

معماری کامپیوتر

نویسنده

موريس مانو

فهرست مطالب

فصل ۱: مدارهای منطقی دیجیتال ۱

- ۱-۱ کامپیوترهای دیجیتال ۱
- ۱-۲ گیت های منطقی ۴
- ۱-۳ جبر بول ۷
- ۱-۴ ساده سازی با نقشه ۱۱
- ۱-۵ مدارهای ترکیبی ۱۸
- ۱-۶ فلیپ فلاپها ۲۲
- ۱-۷ مدارهای ترتیبی ۲۹
- مسائل ۳۸

فصل ۲: قطعات دیجیتال ۴۱

- ۲-۱ مدارهای مجتمع ۴۱
- ۲-۲ دیکدرها ۴۴
- ۲-۳ مولتی پلکسرها ۴۸
- ۲-۴ ثبات ها ۵۰
- ۲-۵ شیفت رجیسترها ۵۴
- ۲-۶ شمارنده های دودویی ۵۷
- ۲-۷ واحد حافظه ۶۱
- مسائل ۶۵

فصل ۳: نمایش داده ها ۶۸

- ۳-۱ انواع داده ها ۶۸
- ۳-۲ متمم ها ۷۶
- ۳-۳ نمایش ممیز - ثابت ۷۹
- ۳-۴ نمایش ممیز شناور ۸۵
- ۳-۵ انواع دیگر کدهای دودویی ۸۶
- ۳-۶ کدهای آشکارسازی خطا ۸۹
- مسائل ۹۲

فصل ۴: انتقال ثبات ها و ریز عمل ها ۹۴

- ۴-۱ زبان انتقال ثبات ۹۴
- ۴-۲ انتقال ثبات ۹۶

- ۴-۳ انتقال های گذرگامی و حافظه ای ۹۹
- ۴-۴ ریزعمل های حسابی ۱۰۳
- ۴-۵ ریزعمل های منطقی ۱۱۰
- ۴-۶ ریزعمل های شیفت ۱۱۵
- ۴-۷ واحد حساب، منطق و شیفت ۱۱۸
- مسائل ۱۲۰

فصل ۵: سازمان و طراحی یک کامپیوتر پایه ۱۲۴

- ۵-۱ کدهای دستورالعمل ها ۱۲۴
- ۵-۲ ثبات های کامپیوتر ۱۲۸
- ۵-۳ دستورالعمل های کامپیوتر ۱۳۳
- ۵-۴ زمانبندی و کنترل ۱۳۶
- ۵-۵ سیکل دستورالعمل ۱۴۰
- ۵-۶ دستورالعمل های ارجاع به حافظه ۱۴۵
- ۵-۷ ورودی - خروجی و وقفه ۱۵۰
- ۵-۸ تشریح کامل کامپیوتر ۱۵۷
- ۵-۹ طراحی کامپیوتر پایه ۱۵۹
- ۵-۱۰ طراحی مدار منطقی انباره ۱۶۴
- مسائل ۱۶۸

فصل ۶: برنامه نویسی کامپیوتر پایه ۱۷۳

- ۶-۱ مقدمه ۱۷۳
- ۶-۲ زبان ماشین ۱۷۴
- ۶-۳ زبان اسمبلی ۱۷۹
- ۶-۴ اسمبلر ۱۸۳
- ۶-۵ حلقه در برنامه نویسی ۱۹۰
- ۶-۶ برنامه نویسی اعمال حسابی و منطقی ۱۹۳
- ۶-۷ زیرروال ها ۱۹۸
- ۶-۸ برنامه نویسی ورودی - خروجی ۲۰۲
- مسائل ۲۰۹

فصل ۷: کنترل ریز برنامه نویسی شده ۲۱۳

- ۷-۱ حافظه کنترل ۲۱۳
- ۷-۲ دنبال کردن آدرس ۲۱۶

۷-۳ مثال ریزبرنامه ۲۲۱

۷-۴ طراحی واحد کنترل ۲۳۱

مسائل ۲۳۶

فصل ۸: واحد مرکزی پردازش ۲۴۰

۸-۱ مقدمه ۲۴۰

۸-۲ سازمان ثبات های عمومی ۲۴۱

۸-۳ سازمان پشته ۲۴۶

۸-۴ قالب دستورالعمل ها ۲۵۴

۸-۵ روشهای آدرس دهی ۲۵۹

۸-۶ انتقال و دستکاری داده ها ۲۶۵

۸-۷ کنترل برنامه ۲۷۲

۸-۸ کامپیوتر کم دستور RISC ۲۸۲

مسائل ۲۹۱

فصل ۹: پردازش خط لوله ای و برداری ۲۹۸

۹-۱ پردازش موازی ۲۹۸

۹-۲ خط لوله ۳۰۱

۹-۳ خط لوله حسابی ۳۰۶

۹-۴ خط لوله دستورالعمل ۳۰۹

۹-۵ خط لوله RISC ۳۱۴

۹-۶ پردازش برداری ۳۱۹

۹-۷ پردازشگر آرایه ۳۲۶

مسائل ۳۲۸

فصل ۱۰: معماری کامپیوتر ۳۳۱

۱۰-۱ مقدمه ۳۳۱

۱۰-۲ جمع و تفریق ۳۳۲

۱۰-۳ الگوریتم ضرب ۳۳۸

۱۰-۴ الگوریتم های تقسیم ۳۴۵

۱۰-۵ عمل های حسابی ممیز شناور ۳۵۲

۱۰-۶ واحد حساب دهمی ۳۶۲

۱۰-۷ اعمال حسابی دهمی ۳۶۸

مسائل ۳۷۴

فصل ۱۱: سازمان ورودی - خروجی ۳۷۹

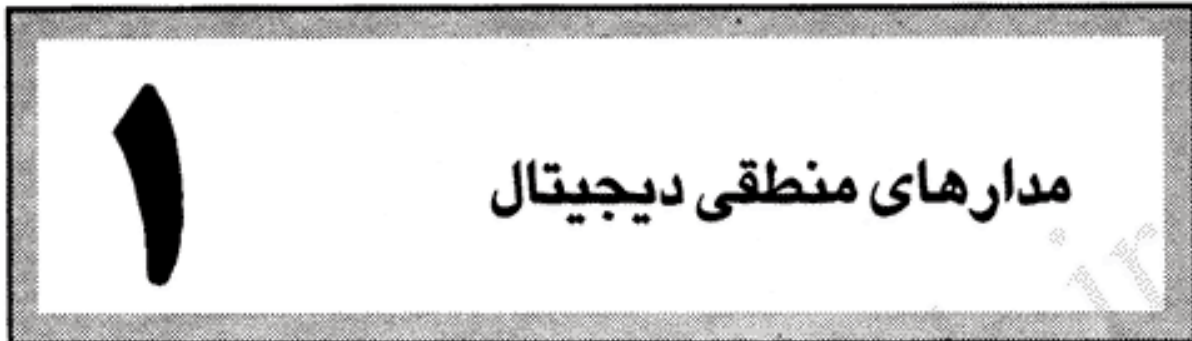
- ۱۱-۱ وسایل جانبی ۳۷۹
- ۱۱-۲ واسطه ورودی - خروجی ۳۸۳
- ۱۱-۳ انتقال غیرهمزمان داده ۳۹۰
- ۱۱-۴ شیوه های انتقال ۴۰۱
- ۱۱-۵ وقفه اولویت دار ۴۰۶
- ۱۱-۶ دستیابی مستقیم به حافظه (DMA) ۴۱۶
- ۱۱-۷ پردازنده ورودی و خروجی IOP ۴۲۰
- ۱۱-۸ تبادل اطلاعات سری ۴۳۰
- مسائل ۴۴۱

فصل ۱۲: سازمان حافظه ۴۴۵

- ۱۲-۱ سلسله مراتب حافظه ۴۴۵
- ۱۲-۲ حافظه اصلی ۴۴۸
- ۱۲-۳ حافظه کمکی ۴۵۴
- ۱۲-۴ حافظه تداعیگر ۴۵۷
- ۱۲-۵ حافظه کش ۴۶۳
- ۱۲-۶ حافظه مجازی ۴۷۱
- ۱۲-۷ سخت افزار مدیریت حافظه ۴۷۸
- مسائل ۴۸۵

فصل ۱۳: چند پردازنده ها ۴۸۹

- ۱۳-۱ مشخصات چند پردازنده ها ۴۸۹
- ۱۳-۲ ساختارهای اتصالات متقابل ۴۹۱
- ۱۳-۳ دآوری بین پردازنده ها ۵۰۰
- ۱۳-۴ ارتباط و همگامی بین پردازنده ها ۵۰۷
- ۱۳-۵ همبستگی حافظه کش ۵۱۰
- مسائل ۵۱۴



- ۱-۱ کامپیوترهای دیجیتال
- ۱-۲ گیت‌های منطقی
- ۱-۳ جبر بول
- ۱-۴ ساده سازی با نقشه
- ۱-۵ مدارهای ترکیبی
- ۱-۶ فلیپ فلاپها
- ۱-۷ مدارهای ترتیبی

۱-۱ کامپیوترهای دیجیتال

کامپیوتر دیجیتال سیستمی دیجیتالی^۱ (رقمی) است که انواع کارهای محاسباتی را انجام می‌دهد. کلمه دیجیتال بدان معنی است که اطلاعات در کامپیوتر توسط متغیرهایی که تعداد محدودی از مقادیر گسسته را بخود اختصاص می‌دهند نمایش داده می‌شوند. این مقادیر در داخل کامپیوتر بوسیله اجزائی که می‌توانند تعداد محدودی از حالت‌های گسسته را در خود حفظ کنند پردازش می‌شوند. مثلاً ارقام دهدهی 0، 1، 2، ...، 9، ده مقدار گسسته را فراهم می‌آورند. اولین کامپیوترهای الکترونیک دیجیتال که در اواخر دهه ۱۹۴۰ ساخته شدند اساساً برای محاسبات عددی بکار می‌رفتند. در این مورد، عناصر گسسته ارقام هستند و در واقع اصطلاح کامپیوتر دیجیتال از این کاربرد گرفته شده است. در عمل، کامپیوترهای دیجیتال با قابلیت اطمینان بیشتری کار می‌کنند بشرطی که تنها از دو حالت استفاده شود. به دلیل محدودیت‌های فیزیکی اجزاء، و نیز بعلت گرایش انسان به منطقی دودویی (یعنی بکارگیری

1- Digital

عبارات صحیح و غلط، بله یا خیر) قطعات دیجیتال که مقید به نگهداری مقادیر گسسته هستند مقید به نگهداری و حفظ فقط دو مقدار که دودویی¹ نامیده می‌شوند نیز می‌باشند.

کامپیوترهای دیجیتال از سیستم اعداد دودویی² استفاده می‌کنند که دو رقم بیشتر ندارد: 0 و 1. یک رقم دودویی بیت خوانده می‌شود. اطلاعات در کامپیوترهای دیجیتال بوسیله گروههایی از بیت‌ها نشان داده می‌شوند. با استفاده از تکنیک‌های کدگذاری، گروه‌های بیت‌ها نه تنها برای نمایش اعداد دودویی بلکه برای سایر سمبل‌های گسسته، همچون ارقام دهدهی یا حروف الفبا نیز بکار برده می‌شوند. با استفاده صحیح از مجموعه‌های دودویی و بکارگیری روشهای مختلف کدگذاری می‌توان گروه‌های بیت‌ها را برای ساخت مجموعه‌های کامل دستورالعمل³ها جهت انجام محاسبات مختلف بکار برد.

برخلاف اعداد دهدهی متداول که سیستم مبنای 10 را بکار می‌برند، اعداد دودویی سیستم مبنای 2 با دو رقم 0 و 1 را استفاده می‌نمایند. معادل دهدهی یک عدد دودویی را می‌توان با بسط آن به صورت یک سری از توان‌های 2 بدست آورد. مثلاً عدد دودویی 1001011 کمیتی را نشان می‌دهد که قابل تبدیل به یک عدد دهدهی است و از ضرب هر بیت در توان صحیحی از 2 بترتیب زیر بدست می‌آید:

$$1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 75$$

هفت بیت 1001011 یک عدد دودویی را نشان می‌دهند که معادل دهدهی آن 75 است. با این وجود همین گروه هفت بیتی وقتی بعنوان کد دودویی حروف الفبا در نظر گرفته شود نمایشگر حرف K است. این گروه می‌تواند یک کد کنترلی را هم برای مشخص کردن برخی تصمیمات منطقی در کامپیوترهای دیجیتال نمایش دهد. به بیان دیگر، گروه بیت‌ها در یک کامپیوتر دیجیتال برای نمایش بسیاری از چیزهاست. این مطلب شبیه به مفهوم استفاده از حروف الفبا یکسان در ساخت زبانهای مختلف مانند انگلیسی و فرانسه است.

گاهی اوقات یک سیستم کامپیوتر به دو بخش عملیاتی با ماهیت نرم‌افزاری⁴ و سخت‌افزاری⁵ تقسیم می‌گردد. سخت‌افزار کامپیوتر شامل تمام قطعات الکترونیک و الکترومکانیک تشکیل دهنده ماهیت فیزیکی آن است. نرم‌افزار کامپیوتر از دستورالعمل‌ها و داده‌هایی⁶ تشکیل شده که کامپیوتر برای انجام کارهای مختلف داده پرداز از آنها استفاده می‌نماید. رشته‌ای از دستورات، برنامه⁷ خوانده می‌شود. داده‌هایی که برنامه با آنها کار می‌کند پایگاه داده‌ها⁸ را تشکیل می‌دهد.

یک سیستم کامپیوتر از ترکیب سخت‌افزار، و نرم‌افزار سیستم برای آن تشکیل شده است. نرم‌افزار سیستم یک کامپیوتر گزیده‌ای از برنامه‌هاست که هدف از کاربرد آن بهره‌گیری موثرتر از کامپیوتر

1- Binary

2- Binary Number System

3- Instruction

4- Software

5- Hardware

6- Data

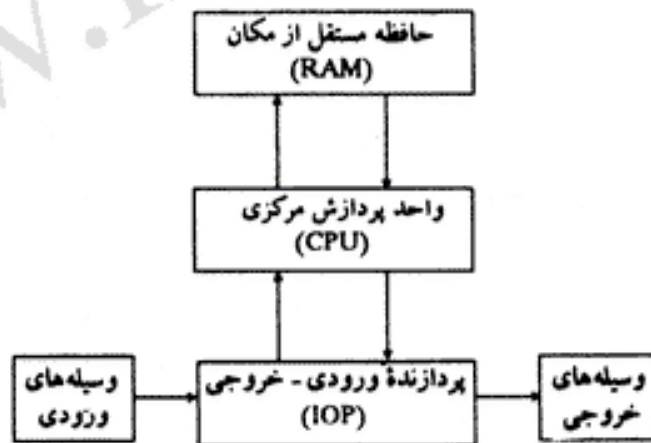
7- Program

8- Data Base

می‌باشد. برنامه‌هایی که بصورت یک بسته نرم‌افزاری سیستم درآمده‌اند سیستم عامل^۱ خوانده می‌شوند. این برنامه‌ها از برنامه‌هایی که کاربر برای حل مسائل خاصی می‌نویسد متمایز است. بعنوان مثال، یک برنامه به زبان سطح بالا که توسط کاربر برای حل یک داده‌پردازی خاص نوشته شده است یک برنامه کاربردی است ولی یک کامپایلر^۲ که برنامه زبان سطح بالا را به برنامه‌ای به زبان ماشین تبدیل می‌کند یک برنامه سیستم است. خریداری که کامپیوتر را می‌خرد علاوه بر سخت‌افزار به انواع نرم‌افزارهای موجود برای بکارگیری موثرتر کامپیوتر هم نیاز دارد. نرم‌افزار سیستم بخش لاینفک یک سیستم کامپیوتری است. نقش آن جبران اختلافات موجود بین نیازهای کاربر و توانایی سخت‌افزار است.

سخت‌افزار کامپیوتر معمولاً به سه بخش عمده مطابق شکل ۱-۱ تقسیم می‌شود. واحد پردازش مرکزی^۳ CPU حاوی یک واحد حساب و منطق^۴ ALU برای کار روی داده‌ها، تعدادی ثبات برای ذخیره کردن داده‌ها، واحدهای کنترل برای برداشت^۵ و اجرای^۶ دستورالعمل‌هاست. حافظه یک کامپیوتر محل نگهداری دستورالعمل‌ها و داده‌هاست. این حافظه، حافظه با دستیابی تصادفی^۷ خوانده می‌شود زیرا CPU قادر است به هر مکانی از حافظه بطور تصادفی دستیابی نموده و در یک فاصله زمانی ثابت اطلاعات دودویی را از آنجا بردارد. پردازنده^۸ ورودی و خروجی (IOP) حاوی مدارات الکترونیک برای کنترل تبادل و انتقال اطلاعات بین کامپیوتر و دنیای خارج است. وسایل^۹ ورودی - خروجی متصل به کامپیوتر عبارتند از صفحه کلیدها، چاپگرها، پایانه‌ها، راه‌اندازهای دیسک مغناطیسی و سایر وسایل تبادل اطلاعات و ارتباطی دیگر است.

این کتاب دانش ابتدایی لازم را برای درک عملیات سخت‌افزاری یک سیستم کامپیوتر فراهم



شکل ۱-۱ بلاک دیاگرام یک کامپیوتر دیجیتال

- | | | |
|--------------------------|-------------|----------------------------|
| 1- Operating System | 2- Compiler | 3- Central Processing Unit |
| 4- Arithmetic Logic Unit | 5- Fetch | 6- Execute |
| 7- Random Access Memory | | 8- Processor |
| 9- Device | | |

می سازد. گاهی اوقات موضوع، بسته به علاقه محقق، از سه دیدگاه مورد بررسی قرار می گیرد. بهنگام بحث درباره سخت افزار کامپیوتر لازم است تا بین سازمان کامپیوتر^۱، طراحی کامپیوتر^۲ و معماری کامپیوتر^۳ تمایز قایل شویم.

سازمان کامپیوتر به نحوه عملکرد اجزاء قطعات سخت افزاری و چگونگی اتصالات آنها در تشکیل سیستم کامپیوتر اشاره دارد. فرض بر این است که قطعات مختلف در مکان های صحیح خود قرار دارند و آنچه لازم است انجام شود بررسی ساختار سازمانی است جهت تحقیق عملکرد کامپیوتر.

طراحی کامپیوتر به طراحی سخت افزاری کامپیوتر مربوط می شود. پس از تهیه مشخصات کامپیوتر، طراح باید سخت افزار سیستم را بوجود آورد. طراحی کامپیوتر به تعیین نوع سخت افزار مربوط می شود و نیز اینکه چگونه آنها بهم متصل گردند. این جنبه از سخت افزار کامپیوتر را گاهی پیاده سازی^۴ کامپیوتر می خوانند.

معماری کامپیوتر به ساختار و رفتار کامپیوتر از دیدگاه کاربر مربوط می شود. این بحث شامل مطالبی درباره قالب اطلاعات، مجموعه دستورالعمل ها، و تکنیک های آدرس دهی حافظه است. طراحی معماری یک سیستم کامپیوتری درباره مشخصات انواع ماژول های^۵ عملیاتی، مانند پردازنده ها و حافظه ها و ساختاردهی آنها برای تشکیل یک سیستم کامپیوتر بحث می کند.

این کتاب درباره هر سه نوع موضوع مربوط به سخت افزار کامپیوتر بحث می نماید. در فصل های ۱ الی ۴، ما انواع قطعات دیجیتال را که در سازمان و طراحی کامپیوتر بکار رفته اند ارائه می نمائیم. فصل ۵ تا ۷ مراحل را که یک طراح باید در طراحی یک کامپیوتر دیجیتال ساده طی کند پوشش می دهد. فصل های ۸ و ۹ درباره معماری واحد پردازش مرکزی است. در فصل های ۱۱ و ۱۲ سازمان و معماری پردازنده ورودی و خروجی و واحد حافظه ارائه شده است.

۱-۲ گیت های منطقی

اطلاعات دودویی در کامپیوترهای دیجیتال با کمیت هایی فیزیکی که سیگنال نامیده می شود نمایش داده می شود. سیگنال های الکتریکی همچون ولتاژهای موجود در سرتاسر کامپیوتر در یکی از دو حالت قابل تشخیص از هم قرار دارند. این دو حالت نماینده یک متغیر دودویی هستند که می تواند برابر 1 یا 0 تصور شود. مثلاً یک کامپیوتر دیجیتال ممکن است از یک سیگنال 3 ولت برای نمایش 1 دودویی و 0.5 ولت برای نشان دادن 0 دودویی استفاده نماید. پایانه های ورودی مدارهای دیجیتال سیگنال های 3 و 0.5 را پذیرفته و پایانه های خروجی هم با سیگنال های 3 و 0.5 به ورودی ها پاسخ می دهند. ورودی ها و خروجی های 3 و 0.5 بترتیب متناظر با مقادیر دودویی 1 و 0 می باشند.

منطق دودویی با متغیرهای دودویی و نیز با اعمالی که مفهوم منطقی دارند سروکار دارد. از این منطق برای توصیف عملیات و یا پردازش اطلاعات دودویی بصورت عبارات جبری و یا جداول

1- Computer Organization

2- Computer Design

3- Computer Architecture

4- Implementation

5- Madule

استفاده می شود. دستکاری اطلاعات دودویی توسط مدارهای منطقی بنام گیت¹ انجام می شود. گیت ها بلاک های سخت افزاری هستند که سیگنال های دودویی 1 یا 0 را در خروجی، بسته به شرایط ورودی هایشان تولید می کنند. امروزه انواع گیت های منطقی در سیستم های کامپیوتر دیجیتال بکار برده می شوند. هر گیت سمبل گرافیک خاص خود را داشته و عملکردش توسط یک عبارت جبری نمایش داده می شود. رابطه ورودی و خروجی برای متغیرهای دودویی برای هر گیت را می توان توسط یک جدول بنام جدول درستی² نشان داد.

نام، سمبل گرافیکی، توابع جبری و جداول درستی هشت گیت منطقی در شکل ۱-۲ لیست شده اند. هر گیت دارای یک یا دو متغیر ورودی است که با A و B و یک خروجی که با X مشخص شده است. گیت AND تابع منطقی AND را تولید می کند، یعنی خروجی آن 1 است اگر هر دو ورودی A و B برابر 1 باشد؛ در غیر این صورت خروجی آن 0 است. این شرایط برای گیت AND بصورت جدول مشخص شده است. جدول همچنین نشان می دهد که X برابر 1 است اگر هر دو ورودی A و B برابر 1 باشند. سمبل جبری عملکرد تابع AND شبیه سمبل ضرب در حساب معمولی است. ما می توانیم برای نمایش عمل از یک نقطه در بین دو متغیر و یا بدون هر نوع علامتی در بین آنها استفاده کنیم. گیت های AND ممکن است بیش از دو ورودی داشته باشند و بنا به تعریف خروجی آنها هنگامی 1 است که تمام ورودی ها 1 باشند. گیت OR تابع OR غیر انحصاری را تولید می کند، یعنی خروجی 1 است اگر ورودی A یا B یا هر دو 1 باشند؛ در غیر این صورت خروجی 0 است. سمبل جبری تابع OR علامت + است که شبیه علامت جمع حسابی است. گیت های OR ممکن است بیش از دو ورودی داشته باشند، و بنا به تعریف، خروجی هنگامی 1 است که هر یک از ورودی ها 1 باشند.

مدار معکوس کننده³ وضعیت منطقی هر سیگنال دودویی را معکوس می کند. این مدار تابع NOT یا متمم را تولید می نماید. سمبل جبری مورد استفاده برای متمم منطقی، علامت پریم یا خط روی سمبل متغیر مورد نظر است. در این کتاب ما از علامت پریم برای متمم منطقی یک متغیر دودویی استفاده می کنیم و خط روی حرف را برای مشخص کردن زیر عمل متمم که در فصل ۴ تعریف خواهد شد نگه میداریم. دایره کوچک در خروجی سمبل گرافیکی معکوس کننده، مشخص کننده متمم منطقی است.



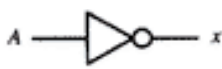





یک سمبل مثلثی شکل، مربوط به یک بافر⁴ (میانگیر) است. یک بافر هیچگونه تابع منطقی را تولید نمی کند چون مقدار دودویی خروجی آن برابر با مقدار دودویی ورودی است. این مدار صرفاً به منظور تقویت توان مورد استفاده قرار می گیرد. مثلاً، بافری که ولتاژ 3 ولت را برای عدد دودویی 1 مورد استفاده قرار می دهد. هرگاه در ورودی 3 ولت را دریافت کند خروجی 3 ولت را تولید خواهد کرد. با این وجود مقدار توان الکتریکی لازم در ورودی بافر بسیار کمتر از توان تولید شده در خروجی آن است. هدف اصلی از بکارگیری بافر راه اندازی سایر گیت ها است که توان قابل ملاحظه ای را نیاز دارند.

1- Gate

2- Truth table

3- Inverter

4- Buffer

نام	سمبل گرافیکی	تابع جبری	جدول درستی															
AND		$x = A \cdot B$ یا $x = AB$	<table border="1"> <tr><td>A</td><td>B</td><td>x</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	A	B	x	0	0	0	0	1	0	1	0	0	1	1	1
A	B	x																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
OR		$x = A + B$	<table border="1"> <tr><td>A</td><td>B</td><td>x</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	A	B	x	0	0	0	0	1	1	1	0	1	1	1	1
A	B	x																
0	0	0																
0	1	1																
1	0	1																
1	1	1																
معکوس کننده		$x = A'$	<table border="1"> <tr><td>A</td><td>x</td></tr> <tr><td>0</td><td>1</td></tr> <tr><td>1</td><td>0</td></tr> </table>	A	x	0	1	1	0									
A	x																	
0	1																	
1	0																	
میانگیر		$x = A$	<table border="1"> <tr><td>A</td><td>x</td></tr> <tr><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td></tr> </table>	A	x	0	0	1	1									
A	x																	
0	0																	
1	1																	
NAND		$x = (AB)'$	<table border="1"> <tr><td>A</td><td>B</td><td>x</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	A	B	x	0	0	1	0	1	1	1	0	1	1	1	0
A	B	x																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
NOR		$x = (A+B)'$	<table border="1"> <tr><td>A</td><td>B</td><td>x</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	A	B	x	0	0	1	0	1	0	1	0	0	1	1	0
A	B	x																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
بای انحصاری (XOR)		$x = A \oplus B$ یا $x = A'B + AB'$	<table border="1"> <tr><td>A</td><td>B</td><td>x</td></tr> <tr><td>0</td><td>0</td><td>0</td></tr> <tr><td>0</td><td>1</td><td>1</td></tr> <tr><td>1</td><td>0</td><td>1</td></tr> <tr><td>1</td><td>1</td><td>0</td></tr> </table>	A	B	x	0	0	0	0	1	1	1	0	1	1	1	0
A	B	x																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
NOR انحصاری با تابع برابری		$x = (A \oplus B)'$ یا $x = A'B' + AB$	<table border="1"> <tr><td>A</td><td>B</td><td>x</td></tr> <tr><td>0</td><td>0</td><td>1</td></tr> <tr><td>0</td><td>1</td><td>0</td></tr> <tr><td>1</td><td>0</td><td>0</td></tr> <tr><td>1</td><td>1</td><td>1</td></tr> </table>	A	B	x	0	0	1	0	1	0	1	0	0	1	1	1
A	B	x																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

شکل ۱-۲ گیت‌های منطقی دیجیتال

تابع NAND متمم تابع AND است و همانطور که از سمبل گرافیکی آن مشخص است از یک سمبل گرافیکی AND که بدنبال آن دایره کوچکی تشکیل شده است. کلمه NAND از اختصار NOT-AND بدست آمده است. گیت NOR متمم OR است و سمبل گرافیکی OR که بدنبال آن دایره کوچکی آمده است را بکار می برد. هر دو گیت NAND و NOR ممکن است بیش از دو ورودی داشته باشند، و خروجی آنها همیشه بترتیب متمم توابع AND و OR است.

گیت OR انحصاری^۱ دارای سمبل گرافیکی شبیه به گیت OR است جز اینکه یک خط منحنی اضافی در سمت ورودی آن وجود دارد. خروجی این گیت هنگامی 1 است که فقط یکی از ورودی ها 1 باشد ولی نباید هر دوی آنها توأمأ یک باشند. تابع OR انحصاری (XOR یا EOR) می تواند توسط سمبل گرافیکی خاص خودش و یا برحسب گیت های AND و OR و متمم طبق شکل ۱-۲ نشان داده شود. NOR انحصاری^۲، همانطور که از سمبل گرافیکی آن مشخص است متمم OR انحصاری است. خروجی این گیت فقط هنگامی 1 است که هر دو ورودی 1 و یا هر دو ورودی 0 باشند. نام مناسبتری برای گیت OR انحصاری تابع فرد^۳ است، یعنی خروجی 1 است اگر تعداد فردی از ورودی ها 1 باشند. بنابراین در یک تابع OR انحصاری سه ورودی، خروجی هنگامی 1 است که فقط یکی از ورودی ها 1 و یا هر سه ورودی 1 باشد. گیت های EOR و ENOR معمولاً دو ورودی هستند و به ندرت با سه ورودی یا بیشتر یافت می شوند.

۱-۳ جبر بول

جبر بول جبری است که با متغیرهای دودویی و اعمال منطقی سروکار دارد. متغیرها با حروفی همچون A و B و X و Y مشخص می شوند. سه عمل اصلی منطقی عبارتند از AND، OR و متمم. یک تابع بول را می توان با استفاده از متغیرهای دودویی، سمبل های عملیات منطقی، پرانتزها و یک علامت تساوی نشان داد. تابع بول برای هر مقدار مفروض از متغیرها می تواند 1 یا 0 باشد. بعنوان مثال تابع بول زیر را در نظر بگیرید

$$F = x + y'z$$

تابع F برابر با 1 است اگر X برابر 1 و یا هر دو y' و z برابر 1 باشند؛ در غیر اینصورت F برابر 0 است. ولی اینکه می گوئیم $y' = 1$ معادل است با اینکه بگوئیم $y = 0$ ، چون y' متمم y است. بنابراین ما ممکن است بگوئیم که F هنگامی 1 است که $x = 1$ یا اگر $yz = 01$ باشد. رابطه بین یک تابع و متغیرهای دودویی اش را با یک جدول درستی می توان نشان داد. برای نمایش یک تابع در یک جدول درستی، لیستی از 2^n ترکیب از n متغیر دودویی نیاز داریم. همانطور که در جدول شکل ۱-۳ (الف) دیده می شود

1- Exclusive OR (XOR, EOR)

2- Exclusive NOR (ENOR) (Equivalence)

3- Odd Function

هشت ترکیب جداگانه برای 3 متغیر x و y و z وجود دارد. تابع F برای ترکیب‌هایی 1 است که در آنها $x = 1$ ، $yz = 01$ باشد؛ این تابع برای سایر ترکیب‌ها صفر است.

یک تابع بول را می‌توان از یک عبارت جبری به یک دیاگرام منطقی متشکل از گیت‌های AND، OR و یا معکوس کننده تبدیل کرد. دیاگرام منطقی F در شکل ۱-۳ (ب) نشان داده شده است برای ورودی y ، معکوس کننده‌ای وجود دارد که متمم y' را تولید می‌نماید. برای جمله $y'z$ یک گیت AND وجود دارد و از یک گیت OR برای ترکیب دو جمله استفاده شده است. در یک دیاگرام منطقی، متغیرهای تابع ورودی‌های مدار و سمبل تابع، خروجی آن را تشکیل می‌دهد.

هدف از جبر بول تسهیل تحلیل و طراحی مدارهای دیجیتال است. این جبر ابزار مناسبی برای اعمال زیر نیز هست.

۱- بیان رابطه جبری بین متغیرها بصورت یک جدول درستی

۲- بیان رابطه‌ای بین ورودی- خروجی یک دیاگرام منطقی بصورت عبارت جبری

۳- یافتن مدار ساده‌تر برای تابع

نمایش یک جدول درستی بصورت یک عبارت جبری به طرق مختلف امکان پذیر است. برطبق قوانین حاکم در جبر بول، با دستکاری یک عبارت بولی می‌توان فرم ساده‌تری که گیت‌های کمتری نیاز دارد را برای آن بدست آورد. برای دیدن چگونگی انجام این اعمال، ابتدا توانایی‌های دستکاری جبر بول را می‌بینیم. جدول ۱-۱ ابتدایی‌ترین اتحادهای^۱ جبر بول را نشان می‌دهد. تمام اتحادهای جدول بوسیله جدول درستی قابل اثباتند. هشت اتحاد اول رابطه اساسی بین یک متغیر و خودش، یا با ثابتهای 1 و 0 را نشان می‌دهد. پنج اتحاد بعدی (از معمولی 9 تا 13) شبیه به روابط در جبر معمولی هستند. اتحاد 14 در جبر معمولی صادق نیست ولی در دستکاری^۲ عبارات بولی مفید است. اتحادهای 15 و 16 تئوری‌های دومورگان نامیده می‌شوند و در زیر بحث شده‌اند. آخرین اتحاد بیان می‌دارد که اگر متغیری دوبار متمم شود مقدار اولیه آن بدست می‌آید.



(ب) دیاگرام منطقی

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(الف) جدول درستی

شکل ۱-۳ جدول درستی و دیاگرام منطقی برای $F = x + y'z$.

جدول ۱-۱ اتحادهای اساسی جبر بول

(1) $x + 0 = x$	(2) $x \cdot 0 = 0$
(3) $x + 1 = 1$	(4) $x \cdot 1 = x$
(5) $x + x = x$	(6) $x \cdot x = x$
(7) $x + x' = 1$	(8) $x \cdot x' = 0$
(9) $x + y = y + x$	(10) $xy = yx$
(11) $x + (y + z) = (x + y) + z$	(12) $x(yz) = (xy)z$
(13) $x(y + z) = xy + xz$	(14) $x + yz = (x + y)(x + z)$
(15) $(x + y)' = x'y'$	(16) $(xy)' = x' + y'$
(17) $(x')' = x$	

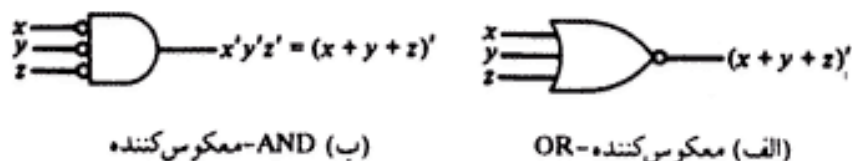
اتحادهایی که در جدول لیست شده‌اند هم در مورد متغیرهای تکی و هم در مورد توابع بولی که برحسب مجموعه‌ای از متغیرهای دودویی بیان شده‌اند صادق‌اند. مثلاً عبارت جبری بولی زیر را در نظر بگیرید:

$$AB' + C'D + AB' + C'D$$

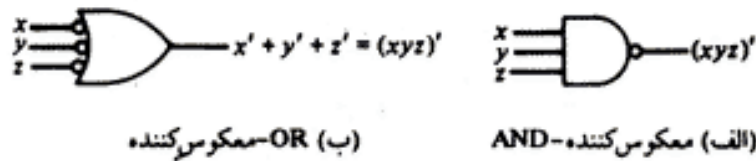
با فرض $x = AB' + C'D$ رابطه فوق برابرست با $x + x$. با استفاده از اتحاد 5 در جدول 1-1 می‌بینیم که $x + x = x$ است. بنابراین عبارت فوق بصورت زیر کاهش می‌یابد

$$AB' + C'D + AB' + C'D = AB' + C'D$$

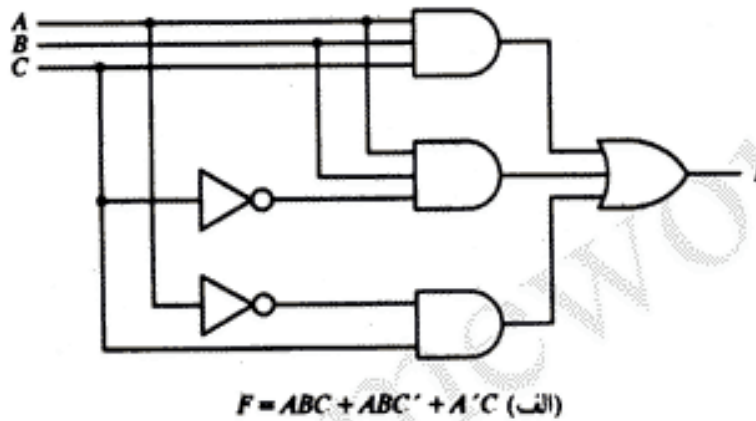
تئوری دو مورگان در کار با گیت‌های NOR و NAND بسیار با اهمیت است. این تئوری بیان می‌کند که یک گیت NOR که عمل $(x + y)'$ را انجام می‌دهد معادل با $x'y'$ است. بطور مشابه یک گیت NAND می‌تواند به یکی از دو صورت $(xy)'$ یا $(x' + y')$ بیان شود. باین دلیل همانطور که در شکل‌های 4-1 و 5-1 دیده می‌شود گیت‌های NOR و NAND دو سمبل نمایشی متمایز دارند. بعوض نمایش گیت NOR با یک دایره کوچکی بدنبال آن، می‌توان آن را با یک گیت AND با دایره‌های کوچکی در تمام ورودی‌هایش نشان داد. سمبل معکوس‌کننده-AND برای گیت NOR از تئوری دو مورگان حاصل می‌شود و در آن دایره کوچک به معنی متمم سازی است. بطور مشابه، گیت NAND دارای دو سمبل متمایز است، شکل 5-1.



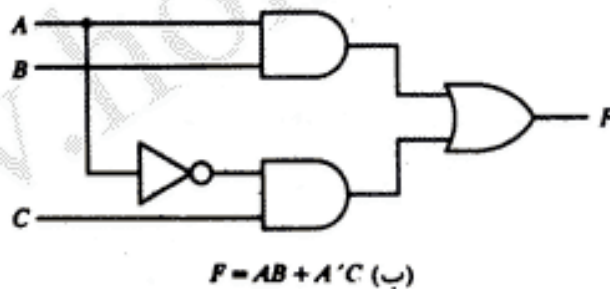
شکل 4-1 دو سمبل گرافیکی برای گیت NOR.



شکل ۵-۱ دو سمبل گرافیکی برای گیت NAND.



$F = ABC + ABC' + A'C$ (الف)



$F = AB + A'C$ (ب)

شکل ۶-۱ دو دیاگرام منطقی برای یک تابع بولی.

برای دیدن چگونگی بکارگیری دستکاری جبر بول در ساده کردن مدارهای دیجیتال، دیاگرام منطقی شکل ۶-۱ (الف) را ملاحظه کنید. خروجی مدار می تواند بصورت عبارت جبری زیر بیان شود

$$F = ABC + ABC' + A'C$$

هر جمله متعلق به یک گیت AND است و گیت OR جمع منطقی سه جمله را تشکیل می دهد. برای متمم های A' و C' دو معکوس کننده^۱ لازم است. عبارت فوق می تواند بصورت زیر ساده شود.

$$F = ABC + ABC' + A'C = AB(C + C') + A'C = AB + A'C$$

1- Inverter

توجه کنید طبق رابطه 7 که در جدول 1-1 $(C + C') = 1$ و به موجب اتحاد 4، $AB \cdot 1 = AB$ است. دیاگرام منطقی عبارت ساده شده در شکل 6-1 (ب) کشیده شده است. این دیاگرام تنها به چهارگیت در عوض شش گیت شکل 6-1 (الف) نیاز دارد. دو مدار معادل بوده و جدول درستی یکسانی را برای ورودی های A، B و C و خروجی F فراهم می آورند.

متمم یک تابع

متمم یک تابع مانند F را، وقتی که بصورت جدول درستی نشان داده شده باشد، می توان از تعویض 1 ها به 0 ها بدست آورد. بهنگام نمایش تابع بصورت عبارت جبری، متمم تابع با توجه به تئوری دُمورگان حاصل می شود. فرم کلی تئوری دُمورگان بصورت زیر نوشته می شود

$$(x_1 + x_2 + x_3 + \dots + x_n)' = x_1' x_2' x_3' \dots x_n'$$

$$(x_1 x_2 x_3 \dots x_n)' = x_1' + x_2' + x_3' + \dots + x_n'$$

از تئوری کلی دُمورگان می توانیم رویه ساده ای برای بدست آوردن متمم یک تابع نتیجه بگیریم. این کار با تبدیل تمام گیت های AND به OR و تمام OR ها به AND و سپس متمم سازی هر متغیر بتنهایی، حاصل می شود. بعنوان مثال رابطه زیر و متمم آن را در نظر بگیرید

$$F = AB + C'D' + B'D$$

$$F' = (A' + B')(C + D)(B + D')$$

عبارت متمم شده از تعویض اعمال AND و OR و متمم کردن هر متغیر بتنهایی، حاصل گشته است. توجه کنید که C' متمم C است.

4-1 ساده سازی با نقشه

پیچیدگی دیاگرام منطقی که یک تابع بول را پیاده سازی می کند مستقیماً به پیچیدگی عبارت جبری که تابع از روی آن پیاده سازی می شود بستگی دارد. هرچند نمایش هر تابع بصورت جدول درستی تنها به یک شکل ممکن است ولی همان تابع می تواند بصورت مختلف جبری نشان داده شود. عبارت تابع را می توان با استفاده از روابط اصلی جبر بول ساده کرد. با این وجود، این رویه گاهی اوقات مشکل است زیرا مراحل متوالی دستکاری تابع بعلت عدم کفایت قوانین قابل پیش بینی نیستند. روش نقشه (جدول) رویه ساده و مستقیمی را برای ساده کردن عبارت بول در اختیار می گذارد. این روش را می توان یک نمایش تصویری از جدول درستی تصور نمود که امکان تفسیری آسان و انتخاب جملات می نیمم برای بیان جبری تابع را فراهم می آورد. روش نقشه را روش کارنو یا روش نقشه K نیز می نامند.

هر ترکیبی از متغیرها در یک جدول درستی مینترم نام دارد. مثلاً، جدول درستی شکل 3-1 هشت مینترم دارد. هنگامی که تابع n متغیره بوسیله نقشه کارنو نشان داده شود دارای 2^n مینترم خواهد بود که

معادل 2^n عدد دودویی حاصل از n بیت است. تابع بولی به ازاء بعضی از مینترمها برابر 1 و برخی دیگر 0 است. با نوشتن معادل دهدهی مینترمهایی که مقدار تابع را 1 می کنند می توان اطلاعات موجود در جدول درستی را بشکل فشرده ای درآورد. مثلاً جدول درستی شکل ۱-۳ بصورت زیر بیان می شود

$$F = (x, y, z) = \sum (1, 4, 5, 6, 7)$$

حروف داخل پرانتز متغیرهای دودویی را بترتیبی که در جملات مینترم ظاهر می شوند نشان می دهند. سمبل \sum نشاندهنده مجموع مینترمهایی است که پس از آن در پرانتز آمده اند. مینترمهایی که مقدار 1 را برای تابع تولید می کنند با معادل دهدهی شان در پرانتز آمده اند. مینترمهایی که در پرانتز وجود ندارند مقدار 0 را برای تابع ایجاد می کنند.

نقشه کارنو دیاگرامی است متشکل از تعدادی مربع که هر مربع نشاندهنده یک مینترم است. در مربعات مربوط به مینترمهایی که عدد 1 را برای تابع تولید کنند عدد 1 و در بقیه 0 یا چیزی نمی نویسند. با تشخیصی الگوهای مختلف و ترکیب مربعهایی که در نقشه با 1 ها مشخص شده اند می توان عبارات جبری دیگری برای تابع بدست آورد و نهایتاً از میان آنها مناسبترین را برگزید.

نقشه های توابع دو، سه و چهار متغیره در شکل ۱-۷ نشان داده شده اند. تعداد مربع های تشکیل دهنده نقشه n متغیره 2^n است. 2^n مینترم، بخاطر سهولت ارجاع به آنها با معادل های دهدهی شان

	BC		B	
	00	01	11	10
A	0	1	3	2
A { 1	4	5	7	6

(ب) نقشه سه متغیره

	B	
	0	1
A	0	1
A { 1	2	3

(الف) نقشه دو متغیره

	CD				C			
	00		01		11		10	
AB	00	01	11	10	00	01	11	10
	0	1	3	2	4	5	7	6
	11	12	13	15	14	11	10	10
A { 10	8	9	11	10				

(ج) نقشه چهار متغیره

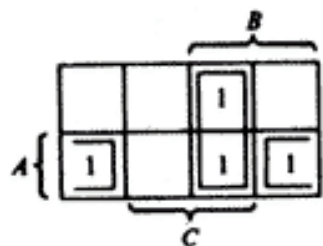
شکل ۱-۷ نقشه برای توابع دو، سه و چهار متغیره

مشخص می‌شوند. شماره‌های مینترم‌ها با ترتیب خاصی به خانه‌ها تخصیص داده شده‌اند بطوری که مینترم‌های مربوط به دو مربع مجاور فقط در یک متغیر با هم اختلاف دارند. نام متغیرها در دو طرف یک خط مورب در گوشه نقشه نوشته می‌شوند. 0ها و 1های نوشته شده در کنار سطرها و ستون‌ها مقدار متغیرها را مشخص می‌کنند. در زیر آکلاد مربوط به هر متغیر، نیمی از خانه‌های نقشه مشخص شده که در آنها متغیر مربوطه بدون پریم است. در بقیه خانه‌های نقشه متغیر دارای پریم (متمم) می‌باشد. مینترم متناظر با هر مربع از روی اعداد دودویی مربوط به متغیرها که در طول سمت چپ و بالای نقشه نوشته شده‌اند تعیین می‌شود. مثلاً مینترم 5 در نقشه سه متغیره در سیستم دودویی 101 است، که می‌تواند از کنار هم قرار گرفتن 1 در سطر دوم و 01 از ستون دوم بدست آید. این مینترم مقداری را برای متغیرهای A و B و C مشخص می‌کند که در آن A و C بدون پریم و B پریم‌دار است (یعنی $AB'C$). از طرف دیگر مینترم 5 در یک نقشه چهار متغیره دارای چهار متغیر می‌باشد. عدد دودویی برای چهار بیت 0101 می‌باشد و مینترم مربوطه هم $A'BC'D$ می‌باشد.

مینترم‌های مربعات مجاور در نقشه بجز در یک متغیر یکسانند. این متغیر در یک مربع نسبت به مربع مجاور متمم است. براساس این تعریف از مجاورت، مربعات انتهایی در سطرها نیز مجاور تصور خواهند شد. موضوع برای مربعات انتهایی در ستون‌ها نیز صادق است. در نتیجه، چهار مربع در گوشه‌های نقشه نیز مجاور خواهند بود. یک تابع بول که بوسیله جدول درستی‌اش نمایش داده شده است به یک نقشه منتقل می‌گردد. بدین ترتیب که در مربع‌هایی که تابع در آنها 1 است عدد 1 نوشته می‌شود. مربع‌های مجاور که در آنها 1 وجود دارد بصورت گروهی باهم ترکیب می‌شوند. تعداد مربع‌های هر گروه باید توان صحیحی از 2 باشد. هر گروه می‌تواند در یک یا چند مربع با یک یا چند گروه دیگر اشتراک داشته باشد. هر گروه از مربعات یک عبارت جبری را نشان می‌دهد، و OR این جملات، عبارت ساده شده جبری را برای تابع فراهم می‌آورد. مثالهای زیر کاربرد نقشه را در ساده سازی تابع بول نشان می‌دهند. در مثال اول تابع تولی زیر را ساده می‌کنیم:

$$F(A, B, C) = \sum (3, 4, 6, 7)$$

نقشه سه متغیره این تابع در شکل ۸-۱ نشان داده شده است. در این نقشه چهار مربع با 1 مشخص شده‌اند که هر کدام متعلق به یکی از مینترم‌هایی است که برای تابع، 1 را ایجاد می‌نمایند. این مربع‌ها متعلق به مینترم‌های 3، 4، 6 و 7 بوده و از شکل ۷-۱ (ب) تشخیص داده می‌شوند. در ستون سوم دو



شکل ۸-۱ نقشه برای $F(A, B, C) = \sum (3, 4, 6, 7)$

مربع مجاور با هم ترکیب می شوند. این ستون متعلق به B و C بوده و جمله BC را تولید می کنند. دو مربع باقیمانده حاوی 1 در دو گوشه سطر دوم، مجاورند و متعلق به سطر A و دو ستون C' می باشند. بنابراین این دو جمله AC' را تولید می کنند. عبارت جبری ساده شده برای تابع عبارتست از OR دو جمله:

$$F = BC + AC'$$

در مثال دوم تابع بولی زیر را ساده می کنیم:

$$F(A, B, C) = \sum (0, 2, 4, 5, 6)$$

پنج مینترم در مربع های مربوطه خود در نقشه سه متغیره در شکل ۹-۱ با 1 علامت زده شده اند. چهار مربع در اولین و چهارمین ستون مجاورند و جمله C' را بدست می دهند. تنها مربع باقیمانده که با 1 مشخص شده متعلق به مینترم 5 بوده و قابل ترکیب با مربع مینترم 4 می باشد که جمله AB' از آن نتیجه می شود. تابع ساده شده عبارتست از:

$$F = C' + AB'$$

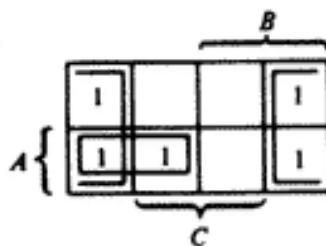
مثال سوم نیاز به یک نقشه چهار متغیره دارد

$$F(A, B, C, D) = \sum (0, 1, 2, 6, 8, 9, 10)$$

بخشی از نقشه که توسط این تابع چهار متغیره پوشش یافته است مربع هایی است که در شکل ۱۰-۱ با 1 مشخص شده اند. تابع دارای 1 هایی در چهار گوشه نقشه است که اگر بصورت یک گروه با هم ترکیب شوند

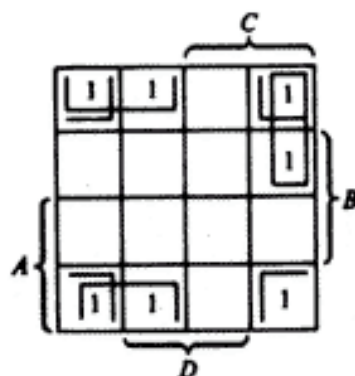
شکل ۹-۱ نقشه برای

$$F(A, B, C) = \sum (0, 2, 4, 5, 6)$$



شکل ۱۰-۱ نقشه برای

$$F(A, B, C, D) = \sum (0, 1, 2, 6, 8, 9, 10)$$



جمله $B'D'$ را تولید می کنند. دلیل ترکیب این مریعات مجاور بودن آنها بعلت مجاور بودن لبه های بالا و پایین و یا چپ و راست است که جمله $B'C'$ را بدست می دهند. تنها 1 باقیمانده در مربع مربوط به میترم 6 است که با میترم 2 ترکیب می شود و جمله $A'CB'$ را می سازد. تابع ساده شده عبارتست از

$$F = B'D' + B'C' + A'CD'$$

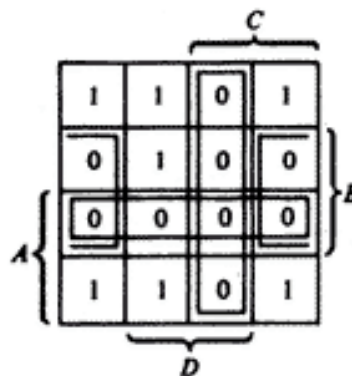
ساده سازی با ضرب حاصل جمع ها

در تمام مثال های قبلی توابع بول حاصل از نقشه ها به فرم جمع حاصلضرب ها بیان شده بودند. جملات ضرب در واقع AND جملات و جمع آنها بمعنی OR این جملات است. گاهی هم مناسبتر است تا یک عبارت جبری را بصورت ضرب حاصل جمع ها بدست آوریم. مجموع، همان OR جملات و حاصلضرب نیز AND آنها هستند. با کمی تصحیح، ضرب حاصل جمع ها را از نقشه می توان بدست آورد. روش بدست آوردن عبارت ضرب حاصل جمع ها، از خصوصیات اساسی جبر بول حاصل می گردد. 1 ها در نقشه نشاندهنده میترم هایی است که برای تابع تولید 1 می نمایند. مربع هایی که با 1 پر نشده اند، برای تابع 0 تولید می کنند. اگر ما مربع های خالی را با 0 پر کنیم و سپس آنها را با هم بر اساس مربع های مجاور ترکیب نمائیم، متمم تابع را خواهیم داشت. با متمم نمودن F' ، عبارتی برای F بصورت ضرب حاصل جمع ها بدست می آید. بهترین روش برای درک بهتر ارائه یک مثال است. می خواهیم تابع بولی زیر را به هر دو روش جمع حاصلضرب ها و ضرب حاصل جمع ها ساده کنیم.

$$F(A, B, C, D) = \sum (0, 1, 2, 5, 8, 9, 10)$$

1 های موجود در نقشه شکل ۱-۱۱ نشان دهنده تمام میترم هایی است که 1 را برای تابع تولید می کنند. مربع هایی که با 0 مشخص شده اند نشان دهنده میترم هایی هستند که در F وجود ندارند و بنابراین متمم F می باشند. ترکیب مربع های 1، فرم ساده شده تابع را بصورت مجموع حاصلضرب ها فراهم می کند:

$$F = B'D' + B'C' + A'CD'$$



شکل ۱-۱۱ نقشه برای

$$F(A, B, C, D) = \sum (0, 1, 2, 5, 8, 9, 10)$$

اگر مربع های 0 ترکیب شوند، طبق شکل، متمم ساده شده تابع بدست می آید.

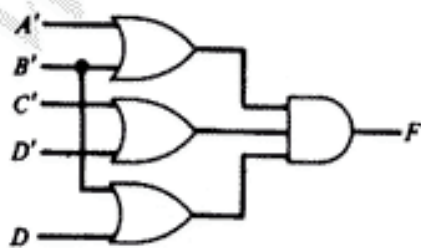
$$F' = AB + CD + BD'$$

با متمم گیری از F' فرم ساده شده تابع ضرب حاصل جمع ها بدست می آید:

$$F = (A' + B')(C' + D')(B' + D)$$

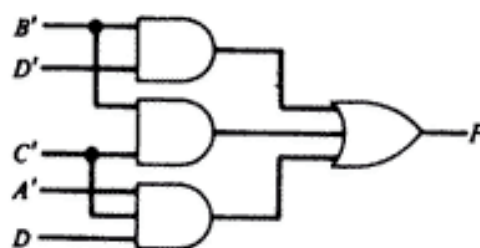
دیاگرام های منطقی دو عبارت ساده شده در شکل ۱۲-۱ نشان داده شده اند. عبارت جمع حاصل ضرب با تعدادی گیت AND در شکل ۱۲-۱ (الف) پیاده سازی شده است که هر عبارت مربوط به یک گیت AND می باشد. خروجی های گیت های AND به ورودی های یک گیت OR متصل است. همان تابع یکمک گیت های OR بصورت ضرب حاصل جمع ها در شکل ۱۲-۱ (ب) پیاده سازی شده است. خروجی های گیت های OR به ورودی های گیت AND متصل شده اند و هر گیت OR نیز متناظر با یک جمله OR است. در هر یک از دو فرم فرض شده است که متغیرهای ورودی مستقیماً بصورت متمم نیز موجود باشند، بنابراین معکوس کننده ای در مدار بکار نرفته است. الگوی بکار رفته در شکل ۱۲-۱ فرم کلی پیاده سازی هر تابع بولی است که به یکی از شکل های استاندارد بیان شده باشد. در فرم جمع حاصل ضرب ها تعدادی گیت AND به یک گیت OR متصل می شوند. در نوع ضرب حاصل جمع ها تعدادی گیت OR به یک گیت AND وصل می گردند.

همانطور که در شکل ۱۳-۱ (الف) نشان داده شده است جمع حاصل ضرب ها می تواند با گیت های NAND پیاده سازی شود. توجه کنید که دومین گیت NAND با استفاده از سمبل گرافیکی شکل ۱-۵ (ب) رسم شده است. دوایر کوچکی در دو انتهای سه خط از این دیاگرام دیده می شوند. دو دایره در یک خط به معنی دوبار متمم شدن می باشد، و چون $(X')' = X$ می باشد دو دایره را می توان حذف



(ب) ضرب حاصل جمع ها

$$F = (A' + B')(C' + D')(B' + D)$$

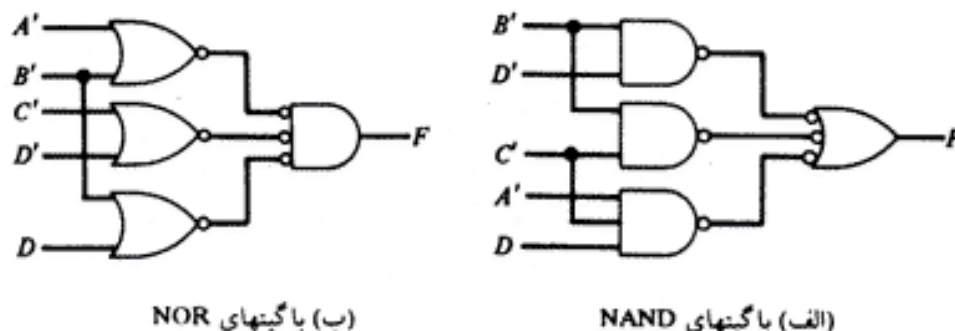


(الف) جمع حاصل ضرب ها

$$F = B'D' + B'C' + A'C'D$$

شکل ۱۲-۱ دیاگرام های منطقی با گیت های AND و OR

کرد و دیاگرام نتیجه معادل با مدار ۱۲-۱ (الف) خواهد بود. بطور مشابه عبارت ضرب حاصل جمع‌ها توسط گیت‌های NOR طبق شکل ۱۳-۱ (ب) پیاده سازی می‌شود. گیت NOR دوم با استفاده از سمبل گرافیکی شکل ۴-۱ (ب) رسم شده است. مجدداً دو دایره در دو انتهای هر خط حذف می‌شوند، و دیاگرام حاصله معادل شکل ۱۲-۱ (ب) خواهد بود.



شکل ۱۳-۱ دیاگرام‌های منطقی با گیت‌های NAND و NOR

حالات بی‌اهمیت

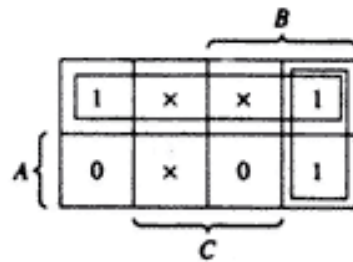
۱ها و ۰ها در نقشه مینترم‌هایی را نشان می‌دهند که بترتیب تابع را ۱ یا ۰ می‌کنند. گاهی مهم نیست که تابع به ازاء یک مینترم معین، ۰ ایجاد کند و یا ۱. چون تابع ممکن است تابع ۰ یا ۱ باشد، می‌گوییم اهمیتی ندارد که خروجی تابع برای این مینترم چیست. مینترم‌هایی که ۰ یا ۱ را برای یک تابع تولید می‌نمایند حالات بی‌اهمیت خوانده شده و با علامت X در نقشه مشخص می‌شوند. این حالات بی‌اهمیت را می‌توان برای ساده‌تر کردن عبارت جبری بکار برد.

وقتی که مربع‌های مجاور برای تابع در نقشه انتخاب می‌شوند، Xها ممکن است ۰ یا ۱ فرض شوند، بسته به اینکه کدام یک عبارت ساده‌تری را فراهم آورند. بعلاوه اگر یک X به ساده شدن تابع کمک نکند لزومی به استفاده از آن نیست. در هر حال، انتخابی که بعمل می‌آید فقط به حد ساده سازی حاصل بستگی دارد. بعنوان مثال، تابع بولی زیر را با جملات بی‌اهمیت آن در نظر بگیرید.

$$F(A, B, C) = \sum (0, 2, 6)$$

$$d(A, B, C) = \sum (1, 3, 5)$$

مینترم‌هایی که در F آمده‌اند مقدار ۱ را برای تابع تولید می‌کنند. مینترم‌های بی‌اهمیت که در d آمده‌اند هر یک از دو مقدار ۰ یا ۱ را می‌توانند تولید نمایند. باقیمانده مینترم‌ها یعنی مینترم‌های ۴ و ۷ مقدار ۰ را برای تابع تولید می‌کنند. نقشه در شکل ۱۴-۱ نشان داده شده است. مینترم‌های F با ۱، مینترم‌های d با X، و مربع‌های باقیمانده با ۰ مشخص شده‌اند. ۱ها و Xها به هر روش مناسبی با هم ترکیب می‌شوند



شکل ۱۴-۱ مثالی از نقشه با حالات بی‌اهمیت

تا حداکثر تعداد مربع‌های مجاور را در برگیرند. لزومی ندارد که تمام یا هر یک از X ها را در این ترکیب‌ها در نظر بگیریم ولی تمام 1 ها باید در نظر گرفته شوند. با جایگزینی X ها در مینترم‌های 1، 3 با 1 ها، جمله A' را از سطر اول بدست می‌آوریم. 1 باقیمانده برای مینترم 6 هم با مینترم 2 ترکیب شده و از آن جمله BC' بدست می‌آید. عبارت ساده شده بصورت زیر است:

$$F = A' + BC'$$

توجه کنید که مینترم بی‌اهمیت 5 در ترکیب گنجانده نشد زیرا به ساده شدن عبارت کمکی نمی‌نماید. همچنین اگر مینترم‌های بی‌اهمیت 1 و 3 با 1 جایگزین نشده بودند عبارت ساده شده بصورت زیر می‌بود

$$F = A'C' + BC'$$

این عبارت به دو گیت AND و یک گیت OR نیاز دارد در حالی که عبارت قبلی فقط به یک AND و یک OR نیاز داشت.

اگر به X های نقشه 0 یا 1 اختصاص یابد تابع بطور کامل مشخص می‌گردد. بنابراین عبارت

$$F = A' + BC'$$

فرم ساده شده تابع

$$F(A, B, C) = \sum (0, 1, 2, 3, 6)$$

است. تابع ساده شده ترکیبی از مینترم‌های اصلی 0، 2 و 6 و مینترم بی‌اهمیت 1 و 3 است. مینترم 5 در تابع گنجانده نشده است. از آنجا که مینترم‌های 1، 3 و 5 بعنوان حالات بی‌اهمیت در تابع در نظر گرفته شدند ما مینترم‌های 1 و 3 را با مقدار 1 و مینترم 5 را 0 انتخاب کردیم. این انتخاب باین علت صورت گرفت تا ساده‌ترین فرم برای عبارت بولی حاصل شود.

۵-۱ مدارهای ترکیبی

یک مدار ترکیبی آرابشی از گیت‌های منطقی متصل بهم با مجموعه‌ای از ورودی‌ها و خروجی‌هاست. در

هر لحظه از زمان، مقادیر دودویی خروجی ها تابعی از ترکیب دودویی ورودی هاست. بلاک دیاگرام مدار ترکیبی در شکل ۱-۱۵ دیده می شود. n متغیر دودویی ورودی از یک منبع خارجی سرچشمه گرفته و m متغیر دودویی خروجی هم به یک مقصد خارجی می روند و در بین این دو، اتصالات داخلی گیت های منطقی قرار دارد. یک مدار ترکیبی اطلاعات دودویی داده های ورودی را به داده های خروجی تبدیل می نماید. مدارهای ترکیبی در کامپیوترهای دیجیتال به منظور تولید تصمیمات کنترلی دودویی استفاده می شوند. همچنین بخش های لازم برای پردازش داده هم از این مدارات ساخته شده اند.

مدار ترکیبی بوسیله جدول درستی اش که رابطه دودویی بین n متغیر ورودی و m متغیر خروجی را نشان می دهد توصیف می شود. جدول درستی مقادیر دودویی خروجی را برای هر 2^n ترکیب ورودی نشان می دهد. مدار ترکیبی می تواند بوسیله m تابع بولی، که هر یک متعلق به یک متغیر خروجی است، مشخص شود. هر تابع برحسب n متغیر ورودی بیان می گردد.

تحلیل یک مدار ترکیبی با دیاگرام مدار منطقی شروع و با مجموعه ای از توابع بولی یا جدول درستی ختم می گردد. اگر مدار دیجیتال بوسیله توصیف کلامی تابعش بیان شود، تابع بولی یا جدول درستی برای تحقیق عملکرد آن کافی است. اگر هدف کار مدار باشد، باید عملکرد آن را با استفاده از توابع بولی یا جدول درستی بدست آمده تفسیر کرد. موفقیت در این کار مستلزم تجربه و آشنایی با مدارهای دیجیتال است. توانایی در مرتبط ساختن یک جدول درستی یا مجموعه ای از توابع بولی با یک کار پردازش اطلاعات هنری است که از تجربه فرد حاصل می شود.

طراحی مدارهای ترکیبی از تشریح کلامی مسئله آغاز می شود و به یک دیاگرام مدار منطقی ختم می گردد. رویه طراحی مراحل زیر را شامل می شود

۱- بیان مسئله

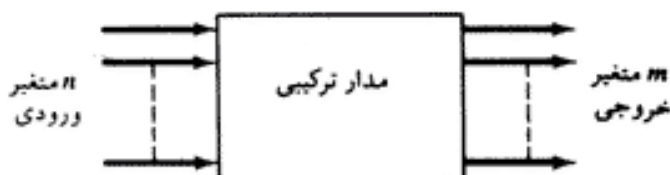
۲- اختصاص سمبل های حرفی به متغیرهای ورودی و خروجی

۳- بدست آوردن جدول درستی که رابطه بین ورودی ها و خروجی ها را تعریف می کند

۴- بدست آوردن توابع بولی برای هر یک از خروجیها

۵- رسم دیاگرام منطقی

ما برای نشان دادن روش طراحی مدارهای ترکیبی، دو مثال از مدارهای حسابی را ارائه می نمایم.



شکل ۱-۱۵ بلاک دیاگرام یک مدار ترکیبی

این مدارها به عنوان بلاک‌های ساختمانی پایه در ساخت مدارهای حسابی پیچیده‌تر بکار می‌روند.

نیم جمع کننده^۱

اساسی‌ترین مدار محاسباتی دیجیتال، مدار جمع دو رقم دودویی است. مدار ترکیبی که جمع حسابی دو بیت را انجام می‌دهد نیم جمع کننده (HA) نامیده می‌شود. مداری که جمع سه بیت را انجام می‌دهد (جمع دو بیت و بیت رقم نقلی قبلی) تمام جمع کننده^۲ خوانده می‌شود. نام تمام جمع کننده از این حقیقت ناشی می‌شود که برای پیاده سازی یک تمام جمع کننده، دو نیم جمع کننده لازم است. متغیرهای ورودی یک نیم جمع کننده مضاف و مضاف‌الیه خوانده می‌شوند. متغیرهای خروجی نیز مجموع و رقم نقلی نام دارند. در این مدار دو خروجی باید وجود داشته باشد زیرا جمع $1 + 1$ عدد دودویی 10 است که دو رقم دارد. ما سمبل‌های x و y را به دو متغیر ورودی و S و C را به دو متغیر خروجی اختصاص می‌دهیم. جدول درستی نیم جمع کننده در شکل ۱-۱۶ (الف) نشان داده شده است. خروجی C برابر 0 است مگر اینکه هر دو ورودی 1 باشند. خروجی S بیت کم ارزشتر^۵ (پایین رتبه) مجموع می‌باشد. توابع بولی دو خروجی را می‌توان مستقیماً از جدول درستی بدست آورد.

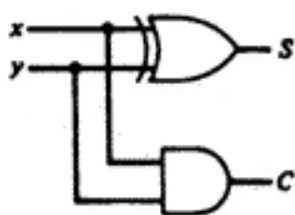
$$S = x'y + xy' = x \oplus y$$

$$C = xy$$

دیاگرام منطقی تابع در شکل ۱-۱۶ (ب) نشان داده شده است. این دیاگرام از یک گیت OR انحصاری و یک گیت AND تشکیل شده است.

تمام جمع کننده

یک تمام جمع کننده (FA) مداری ترکیبی است که مجموع حسابی سه بیت را تشکیل می‌دهد. مدار



(ب) دیاگرام منطقی

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(الف) جدول درستی

شکل ۱-۱۶ نیم جمع کننده

1- Half Adder

2- Full Adder

3- Sum

4- Carry

5- Least Significant

دارای سه ورودی و دو خروجی است. دو متغیر ورودی که با x و y مشخص شده‌اند بیانگر دو بیت با معنایی هستند که باید جمع شوند. ورودی سوم یعنی z رقم نقلی حاصل از عمل جمع در مکان کم ارزشتر قبلی در عدد است. برای مدار دو خروجی لازم است زیرا جمع حسابی سه رقم دودویی بین 0 تا 3 است و اعداد دودویی 2 یا 3 به دو رقم نیاز دارند. خروجی‌ها با S و C مشخص شده‌اند. متغیر S مقدار کم ارزشتر مجموع و C مقدار نقلی خروجی که مقدار با ارزشتر مجموع نیز هست را می‌دهند. جدول درستی تمام جمع‌کننده در جدول 1-2 نشان داده شده است. هشت سطر زیر متغیرهای ورودی تمام ترکیب‌های ممکن را که متغیرها می‌توانند اختیار کنند مشخص می‌نمایند. مقادیر متغیرهای خروجی از حاصل جمع بیت‌های ورودی معین می‌شوند. وقتی همه بیت‌ها 0 باشند، خروجی 0 است. خروجی S هنگامی 1 می‌شود که فقط یک ورودی یا همه ورودی‌ها 1 باشند. خروجی C نیز برابر 1 است اگر هر دو یا هر سه ورودی 1 باشند.

نقشه‌های شکل 1-17 برای یافتن عبارات جبری دو متغیر خروجی بکار می‌روند. 1ها در مربع‌های نقشه‌های S و C مستقیماً از میترم‌های جدول درستی تعیین می‌شوند. مربعات مربوط به خروجی S که در آن‌ها 1ها قرار دارند با هم ترکیب نمی‌شوند زیرا با هم مجاور نیستند. ولی چون خروجی هنگامی 1 است که تعداد فردی از ورودی‌ها 1 باشند، پس S یک تابع فرد بوده و رابطه OR انحصاری بین ورودی‌ها برقرار خواهد بود (بحث انتهای بخش 2-1 ملاحظه شود). مربع‌هایی که برای C مقدار 1 دارند را می‌توان بطرق مختلف با هم ترکیب کرد. یک عبارت ممکن برای C بفرم زیر است:

$$C = x'y + (xy + xy')z$$

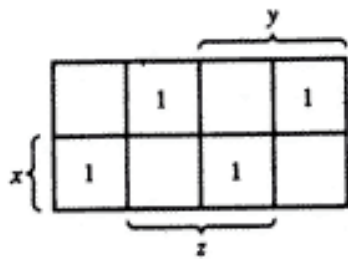
نظر به رابطه $x'y + xy' = x \oplus y$ ، دو عبارت بولی مربوط به تمام جمع‌کننده بصورت زیر خواهد بود

$$S = x \oplus y \oplus z$$

$$C = xy + (x \oplus y)z$$

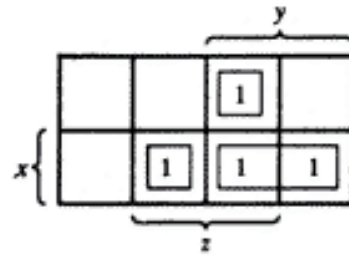
جدول 1-2 جدول درستی برای تمام جمع‌کننده

خروجی‌ها			ورودی‌ها	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$$S = x'y'z' + x'y'z + xy'z' + xyz$$

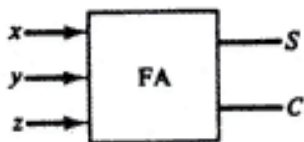
$$= x \oplus y \oplus z$$



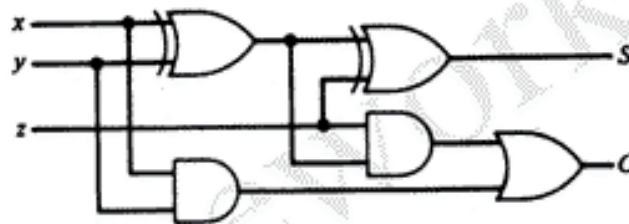
$$C = xy + xz + yz$$

$$= xy + (x'y + xy'z)$$

شکل ۱۷-۱ نقشه‌ها برای تمام جمع‌کننده



(ب) بلاک دیاگرام



(الف) دیاگرام منطقی

شکل ۱۸-۱ مدار تمام جمع‌کننده

دیاگرام منطقی تمام جمع‌کننده در شکل ۱۸-۱ رسم شده است. توجه کنید که مدار تمام جمع‌کننده از دو نیم جمع‌کننده و یک گیت OR ساخته شده است. هنگام استفاده از یک تمام جمع‌کننده (FA)، شکل ۱۸-۱ (ب) بکار گرفته خواهد شد.

۱-۶ فلیپ فلاپها^۱

تاکنون مدارهای بررسی شده دیجیتال از نوع ترکیبی بودند، که در آنها خروجی‌ها در هر لحظه کلاً به ورودی‌های همان لحظه بستگی داشتند. هرچند هر سیستم دیجیتال احتمالاً یک مدار ترکیبی دارد، اغلب سیستم‌هایی که در عمل با آنها مواجه می‌شویم دارای عناصر حافظه نیز هستند و لذا سیستم باید در چهارچوب مدارهای ترتیبی مورد بررسی قرار گیرد. متداول‌ترین نوع مدار ترتیبی نوع همگام^۲ (همزمان یا سنکرون) آن است. مدارهای ترتیبی همگام در لحظه‌های گسسته و معینی از زمان بر عناصر حافظه اثر می‌گذارند. همزمان سازی با یک وسیله زمانبندی یا یک مولد پالس ساعت^۳ حاصل می‌شود که رشته‌ای از پالس‌های ساعت را بطور پی‌در‌پی تولید می‌نماید. پالس‌های ساعت در سراسر سیستم توزیع می‌شوند و عناصر حافظه فقط با رسیدن پالس همگام‌کننده تأثیر می‌پذیرند. مدارهای همگام

1- Flip Flop

2- Synchronous

3- Clock Pulse Generator

ساعت دار مدارهایی هستند که بیش از همه در عمل با آنها مواجه می شویم. این مدارها ندرتاً با مسئله ناپایداری روبرو هستند و زمانبندی آنها به مقاطع زمانی مستقل قابل تفکیک می باشند. هر یک از این مقاطع بطور جداگانه قابل بررسی می باشند.

عناصر حافظه بکار رفته در مدارهای ترتیبی ساعت دار فلیپ فلاپ خوانده می شوند. فلیپ فلاپ یک سلول^۱ دودویی است که می تواند یک بیت اطلاعات را در خود ذخیره نماید. این سلول دارای دو خروجی است، یکی برای مقدار عادی بیتی که در آن ذخیره می شود و دیگری مقدار متمم آن. فلیپ فلاپ یک وضعیت^۲ دودویی را در خود نگهدارند تا اینکه یک پالس ساعت موجب تغییر آن حالت شود. تفاوت بین انواع فلیپ فلاپها در تعداد ورودی های آنها و نحوه تأثیر این ورودی ها بر وضعیت یا حالت دودویی آنهاست. متداول ترین انواع فلیپ فلاپها در بخش های زیر ارائه شده اند.

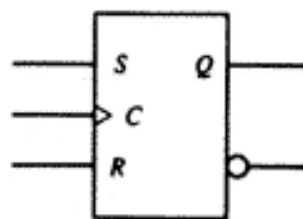
فلیپ فلاپ SR

سمبل گرافیکی فلیپ فلاپ SR در شکل ۱۹-۱ (الف) دیده می شود. این فلیپ فلاپ دارای سه ورودی با نام های نشاندن^۳ (S) حذف کردن^۴ (R) و ساعت^۵ (C) است. خروجی فلیپ فلاپ Q نامیده می شود و گاهی دارای خروجی متمم شده نیز هست که با دایره کوچکی در پایانه خروجی دیگر مشخص شده است. در جلو حرف C یک نوک فلش قرار گرفته که بمعنی دینامیک بودن (پویا بودن) ورودی است. سمبل دینامیکی نشاندهنده این واقعیت است که فلیپ فلاپ به گذار مثبت (لبه بالا رونده) پالس پاسخ می دهد.

عملکرد فلیپ فلاپ SR بشرح زیر است. اگر سیگنالی در ورودی ساعت C وجود نداشته باشد مقادیر در ورودی های S و R هرچه باشند، خروجی مدار تغییری نمی یابد. فقط وقتی که سیگنال ساعت از 0 به 1 تغییر نماید خروجی تحت تأثیر ورودی های S و R قرار خواهد گرفت. بهنگام تغییر C از 0 به

S	R	Q (t+1)	
0	0	Q (t)	بلا تغییر
0	1	0	پاک شدن، 0
1	0	1	نشانده شدن، 1
1	1	?	نامعین

(ب) جدول مشخصه



(الف) سمبل گرافیکی

شکل ۱۹-۱ فلیپ فلاپ SR

1- Cell

2- State

3- Set

4- Reset

5- Clock

1 اگر $S = 1$ و $R = 0$ باشند در خروجی Q مقدار 1 نشانده می شود. اگر در تغییر C از 0 به 1 مقادیر $S=0$ و $R=1$ باشند خروجی Q با 0 پاک می شود. اگر در گذار ساعت (از 0 به 1)، S و R هر دو 0 باشند خروجی تغییری نمی کند. هنگامی که S و R هر دو 1 باشند، خروجی غیر قابل پیش بینی است و به تأخیرهای داخلی که در مدار رخ می دهد بستگی داشته و ممکن است 0 و یا 1 گردد.

جدول مشخصه شکل ۱۹-۱ (ب) عملکرد یک فلیپ فلاپ SR را جدول وار نشان می دهد. در ستون های S و R مقادیر دودویی دو ورودی مشخص شده اند. $Q(t)$ نیز حالت خروجی Q در یک زمان معین است (که به آن حالت فعلی می گویند). $Q(t+1)$ حالت دودویی خروجی Q پس از لبه مثبت پالس ساعت است (حالت بعدی). اگر $S = R = 0$ باشد، لبه (گذار) پالس تغییری در حالت بوجود نمی آورد [یعنی $Q(t+1) = Q(t)$]. اگر $S = 0$ و $R = 1$ باشد فلیپ فلاپ به حالت 0 (پاک شده) می رود. اگر $S = 1$ و $R = 0$ باشد فلیپ فلاپ به وضعیت 1 (نشانده شده) می رود. به فلیپ فلاپ در $S=1$ و $R=1$ نباید پالس داد زیرا تولید حالت بعدی نامعینی را خواهد کرد. این حالت نامعین کنترل فلیپ فلاپ را مشخص می سازد و لذا در عمل بندرت مورد استفاده قرار می گیرد.

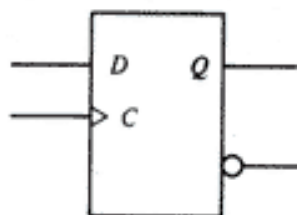
فلیپ فلاپ D

فلیپ فلاپ D (داده) با تغییر کوچکی در فلیپ فلاپ SR بدست می آید. فلیپ فلاپ SR با قرار دادن یک معکوس کننده بین S و R و اختصاص سمبل D به تنها ورودی حاصل، به فلیپ فلاپ D تبدیل می شود. ورودی D هنگام انتقال پالس ساعت از 0 به 1 نمونه برداری می شود. اگر $D = 1$ باشد، خروجی فلیپ فلاپ به حالت 1 می رود، اما در $D = 0$ ، خروجی فلیپ فلاپ به وضعیت 0 خواهد رفت.

سمبل گرافیکی و جدول مشخصه فلیپ فلاپ D در شکل ۲۰-۱ نشان داده شده است. از جدول مزبور درمی یابیم که حالت بعدی $Q(t+1)$ توسط ورودی D معین می شود. این ارتباط بوسیله معادله

D	$Q(t+1)$
0	0 پاک شدن، 0
1	1 نشانده شدن، 1

(ب) جدول مشخصه



(الف) سمبل گرافیکی

شکل ۲۰-۱ فلیپ فلاپ D

مشخصه زیر بیان می‌گردد

$$Q(t+1) = D$$

این بدان معنی است که خروجی Q فلیپ فلاپ مقدار خود را، هر بار بهنگام انتقال سیگنال ساعت از 0 به 1 از ورودی D دریافت می‌کند.

دقت کنید که حالتی از فلیپ فلاپ وجود ندارد که خروجی را تغییر ندهد. هرچند فلیپ فلاپ D از مزیت داشتن فقط یک ورودی (بدون احتساب C) بهره می‌برد ولی عیب آن این است که جدول مشخصه آن فاقد حالت "بلا تغییر" $Q(t+1) = Q(t)$ می‌باشد. در این فلیپ فلاپ حالت بلا تغییر بدین صورت انجام می‌شود که یا باید سیگنال ساعت غیر فعال گردد و یا خروجی فلیپ فلاپ به ورودی آن بازگردانده^۲ شود، بطوری که پالس‌های ساعت حالت فلیپ فلاپ را تغییر ندهند.

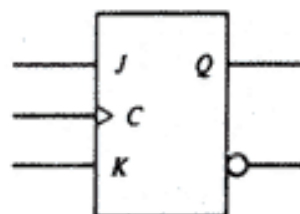
فلیپ فلاپ JK

یک فلیپ فلاپ JK نوع اصلاح شده فلیپ فلاپ SR است که حالت "نامعین"^۳ در فلیپ فلاپ SR در آن "معین" شده است. ورودی‌های J و K در نشانیدن و یا پاک کردن فلیپ فلاپ بترتیب مانند S و R عمل می‌کنند. وقتی هر دو ورودی J و K برابر 1 باشند، انتقال یا گذار پالس ساعت خروجی‌های فلیپ فلاپ را به حالت متممشان سوئیچ می‌کنند.

سمبل گرافیکی و جدول مشخصه فلیپ فلاپ JK معادل با ورودی R در شکل ۱-۲۱ نشان داده شده است. ورودی J معادل ورودی S و ورودی K معادل ورودی R در فلیپ فلاپ SR است. در فلیپ فلاپ JK اگر هر دو ورودی J و K برابر 1 باشند، بجای حالت نامعین، حالت متمم $Q(t+1) = Q'(t)$ را خواهیم داشت.

J	K	$Q(t+1)$	
0	0	$Q(t)$	بلا تغییر
0	1	0	پاک شدن، 0
1	0	1	نشانه شدن، 1
1	1	$Q'(t)$	متمم

(ب) جدول مشخصه



(الف) سمبل گرافیکی

شکل ۱-۲۱ فلیپ فلاپ JK

1- Unchanged

2- Feedback

3- Indeterminate

فلیپ فلاپ T^۱

نوع دیگری از فلیپ فلاپها که در کتب درسی دیده می شود فلیپ فلاپ T می باشد. این فلیپ فلاپ، که در شکل ۱-۲۲ نشان داده شده است از فلیپ فلاپ نوع JK و با اتصال ورودی های J و K به یکدیگر و در نتیجه ایجاد یک ورودی واحد که با T نشان داده می شود بدست می آید. بنابراین فلیپ فلاپ T فقط دو حالت دارد. وقتی $T = 0$ باشد $J = K = 0$ یک گذار ساعت حالت فلیپ فلاپ را تغییر نمی دهد. وقتی $T = 1$ باشد $J = K = 1$ انتقال ساعت حالت فلیپ فلاپ را متمم می کند. این حالات را می توان با معادله مشخصه زیر بیان کرد

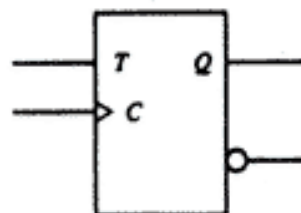
$$Q(t+1) = Q(t) \oplus T$$

فلیپ فلاپ های تحریک شونده با لبه^۲

متداول ترین نوع فلیپ فلاپ های مورد استفاده در همگام کردن تغییر حالت به هنگام تغییر وضعیت یا گذار پالس ساعت فلیپ فلاپ تحریک شونده با لبه (یا فلیپ فلاپ حساس به لبه) می باشد. در این نوع فلیپ فلاپ ها، تغییرات در خروجی بازاء سطح خاصی از ورودی پالس ساعت رخ می دهد. وقتی که سطح ورودی پالس ساعت از این سطح آستانه^۳ تجاوز کند ورودی ها قفل می شوند بطوری که فلیپ فلاپ به تغییرات دیگر در ورودی، تا بازگشت پالس ساعت به 0 و ظهور پالس ساعتی دیگر، پاسخی نمی دهد. در بعضی از این نوع فلیپ فلاپ ها تأثیر ورودی ها در خروجی در لبه بالا رونده سیگنال ساعت (گذار یا انتقال در لبه مثبت) و برخی نیز در لبه پایین رونده سیگنال (گذار در لبه منفی) میسر است. شکل ۱-۲۳ (الف) سیگنال پالس ساعت را در یک فلیپ فلاپ D حساس به لبه مثبت نشان

T	Q(t+1)	
0	Q(t)	بلا تغییر
1	Q'(t)	متمم

(ب) جدول مشخصه



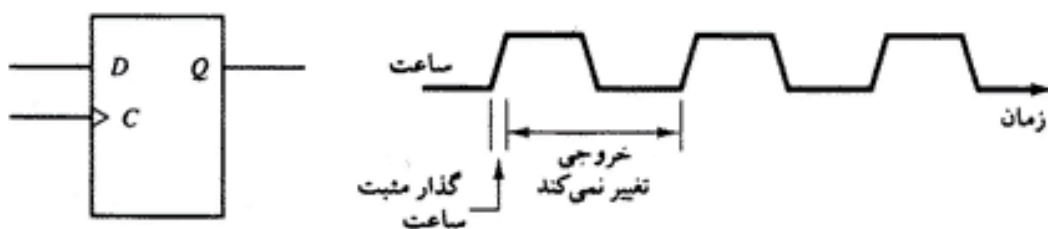
(الف) سیمبل گرافیکی

شکل ۱-۲۲ فلیپ فلاپ T

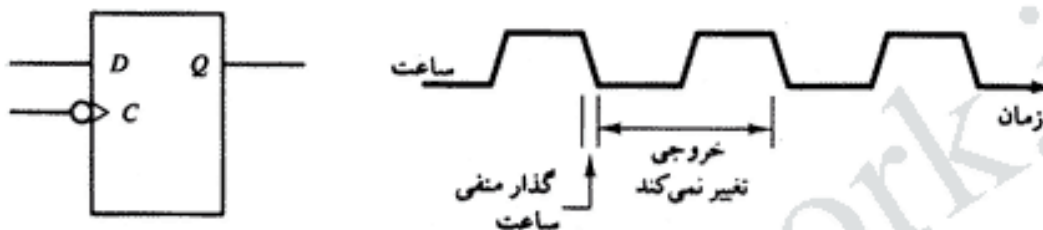
1- Toggle

2- Edge- Triggered Flip Flop

3- Threshold level



(الف) فلیپ فلاپ D تحریک شونده با لبه مثبت



(ب) فلیپ فلاپ D تحریک شونده با لبه منفی

شکل ۱-۲۳ فلیپ فلاپ تحریک شونده با لبه

می دهد. مقدار ورودی D در لبه مثبت پالس ساعت به خروجی Q منتقل می شود. هنگامی که پالس ساعت در سطح 1، در سطح 0 و یا در حال انتقال از سطح 1 به 0 باشد خروجی تغییری نمی کند. گذار یا انتقال مثبت پالس شامل دو فاصله زمانی است که یکی حداقل زمان آماده سازی^۱ می باشد و طی آن ورودی D باید در یک سطح ثابتی قبل از گذار پالس ساعت باقی بماند، و دیگری زمان معینی با نام زمان نگهداری^۲ است که طی آن ورودی D نباید پس از گذار پالس ساعت تغییر کند. زمان مربوط به انتقال یا گذار مثبت کسر بسیار کوچکی از کل دوره تناوب پالس ساعت است.

شکل ۱-۲۳ (ب) سمبل های گرافیکی و دیاگرام زمانی مربوط به فلیپ فلاپ D حساس به لبه منفی را نشان می دهد. در سمبل گرافیکی، یک دایره کوچک منفی کننده^۳ در جلو علامت دینامیکی در ورودی C قرار دارد. این علامت بیانگر تحریک فلیپ فلاپ در لبه منفی پالس است. در این حالت پاسخ فلیپ فلاپ در قبال انتقال سطح سیگنال ساعت از سطح 1 به 0 می باشد.

نوع دیگری از فلیپ فلاپ که در بعضی از سیستم ها بکار می رود فلیپ فلاپ ماستر-اسلیو^۴ (بمعنی حاکم و پیرو) است. این نوع مدار از دو فلیپ فلاپ تشکیل شده است. اولین فلیپ فلاپ ماستر نامیده می شود و به سطح مثبت پالس ساعت پاسخ می دهد، و دومین فلیپ فلاپ اسلیو می باشد که به سطح منفی پالس ساعت واکنش نشان می دهد. در نتیجه خروجی بهنگام تغییر 1 به 0 سیگنال ساعت تغییر

1- Set UP Time

2- Hold Time

3- Negation

4- Master - Slave

می نماید. امروزه گرایش در کنار گذاشتن فلیپ فلاپ ماستر-اسلیو و بکارگیری فلیپ فلاپ حساس به لبه دیده می شود.

فلیپ فلاپ های موجود بصورت مدار مجتمع گاهی دارای پایانه های خاصی برای نشان دادن^۱ و پاک کردن^۲ غیر همگام^۳ می باشند. این ورودی ها معمولاً نشان دادن اولیه^۴ و پاک کردن نامیده می شوند و با سطح منفی سیگنال ورودی خود بدون نیاز به پالس ساعت بر فلیپ فلاپ اثر می گذارند. این ورودی ها برای قرار دادن فلیپ فلاپ ها در یک وضعیت اولیه قبل از شروع بکار با پالس ساعت مفیدند.

جداول تحریک

جداول تحریک فلیپ فلاپ ها حالت بعدی را هرگاه ورودی ها و حالت فعلی معلوم باشند مشخص می کنند. معمولاً هنگام طراحی مدارهای ترتیبی انتقال از حالت فعلی به حالت بعدی برای ما شناخته شده است و ما مایلیم تا ورودی های فلیپ فلاپ را طوری بیابیم که انتقال مناسب انجام یابد. به همین دلیل به جدولی نیاز داریم که برحسب تغییر مفروضی در حالت، ورودی های مورد نیاز را نشان دهد. چنین لیستی از حالت ها، جدول تحریک فلیپ فلاپ نامیده می شود.

جدول ۱-۳ جداول تحریک چهار نوع فلیپ فلاپ را نشان می دهد. هر جدول متشکل از دو ستون $Q(t)$ و $Q(t+1)$ و یک ستون برای هر ورودی است که چگونگی وقوع انتقال مورد نظر را نشان می دهد. انتقال از حالت فعلی $Q(t)$ به حالت بعدی $Q(t+1)$ به چهار طریق امکان پذیر است.

جدول ۱-۳ جدول تحریک برای چهار فلیپ فلاپ

فلیپ فلاپ SR				فلیپ فلاپ D		
$Q(t)$	$Q(t+1)$	S	R	$Q(t)$	$Q(t+1)$	D
0	0	0	x	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	0
1	1	x	0	1	1	1

فلیپ فلاپ JK				فلیپ فلاپ T		
$Q(t)$	$Q(t+1)$	J	K	$Q(t)$	$Q(t+1)$	T
0	0	0	x	0	0	0
0	1	1	x	0	1	1
1	0	x	1	1	0	1
1	1	x	0	1	1	0

1- Set

2- Clear

3- asynchronous

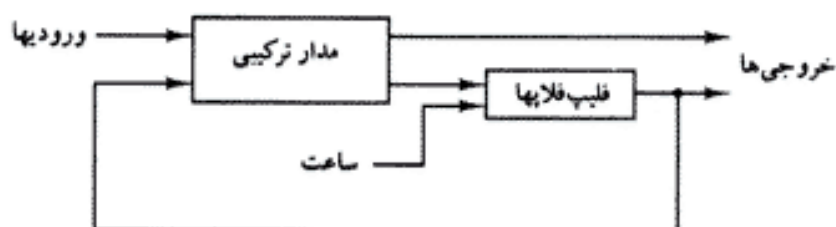
4- Preset

ورودی‌های مورد نیاز برای هر یک از چهار انتقال، از اطلاعات موجود در جدول مشخصه بدست آمده‌اند. سمبل X در جداول نشان‌دهنده حالت بی‌اهمیت است و باین معنی است که 0 یا 1 بودن ورودی مهم نیست.

دلیل وجود حالات بی‌اهمیت در جداول تحریک این است که برای دستیابی به یک انتقال مورد نظر دو راه وجود دارد. مثلاً در یک فلیپ فلاپ JK، انتقال از حالت فعلی 0 به حالت بعدی 0 را می‌توان با قرار دادن J و K در حالت 0 (برای حالت بلا تغییر) یا با قرار دادن $J = 0$ و $K = 1$ برای پاک کردن فلیپ فلاپ (گرچه از قبل پاک شده است) بدست آورد. در هر دو مورد J باید 0 باشد ولی K در مورد اول 0 و در مورد دوم 1 است. چون در هر صورت انتقال مورد نظر رخ خواهد داد، ما ورودی K را با X علامت می‌زنیم و لذا طراح اجازه خواهد داشت مقدار K را با توجه به اینکه کدام مناسبتر است، 1 یا 0 انتخاب کند.

۷-۱ مدارهای ترتیبی

مدار ترتیبی از اتصال مجموعه‌ای از فلیپ‌فلاپها و گیت‌ها بوجود می‌آید. گیت‌ها به تنهایی مدارهای ترکیبی را تشکیل می‌دهند ولی وقتی با فلیپ فلاپها همراه باشند مدار حاصل از رده مدارهای ترتیبی خواهد بود. بلوک دیاگرام یک مدار ترتیبی ساعت‌دار در شکل ۱-۲۴ نشان داده شده است. این مدار از یک مدار ترکیبی و تعدادی فلیپ فلاپ ساعت‌دار تشکیل شده است. بطور کلی، هر تعداد یا هر نوع فلیپ فلاپ می‌تواند در مدار بکار رود. همانطور که در دیاگرام نشان داده شده است، بلاک مدار ترکیبی سیگنالهای دودویی را از ورودی‌های خارجی و از خروجی‌های فلیپ‌فلاپها دریافت می‌کند. خروجی‌های مدار ترکیبی به خروجی‌های خارجی و به ورودی‌های فلیپ فلاپها می‌روند. گیت‌های تشکیل دهنده مدار ترکیبی مقادیر ذخیره شونده در فلیپ فلاپها را بعد از هر گذار پالس ساعت معین می‌کنند. خروجی فلیپ فلاپها به نوبه خود به ورودی‌های مدار ترکیبی اعمال شده و رفتار مدار را تعیین می‌کنند. این فرآیند نشان می‌دهد که خروجی‌های خارجی مدار ترتیبی تابعی از ورودی‌های خارجی و حالت فعلی^۱ فلیپ فلاپها است. بعلاوه، حالت بعدی^۲ فلیپ فلاپها نیز تابعی از حالت فعلی آنها و ورودی‌های خارجی



شکل ۱-۲۴ بلاک دیاگرام یک مدار ترتیبی همگام ساعت‌دار

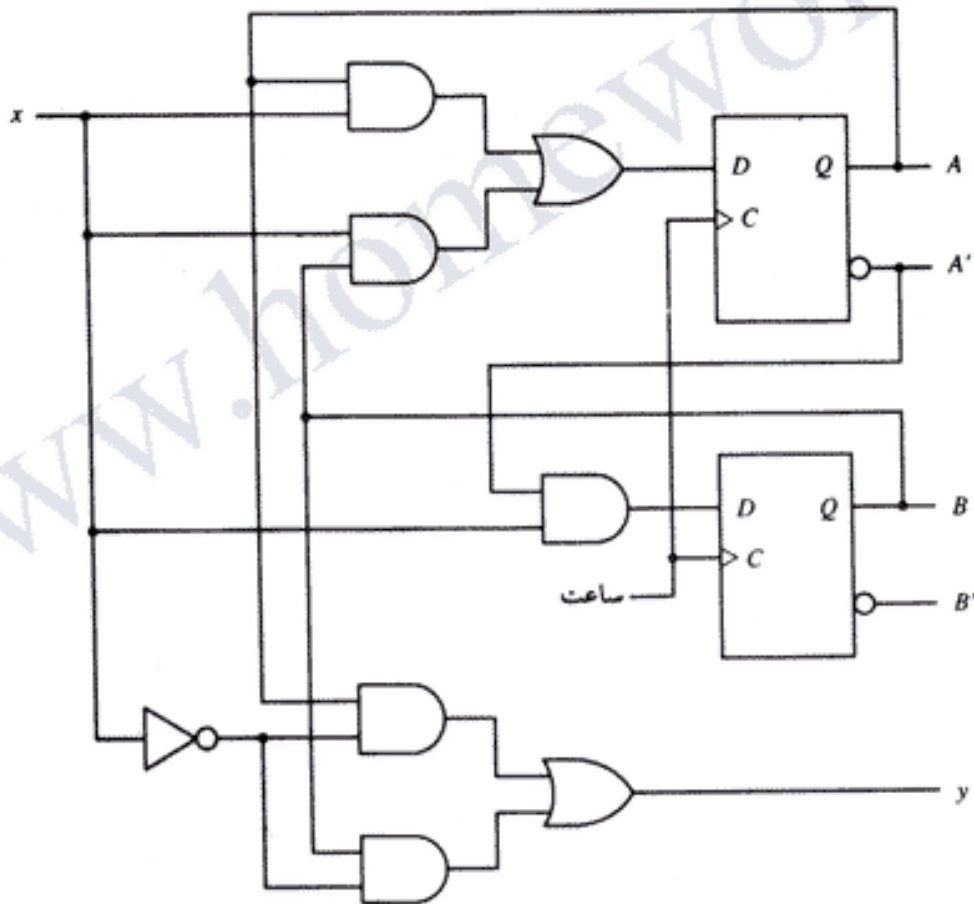
1- Present State

2- Next State

می باشد. بنابراین یک مدار ترتیبی با یک رشته ورودی های زمانبندی شده، خروجی های خارجی و حالات دودویی فلیپ فلاپهای داخلی اش مشخص می گردد.

معادلات ورودی فلیپ فلاپها^۱

مثالی از یک مدار ترتیبی در شکل ۱-۲۵ دیده می شود. این مدار دارای یک متغیر ورودی x ، یک متغیر خروجی y و دو فلیپ فلاپ D ساعت دار است. گیت های AND ، OR و معکوس کننده بخش ترکیبی مدار منطقی را تشکیل می دهند. اتصالات بین گیت ها در مدار ترکیبی می تواند بوسیله مجموعه ای از عبارات بولی مشخص گردد. بخشی از مدار ترکیبی که ورودی های فلیپ فلاپ را تولید می کنند بوسیله مجموعه ای از عبارات بولی که معادلات ورودی فلیپ فلاپها نام دارند توصیف می شود. ما سمبل ورودی هر فلیپ فلاپ را بعنوان متغیر معادله ورودی و اندیس را برای مشخص کردن سمبل



شکل ۱-۲۵ مثالی از یک مدار ترتیبی

1- Flip Flop Input Equation

انتخاب شده برمی‌گزینیم. بنابراین در شکل ۱-۲۵ ما دو معادله ورودی خواهیم داشت که با D_B و D_A مشخص می‌شوند. اولین حرف در هر مورد نماینده ورودی D یک فلیپ فلاپ D و حرف اندیس نام فلیپ فلاپ مربوطه است. معادلات ورودی، توابع بولی برای متغیرهای ورودی فلیپ فلاپ بوده و از بررسی مدار بدست می‌آیند. چون خروجی گیت OR به ورودی D از فلیپ فلاپ A متصل شده است ما اولین معادله ورودی را می‌نویسیم

$$D_A = Ax + Bx$$

که A و B خروجی‌های دو فلیپ فلاپ x و ورودی خارجی است. دومین معادله ورودی از خروجی گیت AND متصل به ورودی D فلیپ فلاپ B حاصل می‌شود.

$$D_B = A'x$$

این مدار ترتیبی یک خروجی خارجی نیز دارد که تابعی از متغیر ورودی و حالات فلیپ فلاپها است. این خروجی می‌تواند بوسیله عبارت جبری زیر مشخص شود

$$y = Ax' + Bx'$$

از این مثال در می‌یابیم که یک معادله ورودی فلیپ فلاپ عبارتی بولی برای یک مدار ترکیبی است. متغیر اندیس‌دار نام یک متغیر دودویی برای خروجی یک مدار ترکیبی است. این خروجی همواره به یکی از ورودی‌های فلیپ فلاپها متصل است.

جدول حالت^۱

رفتار یک مدار ترتیبی از ورودیها، خروجیها و حالت فلیپ فلاپهای آن معین می‌شود. خروجیها و حالت بعدی تابعی از ورودیها و حالت فعلی هستند. یک مدار ترتیبی توسط جدول حالتی مشخص می‌شود که خروجیها و حالات بعدی را به صورت تابعی از ورودی و حالات فعلی مرتبط می‌سازد. در مدارهای ترتیبی ساعت‌دار، انتقال از حالت فعلی به حالت بعدی بوسیله رخداد یک سیگنال ساعت ممکن می‌شود.

جدول حالت شکل ۱-۲۵ در جدول ۱-۴ نشان داده شده است. جدول متشکل از چهار بخش است که با عناوین حالت فعلی، ورودی، حالت بعد و خروجی مشخص شده‌اند. بخش حالت فعلی حالات فلیپ فلاپهای A و B را در هر لحظه از زمان مفروض t نشان می‌دهد. بخش ورودی مقداری از x را در هر حالت فعلی معین می‌سازد. بخش حالت بعدی حالات فلیپ فلاپها را در پرورد زمانی بعدی، $t+1$ ، بدست می‌دهد. بخش خروجی نیز مقدار y را در حالت فعلی و وضعیت ورودی می‌دهد.

1- State Table

جدول ۱-۴ جدول حالت برای مدار شکل ۱-۲۵

حالت فعلی		ورودی x	حالت بعدی		خروجی y
A	B		A	B	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

در تشکیل یک جدول حالت، ابتدا لیست تمام ترکیبات دودویی حالت فعلی و ورودی را می‌نویسیم. در این حالت ما هشت ترکیب دودویی از 000 تا 111 خواهیم داشت. سپس مقادیر حالت‌های بعدی از دیاگرام منطقی یا از معادلات ورودی بدست می‌آیند. معادله ورودی برای فلیپ فلاپ A برابر است با

$$D_A = Ax + Bx$$

مقدار حالت بعدی هر فلیپ فلاپ برابر با مقدار ورودی D در حالت فعلی است. انتقال از حالت فعلی به حالت بعدی پس از اعمال سیگنال ساعت بوقوع می‌پیوندد. بنابراین هرگاه حالات فعلی و ورودی شرایط $Ax = 1$ یا $Bx = 1$ را ایجاد کنند D_A برابر 1 و در نتیجه حالت بعدی A هم 1 خواهد شد. این مطلب در جدول حالت با سه عدد 1 زیر ستون حالت بعدی A آورده شده است. بطور مشابه، معادله ورودی برای فلیپ فلاپ B با رابطه زیر داده می‌شود.

$$D_B = A'x$$

حالت بعدی B در جدول حالت هنگامی 1 است که A برابر 0 و x برابر 1 باشد. ستون خروجی نیز از معادله خروجی زیر بدست می‌آید.

$$y = Ax' + Bx'$$

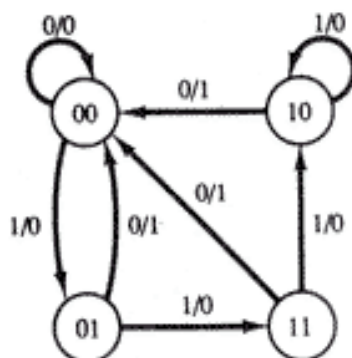
جدول حالت هر مدار ترتیبی با استفاده از روشی که در این مثال دیده شد بدست می‌آید. بطور کلی، یک مدار ترتیبی m فلیپ فلاپ، n متغیر ورودی و p متغیر خروجی دارای m ستون حالت فعلی، n ستون برای ورودی‌ها، m ستون برای حالت بعدی و p ستون برای خروجی‌هاست. ستون‌های حالت فعلی و ورودی با هم ترکیب شده و ما 2^{m+n} ترکیب دودویی، از 0 تا $2^{m+n} - 1$ را در زیر آنها می‌نویسیم. ستون‌های حالت بعدی و خروجی توابعی از حالت فعلی و مقادیر ورودی بوده و مستقیماً

از مدار یا معادلات بولی که مدار را توصیف می کنند بدست می آیند.

دیاگرام حالت

اطلاعات موجود در جدول حالت را می توان بصورت گرافیکی در یک دیاگرام حالت نشان داد. در این نوع دیاگرام، یک حالت بوسیله یک دایره و انتقال بین حالات بوسیله خطوط جهت داری که دایره را بهم وصل می کنند نشان داده می شود. دیاگرام حالت مدار ترتیبی شکل ۲۵-۱ در شکل ۲۶-۱ نشان داده شده است. دیاگرام حالت همان اطلاعات جدول حالت را در دسترس می گذارد. و مستقیماً از جدول ۴-۱ بدست می آید. عدد دودویی داخل هر دایره حالت فلیپ فلاپ ها را مشخص می کند. در کنار خطوط جهت دار دو عدد دودویی که با ممیز از هم جدا شده اند نوشته شده است. اولین عدد مقدار ورودی را در حالت فعلی و عدد بعد از ممیز، خروجی را در همان حالت نشان می دهد. مثلاً کنار خط جهت داری که حالت 00 را به 01 وصل می کند 0/1 نوشته شده است، و بدان معنی است که وقتی مدار ترتیبی در حالت 00 است بازاء ورودی 1، خروجی 0 خواهد بود. پس از اعمال یک پالس ساعت، مدار به حالت بعدی 01 می رود. همین اعمال پالس ساعت ممکن است ورودی را عوض کند. اگر ورودی 0 شود خروجی 1 شده ولی اگر ورودی در 1 باقی بماند خروجی در 0 خواهد ماند. این اطلاعات از دیاگرام حالت و با دنبال کردن دو خط جهت داری که به دایره 01 وصل شده اند حاصل می گردد. خط جهت داری که دایره را به خودش وصل می کند نشان می دهد که تغییر حالتی رخ نداده است.

بین جدول حالت و دیاگرام حالت هیچ تفاوتی جز در نحوه نمایش وجود ندارد. بدست آوردن جدول حالت از دیاگرام منطقی آسان تر است و نمودار حالت مستقیماً از جدول حالت بدست می آید. دیاگرام حالت یک دید مصور را از انتقال حالات فراهم می نماید و بشکلی است که برای تفسیر عملکرد مدار مناسب تر می باشد. مثلاً دیاگرام حالت شکل ۲۶-۱ بوضوح نشان می دهد که با شروع حالت از 00 مادامی که ورودی 1 است خروجی 0 خواهد بود. اولین ورودی 0 پس از رشته ای از 1 ها، خروجی 1 تولید خواهد کرد و مدار را به حالت اولیه 00 برمی گرداند.



شکل ۲۶-۱ دیاگرام حالت یک مدار ترتیبی

مثال طراحی

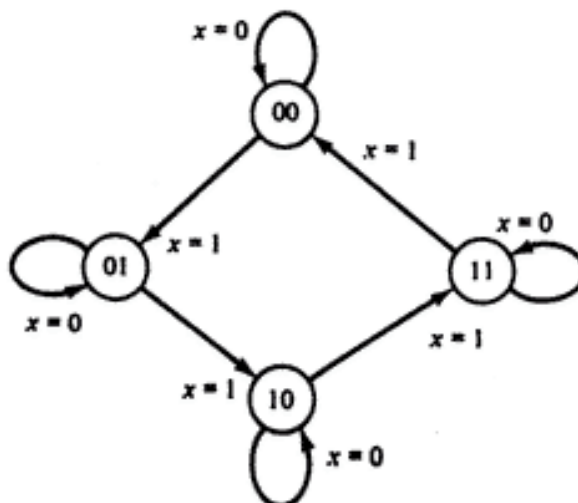
اکنون روش طراحی مدارهای ترتیبی بوسیله ارائه یک مثال خاص نشان داده می شود. در روش طراحی ابتدا مشخصات مدار را به یک دیاگرام حالت تبدیل می کنیم. سپس دیاگرام حالت به یک جدول حالت تبدیل می شود. بالاخره از جدول حالت برای تهیه دیاگرام مدار منطقی استفاده می شود.

ما می خواهیم یک مدار ترتیبی طراحی کنیم که هر وقت ورودی خارجی x برابر 1 باشد رشته حالات دودویی 00، 01، 10 و 11 را تکرار نماید. بازه $x = 0$ هم حالت مدار تغییری نکند. این نوع مدار یک شمارنده دودویی دو بیتی است زیرا رشته حالات با شمارش متوالی دو رقم دودویی یکسان می باشد. ورودی x یک متغیر کنترل است که مشخص می کند شمارش چه زمانی آغاز شود.

شمارنده دودویی به دو فلیپ فلاپ برای نمایش دو رقم نیاز دارد. دیاگرام حالت برای مدار ترتیبی در شکل ۱-۲۷ نشان داده شده است. دیاگرام بدین منظور رسم شده است تا نشان دهیم مادامی که $x = 1$ است مدار شمارش دودویی را ادامه می دهد. حالت بعد از 11، حالت 00 است که باعث تکرار شمارش می گردد. اگر $x = 0$ باشد، حالت مدار بدون تغییر باقی می ماند. این مدار ترتیبی خروجی خارجی ندارد و بنابراین فقط مقدار دودویی در جوار خطوط واصل در دیاگرام نوشته شده است. در این مدار حالت فلیپ فلاپ ها بعنوان خروجی های شمارنده تصور شده اند.

ما قبلاً سمبل x را به متغیر ورودی اختصاص دادیم. حال سمبل های A و B را به خروجی های دو فلیپ فلاپ تخصیص می دهیم. حالت بعدی A و B را، می توان بعنوان تابعی از حالت فعلی و ورودی x از دیاگرام حالت به جدول حالت منتقل ساخت. پنج ستون اول جدول ۵-۱ جدول حالت را بوجود می آورند. واردهای این جدول مستقیماً از دیاگرام حالت بدست می آیند.

جدول تحریک یک مدار ترتیبی نوع تعمیم یافته یک جدول حالت است. این تعمیم شامل لیستی



شکل ۱-۲۷ دیاگرام حالت برای شمارنده دودویی

از تحریک های ورودی فلیپ فلاپ می باشد که موجب انتقال به حالت مورد نظر می شوند. شرایط ورودی فلیپ فلاپها تابعی از نوع فلیپ فلاپ بکار رفته می باشد. ما اگر از فلیپ فلاپهای JK استفاده کنیم، نیاز به ستون های ورودی های J و K در هر فلیپ فلاپ داریم. ورودیهای فلیپ فلاپ A با J_A و K_A و ورودی های فلیپ فلاپ B با J_B و K_B نشان داده می شوند.

جدول تحریک برای فلیپ فلاپ JK که در جدول ۱-۳ نشان داده شد در اینجا برای بدست آوردن جدول تحریک مدار ترتیبی بکار گرفته می شود. مثلاً در اولین سطر جدول ۱-۵، انتقالی را از حالت فعلی 0 به حالت بعدی 0 در فلیپ فلاپ A داریم. در جدول ۱-۳ می بینیم که انتقال حالتی از $Q(t)=0$ به $Q(t+1)=0$ در یک فلیپ فلاپ JK مستلزم ورودی $J=0$ و $K=X$ می باشد. بنابراین 0 و X بترتیب در سطر اول و در زیر J_A و K_A نوشته می شوند. چون اولین سطر انتقال فلیپ فلاپ B را هم از حالت فعلی 0 به حالت بعدی 0 نشان می دهد، لذا مجدداً 0 و X در سطر اول و در زیر J_B و K_B نوشته شده اند. دومین سطر جدول ۱-۵ انتقال فلیپ فلاپ B را از حالت فعلی 6 به حالت بعدی 1 نشان می دهد. با توجه به جدول ۱-۳ می بینیم که انتقال از $Q(t)=0$ به $Q(t+1)=1$ مستلزم ورودی $J=1$ و $K=X$ است. بنابراین 1 و 0 در سطر دوم و بترتیب در زیر J_B و K_B نوشته می شوند. این روند برای هر سطر جدول و هر فلیپ فلاپ ادامه می یابد و شرایط ورودی در جدول ۱-۳ در سطر مناسب و برای هر فلیپ فلاپ مورد نظر نوشته می شود.

اکنون اطلاعات موجود در یک جدول تحریک مانند جدول ۱-۵ را ملاحظه می کنیم. می دانیم که یک مدار ترتیبی از تعدادی فلیپ فلاپ و یک مدار ترکیبی تشکیل شده است. با توجه به شکل ۱-۲۴ می بینیم که خروجی های مدار ترتیبی باید به چهار ورودی فلیپ فلاپها یعنی J_A ، K_A ، J_B و K_B متصل شوند. ورودی های مدار ترکیبی عبارت از ورودی خارجی X و مقادیر حالات فعلی فلیپ فلاپ های A و B می باشند. بعلاوه توابع بولی که یک مدار ترکیبی را مشخص می کنند از یک جدول درستی که رابطه ورودی - خروجی مدار را نشان می دهد حاصل می گردند. واردهایی که ورودی های مدار ترکیبی را

جدول ۱-۵ جدول تحریک برای شمارنده دودویی

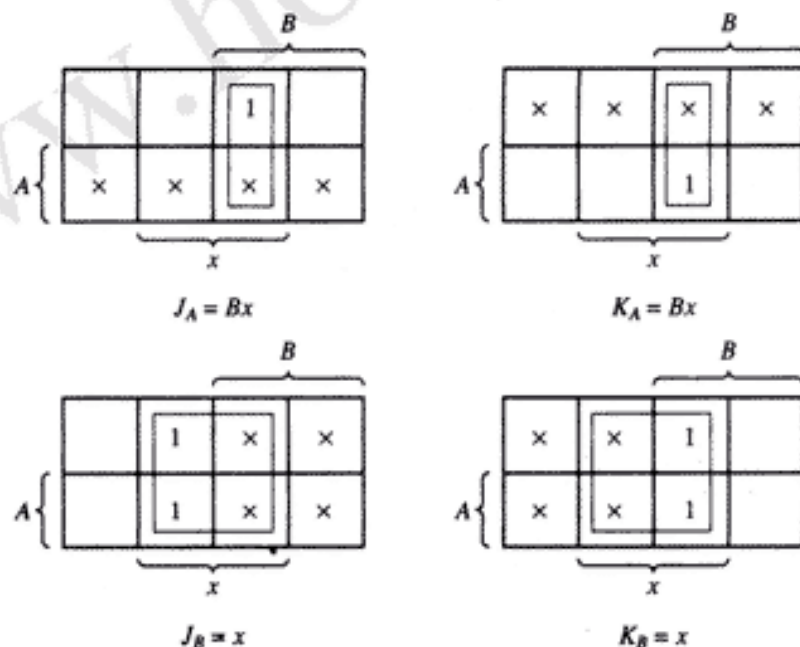
حالت فعلی			ورودی		حالت بعدی		ورودی های فلیپ فلاپ			
A	B	x	A	B	J_A	K_A	J_B	K_B		
0	0	0	0	0	0	x	0	x		
0	0	1	0	1	0	x	1	x		
0	1	0	0	1	0	x	x	0		
0	1	1	1	0	1	x	x	1		
1	0	0	1	0	x	0	0	x		
1	0	1	1	1	x	0	1	x		
1	1	0	1	1	x	0	x	0		
1	1	1	0	0	x	1	x	1		

جدول تحریک، یک دیاگرام حالت رابه جدول درستی مورد نیاز در طراحی مدار ترکیبی که بخشی از مدار ترتیبی است مبدل می سازد.

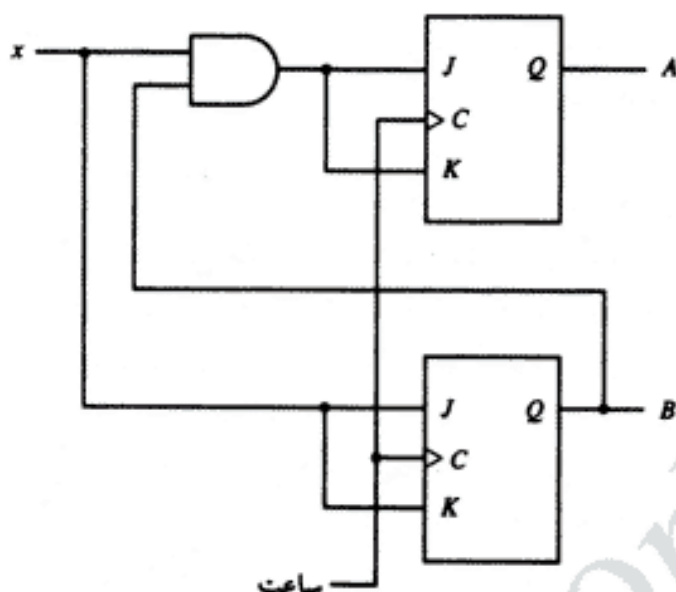
اکنون توابع بولی ساده شده در مدار ترکیبی را می توان بدست آورد. ورودیها متغیرهای A ، B و x می باشند. خروجیها نیز متغیرهای J_A ، K_A ، J_B و K_B هستند. اطلاعات حاصل از جدول تحریک به نقشه های شکل ۱-۲۸ انتقال یافته و چهار معادله ورودی ساده شده فلیپ فلاپها از آن بدست آمده است

$$\begin{aligned} J_A &= Bx & K_A &= Bx \\ J_B &= x & K_B &= x \end{aligned}$$

دیاگرام منطقی که در شکل ۱-۲۹ رسم شده است از دو فلیپ فلاپ JK و یک گیت AND تشکیل یافته است. توجه کنید که ورودی های J و K حالت بعدی شمارنده را پس از وقوع پالس ساعت معین می کنند. اگر هر دو ورودی J و K برابر 0 باشند پالس ساعت تأثیری نخواهد داشت. یعنی حالت فلیپ فلاپها تغییری نمی کند. بنابراین وقتی $x = 0$ باشد، تمام ورودی های فلیپ فلاپها برابر 0 و حالت فلیپ فلاپها بلا تغییر باقی می ماند حتی اگر پالسهای ساعت مرتباً به مدار اعمال شوند.



شکل ۱-۲۸ نقشه های مدار ترکیبی شمارنده



شکل ۲۹-۱ دیاگرام منطقی یک شمارنده دو بیتی

روش طراحی

طراحی مدارهای ترتیبی رتوس مطالبی را که در مثال طراحی دیده شد دنبال می‌کند. ابتدا رفتار مدار بصورت دیاگرام حالت فرموله می‌شود. تعداد فلیپ فلاپ‌های لازم برای مدار با توجه به تعداد بیت‌های داخل دایره‌ها در دیاگرام حالت معین می‌گردد. تعداد ورودی‌ها برای مدار در امتداد خطوط واصل بین دایره‌ها مشخص می‌شود. ما سپس حروفی را برای علامت گذاری تمام فلیپ فلاپها، متغیرهای ورودی و خروجی بکار گرفته و برای بدست آوردن جدول حالت به مرحله بعد می‌رویم.

بازاء m فلیپ فلاپ و n ورودی، جدول حالت شامل m ستون برای حالت فعلی، n ستون برای ورودی‌ها و m ستون برای حالت بعدی خواهد بود. تعداد سطرهای جدول تا 2^{m+n} می‌باشد که هر سطر مربوط به یک ترکیب دودویی از حالت فعلی و ورودی‌هاست. در هر سطر ما حالت بعدی را آنچنانکه در دیاگرام حالت مشخص شده لیست می‌کنیم. سپس نوع فلیپ فلاپ بکار رفته در مدار انتخاب می‌شود. جدول حالت به جدول تحریک با اضافه کردن ستون‌های هر ورودی در هر فلیپ فلاپ تعمیم می‌یابد. جدول تحریک برای فلیپ فلاپ بکار رفته می‌تواند از جدول ۳-۱ بدست آید. از اطلاعات موجود در این جدول و با بررسی انتقال حالت فعلی به حالت بعدی در جدول حالت، اطلاعات مربوط به شرایط ورودی فلیپ فلاپها را در جدول تحریک بدست می‌آوریم.

جدول درستی بخش ترکیبی مدار ترتیبی در جدول تحریک موجود است. ستون‌های حالت فعلی و ورودی‌ها، ورودی‌های جدول درستی را تشکیل می‌دهند. وضعیت ورودی فلیپ فلاپها خروجی‌های جدول درستی را تشکیل می‌دهند. با استفاده از مفهوم ساده سازی بکمک نقشه، ما مجموعه‌ای از

معادلات ورودی فلیپ فلاپها را برای مدار ترکیبی بدست می آوریم. هر معادله ورودی فلیپ فلاپ یک نمودار منطقی را مشخص می کند که خروجی آن باید به یکی از ورودی های فلیپ فلاپها وصل شود. مدار ترکیبی حاصل، همراه با فلیپ فلاپها، مدار ترتیبی را تشکیل می دهد. اغلب، خروجی فلیپ فلاپها بعنوان بخشی از خروجی های مدار ترتیبی تصور می شوند. با این وجود مدار ترکیبی ممکن است خروجی خارجی هم داشته باشد. در چنین حالتی توابع بولی خروجی های خارجی از جدول حالت با کمک تکنیک های طراحی بدست می آیند. مجموعه ای از معادلات ورودی فلیپ فلاپها یک مدار ترتیبی را بفرم جبری مشخص می کنند. روش یافتن دیاگرام منطقی از مجموعه معادلات ورودی فلیپ فلاپها یک روند مستقیم و سر راست است. ابتدا تمام ورودی ها و خروجی ها را نام گذاری می کنیم. سپس مدار ترکیبی را با توجه به معادلات بولی حاصل از معادلات ورودی فلیپ فلاپها رسم می نماییم. و بالاخره خروجی های فلیپ فلاپها را به ورودی های مدار ترکیبی و خروجی های مدار ترکیبی را به ورودی های فلیپ فلاپها وصل می نماییم.

مسائل

۱-۱ با استفاده از جدول درستی صحت تئوری دومورگان را برای سه متغیر تحقیق کنید:

$$(ABC)' = A' + B' + C'$$

۱-۲ جدول درستی یک تابع OR انحصاری (تابع فرد) سه متغیر را تشکیل دهید:

$$x = A \oplus B \oplus C$$

۱-۳ عبارات زیر را با استفاده جبر بول ساده کنید:

$$(الف) A + AB$$

$$(ب) AB + AB'$$

$$(ج) A'BC + AC$$

$$(د) A'B + ABC' + ABC$$

۱-۴ عبارات زیر را با استفاده از جدول بول ساده کنید:

$$(الف) AB + A(CD + CD')$$

$$(ب) (BC' + A'D)(AB' + CD')$$

۱-۵ با استفاده از تئوری دومورگان نشان دهید که:

$$(الف) (A + B)' (A' + B')' = 0$$

$$(ب) A + A'B + A'B' = 1$$

۱-۶ با فرض عبارت بولی $F = x'y + xyz'$:

(الف) یک عبارت جبری برای متمم F' بدست آورید.

$$(ب) نشان دهید که $FF' = 0$$$

$$(ج) نشان دهید که $F + F' = 1$$$

۱-۷ با فرض تابع بول

$$F = xy'z + x'y'z + xyz$$

- (الف) جدول درستی تابع را بدست آورید.
 (ب) دیاگرام منطقی را با استفاده از عبارت بولی اصلی رسم کنید.
 (ج) عبارت جبری را با بکارگیری جبر بول ساده کنید.
 (د) جدول درستی را با استفاده از عبارت ساده شده تابع تشکیل داده و نشان دهید که همان جدول بخش (الف) است.
 (ه) دیاگرام منطقی را با استفاده از عبارت ساده شده بدست آورده و تعداد کل گیت ها را با دیاگرام بخش (ب) مقایسه کنید.

1-8 توابع بولی زیر را با استفاده از نقشه سه متغیره ساده کنید.

$$F(x, y, z) = \sum(1, 2, 3, 6, 7) \quad (\text{ب}) \quad F(x, y, z) = \sum(0, 1, 5, 7) \quad (\text{الف})$$

$$F(x, y, z) = \sum(0, 2, 3, 4, 6) \quad (\text{د}) \quad F(x, y, z) = \sum(3, 5, 6, 7) \quad (\text{ج})$$

1-9 توابع بولی زیر را با استفاده از نقشه چهار متغیره ساده کنید:

$$F(A, B, C, D) = \sum(4, 6, 7, 15) \quad (\text{الف})$$

$$F(A, B, C, D) = \sum(3, 7, 11, 13, 14, 15) \quad (\text{ب})$$

$$F(A, B, C, D) = \sum(0, 1, 2, 4, 5, 7, 11, 15) \quad (\text{ج})$$

$$F(A, B, C, D) = \sum(0, 2, 4, 5, 6, 7, 8, 10, 13, 15) \quad (\text{د})$$

1-10 عبارات زیر را (۱) بصورت جمع حاصلضربها و (۲) بصورت ضرب حاصل جمع ها ساده کنید:

$$x'z' + y'z' + yz' + xy \quad (\text{الف})$$

$$AC' + B'D + A'CD + ABCD \quad (\text{ب})$$

1-11 توابع بولی زیر را بشکل جمع حاصلضربها با استفاده از نقشه چهار متغیره ساده کنید. دیاگرام منطقی را، الف) با گیت های AND-OR ؛ ب) با گیت های NAND رسم کنید.

$$F(A, B, C, D) = \sum(0, 2, 8, 9, 10, 11, 14, 15)$$

1-12 تابع بولی زیر را بشکل ضرب حاصل جمع ها با استفاده از نقشه چهار متغیره ساده کنید. دیاگرام منطقی را، الف) با گیت های OR-AND ؛ ب) با گیت های NOR رسم کنید.

$$F(w, x, y, z) = \sum(2, 3, 4, 5, 6, 7, 11, 14, 15)$$

1-13 تابع بولی F را همراه با حالات بی اهمیت d در فرم (۱) جمع حاصلضربها و (۲) ضرب حاصلجمع ها ساده کنید.

$$F(w, x, y, z) = \sum(0, 1, 2, 3, 7, 8, 10)$$

$$F(w, x, y, z) = \sum(5, 6, 11, 15)$$

۱-۱۴ با استفاده از جدول ۱-۲ عبارت بولی S (جمع) خارج شده از یک تمام جمع کننده بشکل جمع حاصلضربها را بدست آورید. سپس با دستکاریهای جبری نشان دهید که S می تواند بصورت OR انحصاری سه متغیره بیان شود.

$$S = x \oplus y \oplus z$$

۱-۱۵ یک مدار ترتیبی بنام تابع اکثریت بدین صورت تعریف می شود که هرگاه مقدار متغیرهای ورودی 1 بیشتر از ورودی های 0 باشند خروجی 1 است. تابع اکثریت سه ورودی را طراحی کنید.

۱-۱۶ یک مدار ترکیبی با سه متغیر x، y و z و سه خروجی A، B و C طراحی کنید. اگر ورودی های دودویی 0، 1، 2 یا 3 باشد خروجی یکی بیشتر از ورودی است. وقتی ورودی ها 4، 5، 6 و 7 باشند خروجی دودویی یکی کمتر از ورودی باشند.

۱-۱۷ نشان دهید که یک فلیپ فلاپ JK را با قرار دادن یک معکوس کننده بین ورودی های J و K می توان به یک فلیپ فلاپ D تبدیل کرد.

۱-۱۸ با استفاده از اطلاعات موجود در جدول مشخصه فلیپ فلاپ JK در جدول شکل ۱-۲ (ب) جدول تحریک را برای فلیپ فلاپ JK بدست آورده و جواب خود را با جدول ۱-۳ مقایسه کنید.

۱-۱۹ یک مدار ترتیبی دارای دو فلیپ فلاپ A و B، دو ورودی x و y و یک خروجی z است. معادلات ورودی فلیپ فلاپ و نیز خروجی مدار بقرار زیرند:

$$D_A = x'y + xA$$

$$D_B = x'B + xA$$

$$z = B$$

(الف) دیاگرام منطقی را برای مدار رسم کنید

(ب) جدول حالت را برپا کنید.

۱-۲۰ یک پایین شمار دو بیتی را طراحی کنید، این یک مدار ترتیبی دارای دو فلیپ فلاپ و یک ورودی x می باشد. وقتی $x = 0$ است حالت فلیپ فلاپ تغییر نمی کند. وقتی $x = 1$ باشد، ترتیب حالات 11، 10، 01، 00 و 11 و تکرار آن خواهد بود.

۱-۲۱ یک مدار ترتیبی با دو فلیپ فلاپ JK و دو ورودی E و x طراحی کنید. اگر $E = 0$ باشد، مدار بدون توجه به مقدار x در همان حالت باقی می ماند. وقتی $E = 1$ و $x = 1$ است مدار به یک سری حالات از 00 به 01، به 10، به 11 و 00 و تکراری از آن وارد می شود. وقتی $E = 1$ و $x = 0$ باشد مدار به حالات انتقالی 00 به 11، به 10، به 01 و 00 و تکراری از آن وارد می شود.



قطعات دیجیتال

- ۲-۱ مدارهای مجتمع
- ۲-۲ دیگدرها
- ۲-۳ مولتی پلکسرها
- ۲-۴ ثباتها
- ۲-۵ شیفت رجیسترها
- ۲-۶ شمارندههای دودویی
- ۲-۷ واحد حافظه

۲-۱ مدارهای مجتمع^۱

مدارهای دیجیتال با مدارهای مجتمع ساخته می شوند. یک مدار مجتمع (یا آی سی)، یک کریستال کوچک نیمه هادی بنام تراشه^۲ است که قطعات الکترونیکی را برای گیت های دیجیتال در خود دارد. اتصالات داخل تراشه، مدار مورد نیاز را بوجود می آورد. تراشه در داخل یک محفظه پلاستیک و یا سرامیک جاسازی می شود و اتصالات آن با سیم های طلایی نازک به پایه های خارجی جوش داده می شود تا مدار مجتمع بوجود آید. تعداد پایه ها ممکن است از 14 پایه در بسته های کوچک تا 100 پایه یا بیشتر در بسته های بزرگتر تغییر کند. هر مدار مجتمع یا آی سی دارای یک مشخصه عددی است که روی سطح بسته بندی آن برای شناسایی چاپ می شود. هر سازنده یک کتابچه راهنما یا کاتالوگ با شرح دقیق و تمام اطلاعات لازم درباره آی سی های ساخت خود را چاپ می کند.

1- Integrated Circuit (IC) 2- Chip

با پیشرفت تکنولوژی مدارهای مجتمع، تعداد گیت‌هایی که می‌توانست در یک تراشه جای گیرد به میزان قابل ملاحظه‌ای افزایش یافت. تراشه‌هایی که دارای چند گیت داخلی بودند و آن دسته که چند صد گیت را دارا بودند در بسته‌هایی با ظرفیت یا مقیاس کوچک، متوسط و یا بزرگ جای داده شدند. مدارهای مجتمع با مقیاس کوچک^۱ (SSI) دارای چند گیت مستقل در یک بسته واحد هستند. ورودی‌ها و خروجی‌های گیت‌ها مستقیماً به پایه‌های بسته متصل‌اند. تعداد گیت‌ها معمولاً کمتر از ۱۰ و محدود به تعداد پایه‌ها در IC می‌باشد.

قطعات مجتمع با مقیاس متوسط^۲ (MSI) دارای تقریباً ۱۰ الی ۲۰۰ گیت در هر بسته می‌باشند. این وسیله‌ها معمولاً توابع دیجیتال ساده همچون دیکدرها، جمع‌کننده‌ها و ثبات‌ها را اجرا می‌نمایند. مدارها یا وسایل مجتمع با مقیاس بزرگ^۳ (LSI) بین ۲۰۰ تا چند هزار گیت در هر بسته دارند. این بسته‌ها سیستم‌های دیجیتالی همچون پردازنده‌ها، تراشه‌های حافظه و ماژول‌های قابل برنامه‌ریزی را شامل می‌شوند. قطعات مجتمع با مقیاس بسیار بزرگ^۴ (VLSI) حاوی هزاران گیت در یک بسته‌اند. مثالهایی از این گروه عبارتند از، آرایه‌های بزرگ حافظه و تراشه‌های پیچیده ریز کامپیوترها. VLSI‌ها بدلیل کوچکی و ارزانی انقلابی در تکنولوژی ساخت سیستم‌های کامپیوتری بوجود آورده و به طراحان امکان ساخت و ایجاد ساختارهایی را دادند که قبلاً اقتصادی نبودند.

مدارهای مجتمع دیجیتال نه تنها براساس عملکرد منطقی‌شان طبقه‌بندی می‌شوند بلکه از نظر تکنولوژی خاص مدارهایی که به آن تعلق دارند نیز دسته‌بندی می‌گردند. تکنولوژی بکار رفته در مدار را خانواده منطقی دیجیتال^۵ می‌خوانند. هر خانواده منطقی، مدار الکترونیکی پایه خاصی را داراست که مدارها و توابع دیجیتال پیچیده‌تر براساس آن تهیه می‌شوند. مدار پایه در هر تکنولوژی یک گیت NAND، NOR و یا معکوس کننده است. در نام‌گذاری تکنولوژی، از قطعات الکترونیک بکار رفته در ساخت مدار پایه معمولاً استفاده می‌شود. بسیاری از خانواده‌های مختلف منطقی بصورت مدارهای مجتمع در سطح تجاری عرضه شده‌اند. متداول‌ترین خانواده‌ها در زیر معرفی شده‌اند.

TTL - منطق ترانزیستور - ترانزیستور^۶

ECL - منطق کوپل امیتر^۷

MOS - منطق فلز - اکسید - نیمه هادی^۸

CMOS - منطق فلز - اکسید - نیمه هادی مکمل^۹

1- Small Scale Integration

2- Medium Scale Integration

3- Large Scale Integration

4- Very Large Scale Integration

5- Digital Logic Family

6- Transistor - Transistor Logic

7- Emitter Coupled Logic

8- Metal - Oxide Semiconductor

9- Complementary Metal Oxide Semiconductor

TTL یک خانواده متداول است که سالها مورد استفاده بوده و بعنوان استاندارد تلقی می شود. ECL در سیستم هایی که به سرعت عمل بالا نیاز دارند ترجیح داده می شود. MOS برای مدارهایی که نیاز به تراکم بالا دارند مناسب است، و CMOS در سیستم های کم مصرف بکار می رود.

خانواده منطقی ترانزیستور - ترانزیستور گونه تکامل یافته تکنولوژی قدیمی تری است که در آن از دیود و ترانزیستور برای ساخت گیت پایه NAND استفاده می شده است. این تکنولوژی منطق دیود - ترانزیستور¹ (DTL) خوانده می شده است. بعدها برای بهبود عملکرد مدار بجای دیود از ترانزیستور استفاده شد و نام خانواده جدید، منطق ترانزیستور - ترانزیستور گذاشته شد. به همین دلیل نام "ترانزیستور" در این عبارت دو بار تکرار شده است. علاوه بر نوع استاندارد TTL، انواع دیگری از این خانواده عبارتند از: TTL سرعت بالا،² TTL توان پائین³ (یا کم مصرف)، TTL شوکتی⁴، TTL شوکتی توان پایین⁵ و TTL شوکتی پیشرفته⁶. منبع تغذیه مدارهای TTL، 5 ولت است و دوسطح منطقی تقریباً 0 و 3/5 ولت می باشند.

خانواده کوپل امیتر سریع ترین مدارهای دیجیتال را بفرم مجتمع در اختیار می گذارد. ECL در مدارهایی مانند سوپر کامپیوترها و پردازنده های سیگنال که در آنها سرعت بالا ضرورت دارد، بکار می رود. ترانزیستورها در گیت های ECL در حالت غیر اشباع کار می کنند و رسیدن به تأخیرهای انتشاری در حد 1 تا 2 نانو ثانیه در آنها میسر است.

منطق فلز - اکسید - نیمه هادی (MOS) یک ترانزیستور تک قطبی است که به جریان یک نوع حامل الکتریکی وابسته است. این حامل ها ممکن است الکترون (در نوع کانال n) یا حفره (در نوع کانال p) باشند. این، برخلاف ترانزیستور دو قطبی بکار رفته در گیت های TTL و ECL است، که در حین عملکرد هر دو نوع حامل در آن وجود دارند. یک MOS کانال p را PMOS و یک MOS کانال n را NMOS می نامند. معمولاً در مدارهایی که فقط یک نوع ترانزیستور MOS وجود دارد از NMOS استفاده می شود. در تکنولوژی CMOS هر دو نوع ترانزیستور PMOS و NMOS که بشکل مکمل در تمام مدارها بسته شده اند بکار رفته است. بزرگترین مزیت CMOS نسبت به دو قطبی تراکم بالای مدارها در بسته بندی، ساده بودن تکنیک ساخت و عملکرد مقرون به صرفه آن بدلیل مصرف توان کم است.

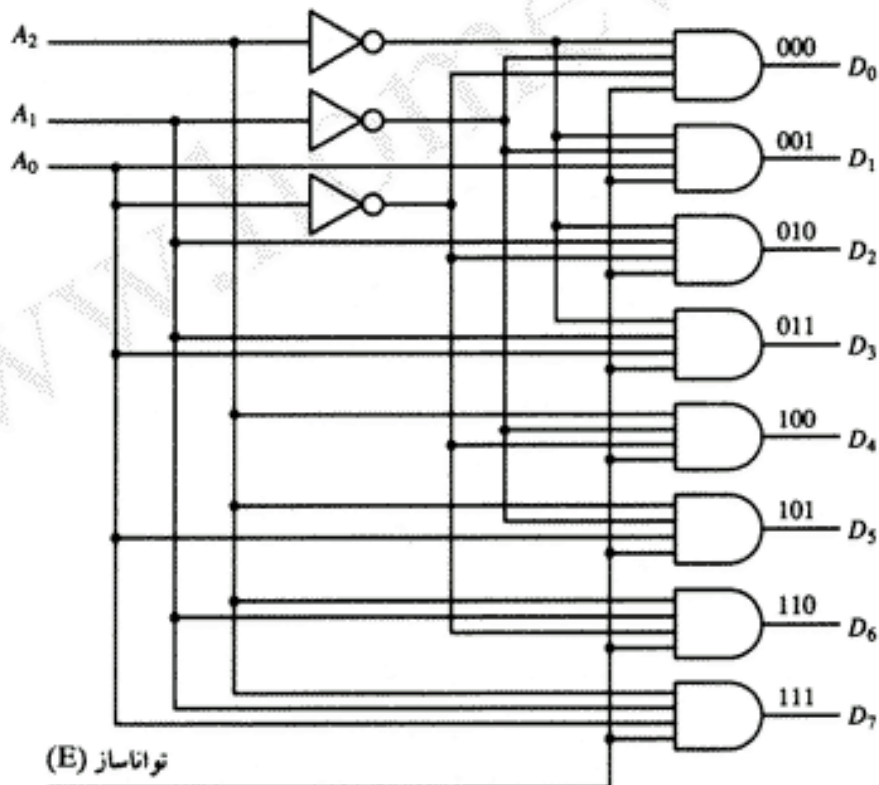
بعلت مزایای بی شمار، مدارهای مجتمع انحصاراً در تهیه انواع قطعات لازم در طراحی سیستم های کامپیوتر به کار می روند. برای درک سازمان و طراحی کامپیوترها، آشنایی با انواع قطعات و اجزاء بکار رفته در مدارهای مجتمع اهمیت دارد. به این دلیل، اجزاء اصلی به همراه خواص منطقی آنها تشریح شده است. این اجزاء مجموعه ای از واحدهای عملیاتی دیجیتال را فراهم می کنند که در طراحی کامپیوترهای دیجیتال بعنوان بلاک های ساختمانی اصلی (پایه) به کار می روند.

-
- | | | |
|-----------------------------|-----------------------------|------------------|
| 1- Diode - Transistor Logic | 2- High Speed TTL | 3- Low Power TTL |
| 4- Schottky TTL | 5- Low - Power Schottky TTL | |
| 6- Advanced Schottky TTL | | |

۲-۲ دیکدرها^۱

کمیت های اطلاعاتی گسسته در سیستم های کامپیوتر دیجیتال با کدهای دودویی نشان داده می شوند. یک کد دودویی ۸ بیتی قادر است 2^n عنصر مستقل اطلاعاتی کد شده را نمایش دهد. یک دیکدر مداری است ترکیبی که اطلاعات دودویی را از طریق n خط ورودی دریافت نموده و آنها را به حداکثر 2^n خط خروجی مستقل منحصر بفرد تبدیل می کند. اگر اطلاعات دیکد شده n بیتی ترکیبات بلااستفاده یا بی اهمیت داشته باشد در اینصورت دیکدر دارای خروجی های کمتر از 2^n خواهد بود. دیکدرهایی که در این بخش مورد بحث قرار گرفته اند دیکدرهای n به m خطی نام دارند که در آن $m \leq 2^n$ است. هدف از بکارگیری آنها تولید 2^n ترکیب دودویی از n متغیر ورودی است. دیکدر دارای n ورودی و m خروجی بوده و دیکدر $n \times m$ نیز نامیده می شود.

دیاگرام منطقی یک دیکدر ۳ به ۸ در شکل ۲-۱ نشان داده شده است. سه ورودی داده A_2 و A_1 و A_0 به هشت خروجی داده تبدیل شده اند که هر خروجی یکی از ترکیبات سه متغیر ورودی دودویی را نشان



شکل ۲-۱ دیکدر ۳ به ۸ خطی

می دهد. سه معکوس کننده، متمم ورودی ها را فراهم می کنند و هر یک از هشت گیت AND یکی از ترکیبات دودویی را تولید می نماید. کاربرد خاصی از این دیکدر تبدیل دودویی به هشتایی¹ است. متغیرهای ورودی یک عدد دودویی را نشان می دهند و خروجی ها، هشت رقم یک عدد سیستم هشتایی را نمایش می دهند. با این وجود یک دیکدر 3 به 8 خطی می تواند برای دیکد کردن هر کد سه بیتی بکار رفته و هشت خروجی را که هر بار یکی از ترکیبات کد دودویی است، فراهم آورد.

دیکدرهای تجاری دارای یک یا چند ورودی تواناسازی² (فعال سازی) برای کنترل کار مدار هستند. دیکدر شکل 1-2 دارای یک ورودی تواناساز، E می باشد. دیکدر هنگامی فعال شده است که E برابر 1 باشد و اگر E مساوی 0 بود دیکدر ناتوان (غیرفعال) شده است. عملکرد دیکدر با استفاده از یک جدول درستی که در جدول 1-2 دیده می شود آشکارتر خواهد شد. هر گاه ورودی تواناساز E برابر 0 باشد، تمام خروجی ها، بدون توجه به مقادیر سه ورودی داده، 0 خواهند شد. سه علامت x در جدول حالات بی اهمیت ورودی ها را نشان می دهند. اگر ورودی تواناساز، 1 شود، دیکدر در وضع عادی خودکار خواهد کرد. برای هر ترکیب ممکن ورودی، هفت خروجی 0 و فقط یک خروجی 1 وجود دارد که بیانگر عدد هشتایی معادل عدد دودویی در خطوط ورودی داده است.

دیکدر با گیت NAND

بعضی از دیکدرها بجای گیت AND با گیت NAND ساخته می شوند. چون گیت NAND عمل گیت AND را با خروجی معکوس شده انجام می دهد، تولید خروجی های معکوس شده با دیکدر مقرون به صرفه تر است. یک دیکدر 2 به 4 همراه با ورودی تواناساز که از گیت های NAND ساخته شده است در شکل 2-2 نشان داده شده است. مدار دارای خروجی های متمم بوده و با ورودی تواناساز متمم شده کار می کند. دیکدر هنگامی تواناست که E برابر 0 شود. همانطور که از جدول درستی

جدول 1-2 جدول درستی برای دیکدر 3 به 8 خطی

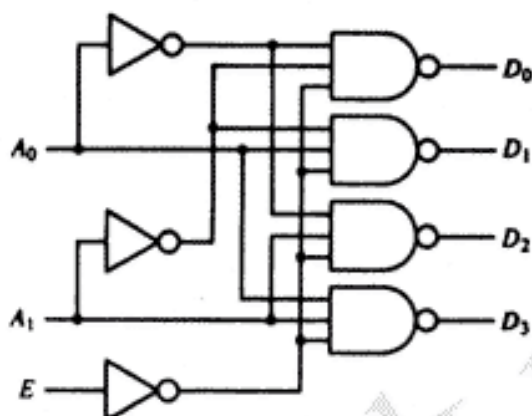
تواناساز E	ورودی ها			خروجی ها								
	A ₂	A ₁	A ₀	D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀	
0	x	x	x	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0	0
1	0	1	0	0	0	0	0	0	1	0	0	0
1	0	1	1	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0

1- Octal

2- Enable

E	A_1	A_0	D_0	D_1	D_2	D_3
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0
1	x	x	1	1	1	1

(ب) جدول درستی



(الف) دیاگرام منطقی

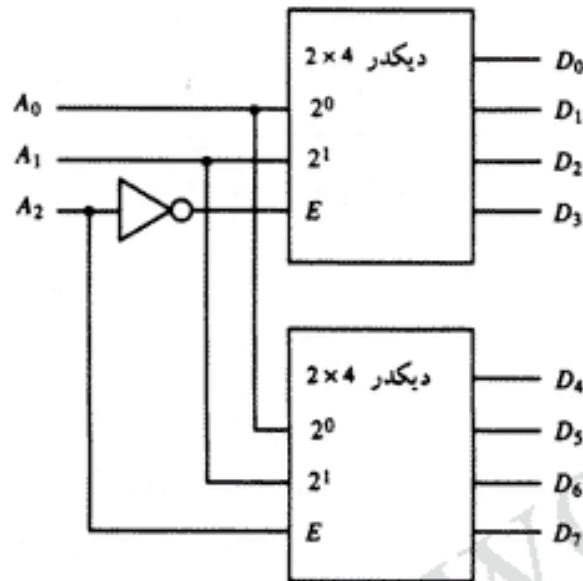
شکل ۲-۲ دیکدر 2 به 4 خطی با گیت های NAND

پیداست، تنها در هر لحظه از زمان فقط یک خروجی برابر 0، و بقیه سه خروجی دیگر برابر 1 هستند. خروجی که نماینده عدد دودویی واقع در ورودی های A_1 و A_0 است دارای مقدار 0 است. هرگاه E برابر 1 باشد، مدار صرف نظر از مقدار دودویی ورودی های دیگر غیرفعال خواهد بود. وقتی که مدار غیرفعال است، هیچ یک از خروجی ها انتخاب نمی شود و تمام خروجی ها برابر 1 خواهند بود. بطور کلی، یک دیکدر ممکن است با خروجی های متمم شده یا نشده کار کند. ورودی تواناساز هم ممکن است با یک سطح سیگنال 0 یا 1 فعال شود. برخی دیکدرها دارای دو یا چند ورودی تواناساز هستند و برای فعال شدن مدار شرایط منطقی معینی باید در آنها برقرار باشد.

گسترش دیکدر

مواردی پیش می آید که دیکدری با اندازه مشخصی مورد نیاز است ولی تنها اندازه کوچکتر آن موجود است در چنین مواقعی می توان دو و یا چند دیکدر را با هم ترکیب نمود و با استفاده از ورودی های فعال ساز (تواناساز) دیکدر بزرگتری بوجود آورد. بنابراین اگر یک دیکدر 6 به 64 مورد نیاز باشد می توان آن را با چند دیکدر 4 به 16 تشکیل داد.

شکل ۲-۳ نشان می دهد که چگونه از اتصال دیکدرهای دارای ورودی های تواناساز می توان دیکدرهای بزرگتری ساخت. در این شکل دو دیکدر 2 به 4 با هم ترکیب شده اند تا یک دیکدر 3 به 8 تشکیل شود. دو بیت کم ارزشتر ورودی ها به هر دو دیکدر متصل شده اند. بیت پرارزشتر به ورودی تواناساز یکی از دیکدرها و از طریق یک معکوس کننده به ورودی تواناساز دیکدر دیگر وصل است. فرض بر این است که دیکدر وقتی که $E=1$ است فعال شود. اگر $E=0$ باشد دیکدر غیرفعال و تمام خروجی های آن در سطح 0 است. هرگاه $A_2=0$ باشد، دیکدر بالایی فعال و دیکدر پایینی غیرفعال است. خروجی های



شکل ۲-۳ یک دیکدر ۳×۸ (یا ۳ به ۸) ساخته شده با دو دیکدر ۲×۴

دیکدر تحتانی همگی غیرفعال و در وضعیت ۰ می باشند. خروجی های دیکدر فوقانی خروجی های D_0 الی D_3 را، بسته به مقادیر A_0 و A_1 تولید می کنند (در این حالت $A_2=0$ است). اگر $A_2=1$ باشد، دیکدر پائینی فعال و دیکدر بالایی غیرفعال است. دیکدر پائینی در این وضعیت معادل دودویی D_4 تا D_7 را تولید می کند زیرا این عددهای دودویی رقم ۱ را در مکان A_2 دارا هستند.

مثال فوق فوائد ورودی فعال ساز (تواناساز) را در دیکدرها یا هر مدار منطقی ترکیبی نشان داد. ورودی های تواناساز ابزار مناسبی برای اتصال دو یا چند مدار به منظور گسترش ورودی ها و خروجی ها می باشند.

انکدرها^۱

انکدر (کدگذار) یک مدار دیجیتال است که عکس عمل دیکدر را انجام می دهد. یک انکدر دارای 2^n (یا کمتر) خط ورودی و n خط خروجی است. خطوط خروجی که دودویی متناظر مقدار ورودی را تولید می کنند. مثالی از یک انکدر، انکدر هشت هشتی به دودویی است، که جدول درستی آن در جدول ۲-۲ داده شده است. این مدار دارای هشت ورودی، که هر یک به یک رقم هشت هشتی (هشتایی) متعلق بوده و سه خروجی که عدد دودویی متناظر را تولید می کنند، است. فرض براین است که فقط یک ورودی دارای مقدار ۱ در هر لحظه از زمان است؛ در غیر این صورت مدار مفهومی ندارد.

انکدر را می توان با گیت های OR، که ورودی هایشان مستقیماً از جدول درستی تعیین می شوند،

1- Encoder

پیاده سازی کرد. خروجی A_0 بشرطی 1 است که رقم ورودی هشت هشتی 1 یا 3 یا 5 یا 7 باشد. شرایط مشابهی برای دو خروجی دیگر برقرار است. این شرایط را می توان بوسیله توابع بولی زیر بیان کرد.

$$A_0 = D_1 + D_3 + D_5 + D_7$$

$$A_1 = D_2 + D_3 + D_6 + D_7$$

$$A_2 = D_4 + D_5 + D_6 + D_7$$

انکدر فوق با سه گیت OR قابل پیاده سازی است.

۲-۳ مولتی پلکسرها^۱

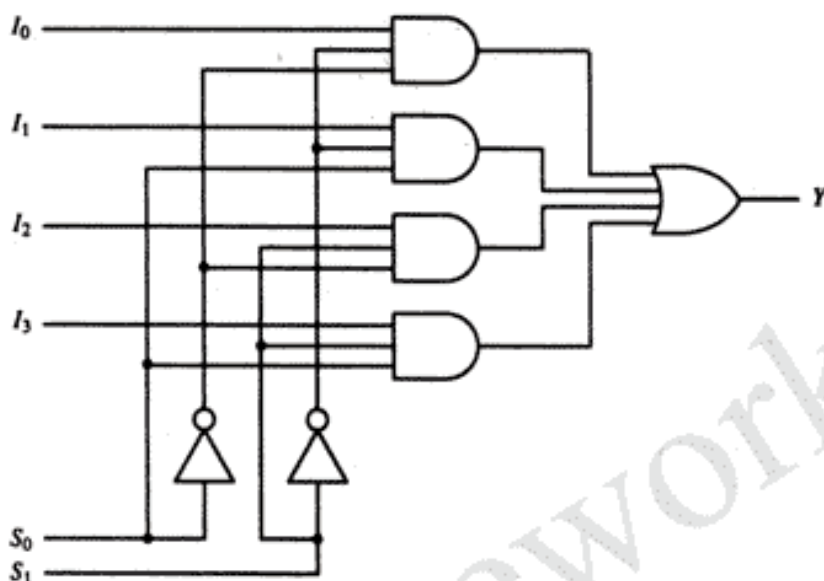
مولتی پلکسر یک مدار ترکیبی است که اطلاعات دودویی را از یکی از 2^n ورودی دریافت و آن را به یک مسیر خروجی هدایت می نماید. مجموعه ای از خطوط ورودی انتخاب^۲ خط داده خاصی را برای خروجی انتخاب می کنند. یک مولتی پلکسر 2^n به 1 دارای 2^n خط ورودی داده و n خط ورودی انتخاب می باشد. ترکیب این ورودی ها تعیین کننده خط داده ورودی انتخاب شده برای خروجی است. یک مولتی پلکسر 4 به 1 در شکل ۲-۴ نشان داده شده است. هر یک از چهار ورودی داده I_0 تا I_3 به ورودی یک گیت AND اعمال شده است. دو خط انتخاب S_0 و S_1 برای انتخاب یک گیت AND بخصوص دیکد شده اند. خروجی گیت های AND به ورودی های یک گیت OR اعمال شده تا تنها خروجی مدار را ایجاد نماید. برای درک طرز کار مدار، حالت $S_1S_0=10$ را در نظر بگیرید. دو ورودی گیت AND مربوط به ورودی I_2 در وضعیت 1 هستند. سومین ورودی این گیت به I_2 متصل است. سه

جدول ۲-۲ جدول درستی برای انکدر هشت هشتی به دودویی

ورودی ها								خروجی ها		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	A_2	A_1	A_0
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1

1- Multiplexer

2- selection



شکل ۲-۴ مولتی پلکسر 4 به 1

گیت AND دیگر، هر یک حداقل یک ورودی 0 دارند که در نتیجه آن خروجی شان 0 می شود. بنابراین خروجی گیت OR برابر مقدار I_2 است و باین ترتیب مسیری از ورودی های منتخب به خروجی ایجاد شده است.

مولتی پلکسر 4 به 1 شکل ۲-۴ دارای شش ورودی و یک خروجی است. جدول درستی که عملکرد مدار را نشان می دهد نیاز به 64 سطر خواهد داشت زیرا شش متغیر ورودی 2^6 ترکیب دودویی دارد. این جدول بیش از حد طولانی است و در اینجا نشان داده نخواهد شد.

روش مناسب تری برای بیان عملکرد مولتی پلکسر استفاده از جدول تابع است. جدول تابع برای مولتی پلکسر در جدول ۲-۳ نشان داده شده است. این جدول رابطه بین چهار ورودی داده و تنها خروجی مدار را بصورت تابعی از ورودی های انتخاب S_0 و S_1 نشان می دهد. هرگاه ورودی های انتخاب برابر 00 باشند، خروجی Y برابر I_0 است. اگر ورودی های انتخاب 01 شوند، ورودی I_1 به خروجی Y متصل است، و برای دو ورودی دیگر نیز استدلال مشابهی وجود دارد. مولتی پلکسر را انتخاب کننده داده 1 هم می نامند، چون این مدار یکی از چند ورودی داده را برمیگزیند و اطلاعات دودویی را به خروجی هدایت می نماید.

گیت های AND و معکوس کننده ها در مولتی پلکسر مشابه یک مدار دیگر می باشند و خطوط ورودی انتخاب را دیکد می کنند. بطور کلی، یک مولتی پلکسر 2^n به 1 خطی، از یک دیکدر n به 2^n

1- Data Selector (or data switch)

جدول ۲-۳ جدول تابع برای مولتی پلکسر 4 به 1 خطی

انتخاب		ورودی
S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

با اضافه کردن 2^n خط ورودی به آن، یعنی یک خط از هر ورودی داده، ساخته می شود. اندازه 1 یک مولتی پلکسر بوسیله تعداد 2^n ورودی داده و تنها خروجی اش مشخص می شود. این اندازه بطور ضمنی بیانگر این واقعیت است که تعداد خطوط انتخاب نیز n است. مولتی پلکسر را غالباً با نام اختصاری MUX می خوانند.

همانند دیگرها، مولتی پلکسرها هم ممکن است دارای ورودی تواناساز باشند تا عملکرد آن را کنترل نمایند. هر وقت ورودی تواناساز در حالت غیرفعال باشد، خروجی ها از کار می افتند، و اگر در حالت فعال قرار گیرد، مدار به صورت یک مولتی پلکسر بطور عادی عمل می کند. ورودی تواناساز برای اتصال دو یا چند مولتی پلکسر و تشکیل یک مولتی پلکسر با تعداد ورودی های بیشتر مفید است.

در برخی از موارد دو یا چند مولتی پلکسر در یک بسته مدار مجتمع جای داده می شود. ورودی های انتخاب و تواناساز در ساختار چند واحدی، برای همه مولتی پلکسرها مشترک اند. بلاک دیاگرام یک مولتی پلکسر چهارتایی 2 به 1، بعنوان مثال در شکل 5-2 نشان داده شده است. مدار دارای چهار مولتی پلکسر است که هر یک قادر به انتخاب یکی از دو خط ورودی می باشد. می توان خروجی Y_0 را چنان تعیین کرد که از ورودی A_0 یا B_0 گرفته شود. بطور مشابه Y_1 می تواند مقدار A_1 یا B_1 را داشته باشد و الی آخر. خط ورودی انتخاب S یکی از خطوط را در هر یک از چهار مولتی پلکسر انتخاب می کند. ورودی تواناساز E برای عملکرد عادی فعال می شود. اگرچه مدار دارای چهار مولتی پلکسر است، ما می توانیم این تصور را داشته باشیم که مدار یکی از دو خط داده چهار بیتی را انتخاب می نماید. همان طور که در جدول تابع نشان داده شد، هر گاه $E=1$ باشد، مدار فعال می شود. آنگاه اگر $S=0$ باشد چهار ورودی A مسیری به خروجی خواهند داشت. از طرف دیگر، اگر $S=1$ باشد، چهار ورودی B به خروجی اعمال می شوند. هر گاه $E=0$ باشد، همه خروجی ها مستقل از مقدار S ، تماماً 0 خواهند بود.

۲-۴ ثباتها^۲

ثبات مجموعه ای از فلیپ فلاپ هاست و هر فلیپ فلاپ قادر است یک بیت از اطلاعات را در خود

1- Size

2- Register



شکل ۵-۲ مولتی‌پلکسر 2 به 1 چهارتایی

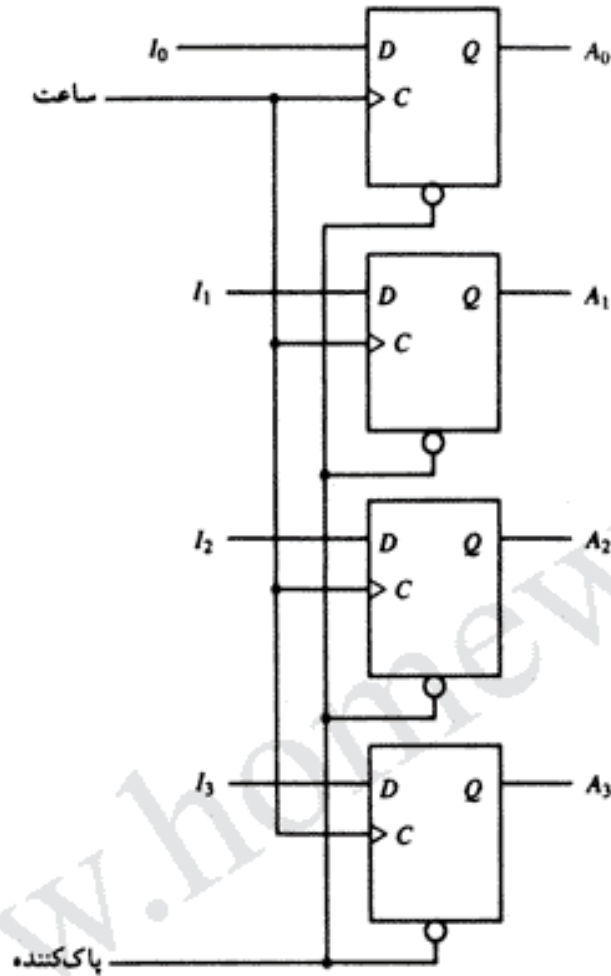
ذخیره کند. یک ثبات n بیتی n فلیپ فلاپ دارد و می‌تواند هر اطلاعات دودویی n بیتی را در خود ذخیره کند. یک ثبات، علاوه بر فلیپ فلاپ‌ها، ممکن است گیت‌های ترکیبی، که کارهای پردازش داده معینی را انجام می‌دهند، را هم در خود داشته باشد. یک ثبات در تعریف جامع‌تر خود متشکل از گروهی از فلیپ فلاپ‌ها و گیت‌هاست که بر تغییر حالت (خروجی) فلیپ فلاپ‌ها تأثیر دارند. فلیپ فلاپ‌ها اطلاعات دودویی را ذخیره می‌کنند و گیت‌ها زمان و چگونگی انتقال اطلاعات به ثبات را کنترل می‌نمایند.

انواع مختلفی از ثبات‌های تجاری وجود دارند. ساده‌ترین ثبات حاوی فلیپ فلاپ‌ها و فاقد هر نوع گیت خارجی است. شکل ۶-۲ چنین ثباتی را که از چهار فلیپ فلاپ D ساخته شده نشان می‌دهد. ورودی پالس ساعت مشترک در لبه بالارونده خود همه فلیپ فلاپ‌ها را فعال^۱ می‌سازد و داده دودویی موجود در چهار ورودی بداخل ثبات چهاربیتی منتقل می‌شود. چهار خروجی را می‌توان در هر زمانی نمونه برداری کرد و اطلاعات ذخیره شده در ثبات را بدست آورد. ورودی پاک کردن^۲ به پایه خاصی در هر فلیپ فلاپ متصل است. هرگاه این ورودی 0 شود تمام فلیپ فلاپ‌ها بطور غیرهمزمان (آسترون) و قبل از عملکرد پالس ساعت پاک^۳ می‌شوند. ورودی پاک کردن بهنگام عملکرد عادی فلیپ فلاپ‌ها باید در منطق 1 نگهداری شوند. توجه کنید که پالس ساعت، ورودی D را توانا می‌سازد ولی ورودی پاک کردن مستقل از پالس ساعت است.

1- Trigger

2- Clear

3- Reset



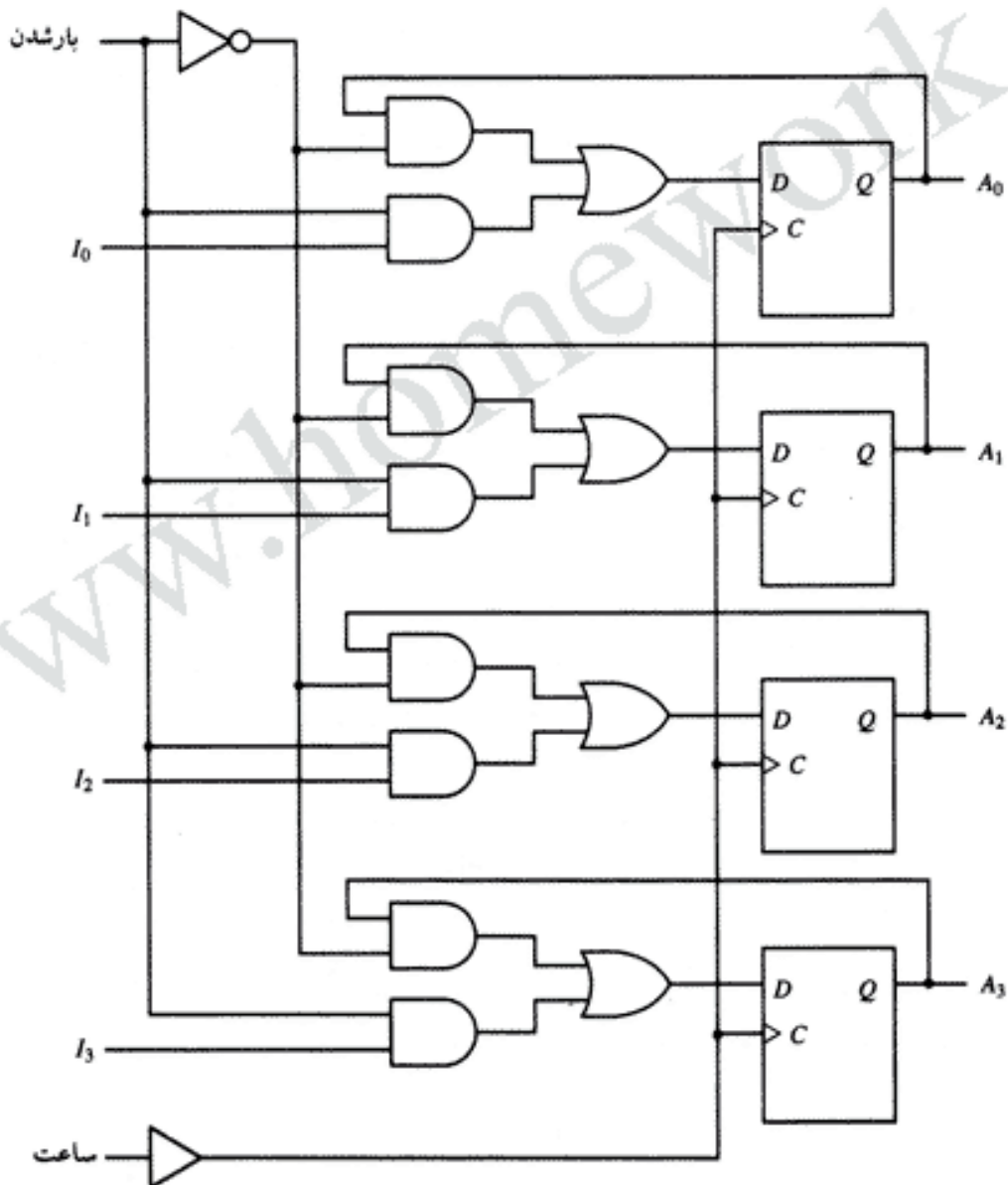
شکل ۶-۲ ثبات 4 بیت

انتقال اطلاعات جدید بداخل ثبات "بارکردن" ثبات نامیده می شود. اگر تمام بیت های ثبات بطور همزمان با یک انتقال پالس ساعت بار شوند، گوئیم بارشدن بصورت موازی صورت گرفته است. یک گذار یا انتقال پالس ساعت اعمال شده به ورودی های C در ثبات شکل ۶-۲ تمام چهار ورودی I_0 تا I_3 را به صورت موازی بار خواهد کرد. در این آرایش، اگر بخواهیم محتوای ثبات بدون تغییر باقی بماند باید پالس ساعت به مدار اعمال نگردد.

ثبات با بار شدن موازی

بسیاری از سیستم های دیجیتال دارای یک مدار ساعت اصلی می باشند که رشته ای از پالس ساعت را

فراهم می‌نماید. این پالس‌های ساعت به تمام فلیپ فلاپها و ثبات‌ها در سیستم اعمال می‌شوند. ساعت اصلی مانند پمپی که ضربان ثابتی را به تمام بخش‌ها ارسال کند عمل می‌نماید. سیگنال کنترل جداگانه را باید بکار برد تا در مورد تأثیر پالس ساعت خاص بر روی ثبات خاص تصمیم‌گیری نماید. یک ثبات چهار بیتی با ورودی کنترل بارکردن که از طریق گیت‌ها به ورودی‌های D می‌رسند در شکل ۷-۲ نشان داده شده‌اند. ورودی‌های C پالس‌های ساعت را در همه زمانها دریافت می‌کنند. گیت بافر در ورودی ساعت توان دریافتی لازم را از مولد ساعت کاهش می‌دهد. هرگاه پالس ساعت فقط به ورودی یک گیت در عوض چندگیت وصل شود توان کمتری لازم است ولی چنانچه این گیت بافر مورد



شکل ۷-۲ یک ثبات 4 بیت با بارشدن موازی

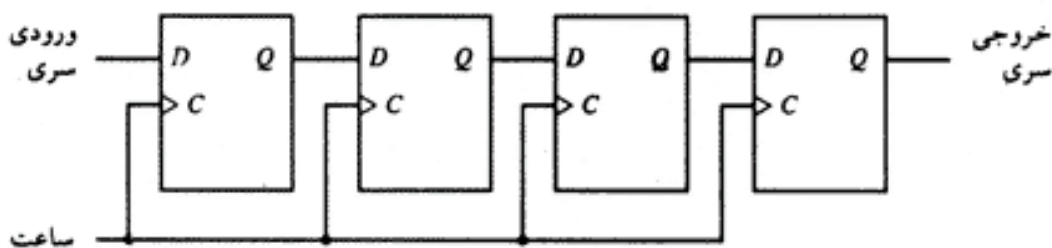
استفاده قرار نگرفته و پالس ساعت به هر چهار ورودی مستقیماً وصل شود توان بیشتری از مولد پالس اخذ خواهد شد.

ورودی بارکردن عملی را که با هر پالس ساعت در ثبات انجام می شود تعیین می کند. هرگاه ورودی بارکردن برابر 1 باشد، داده های چهار ورودی با لبه مثبت پالس ساعت (گذار مثبت) به داخل ثبات انتقال می یابند. اگر ورودی باردهی 0 باشد، ورودی های داده ممانعت شده و ورودی های فلیپ فلاپ D به خروجی های آن وصل می شوند. اتصال برگشت¹ (پسخور) از خروجی به ورودی لازم است زیرا فلیپ فلاپ D حالت بلا تغییر ندارد. با هر پالس ساعت ورودی D حالت بعدی خروجی را معین می کند. برای حفظ خروجی و عدم تغییر آن، باید ورودی D را با خروجی Q برابر کنیم.

توجه کنید که پالس های ساعت همیشه به ورودی های C اعمال می شوند. ورودی بارکردن تعیین می کند که پالس بعدی موجب پذیرش اطلاعات شود یا اطلاعات موجود در ثبات بدون تغییر باقی بماند. انتقال اطلاعات از ورودی به ثبات برای هر چهار بیت در یک گذر پالس ساعت بطور همزمان صورت می گیرد.

۵-۲ شیفت رجیسترها^۲

ثباتی که قادر است اطلاعات دودویی را در یک یا دو جهت انتقال دهد شیفت رجیستر (یا ثبات انتقال) نامیده می شود. آرایش منطقی یک شیفت رجیستر از زنجیره ای از فلیپ فلاپهای متوالی تشکیل شده، که در آن خروجی یک فلیپ فلاپ به ورودی فلیپ فلاپ بعدی متصل شده است. تمام فلیپ فلاپها پالس ساعت مشترکی را که موجب جابجایی یک طبقه به طبقه بعدی است دریافت می کنند. ساده ترین شیفت رجیستر ممکن، فقط از فلیپ فلاپها استفاده می کند، شکل ۸-۲. خروجی هر فلیپ فلاپ به ورودی D فلیپ فلاپ سمت راستی وصل شده است. پالس ساعت برای همه آنها مشترک است. ورودی سری تعیین کننده آنچه به سمت چپ ترین فلیپ فلاپ وارد می شود می باشد. خروجی سری نیز از سمت راست ترین فلیپ فلاپ دریافت می شود.



شکل ۸-۲ شیفت رجیستر 4 بیت

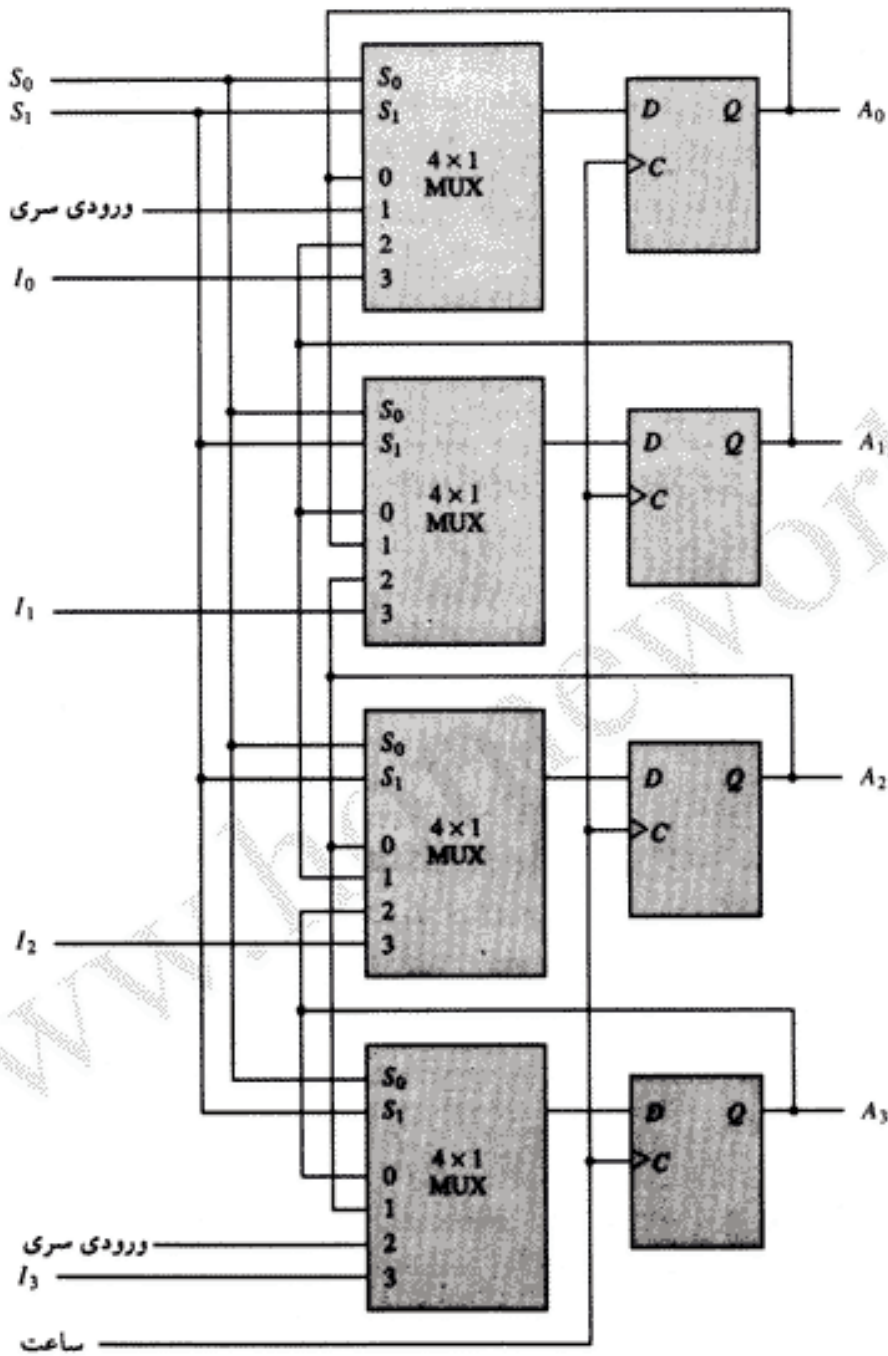
گاهی لازم است شیفت را چنان کنترل کنیم که با وقوع پالس های معینی و نه همه پالس ها بوقوع بپیوندد. این عمل با ممانعت از اعمال پالس ساعت به ورودی ثبات میسر است. اگر شیفت رجیستر شکل ۸-۲ بکار رود، عمل شیفت را می توان با اتصال ساعت به ورودی یک گیت AND کنترل کرد. در این صورت ورودی دوم این گیت می تواند جابجایی را با راه دادن و یا راه ندادن آن کنترل نماید. همچنین می توان با اضافه کردن مدارهایی، عمل جابجایی را از طریق ورودی های D فلیپ فلاپ ها کنترل نمود و نه از طریق ورودی ساعت.

شیفت رجیستر دو جهته با بارشدن موازی

ثباتی که بتواند عمل شیفت را فقط در یک جهت انجام دهد ثبات شیفت رجیستر یک طرفه خوانده می شود. ثباتی که قادر باشد شیفت یا جابجایی را در هر دو جهت انجام دهد شیفت رجیستر دو جهته نامیده می شود. برخی شیفت رجیسترها پایانه های ورودی و خروجی لازم را برای انتقال موازی نیز در اختیار دارند. کامل ترین شیفت رجیستر تمام توانایی های ذکر شده زیر را داراست. ثبات های دیگر ممکن است برخی از این قابلیت ها را داشته باشند. البته هر شیفت رجیستر حداقل یک عمل شیفت را انجام می دهد.

- ۱- یک ورودی پالس های ساعت برای همزمانی تمام عملیات
- ۲- عمل شیفت به راست و یک خط ورودی سری برای شیفت به راست
- ۳- عمل شیفت به چپ و یک خط ورودی سری مربوط به شیفت به چپ
- ۴- عمل بارشدن موازی و n خط ورودی برای این انتقال موازی
- ۵- n خط خروجی موازی
- ۶- یک حالت کنترلی که با وجود اعمال مداوم پالس های ساعت، اطلاعات موجود در ثبات را بدون تغییر باقی گذارد.

یک شیفت رجیستر چهار بیتی با بارشدن موازی در شکل ۹-۲ نشان داده شده است. هر طبقه از یک فلیپ فلاپ D و یک مولتی پلکسر 4×1 تشکیل شده است. دو ورودی انتخاب S_0 و S_1 یکی از ورودی های داده مولتی پلکسر را برای فلیپ فلاپ D انتخاب می کند. خطوط انتخاب شده عملکرد ثبات را طبق جدول تابع ۴-۲ کنترل می کند. هرگاه ورودی های کنترل $S_0 = S_1 = 00$ باشند ورودی داده 0 هر مولتی پلکسر انتخاب می شود. در این وضعیت مسیری از خروجی هر فلیپ فلاپ به ورودی همان فلیپ فلاپ برقرار می شود. پالس ساعت بعدی مقدار دودویی قبلی هر فلیپ فلاپ را بداخل آن انتقال می دهد. و در نتیجه تغییر حالتی رخ نمی دهد. وقتی که $S_0 = S_1 = 01$ باشد، پایانه ای در هر مولتی پلکسر که با 1 علامت زده شده است به ورودی D فلیپ فلاپ مربوطه اش راه می یابد. این خود سبب یک شیفت



شکل ۹-۲ شیفت رجیستر دوطرفه با بار شدن موازی

به راست می‌گردد و ضمن آن داده ورودی سری بداخل فلیپ فلاپ A_0 منتقل می‌شود و بهمین ترتیب محتوای هر فلیپ فلاپ A_{i-1} به فلیپ فلاپ A_i انتقال می‌یابد که در آن $i = 1, 2, 3$ می‌باشد. اگر $S_1 S_0 = 10$ باشد، اطلاعات دودویی از هر ورودی I_0 تا I_3 و از فلیپ فلاپ مربوط به خود می‌شوند و در نتیجه یک

جدول ۲-۴ جدول تابع برای ثبات شکل ۲-۹

کنترل شیوه		عملکرد ثبات
S_1	S_0	
0	0	پلاتنیر
0	1	شیفت به راست (پایین)
1	0	شیفت به چپ (بالا)
1	1	بار شدن موازی

بار شدن موازی صورت گرفته است. دقت کنید که براساس شکل ترسیم شده، شیفت به راست محتوای ثبات‌ها را روبه پائین و شیفت به چپ جابجایی را بسمت بالا باعث می‌گردد. شیفت رجیسترها اغلب برای ارتباط سیستم‌های دیجیتال، با فواصل دور از یکدیگر، بکار می‌روند. مثلاً فرض کنید که بخواهیم کمیتی n بیتی را بین دو نقطه انتقال دهیم. اگر فاصله بین مبدا و مقصد خیلی زیاد باشد، استفاده از n خط برای انتقال موازی آنها پر هزینه خواهد بود. ممکن است استفاده از یک خط واحد مقرون به صرفه‌تر باشد. فرستنده، داده n بیتی موازی را در شیفت رجیستر بار کرده و سپس داده را از خروجی سریال انتقال می‌دهد. گیرنده، داده سری را از طریق خط ورودی سری بداخل یک شیفت رجیستر منتقل می‌سازد. وقتی که محل n بیت در شیفت رجیسترها جای گرفت می‌توان آن را از طریق خروجی‌های موازی در ثبات برداشت کرد. بنابراین فرستنده یک عمل تبدیل موازی به سری داده‌ها را انجام می‌دهد و بالعکس گیرنده داده‌های سری را به موازی باز می‌گرداند.

۲-۶ شمارنده‌های دودویی

ثباتی که با اعمال پالس‌های ورودی رشته‌ای از حالت‌های از پیش تعیین شده را طی می‌کند، شمارنده خوانده می‌شود. پالس‌های ورودی ممکن است پالس‌های ساعت و یا از یک منبع خارجی حاصل شده باشند. همچنین ممکن است فواصل زمانی پالس یکنواخت و یا تصادفی (غیریکنواخت) باشند. شمارنده‌ها را تقریباً در تمام تجهیزاتی که در آن‌ها مدارهای منطقی دیجیتال وجود دارد می‌توان یافت. از شمارنده‌ها می‌توان برای شمارش تعداد دفعات وقوع یک واقعه استفاده کرد و نیز برای تولید سیگنال‌های زمان بندی در کنترل رشته‌ای از اعمال در کامپیوترها بکار می‌برد.

از میان رشته‌های مختلفی که یک شمارنده می‌تواند طی کند، رشته دودویی ساده‌ترین و سراسرترین است. شمارنده‌ای که رشته اعداد دودویی را دنبال می‌کند شمارنده دودویی نام دارد. یک شمارنده دودویی n بیتی، ثباتی است که از n فلیپ فلاپ و گیت‌های مربوطه‌اش تشکیل شده و رشته از حالات را براساس شمارش دودویی n بیت، از 0 تا $2^n - 1$ دنبال می‌کند. طراحی شمارنده‌های دودویی با

توجه به روش ارائه شده برای مدارهای ترتیبی که در بخش ۷-۱ کلیات آن بیان شد انجام می‌گیرد. روش ساده‌تری در طراحی، مستقیماً از بررسی حالاتی که ثبات باید اختیار نماید تا شمارش انجام شود امکان‌پذیر است.

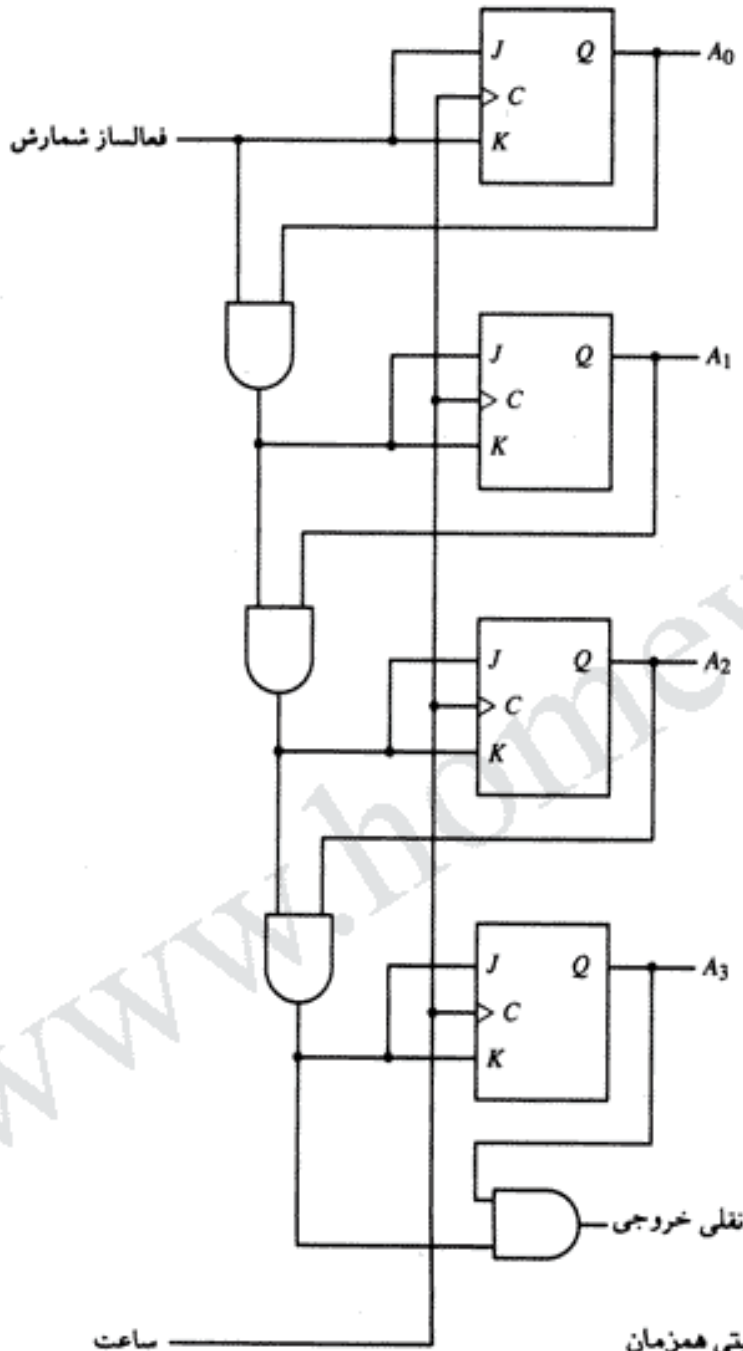
با مرور رشته‌ای از اعداد دودویی مانند 0000، 0001، 0010، 0011 و غیره، در می‌یابیم که بیت پائین رتبه‌تر با هر شمارش متمم می‌شود و هر یک از بیت‌های دیگر هنگام افزایش اگر فقط اگر همه بیت‌های کم ارزش‌تر از آن 1 باشند متمم می‌شود. مثلاً شمارش دودویی از 0111 (7) به 1000 (8) با الف) متمم کردن بیت کم ارزش‌تر، ب) متمم کردن بیت دوم، چون اولین بیت در 0111 برابر 1 است، ج) متمم کردن بیت مرتبه سوم، چون دو بیت اول در 0111 برابر 1 می‌باشد، د) متمم کردن بیت چهارم، چون سه بیت اول در 0111 همگی برابر 1 می‌باشند، بدست می‌آید.

مدار شمارنده معمولاً از فلیپ فلاپهایی که قابلیت متمم شدن دارند استفاده می‌نماید. فلیپ فلاپ‌های T و JK این خاصیت را دارا هستند. بخاطر بیابورید که فلیپ فلاپ JK در صورتی که هر دو ورودی J و K برابر 1 باشند و ساعت گذار مثبت انجام دهد متمم می‌شود. در $J=K=0$ خروجی فلیپ فلاپ تغییر نمی‌کند. بعلاوه، شمارنده را می‌توان توسط یک ورودی تواناساز کنترل کرد تا بدون جدا کردن سیگنال ساعت از فلیپ فلاپها شمارنده را روشن و یا خاموش کند.

همانطور که در شمارنده دودویی 4 بیتی شکل ۱۰-۲ دیده می‌شود اینگونه شمارنده‌های دودویی همزمان (سنکرون) الگوی منظمی دارند. ورودی‌های C در تمام فلیپ فلاپها ساعت مشترکی را دریافت می‌کنند. اگر فعال ساز شمارش 1 باشد تمام ورودی‌های J و K در 0 باقی می‌مانند و خروجی‌های شمارنده تغییری نمی‌کند. طبقه اول یعنی A_0 وقتی که شمارنده فعال شود و ساعت گذار مثبت انجام دهد متمم می‌گردد. هر یک از سه فلیپ فلاپ دیگر هنگامی متمم می‌شوند که تمام فلیپ فلاپ‌های پائین رتبه‌شان 1 و شمارش نیز فعال شود. زنجیره گیت‌های AND منطبق لازم را برای ورودی‌های J و K تهیه می‌کنند. با استفاده از رقم نقلی خروجی می‌توان شمارنده را به طبقات بیشتری گسترش داد که در هر طبقه یک فلیپ فلاپ و یک گیت AND مورد نیاز است.

شمارنده دودویی با بارشدن موازی

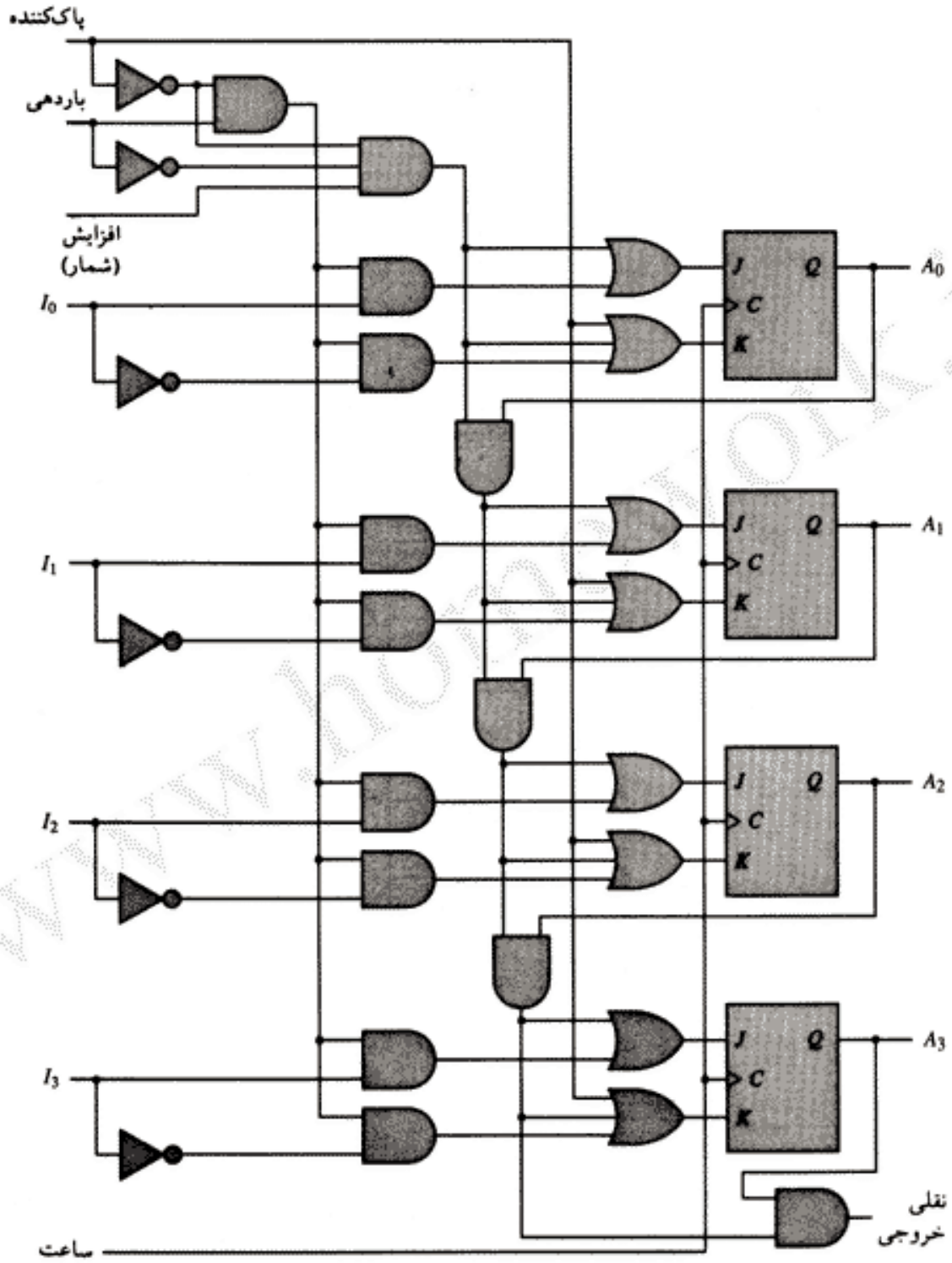
شمارنده‌هایی که در سیستم‌های دیجیتال مورد استفاده قرار می‌گیرند اغلب نیاز به امکان بارشدن موازی برای دریافت یک مقدار اولیه قبل از شمارش دارند. شکل ۱۱-۲ دیاگرام منطقی یک شمارنده دودویی که دارای امکان بارشدن موازی و نیز پاک شدن همزمان به 0 می‌باشد را نشان می‌دهد. اگر ورودی پاک کننده 1 باشد، همه ورودی‌های K را 1 می‌کند و بنابراین با گذار بعدی ساعت همه فلیپ فلاپ‌ها را 0 می‌نماید. اگر ورودی کنترل بارکردن برابر 1 باشد، عمل شمارش را غیرفعال نموده و موجب می‌شود تا چهار ورودی داده بداخل چهار فلیپ فلاپ انتقال یابند (بشرطی که ورودی پاک کردن



شکل ۱۰-۲ شمارنده همزمان ۴ بیتی همزمان

0 باشد). اگر هر دو ورودی پاک کردن و بار کردن 0 و ورودی افزایش 1 باشد مدار بصورت یک شمارنده دودویی عمل خواهد کرد.

1- Increment



شکل ۱۱-۲ شمارنده دودویی 4 بیتی با بارشدهن موازی و پاک شدن همزمان

جدول ۲-۵ جدول تابع برای ثبات شکل ۲-۱۱

عمل	افزایش	بار شدن	پاک شدن	ساعت
بلا تغییر	0	0	0	↑
1 واحد افزایش شمارش	1	0	0	↑
بار کردن ورودی های I ₀ تا I ₃	x	1	0	↑
پاک کردن خروجی ها به 0	x	x	1	↑

عملکرد مدار در جدول ۲-۵ خلاصه شده است. اگر همه ورودی های پاک کردن، بار کردن و افزایش 0 باشند هیچ تغییر در خروجی نخواهد بود حتی اگر پالس ها به ورودی C اعمال گردند. اگر ورودی های پاک و بار کردن در 0 نگهداشته شود، ورودی افزایش عملکرد را کنترل نموده و خروجی ها بازاء هر گذار پالس ساعت به عدد دودویی بعدی در شمارش تغییر خواهند کرد. هر وقت ورودی بار کردن 1 باشد، بشرطی که ورودی پاک کردن ناتوان (غیر فعال) ولی ورودی افزایش 0 یا 1 باشد، داده های ورودی بداخل فلیپ فلاپ ها بار می شوند. با اعمال ورودی پاک کردن، ثبات بدون توجه به مقادیر ورودی های بار کردن و افزایش، 0 خواهد شد.

شمارنده هایی که دارای قابلیت بار کردن موازی هستند در طراحی کامپیوترهای دیجیتال خیلی مفیدند. در فصل های بعدی، ما از آنها بعنوان ثبات هایی که امکان افزایش و بار شدن دارند استفاده خواهیم کرد. عمل افزایش محتوای یک ثبات را یک واحد افزایش می دهد. با توانا کردن ورودی شمارش در طول یک پرپود ساعت، محتوای ثبات می تواند یک واحد افزایش یابد.

۲-۷ واحد حافظه

واحد حافظه مجموعه ای از سلول های ذخیره سازی به همراه مدارات لازم برای انتقال اطلاعات به داخل و خارج آنهاست. حافظه اطلاعات دودویی را بصورت دسته هایی از بیت بنام کلمه^۱ ذخیره می کند. یک کلمه در حافظه مجموعه ای از بیت هاست که بصورت یک واحد، به داخل و خارج حافظه انتقال می یابد. کلمه حافظه^۲ گروهی از 1 ها و 0 هاست و ممکن است بعنوان یک عدد، یک کد دستورالعمل، یک یا چند کاراکتر الفبا عددی یا هر اطلاعات دودویی کد شده دیگر باشد. یک گروه هشت بیتی، بایت^۳ خوانده می شود. در اغلب حافظه های کامپیوتر، تعداد بیت های هر کلمه مضربی از 8 است. بنابراین یک کلمه 16 بیتی حاوی دو بایت است، و یک کلمه 32 بیتی نیز از چهار بایت ساخته می شود. ظرفیت حافظه کامپیوترهای تجاری معمولاً بصورت تعداد بایت هایی که می توان در آن ذخیره کرد، بیان می شود. ساختمان درونی یک واحد حافظه بوسیله تعداد کلمات آن و تعداد بیت های هر کلمه مشخص

1- word

2- Memory word

3- Byte

می‌گردد. خطوط ورودی خاصی که کلمه بخصوصی را انتخاب می‌کنند خطوط آدرس¹ نام دارند. به هر کلمه در حافظه یک شماره شناسایی تعلق می‌گیرد که به آن آدرس اتلاق می‌شود، و از 0 آغاز و با 1، 2، 3 تا 2^k-1 ادامه می‌یابد که در آن k تعداد خطوط آدرس است. انتخاب یک کلمه خاص در داخل حافظه با اعمال آدرس دودویی K بیتی به خطوط آدرس تحقق می‌یابد. دیکدوری در داخل حافظه این آدرس را دریافت کرده و مسیرهای لازم را برای انتخاب بیت‌های کلمه خاص باز می‌کند. حافظه‌های کامپیوتر ممکن است از 1024 کلمه، که 10 خط آدرس نیاز دارند، تا 2^{32} کلمه، که 32 خط آدرس لازم دارند، متغیر باشد. رسم براین است که تعداد کلمات (یا بایت‌ها) را در یک حافظه با یکی از حروف K (کیلو)²، M (مگا)³ یا G (گیگا)⁴ بیان کنند. K برابر با 2^{10} ، M برابر با 2^{20} و G برابر با 2^{30} می‌باشد. بنابراین $64K=2^{16}$ ، $2M=2^{21}$ و $4G=2^{32}$.

دو نوع عمده حافظه که در سیستم‌های کامپیوتری بکار می‌روند عبارتند از: حافظه با دستیابی تصادفی⁵ (RAM) و حافظه فقط خواندنی⁶ (ROM).

حافظه با دستیابی تصادفی

در حافظه با دستیابی تصادفی (RAM)، انتقال اطلاعات از هر مکان دلخواه بصورت تصادفی و دلخواه امکان‌پذیر است. یعنی عمل نشانیدن یک کلمه در هر حافظه یکسان بوده و زمان لازم برای آن مستقل از مکان فیزیکی سلول در حافظه است. مفهوم "دستیابی تصادفی" نیز از همین مطلب ناشی می‌شود. ارتباط بین حافظه و محیط اطراف آن از طریق خطوط داده ورودی و خروجی، انتخاب آدرس، و خطوط کنترل که جهت انتقال را مشخص می‌کنند صورت می‌گیرد. بلاک دیاگرام یک واحد RAM در شکل ۱۲-۲ نشان داده شده است. n خط ورودی داده اطلاعات را که باید در حافظه ذخیره شوند فراهم می‌آورند، و n خط خروجی داده نیز اطلاعاتی را که از حافظه بیرون می‌آیند در اختیار می‌گذارند. k خط آدرس یک عدد دودویی k بیتی را برای انتخاب کلمه موردنظر از میان 2^k کلمه، در داخل حافظه فراهم



شکل ۱۲-۲ بلاک دیاگرام یک حافظه با دستیابی تصادفی (RAM)

- | | | |
|--------------------------|------------------------------|---------|
| 1- Address | 2- kilo | 3- Mega |
| 4- Giga | 5- Random Access Memory, RAM | |
| 6- Read Only Memory, ROM | | |

می آورند. دو خط کنترل ورودی جهت انتقال موردنظر را مشخص می کنند. در یک حافظه RAM می توان دو عمل نوشتن^۱ و خواندن^۲ را انجام داد. سیگنال نوشتن، انتقال به داخل و خواندن، انتقال به خارج را میسر می سازند. در پذیرش یکی از این دو سیگنال کنترل، مدارات داخلی حافظه اعمال لازم را انجام می دهند. مراحلی که برای انتقال یک کلمه جدید و ذخیره آن در حافظه باید طی شود بقرار زیر است.

- ۱- اعمال آدرس کلمه مورد نظر به خطوط آدرس
- ۲- اعمال بیت های داده ای که قرار است در حافظه ذخیره شوند، بر روی خطوط ورودی داده.
- ۳- فعال کردن ورودی نوشتن.

سپس واحد حافظه بیت های موجود بر روی خطوط ورودی داده را دریافت کرده و آنها را در کلمه تعیین شده بوسیله خطوط آدرس جای می دهد.

مراحلی که برای انتقال یک کلمه به خارج از حافظه باید طی شود بقرار زیر است.

- ۱- اعمال آدرس دودویی کلمه موردنظر بر روی خطوط آدرس
- ۲- فعال کردن ورودی خواندن

سپس واحد حافظه، بیت های کلمه انتخاب شده بوسیله آدرس را برداشته و بر روی خطوط خروجی داده قرار می دهد. محتوای کلمه انتخابی پس از خواندن تغییری نمی کند.

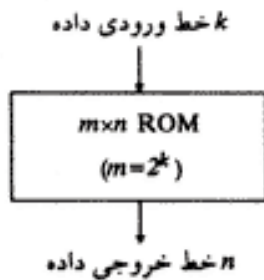
حافظه فقط خواندنی

حافظه فقط خواندنی (ROM) همانطور که از نامش مشخص است، حافظه ای است که فقط عمل خواندن را انجام می دهد؛ یعنی دارای امکان نوشتن نیست. این بدان معنی است که اطلاعات دودویی ذخیره شده در ROM حین تولید سخت افزار در آن بصورت دائمی در آمده است و نمی توان با نوشتن کلمات مختلف در آن، محتوای آن را عوض کرد. در حالیکه RAM یک وسیله همه منظوره است و محتوای آن در طول پردازش قابل تغییر می باشد قطعه ROM محدود به خواندن کلماتی است که در داخل آن بطور دائم ذخیره شده است. اطلاعات دودویی که باید در ROM ذخیره شود بوسیله طراح مشخص شده و سپس هنگام ساخت قطعه با شکل خاصی از اتصالات داخلی در آن پیاده می شود. نوعی از ROM ها دارای فیوزهای الکترونیکی خاص هستند که می توان آنها را برای آرایش خاصی "برنامه دهی" کرد. به محض ایجاد الگوی موردنظر، حتی با قطع برق نیز اطلاعات در آن باقی خواهد ماند.

یک ROM با اندازه $n \times m$ آرایه ای از سلول های دودویی است که بصورت m کلمه n بیتی سازمان دهی شده است. همانطور که در شکل ۱۳-۲ دیده می شود، ROM دارای k خط ورودی آدرس

1- Write

2- Read



شکل ۱۳-۲ بلاک دیاگرام حافظه فقط خواندنی (ROM)

برای انتخاب $m = 2^k$ کلمه از حافظه، و n خط خروجی است که هر یک از این خطوط به یک بیت از هر کلمه مرتبط است. یک مدار مجتمع ROM ممکن است دارای یک یا چند ورودی فعال ساز (تواناساز) بوده و برای تشکیل یک ROM با ظرفیت بیشتر بهم متصل شوند.

ROM به خط کنترل خواندن نیازی ندارد زیرا در هر زمان معین، خطوط خروجی خودبخود n بیت کلمه انتخاب شده توسط مقدار آدرس را در اختیار می‌گذارند. چون خروجی‌ها فقط تابعی از ورودی‌های همان لحظه (خطوط آدرس) هستند، ROM را بعنوان مدارهای ترکیبی طبقه‌بندی می‌نمایند. در واقع ساختار داخلی یک ROM از یک دیکدر و مجموعه‌ای از گیت‌های OR تشکیل شده است. در این وسیله نیازی به وجود عنصر ذخیره کننده، همچون RAM، نیست زیرا مقادیر بیت‌ها در ROM بصورت دائمی تثبیت شده‌اند.

ROM ها کاربرد گسترده‌ای در طراحی سیستم‌های دیجیتال دارند. یک ROM اساساً یک رابطه ورودی - خروجی مشخص شده بوسیله یک جدول درستی را تولید می‌کند. بنابراین، هر مدار ترکیبی را با k ورودی و n خروجی پیاده‌سازی می‌کند. وقتی که ROM در یک سیستم کامپیوتری بعنوان واحد حافظه بکار گرفته شد به منظور ذخیره کردن برنامه‌های ثابتی که تغییری نمی‌کنند و یا جداول حاوی ثابت‌ها مورد استفاده قرار گرفته است. ROM در ساخت واحد کنترل کامپیوتر دیجیتال نیز بکار گرفته می‌شود. در این حال، این واحد برای ذخیره اطلاعات کد شده‌ای که نماینده رشته مستغیرهای کنترل داخلی لازم برای فعال کردن اعمال مختلف در کامپیوتر است استفاده می‌شود. یک واحد کنترل که از ROM برای ذخیره سازی اطلاعات دودویی کنترل استفاده می‌کند یک واحد کنترل ریزبرنامه نویسی^۱ نام دارد. این موضوع در فصل ۷ مفصل‌تر بحث شده است.

انواع ROM

مسیرهای لازم در یک ROM به سه طریق مختلف قابل برنامه ریزی است. اولین روش، یعنی

برنامه ریزی با ماسک^۱، در حین آخرین فرآیند ساخت توسط کمپانی سازنده نیمه رسانا، انجام می شود. در روش ساخت ROM، مشتری جدول درستی آنچه را که ROM باید بر آن اساس کار کند پر می نماید. جدول درستی ممکن است بروش های مختلفی تهیه و یا با قالب خاصی در خروجی کامپیوتر ارائه شود. سازنده، ماسک مناسبی را برای مسیرها تهیه می کند تا 1ها و 0ها بر طبق جدول درستی مشتری تهیه شوند. این روش پرهزینه است زیرا فروشنده، مبلغ خاصی را بخاطر ساخت سفارشی ROM تقاضا می کند. به این دلیل، برنامه ریزی با ماسک فقط هنگامی اقتصادی است که تعداد زیادی ROM با آرایش یکسان سفارش گردد.

برای تعداد کم، روش دوم که حافظه قابل برنامه ریزی فقط خواندنی،^۲ PROM نام دارد اقتصادی تر است. بهنگام سفارش تمام فیوزهای PROM دست نخورده هستند و لذا تمام بیت های ذخیره شده در کلمه 1 خواهند بود. فیوزهای PROM را می توان با اعمال پالس های جریان از طریق پایانه های خروجی برای هر یک از آدرس ها سوزاند. یک فیوز سوخته حالت دودویی 0 را بوجود می آورد و فیوز دست نخورده حالت دودویی 1 را می دهد. این مطلب به کاربر این امکان را می دهد تا PROM مورد نظر خود را در آزمایشگاه خود برای بدست آوردن رابطه بین آدرس ورودی و کلمه ذخیره شده برنامه نویسی کند. دستگاه های خاصی بنام برنامه نویس PROM برای انجام این عمل بصورت تجاری موجود است. در هر صورت، روش های موجود در برنامه نویسی ROMها سخت افزاری می باشند هر چند که از کلمه "برنامه نویسی" هم در بیان آن استفاده شده باشد.

اعمال برنامه نویسی سخت افزاری ROMها و PROMها غیرقابل برگشتند، و به محض اینکه برنامه نویسی شدند، الگوی ثابتی در آنها ایجاد شده و دیگر تغییر نخواهند کرد. چنانچه قرار باشد پس از ایجاد الگوی بیت ها، آنها را تغییر دهیم، قطعه باید کلاً دور انداخته شود. نوع سومی از ROMها نیز وجود دارد که PROM پاک شونده یا EPROM خوانده می شود. EPROM، حتی اگر فیوزهایش قبلاً سوزانده شده باشد، می توان بصورت اولیه بازسازی کرد. وقتی EPROM در زیر اشعه ماوراء بنفش برای مدت معینی قرار گیرد، تابش طول موج کوتاه گیت های داخلی را که نقش فیوز را به عهده دارند تخلیه می نماید. پس از پاک کردن، EPROM به حالت اولیه اش باز می گردد و می تواند با مجموعه جدیدی از کلمات برنامه نویسی گردد. انواع بخصوصی از PROM را می توان با سیگنال های الکتریکی، بجای نور ماوراء بنفش، پاک کرد. این PROMها را PROM پاک شدنی الکتریکی یا EEPROM می نامند.

مسائل

۲-۱ قطعات TTL SSI اغلب بصورت مدارات مجتمع 14 پایه عرضه می شوند. دو پایه برای منبع تغذیه در نظر گرفته شده و بقیه برای پایانه های ورودی و خروجی بکار می روند. چند مدار از انواع زیر را در یک چنین بسته ای می توان گنجانید؟ الف) معکوس کننده، ب) گیت OR انحصاری

1- Mask Programming

2- Programmable Read Only Memory

- دو ورودی، ج) گیت OR سه ورودی، د) گیت AND چهار ورودی، ه) گیت NOR پنج ورودی، و) گیت NAND هشت ورودی، ز) فلیپ فلاپ JK ساعت دار با پاک کننده غیرهمزمان.
- ۲-۲ تراشه های MSI وجود توابع دیجیتال ساده ای همچون دیکدر، مولتی پلکسر، ثبات ها و شمارنده ها را امکان پذیر می سازند. تراشه های زیر مدارهای مجتمع از نوع TTL هستند که چنین توابعی را تولید می نمایند. مشخصات آنها را در کتاب راهنما یافته و آنها را با قطعات متناظر ارائه شده در این فصل مقایسه کنید.
- الف) IC نوع 74155، دیکدر دو تایی 2 به 4
- ب) IC نوع 74157، مولتی پلکسر چهار تایی 2 به 1 خطی
- ج) IC نوع 74194، شیفت رجیستر چهار بیت دوطرفه با بارشدن موازی
- د) IC نوع 74163، شمارنده دودویی چهاربیت بآبارشدن موازی و پاک شدن همزمان
- ۲-۳ یک دیکدر 5 به 32 را با چهار دیکدر 3 به 8 که دارای ورودی تواناساز است و یک دیکدر 2 به 4 بسازید. از بلاک دیاگرام مشابه شکل ۲-۳ استفاده کنید.
- ۲-۴ دیاگرام منطقی یک دیکدر 2 به 4 را فقط با گیت NOR ترسیم کنید. ورودی فعال ساز نیز در نظر گرفته شود.
- ۲-۵ دیکدر شکل 2-2 را طوری تصحیح کنید که مدار اگر $E=1$ باشد فعال و اگر $E=0$ باشد غیرفعال باشد. جدول درستی را برای نوع تصحیح شده رسم کنید.
- ۲-۶ دیاگرام منطقی یک آنکدر با هشت ورودی و سه خروجی، که جدول درستی آن در جدول ۲-۲ آورده شد را ترسیم کنید. وقتی تمام ورودی ها 0 باشند خروجی چیست؟ اگر فقط ورودی D برابر 0 باشد خروجی چیست. روشی را پیشنهاد کنید که این دو حالت را از هم تمیز دهد.
- ۲-۷ یک مولتی پلکسر 16 به 1 را با دو مولتی پلکسر 8 به 1 و یک مولتی پلکسر 2 به 1 بسازید. برای هر سه مولتی پلکسر از بلاک دیاگرام استفاده کنید.
- ۲-۸ بلاک دیاگرام یک مولتی پلکسر را ترسیم کرده و عملکرد آن را بوسیله جدول تابع توضیح دهید.
- ۲-۹ یک گیت AND دو ورودی را در ثبات شکل ۲-۶ بگنجانید و خروجی این گیت را به ورودی های ساعت همه فلیپ فلاپ ها وصل کنید. یکی از ورودی های گیت AND پالس های ساعت را از مولد پالس ساعت دریافت می کند. ورودی دیگر گیت AND کنترل بارشدن موازی را فراهم می آورد. عملکرد ثبات جدید را تشریح کنید.
- ۲-۱۰ هدف از گیت بافر در ورودی ساعت ثبات شکل ۲-۷ چیست؟
- ۲-۱۱ به ثباتی که دارای امکان بارشدن موازی در شکل ۲-۷ است، امکان ورودی پاک کننده را اضافه کنید.
- ۲-۱۲ مقدار اولیه محتوای یک ثبات 1101 است. ثبات شش بار با استفاده از ورودی سری 101101 به راست شیفت داده می شود. محتوای ثبات پس از شیفت چیست.

- ۲-۱۳ فرق بین انتقال سری و موازی چیست؟ با استفاده از یک شیفت رجیستر که قابلیت بارشدن موازی دارد توضیح دهید چگونه می توان ورودی سری را به خروجی موازی و بالعکس تبدیل کرد.
- ۲-۱۴ شمارنده حلقوی یک شیفت رجیستر مطابق شکل ۲-۸ می باشد که خروجی سری اش به ورودی سری آن وصل شده است. با شروع از حالت اولیه 1000، رشته حالات چهار فلیپ فلاپ را پس از هر شیفت لیست کنید.
- ۲-۱۵ یک شیفت رجیستر چهاربیتی دو طرفه با قابلیت بار شدن موازی طبق شکل ۲-۹ در یک بسته مدار مجتمع بسته بندی شده است.
- الف) بلاک دیاگرام مدار مجتمع را با تمام ورودی ها و خروجی هایش ترسیم کنید.
ب) با استفاده از دو مدار مجتمع بلاک دیاگرام یک شیفت رجیستر 8 بیت با قابلیت بار شدن موازی را رسم کنید.
- ۲-۱۶ در یک شمارنده دودویی ده بیت چند فلیپ فلاپ در شمارش بعدی متمم می شوند.
الف) 1001100111 (ب) 0011111111
- ۲-۱۷ اتصالات لازم بین چهار شمارنده دودویی چهاربیتی با بار شدن موازی (شکل ۲-۱۱) برای ایجاد یک شمارنده دودویی 16 بیت با بار شدن موازی را نشان دهید. برای هر شمارنده چهاربیتی از یک بلاک دیاگرام استفاده کنید.
- ۲-۱۸ نشان دهید که چگونه یک شمارنده دودویی با بار شدن موازی شکل ۲-۱۱ را می توان به یک شمارنده تقسیم بر N تبدیل کرد (یعنی شمارنده ای که از 0000 تا N شمرد و به 0000 باز گردد). بعنوان یک حالت خاص مدار یک تقسیم بر 10 را با استفاده از شکل ۲-۱۱ و یک گیت AND خارجی رسم کنید.
- ۲-۱۹ واحدهای حافظه زیر با تعداد کلمات، ضرب در تعداد بیت ها مشخص شده اند. در هر مورد چند خط آدرس و چند خط داده ورودی - خروجی لازم است؟
الف) 2K×16 (ب) 64K×8
ج) 16M×32 (د) 4G×64
- ۲-۲۰ تعداد بایت هایی را که می توان در حافظه های مسئله ۲-۱۹ ذخیره کرد مشخص کنید.
- ۲-۲۱ چند تراشه حافظه 128×8 برای ایجاد یک حافظه 4096×16 لازم است؟
- ۲-۲۲ با فرض داشتن یک ROM به ظرفیت 32×8 بیت و یک ورودی تواناساز، اتصالات خارجی لازم برای ساخت یک تراشه ROM با ظرفیت 128×8 و یک دیکدر را نشان دهید.
- ۲-۲۳ یک تراشه ROM با ظرفیت 4096×8 دارای دو ورودی تواناساز بوده و با منبع تغذیه 5 ولت کار می کند. چند پایه برای بسته مدار مجتمع لازم است؟ بلاک دیاگرام آن را رسم و پایانه های ورودی و خروجی را نام گذاری کنید.

۳

نمایش داده‌ها

۳-۱ انواع داده‌ها

۳-۲ متمم‌ها

۳-۳ نمایش ممیز - ثابت

۳-۴ نمایش ممیز شناور

۳-۵ انواع دیگر کدهای دودویی

۳-۶ کدهای آشکارسازی خطا

۱-۳ انواع داده‌ها

اطلاعات دودویی در کامپیوترهای دیجیتال در حافظه‌ها یا ثبات‌های پردازنده ذخیره می‌شوند. ثبات‌ها حاوی داده‌ها یا اطلاعات کنترل هستند. اطلاعات کنترل متشکل از یک بیت یا گروهی از بیت‌هاست که برای مشخص کردن سیگنال‌های فرمان لازم در دستکاری داده‌های سایر ثبات‌ها استفاده می‌شود. داده‌ها، اعداد و اطلاعات دودویی کد شده دیگری می‌باشند که برای حصول به نتایج محاسباتی مورد نظر روی آنها کار می‌شود. در این فصل متداول‌تر نوع داده‌ها را که در کامپیوترهای دیجیتال یافت می‌شوند معرفی می‌کنیم و چگونگی نمایش انواع مختلف داده‌ها، بشکل کد شده دودویی در ثبات‌های کامپیوتر را نشان خواهیم داد.

انواع داده‌های موجود در ثبات‌های کامپیوترهای دیجیتال را می‌توان به ۳ فرم زیر طبقه‌بندی کرد:

۱) اعدادی که در عملیات حسابی بکار می‌روند، ۲) حروف الفبا مورد استفاده در پردازش داده، و ۳) سایر سمبل‌های گسسته که برای اهداف خاصی بکار می‌روند. همه انواع داده‌ها، بجز اعداد دودویی، به فرم دودویی کد شده در ثبات‌های کامپیوتر نمایش داده می‌شوند. این بدان علت است که ثبات‌ها از

فلیپ فلاپ ساخته شده‌اند و فلیپ فلاپ‌ها وسیله دوحالتی هستند که قادرند فقط 1ها و 0ها را در خود ذخیره کنند. سیستم اعداد دودویی طبیعی‌ترین سیستم بکار رفته در یک کامپیوتر دیجیتال است. ولی گاهی مناسبتر است سیستم اعداد دیگری را بکار برد. بخصوص سیستم اعداد دهدهی به این دلیل که بوسیله انسان جهت عملیات حسابی بکار می‌روند.

سیستم‌های عددنویسی

یک سیستم عددنویسی با پایه 1 یا مبنای 2 سیستمی است که سمبل‌های متمایز از همی را برای نمایش 2 رقم بکار می‌برد. اعداد بارشته‌ای از سمبل‌های ارقام نمایش داده می‌شوند. برای تعیین کمیتی که یک عدد نمایشگر آنست، لازم است تا هر رقم را در توان صحیحی از 2 ضرب کرده و سپس مجموع همه این حاصلضرب‌ها محاسبه شود. مثلاً، سیستم اعداد دهدهی که همه روزه مورد استفاده است سیستم مبنای 10 را بکار می‌برد. 10 سمبل عبارتند $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$. رشته رقم 724.5 بیانگر کمیت زیر تلقی می‌شود.

$$7 \times 10^2 + 2 \times 10^1 + 4 \times 10^0 + 5 \times 10^{-1}$$

یعنی 7 صدتایی بعلاوه 2 ده تایی بعلاوه 4 یکی بعلاوه 5 دهم. بطور مشابه هر عدد دهدهی را می‌توان بهمین ترتیب برای یافتن کمیت آن تفسیر کرد
سیستم عددنویسی دودویی مبنای 2 را بکار می‌برد. دو سمبل ارقام مورد استفاده، 0 و 1 هستند. رشته ارقام 101101 بیانگر کمیت زیر است.

$$1 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 45$$

برای تفکیک اعداد با مبناهای مختلف، اعداد در پرانتزهایی قرار داده شده و مبنای عدد بصورت اندیس در زیر پرانتز قرار می‌گیرد. مثلاً برای نشان دادن تساوی بین 45 و 101101 می‌نویسیم $(101101)_2 = (45)_{10}$.

علاوه بر سیستم‌های اعداد دودویی و دهدهی، سیستم‌های هشت هشتی (مبنای 8) و شانزده شانزدهی (مبنای 16) نیز در کامپیوترهای دیجیتال اهمیت دارند. هشت سمبل اعداد هشت هشتی عبارتند از: $0, 1, 2, 3, 4, 5, 6, 7$. شانزده سمبل اعداد شانزده شانزدهی عبارتند از: $0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F$. شش سمبل آخر، متاسفانه، همان حروف الفبا هستند و گاهی ممکن است موجب اشتباه شوند. با این وجود، این قراردادی است که پذیرفته شده است. وقتی که حروف A, B, C, D, E, F در مبنای شانزده بکار روند به ترتیب متناظر با اعداد $10, 11, 12, 13, 14, 15$ می‌باشند.

یک عدد در مبنای 2 را با تشکیل جمع حاصلضرب‌های ارقام در وزن‌هایشان می‌توان به سیستم

1- Base

2- Radix

دهدهی تبدیل کرد. مثلاً عدد هشت هشتی 736.4 به صورت زیر به دهدهی تبدیل می شود:

$$(736.4)_8 = 7 \times 8^2 + 3 \times 8^1 + 6 \times 8^0 + 4 \times 8^{-1} \\ = 7 \times 64 + 3 \times 8 + 6 \times 1 + 4 : 8 = (478.5)_{10}$$

عدد دهدهی معادل عدد F3 در مبنای 16 از محاسبه زیر بدست می آید

$$(F3)_{16} = F \times 16 + 3 = 15 \times 16 + 3 = (243)_{10}$$

تبدیل از دهدهی به معادل مبنای r آن، با جداسازی بخش صحیح عدد از کسری آن و تبدیل هر بخش بطور جداگانه صورت می گیرد. تبدیل یک عدد صحیح دهدهی به مبنای r بوسیله تقسیمات متوالی بر r و کنار هم گذاشتن باقیمانده ها انجام می شود. شکل ۱-۳ این عملیات را نشان می دهد.

تبدیل عدد دهدهی 41.6875 به دودویی ابتدا با جدا کردن عدد به دو قسمت صحیح 41 و کسری 0.6875 آغاز می شود. برای تبدیل بخش صحیح، 41 را بر 2 تقسیم می کنیم تا خارج قسمت صحیح 20 و باقیمانده 1 بدست آید. خارج قسمت را دوباره بر 2 تقسیم می کنیم تا خارج قسمت و باقیمانده جدیدی بدست آید. این روند تا 0 شدن خارج قسمت ادامه می یابد. ضرائب عدد دودویی از باقیمانده ها حاصل می گردد، بدین ترتیب که اولین باقیمانده بیت مرتبه پائین عدد دودویی تبدیل شده را تشکیل می دهد.

بخش کسری عدد با ضرب آن در $r=2$ حاصل می گردد تا یک عدد صحیح همراه با کسری تولید کند. قسمت کسری جدید (بدون بخش صحیح آن) دوباره در 2 ضرب می شود تا عدد صحیح همراه با کسر دیگری بدست آید. این روند تا صفر شدن بخش کسری و یا تاجایی که ارقام حاصل پاسخگوی دقت مورد نظر بشوند ادامه دارد. ضرائب کسری دودویی از ارقام صحیح و با توجه به قرار گرفتن اولین عدد صحیح بعنوان رقم بعد از ممیز تشکیل می گردد. نهایتاً دو قسمت برای تبدیل کل با یکدیگر ترکیب می گردند.

41 = بخش صحیح

$$\begin{array}{r|l} 41 & \\ 20 & 1 \\ 10 & 0 \\ 5 & 0 \\ 2 & 1 \\ 1 & 0 \\ 0 & 1 \end{array}$$

$$(41)_{10} = (101001)_2$$

0.6875 = بخش کسری

$$\begin{array}{r} 0.6875 \\ \underline{2} \\ 1.3750 \\ \times 2 \\ \underline{0.}7500 \\ \times 2 \\ \underline{1.}5000 \\ \times 2 \\ \underline{1.}0000 \end{array}$$

$$(0.6875)_{10} = (0.1011)_2$$

$$(41.6875)_{10} = (101001.1011)_2$$

شکل ۱-۳ تبدیل 41.6875 دهدهی به دودویی

جدول ۱-۳ اعداد هشت هشتی کد شده با دودویی

عدد هشت هشتی	هشت هشتی کد شده با دودویی	معادل دهدهی	
0	000	0	↑ کد یک رقم هشت هشتی ↓
1	001	1	
2	010	2	
3	011	3	
4	100	4	
5	101	5	
6	110	6	
7	111	7	
<hr/>			
10	001 000	8	
11	001 001	9	
12	001 010	10	
24	010 100	20	
62	110 010	50	
143	001 100 011	99	
370	011 111 000	248	

جدول ۲-۳ اعداد شانزده شانزدهی کد شده با دودویی

عدد شانزده شانزدهی	شانزده شانزدهی کد شده با دودویی	معادل دهدهی	
0	0000	0	↑ کد یک رقم شانزده شانزدهی ↓
1	0001	1	
2	0010	2	
3	0011	3	
4	0100	4	
5	0101	5	
6	0110	6	
7	0111	7	
8	1000	8	
9	1001	9	
A	1010	10	
B	1011	11	
C	1100	12	
D	1101	13	
E	1110	14	
F	1111	15	
<hr/>			
14	0001 0100	20	
32	0011 0010	50	
63	0110 0011	99	
F8	1111 1000	248	

1100011 خواهد بود. معادل هشت هشتی عدد دهدهی 99 برابر است با 001 100 011 و معادل شانزده شانزدهی آن 0110 0011 می باشد. اگر ما صفرهای سمت چپ را در این سه فرم نادیده بگیریم، می بینیم که ترکیب بیت های آنها یکی است. در واقع باید هم همینطور باشد زیرا تبدیل مستقیمی بین اعداد دودویی و هشت هشتی یا شانزده شانزدهی صورت گرفته است. نکته ای که در این بحث وجود دارد این است که یک رشته از 1ها و 0ها که در یک ثبات ذخیره شده اند می توانند یک عدد دودویی را نمایش دهند، ولی همین رشته می تواند بعنوان یک عدد هشت هشتی با کد دودویی هم تفسیر شود (اگر ما بیت ها را به گروه های سه بیتی تقسیم کنیم) و یا بعنوان یک عدد شانزده شانزدهی با کد دودویی تعبیر گردد.

ثبات ها در کامپیوترهای دیجیتال حاوی بیت های زیادی هستند. مشخص کردن محتوای ثبات ها با مقادیر دودویی آنها، رشته ای طولانی از ارقام دودویی را نیاز خواهد داشت. بهتر این است که محتوای ثبات ها با معادل هشت هشتی یا شانزده شانزدهی مشخص شوند. بدین ترتیب تعداد ارقام به یک سوم در هشت هشتی، و به یک چهارم در شانزده شانزدهی تقلیل می یابد. مثلاً عدد دودویی 1111 1111 دارای 12 رقم است. این عدد در نمایش هشت هشتی بصورت 7777 (چهاررقم) و یا در نمایش شانزده شانزدهی بشکل FFF (سه رقم) می باشد. در کتب راهنمای کامپیوترها همیشه از نمایش هشت هشتی یا شانزده شانزدهی برای مشخص کردن محتوای ثبات ها استفاده می شود.

نمایش دهدهی

سیستم اعداد دودویی طبیعی ترین سیستم برای یک کامپیوتر است، ولی انسان ها به دستگاه دهدهی عادت دارند. یک راه حل برای این مشکل تبدیل تمام اعداد دهدهی ورودی به اعداد دودویی، انجام تمام اعمال حسابی به شکل دودویی و سپس تبدیل مقادیر دودویی به دهدهی برای درک فرد کاربر است. با این وجود، می توان اعمال محاسباتی را در کامپیوتر مستقیماً با اعداد دهدهی انجام داد بشرطی که آنها در ثبات ها بشکل کد شده ذخیره باشند. اعداد دهدهی معمولاً بصورت کاراکترهای الفبا عددی کد شده دودویی وارد کامپیوتر می شوند. این کدها، که بعداً معرفی خواهند شد، برای هر رقم دهدهی بین شش تا هشت بیت است. وقتی که اعداد دهدهی برای محاسبات حسابی داخلی بکار روند، تبدیل به یک دودویی چهار بیت برای هر رقم می گردند.

یک کد دودویی گروهی از n بیت است که تا 2^n ترکیب متمایز از 0ها و 1ها را بخود می گیرد. هر ترکیب نمایشگر یک عنصر از مجموعه ای است که قرار است بصورت کد درآید. مثلاً، مجموعه ای از چهار عنصر بوسیله یک کد دوبیتی بصورت کد درمی آیند که هر عنصر متعلق بیکی از ترکیبهای بیتی؛ 00، 01، 10 و 11 است. مجموعه ای از هشت عنصر نیازمند یک کد سه بیتی است و یک مجموعه شانزده عنصری به کد چهاربیتی احتیاج دارد و بهمین ترتیب. اگر تعداد عناصر در مجموعه توانی صحیح از 2 نباشد، برخی از ترکیب بیت ها در کد دودویی به هیچ عنصری اختصاص نخواهد یافت. ارقام دهدهی

چنین مجموعه‌ای را تشکیل می‌دهند. یک کد دودویی که بین ده عنصر تمایز بوجود آورد باید حداقل چهار بیت داشته باشد، ولی شش ترکیب بدون استفاده خواهند ماند. کدهای متفاوت متعددی را با چهار بیت بصورت ده ترکیب مختلف و متمایز می‌توان بدست آورد. معمول‌ترین تخصیص بیت‌های بکار رفته برای ارقام دهدهی همان تخصیص دودویی سراسمی است که در ده وارده اول جدول ۳-۳ لیست شده است. این کد خاص، دهدهی کد شده با دودویی^۱ نامیده می‌شود و غالباً با BCD نشان داده می‌شود. سایر کدهای دهدهی گاهی اوقات بکار می‌روند و چند تا از آنها در بخش ۳-۵ آورده شده‌اند. دانستن اختلاف بین تبدیل اعداد دهدهی به دودویی و کد دودویی یک عدد دهدهی بسیار مهم است. مثلاً، وقتی که عدد دهدهی 99 به دودویی تبدیل شود حاصل 1100011 خواهد شد، ولی وقتی بصورت BCD درآید حاصل 1001 1001 خواهد بود. تنها اختلاف بین عدد دهدهی که بصورت سمبل‌های رقمی 0، 1، 2، ...، 9 نمایش داده شده است و سمبل‌های BCD 0001، 0010، در نمادهای مورد استفاده برای نمایش ارقام است و در واقع خود عدد دقیقاً یکسان است. چند عدد دهدهی و نمایش BCD آنها در جدول ۳-۳ آورده شده است.

نمایش الفبا عددی

بسیاری از کاربردهای کامپیوترهای دیجیتال مستلزم کار با داده‌هایی است که علاوه بر اعداد، حروف

جدول ۳-۳ اعداد دهدهی کد شده با دودویی (BCD)

عدد دهدهی	عدد دهدهی کد شده با دودویی (BCD)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	0001 0000
20	0010 0000
50	0101 0000
99	1001 1001
248	0010 0100 1000

1- Binary Coded Decimal

الفبا و کاراکترهای خاصی را نیز شامل می‌شود. یک مجموعه از کاراکترهای الفبا عددی متشکل از مجموعه 10 عنصر ارقام دهدهی، 26 حرف الفبا و تعدادی کاراکترهای خاص مانند \$، + و = است. چنین مجموعه‌ای بین 32 الی 64 عنصر (اگر فقط حروف بزرگ در نظر گرفته شود) و یا 64 تا 128 عنصر (اگر هر دو حروف بزرگ و کوچک بکار روند) خواهد داشت. در حالت اول، کد دودویی، شش بیت نیاز دارد و در حالت دوم هفت بیت مورد نیاز است. کد استاندارد الفبا عددی دودویی ASCII¹ است که از هفت بیت برای 128 کاراکتر استفاده می‌کند. کدهای دودویی برای حروف بزرگ، ارقام دهدهی، و چند کاراکتر خاص دیگر در جدول ۳-۴ لیست شده‌اند. توجه کنید که ارقام دهدهی در ASCII با حذف سه بیت مرتبه بالاتر 011، قابل تبدیل به BCD هستند. لیست کامل کاراکترهای ASCII در جدول ۱-۱۱ آورده شده است.

جدول ۳-۴ کد استاندارد آمریکایی برای تبادل اطلاعات (ASCII)

کاراکتر	کد دودویی	کاراکتر	کد دودویی
A	100 0001	0	011 0000
B	100 0010	1	011 0001
C	100 0011	2	011 0010
D	100 0100	3	011 0011
E	100 0101	4	011 0100
F	100 0110	5	011 0101
G	100 0111	6	011 0110
H	100 1000	7	011 0111
I	100 1001	8	011 1000
J	100 1010	9	011 1001
K	100 1011		
L	100 1100		
M	100 1101	space	010 0000
N	100 1110		010 1110
O	100 1111	(010 1000
P	101 0000	+	010 1011
Q	101 0001	\$	010 0100
R	101 0010	*	010 1010
S	101 0011)	010 1001
T	101 0100	-	010 1101
U	101 0101	/	010 1111
V	101 0110	,	010 1100
W	101 0111	=	011 1101
X	101 1000		
Y	101 1001		
Z	101 1010		

1- American Standard Code for Information Interchange

کدهای دودویی نقش مهمی را در عملیات کامپیوترهای دیجیتال ایفا می‌کنند. کدها باید به فرم دودویی باشند زیرا ثبات‌ها قادرند فقط اطلاعات دودویی را نگهداری نمایند. باید توجه داشت که کدهای دودویی صرفاً سمبل‌ها را تغییر می‌دهند و نه مفهوم عناصر گسسته‌ای را که نمایش می‌دهند. در عملیات کامپیوترهای دیجیتال باید مفهوم بیت‌های ذخیره شده در ثبات‌ها در نظر گرفته شود تا اعمال روی عملوندهای یکسان انجام شوند. در بررسی تصادفی بیت‌های یک ثبات کامپیوتری ملاحظه می‌شود که محتوای آن اغلب اطلاعات کد شده است و نه یک عدد دودویی.

کدهای دودویی برای هر مجموعه از عناصر گسسته مانند نت‌های موسیقی و مهره‌های شطرنج و مکان‌های آنها در روی صفحه شطرنج قابل تعریف‌اند. همچنین این کدها برای معرفی دستوراتی که اطلاعات کنترلی را در کامپیوترها مشخص می‌کنند بکار می‌روند. این فصل درباره نمایش داده‌ها بحث می‌کند. کد دستورات در فصل پنج بحث خواهد شد.

۲-۳ متمم‌ها

در کامپیوترهای دیجیتال از متمم‌ها برای ساده کردن عمل تفریق و عملیات منطقی استفاده می‌شود. در هر سیستم مبنای r دو نوع متمم وجود دارد: متمم r و متمم $r-1$. هرگاه به جای r مقدار آن در نام متمم قرار گیرد، برای اعداد دودویی متمم 2 و متمم 1 و برای اعداد دهدهی متمم 10 و متمم 9 را خواهیم داشت.

متمم $r-1$

اگر عدد N در پایه r بصورت n رقم موجود باشد، متمم $r-1$ عدد N بصورت $(r^n - 1) - N$ تعریف می‌شود. برای اعداد دهدهی $r=10$ و $r-1=9$ است، بنابراین متمم 9 عدد N برابر است با $(10^n - 1) - N$. عدد 10^n از یک 1 و n عدد 0 تشکیل شده است. مقدار $10^n - 1$ برابر خواهد بود با n عدد 9. مثلاً، با $n=4$ داریم $10^4 = 10000$ و $10^4 - 1 = 9999$. نتیجه می‌شود که متمم 9 یک عدد دهدهی با تفریق هر یک از ارقام آن از 9 بدست می‌آید. مثلاً متمم 9 عدد 546700 برابریست با $999999 - 546700 = 453299$ و متمم 9 عدد 12389 برابریست با $99999 - 12389 = 87610$.

برای اعداد دودویی، $r=2$ و $r-1=1$ است، بنابراین متمم 1 عدد N برابر $(2^n - 1) - N$ خواهد بود. مجدداً، 2^n بوسیله یک عدد دودویی نمایش داده می‌شود و متشکل است از یک 1 و n عدد 0 که بدنبال آن آمده است. $2^n - 1$ یک عدد دودویی است که بوسیله n عدد 1 نشان داده می‌شود. مثلاً برای $n=4$ داریم $2^4 = (10000)_2$ و $2^4 - 1 = (1111)_2$. بنابراین متمم 1 یک عدد دودویی از تفریق هر رقم از 1 بدست می‌آید. با این وجود تفریق هر رقم از 1 سبب می‌شود تا هر بیت از 0 به 1 و از 1 به 0 تبدیل شود. بنابراین متمم 1 یک عدد دودویی از تبدیل 1ها به 0ها و 0ها به 1ها بدست می‌آید. مثلاً متمم 1 عدد 1011001 برابر است با 0100110، و متمم 1 عدد 0001111 عبارتست از 1110000.

متمم $(r-1)$ اعداد هشت هشتی یا شانزده شانزدهی بترتیب از تفریق هر رقم از 7 یا F (15 دهمی) بدست می آید.

متمم r

متمم r یک عدد n رقمی N در پایه r با r برابرست با $r^n - N$ و با $N=0$ برابر 0 است. می بینیم که متمم r در مقایسه با متمم $r-1$ ، از جمع 1 با متمم $r-1$ حاصل می شود زیرا $r^n - N = [(r^n - 1) - N] + 1$ می باشد. بنابراین متمم 10 عدد دهمی 2389 برابر است با $7610 + 1 = 6711$ که از جمع 1 با مقدار متمم 9 بدست می آید. متمم 2 عدد دودویی 101100 برابرست با $010100 + 1 = 010011$ که از جمع 1 با مقدار متمم 1 بدست آمده است.

چون 10^n عددی است که با یک 1 و بدنبال آن با n عدد 0 نشان داده می شود، لذا $10^n - N$ که متمم 10 عدد N است، نیز می تواند بهمین صورت تعریف شود. بدین ترتیب که تمام 0های کم ارزشتر را دست نخورده باقی گذاشته و اولین رقم غیر 0 را از 10 کم می کنیم، و سپس تمام رقم های مرتبه بالاتر را از 9 کسر می نماییم. متمم 10 عدد 246700 برابر است با 753300 است و بدین ترتیب حاصل می شود که دو رقم 00 را باقی گذاشته، 7 را از 10 کم می کنیم، و سه رقم باقیمانده را از 9 کسر می کنیم. بطور مشابه، برای بدست آوردن متمم 2 می توان همه 0های سمت راست و اولین رقم 1 را دست نخورده باقی گذاشت، و سپس 0ها بجای 1 و 1ها بجای 0 در بقیه مکان های با ارزشتر گذاشته می شوند. متمم 2 عدد 1101100 برابر است با 0010100 و با باقی گذاشتن دو 0 مرتبه پائین تر و اولین 1، و سپس با جایگزینی 1ها با 0ها و 0ها با 1ها در چهاربیت مرتبه بالاتر بدست می آید.

در تعریف فوق فرض شد که اعداد دارای ممیز نیستند. اگر عدد اولیه N دارای نقطه ممیز باشد، باید بطور موقت حذف گردد تا بتوان متمم های r و $r-1$ را تشکیل داد. سپس نقطه ممیز در همان موقعیت نسبی اولیه اش وارد می شود. ذکر این نکته هم مفید است که متمم یک متمم همان مقدار اولیه را می دهد. متمم r عدد N برابر $r^n - N$ است. متمم این متمم $(r^n - (r^n - N)) = N$ خواهد بود که در واقع همان عدد اولیه می باشد.

تفریق اعداد بی علامت

در روش مستقیم تفریق که در دبستان تدریس شده از مفهوم "قرض کردن" استفاده شد. در این روش هرگاه رقمی از مفروق منه کوچکتر از رقم منناظر مفروق باشد یک واحد از عدد مرتبه بالاتر قرض می کنیم. بنظر می رسد که این ساده ترین کار بهنگام انجام تفریق با قلم و کاغذ است. وقتی که تفریق را با سخت افزار دیجیتال انجام دهیم، این روش در مقایسه با روش بکارگیری متمم از کارایی کمتری برخوردار است.

تفریق دو عدد n رقمی بی علامت $M-N$ ($N \neq 0$) در مبنای r بطریق زیر انجام می شود

۱- مفروق منه M را با متمم r مفروق N جمع می کنیم. نتیجه این عمل عبارتست از:

$$M + (r^n - N) = M - N + r^n$$

۲- اگر $M \geq N$ باشد، مجموع، یک رقم نقلی r^n تولید می کند که چشم پوشی می شود، و آنچه باقی می ماند $M-N$ است.

۳- اگر $M < N$ باشد، مجموع، رقم نقلی تولید نکرده و برابرست با $r^n - (N-M)$ ، که متمم r عدد $(M-N)$ است. برای بدست آوردن جواب بصورت قابل درک، متمم r مجموع را بدست آورده و در جلو آن یک علامت منفی می گذاریم

برای مثال، تفریق $72532 - 13250 = 59282$ را در نظر بگیرید. متمم 10 عدد 13250 برابرست با 86750. بنابراین:

$$\begin{array}{r} M = 72532 \\ N \text{ عدد } 10 \text{ متمم} = 86750 \\ \text{مجموع} = 159282 \\ \text{با حذف رقم نقلی } 10^5 = -100000 \\ \hline \text{جواب} = 59282 \end{array}$$

حال مثالی را با $M < N$ در نظر بگیرید. تفریق $13250 - 72532$ عدد منفی 59282 را تولید می کند. با استفاده از متمم ها داریم

$$\begin{array}{r} M = 13252 \\ N \text{ عدد } 10 \text{ متمم} = +27468 \\ \hline \text{جمع} = 40718 \end{array}$$

در این مثال رقم نقلی وجود ندارد
جواب نهایی 59282 است که برابر با متمم 10 عدد 40718 می باشد.

چون ما با اعداد بی علامت سروکار داریم، در واقع راهی برای بدست آوردن نتیجه بی علامت در مثال دوم وجود ندارد. بهنگام کار با قلم و کاغذ، تشخیص می دهیم که جواب باید به یک عدد علامت دار منفی تبدیل شود. در تفریق با روش متمم، جواب منفی از غیاب رقم نقلی نهایی و نتیجه متمم شده قابل تشخیص است.

تفریق با استفاده از روش متمم برای اعداد دودویی نیز مشابه روش فوق است. با بکارگیری دو عدد

$X=1010100$ و $Y=1000011$ ، تفریق $X-Y$ و $Y-X$ را بکمک متمم های 2 انجام می دهیم.

$$\begin{array}{r}
 X = \quad 1010100 \\
 Y \text{ عدد 2 متمم} = + \quad 0111101 \\
 \hline
 \text{مجموع} = \quad 10010001 \\
 \text{حذف رقم نقلی } 2^7 = - \quad 10000000 \\
 \hline
 X-Y: \text{ جواب} = \quad 0010001 \\
 \\
 Y = \quad 1000011 \\
 X \text{ عدد 2 متمم} = + \quad 0101100 \\
 \hline
 \text{مجموع} = \quad 1101111
 \end{array}$$

که در آن رقم نقلی وجود ندارد
جواب عدد منفی 0010001 است که متمم 2 عدد 1101111 می باشد

۳-۳ نمایش ممیز - ثابت

اعداد صحیح مثبت، از جمله صفر، بصورت اعداد بی علامت نمایش داده می شوند. معهذا برای نمایش اعداد صحیح منفی، ما علامتی را برای نمایش مقادیر منفی نیاز داریم. در حساب معمولی اعداد منفی با علامت منفی و اعداد مثبت با علامت مثبت مشخص می شوند. بعلت محدودیت های سخت افزاری، کامپیوترها باید هر چیزی، حتی علامت اعداد را، با 0ها و 1ها نمایش دهند. در نتیجه، مرسوم است که علامت را در انتهاالیه سمت چپ عدد قرار دهند. بنا به قرارداد بیت علامت را برای اعداد مثبت برابر 0 و برای اعداد منفی برابر 1 در نظر می گیرند.

یک عدد، علاوه بر علامت، ممکن است نقطه ممیز دودویی (با اعشاری) هم داشته باشد. مکان نقطه دودویی برای نمایش کسرها، اعداد صحیح، و اعداد مخلوط صحیح و کسری بکار می رود. نمایش نقطه ممیز دودویی در یک ثبات پیچیده است زیرا مکان آن باید در محلی در ثبات مشخص گردد. دو راه برای مشخص کردن محل ممیز دودویی در یک ثبات وجود دارد: یکی اختصاص یک محل ثابت به آن و دیگری استفاده از نمایش ممیز شناور می باشد. در روش ممیز ثابت فرض می شود که ممیز همیشه در یک مکان ثابت قرار گرفته است. دومکانی که بیشتر مورد استفاده اند عبارتند از: (۱) ممیز دودویی در انتهاالیه سمت چپ ثبات قرار گیرد و در نتیجه عدد ذخیره شده به یک کسر تبدیل شود؛ (۲) ممیز دودویی در انتهاالیه سمت راست ثبات واقع شود و در نتیجه عدد ذخیره شده یک عدد صحیح گردد. در هر حال، ممیز دودویی واقعاً وجود خارجی ندارد، بلکه با این فرض با عدد بصورت یک کسر یا یک عدد صحیح برخورد می شود. در نمایش ممیز شناور با استفاده از یک ثبات دیگر و ذخیره کردن عددی در آن محل

ممیز در ثبات اول مشخص می شود. درباره ممیز شناور در بخش بعدی بیشتر صحبت خواهیم کرد.

نمایش صحیح

وقتی که یک عدد دودویی صحیح مثبت باشد، علامت آن با 0 و اندازه آن با یک عدد دودویی مثبت نشان داده می شود. هرگاه عدد منفی باشد، علامت توسط 1 ولی بقیه عدد ممکن است به یکی از سه روش زیر نمایش داده شود.

۱- نمایش مقدار (اندازه) علامت دار¹

۲- نمایش متمم 1 علامت دار²

۳- نمایش متمم 2 علامت دار³

نمایش مقدار علامت دار برای یک عدد منفی از مقدار و یک علامت منفی تشکیل شده است. در دو نمایش دیگر، عدد منفی به یکی از دو صورت متمم 1 و متمم 2 مقدار مثبت عدد نمایش داده می شود. مثلاً، عدد علامت دار 14 را در یک ثبات در نظر بگیرید. 14 + بوسیله بیت علامت 0 که به دنبال آن مقدار معادل دودویی آن آمده است نشان داده می شود، 00001110: 14. توجه کنید که هر یک از هشت بیت ثبات باید مقداری داشته باشند و لذا 0هایی باید در مکان های پرارزشتری که بدنبال بیت علامت آمده اند قرار گیرند. هرچند تنها یک راه برای نمایش 14 + وجود دارد، ولی برای نمایش 14 - سه روش موجود است.

در نمایش مقدار علامت دار 10001110

در نمایش متمم 1 علامت دار 1110001

در نمایش متمم 2 علامت دار 11110010

نمایش مقدار علامت دار برای 14- فقط از متمم کردن بیت علامت بدست می آید. نمایش متمم 1 علامت دار 14- با متمم کردن تمام بیت های 14+ از جمله بیت علامت، فراهم می شود. نمایش متمم 2 علامت دار با بدست آوردن متمم 2 عدد مثبت، از جمله بیت علامت حاصل می شود.

سیستم مقدار علامت دار در محاسبات معمولی بکار می رود، اما اگر در محاسبات کامپیوتری بکار رود کارها کند و پیچیده خواهد شد. بنابراین معمولاً متمم علامت دار مورد استفاده قرار می گیرد. از متمم 1 به این علت که در آن دو نحوه نمایش 0 وجود دارد (0+ , 0-)، به ندرت، بجز در عملیات حسابی در برخی کامپیوترهای قدیمی استفاده می شود. متمم 1 در عملیات منطقی بکار گرفته می شود چون تغییر 1 به 0 یا 0 به 1 معادل با عمل متمم سازی منطقی است. در بحث زیر که درباره حساب دودویی علامت دار خواهیم کرد برای عددهای منفی منحصراً از نمایش متمم 2 علامت دار استفاده خواهد شد.

1- signed – magnitude

2- signed –1’s complement

3- Signed –2’s Comploment

جمع حسابی

جمع دو عدد در سیستم مقدار (اندازه) علامت دار از قواعد حساب معمولی پیروی می کند. اگر علامت ها یکسان باشند، دو عدد را جمع می کنیم و علامت مشترک آنها را به مجموع می دهیم. اگر علامت ها مختلف باشند، مقدار کوچکتر را از بزرگتر کم می کنیم و علامت عددی که مقدار آن بزرگتر باشد را به حاصل می دهیم. مثلاً، $-12 = -(37-25) = (-37) + (25)$ که با تفریق عددی که اندازه کوچکتر دارد، یعنی 25، از عدد بزرگتر 37 و بکاربردن علامت 37 برای نتیجه حاصل انجام شده است. این روند مستلزم مقایسه علامت ها و مقادیر و سپس انجام جمع یا تفریق است. (روش جمع اعداد دودویی در نمایش مقدار علامت دار در بخش ۲-۱۰ بحث شده است) در مقابل، قاعده جمع کردن اعداد در سیستم متمم 2 نیازی به مقایسه و تفریق ندارد. رویه بسیاری ساده است و به طریق زیر می توان آن را بیان کرد. دو عدد را همراه با بیت های علامت آنها جمع کنید، و در صورت بوجود آمدن رقم نقلی از محل بیت علامت (منتهاالبه سمت چپ)، از آن صرف نظر کنید. مثال های عددی در زیر نشان داده شده اند. توجه کنید که اعداد منفی باید ابتدا بصورت متمم 2 در آیند و اگر حاصل جمع بدست آمده منفی باشد بشکل متمم 2 خواهد بود.

+ 6	00000110		- 6	11111010
+ 13	00001101		+ 13	00001101
+ 19	00010011		+ 7	00000111
+ 6	00000110		- 6	11111010
- 13	11110011		- 13	11110011
- 7	11111001		- 19	11101101

در هر یک از چهار حالت، عمل انجام شده، حتی بر روی بیت علامت، جمع است. هر رقم نقلی خارج شده از بیت علامت نادیده گرفته می شود و نتایج منفی خود بخود به فرم متمم 2 خواهند بود. نمایش اعداد منفی به فرم متمم برای کسانی که با سیستم مقدار علامت دار کار کرده اند ناآشناست. برای تعیین مقدار یک عدد منفی وقتی که بشکل متمم 2 علامت دار باشد، لازم است به عدد مثبتی تبدیل شود تا به فرم آشناتری درآید. مثلاً عدد دودویی علامت دار 11111001 منفی است زیرا سمت چپ ترین بیت 1 است. متمم 2 آن 00000111 می باشد که معادل عدد دودویی +7 است. بنابراین عدد منفی اصلی -7 بوده است.

تفریق حسابی

تفریق دو عدد دودویی وقتی که اعداد منفی به شکل متمم 2 هستند بسیار ساده است و می تواند به

شکل زیر بیان شود: متمم 2 مفروق را بگیرد (با در نظر گرفتن بیت علامت) و آنرا با مفروق منه جمع کنید (به همراه بیت علامت). رقم نقلی خروجی از مکان بیت علامت را چشم پوشی نمائید. این رویه از این واقعیت نتیجه شده است که یک عمل تفریق می تواند به یک جمع بدل شود بشرطی که علامت مفروق عوض شود. این مطلب با رابطه زیر نشان داده شده است.

$$(\pm A) - (+B) = (\pm A) + (-B)$$

$$(\pm A) - (-B) = (\pm A) + (+B)$$

اما تغییر یک عدد مثبت به منفی بسادگی با بدست آوردن متمم 2 امکان پذیر است. عکس این موضوع نیز صادق است زیرا متمم یک عدد منفی یک عدد معادل مثبت را بدست می دهد. تفریق $7 = + - (13) - (-6)$ را در نظر بگیرید. این تفریق بصورت هشت بیت دودویی بفرم $111110011 - 111110110$ نوشته می شود. با گرفتن متمم 2 عدد (-13) که $(+13)$ را می دهد تفریق به جمع تبدیل می شود و در نتیجه داریم $111110110 + 000011101 = 100000111$. با حذف رقم نقلی انتهایی، پاسخ صحیح 00000111 یعنی $(+7)$ بدست می آید.

اشاره به این نکته لازم به نظر می رسد که بگوئیم جمع و تفریق اعداد متمم 2 علامت دار با همان قواعد جمع و تفریق اعداد بی علامت صورت می گیرد. بنابراین، در کامپیوتر یک مدار سخت افزاری مشترک برای انجام هر دو عمل حسابی مورد نیاز است. کاربر یا برنامه نویس باید نتایج چنین جمع یا تفریق را، بسته به اینکه اعداد علامت دار فرض شوند، یا بی علامت بگونه ای متفاوت تفسیر کند.

سرریز¹

وقتی دو عدد n رقمی جمع شده و حاصل جمع $n+1$ رقم را اشغال کند، گوئیم سرریز رخ داده است. وقتی جمع را با کاغذ و قلم انجام دهیم، سرریز مسئله ای را ایجاد نمی کند زیرا محدودیتی در عرض صفحه برای نوشتن حاصل جمع وجود ندارد. سرریز در کامپیوترهای دیجیتال ایجاد اشکال می کند زیرا عرض ثبات ها محدودیت دارد. نتیجه ای که $n+1$ بیتی باشد را نمی توان در ثباتی که دارای عرض استاندارد n بیت است جای داد. با این دلیل اغلب کامپیوترها رخداد یک سرریز را ردیابی می کنند، و وقتی که اتفاق بیفتد، فلیپ فلاپ بخصوصی 1 می شود که کاربر می تواند در هر زمان آن را چک کند. درک یک سرریز پس از جمع دو عدد دودویی بستگی به علامت دار بودن یا نبودن آنها دارد. اگر دو عدد بدون علامت با هم جمع شوند، سرریز هنگامی آشکار می شود که رقم نقلی از با ارزش ترین مکان خارج شود. در حالت علامت دار بودن عدد، سمت چپ ترین بیت همیشه علامت را نمایش می دهد و اعداد منفی به فرم متمم 2 خواهند بود. وقتی دو عدد علامت دار جمع شوند، بیت علامت بعنوان

1- Overflow

بخشی از عدد تلقی می شود و رقم نقلی مشخص کننده یک سرریز نیست. بنابراین از جمع دو عدد که یکی مثبت و دیگری منفی باشد سرریز رخ نمی دهد، زیرا جمع یک عدد مثبت با یک عدد منفی نتیجه ای را تولید می کند که از بزرگترین آنها کوچکتر است. سرریز هنگامی روی می دهد که دو عددی که با هم جمع می شوند هر دو مثبت و یا هر دو منفی باشند. برای اینکه ببینیم چگونه این اتفاق ممکن است رخ بدهد مثال زیر را در نظر بگیرید. دو عدد علامت دار دودویی $+70$ و $+80$ در دو ثبات 8 بیت ذخیره شده اند دامنه اعدادی که هر ثبات می تواند در خود جای دهد از عدد دودویی $+127$ تا -128 است. چون مجموع $+150$ است، از ظرفیت ثبات هشت بیتی تجاوز می کند. این وضعیت هنگامی رخ می دهد که اعداد هر دو مثبت یا منفی باشند. دو جمع دودویی در زیر همراه با دو رقم نقلی آخر نشان داده شده اند.

ارقام نقلی: 01	ارقام نقلی: 10
$+70$ 0 1000110	-70 1 0111010
$+80$ 0 1010000	-80 1 0110000
$+150$ 1 0010110	-150 0 1101010

توجه کنید که نتیجه هشت بیتی که می باید مثبت باشد، دارای بیت علامت منفی است و نتیجه هشت بیتی که باید منفی می بود دارای بیت علامت مثبت است. با این وجود اگر، رقم نقلی خارج شده از محل بیت علامت بعنوان بیت علامت در نظر گرفته شده است، جواب 9 بیتی حاصل صحیح خواهد بود. چون جواب نمی تواند در هشت بیت جای گیرد گوئیم سرریز رخ داده است: حالت سرریز را با مشاهده رقم نقلی وارده به بیت علامت و نقلی خارج شدن از بیت علامت می توان درک کرد. اگر این دو نقلی با هم مساوی نباشد، یک وضعیت سرریز تولید شده است. این مطلب در مثال های فوق که هر دو رقم نقلی نشان داده شده اند دیده می شود. چنانچه دو رقم نقلی به یک گیت OR انحصاری اعمال شوند، سرریز هنگامی رخ داده است که خروجی این گیت 1 شود.

نمایش ممیز - ثابت ددهی

نمایش اعداد ددهی در ثبات ها تابعی از کد دودویی مورد استفاده برای نمایش رقم ددهی است. یک کد ددهی چهاربیتی نیاز به چهار فلیپ فلاپ برای هر رقم ددهی دارد. نمایش 4385 در BCD به 16 فلیپ فلاپ نیاز دارد. این عدد در یک ثبات با 16 فلیپ فلاپ نمایش داده می شود.

0100 0011 1000 0101

با نمایش اعداد به شکل ددهی مقدار قابل توجهی از فضای ذخیره سازی تلف می شود زیرا تعداد بیت های لازم برای ذخیره سازی یک عدد ددهی در کد دودویی بزرگتر از تعداد بیت های لازم در نمایش معادل دودویی آن است. همچنین، مدارهای لازم برای انجام محاسبات ددهی پیچیده تر هستند. با این

وجود، استفاده از نمایش دهدهی دارای مزایایی است زیرا داده‌های ورودی و خروجی کامپیوتر بوسیله انسانها که سیستم دهدهی را مورد استفاده قرار می‌دهند تولید می‌شود. در برخی از کاربردها، مانند داده‌پردازی تجاری، حجم محاسبات ریاضی لازم در مقایسه با مقدار لازم برای ورودی و خروجی داده‌های دهدهی کم است. به همین دلیل برخی از کامپیوتر و ماشین حساب‌ها اعمال حسابی را مستقیماً با داده‌های دهدهی (بصورت یک کد دودویی) انجام می‌دهند، و بنابراین در آنها نیازی به تبدیل به دودویی و تبدیل مجدد به دهدهی وجود ندارد. بعضی از سیستم‌های کامپیوتری دارای بخش سخت افزاری برای هر دو نوع محاسبه حسابی بر روی داده‌های دودویی و دهدهی می‌باشند.

نمایش اعداد دهدهی علامت دار در BCD مشابه با نمایش اعداد علامت دار دودویی است. ما می‌توانیم سیستم آشنای مقدار علامت دار یا سیستم متمم علامت دار را بکار ببریم. علامت یک عدد دهدهی معمولاً با چهاربیت نشان داده می‌شود تا با کد 4 بیتی ارقام دهدهی مطابقت داشته باشد. معمول است که علامت مثبت را با چهار 0 و علامت منفی را با معادل BCD عدد 9 که 1001 است نشان دهند. بکارگیری سیستم اعداد مقدار علامت دار در کامپیوترها مشکل است. سیستم متمم علامت دار هم در فرم متمم 9 و یا متمم 10 می‌باشد. ولی سیستم متمم 10 علامت دار بیشتر استفاده می‌شود. برای بدست آوردن متمم 10 یک عدد BCD، ابتدا متمم 9 را بدست آورده و سپس به کم ارزش‌ترین بیت یک واحد اضافه می‌کنیم. متمم 9 نیز از تفریق هر رقم از 9 حاصل می‌گردد.

روشهای بدست آمده برای سیستم متمم 2 علامت دار در مورد سیستم متمم 10 علامت دار برای اعداد دهدهی نیز صادق است. عمل جمع با اضافه کردن تمام ارقام، از جمله رقم علامت، و چشم پوشی از رقم نقلی نهایی انجام می‌شود. البته فرض بر این است که همه اعداد منفی به شکل متمم 10 هستند. به جمع $+135 = (-240) + (+375)$ که در سیستم متمم 10 علامت دار انجام شده توجه کنید.

$$\begin{array}{r} 0\ 375\ (0000\ 0011\ 0111\ 0101)_{BCD} \\ +9\ 760\ (1001\ 0111\ 0110\ 0000)_{BCD} \\ \hline 0\ 135\ (0000\ 0001\ 0011\ 0101)_{BCD} \end{array}$$

عدد 9 در سمت چپ‌ترین مکان عدد دوم نشان دهنده منفی بودن آنست. عدد 9760 متمم 10 عدد 0240 می‌باشد. دو عدد با یکدیگر جمع شده و رقم نقلی نیز چشم پوشی شده است تا $+135$ حاصل گردد. البته اعداد دهدهی، از جمله ارقام علامت، در داخل کامپیوتر باید بشکل BCD باشند. عمل جمع با جمع‌کننده‌های BCD صورت می‌گیرد (شکل ۱۸-۱۰ را ملاحظه کنید).

تفریق اعداد دهدهی بدون علامت یا سیستم متمم 10 علامت دار همانند حالت دودویی است. متمم 10 مفروق را با مفروق منه جمع می‌کنیم. بسیاری از کامپیوترها سخت افزار خاصی را برای انجام مستقیم محاسبات حسابی با اعداد دهدهی دارند. کاربر کامپیوتر می‌تواند با کمک دستورالعمل‌هایی در برنامه مشخص کند که اعمال حسابی مستقیماً با عددهای دهدهی و بدون تبدیل آنها به دودویی انجام شود.

۳-۴ نمایش ممیز شناور^۱

نمایش ممیز شناور یک عدد دارای دو بخش است. بخش اول یک عدد ممیز ثابت علامت دار است که مانتیس^۲ خوانده می شود. بخش دوم محل ممیز دهدهی (یا دودویی) را معین می کند و نما^۳ نام دارد. مانتیس ممیز ثابت، ممکن است یک عدد کسری و یا صحیح باشد. مثلاً، عدد دهدهی 6132.784 به شکل ممیز شناور و بصورت یک کسر و یک نما نشان داده می شود.

کسر	نما
+0.6132789	+04

مقدار نما مشخص می کند که مکان واقعی ممیز چهار واحد به سمت راست ممیز نشان داده شده در کسر است. این نمایش معادل با نماد علمی $0.6132789 \times 10^{+4}$ است. ممیز شناور همواره برای تفسیر اعدادی به شکل کلی زیر بکار می رود.

$$m \times r^e$$

در ثبات ها فقط مانتیس m و نمای e بطور فیزیکی نمایش داده می شوند (از جمله علامت آنها). پایه r و محل ممیز در مانتیس همیشه بطور ضمنی مساوی با مقدار معینی فرض می شوند. مدارهایی که بر روی اعداد ممیز شناور در ثباتها عمل می کنند براساس این فرضیات برای ایجاد محاسبات صحیح، طراحی می شوند.

یک عدد ممیز شناور دودویی نیز به نحوی مشابه نشان داده می شود بجز اینکه از پایه 2 برای نما استفاده می شود. مثلاً عدد دودویی 1001.11 با بخش کسری 8 بیتی و نمای 6 بیتی بشکل زیر نمایش داده می شود.

کسر	نما
01001110	000100

بخش کسری دارای یک 0 در انتها الیه سمت چپ است که مثبت بودن آن را مشخص می کند. نقطه ممیز دودویی بدنبال بیت علامت آمده است ولی در ثبات نشان داده نمی شود. نما دارای معادل دودویی عدد $+4$ است. عدد ممیز شناور معادل است با

$$m \times 2^e = + (1001110)_2 \times 2^{+4}$$

یک عدد ممیز شناور، در صورتی که پرارزترین رقم مانتیس غیر صفر باشد، نرمالیزه^۵ (یا بهنجار)

-
- | | | |
|-------------------|---------------|-------------|
| 1- Floating point | 2- Mantissa | 3- Exponent |
| 4- Radix | 5- Normalized | |

خوانده می شود. مثلاً، عدد دهدهی 350 نرمالیزه است ولی 00035 نرمالیزه نیست. صرف نظر از اینکه محل ممیز در کجای مانتیس فرض شود، عدد هنگامی نرمالیزه است که رقم متناهی سمت چپ آن غیر صفر باشد. مثلاً عدد دودویی 00011010 به دلیل داشتن سه 0 در ابتدای آن نرمالیزه نیست. این عدد با سه بار شیفت به چپ و چشم پوشی از 0های ابتدای آن نرمالیزه می شود و عدد حاصل 11010000 خواهد بود. سه بار عمل شیفت عدد را در $2^3=8$ ضرب می کند. برای حفظ مقدار عدد ممیز شناور، باید سه واحد از نما کسر شود. فرم نرمالیزه بیشترین دقت را برای اعداد ممیز شناور فراهم می کند. صفر را نمی توان نرمالیزه کرد زیرا دارای رقم غیر صفر نیست. این عدد در نمایش ممیز شناور با مانتیس و نمای تماماً 0 نشان داده می شود.

عملیات حسابی با اعداد ممیز شناور پیچیده تر از اعمال حسابی با اعداد ممیز ثابت است و اجرای آنها طولانی تر است و بعلاوه به سخت افزار پیچیده تری نیاز دارد. با این وجود، نمایش ممیز شناور اغلب برای محاسبات علمی بکار می رود زیرا مسائل و مشکلاتی از نظر ضرب و تقسیم در نمایش ممیز ثابت وجود دارد. بسیاری از کامپیوترها و تمام ماشین حساب های الکترونیکی قابلیت درونی انجام عملیات محاسباتی ممیز شناور را دارا هستند. کامپیوترهایی که فاقد سخت افزار لازم برای محاسبات ممیز شناور می باشند مجموعه ای از دستورالعمل ها را برای کاربر در برنامه نویسی مسائل علمی، با اعداد ممیز شناور در اختیار می گذارند. عملیات حسابی با اعداد ممیز شناور در بخش ۵-۱۰ بحث شده اند.

۳-۵ انواع دیگر کدهای دودویی

در بخش های قبلی متداول ترین نوع داده های کد شده دودویی در کامپیوترهای دیجیتال را معرفی کردیم. گاهی اوقات از کدهای دیگری هم برای اعداد دهدهی و کاراکترهای الفبا عددی استفاده می شود. گاهی هم کامپیوترهای دیجیتال از کدهای دودویی دیگری برای کاربردهای خاص استفاده می کنند. در این بخش چند کد دودویی، که در کامپیوترهای دیجیتال با آنها مواجه می شویم ارائه می شود.

کدگری^۱

سیستم های دیجیتال فقط قادرند داده ها را بشکل گسسته پردازش کنند. بسیاری از سیستم های فیزیکی داده های خروجی پیوسته ای را در اختیار می گذارند. این داده ها باید، قبل از بکار رفتن بوسیله کامپیوترهای دیجیتال، به فرم دیجیتال تبدیل شوند. اطلاعات پیوسته یا آنالوگ^۲ بکمک مبدل های آنالوگ به دیجیتال به کمیت های دیجیتال بدل می شوند. از کد انعکاسی^۳ یا کدگری که در جدول ۳-۵ نشان داده شده است گاهی اوقات برای داده های دیجیتال تبدیل شده استفاده می شود. مزیت کدگری بر

1- Gray Code

2- Analog to Digital Convertor

3- Reflected Code

اعداد دودویی مستقیم این است که کدگری با رفتن از هر عدد به عدد بعدی فقط در یک بیت تغییر می‌کند. به عبارت دیگر، تغییر از هر عدد به عدد بعدی فقط مستلزم تغییر یک بیت از 0 به 1 یا از 1 به 0 است. یکی از کاربردهای کدگری هنگامی است که داده‌های آنالوگ نشان دهنده تغییر مکان پیوسته یک محور دوار^۱ باشد. محور به قسمت‌هایی تقسیم بندی می‌شود و به هر قسمت عددی تخصیص می‌یابد. اگر قطعات مجاور، متناظر با عددهای مجاور در کدگری باشند، ابهام در موقعیت شفت هنگامی که در وضعیت بین دو نقطه قرار گیرد کمتر خواهد شد.

گاهی اوقات از شمارنده‌های کدگری برای تهیه رشته‌های زمانبندی^۲ (تنظیم زمان) در کنترل عملیات یک سیستم دیجیتال استفاده می‌شود. شمارنده کدگری شمارنده‌ای است که فلیپ فلاپ‌هایش رشته حالات جدول ۳-۵ را اختیار می‌کنند. این شمارنده‌ها در تغییر از یک حالت به حالت دیگر ابهامی ندارند زیرا در حالت گذار فقط یک بیت آن تغییر می‌کند

کدهای دهمی دیگر

کدهای دودویی برای ارقام دهمی حداقل به چهاربیت نیاز دارند. کدهای متعدد مختلفی را با کمک چهاربیت می‌توان بصورت 10 ترکیب مختلف یا بیشتر درآورد. چند نمونه از این کدها در جدول ۳-۶ نشان داده شده است.

BCD را قبلاً معرفی کردیم. در این کد از تخصیص مستقیم معادل دودویی به رقم استفاده شده است. هنگام استفاده از BCD، شش ترکیب بدون استفاده بیت‌ها که در جدول نشان داده شده‌اند مفهومی ندارند، درست مانند حرف H که هنگام استفاده از سمبل‌های ارقام دهمی مفهوم ندارد. مثلاً بیان اینکه 1001 1110 یک عدد دهمی BCD است مانند این است که بگوئیم 9H یک عدد دهمی است که با سمبل‌های مرسوم نوشته شده باشد. در هر دوی اینها یک سمبل غیرمعتبر وجود دارد و بنابراین یک

جدول ۳-۵ کدگری 4 بیتی

Binary code	Decimal equivalent	Binary code	Decimal equivalent
0000	0	1100	8
0001	1	1101	9
0011	2	1111	10
0010	3	1110	11
0110	4	1010	12
0111	5	1011	13
0101	6	1001	14
0100	7	1000	15

1- Shaft

2- Timing Sequence

جدول ۳-۶ چهار کد دودویی مختلف برای ارقام دهدهی

رقم دهدهی	BCD 8421	2421	افزونی 3	افزونی 3 مغری
0	0000	0000	0011	0010
1	0001	0001	0100	0110
2	0010	0010	0101	0111
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1100
6	0110	1100	1001	1101
7	0111	1101	1010	1111
8	1000	1110	1011	1110
9	1001	1111	1100	1010
ترکیب های	1010	0101	0000	0000
بلا استفاده	1011	0110	0001	0001
بیت ها	1100	0111	0010	0011
	1101	1000	1101	1000
	1110	1001	1110	1001
	1111	1010	1111	1011

عدد بی معنی را نشان می دهند.

یکی از معایب استفاده از BCD مشکلات ناشی از محاسبه متمم 9 عدد است. از طرف دیگر، متمم 9 در کدهای 2421 و افزونی 3¹ که در جدول ۳-۶ نشان داده شده اند، به آسانی قابل محاسبه است. این دو کد دارای خاصیت خود متممی^۲ هستند. بدان معنی که متمم 9 عدد دهدهی هنگامی که یکی از این دو نوع تبدیل شود ب راحتی با تعویض 1 ها به 0 و 0 ها به 1 بدست می آید. این خاصیت هنگام انجام عملیات حسابی در نمایش متمم علامت دار مفیدند.

2421 مثالی از کدهای وزن دار^۳ است. در یک کد وزن دار بیتها در وزنه های مشخص شده ضرب می شوند، و مجموع بیت های وزن دار شده رقم دهدهی را می دهد. مثلاً ترکیب بیتی 1101 وقتی که در رقم های مربوطه در 2421 ضرب شوند، معادل دهدهی این ترکیب یعنی $2 \times 1 + 4 \times 1 + 2 \times 0 + 1 \times 1 = 7$ بدست می آید. به کد BCD می توان وزنه های 8421 را اختصاص داد و بهمین دلیل کد 8421 خوانده می شود

کد افزونی 3 یک کد دهدهی است که در کامپیوترهای قدیمی بکار می رفته است. این یک کد غیر وزندار است. کد دودویی اختصاص یافته این کد از عدد دودویی BCD معادل آن پس از جمع با عدد 3 (0011) بدست می آید.

1- Excess-3

2- Self Complementing

3- Weighted Code

با توجه به جدول ۵-۳ می بینیم که کدگری برای 10 وارده اول جدول، کد دهمی مناسبی نیست. این بدان علت است که انتقال از 9 به 0 مستلزم تغییر سه بیت می باشد (از 1101 به 0000). برای غلبه بر این مشکل، ما 10 عدد را، که از سومین وارده 0010 شروع و به 1010 ختم می شود، انتخاب می کنیم. حال گذر از 1010 به 0010 تنها تغییر یک بیت را موجب می شود. چون کد به میزان سه عدد به بالا شیفت داده شده است کدگری افزونی 3 خواننده می شود. این کد به همراه سایر کدهای دهمی در جدول ۶-۳ آورده شده است.

کدهای الفبا عددی دیگر

کد ASCII (جدول ۴-۳) کد استاندارد است که معمولاً برای ارسال اطلاعات دودویی بکار می رود. هر مشخصه بوسیله یک کد هفت بیتی نمایش داده می شود و معمولاً بیت هشتمی هم برای توازن به آن اضافه می گردد. (بخش ۶-۳ ملاحظه شود). کد دارای 128 کاراکتر می باشد. نود و پنج کاراکتر سمبل های گرافیکی^۱ را نمایش می دهند که شامل حروف بزرگ و کوچک، ارقام 0 تا 9، علائم نقطه گذاری و سمبل های خاص هستند. بیست و سه کاراکتر نماینده شکل دهنده ها^۲ می باشند که شامل کاراکترهای عملیاتی برای کنترل دستگاه های چاپ و یا نمایشگرها از قبیل برگشت نورد^۳، تعویض سطر^۴، جدول بندی افقی^۵، و پستبر^۶ می باشند. 10 کاراکتر دیگر برای هدایت جریان انتقال داده ها به کار می روند و وضعیت آن را گزارش می دهد.

کد الفباء عددی دیگر که در دستگاه های IBM بکار می رود EBCDIC^۷ می باشد. در این کد از هشت بیت برای هر کاراکتر استفاده می شود (بیت نهم متعلق به توازن است). کاراکترهای EBCDIC همان کاراکترهای ASCII می باشند ولی بیت های اختصاص یافته به آنها متفاوت است.

هرگاه کاراکترهای الفباء عددی برای پردازش داخلی در کامپیوتر بکار روند (ونه به منظور ارسال داده ها) بهتر است از یک کد 6 بیتی برای نمایش 64 کاراکتر استفاده شود. با یک کد 6 بیتی می توان 26 حرف بزرگ، ارقام 0 تا 9، و تا 28 کاراکتر خاص را مشخص کرد. این مجموعه از کاراکترها معمولاً برای اهداف پردازش داده کافی است. بکار گرفتن بیت های کمتر برای کد کاراکترها این مزیت را دارد که فضای حافظه لازم برای ذخیره سازی کمیت های داده های الفبایی کاهش می یابد.

۶-۳ کدهای آشکارسازی خطا

اطلاعات دودویی انتقالی از طریق برخی از وسایل ارتباطی در معرض پارازیت های^۸ خارجی قرار دارند که ممکن است بیت ها را از 0 به 1 و بالعکس تبدیل کند. کد کشف خطا یک کد دودویی است که

1- Graphic	2- Format Effectors	3- Carriage Return
4- Line Feed	5- Horizontal Tabulation	6- Back Space
7- Extended BCD Interchange Code		8- Noise

خطاهای رقمی را در طول ارسال کشف می‌کند. خطاهای درک شده را نمی‌توان اصلاح کرد و فقط وجود آنها معلوم می‌شود. معمول این است که تواتر خطاها مشاهده شود. اگر خطاها غیرمتوالی و تصادفی رخ دهند، اطلاعات خاصی که خطا دارد مجدداً ارسال می‌گردد. اگر خطا بیش حد معقول رخ دهد سیستم از نظر وجود اشکال چک می‌شود.

متداول‌ترین کد کشف خطا، بیت توازن^۱ است. بیت توازن بیتی است اضافی که به یک پیام دودویی اضافه می‌شود تا تعداد 1ها را در آن زوج یا فرد نماید. یک پیام سه بیتی و دونوع بیت توازن ممکن در جدول ۳-۷ نشان داده شده است. بیت فرد P چنان اختیار می‌شود تا مجموع تمام 1ها را فرد نماید (در تمام چهاربیت). بیت زوج P چنان اختیار می‌شود تا مجموع تمام 1ها را زوج نماید. در هر صورت مجموع 1ها در پیام ارسالی و بیت P در نظر گرفته می‌شود. در هر کاربرد خاص یکی از انواع توازن بکار می‌رود. روش توازن زوج این اشکال را دارد که یکی از ترکیبات بیت‌ها تماماً 0 خواهد داشت، در حالی که در توازن فرد همیشه یکی از بیت‌ها (از چهاربیتی که پیام و P را تشکیل می‌دهند) 1 خواهد بود. دقت کنید که P فرد متمم P زوج است.

هنگام انتقال اطلاعات از یک مکان به مکانی دیگر، با بیت توازن باین صورت برخورد می‌شود که در سمت فرستنده، پیام به یک مولد توازن^۲ اعمال می‌شود تا بیت توازن لازم توسط آن تولید شود. پیام، همراه با بیت توازن به مقصد ارسال می‌گردد. در سمت گیرنده، تمام بیت‌های رسیده (دراین حالت چهاربیت) به یک چک‌کننده توازن^۳ اعمال می‌شوند تا توازن انتخابی را چک کنند (فرد یا زوج). اگر توازن بیت‌ها با توازن انتخابی مغایرت داشته باشد، خطا اعلام می‌شود. روش توازن می‌تواند وجود یک، سه یا هر تعداد فردی از خطا را کشف کند. اگر تعداد خطاها زوج باشند کشف نمی‌شوند. شبکه‌های مولد و چک‌کننده توازن مدارهایی منطقی هستند که با توابع OR انحصاری ساخته می‌شوند. دلیل این امر این است که، همانطور که در بخش ۲-۱ ذکر شد، توابع OR انحصاری برای سه یا

جدول ۳-۷ تولید بیت توازن

پیام xyz	فرد (P)	زوج (P)
000	1	0
001	0	1
010	0	1
011	1	0
100	0	1
101	1	0
110	1	0
111	0	1

1- Parity Bit

2- Parity Generator

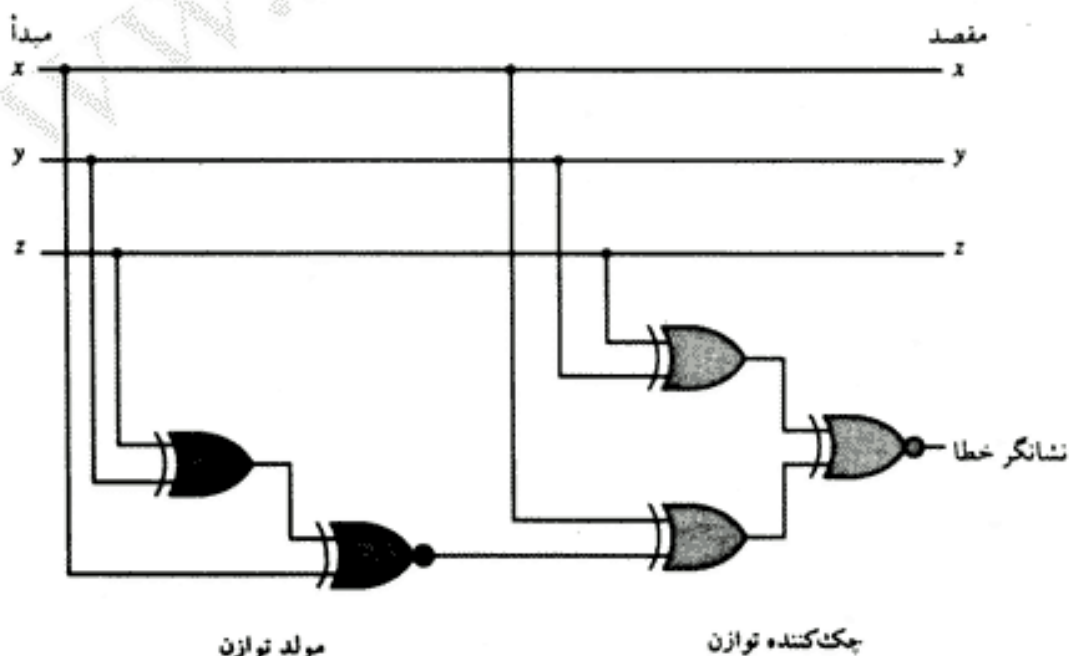
3- Parity Checker

چند متغیر بنا به تعریف یک تابع فرد است. تابع فرد تابعی منطقی است که مقدار آن 1 دودویی است، اگر و فقط اگر تعداد فردی از متغیرها 1 باشند. طبق این تعریف، تابع زوج P عبارتست از OR انحصاری x, y, z ، زیرا هر گاه یکی از این سه متغیر 1 یا هر سه آنها 1 باشند، تابع برابر 1 خواهد بود (جدول ۳-۷). تابع فرد متمم تابع زوج است.

بعنوان یک مثال، یک پیام سه بیتی را که قرار است با بیت توازن فرد ارسال شود در نظر بگیرید. در سمت فرستنده، بیت توازن فرد بوسیله یک مولد توازن تولید می شود. این مدار، همانطور که در شکل ۳-۳ دیده می شود، از یک OR انحصاری و یک گیت NOR انحصاری تشکیل شده است. چون P زوج، OR انحصاری x, y, z و P فرد متمم P زوج است، برای متمم لازم باید یک گیت NOR انحصاری بکار گرفته شود. پیام و بیت توازن فرد به مقصدشان ارسال و به یک چک کننده توازن اعمال می شوند. در حین ارسال پیام اگر توازن چهاربیت رسیده زوج باشد خطایی رخ داده است، زیرا اطلاعات ارسالی ابتدا فرد بوده است. اگر خطایی رخ داده باشد خروجی چک کننده توازن 1 خواهد شد، یعنی تعداد 1 های چهار ورودی زوج است. چون تابع OR انحصاری چهار ورودی یک تابع فرد است، ما می باید مجدداً با استفاده از یک گیت NOR انحصاری، خروجی را متمم کنیم.

ذکر این نکته مفید است که بدانیم مولد توازن می تواند همان مدار چک کننده توازن را داشته باشد بشرطی که ورودی چهارم دائماً در 0 منطقی نگهداشته شود. مزیت این خاصیت این است که برای تولید و چک کردن توازن می توان از یک مدار استفاده کرد.

با توجه به مثال فوق واضح است که مولدها و چک کننده های توازن با توابع OR انحصاری پیاده سازی می شوند. مدارات توازن فرد برای متمم کردن خروجی تابع به یک NOR انحصاری نیاز دارند.



شکل ۳-۳ کشف خطا بوسیله بیت توازن فرد

مسائل

- ۳-۱ اعداد دودویی زیر را به دهدهی تبدیل کنید: 110110100؛ 1110101؛ 101110
- ۳-۲ اعداد زیر، که در آنها پایه‌هایشان مشخص شده است را به دهدهی تبدیل کنید: $3(12121)$ ؛ $5(4310)$ ، $7(50)$ ؛ $12(198)$.
- ۳-۳ اعداد دهدهی زیر را به دودویی تبدیل کنید: 1231؛ 673 و 1998.
- ۳-۴ اعداد دهدهی زیر را به مبناهای مشخص شده تبدیل کنید
الف) 7562 به هشت هشتی (ب) 1938 به شانزده شانزدهی (ج) 175 به دودویی
- ۳-۵ عدد شانزده شانزدهی F3A7C2 را به دودویی و هشت هشتی تبدیل نمایید.
- ۳-۶ اگر جواب معادله $x^2 - 10x + 31 = 0$ برابر $x=5$ و $x=8$ باشد. پایه عددها چیست؟
- ۳-۷ مقدار همه بیت‌های یک ثبات 12 بیت که عددی معادل 215 دهدهی را نگهدارند:
الف) به دودویی؛ (ب) هشت هشتی کد شده بادودویی، (ج) شانزده شانزدهی کد شده با دودویی
د) دهدهی کد شده با دودویی (BCD)، نشان دهید.
- ۳-۸ آرایش یک ثبات 24 بیتی را که محتوایش معادل یک عدد دهدهی 295 است؛ (الف) به دودویی؛
(ب) به BCD؛ (ج) به ASCII با استفاده از هشت هشتی با توازن زوج، نشان دهید.
- ۳-۹ نام خود را با استفاده از یک کد هشت بیتی به ASCII نوشته و بیت سمت چپ منتهالیه را در نظریگیرید. بین نام و نام خانوادگی خود از کارکتر فضای خالی استفاده کنید و در صورت استفاده از فقط حرف اول نام، پس از آن نقطه بگذارید.
- ۳-۱۰ کد ASCII زیر را دیکد کنید
1001010 1001111 1001000 1001110 0100000 1000100 1001111 1000101
- ۳-۱۱ متمم 9 اعداد دهدهی هشت رقمی زیر را بدست آورید.
12349876 ; 00980100 * 90009951 ; , 00000000
- ۳-۱۲ متمم 10 اعداد دهدهی شش رقمی زیر را بدست آورید.
123900; 090657 ; 100000; , 000000
- ۳-۱۳ متمم‌های 1 و 2 اعداد دودویی هشت رقمی زیر را بدست آورید
10101110 ; 10000001 ; 10000000; 00000001; 00000000
- ۳-۱۴ تفریق را با اعداد دهدهی بدون علامت زیر و با استفاده از متمم 10 مفروق انجام دهید.
الف) 5250-1321 (ب) 1753-8640
ج) 20-100 (د) 1200-250
- ۳-۱۵ تفریق را با اعداد دودویی بدون علامت زیر و با استفاده از متمم 2 مفروق انجام دهید.
الف) 11010-10000 (ب) 11010-1101
ج) 100-110000 (د) 1010100-1010100

- ۳-۱۶ اعمال حسابی $(-13)+(+42)$ و $(-13)-(-42)$ را در دودویی و با استفاده از متمم 2 علامت دار برای اعداد منفی انجام دهید.
- ۳-۱۷ اعمال حسابی $(+80)+(+70)$ و $(-80)+(-70)$ را با نمایش اعداد دودویی به شکل متمم 2 علامت دار انجام دهید. از هشت بیت برای نمایش هر عدد به همراه علامت آن استفاده کنید. نشان دهید که در هر مورد سرریز رخ می دهد، دو نقلی آخر نامساویند، و علامت معکوس می شود.
- ۳-۱۸ اعمال حسابی زیر را برای اعداد دهدهی مشخص شده با استفاده از نمایش متمم 10 علامت دار برای اعداد منفی انجام دهید
- الف) $(+785) + (-638)$ ب) $(+785) - (-638)$
- ۳-۱۹ یک عدد دودویی ممیز شناور 36 بیتی دارای 8 بیت بعلاوه علامت برای نما و 26 بیت بعلاوه علامت برای مانیتس می باشد. مانیتس یک کسر نرمالیزه شده است. اعداد مانیتس و نما به فرم مقدار علامت دار هستند. بزرگترین و کوچکترین عددی که با این شکل می توان نشان داد، بجز صفر، کدامند؟
- ۳-۲۰ عدد $10(+46.5)$ را بصورت یک عدد دودویی ممیز شناور 24 بیتی نشان دهید. مانیتس که یک کسر نرمالیزه است 24 بیت و نما 8 بیت دارند.
- ۳-۲۱ کدگری گاهی اوقات کدانعکاسی خواننده می شود زیرا مقادیر بیت ها در دو طرف هریک از توان های 2^n انعکاس یافته یکدیگرند. مثلاً همانطور که در جدول 3-5 ملاحظه می شود، مقدار سه بیت پائین رتبه در دو طرف خطی که بین 7 و 8 کشیده شود انعکاس یکدیگرند. با استفاده از این خاصیت کدگری مطلوب است:
- الف) اعداد کدگری برای 16 تا 31 بعنوان ادامه جدول 3-5
- ب) کدگری افزونی 3 برای اعداد دهدهی 10 تا 19 به عنوان ادامه جدول 3-6
- ۳-۲۲ عدد دهدهی 8620 را به شکل (الف) BCD؛ (ب) افزونی 3؛ (ج) کد 2421؛ (د) عدد دودویی نمایش دهید.
- ۳-۲۳ ده رقم BCD را با توازن زوج در متتالیه سمت چپ آن (کلاً پنج بیت در هر رقم) نشان دهید. عمل را با توازن فرد تکرار کنید.
- ۳-۲۴ عدد دهدهی 3984 را بصورت کد 2421 جدول 3-6 نشان دهید. تمام بیت های عدد کد شده را متمم کنید و نشان دهید که متمم 9 همان عدد 3984 با کد 2421 است.
- ۳-۲۵ نشان دهید که تابع OR انحصاری $x=A\oplus B\oplus C\oplus D$ یک تابع فرد است. یکی از راه ها این است که جدول درستی $y=A\oplus B$ و $z=C\oplus D$ را بدست آورده و سپس جدول درستی $x=y\oplus z$ را بدست آورید. نشان دهید که $x=1$ است بشرطی که تعداد 1ها در A، B و C و D فرد باشد.
- ۳-۲۶ مدارهای یک مولد توازن سه بیتی و چک کننده توازن چهاربیتی را با استفاده از بیت توازن زوج بدست آورید (مدارات شکل 3-3 توازن فرد را بکار برده اند).

انتقال ثباتها و ریز عملها



۴-۱ زبان انتقال ثبات

۴-۲ انتقال ثبات

۴-۳ انتقالهای گذرگاهی و حافظه‌ای

۴-۴ ریز عملهای حسابی

۴-۵ ریز عملهای منطقی

۴-۶ ریز عملهای شیفت

۴-۷ واحد حساب، منطق و شیفت

۴-۱ زبان انتقال ثبات

یک سیستم دیجیتال مجموعه‌ای از ماژول‌های^۱ سخت افزاری متصل بهم است که کاری خاص را در زمینه پردازش اطلاعات انجام می‌دهند. سیستم‌های دیجیتال از نظر اندازه و پیچیدگی از چند مدار مجتمع تا مجموعه‌ای از کامپیوترهای مرتبط متغیرند. در طراحی سیستم‌های دیجیتال از روش ماژولی استفاده می‌شود. ماژول‌ها از اجزایی^۲ چون ثبات‌ها، دیکدرها عناصر حسابی و کنترل منطقی ساخته می‌شوند. ماژول‌های مختلف با مسیرهای مشترک داده و کنترل به هم متصل می‌شوند تا یک سیستم کامپیوتر دیجیتال بوجود آید.

ماژول‌های دیجیتال در بهترین فرم براساس ثبات‌ها و عملیاتی که روی داده‌های ذخیره شده در آنها انجام می‌شود تعریف می‌گردند. عملیاتی که روی داده‌های ذخیره شده در ثبات‌ها صورت می‌گیرد ریز عمل^۳ نام دارد. به بیان دیگر ریز عمل یک جزء عملیاتی است که بر روی اطلاعات ذخیره شده در یک یا چند ثبات انجام می‌شود. نتیجه عمل ممکن است جایگزین اطلاعات دودویی قبلی در ثبات شود و یا به ثبات دیگری انتقال یابد. مثال‌هایی از ریز عمل عبارتند از، شیفت^۴، شمارش^۵، پاک کردن^۶ و بارکردن^۷

1- Module

2- Component

۳- Microoperation

4- Shift

5- Count

6- Clear

7- Load

- است. برخی از مؤلفه‌های دیجیتال که در فصل 2 معرفی شدند ثبات‌هایی هستند که ریزعمل‌ها را پیاده سازی می‌کنند. مثلاً، یک شمارنده با قابلیت بارشدن موازی می‌تواند ریزعمل‌های افزایش و بارشدن را اجرا کند. یک ثبات شیفت (جابجایی) دوطرفه قادر است ریزعمل‌های شیفت به راست و چپ را انجام دهد. سازمان داخلی یک کامپیوتر دیجیتال به بهترین نحو توسط موارد زیر مشخص می‌شود:
- 1- مجموعه ثبات‌های آن و وظایف آنها
 - 2- رشته ریزعمل‌های انجام شده روی اطلاعات دودویی ذخیره شده در ثبات‌ها
 - 3- واحد کنترلی که موجب آغاز رشته ریز عمل‌ها می‌شود.

رشته ریزعمل‌ها در کامپیوتر را می‌توان با تشریح لفظی هر عمل مشخص کرد، ولی این روش معمولاً تشریحی طولانی خواهد بود. بهتر آنست تا روشی سمبلیک را برای توصیف رشته انتقال‌ها بین ثبات‌ها و ریزعمل‌های حسابی و منطقی مختلف مربوط به این انتقال‌ها، مشخص کنیم. استفاده از سمبل بجای توضیحات، روشی سازمان یافته و فشرده‌ای را برای نشان دادن رشته ریزعملها در ثبات‌ها و توابع کنترلی که موجب اجرای آنها می‌شوند فراهم می‌کند.

نحوه بیان سمبلیک مورد استفاده برای بیان انتقال‌های ریزعملی در بین ثبات‌ها، زبان انتقال ثباتها خواننده می‌شود. اصطلاح "انتقال ثباتها" بیانگر وجود مدارات منطقی سخت افزاری است که می‌تواند یک ریزعمل بیان شده را اجرا نماید و نتیجه عمل را به همان ثبات یا ثبات دیگر انتقال می‌دهد. کلمه "زبان" از برنامه نویسان اقتباس شده است، که این کلمه را برای زبان‌های برنامه نویسی بکار می‌برند. زبان برنامه نویسی رویه‌ای برای نوشتن سمبل‌هایی است که فرآیند محاسباتی خاصی را مشخص می‌کند. بطور مشابه، یک زبان طبیعی مانند انگلیسی، سیستمی است برای نوشتن سمبل‌ها و ترکیب آنها بصورت کلمات و جملات برای ارتباط بین انسانها. زبان انتقال ثبات هم سیستمی است برای بیان رشته ریزعمل‌ها بین ثبات‌های یک ماژول دیجیتال بصورت سمبلیک. چنین زبانی ابزار مناسبی برای توصیف فشرده و دقیق سازمان داخلی کامپیوترهای دیجیتال است. همچنین می‌توان از آن برای تسهیل روند طراحی سیستم‌های دیجیتال استفاده کرد.

اعتقاد براین است که زبان انتقال ثبات بکار رفته تا حد امکان ساده باشد، بنابراین بخاطر سپردن آن طولی نخواهد کشید. ما همچنان به تعریف سمبل برای انواع ریزعمل‌ها ادامه خواهیم داد، و همزمان با آن سخت افزاری که ریزعمل بیان شده را پیاده سازی کند معرفی می‌کنیم. علائم سمبلیک معرفی شده در این فصل، در فصل‌های بعدی برای مشخص کردن انتقالات ثبات‌ها، ریزعمل‌ها، و توابع کنترلی که سازمان سخت افزاری داخلی کامپیوترهای دیجیتال را بیان می‌کنند، بکار می‌روند. به محض آشنایی با این زبان، سایر سمبل‌های بکار رفته بسادگی قابل آموختن است زیرا بیشتر تفاوت‌های بین زبان‌های انتقال ثبات در جزئیات آنهاست، نه در کلیات.

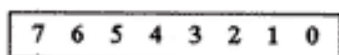
۲-۴ انتقال ثبات

ثبات‌های کامپیوتر با حروف بزرگ الفبای انگلیسی (وگاهی عددهایی به دنبال آنها) برای نشان دادن کار ثبات مشخص می‌شوند. مثلاً، ثباتی که آدرس را برای یک واحد حافظه نگه‌دارد ثبات آدرس حافظه^۱ نام داشته و با MAR مشخص می‌شود. سایر نام‌ها برای ثبات‌ها عبارتند از PC (برای شمارنده برنامه)^۲، IR (برای ثبات دستورالعمل)^۳، و R1 (برای ثبات پردازنده). فلیپ فلاپ‌های تشکیل دهنده یک ثبات n بیتی به طور متوالی از 0 الی n-1 از راست به چپ شماره گذاری می‌شوند. شکل ۱-۴ نمایش ثباتها را بصورت بلاک دیاگرام نشان می‌دهد. متداول‌ترین راه نمایش یک ثبات، استفاده از کادر مستطیل شکل همراه با نام ثبات در داخل آن است، شکل ۱-۴ (الف). بیت‌های مختلف ثبات مانند قسمت (ب) در شکل متمایز می‌شوند. شماره گذاری بیت‌ها در یک ثبات 16 بیتی، مانند قسمت (ج)، در بالای آن مشخص می‌شود. یک ثبات 16 بیتی هم در قسمت (د) به دو قسمت تقسیم شده است. به بیت‌های 0 تا 7 علامت L (برای بایت کم ارزشتر) و به بیت‌های 8 تا 15 سمبل H (برای بایت با ارزشتر) بکار می‌رود. نام ثبات 16 بیتی PC است. سمبل PC(0-7) به بایت کم ارزشتر اطلاق می‌شود و PC(8-15) یا PC(H) به بایت با ارزشتر اشاره می‌کند.

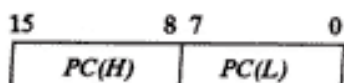
انتقال اطلاعات از یک ثبات به ثبات دیگر بصورت سمبلیک با استفاده از عملگر جایگزینی مشخص می‌شود. عبارت:

$$R2 \leftarrow R1$$

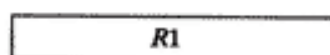
بیانگر یک انتقال از ثبات R1 به ثبات R2 است. این عبارت جایگزینی شدن محتوای R2 را با R1 مشخص می‌نماید. بنابه تعریف، محتوای ثبات R1 پس از انتقال تغییر نمی‌کند. عبارتی که مشخص کننده انتقال ثباتها باشد دلالت بر این دارد که مدارهایی از خروجی‌های ثبات مبدأ به ورودی‌های ثبات مقصد وجود دارند و ثبات مقصد دارای قابلیت بارشدن موازی است. معمولاً



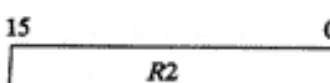
(ب) نشان دادن بیت‌های مختلف



(د) ثباتی که به دو بخش تقسیم شده



(الف) ثبات R



(ج) شماره گذاری بیت‌ها

شکل ۱-۴ بلاک دیاگرام ثبات

1- Memory Address Register 2- Program Counter

3- Instruction Register

می خواهیم که انتقال تحت شرایط کنترل از پیش تعیین شده ای انجام شود. این موضوع را می توان با یک عبارت مانند رابطه زیر نشان داد.

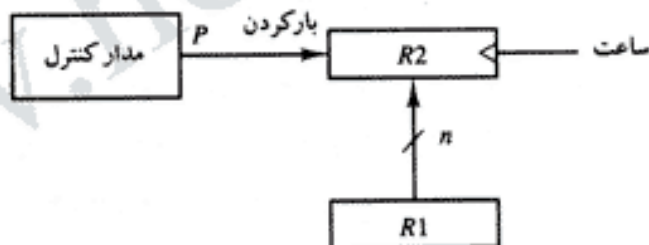
$$\text{If } (P = 1) \text{ then } (R2 \leftarrow R1)$$

که در آن P یک سیگنال کنترلی است که در بخش کنترل تولید می شود. گاهی بهتر است که متغیرهای کنترل را با مشخص کردن یک تابع کنترل از عمل انتقال ثبات جدا کنیم. تابع کنترل یک متغیر بولی است که برابر 1 یا 0 است. تابع کنترل در عبارت بصورت زیر مشخص می شود

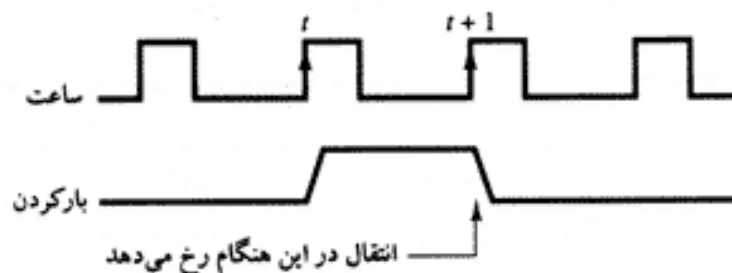
$$P: P2 \leftarrow R1$$

شرط کنترل با علامت نقل ¹، خاتمه می یابد. این سمبل بیان می دارد که لازمه عمل انتقال توسط سخت افزار این است که $P=1$ باشد.

هر یک از عبارت های نوشته شده به زبان انتقال ثبات ها، دلالت بر وجود یک ساختمان سخت افزاری برای پیاده سازی انتقال دارد. شکل ۲-۴، انتقال از R1 به R2 را بصورت بلاک دیاگرام نمایش می دهد. n خروجی ثبات R1 به n ورودی ثبات R2 متصل است. از حرف n برای نشان دادن تعداد بیت های ثبات استفاده شده است. اگر طول ثبات معین باشد، به جای n، یک عدد واقعی گذاشته



(الف) بلاک دیاگرام



(ب) دیاگرام زمانی

شکل ۲-۴ انتقال از ثبات R1 به R2 به هنگام $P = 1$

می شود. ثبات $R2$ دارای ورودی بارکردن است که بوسیله متغیر کنترل P فعال می شود. فرض بر این است که متغیر کنترل با همان ساعتی که به ثبات اعمال می شود همگام¹ است. همانطور که در دیاگرام زمانبندی نشان داده شده است، P در بخش کنترل با لبه بالارونده پالس ساعت در زمان t فعال می شود. گذر مثبت بعدی ساعت در زمان $t+1$ ورودی بارکردن را فعال یافته و سپس داده های ورودی $R2$ بداخل ثبات بصورت موازی بار می شوند. در زمان $t+1$ کنترل P می تواند به 0 بازگردد؛ در غیر این صورت، عمل انتقال با هر گذر پالس ساعت مادامی که P فعال است تکرار می شود.

توجه کنید که پالس ساعت بعنوان یک متغیر در عبارت انتقال ثبات آورده نشده است. فرض بر این است که تمام انتقال ها در یک لبه گذر پالس ساعت رخ می دهند. هرچند که شرایط کنترلی مانند P درست پس از زمان t فعال می شود، ولی عمل انتقال تا وقتی که ثبات با گذر مثبت بعدی ساعت در زمان $t+1$ فعال نشود رخ نمی دهد.

سمبل های اصلی زبان انتقال ثبات ها در جدول ۱-۴ لیست شده اند. ثبات ها با حروف بزرگ نشان داده می شوند و ممکن است اعدادی به دنبال این حروف آورده شوند. برای مشخص کردن بخشی از یک ثبات از پرانتز استفاده می شود و محدوده بیت ها یا یک نام سمبلیک برای آن بخش در داخل پرانتز ذکر می شود. علامت فلش بیانگر انتقال اطلاعات و جهت آن است. از کاما^۲ برای جدا کردن دو یا چند عمل که همزمان انجام می شوند استفاده می شود. عبارت

$$T: R2 \leftarrow R1, R1 \leftarrow R2$$

نشان دهنده عملی است که محتوای دو ثبات در طول یک پالس ساعت مشترک با یکدیگر تعویض می شوند، مشروط بر این که $T=1$ باشد. این عمل همزمان با ثبات هایی امکان پذیر است که فلیپ فلاپ های حساس به لبه پالس داشته باشند.

جدول ۱-۴ سمبل های اصلی انتقال ثبات ها

مثال	توضیح	سمبل
$MAR R2$	یک ثبات را مشخص می کند	حروف (و ارقام)
$R2(0-7) R2(L)$	بخشی از یک ثبات را مشخص می کند	پرانتز ()
$R2 \leftarrow R1$	انتقال اطلاعات را مشخص می کند	پیکان \leftarrow
$R2 \leftarrow R1 R1 \leftarrow R2$	دو ریز عمل را از هم جدا می کند	کاما

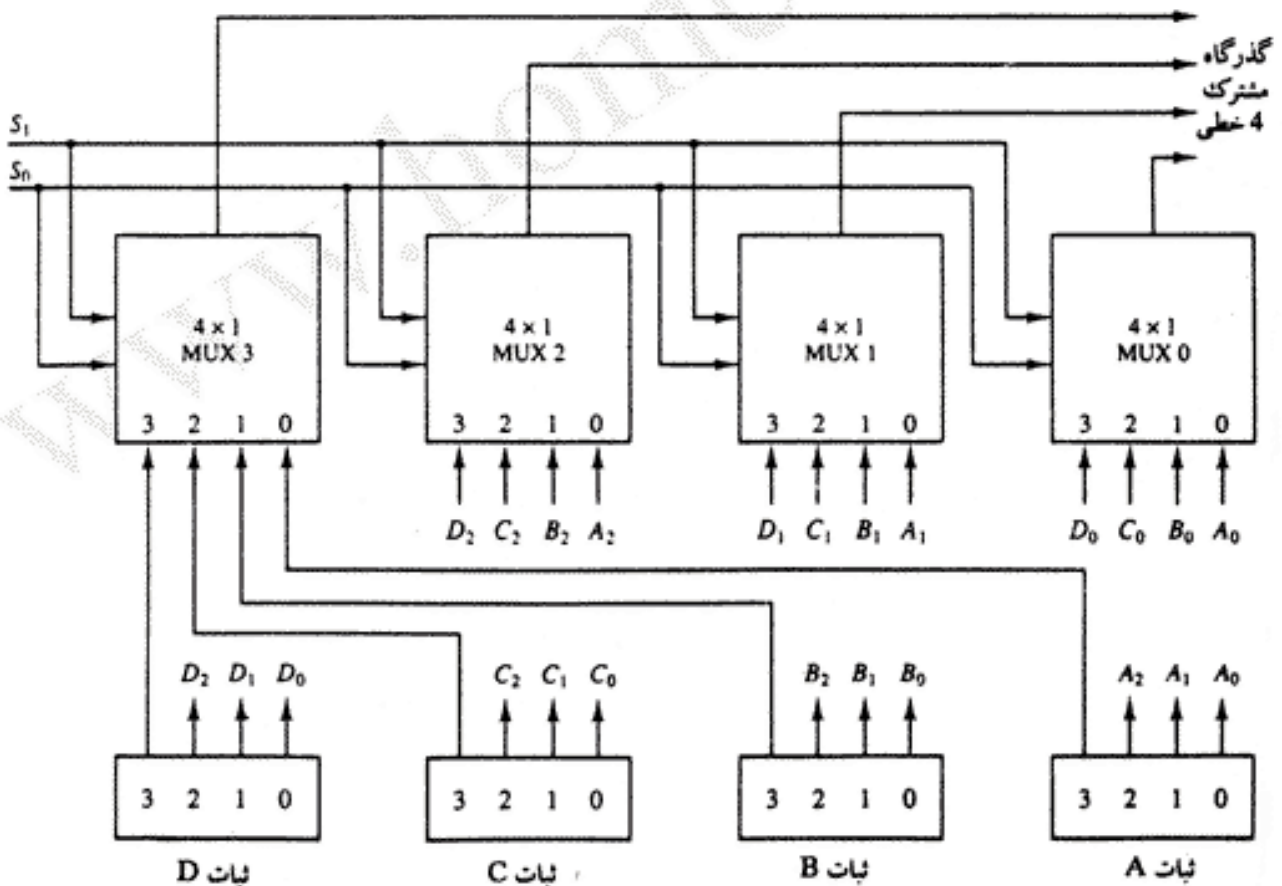
1- synchro

2- Comma

۴-۳ انتقال‌های گذرگاهی و حافظه‌ای

یک کامپیوتر دیجیتال نوعاً ثبات‌های زیادی دارد و باید در آن مسیرهایی برای انتقال اطلاعات از یک ثبات به ثبات دیگر فراهم شود. اگر خطوط جداگانه‌ای بین هر ثبات و دیگر ثبات‌های سیستم به کار رود، تعداد سیم‌ها بیش از حد خواهد شد. روش کاراتر برای انتقال اطلاعات بین ثبات‌ها در یک آرایش چند ثباتی، سیستم گذرگاه مشترک است. ساختار یک گذرگاه از مجموعه‌ای از خطوط مشترک تشکیل می‌شود، که تعداد آنها، یک خط بازای هر یک از بیت‌های ثبات‌هاست، و از طریق آنها اطلاعات دودویی یکی یکی انتقال می‌یابند. سیگنال کنترل، تعیین‌کننده ثبات انتخاب شده بوسیله گذرگاه هستند.

یک راه ساخت یک سیستم گذرگاه مشترک استفاده از مولتی پلکسر است. مولتی پلکسرها ثبات مبداء را انتخاب می‌کنند و سپس اطلاعات آنها روی گذرگاه قرار می‌گیرد. ساخت یک سیستم گذرگاه برای چهار ثبات در شکل ۴-۳ نشان داده شده است. هر ثبات دارای چهار بیت است، که از 0 تا 3 شماره‌گذاری شده‌اند. گذرگاه از چهار مولتی پلکسر 4×1 تشکیل شده که هر یک چهار ورودی داده 0 تا 3 و دو ورودی انتخاب S_1 و S_0 دارند. برای جلوگیری از پیچیده شدن دیاگرام بعلاوه تقاطع 16 خط متقاطع،



شکل ۴-۳ سیستم گذرگاه برای چهار ثبات

برای نشان دادن اتصالات خروجی های ثابت ها به ورودی های مولتی پلکسرها از پرچسب یا حروف استفاده می شود. مثلاً خروجی 1 در ثابت A به ورودی 0 از MUX1 متصل است زیرا این ورودی دارای پرچسب A1 است. با توجه به دیاگرام دیده می شود که بیت های هم ارزش در هر ثابت به ورودی های یک مولتی پلکسر متصل می شوند تا یک خط گذرگاه را تشکیل دهند. بنابراین MUX0 چهار بیت 0 ثابتها را راه دهی می کند، MUX1 چهار بیت 1 ثابتها را راه دهی می نماید، و برای دوبیت دیگر نیز به همین ترتیب.

دو خط انتخاب S_0 و S_1 به ورودی های انتخاب هر چهار مولتی پلکسر متصل اند. خطوط انتخاب چهار بیت یک ثابت را انتخاب کرده و آنها را به گذرگاه مشترک منتقل می نمایند. اگر $S_1S_0=00$ باشد، ورودی داده 0 هر چهار مولتی پلکسر، انتخاب شده و به خروجی ها که گذرگاه را تشکیل می دهند اعمال می گردند. این موجب می شود تا خطوط گذرگاه محتویات ثابت A را دریافت دارند زیرا خروجی این ثابت به ورودی داده 0 مولتی پلکسرها متصل شده است. بطور مشابه، ثابت B بشرطی انتخاب می شود که $S_1S_0=01$ باشد والی آخر. جدول ۲-۴ ثباتی را نشان می دهد که بوسیله گذرگاه بازاء هر یک از چهار مقدار دودویی ممکن خطوط انتخاب، برگزیده شده است.

بطور کلی، یک سیستم گذرگاه، k ثابت n بیتی را برای تشکیل یک گذرگاه مشترک n خطی راه دهی می نماید. تعداد مولتی پلکسرها مورد نیاز برای ساخت گذرگاه برابر n است که همان تعداد تعداد بیت ها در هر ثابت است. اندازه هر مولتی پلکسر باید $k \times 1$ باشد زیرا که خط داده را راه دهی می نماید. مثلاً یک گذرگاه مشترک برای هشت ثابت 16 بیتی به 16 مولتی پلکسر احتیاج دارد که هر یک متعلق به یک خط در گذرگاه است. هر مولتی پلکسر باید دارای هشت خط ورودی داده و سه خط انتخاب برای راه دهی مجموعه بیت های هم ارزش در هشت ثابت باشد.

انتقال اطلاعات از یک گذرگاه بداخل یکی از چند ثابت مقصد با اتصال خطوط گذرگاه به ورودی های تمام ثابت های مقصد و فعال کردن کنترل بازکردن ثابت مقصد خاصی که انتخاب شده تحقق می یابد. گذرگاه ممکن است در عبارت سبملیک انتقال گذرگاه ذکر شود و یا وجود آن در عبارت بطور ضمنی فرض گردد. وقتی که گذرگاه در عبارت فید شود انتقال ثابت بصورت زیر نشان داده می شود.

محتوای ثابت C روی گذرگاه قرار گرفته است و محتوای گذرگاه نیز با فعال کردن ورودی کنترل بازکردن

$$BUS \leftarrow C, \quad R1 \leftarrow BUS$$

جدول ۲-۴ جدول تابع برای گذرگاه شکل ۲-۴

S_1	S_0	ثباتی که انتخاب می شود
0	0	A
0	1	B
1	0	C
1	1	D

در ثبات $R1$ قرار می‌گیرد. اگر وجود گذرگاه در سیستم مسجل باشد، مناسبتر خواهد بود که فقط جهت انتقال نشان داده شود.

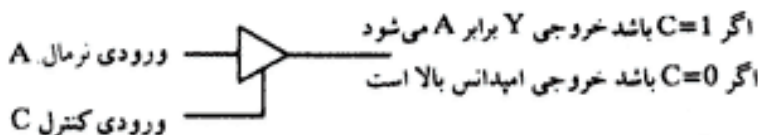
$$R1 \leftarrow C$$

باتوجه به عبارت فوق طراح متوجه می‌شود که سیگنال‌های کنترلی باید فعال شوند تا انتقال از طریق گذرگاه انجام گردد.

بافرهای سه حالتی گذرگاه

گذرگاه سیستم می‌تواند بجای مولتی پلکسر از گیت‌های سه حالتی ساخته شود. گیت‌های سه حالتی یک مدار دیجیتال است که سه حالت را در خروجی خود ایجاد می‌کند. دو تا از این حالات سیگنالهای معادل 1 و 0 منطقی همانند گیت‌های معمولی هستند. حالت سوم، حالت امپدانس بالاست. این حالت مانند مدار باز عمل می‌کند، یعنی خروجی قطع است و مفهوم منطقی ندارد. گیت‌های سه حالتی می‌توانند هر یک از انواع گیت‌های منطقی مانند AND یا NAND باشند. اما نوعی که بیش از همه در طراحی یک سیستم گذرگاه به کار می‌رود گیت بافر¹ (یا میانگیر) می‌باشد.

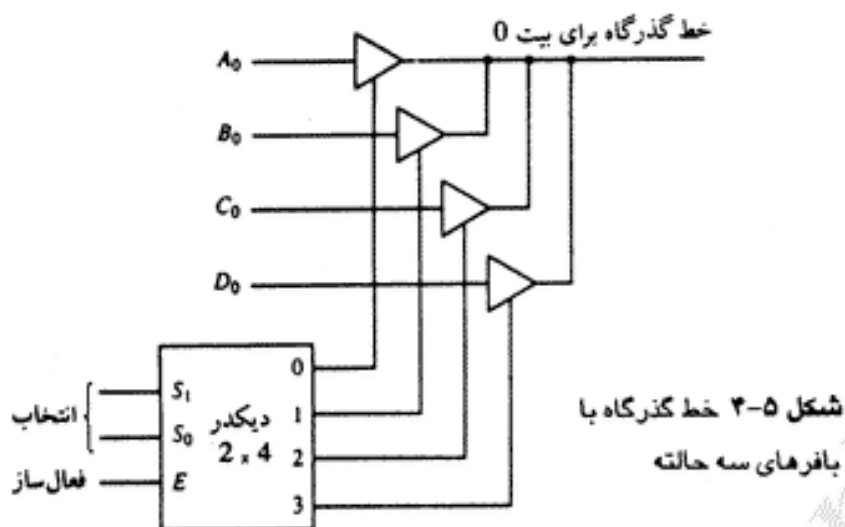
سمبل گرافیکی یک گیت بافر سه حالتی در شکل ۴-۴ نشان داده شده است. وجود ورودی کنترل، علاوه بر ورودی عادی، آن را از میانگیرهای عادی متمایز می‌کند. اگر ورودی کنترل 1 باشد، خروجی فعال می‌شود و گیت مانند هر بافر معمولی کار کرده و خروجی آن برابر ورودی نرمال* است. وقتی ورودی کنترل 0 باشد، خروجی غیرفعال شده و گیت به حالت امپدانس بالا می‌رود و به ورودی نرمال بستگی نخواهد داشت. حالت امپدانس بالای یک گیت سه حالتی ویژگی خاصی را فراهم می‌آورد که در گیت‌های دیگر موجود نیست. بعلاوه این ویژگی، خروجی تعداد زیادی گیت سه حالتی را می‌توان با سیم‌هایی بهم متصل نموده و بدون اینکه خطر تأثیر بارشدن پیش آید یک گذرگاه مشترک را تشکیل داد. ساختن یک سیستم گذرگاه با میانگیرهای سه حالتی در شکل ۴-۵ نشان داده شده است. خروجی‌های چهار بافر به هم متصل شده‌اند تا یک خط گذرگاه بوجود آید. (دقت داشته باشید که این نوع اتصالات بر روی گیت‌هایی که فاقد خروجی سه حالتی هستند امکان پذیر نیست.) ورودی‌های کنترل بافرها تعیین می‌کنند که کدام یک از چهار ورودی نرمال با خط گذرگاه ارتباط برقرار کند. در هر



شکل ۴-۴ سمبل گرافیکی بافر سه حالتی

1- Buffer

* - منظور از بکارگیری کلمه "نرمال" تفکیک ورودی از ورودی کنترل است.



لحظه از زمان فقط یک بافر می تواند فعال باشد. بافرهای متصل به خط باید طوری کنترل شوند که فقط یک بافر سه حالت به گذرگاه دسترسی داشته و سایر بافرها در حالت امپدانس بالا باشند.

برای تضمین این که در هر لحظه از زمان فقط یک ورودی کنترل فعال باشد از یک دیکدر مطابق شکل ۴-۵ استفاده می شود. وقتی که ورودی تواناساز دیکدر 0 است، هر چهار خروجی آن 0 بوده و خط گذرگاه در حالت امپدانس بالاست زیرا تمام چهار بافر ناتوان شده اند. وقتی که ورودی تواناساز فعال باشد، یکی از بافرهای سه حالت، بسته به مقدار دودویی در ورودی های انتخاب دیکدر، فعال می شود. بررسی دقیق شکل ۴-۵ نشان می دهد که این روش راه دیگری برای ساختن یک مولتی پلکسر 4x1 است زیرا این مدار می تواند جایگزین مولتی پلکسر شکل ۳-۴ گردد.

برای ایجاد یک گذرگاه مشترک برای چهار ثبات n بیتی با استفاده از بافرهای سه حالت، ما به n مدار، که در هر کدام چهار بافر سه حالت مانند شکل ۴-۵ وجود داشته باشد نیاز داریم. هر گروه چهار بافره یک بیت هم ارزش را از چهار ثبات دریافت می کند. هر خروجی مشترک یکی از خطوط را برای گذرگاه مشترک ایجاد می کند که در مجموع n خط بوجود خواهد آمد. برای انتخاب بین چهار ثبات فقط یک دیکدر لازم است.

انتقال های حافظه ای

عملکرد حافظه در بخش ۷-۲ شرح داده شد. انتقال اطلاعات از یک کلمه حافظه به محیط خارج، "خواندن" نامیده می شود. انتقال اطلاعات جدیدی که در حافظه ذخیره می شود، "نوشتن" نام دارد. یک کلمه حافظه را با حرف M نشان خواهیم داد. انتخاب یک کلمه حافظه خاص از میان چندین حافظه موجود بوسیله آدرس حافظه در حین انتقال صورت می گیرد. وقتی که عمل نوشتن انجام می شود لازم

است آدرس حافظه مشخص شود. این عمل با قرار دادن آدرس در گروهی که دنبال حرف M آمده است انجام می‌گردد.

واحد حافظه‌ای را در نظر بگیرید که آدرس را از یک ثبات، که آن را ثبات آدرس می‌نامیم، و با AR نشان داده می‌شود، دریافت می‌کند. داده‌ها به ثبات دیگری بنام ثبات داده، DR، منتقل می‌شوند. این عمل خواندن را با عبارت زیر می‌توان بیان کرد.

Read: $DR \leftarrow M[AR]$

این عبارت موجب انتقال اطلاعات از کلمه حافظه M، که بوسیله آدرس موجود در AR انتخاب شده، به DR می‌گردد.

عمل نوشتن محتوای یک ثبات داده را به حافظه M، انتخاب شده بوسیله آدرسی، انتقال می‌دهد. فرض کنید که داده ورودی در ثبات R1 و آدرس در AR باشد. عمل نوشتن بصورت زیر می‌تواند بیان شود.

Write: $M[AR] \leftarrow R1$

عبارت فوق موجب انتقال اطلاعات از R1 به کلمه حافظه M، انتخاب شده بوسیله آدرس موجود در AR، می‌گردد.

۴-۴ ریزعمل‌های حسابی

ریزعمل^۱ یک عمل جزئی است که روی داده‌های ذخیره شده در ثبات‌ها انجام می‌شود. ریزعمل‌هایی که بیش از همه در کامپیوترهای دیجیتال با آن مواجه می‌شویم به چهار دسته زیر طبقه بندی می‌شوند:

۱- ریزعمل‌های انتقال ثبات، اطلاعات دودویی را از یک ثبات به ثبات دیگر منتقل می‌کنند.

۲- ریزعمل‌های حسابی عملیات محاسباتی را روی داده‌های عددی ذخیره شده در ثبات‌ها انجام می‌دهند.

۳- ریزعمل‌های منطقی عملیات دستکاری بیت‌ها را روی داده‌های غیر عددی ذخیره شده در ثبات‌ها انجام می‌دهند.

۴- ریزعمل‌های شیفت^۲ اعمال شیفت (جابجایی) را روی داده‌های ذخیره شده در ثبات‌ها اجرا می‌کنند.

ریزعمل انتقال ثبات، در بخش ۲-۴ معرفی شد. این نوع ریزعمل محتوای اطلاعاتی را، هنگامی که از ثبات‌های مبدأ به ثبات مقصد منتقل می‌شود، تغییر نمی‌دهد. سه نوع دیگر ریزاعمال محتوای اطلاعاتی را حین انتقال تغییر می‌دهند. در این بخش ما مجموعه‌ای از ریزعمل‌های حسابی را معرفی

1- Microoperation

2- Shift

خواهیم کرد. در دو بخش بعد ما ریز عمل های منطقی و شیفت را ارائه خواهیم داد. ریز عمل های حسابی پایه عبارتند از جمع، تفریق، افزایش، کاهش و شیفت. شیفت های حسابی بعداً همراه با ریز عمل های شیفت توضیح داده خواهد شد. ریز عمل حسابی زیر

$$R3 \leftarrow R1 + R2$$

بیانگر ریز عمل "جمع" است. این عبارت بیان می کند که محتوای ثبات $R1$ با محتوای ثبات $R2$ جمع شده و حاصل جمع به ثبات $R3$ انتقال می یابد. برای پیاده سازی این عبارت با سخت افزار، ما به سه ثبات و قطعات منطقی که عمل جمع را انجام دهند نیاز داریم. سایر ریز عمل های اصلی حسابی در جدول ۳-۴ لیست شده اند. تفریق غالباً با استفاده از متمم سازی و جمع پیاده سازی می شود. در عوض بکارگیری عملگر "منها"، ما تفریق را بوسیله عبارت زیر مشخص می کنیم.

$$R3 \leftarrow R1 + \overline{R2} + 1$$

$\overline{R2}$ سمبل متمم 1 ثبات $R2$ است. با جمع 1 با متمم 1 متمم 2 تولید می شود. جمع محتوای $R1$ با متمم 2 محتوای ثبات $R2$ ، برابر ست با $R1-R2$.

ریز اعمال افزایش و کاهش را بترتیب بوسیله عمل های جمع با 1 و تفریق 1 انجام می دهیم. این ریز اعمال با مدارهای ترکیبی یا با شمارنده های بالا - پایین¹ پیاده سازی می شوند. اعمال حسابی ضرب و تقسیم در جدول ۳-۴ لیست نشده اند. این دو عمل، اعمال حسابی معتبری هستند ولی در مجموعه دستورات پایه منظور نشده اند. تنها جایی که این اعمال بعنوان ریز عمل در نظر گرفته می شوند، در یک سیستم دیجیتال است که در آن این اعمال بصورت مدارهای ترکیبی پیاده سازی می شوند. در چنین حالتی، سیگنال هایی این اعمال را در طول گیت ها منتشر می نمایند و نتیجه

جدول ۳-۴ ریز عمل های حسابی

نمایش سمبلیک	شرح
$R3 \leftarrow R1 + R2$	محتوای $R1$ بعلاوه $R2$ به $R3$ منتقل می شود
$R3 \leftarrow R1 - R2$	محتوای $R1$ منهای $R2$ به $R3$ منتقل می گردد
$R2 \leftarrow \overline{R2}$	محتوای $R1$ متمم می شود (متمم 1)
$R2 \leftarrow \overline{R2} + 1$	محتوای $R2$ متمم می شود (منفی می شود)
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ بعلاوه متمم 2 از $R2$ (تفریق)
$R1 \leftarrow R1 + 1$	یک واحد افزایش در محتوای $R1$
$R1 \leftarrow R1 - 1$	یک واحد کاهش در محتوای $R1$

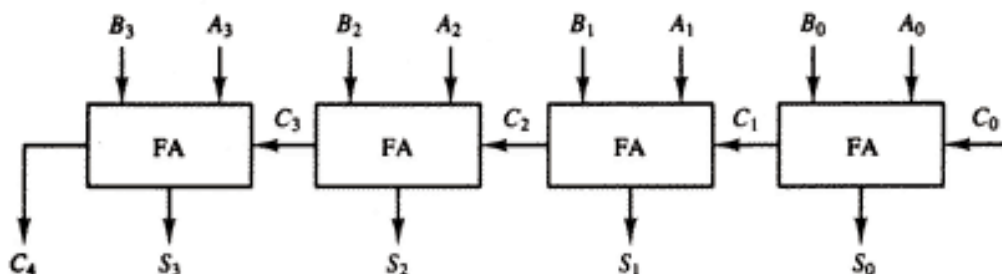
1- UP-Down counter

عمل می‌تواند به محض اینکه سیگنال خروجی از مدار ترکیبی عبور کرد با یک پالس ساعت به داخل ثبات مقصد انتقال یابد. در اکثر کامپیوترها، عمل ضرب با رشته‌ای از ریزعمل‌های جمع و شیف‌ت اجرا می‌شود. تقسیم با رشته‌ای از ریزعمل‌های تفریق و شیف‌ت صورت می‌گیرد. مشخص کردن سخت افزار برای این موارد مستلزم استفاده از لیست عباراتی است که در آنها ریزعمل‌های جمع و تفریق و شیف‌ت بکار رفته‌اند.

جمع کننده دودویی

برای پیاده سازی ریز عمل جمع بوسیله سخت افزار، ما به ثبات‌هایی که داده را نگهدارند و نیز قطعات یا مؤلفه‌های دیجیتالی که جمع حسابی را اجرا نمایند، نیاز داریم. مدار دیجیتالی که مجموع حسابی دو بیت و رقم نقلی قبلی را تشکیل می‌دهد تمام جمع کننده¹ نامیده می‌شود (شکل ۱-۱۷ ملاحظه شود). مدار دیجیتالی که جمع دو عدد دودویی با هر طول را اجرا می‌نماید جمع کننده دودویی^۲ خوانده می‌شود. جمع کننده دودویی از مدارهای تمام جمع کننده که بصورت سری بهم متصلند ساخته می‌شود، باین ترتیب که رقم نقلی خروجی یک تمام جمع کننده به ورودی رقم نقلی تمام جمع کننده بعدی متصل است. شکل ۴-۶ اتصالات درونی چهار تمام جمع کننده (FA) را برای فراهم کردن یک جمع کننده دودویی چهاربیت نشان می‌دهد. بیت‌های مضاف A و مضاف الیه B بوسیله زیرنویس‌هایی از راست به چپ مشخص شده‌اند و در آن زیرنویس 0 متعلق به کم ارزشترین بیت است. ارقام نقلی، بصورت زنجیره‌ای بین تمام جمع کننده‌ها کشیده شده‌اند. رقم نقلی ورودی به جمع کننده دودویی، C_0 و رقم نقلی خروجی C_4 است. خروجی S تمام جمع کننده‌ها بیت‌های حاصل جمع موردنظر را تولید می‌نمایند.

یک جمع کننده دودویی n بیتی به n تمام جمع کننده نیاز دارد. رقم نقلی خروجی از هر تمام جمع کننده به ورودی نقلی تمام جمع کننده مرتبه بالاتر بعدی متصل است. n بیت داده برای ورودی‌های A از یک ثبات (مانند R1) می‌آیند، و n بیت داده برای ورودی‌های B نیز از ثبات دیگری (مانند R2) گرفته



شکل ۴-۶ جمع کننده دودویی 4 بیتی

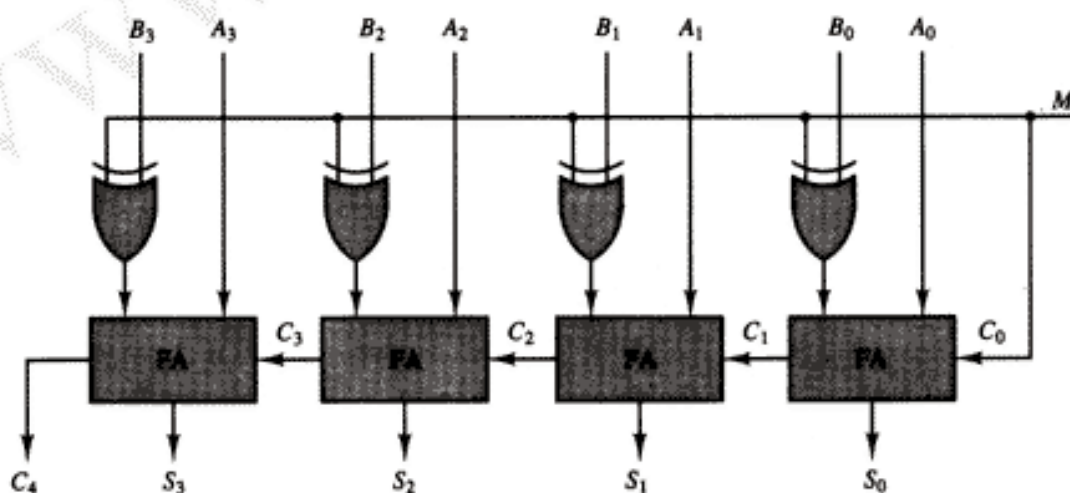
1- Full Adder

2- Binary Adder

می‌شوند. مجموع می‌تواند به ثبات سومی و یا بیکی از ثبات های مبدأ ($R1$ یا $R2$) منتقل شده و جایگزین محتوای قبلی آن گردد.

جمع و تفریق کننده دودویی

تفریق اعداد دودویی را می‌توان به بهترین وجه با استفاده از متمم طبق آنچه در بخش ۲-۳ بحث شده انجام داد. به خاطر بیاورید که تفریق $A-B$ با بدست آوردن متمم B و جمع آن با A انجام می‌شود. متمم ۲، از متمم ۱ و جمع کم ارزشترین جفت بیت‌ها با ۱ حاصل می‌گردد. متمم ۱ را هم با گیت‌های معکوس کننده می‌توان ایجاد کرد و جمع کردن یک واحد را نیز می‌توان از طریق ورودی نقلی انجام داد. اعمال جمع و تفریق را می‌توان با افزودن یک گیت OR انحصاری با هم ترکیب کرده و بصورت یک مدار مشترک درآورد. یک مدار جمع و تفریق کننده چهاربیتی در شکل ۴-۷ نشان داده شده است. ورودی حالت M عمل را کنترل می‌کند. اگر $M=0$ باشد، مدار یک جمع کننده و هر وقت $M=1$ باشد مدار تبدیل به یک تفریق گر می‌شود. هر گیت OR انحصاری ورودی M و یکی از ورودی‌های B را دریافت می‌کند. اگر $M=0$ باشد داریم $B \oplus 0 = B$. تمام جمع کننده‌ها مقدار B و رقم نقلی ۰ را دریافت کرده و مدار عمل A بعلاوه B را انجام می‌دهد. اگر $M=1$ باشد، داریم $B \oplus 1 = B'$ و $C_0=1$. ورودی‌های B کلاً متمم شده و مقدار ۱ از طریق نقلی ورودی با آن جمع می‌شود. مدار عمل A بعلاوه متمم ۲ عدد B را انجام می‌دهد. برای اعداد بدون علامت، این عمل اگر $A \geq B$ باشد برابر $A-B$ و اگر $A < B$ باشد متمم ۲ مقدار $B-A$ خواهد بود. برای اعداد علامت دار نتیجه $A-B$ می‌باشد، بشرطی که سرریزی وجود نداشته باشد.



شکل ۴-۷ جمع و تفریق کننده ۴ بیتی

1- Mode Input

افزایشگر دودویی

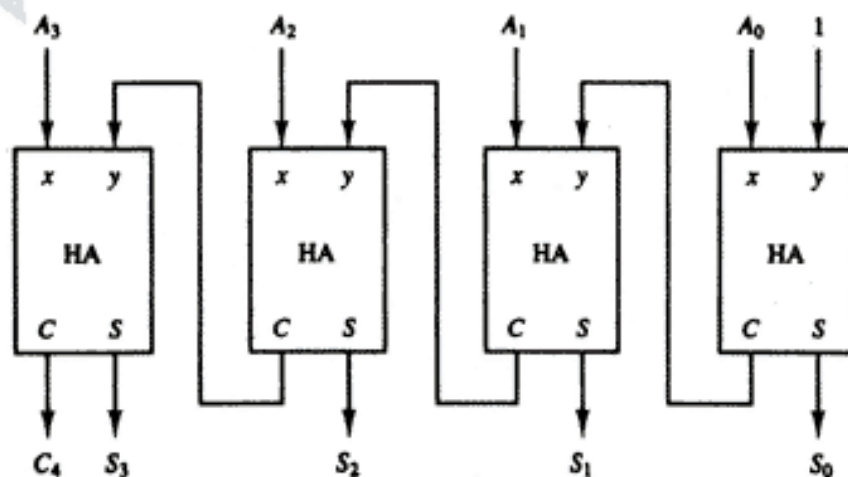
ریزعمل افزایش، به عدد داخل یک ثبات یک واحد اضافه می‌کند. مثلاً اگر یک ثبات 4 بیتی دارای یک مقدار دودویی 0110 باشد، پس از یک افزایش به 0111 بدل خواهد شد. این ریزعمل بسادگی با یک شمارنده (شکل ۱۰-۲ ملاحظه شود) پیاده سازی می‌شود. هر دفعه که تواناساز شمارش فعال شود، لبه پالس ساعت محتوای ثبات را یک واحد افزایش می‌دهد. مواردی هم پیش می‌آید که ریزعمل افزایش باید با یک مدار ترکیبی و مستقل از یک ثبات انجام شود. این کار با استفاده از نیم جمع کننده‌هایی که با هم متوالیاً بسته شده‌اند (شکل ۱۶-۱) امکان پذیر است.

دیاگرام مدار ترکیبی افزایشگر چهاربیتی در شکل ۸-۴ نشان داده شده است. یکی از ورودی‌های نیم‌جمع‌کننده^۱ با ارزش کم‌تر به 1 متصل شده است و ورودی دیگر به بیت کم ارزشتر عددی که قرار است افزایش یابد وصل می‌شود. رقم نقلی خروجی از یک نیم جمع‌کننده به یکی از ورودی‌های نیم جمع‌کننده مرتبه بالاتر بعدی اتصال یافته است. مدار، چهار بیت A_0 تا A_3 را دریافت کرده، یک واحد به آن اضافه می‌کند، و خروجی افزایش یافته را در S_0 تا S_3 تولید می‌نماید. رقم نقلی خروجی C_4 فقط پس از حالت دودویی 1111 برابر 1 خواهد شد. در این حالت S_0 تا S_3 هم 0 می‌شوند.

مدار شکل ۸-۴ می‌تواند به یک افزایشگر n بیتی گسترش یابد و این با گسترش دیاگرام توسط n نیم جمع‌کننده امکان پذیر است. یکی از بیت‌های کم ارزشتر باید به 1 منطقی متصل گردد. سایر ورودی‌ها، عددی را که باید افزایش یابد و یا رقم نقلی مرحله قبلی را دریافت می‌کنند.

مدار حسابی

ریزعمل‌های لیست شده در جدول ۳-۴ را می‌توان با یک مدار حسابی تکمیل شده‌ای پیاده سازی



شکل ۸-۴ افزایشگر دودویی 4 بیتی

1- Half Adder

کرد. قطعه اصلی یک مدار حسابی، جمع کننده موازی است. با کنترل ورودی های داده به جمع کننده، می توان انواع اعمال حسابی مختلف را بدست آورد.

دیاگرام یک مدار حسابی 4 بیتی در شکل ۹-۴ نشان داده شده است. در این دیاگرام چهار مدار تمام کننده وجود دارد که جمع چهار بیت را بوجود می آورد و چهار مولتی پلکسر نیز اعمال مختلف را انتخاب می کنند. دو ورودی چهار بیت A و B و یک خروجی 4 بیت نیز وجود دارند. چهار ورودی A مستقیماً به ورودی های X جمع کننده دودویی می روند. هر یک از چهار ورودی B هم به ورودی های داده مولتی پلکسر ها متصلند. ورودی های داده مولتی پلکسر ها متمم B را نیز دریافت می نمایند. دو ورودی دیگر داده به 0 و 1 منطقی وصل شده اند. 0 منطقی یک سطح ولتاژ ثابتی است (0 ولت در مدارهای مجتمع TTL) و سیگنال 1 منطقی می تواند توسط یک معکوس کننده که ورودی آن در 0 است تولید گردد. چهار مولتی پلکسر بوسیله دو خط انتخاب S_0 و S_1 کنترل می شوند. رقم نقلی C_{in} به ورودی نقلی FA در مکان کم ارزشتر وارد می شود. سایر ارقام نقلی از یک مرحله به مرحله بعد متصل می شوند. خروجی جمع کننده دودویی از جمع حسابی زیر محاسبه می شود

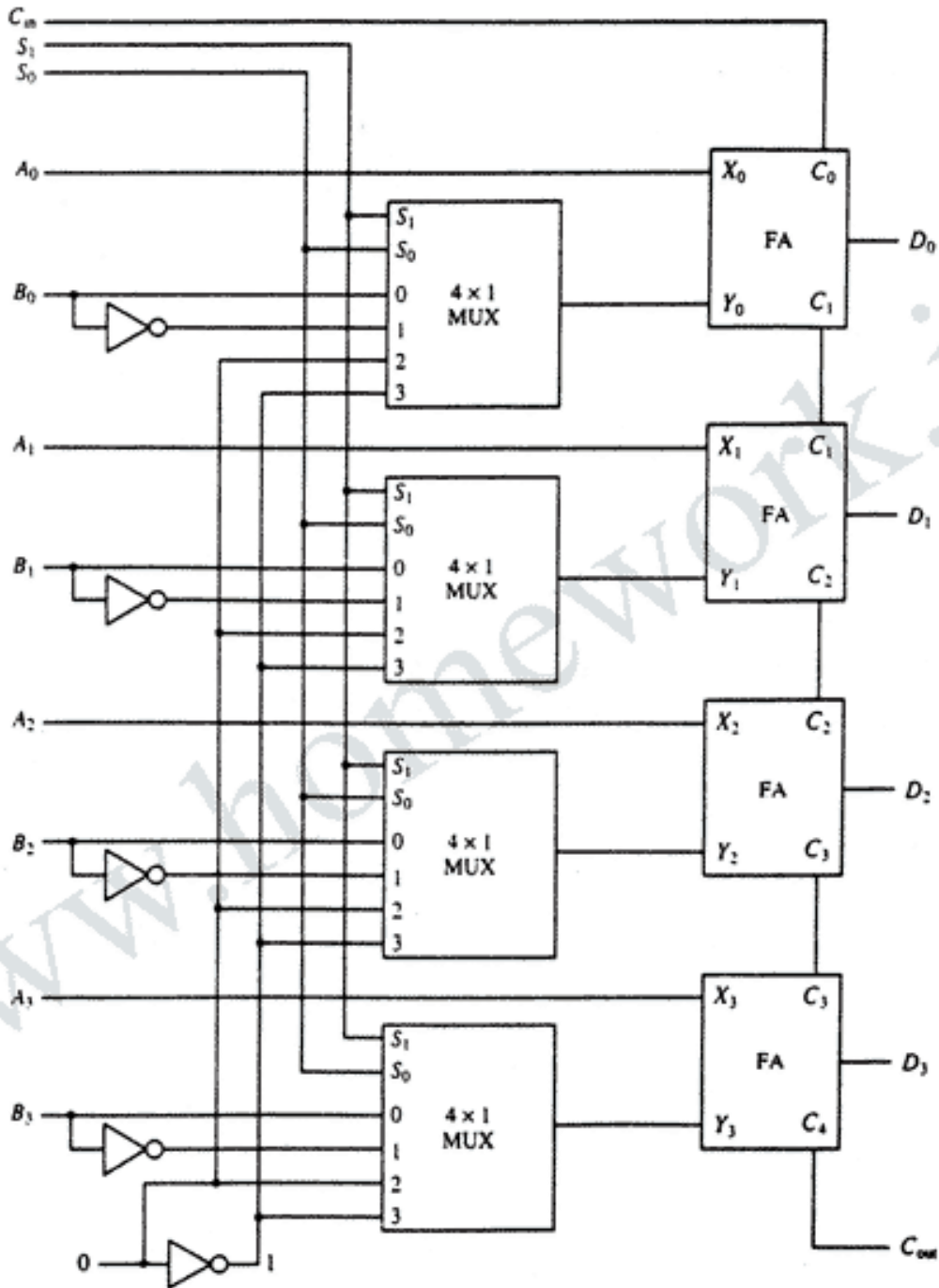
$$D = A + Y + C_{in}$$

که در آن A یک عدد دودویی 4 بیت در ورودی های X، و Y یک عدد دودویی 4 بیت در ورودی های Y جمع کننده دودویی است. C_{in} رقم نقلی ورودی است که می تواند 0 یا 1 باشد. با کنترل مقدار Y توسط دو ورودی S_0 و S_1 با 0 یا 1 کردن C_{in} می توان هشت ریز عمل حسابی لیست شده در جدول ۴-۴ را تولید کرد. اگر $S_1 S_0 = 00$ باشد، مقدار B به ورودی های Y جمع کننده اعمال می شود. اگر $C_{in} = 0$ باشد، خروجی $D = A + B$ است. اگر $C_{in} = 1$ باشد خروجی $D = A + B + 1$ خواهد بود. هر دو حالت ریز عمل جمع را با و یا بدون نقلی ورودی انجام می دهند.

اگر $S_1 S_0 = 01$ باشد، متمم B به ورودی های Y جمع کننده اعمال می گردد. اگر $C_{in} = 1$ باشد سپس $D = A + \bar{B} + 1$ خواهد بود. این عمل A بعلاوه متمم 2 عدد B را که معادل تفریق $A - B$ است تولید می نماید. وقتی که $C_{in} = 0$ باشد سپس $D = A + \bar{B}$ خواهد شد. این عمل معادل با تفریق بدون قرض کردن می باشد، یعنی، $A - B - 1$.

در $S_1 S_0 = 10$ ، از ورودی های B چشم پوشی می شوند، و در عوض، به ورودی های Y تماماً 0 اعمال می گردد. خروجی برابر خواهد بود با $D = A + 0 + C_{in}$. در این حال اگر $C_{in} = 0$ باشد داریم $D = A$ و اگر $C_{in} = 1$ باشد داریم $D = A + 1$. در حالت اول انتقال مستقیمی از ورودی A به خروجی D وجود دارد. در حالت دوم مقدار A به میزان 1 واحد افزایش یافته است.

اگر $S_1 S_0 = 11$ باشد، بداخل ورودی های Y جمع کننده تماماً 1 وارد می شود تا عمل کاهش 1 واحد، وقتی که $C_{in} = 0$ است، یعنی $D = A - 1$ ، صورت گیرد. دلیل این مطلب این است که عددی که تماماً از 1 ساخته شده باشد متمم 2 عدد 1 است (متمم 2 عدد 0001 عدد 1111 است). جمع عدد A با



شکل ۹-۴ مدار حساب 4 بیتی

متمم 2 عدد 1 برابرست با $A-1$. اگر $C_{in} = 1$ باشد، سپس $D=A-1+1=A$ خواهد شد، که انتقال مستقیمی را از ورودی A به خروجی D ایجاد می نماید. توجه کنید که ریز عمل $D=A$ دوبار تولید شده است. بنابراین تنها هفت ریز عمل متمایز از هم در مدار حسابی وجود دارد.

جدول ۴-۴ جدول تابع مدار حساب

انتخاب			ورودی Y	خروجی $D = A + Y + C_n$	ریز عمل
S_1	S_0	C_n			
0	0	0	B	$D = A + B$	جمع
0	0	1	B	$D = A + B + 1$	جمع با نقلی
0	1	0	\overline{B}	$D = A + \overline{B}$	تفریق با فرض
0	1	1	\overline{B}	$D = A + \overline{B} + 1$	تفریق
1	0	0	0	$D = A$	انتقال A
1	0	1	0	$D = A + 1$	افزایش A
1	1	0	1	$D = A - 1$	کاهش A
1	1	1	1	$D = A$	انتقال A

۴-۵ ریز عمل های منطقی

ریز عمل های منطقی اعمال دودویی را برای رشته ای از بیت های ذخیره شده در ثبات ها مشخص می نمایند. اعمال هر بیت ثبات را بطور جداگانه در نظر گرفته و با آنها بعنوان متغیر دودویی رفتار می کند. مثلاً ریز عمل OR انحصاری^۱ را با محتوای دو ثبات R1 و R2 با عبارت زیر نشان می دهیم.

$$P: R1 \leftarrow R1 \oplus R2$$

این عبارت یک ریز عمل منطقی را روی بیت های متمایز ثبات ها مشخص می کند بشرطی که متغیر کنترل $P = 1$ باشد. بعنوان یک مثال عددی، فرض کنید که هر ثبات دارای چهار بیت باشد. اجازه بدهید محتوای R1 برابر 1010 و محتوای R2 برابر 1100 باشد. ریز عمل OR انحصاری مشخص شده بالا محاسبات منطقی زیر را انجام خواهد داد

$$\begin{array}{r} 1010 \quad \text{محتوای R1} \\ 1100 \quad \text{محتوای R2} \\ \hline 0110 \quad \text{محتوای R1 پس از } P=1 \end{array}$$

محتوای R1 پس از اجرای ریز عمل برابرست با OR انحصاری بیت به بیت بر روی جفت ثبات های R2 و ثبات های اولیه R1. ریز عمل های منطقی به ندرت در محاسبات علمی بکار می روند، ولی برای دستکاری داده های دودویی و تصمیم گیری های منطقی بسیار مفیدند.

برای تفکیک ریز عمل های OR و AND سمبل های خاصی بکار خواهند رفت تا از سمبل های بکار رفته از توابع بول متمایز شوند. سمبل ۷ برای ریز عمل OR و سمبل ۸ برای ریز عمل AND بکار خواهد رفت. ریز عمل متمم سازی همان متمم 1 بوده و با یک خط که روی نام ثبات قرار می گیرد

1- Exclusive OR (XOR, EXOR)

شناخته می شود. با بکارگیری سمبل های متفاوت می توان یک ریزعمل منطقی را از یک تابع کنترل (یا تابع بول) تفکیک کرد. دلیل دیگر برای استفاده از دو مجموعه سمبل این است که سمبل + را وقتی بمعنی جمع حسابی بکار می رود، از عمل OR منطقی متمایز نماییم. هر چند + دارای دو مفهوم است، ولی می توان بین آنها با توجه به اینکه سمبل در کجا آمده است تفاوت قائل شد. وقتی سمبل + در یک ریزعمل واقع شود، بمعنی جمع حسابی است. وقتی که در تابع کنترل واقع شود (یا تابع بول)، بمعنی عمل OR است. ما هرگز از این علامت برای ریزعمل OR استفاده نخواهیم کرد. مثلاً در عبارت

$$P + Q : R1 \leftarrow R2 + R3, R4 \leftarrow R5 \vee R6$$

علامت + بین P و Q یک عمل OR بین دو متغیر دودویی یک تابع کنترل است. علامت + در بین R2 و R3 یک ریزعمل جمع را مشخص می نماید. ریزعمل OR بین R5 و R6 با سمبل \vee تعریف شده است.

لیست ریزعمل های منطقی

شانزده عمل منطقی وجود دارد که می توان با دو متغیر دودویی انجام داد. این عمل ها را از روی جدول درستی کلی که می توان با دو متغیر، مطابق جدول ۴-۵، بدست آورد تعریف کرد. در این جدول هر یک از شانزده ستون F_0 تا F_{15} یک جدول درستی را برای هر تابع بول دو متغیر X و Y نمایش می دهند. توجه کنید که تابع از شانزده ترکیب ممکن که برای F می تواند تخصیص یابد بدست آمده است. 16 تابع بول دو متغیر X و Y بصورت جبری در جدول ۴-۶ آورده شده اند. 16 ریزعمل منطقی با جایگزینی متغیر X با محتوای دودویی ثبات A و متغیر Y با محتوای ثبات B حاصل شده اند. دانستن این نکته که توابع بولی لیست شده در اولین ستون جدول ۴-۶ رابطه ای را بین دو متغیر X و Y نشان می دهند، اهمیت دارد. ریزعمل های لیست شده در دومین ستون رابطه ای را بین محتوای دودویی ثبات های A و B بیان می دارند. با هر بیت از ثبات بصورت یک متغیر دودویی رفتار شده و ریزعمل روی رشته ای از بیت های ذخیره شده در ثبات انجام می گیرد.

پیاده سازی سخت افزاری

پیاده سازی سخت افزاری ریزعمل های منطقی مستلزم این است که برای همه بیت ها یا جفت

جدول ۴-۵ جدولهای ارزش برای 16 تابع دو متغیره

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

جدول ۴-۶ شانزده ریزعمل منطقی

تایع بولی	ریزعمل	توضیح
$F_0 = 0$	$F \leftarrow 0$	صفر کردن
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \overline{B}$	
$F_3 = x$	$F \leftarrow A$	انتقال A
$F_4 = x'y$	$F \leftarrow \overline{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	انتقال B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	OR انحصاری
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x + y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	NOR انحصاری
$F_{10} = y'$	$F \leftarrow \overline{B}$	متمم کردن B
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	متمم کردن A
$F_{13} = x' + y$	$F \leftarrow \overline{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow \text{all 1's}$	همه بیت ها در وضعیت 1

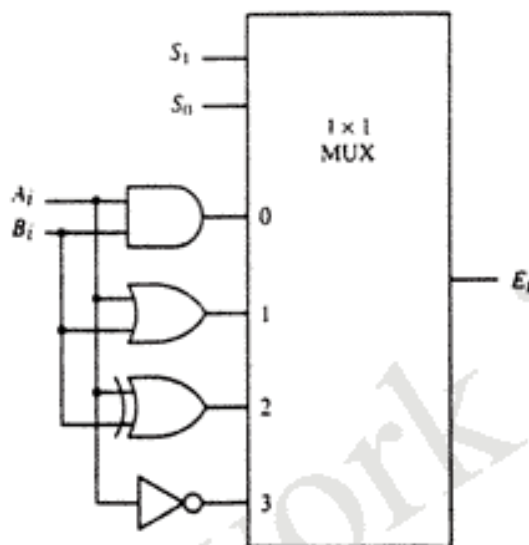
بیت های ثبات ها گیت های منطقی گذاشته شود تا تایع منطقی مورد نظر را اجرا نماید. اگرچه 16 ریزعمل منطقی وجود دارد اغلب کامپیوترها فقط از چهار ریزعمل (AND, OR, XOR, OR) انحصاری) و متمم استفاده می کنند، و سایر ریزعمل ها را نیز می توان از این چهار عمل بدست آورد. شکل ۴-۱۰ یک طبقه از مدار را که چهار ریزعمل اصلی منطقی را اجرا می نماید، نشان می دهد. این طبقه متشکل از چهار گیت و یک مولتی پلکسر است. هر یک از چهار عمل منطقی از طریق یک گیت که منطق لازم را انجام می دهد تولید می شود. خروجی گیت ها به ورودی های داده مولتی پلکسر اعمال می شود. دو ورودی انتخاب S_0 و S_1 یکی از ورودی های داده به مولتی پلکسر را انتخاب کرده و مقدار آن را به خروجی هدایت می نماید. دیاگرام، نمونه ای از طبقات را که با اندیس i مشخص شده نشان می دهد. برای یک مدار منطقی n بیتی، دیاگرام باید n بار برای $i=0, 1, 2, \dots, n-1$ تکرار شود. متغیرهای انتخاب به تمام طبقات اعمال می شوند. جدول تایع شکل ۴-۱۰ (ب) ریزعمل های منطقی حاصل برای هر ترکیبی از متغیرهای انتخاب را لیست نموده است.

چند کاربرد

ریزعمل های منطقی در دستکاری بیت های متمایز یا بخشی از کلمات ذخیره شده در یک ثبات بسیار مفیدند. از این ریزعمل ها می توان برای تغییر مقادیر بیت ها، پاک کردن گروهی از بیت ها، یا وارد کردن بیت های جدید در ثبات استفاده کرد. مثال های زیر چگونگی دستکاری بیت های یک ثبات (فرضاً A) را با ریزعملی که تابعی از بیت های ثبات دیگر است (فرضاً B) نشان می دهد. در یک نمونه از

S_1	S_0	خروجی	عمل
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \bar{A}$	نم

(ب) جدول تابع



(الف) نمودار منطقی

شکل ۱۰-۴ یک طبقه از مدار منطقی

کاربرد، ثبات A می تواند ثبات یک پردازشگر و بیت های ثبات B عملوندی منطقی حاصل از حافظه باشد که در ثبات B قرار گرفته است.

عمل نشانیدن انتخابی 1 هایی را در بیت های ثبات A که متناظرشان در ثبات B برابر 1 هستند، می نشانند. این عمل مکان هایی را که متناظرشان در B برابر 0 باشد تغییر نمی دهد. مثال عددی زیر این عمل را واضحتر بیان می دارد.

$$\begin{array}{r}
 1010 \quad \text{A قبل از عمل} \\
 \underline{1100} \quad \text{B (عملوند منطقی)} \\
 1110 \quad \text{A بعد از عمل}
 \end{array}$$

دو بیت سمت چپ B ، 1 هستند، بنابراین بیت های متناظرشان در A ، 1 خواهند شد. یکی از این دو بیت قبلاً 1 بوده و دیگری از 0 به 1 تبدیل شده است. دو بیت A که نظیرشان در B برابر 0 می باشند بلا تغییر باقی مانده اند. مثال فوق مانند یک جدول درستی است زیرا تمام چهار بیت ترکیب دو متغیر دودویی را داراست. از جدول درستی درمی یابیم که بیت A پس از عمل OR منطقی بیت ها در ثبات B و مقادیر قبلی A بدست می آیند. بنابراین، ریز عمل OR می تواند برای 1 کردن بیت های انتخاب شده یک ثبات بکار روند.

1- Selective set

عمل متمم سازی انتخابی¹ بیت های A را که متناظرشان در B، برابر 1 باشد متمم می نماید. این عمل مکان هایی را که نظیرشان در B برابر 0 باشد تغییر نمی دهد. مثلاً

1010	A قبل از عمل
<u>1100</u>	عملوند منطقی
0110	A بعد از عمل

مجدداً دو بیت منتهالیه سمت چپ B، 1 می باشند، بنابراین بیت های متناظر در A متمم شده اند. این مثال هم مانند یک جدول درستی است که از آن می توان نتیجه گرفت که عمل متمم کردن انتخابی دقیقاً همان ریز عمل OR انحصاری است. بنابراین از ریز عمل OR انحصاری می توان برای متمم کردن انتخابی بیت های یک ثبات استفاده کرد.

عمل پاک کردن انتخابی² فقط بیت هایی از A را، که بیت های متناظر آنها در B مقدار 1 دارد، 0 می کند مثلاً

1010	A قبل از عمل
<u>1100</u>	عملوند منطقی
0010	A بعد از عمل

باز هم دو بیت منتهالیه چپ B مقدار 1 دارند، بنابراین بیت های متناظر آنها در A برابر 0 شده اند. می توان نتیجه گرفت که عمل بولی انجام شده روی بیت ها $A\bar{B}$ است. ریز عمل منطقی مربوطه عبارتست از

$$A \leftarrow A \wedge \bar{B}$$

عمل ماسک³ (پوشش) مشابه عمل پاک کردن انتخابی است بجز اینکه بیت های A فقط در صورتی که نظیرشان در B مقدار 0 باشد، پاک می شوند. عمل ماسک، همانطور که از مثال عددی زیر دیده می شود، یک ریز عمل AND است.

1010	A قبل از عمل
<u>1100</u>	B عملوند منطقی
1000	A پس از ماسک

دو بیت منتهالیه سمت راست A پاک شده اند زیرا بیت های متناظرشان در B، 0 هستند. دو بیت سمت چپ بلا تغییر مانده اند زیرا بیت های نظیرشان در B، 1 می باشند. عمل ماسک از عمل پاک کردن انتخابی مناسبتر است زیرا اکثر کامپیوترها دارای دستور AND می باشند، و آنهایی که دستوراتی را برای اجرای ریز عمل پاک کردن انتخابی دارا هستند اندکند.

1- Selective Complement

2- Selective Clear

3- Mask

عمل درج¹، یک مقدار جدیدی را در گروهی وارد می‌کند. این عمل ابتدا با پوشش بیت‌ها و سپس OR کردن آنها با مقدار مورد نظر انجام می‌شود. مثلاً فرض کنید که ثبات A حاوی هشت بیت 01101010 باشد. برای جایگزینی چهار بیت سمت چپ با 1001، ابتدا چهار بیت ناخواسته را ماسک می‌کنیم:

$$\begin{array}{r} 0110 \ 1010 \\ 0000 \ 1111 \\ \hline 0000 \ 1010 \end{array}$$

A قبل از عمل

B (ماسک)

A پس از عمل

و سپس مقدار جدید را درج می‌نمائیم

$$\begin{array}{r} 0000 \ 1010 \\ 1001 \ 0000 \\ \hline 1001 \ 1010 \end{array}$$

A قبل از عمل

B (درج)

A بعد از عمل

عمل ماسک یک ریز عمل AND و عمل درج یک ریز عمل OR می‌باشد. عمل پاک کردن²، دو کلمه A و B را مقایسه کرده و اگر دو عدد مساوی باشند نتیجه تمام 0 را تولید می‌نماید. این عمل توسط ریز عمل OR انحصاری تحقق می‌یابد و بوسیله مثال زیر نشان داده شده است.

$$\begin{array}{r} 1010 \quad A \\ 1010 \quad B \\ \hline 0000 \quad A \leftarrow A \oplus B \end{array}$$

اگر A و B مساوی باشند، بیت‌های نظیر به نظیر هر دو یا 0 یا 1 می‌باشند. در هر یک از حالات عمل OR انحصاری، 0 تولید خواهد کرد. نتیجه تمام 0 سپس برای تعیین مساوی بودن دو عدد چک می‌شود.

۴-۶ ریز عمل‌های شیفت

ریز عمل‌های شیفت برای انتقال یا جابجایی سری داده‌ها بکار گرفته می‌شوند. آنها همچنین همراه با عمل‌های حسابی، منطقی و سایر اعمال داده پردازشی بکار می‌روند. محتوای یک ثبات می‌تواند به چپ یا به راست شیفت پیدا کند. در همان زمانی که بیت‌ها شیفت داده می‌شوند، اولین فلیپ فلاپ اطلاعات دودویی خود را از ورودی سری دریافت می‌کند. در حین شیفت به چپ، ورودی سری یک بیت را به سمت راست ترین مکان منتقل می‌نماید. در حین عمل شیفت به راست، ورودی سری یک بیت را به سمت چپ‌ترین مکان انتقال می‌دهد. اطلاعاتی که از طریق ورودی سری منتقل می‌گردد تعیین کننده نوع شیفت است. سه نوع شیفت وجود دارد: منطقی، چرخشی و حسابی.

شیفت منطقی^۱ مقدار 0 را از طریق ورودی سری انتقال می دهد. ما سمبل shl و shr را بترتیب برای ریز عمل های شیفت به چپ و شیفت به راست برمی گزینیم. مثلاً

$$R1 \leftarrow \text{shl } R1$$

$$R2 \leftarrow \text{shr } R2$$

دو نمونه ریز عمل می باشند که محتوای ثبات R1 را یک بیت به چپ و محتوای ثبات R2 را یک بیت به راست شیفت می دهند. سمبل ثبات در هر دو طرف فلش باید یکسان باشند. فرض بر این است که حین شیفت، بیت انتقالی به مکان انتهایی از طریق ورودی سری، 0 باشد.

شیفت چرخشی (که عمل چرخش^۲ نیز خوانده می شود) بیت های ثبات را از طریق دو انتها بدون از دست دادن هرگونه اطلاعات می چرخاند. این عمل با اتصال خروجی سری به ورودی سری ثبات تحقق می یابد. ما سمبل های cil و cir را بترتیب برای چرخش به چپ و چرخش به راست بکار خواهیم برد. سمبل های بکار رفته و ریز عمل های شیفت در جدول ۴-۷ نشان داده شده اند.

شیفت حسابی^۳ ریز عملی است که یک عدد دودویی علامت دار را به چپ یا راست شیفت می دهد. شیفت حسابی به چپ، یک عدد دودویی علامت دار را در 2 ضرب می کند. یک شیفت حسابی به راست نیز عدد را بر 2 تقسیم می نماید. شیفت های حسابی نباید بیت علامت را تغییر دهند زیرا وقتی عدد را در 2 ضرب یا تقسیم می کنیم علامت همچنان باقی می ماند. سمت چپ ترین بیت در ثبات بیت علامت را نگه می دارد، و بقیه بیت ها عدد را حفظ می کنند. بیت علامت برای اعداد مثبت، 0 و برای منفی، 1 است. اعداد منفی در فرم متمم 2 هستند. شکل ۴-۱۱ نمونه ای از یک ثبات n بیت را نشان می دهد. بیت R_{n-1} در سمت چپ ترین مکان بیت علامت را نگه می دارد. بیت R_{n-2}، با ارزش ترین بیت و R₀ کم ارزش ترین بیت است. شیفت به راست حسابی بیت علامت را عوض نمی کند و همه بیت ها (از جمله علامت) را به راست شیفت می دهد. بنابراین R_{n-1} تغییر نمی کند، R_{n-2} بیت R_{n-1} را دریافت می نماید و برای سایر بیت های ثبات نیز به همین ترتیب. بیت واقع در R₀ از دست می رود.

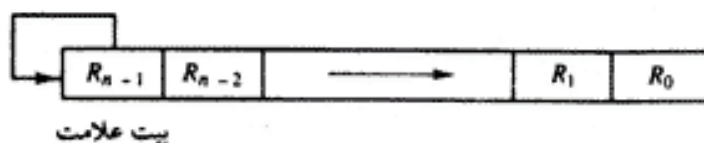
جدول ۴-۷ ریز عمل های شیفت

سمبل نشان دهنده	توضیح
$R \leftarrow \text{shl } R$	شیفت ثبات R به چپ
$R \leftarrow \text{shr } R$	شیفت ثبات R به راست
$R \leftarrow \text{cil } R$	شیفت چرخشی ثبات R به چپ
$R \leftarrow \text{cir } R$	شیفت چرخشی ثبات R به راست
$R \leftarrow \text{ashl } R$	شیفت حسابی ثبات R به چپ
$R \leftarrow \text{ashr } R$	شیفت حسابی ثبات R به راست

1- Logical Shift

2- Rotate

3- Arithmetic Shift



شکل ۱۱-۴ شیفت حسابی به راست

شیفت حسابی به چپ یک 0 وارد R_0 می نماید، و کلیه بیت های دیگر را به چپ شیفت می دهد. بیت اولیه R_{n-1} از دست رفته و با بیت R_{n-2} جایگزین می شود. اگر بیت واقع در R_{n-1} پس از شیفت عوض شود، علامت معکوس شده است. این هنگامی رخ می دهد که ضرب در 2 سبب سرریز گردد. سرریز در شیفت به چپ هنگامی رخ می دهد که قبل از شیفت R_{n-1} و R_{n-2} مساوی نباشد. یک فلیپ فلاپ سرریز V_s برای کشف یک سرریز حاصل از شیفت حسابی به چپ می تواند مورد استفاده قرار گیرد.

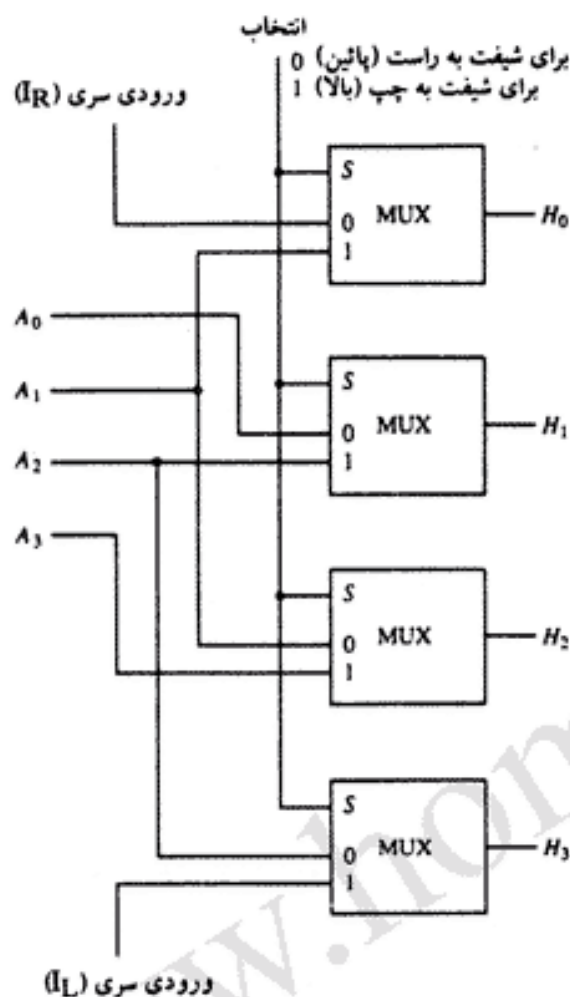
$$V_s = R_{n-1} \oplus R_{n-2}$$

اگر $V_s = 0$ باشد، سرریز وجود ندارد، ولی اگر $V_s = 1$ گردد، سرریز وجود داشته و پس از شیفت علامت عوض خواهد شد. V_s باید با همان پالس ساعت شیفت ثبات وارد فلیپ فلاپ سرریز گردد.

پیاده سازی سخت افزاری

یکی از انتخاب های ممکن برای واحد شیفت، ثبات شیفت دوطرفه با بارشدن موازی است (شکل ۹-۲ ملاحظه شود). اطلاعات را می توان به صورت موازی وارد ثبات نمود و سپس آنرا به چپ یا راست شیفت داد. در این نوع آرایش، یک پالس ساعت برای بارشدن داده بداخل ثبات مورد نیاز است و پالس دیگری برای شیفت دادن لازم است. در یک واحد پردازنده با ثبات های متعدد، بهتر است برای کارایی بیشتر، عمل شیفت را با مدار ترکیبی پیاده سازی کنیم. در این روش محتوای ثباتی که قرار است شیفت کند ابتدا روی گذرگاه مشترک قرار گرفته و خروجی آن به یک شیفت دهنده ترکیبی متصل می شود، و سپس عدد شیفت داده شده به ثبات بازگردانده می شود. این نیز تنها یک پالس ساعت برای بارکردن مقدار شیفت یافته بداخل ثبات نیاز دارد.

یک شیفت دهنده ترکیبی را می توان مانند شکل ۱۲-۴ با مولتی پلکسر ساخت. شیفت دهنده 4 بیتی دارای چهار ورودی داده A_0 الی A_3 و چهار خروجی داده H_0 الی H_3 است. دو ورودی سری نیز وجود دارد، یکی برای شیفت به چپ (L) و دیگری برای شیفت به راست (R). وقتی که ورودی انتخاب وجود دارد، $S=0$ باشد، داده ورودی به راست شیفت داده می شود (در دیاگرام به سمت پائین). اگر $S=1$ باشد، داده ورودی به چپ شیفت می یابد (به سمت بالای دیاگرام). با n ورودی و خروجی داده به n مولتی پلکسر نیاز دارد. دو ورودی سری بوسیله مولتی پلکسر دیگری قابل کنترل اند تا سه نوع شیفت ممکن را فراهم آورند.



جدول تابع

انتخاب	خروجی			
	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

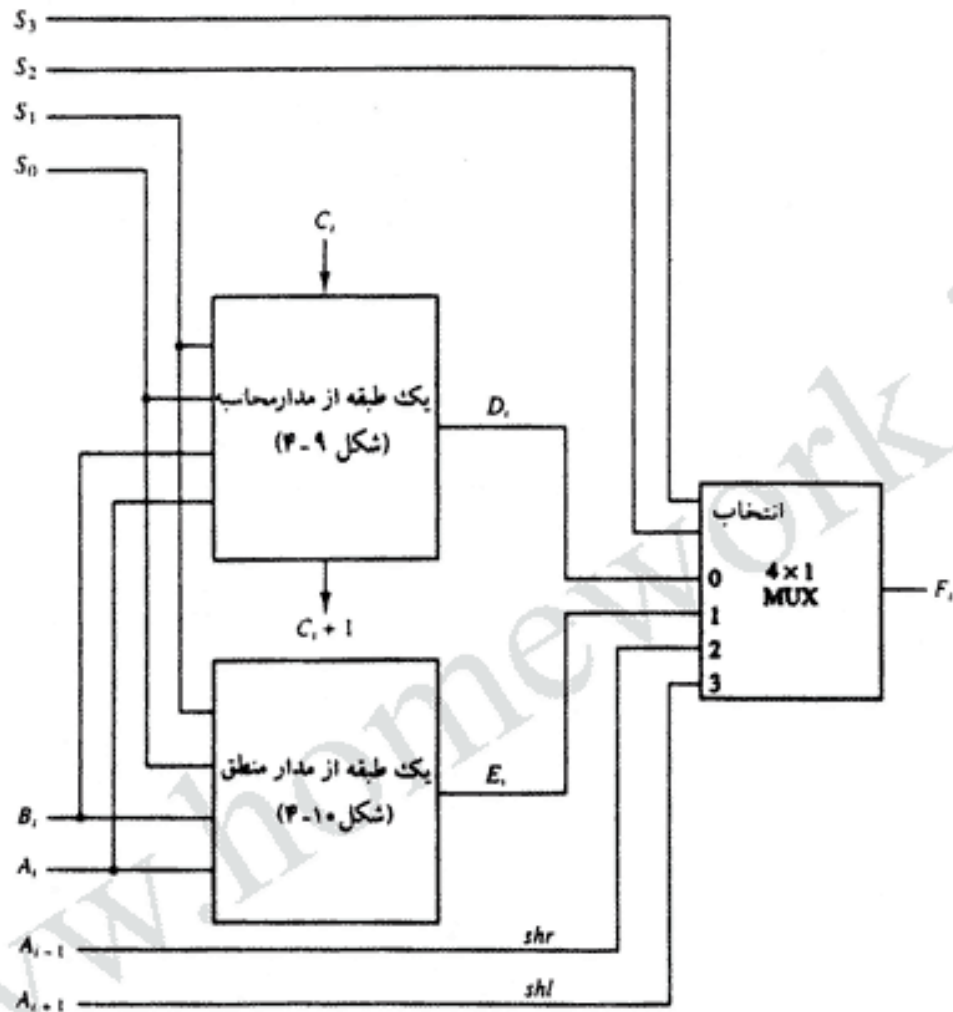
شکل ۱۲-۴ شیفت ترکیبی 4 بیتی

۴-۷ واحد حساب، منطق و شیفت

در سیستم‌های کامپیوتری به جای اینکه ثباتهای مختلف ریزعملها را مستقیماً اجزا کنند، از تعدادی ثبات ذخیره سازی استفاده می‌شود که به یک واحد عملیاتی مشترک بنام واحد حساب و منطق^۱ متصل‌اند. برای اجرای یک ریزعمل، محتوای ثبات خاصی در ورودی ALU مشترک قرار می‌گیرد. ALU عملی را انجام داده و سپس نتیجه را به ثبات مقصد ارسال می‌نماید. چون ALU یک مدار ترکیبی است بنابراین عمل انتقال ثبات از ثبات مبدأ به ALU و از آنجا به ثبات مقصد در یک پریود پالس ساعت صورت می‌گیرد. ریز عمل شیفت اغلب در یک واحد جدا انجام می‌شود، و گاهی نیز واحد شیفت جزئی از ALU می‌باشد.

مدارهای حساب، منطق و شیفت که در بخش قبل معرفی شدند را می‌توان بصورت ALU واحد با متغیرهای انتخاب مشترک با هم ترکیب کرد. یک طبقه از واحد حساب، منطق و شیفت در شکل ۱۳-۴

1- Arithmetic and Logic Unit



شکل ۴-۱۳ یک طبقه از واحد حساب، منطق و شیفت

دیده می شود. زیرنویس i نمایانگر یک طبقه نمونه است. ورودی های A_i و B_i به هر دو مدار حساب و منطق اعمال شده اند. یک ریز عمل خاصی با ورودی های S_0 و S_1 انتخاب می شوند. یک مولتی پلکسر 4×1 در خروجی، خروجی حسابی E_i یا منطقی H_i را برمیگزیند. داده ها در این مولتی پلکسر با S_2 و S_3 انتخاب می شوند. دو ورودی دیگر داده به مولتی پلکسر، A_{i-1} را برای عمل شیفت به راست و A_{i+1} را برای عمل شیفت به چپ دریافت می کند. توجه کنید که دیاگرام تنها یک طبقه را نشان می دهد. برای یک ALU با ساختمان n بیتی مدار شکل ۴-۱۳ باید n بار تکرار شود. خروجی نقلی C_{i+1} برای یک طبقه ALU مفروض باید به رقم نقلی C_i طبقه سری بعدی وصل شود. رقم نقلی ورودی به اولین طبقه، C_{in} است که یک متغیر انتخاب برای اعمال حسابی خواهد بود.

مداری که یک طبقه اش در شکل ۴-۱۳ مشخص شد. یک عمل حسابی هشت بیتی، چهار عمل

منطقی، و دو عمل شیفت را اجرا می نماید. هر عمل توسط پنج متغیر S_3, S_2, S_1, S_0 و C_{in} انتخاب می گردد. رقم نقلی ورودی فقط برای انتخاب اعمال حسابی بکار می رود. جدول ۴-۸ چهارده عمل ALU را لیست نموده است. هشتای اول اعمال حسابی هستند. (جدول ۴-۴ را ببینید) و توسط $S_3S_2=00$ انتخاب می شوند. چهار عمل بعدی منطقی هستند (جدول ۴-۱۰ را ببینید) و بوسیله $S_3S_2=01$ انتخاب می گردند. ورودی نقلی در طول اعمال منطقی هیچ اثری ندارد و با علامت بی اهمیت X مشخص شده است. دو عمل آخر اعمال شیفت هستند که با 11, 10, $S_3S_2=10$ انتخاب می شوند. سه ورودی انتخاب دیگر تأثیری بر روی شیفت ندارند.

جدول ۴-۸ جدول تابع برای واحد حساب، منطقی و شیفت

انتخاب کننده عمل					عمل	تابع
S_3	S_2	S_1	S_0	C_{in}		
0	0	0	0	0	$F = A$	انتقال
0	0	0	0	1	$F = A + 1$	افزایش
0	0	0	1	0	$F = A + B$	جمع
0	0	0	1	1	$F = A + B + 1$	جمع با رقم نقلی
0	0	1	0	0	$F = A + \bar{B}$	تفریق با فرض
0	0	1	0	1	$F = A + \bar{B} + 1$	تفریق
0	0	1	1	0	$F = A - 1$	کاهش A
0	0	1	1	1	$F = A$	انتقال A
0	1	0	0	x	$F = A \wedge B$	AND
0	1	0	1	x	$F = A \vee B$	OR
0	1	1	0	x	$F = A \oplus B$	XOR
0	1	1	1	x	$F = \bar{A}$	منم کردن A
1	0	x	x	x	$F = shr A$	شیفت A به راست و به داخل F
1	1	x	x	x	$F = shl A$	شیفت A به چپ و به داخل F

مسائل

۴-۱) بلاک دیاگرام سخت افزاری مشابه شکل ۴-۲ (الف) که عبارت انتقال ثبات زیر را پیاده سازی کنید را رسم کنید:

$$yT_2: R2 \leftarrow R1, R1 \leftarrow R2$$

۴-۲) خروجی چهار ثبات $R0, R1, R2$ و $R3$ از طریق یک مولتی پلکسر 4×1 به ورودی های ثبات پنجم $R5$ وصل شده اند. هر ثبات هشت بیتی است. انتقال لازم توسط چهار متغیر زمان بندی T_0 تا T_3 مطابق زیر تعیین می شود

$$T_0: R5 \leftarrow R0$$

$$T_1 : R5 \leftarrow R1$$

$$T_2 : R5 \leftarrow R2$$

$$T_3 : R5 \leftarrow R3$$

متغیرهای زمانی دو به دوازهم جدا هستند. یعنی در هر زمان معین فقط یک متغیر 1 است، در حالی که سه متغیر دیگر 0 هستند. بلاک دیاگرامی رسم کنید که نشان دهنده سخت افزار پیاده سازی انتقال ثبات باشد. اتصالات لازم برای چهار متغیر زمانبندی به خطوط انتخاب مولتی پلکسر و ورودی بازکردن ثبات R5 را نیز نشان دهید.

۳-۴ عبارت کنترل شرطی زیر را بوسیله دو عبارت انتقال ثبات با توابع کنترل نشان دهید.

$$\text{If}(P=1) \text{ then } (R1 \leftarrow R2) \text{ else if } (Q=1) \text{ then } (R1 \leftarrow R3)$$

۴-۴ برای اینکه سیستم گذرگاه شکل ۳-۴ اطلاعات را از هر ثبات به هر ثبات دیگر منتقل کند چه باید کرد؟ خصوصاً اتصالاتی را که باید برای تهیه یک مسیر از خروجی های ثبات C به ورودی های ثبات A وصل نمود نشان دهید.

۴-۵ یک سیستم گذرگاه مانند آنچه در شکل ۳-۴ دیده شد رسم کنید، ولی از بافرهای سه حالت و دیگر در بجای مولتی پلکسر استفاده نمائید.

۴-۶ یک کامپیوتر دیجیتال دارای سیستم گذرگاه مشترک برای 16 ثبات 32 بیتی است. گذرگاه با مولتی پلکسر ساخته شده است.

الف) در هر مولتی پلکسر چند ورودی انتخاب وجود دارد.

ب) اندازه مولتی پلکسر چیست

ج) چند مولتی پلکسر در گذرگاه است

۴-۷ عبارات زیر انتقال در یک حافظه را مشخص می کنند. در هر حالت عمل حافظه را توضیح دهید.

$$\text{الف) } R2 \leftarrow M[AR] \quad \text{ب) } M[AR] \leftarrow R3 \quad \text{ج) } R5 \leftarrow M[R5]$$

۴-۸ بلاک دیاگرام سخت افزاری را رسم کنید که عبارات زیر را پیاده سازی کند.

$$x+yZ : AR \leftarrow AR+BR$$

که AR و BR دو ثبات n بیت و x و y و Z متغیرهای کنترل اند. برای تابع کنترل گیت ها را نیز نشان دهید.

(بیاد داشته باشید که سمبل + در بخش کنترل عمل OR و در ریز عمل جمع بمعنی بعلاوه است.)

۴-۹ سخت افزاری را نشان دهید که عبارت زیر را پیاده سازی کند. برای تابع کنترل گیت ها و برای

شمارنده دودویی با ورودی فعال ساز (تواناساز) شمارش، بلاک دیاگرام را رسم کنید.

$$xyT_0 + T_1 + y' T_2 : AR \leftarrow AR+1$$

۴-۱۰ عبارت انتقال ثبات زیر را برای دو ثبات 4 بیتی R_1 و R_2 ملاحظه کنید

$$xT: R_1 \leftarrow R_1 + R_2$$

$$x'T: R_1 \leftarrow R_2$$

هر بار که $T=1$ است، اگر $x=1$ باشد. محتوای R_2 به R_1 اضافه می شود و اگر $x=0$ باشد R_2 به R_1 انتقال می یابد. دیاگرامی رسم کنید که پیاده سازی سخت افزاری دو عبارت را نشان دهد. برای دو ثبات 4 بیتی، یک جمع کننده 4 بیت، و یک مولتی پلکسر چهارتایی 1:2 که ورودی ها R_1 را انتخاب می کند از بلاک دیاگرام استفاده کنید. در دیاگرام نشان دهید که متغیرهای کنترلی x و T چگونه ورودی های مولتی پلکسر و ورودی بار کردن ثبات R_1 را انتخاب می کنند.

۴-۱۱ با استفاده از یک شمارنده 4 بیتی با بار شدن موازی مانند شکل ۱۱-۲ و یک جمع کننده 4 بیتی مانند شکل ۶-۴، یک بلاک دیاگرام رسم کرده و چگونگی پیاده سازی عبارت های زیر را نشان دهید.

$$x: R_1 \leftarrow R_1 + R_2 \quad \text{R}_2 \text{ را به } R_1 \text{ اضافه کن}$$

$$x'y: R_1 \leftarrow R_1 + 1 \quad \text{R}_1 \text{ را افزایش بده}$$

که R_1 یک شمارنده با بار شدن موازی و R_2 یک ثبات 4 بیت است.

۴-۱۲ مدار جمع - تفریق کننده شکل ۷-۴ دارای مقادیر زیر برای ورودی مد M و ورودی های داده A و B می باشد. در هر حالت، مقادیر خروجی S_0, S_1, S_2, S_3 و C_4 را معین کنید.

	M	A	B
الف	0	0111	0110
ب	0	1000	1001
ج	1	1100	1000
د	1	0101	1010
ه	1	0000	0001

۴-۱۳ یک مدار ترکیبی کاهش گر چهاربیت با چهار مدار تمام جمع کننده بسازید.

۴-۱۴ فرض کنید که مدار چهاربیت شکل ۹-۴ در یک مدار مجتمع قرار گرفته باشد. اتصالات لازم برای دو IC از این نوع را برای ساختن یک مدار حسابی 8 بیتی نشان دهید.

۴-۱۵ یک مدار حسابی با یک متغیر انتخاب s و دو خط ورودی n بیتی A و B طراحی کنید. مدار چهار عمل حسابی زیر را با توجه به نقلی ورودی C_{in} تولید می کند. بلاک دیاگرام را برای دو طبقه اول مدار رسم کنید.

S	$C_{in} = 0$	$C_{in} = 1$
0	$D=A+B$ (جمع)	$D = A+1$ (افزایش)
1	$D=A-1$ (کاهش)	$D = A+B+1$ (تفریق)

۴-۱۶ مدار ترکیبی انتخاب و تولید هر یک از 16 تابع منطقی جدول ۴-۵ را رسم کنید.

۴-۱۷ یک مدار دیجیتال که چهار عمل منطقی OR انحصاری، NOR انحصاری، NOR و NAND را انجام دهد طراحی کنید. دو متغیر انتخاب بکار برید. دیاگرام منطقی یک طبقه نمونه را نشان دهید.

۴-۱۸ ثبات A عدد هشت بیتی 11011001 را حفظ می کند. عملوند B و ریز عمل منطقی لازم را برای تغییر A به هر یک از حالات زیر معین کنید

الف) 01101101 ب) 11111101

۴-۱۹ ثبات های 8 بیتی AR، BR، CR و DR ابتدا دارای مقادیر زیرند

$$AR = 11110010$$

$$BR = 11111111$$

$$CR = 10111001$$

$$DR = 11101010$$

محتوای هشت بیتی هر یک از ثبات ها را پس از اجرای هر یک از رشته ریز عمل ها مشخص کنید.

$$AR \leftarrow AR + BR \quad \text{BR را با هم جمع کن}$$

$$CR \leftarrow CR \wedge DR, \quad BR \leftarrow BR + 1 \quad \text{DR و CR راه AND کن، BR را افزایش بده}$$

$$AR \leftarrow AR - CR \quad \text{CR را از AR کم کن}$$

۴-۲۰ یک ثبات هشت بیتی حاوی عدد دودویی 10011100 است. پس از یک شیفت به راست حسابی

مقدار ثبات چقدر است؟ با شروع از مقدار اولیه 10011100، مقدار ثبات را پس از یک شیفت به

چپ حسابی معین کنید، و بگوئید آیا سرریز وجود دارد؟

۴-۲۱ با شروع از مقدار اولیه $R=11011101$ رشته مقادیر دودویی را در R پس از یک شیفت به چپ

منطقی و بدنبال آن یک شیفت به راست چرخشی، و سپس با یک شیفت به راست منطقی و

نهایتاً یک شیفت به چپ چرخشی تعیین کنید.

۴-۲۲ مقدار H در شکل ۴-۱۲ چیست بشرطی که $I_L=0$ و $I_R=1$ ، $S=1$ ، $A=1001$ باشد؟

۴-۲۳ چه چیزی در عبارات انتقال ثبات زیر غلط است

yT: $R1 \leftarrow R2, R1 \leftarrow R3$ (ب)

xT: $AR \leftarrow \overline{AR}, AR \leftarrow 0$ (الف)

zT: $PC \leftarrow AR, PC \leftarrow PC + 1$ (ج)

سازمان و طراحی یک کامپیوتر پایه



- | | |
|------|------------------------------|
| ۵-۱ | کدهای دستورالعملها |
| ۵-۲ | ثباتهای کامپیوتر |
| ۵-۳ | دستورالعملهای کامپیوتر |
| ۵-۴ | زمانبندی و کنترل |
| ۵-۵ | سیکل دستورالعمل |
| ۵-۶ | دستورالعملهای ارجاع به حافظه |
| ۵-۷ | ورودی - خروجی و وقفه |
| ۵-۸ | تشریح کامل کامپیوتر |
| ۵-۹ | طراحی کامپیوتر پایه |
| ۵-۱۰ | طراحی مدار منطقی انباره |

۵-۱ کدهای دستورالعملها

در این فصل یک کامپیوتر پایه^۱ را معرفی می‌کنیم و نشان خواهیم داد که چگونه می‌توان عملکرد آن را توسط عبارات انتقال ثبات مشخص نمود. سازمان کامپیوتر بوسیله ثبات های داخلی اش، زمانبندی و ساختار کنترل، و مجموعه دستوراتی که به کار می‌برد تعریف می‌گردد. پس از آن طراحی کامپیوتر مشروحاً انجام شده است. هرچند که کامپیوتر پایه‌ای که در این فصل ارائه شده در مقایسه با کامپیوترهای تجاری بسیار کوچک است، ولی مزیت سادگی آن ما را در ارائه روند طراحی بدون برخورد با اشکالات متعدد، قادر می‌سازد.

سازمان داخلی یک سیستم دیجیتال با رشته ریز عملهایی^۲ که روی داده‌های ذخیره شده در ثباتهایش انجام می‌دهد تعریف می‌شود. کامپیوتر دیجیتال همه منظوره^۳ قادر به اجرای انواع ریز عملها بوده و بعلاوه می‌توان اجرای عملیاتی با توالی خاص را به آن دیکته کرد. کاربر یک کامپیوتر می‌تواند کنترل پردازش را با استفاده از یک برنامه بدست بگیرد. برنامه هم مجموعه‌ای از دستورالعملهاست که اعمال، عملوندها و توالی رخداد پردازش را مشخص می‌کند. کار پردازش داده را ممکن است بوسیله برنامه جدیدی با دستورات متفاوت و یا با همان دستورات ولی با داده‌های مختلف تغییر داد.

-
- 1- Basic Computer 2- Microoperation
3- General Purpose Digital Computer

دستورالعمل کامپیوتر یک کد دودویی است که رشته‌ای از ریزاعمال را برای کامپیوتر مشخص می‌کند. کدهای دستورات همراه با داده‌هایشان در حافظه ذخیره می‌شوند. کامپیوتر هر دستور را از حافظه خوانده و آن را در یک ثبات کنترل قرار می‌دهد. پس از آن واحد کنترل کد دودویی دستورالعمل را تفسیر می‌کند و بدنبال آن با صادر کردن رشته‌ای از ریزعمل‌ها آن را اجرا می‌نماید. هر کامپیوتر مجموعه دستورالعمل‌های خاص خود را دارد. توانایی در ذخیره و اجرای دستورات، یا مفهوم توانایی برنامه ذخیره شده، مهمترین خاصیت یک کامپیوتر همه منظوره است.

یک کد دستورالعمل مجموعه‌ای از بیت‌هاست که انجام یک عمل خاص را به کامپیوتر فرمان می‌دهد. این کد معمولاً به دو قسمت تقسیم می‌شود و هر یک تفسیر خاص خود را داراست. اصلی ترین بخش کد دستور بخش عمل آنست. کد عمل در یک دستور گروهی از بیت‌هاست که اعمالی مانند جمع، تفریق، ضرب، شیفت و متمم سازی را تعریف می‌نمایند. تعداد بیت‌های مربوط به عمل یک دستور به تعداد کل اعمال موجود در کامپیوتر بستگی دارد. برای 2^n عمل مستقل از هم (یا کمتر)، کد عمل حداقل n بیتی است. بعنوان توضیحی بیشتر در این مورد، کامپیوتری را با 64 عمل جدا از هم در نظر بگیرید که یکی از آنها عمل ADD باشد. کد عمل شش بیتی و ترکیب 110010 به ADD اختصاص یافته است. وقتی که این کد در واحد کنترل دیکد شود، کامپیوتر سیگنال‌های کنترل خواندن عملوند را از حافظه صادر و آنرا با یک ثبات پردازشگر جمع می‌نماید.

در این مقطع ما باید ارتباط بین یک عمل و یک ریزعمل را در یک کامپیوتر بدانیم. عمل بخشی از دستور ذخیره شده در حافظه است. این بخش که یک کد دودویی است انجام عملی خاص را به کامپیوتر دستور می‌دهد. واحد کنترل دستورالعمل را از حافظه دریافت و بیت‌های عمل را تفسیر می‌کند. سپس رشته‌ای از سیگنال کنترل را برای آغاز اجرای ریزاعمال در ثبات‌های داخلی کامپیوتر صادر می‌نماید. واحد کنترل برای هر کد عمل، رشته‌ای از ریزاعمال لازم را برای تحقق سخت افزاری عمل صادر می‌کند. به همین علت گاهی یک کد عمل را درشت عمل¹ (کلان) هم می‌گویند زیرا مجموعه‌ای از ریزعمل‌ها را مشخص می‌نماید.

بخش عمل در یک کد دستورالعمل، کاری را که قرار است انجام بگیرد معین می‌کند. این عمل باید بر روی داده‌هایی که در ثبات‌های پردازشگر و یا حافظه ذخیره شده‌اند صورت گیرد. بنابراین در کد دستور نه تنها عمل بلکه ثبات و یا حافظه‌هایی که عملوندها در آنها می‌توانند یافت شوند و یا در آنها ذخیره شوند نیز مشخص می‌شود. کلمات حافظه توسط آدرسشان در دستورالعمل معین می‌گردند. ثبات‌های پردازشگر نیز با تخصیص k بیت دودویی دیگر که 2^k ثبات را مشخص می‌کند تعیین می‌شوند. برای تخصیص یک کد دودویی برای دستورات انتخاب‌های متعددی وجود دارد و هر کامپیوتر ساختار کد دستورالعمل خاص خود را داراست. قالب کدهای دستور توسط طراحان که معماری آن را مشخص می‌نمایند تعیین می‌شود. در این فصل ما قالب کدهای خاصی را برای تشریح سازمان و طراحی کامپیوترهای دیجیتال برمی‌گزینیم.

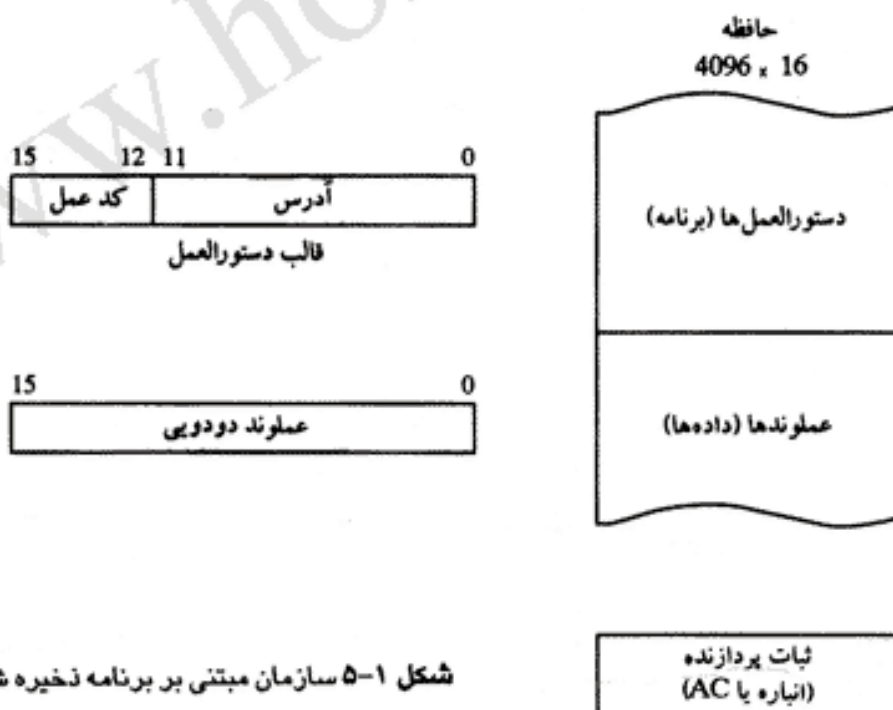
1- Macrooperation

سازمان مبتنی بر برنامه ذخیره شده

ساده ترین راه سازماندهی یک کامپیوتر داشتن یک ثبات پردازنده و قالب کد دستورالعملی متشکل از دو بخش است. قسمت اول عملی را که انجام خواهد شد مشخص می کند و قسمت دوم آدرس را معین می نماید. آدرس به کنترل، محل عملوند^۱ را در حافظه نشان می دهد. این عملوند از حافظه خوانده شده و بر روی آن بعنوان داده ذخیره شده در ثبات پردازشگر عمل می شود.

شکل ۱-۵ این نوع سازماندهی را نشان می دهد. دستورالعملها در یک بخش دیگری ذخیره می شوند. برای واحد حافظه ای با 4096 کلمه به 12 بیت برای مشخص کردن آدرس نیاز است، زیرا $2^{12} = 4096$ می باشد. اگر هر کد دستورالعمل در یک کلمه 16 بیتی حافظه جای گیرد، چهار بیت برای کد عمل لازم است تا یکی از 16 عمل ممکن را مشخص نماید و نیز 12 بیت برای مشخص کردن آدرس عملوند خواهیم داشت. کنترل یک کد 16 بیتی دستورالعمل را از آن قسمت از حافظه که برنامه در آن است می خواند. سپس از بخش 12 بیتی آدرس در دستورالعمل برای خواندن یک عملوند 16 بیتی از ناحیه داده ها استفاده می شود. آنگاه عملی را که کد عمل مشخص می نماید اجرا می کند. در کامپیوترهایی که فقط یک ثبات پردازش دارند نام انباره^۲ را به آن اختصاص می دهند. اعمال در این کامپیوتر معمولاً بر روی عملوند و محتویات انباره AC صورت می گیرد.

اگر عمل مربوط به یک کد دستورالعمل نیاز به عملوند نداشته باشد بیت های مربوطه برای اهداف



شکل ۱-۵ سازمان مبتنی بر برنامه ذخیره شده

1- Operand

2-Accumulator

دیگری بکار گرفته می شوند. مثلاً، اعمالی مانند پاک کردن¹ AC، متمم کردن AC و افزایش AC روی داده ذخیره شده در AC عمل می کنند. این اعمال به عملوندی از حافظه نیازی ندارند. برای این گونه اعمال، بخش دوم کد دستور (بیت های 0 تا 11) برای تعیین آدرس حافظه مورد نیاز نیستند و می توانند برای اعمال دیگری در کامپیوتر بکار روند.

آدرس غیرمستقیم

گاهی اوقات مناسبتر است از بیت های آدرس دستورالعمل نه بعنوان آدرس بلکه عملوند واقعی استفاده شود. در اینصورت گوئیم دستورالعمل دارای عملوند بلافصل² است. هرگاه این بخش، آدرس عملوند را مشخص کند گوئیم دستور دارای آدرس مستقیم³ است. دلیل این نام وجود امکان سومی است که آدرس غیرمستقیم⁴ نامیده می شود و در آن بیت های بیت بخش دوم، آدرس کلمه حافظه ای را مشخص می کنند که آدرس عملوند در آن قرار دارد. یکی از بیت های کد دستور را برای تفکیک آدرس مستقیم و غیرمستقیم می توان بکار برد.

به منظور توضیح بیشتر، قالب کد دستورالعمل شکل ۲-۵ (الف) را در نظر بگیرید. این دستورالعمل شکل است از سه بیت کد عملیات، 12 بیت آدرس و یک بیت مربوط به روش آدرس دهی مستقیم یا غیرمستقیم، که با I نشان داده شده است. بیت I برای آدرس دهی مستقیم برابر 0 و برای آدرس دهی غیرمستقیم 1 می باشد. در شکل ۲-۵ (ب) دستورالعملی با آدرس مستقیم نشان داده شده است. این دستورالعمل در آدرس 22 از حافظه قرار دارد. بیت I برابر 0 است بنابراین دستور از نوع آدرس مستقیم است. کد عمل مشخص کننده دستور ADD است و بخش آدرس یک عدد دودویی معادل با 457 می باشد. واحد کنترل عملوند را در آدرس 457 یافته و آن را با محتوای AC جمع می کند. دستورالعمل موجود در آدرس 35 در شکل ۲-۵ (ج) دارای بیت $I=1$ است. بنابراین این دستور بعنوان یک دستورالعمل با آدرس غیرمستقیم شناخته می شود. بخش آدرس عدد دودویی معادل 300 است. کنترل به آدرس 300 مراجعه می نماید تا آدرس عملوند را بیابد. آدرس عملوند در این مورد 1350 است. سپس عملوند حاصل از آدرس 1350 با محتوای AC جمع می شود. لذا دستورالعمل ها با آدرس غیرمستقیم برای برداشتن عملوند نیاز به دو ارجاع به حافظه دارند. اولین ارجاع برای خواندن آدرس عملوند؛ و دومی برای خواندن خود عملوند است. ما آدرس موثر را در دستورات محاسباتی همان آدرس عملوند و در دستورالعمل های انشعاب آدرس هدف یا مقصد⁵ تعریف می کنیم. بنابراین آدرس موثر در دستورالعمل شکل ۲-۵ (ب) عبارتست از 457 و در دستورالعمل شکل ۲-۵ (ج) برابر 1350 می باشد.

روش های آدرس دهی مستقیم و غیرمستقیم در کامپیوتری که در این بخش ارائه شده بکار رفته اند. کلمه ای از حافظه برای نگهداری آدرس عملوند در دستورالعملی که با آدرس غیرمستقیم است بعنوان

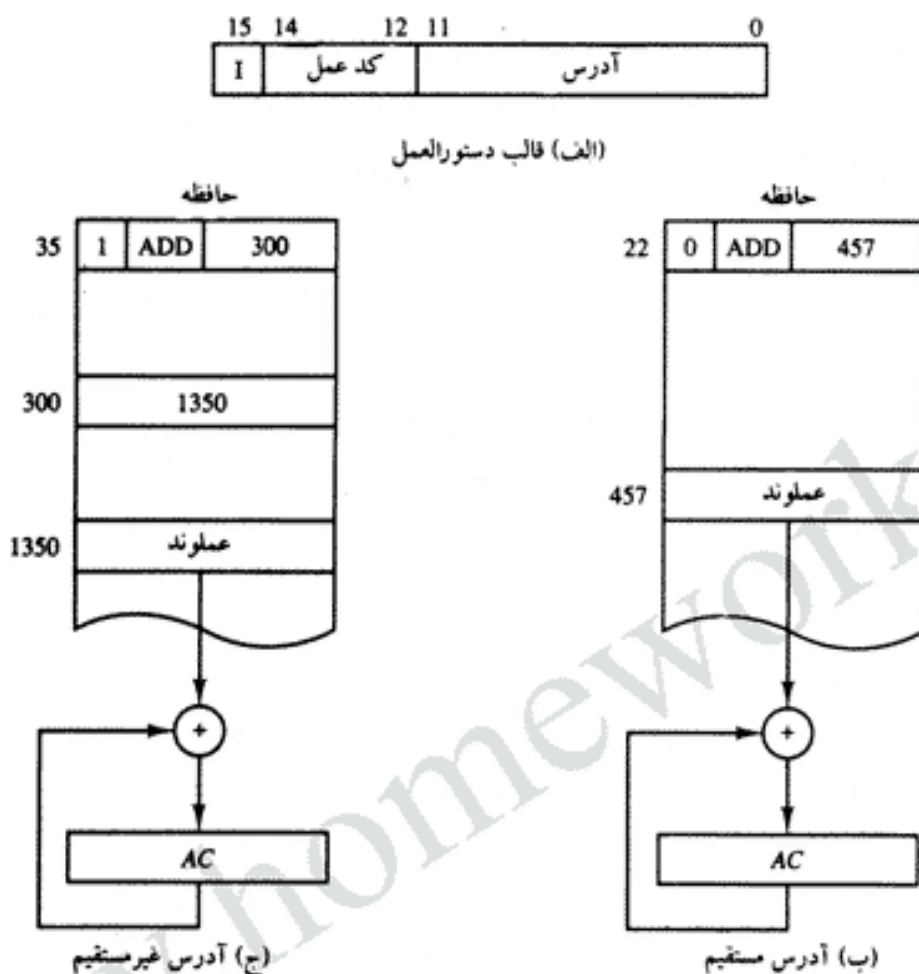
1- Clear

2- Immediate

3- Direct Address

4- Indirect Address

5- Destination



شکل ۲-۵ نمایش آدرس مستقیم و غیرمستقیم

اشاره‌گری به آرایه داده بکار می‌رود. همانطور که در کامپیوترهای تجاری مرسوم است این اشاره‌گر را می‌توان بجای قرار دادن در یک حافظه، در یک ثابت پردازشگر قرار داد.

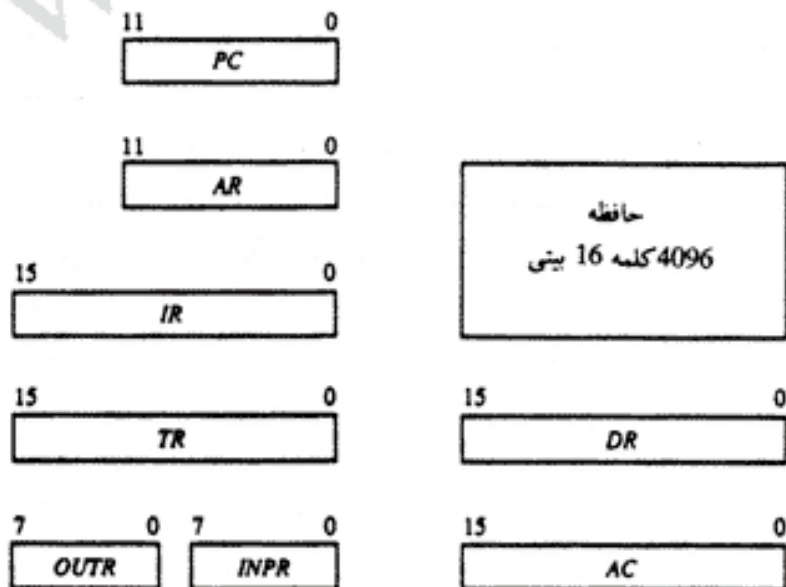
۲-۵ ثبات‌های کامپیوتر

دستورالعمل‌های کامپیوتر معمولاً در مکان‌هایی از حافظه بطور متوالی ذخیره شده و هر یک بنوبت اجرا می‌گردند. واحد کنترل یک دستور را از آدرس خاصی در حافظه خوانده و آن را اجرا می‌کند. سپس بهمین ترتیب با خواندن دستور بعدی و اجرای آن، روند را ادامه می‌دهد. اینگونه توالی در خواندن و اجرای دستورات، به شمارنده نیاز دارد تا آدرس دستورالعمل بعدی را پس از اتمام اجرای دستور جاری محاسبه نماید. همچنین یک ثابت در واحد کنترل برای ذخیره کد دستورالعمل، پس از خواندن آن از حافظه، لازم است. علاوه کامپیوتر به ثبات‌هایی در پردازشگر جهت دستکاری داده‌ها و نیز یک ثابت برای ذخیره‌سازی آدرس حافظه نیاز دارد. این نیازها، وجود ثبات‌های شکل ۳-۵ را در سیستم الزامی می‌دارد.

این ثابت‌ها در جدول ۱-۵ هم همراه با شرحی مختصر از کارشان و تعداد بیت‌های آنها آورده شده‌اند. واحد حافظه 4096 کلمه‌ی 16 بیتی ظرفیت دارد. دوازده بیت از کلمه برای مشخص کردن آدرس عملوند لازم است. لذا سه بیت برای بخش عمل دستور و یک بیت برای تعیین مستقیم یا غیرمستقیم بودن آدرس باقی می‌ماند. ثابت داده (DR) عملوند خوانده شده از حافظه را نگه می‌دارد. ثابت انباره (AC) یک ثابت همه منظوره در پردازش است. دستور خوانده شده از حافظه در ثابت دستورالعمل (IR) قرار می‌گیرد. ثابت موقت (TR) برای نگهداری موقت داده‌ها در حین پردازش به کار می‌رود. ثابت آدرس حافظه (AR) 12 بیتی است زیرا عرض آدرس حافظه برابر دوازده است. شمارنده برنامه (PC) نیز 12 بیت دارد و آدرس دستورالعمل بعدی را که پس از اجرای دستور جاری خوانده خواهد شد

جدول ۱-۵ لیست ثابت‌های کامپیوتر ساده

سمبل ثابت	تعداد بیت‌ها	نام ثابت	وظیفه
DR	16	ثابت داده	نگهداری عملوند حافظه
AR	12	ثابت آدرس	نگهداری آدرس حافظه
AC	16	انباره	ثابت پردازنده
IR	16	ثابت دستورالعمل	نگهداری کد دستور
PC	12	شمارنده برنامه	نگهداری آدرس دستور
TR	16	ثابت موقت	نگهداری داده‌های موقت
INPR	8	ثابت ورودی	نگهداری کاراکتر ورودی
OUTR	8	ثابت خروجی	نگهداری کاراکتر خروجی



شکل ۳-۵ ثابت‌ها و حافظه کامپیوتر پایه

نگه می‌دارد. PC یک رشته شمارشی را طی می‌نماید و سبب می‌شود تا کامپیوتر دستورات از قبل ذخیره شده در حافظه را متوالیاً بخواند. کلمات دستورالعمل یکی پس از دیگری خوانده شده و اجرا می‌شوند مگر اینکه به دستورانشعاب در برنامه برخورد شود. دستورانشعاب موجب انتقال به یک دستور غیرمتوالی در برنامه که در مکانی دیگر است می‌گردد. در تحقق این انتقال، بخش آدرس دستورانشعاب به PC منتقل می‌شود تا آدرس دستور قابل اجرای بعدی باشد. برای خواندن یک دستورالعمل، محتوای PC به عنوان آدرس به حافظه ارسال شده و بدین ترتیب سیکل خواندن از حافظه آغاز می‌شود. سپس PC به میزان یک واحد افزایش می‌یابد تا آدرس دستور بعدی تولید گردد.

دو ثبات برای ورودی و خروجی بکار رفته‌اند. ثبات ورودی (INPR) یک کاراکتر هشت بیتی را از یک وسیله ورودی دریافت می‌کند. ثبات خروجی (OUTR) یک کاراکتر هشت بیتی را برای یک وسیله خروجی نگه می‌دارد.

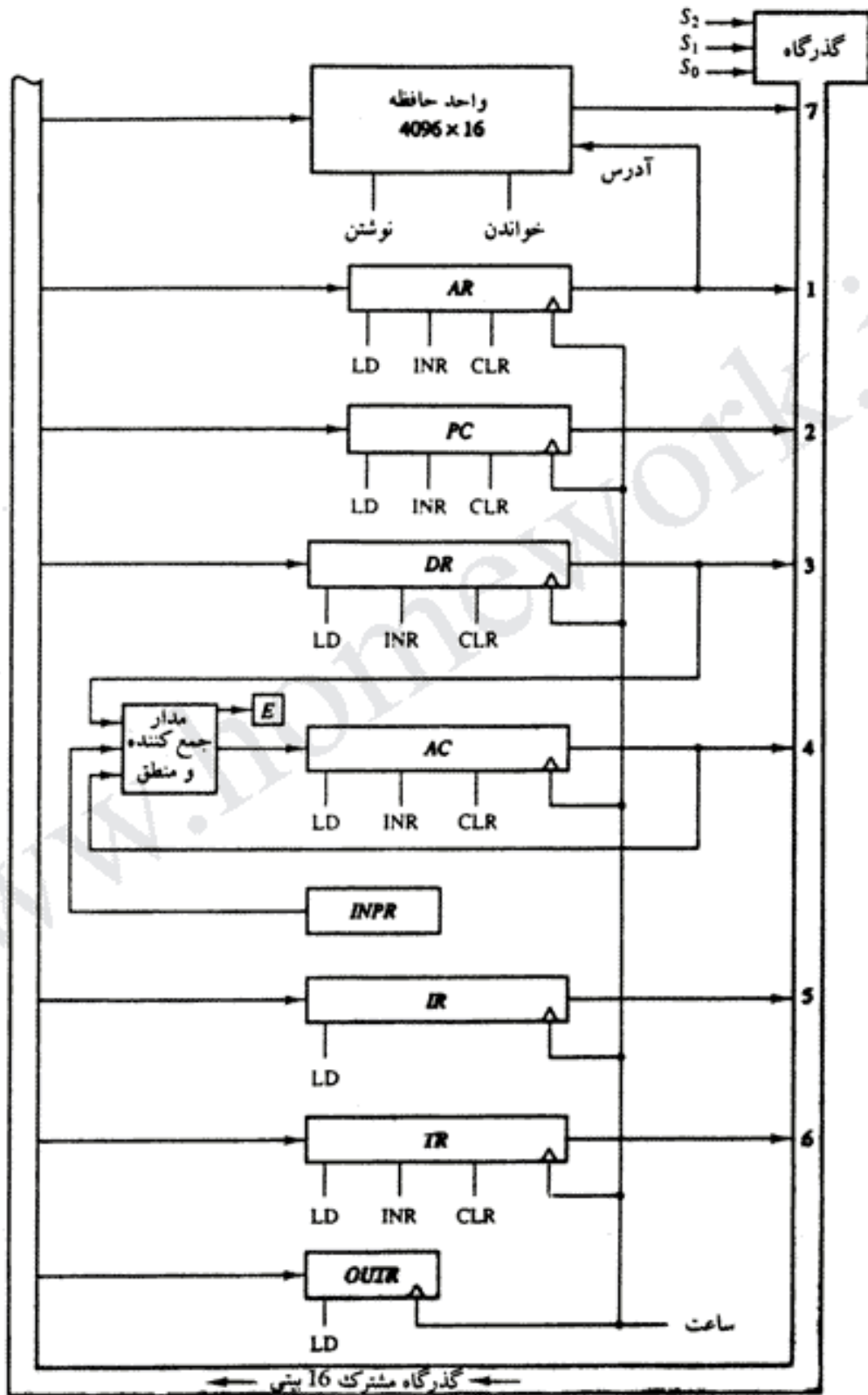
سیستم گذرگاه مشترک

کامپیوتر مورد نظر دارای هشت ثبات، یک واحد حافظه و یک واحد کنترل، (که در بخش ۴-۵ ارائه خواهد شد) می‌باشد. برای انتقال اطلاعات از یک ثبات به ثبات دیگر و یا بین حافظه و ثبات‌ها، مسیرهایی باید ایجاد شود. اگر بین خروجی هر ثبات با ورودی ثبات دیگری اتصال برقرار شود، تعداد سیم‌ها بیش از حد زیاد خواهد شد. روش بهتر در انتقال اطلاعات در یک سیستم که دارای تعداد قابل توجهی ثبات است استفاده از گذرگاه مشترک است. در بخش ۳-۴ نشان دادیم که چگونه با استفاده از مولتی پلکسر یا بافرهای سه حالته می‌توان یک گذرگاه سیستم ساخت. اتصالات ثبات‌ها و حافظه‌های کامپیوتر مورد نظر به گذرگاه مشترک فوق در شکل ۴-۵ نشان داده شده است.

خروجی‌های هفت ثبات و حافظه‌ها به گذرگاه مشترک متصل شده‌اند. در هر مقطع زمانی، خروجی خاصی که برای خطوط گذرگاه انتخاب می‌شود توسط مقادیر دودویی انتخاب S_1, S_0 و S_2 معین می‌گردد. مقادیر دهدهی که در امتداد هر خروجی نوشته شده معادل عدد دودویی لازم را برای انتخاب نشان می‌دهد. مثلاً در کنار خروجی DR عدد 3 نوشته شده است، لذا 16 بیت خروجی DR هنگامی روی گذرگاه قرار می‌گیرد که $S_2S_1S_0 = 011$ باشد زیرا این مقدار معادل دودویی عدد 3 است. خطوط گذرگاه مشترک به ورودی‌های همه ثبات‌ها و ورودی داده حافظه‌ها نیز وصل شده‌اند.

هر ثبات خاصی که ورودی LD آن فعال باشد، داده‌ها را با پالس بعدی از گذرگاه دریافت می‌کند. حافظه هم بهنگام فعال شده ورودی "نوشتن" W محتوای گذرگاه را دریافت می‌نماید. هنگام فعال شدن ورودی "خواندن" R حافظه، خروجی آن روی گذرگاه قرار می‌گیرد.

چهار ثبات DR، AC، IR و TR هر یک 16 بیتی هستند. دو ثبات AR و PC هر یک 12 بیت دارند زیرا آدرس حافظه را نگه می‌دارند. وقتی که محتوای AR یا PC به گذرگاه مشترک 16 بیتی اعمال شود،



شکل ۴-۵ ثبات های کامپیوتر پایه متصل به یک گذرگاه مشترک

چهار بیت بالا رتبه 0 می شوند. وقتی که AR یا PC اطلاعات را از گذرگاه دریافت می کنند، تنها 12 بیت پایین رتبه گذرگاه به داخل ثبات منتقل می گردد.

ثبات ورودی INPR و OUTR هر یک هشت بیت داشته و با هشت بیت پایین رتبه در گذرگاه ارتباط برقرار می کنند. INPR برای گذرگاه، اطلاعات را از خارج تهیه می کند ولی OUTR فقط اطلاعات را از گذرگاه دریافت می نماید. دلیل این است که INPR کاراکتری را از وسیله خارجی دریافت می کند که باید بعداً به AC انتقال یابد. بالعکس OUTR کاراکتر را از AC دریافت و به وسیله خارجی تحویل می دهد. ضمناً هیچگونه انتقالی از OUTR به ثبات های دیگر صورت نمی گیرد.

شانزده خط گذرگاه مشترک، اطلاعات را از شش ثبات و حافظه دریافت می کنند. خطوط گذرگاه ورودی شش ثبات و حافظه متصلند. پنج ثبات دارای سه ورودی کنترل می باشند: LD (بارشدن)¹ INR (افزایش)² و CLR (پاک شدن)³. این نوع ثبات معادل یک شمارنده دودویی با بارشدن موازی و نیز پاک کننده همزمان⁴ مطابق شکل ۱۱-۲ است. عمل افزایش با فعال شدن ورودی "شمارش"⁵ شمارنده میسر است. دو ثبات هر یک فقط دارای یک ورودی LD می باشند. این نوع ثبات در شکل ۷-۲ نشان داده شد.

ورودی داده ها و خروجی داده های حافظه به گذرگاه مشترک متصل اند. ولی ورودی آدرس حافظه به AR متصل است. بنابراین از AR همیشه باید برای معین کردن آدرس حافظه استفاده شود. با استفاده از یک ثبات واحد برای آدرس، نیاز به یک گذرگاه آدرس که در غیراینصورت لازم بود از بین می رود. هنگام عمل نوشتن، محتوای هر ثبات می تواند بعنوان داده ورودی حافظه مشخص شود. به طور مشابه، هر ثباتی، بجز AC، می تواند داده های حافظه را پس از عمل خواندن دریافت کند.

شانزده ورودی AC از یک مدار جمع کننده و منطقی می آیند. این مدار جمع کننده دارای سه مجموعه ورودی است. یک مجموعه از ورودی ها از خروجی AC می آید. این خطوط برای پیاده سازی ریزعمل هایی⁶ مانند متمم سازی AC و شیفت AC بکار می روند. مجموعه ورودی 16 بیتی دیگری از ثبات DR می آید. ورودی هایی که از DR و AC می آیند برای ریزعمل های حسابی و منطقی مانند جمع DR با AC یا AND کردن DR با AC بکار می روند. نتیجه جمع به AC و رقم نقلی نهایی⁷ به فلیپ فلاپ E (بیت گسترش⁸ AC) منتقل می شوند. سومین مجموعه هشت بیتی به ثبات ورودی INPR وصل است. عملکرد INPR و OUTR در بخش ۷-۵ شرح داده خواهد شد.

توجه کنید که در یک سیکل ساعت، می توان هم محتوای هر ثبات را روی گذرگاه قرار داد و هم بطور همزمان یک عمل را در جمع کننده و مدار منطقی اجرا نمود. لبه پالس سیکل ساعت محتوای گذرگاه را به ثبات معین شده و خروجی جمع کننده و مدار منطقی را به AC منتقل می نماید. مثلاً دو

1- Load
4- Synchronous
7- End carry out

2- Increment
5- Count
8- Extended Ac bit

3- Clear
6- Microoperation

ریز عمل

$$DR \leftarrow AC \text{ و } AC \leftarrow DR$$

می توانند هم زمان اجرا شوند. این اعمال با قرار گرفتن محتوای AC بر روی گذرگاه (با $S_2S_1S_0=100$)، فعال شدن LD (بارشدن) در ورودی DR، و همزمان با آن انتقال محتوای DR از طریق مدار جمع کننده و منطقی، و فعال شدن LD در ورودی AC انجام می شوند. دو انتقال فوق با وقوع لبه پائین رونده پالس ساعت رخ می دهد.

۳-۵ دستورالعمل های کامپیوتر

کامپیوتر پایه مورد بحث دارای سه قالب دستورالعمل مطابق شکل ۵-۵ می باشد. این قالب ها 16 بیتی هستند. بخش عمل کد دستور سه بیتی است و مفهوم بقیه بیت ها به نوع کد وابسته است. دستورالعمل های ارجاع به حافظه از 12 بیت برای تعیین آدرس و 1 بیت برای مشخص کردن روش آدرس دهی I استفاده می کنند. برای آدرس دهی مستقیم، I برابر 0 و برای آدرس غیرمستقیم I مساوی 1 است، شکل ۲-۵. دستورالعمل های ارجاع به ثبات با کد عملی 111 و یک بیت 0 در انتها الیه سمت چپ آن (بیت 15) قابل تشخیص اند. دستورالعمل های ثباتی عمل را بر روی AC و یا هر تستی بر روی آن را مشخص می نمایند. در این دستورات عملوندی از حافظه مورد نیاز نیست؛ بنابراین از 12 بیت باقیمانده برای مشخص کردن عمل و یا تست مورد نظر استفاده می شود. بطور مشابه، دستورات ورودی - خروجی نیز نیاز به ارجاع به حافظه ندارند و با کد عملیاتی 111 و 1 در بیت انتها الیه سمت چپ دستور قابل تشخیص اند. بقیه 12 بیت نوع عمل ورودی - خروجی و یا تستی که باید انجام شود را مشخص می سازد. واحد کنترل کامپیوتر نوع دستورالعمل را با توجه به بیت های مکان های 12 تا 15 دستورالعمل تشخیص می دهد. اگر سه بیت کد عمل در مکان های 12 تا 14 برابر 111 نباشد دستور از نوع ارجاع به حافظه است و بیت 15 بعنوان بیت روش آدرس دهی I در نظر گرفته می شود. اگر سه بیت کد عمل 111

15	14	12	11	0	
I	کد عمل	آدرس			کد عمل از 000 تا 110

(الف) دستورالعمل های حافظه ای

15	12	11	0		
0	1	1	1	عمل ثباتی	کد عمل برابر 111، I برابر 0

(ب) دستورالعمل های ثبات

15	12	11	0		
1	1	1	1	عمل I/O	کد عمل برابر 111، I برابر 1

(ج) دستورالعمل های ورودی - خروجی

شکل ۵-۵ قالب دستورالعمل ها در کامپیوتر پایه

باشد، واحد کنترل بیت مکان 15 را بررسی می نماید. اگر این بیت 0 باشد، دستور از نوع ارجاع به ثبات است. اگر بیت مذکور 1 باشد، دستور از نوع ورودی - خروجی است. هنگامی که کد عمل 111 باشد بیت مکان 15 با I مشخص می شود ولی بعنوان بیت روشن آدرسی دهی بکار نمی رود. گفتیم که برای کد عمل تنها سه بیت به کار رفته است. لذا ممکن است به نظر برسد که کامپیوتر به حداکثر هشت عمل محدود شده است. با این وجود، چون دستورات ارجاع به ثبات و ورودی خروجی از 12 بیت باقیمانده کد عمل استفاده می کنند، کل دستورات عملیها از هشت تجاوز می کنند. در واقع در این حال، دستورات عملیهای مورد استفاده در کامپیوتر 25 خواهد بود

دستورات عملیهای کامپیوتر در جدول ۲-۵ ملاحظه می شوند. سمبل مشخص کننده دستورات عملیها از یک کلمه سه حرفی تشکیل شده و خلاصه عمل را برای برنامه نویس و کاربر مشخص می کنند. کد مبنای

جدول ۲-۵ دستورات عملیهای کامپیوتر پایه

سمبل	کد شانزده شانزدهمی		شرح
	I = 0	I = 1	
AND	0xxx	8xxx	AND کردن کلمه حافظه با AC
ADD	1xxx	9xxx	جمع کردن کلمه حافظه با AC
LDA	2xxx	Axxx	بار کردن کلمه حافظه در AC
STA	3xxx	Bxxx	ذخیره محتوای AC در حافظه
BUN	4xxx	Cxxx	انشعاب نامشروط
BSA	5xxx	Dxxx	انشعاب و ضبط آدرس بازگشت
ISZ	6xxx	Exxx	افزایش و گذر در صورت نتیجه صفر
CLA	7800		پاک کردن AC
CLE	7400		پاک کردن E
CMA	7200		متعم کردن AC
CME	7100		متعم کردن E
CIR	7080		چرخش AC و E به راست
CIL	7040		چرخش AC و E به چپ
INC	7020		افزایش AC
SPA	7010		گذر از دستور بعدی اگر AC مثبت باشد
SNA	7008		گذر از دستور بعدی اگر AC منفی باشد
SZA	7004		گذر از دستور بعدی اگر AC صفر باشد
SZE	7002		گذر از دستور بعدی اگر E صفر باشد
HLT	7001		توقف کامپیوتر
INP	F800		دریافت کاراکتر و انتقال آن به AC
OUT	F400		برداشتن کاراکتر از AC و انتقال آن به خروجی
SKI	F200		گذر مبتنی بر پرچم ورودی
SKO	F100		گذر مبتنی بر پرچم خروجی
ION	F080		فعال کردن وقفه ها
IOF	F040		غیرفعال کردن وقفه ها

شانزده برابر با عدد معادل شانزده کد دودویی بکار رفته در دستورالعمل است. با استفاده از معادل شانزده، ما 16 بیت کد دستورالعمل را به چهار رقم کاهش می‌دهیم که هر رقم معادل چهار بیت است. دستورالعمل ارجاع به حافظه دارای بخش آدرس 12 بیتی است. این بخش از دستور با سه مشخص شده و به مفهوم سه رقم در مبنای شانزده است. آخرین بیت دستورالعمل با نماد I مشخص گردیده است. هرگاه $I=0$ باشد چهار بیت آخر دستور یک رقم مبنای شانزده معادل با 0 تا 6 است. وقتی که $I=1$ باشد چهار بیت آخر دستور یک رقم مبنای شانزده معادل با 8 تا E می‌باشد.

دستورات ارجاع به ثبات برای مشخص کردن عملوند از 16 بیت استفاده می‌نمایند. چهاربیت سمت چپ همیشه 0111 هستند. که معادل 7 در مبنای شانزده می‌باشد. سه رقم دیگر معادل دودویی 12 بیت باقیمانده را در مبنای شانزده نشان می‌دهند. دستورات ورودی - خروجی هم از تمام 16 بیت برای مشخص کردن عمل استفاده می‌کنند. چهاربیت آخر همیشه 1111 می‌باشند که معادل F در مبنای شانزده است.

کامل بودن مجموعه دستورات

قبل از بررسی عمل‌هایی که دستورات انجام می‌دهند، اجازه بدهید درباره نوع دستوراتی که باید در کامپیوتر در نظر داشت بحث کنیم. یک کامپیوتر باید دستورالعمل‌هایی را داشته باشد که کاربر بتواند بکمک آنها برنامه‌هایی بزبان ماشین برای محاسبه توابعی که قابل محاسبه هستند بنویسد. اگر کامپیوتر دارای دستورالعمل‌های کافی در هر یک از رسته‌های زیر باشد مجموعه دستورات آن کافی تلقی می‌گردد:

۱- دستورالعمل‌های حسابی، منطقی و شیفت

۲- دستورالعمل‌هایی برای تبادل اطلاعات با حافظه و ثبات‌های پردازشگر

۳- دستورات کنترل برنامه همراه با دستورالعمل‌هایی که شرایط وضعیتی^۱ را چک می‌کنند

۴- دستورات ورودی و خروجی

دستورات حسابی منطقی قابلیت‌های محاسباتی را برای پردازش داده‌هایی که کاربر مایل است بکار برد فراهم می‌آورند. بخش اعظم اطلاعات دودویی در یک کامپیوتر دیجیتال در حافظه ذخیره می‌شود، ولی تمام محاسبات در ثبات‌های پردازشگر انجام می‌گردد. بنابراین کاربر باید امکان انتقال اطلاعات را بین دو واحد داشته باشد. تصمیم‌گیری، از جمله قابلیت‌های مهم در کامپیوترهای دیجیتال است. بعنوان مثال دو عدد را می‌توان مقایسه کرد و اگر اولی بزرگتر از دومی باشد ادامه کار ممکن است با حالتی که دومی از اولی بزرگتر است متفاوت باشد. در اینحالت دستورالعمل‌های کنترل برنامه مانند دستورات انشعاب برای تغییر ترتیب اجرای برنامه بکار می‌روند. دستورات ورودی - خروجی برای ارتباط بین کامپیوتر و کاربر لازمند. برنامه‌ها و داده‌ها باید به حافظه منتقل گردند و نتایج محاسبات باید به کاربر بازگردانده شود.

1- Status Condition

دستورات لیست شده در جدول ۲-۵ از حداقل مجموعه‌ای تشکیل یافته که قادر است قابلیت‌های ذکر شده در بالا را فراهم آورد. در این جدول یک دستورالعمل حسابی ADD، دو دستور مرتبط با اعمال حسابی منطقی، یعنی متمم کردن AC (CMA) و افزایش AC (INC) وجود دارد. با این سه دستور می‌توان عددهای دودویی را جمع و تفریق کرد بشرطی که اعداد منفی بکسل متمم 2 علامت دار نشان داده شوند. با استفاده از دستورالعمل‌های چرخش CIR و CIL می‌توان شیفت‌های حسابی و یاهر گونه شیفت دیگر را انجام داد. ضرب و تقسیم با استفاده از جمع و تفریق امکان پذیر است. سه عمل منطقی وجود دارد: AND، متمم سازی AC (CMA) و پاک کردن AC (CLA). دستور AND و مکمل با هم امکان NAND را فراهم می‌کنند. می‌توان نشان داد که با عمل NAND انجام کلیه اعمال منطقی دو متغیره امکان پذیر است (جدول ۶-۴).

انتقال اطلاعات از حافظه به AC با دستور بارکردن AC (LDA) انجام می‌شود. ذخیره اطلاعات از AC در حافظه هم توسط دستور ذخیره کردن در AC (STA) صورت می‌گیرد. دستورالعمل‌های انشعاب BUN و BSA و ISZ همراه با چهار دستور گذر^۱ امکان کنترل برنامه و بررسی شرایط وضعیتی بوجود می‌آورند. دستورالعمل‌های ورودی INP و خروجی OUT موجب انتقال اطلاعات بین کامپیوتر و وسایل خارجی می‌گردند.

هر چند مجموعه دستورالعمل‌ها برای کامپیوتر پایه نسبتاً کافی است ولی کارا نیست زیرا دستوراتی که کسراراً بکار می‌روند بسرعت اجرا نمی‌شوند. مجموعه کارایی از دستورات باید شامل دستورالعمل‌های تفریق، ضرب، OR و XOR^۲ باشد. در کامپیوتر پایه برای این اعمال باید برنامه نویسی کرد. این برنامه‌ها برای کامپیوتر پایه به همراه مثالهای برنامه نویسی دیگری در فصل ۶ آورده شده است. با استفاده از تعداد محدودی از دستورالعمل‌ها می‌توان جزئیات طراحی منطقی کامپیوتر را نشان داد. مجموعه کامل تری از دستورات سبب می‌شود تا طراحی بیش از حد پیچیده شود. با روش انتخابی، ما اصول سازمان کامپیوتر را نشان داده و طراحی را بدون بحث‌های مفصل دنبال خواهیم کرد. در فصل ۸ لیست کاملی از دستورالعمل‌های کامپیوتر که در اغلب کامپیوترهای تجاری، وجود دارد را ارائه خواهیم نمود.

وظیفه هر دستور در جدول ۲-۵ لیست شده است و ریز عمل‌های لازم برای اجرای هر یک از آنها در بخش ۵-۵ تا ۵-۷ آورده شده است. ما فعلاً بدلیل اولویت در آشنایی با واحد کنترل و سازمان آن، این بحث را بتأخیر می‌اندازیم.

۴-۵ زمانبندی و کنترل

زمانبندی همه ثبات‌ها در کامپیوتر پایه بوسیله یک مولد پالس ساعت اصلی کنترل می‌شود. پالس‌های ساعت به همه فلیپ‌فلاپ‌ها و ثبات‌ها، از جمله آنهایی که در واحد کنترل سیستم قرار دارند

1- Skip

2- Exclusive OR (XOR)

اعمال می‌شوند. پالس‌های ساعت هیچ ثباتی را عوض نمی‌کنند مگر اینکه ثبات توسط یک سیگنال کنترل فعال شود. سیگنال‌های کنترل در واحد کنترل تولید می‌شوند و ورودی‌های کنترل را برای مولتی پلکسر در گذرگاه مشترک، ورودی‌های کنترل در ثبات‌های پردازشگر و ریزاعمال در انبار AC فراهم می‌آورند.

دو نوع سازمان کنترل عمده وجود دارد که عبارتند از: کنترل سخت افزاری و کنترل ریز برنامه نویسی شده. در نوع سخت افزاری، مدار منطقی کنترل با گیت‌ها، فلیپ فلاپ‌ها، دیکدرها و سایر مدارهای دیجیتال پیاده‌سازی می‌شود. حسن این روش داشتن سرعت بالا است که با بهینه کردن آن حاصل می‌شود. در سازمان ریز برنامه نویسی شده اطلاعات کنترل در یک حافظه کنترل ذخیره شده است. حافظه کنترل به صورتی برنامه نویسی می‌شود که رشته ریز عمل‌های لازم را اجرا کند. در کنترل سخت‌افزاری، همانطور که از نامش پیداست، در صورت نیاز به تغییر یا تصحیح آن، باید سیم بندی بین قطعات مختلف را تغییر داد. در کنترل ریز برنامه نویسی شده هرگونه تغییر تنها با تصحیح ریز برنامه در حافظه کنترل امکان پذیر است. در این بخش یک واحد کنترل ریز برنامه نویسی برای کامپیوتر پایه ارائه می‌کنیم. واحد کنترل ریز برنامه نویسی شده برای کامپیوتری مشابه در فصل ۷ آمده است.

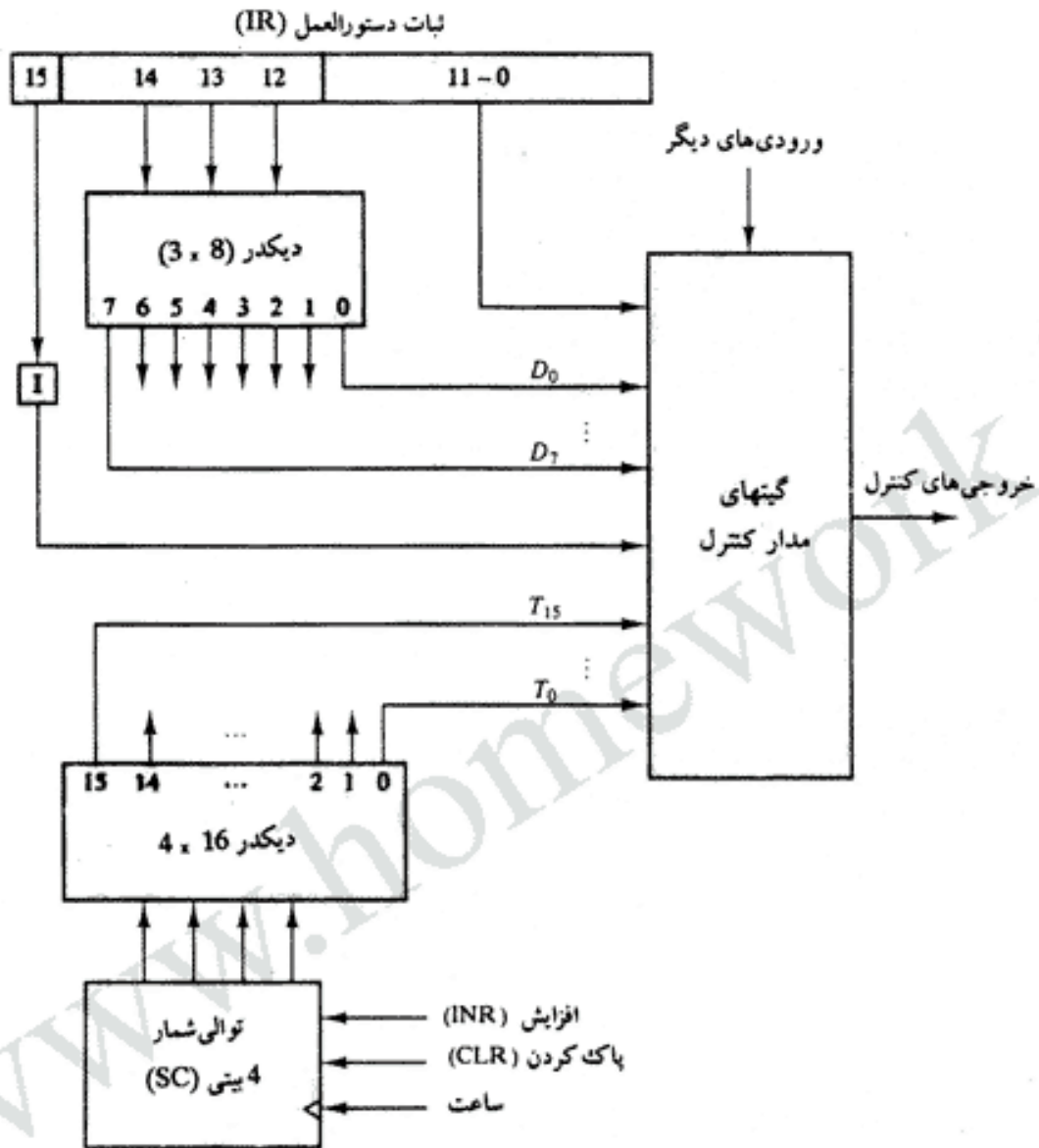
بلاک دیاگرام واحد مزبور در شکل ۶-۵ دیده می‌شود. این واحد از دو دیکدر، یک توالی شمار SC و چند گیت تشکیل شده است. هر دستوری که از حافظه خوانده شود در ثبات دستورالعمل^۱ (IR) قرار می‌گیرد. مکان این ثبات در سیستم گذرگاه مشترک در شکل ۴-۵ مشخص شده است. ثبات دستورالعمل که مجدداً در شکل ۶-۵ آورده شده به سه بخش تقسیم شده است: بیت I_1 ، کد عمل، و بیت 0 تا 11. کد عمل واقع در بیت‌های 12 تا 14، توسط یک دیکدر 3×8 دیکد می‌شود. هشت خروجی دیکدر با D_0 تا D_7 مشخص شده‌اند. زیر نویس دهنده میبین مقدار دودویی کد عمل است. بیت 15 دستورالعمل به فلیپ فلاپ I منتقل می‌شود. بیت‌های 0 تا 11 به مدار کنترل اعمال می‌گردند. شمارنده چهاربیتی می‌تواند از 0 تا 15 متوالیاً بشمارد. خروجی‌های این شمارنده به صورت 16 سیگنال زمانبندی T_0 تا T_{15} دیکد می‌شود. مدار داخلی گیت‌های کنترل بعداً بهنگام بررسی مشروحتر طراحی کامپیوتر بدست خواهد آمد.

شمارنده SC را می‌توان بطور همزمان^۲ افزایش داد یا پاک کرد (شمارنده شکل ۱۱-۲ ملاحظه شود). اغلب اوقات، شمارنده افزایش می‌یابد تا رشته سیگنال‌های زمانبندی متوالی را از خروجی دیکدر ایجاد کند. هر از گاه یکبار این شمارنده 0 می‌شود و سبب می‌گردد تا سیگنال زمانبندی بعدی T_0 شود. مثلاً حالتی را در نظر بگیرید که SC افزایش داده شود و تا سیگنال‌های T_0 ، T_1 ، T_2 ، T_3 ، T_4 را بطور متوالی فراهم نماید. در T_4 ، اگر خروجی دیکدر D_3 فعال باشد، SC صفر می‌شود. این موضوع بصورت نمادی با عبارت زیر نشان داده می‌شود:

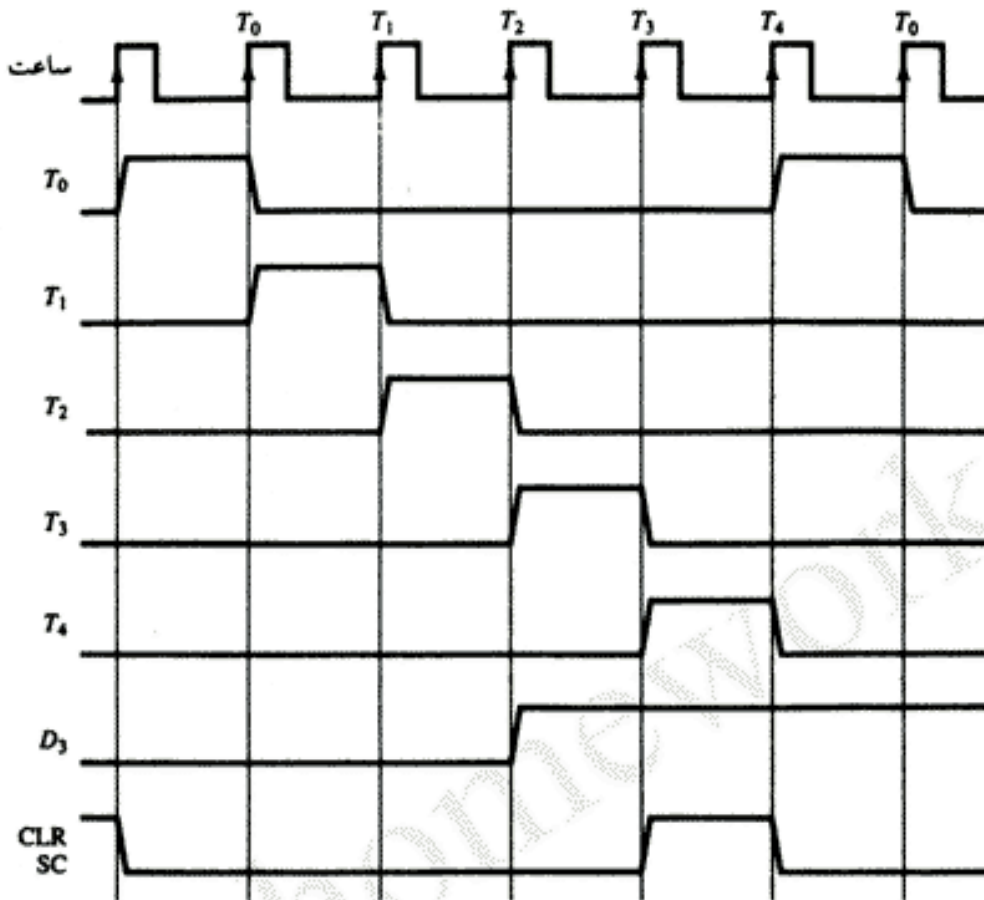
$$D_3 T_4: SC \leftarrow 0$$

1- Instruction Register

2- Synchronous



دیاگرام زمانی شکل ۵-۷ ارتباط زمانی سیگنال های کنترل را نشان می دهد. شمارنده SC بهنگام لبه مثبت پالس ساعت واکنش نشان داده و عمل می کند. در آغاز، ورودی CLR شمارنده SC فعال است. اولین لبه مثبت پالس ساعت، SC را 0 می کند که این بنوبه خود سیگنال زمانبندی T_0 را که از دیکدر خارج می شود فعال می نماید. T_0 در طول یک سیکل پالس ساعت فعال باقی می ماند. لبه مثبت پالس ساعت فقط آن دسته از ثباتها را فعال می کند که ورودی کنترلشان به سیگنال زمانبندی T_0 متصل باشد. SC با هر لبه مثبت پالس ساعت یک واحد افزایش می یابد، مگر اینکه CLR آن فعال باشد. این افزایش سبب تولید سیگنال های T_0, T_1, T_2, T_3, T_4 و غیره مطابق شکل می گردد (به رابطه بین سیگنال



شکل ۵-۷ مثالی از سیگنال‌های زمانبندی واحد کنترل

زمانبندی و لبه مثبت پالس مربوطه توجه کنید). اگر SC صفر نشود، سیگنال‌های زمانبندی با T_5 و T_6 تا T_{15} ادامه یافته و مجدداً به T_0 باز می‌گردند.

سه موج آخر در شکل ۵-۷ نشان می‌دهد که چگونه SC، وقتی که $D_3T_4=1$ است، پاک می‌شود. خروجی D_3 هنگامی فعال می‌شود که سیگنال T_2 پایان یافته باشد. وقتی سیگنال T_4 فعال می‌شود خروجی گیت AND بکار رفته در تابع کنترل D_3T_4 فعال می‌گردد. این به ورودی CLR مربوط به SC اعمال شده است. در لبه مثبت پالس بعد از T_3 ، یعنی T_4 شمارنده پاک می‌شود. این به نوبه خود سبب می‌شود تا در عوض T_5 که می‌باید با افزایش SC فعال می‌شد، سیگنال T_0 بعثت پاک شدن شمارنده SC تولید شود.

سیکل خواندن یا نوشتن با لبه بالارونده لبه سیگنال زمانبندی آغاز می‌شود. فرض بر این است که سیکل زمانی عملکرد حافظه کمتر از سیکل پالس ساعت باشد. با این فرض، سیکل خواندن یا نوشتن توسط یک سیگنال زمانبندی آغاز و در لبه پالس ساعت بعدی خاتمه می‌یابد. سپس لبه پالس برای بارکردن حافظه در یک ثبات بکار می‌رود. این ارتباط زمانی در بسیاری از کامپیوترها برقرار نیست زیرا معمولاً سیکل زمانی حافظه طولانی‌تر از سیکل پالس ساعت است. در این حالت لازم است زمان

انتظاری^۱ در پردازشگر برای فرصت دادن به حافظه ایجاد کرد تا کلمه حافظه مهیا گردد. برای سهولت در ارائه مطلب ما فرض خواهیم کرد که در کامپیوتر پایه، زمان انتظار لازم نیست. برای درک کامل عملکرد یک کامپیوتر، باید رابطه زمانی بین لبه های پالس ساعت و سیگنال های زمانبندی را دانست. بعنوان مثال عبارت انتقال ثبات

$$T_0 : AR \leftarrow PC$$

انتقال محتوای PC را به AR بهنگام فعال شدن T_0 مشخص می کند. T_0 در تمام مدت یک سیکل ساعت فعال است. در طول این مدت محتوای PC روی گذرگاه قرار می گیرد. (با $S_2S_1S_0=010$) و ورودی بارشدن LD ثبات AR فعال می شود. انتقال واقعی تا پایان ساعت و در واقع تا آغاز لبه مثبت پالس بعدی رخ نمی دهد. همین لبه SC را از 0000 به 0001 افزایش می دهد. پالس ساعت بعدی T_1 را فعال و T_0 را غیرفعال می سازد.

۵-۵ سیکل دستورالعمل

برنامه ای که در واحد حافظه کامپیوتر ذخیره شده از دنباله ای از دستورالعمل ها تشکیل شده است. این برنامه طی سیکل های اجرایی دستورالعمل اجرا می شود. هر سیکل دستورالعمل نیز خود به مجموعه ای از سیکل های جزئی یا فازها تقسیم می شود. در کامپیوتر پایه هر سیکل دستورالعمل متشکل از فازهای زیر است.

۱- برداشت یک دستور از حافظه

۲- دیکد کردن دستور

۳- خواندن آدرس مؤثر

۴- اجرای دستورالعمل

پس از پایان مرحله ۴، کنترل به مرحله ۱ بازگشته و برداشت^۲، دیکد و اجرای^۳ دستور بعدی را انجام می دهد. این فرآیند بطور نامحدود ادامه می یابد مگر اینکه کامپیوتر با دستور HALT مواجه شود.

برداشت و دیکد

در آغاز شمارنده برنامه، PC، با آدرس اولین دستورالعمل برنامه بار می شود. شمارنده SC پاک شده و در نتیجه سیگنال T_0 را تولید می نماید. پس از هر پالس ساعت، SC یک واحد افزایش می یابد تا رشته سیگنال های T_0 ، T_1 ، T_2 و غیره متوالیاً تولید گردند. ریز عمل های مربوط به فاز برداشت و دیکد را می توان توسط عبارات انتقال ثبات زیر مشخص نمود.

$$T_0 : AR \leftarrow PC$$

1- Wait

2- Fetch

3- Execute

$$T_1 : IR \leftarrow M [AR] , PC \leftarrow PC + 1$$

$$T_2 : D_0, \dots, D_7 \leftarrow \text{دیکد} , AR \leftarrow IR (0-11), I \leftarrow IR(15)$$

از آنجا که فقط AR به ورودی های آدرس حافظه متصل می باشد، لازم است که در حین سیگنال زمانبندی T_0 ، آدرس از PC به AR منتقل شود. سپس با خواندن دستورالعمل از حافظه، در سیگنال زمانبندی T_1 ، مقدار دستور در ثبات دستورالعمل قرار می گیرد. همزمان با عمل اخیر، PC یک واحد افزایش می یابد تا حاوی دستورالعمل بعدی در برنامه باشد. در زمان T_2 ، کد عمل موجود در IR، دیکد می شود و بیت غیرمستقیم به فلیپ فلاپ I منتقل می گردد و نهایتاً بخش آدرس دستورالعمل به AR انتقال می یابد. توجه داشته باشید که SC پس از هر پالس ساعت افزایش می یابد تا سیگنال های T_0 ، T_1 ، T_2 متوالیاً تولید گردند.

شکل ۵-۸ چگونگی پیاده سازی دو عبارت انتقال ثبات را در سیستم گذرگاه نشان می دهد. برای فراهم کردن مسیر داده ها در انتقال PC به AR، باید سیگنال زمانبندی T_0 را به مدار اعمال نمائیم تا اتصالات زیر فراهم شود.

۱- قرار دادن محتوای PC روی گذرگاه با ورودی های انتخاب کننده $S_2S_1S_0=010$

۲- انتقال محتوای گذرگاه به AR با توانا کردن ورودی LD مربوط به AR

پالس ساعت بعدی انتقال از PC به AR را موجب می شود زیرا $T_0=1$ است. برای پیاده کردن عبارت

دوم

$$T_1 : IR \leftarrow M[AR], \quad PC \leftarrow PC + 1$$

لازم است با استفاده از سیگنال زمانبندی T_1 اتصالات زیر را در سیستم گذرگاه برقرار کنیم.

۱- ورودی خواندن حافظه فعال شود

۲- محتوای حافظه به کمک $S_2S_1S_0=111$ روی گذرگاه قرار گیرد.

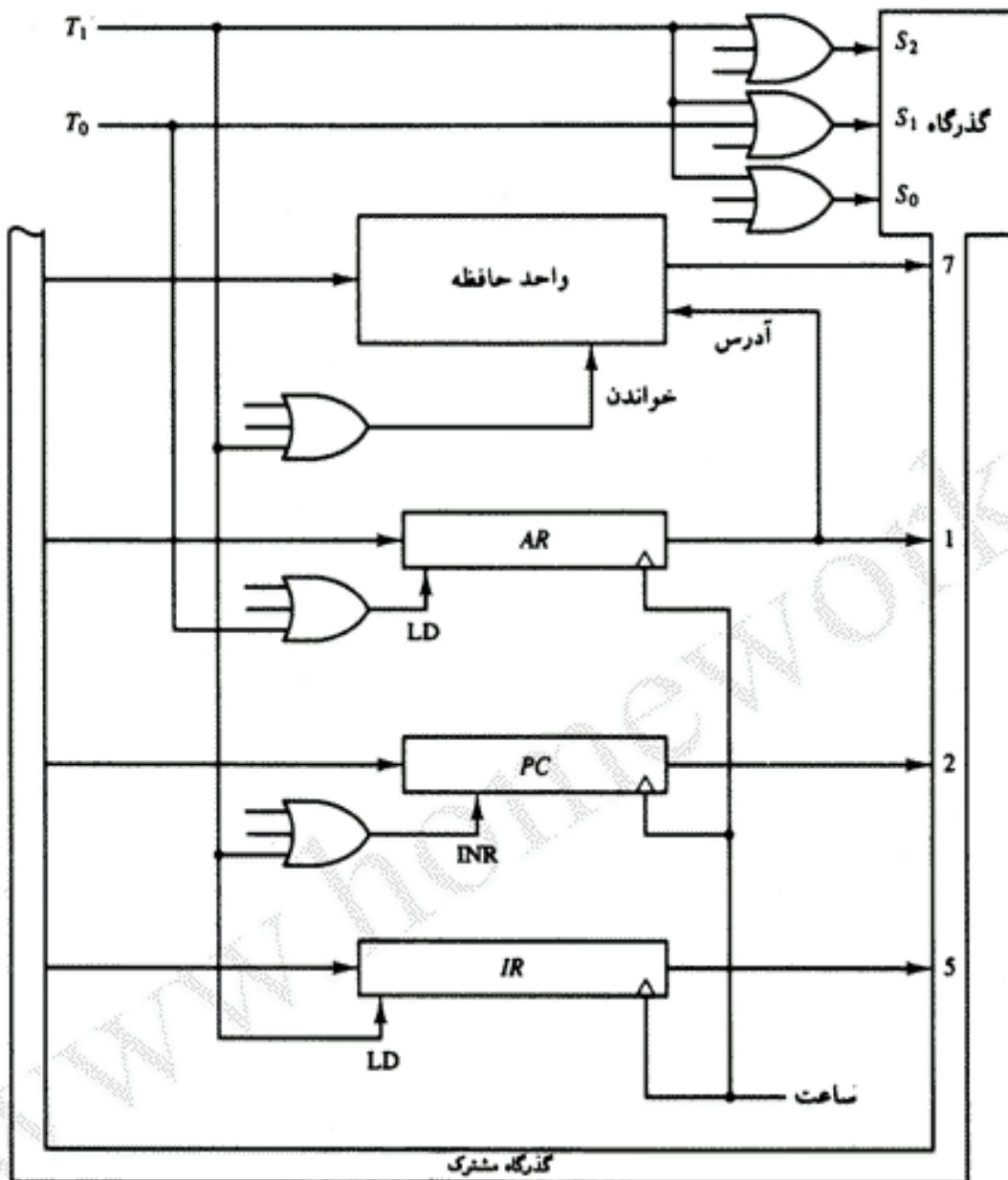
۳- محتوای گذرگاه با توانا شدن LD ثبات IR، به آن منتقل گردد.

۴- PC با توانا شدن ورودی INR، یک واحد افزایش می یابد.

شکل ۵-۸ بخشی از گذرگاه سیستم را همراه با اتصالات T_0 و T_1 به ورودی های کنترل ثبات ها، حافظه ها و ورودی های انتخاب کننده نشان می دهد. چون توابع کنترلی دیگری که اعمال مشابهی را انجام می دهند نیز وجود دارند لذا گیت های OR چند ورودی در دیاگرام آورده شده اند.

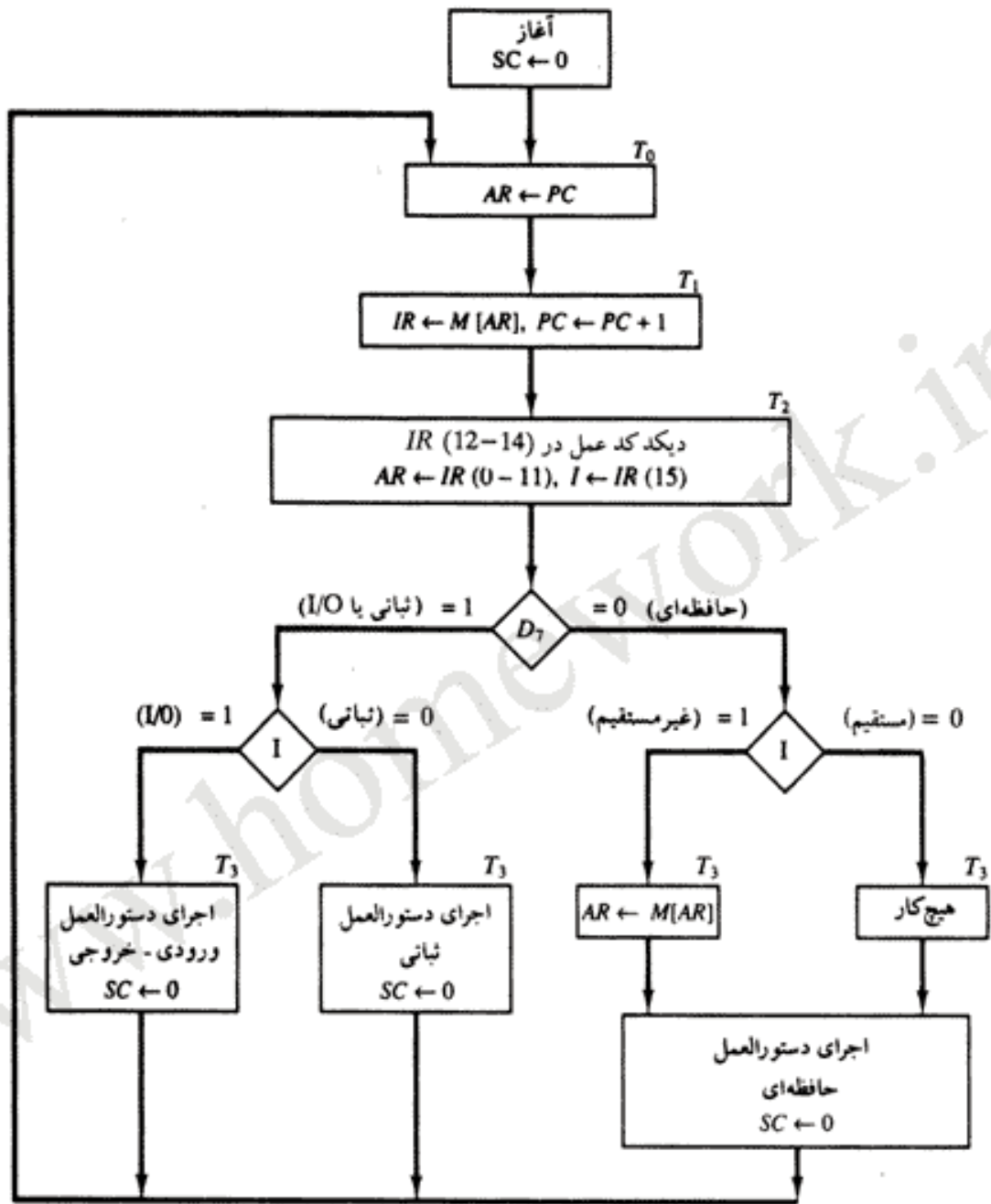
تعیین نوع دستورالعمل

سیگنال زمانبندی بعدی که پس از دیکد شدن فعال است T_3 می باشد. در مدت زمان T_3 ، واحد



شکل ۸-۵ انتقال ثبات‌های مربوط به فاز برداشت

کنترل نوع دستوری را که درست چند لحظه قبل خوانده شده معین می‌کند. فلوچارت شکل ۹-۵ آرایش اولیه‌ای را برای سیکل دستورالعمل و چگونگی تعیین نوع دستورات را توسط کنترل پس از دیکد شدن نشان می‌دهد. سه نوع دستورالعمل موجود در کامپیوتر پایه در شکل ۵-۵ دیده می‌شود. اگر کد عمل برابر با 111 باشد خروجی D_7 دیکدر 1 خواهد بود. با ملاحظه شکل ۵-۵ دیده می‌شود که اگر $D_7=1$ باشد دستور از نوع ارجاع به ثبات یا ورودی- خروجی است. برای $D_7=0$ کد عمل یکی از هفت نوع دیگر 000 تا 110 خواهد بود که همگی از نوع ارجاع به حافظه می‌باشند. سپس کنترل اولین



شکل ۵-۹ فلوجارت سیکل دستورالعمل (آرایش اولیه)

بیت دستورالعمل را که در این لحظه در فلیپ فلاپ I است چک می‌کند. اگر $D_7=0$ و $I=1$ باشد دستور از نوع ارجاع به حافظه با آدرس غیرمستقیم است. در اینصورت لازم است که آدرس موثر از حافظه خوانده شود. ریزعمل مربوط به حالت آدرس غیرمستقیم توسط عبارت انتقال ثبات زیر صورت می‌گیرد.

$$AR \leftarrow M[AR]$$

در ابتدا، AR بخش آدرس دستورالعمل را در خود دارد. این آدرس در طول مدت خواندن حافظه بکار می‌رود. کلمه‌ای که توسط آدرس موجود در AR معین می‌شود از حافظه خوانده شده و روی گذرگاه مشترک قرار می‌گیرد. سپس ورودی LD مربوط به AR فعال می‌شود تا آدرس غیرمستقیم 12 بیتی با ارزش کمتر کلمه حافظه را دریافت نماید.

سه نوع دستورالعمل موجود به چهار مسیر فرعی جداگانه تقسیم می‌شوند. عمل انتخاب شده بوسیله سیگنال مرتبط با T₃ فعال می‌گردد که برای چهارمسیر توسط روابط زیر مشخص می‌شود.

$$D_7IT_3 : AR \leftarrow M[AR]$$

$$D_7I'T_3 : \text{هیچ کار}$$

$$D_7I'T_3 : \text{اجرای دستور ارجاع به ثبات}$$

$$D_7IT_3 : \text{اجرای یک دستور ورودی - خروجی}$$

وقتی که دستورالعمل ارجاع حافظه با $I=0$ پیش آید، لازم نیست کاری انجام شود زیرا در این حالت آدرس موثر از قبل در AR است. با این وجود شمارنده SC، وقتی که $D_7T_3 = 1$ باشد، باید افزایش یابد تا اجرای دستورالعمل ارجاع به حافظه با متغیر زمانی T₄ ادامه یابد. یک دستورالعمل ارجاع به ثبات یا ورودی - خروجی با پالس ساعت مربوط به T₃ اجرا می‌شود. پس از اجرای دستور، SC پاک شده و کنترل با $T_0=1$ به فاز برداشت باز می‌گردد.

دقت کنید که شمارنده SC با هر لبه مثبت پالس افزایش یافته و یا پاک می‌شود. ما در این جا این قرارداد را می‌پذیریم که بازاء هر افزایش عبارت $SC \leftarrow SC + 1$ را ننویسیم ولی بطور ضمنی می‌دانیم که کنترل به مرحله بعدی منتقل می‌شود. چنانچه قرار باشد SC پاک شود عبارت $SC \leftarrow 0$ را خواهیم نوشت. در این بخش انتقال‌های ثباتی لازم برای اجرای دستورات ارجاع به ثبات ارائه شده است. دستورالعمل‌های ارجاع به حافظه در بخش بعد شرح داده شده‌اند. دستورالعمل‌های ورودی - خروجی در بخش ۵-۷ توضیح داده شده‌اند.

دستورالعمل‌های ارجاع به ثبات

این نوع دستورات وقتی که $D_7=1$ و $I=0$ است توسط کنترل قابل تشخیص‌اند. بیت‌های 0 تا 11 این دستورات برای تعیین یکی از 12 دستورالعمل بکار می‌روند. این 12 بیت در (0-11) IR قرار دارند و در طول T₂ نیز به AR منتقل شده‌اند. توابع کنترل و ریزعمل‌ها برای دستورات ارجاع به ثبات در جدول ۳-۵ لیست شده‌اند. این دستورالعمل‌ها با انتقال پالس ساعت در T₃ اجرا می‌شوند. هر یک از توابع کنترل به یک $D_7I'T_3$ نیاز دارند که ما آن را به I نشان می‌دهیم. توابع کنترل توسط یکی از بیت‌های (0-11) IR از هم متمایز می‌گردند. با تخصیص نماد یا سمبل B_i به بیت i از IR، همه توابع کنترلی را می‌توان با B_i نشان داد. مثلاً دستورالعمل CLA دارای کد مبنای شانزده 7800 است (جدول ۲-۵)

جدول ۵-۳ اجرای دستورالعمل‌های ثابتی

مشترک در همه دستورالعمل‌های ثابتی $D_7I'T_3 = r$			
بیتی در IR (0-11) که عمل را مشخص می‌کند $IR(i) = B_i$			
	$r:$	$SC \leftarrow 0$	پاک کردن SC
CLA	$rB_{11}:$	$AC \leftarrow 0$	پاک کردن AC
CLE	$rB_{10}:$	$E \leftarrow 0$	پاک کردن E
CMA	$rB_6:$	$AC \leftarrow \overline{AC}$	منم کردن AC
CME	$rB_6:$	$E \leftarrow \overline{E}$	منم کردن E
CIR	$rB_7:$	$AC \leftarrow shr AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	شیفت چرخشی به راست
CIL	$rB_6:$	$AC \leftarrow shl AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	شیفت چرخشی به چپ
INC	$rB_5:$	$AC \leftarrow AC + 1$	افزایش AC
SPA	$rB_5:$	If $(AC(15) = 0)$ then $(PC \leftarrow PC + 1)$	گذر در صورت مثبت بودن
SNA	$rB_5:$	If $(AC(15) = 1)$ then $(PC \leftarrow PC + 1)$	گذر در صورت منفی بودن
SZA	$rB_5:$	If $(AC = 0)$ then $(PC \leftarrow PC + 1)$	گذر در صورت صفر بودن AC
SZE	$rB_5:$	If $(E = 0)$ then $(PC \leftarrow PC + 1)$	گذر در صورت صفر بودن E
HLT	$rB_0:$	$S \leftarrow 0$	توقف کامپیوتر

ملاحظه شود)، که معادل عدد دودویی 0111 1000 0000 0000 می‌باشد. اولین بیت 0 است که معادل با I' می‌باشد. سه بیت بعدی کد عمل را تشکیل می‌دهند و از خروجی D_7 دیکدر قابل تشخیص‌اند. بیت 11 در IR برابر 1 است که از B_{11} قابل تشخیص است. بنابراین تابع کنترلی که ریز عملیات را برای این دستور آغاز می‌کند $D_7I'T_3B_{11} = rB_{11}$ خواهد بود. اجرای دستورالعمل ارجاع به ثابت در زمان T_3 پایان می‌پذیرد. شمارنده SC پاک شده و کنترل برای برداشت دستوری بعدی با T_0 به عقب برمی‌گردد.

هفت دستور اول ثابتی، اعمال پاک کردن، مکمل سازی، چرخش و افزایش روی ثابت‌های AC و E را انجام می‌دهند. چهار دستوری بعدی موجب گذر^۱ (رد شدن) از دستور بعدی در صورت برقرار بودن یک شرط معین می‌شود. این گذر با افزایش PC صورت می‌گیرد (علاوه برافزایش آن در فاز برداشت دستورالعمل در فاز T_1). عبارت‌های کنترل شرطی را باید بخشی از شرایط کنترلی تلقی کرد. هرگاه بیت علامت $AC, AC(15) = 0$ شود محتوای آن یک عدد مثبت است و اگر $AC = 1$ باشد منفی است. اگر تمام فلیپ فلاپ‌های ثابت صفر باشند محتوای AC صفر است. دستور HLT فلیپ فلاپ استارت - استاپ^۲ را پاک می‌کند که موجب توقف شمارنده SC می‌گردد. برای بکار انداختن مجدد کامپیوتر فلیپ - فلاپ استارت - استاپ باید بصورت دستی 1 شود.

۵-۶ دستورالعمل‌های ارجاع به حافظه

در تعیین ریز عمل‌های مورد نیاز برای اجرای هر دستورالعمل، لازم است تابعی که باید آنرا اجرا کند دقیقاً تعریف شود. با نگرشی به جدول ۵-۲، که در آن دستورالعمل‌ها تعریف شدند، می‌بینیم که برخی

1- skip جدول ۵-۳ اجرای دستورات ارجاع به ثابت

2- Start - stop

از دستورات شرح مبهمی دارند. دلیل این است که این دستورات با جملات طولانی توصیف شده‌اند و فضای کافی برای چنین توضیح طولانی در جدول وجود ندارد. حال ما نشان خواهیم داد که دستور ارجاع به حافظه کاملاً با توجه به علائم و زبان انتقال ثبات قابل تعریف است.

جدول ۴-۵ هفت دستور ارجاع به حافظه را نشان می‌دهد. خروجی دیکد شده D_i برای $i=0,1,2,3,4,5,6$ متعلق به هر دستور نیز در جدول آورده شده است. آدرس موثر در ثبات AR و در مقطع T_2 بهنگام $I=0$ و یا در T_3 با $I=1$ جای می‌گیرد. اجرای دستورهای حافظه‌ای با سیگنال زمانبندی T_4 شروع می‌شود. توصیف سبلیک هر دستور هم در جدول برحسب علائم انتقال ثبات مشخص شده است. اجرای واقعی هر دستور در سیستم نیاز به یک رشته ریزعمل دارد. دلیل این است که داده‌های ذخیره شده در حافظه نمی‌توانند مستقیماً پردازش شوند. این داده‌ها باید ابتدا از یک حافظه خوانده شده و در ثباتی قرار گیرند که مدارهای منطقی بتوانند روی آنها عمل کنند. حال ما عملکرد هر دستور را توضیح داده و توابع کنترل و ریزعمل‌های لازم را برای اجرای آنها لیست می‌کنیم. فلوچارتی که تمام این ریز اعمال را در اختیار می‌گذارد در آخر این بخش آمده است.

AND با AC

این دستور عمل منطقی AND بر روی جفت بیت‌های AC و حافظه‌ای که توسط آدرس موثرش معین می‌شود را انجام می‌دهد. نتیجه عمل فوق در انتها به AC منتقل می‌شود. ریزعمل‌هایی که این دستورالعمل را اجرا می‌کنند عبارتند از

$$D_0 T_4: DR \leftarrow M[AR]$$

$$D_0 T_5: AC \leftarrow AC \wedge DR, SC \leftarrow 0$$

تابع کنترل این دستور خروجی دیکدر D_0 را بکار می‌برد زیرا این خروجی وقتی که عمل AND با

جدول ۴-۵ دستورالعمل‌های حافظه‌ای

سمبل	دیکدر عمل	توضیح سبلیک
AND	D_0	$AC \leftarrow AC \wedge M[AR]$
ADD	D_1	$AC \leftarrow AC + M[AR], E \leftarrow C_{out}$
LDA	D_2	$AC \leftarrow M[AR]$
STA	D_3	$M[AR] \leftarrow AC$
BUN	D_4	$PC \leftarrow AR$
BSA	D_5	$M[AR] \leftarrow PC, PC \leftarrow AR + 1$
ISZ	D_6	$M[AR] \leftarrow M[AR] + 1$ If $M[AR] + 1 = 0$ then $PC \leftarrow PC + 1$

کد 000 اجرا می شود فعال می گردد. دو سیگنال زمانبندی برای اجرای دستور فوق لازم است. لبه پالس مربوط به سیگنال T_4 ، عملوند را از حافظه به DR انتقال می دهد. لبه پالس بعدی یعنی T_5 ، نتیجه عمل AND بر روی AC و DR را به AC منتقل و SC را هم صفر می کند که این خود سبب واگذاری کنترل به سیگنال T_0 می گردد تا سیکل دستورالعمل جدیدی آغاز شود.

AC با ADD

این دستور محتوای حافظه مشخص شده توسط آدرس موثر را با مقدار AC جمع می کند. حاصل جمع به AC و رقم نقلی خروجی C_{out} به فلیپ فلاپ E (بیت گسترش انباره)¹ منتقل می شود. ریز اعمال لازم برای این دستور عبارتست از

$$D_1 T_4: DR \leftarrow M[AR]$$

$$D_1 T_5: AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$$

مشابه با دستور قبل دو سیگنال زمانی T_4 و T_5 بکار رفته اند ولی خروجی D_1 دیکدر بجای D_0 که در عمل AND بکار رفت فعال است. پس از برداشت و دیکدر دستور تنها یک خروجی دیکدر فعال خواهد بود و این خروجی رشته ریز عملهایی که واحد کنترل هنگام اجرای یک دستورالعمل حافظه ای دنبال می کند را تعیین می نماید.

LDA: بار کردن AC

این دستور محتوای حافظه ای که توسط آدرس موثر مشخص شده را به AC منتقل می کند.

$$D_2 T_4: DR \leftarrow M[AR]$$

$$D_2 T_5: AC \leftarrow DR, SC \leftarrow 0$$

با بازنگری مجدد شکل ۴-۵ ملاحظه می شود که مسیر مستقیمی از گذرگاه به AC وجود ندارد. مدار جمع کننده و منطقی، اطلاعات را از DR دریافت می کند که قابل انتقال به AC است. بنابراین لازم است تا ابتدا کلمه حافظه به DR فراخوانده شده و سپس از DR به AC منتقل شود. دلیل عدم ارتباط گذرگاه به AC، تاخیری است که در مدارهای جمع کننده و منطقی وجود دارد. در اینجا فرض بر این است که زمان لازم برای خواندن از حافظه و انتقال به گذرگاه و مدارهای جمع کننده و منطقی بیش از یک پالس ساعت است. با عدم اتصال گذرگاه به ورودی های AC می توانیم زمان یک سیکل ساعت را برای هر ریز عمل حفظ کنیم.

1- Extended AC

STA: ذخیره کردن AC در حافظه

این دستور محتوای AC را در حافظه‌ای که با آدرس موثرش مشخص شده ذخیره می‌نماید. چون خروجی AC به گذرگاه وصل است و ورودی حافظه هم به گذرگاه متصل شده، می‌توانیم این دستور را با یک ریز عمل انجام دهیم.

$$D_3 T_4: M[AR] \leftarrow AC, SC \leftarrow 0$$

BUN: انشعاب بدون شرط

این دستور برنامه را به دستورالعملی که توسط آدرس موثرش مشخص شده منتقل می‌نماید. بخاطر داشته باشید که PC آدرس دستوری که در سیکل بعدی خوانده می‌شود را در خود نگه میدارد. PC در زمان T_1 افزایش می‌یابد تا برای دستور بعدی در برنامه آماده باشد. دستورالعمل BUN به برنامه نویس امکان می‌دهد تا دستوری را در خارج از رشته متوالی برنامه مشخص کند و در این حالت گوئیم برنامه بدون شرط انشعاب یافته است. این دستورالعمل با یک ریز عمل اجرا می‌شود.

$$D_4 T_4: PC \leftarrow AR, SC \leftarrow 0$$

آدرس موثر از AR و از طریق گذرگاه مشترک به PC منتقل می‌شود. با پاک شدن SC، کنترل به T_0 منتقل می‌گردد. سپس دستور بعدی توسط مقدار جدید در PC از حافظه برداشت و اجرا می‌گردد.

BSA: انشعاب با ذخیره آدرس برگشت

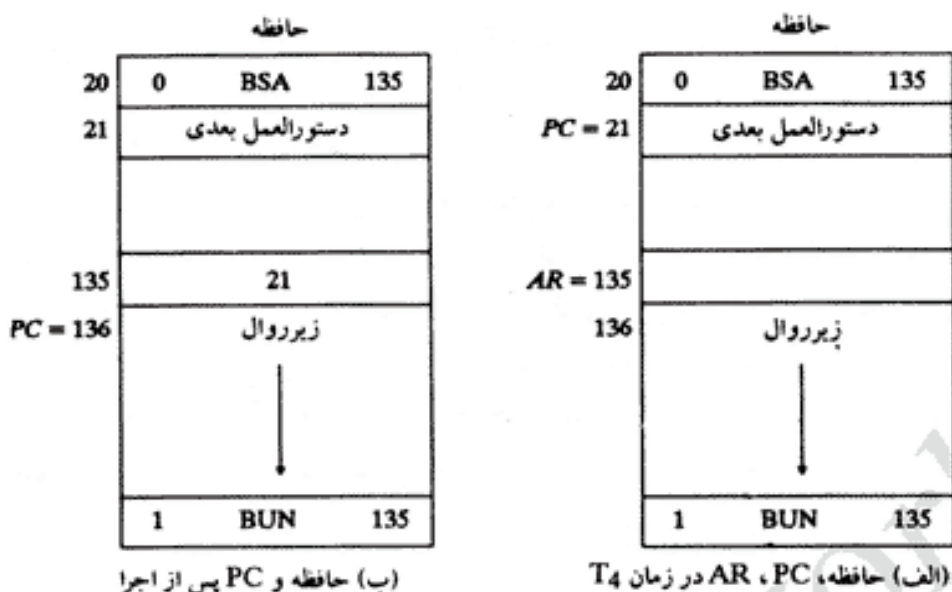
این دستور برای انشعاب به بخشی از برنامه که زیرروال¹ یا رویه خوانده می‌شود مفید است. هر وقت این دستور اجرا شود BSA آدرس دستور بعدی را که در PC موجود است در حافظه‌ای که توسط آدرس موثر معین می‌شود ذخیره می‌نماید. سپس آدرس موثر بعلاوه 1 به PC منتقل می‌شود تا بعنوان آدرس اولین دستور در زیرروال مورد استفاده قرارگیرد. این عمل در جدول ۴-۵ همراه با عبارت انتقال ثبات در زیر آمده است.

$$M[AR] \leftarrow PC, PC \leftarrow AR+1$$

یک مثال عددی که نشان دهنده چگونگی بکار رفتن این دستور است در شکل ۱۰-۵ دیده می‌شود فرض بر این است که دستور BSA در حافظه 20 قرار گرفته باشد. بیت 1 برابر 0 و بخش آدرس دستور 135 تصور می‌شود. پس از فازهای برداشت و دیکد، PC حاوی 21 خواهد بود که آدرس دستور بعدی در برنامه است (این همان آدرس بازگشت است). AR حاوی آدرس موثر 135 است. این مطلب در قسمت (الف) شکل ملاحظه می‌گردد. دستورالعمل BSA عمل عددی زیر را اجرا می‌نماید.

$$M[135] \leftarrow 21, PC \leftarrow 135+1 = 136$$

1- Subroutine



شکل ۱۰-۵

نتیجه این عمل در بخش (ب) از شکل مزبور دیده می شود. آدرس بازگشت 21 در حافظه 135 ذخیره شده و کنترل از 136 زیرروال بکار خود ادامه می دهد. بازگشت به برنامه اصلی (در آدرس 21) توسط دستور BUN که در انتهای زیرروال قرار دارد انجام می شود. وقتی این دستور اجرا شود کنترل آدرس موثر را که در مکان 135 قرار دارد و همان آدرس 21 است می خواند. پس از اجرای BUN، آدرس 21 به PC منتقل می شود و در نتیجه دستور واقع در آدرس بازگشت اجرا خواهد شد.

دستورالعمل BSA عملی را انجام می دهد که معمولاً به آن فراخوانی^۱ زیرروال می گویند. دستور BUN در انتهای زیرروال عملی را که به آن بازگشت از زیرروال گویند انجام می دهد. در بسیاری از کامپیوترهای تجاری آدرس برگشت از زیرروال در یکی از ثباتهای پردازشگر یا در بخشی از حافظه بنام پشته^۲ ذخیره می شود. این موضوع در بخش ۷-۸ مشروحاً توضیح داده خواهد شد.

امکان اجرای دستور BSA در یک پالس ساعت، در صورتی که از سیستم گذرگاه کامپیوتر پایه استفاده کنیم، میسر نیست. برای استفاده صحیح از حافظه و گذرگاه، دستور BSA باید با دو ریزعمل زیر اجرا شود

$$D_5 T_4: M[AR] \leftarrow PC, AR \leftarrow AR+1$$

$$D_5 T_5: PC \leftarrow AR, SC \leftarrow 0$$

سیگنال زمانبندی T4 یک عمل نوشتن در حافظه را آغاز می کند و طی آن محتوای PC را روی گذرگاه قرار می دهد، و ورودی INR مربوط به AR را فعال می کند. نوشتن در حافظه و افزایش AR تالبه پالس

1- Call

2- Stack

بعدی تکمیل می‌گردد. در T₅ گذرگاه برای انتقال AR به PC مورد استفاده قرار می‌گیرد.

ISZ: افزایش و گذر اگر نتیجه صفر

این دستور کلمه‌ای که توسط آدرس موثر مشخص می‌شود را یک واحد افزایش می‌دهد و اگر مقدار افزایش یافته 0 شود، PC را یک واحد افزایش می‌دهد. برنامه نویس معمولاً یک عدد منفی را (بفرم متمم 2) در حافظه قرار می‌دهد. این عدد منفی مرتباً افزایش می‌یابد و نهایتاً به صفر می‌رسد. در این لحظه PC یک واحد اضافه می‌شود تا از دستور بعدی در برنامه گذر¹ کند.

چون امکان افزایش کلمه در داخل حافظه ممکن نیست، لازم است کلمه به DR منتقل و در DR یک واحد افزایش و سپس به حافظه بازگردانده شود. این کار با ریزعمل‌های زیر انجام می‌شود:

$$D_6 T_4: DR \leftarrow M[AR]$$

$$D_6 T_5: DR \leftarrow DR + 1$$

$$D_6 T_6: M[AR] \leftarrow DR \text{ if } (DR=0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$$

فلوچارت کنترل

فلوچارت نمایش اجرای ریزعمل‌های مربوط به هفت دستورالعمل ارجاع به حافظه در شکل ۱۱-۵ نشان داده شده است. توابع کنترل در بالای هر چهارگوشه آمده است. ریزعمل‌های اجرا شده در T₄، T₅ و T₆ به مقدار کد عمل بستگی دارد. این مطلب در فلوچارت توسط شش مسیر مختلف مشخص شده که هر یک از آنها پس از دیکد شدن دستورالعمل توسط کنترل انتخاب می‌شوند. شمارنده برنامه SC در هر حالت پس از آخرین سیگنال زمانبندی، 0 می‌شود. این امر باعث انتقال کنترل به سیگنال زمانبندی T₀ برای آغاز سیکل دستورالعمل بعدی می‌گردد.

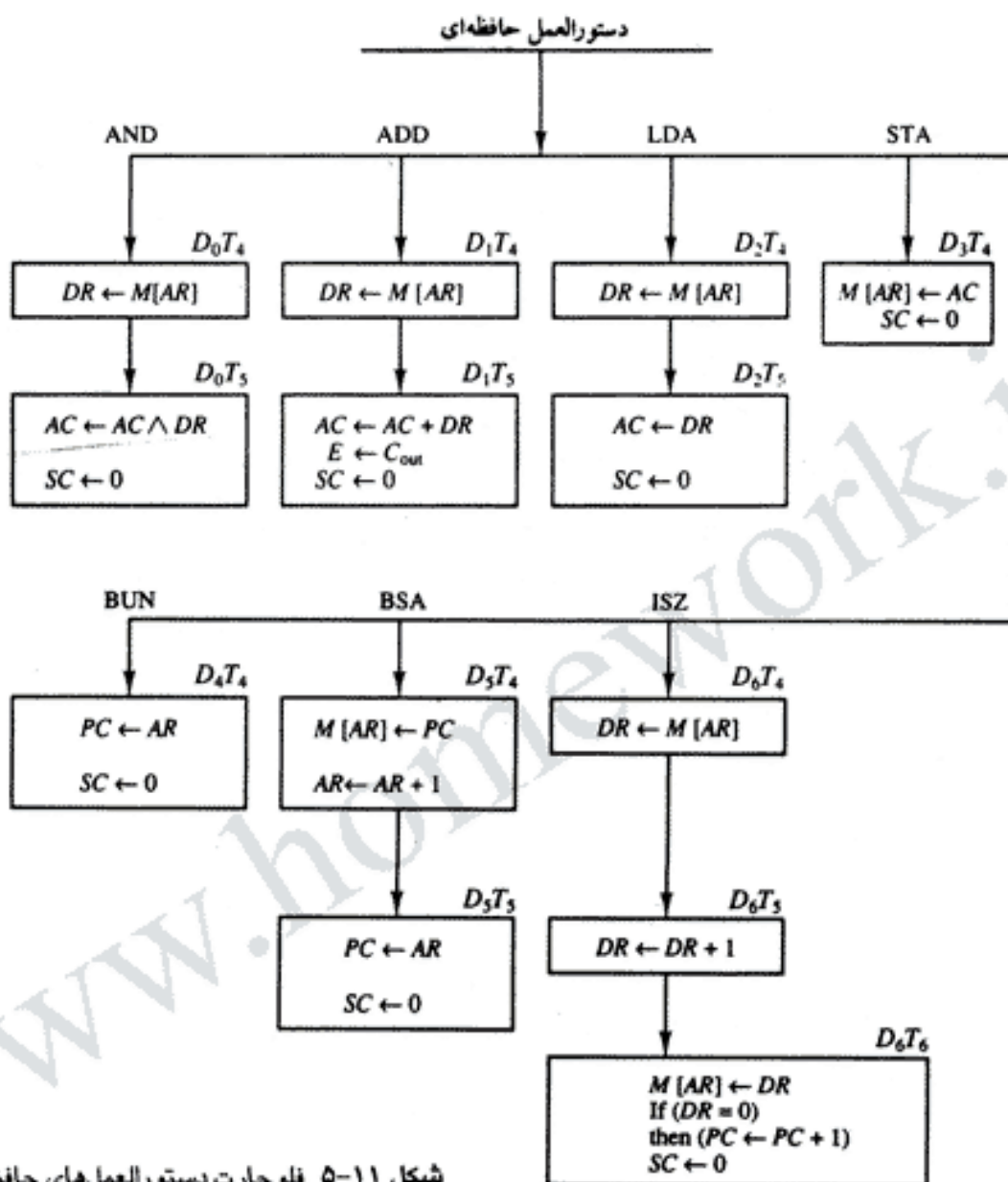
توجه کنید که برای اجرای طولانی‌ترین دستورالعمل (ISZ) به هفت سیگنال زمانبندی نیاز داریم. لذا کامپیوتر را می‌توان با یک شمارنده سه بیتی SC طراحی کرد. دلیل بکارگیری چهاربیت در SC این است که سیگنال‌های زمانی لازم برای سایر دستورات در بخش تمرینات میسر باشد.

۷-۵ ورودی - خروجی و وقفه^۲

یک کامپیوتر مادامی که با محیط خارج خود ارتباط برقرار نکرده است نمی‌تواند مفید واقع شود. دستورات و داده‌های ذخیره شده در حافظه باید از برخی وسایل خارجی متصل به ورودی گرفته شوند. نتایج محاسبات هم باید از طریق نوعی وسیله خروجی به کاربر انتقال یابد. کامپیوترهای تجاری مجهز به انواع وسایل ورودی و خروجی هستند. برای نمایشی از اساسی‌ترین ملزومات ارتباطی ورودی و

1- Skip

2- Interrupt



شکل ۱۱-۵ فلوجارت دستورالعمل‌های حافظه‌ای

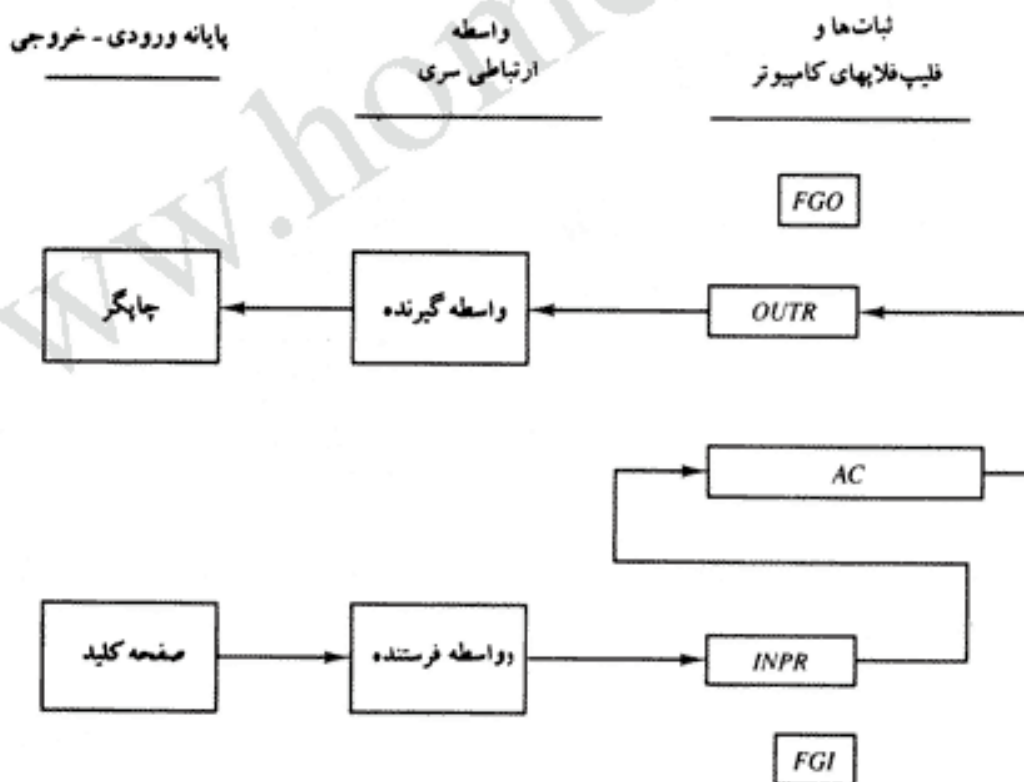
خروجی، ما واحد پایانه‌ای را با صفحه کلید و چاپگر نشان خواهیم داد. سازمان ورودی - خروجی در فصل ۱۱ بحث خواهد شد.

آرایش ورودی - خروجی

پایانه، اطلاعات سری را ارسال و دریافت می‌کند. هر یک از کمپیت‌های اطلاعاتی حاوی یک کد الفباء عددی هشت بیتی است. اطلاعات سری وارده از صفحه کلید بداخل INPR شیفتر داده می‌شود. اطلاعات سری چاپگر در OUTFR ذخیره می‌گردد. این دو ثبات بایک مدار واسط وسیله جانبی بطور سری وبا

AC ارتباط موازی دارند. آرایش ورودی - خروجی در شکل ۱۲-۵ دیده می شود. مدار واسطه وسیله ارسال کننده^۱ (فرستنده) اطلاعات سری را از صفحه کلید دریافت و به INPR می فرستد. مدار واسطه دریافت کننده^۲ اطلاعات را از OUTR دریافت و به چاپگر ارسال می دارد. عملکرد مدار واسطه در بخش ۱۱-۳ تشریح شده است.

ثبات INPR از هشت بیت تشکیل و اطلاعات الفبا عددی را در خود نگه می دارد. پرچم یک بیتی FGI یک فلیپ فلاپ کنترل است و این پرچم هر وقت اطلاعات جدیدی در وسیله ورودی موجود باشد برابر 1 می شود و وقتی کامپیوتر این اطلاعات را پذیرفت برابر 0 می گردد. این پرچم برای همزمان کردن سرعت جریان اطلاعات بین وسیله ورودی و کامپیوتر لازم است. فرآیند انتقال اطلاعات بقرار زیر است. در ابتدا به پرچم ورودی FGI مقدار 0 داده می شود. وقتی کلیدی روی صفحه کلید فشار داده می شود، یک کد الفبا عددی هشت بیتی بطور سری به INPR شیفت پیدا کرده و پرچم FGI برابر 1 می گردد. مادامی که این پرچم 1 است محتوای INPR قابل تغییر بوسیله کلید دیگر نیست. کامپیوتر بیت پرچم را چک می نماید، اگر پرچم 1 باشد، اطلاعات از INPR بطور موازی به AC منتقل و FGI پاک



شکل ۱۲-۵ آرایش ورودی-خروجی

1- Transmitter Interface

2- Receiver Interface

می شود. به محض پاک شدن پرچم، اطلاعات جدید حاصل از کلید جدید می تواند وارد INPR گردد. ثبات خروجی OUTR بطور مشابه کار می کند ولی جهت انتقال اطلاعات معکوس است. در آغاز پرچم خروجی FGO برابر 1 است. کامپیوتر این پرچم را چک می کند؛ اگر برابر 1 باشد، اطلاعات از AC بطور موازی به OUTR منتقل و FGO پاک می شود. وسیله جانبی خروجی اطلاعات کد شده را پذیرفته و کاراکتر مربوطه را چاپ و در پایان FGO را 1 می کند. کامپیوتر مادامی که FGO برابر 0 است اطلاعاتی را در OUTR ذخیره نمی کند زیرا صفر بودن FGO نشانه این است که خروجی در حال چاپ کاراکتر است.

دستورات ورودی - خروجی

دستورات ورودی و خروجی برای تبادل اطلاعات از ثبات AC، و چک بیت های پرچم برای کنترل وقفه مورد نیازند. دستورات ورودی و خروجی دارای کد عملیاتی 1111 بوده و هنگامی که $D_7=1$ و $I=1$ باشد توسط کنترل قابل تشخیص اند. بیت های دیگر معین کننده دستورالعمل خاص مورد نظر هستند. توابع کنترل و ریز عمل های ورودی - خروجی در جدول 5-5 لیست شده اند. اجرای این دستورات با لجه پالس ساعت T_3 آغاز می شود. همه توابع کنترل نیاز به رابطه بولی D_7IT_3 دارند که برای راحتی، آنرا با P نشان می دهیم. هر تابع کنترل خاص مورد نظر توسط یکی از بیت های $IR(6-11)$ از دیگری تفکیک می شود. با فرض نمایش بیت IR با B_i تمام توابع کنترل را می توان با PB_i نشان داد که در آن i از 6 تا 11 تغییر می کند. شمارنده SC وقتی که $P=D_7IT_3=1$ باشد پاک می شود.

دستور INP اطلاعات ورودی را از INPR به هشت بیت پائین رتبه AC انتقال داده و پرچم ورودی را پاک می کند. دستور OUT هشت بیت پائین رتبه AC را به ثبات OUTR منتقل و پرچم خروجی را 0 می نماید. دو دستور بعدی در جدول (5-5) وضعیت پرچم ها را تست نموده و در صورت 1 بودن موجب گذر از دستور بعدی می شوند. دستورالعملی که از آن گذر می شود معمولاً یک دستور انشعاب برای بازگشت و کنترل مجدد پرچم است. اگر پرچم 0 باشد از دستور انشعاب گذر نمی شود. ولی اگر پرچم 1 باشد دستور انشعاب گذر شده و دستوری از نوع ورودی یا خروجی اجرا گردد. (مثال هایی از برنامه های ورودی و خروجی در بخش 8-6 آمده است). دو دستور آخر یگ قلیپ فلاپ IEN وقفه را

جدول 5-5 دستورالعمل های ورودی - خروجی

		(مشترک در همه دستورالعمل های ورودی - خروجی) $D_7IT_3 = p$	
		$IR(i) = B_i$ [بیتی در $IR(0-11)$ که دستورالعمل را مشخص می کند]	
	$p:$	$SC \leftarrow 0$	پاک کردن SC
INP	$pB_{11}:$	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	دریافت کاراکتر ورودی
OUT	$pB_{10}:$	$OUTR \leftarrow AC(0-7), FGO \leftarrow 0$	ارسال کاراکتر خروجی
SKI	$pB_9:$	If $(FGI = 1)$ then $(PC \leftarrow PC + 1)$	گذر بر اساس پرچم ورودی
SKO	$pB_8:$	If $(FGO = 1)$ then $(PC \leftarrow PC + 1)$	گذر بر اساس پرچم خروجی
ION	$pB_7:$	$IEN \leftarrow 1$	روشن کردن فعال ساز وقفه
IOF	$pB_6:$	$IEN \leftarrow 0$	خاموش کردن فعال ساز وقفه

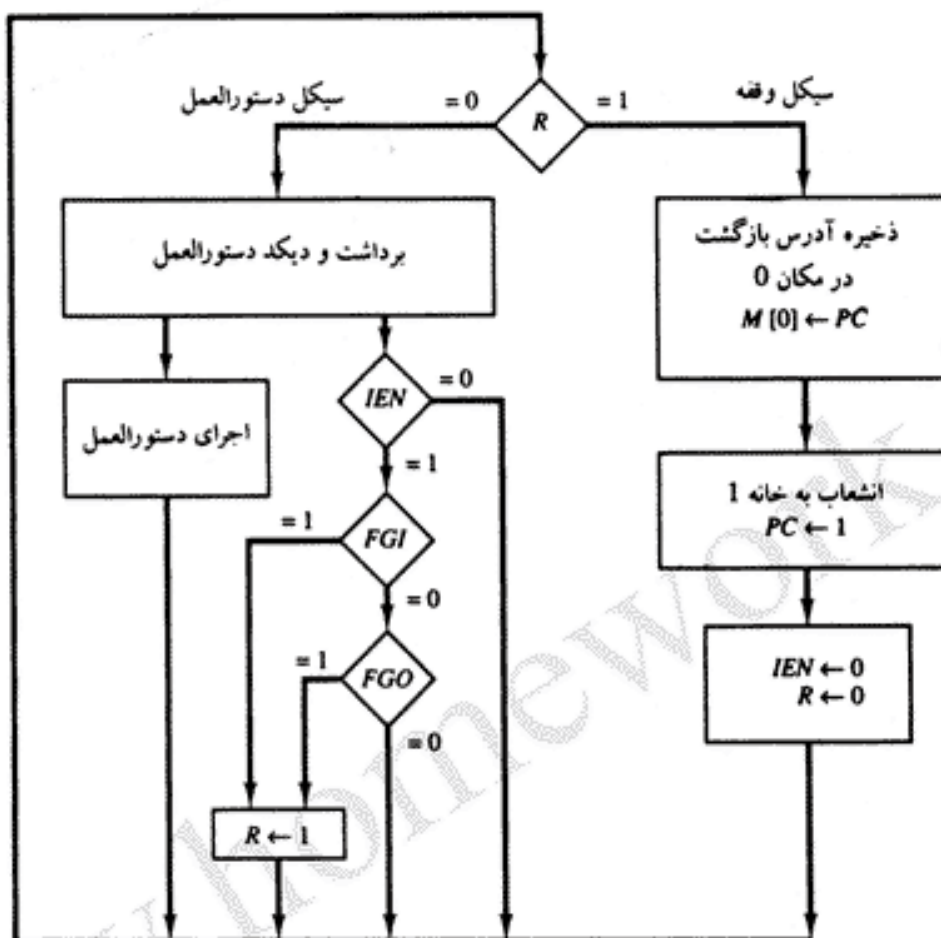
پاک و یا برقرار¹ می نمایند (می نشانند). هدف از IEN به همراه بحث مربوط به وقفه بیان شده است.

وقفه برنامه

فرآیند ارتباطی بحث فوق بنام انتقال تحت کنترل برنامه خوانده می شود. کامپیوتر بطور مداوم بیت پرچم را چک کرده و وقتی آن را 1 بیابد انتقال را آغاز می کند. بدلیل تفاوت سرعت جریان اطلاعات بین کامپیوتر و وسیله ورودی - خروجی کارایی این نوع انتقال پائین است. برای درک این مطلب، کامپیوتری را در نظر بگیرید که می تواند سیکل دستور را در 1 μ s انجام دهد. فرض کنید که وسیله ورودی - خروجی می تواند اطلاعات را حداکثر با میزان 10 کاراکتر در ثانیه منتقل نماید. این سرعت معادل با 100,000 میکروثانیه برای هر کاراکتر می باشد. وقتی که کامپیوتر بیت پرچم را چک می کند و تصمیم می گیرد اطلاعات را انتقال ندهد دو دستورالعمل اجرا می شود. این بدان معنی است که در سرعت ماکزیمم، کامپیوتر بین هر دو انتقال 50000 بار پرچم را چک خواهد کرد. کامپیوتر مادامی که در حال چک کردن پرچم هاست بعوض انجام کار مفیدی وقت را تلف می کند.

روشی دیگر این است که وسیله خارجی بهنگام آماده بودن انتقال، کامپیوتر را آگاه سازد. در این هنگام کامپیوتر مشغول به کارهای دیگری است. این نوع انتقال از امکان وقفه استفاده می کند. کامپیوتر در حالت اجرای یک برنامه، دیگر نیاز به چک کردن پرچم ها ندارد. با این وجود هر وقت پرچمی 1 شد کامپیوتر از پیشروی در برنامه جاری بازداشته شده و از 1 شدن پرچم آگاه می شود. سپس کامپیوتر انتقال اطلاعات را از ورودی یا خروجی انجام می دهد. در پایان به برنامه جاری بازگشته و به آنچه قبل از وقفه انجام می داد ادامه می دهد.

فلیپ فلاپ تواناساز (فعال ساز) وقفه IEN را با دو دستور می توان 0 یا 1 کرد. وقتی IEN پاک شود (با دستور IOF) پرچم ها قادر به ایجاد وقفه در کامپیوتر نیستند. وقتی IEN برقرار شود (1 شود) (با دستور ION) کامپیوتر قابل توقف است. این دو دستور برنامه نویس را قادر می سازند تا از امکانات وقفه استفاده نمایند. نحوه واکنش کامپیوتر به وقفه با توجه به شکل 13-5 قابل تشریح است. یک فلیپ فلاپ R در کامپیوتر منظور شده است. وقتی $R=0$ باشد کامپیوتر وارد سیکل دستورالعمل می گردد. به موازات سیکل اجرای دستورالعمل، IEN توسط کنترل چک می شود. 0 بودن این فلیپ فلاپ بمعنی این است که برنامه نویس مایل به استفاده از وقفه نیست. لذا کنترل با دستور بعدی بکار ادامه می دهد. اگر $IEN=1$ باشد کنترل بیت های پرچم را چک می نماید. اگر هر دو پرچم 0 باشند نه ورودی و نه خروجی آماده انتقال اطلاعات نیستند. در این حالت کنترل با دستور بعدی به کار ادامه می دهد. اگر ضمن $IEN=1$ هر یک از دو پرچم ورودی یا خروجی 1 شوند $R=1$ می گردد. در انتهای فاز اجرای دستور، کنترل R را چک می نماید و چنانچه مقدار آن برابر 1 باشد کنترل در عوض اجرای دستور بعدی وارد سیکل وقفه می گردد. سیکل وقفه، پیاده سازی سخت افزاری یک انشعاب و ذخیره آدرس برگشت است. آدرس موجود در

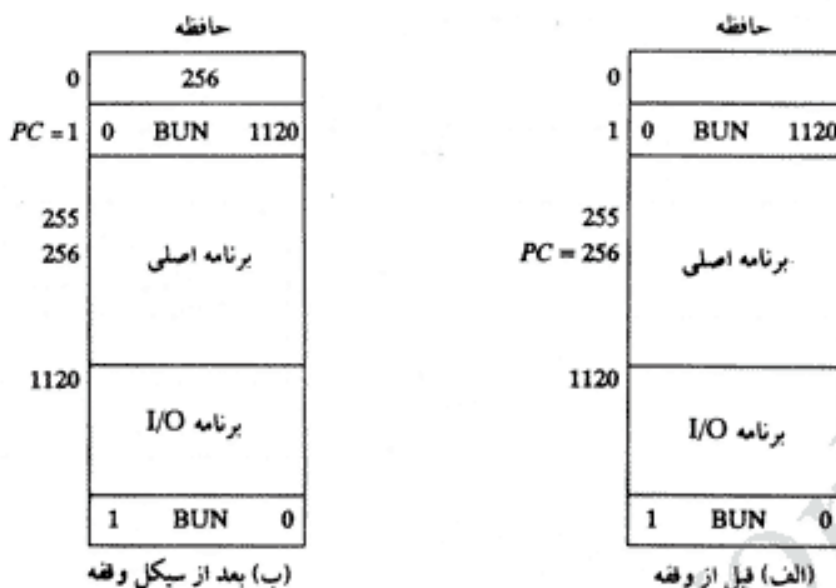


شکل ۱۴-۵ فلوجارت سیکل وقفه

PC در یک مکان خاص ذخیره می شود تا بهنگام بازگشت برنامه به دستوری که آنرا ترک کرده است از آن استفاده نماید. این مکان ممکن است یک ثبات پردازشگر، یک حافظه پشته یا یک مکان خاصی از حافظه باشد. در اینجا ما حافظه با آدرس 0 را برای نگهداری آدرس برگشت انتخاب می کنیم. کنترل سپس آدرس 1 را در PC وارد کرده و IEN و R را پاک می کند در نتیجه هیچ وقفه ای، مادامی که تقاضای وقفه قبلی سرویس نشده باشد اتفاق نمی افتد.

مثالی که نشان دهنده رخدادها در سیکل وقفه است در شکل ۱۴-۵ دیده می شود. فرض کنید که وقفه ای رخ داده و $R=1$ است و بموازات آن هم کنترل مشغول اجرای دستوری در آدرس 255 می باشد. در این زمان، آدرس برگشت در PC برابر با 256 است. برنامه نویس قبلاً برنامه ای را برای سرویس دادن به ورودی و خروجی در حافظه و در آدرس 1120 قرار داده و یک BUN 1120 را در آدرس 1 قرار داده است. این مطلب در شکل ۱۴-۵ (الف) نشان داده شده است.

وقتی کنترل به T_0 رسیده و R را برابر 1 بیابد، وارد سیکل وقفه می گردد. محتویات PC(256) در



شکل ۱۴-۵ مثالی از سیکل وقفه

مکان 0 حافظه ذخیره می شود. PC مساوی 1 می شود و R با 0 پاک می شود. در آغاز سیکل دستورالعمل بعدی، دستوری که از حافظه خوانده شده در آدرس 1 قرار دارد زیرا این عدد همان محتوای PC است. دستور انشعاب واقع در آدرس 1 سبب می شود تا اجرا به برنامه سرویس ورودی - خروجی واقع در آدرس 1120 منتقل گردد. این برنامه پرچم ها را چک و تعیین می کند که کدام پرچم 1 شده است و سپس اطلاعات ورودی یا خروجی را منتقل می نماید. پس از انجام این عمل، دستورالعمل ION برای 1 کردن IEN اجرا می شود (تا وقفه را ادامه دهد) و برنامه به مکانی که وقفه در آن رخ داده بود باز می گردد. دستورالعملی که کامپیوتر را به مکان اولیه اش در برنامه اصلی باز می گرداند یک انشعاب غیر مستقیم است که بخش آدرس آن 0 است. این دستور در انتهای برنامه سرویس I/O قرار دارد. پس از اینکه این دستور از حافظه دریافت شده کنترل به فاز غیر مستقیم (زیرا $I=1$) رفته و آدرس مؤثر را می خواند. آدرس مؤثر در مکان 0 است و همان آدرس برگشتی است که در لحظه بروز وقفه ذخیره شده بود. اجرای دستور غیر مستقیم BUN نشان دادن آدرس برگشت در PC است.

سیکل وقفه

ما حالا آماده ایم تا عبارات انتقال ثبات را برای سیکل وقفه بنویسیم. سیکل وقفه پس از آخرین فاز اجرا، اگر فلیپ فلاپ $R=1$ باشد آغاز می شود. این فلیپ فلاپ به شرطی 1 می شود. که $IEN=1$ و یکی از دو فلیپ فلاپ FGI یا FGO برابر 1 باشد. این وضعیت در هر انتقال پالس ساعتی امکان پذیر است

بجز وقتی که سیگنالهای T_0 ، T_1 و یا T_2 فعال باشند. شرایط 1 شدن فلیپ فلاپ R با عبارت انتقال ثبات زیر می تواند بیان شود:

$$T_0 T_1 T_2 (IEN) (FGI+FGO): R_1 \leftarrow 1$$

نماد + بین FGI و FGO در تابع کنترل یک عمل منطقی OR را بیان می دارد. این عمل با IEN و AND، $T_0 T_1 T_2$ می شود.

حال ما فازهای برداشت و دیکد سیکل دستور را تصحیح می کنیم. بجای استفاده از T_0 ، T_1 و T_2 (طبق شکل ۹-۵) سه سیگنال مذکور را با R' ، AND می نمایم در نتیجه فازهای برداشت و دیکد با سه تابع کنترل $R'T_0$ ، $R'T_1$ و $R'T_2$ تعریف می شوند. دلیل این مورد این است که پس از فاز اجرای دستورالعمل و پاک شدن SC، کنترل فقط با شرط $R=0$ به فاز برداشت خواهد رفت. در غیراینصورت اگر $R=1$ باشد کنترل به سیکل وقفه خواهد رفت. سیکل وقفه آدرس برگشت (موجود در PC) را در آدرس 0 ذخیره ساخته و سپس به حافظه 1 انشعاب کرده و IEN و R و SC را هم 0 می کند. این اعمال با رشته ریزعمل های زیر امکان پذیر است.

$$RT_0: AR \leftarrow 0, TR \leftarrow PC$$

$$RT_1: M[AR] \leftarrow TR, PC \leftarrow 0$$

$$RT_2: PC \leftarrow PC+1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$$

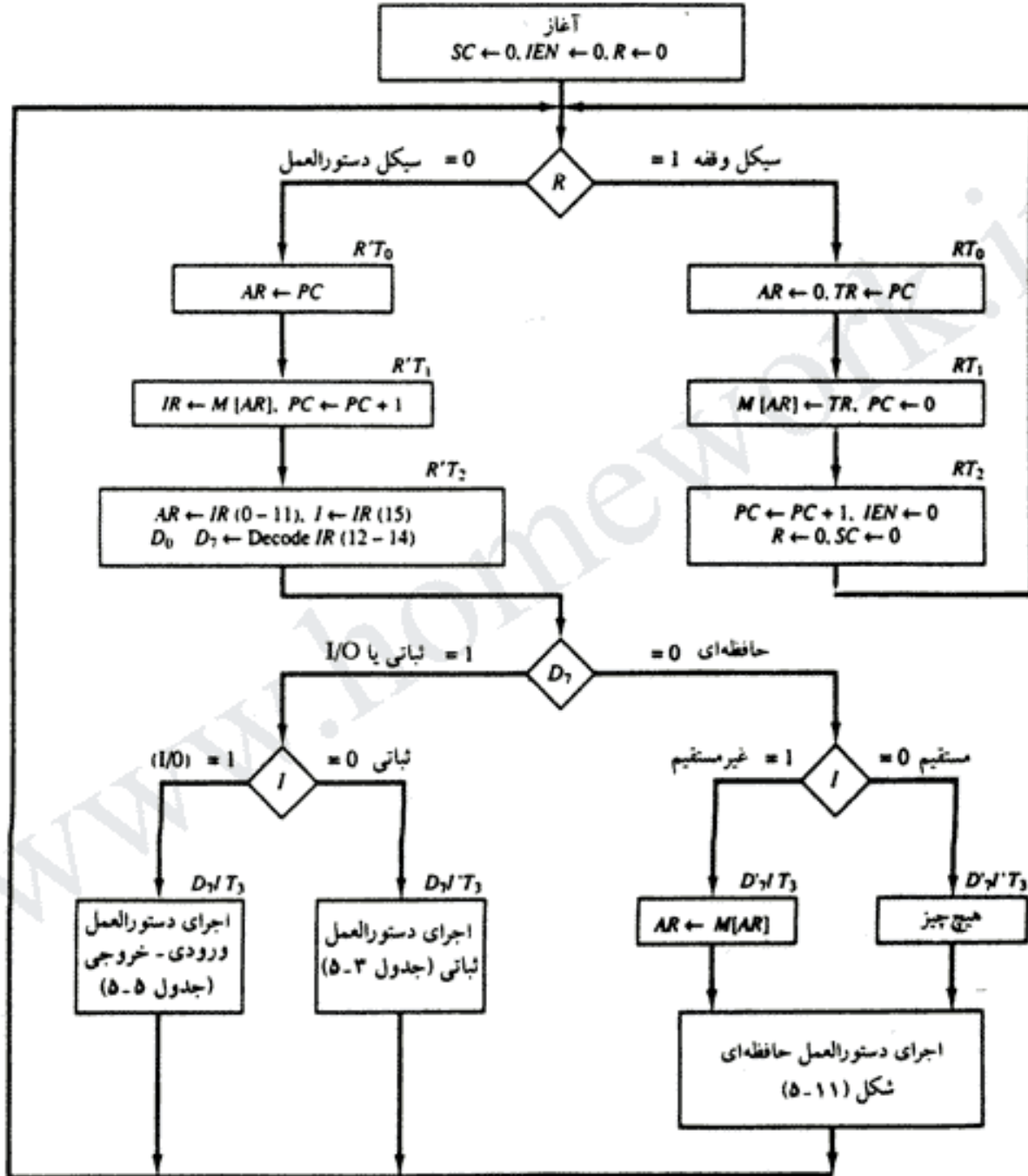
در طول اولین سیگنال زمانبندی، AR پاک می شود و PC به ثبات موقتی بنام TR انتقال می یابد. با دومین سیگنال آدرس برگشت در مکان 0 حافظه ذخیره شده و PC پاک می شود. سومین سیگنال زمانبندی PC را یک واحد افزایش می دهد، IEN و R را پاک می کند و کنترل نیز با پاک شدن SC به T_0 باز می گردد. شروع دستورالعمل بعدی شرط $R'T_0$ را داراست و محتویات PC نیز برابر 1 است. سپس کنترل وارد سیکل دستور شده و دستور BUN را از حافظه 1 برداشت و اجرا می کند.

۵-۸ تشریح کامل کامپیوتر

فلوچارت نهایی سیکل دستورالعمل به همراه سیکل وقفه برای کامپیوتر پایه در شکل ۱۵-۵ نشان داده شده است. فلیپ فلاپ R هر لحظه ممکن است در طول فازهای غیرمستقیم^۱ یا اجرا^۲ برابر 1 شود. کنترل پس از $SC=0$ به T_0 برمی گردد. اگر $R=1$ باشد کنترل وارد سیکل وقفه می گردد. اگر $R=0$ باشد کامپیوتر وارد سیکل دستورالعمل می شود. اگر دستور از نوع ارجاع به حافظه باشد، کامپیوتر در صورت وجود آدرس غیرمستقیم اجرای دستور دیکد شده را طبق شکل ۱۱-۵ ادامه می دهد. اگر دستورالعمل از نوع ارجاع به ثبات باشد، اجرای آن طبق جدول ۳-۵ خواهد بود و بالاخره اگر دستور از نوع ورودی -

1- Indirect

2- Execute



شکل ۵-۱۵ فلوچارت برای اعمال کامپیوتر

خروجی باشد، اجرای آن با یکی از ریز عمل های جدول ۵-۵ می باشد.
طرز کار کامپیوتر را می توان با لیستی از عبارات انتقال ثبات بجای فلوچارت نشان داد. این عمل با سرجمع کردن تمام ریز اعمال در یک جدول انجام می شود. جدول مذکور با استفاده از شکل های ۱۱-۵ و ۱۵-۵ و جدول ۳-۵ و ۵-۵ تکمیل گردیده است.
توابع کنترل و ریز اعمال برای کل کامپیوتر در جدول ۶-۵ آورده شده است. در این جدول عبارات انتقال ثبات، سازمان داخلی کامپیوتر را بفرم فشرده توصیف می کنند. این عبارات اطلاعات لازم را برای طراحی مدارهای منطقی نیز فراهم می آورند. توابع کنترل و عبارات کنترل شرطی که در جدول لیست شده اند توابع بول گیت ها را در واحد کنترل فرموله می کنند. لیست ریز اعمال، نوع ورودی های کنترل لازم برای ثبات ها و حافظه ها را معین می سازد. یک عبارت انتقال ثبات نه فقط برای توصیف سازمان داخلی یک سیستم دیجیتال مفید است بلکه برای مشخص کردن مدارهای منطقی لازم در طراحی نیز می باشد.

۵-۹ طراحی کامپیوتر پایه

کامپیوتر پایه متشکل از بخش های سخت افزاری زیر است.

- ۱- یک واحد حافظه با 4096 کلمه 16 بیتی
- ۲- 9 ثبات: AR, PC, DR, AC, JR, TR, OTR, INPR, SC
- ۳- 7 فلیپ فلاپ: J, K, S, R, E, JEN, FGI و FGO
- ۴- دو دیکدر: یک دیکدر عملیاتی 3x8 و یک دیکدر زمانی 4x16
- ۵- یک گذرگاه مشترک 16 بیتی
- ۶- کنترل منطقی گیتی
- ۷- جمع کننده و مدار منطقی متصل به ورودی AC

واحد حافظه قطعه استاندارد است و بصورت آماده قابل تهیه از منابع تجاری است. ثبات ها از نوع شکل ۱۱-۲ و مشابه آی سی 74163 می باشند. فلیپ فلاپها می توانند از نوع D یا JK مطابق بخش ۶-۱ باشند. دو دیکدر نیز استاندارد و مشابه آنچه در بخش (۲-۲) گفته شد می باشند. سیستم گذرگاه مشترک می تواند با شانزده مولتی پلکسر 8x1 با آرایشی مشابه با شکل ۳-۴ ساخته شود. حالا تصمیم داریم نشان دهیم که چگونه مدار منطقی کنترل را می توان طراحی کرد. بخش بعد طراحی جمع کننده و مدارهای منطقی مربوط به AC را بحث می کند.

گیت های مدار کنترل

دیباگرام مدار کنترل در شکل ۶-۵ دیده می شود. ورودی های این مدار شامل دو دیکدر، فلیپ فلاپ I و بیت های 0 تا 11 ثبات IR می باشند. سایر ورودی ها به کنترل عبارتند از بیت های 0 تا 15 ثبات AC

جدول ۶-۵ توابع کنترل و اعمال جزئی کامپیوتر پایه

برداشت	$R'T_0: AR \leftarrow PC$
	$R'T_1: IR \leftarrow M[AR], PC \leftarrow PC + 1$
دیکد	$R'T_2: D_0, \dots, D_7 \leftarrow \text{Decode } IR(12-14),$ $AR \leftarrow IR(0-11), I \leftarrow IR(15)$
غیرمستقیم	$D_5IT_3: AR \leftarrow M[AR]$
وقفه:	$T_0T_1T_2(IEN)(FGI + FGO): R \leftarrow 1$ $RT_0: AR \leftarrow 0, TR \leftarrow PC$ $RT_1: M[AR] \leftarrow TR, PC \leftarrow 0$ $RT_2: PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$
حافظه‌ای:	
AND	$D_0T_4: DR \leftarrow M[AR]$ $D_0T_5: AC \leftarrow AC \wedge DR, SC \leftarrow 0$
ADD	$D_1T_4: DR \leftarrow M[AR]$ $D_1T_5: AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$
LDA	$D_2T_4: DR \leftarrow M[AR]$ $D_2T_5: AC \leftarrow DR, SC \leftarrow 0$
STA	$D_3T_4: M[AR] \leftarrow AC, SC \leftarrow 0$
BUN	$D_4T_4: PC \leftarrow AR, SC \leftarrow 0$
BSA	$D_5T_4: M[AR] \leftarrow PC, AR \leftarrow AR + 1$ $D_5T_5: PC \leftarrow AR, SC \leftarrow 0$
ISZ	$D_6T_4: DR \leftarrow M[AR]$ $D_6T_5: DR \leftarrow DR + 1$ $D_6T_6: M[AR] \leftarrow DR, \text{ if } (DR = 0) \text{ then } (PC \leftarrow PC + 1) \quad SC \leftarrow 0$
ثباتی:	$D_7IT_7 = r$ (مشترک در همه دستورالعمل‌های ثباتی) $IR(i) = B_i, (i = 0, 1, 2, \dots, 11)$ $r: SC \leftarrow 0$
CLA	$rB_{11}: AC \leftarrow 0$
CLE	$rB_{10}: E \leftarrow 0$
CMA	$rB_9: AC \leftarrow \overline{AC}$
CME	$rB_8: E \leftarrow \overline{E}$
CIR	$rB_7: AC \leftarrow \text{shr } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$rB_6: AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$rB_5: AC \leftarrow AC + 1$
SPA	$rB_4: \text{If } (AC(15) = 0) \text{ then } (PC \leftarrow PC + 1)$
SNA	$rB_3: \text{If } (AC(15) = 1) \text{ then } (PC \leftarrow PC + 1)$
SZA	$rB_2: \text{If } (AC = 0) \text{ then } (PC \leftarrow PC + 1)$
SZE	$rB_1: \text{If } (E = 0) \text{ then } (PC \leftarrow PC + 1)$
HLT	$rB_0: S \leftarrow 0$
ورودی - خروجی:	$D_8IT_8 = p$ (مشترک در همه دستورالعمل‌های ورودی خروجی) $IR(i) = B_i, (i = 6, 7, 8, 9, 10, 11)$ $p: SC \leftarrow 0$
INP	$pB_{11}: AC(0-7) \leftarrow INPR, FGI \leftarrow 0$
OUT	$pB_{10}: OTR \leftarrow AC(0-7), FGO \leftarrow 0$
SKI	$pB_9: \text{If } (FGI = 1) \text{ then } (PC \leftarrow PC + 1)$
SKO	$pB_8: \text{If } (FGO = 1) \text{ then } (PC \leftarrow PC + 1)$
ION	$pB_7: IEN \leftarrow 1$
IOF	$pB_6: IEN \leftarrow 0$

برای کنترل $AC=0$ و کنترل بیت علامت $AC(15)$ ؛ بیت های 0 تا 15 ثبات DR برای کنترل $DR=0$ و مقادیر هفت فلیپ فلاپ.

خروجی های مدار کنترل منطقی عبارتند از:

- ۱- سیگنال های کنترل ورودی 9 عدد ثبات
- ۲- سیگنال های کنترل ورودی های خواندن و نوشتن
- ۳- سیگنال های نشان دادن^۱، پاک کردن^۲ و مکمل سازی فلیپ فلاپها
- ۴- سیگنال S_0, S_1, S_2 و S_0 برای انتخاب ثبات برای گذرگاه
- ۵- سیگنال های کنترل مدار منطقی جمع کننده AC

مشخصات انواع سیگنال های کنترل مستقیماً از لیست عبارات انتقال ثبات در جدول ۵-۶ قابل دسترسی است.

کنترل ثبات ها و حافظه

ثبات های کامپیوتر متصل به سیستم گذرگاه مشترک در شکل ۴-۵ دیده می شوند. ورودی های کنترل ثبات ها LD (بارشدن)، INR (افزایش) و CLR (پاک کردن) می باشند. فرض کنید که می خواهیم ساختار گیتی ورودی های کنترل AR را بدست آوریم. برای این منظور جدول ۵-۶ را برای یافتن تمام عباراتی که محتوای AR را تغییر می دهد جستجو می کنیم.

$$R'T_0 : AR \leftarrow PC$$

$$R'T_2 : AR \leftarrow IR (0-11)$$

$$D'7IT_3 : AR \leftarrow M[AR]$$

$$RT_0 : AR \leftarrow 0$$

$$D_5T_4 : AR \leftarrow AR+1$$

سه عبارت اول بیانگر انتقال اطلاعات از یک ثبات یا حافظه به AR است. محتوای ثبات مبدأ یا حافظه روی گذرگاه قرار گرفته و محتویات گذرگاه هم با تواناسازی ورودی کنترل LD منتقل می شود. عبارت چهارم AR را 0 می کند. عبارت آخر هم AR را یک واحد افزایش می دهد. توابع کنترل می توانند با هم ترکیب و با سه عبارت بولی نشان داده شوند.

$$LD(AR) = R'T_0 + R'T_2 + D'7IT_3$$

$$CLR(AR) = RT_0$$

$$INR(AR) = D_5 T_4$$

که در آنها LD(AR) ورودی بارکردن AR، CLR (AR) ورودی پاک کردن AR و INR(AR) ورودی افزایش AR است. مدار منطقی کنترل گیت AR در شکل ۱۶-۵ دیده می شود. بطریق مشابه می توان مدار کنترل سایر ثبات ها و نیز ورودی های کنترل حافظه را نیز بدست آورد. گیت های منطقی مربوط به ورودی خواندن با جستجو در جدول ۶-۵ برای یافتن عبارات خواندن نیز امکان پذیر است. نشانه عمل خواندن با نماد $M[AR] \leftarrow$ قابل تشخیص است.

$$\text{Read: } R'T_1 + D_7IT_3 + (D_0 + D_1 + D_2 + D_6) T_4$$

خروجی گیت های منطقی که عبارت فوق را پیاده سازی می کنند باید به ورودی خواندن حافظه متصل گردد.

کنترل فلیپ فلاپ های منفرد

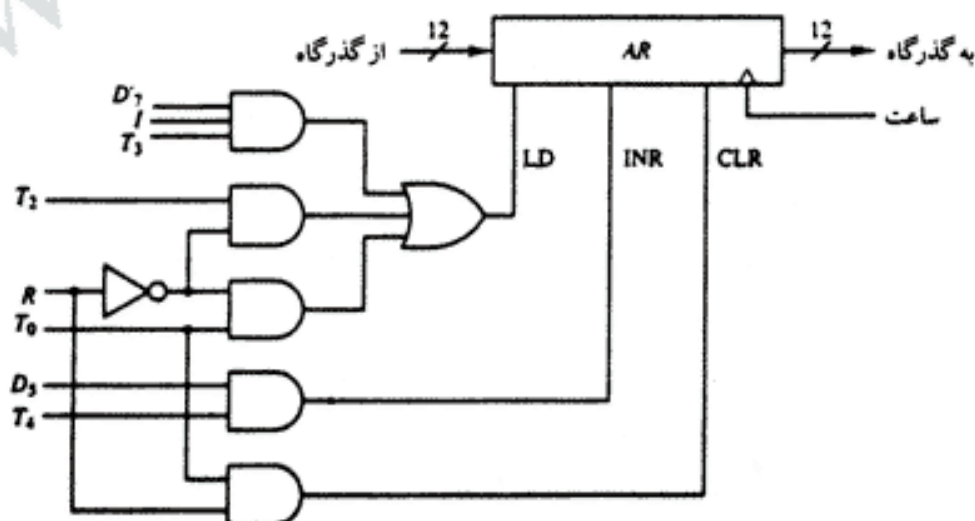
گیت های کنترل هفت فلیپ فلاپ بروش مشابهی تعیین می شود. بعنوان مثال جدول ۶-۵ نشان می دهد که IEN توسط دو دستورالعمل ممکن است تغییر یابد.

$$pB_7: IEN \leftarrow 1$$

$$pB_6: IEN \leftarrow 0$$

که در آن $p = D_7IT_3$ و B_6 و B_7 بیت های ۶ و ۷ از ثبات IR هستند. بعلاوه در انتهای سیکل وقفه هم IEN پاک می شود.

$$RT_2: IEN \leftarrow 0$$



شکل ۱۶-۵ گیت های کنترل AR

چنانچه یک فلیپ فلاپ JK برای IEN بکار رود، مدار کنترل منطقی مطابق شکل ۵-۱۷ خواهد بود.

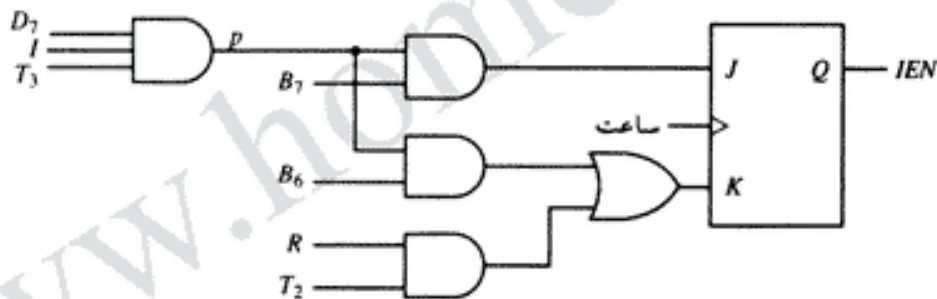
کنترل گذرگاه مشترک

گذرگاه 16 بیتی شکل ۴-۵ توسط ورودی‌ها انتخاب S_2 ، S_1 و S_0 کنترل می‌شود. عدد اعشاری مقابل هر ورودی گذرگاه عدد معادل دودویی اعمال شده به ورودی‌های کنترل است تا ثبات مربوط به آن عدد انتخاب شود. جدول ۵-۷ اعداد اعمال شده به $S_2S_1S_0$ برای انتخاب هر ثبات را نشان می‌دهد. هر عدد دودویی مرتبط با یک متغیر بولی x_1 تا x_7 است که با ساختار گیتی که باید برای انتخاب ثبات یا حافظه فعال باشد مرتبط است. بعنوان مثال وقتی $x_1=1$ است مقدار $S_2S_1S_0$ باید 001 باشد و خروجی AR برای گذرگاه انتخاب می‌شود. جدول ۵-۷ می‌تواند بعنوان جدول درستی انکدر دودویی در نظر گرفته شود. مکان قرار گرفتن انکدر در ورودی‌های انتخاب گر در شکل ۵-۱۸ دیده می‌شود. توابع بول برای انکدر برابر است با:

$$S_0 = x_1 + x_3 + x_5 + x_7$$

$$S_1 = x_2 + x_3 + x_6 + x_7$$

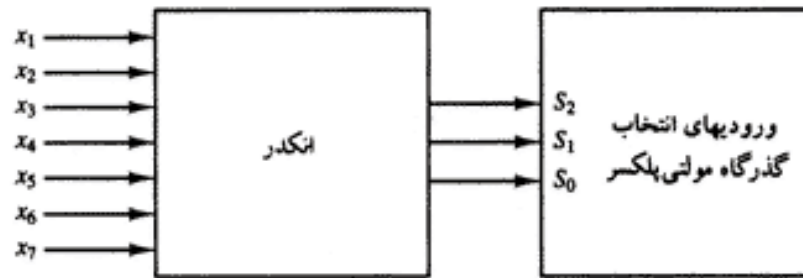
$$S_2 = x_4 + x_5 + x_6 + x_7$$



شکل ۵-۱۷ ورودی‌های کنترل برای IEN

جدول ۵-۷ انکدر برای مدار انتخاب گذرگاه

ورودیها							خروجیها			ثبات انتخاب شده برای گذرگاه
x_1	x_2	x_3	x_4	x_5	x_6	x_7	S_2	S_1	S_0	
0	0	0	0	0	0	0	0	0	0	هیچکدام
1	0	0	0	0	0	0	0	0	1	AR
0	1	0	0	0	0	0	0	1	0	PC
0	0	1	0	0	0	0	0	1	1	DR
0	0	0	1	0	0	0	1	0	0	AC
0	0	0	0	1	0	0	1	0	1	IR
0	0	0	0	0	1	0	1	1	0	TR
0	0	0	0	0	0	1	1	1	1	حافظه



شکل ۱۸-۵ انکدر برای ورودی‌های انتخاب گذرگاه

برای تعیین مدار منطقی ورودی انکدر، لازم است توابع کنترلی که ثبات مربوطه را روی گذرگاه قرار دهند تعیین گردد. برای مثال، برای یافتن مدار منطقی مربوط به $x_1=1$ ما تمام عبارات انتقال ثبات در جدول ۵-۶ را جستجو کرده و عباراتی که در آنها AR منبع است را استخراج می‌کنیم.

$$D_4 T_4: PC \leftarrow AR$$

$$D_5 T_5: PC \leftarrow AR$$

بنابراین، تابع بول x_1 برابرست با

$$x_1 = D_4 T_4 + D_5 T_5$$

خروجی داده حافظه وقتی انتخاب می‌شود که $x_7=1$ و $S_2 S_1 S_0=111$ باشد. مدار گیتی که x_7 را تولید می‌کند نیز باید به ورودی خواندن حافظه وصل شود. بنابراین تابع بول برای x_7 مشابه آنچه قبلاً برای "خواندن" بدست آمد خواهد بود.

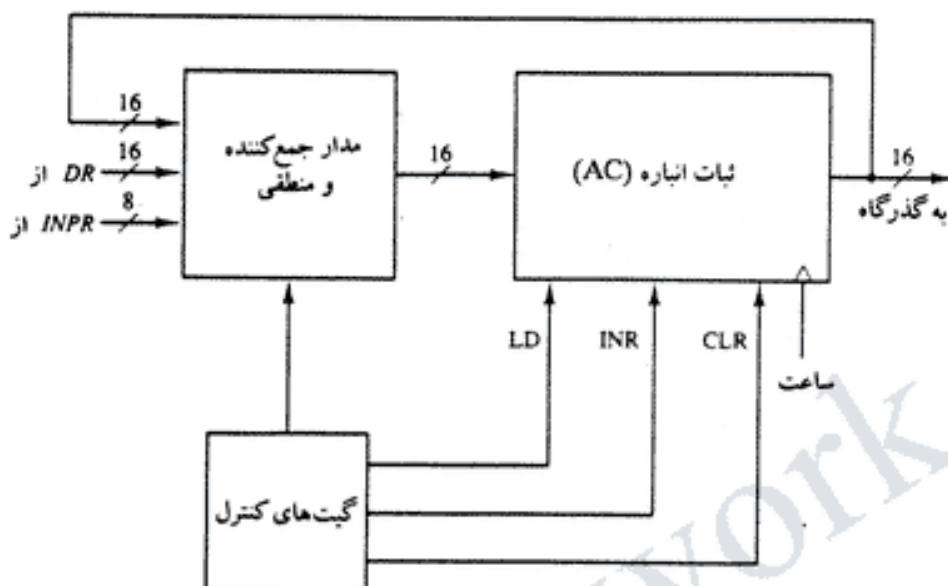
$$x_7 = R' T_1 + D_7 I T_3 + (D_0 + D_1 + D_2 + D_6) T_4$$

به روشی مشابه می‌توانیم مدار گیتی را برای سایر ثبات‌ها بدست آوریم

۵-۱۰ طراحی مدار منطقی انباره^۱

مدارهای مربوط به ثبات AC در شکل ۵-۱۹ دیده می‌شود. جمع‌کننده و مدار منطقی دارای سه مجموعه ورودی است. یک مجموعه 16 ورودی از خروجی AC می‌آید. مجموعه 16 ورودی دیگر از DR می‌رسد. سومین مجموعه هشت ورودی از ثبات ورودی INPR گرفته شده است. خروجی جمع‌کننده و مدار منطقی ورودی‌های داده را برای ثبات AC فراهم می‌آورند. بعلاوه مدارهای منطقی لازم

1- Accumulator



شکل ۱۹-۵ مدار مربوط به AC

برای کنترل LD، INR و CLR در ثبات و کنترل جمع کننده و مدار منطقی مربوطه نیز باید مد نظر باشند.

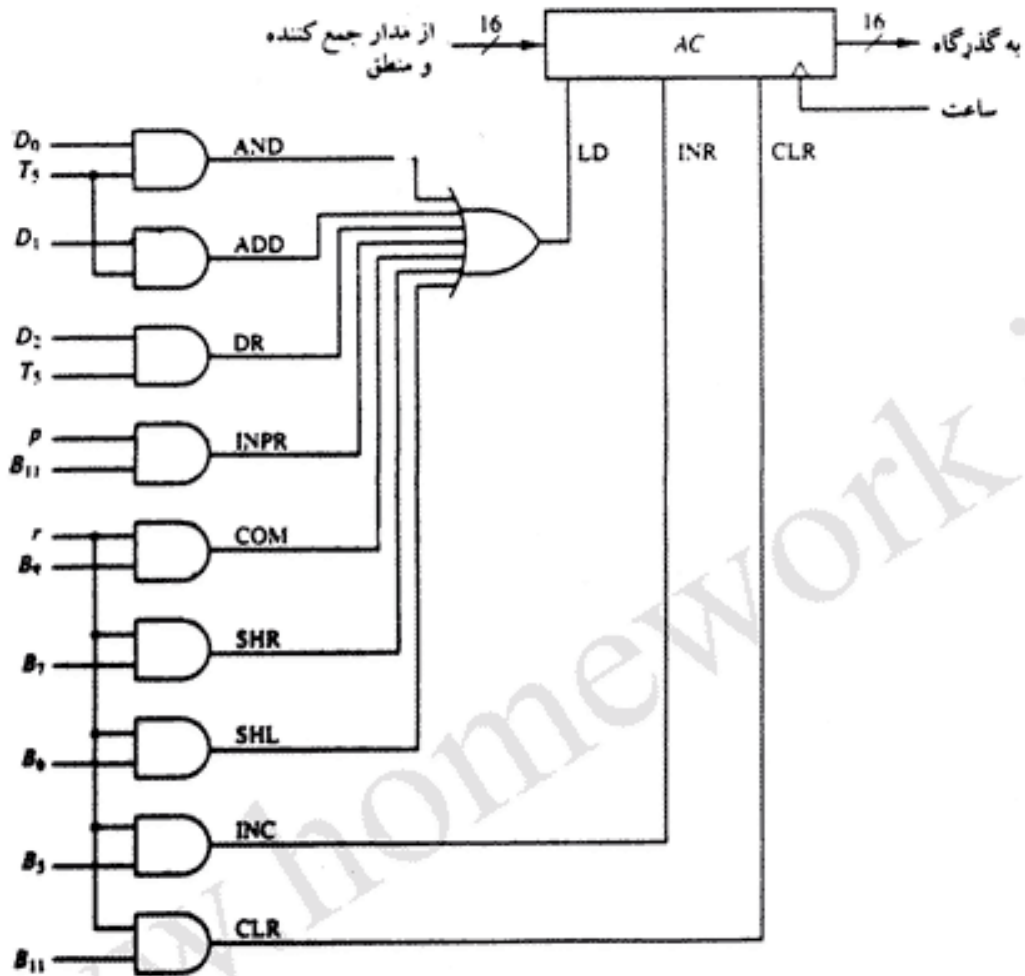
برای طراحی مدار منطقی مربوط به AC، باید به عبارات انتقال ثبات در جدول ۶-۵ مراجعه و تمام عباراتی که محتویات AC را تغییر می دهند استخراج شوند.

$D_0T_5:$	$AC \leftarrow AC \wedge DR$	DR با AND
$D_1T_5:$	$AC \leftarrow AC + DR$	DR با Add
$D_2T_5:$	$AC \leftarrow DR$	انتقال از DR
$pB_{11}:$	$AC(0-7) \leftarrow INPR$	انتقال از INPR
$rB_4:$	$AC \leftarrow \overline{AC}$	متمم
$rB_7:$	$AC \leftarrow shr AC, AC(15) \leftarrow E$	شیفت به راست
$rB_6:$	$AC \leftarrow shl AC, AC(0) \leftarrow E$	شیفت به چپ
$rB_{11}:$	$AC \leftarrow 0$	پاک
$rB_5:$	$AC \leftarrow AC + 1$	افزایش

از این لیست می توانیم مدار منطقی جمع کننده و گیت های منطقی آنرا بدست آوریم

کنترل ثبات AC

ساختار گیتی کنترل کننده ورودی های AC یعنی LD، INR و CLR در شکل ۲۰-۵ نشان داده شده اند. پیکربندی گیت ها از توابع کنترل در لیست فوق بدست آمده اند. تابع کنترل برای ریز عمل های پاک



شکل ۲۰-۵ ساختار گیت‌های مربوط به کنترل در ورودی‌های LD، INR و CLR از AC

کردن^۱، rB_{11} است که در آن $r = D_7I'T_3$ و $B_{11} = IR(11)$ می‌باشد. خروجی گیت AND که این تابع کنترل را تولید می‌کند به ورودی CLR ثابت متصل است. بطور مشابه خروجی گیتی که ریز عمل افزایش^۲ را پیاده سازی می‌نماید به ورودی INR ثابت متصل است. هفت ریز عمل دیگر در جمع کننده و مدار منطقی تولید و در زمان معینی به AC اعمال می‌شوند. خروجی‌های گیت‌ها برای هر تابع کنترل با یک نام سمبلیک علامت گذاری شده است. این خروجی‌ها در طراحی جمع کننده و مدار منطقی بکار می‌روند.

جمع کننده و مدار منطقی

جمع کننده و مدار منطقی مربوطه می‌تواند به 16 قسمت شود که هر قسمت مربوط به یک بیت از

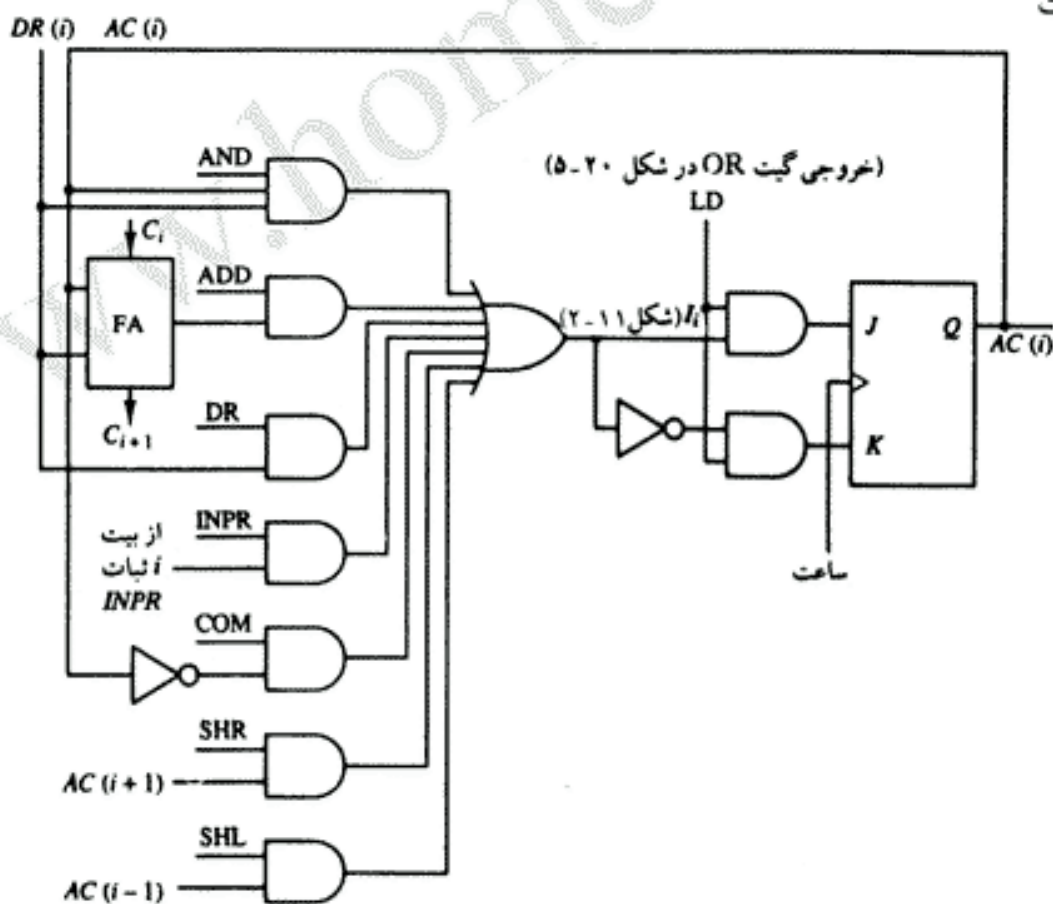
1- Clear

2- Increment

AC است. ساختمان درونی ثبات مطابق شکل ۱۱-۲ است. با بازگشت به این شکل می بینیم که هر گیت دارای یک فلیپ فلاپ JK، دو گیت OR و دو گیت AND است. ورودی بار (LD) به ورودی گیت های AND وصل است. شکل ۲۱-۵ این بخش از ثبات AC را نشان می دهد (یک گیت OR حذف شده است). ورودی با I_i و خروجی با $AC(i)$ نام گذاری شده است. وقتی که ورودی LD فعال است، ۱۶ ورودی I_i که در آن $i=0,1,2,\dots,15$ است به $AC(0-15)$ منتقل می شوند.

یک طبقه از مدار جمع کننده و منطقی از هفت گیت AND، یک گیت OR و یک تمام جمع کننده (FA) مطابق شکل ۲۱-۵ ساخته شده است. ورودی این گیت ها با نام هایی سمبلیکی از خروجی گیت ها با همان نام در شکل ۲۰-۵ آمده اند. بعنوان مثال، ورودی علامت زده شده با ADD در شکل ۲۱-۵ به خروجی مشابه با همان نام در شکل ۲۰-۵ وصل است.

عمل AND با $AC(i)$ کردن $AC(i)$ با بیت متناظرش در ثبات داده $DR(i)$ صورت می گیرد. عمل ADD هم با جمع کننده ای مشابه با شکل ۶-۴ انجام می شود. در طبقه جمع کننده از یک تمام جمع کننده با ورودی و خروجی نقلی استفاده شده است. انتقال از INPR به AC فقط از بیت ۰ تا ۷ میسر است. ریز عمل مکمل سازی با معکوس سازی مقادیر بیت های AC بدست می آید. عمل شیفت به راست



شکل ۲۱-۵ یک طبقه از مدار جمع کننده و منطقی

انتقال را از بیت $AC(i+1)$ به $AC(i)$ و شیفت به چپ انتقال را از بیت $AC(i-1)$ به $AC(i)$ انجام می دهد. مدار کامل جمع کننده و مدار منطقی از 16 بخش که بهم متصلند تشکیل شده اند.

مسائل

- ۵-۱ یک کامپیوتر از حافظه ای با 256k کلمه 32 بیتی استفاده می کند. یک دستورالعمل دودویی در یک کلمه از حافظه ذخیره شده است. دستورالعمل چهاربخش دارد: بیت غیرمستقیم، یک کد عملیاتی، یک کد ثبات برای تعیین یکی از 64 ثبات و بخش آدرس.
- (الف) چند بیت در کد عملیاتی، کد ثبات و آدرس وجود دارد؟
- (ب) قالب کلمه دستورالعمل را ترسیم و تعداد بیت در هر قسمت را معین کنید.
- (پ) در ورودی های داده و آدرس حافظه چند بیت وجود دارد.
- ۵-۲ اختلاف بین دستور یا آدرس مستقیم و غیرمستقیم چیست؟ چند ارجاع به حافظه برای هر نوع دستورالعمل لازم است تا عملوند را به ثبات پردازشگر منتقل کند؟
- ۵-۳ ورودی های کنترل زیر در سیستم گذرگاه شکل ۴-۵ فعالند. برای هر حالت انتقال ثباتی که در پالس ساعت بعدی اجرا شود را معین کنید.

	S_2	S_1	S_0	LD ثبات	حافظه	جمع کننده
(الف)	1	1	1	IR	خواندن	—
(ب)	1	1	0	PC	—	—
(ج)	1	0	0	DR	نوشتن	—
(د)	0	0	0	AC	—	جمع

- ۵-۴ انتقال ثبات های زیر قرار است در سیستم شکل ۴-۵ اجرا شوند. برای هر انتقال: (۱) مقدار دودویی که باید به ورودی های انتخاب گذرگاه S_2 , S_1 و S_0 اعمال شوند را معین کنید؛ (۲) ثباتی که کنترل LD آن باید فعال شود (اگر وجود دارد)؛ (۳) عمل نوشتن یا خواندن حافظه (اگر لازم است)؛ و (۴) عمل در جمع کننده و مدار منطقی (اگر وجود دارد).

$$\begin{array}{ll} \text{الف) } AR \leftarrow PC & \text{ب) } IR \leftarrow M[AR] \\ \text{ج) } M[AR] \leftarrow TR & \text{د) } AC \leftarrow DR, DR \leftarrow AC \end{array}$$

- ۵-۵ توضیح دهید چرا هیچک از ریزعمل های زیر در طول یک پالس ساعت در سیستم شکل ۴-۵ اجرا نمی شوند. رشته ریزاعمال لازم برای انجام عملیات را معین کنید

$$\begin{array}{ll} \text{الف) } IR \leftarrow M[PC] & \\ \text{ب) } AC \leftarrow AC + TR & \end{array}$$

(ج) $DR \leftarrow DR + AC$ (AC تغییر نمی‌کند)

۵-۶ قالب دستورات کامپیوتر پایه شکل ۵-۵ و لیست دستورات جدول ۲-۵ را ملاحظه کنید. برای هر یک از دستورات 16 بیتی، کد معادل چهاربیتی مبنای شانزده را نوشته و بزبان ساده بگوئید این دستور چه کاری انجام می‌دهد.

(الف) 0001 0000 0010 0100

(ب) 1011 0001 0010 0100

(ج) 0111 0000 0010 0000

۵-۷ کدام دودستور برای 1 کردن فلیپ فلاپ E در کامپیوتر پایه بکار می‌روند؟
 ۵-۸ یک دیاگرام زمانبندی مشابه شکل ۵-۷ ترسیم و فرض کنید SC در T_3 برابر 0 شده باشد بشرط اینکه سیگنال کنترل C_7 فعال باشد.

$C_7 T_3: SC \leftarrow 0$

C_7 با لبه پالس مربوط به T_1 فعال می‌شود.

۵-۹ محتویات AC در کامپیوتر پایه عدد مبنای شانزده A937 است و مقدار اولیه E برابر 1 است. محتویات AC، E، PC، AR و IR در مبنای 16 پس از اجرای دستور CLA چیست. عمل قبل را 11 بار با هر یک از دستورات عملی تکرار کنید. مقدار اولیه PC را عدد مبنای شانزده 021 فرض کنید.
 ۵-۱۰ دستور عملی در آدرس 021 کامپیوتر پایه دارای $I=0$ ، کد عملیاتی AND و آدرس 083 است (تمام ارقام در مبنای شانزده است). کلمه حافظه واقع در آدرس 083 دارای عملوند B8F2 و محتویات AC هم A937 است. در طول سیکل دستور محتویات ثبات‌های زیر را در پایان فاز اجرا معین کنید: PC، AR، DR، AC و IR. مسئله را شش بار دیگر برای دستورالعمل ارجاع به حافظه دیگری تکرار کنید.

۵-۱۱ محتویات ثبات‌های PC، AR، DR، IR و SC در مبنای شانزده وقتی که در کامپیوتر پایه دستور غیرمستقیم ISZ دریافت و اجرا شود چیست. مقدار اولیه PC را 7FF در نظر بگیرید. محتویات حافظه در آدرس 7FF برابر EA9F است. محتویات حافظه در آدرس A9F هم 0C35 می‌باشد. محتویات حافظه C35 برابر با FFFF می‌باشد. پاسخ خود را بصورت جدولی با پنج ستون با هر ستون برای یک ثبات، و هر سطر برای یک سیگنال زمان بندی تهیه کنید. محتویات هر ثبات را پس از لبه مثبت هر پالس ساعت نشان دهید.

۵-۱۲ محتویات PC در کامپیوتر پایه 3AF است (تمام اعداد در مبنای شانزده). محتویات AC هم 7EC3 است. محتویات حافظه آدرس 3AF برابر با 932E می‌باشد. محتویات حافظه در آدرس 32E برابر 09AC و در آدرس 9AC هم 8B9F است

الف) دستورالعملی که بعداً دریافت و اجرا شود چیست؟
 ب) عمل دودویی که در AC پس از اجرای دستورالعمل رخ می دهد چیست
 ج) محتویات ثبات های PC، AR، DR، AC و IR در مبنای شانزده چیست. همچنین مقادیر E و I و SC در انتهای سیکل دستورالعمل را معین کنید.

۵-۱۳ فرض کنید که شش دستور ارجاع به حافظه در کامپیوتر پایه در جدول ۴-۵ با جدول زیر تعویض شوند. EA آدرس موثر واقع در AR در T₄ است. فرض کنید که جمع کننده و مدار منطقی شکل ۴-۵ می تواند عمل XOR را انجام دهد $AC \leftarrow AC \oplus DR$. بعلاوه فرض کنید که جمع کننده و مدار منطقی نمی توانند مستقیماً تفریق را انجام دهند. تفریق باید بکمک مکمل 2 انجام شود. رشته عبارات انتقال ثبات لازم برای اجرای هر دستور لیست شده را از T₄ به بعد مشخص کنید. دقت کنید که هیچ تغییری در AC رخ نمی دهد مگر اینکه دستورالعمل تغییری را در آن معین کند. شما می توانید با استفاده از TR محتویات AC را موقتاً ذخیره و یا محتویات AC و DR را باهم عوض کنید.

توضیح	نمایش سبلیک	کد عمل	سمبل
OR انحصاری با AC	$AC \leftarrow AC \oplus M[EA]$	000	XOR
جمع AC با حافظه	$M[EA] \leftarrow M[EA] + AC$	001	ADM
تفریق حافظه از AC	$AC \leftarrow AC - M[EA]$	010	SUB
تبادل AC با حافظه	$AC \leftarrow M[EA], M[EA] \leftarrow AC$	011	XCH
گذر در صورت برابری	If $(M[EA] = AC)$ then $(PC \leftarrow PC + 1)$	100	SEQ
انشعاب اگر AC مثبت و غیر صفر باشد	If $(AC > 0)$ then $(PC \leftarrow EA)$	101	BPA

۵-۱۴ تغییرات زیر را در کامپیوتر پایه بعمل آورید.
 ۱- یک ثبات CTR (ثبات شمارنده) را به سیستم گذرگاه اضافه کنید و آنرا با $S_2S_1S_0 = 000$ انتخاب نمایید.
 ۲- ISZ را با دستوری که یک عدد را در CTR بار کند عوض کنید.

$$\text{LDC Address} \quad \text{CTR} \leftarrow M[\text{address}]$$

۳- یک دستور ارجاع به ثبات ICSZ به مجموعه اضافه کنید: CTR را یک واحد اضافه کرده و از اجرای دستور بعدی اگر حاصل افزایش صفر است صرفنظر نمایید. مزیت این تغییر را بیان کنید.

۵-۱۵ واحد حافظه کامپیوتر پایه در شکل ۱۳-۵ را با یک حافظه 16×65536 عوض کنید. این

حافظه آدرس 16 بیتی نیاز دارد. قالب دستورالعمل ارجاع به حافظه در شکل (5-5) الف) برای $I=1$ بلا تغییر و بخش آدرس در مکان های 0 تا 11 قرار دارد. اما وقتی $I=0$ است (آدرس مستقیم) آدرس دستورالعمل توسط 16 بیت در کلمه دیگری که بدنبال دستور آمده است داده شده است. ریز اعمال را در T_2 ، T_3 (و T_4 اگر لازم باشد) تصحیح کنید تا با این پیکربندی هماهنگ باشد.

5-16 کامپیوتری از یک حافظه هشت بیتی 65536 کلمه ای استفاده می کند. این کامپیوتر دارای ثبات های PC ، AR ، TR (هریک شانزده بیت) و AC و DR و IR (هریک هشت بیت) است. یک دستور ارجاع به حافظه متشکل از سه کلمه است: یک کد عملیات 8 بیتی (یک کلمه) و یک آدرس 16 بیتی (در دو کلمه بعدی). تمام عملوندها هشت بیت هستند. بیت غیرمستقیم هم وجود ندارد.

الف) بلاک دیاگرامی از کامپیوتر ترسیم کنید و ثبات ها و حافظه ها را مطابق شکل 3-5 نشان دهید. (از یک گذرگاه مشترک استفاده نکنید).

ب) طرز قرار گرفتن یک نمونه دستور سه کلمه ای را به همراه عملوند 8 بیتی در حافظه نشان دهید.

رشته ریزاعمال برای دریافت یک دستور ارجاع به حافظه را لیست کنید و سپس عملوند را در DR قرار دهید. از سیگنال زمانی T_0 شروع کنید.

5-17 یک کامپیوتر دیجیتال دارای 16384 حافظه 40 بیتی در هر کلمه است. قالب کد دستور از شش بیت برای عملوند و 14 بیت برای آدرس تشکیل شده است. (بیت غیرمستقیم ندارد). دو دستورالعمل در یک کلمه جای داده شده اند و یک ثبات دستورالعمل 40 بیتی IR هم در واحد کنترل وجود دارد. برنامه ای را برای فازهای برداشت و اجرا در این کامپیوتر بنویسید.

5-18 یک برنامه خروجی از آدرس 2300 نوشته شده است. این برنامه وقتی کامپیوتر یک وقفه را در $FGO=1$ تشخیص دهد اجرا می گردد (در حالیکه $IEN=1$ است)

الف - چه دستوری باید در آدرس 1 قرار گیرد؟

ب - دو دستور آخر برنامه خروجی چیست؟

5-19 عبارات انتقال ثبات برای ثبات R و حافظه در یک کامپیوتر مطابق زیر است (Xها توابع کنترل هستند و بطور تصادفی رخ می دهند).

$$X'_3 X_1: R \leftarrow M [AR]$$

کلمه حافظه را در R بنویس

$$X'_1 X_2: R \leftarrow AC$$

انتقال AC به R

$$X'_1 X_3: M [AR] \leftarrow R$$

R را در حافظه بنویس

حافظه دارای ورودی های داده، خروجی های داده، ورودی های آدرس و ورودی های کنترل برای

خواندن و نوشتن مطابق شکل ۱۲-۲ است. سخت افزار R و حافظه را بشکل بلاک دیاگرام بکشید نشان دهید که چگونه توابع کنترل X_1 تا X_3 ورودی R، ورودی مولتی پلکسرهایی که شما در دیاگرام وارد کرده‌اید، و ورودی‌های خواندن و نوشتن حافظه را انتخاب می‌کنند. ۵-۲۰ رشته اعمالی که باید توسط فلیپ فلاپ F انجام شوند با عبارت انتقال ثبات داده شده‌اند

$XT_3: F \leftarrow 1$	1 در F نشانده شود
$yT_1: F \leftarrow 0$	F با 0 پاک شود
$ZT_2: F \leftarrow \bar{F}$	F مکمل
$WT_5: F \leftarrow G$	مقدار G را به F انتقال بده

در غیراینصورت F نباید تغییر یابد. دیاگرام منطقی مربوط به اتصالات گیت‌ها که توابع کنترل و ورودی‌های فلیپ فلاپ F تشکیل می‌دهند را ترسیم نمائید. از یک فلیپ فلاپ JK استفاده کرده و تعداد گیت‌ها را حداقل کنید.

- ۵-۲۱ مدار کنترل گیتی مربوط به شمارنده برنامه PC را در کامپیوتر پایه بدست آورید.
- ۵-۲۲ مدار کنترل گیتی برای ورودی خواندن یک حافظه را در کامپیوتر پایه بدست آورید.
- ۵-۲۳ مدار کامل منطقی وقفه را در کامپیوتر پایه نشان دهید. از فلیپ فلاپ JK استفاده کرده و گیت‌ها را به حداقل برسانید.
- ۵-۲۴ عبارت بولی را برای x_2 (جدول ۷-۵ را ملاحظه کنید) بدست آورید. نشان دهید که x_2 می‌تواند با یک گیت AND و یک OR تولید شود.
- ۵-۲۵ عبارت بولی برای یک مدار گیتی که شمارنده SC را پاک کند بدست آورید. دیاگرام منطقی آنرا رسم نموده و نشان دهید که چگونه خروجی به ورودی‌های INR و CLR از شمارنده SC وصل می‌شود شکل ۶-۵. تعداد گیت‌ها را حداقل نمائید.

۶

برنامه نویسی کامپیوتر پایه

- | | |
|-----------------|--------------------------------------|
| ۶-۱ مقدمه | ۶-۵ حلقه در برنامه نویسی |
| ۶-۲ زبان ماشین | ۶-۶ برنامه نویسی اعمال حسابی و منطقی |
| ۶-۳ زبان اسمبلی | ۶-۷ زیرروالها |
| ۶-۴ اسمبلر | ۶-۸ برنامه نویسی ورودی - خروجی |

۶-۱ مقدمه

یک سیستم کامپیوتری کامل متشکل از سخت افزار و نرم افزار است. سخت افزار از قطعات فیزیکی و همه تجهیزات مرتبط با آنها تشکیل می شود. منظور از نرم افزار برنامه هایی است که برای کامپیوتر نوشته می شود. می توان از جهات مختلف با نرم افزار کامپیوتر، بدون توجه به جزئیات چگونگی عملکرد آن، آشنا شد. همچنین می توان بخش هایی از سخت افزار را بدون اطلاع از توانایی های نرم افزار طراحی نمود. در هر صورت، افرادی که با معماری کامپیوتر سر و کار دارند باید هم از سخت افزار و هم از نرم افزار اطلاعاتی داشته باشند زیرا این دو شاخه بر یکدیگر تأثیر می گذارند.

نوشتن یک برنامه برای کامپیوتر به معنی مشخص کردن رشته ای از دستورالعمل های ماشین، به طور مستقیم یا غیر مستقیم است. دستورالعمل های ماشین در داخل کامپیوتر الگویی دودویی را تشکیل می دهند که کار با آن و یا درک آن، اگر غیر ممکن نباشد، مشکل هست. لذا نوشتن برنامه ها با مجموعه کاراکترهای آشنا تر ترجیح داده می شود. در نتیجه، ترجمه برنامه هایی که بر اساس سلیقه کاربر نوشته شده به برنامه های قابل تشخیص بوسیله سخت افزار، لازم به نظر می رسد.

برنامه ای که بوسیله یک کاربر نوشته می شود ممکن است وابسته و یا مستقل از کامپیوتر فیزیکی باشد که برنامه را اجرا می کند. مثلاً برنامه ای که بزیان فرتن استاندارد نوشته می شود مستقل از ماشین

است، زیرا اکثر کامپیوترها برنامه مترجمی دارند که فرترن استاندارد را به کد دودویی آن کامپیوتر تبدیل می‌کنند. اما خود برنامه مترجم وابسته به ماشین است زیرا باید برنامه فرترن را به کد دودویی قابل تشخیص بوسیله سخت افزار کامپیوتر خاص مورد استفاده ترجمه کند.

این فصل برخی از مفاهیم ابتدایی برنامه نویسی را معرفی می‌کند و رابطه آنها را با نمایش سخت‌افزاری دستورالعمل‌ها نشان می‌دهد. اولین بخش، عملکرد و ساختار اساسی برنامه‌ای که بزبان سمبلیک نوشته شده و بوسیله کاربر به یک برنامه معادل دودویی ترجمه می‌شود را ارائه می‌نماید. تأکید در بحث بیشتر بر روی مفاهیم مهم برنامه مترجم است و نه در چگونگی نوشتن خود برنامه. پس از آن فواید انواع دستورات ماشین به وسیله چند مثال برنامه‌نویسی ساده نشان داده شده است.

مجموعه دستورات پایه، که سازمان سخت‌افزار آن در فصل ۵ بررسی شد، در این فصل به منظور تشریح بسیاری از تکنیک‌های مورد استفاده در برنامه‌نویسی یک کامپیوتر، بکار رفته است. به این ترتیب امکان خواهد داشت تا رابطه بین یک برنامه و عملیات سخت‌افزاری که دستورات را اجرا می‌کنند مورد تحقیق قرار گیرد.

۲۵ دستور پایه کامپیوتر در جدول ۱-۶ تکرار شده‌اند تا مراجعه به آنها بهنگام مثال‌های برنامه‌نویسی تسهیل گردد. به هر دستورالعمل یک سمبل سه حرفی اختصاص یافته است تا نوشتن برنامه سمبلیک ساده شود. هفت دستورالعمل اول، دستورات ارجاع به حافظه (حافظه‌ای) است و هجده دستورالعمل بعدی دستورات ارجاع به ثبات (ثباتی) و ورود و خروج هستند. یک دستورالعمل ارجاع به حافظه دارای سه بخش است: بیت روش^۱ (شیوه) سه بیت کد عمل، و دوازده بیت آدرس. اولین رقم شانزده شانه‌دهی یک دستورالعمل حافظه‌ای شامل بیت روش و کد عمل آنست. سه رقم دیگر آدرس را مشخص می‌کنند. در یک دستورالعمل با آدرس غیرمستقیم بیت روش برابر 1 و اولین رقم شانزده شانه‌دهی بین 8 تا E می‌باشد. در حالت مستقیم محدوده از 0 تا 6 است. هجده دستور دیگر دارای کد عمل شانزده بیتی هستند. کد هر دستورالعمل بصورت یک عدد چهار رقمی شانزده شانه‌دهی لیست شده است. اولین رقم یک دستورالعمل ثباتی همواره 7 است. اولین رقم یک دستور ورودی-خروجی همیشه F است. سمبل m در ستون توضیح بمعنی آدرس موثر است. حرف M کلمه‌ای از حافظه را که در آدرس موثر قرار دارد نشان می‌دهد.

۲-۶ زبان ماشین

برنامه مجموعه‌ای از دستورالعمل‌ها برای هدایت کامپیوتر در اجرای یک کار داده پردازی مورد نظر است. زبان‌های مختلفی برای نوشتن یک برنامه کامپیوتری وجود دارد. ولی کامپیوتر می‌تواند تنها برنامه‌هایی را اجرا^۲ کند که در داخل آن به شکل دودویی ارائه شده باشند. برنامه‌های نوشته شده به هر زبان دیگر، قبل از اجرا بوسیله کامپیوتر باید به دستورالعمل دودویی ترجمه شوند. برنامه‌های نوشته

1- Mode bit

2- Execute

جدول ۶-۱ دستورالعمل های کامپیوتر

سمبل	کد شانزده شانزدهمی	شرح
AND	0 یا 8	AND کردن M با AC
ADD	1 یا 9	جمع M با AC، نقلی به E
LDA	2 یا A	بار کردن AC از M
STA	3 یا B	ذخیره AC در M
BUN	4 یا C	انشعاب غیرشرطی به m
BSA	5 یا D	ذخیره آدرس برگشت در m و انشعاب به m+1
ISZ	6 یا E	افزایش M و گذر در صورت صفر بودن نتیجه
CLA	7800	صفر کردن AC
CLE	7400	صفر کردن E
CMA	7200	متسم کردن AC
CME	7100	متسم کردن E
CIR	7080	چرخش به راست E و AC
CIL	7040	چرخش به چپ E و AC
INC	7020	افزایش AC
SPA	7010	گذر در صورت مثبت بودن AC
SNA	7008	گذر در صورت منفی بودن AC
SZA	7004	گذر در صورت صفر بودن AC
SZE	7002	گذر در صورت صفر بودن E
HLT	7001	توقف کامپیوتر
INP	F800	دریافت اطلاعات ورودی و 0 کردن پرچم
OUT	F400	ارسال اطلاعات خروجی و 0 کردن پرچم
SKI	F200	گذر در صورت 1 بودن پرچم ورودی
SKO	F100	گذر در صورت 1 بودن پرچم خروجی
ION	F080	فعال کردن وقفه
IOF	F040	غیرفعال کردن وقفه

شده برای یک کامپیوتر می توانند یکی از رشته های زیر باشند.

۱- کد دودویی. این کد رشته ای از دستورالعمل ها و عملوندها به شکل دودویی است که شکل واقعی

آنها را آنطور که در حافظه کامپیوتر ظاهر می شوند، نشان می دهد.

۲- کد هشت هشتی یا شانزده شانزدهمی. این کد معادل ترجمه شده کد دودویی به هشت هشتی یا

شانزده شانزدهمی است.

۳- کد سمبلیک. در این کد، کاربرد از سمبل ها (حروف، اعداد یا کاراکترهای خاص) برای بخش عملیاتی،

بخش آدرس، و سایر قسمت های کد دستورالعمل استفاده می کند. هر دستورالعمل سمبلیک را

می توان به یک دستورالعمل کد شده با دودویی ترجمه کرد. این ترجمه توسط برنامه خاصی بنام

اسمبلر^۱ انجام می شود. چون اسمبلر سمبل ها را ترجمه می کند، این نوع برنامه سمبلیک، برنامه

بزیان اسمبلی خوانده می شود.

1- Assembler

۴- زبان‌های برنامه‌نویسی سطح بالا، این برنامه‌ها که زبان‌های خاصی نوشته می‌شوند بخاطر دریافت تأثیر رویه‌هایی است که به منظور حل مسئله خاصی بکار می‌روند و نه صرفاً بخاطر تأثیر بر رفتار سخت‌افزار کامپیوتر. مثالی از یک برنامه زبان سطح بالا فرترن^۱ است. برنامه بصورت رشته‌ای از عبارات بر اساس نحوه تفکر فرد بهنگام حل یک مسئله نوشته می‌شود. با این وجود، هر عبارت باید قبل از اجرا در کامپیوتر به رشته‌ای از دستورات دودویی تبدیل شود. برنامه‌ای که یک برنامه دیگر زبان سطح بالا را به دودویی ترجمه می‌کند کامپایلر نامیده می‌شود.

دقیقتاً بگوئیم، برنامه زبان ماشین یک برنامه دودویی از رسته ۱ است. بعلت شباهت بین نمایش‌های دودویی، هشت هشتی و شانزده شانزده‌می، رسم بر این است که رسته ۲ بعنوان زبان ماشین شناخته شود. بدلیل رابطه یک به یک بین دستور سمبلیک و معادل دودویی‌اش، زبان اسمبلی بعنوان زبان در سطح ماشین شناخته می‌شود.

اکنون با بکارگیری کامپیوتر پایه رابطه بین زبان‌های دودویی و اسمبلی را شرح می‌دهیم. برنامه لیست شده در جدول ۲-۶ را ملاحظه کنید. اولین ستون، مکان حافظه (به دودویی) را برای هر دستور یا عملوند می‌دهد. ستون دوم محتوای دودویی این مکان‌هاست. (مکان، آدرس کلمه حافظه‌ای است که دستورات عمل در آن ذخیره شده است. شناخت تفاوت این آدرس، از بخش آدرس دستورات عمل اهمیت دارد) برنامه در بخش مشخصی از حافظه ذخیره شده و سپس بوسیله کامپیوتر و با شروع از آدرس 0 اجرا می‌شود. سخت‌افزار کامپیوتر این دستورات عمل‌ها را اجرا کرده و کار مورد نظر را انجام می‌دهد. معهداً، فردی که به این برنامه نگاه می‌کند به سختی درخواهد یافت که پس از اجرای برنامه چه رخ خواهد داد. با این حال، سخت‌افزار کامپیوتر تنها این نوع کدهای دستورات عمل را تشخیص می‌دهد.

نوشتن 16 بیت برای هر دستورات عمل خسته کننده است زیرا رقم‌ها زیادند. ما می‌خواهیم تعداد ارقام برای هر دستورات عمل را با بکاربردن کد هشت هشتی معادل کد دودویی کاهش دهیم. این کار نیاز به شش

جدول ۲-۶ برنامه دودویی برای جمع کردن دو عدد

خانه حافظه	کد دستورات عمل
0	0010 0000 0000 0100
1	0001 0000 0000 0101
10	0011 0000 0000 0110
11	0111 0000 0000 0001
100	0000 0000 0101 0011
101	1111 1111 1110 1001
110	0000 0000 0000 0000

1- Fortran

رقم برای هر دستورالعمل دارد. از طرف دیگر، ما می‌توانیم هر دستورالعمل را به چهار رقم تقلیل دهیم بشرطی که از کد معادل شانزده شانزده جدول ۳-۶ استفاده کنیم. نمایش شانزده شانزده در کاربرد ساده است؛ اما باید دانست که هر رقم شانزده شانزده وقتی که برنامه وارد کامپیوتر می‌شود باید تبدیل به معادل چهاربیتی‌اش شود. مزیت نوشتن برنامه‌های دودویی به معادل هشت هشتی یا شانزده شانزده از این مثال باید آشکار شده باشد.

برنامه جدول ۴-۶ از نام‌های سمبلیک دستورالعمل‌ها (این دستورالعمل‌ها در جدول ۱-۶ معرفی شدند) بجای معادل دودویی یا شانزده شانزده آنها استفاده کرده است. بخش آدرس دستورات حافظه‌ای، همانند عملوندها، به همان شکل شانزده شانزده شان باقی می‌مانند. توجه کنید که در مکان 005 مقدار عملوند منفی است زیرا بیت علامت در سمت چپ‌ترین مکان 1 است. در نظر گرفتن ستونی برای توضیحات راهی را برای شرح عملکرد دستورات در اختیار می‌گذارد. کار با برنامه‌های سمبلیک آسانتر است، و در نتیجه، بهتر است برنامه‌ها با سمبل‌ها نوشته شوند. این برنامه‌ها قابل تبدیل به کد دودویی معادل برای تولید برنامه دودویی می‌باشند.

حال یک قدم جلوتر می‌رویم و هر آدرس شانزده شانزده را با یک آدرسی سمبلیک و نیز هر عملوند شانزده شانزده را با عملوند دهنده جایگزین می‌کنیم. این اقدام مناسبی است زیرا معمولاً نویسنده برنامه بهنگام برنامه‌نویسی عدد واقعی مکان حافظه را نمی‌داند. اگر عملوند در حافظه بعد از

جدول ۳-۶ برنامه شانزده شانزده برای جمع کردن دو عدد

خانه حافظه	دستورالعمل
000	2004
001	1005
002	3006
003	7001
004	0053
005	FFE9
006	0000

جدول ۴-۶ برنامه با کدهای عمل سمبلیک

خانه حافظه	دستورالعمل	توضیحات
000	LDA 004	بار کردن عملوند اول در AC
001	ADD 005	جمع کردن عملوند دوم در AC
002	STA 006	ذخیره مجموع در خانه 006
003	HLT	توقف کامپیوتر
004	0053	عملوند اول
005	FFE9	عملوند دوم
006	0000	محل ذخیره مجموع

دستورالعمل قرار داده شود و اگر طول برنامه از قبل مشخص نباشد، محل عددی عملوندها تا پایان برنامه معلوم نخواهد بود. بعلاوه اعداد دهمی آشناتر از معادل شانزده شانزدهمی شان است. برنامه جدول ۵-۶ یک برنامه به زبان اسمبلی برای جمع دو عدد است. سمبل ORG که بدنبال آن اعدادی آمده است یک دستورالعمل ماشین نیست. هدف از آن تعیین مبدأ، یعنی، تعیین مکانی از حافظه است که دستورالعمل بعدی در زیر آن قرار می‌گیرد. سه خط بعدی دارای آدرس‌های سمبلیک (نمادین) هستند. مقدار هر یک از سمبل‌ها از سطرهای که آن سمبل را بصورت عنوان در ستون اول خود دارد مشخص می‌شود. عملوندهای دهمی با سمبل DEC مشخص می‌گردند. عددها می‌توانند مثبت یا منفی باشند، اما اگر منفی باشند باید به فرم متمم 2 علامتدار دودویی درآیند. آخرین خط سمبل END را دارد که انتهای برنامه را نشان می‌دهد. سمبل‌های ORG، DEC و END، شبه دستورالعمل^۱ها خوانده می‌شوند و در بخش بعد تعریف شده‌اند. توجه کنید که قبل از تمام توضیحات علامت ممیز (/) آمده است. برنامه معادل بزبان فرترن برای جمع دو عدد صحیح در جدول ۶-۶ لیست شده است. دو مقدار برای A و B را می‌توان با دستور INPUT یا با دستور DATA مشخص کرد. عمل حسابی روی دو عدد با یک دستور ساده انجام می‌شود. ترجمه این برنامه فرترن به یک برنامه دودویی متشکل از تخصیص سه مکان حافظه به مضاف، مضاف‌الیه و حاصل جمع، و سپس بدست آوردن رشته دستورالعمل‌های دودویی که مجموع را تشکیل می‌دهند می‌باشد. بنابراین یک برنامه مترجم سمبل‌های برنامه فرترن را به لیستی از برنامه جدول ۲-۶ که بشکل دودویی است ترجمه می‌کند.

جدول ۵-۶ برنامه به زبان اسمبلی برای جمع دو عدد

	ORG 0	مبدأ برنامه خانه 0 است /
	LDA A	بارکردن عملوند از A /
	ADD B	جمع کردن عملوند در B /
	STA C	ذخیره مجموع در C /
	HLT	توقف کامپیوتر /
A,	DEC 83	عملوند دهمی /
B,	DEC -23	عملوند دهمی /
C,	DEC 0	مجموع در خانه C ذخیره می‌شود /
	END	پایان برنامه نمادین /

جدول ۶-۶ برنامه فرترن برای جمع دو عدد

```

INTEGER A, B, C
DATA A, 83 B, -23
C = A + B
END

```

۳-۶ زبان اسمبلی

هر زبان برنامه نویسی با مجموعه‌ای از قواعد تعریف می‌شود. اگر کاربران بخواهند برنامه‌هایشان بطور صحیح ترجمه شود باید این قواعد را رعایت کنند. تقریباً هر کامپیوتر تجاری دارای زبان اسمبلی خاص خود است. قواعد نوشتن برنامه‌ها به زبان اسمبلی بصورت راهنماها مستندسازی و چاپ می‌شود و معمولاً توسط سازندگان کامپیوتر در دسترس قرار می‌گیرد. واحد پایه یک برنامه اسمبلی یک سطر کد است. یک زبان خاص بر اساس مجموعه قواعد خاصی تعریف می‌شود و این قواعد بیان می‌دارند که سمبل‌ها چگونه می‌توانند استفاده شوند یا با یکدیگر ترکیب شوند تا یک خط از کد را بسازند. حال ما قواعد یک زبان اسمبلی را برای نوشتن یک برنامه سمبلیک برای کامپیوتر پایه فرموله می‌کنیم.

قواعد زبان

هر خط از برنامه زبان اسمبلی به صورت سه ستون مرتب شده‌اند که هر یک میدان^۱ نامیده می‌شوند. میدان‌ها اطلاعات زیر را مشخص می‌کنند.

- ۱- میدان عنوان (برچسب) که می‌تواند خالی باشد یا یک آدرس سمبلیک را مشخص کند.
- ۲- میدان دستورالعمل که یک دستورالعمل ماشین یا یک شبه دستورالعمل را مشخص می‌نماید.
- ۳- میدان توضیحات که می‌تواند خالی یا حاوی یک توضیح باشد.

هر آدرس سمبلیک حاوی یک، دو، سه ولی نه بیش از سه کاراکتر الفبا عددی است. اولین کاراکتر حتماً باید یک حرف، دو کاراکتر بعدی حرف یا عدد باشند. سمبل می‌تواند بدخواه توسط برنامه‌نویس انتخاب شود. یک آدرس سمبلیک در میدان عنوان به یک کاما ختم می‌شود تا اسمبلر تشخیص دهد که عنوان است. میدان دستورالعمل در یک برنامه زبان اسمبلی یکی از اقلام زیر را مشخص می‌نماید.

- ۱- دستورالعمل‌های حافظه‌ای^۲ MRI
- ۲- دستورالعمل‌های ثباتی^۳ یا ورودی خروجی از نوع غیر ارجاع به حافظه non - MRI
- ۳- شبه دستورالعمل‌ها با، و بدون عملوند

یک دستورالعمل حافظه‌ای دو یا سه سمبل را که با فواصلی از هم جدا شده‌اند استفاده می‌کند. سمبل اول باید سه حرفی باشد که یکی از کدهای عمل MRI جدول ۱-۶ را تعریف می‌نماید. دومین سمبل، آدرس سمبلیک است. سومین سمبل، که ممکن است وجود داشته یا نداشته باشد، حرف I است. اگر I وجود نداشته باشد، سطر، بیانگر آدرس مستقیم است. حضور I بمعنی غیرمستقیم بودن

1- Field

2- Memory- Reference Instructions

3- Register - Reference Instruction

آدرس دستورالعمل است.

دستورالعمل غیرارجاع به حافظه دستوری است که بخش آدرس ندارد. نشانه غیرارجاع به حافظه در میدان دستورالعمل برنامه بوسیله هر یک از سمبل های سه حرفی در لیست جدول ۶-۱ برای دستورات ثباتی و ورودی تشخیص داده می شود.

مثالهای زیر نمونه ایی از سمبل هایی است که می توان در میدان دستورالعمل یک برنامه قرار داد.

CLA	non-MRI
ADD OPR	MRI با آدرس مستقیم
ADD PRT I	MRI با آدرس غیرمستقیم

سمبل سه حرفی هر سطر باید یکی از دستورالعمل های کامپیوتر در جدول ۶-۱ باشد. بدنبال یک دستور، مانند ADD، باید یک آدرس سمبلیک وجود داشته باشد. وجود حرف I اختیاری است. آدرس سمبلیک در میدان آدرس، مکان حافظه مربوط به عملوند را معین می کند. این مکان مجدداً باید در جایی از برنامه بصورت عنوان در ستون اول در جلو دستورالعمل آورده شود. برای اینکه بتوان یک برنامه بزیان اسمبلی را به یک برنامه دودویی ترجمه کرد، ضروری است، هر آدرس سمبلیک که در میدان آدرس ذکر شده است مجدداً در میدان عنوان (برچسب) ظاهر گردد.

شبه دستور، یک دستورالعمل ماشین نیست ولی دستورالعمل برای اسمبلر است تا اطلاعاتی از بعضی فازهای ترجمه را در اختیار آن قرار دهد. چهارشبه دستوری که اسمبلر آنها را می شناسد در جدول ۶-۷ ارائه شده اند. (برنامه های اسمبلی دیگر شبه دستورات متعدد دیگری دارند). شبه دستور ORG به اسمبلر اطلاع می دهد که دستورالعمل یا عملوند سطر بعدی قرار است در مکانی از حافظه که بوسیله عدد بعد از ORG مشخص می شود قرار گیرد. امکان استفاده بیش از یکبار ORG در برنامه برای مشخص کردن بیش از یک قطعه از حافظه وجود دارد. سمبل END در انتهای برنامه قرار گرفته است تا اسمبلر را از ختم برنامه مطلع سازد. دو شبه دستورالعمل دیگر پایه^۱ عملوند را مشخص کرده و به

جدول ۶-۷ تعریف شبه دستورالعمل ها

سمبل	اطلاعاتی که به اسمبلر داده می شود
ORG N	عدد شانزده شازدهمی N مکانی از حافظه است که دستورالعمل با عملوند واقع در سطر بعدی باید در آنجا قرار داده شود
END	انتهای برنامه سمبلیک
DEC N	عدد دهدهمی علامتدار N که باید به دودویی تبدیل گردد
HEX N	عدد شانزده شازدهمی N که باید به دودویی تبدیل گردد

اسمبلر می‌گویند که چگونه اعداد لیست شده را به اعداد دودویی تبدیل کند. میدان سوم در یک برنامه برای ارائه توضیحات در نظر گرفته شده است. هر خط از کد ممکن است توضیح داشته و یا نداشته باشد، ولی اگر داشت، باید برای اسمبلر بوسیله یک ممیز آغاز شود. توضیحات برای تشریح برنامه مفید بوده و به درک قدم به قدم روش اتخاذ شده بوسیله برنامه کمک می‌کند. چون توضیحات فقط نقش تشریحی را دارند لذا در مرحله ترجمه از آنها چشم‌پوشی می‌شود.

مثال

برنامه جدول ۸-۶ مثالی از یک برنامه زبان اسمبلی است. در اولین سطر، شبه دستور ORG برای تعریف مبدأ برنامه در مکان حافظه 100 قرار دارد. شش خط بعد دستورالعمل‌های ماشین را تعریف می‌کنند و چهار سطر آخر هم شبه دستورالعمل می‌باشند. سه آدرس سمبلیک بکاررفته و هر یک از آنها در ستون ۱ به صورت عنوان و در ستون ۲ به صورت آدرس یک دستور حافظه‌ای آورده شده‌اند. سه شبه دستور عملوند را مشخص می‌نمایند، و آخرین سطر نیز پایان برنامه را مشخص می‌کند. وقتی که برنامه به کد دودویی ترجمه شده و توسط کامپیوتر اجرا شود، تفریق بین دو عدد را انجام می‌دهد. عمل تفریق بوسیله جمع مفروق‌منه با متمم 2 مفروق انجام می‌شود. مفروق یک عدد منفی است. این عدد به یک عدد دودویی در نمایش متمم 2 علامت‌دار بدل می‌شود زیرا خواسته‌ایم تا تمام اعداد دودویی بشکل متمم 2 باشند. وقتی که متمم 2 مفروق بدست آمد (با متمم کردن و افزایش دادن AC)، -23 به $+23$ تبدیل و اختلاف $106 = 83 + 23 = 83 + (-23)$ (متمم 2 عدد -23) بدست می‌آید.

ترجمه به دودویی

ترجمه برنامه سمبلیک به دودویی توسط برنامه خاصی بنام اسمبلر صورت می‌گیرد. اگر ابتدا ترجمه را روی کاغذ انجام دهیم کارهای انجام شده بوسیله اسمبلر بهتر درک خواهد شد. ترجمه برنامه

جدول ۸-۶ برنامه زبان اسمبلی برای تفریق دو عدد

ORG 100	مبدأ برنامه مکان 100 است /
LDA SUB	بار کردن مفروق در AC /
CMA	متمم کردن AC /
INC	افزایش AC /
ADD MIN	جمع کردن مفروق‌منه با AC /
STA DIF	ذخیره تفاضل /
HLT	توقف کامپیوتر /
MIN, DEC 83	مفروق منه /
SUB, DEC -23	مفروق /
DIF, HEX 0	محل ذخیره تفاضل /
END	پایان برنامه سمبلیک /

جدول ۸-۶، به کد دودویی معادل آن را می توان با مرور برنامه و جایگزین کردن سمبل ها با معادل دودویی کد ماشین آنها انجام داد. با شروع از سطر اول، ابتدا با شبه دستورالعمل ORG مواجه می شویم. این دستور می گوید که برنامه دودویی را از آدرس 100 شانزده شانزدهمی شروع کنیم. در سطر دوم دو سمبل بکار رفته است. این سطر باید یک دستورالعمل حافظه ای باشد که هر خانه 100 حافظه قرار دارد. چون حرف I وجود ندارد بیت اول کد دستورالعمل باید 0 باشد. نام سمبلیک دستورالعمل LDA است. با ملاحظه جدول ۱-۶ می بینیم که اولین رقم شانزده شانزدهمی دستور باید 2 باشد، مقدار دودویی بخش آدرس باید از آدرس SUB بدست آید. ستون عنوان را مرور می کنیم و می بینیم که این سمبل در سطر ۹ واقع شده است. برای تعیین مقدار شانزده شانزدهمی می بینیم که سطر ۲ حاوی دستورالعملی برای مکان 100 است و هر یک از سطرهای بعدی یک دستور ماشین یا یک عملوند را برای خانه های متوالی حافظه مشخص می کنند. با شمارش سطرها، می بینیم که عنوان SUB در سطر ۹ مربوط به مکان 107 است. بنابراین آدرس شانزده شانزدهمی دستور LDA باید 107 باشد. وقتی که دو بخش دستور مونتاز شوند، کد شانزده شانزدهمی 2107 حاصل می گردد. سطرهای دیگری که نمایش دهنده دستورات ماشین هستند به روش مشابهی ترجمه شده و کد شانزده شانزدهمی آنها در جدول ۹-۶ لیست شده است.

دو سطر در برنامه سمبلیک که با شبه دستور DEC مشخص شده اند عملوندهای دهنده را مشخص می نمایند. سومین سطر صفری را با شبه دستور HEX معین می کند (DEC نیز می تواند بکار رود). عدد دهنده 83 به دودویی تبدیل شده و در مکان 106 بفرم شانزده شانزدهمی قرار داده شده است. عدد دهنده 23- یک عدد منفی است و باید بفرم متمم 2 علامت دار دودویی تبدیل شود. معادل شانزده شانزدهمی عدد دودویی در مکان 107 قرار دارد. سمبل END انتهای برنامه را خبر می دهد و می گوید که دیگر سطری برای ترجمه وجود ندارد.

جدول ۹-۶ لیست برنامه ترجمه شده جدول ۸-۶

کد شانزده شانزدهمی		برنامه سمبلیک	
عمل	محتوا		
			ORG 100
100	2107		LDA SUB
101	7200		CMA
102	7020		INC
103	1106		ADD #MIN
104	3108		STA DIF
105	7001		HLT
106	0053	MIN,	DEC 83
107	FFE9	SUB,	DEC -23
108	0000	DIF,	HEX 0
			END

اگر ما دو مرور کامل بر روی برنامه سمبلیک داشته باشیم عمل ترجمه ساده می شود. در مرور اول ترجمه ای صورت نمی گیرد بلکه فقط مکانی برای هر دستورالعمل ماشین و عملوند در حافظه تعلق می گیرد. تخصیص مکان مقدار آدرس را برای عناوین تعریف کرده و برای فرآیند ترجمه که در مرور دوم صورت می گیرد تسهیلات لازم را فراهم می آورد. بنابراین ما در جدول ۹-۶ مکان 100 را به اولین دستور بعد از ORG اختصاص می دهیم. سپس مکان های متوالی بعدی را تا انتهای برنامه برای هر یک از سطرهای کد که حاوی دستور یا عملوند هستند، اختصاص می دهیم. (به ORG و END عددی اختصاص نمی یابد زیرا دستور و یا عملوند نیستند). وقتی که اولین مرور برنامه تکمیل شد، برای هر عنوان شماره مکان آن را مشخص می کنیم و جدولی را تشکیل می دهیم که مقدار شانزده شانزدهمی هر آدرس سمبلیک را نشان می دهد. برای این برنامه جدول سمبل آدرس برابر زیر است:

آدرس شانزده شانزدهمی	سمبل آدرس
106	MIN
107	SUB
108	DIF

در دومین مرور به برنامه سمبلیک، با مراجعه به جدول سمبل آدرس ها مقدار آدرس هر دستورالعمل حافظه ای را تعیین می کنیم. مثلاً سطر کد LDA SUB در مرور دوم با برداشتن مقادیر شانزده شانزدهمی LDA از جدول ۱-۶ و SUB از لیست جدول سمبل آدرس ترجمه می شود. سپس دو بخش را بصورت یک دستورالعمل در مبنای شانزده کنار هم^۱ می گذاریم. اگر بخواهیم نحوه قرار گرفتن این برنامه را در کامپیوتر بدانیم می توانیم به آسانی کد شانزده شانزدهمی حاصل را به دودویی تبدیل کنیم. وقتی که ترجمه از سمبل ها به دودویی بوسیله یک برنامه اسمبلر انجام شد، اولین مرور، اولین گذر خوانده شده و دومین مرور نیز دومین گذر نامیده می شود.

۴-۶ اسمبلر

اسمبلر برنامه ای است که یک برنامه بزبان اسمبلی را دریافت و معادل زبان ماشین دودویی آن را تولید می کند. برنامه سمبلیک ورودی برنامه منبع^۲ و برنامه دودویی حاصل برنامه مقصد^۳ خوانده می شود. در واقع برنامه اسمبلر روی رشته های حرفی عمل کرده و معادل دودویی آنها را تولید می نماید.

1- Assemble

2- Source Program

3- Object Program

نمایش برنامه سمبلیک در حافظه

قبل از آغاز اسمبل کردن (مونتاژ کردن)، برنامه سمبلیک باید در حافظه ذخیره شود. کاربر برنامه سمبلیک را از یک پایانه تایپ می‌کند. با استفاده از یک برنامه پارکننده، کاراکترهای برنامه سمبلیک را در حافظه وارد می‌نماید. چون برنامه از مجموعه‌ای از سمبل‌ها ساخته شده است، نمایش آن در حافظه با استفاده از کد کاراکترهای الفباء عددی است. در کامپیوتر پایه، هر کاراکتر با یک کد هشت بیتی نمایش داده می‌شود. همواره با ارزشترین بیت 0 است و هفت بیت دیگر بوسیله ASCII مشخص می‌شود. معادل شانزده شانزدهی مجموعه کاراکترها در جدول ۱۰-۶ آورده شده است. به هر کاراکتر دو رقم مبنای شانزدهی تخصیص یافته است که بسادگی قابل تبدیل به کد معادل هشت بیتی‌اش می‌باشد. آخرین وارده در جدول، کاراکتری را تولید نمی‌کند بلکه مربوط به حرکت فیزیکی مکان‌نما^۱ بر روی صفحه پایانه است. کد مربوطه به CR هنگامی تولید می‌شود که کلید Return فشار داده شود. این باعث بازگشت "تورد"^۲ به مکان اولیه‌اش برای شروع سطر جدید می‌گردد. اسمبلر کد CR را بعنوان کد انتهای سطر می‌شناسد. هر سطر کد در خانه‌های متوالی حافظه بصورت دو کاراکتر در هر مکان ذخیره می‌شود. در هر کلمه دو کاراکتر را می‌توان ذخیره کرد زیرا هر کلمه حافظه دارای 16 بیت ظرفیت است. هر سمبل عنوان به یک کاما ختم می‌شود. سمبل‌های عملیات و آدرس با یک فاصله^۳ و پایان هر سطر را، کد CR مشخص می‌کند. مثلاً سطر کد

PL3, LDA SUB I

جدول ۱۰-۶ کد شانزده شانزدهی کاراکترها

کاراکتر	کد	کاراکتر	کد	کاراکتر	کد
A	41	Q	51	6	36
B	42	R	52	7	37
C	43	S	53	8	38
D	44	T	54	9	39
E	45	U	55	space	20
F	46	V	56	(28
G	47	W	57)	29
H	48	X	58	*	2A
I	49	Y	59	+	2B
J	4A	Z	5A	.	2C
K	4B	0	30	-	2D
L	4C	1	31	.	2E
M	4D	2	32	/	2F
N	4E	3	33	=	3D
O	4F	4	34	CR	0D برگشت تورد
P	50	5	35		

1- Cursor

2- Carriage

3- Space

در هفت کلمه حافظه متوالی، مطابق جدول ۱۱-۶، ذخیره شده است. عنوان PL3 دو کلمه را اشغال می‌کند و به یک کاما (2C) منتهی می‌شود. میدان دستورالعمل در سطر کد ممکن است دارای یک یا چند سمبل باشد. هر سمبل به کد مربوط به فاصله (20) ختم می‌شود، بجز آخرین سمبل که به کد برگشت نورد (0D) ختم می‌گردد. اگر سطر کد دارای توضیحات باشد، اسمبلر آن را بوسیله ممیز یعنی 2F شناسایی می‌کند. اسمبلر کلیه کاراکترهای میدان توضیحات را چشم‌پوشی کرده و بدنبال کد CR می‌گردد. پس از اینکه با این کد مواجه شد، به جای آن کد فاصله را پس از آخرین سمبل سطر قرار می‌دهد. ورودی برنامه اسمبلر بزبان سمبلیک در ASCII است. این ورودی بوسیله اسمبلر دوبار مرور می‌شود تا برنامه معادل دودویی آن تولید شود. برنامه دودویی، خروجی تولید شده توسط اسمبلر می‌باشد. اکنون ما بطور مختصر عمده کارهای اسمبلر را حین ترجمه تشریح می‌کنیم.

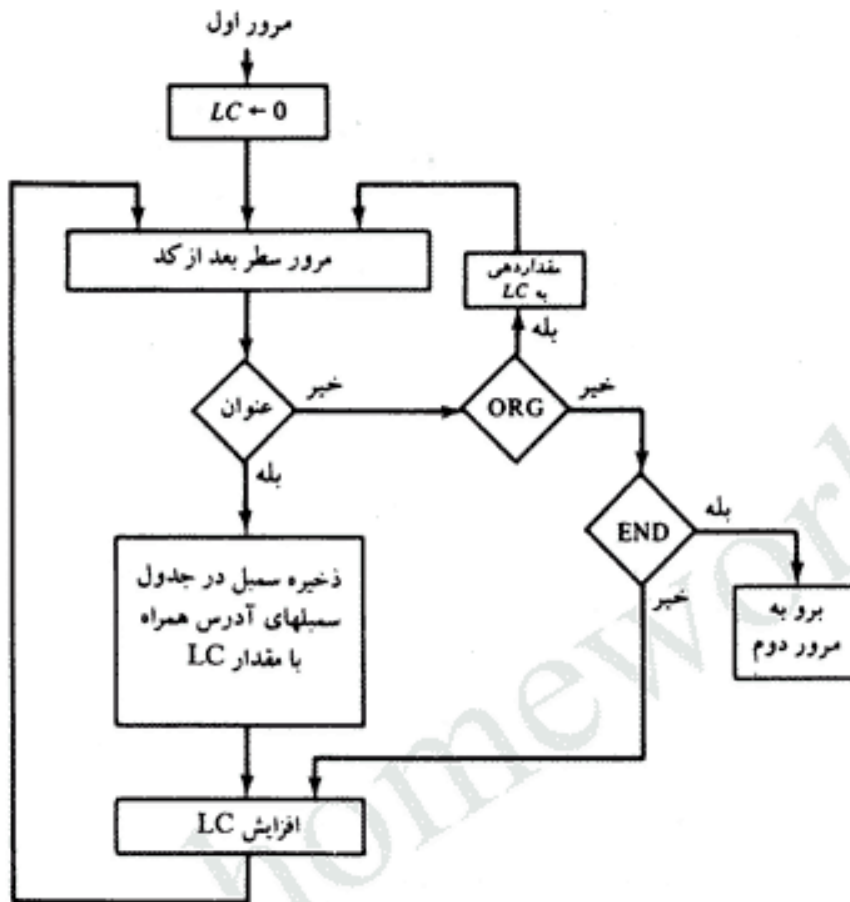
مرور اول

یک اسمبلر دو مروری سراسر برنامه سمبلیک را دوبار مرور می‌کند. در مرور اول، جدولی را تولید می‌کند که مقدار معادل دودویی همه سمبل‌های آدرس تعریف شده توسط کاربر را نشان می‌دهد. ترجمه به دودویی در مرور دوم صورت می‌گیرد. برای دنبال کردن محل دستورالعمل‌ها، اسمبلر از کلمه‌ای در حافظه به نام مکان‌شمار^۱ (بطور خلاصه LC) استفاده می‌کند. محتوای LC، آدرس محلی از حافظه را که به دستورالعمل یا عملوند در دست پردازش اختصاص داده می‌شود، مشخص می‌کند. شبه دستور ORG آدرس اولین مکان حافظه در نظر گرفته شده برای برنامه را به عنوان مقدار اولیه به مکان شمار می‌دهد. چون دستورالعمل‌ها متوالیاً ذخیره می‌شوند، پس از پردازش هر سطر کد، LC افزایش می‌یابد. برای جلوگیری از هرگونه ابهام در غیاب ORG، اسمبلر در آغاز مقدار 0 را به LC می‌دهد. کارهایی که توسط اسمبلر در اولین مرور صورت می‌گیرد در فلوچارت شکل ۱-۶ توصیف شده

جدول ۱۱-۶ نمایش سطر کد PL3, LDA SUB I در کامپیوتر

نمایش دودویی	کد شانزده شانزده می	سمبل	کلمه حافظه
0101 0000 0100 1100	50 4C	P L	1
0011 0011 0010 1100	33 2C	3 ,	2
0100 1100 0100 0100	4C 44	L D	3
0100 0001 0010 0000	41 20	A	4
0101 0011 0101 0101	53 55	S U	5
0100 0010 0010 0000	42 20	B	6
0100 1001 0000 1101	49 0D	I CR	7

1- Location Counter



شکل ۶-۱ فلوچارت مرور اول اسمبلر

است. LC ابتدا 0 می شود. خط اول کد سمبلیک تحلیل می شود تا معین شود آیا دارای عنوان است یا خیر (این کار با وجود کاما مشخص می گردد). اگر سطر فاقد عنوان باشد، اسمبلر سمبل را در میدان دستورالعمل چک می کند. اگر این میدان حاوی شبه دستور ORG باشد، اسمبلر LC را با مقداری که در جلوی ORG آمده است بار می کند و برای پردازش سطر بعدی باز می گردد. اگر سطر دارای شبه دستور END باشد، اسمبلر مرور اول را خاتمه می دهد و به سراغ مرور دوم می رود (دقت کنید سطر ORG و END دارای عنوان نیست). اگر سطر مورد نظر دارای عنوان باشد، این عنوان همراه با معادل دودویی اش که در LC قرار دارد در جدول سمبل آدرس قرار داده می شود. اگر عنوان وجود نداشته باشد چیزی در جدول ذخیره نمی شود. سپس LC یک واحد افزایش یافته و سطر جدیدی پردازش می گردد.

برای برنامه جدول ۸-۶، اسمبلر جدول سمبل آدرس در جدول ۱۲-۶ را تولید می کند. هر سمبل عنوان در دو مکان حافظه ذخیره می شود و بوسیله یک کاما خاتمه می یابد. اگر عنوان کمتر از سه کاراکتر داشته باشد، مکان های حافظه اضافی با کد فاصله پر می شود. مقداری که در LC در حین پردازش سطر وجود دارد در حافظه بعدی ذخیره می شود. این برنامه سه آدرس سمبلیک دارد: SUB، MIN و DIF.

جدول ۱۲-۶ جدول سمبل های آدرس برای برنامه جدول ۸-۶

کلمه حافظه	سیل با (LC)*	کد شانزده شانزدهمی	نمایش دودویی
1	M I	4D 49	0100 1101 0100 1001
2	N ,	4E 2C	0100 1110 0010 1100
3	(LC)	01 06	0000 0001 0000 0110
4	S U	53 55	0101 0011 0101 0101
5	B ,	42 2C	0100 0010 0010 1100
6	(LC)	01 07	0000 0001 0000 0111
7	D I	44 49	0100 0100 0100 1001
8	F ,	46 2C	0100 0110 0010 1100
9	(LC)	01 08	0000 0001 0000 1000

* محتوای شانزده مکان را مشخص می کند (LC)

این سمبل ها بترتیب آدرس های 12 بیتی را که معادل 106 ، 107 و 108 می باشند نمایش می دهند. جدول سمبل آدرس سه کلمه حافظه فوق را برای هر سمبل عنوان که با آن مواجه شود اشغال می کند و داده های خروجی را که اسمبلر در حین مرور اول ایجاد می کند تشکیل می دهد.

مرور دوم

دستورات ماشین در حین مرور دوم با استفاده از روش های نظاره به جدول^۱ ترجمه می شوند. روش نظاره به جدول جستجویی است بر وارده های جدول برای اینکه تعیین کنیم آیا یک نمونه مورد نظر با نمونه های ذخیره شده در جدول مطابقت دارد یا خیر. اسمبلر چهار جدول را بکار می برد. هر سمبلی که در برنامه با آن مواجه شود باید بعنوان یکی از وارده ها در این جدول موجود باشد؛ در غیر اینصورت سمبل قابل تفسیر نیست. ما نام های زیر را به چهار جدول اختصاص می دهیم:

۱- جدول شبه دستورالعمل ها

۲- جدول MRI

۳- جدول non-MRI

۴- جدول سمبل آدرس

وارده های جدول شبه دستورالعمل ها چهار سمبل دارد: END ، ORG ، DEC و HEX. هر وارده هنگام برخورد با آن در برنامه، اسمبلر را به زیرروال شبه دستور مربوطه، هدایت می کند. جدول MRI حاوی هفت سمبل دستورالعمل های حافظه ای و کد عمل معادل سه بیتی آنهاست. جدول non-MRI حاوی هجده دستورالعمل ثباتی و ورودی- خروجی همراه با شانزده کد دودویی معادل آنها می باشد.

1- Table look up

جدول سمبل آدرس در مرور اول پردازش اسمبلی تولید می‌گردد. اسمبلر این جداول را برای یافتن سمبلی که در حال پردازش است جستجو می‌کند تا مقدار دودویی آن را معین کند.

کاری که در حین دومین مرور توسط اسمبلر انجام می‌شود در شکل ۲-۶ آمده است. LC در آغاز با 0 بار می‌شود. سپس سطرهای کد یک به یک تحلیل می‌شوند. عناوین در مرور دوم چشم‌پوشی می‌شوند، لذا اسمبلر بلافاصله به میدان آدرس رفته و اولین سمبلی را که با آن مواجه می‌شود واری می‌کند. ابتدا جدول شبه دستور را چک می‌نماید. تطابق با ORG، اسمبلر را به زیرروالی می‌فرستد که در آن LC با یک مقدار اولیه بار می‌شود. هر تطابق با END، فرآیند ترجمه را خاتمه می‌دهد. شبه دستورات عملوندی موجب تبدیل عملوند به دودویی‌شان می‌گردند. این عملوند در خانه‌ای از حافظه که بوسیله LC مشخص می‌شود قرار می‌گیرد. سپس شمارنده مکان، LC، یک واحد افزایش می‌یابد و اسمبلر به تحلیل کد سطر بعدی ادامه می‌دهد.

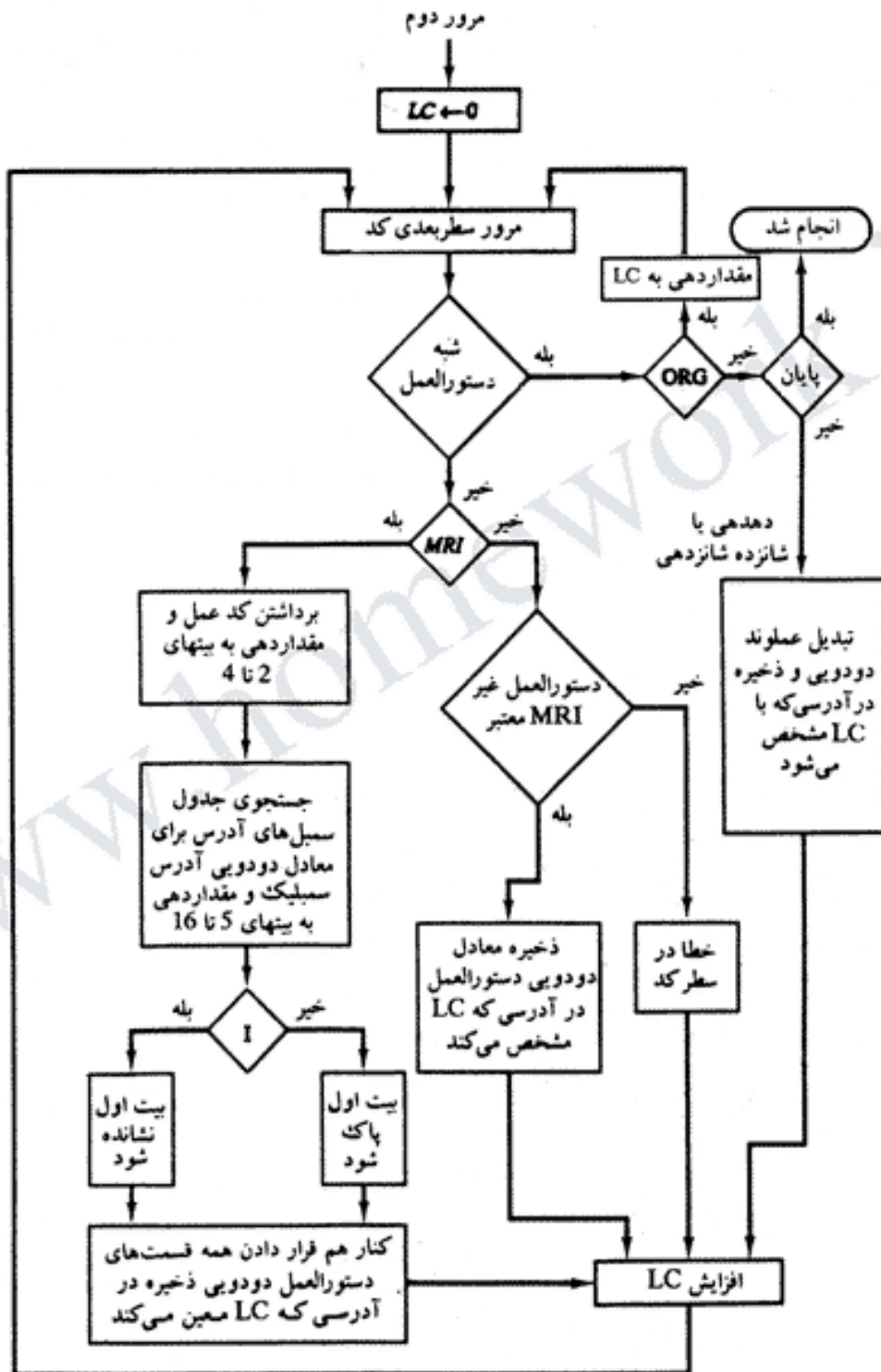
اگر سمبلی که با آن برخورد شده شبه دستور نباشد، اسمبلر به جدول MRI مراجعه می‌کند. اگر سمبل در این جدول پیدا نشود، اسمبلر به جدول non-MRI می‌رود. سمبلی که در این جدول پیدا شود متعلق به دستورات عمل‌های ثابتی یا ورودی-خروجی است. اسمبلر کد شانزده بیتی دستورات عمل را در کلمه حافظه‌ای که بوسیله LC مشخص می‌گردد ذخیره می‌کند. مکان‌شمار، LC، افزایش یافته و تحلیل ادامه می‌یابد.

هنگامی که سمبلی در جدول MRI پیدا شود، اسمبلر کد سه بیتی آن را استخراج کرده و آن را در بیت‌های 2 الی 4 از یک کلمه قرار می‌دهد. یک دستورات عمل حافظه‌ای با دو یا سه سمبل مشخص می‌شود. سمبل دوم آدرس سمبلیک است و سومی، که ممکن است وجود داشته باشد یا نداشته باشد حرف I است. آدرس سمبلیک با جستجوی جدول سمبل آدرس به دودویی تبدیل می‌گردد. بیت اول دستور، بسته به اینکه حرف I وجود داشته باشد یا خیر، 1 یا 0 است. سه جزء کد دستورات عمل در کنار هم گذاشته شده¹ (یا به بیان دیگر مونتاژ شده) و سپس در مکانی از حافظه که توسط محتوای LC مشخص می‌گردد، ذخیره می‌شود. مکان‌شمار افزایش یافته و اسمبلر به پردازش سطر بعدی ادامه می‌دهد.

عمل مهمی که یک اسمبلر می‌باید انجام دهد یافتن خطاهای ممکن در برنامه سمبلیک است. این کار تشخیص خطا² خوانده می‌شود. چنین خطایی ممکن است سمبل نامعتبر کد ماشین باشد که نشانه آن عدم وجودش در جدول‌های MRI و non-MRI است. اسمبلر قادر به ترجمه چنین سمبلی نیست زیرا مقدار معادل دودویی آن را نمی‌داند. در چنین حالتی، اسمبلر پیام خطایی را چاپ می‌کند تا برنامه‌نویس را از وجود خطا در سطر خاصی از کد برنامه سمبلیک مطلع سازد. خطای ممکن دیگر این است که برنامه دارای آدرس سمبلیکی باشد که در جدول وجود ندارد اسمبلر قادر نخواهد بود این سطر کد را بطور صحیحی ترجمه کند زیرا معادل دودویی آن سمبل را در جدول سمبل آدرس که در مرور اول تولید شده نمی‌یابد. خطاهای دیگری هم ممکن است رخ دهند و یک اسمبلر قابل استفاده در عمل باید

1- Assembled

2- Diagnostic



شکل ۲-۶ فلوجارت مرور دوم اسمبلر

همه این گونه خطاها را کشف و برای هر یک پیغام خطایی را چاپ کند. باید تأکید کرد که یک اسمبلر عملی بسیار پیچیده تر از آنچه در اینجا شرح داده ایم می باشد. اکثر کامپیوترها انعطاف زیادی را در نوشتن برنامه ها ب زبان اسمبلی در اختیار برنامه نویس می گذارند. مثلاً ممکن است به کاربر اجازه داد تا برای مشخص کردن آدرس از عدد یا یک اسمبل استفاده کند. بسیاری از اسمبلرها این امکان را به کاربر می دهند که آدرس را با یک عبارت ریاضی مشخص کند. شبه دستورالعمل های بسیار زیادی را نیز ممکن است برای تسهیل در کار برنامه نویس مشخص کرد. با کاراتر شدن و تکمیل زبان اسمبلی، اسمبلر پیچیده تر می شود.

۵-۶ حلقه در برنامه نویسی

حلقه برنامه رشته ای از دستورات است که چندین بار، و هر بار با مجموعه متفاوتی از داده ها، اجرا می شود. حلقه برنامه در فرترن با عبارت DO مشخص می شود. مثال زیر برنامه ای ب زبان فرترن است که مجموع 100 عدد صحیح را تهیه می کند.

```
DIMENSION A(100)
INTEGER SUM, A
SUM = 0
DO 3 J = 1, 100
3 SUM = SUM + A (J)
```

عبارت شماره 3، بتعداد 100 بار تکرار می شود و هر بار عملوند متفاوت بوده و مقادیر $A(J)$ که در آن $J = 0, 1, \dots, 100$ می باشد بکار می رود.

برنامه سیستمی که یک برنامه ب زبان سطح بالا را، مانند آنچه در بالا دیدیم به زبان ماشین ترجمه کند، کامپایلر^۱ نام دارد. کامپایلر در مقایسه با اسمبلر، برنامه پیچیده تری است و درک کامل عملکرد آن مستلزم دانش برنامه نویسی سیستم است. معهذاً، می توانیم کارهای اصلی یک کامپایلر را با دنبال کردن فرآیند ترجمه یک برنامه و رای آنچه که در برنامه زبان اسمبلی بود نشان دهیم. یک کامپایلر می تواند از یک برنامه اسمبلی بعنوان مرحله میانی برای ترجمه استفاده کند و یا مستقیماً آنرا به دودویی تبدیل نماید.

اولین عبارت در برنامه فرترن یک دستور DIMENSION است. این عبارت به کامپایلر فرمان می دهد تا 100 کلمه حافظه را برای 100 عملوند رزرو نماید. مقدار عملوندها از یک عبارت ورودی^۲ بدست می آیند (دستور مزبور در برنامه بالا آورده نشده است). عبارت دوم به کامپایلر خبر می دهد که

1- Compiler

2- INPUT

اعداد صحیح هستند. اگر اعداد از نوع غیر صحیح^۱ باشند، کامپایلر باید مکان‌هایی را برای اعداد ممیز شناور رزرو نموده و دستوراتی را تولید کند که عملیات محاسباتی بعدی را با داده‌های ممیز شناور انجام دهند. این دو دستور همانند شبه دستورات عمل‌ها در اسمبلی، غیر قابل اجرا هستند. فرض کنید کامپایلر مکان‌های 16(150) تا 16(1B3) را برای 100 عملوند رزرو نماید. این کلمات حافظه رزرو شده در سطرهای 19 الی 118 برنامه ترجمه شده جدول ۱۳-۶ لیست شده‌اند. این کار توسط شبه دستور ORG در سطر 18 انجام می‌شود که ابتدای عملوندها را مشخص می‌کند. اولین و آخرین عملوندها توسط عدد دهمی خاصی نشان داده شده‌اند، اگرچه این مقادیر ضمن ترجمه مشخص نیستند. کامپایلر فقط فضای داده‌ها را در حافظه رزرو کرده و مقادیر بعداً پس از اجرای عبارت ورودی INPUT در آنها جای می‌گیرد. شماره سطرها در برنامه سمبلیک فقط به منظور ارجاع می‌باشند و نه بخشی از برنامه ترجمه شده آن. اندیس موجود در عبارت DO بصورت دستورات عمل‌های سطرهای 2 الی 5 و ثابت‌های سطرهای

جدول ۱۳-۶ برنامه سمبلیک برای جمع 100 عدد

سطر			
1		ORG 100	مبداء برنامه 100 شانزده شانزدهی /
2		LDA ADS	بار کردن آدرس ابتدای عملوندها /
3		STA PTR	ذخیره در اشاره گر /
4		LDA NBR	بار کردن 100 منفی /
5		STA CTR	ذخیره در شمارنده /
6		CLA	پاک کردن انباشتگر /
7	LOP,	ADD PTR I	جمع یک عملوند با AC /
8		ISZ PTR	افزایش اشاره گر /
9		ISZ CTR	افزایش شمارنده /
10		BUN LOP	تکرار مجدد حلقه /
11		STA SUM	ذخیره مجموع /
12		HLT	توقف /
13	ADS,	HEX 150	آدرس ابتدای عملوندها /
14	PTR,	HEX 0	محل نگهداری شده برای اشاره گر /
15	NBR,	DEC -100	ثابتی که بعنوان مقدار اولیه به اشاره گر داده می‌شود /
16	CTR,	HEX 0	محل ذخیره شده برای شمارنده /
17	SUM,	HEX 0	محل ذخیره مجموع /
18		ORG 150	مبداء عملوندها 150 در مبنای شانزده شانزدهی است /
19		DEC 75	نخستین عملوند /
.			
.			
.			
118		DEC 23	آخرین عملوند /
119		END	انتهای برنامه نمادین /

1- Real

13 تا 16 ترجمه می شود. آدرس اولین عملوند (150) در مکان ADS در سطر 13 ذخیره می شود. تعداد دفعاتی که دستور شماره 3 برنامه فترن اجرا می شود 100 است. بنابراین عدد 100- در مکان NBR ذخیره می گردد. سپس کامپایلر دستورات سطرهای 2 الی 5 را برای ایجاد مقادیر اولیه حلقه برنامه تولید می نماید. آدرس اولین عملوند به مکان PTR منتقل می گردد. این کار با توجه به (1)A حاصل از (J)A صورت می گیرد. سپس عدد 100- به مکان CTR منتقل می گردد. این مکان بعنوان شمارنده عمل کرده و هر بار که حلقه برنامه اجرا شود یک واحد افزایش می یابد. وقتی که شمارنده به صفر برسد، عمل مورد نظر یکصد بار انجام شده و برنامه از حلقه خارج می گردد.

برخی از کامپایلرها دستور $SUM = 0$ را به یک دستورالعمل ماشین که مکان SUM را بعنوان مقدار اولیه صفر مینماید ترجمه می کنند. سپس ارجاع به این مکان هر بار که سطر 3 برنامه فترن اجرا شود صورت می گیرد. کامپایلر هوشمندتر می تواند چنین تشخیص دهد که جمع در انباره¹ صورت گرفته و فقط نتیجه نهایی در SUM قرار گیرد. کامپایلر دستورالعملی در سطر 6 تولید خواهد کرد تا انباره AC را پاک کند. همچنین مکانی از حافظه که با SUM مشخص شده است (در خط 17) برای ذخیره مقدار این متغیر در انتهای حلقه نیز بوسیله کامپایلر رزرو می شود.

حلقه مشخص شده با عبارت DO به رشته دستورالعمل های واقع در سطرهای 7 تا 10 ترجمه می شود. سطر 7 دستورالعمل ADD غیر مستقیم است زیرا دارای سمبل I می باشد. آدرس عملوند جاری در خانه PTR ذخیره می شود. وقتی این مکان بطور غیرمستقیم آدرس دهی شود کامپیوتر محتوای PTR را بعنوان آدرس عملوند تلقی می نماید. در نتیجه، عملوند در مکان 150 با انباره جمع می شود. سپس PTR با دستورالعمل ISZ در سطر 8 یک واحد افزایش می یابد، و مقدار آن برابر آدرس عملوند بعدی می گردد. مکان CTR در سطر 9 افزایش می یابد و اگر صفر نباشد، کامپیوتر از دستورالعمل بعدی گذر نمی کند. دستور بعدی یک دستورالعمل انشعاب (BUN) به ابتدای حلقه است، لذا کامپیوتر برای تکرار مجدد حلقه باز می گردد. وقتی مکان CTR صفر شود (پس از 100 بار تکرار حلقه) دستور بعدی گذر² (صرفنظر) می شود و کامپیوتر دستورالعمل سطر 11 و 12 را اجرا می نماید. جمع حاصل در انباره در SUM ذخیره شده و کامپیوتر متوقف می شود. دستورالعمل توقف³ برای وضوح برنامه در اینجا قرار داده شده است؛ در واقع برنامه به محلی دیگر برای ادامه اجرای برنامه منتقل می گردد و یا به آغاز برنامه دیگری انشعاب می نماید. توجه کنید که ISZ در سطر 8 صرفاً برای افزایش اشاره گر آدرس PTR بکار رفته است. چون آدرس عددی مثبت است هرگز گذری رخ نمی دهد.

برنامه جدول ۱۳-۶ برای نخستین بار مفاهیم اشاره گر، شمارنده همراه با عمل آدرس دهی غیرمستقیم را بکار برده است تا یک حلقه را در برنامه بوجود آورد. اشاره گر به آدرس عملوند جاری اشاره می نماید و شمارنده تعداد دفعات اجرای حلقه برنامه را می شمارد. در این مثال ما دو حافظه را برای این مقاصد استفاده کردیم. در کامپیوترهایی که بیش از یک ثبات پردازنده دارند می توان از یک ثبات بعنوان

1- Accumulator

2- Skip

3- HALT

اشاره گر، از دیگری بعنوان شمارنده و از سومی بجای انباره استفاده کرد. وقتی که از ثبات های پردازنده بعنوان اشاره گر و شمارنده استفاده شود به آنها ثبات های شاخص^۱ گفته می شود. ثبات های شاخص در بخش ۵-۸ مورد بحث قرار گرفته اند...

۶-۶ برنامه نویسی اعمال حسابی و منطقی

تعداد دستورالعمل های موجود در یک کامپیوتر از چند صد دستور در سیستم های بزرگ تا چند دوجین در کامپیوترهای کوچک متغیر است. برخی کامپیوترها عمل مفروضی را با یک دستورالعمل انجام می دهند و برخی دیگر برای انجام همان عمل ممکن است تعداد زیادی دستورات ماشین نیاز داشته باشند. مثلاً چهار عمل اصلی حساب را در نظر بگیرید. بعضی کامپیوترها دارای دستورات ماشین برای جمع، تفریق، ضرب و تقسیم هستند. برخی دیگر، مانند کامپیوتر پایه، فقط دارای یک دستور حسابی، ADD، است. اعمالی که در مجموعه دستورالعمل های ماشین گنجانده نشده باشد باید بوسیله یک برنامه پیاده سازی شود. ما در جدول ۸-۶ برنامه ای را برای تفریق دو عدد دیدیم. برنامه های مربوط به سایر عملیات حسابی به روشی مشابه بدست می آید.

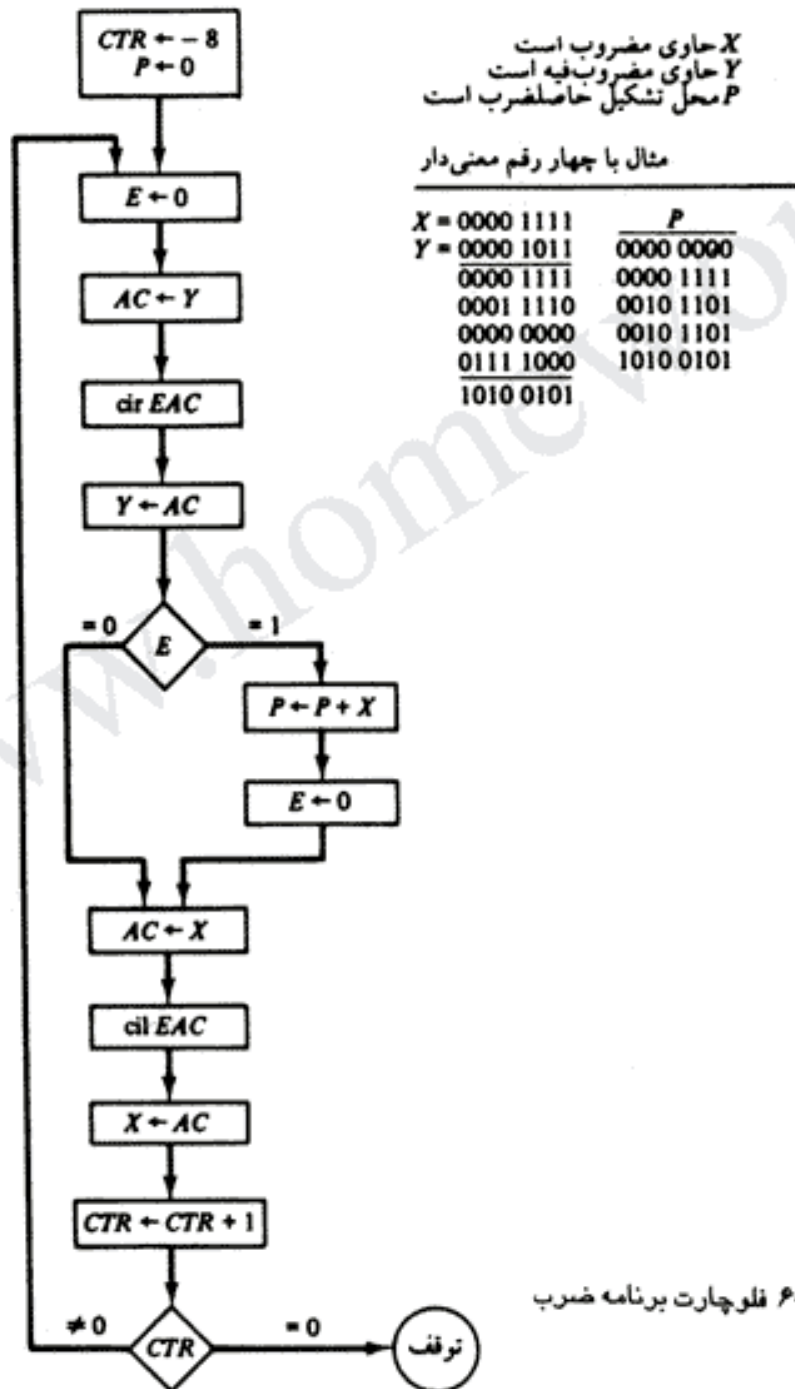
اعمالی که در یک کامپیوتر توسط یک دستورالعمل ماشین پیاده سازی شده باشند گوییم بطور سخت افزاری پیاده سازی شده اند. به اعمالی که بوسیله مجموعه ای از دستورالعمل ها پیاده سازی می شوند گوییم که بطور نرم افزاری پیاده سازی شده اند. برخی از کامپیوترها مجموعه گسترده ای از دستورات سخت افزاری را دارا هستند که برای بالا بردن سرعت انجام کارهای متداول طراحی شده اند. دسته ای دیگر دارای دستورالعمل های سخت افزاری کمتری هستند و بر پیاده سازی نرم افزاری اعمال متکی اند. پیاده سازی سخت افزاری بسیار پرهزینه است زیرا مدارهای اضافه تری برای پیاده سازی اعمال نیاز دارد. پیاده سازی نرم افزاری موجب ایجاد برنامه های طولانی هم از نظر تعداد دستورات و هم از نظر زمان اجرا می شود. این بخش پیاده سازی نرم افزاری چند عمل حسابی و منطقی را نشان می دهد. برنامه ها برای هر نوع عمل حسابی نه فقط برای داده های ممیز ثابت بلکه برای داده های دهدهی و ممیز شناور نوشته شده اند. پیاده سازی سخت افزاری عملیات حسابی در فصل ۱۰ آورده شده است.

برنامه ضرب

اینک برنامه ای را برای ضرب دو عدد می نویسیم. برای ساده کردن برنامه از بیت علامت صرف نظر کرده و فرض می کنیم اعداد مثبت باشند. همچنین فرض می کنیم که دو عدد دودویی بیش از هشت بیت ندارند، لذا حاصلضرب آنها از 16 بیت تجاوز نمی کند. می توان برنامه را اصلاح کرد تا علامت اعداد نیز در نظر گرفته شود و یا از اعداد 16 بیتی استفاده گردد. در این صورت حاصلضرب ممکن است تا 31 بیت طول داشته و دو کلمه از حافظه را اشغال نماید.

1- Index Register

برنامه ضرب دو عدد مبتنی بر روش ضرب اعداد با قلم و کاغذ است. همانطور که در مثال عددی شکل ۳-۶ نشان داده شده است، فرآیند ضرب از واریسی بیت‌های مضروب‌فیه Y و جمع مضروب X بتعداد 1 های موجود در Y تشکیل می‌شود، بشرطی که مقدار X از هر سطر به سطر بعدی یک واحد بسمت چپ جابجا شود. چون کامپیوتر در هر لحظه قادر است دو عدد را با هم جمع کند، ما خانه‌ای از



شکل ۳-۶ فلوجارت برنامه ضرب

حافظه را که با P نشان داده شده برای حفظ مجموع‌های میانی^۱ در نظر می‌گیریم. مجموع‌های میانی، حاصلضرب‌های جزئی خوانده می‌شوند زیرا حاوی بخشی از حاصلضرب هستند که تا آن زمان بدست آمده است. همانطور که در مثال عددی در شکل نشان داده شده است، حاصلضرب جزئی با صفر شروع می‌شود، مضروب، X ، به محتوای P ، بازا هر بیت از مضروب فیه Y که 1 است، اضافه می‌شود. پس از واری هر یک از بیت‌های مضروب فیه، مقدار X بسمت چپ شیفت داده می‌شود. مقدار نهایی حاصل در P حاصلضرب را تشکیل می‌دهد. وقتی که اعداد در هم ضرب شدند، حاصلضرب دارای چهار بیت معنی‌دار است. پس از عمل ضرب، حاصلضرب هشت بیت معنی‌دار خواهد داشت. کامپیوتر می‌تواند از اعداد با هشت بیت معنی‌دار استفاده کند و لذا حاصلضرب تا 16 بیت طول خواهد داشت.

فلوچارت شکل ۳-۶ رویه برنامه‌نویسی عمل ضرب را قدم به قدم نشان می‌دهد. برنامه دارای حلقه‌ای است که هشت بار طی می‌شود، یعنی هر بار بازا یک بیت معنی‌دار مضروب فیه. در ابتدا، مکان X حاوی مضروب و مکان Y حاوی مضروب فیه است. شمارنده CTR با 8- بار می‌شود و مکان P به 0 پاک می‌گردد. هر بیت مضروب فیه پس از انتقال به E واری می‌شود. این عمل با پاک کردن E ، بارکردن Y در AC ، چرخش E و AC به راست و ذخیره عدد شیفت یافته به Y انجام می‌گردد. بیت ذخیره شده در E کم‌ارزشترین بیت مضروب فیه است. اکنون مقدار E را واری می‌کنیم. اگر E برابر 1 باشد، مضروب، X ، با حاصلضرب جزئی P جمع می‌شود. ولی اگر E برابر 0 باشد حاصلضرب جزئی تغییری نمی‌کند. سپس مقدار X را با بارکردن آن در AC و چرخش E و AC ، یک واحد بچپ شیفت می‌دهیم. حلقه هشت بار، با افزایش CTR و رسیدن آن به صفر تکرار می‌گردد. وقتی که شمارنده به صفر برسد، برنامه از حلقه خارج می‌شود در حالی که نتیجه در مکان P قرار دارد.

برنامه جدول ۱۴-۶ دستورات ضرب دو عدد بدون علامت را نشان می‌دهد. مرحله مقداردهی اولیه^۲ آورده نشده است ولی بهنگام بارکردن برنامه در کامپیوتر باید انجام شود. مقداردهی اولیه شامل آوردن مضروب و مضروب فیه به ترتیب در خانه‌های X و Y ؛ دادن مقدار 8- به شمارنده؛ و دادن مقدار صفر به P می‌باشد. اگر این مکان‌ها مقداردهی نشوند، برنامه ممکن است با داده‌های غلط اجرا شود. برنامه، خود سر راست است و مراحل لیست شده در فلوچارت را دنبال می‌کند. توضیحات برنامه هم می‌تواند به دنبال کردن قدم به قدم رویه انجام کار کمک کند.

مثال فوق نشان داد که اگر کامپیوتری فاقد دستور ماشین برای کار مورد نظر می‌باشد، آن عمل را می‌توان با رشته‌ای از دستورات عمل‌ها برنامه‌نویسی کرد. بنابراین ما پیاده‌سازی نرم‌افزاری عمل ضرب را نشان دادیم. پیاده‌سازی سخت‌افزاری متناظر آن در بخش ۳-۱۰ آمده است.

جمع با دقت مضاعف

هنگامی که دو عدد 16 بیتی بی‌علامت در هم ضرب شوند، نتیجه حاصلضرب 32 بیتی خواهد بود که باید در

1- Intermediate Sum

2- Initialization

جدول ۱۴-۶ برنامه ضرب دو عدد مثبت

	ORG 100	
LOP,	CLE	پاک کردن E /
	LDA Y	بارکردن مضروب فیه /
	CIR	انتقال بیت مضروب فیه به E /
	STA Y	ذخیره مضروب فیه جایجا شده /
	SZE	چک بیت، صفر است یا نه /
	BUN ONE	بیت 1 است، به ONE برو /
	BUN ZRO	بیت 0 است، به ZRO برو /
ONE,	LDA X	بارکردن مضروب /
	ADD P	جمع با حاصلضرب جزئی /
	STA P	ذخیره حاصلضرب جزئی /
	CLE	پاک کردن E /
ZRO,	LDA X	بارکردن مضروب /
	CIL	چرخش به چپ /
	STA X	ذخیره مضروب جایجا شده /
	ISZ CTR	افزایش شمارنده /
	BUN LOP	شمارنده صفر نیست، تکرار حلقه /
	HLT	شمارنده صفر است؛ توقف /
CTR,	DEC -8	محل شمارنده /
X,	HEX 000F	محل ذخیره مضروب /
Y,	HEX 000B	محل ذخیره مضروب فیه /
P,	HEX 0	محل تشکیل حاصلضرب /
	END	

دو کلمه حافظه ذخیره شود. هرگاه عددی در دو کلمه حافظه جای گیرد گوییم دقت مضاعف دارد. وقتی که یک حاصلضرب جزئی محاسبه می شود، لازم است یک عدد با دقت مضاعف، با مضروب شیفت یافته که خود عددی با دقت مضاعف است، جمع شود. برای دقت بیشتر، برنامه نویس ممکن است بخواهد از اعداد با دقت مضاعف استفاده کند و محاسبه را با عملوندهایی که دو کلمه از حافظه را اشغال می کنند انجام دهد. در این بخش برنامه ای را ارائه می دهیم که دو عدد با دقت مضاعف را با هم جمع می کند. یکی از دو عدد با دقت مضاعف در دو حافظه متوالی، یعنی AL و AH، قرار داده می شود که AL حاوی 16 بیت کم ارزشتر است. عدد دیگر در BL و BH قرار داده می شود. برنامه در جدول ۱۵-۶ لیست شده است. دو بخش کم ارزشتر با هم جمع شده و رقم نقلی به E منتقل می شود. AC پاک شده و بیت موجود در E به داخل کم ارزشترین بیت AC چرخش داده می شود. سپس دو بخش پرارزشتر با نقلی جمع می شوند و مجموع با دقت مضاعف در CL و CH ذخیره می شود.

اعمال منطقی

کامپیوتر پایه سه دستورالعمل ماشین منطقی AND، CMA و CLA را داراست. دستورالعمل LDA را هم می توان یک دستورالعمل منطقی تصور کرد که یک عملوند منطقی را به AC منتقل می کند.

جدول ۱۵-۶ برنامه جمع دو عدد با دقت مضاعف

LDA AL	بار کردن بخش کم ارزشتر A
ADD BL	جمع بخش کم ارزشتر B، نقلی در E
STA CL	ذخیره در بخش کم ارزشتر C
CLA	پاک کردن AC
CIL	چرخش برای آوردن نقلی به درون AC(0)
ADD AH	جمع کردن بخش با ارزشتر A و نقلی
ADD BH	جمع کردن بخش پر ارزشتر B
STA CH	ذخیره در بخش پر ارزشتر C
HLT	
AL,	محل عملوندها
AH,	—
BL,	—
BH,	—
CL,	—
CH,	—

در بخش ۴-۵ فهرست ۱۶ عمل منطقی مختلف را ارائه نمودیم. هر شانزده عمل منطقی را می توان بصورت نرم افزاری پیاده سازی کرد زیرا هر تابع منطقی را می توان با AND و متمم سازی پیاده سازی نمود. مثلاً عمل OR در دستورات کامپیوتر پایه وجود ندارد. اما با استفاده از تئوری دموورگان رابطه $x + y = (x'y)'$ را می توان نتیجه گرفت. عبارت دوم فقط از اعمال AND و متمم سازی تشکیل شده است. برنامه ای که عمل OR را روی دو عملوند منطقی A و B انجام می دهد بصورت زیر است:

LDA A	Load first operand A	اولین عملوند در A را بارکن
CMA	Complement to get \bar{A}	با متمم سازی \bar{A} را بدست بیاور
STA TMP	Store in a temporary location	حاصل را در یک مکان موقت ذخیره کن
LDA B	Load Second operand B	عملوند دوم را در B بارکن
CMA	Complement to get \bar{B}	با متمم سازی \bar{B} را بدست بیاور
AND TMP	AND with \bar{A} to get $\bar{A} \wedge \bar{B}$	حاصل را \bar{A} با برای بدست آوردن $\bar{A} \wedge \bar{B}$ ، AND کن
CMA	Complement again to get $A \wedge B$	حاصل را مجدداً متمم کن تا AVB بدست آید

سایر اعمال منطقی را نیز می توان بطور مشابه با نرم افزار پیاده سازی کرد.

اعمال شیفت

اعمال شیفت چرخشی از جمله دستورات ماشین در کامپیوترها هستند. شیفت های مورد نظر دیگر عبارتند از شیفت های منطقی و شیفت های حسابی. این دو نوع شیفت را بسادگی با تعداد کمی از دستورات عمل ها می توان برنامه نویسی کرد.

شیفت منطقی مستلزم این است که رقم 0 به یکی از دو انتها اضافه شود. این بسادگی با پاک کردن E و

چرخش AC و E میسر است. بنابراین برای یک عمل منطقی شیفت به راست ما نیاز به دو دستورالعمل زیر داریم:

CLE

CIR

برای شیفت به چپ دو دستور زیر مورد نیاز است:

CLE

CIL

شیفت حسابی به نوع نمایش اعداد منفی بستگی دارد. برای کامپیوتر پایه، ما نمایش متمم 2 علامت دار را برگزیدیم. قواعد شیفت حسابی در بخش ۶-۴ لیست شده است. برای یک عمل شیفت به راست حسابی لازم است تا بیت علامت در سمت چپ ترین مکان بلا تغییر بماند. اما خود بیت علامت بداخل بیت مرتبه بالاتر عدد شیفت پیدا می کند. برنامه شیفت براساس حسابی مستلزم این است تا E را برابر مقدار بیت علامت کنیم و سپس چرخش به راست را انجام دهیم. بنابراین:

CLE

SPA

CME

CIR

E را با 0 پاک کن

اگر AC مثبت است گذر کن

AC منفی است، E برابر 1 شود

چرخش E و AC

برای شیفت به چپ حسابی لازم است تا بیت اضافه شده به منتهی الیه سمت راست 0 باشد. این کار بسادگی با صفر کردن E قبل از عمل چرخش به چپ انجام می شود. بیت علامت نباید در حین شیفت تغییر نماید. با یک دستور چرخش، بیت علامت بداخل E منتقل می شود. سپس لازم است تا بیت علامت را پس از عمل با E مقایسه کنیم. اگر دو مقدار برابر باشند، شیفت حسابی بطور صحیحی انجام گرفته است، ولی اگر برابر نباشند، سرریز رخ داده است. سرریز بدان معنی است که عدد قبل از شیفت بیش از حد بزرگ بوده است و هنگامی که در 2 ضرب شده (توسط عمل شیفت) عدد حاصل از ظرفیت AC تجاوز کرده است.

۶-۷ زیرروالها^۱

مکرراً اتفاق افتاده است که یک قطعه از کد در بخش های مختلفی از برنامه به طور تکراری نوشته شود. مسلماً بجای نوشتن کد در هر بار که لازم باشد بهتر است دستورات مشترک فقط یکبار نوشته شوند. مجموعه دستورالعمل های مشترکی که در یک برنامه بتوان از آنها بدفعات زیاد استفاده کرد زیر روال نام دارد. هر بار که زیر روال در بخش اصلی برنامه بکار رود، انشعابی به ابتدای زیرروال صورت

1- Subroutine

می‌گیرد. پس از اجرای زیرروال، انشعابی دیگر موجب بازگشت به برنامه اصلی می‌گردد. زیرروال از رشته مستقلی از دستورالعمل‌ها تشکیل می‌شود که کار معینی را انجام می‌دهند. از هر جای برنامه اصلی می‌توان به زیرروال انشعاب کرد. در اینجا این سؤال مطرح می‌شود که زیرروال چگونه می‌فهمد که به چه محلی بازگردد، زیرا ممکن است از مکان‌های متعددی در برنامه اصلی به زیرروال مزبور انشعاب گردد. بنابراین لازم است آدرس بازگشت را جایی در کامپیوتر ذخیره کنیم تا زیرروال بداند به کجا بازگردد. چون انشعاب به زیرروال و بازگشت به برنامه اصلی عمل متداولی است، تمام کامپیوترها دستورات خاصی را برای ورود به زیرروال و بازگشت از آن در اختیار می‌گذارند. در کامپیوتر پایه، رابط بین برنامه اصلی و یک زیرروال دستورالعمل BSA¹ می‌باشد. برای شرح چگونگی استفاده از این دستور، زیرروالی می‌نویسیم که محتوای یک انباره را چهار بار به چپ شیفت دهد. چهار بار شیفت یک کلمه برای پردازش اعداد دهدهی کد شده با دودویی، BCD، و یا کاراکترهای الفبا عددی عمل مفیدی است. چنین دستوری می‌توانست بعنوان دستورالعمل ماشین در کامپیوتر پیش‌بینی شود. اما چون این کار صورت نگرفته است، زیرروالی برای این کار ایجاد می‌شود. برنامه جدول ۱۶-۶ با بارکردن X در AC شروع می‌شود. دستورالعمل بعدی که با آن مواجه می‌شویم BSA

جدول ۱۶-۶ برنامه‌ای برای نمایش طریقه استفاده از زیرروالها

مکان			
		ORG 100	برنامه اصلی /
100		LDA X	باردهی X /
101		BSA SH4	انشعاب به زیرروال /
102		STA X	ذخیره عدد جابجا شده /
103		LDA Y	بارکردن Y /
104		BSA SH4	انشعاب دوباره به زیرروال /
105		STA Y	ذخیره عدد جابجا شده /
106		HLT	
107	X,	HEX 1234	
108	Y,	HEX 4321	
			زیرروال چهار بار جابجایی /
109	SH4,	HEX 0	عمل ذخیره آدرس بازگشت /
10A		CIL	یکت بار چرخش به چپ /
10B		CIL	
10C		CIL	
10D		CIL	چرخش به چپ برای چهارمین بار /
10E		AND MSK	صفر کردن (13-16) AC /
10F		BUN SH4 I	بازگشت به برنامه اصلی /
110	MSK,	HEX FFF0	عملوند پوشش /
		END	

1- Branch and Save Address

SH4 است. دستورالعمل BSA در مکان 101 است. زیر روال SH4 پس از اتمام کار خود باید به مکان 102 باز گردد. هنگامی که دستور BSA اجرا شود واحد کنترل آدرس بازگشت 102 را در مکانی که توسط آدرس سمبلیک SH4 تعریف می شود (که 109 است) ذخیره می کند. همچنین محتوای SH4+1 را بداخل شمارنده برنامه منتقل می سازد. پس از اجرای این دستورالعمل، مکان حافظه 109 حاوی دودویی معادل با شانزده شانزدهمی 102 می باشد و شمارنده برنامه حاوی دودویی معادل با شانزده شانزدهمی 10A خواهد بود. با این عمل آدرس بازگشت ذخیره شده است و اینک زیرروال با شروع از 10A اجرا می شود (زیرا این مقدار محتوای PC در برداشت سیکل بعدی است. محاسبات در زیر روال محتوای AC را چهار بار به چپ شیفت می دهد. برای انجام یک عمل شیفت منطقی چهار بیت پائین رتبه باید 0 شوند. این کار با ماسک کردن (پوشش دادن) FFF0 با محتوای AC انجام می شود. عمل ماسک، یک عمل AND منطقی است که هر جا بیت عملوند ماسک صفر باشند بیت متناظر در AC را پاک می کند و هر جا بیت های عملوند ماسک 1 باشند بیت های AC را بدون تغییر باقی می گذارند.

آخرین دستورالعمل، زیر روال کامپیوتر را به برنامه اصلی باز می گرداند. این عمل با دستورالعمل انشعاب غیرمستقیم و سمبل آدرس مورد استفاده برای نام زیر روال صورت می گیرد. آدرسی که کامپیوتر به آن انشعاب می کند SH4 نیست بلکه مقداری است که در مکان SH4 قرار دارد، زیرا دستورالعمل از نوع غیرمستقیم است. آنچه در SH4 یافت می شود آدرس بازگشت 102 می باشد که قبلاً در آنجا با دستورالعمل BSA ذخیره شد. کامپیوتر برای اجرای دستورالعمل واقع در مکان 102 باز می گردد. برنامه اصلی با ذخیره کردن عدد شیفت یافته در مکان X به کار خود ادامه می دهد. سپس عدد جدیدی از Y بداخل AC بار شده و انشعاب دیگری به زیرروال صورت می گیرد. این بار مکان SH4 حاوی آدرس برگشت 105 خواهد بود زیرا این مقدار مکان دستورالعمل بعدی پس از BSA است. عملوند جدید شیفت پیدا کرده و زیر روال به برنامه اصلی در مکان 105 باز می گردد.

از این مثال می بینیم که اولین مکان حافظه هر زیر روال نقش یک رابط بین برنامه اصلی و زیرروال را بعهدده دارد. رویه انشعاب به زیرروال و بازگشت به برنامه اصلی پیوند¹ زیر روال نامیده می شود. عملی را که دستورالعمل BSA انجام می دهد معمولاً فراخوانی² زیرروال می نامند. آخرین دستورالعمل زیرروال نیز عملی را که بازگشت³ از زیرروال می خوانند انجام می دهد.

روش بکار رفته در کامپیوتر پایه برای پیوند زیرروال عمدتاً در کامپیوترهایی یافت می شود که فقط یک ثبات پردازنده دارند. بسیاری از کامپیوترها دارای چندین ثبات پردازنده هستند و به برخی از آنها نام ثبات اندیس⁴ (شاخص) داده شده است. در چنین کامپیوترهایی، یک ثبات اندیس برای پیوند زیرروال بکار می رود. دستورالعمل انشعاب به زیر روال، آدرس برگشت را در ثبات اندیس ذخیره می کند. دستورالعمل بازگشت از زیر روال، با انشعاب به آدرسی که در آن موقع در ثبات شاخص است صورت می گیرد.

1- Linkage

2- Call

3- Return

4- Index Register

پارامترهای زیرروال و داده‌های پیوند

هنگامی که زیرروالی فراخوانده شود، برنامه اصلی باید داده‌هایی را که مایل است زیرروال با آنها کار کند به آن انتقال دهد. در مثال قبل، داده از طریق انباره منتقل شد. عملوند قبل از انشعاب در AC بار گردید. زیرروال عدد را شیفت داد و آن را در همان جا باقی گذاشت تا برنامه اصلی از آن استفاده کند. بطور کلی، لازم است تا زیرروال به داده‌های برنامه خواننده دسترسی داشته و نتایج را به آن بازگرداند. از انباره می‌توان برای یک پارامتر ورودی و یک پارامتر خروجی استفاده کرد. در کامپیوترهایی که ثبات‌های متعدد دارند تعداد بیشتری پارامتر بدین طریق قابل انتقال است. راه دیگری برای انتقال داده به یک زیرروال از طریق حافظه است. داده‌ها غالباً در مکان‌هایی از حافظه که بدنال فراخوانی آمده‌اند قرار داده می‌شوند. همچنین می‌توان آنها را در بلاک‌هایی از حافظه نیز ذخیره کرد. سپس اولین آدرس بلاک در مکان بعد از فراخوانی قرار می‌گیرد. در هر حال، آدرس بازگشت همواره اطلاعات ارتباطی برای انتقال داده بین برنامه اصلی و زیرروال را فراهم می‌آورد.

بعنوان مثال، زیرروالی را در نظر بگیرید که عمل منطقی OR را انجام دهد. دو عملوند باید به زیرروال منتقل و زیرروال هم باید نتیجه عمل را بازگرداند. از انباره می‌توان برای انتقال یک عملوند و دریافت نتیجه استفاده کرد. عملوند دیگر در مکانی که بدنال دستورالعمل BSA می‌آید وارد می‌شود. این مطلب در برنامه جدول ۱۷-۶ نشان داده شده است. اولین عملوند در مکان X بداخل AC بار می‌شود. عملوند دوم در مکان 202 که بدنال دستور BSA است ذخیره می‌شود. پس از انشعاب، اولین

جدول ۱۷-۶ برنامه‌ای برای نمایش اشتراک پارامترها

مکان			
		ORG 200	
200		LDA X	بارکردن عملوند اول در AC /
201		BSA OR	انشعاب به زیرروال OR /
202		HEX 3AF6	محل ذخیره عملوند دوم /
203		STA Y	محل برگشت از زیرروال /
204		HLT	
205	X,	HEX 7B95	محل ذخیره عملوند اول /
206	Y,	HEX 0	محل ذخیره نتیجه /
207	OR,	HEX 0	زیرروال OR /
208		CMA	متسم کردن عملوند اول /
209		STA TMP	ذخیره در مکان موقت /
20A		LDA OR I	بارکردن عملوند دوم /
20B		CMA	متسم کردن عملوند دوم /
20C		AND TMP	AND کردن متسم عملوند اول /
20D		CMA	متسم مجدد برای یافتن OR /
20E		ISZ OR	افزایش آدرس بازگشت /
20F		BUN OR I	بازگشت به برنامه اصلی /
210	TMP	HEX 0	محل ذخیره موقت /
		END	

مکان در زیر روال، عدد 202 را نگه میدارد. دقت کنید که در این حالت، 202 آدرس بازگشت نیست بلکه آدرس عملوند دوم است. زیر روال، عمل OR را با متمم کردن عملوند اول که در AC قرار دارد و ذخیره آن در یک مکان موقت به نام TMP آغاز می‌کند. دومین عملوند توسط دستورالعمل غیرمستقیم در مکان OR به AC بار می‌شود. بیاد بیاورید که مکان OR حاوی عدد 202 است. با مراجعه غیرمستقیم دستور به این مکان عملوند موجود در مکان 202 در AC بار می‌شود. این عملوند متمم شده و سپس با عملوند ذخیره شده در TMP، AND می‌شود. متمم کردن نتیجه، حاصل عمل OR را می‌دهد.

بازگشت از زیر روال باید بگونه‌ای صورت گیرد که برنامه اصلی از خانه 203، که دستورالعمل بعدی در آن قرار دارد ادامه یابد. این کار با افزایش آدرس OR توسط دستورالعمل ISZ صورت می‌گیرد. اکنون مکان OR عدد 203 را نگه میدارد و یک دستورالعمل BUN غیرمستقیم موجب بازگشت به محل صحیح می‌گردد. امکان دارد بیش از یک عملوند بدنبال دستورالعمل BSA داشت. زیر روال باید آدرس برگشت ذخیره شده در اولین خانه خود را، بازاء هر عملوندی که از برنامه فراخواننده استخراج می‌کند، افزایش دهد. به علاوه برنامه فراخواننده می‌تواند یک یا چند مکان را برای نتایج محاسبه شده در زیر روال حفظ کند. اولین مکان در زیر روال باید برای این مکان‌ها قبل از بازگشت افزایش داده شود. اگر حجم زیادی از داده‌ها برای انتقال وجود داشته باشد، آنها را می‌توان بصورت بلاک ذخیره کرده و آدرس اولین قلم بعنوان پارامتر پیوندی بکار گرفته شود.

زیر روالی که یک بلاک داده را از آدرس 100 به بلاک دیگری از آدرس 200 منتقل می‌کند در جدول ۱۸-۶ آورده شده است. طول بلاک 16 کلمه می‌باشد. اولین دستورالعمل، انشعاب به MVE است. اولین بخش زیر روال سه پارامتر 100، 200 و 16- را از برنامه اصلی منتقل و در مکان‌های مربوطه‌شان قرار می‌دهد. اقلام داده از بلاک‌های خود بکمک دو اشاره‌گر برداشته می‌شوند. شمارنده انتقال تنها 16 قلم را تضمین می‌کند. وقتی که زیر روال کار خود را تمام می‌کند، داده‌های مورد نظر در بلاکی قرار دارند که با مکان 200 شروع می‌شود. بازگشت برنامه به دستورالعمل HLT است.

۸-۶ برنامه نویسی ورودی - خروجی

کاربران کامپیوتر برنامه‌ها را با استفاده از سمبل‌هایی می‌نویسند که بوسیله زبان برنامه‌نویسی بکار رفته تعریف شده‌اند. سمبل‌ها رشته‌هایی از کاراکترها هستند و به هر کاراکتر یک کد 8 بیت اختصاص می‌یابد بطوری که می‌توان آنرا در حافظه کامپیوتر ذخیره کرد. یک کاراکتر کد شده به دودویی هنگامی وارد کامپیوتر می‌شود که یک دستورالعمل INP^۱ (ورودی) اجرا شود. یک کاراکتر کد شده به دودویی وقتی به دستگاه خروجی^۲ منتقل می‌شود که یک دستورالعمل OUT^۳ اجرا گردد. وسیله خروجی کد دودویی را تشخیص داده و کاراکتر مربوط به آن را تایپ می‌کند.

جدول ۱۹-۶ (الف) دستورات لازم برای وارد کردن یک کاراکتر و ذخیره آن را در حافظه نشان

1- Input

2- Output Device

3- Output

جدول ۱۸-۶ زیرروال انتقال بلاک دادهها

		برنامه اصلی /
	BSA MVE	انشعاب به زیرروال /
	HEX 100	آدرس شروع مبدأ دادهها /
	HEX 200	آدرس شروع مقصد دادهها /
	DEC -16	تعداد اقلامی که باید انتقال یابد /
	HLT	
MVE,	HEX 0	زیرروال MVE /
	LDA MVE I	بارکردن آدرس مبدأ /
	STA PT1	ذخیره در اشاره گر اول /
	ISZ MVE	افزایش آدرس بازگشت /
	LDA MVE I	بارکردن آدرس مقصد /
	STA PT2	ذخیره در اشاره گر دوم /
	ISZ MVE	افزایش آدرس برگشت /
	LDA MVE I	بارکردن تعداد اقلام /
	STA CTR	ذخیره در شمارنده /
	ISZ MVE	افزایش آدرس برگشت /
LOP,	LDA PT1 I	بارکردن یک فم از دادههای مقصد /
	STA PT2 I	ذخیره در مقصد /
	ISZ PT1	افزایش اشاره گر مبدأ /
	ISZ PT2	افزایش اشاره گر مقصد /
	ISZ CTR	افزایش شمارنده /
	BUN LOP	16 بار تکرار /
	BUN MVE I	بازگشت به برنامه اصلی /
PT1,	—	
PT2,	—	
CTR,	—	

جدول ۱۹-۶ برنامه ورود و خروج یک کاراکتر

(الف) ورودی یک کاراکتر:		
CIF,	SKI	چک کردن پرچم ورودی /
	BUN CIF	پرچم برابر 0 است، انشعاب برای واری مجدد /
	INP	پرچم 1 است، دریافت کاراکتر /
	OUT	چاپ کاراکتر /
	STA CHR	ذخیره کاراکتر /
	HLT	
CHR,	—	محل ذخیره کاراکتر /
(ب) خروجی یک کاراکتر:		
	LDA CHR	بارکردن کاراکتر در AC /
COF,	SKO	چک کردن پرچم خروجی /
	BUN COF	پرچم 0 است، انشعاب برای واری مجدد /
	OUT	پرچم برابر 1 است، ارسال کاراکتر /
	HLT	
CHR,	HEX 0057	کاراکتر مورد نظر "W" است /

می دهد. دستورالعمل SKI پرچم¹ ورودی را چک می کند تا وجود یک کاراکتر را برای انتقال دریابد. اگر پرچم ورودی 1 باشد دستور بعدی گذر می شود. دستورالعمل INP کاراکتر کد شده به دودویی را به AC (0-7) می فرستد. سپس کاراکتر توسط دستورالعمل OUT چاپ می شود. واحد پایانه ای² (ترمینال) که مستقیماً به کامپیوتر متصل است، با فشار دادن کلید روی صفحه کلید، کاراکتری را چاپ نمی کند. برای تایپ آن، لازم است دستورالعمل OUT برای چاپگر ارسال شود. باین ترتیب، کاربر از یک انتقال صحیح مطمئن خواهد شد. اگر دستورالعمل SKI بیت پرچم را برابر 0 بیابد، دستورالعمل متوالی بعدی را اجرا خواهد کرد. این دستورالعمل، انشعاب برای بازگشت و واریسی مجدد بیت پرچم است. چون دستگاه (وسيله) ورودی، خیلی آهسته تر از کامپیوتر است، دو دستورالعمل حلقه، قبل از انتقال یک کاراکتر به داخل انباره، چند بار اجرا خواهند شد.

جدول ۱۹-۶ (ب) لیست دستورات لازم برای چاپ یک کاراکتر را که قبلاً در حافظه ذخیره شده نشان می دهد. ابتدا کاراکتر در AC بار می شود. سپس پرچم خروجی چک می گردد. اگر پرچم 0 باشد، کامپیوتر در حلقه متشکل از دو دستور باقی می ماند و بیت پرچم را چک می نماید. وقتی که پرچم به 1 تغییر یابد، کاراکتر از انباره به چاپگر منتقل می گردد.

دستکاری کاراکتر

یک کامپیوتر صرفاً یک ماشین حساب نیست بلکه دستکاری کننده سمبل ها نیز هست. کاراکترهای کد شده به دودویی که سمبل ها را می سازند می توانند بوسیله دستورالعمل های کامپیوتر برای انجام انواع کارهای داده پردازشی دستکاری شوند. یکی از این کارها فشردن دو کاراکتر در یک کلمه است. این کار ساده است زیرا هر کاراکتر 8 بیت را اشغال می کند و یک کلمه حافظه از 16 بیت تشکیل شده است. برنامه لیست شده در جدول ۲۰-۶ عبارتست از زیرروالی بنام IN2 که دو کاراکتر را وارد کرده و آنها را در یک کلمه فشرده می سازد. کلمه فشرده شده در انباره باقی می ماند. توجه کنید که زیر روال SH4 (جدول ۱۶-۶) برای هشت بار شیفت به چپ انباره دوبار فراخوانی شود.

در بحث اسمبلر فرض شد که برنامه سمبلیک در بخشی از حافظه که گاهی بافر³ نامیده می شود، ذخیره می گردد. برنامه سمبلیک تایپ شده از طریق وسایل ورودی داخل شده و در مکان های متوالی حافظه در بافر ذخیره می گردد. برنامه لیست شده در جدول ۲۱-۶ می تواند برای ورود برنامه سمبلیک از صفحه کلید و فشردن دو کاراکتر در یک کلمه و ذخیره آن در یک بافر بکار رود. اولین آدرس بافر 500 است. اولین جفت کاراکتر در مکان 500 ذخیره می شود و سایر کاراکترها در مکان های متوالی ذخیره می شوند. برنامه از اشاره گر برای دنبال کردن مکان های خالی موجود در بافر استفاده می کند. در برنامه از شمارنده استفاده نشده است، بنابراین مادامی که کاراکترها موجود باشند و یا تا رسیدن بافر به 0 (پس از FFFF) خوانده می شوند. در شرایط عملی ممکن است محدود کردن ظرفیت بافر لازم بنظر برسد و از یک

1- Flag

2- Terminal

3- Buffer

جدول ۲۰-۶ زیرروالی برای دریافت و فشرده کردن دو کاراکتر

IN2, FST,	— SKI BUN FST INP OUT	محل ورود به زیرروال / دریافت اولین کاراکتر /
SCD,	BSA SH4 BSA SH4 SKI BUN SCD INP OUT BUN IN2 I	چهار بار شیفت به چپ / چهار بار شیفت به چپ / دریافت دومین کاراکتر / بازگشت /

جدول ۲۱-۶ برنامه ذخیره کاراکترهای ورودی در یک بافر

LOP,	LDA ADS STA PTR BSA IN2 STA PTR I ISZ PTR BUN LOP HLT	بار کردن آدرس ابتدای بافر / آغاز اشاره گر / پرش به زیرروال IN2 (جدول ۶-۲۰) / ذخیره کلمه دو کاراکتری در بافر / افزایش اشاره گر / انشعاب برای دریافت کاراکترهای دیگر /
ADS, PTR,	HEX 500 HEX 0	آدرس ابتدای بافر / محل اشاره گر /

شمارنده برای این منظور استفاده گردد. توجه داشته باشید که زیر روال IN2 جدول ۲۰-۶ برای وارد کردن و فشرده کردن جفت کاراکترها فراخوانی می شود.

در بحث مربوط به مرور دوم اسمبلر در بخش ۴-۶، گفته شد که یکی از متداول ترین عملیات یک اسمبلر جستجوی جدول^۱ است. این عملی است که طی آن یک جدول برای یافتن یک سمبل مورد نظر جستجو می شود. جستجو ممکن است، مقایسه سمبل مفروض با هر یک از سمبل های ذخیره شده در جدول باشد. وقتی که تطابقی رخ دهد و یا اینکه هیچ یک از سمبل ها با آن مطابقت نداشته باشند جستجو خاتمه می یابد. اگر تطابق اتفاق بیفتد، اسمبلر کد دودویی معادل را استخراج می کند. برای مقایسه دو کلمه برنامه ای در جدول ۲۲-۶ ارائه شده است. عمل مقایسه بدین ترتیب صورت می گیرد که متمم 2 یک کلمه بدست آمده و با کلمه دوم جمع حسابی می شود. اگر نتیجه صفر باشد، دو کلمه برابرند و تطابق رخ داده است. اگر نتیجه صفر نباشد، کلمات یکسان نیستند. این برنامه را می توان بصورت زیر روالی در یک برنامه جستجوی جدول مورد استفاده قرار داد.

1- Table look up

جدول ۲۲-۶ برنامه مقایسه دو کلمه

LDA WD1	بارکردن کلمه اول /
CMA	
INC	تشکیل متمم 2 /
ADD WD2	اضافه کردن کلمه دوم /
SZA	گذر اگر AC صفر باشد /
BUN UEQ	انشعاب به روال ۳ اگر نامساوی * /
BUN EQL	انشعاب به روال ۳ اگر مساوی * /
WD1,	—
WD2,	—

برنامه وقفه^۱

زمان اجرای برنامه‌های ورودی و خروجی-عمدتاً مربوط به زمانی است که کامپیوتر با صرف آن، منتظر 1 شدن پرچم وسیله خارجی می‌ماند. این اتلاف زمانی ناشی از حلقه انتظاری است که عمل چک کردن پرچم را انجام می‌دهد و لذا کامپیوتر را به کاری که جز اتلاف وقت زیادی نیست مشغول نگه‌میدارد. این زمان انتظار می‌تواند حذف شود بشرطی که برای اطلاع دادن 1 شدن پرچم از قابلیت وقفه استفاده شود. مزیت استفاده از وقفه این است که انتقال اطلاعات با درخواست از سوی وسیله خارجی آغاز می‌شود. در این احوال کامپیوتر می‌تواند مشغول اجرای کارهای مفید دیگری باشد. واضح است که اگر برنامه دیگری در حافظه مقیم نشده باشد کاری که کامپیوتر انجام دهد وجود نخواهد داشت، بنابراین می‌تواند همان چک کردن پرچم‌ها را انجام دهد. قابلیت وقفه در محیط‌های چند برنامه‌ای که دو یا چند برنامه در حافظه مقیم‌اند مفید است.

در هر لحظه از زمان، حتی اگر چند برنامه در حافظه مقیم باشد، تنها یک برنامه می‌تواند اجرا شود. برنامه‌ای که در حال حاضر در حال اجرا شدن است، برنامه جاری خوانده می‌شود. سایر برنامه‌ها معمولاً منتظر داده‌های ورودی یا خروجی هستند. کار وقفه این است که در حین اجرای برنامه، به انتقال داده‌های برنامه دیگر پردازد. برنامه جاری باید شامل یک دستور ION برای فعال کردن وقفه‌ها باشد. اگر قابلیت وقفه بکار گرفته نشود، برنامه باید دارای دستور IOF باشد تا وقفه را از کار بیندازد. (کلید start کامپیوتر نیز وقفه را از کار می‌اندازد).

قابلیت وقفه اجازه می‌دهد تا برنامه جاری به کار خود ادامه دهد تا اینکه وسیله داخلی یا خارجی پرچم خود را 1 نماید. هر وقت پرچمی 1 شد، کامپیوتر اجرای دستور جاری را تمام کرده و به وقفه پاسخ^۲ می‌دهد. نتیجه این عمل این است که آدرس بازگشت در مکان 0 ذخیره می‌گردد. سپس دستورالعمل در مکان 1 اجرا می‌شود؛ باین ترتیب یک روال سرویس‌دهی برای انتقال ورودی یا خروجی آغاز می‌گردد. روال سرویس‌دهی در هر جا از حافظه می‌تواند ذخیره شود بشرطی که

1- Interrupt

2- Acknowledge

دستورالعملی برای انشعاب به ابتدای روال در خانه 1 ذخیره شود. روال سرویس دهی بایستی دستوراتی برای انجام کارهای زیر داشته باشد.

۱- ذخیره محتویات ثبات های پردازنده

۲- واریسی اینکه کدام پرچم 1 است

۳- سرویس دهی به وسیله ای که پرچمش 1 است

۴- بازگرداندن محتوای ثبات های پردازنده

۵- فعال کردن قابلیت وقفه

۶- بازگشت به برنامه جاری

محتوای ثبات های پردازنده قبل از وقفه و بعد از بازگشت به برنامه جاری باید یکسان باشند؛ در غیر اینصورت برنامه جاری ممکن است در وضعیت خطا کار کند. چون روال سرویس دهی ممکن است این ثبات ها را بکار گیرد، لازم است تا محتوای آنها را در ابتدای روال ذخیره کرده و در انتها نیز بازگردانده شوند. ترتیب واریسی پرچم ها اولویت هر وسیله را تعیین می کند. اگرچه ممکن است دو یا چند پرچم بطور همزمان نشانده¹ شوند، ولی در هر زمان به یک وسیله سرویس داده می شود. ترتیب سرویس دهی بر اساس اولویت وسایل است.

وقوع یک وقفه مانع وقوع وقفه های دیگر می گردد. روال سرویس باید قبل از بازگشت به برنامه جاری وقفه ها را فعال کند. در نتیجه کامپیوتر حین اجرای برنامه جاری قادر خواهد بود وقفه های بعدی را بپذیرد. قابلیت وقفه نباید تا زمانی که آدرس بازگشت در شمارنده برنامه قرار نگرفته است فعال شود.

مثالی از یک برنامه که به یک وقفه سرویس می دهد در جدول ۲۳-۶ ارائه شده است. مکان 0 برای آدرس بازگشت نگه داشته شده است. مکان 1 حاوی دستورالعمل انشعاب به ابتدای روال سرویس SRV است. بخشی از برنامه جاری که نشان داده شده حاوی یک دستورالعمل ION است که وقفه ها را فعال می کند. فرض کنید در حالی که کامپیوتر دستورالعمل واقع در مکان 103 را اجرا می کند وقفه ای رخ دهد. سیکل وقفه معادل دودویی عدد شانزده شانزدهمی 104 را در مکان 0 ذخیره کرده و به مکان 1 انشعاب می نماید. دستورالعمل انشعاب در مکان 1، کامپیوتر را به روال SRV می فرستد.

روال سرویس دهی شش کار ذکر شده فوق را اجرا می نماید. محتوای AC و E در مکان های خاصی ذخیره می شوند. (این مکان ها در کامپیوتر پایه فقط ثبات های پردازشگر هستند. پرچم ها متوالیاً واریسی می شوند مثلاً ابتدا پرچم ورودی و سپس پرچم خروجی. اگر یکی یا هر دو پرچم 1 شوند (نشانده شوند) یک قلم از داده به و یا از حافظه مربوطه منتقل می شود. قبل از بازگشت به برنامه جاری محتوای قبلی E و AC بازیابی شده و قابلیت وقفه فعال می گردد. آخرین دستور، انشعابی را به آدرس ذخیره شده در مکان 0 موجب می شود. این آدرسی است که قبلاً در حین وقفه ذخیره شده است. از اینرو برنامه جاری

1- Set

جدول ۲۳-۶ برنامه سرویس‌دهی به یک وقفه

مکان			
0	ZRO,	—	محل ذخیره آدرس برگشت /
1		BUN SRV	شعبه به روال سرویس /
100		CLA	قسمتی از برنامه در حال اجرا /
101		ION	فعال کردن قابلیت وقفه /
102		LDA X	
103		ADD Y	شروع وقفه /
104		STA Z	برنامه بعد از وقفه به این مکان بازمی‌گردد /
.		.	
.		.	
.		.	
200	SRV,	STA SAC	روال سرویس‌دهی به وقفه /
		CIR	ذخیره محتوای AC /
		STA SE	انتقال E به AC(1) /
		SKI	ذخیره محتوای E /
		BUN NXT	چک کردن پرچم ورودی /
		INP	پرچم صفر است، پرچم بعدی چک شود /
		OUT	پرچم 1 است، دریافت کاراکتر /
		STA PT1 I	چاپ کاراکتر /
		ISZ PT1	ذخیره کاراکتر در بافر ورودی /
	NXT,	SKO	افزایش اشاره‌گر ورودی /
		BUN EXT	چک کردن پرچم خروجی /
		LDA PT2 I	پرچم صفر است، خروج /
		OUT	بار کردن کاراکتر از بافر خروجی /
		ISZ PT2	ارسال کاراکتر /
	EXT,	LDA SE	افزایش اشاره‌گر خروجی /
		CIL	بازیابی مقدار AC(1) /
		LDA SAC	انتقال آن به E /
		ION	بازیابی محتوای AC /
		BUN ZRO I	فعال کردن وقفه /
	SAC,	—	بازگشت به برنامه در حال اجرا /
	SE,	—	محل ذخیره AC /
	PT1,	—	محل ذخیره E /
	PT2,	—	اشاره‌گر میانگیر ورودی /
			اشاره‌گر میانگیر خروجی /

از مکان 104 به کار ادامه خواهد داد، یعنی به همان جایی که در آن وقفه رخ داده بود. در یک نمونه کامپیوتر امکان دارد چندین وسیله ورودی یا خروجی به مدار وقفه متصل شده باشد. بعلاوه، منابع وقفه محدود به وسایل ورودی و خروجی محدود نیستند. وقفه‌ها را می‌توان برای اهداف دیگری همچون خطاهای پردازشی درونی یا اخطارهای ویژه بکار برد. بحث‌های بیشتر درباره وقفه‌ها و برخی مفاهیم پیشرفته که اهمیت موضوع را بیان کنند در بخش ۵-۱۱ آورده شده است.

مسائل

۶-۱ برنامه زیر در واحد حافظه کامپیوتر پایه ذخیره شده است. محتوای AC، PC و IR را به شانزده شانزدهمی در پایان اجرای هر دستورالعمل نشان دهید. تمام اعداد در جدول به شانزدهمی هستند.

مکان	دستورالعمل
010	CLA
011	ADD 016
012	BUN 014
013	HLT
014	AND 017
015	BUN 013
016	C1A5
017	93C6

۶-۲ برنامه زیر لیستی از دستورالعملها در مبنای شانزده است. کامپیوتر با شروع از آدرس 100 دستورالعملها را اجرا می کند. محتوای AC و کلمه حافظه درون آدرس 103 بهنگام توقف کامپیوتر چیست؟

مکان	دستورالعمل
100	5103
101	7200
102	7001
103	0000
104	7800
105	7020
106	C103

۶-۳ لیست برنامه زبان اسمبلی تولید شده بوسیله یک کامپایلر را از برنامه فرتن زیر بنویسید. متغیرها را عدد صحیح تصور کنید.

$$\text{SUM} = 0$$

$$\text{SUM} = \text{SUM} + \text{A} + \text{B}$$

$$\text{DIF} = \text{DIF} - \text{C}$$

$$\text{SUM} = \text{SUM} + \text{DIF}$$

۶-۴ آیا از حرف I می توان بعنوان یک آدرس سمبلیک در برنامه زبان اسمبلی برای کامپیوتر پایه استفاده کرد؟ توضیح دهید.

۶-۵ اگر سطری از کد که حاوی شبه دستورالعمل ORG یا END است عنوان نیز داشته باشد در مرور اول اسمبلر (شکل ۶-۱) چه رخ خواهد داد. فلوچارت را تغییر دهید تا در این صورت پیغام خطا بدهد.

۶-۶ سطری از کد در یک برنامه زبان اسمبلی بصورت زیر است:

DEC - 35

(الف) نشان دهید که برای ذخیره سطر کد به چهار کلمه حافظه نیاز است و محتوای دودویی آنها را بنویسید.

(ب) نشان دهید که یک کلمه حافظه کد ترجمه شده دودویی را ذخیره می نماید. محتوای دودویی آنها را بنویسید.

۶-۷ الف) جدول سمبل آدرس تولید شده برای برنامه جدول ۱۳-۶ در حین مرور اول بوسیله اسمبلر را بدست آورید.

(ب) برنامه ترجمه شده را به شانزده شانزده بنویسید.

۶-۸ شبه دستورالعمل BSS N (بلاک آغاز شده بوسیله سمبل) گاهی برای رزرو N کلمه حافظه برای گروهی از عملوندها به کار می رود. مثلاً سطر کد

A, BSS 10

به اسمبلر اطلاع می دهد که بلاکی از 10 (دهدهی) مکان را، با شروع از مکان متناظر با سمبل A خالی بگذارد. این مشابه جمله DIMENSION A(10) در فورترن است. فلوجارت شکل ۱-۶ را برای پردازش این شبه دستور تصحیح کنید.

۶-۹ فلوجارت شکل ۲-۶ را تصحیح کنید بطوری که اگر یک دستور سمبلیک با عنوان تعریف نشده باشد یک پیام خطا تولید شود.

۶-۱۰ نشان دهید که چگونه جداول MRI و non-MRI می توانند در حافظه ذخیره شوند.

۶-۱۱ لیست برنامه زبان اسمبلی (معادل دودویی دستورالعملها) تولید شده بوسیله کامپایلر را برای عبارت زیر بدست آورید:

IF(A - B) 10, 20, 30

برنامه به عبارت 10 انشعاب می کند اگر $A - B < 0$ باشد؛ یا به 20 اگر $A - B = 0$ باشد؛ و یا به 30 اگر $A - B > 0$ باشد.

۶-۱۲ الف) توضیح دهید که اجرای برنامه زیر موجب انجام چه کاری می شود. مقدار مکان CTR وقتی که کامپیوتر متوقف می شود چیست؟

ORG 100
CLE
CLA
STA CTR
LDA WRD
SZA
BUN ROT
BUN STP

```

ROT,   CIL
        SZE
        BUN AGN
        BUN ROT
AGN,   CLE
        ISZ CTR
        SZA
        BUN ROT
STP,   HLT
CTR,   HEX 0
WRD,   HEX 62C1

        END

```

ب) جدول سمبل های آدرس بدست آمده در مرور اول اسمبلر را بنویسید.

ج) کد شانزده شانزدهمی برنامه ترجمه شده را بنویسید.

۶-۱۳ یک حلقه برنامه با استفاده از اشاره گر و شمارنده بنویسید تا محتویات مکان شانزده شانزدهمی 500 تا 5FF را 0 کند.

۶-۱۴ برنامه ای بنویسید تا دو عدد را با روش جمع تکراری، در هم ضرب کند. مثلاً، برای ضرب 4×5 برنامه 5 را چهار بار با هم جمع کند، یعنی $5 + 5 + 5 + 5$.

۶-۱۵ برنامه ضرب جدول ۶-۱۴ مقدار دهی اولیه نشده است. پس از یکبار اجرای برنامه، مکان CTR مقدار صفر می گیرد. نشان دهید که اگر برنامه مجدداً از مکان 100 شروع شود، حلقه 65536 بار تکرار خواهد شد. دستورات لازم برای مقداردهی اولیه^۱ را برای برنامه اضافه کنید.

۶-۱۶ برنامه ای برای ضرب دو عدد مثبت بدون علامت 16 بیتی بنویسید و حاصلضرب آنها را بصورت یک عدد بدون علامت با دقت مضاعف به دست آورید.

۶-۱۷ برنامه ای برای ضرب دو عدد علامت دار بنویسید. اعداد منفی ابتدا بصورت متمم 2 علامت دار نشان داده می شوند. حاصلضرب باید با دقت منفرد باشد و در صورت منفی بودن هم بصورت متمم 2 علامت دار نشان داده شود.

۶-۱۸ برنامه ای بنویسید که دو عدد با دقت مضاعف را از هم تفریق کند.

۶-۱۹ برنامه ای بنویسید که OR انحصاری منطقی دو عملوند منطقی را بدست آورد.

۶-۲۰ برنامه ای برای شیفت حسابی به چپ بنویسید که در صورت سرریز به OVF انشعاب کند.

۶-۲۱ برنامه ای برای تفریق دو عدد بنویسید. در برنامه فراخواننده، مفروق و مفروق منه بدنبال دستورالعمل BSA آمده اند. تفاضل به سومین مکان پس از BSA در برنامه اصلی بازمی گردد.

۶-۲۲ برنامه ای بنویسید که هر داده را در یک بلاک متمم کند. در برنامه فراخواننده، دو پارامتر

1- Initialize

- آدرس شروع بلاک و تعداد کلمات بلاک پس از دستورالعمل BSA آمده‌اند.
- ۶-۲۳ برنامه‌ای بنویسید که E و AC را چهار بار به راست بچرخاند. اگر ابتدا محتوای AC برابر $E = 1$ (079C)₁₆ باشد محتوای آنها پس از اجرای زیر روال چیست؟
- ۶-۲۴ برنامه‌ای بنویسید که کاراکترهای ورودی را بپذیرد، هر دو کاراکتر بصورت یک کلمه فشرده کند و متوالیاً در بافر حافظه ذخیره نماید. اولین آدرس بافر $16(400)$ و اندازه آن $10(512)$ است. اگر بافر سرریز کند کامپیوتر متوقف شود.
- ۶-۲۵ برنامه‌ای بنویسید که دو کاراکتر واقع در مکان WRD را بردارد و از حالت فشرده خارج کند و آنها را در بیت‌های 0 تا 7 از مکان‌های CH1 و CH2 ذخیره کند. بیت‌های 9 تا 15 باید صفر باشد.
- ۶-۲۶ فلوجارتی را برای برنامه‌ای بدست آورید که کد CR (0D در مبنای شانزده) را در بافر حافظه جستجو کند. بافر دارای ۲ کد در هر کلمه است. وقتی به کد CR برخورد شود، برنامه آن را به بیت‌های 0 تا 7 مکان LNE منتقل می‌سازد بدون آن که بیت‌های 8 تا 15 تغییر کنند.
- ۶-۲۷ روال سرویس‌دهی SRV را از جدول ۶-۲۳ به کد شانزده شانزده می‌معادلش ترجمه کنید. فرض کنید که روال از مکان 200 شروع شده باشد.
- ۶-۲۸ یک روال سرویس‌دهی وقفه که تمام عملیات لازم را انجام دهد بنویسید ولی وسیله ورودی فقط اگر مکان MOD تماماً 1 باشد سرویس‌دهی شود. وسیله خروجی هم فقط اگر مکان MOD تماماً 0 باشد سرویس‌دهی گردد.



کنترل ریز برنامه نویسی شده

- ۷-۱ حافظه کنترل
- ۷-۲ دنبال کردن آدرس
- ۷-۳ مثال ریز برنامه
- ۷-۴ طراحی واحد کنترل

۷-۱ حافظه کنترل

وظیفه یک واحد کنترل در یک کامپیوتر دیجیتال بکارگیری دنباله‌هایی از ریز عمل‌هاست. تعداد انواع ریز عمل‌های موجود در یک سیستم معین محدود است. پیچیدگی سیستم دیجیتال به تعداد ریز عمل‌های متوالی که اجرا می‌کند بستگی دارد. هنگامی که سیگنال‌های کنترلی توسط سخت‌افزار حاصل از تکنیک‌های مدارهای منطقی رایج تولید شود، گوئیم واحد کنترل سخت‌افزاری^۱ است. ریز برنامه‌نویسی روش دیگری برای طراحی واحد کنترل یک کامپیوتر دیجیتال است. اصل ریز برنامه‌نویسی روشی شاخص و سیستماتیک در کنترل رشته ریز عمل‌ها در یک کامپیوتر دیجیتال است. تابع کنترلی که مشخص کننده یک ریز عمل است یک متغیر دودویی می‌باشد. وقتی که این متغیر در یکی از حالات دودویی باشد، ریز عمل متناظر آن اجرا می‌شود. اگر متغیر کنترلی که در وضعیت دودویی مخالف باشد وضعیت ثبات‌های سیستم را تغییر نمی‌دهد. وضعیت فعال یک متغیر کنترل ممکن است بسته به کاربرد هر یک از دو حالت 1 یا 0 باشد. در سیستم مبتنی بر گذرگاه^۲، سیگنال‌های کنترلی که ریز عمل را مشخص می‌نمایند بصورت گروه‌هایی از بیت‌ها هستند که مسیرها را در مولتی‌پلکسرها، دیکدرها و واحدهای حساب و منطق انتخاب می‌نمایند.

واحد کنترل موجب اجرای یک سری مراحل متوالی از ریز عمل‌ها می‌شود. در هر زمان مفروض،

1- Hardware = Hardwired 2- Bus Organized System

ریزعمل های معینی باید اجرا شوند، در حالیکه سایر ریزعمل ها را کد هستند. متغیرهای کنترلی در هر لحظه از زمان می توانند با رشته ای از 1 ها و 0 ها که کلمه کنترل نامیده می شود نمایش داده شوند. باین ترتیب می توان با برنامه نویسی کلمات کنترلی، عملیات مختلف را روی اجزاء سیستم انجام داد. واحد کنترلی که متغیرهای دودویی کنترل آن در حافظه ذخیره شده باشد واحد کنترل ریزبرنامه نویسی شده نام دارد. هر کلمه در واحد کنترل حاوی یک ریز دستورالعمل است. ریز دستورالعمل مشخص کننده یک یا چند ریزعمل برای سیستم است. رشته ای از ریزدستورالعمل ها، ریزبرنامه را تشکیل می دهند. با توجه به اینکه پس از نصب واحد کنترل در سیستم لازم نیست تغییری در ریزبرنامه داده شود، حافظه کنترل می تواند از نوع فقط خواندنی¹ (ROM) باشد. محتوای کلمات ROM ثابت است و آنها را بسادگی نمی توان با برنامه نویسی تغییر داد زیرا امکان نوشتن در ROM وجود ندارد. کلمات ROM حین ساخت واحد سخت افزار بطور دائمی ایجاد می شوند. بکارگیری یک ریزبرنامه بمعنی در اختیار قرار دادن تمام متغیرهای کنترل در کلمات ROM برای استفاده واحد کنترل از طریق خواندن های متوالی است. محتوای کلمه در ROM در یک آدرس مشخص یک ریزدستورالعمل را مشخص می کند.

در روش پیشرفته تری که برنامه نویسی دینامیکی نام دارد این امکان وجود دارد که ریزبرنامه ابتدا از طریق یک حافظه کمکی مانند دیسک مغناطیسی به سیستم بار شود. واحدهای کنترلی که از برنامه نویسی دینامیکی استفاده می کنند دارای حافظه هایی با قابلیت نوشتن هستند. این نوع حافظه را می توان برای نوشتن (در تغییر برنامه) بکار برد، ولی اکثراً برای خواندن مورد استفاده قرار می گیرند. حافظه ای که بخشی از واحد کنترل باشد حافظه کنترل² نامیده می شود.

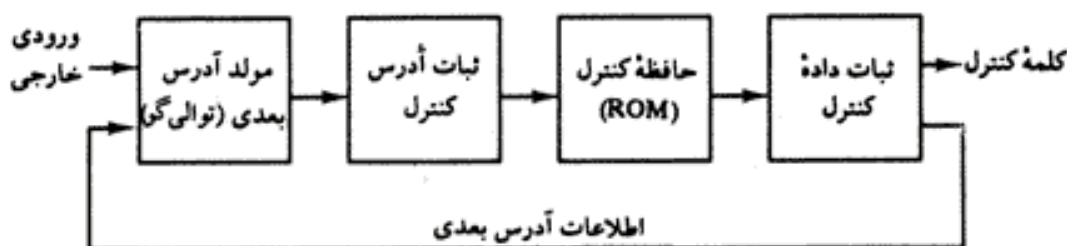
کامپیوتری که از واحد کنترل ریزبرنامه نویسی شده استفاده می کند دو حافظه مجزا خواهد داشت: یک حافظه اصلی و یک حافظه کنترل. حافظه اصلی در اختیار کاربر است تا برنامه های خود را در آن ذخیره کند. محتوای حافظه اصلی بهنگام دستکاری داده ها و یا هر بار که برنامه عوض شود تغییر می نماید. برنامه کاربر در حافظه اصلی شامل دستورات ماشین و داده هاست. متقابلاً، حافظه کنترل یک ریزبرنامه ثابتی را نگهداری می کند و بسوسيله هر کاربری قابل تغییر نیست. ریزبرنامه از ریزدستورالعمل هایی تشکیل شده که بنوبه خود انواع سیگنال های کنترل را برای اجرای ریزاعمال روی ثبات ها معین می کنند. هر دستورالعمل ماشین موجب اجرای رشته ای از ریزدستورالعمل های موجود در حافظه کنترل می شود. وظیفه این ریزدستورالعمل ها برداشت³ دستورالعمل ماشین از حافظه اصلی، محاسبه آدرس موثر، اجرای عملی که توسط دستورالعمل ماشین مشخص شده، و بازگرداندن کنترل به فاز برداشت برای تکرار سیکل مربوط به دستورالعمل بعدی است.

آرایش کلی واحد کنترل ریزبرنامه نویسی شده در دیاگرام شکل ۱-۷ نشان داده شده است. فرض براین است که حافظه کنترل از نوع ROM بوده و همه اطلاعات کنترلی بصورت دائمی در آن ذخیره شده است. ثبات آدرس حافظه کنترل، آدرس ریزدستورالعمل را مشخص می کند، و ثبات داده کنترل

1- Read Only Memory

2- Control Memory

3- Fetch



شکل ۱-۷ سازمان کنترل ریزبرنامه نویسی شده

ریزدستورالعمل خوانده شده از حافظه را نگهدارند. ریزدستورالعمل حاوی یک کلمه کنترل است که یک یا چند ریزعمل را برای پردازشگر داده مشخص می نماید. به محض اجرای این ریزاعمال، کنترل باید آدرس بعدی را معین کند. مکان ریز دستورالعمل بعدی ممکن است خانه بعدی باشد، و یا ممکن است در جای دیگری از حافظه کنترل قرار داشته باشد. باین دلیل لازم است تعدادی از بیت های ریزدستورالعمل فعلی را برای کنترل ایجاد آدرس ریزدستورالعمل بعدی بکار برد. آدرس بعدی می تواند تابعی از شرایط ورودی خارجی نیز باشد. ضمن اجرای ریزاعمال، آدرس بعدی در مدار مولد آدرس بعدی تولید شده و سپس به ثبات آدرس کنترل منتقل می گردد تا ریزدستورالعمل بعدی را بخواند. بنابراین یک ریزدستورالعمل حاوی بیت هایی برای آغاز ریز اعمال در بخش پردازشگر داده و بیت هایی برای تعیین رشته آدرس ها برای حافظه کنترل است.

مولد آدرس بعدی گاهی اوقات توالی گر^۱ ریزبرنامه نامیده می شود، زیرا آدرس های متوالی کلماتی را که از حافظه کنترل خوانده می شوند معین می سازد. آدرس ریزدستورالعمل بعدی بسته به ورودی توالی گر به چند طریق معین می شود. عملیات توالی گر ریزبرنامه عمدتاً عبارتست از افزایش ثبات آدرس کنترل به میزان یک واحد، بارکردن آدرس حافظه کنترل در ثبات آدرس کنترل، انتقال آدرس خارجی، یا بارکردن یک آدرس اولیه برای آغاز اعمال کنترل.

ثبات داده کنترل در حالی که آدرس بعدی محاسبه و از حافظه خوانده می شود، حاوی دستورالعمل فعلی است. ثبات داده گاهی اوقات ثبات خط لوله^۲ هم خوانده می شود. این ثبات امکان اجرای ریزاعمال مشخص شده با کلمه کنترل، همزمان با تولید ریزدستورالعمل بعدی را فراهم می آورد. (برای توضیح بیشتر در مورد ثبات خط لوله به فصل های ۸ و ۹ مراجعه شود). این آرایش نیاز به یک پالس ساعت دو فاز دارد، که یکی از آنها به ثبات آدرس و دیگری به ثبات داده اعمال می شود.

سیستم می تواند با اعمال یک ساعت تک فاز به ثبات آدرس، بدون ثبات داده کنترل کار کند. کلمه کنترل و اطلاعات آدرس بعدی مستقیماً از حافظه کنترل دریافت می شود. باید دانست که ROM بهمانند یک مدار ترکیبی کار می کند، که در آن مقدار آدرس ورودی و کلمه مربوط به آن آدرس خروجی آن است.

1- Sequencer

2- Pipeline Register

محتوای یک کلمه معین شده در ROM تا زمانی که مقدار آدرس در ثبات آدرس باقی بماند، در سیستم‌های خروجی باقی می‌ماند. در حافظه فقط خواندنی هیچ سیگنال خواندنی مورد نیاز نیست. هر پالس ساعت، ریز عمل‌های مشخص شده بوسیله کلمه کنترل را اجرا نموده و آدرس جدیدی را نیز به ثبات آدرس کنترل انتقال می‌دهد. در مثالی که بدنبال آمده است فرض خواهیم کرد پالس ساعت تک فاز باشد و لذا از ثبات داده کنترل استفاده نخواهد شد. باین ترتیب ثبات آدرس تنها جزء موجود در سیستم است که پالس‌های ساعت را دریافت می‌کند. دو جزء دیگر؛ توالی‌گر و حافظه کنترل مدارهای ترکیبی بوده و به پالس ساعت نیازی ندارند.

مزیت عمده کنترل ریز برنامه نویسی شده این است که وقتی پیکربندی سخت‌افزاری تکمیل شد، سخت‌افزار اضافی و یا تغییر سیم‌بندی نخواهیم داشت. اگر خواهیم کنترل متفاوتی را برای سیستم ایجاد کنیم، فقط لازم است مجموعه‌ای از دستورالعمل‌های متفاوتی را برای حافظه کنترل اختیار کنیم. پیکربندی سخت‌افزار برای عملیات متفاوت تغییر نمی‌کند؛ و تنها تغییری که باید صورت گیرد ریز برنامه‌ای است که در حافظه کنترل جای داده می‌شود.

باید یادآوری کرد اکثر کامپیوترهای پایه‌ریزی شده بر مفهوم معماری کامپیوترهای کم‌دستور^۱ یا RISC (به بخش ۸-۸ مراجعه شود) از کنترل سخت‌افزاری استفاده می‌کنند نه از حافظه کنترل ریز برنامه. مثالی از کنترل سخت‌افزاری برای یک کامپیوتر ساده در بخش ۴-۵ ملاحظه شد.

۷-۲ توالی آدرس

ریزدستورالعمل‌ها بصورت گروهی در حافظه ذخیره می‌شوند و هر گروه نیز یک روال را تشکیل می‌دهند. هر دستورالعمل کامپیوتر دارای روال ریز برنامه متعلق به خود در حافظه کنترل است که ریز اعمال مربوط به اجرای دستورالعمل را تولید می‌نماید. سخت‌افزاری که توالی آدرس حافظه کنترل را کنترل می‌نماید باید قادر به دنبال کردن ریز دستورالعمل‌های یک روال بوده و قابلیت انشعاب از یک روال به روال دیگر را هم داشته باشد. برای تحقق توالی دستورات در یک واحد ریز برنامه کنترل، اجازه بدهید مراحل را که کنترل باید حین اجرای یک تک دستورالعمل کامپیوتر طی کند بشماریم.

هنگام روشن شدن کامپیوتر، آدرس اولیه‌ای در ثبات آدرس کنترل بار می‌شود. معمولاً این آدرس، آدرس اولین ریز دستورالعمل است که روال برداشت^۲ دستورالعمل را فعال می‌سازد. اجرای روال برداشت می‌تواند با افزایش ثبات آدرس کنترل تا طی شدن کلیه ریز دستورالعمل‌های آن دنبال شود. در پایان روال برداشت، دستورالعمل در ثبات دستورالعمل کامپیوتر قرار گرفته است.

سپس حافظه کنترل باید به روالی مراجعه کند که طی آن آدرس موثر عملوند تعیین گردد. یک دستورالعمل ماشین ممکن است دارای بیت‌هایی باشد که روش‌های متعدد آدرس‌دهی مانند آدرس‌دهی غیرمستقیم و ثبات شاخص را مشخص کنند. روال محاسبه آدرس موثر در حافظه کنترل از طریق

1- Reduced Instruction Set Computer (RISC)

2- Fetch

ریزدستورالعمل انشعاب قابل دسترسی است، که این انشعابات مشروط بر وضعیت بیت های حالت دستورالعمل صورت می گیرد. وقتی که روال محاسبه آدرس موثر پایان یافت، آدرس عملوند در ثبات آدرس حافظه قرار خواهد داشت.

قدم بعدی تولید ریزاعمالی است که دستورالعمل برداشت شده از حافظه را اجرا می نمایند. مراحل ریزاعمالی که باید در ثبات های پردازنده تولید شوند به کد عمل که بخشی از دستورالعمل است بستگی دارد. به هر دستورالعمل روال ریزبرنامه خاصی تعلق می گیرد که در مکان معینی از حافظه کنترل ذخیره شده است. تبدیل بیت های کد دستورالعمل به یک آدرس در حافظه کنترل که روال در آن قرار دارد نگاشت^۱ نامیده می شود. رویه نگاشت قاعده ای است که کد دستورالعمل را به آدرس حافظه کنترل تبدیل می کند. هنگامی که به روال مورد نظر دسترسی پیدا شد، ریزدستورالعمل هایی که دستورالعمل را اجرا می کنند می توانند با افزایش ثبات آدرس کنترل طی شوند، اما گاهی هم توالی ریزدستورالعمل ها به مقادیر معینی از بیت های وضعیت در ثبات های پردازشگر بستگی دارد. ریزبرنامه هایی که از زیرروال ها استفاده می کنند به یک ثبات خارجی برای نگهداری آدرس بازگشت نیاز دارند. آدرس های بازگشت نمی توانند در ROM ذخیره شوند زیرا ROM دارای قابلیت نوشتن نیست.

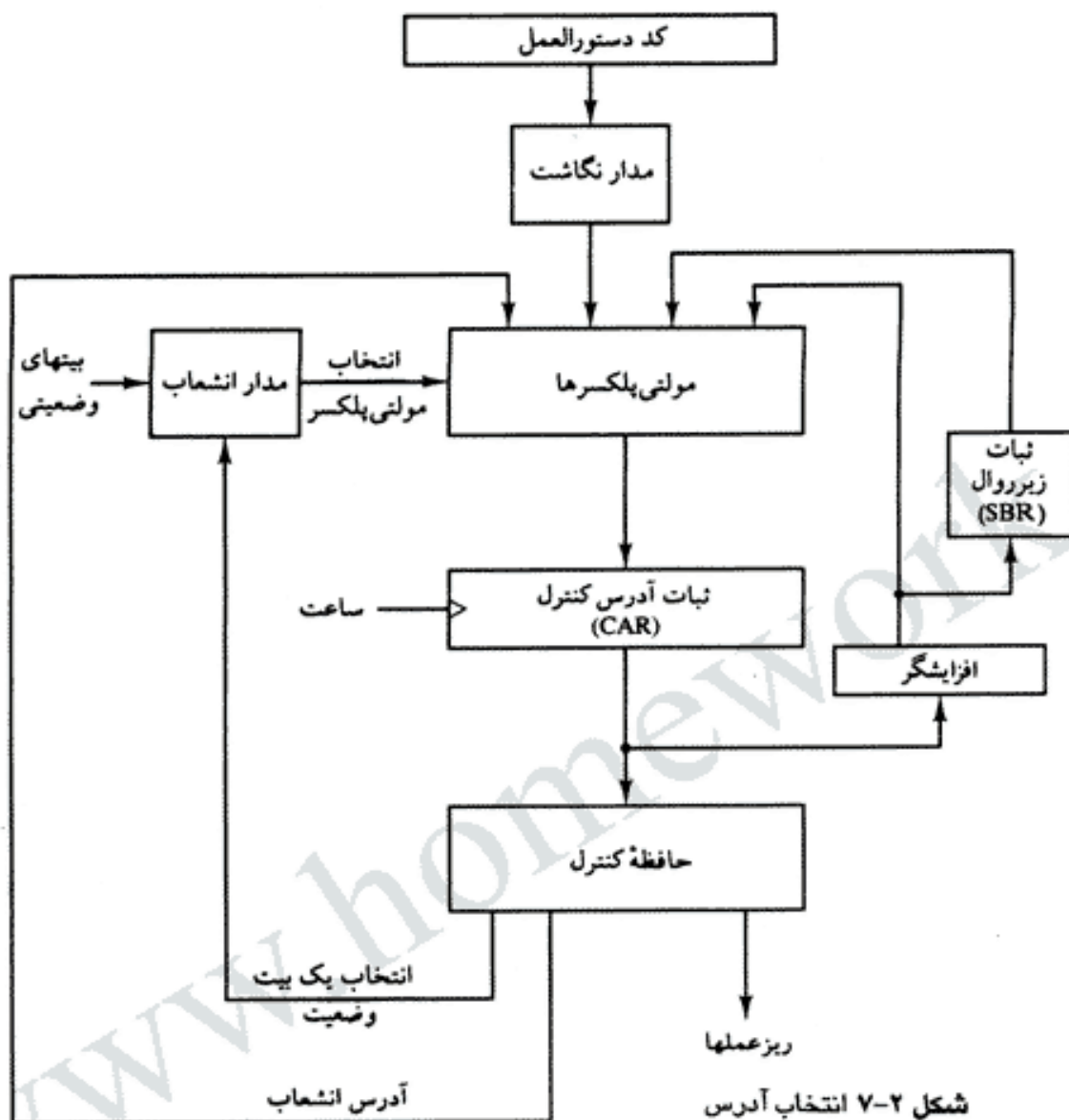
هنگامی که اجرای دستورالعملی پایان یابد، کنترل باید به روال برداشت (واکشی) بازگردد. این عمل با اجرای یک ریزدستورالعمل انشعاب غیرشرطی به ابتدای روال برداشت تحقق می یابد. بطور خلاصه، قابلیت های لازم برای توالی آدرس در حافظه کنترل عبارتند از:

- ۱- افزایش ثبات آدرس کنترل
- ۲- انشعاب غیرشرطی و یا شرطی، بسته به شرایط بیت های وضعیت
- ۳- فرآیند نگاشت (تبدیل) از بیت های دستورالعمل به آدرس حافظه کنترل
- ۴- امکاناتی برای فراخوانی زیرروال و بازگشت از آن

شکل ۲-۷ بلاک دیاگرام یک حافظه کنترل و سخت افزار لازم برای انتخاب آدرس ریزدستورالعمل بعدی را نشان می دهد. ریزدستورالعمل واقع در حافظه کنترل متشکل از مجموعه ای از بیت هاست که ریزاعمال را در ثبات های کامپیوتر موجب می شود و نیز مجموعه ای از بیت های دیگر برای تهیه آدرس بعدی است. دیاگرام چهار مسیری که توسط آنها ثبات آدرس کنترل^۲ (CAR) آدرس را دریافت می نماید، نشان می دهد. افزایشگر محتوای ثبات آدرس کنترل را یک واحد افزایش می دهد تا ریزدستورالعمل بعدی را انتخاب نماید. عمل انشعاب پس از مشخص شدن آدرس انشعاب در یکی از میدان های ریزدستورالعمل انجام می شود. انشعاب شرطی با بکارگیری بخشی از ریزدستورالعمل بخاطر انتخاب بیت وضعیت معینی برای مشخص نمودن شرایط انشعاب، صورت می گیرد. آدرس خارجی (که بخشی از دستورالعمل است) از طریق مدار منطقی نگاشت به کنترل حافظه منتقل می شود. آدرس برگشت

1- Mapping

2- Control Address Register



شکل ۷-۲ انتخاب آدرس برای حافظه کنترل

برای زیر روال در ثبات خاصی که مقدارش بعداً بهنگام بازگشت مورد استفاده قرار می گیرد ذخیره می شود.

انشعاب شرطی

منطق انشعاب شکل ۷-۲ قابلیت تصمیم گیری را در واحد کنترل فراهم می آورد. حالت وضعیت سیستم، بیت های خاصی هستند که اطلاعات پارامتری همچون، نقلی خروجی یک جمع کننده، بیت علامت یک عدد، بیت های روش^۱ (شیوه) در یک دستورالعمل، شرایط وضعیتی ورودی یا

1- Mode bit

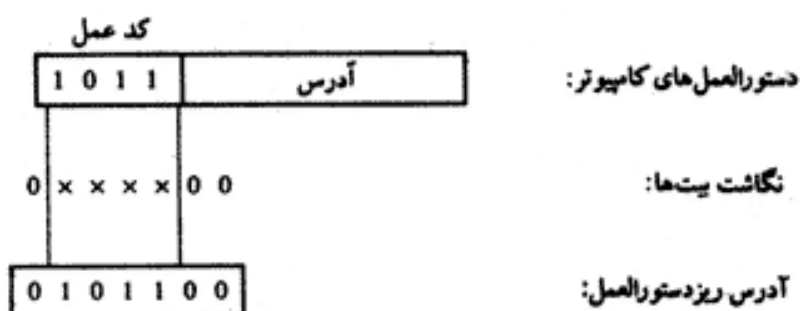
خروجی^۱ را فراهم می‌کنند. اطلاعات موجود در این بیت‌ها قابل تست بوده و تصمیم اتخاذ شده می‌تواند بر اساس حالت آنها یعنی 1 یا 0 باشد. بیت‌های وضعیت، همراه با میدانی در ریزدستورالعمل که آدرس انشعاب را مشخص می‌نماید، تصمیمات انشعاب‌های شرطی را که در مدار انشعاب تولید می‌شود کنترل می‌نمایند.

سخت‌افزار مدار انشعاب بطرق مختلفی قابل پیاده‌سازی است. در ساده‌ترین راه، شرط مشخص شده و انشعاب به آدرس معین شده با برقرار بودن شرط صورت می‌گیرد، در غیر اینصورت ثبات آدرس افزایش می‌یابد. این مدار می‌تواند با یک مولتی‌پلکسر پیاده‌سازی شود. فرض کنید که بیت‌های وضعیت جمعاً بتوانند هشت حالت داشته باشند، از سه بیت در ریزدستورالعمل می‌توان برای مشخص کردن هر یک از هشت حالت وضعیت استفاده کرد. این سه بیت متغیرهای انتخاب را برای مولتی‌پلکسر تشکیل می‌دهند. اگر بیت وضعیت انتخاب شده در حالت 1 باشد، خروجی مولتی‌پلکسر 1 است؛ در غیر اینصورت 0 است. خروجی 1 در مولتی‌پلکسر یک سیگنال کنترل برای انتقال آدرس انشعاب از ریزدستورالعمل به ثبات آدرس کنترل تولید می‌کند. خروجی 0 در مولتی‌پلکسر موجب افزایش ثبات آدرس می‌گردد. در این پیکربندی ریزبرنامه یکی از دو مسیر ممکن را بسته به مقدار بیت وضعیت دنبال خواهد کرد.

ریزدستورالعمل انشعاب غیرشرطی را می‌توان با بارکردن آدرس انشعاب از حافظه کنترل به ثبات آدرس کنترل ایجاد کرد. این کار را می‌توان با ثابت نگهداشتن مقدار یک بیت وضعیت در ورودی مولتی‌پلکسر در منطق 1 انجام داد. ارجاع به این بیت توسط خطوط انتخاب از حافظه کنترل موجب می‌شود تا آدرس انشعاب در ثبات آدرس کنترل بدون هر شرطی بار شود.

نگاشت^۲ دستورالعمل

نوع خاصی از انشعاب هنگامی است که یک ریزدستورالعمل حکم کند که به اولین کلمه در حافظه کنترل که روال ریزبرنامه مربوط به یک دستورالعمل در آنجا واقع است انشعاب شود. بیت‌های وضعیت برای این نوع انشعاب، بیت‌های بخش کد عملیات در دستورالعمل هستند. مثلاً، کامپیوتری با قالب



شکل ۳-۷ نگاشت (تبدیل) کد دستورالعمل به آدرس ریزدستورالعمل

1- Input or output state Condition

2- Mapping

ساده‌ای برای دستورالعمل، مطابق شکل ۳-۷، کد عمل چهار بیتی دارد که می‌تواند تا 16 دستورالعمل متمایز را تشخیص دهد. فرض کنید حافظه کنترل 128 کلمه داشته باشد، در نتیجه به آدرس هفت بیتی نیاز خواهد داشت. برای هر کد عملیات یک روال ریزبرنامه در حافظه وجود دارد که دستورالعمل را اجرا می‌کند. نمونه ساده‌ای از فرآیند نگاشت که کد عملیات 4 بیتی را به آدرس 7 بیتی برای حافظه کنترل تبدیل می‌کند در شکل ۳-۷ نشان داده شده است. این نگاشت بدین صورت انجام می‌شود که یک 0 در پرارزترین مکان آدرس قرار داده می‌شود، چهار بیت کد عملیات منتقل می‌گردد، و دو بیت کم‌ارزتر ثبات آدرس کنترل پاک می‌شوند. لذا برای هر دستورالعمل کامپیوتر می‌توان یک روال ریزبرنامه با ظرفیت چهار ریزدستورالعمل داشت. اگر روال به بیش از چهار ریزدستورالعمل نیاز داشته باشد، می‌توان از آدرس‌های 1000000 تا 1111111 استفاده کرد و اگر کمتر از چهار ریزعمل بکار رود، از مکان‌های حافظه بلااستفاده می‌توان برای روال‌های دیگر استفاده کرد.

می‌توان با استفاده از ROM در مشخص کردن تابع نگاشت، مفهوم نگاشت را تعمیم داد. در این آرایش، بیت‌های دستورالعمل آدرس ROM، نگاشت را مشخص می‌کنند. محتوای ROM نگاشت بیت‌های ثبات آدرس کنترل را تشکیل می‌دهند. به این ترتیب روال ریزبرنامه‌ای که دستورالعمل را اجرا می‌کند در هر جای مطلوبی در حافظه کنترل می‌تواند قرارگیرد. روش نگاشت انعطاف لازم را برای افزایش دستورالعمل‌ها، بعلت بالارفتن تقاضا فراهم می‌نماید.

گاهی تابع نگاشت با استفاده از مدار مجتمعی به نام قطعه منطقی برنامه پذیر¹ یا PLD نیز پیاده‌سازی می‌شود. از نظر مفهوم PLD شبیه ROM است با این تفاوت که از گیت‌های AND و OR با فیوزهای الکترونیکی داخلی استفاده می‌کند. اتصالات بین ورودیها، گیت‌های AND، گیت‌های OR، و خروجیها را می‌توان مانند ROM برنامه‌نویسی کرد. هر تابع نگاشتی که با عبارات بولی بیان شود به آسانی با یک PLD تحقق می‌یابد.

زیرروال‌ها²

زیرروال‌ها برنامه‌هایی هستند که بوسیله سایر روال‌ها برای انجام کار خاصی مورد استفاده قرار می‌گیرند. زیرروال را می‌توان از هر نقطه‌ای در متن اصلی ریزبرنامه فراخوانی کرد. غالباً بسیاری از ریزبرنامه‌ها حاوی بخش‌های یکسانی از کد هستند. می‌توان ریزدستورالعمل را با بکارگیری زیرروال‌هایی که بخش‌های مشترکی از ریزکدها دارند کاهش داد. مثلاً، رشته ریزاعمال لازم برای تولید آدرس موثر عملوند یک دستورالعمل برای تمام دستورالعمل‌های ارجاع به حافظه مشترک است. این رشته می‌تواند زیرروالی باشد که از درون روال‌های متعدد دیگر برای اجرای محاسبات آدرس موثر فراخوانی شود.

ریزبرنامه‌هایی که از زیرروال استفاده می‌کنند باید امکاناتی برای ذخیره آدرس بازگشت بهنگام فراخوانی، و بازیابی آدرس بهنگام برگشت از زیرروال داشته باشد. این کار را می‌توان با قرار دادن

1- Programmable Logic Device

2- Subroutine

خروجی افزایش یافته از ثبات آدرس کنترل در ثبات زیرروال و انشعاب به ابتدای زیرروال انجام داد. ثبات زیرروال سپس می‌تواند بعنوان منبعی جهت انتقال آدرس بازگشت به روال اصلی مورد استفاده قرار گیرد. بهترین راه ایجاد مجموعه‌ای از ثباتها برای ذخیره آدرس‌های زیرروال‌ها، سازماندهی ثبات‌ها به شکل یک پشته^۱ از نوع آخرین ورودی-اولین خروجی^۲ (LIFO) می‌باشد. استفاده از پشته در یک زیرروال فراخوانی و بازگشت از زیرروال‌ها در بخش ۸-۷ مفصلاً شرح داده شده است.

۷-۳ مثال ریزبرنامه

وقتی که آرایش یک کامپیوتر و واحد کنترل ریزبرنامه‌نویسی شده آن مسجل شد، وظیفه طراح، تولید ریزکد برای حافظه کنترل است. این تولیدکد را ریز برنامه‌نویسی می‌نامند که مشابه برنامه‌نویسی معمولی بزبان ماشین است. برای درک بهتر این فرآیند، در اینجا یک کامپیوتر دیجیتال ساده را ارائه داده و نشان می‌دهیم. کامپیوتر مورد بحث در این مثال مشابه کامپیوتر پایه معرفی شده در فصل ۵ است ولی دقیقاً همانند آن نیست.

آرایش کامپیوتر

بلاک دیاگرام کامپیوتر مورد نظر در شکل ۴-۷ نشان داده شده است. این کامپیوتر از دو واحد حافظه تشکیل شده است: یکی حافظه اصلی برای ذخیره دستورالعمل‌ها و داده‌ها، و دیگری حافظه کنترل برای ذخیره ریزبرنامه. چهار ثبات به واحد پردازنده و دو ثبات به واحد کنترل اختصاص یافته‌اند. ثبات‌های پردازنده عبارتند از شمارنده برنامه^۳، PC، ثبات آدرس AR، ثبات داده DR و ثبات انبار AC. کار این ثبات‌ها مشابه کامپیوتر پایه‌ای است که در فصل ۵ معرفی شد (شکل ۳-۵ ملاحظه شود). واحد کنترل دارای ثبات آدرس کنترل CAR و یک ثبات زیرروال SBR است. حافظه کنترل و ثبات‌های متعلق به آن، همان‌طور که در شکل ۲-۷ دیده می‌شود، بصورت یک واحد کنترل ریزبرنامه‌نویسی شده شناخته می‌شوند.

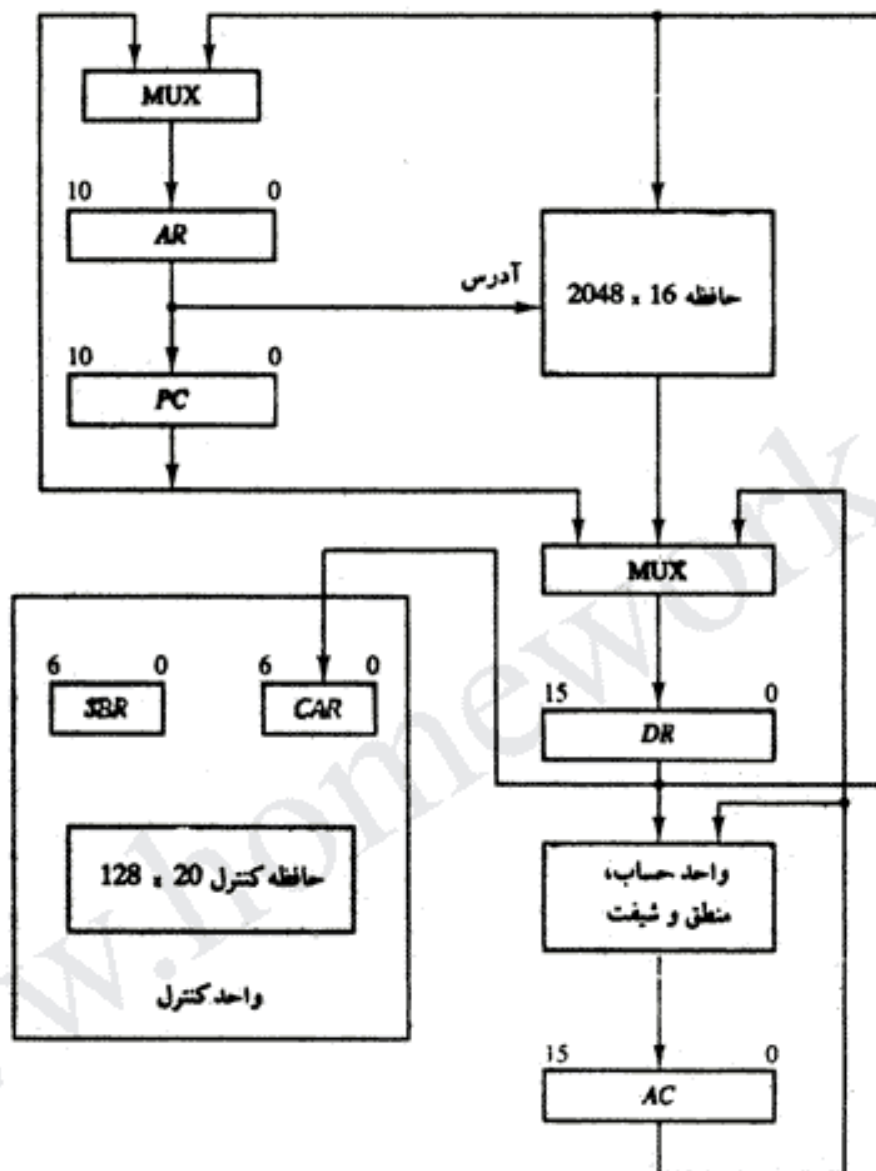
انتقال اطلاعات بین ثبات‌های پردازنده از طریق مولتی‌پلکسر صورت می‌گیرد و نه با استفاده از گذرگاه مشترک. DR می‌تواند اطلاعات را از AC، PC و یا حافظه دریافت کند. PC فقط از AR اطلاعات دریافت می‌کند. واحد حساب، منطق و شیفت ریزاعمال را بر روی داده‌های دریافتی از AC و DR انجام داده و حاصل را در AC قرار می‌دهد. دقت کنید که حافظه آدرس خود را از AR دریافت می‌کند. داده ورودی که در حافظه نوشته می‌شود از طریق DR وارد، و داده خوانده شده از حافظه هم فقط به DR می‌رود.

قالب دستورالعمل‌های کامپیوتر در شکل ۵-۷ (الف) نشان داده شده است. این قالب از سه میدان تشکیل شده است: یک میدان ۱ بیتی برای آدرس دهی غیرمستقیم که با I مشخص شده است، یک کد

1- stack

2- Last In First Out

3- Program Counter



شکل ۴-۷ پیکربندی سخت‌افزار کامپیوتر

عملیات^۱ 4 بیتی (opcode) و یک میدان آدرس 11 بیتی. شکل ۵-۷ (ب) چهار دستور از شانزده دستورالعمل‌های ارجاع به حافظه را نشان می‌دهد. دستورالعمل ADD، عملوند موجود در آدرس موثر را با محتوای AC جمع می‌کند. دستورالعمل BRANCH، انشعابی را به آدرس موثر موجب می‌شود بشرطی که عملوند درون AC منفی باشد. اگر AC منفی نباشد برنامه با دستورالعمل بعدی ادامه می‌یابد. AC در صورتی منفی است که بیت علامت (بیت مستهالیه سمت چپ ثبات) 1 باشد. دستورالعمل STORE محتوای AC را به کلمه‌ای از حافظه که آدرس موثر آن را مشخص می‌کند انتقال

1- Operation Code

15	14	11	10	0
I	کد عمل	آدرس		

(الف) قالب دستورالعمل

توضیح	کد عمل	سمبل
$AC \leftarrow AC + M[EA]$	0000	ADD
If $(AC < 0)$ then $(PC \leftarrow EA)$	0001	BRANCH
$M[EA] \leftarrow AC$	0010	STORE
$AC \leftarrow M[EA], M[EA] \leftarrow AC$	0011	EXCHANGE

EA آدرس مؤثر است

(ب) چهار دستورالعمل کامپیوتر

شکل ۷-۵ دستورالعمل های کامپیوتر

می دهد. دستورالعمل EXCHANGE داده هایی را بین AC و کلمه حافظه مشخص شده بوسیله آدرس مؤثر تعویض می کند.

در دنباله بحث نشان داده خواهد شد که هر دستورالعمل کامپیوتر بایستی ریز برنامه نویسی شود. بخاطر پرهیز از پیچیده شدن مثال ریز برنامه نویسی، در اینجا فقط چهار دستورالعمل مورد بررسی قرار گرفته است. باید دانست که 12 دستورالعمل دیگر می تواند به مجموعه اضافه شود و هر دستورالعمل باید توسط روند شرح داده شده در ذیل ریز برنامه نویسی گردد.

قالب ریز دستورالعمل ها

قالب ریز دستورالعمل ها برای حافظه کنترل در شکل ۷-۶ نشان داده شده است. 20 بیت هر دستورالعمل به چهار بخش عملیاتی تقسیم شده است. سه میدان F1، F2 و F3 ریز اعمال را برای کامپیوتر مشخص می کنند. میدان CD بیت های شرطی وضعیت را انتخاب می نماید. میدان BR نوع

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

F1، F2، F3: میدان های ریز عمل

CD: شرط انشعاب

BR: میدان انشعاب

AD: میدان آدرس

شکل ۷-۶ قالب کد ریز دستورالعمل (20 بیت)

انشعاب بکار رفته را معین می‌کند. میدان AD آدرس انشعاب را داراست. میدان آدرس هفت بیت عرض دارد، زیرا حافظه کنترل دارای $2^7 = 128$ کلمه است.

گفتیم که ریزاعمال به سه میدان جزئی تقسیم شده‌اند که هر یک سه بیتی هستند. هر سه بیت در هر میدان برای مشخص کردن هفت ریزعمل مستقل، مطابق جدول ۷-۱، انکد شده‌اند. لذا در مجموع 21 ریزعمل وجود خواهد داشت. برای هر ریزدستور نمی‌توان بیش از سه ریزعمل انتخاب کرد، یعنی هر یک برای یک میدان. اگر کمتر از سه ریزعمل بکار رود، یک یا چند میدان از کد 000 بمعنی هیچکار^۱ استفاده خواهند کرد. مثلاً یک ریزدستورالعمل می‌تواند بطور همزمان دو ریزعمل را در F2 و F3 و هیچ عمل را در F1 انجام دهد.

$$DR \leftarrow M[AR] \quad \text{با } F2 = 100$$

و

$$PC \leftarrow PC + 1 \quad \text{با } F3 = 101$$

9 بیت میدان ریز عملیات عبارت از 101 100 000 خواهند بود. توجه به این نکته که نمی‌توان دو یا چند عمل متضاد را به طور همزمان اجرا کرد اهمیت دارد. مثلاً میدان ریزاعمال 000 001 010 مفهومی ندارد زیرا این ریزاعمال، عملیاتی مانند پاک کردن AC و تفریق DR از AC را بطور همزمان نشان می‌دهد. هر ریزعمل در جدول ۷-۱ بوسیله یک عبارت انتقال ثبات تعریف شده و سمبلی جهت استفاده در برنامه‌های سمبلیک به آن اختصاص داده شده است. همه سمبل‌های ریزعمل‌های انتقالی از پنج حرف استفاده می‌کنند. دو حرف اول ثبات مبدا را مشخص می‌کند، سومین حرف همواره T است، و دو حرف آخر ثبات مقصد را معین می‌نماید.

میدان شرط^۲ CD دو بیتی است که برای تعیین چهار بیت وضعیت شرطی طبق جدول ۷-۱ انکد شده است. اولین حالت همیشه 1 است، ارجاع به $CD = 00$ (یا سمبل U) همیشه با شرط درست مواجه می‌شود. اگر این حالت همراه با میدان BR (انشعاب) بکار رود یک عمل انشعاب بدون شرط حاصل می‌شود. بیت غیرمستقیم I را پس از خواندن یک دستورالعمل از حافظه می‌توان در بیت 15 ثبات DR بدست آورد. بیت علامت AC بیت وضعیت بعدی را تشکیل می‌دهد. مقدار صفر، که با Z مشخص شده است، یک متغیر دودویی است که اگر تمام بیت‌ها در AC صفر شود، این متغیر 1 خواهد شد. ما هنگامی که ریزبرنامه‌هایی را به فرم سمبلیک بنویسیم از چهار سمبل U، I، S، Z استفاده خواهیم کرد. میدان BR دو بیتی است. این میدان به همراه میدان آدرس AD برای انتخاب آدرس ریزدستور بعدی بکار می‌رود. همان‌طور که در جدول ۷-۱ ملاحظه می‌شود، وقتی $BR = 00$ باشد، کنترل عمل پرش (JMP) را اجرا می‌کند (که مشابه عمل انشعاب است)، و هنگامی که $BR = 01$ باشد عمل فراخوانی زیرروال (CALL) انجام خواهد شد. دو عمل یکسان هستند با این تفاوت که ریزعمل فراخوانی آدرس

1- No operation

2- Condition Field

جدول ۷-۱ سمبلها و کدهای دودویی برای میدانهای ریز دستورات العملها

F1	ریز عمل	سمبل
000	هیچکار	NOP
001	$AC \leftarrow AC + DR$	ADD
010	$AC \leftarrow 0$	CLRAC
011	$AC \leftarrow AC + 1$	INCAC
100	$AC \leftarrow DR$	DRTAC
101	$AR \leftarrow DR(0-10)$	DRTAR
110	$AR \leftarrow PC$	PCTAR
111	$M[AR] \leftarrow DR$	WRITE

F2	ریز عمل	سمبل
000	هیچکار	NOP
001	$AC \leftarrow AC - DR$	SUB
010	$AC \leftarrow AC \vee DR$	OR
011	$AC \leftarrow AC \wedge DR$	AND
100	$DR \leftarrow M[AR]$	READ
101	$DR \leftarrow AC$	ACTDR
110	$DR \leftarrow DR + 1$	INCDR
111	$DR(0-10) \leftarrow PC$	PCTDR

F3	ریز عمل	سمبل
000	هیچکار	NOP
001	$AC \leftarrow AC \oplus DR$	XOR
010	$AC \leftarrow \overline{AC}$	COM
011	$AC \leftarrow \text{shl } AC$	SHL
100	$AC \leftarrow \text{shr } AC$	SHR
101	$PC \leftarrow PC + 1$	INCPC
110	$PC \leftarrow AR$	ARTPC
111	Reserved	

CD	شرط	سمبل	توضیح
00	همیشه = 1	U	انشعاب غیرشرطی
01	$DR(15)$	I	بیت آدرس غیرمستقیم
10	$AC(15)$	S	بیت علامت AC
11	$AC = 0$	Z	مقدار صفر در AC

BR	سمبل	عملکرد
00	JMP	اگر شرط برابر 1 باشد $CAR \leftarrow AD$ اگر شرط برابر صفر باشد $CAR \leftarrow CAR + 1$
01	CALL	اگر شرط برابر 1 باشد $CAR \leftarrow AD, SBR \leftarrow CAR + 1$ اگر شرط برابر 0 باشد $CAR \leftarrow CAR + 1$
10	RET	بازگشت از زیرروال ($CAR \leftarrow SBR$)
11	MAP	$CAR(2-5) \leftarrow DR(11-14), CAR(0,1,6) \leftarrow 0$

بازگشت را در ثبات زیرروال SBR ذخیره می‌کند. اعمال پرش و فراخوانی به میدان CD وابسته هستند. اگر شرایط بیت وضعیت در میدان CD برابر 1 باشد، آدرس بعدی واقع در میدان AD به ثبات آدرس کنترل CAR منتقل می‌گردد. در غیراینصورت CAR یک واحد اضافه می‌شود. بازگشت از زیرروال در صورتی انجام می‌شود که میدان BR برابر 10 باشد. این وضعیت موجب می‌شود آدرس بازگشت از SBR به CAR منتقل شود. نگاشت بیت‌های کد عملیات دستورالعمل به آدرسی برای CAR در صورتی انجام می‌شود که میدان BR برابر 11 باشد. این نگاشت در شکل ۳-۷ نشان داده شده است. بیت‌های کد عملیات پس از خواندن یک دستورالعمل از حافظه در DR(11-14) قرار دارد. توجه کنید که دو حالت آخر در میدان BR مستقل از مقادیر میدان‌های CD و AD است.

ریز دستورالعمل‌های سمبلیک

سمبل‌های ارائه شده در جدول ۱-۷ می‌توانند برای مشخص کردن ریزدستورالعمل‌ها به فرم سمبلیک بکارروند. یک ریزبرنامه سمبلیک می‌تواند بکمک اسمبلر به معادل دودویی‌اش تبدیل شود. اسمبلر یک ریزبرنامه از نظر مفهوم مشابه اسمبلر معمولی در کامپیوتر است که در بخش ۳-۶ تعریف شد. ساده‌ترین و سراسرترین روش تنظیم یک زبان اسمبلی برای یک ریزبرنامه، تعریف سمبل‌ها برای هر میدان ریز دستورالعمل و امکان تعریف آدرس سمبلیک توسط خود کاربران است.

هر سطر از ریزبرنامه زبان اسمبلی، یک ریزدستورالعمل سمبلیک را تعریف می‌کند. هر دستورالعمل سمبلیک به پنج میدان تقسیم می‌شود: عنوان، ریزاعمال، CD، BR، AD. این میدان‌ها اطلاعات زیر را مشخص می‌نمایند.

- ۱- میدان عنوان می‌تواند خالی و یا یک آدرس سمبلیک باشد. هر عنوان با یک علامت نقل قول (:) خاتمه می‌یابد.
- ۲- میدان ریزعمل‌ها از یک، دو و یا سه سمبل تعریف شده در جدول ۱-۷ تعریف می‌شود و بوسیله کاما از هم جدا می‌شوند. نمی‌توان بیش از یک سمبل از هر میدان F داشت. هرگاه ریزدستورالعمل هیچ عملی نداشته باشد از سمبل NOP¹ استفاده می‌شود. اسمبلر این سمبل را به 9 عدد 0 ترجمه می‌کند.
- ۳- میدان CD دارای یک حرف U، I، S و یا Z است.
- ۴- میدان BR حاوی یکی از چهار سمبل تعریف شده در جدول ۱-۷ است.
- ۵- میدان AD مقدار میدان آدرس ریز دستورالعمل را به یکی از سه طریق زیر مشخص می‌کند.
 - (الف) با یک آدرس سمبلیک، که باید بصورت عنوان هم وجود داشته باشد.
 - (ب) با سمبل NEXT برای مشخص کردن آدرس متوالی بعدی
 - (ج) اگر میدان BR حاوی سمبل RET، یا MAP باشد، میدان AD خالی گذاشته می‌شود و اسمبلر آن را به هفت عدد صفر تبدیل می‌نماید.

1- No Operation

ما از شبه دستور ORG نیز برای تعریف مبدا، یا اولین آدرس یک روال ریزبرنامه استفاده می‌کنیم. مثلاً سمبل 64 ORG به اسمبلر اطلاع می‌دهد که ریزدستورالعمل بعدی را در حافظه کنترل در آدرس دهی 64 قرار دهد، که معادل آدرس دودویی 1000 000 است.

روال برداشت^۱

حافظه کنترل دارای 128 کلمه است، و هر کلمه 20 بیت دارد. برای ریزبرنامه‌نویسی حافظه کنترل، لازم است تا مقادیر بیتی هر 128 کلمه را معین کنیم. 64 کلمه اول (آدرس 0 تا 63) بوسیله روال‌های 16 دستورالعمل اشغال می‌شوند. 64 کلمه آخر را می‌توان برای هر هدف دیگری بکار برد. آدرس 64 شروع مناسبی برای روال برداشت (یا واکنشی) است. ریزدستورالعمل‌های لازم برای این روال عبارتند از:

AR ← PC

DR ← M[AR], PC ← PC + 1

AR ← DR (0 - 10), CAR (2 - 5) ← DR (11 - 14), CAR (0, 1, 6) ← 0

آدرس دستورالعمل از PC به AR منتقل شده و سپس دستورالعمل از حافظه به DR خوانده می‌شود. چون هیچ ثبات دستورالعملی وجود ندارد، کد دستورالعمل در DR باقی می‌ماند. بخش آدرس به AR منتقل و سپس کنترل بوسیله نگاشت بخش کد عملیات دستورالعمل از DR به CAR به یکی از 16 روال انتقال می‌یابد.

روال برداشت به سه ریزدستورالعمل نیاز دارد، که در آدرس 64، 65 و 66 حافظه کنترل قرار داده می‌شوند. با استفاده از قواعد زبان اسمبلی که قبلاً تعریف شد، ما می‌توانیم ریزبرنامه سمبلیک را برای روال برداشت بنویسیم:

```

ORG 64
FETCH: PCTAR          U   JMP   NEXT
        READ, INCPC  U   JMP   NEXT
        DRTAR       U   MAP

```

ترجمه ریزبرنامه سمبلیک به دودویی، ریزبرنامه دودویی زیر را تولید می‌کند. مقادیر بیت‌ها از جدول ۱-۷ بدست می‌آیند.

آدرس دودویی	F1	F2	F3	CD	BR	AD
1000000	110	000	000	00	00	1000001
1000001	000	100	101	00	00	1000010
1000010	101	000	000	00	11	0000000

1- Fetch Roution

سه ریز دستورالعمل که روال برداشت را تشکیل می دهند به سه فرم مختلف نشان داده شدند. نمایش انتقال ثباتی عمل انتقال بین ثبات ها را که هر ریزدستورالعمل پیاده می کند نشان می دهد. نمایش سمبلیک برای نوشتن ریزبرنامه در قالب زبان اسمبلی مفید است. نمایش دودویی محتوای داخلی واقعی را که باید در حافظه کنترل ذخیره شود نشان می دهد. مرسوم است که ریزبرنامه ها را به شکل سمبلیک بنویسند و سپس با استفاده از یک برنامه اسمبلر ترجمه دودویی آن را بدست آورند.

ریزبرنامه سمبلیک

اجرای سومین ریزدستورالعمل در روال برداشت (MAP) سبب می شود به آدرس 0xxxx00 انشعاب شود که در آن xxxx چهار بیت کد عملیات است. مثلاً اگر دستورالعمل، ADD باشد که کد عمل آن 0000 است، ریزدستورالعمل MAP آدرس 00000000 که آدرس شروع روال ADD در حافظه کنترل می باشد را به CAR منتقل می سازد. اولین آدرس برای روال های BRANCH و STORE بترتیب عبارتند از 00 0001 00 (دهدهی 4) و 00 0010 00 (دهدهی 8). اولین آدرس سیزده روال دیگر در آدرس های 12، 16، 20، ... و 60 می باشند. باین ترتیب چهار کلمه در حافظه کنترل به هر روال اختصاص می یابد. در هر روال باید ریزدستورالعمل هایی برای محاسبه آدرس موثر و اجرای دستورالعمل در نظر بگیریم. روش آدرس دهی غیرمستقیم در همه دستورالعمل های حافظه ای می تواند بکار رود. با ذخیره دستورالعمل های مربوط به آدرس دهی غیرمستقیم به صورت یک روال، می توان در تعداد کلمه های حافظه کنترل صرفه جویی کرد. این زیرروال، که با INDRCT نشان داده شده، درست پس از روال برداشت قرار دارد، جدول ۲-۷. این جدول ریزبرنامه سمبلیک را برای روال برداشت و روال های ریزدستورالعملی که چهار دستورالعمل کامپیوتر اجرا می کنند را نیز نشان می دهد.

برای اینکه ببینیم چگونه انتقال و برگشت از زیرروال غیرمستقیم INDRCT رخ می دهد، فرض کنید که ریزدستورالعمل MAP در انتهای روال برداشت موجب انشعاب به آدرس 0، یعنی مکانی که روال ADD در آن قرار دارد، شود. اولین ریزدستورالعمل در روال ADD زیرروال INDRCT را با شرط بیت وضعیت I، فرامی خواند. اگر $I = 1$ باشد انشعاب به INDRCT رخ داده و آدرس برگشت (آدرس 1 در این حالت) در ثبات زیر روال SBR ذخیره می شود. زیرروال INDRCT دو ریزدستورالعمل دارد.

```
INDRCT: READ U JUMP NEXT
        DRTAR U RET
```

بخاطر آورید که یک آدرس غیرمستقیم، بخش آدرس دستورالعمل را بعنوان آدرسی که آدرس موثر در آن قرار دارد تلقی می کند و آنرا عملوند تصور نمی نماید. بنابراین حافظه برای بدست آوردن آدرس موثر باید دستیابی شود و سپس به AR منتقل گردد. برگشت از زیرروال (RET) آدرس SBR را به CAR انتقال می دهد، و لذا به دومین ریزدستورالعمل روال ADD بازمی گردد.

جدول ۷-۲ بخشی از برنامه سمبلیک

عنوان	ریزعمل های	CD	BR	AD
ADD:	ORG 0			
	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ADD	U	JMP	FETCH
BRANCH:	ORG 4			
	NOP	S	JMP	OVER
	NOP	U	JMP	FETCH
OVER:	NOP	I	CALL	INDRCT
	ARTPC	U	JMP	FETCH
STORE:	ORG 8			
	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
	WRITE	U	JMP	FETCH
EXCHANGE:	ORG 12			
	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
	ACTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH
FETCH:	ORG 64			
	PCTAR	U	JMP	NEXT
	READ, INCPC	U	JMP	NEXT
INDRCT:	DRTAR	U	MAP	
	READ	U	JMP	NEXT
	DRTAR	U	RET	

اجرای دستورالعمل ADD بوسیله ریزدستورالعمل واقع در آدرس های 1 و 2 انجام می گردد. اولین ریز دستورالعمل عملوند را از حافظه به DR می خواند. دومین ریز دستورالعمل ریزعمل جمع با محتوای DR و AC انجام داده و سپس به ابتدای روال برداشت بازمی گردد.

دستورالعمل BRANCH موجب انشعابی به آدرس موثر می گردد بشرطی که $AC < 0$ باشد. AC هنگامی کوچکتر از صفر است که بیت علامت در آن منفی باشد که این از 1 بودن بیت وضعیت S آشکار می گردد. روال BRANCH در جدول ۷-۲ با واریسی مقدار S آغاز می شود. اگر S برابر 0 باشد، انشعابی رخ نمی دهد و ریزدستورالعمل بعدی موجب پرش به عقب به روال برداشت، بدون تغییر محتوای PC، می گردد. اگر S برابر 1 باشد، اولین ریزدستورالعمل JMP کنترل را به مکان OVER منتقل می سازد. ریزدستورالعمل واقع در این مکان زیرروال INDRCT را، اگر $I = 1$ باشد، فرامی خواند. سپس آدرس

موثر از AR به PC منتقل و زیربرنامه به روال برداشت پرش می‌کند.
 روال STORE مجدداً زیرروال INDRCT را، اگر $I = 1$ باشد، فرامی‌خواند. محتوای AC به DR منتقل می‌شود. یک عمل نوشتن در حافظه، برای ذخیره کردن DR در مکان مشخص شده بوسیله آدرس موثر در AR، اجرا می‌شود.

روال EXCHANGE عملوند را از آدرس موثر خوانده و آنرا در DR قرار می‌دهد. محتوای DR و AC در سومین دستورالعمل با یکدیگر تعویض می‌شوند. این تعویض هنگامی میسر است که ثباتها از نوع حساس به لبه^۱ باشند (شکل ۲۳-۱). محتوای اصلی AC که حالا در DR است مجدداً در حافظه ذخیره می‌شود. توجه کنید که جدول ۲-۷ بخشی از ریزبرنامه را داراست. تنها چهار دستور از ۱۶ دستور کامپیوتر ریزبرنامه نویسی شدند. همچنین کلمات حافظه کنترل در مکان‌های ۶۹ تا ۱۲۷ مورد استفاده قرار نگرفتند. دستوراتی مانند ضرب، تقسیم که نیاز به رشته ریزاعمال طولانی تری دارند برای اجرا به بیش از چهار ریز دستورالعمل نیاز دارند. کلمات حافظه کنترل از ۶۹ تا ۱۲۷ می‌توانند برای این منظور بکار روند.

ریز برنامه دودویی

ریزبرنامه سمبلیک فرم مناسبی برای نوشتن ریزبرنامه‌هایی است که افراد می‌توانند بخوانند و بفهمند. ولی این روش ذخیره ریزبرنامه در حافظه نیست. برنامه سمبلیک باید با یکی از دو روش برنامه اسمبلی یا کاربر به دودویی ترجمه شود بشرطی که ریزبرنامه مانند این مثال ساده باشد.
 فرم دودویی معادل ریزبرنامه در جدول ۳-۷ لیست شده است. آدرس‌های حافظه کنترل به هر دو صورت دهمی و دودویی داده شده است. همانطور که در جدول ۱-۷ دیده شد، محتوای دودویی هر ریزدستورالعمل از سمبل‌ها و معادل‌های دودویی آنها بدست می‌آید.

دقت کنید که آدرس ۳ در ریزبرنامه سمبلیک فاقد معادل است زیرا روال ADD فقط سه ریز دستورالعمل در مکان‌های ۰، ۱ و ۲ دارد. روال بعدی از آدرس ۴ شروع می‌شود. هرچند آدرس ۳ بکارنرفته است ولی باید یک مقدار دودویی برای هر یک از کلمات حافظه کنترل تعیین شود. می‌توانیم مقدار تماماً ۰ را در این کلمه قرار دهیم زیرا این مکان هیچوقت بکار نخواهد رفت. با این وجود اگر خطای پیش‌بینی نشده‌ای رخ دهد، یا اگر یک سیگنال پارازیت^۲ (نویز) مقدار ۳ را در CAR قرار دهد، پرش به آدرس ۶۴ که ابتدای روال برداشت است عاقلانه است.

ریزبرنامه دودویی لیست شده در جدول ۳-۷ محتوای کلمات حافظه کنترل را مشخص می‌نماید. وقتی که از ROM برای حافظه کنترل استفاده شود، لیست دودویی ریزبرنامه جدول درستی را برای ساخت قطعه فراهم می‌آورد. این عمل ساخت یک فرآیند سخت‌افزاری است و با ایجاد ماسکی برای ROM که ۱‌ها و ۰‌ها را برای هر کلمه بوجود می‌آورد، تشکیل شده است. پس از آنکه اتصالات درونی در حین ساخت سوزانده شدند بیت‌های ROM تثبیت می‌گردند. ROM از بسته‌های مدار مجتمع، IC،

1- Edge- Triggered

2- Noise

جدول ۳-۷ بخشی از ریزبرنامه دودویی برای حافظه کنترل

ریزروال	آدرس		ریزدستورالعمل دودویی					
	دهدی	دودویی	F1	F2	F3	CD	BR	AD
ADD	0	0000000	000	000	000	01	01	1000011
	1	0000001	000	100	000	00	00	0000010
	2	0000010	001	000	000	00	00	1000000
	3	0000011	000	000	000	00	00	1000000
BRANCH	4	0000100	000	000	000	10	01	0000110
	5	0000101	000	000	000	00	00	1000000
	6	0000110	000	000	000	01	01	1000011
	7	0000111	000	000	110	00	00	1000000
STORE	8	0001000	000	000	000	01	01	1000011
	9	0001001	000	101	000	00	00	0001010
	10	0001010	111	000	000	00	00	1000000
	11	0001011	000	000	000	00	00	1000000
EXCHANGE	12	0001100	000	000	000	01	01	1000011
	13	0001101	001	000	000	00	00	0001110
	14	0001110	100	101	000	00	00	0001111
	15	0001111	111	000	000	00	00	1000000
FETCH	64	1000000	110	000	000	00	00	1000001
	65	1000001	000	100	101	00	00	1000010
	66	1000010	101	000	000	00	11	0000000
INDRCT	67	1000011	000	100	000	00	00	1000100
	68	1000100	101	000	000	00	10	0000000

ساخته شده و در صورت لزوم می توان آنها را تعویض کرد. برای اصلاح مجموعه دستورالعمل های کامپیوتر، لازم است یک ریزبرنامه جدید تولید و ROM جدیدی ماسک گردد. قطعه قبلی را برداشته و جدید را به جای آن قرار می دهیم.

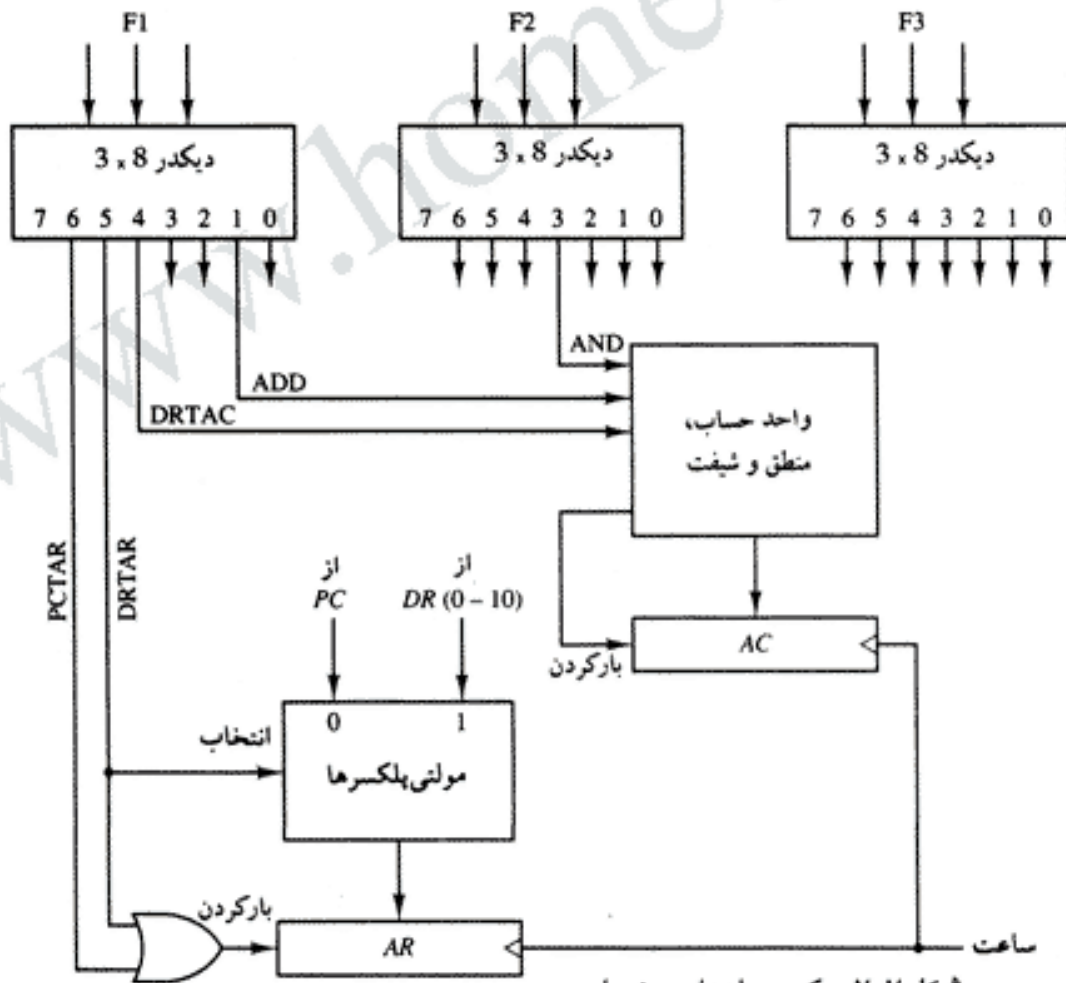
اگر بخواهیم حافظه کنترل قابل نوشتن باشد، ROM با RAM جایگزین می شود. مزیت بکارگیری RAM برای حافظه کنترل این است که ریزبرنامه بسادگی با نوشتن یک الگوی جدید از 1 ها و 0 ها بدون اعمال روش های سخت افزاری میسر است. قابلیت نوشتن در حافظه کنترل این انعطاف را ایجاد می کند که با تغییر ریزبرنامه تحت کنترل پردازنده مجموعه دستورالعمل های کامپیوتر را بطور پویا تغییر داد. با این وجود اکثر سیستم های ریزبرنامه نویسی شده از ROM برای حافظه کنترل استفاده می کنند زیرا از RAM ارزاتر و سریعتر است و نیز هر کاربری نمی تواند معماری سیستم را تغییر دهد.

۴-۷ طراحی واحد کنترل

بیت های ریزدستورالعمل معمولاً به میدانی هایی تقسیم می شود، که هر میدان کار جداگانه و مستقلی را بعهده دارد. انواع میدان های مختلفی که در قالب های دستورالعمل ها با آنها مواجه می شویم،

بیت کنترلی لازم را برای انجام ریز عمل اعمال در سیستم، بیت های خاصی را برای مشخص کردن نحوه محاسبه آدرس بعدی، و میدان آدرسی را برای انشعاب، در اختیار می گذارند. تعداد بیت های کنترلی را که موجب اجرای ریز عمل ها می شوند می توان با دسته بندی متغیرهای جدا از هم به شکل میدان های مختلف و کدگذاری K بیت از هر میدان برای مشخص کردن 2^k ریز عمل، کاهش داد. هر میدان نیاز به یک دیکدر برای ایجاد سیگنال های کنترلی مربوطه دارد. این عمل اندازه بیت های ریز دستورالعمل را کاهش می دهد ولی مستلزم سخت افزار اضافی در خارج از حافظه کنترل است. روش فوق زمان تأخیر سیگنال های کنترلی را نیز افزایش می دهد زیرا این سیگنالها باید از مدار دیکدر عبور نمایند.

انکد کردن بیت های کنترل در مثال برنامه نویسی بخش قبل نشان داده شد. 9 بیت میدان ریز عمل به سه میدان جزئی سه بیتی تقسیم شدند. خروجی حافظه کنترل هر میدان جزئی باید دیکد می شد تا ریز عمل های متمایز ایجاد می شدند. خروجی دیکدرها نیز به ورودی های مناسب در واحد پردازنده متصل می شدند. شکل ۷-۷ سه دیکدر و بعضی از اتصالاتی که باید از خروجی هایشان ساخته شود را نشان می دهد. هر یک از سه میدان ریز دستورالعمل موجود در خروجی حافظه کنترل با یک دیکدر 3×8 دیکد می شود



شکل ۷-۷ دیکدر میدان های ریز عمل

تا هشت خروجی را فراهم آورد. هر یک از این خروجی‌ها بایستی به مدار مناسبی برای آغاز ریزعمل مربوطه، مطابق جدول ۷-۱، متصل شود. مثلاً وقتی که $F_1 = 101$ (5 دودویی) باشد، گذر لبه پالس ساعت بعدی محتوای $DR(0 - 10)$ را به AR (که در جدول ۷-۱ با سمبل $DRTAR$ نشان داده شده است) منتقل می‌نماید. بطور مشابه اگر $F_1 = 110$ (6 دودویی) باشد، انتقالی از PC به AR (که با سمبل $PCTAR$ نشان داده شده است) صورت می‌گیرد. همانطور که در شکل ۷-۷ دیده می‌شود، خروجی‌های 5 و 6 دیکدر F_1 به ورودی بارکردن AR وصل شده‌اند بطوری که وقتی هر یک از خروجی‌ها فعال گردد، اطلاعات از مولتی‌پلکسرها به AR منتقل می‌گردند. مولتی‌پلکسرها اطلاعات را، وقتی که خروجی 5 فعال است از DR ، و وقتی این خروجی غیر فعال باشد از PC دریافت می‌نمایند. انتقال به AR ، با گذر لبه پالس ساعت و هنگامی که خروجی 5 یا 6 دیکدر فعال باشد انجام می‌شود. سایر خروجی‌های دیکدر که انتقال بین ثبات‌ها را انجام می‌دهند باید به روش مشابهی مورد بررسی قرار گیرند.

واحد حساب، منطق، شیفت را می‌توان مطابق شکل‌های ۱۹-۵ و ۲۰-۵ طراحی کرد. به جای استفاده از گیت‌ها در تولید سیگنالهای کنترلی AND ، ADD و DR در شکل ۱۹-۵، این ورودی‌ها بترتیب از خروجی دیکدرهای متناظر AND ، ADD و $DRTAC$ می‌آیند، شکل ۷-۷. سایر خروجی‌های دیکدرها که مربوط به عمل روی AC است باید به نحوی مشابه به واحد حساب، منطق، شیفت متصل گردد.

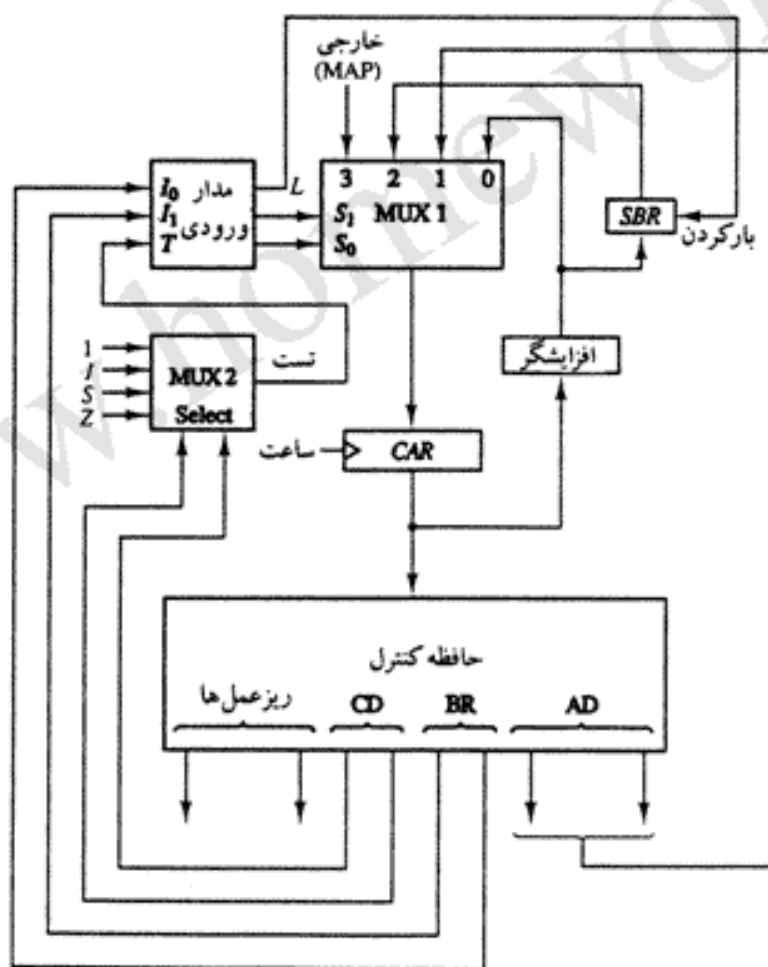
توالی‌گر ریزبرنامه

اجزاء اصلی واحد کنترل ریزبرنامه نویسی شده حافظه کنترل و مدارات انتخاب آدرس بعدی هستند. بخش انتخاب آدرس، توالی‌گر ریزبرنامه خوانده می‌شود. یک توالی‌گر ریزبرنامه را می‌توان با اجزاء دیجیتال، مناسب یک کاربرد خاص ساخت. با این وجود، همان‌طور که واحدهای ROM بزرگی به شکل بسته‌های مدار مجتمع وجود دارند، توالی‌گرهای همه منظوره هم وجود دارند که برای ساخت واحدهای کنترل ریزبرنامه نویسی شده مناسب‌اند. برای تضمین پذیرش وسیع، باید سازمان داخلی مدار مجتمع توالی‌گر بصورتی باشد که بتوان آن را در گستره وسیعی از کاربردها مورد استفاده قرار داد.

هدف از بکارگیری توالی‌گر ریزبرنامه، ارائه آدرس به حافظه کنترل است بطوری که یک ریزدستورالعمل از آن خوانده شده و اجرا گردد. مدار آدرس بعدی توالی‌گر، منبع آدرس بعدی، که در ثبات آدرس کنترل ذخیره خواهد شد را مشخص می‌کند. انتخاب منبع آدرس بوسیله بیت‌های اطلاعات آدرس بعدی که توالی‌گر آنها را از ریزدستورالعمل جاری دریافت می‌کند هدایت می‌شود. توالی‌گرهای تجاری در داخل خود دارای مجموعه‌ای از ثبات‌های داخلی هستند که برای ذخیره موقت آدرسها هنگام اجرای حلقه‌های ریزبرنامه و فراخوانی زیرروال‌ها بکار می‌روند. در بعضی از توالی‌گر یک ثبات خروجی وجود دارد که می‌توان از آن به عنوان ثبات آدرس برای حافظه کنترل استفاده کرد.

برای تشریح ساختار داخلی یک توالی‌گر نوعی ریزبرنامه، یک توالی‌گر خاصی که برای استفاده در ریزبرنامه کامپیوتر مناسب است و در بخش قبلی ارائه کردیم را نشان می‌دهیم. بلاک دیاگرام توالی‌گر ریزبرنامه در شکل ۸-۷ نمایش داده شده است. حافظه کنترل باین علت در شکل آمده است که

عکس العمل بین توالی گر و حافظه متصل به آن ملاحظه شود. در مدار دو مولتی پلکسر وجود دارد. اولین مولتی پلکسر آدرسی را از یکی از چهار منبع انتخاب کرده و آنرا به ثبات آدرس کنترل CAR هدایت می نماید. دومین مولتی پلکسر مقدار بیت وضعیت انتخاب شده را تست کرده و نتیجه تست را به یک مدار منطقی ورودی اعمال می نماید. خروجی CAR آدرس را برای حافظه کنترل تهیه می نماید. محتوای CAR افزایش یافته و به یکی از ورودی های مولتی پلکسر و نیز به ثبات زیر روال SBR اعمال می شود. سه ورودی دیگر مولتی پلکسر شماره ۱ از میدان آدرس ریز دستورالعمل جاری، از خروجی SBR، و از منبع خارجی که دستورالعمل را نگاشت می کند می آیند. اگر چه دیاگرام فقط یک ثبات زیر روال را نشان می دهد، یک توالی گر دارای پشته ای از ثباتها با عمق چهار تا هشت سطح می باشد. باین ترتیب، بطور همزمان چند زیرروال می توانند فعال باشند. اعمال پوش^۱ و پوپ^۲، به همراه اشاره گر



شکل ۸-۷ توالی گر ریز برنامه برای حافظه کنترل

1- Push

2- POP

پشته، آدرس برگشت را در حین اجرای ریزدستورالعمل های فراخوانی و بازگشت ذخیره و بازیابی می کنند. میدان CD (شرط) در ریزدستورالعمل، یکی از بیت های وضعیت را در مولتی پلکسر دوم انتخاب می کند. اگر بیت انتخاب شده 1 باشد، متغیر T (تست) برابر 1 خواهد بود؛ در غیر این صورت برابر 0 است. مقدار T به همراه دو بیت از میدان BR (انشعاب) به یک مدار منطقی ورودی می رود. مدار منطقی ورودی یک توالی گر خاص، نوع اعمالی که توالی گر می تواند انجام دهد را معین می کند. از جمله اعمال توالی گرها عبارتند از: افزایش، انشعاب یا پرش، فراخوانی و بازگشت از زیرروال، بارکردن آدرس خارجی، پوش یا پاپ پشته. سایر اعمال توالی گرهای تجاری سه یا چهار ورودی علاوه بر ورودی T دارند و لذا محدوده وسیعتری از عملیات را در اختیار می گذارند.

مدار منطقی ورودی شکل ۷-۸ دارای سه ورودی I_0 ، I_1 و T و سه خروجی S_0 ، S_1 و L می باشد. متغیرهای S_0 و S_1 یکی از منابع آدرس را برای CAR انتخاب می کنند. متغیر L ورودی بارکردن SBR را فعال می نماید. مقادیر دودویی دو متغیر، انتخاب مسیر موردنظر را در مولتی پلکسر معین می کنند. مثلاً، با $S_1 S_0 = 10$ ، ورودی شماره ۲ مولتی پلکسر انتخاب و مسیری بین SBR و CAR بوجود می آید. توجه کنید که هر یک از چهار ورودی و خروجی MUX1 دارای هفت بیت آدرس هستند.

جدول درستی مدار منطقی ورودی در جدول ۷-۴ نشان داده شده است. ورودی های I_0 و I_1 همان بیت های میدان BR هستند. تابع مشخص شده برای هر وارده از جدول ۷-۱ گرفته شده است. مقادیر بیت های S_0 و S_1 با توجه به عمل مورد نظر و مسیری که انتخاب لازم را برقرار می کند تعیین می شود. اگر شرط بیت وضعیت در حین اجرای ریز دستورالعمل فراخوانی ($BR = 01$) برقرار باشد ($T = 1$)، مقدار افزایش یافته CAR در ثبات زیرروال بار می شود. با استفاده از این جدول می توان توابع ساده شده را برای مدار منطقی ورودی بدست آورد.

$$S_1 = I_1$$

$$S_0 = I_1 I_0 + I_1' T$$

$$L = I_1' I_0 T$$

این مدار را می توان با سه گیت AND، یک OR و یک معکوس کننده ساخت.

جدول ۷-۴ جدول درستی مدار منطقی ورودی توالی شمار ریزبرنامه

BR میدان	ورودی			MUX 1		SBR بارکردن L
	I_1	I_0	T	S_1	S_0	
0 0	0	0	0	0	0	0
0 0	0	0	1	0	1	0
0 1	0	1	0	0	0	0
0 1	0	1	1	0	1	1
1 0	1	0	x	1	0	0
1 1	1	1	x	1	1	0

توجه کنید که مدار افزایشگر در توالی گر شکل ۷-۸ شماره‌ده‌ای که با فلیپ فلاپ ساخته شده باشد نیست. بلکه مداری ترکیبی است که با گیت ساخته شده است. یک مدار افزایشگر ترکیبی را می‌توان با اتصالی سری مدارهای نیم‌جمع‌کننده (شکل ۷-۸) طراحی کرد. رقم نقلی خروجی از یک طبقه باید به ورودی طبقه بعدی وصل شود. یکی از ورودی‌های کم‌ارزشترین طبقه باید برابر 1 باشد تا عمل افزایش یک واحدی صورت گیرد.

مسائل

- ۷-۱ تفاوت بین ریزپردازنده و ریزبرنامه چیست؟ آیا می‌توان ریزپردازنده‌ای بدون ریزبرنامه طراحی کرد؟ آیا همه کامپیوترهای ریزبرنامه‌نویسی شده ریزپردازنده هستند؟
- ۷-۲ تفاوت بین کنترل سخت‌افزاری و ریزبرنامه‌نویسی شده را بیان کنید. آیا می‌توان یک کنترل سخت‌افزاری را همراه با حافظه کنترل داشت؟
- ۷-۳ این اصطلاحات را تعریف کنید: (الف) ریزعمل؛ (ب) ریزدستورالعمل؛ (ج) ریزبرنامه؛ (د) ریزکد.
- ۷-۴ سازمان یک کنترل ریزبرنامه‌نویسی شده که در شکل ۷-۱ نشان داده شده است دارای تأخیرهای انتشاری زیر است. 40ns برای تولید آدرس بعدی، 10ns برای انتقال آدرس بداخل ثبات آدرس کنترل، 40ns برای دستیابی به حافظه کنترل، ROM، 10ns برای انتقال ریزدستورالعمل به ثبات داده کنترل، و 40ns برای اجرای ریزعمل مورد نظر که بوسیله کلمه کنترل مشخص شده است. حداکثر فرکانس پالس ساعتی که کنترل می‌تواند بکار برد چیست؟ اگر ثبات داده کنترل مورد استفاده قرار نگیرد، فرکانس ساعت چیست؟
- ۷-۵ سیستم شکل ۷-۲ از یک حافظه کنترل 1024 کلمه‌ای 32 بیتی استفاده می‌کند. ریزدستورالعمل‌ها دارای سه میدان هستند، میدان ریزعمل‌ها 16 بیتی است. (الف) میدان آدرس انشعاب چند بیتی است (ب) اگر 16 بیت وضعیت وجود داشته باشد، چند بیت منطق انشعاب برای انتخاب یک بیت وضعیت لازم است. (ج) چند بیت برای انتخاب ورودی مولتی‌پلکسر باقی مانده است
- ۷-۶ حافظه کنترل شکل ۷-۲ دارای 4096 کلمه 24 بیتی است. (الف) در ثبات آدرس کنترل چند بیت وجود دارد. (ب) هر یک از چهار ورودی که بداخل مولتی‌پلکسر می‌روند چند بیتی‌اند. (ج) تعداد ورودی‌های هر مولتی‌پلکسر چند است و چند مولتی‌پلکسر لازم است.
- ۷-۷ با استفاده از روش نگاشت که در شکل ۷-۳ توصیف شد، برای هر یک از کدهای عملیاتی زیر

اولین ریزدستورالعمل را بنویسید: (الف) 0010؛ (ب) 1011؛ (ج) 1111.

۷-۸ یک رویه نگاشت تنظیم کنید که هشت ریزدستورالعمل متوالی را برای هر روال در اختیار بگذارد. کد عمل شش بیتی است و حافظه کنترل 2048 کلمه دارد.

۷-۹ توضیح دهید که چگونه با استفاده از یک حافظه فقط خواندنی می توان عمل نگاشت از کد دستورالعمل به آدرس ریز دستورالعمل را انجام داد. مزیت این روش نسبت به روش شکل ۷-۳ چیست.

۷-۱۰ چرا در پیکربندی کامپیوتر شکل ۷-۴ به دو مولتی پلکسر نیاز داریم. آیا راه دیگری برای انتقال اطلاعات چند منبع به یک مقصد مشترک وجود دارد؟

۷-۱۱ با استفاده از جدول ۷-۱، برای ریزاعمال زیر، 9 بیت میدان ریزعمل را مشخص کنید.

(الف) $AC \leftarrow AC + 1$ و $DR \leftarrow DR + 1$

(ب) $PC \leftarrow PC + 1$ و $DR \leftarrow M[AR]$

(ج) $DR \leftarrow AC$ و $AC \leftarrow DR$

۷-۱۲ با استفاده از جدول ۷-۱، ریزاعمال سمبلیک زیر را به عبارات انتقال ثباتی و دودویی تبدیل کنید.

(الف) READ, INCPC

(ب) ACTDR, DRTAC

(ج) ARTPC, DRTAC, WRITE

۷-۱۳ فرض کنید که می خواهیم روال ADD، که در جدول ۷-۲ لیست شد را به دو ریزدستورالعمل زیر تبدیل کنیم:

ADD : READ I CALL INDR2

ADD U JMP FETCH

زیر روال INDR2 چه باید باشد

۷-۱۴ در زیر ریزبرنامه سمبلیک یک دستورالعمل در کامپیوتر بخش ۷-۳ آمده است

ORG 40

NOP S JMP FETCH

NOP Z JMP FETCH

NOP I CALL INDRCT

ARTPC U JMP FETCH

(الف) عملی که پس از اجرای دستورالعمل انجام می شود را مشخص کنید.

(ب) چهار ریزدستورالعمل را به فرم معادل دودویی آنها تبدیل کنید.

۷-۱۵ کامپیوتر بخش ۷-۳ دارای ریزبرنامه دودویی زیر است.

آدرس	ریزبرنامه دودویی																				
60	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1
61	1	1	1	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0
62	0	0	1	0	0	1	0	0	0	1	0	1	0	0	1	1	1	1	1	1	1
63	1	0	1	1	1	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0

الف) آنرا به ریزبرنامه سمبلیک، مطابق جدول ۷-۲، ترجمه کنید. (FETCH در آدرس 64 و INDRCT در آدرس 67 است.)

ب) تمام اشکالاتی را که با اجرای این ریزبرنامه در کامپیوتر پیش می آید بنویسید.

۷-۱۶ دستورات زیر را به کامپیوتر بخش ۷-۳ اضافه کنید (EA آدرس موثر است): برنامه سمبلیک را برای هر روال مطابق جدول ۷-۲ بنویسید. (توجه کنید که مقدار AC نباید تغییر کند مگر اینکه دستورالعمل تغییری در AC را مشخص نماید.)

سجل	کد عمل	شکل سمبلیک	شرح
AND	0100	$AC \leftarrow AC \wedge M[EA]$	AND
SUB	0101	$AC \leftarrow AC - M[EA]$	تفریق
ADM	0110	$M[EA] \leftarrow M[EA] + AC$	جمع با حافظه
BTCL	0111	$AC \leftarrow AC \wedge M[EA]$	صفر کردن بیت ها
BZ	1000	If (AC = 0) then (PC ← EA)	انشعاب در صورت صفر بودن AC
SEQ	1001	If (AC = M[EA]) then (PC ← PC + 1)	گذر در صورت برابری
BPNZ	1010	If (AC > 0) then (PC ← EA)	انشعاب اگر مثبت و غیرصفر

۷-۱۷ یک روال ریزبرنامه برای دستورالعمل ISZ (افزایش و گذر در صورت صفر بودن) که در فصل ۵ (جدول ۵-۴) تعریف شد بنویسید. از قالب دستورالعمل بخش ۷-۳ استفاده کنید. توجه کنید

که شرط وضعیت $DR = 0$ در میدان CD کامپیوتر بخش ۷-۳ وجود ندارد با این وجود می توانید AC و DR را با هم تعویض کرده و با بیت Z چک کنید که آیا $AC = 0$ است یا نه؟

۷-۱۸ روال های ریزبرنامه سمبلیک را برای دستورالعمل BSA (انشعاب و ذخیره آدرس) که در فصل ۵ تعریف شد بنویسید (جدول ۵-۴). از قالب ریزدستورالعمل بخش ۷-۳ استفاده کنید. تعداد ریزدستورالعمل ها را حداقل کنید.

۷-۱۹ نشان دهید که چگونه خروجی های 5 و 6 دیکدر F3 در شکل ۷-۷ به شماره برنامه PC وصل می شوند.

۷-۲۰ نشان دهید که چگونه یک میدان ۹ بیتی ریزعمل در ریزدستورالعمل را می توان به زیر

- میدان‌هایی تقسیم کرد تا 46 ریزعمل را مشخص نمایند؟ چند ریزعمل می‌توان در یک ریزدستورالعمل مشخص کرد؟
- ۷-۲۱ کامپیوتری دارای 16 ثبات، یک ALU با 32 عمل و یک شیفت دهنده با هشت عمل است که کلاً به یک سیستم گذرگاه مشترک متصل‌اند.
- الف) برای یک ریز عمل کلمه کنترلی بنویسید.
- ب) تعداد بیت‌ها را در هر میدان کلمه کنترل مشخص کنید و یک طرح انکد کردن کلی را ارائه نمائید.
- ج) بیت‌های کلمه کنترل ریزعمل $R_4 \leftarrow R_5 + R_6$ را نشان دهید.
- ۷-۲۲ فرض کنید که منطق ورودی توالی‌گر ریزبرنامه شکل ۷-۸ چهار ورودی I_0, I_1, I_2 و I_0 (تست) و سه خروجی S_0, S_1, S_2 را داراست. عملیات اجرا شده در واحد در جدول زیر لیست شده است. یک مدار منطقی ورودی با استفاده از حداقل گیت‌ها طراحی کنید.

I_2	I_1	I_0	عمل
0	0	0	افزایش CAR اگر $T=1$ ، پرش به AD اگر $T=0$
x	0	1	پرش غیرشرطی به AD
1	0	0	افزایش غیرشرطی CAR
0	1	0	پرش به AD اگر $T=1$ ، افزایش CAR اگر $T=0$
1	1	0	فراخوانی زیرروال اگر $T=1$ ، افزایش CAR اگر $T=0$
0	1	1	بازگشت غیرشرطی از زیرروال
1	1	1	نگاشت غیرشرطی آدرس خارجی

- ۷-۲۳ یک افزایشگر ترکیبی ۷ بیتی برای توالی‌گر ریزبرنامه شکل ۷-۸ طراحی کنید (شکل ۷-۸). با اضافه کردن یک ورودی کنترل D افزایشگر را تکمیل کنید. وقتی $D = 0$ است، مدار یک واحد افزایش می‌یابد، ولی وقتی $D = 1$ است مدار دو واحد افزایش می‌یابد.
- ۷-۲۴ بین MUX2 و مدار منطقی ورودی در شکل ۷-۸ یک گیت OR انحصاری اضافه کنید. یکی از ورودی‌های گیت از خروجی تست مولتی‌پلکسر می‌آید. ورودی دیگر گیت از بیتی با نام P^2 (بمعنی قطبیت) در ریزدستورالعمل حاصل از حافظه کنترل می‌آید. خروجی گیت به ورودی T در مدار منطقی ورودی متصل می‌شود. کنترل P چه کاری انجام می‌دهد.

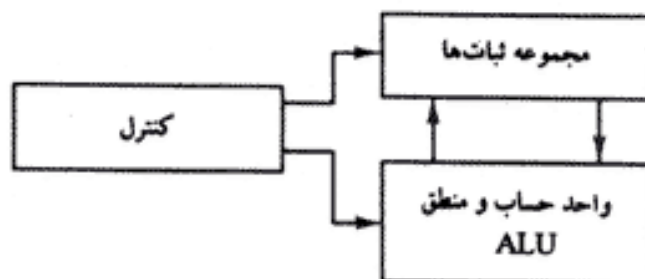
واحد مرکزی پردازش



- ۸-۱ مقدمه
- ۸-۲ سازمان ثبات‌های عمومی
- ۸-۳ سازمان پشته
- ۸-۴ قالب دستورالعمل‌ها
- ۸-۵ روشهای آدرس دهی
- ۸-۶ انتقال و دستکاری داده‌ها
- ۸-۷ کنترل برنامه
- ۸-۸ کامپیوتر کم دستور RISC

۸-۱ مقدمه

بخشی از کامپیوتر که عمده پردازش داده‌ها را انجام می‌دهد واحد مرکزی پردازش نام داشته و آنرا CPU نیز می‌خوانند. CPU از سه بخش عمده تشکیل یافته است، شکل ۸-۱. مجموعه ثبات‌ها، داده‌های میانی که در حین اجرای دستورالعمل بکار می‌روند را ذخیره می‌نمایند. واحد حساب و منطق (ALU) ریزاعمال لازم برای اجرای دستورات را انجام می‌دهد. واحد کنترل مدیریت انتقال داده‌ها را در بین ثبات بعهده داشته و ALU را در انجام اعمال هدایت می‌کند.



شکل ۸-۱ اجزاء مهم یک CPU

CPU انواع متعددی از توابع^۱ را که توسط نوع خاصی دستور دیکته می شود اجرا می نماید. گاهی اوقات معماری^۲ کامپیوتر بوسیله برنامه نویس، که دستورات زبان ماشین را بکار می برد، بعنوان ساختار و رفتار کامپیوتر تعریف می شود. این ساختار و رفتار قالب دستورالعمل ها، روش های آدرس دهی، مجموعه دستورالعمل ها و سازمان کلی ثبات های CPU است.

یکی از مرزهای مشترک دیدگاه های طراح کامپیوتر و برنامه نویس از ماشین، بخشی از CPU است که به مجموعه دستورات مربوط می شود. از نظر طراحان، مجموعه دستورالعمل ها، مشخصات طراحی CPU را فراهم می آورند. بخش عمده طراحی یک CPU انتخاب سخت افزار برای اجرای دستورات ماشین است. کاربری که کامپیوتر را در زبان ماشین یا اسمبلی برنامه ریزی می کند باید از مجموعه ثبات ها، ساختمان حافظه، نوع داده هایی که دستورالعمل ها می توانند با آنها کار می کنند و عملی که هر دستور اجرا می نماید آگاه باشد. مثال هایی از CPU های ساده در فصل ۵ و ۷ آورده شد. این فصل سازمان و معماری یک CPU را با تاکید به نظرات کاربر کامپیوتر توصیف می کند. ابتدا بطور مختصر چگونگی ارتباط ثبات ها را با ALU از طریق گذرگاه ها توضیح داده و طرز کار پشته حافظه را تشریح می کنیم. سپس انواع قالب های دستورات موجود، روش های آدرس دهی بکار رفته در بازیابی داده ها از حافظه و انواع دستوراتی که عموماً در کامپیوترها دیده می شوند را نشان خواهیم داد. آخرین بخش مفهوم کامپیوترهای کم دستور^۳ (RISC) را بیان دارد.

۲-۸ سازمان ثبات های عمومی

در مثال برنامه نویسی فصل ۶ دیدیم که مکان های حافظه برای ذخیره کردن اشاره گرها، شمارنده ها، آدرس برگشت نتایج موقت و حاصل ضرب جزئی^۴ در ضرب لازم هستند. ارجاع به مکان های حافظه برای اهداف فوق، چیزی جز اتلاف وقت نیست، زیرا دستیابی به حافظه وقت گیرترین عمل در یک کامپیوتر است. بهترین و مفیدترین راه برای نگهداری این مقادیر میانی ذخیره کردن آنها در ثبات های پردازشگر است. وقتی که تعداد زیادی ثبات در CPU در نظر گرفته شد، بهترین راه بهره گیری از آنها ارتباطشان از طریق یک سیستم گذرگاه مشترک است. این ثبات ها تنها در انتقال مستقیم داده ها با هم ارتباط برقرار نمی کنند بلکه در حین اجرای انواع ریزعمل ها هم بهم وصل می شوند. از اینرو لازم است واحد مشترکی را فراهم آوریم تا تمام ریزاعمال حسابی، منطقی و شیفت را در پردازشگر اجرا نماید.

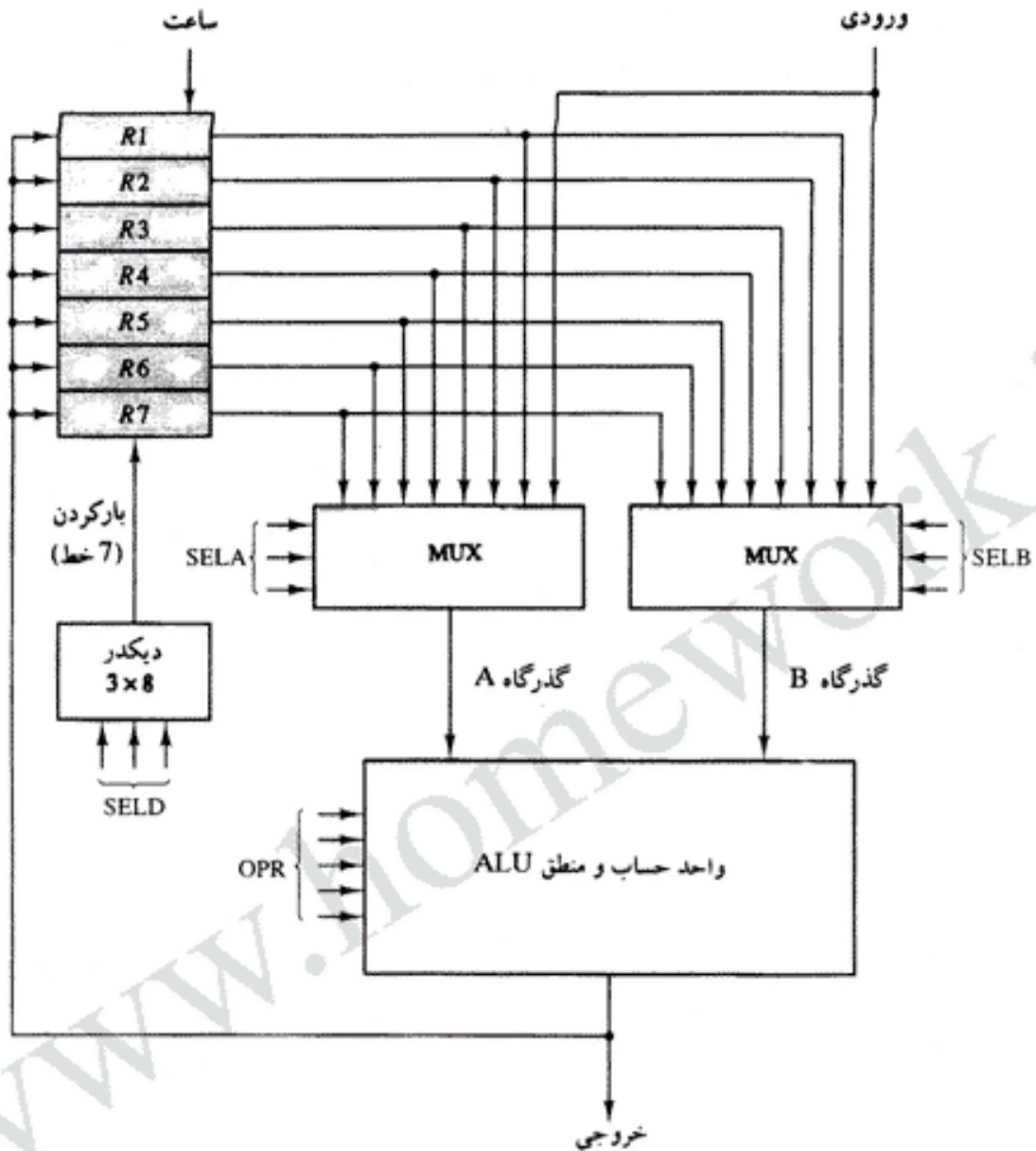
یک سازمان گذرگاه برای هفت ثبات CPU در شکل ۲-۸ نشان داده شده است. خروجی هر ثبات به دو مولتی پلکسر (MUX) وصل شده تا دو گذرگاه A و B را بسازد. خطوط انتخاب در هر مولتی پلکسر، یک ثبات یا داده ورودی را برای گذرگاه خاص انتخاب می نمایند. گذرگاه های A و B ورودی های واحد حساب - منطق (ALU) مشترک را تشکیل می دهند. عمل انتخاب شده در ALU

1- Function

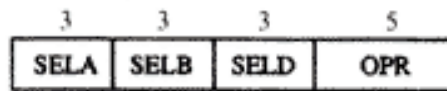
2- Architecture

3- Reduced Instruction Set Computer

4- Partial Product



(الف) بلاک دیاگرام



(ب) کلمه کنترل

شکل ۲-۸ مجموعه ثبات‌ها با ALU مشترک

معین‌کننده ریز عمل‌های حسابی منطقی است که باید اجرا شوند. نتایج ریز عمل‌ها هم در خروجی و یا در ورودی همه ثبات‌ها قابل دسترسی است. ثباتی که اطلاعات را از گذرگاه خروجی دریافت می‌کند توسط یک دیکدر انتخاب می‌شود. دیکدر ورودی بارکردن، ثبات مزبور را فعال کرده و لذا یک مسیر

انتقال را بین داده در گذرگاه خروجی و ورودی ثبات انتخاب شده برقرار می‌سازد. واحد کنترلی که سیستم گذرگاه CPU را اداره می‌کند، جریان اطلاعات بین ثبات‌ها و ALU را با انتخاب بخش‌های مختلف جهت می‌بخشد. مثلاً برای اجرای عمل

$$R_1 \leftarrow R_2 + R_3$$

کنترل باید متغیرهای انتخاب دودویی مناسب را برای ورودی‌های انتخاب کننده زیر تهیه نماید

۱- انتخاب کننده مولتی پلکسر A (SELA): برای قرار دادن محتوای R_2 روی گذرگاه A

۲- انتخاب کننده مولتی پلکسر B (SELB): برای قرار دادن محتوای R_3 روی گذرگاه B

۳- انتخاب‌گر عملیات در ALU (OPR): برای انجام عمل جمع حسابی $A + B$

۴- انتخاب‌گر دیکدر مقصد (SELD): برای انتقال محتویات گذرگاه خروجی به R_1

چهار متغیر کنترل فوق در واحد کنترل تولید و باید در آغاز یکی از پالس‌های ساعت در دسترس باشند. در طول یک سیکل ساعت، داده‌ها از دو ثبات منبع از طریق گیت‌های مولتی پلکسرها به ALU، سپس به گذرگاه خروجی و از آنجا به ورودی ثبات‌های مقصد می‌رسند. بنابراین با رسیدن پالس ساعت بعدی اطلاعات دودویی از گذرگاه خروجی به ثبات R_1 وارد می‌شود. برای دستیابی به یک واکنش سریع، ALU از مدارات سریع العمل ساخته شده است. گذرگاه‌ها از مولتی پلکسرها و گیت‌های سه حالت مطابق آنچه در بخش (۳-۴) دیده شد پیاده سازی شده‌اند.

کلمه کنترل

در سیستم، چهارده ورودی انتخاب دودویی وجود دارد که ترکیب مقدار عددی آنها یک کلمه کنترل^۱ را تشکیل می‌دهد. کلمه کنترل 14 بیتی در شکل ۲-۸ (ب) تعریف شده است. این کلمه از چهاربخش تشکیل می‌شود. سه بخش از آن هر یک سه بیت و یک بخش پنج بیتی است. سه بیت SELA یک ثبات مبدأ را برای ورودی A از واحد ALU انتخاب می‌کنند. سه بیت مربوط به SELB هم یک ثبات دیگر را برای ALU از طریق ورودی B انتخاب می‌نمایند. سه بیت SELD ثبات مقصد را بکمک دیکدر و هفت خروجی مربوطه‌اش انتخاب می‌نمایند. پنج بیت OPR یکی از عملیات ALU را برمی‌گزینند. چهارده بیت کلمه کنترل فوق وقتی کلاً به ورودی‌های انتخاب اعمال شوند یک ریزعمل خاصی را مشخص خواهند کرد.

جدول (۱-۸) کد انتخاب ثبات‌ها را مشخص می‌نماید. سه بیت دودویی ستون اول کد در جدول کد دودویی هر یک از سه میدان را مشخص می‌کند. ثباتی که توسط میدان SELA و SELB و SELD انتخاب می‌شود عدد دهنده‌ی مربوطه‌اش معادل با عدد دودویی کد مربوطه‌اش می‌باشد. هرگاه SELA

1- Control Word

جدول ۸-۱ تخصیص کد به میدانهای انتخاب ثباتها

کد دودویی	SELA	SELB	SELD
000	ورودی	ورودی	هیچ
001	R1	R1	R1
010	R2	R2	R2
011	R3	R3	R3
100	R4	R4	R4
101	R5	R5	R5
110	R6	R6	R6
111	R7	R7	R7

یا SELB برابر 000 باشد، مولتی پلکسر مربوطه داده را از ورودی خارجی می‌پذیرد. وقتی SELD=000 باشد هیچیک از ثبات‌ها انتخاب نشده ولی محتوای گذرگاه روی خروجی خارجی قرار می‌گیرد. ALU اعمال حسابی و منطقی را انجام می‌دهد. بعلاوه CPU باید قابلیت انجام شیفت را داشته باشد. مدار شیفت دهنده می‌تواند در ورودی ALU قرار گرفته و امکان پیش شیفت^۱ را فراهم آورد و یا پس از ALU بوده و امکان پس شیفت^۲ را بوجود آورد. در برخی مواقع این عمل را خود ALU انجام می‌دهد. یک مدار حساب، منطق و شیفت در بخش (۴-۷) طراحی شد. جدول عملیات این ALU در جدول (۴-۸) دیده شد. کدگذاری^۳ عملهای ALU برای CPU از بخش (۴-۷) گرفته شده و در جدول (۸-۲) آورده شده است. میدان OPR از پنج بیت تشکیل شده و هر عمل با یک نام سمبلیک مشخص شده است.

جدول ۸-۲ تخصیص کد به اعمال ALU

OPR انتخابگر	عمل	سبل
00000	انتقال A	TSFA
00001	افزایش A	INCA
00010	جمع A+B	ADD
00101	تفریق A-B	SUB
00110	کاهش A	DECA
01000	AND ثبات‌های A و B	AND
01010	OR ثبات‌های A و B	OR
01100	XOR ثبات‌های A و B	XOR
01110	متمم کردن A	COMA
10000	شیفت A به راست	SHRA
11000	شیفت A به چپ	SHLA

1- Preshift

2- Postshift

3- Encode

مثال هایی از ریزعمل ها

برای تعیین یک ریزعمل در CPU به یک کلمه کنترل 14 بیتی نیاز است. کلمه کنترل مربوط به یک ریزعمل معین را می توان از متغیرهای انتخاب بدست آورد. مثلاً برای ریز عمل تفریق که با عبارت زیر مشخص می شود:

$$R_1 \leftarrow R_2 - R_3$$

R_2 را برای ورودی A از ALU، R_3 را برای ورودی B از ALU و R_1 را بعنوان ثبات مقصد و عمل تفریق $A-B$ را در ALU مشخص می نماید. بنابراین کلمه کنترل از چهارمیدان تشکیل و هر میدان از لیست کدهای جدول (۸-۲) و (۸-۳) بدست می آید. کلمه کنترل دودویی برای ریزعمل تفریق برابرست با 010 011 001 00101 که بطریق زیر بدست می آید.

بخش:	SELA	SELB	SELE	OPR
سمبل:	R_2	R_3	R_1	SUB
کلمه کنترل:	010	011	001	00101

کلمه کنترل برای تفریق و چند دستور دیگر در جدول (۸-۳) آورده شده است.

ریزعمل های افزایش و انتقال، ورودی B از ALU را بکار نمی برند. در این موارد برای میدان B خط تیره (-) قرار داده می شود. برای هر میدان بکار نرفته، مانند فوق، در کلمه کنترل عدد دودویی 000 را تخصیص خواهیم داد هر چند که هر عدد دودویی دیگر نیز قابل استفاده است. برای قرار دادن محتوای هر ثبات در خروجی، محتوا را در ورودی A از ALU قرار می دهیم ولی هیچیک از ثبات ها برای پذیرش داده انتخاب نمی شوند. عمل TSFA در ALU، داده را از ثبات به ALU و از آن جا به خروجی هدایت می کند. انتقال مستقیم از ورودی به خروجی توسط کلمه کنترلی تماماً 0 (برای میدان B هم 000 در نظر گرفته شود) انجام می گردد. یک ثبات هم می تواند با انجام عمل OR

جدول ۸-۳ مثالهایی از ریزعمل های CPU

ریزعمل	مشخص کردن سمبل				کلمه کنترل
	SELA	SELB	SELD	OPR	
$R_1 \leftarrow R_2 - R_3$	R_2	R_3	R_1	SUB	010 011 001 00101
$R_4 \leftarrow R_4 \vee R_5$	R_4	R_5	R_4	OR	100 101 100 01010
$R_6 \leftarrow R_6 + 1$	R_6	—	R_6	INCA	110 000 110 00001
$R_7 \leftarrow R_1$	R_1	—	R_7	TSFA	001 000 111 00000
Output $\leftarrow R_2$	R_2	—	None	TSFA	010 000 000 00000
Output \leftarrow Input	Input	—	None	TSFA	000 000 000 00000
$R_4 \leftarrow \text{shl } R_4$	R_4	—	R_4	SHLA	100 000 100 11000
$R_5 \leftarrow 0$	R_5	R_5	R_5	XOR	101 101 101 01100

انحصاری^۱ (XOR) صفرشود، زیرا $x \oplus x=0$

باتوجه به این مثالها واضح است که ریزعملهای بسیار دیگری را نیز در CPU می توان ایجاد کرد. یک روش مناسب تر در تولید کلمات کنترل با بیت های متعدد ذخیره سازی آنها در یک واحد حافظه است. این حافظه که کلمات کنترل را ذخیره می کند حافظه کنترل خوانده می شود. با خواندن متوالی کلمات کنترل می توان رشته ریزعمل های دلخواه را برای CPU ایجاد کرد. این نوع کنترل به کنترل ریزبرنامه نویسی^۲ شده معروف است. این نوع واحد کنترل در شکل (۷-۸) نشان داده شد. کلمه دودویی کنترل برای CPU از خروجی هائی که در حافظه کنترل با "ریزعملها" مشخص شده می آیند.

۳-۸ سازمان پشته

یک ویژگی مفید که در CPU اغلب کامپیوترها وجود دارد پشته یا لیست اولین ورودی - آخرین خروجی^۳ (LIFO) است. پشته یک وسیله ذخیره سازی است که اطلاعات را بنحوی ذخیره می کند که همیشه آخرین کمیت ذخیره شده اولین کمیت بازبایی شوند می باشد. عمل یک پشته را می توان با رویهم چیدن سینی ها مقایسه کرد. آخرین سینی در بالای مجموعه سینی ها اولین سینی برداشته شده از آن خواهد بود. در یک کامپیوتر دیجیتال، پشته اصولاً یک واحد حافظه است به همراه ثبات آدرسی که تنها می تواند بشمارد. (پس از یک مقدار اولیه ای که می توان در آن بازکرد). ثباتی که آدرس پشته را نگهداری می نماید اشاره گر پشته^۴ نامیده می شود (SP)، زیرا مقدار آن همیشه به بالاترین کمیت در پشته اشاره می کند. برخلاف پشته ای از سینی ها که در آن خود سینی را می توان خارج یا داخل پشته کرد، ثبات های فیزیکی پشته فقط همواره برای خواندن و نوشتن در دسترس قرار دارند. در واقع کلمه است که می تواند در پشته درج یا حذف شود.

دو عمل پشته عبارتند از درج یا حذف کمیت. عمل درج داده پوش^۵ (فشار دادن یا هول دادن) نامیده می شود زیرا عمل را می توان بصورت فشار دادن یک کمیت جدید در بالای آن تصور کرد. عمل حذف را پاپ^۶ (کشیدن یا برداشت) گویند زیرا عمل را می توان بصورت حذف یک کمیت از بالای آن در نظر گرفت. با این وجود چیزی بداخل پشته فشار داده و یا کشیده نمی شود. این اعمال با افزایش یا کاهش ثبات اشاره گر پشته شبیه سازی می گردند.

پشته های ثباتی

یک پشته می تواند دربخشی از یک حافظه بزرگ یا بصورت مجموعه ای از تعداد معینی حافظه یا

1- Exclusive OR

2- Microprogrammed Control

3- Last - In , First - Out

4- Stack Pointor

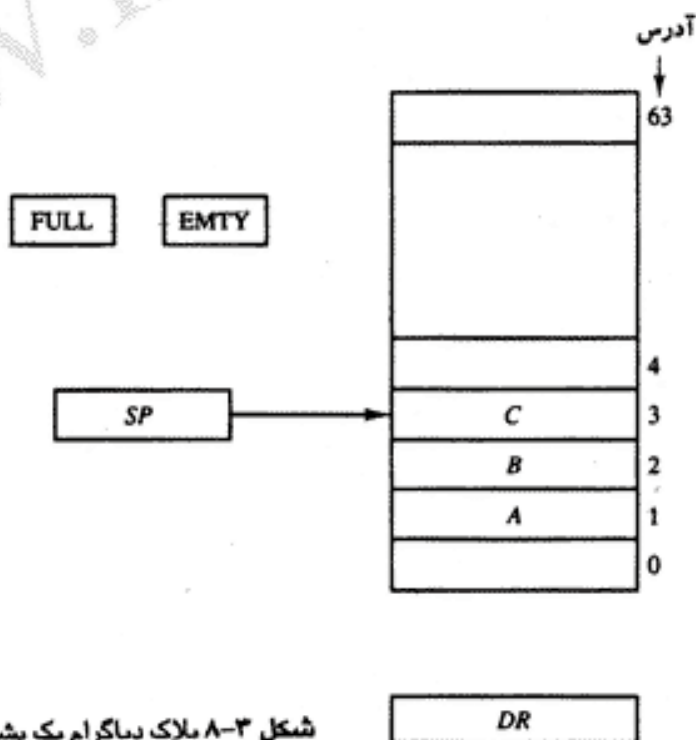
5- Push

6- Pop

ثبات قرار داشته باشد. شکل ۳-۸ سازمان یک پشته ثابتی ۶۴ کلمه‌ای را نشان می‌دهد. محتوای اشاره گر SP را عددی دودویی تشکیل می‌دهد که مقدار آن برابر آدرس کلمه‌ای است که در آن لحظه در بالای پشته قرار دارد. سه کمیت A و B و C برترتیب در پشته قرار گرفته‌اند. کمیت C در بالای پشته واقع و لذا محتوای SP برابر ۳ است. برای برداشتن کمیت فوقانی، پشته با خواندن، کلمه حافظه را از آدرس ۳ برداشته و محتوای SP را یک واحد کم می‌کند. حال کمیت B در بالای پشته است زیرا اشاره گر پشته آدرس ۲ را در خود دارد. برای وارد کردن کمیت جدید، با نوشتن در پشته اشاره گر SP یک واحد اضافه می‌شود. توجه کنید که هرچند کمیت C خوانده شد ولی بطور فیزیکی از بین نرفته است. این مطلب اهمیتی ندارد زیرا وقتی در پشته چیزی نوشته شود، کمیت جدید در همان محل نوشته می‌شود.

در یک پشته ۶۴ کلمه‌ای، اشاره گر پشته ۶ بیتی است زیرا $2^6 = 64$ است. چون اشاره گر SP تنها دارای ۶ بیت است نمی‌تواند از ۶۳ (دودویی ۱۱۱۱۱۱) تجاوز کند، و لذا SP فقط می‌تواند شش رقم پائین رتبه را در خود جای دهد. بطور مشابه وقتی ۰۰۰۰۰۰ یک واحد کم شود نتیجه ۱۱۱۱۱۱ خواهد بود. هرگاه پشته پر باشد، ثبات یک بیتی FULL برابر ۱ می‌شود و هر وقت پشته خالی باشد، ثبات یک بیتی EMTY برابر ۱ می‌گردد. DR ثباتی است که آنچه باید در پشته نوشته شود و یا از آن خوانده شود را در خود نگه می‌دارد.

در ابتدا که SP پاک می‌شود، EMTY برابر ۱ و FULL هم ۰ می‌گردد بدین ترتیب اشاره گر پشته به آدرس ۰ اشاره نموده و پشته خالی است. اگر پشته پر نباشد (FULL=۰) یک کمیت جدید با عمل پوش



شکل ۳-۸ بلاک دیگرام یک پشته ۶۴ کلمه‌ای

وارد آن می شود. عمل پوش با ریز اعمال زیر صورت می گیرد:

$SP \leftarrow SP + 1$	افزایش اشاره گر پشته
$M[SP] \leftarrow DR$	نوشتن داده در مکان بالای داده
$If(SP = 0) \text{ then } (FULL \leftarrow 1)$	آیا پشته پر است یا نه
$EMPTY \leftarrow 0$	علامت خالی نبودن پشته

اشاره گر پشته برای اشاره به کلمه بالاتر افزایش می یابد. سپس اجرای عمل نوشتن در حافظه سبب انتقال DR به مکان بالای پشته می گردد. توجه کنید که SP حاوی آدرس مکان بالای پشته است و $M[SP]$ کلمه ای از حافظه است که توسط آدرس فعلی موجود در اشاره گر پشته SP مشخص می گردد. اولین کمیت ذخیره شده در پشته در آدرس 1 قرار دارد. آخرین کمیت در آدرس 0 ذخیره شده است. اگر اشاره گر پشته به 0 برسد، پشته پر است و بنابراین $FULL=1$ خواهد بود. این وضعیت هنگامی رخ می دهد که بالاترین کمیتی که متعلق به آخرین پوش است در مکان 63 قرار گرفته باشد، و پس از افزایش SP آخرین کمیت جدید در مکان 0 ذخیره خواهد شد. به محض ذخیره شدن این کمیت اخیر در مکان 0 حافظه، دیگر محل خالی وجود ندارد. اگر کمیتی در پشته نوشته شده باشد مسلماً دیگر پشته خالی نیست و لذا $EMPTY=0$ می گردد. اگر پشته خالی نباشد (یعنی $EMPTY=0$) یک کمیت از پشته حذف می شود. عمل پاپ از ریز عملهای زیر تشکیل می شود.

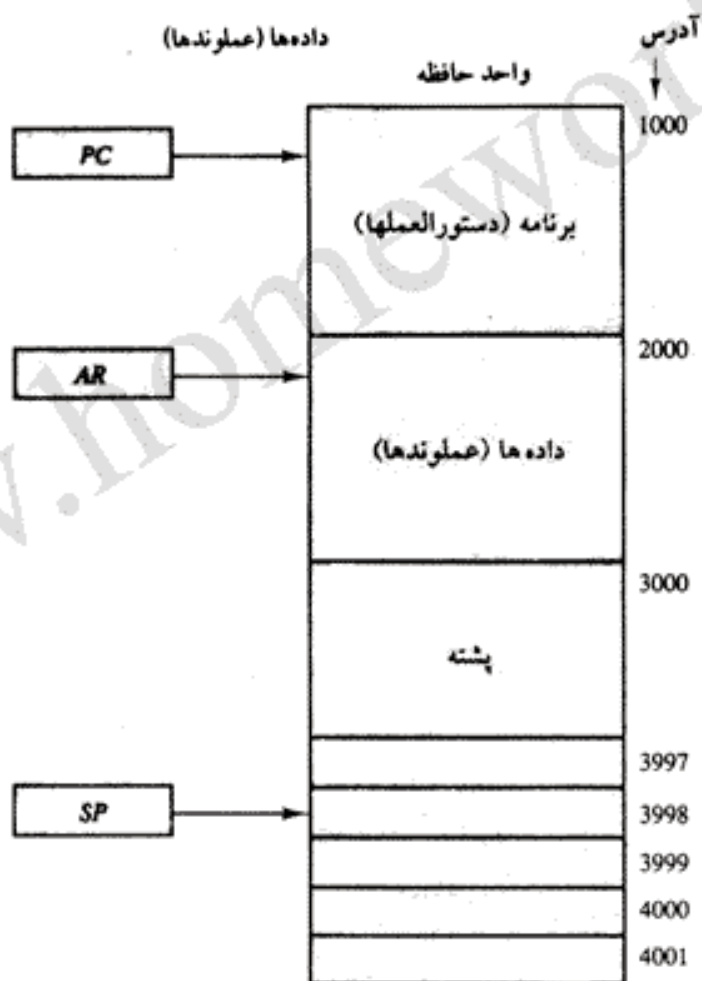
$DR \leftarrow M[SP]$	خواندن داده از مکان بالای پشته
$SP \leftarrow SP - 1$	کاهش اشاره گر پشته
$If (SP=0) \text{ then } (EMPTY \leftarrow 1)$	آیا پشته خالی است
$FULL \leftarrow 0$	علامت پر نبودن پشته

کمیت بالای پشته از پشته به DR منتقل می شود. سپس اشاره گر پشته کاهش می یابد. اگر مقدار آن به صفر برسد، پشته خالی است و $EMPTY = 1$ می شود. اگر وضعیت هنگامی رخ دهد که کمیت خوانده شده در مکان 1 باشد به محض خوانده شدن این کمیت، SP کاهش یافته و به 0 می رسد که مقدار اولیه SP بوده است. توجه کنید که اگر یک عمل پاپ کمیتی را از مکان 0 بخواند و سپس SP کاهش یابد، به SP به 111111 تغییر می یابد که معادل عدد دهدهی 63 است. در این آرایش، کلمه موجود در آدرس 0 آخرین کمیت پشته را دریافت می نماید. همچنین توجه کنید که اگر در $FULL=1$ پشته پوش شود یا در $EMPTY=1$ پشته پاپ شود خطای عملیاتی رخ می دهد.

پشته حافظه ای

پشته می تواند بعنوان یک واحد مستقل طبق شکل ۳-۸ وجود داشته باشد و یا در یک حافظه

RAM متصل به CPU پیاده سازی شود. پیاده سازی پشته در CPU بدین ترتیب صورت می گیرد که بخشی از حافظه به کمک یک ثبات پردازشگر به پشته اختصاص می یابد. شکل ۴-۸ بخشی از حافظه کامپیوتر را که به سه قسمت شده نشان می دهد: این سه قسمت به برنامه، داده و پشته تخصیص یافته است. شمارنده برنامه PC به آدرس دستور بعدی اشاره می کند. ثبات آدرس AR به آرایه ای از داده ها اشاره می کند. اشاره گر پشته نیز به بالای پشته اشاره می نماید. سه ثبات به یک گذرگاه آدرس مشترک متصلند و همواره یکی از آنها می تواند آدرسی را برای حافظه فراهم آورد. PC در طول فاز برداشت^۱ برای خواندن یک دستورالعمل بکار می رود. AR در طول فاز اجرا برای خواندن عملوند بکار گرفته می شود. SP نیز بهنگام پوش یا پاپ کمیت ها را به پشته می فرستد یا از آن برمی دارد.



شکل ۴-۸ حافظه کامپیوتر با
قطعه های برنامه، داده و پشته

DR

1- Fetch

همانطور که در شکل ۴-۸ ملاحظه می شود، مقدار اولیه SP برابر 4001 بوده و پشته با کاهش آدرس پرمی شود. بنابراین اولین کمیت در آدرس 4000 ذخیره می گردد، دومین کمیت در 3999 و بالاخره آخرین کمیت می تواند در آدرس 3000 ذخیره گردد. در اینجا فرض می کنیم که کمیت ها در پشته با ثبات داده DR ارتباط برقرار می کنند. کمیت جدید با عمل پوش طبق عبارات زیر وارد می شود.

$$SP \leftarrow SP - 1$$

$$M[SP] \leftarrow DR$$

دیده می شود که اشاره گر کاهش یافته و به آدرس کلمه بعدی اشاره می نماید. عمل نوشتن در حافظه هم کلمه را از DR وارد قسمت فوقانی پشته می نماید. یک کلمه نیز با عمل پاپ طبق عبارات زیر پاک می شود.

$$DR \leftarrow M[SP]$$

$$SP \leftarrow SP + 1$$

کمیت فوقانی از پشته خوانده شده و در DR قرار می گیرد. سپس اشاره گر پشته یک واحد اضافه شده و به مکان بعدی در پشته اشاره می نماید.

بیشتر کامپیوترها، سخت افزاری برای سرریز شدن^۱ پشته (پرترشدن از حد اکثر) و یا فروریز^۲ (خالی ترشدن از حداقل) ندارند. حدود پشته می تواند با دو ثبات پردازشگر چک شود: یکی از آنها تا حد بالا (در این حالت 3000) و دیگری حد پائین (در این حالت 4000) را نگه میدارند. پس از یک عمل پوش، SP با ثبات حد بالا و پس از پاپ با ثبات حد پائین مقایسه می شود.

دو ریز عمل لازم برای پوش یا پاپ عبارتند از (۱) دستیابی به حافظه توسط SP و (۲) بهنگام کردن^۳ SP. اینکه کدام ریز عمل ابتدا انجام می شود و یا اینکه تصحیح SP با افزایش و یا کاهش صورت می گیرد بستگی به سازمان پشته دارد. در شکل ۴-۸ گسترش پشته با کاهش آدرس حافظه همراه است. پشته می تواند با افزایش آدرس هم مطابق شکل ۳-۸ گسترش یابد. در این حالت SP برای پوش افزایش یافته و برای پاپ کاهش می یابد. می توان پشته را طوری طراحی کرد که SP به مکان خالی بعدی در بالای پشته اشاره کند. در این حالت ترتیب اعمال جزیی با یکدیگر تعویض می شوند.

اشاره گر پشته ابتدا با یک مقدار اولیه ای بار می شود. این مقدار اولیه باید آدرس پایین یک پشته در حافظه باشد. از آن پس SP بطور اتوماتیک با هر پوش یا پاپ افزایش یا کاهش می یابد. مزیت یک پشته حافظه ای این است که CPU می تواند بدون تعیین آدرس از آن استفاده نماید زیرا آدرس همیشه بطور اتوماتیک در اشاره گر بهنگام (آماده) است

1- Overflow

2- Under flow

3- Update

نمایش لهستانی معکوس^۱

سازمان پشته برای ارزیابی عبارات حسابی بسیار مفید است. روشهای ریاضی معمول در نوشتن عبارات حسابی وقتی توسط یک کامپیوتر ارزیابی شوند مشکلاتی را ایجاد می نمایند. عبارات حسابی معمول بصورت نمایش میانوندی^۲ نوشته شده و عملگر بین دو عملوند قرار می گیرد. عبارت ساده حسابی زیرا را در نظر بگیرید

$$A * B + C * D$$

علامت ستاره (که در واقع ضرب می باشد) بین دو عملوند A و B یا C و D قرار گرفته است. علامت جمع بین دو حاصلضرب قرار گرفته است. برای محاسبه این عبارت لازم است $A * B$ را بدست آورده، حاصلضرب را ذخیره و پس از انجام $C * D$ مجموع دو حاصلضرب را بدست آوریم. از این مثال می بینیم در محاسبه عبارات میانوندی لازم است جستجوهای رو به جلو و عقب در عبارت صورت گیرد تا عمل بعدی انجام پذیرد.

ریاضی دان لهستانی لوکاسویچ^۳ نشان داد که عبارات ریاضی می توانند بصورت نمایش پیشوندی^۴ نشان داده شوند. این نمایش عبارات اغلب نمایش لهستانی خوانده می شوند و در آن عملگر را قبل از عملوندها قرار می دهند. نمایش پسوندی^۵ یا نمایش لهستانی معکوس (RPN) عملگر را بعد از عملوندها قرار می دهد. مثال های زیر سه فرم نمایش عبارات را نشان می دهد.

A + B نمایش میانوندی

+AB نمایش پیشوندی لهستانی

AB+ نمایش پسوندی لهستانی

روش لهستانی معکوس برای عملیات پشته مناسب است، عبارت

$$A * B + C * D$$

در این روش بشکل زیر در می آید

$$AB * CD * +$$

و بطریق زیر ارزیابی می شود: عبارت را از چپ به راست جستجو نمایید، وقتی که به عملگری رسیدید آنرا بر روی دو عملوند قبلی اعمال کنید. سپس دو عملوند را حذف و آنها را با حاصل جایگزین نمایید. این عمل ادامه می یابد تا اینکه دیگر به عملگری برخورد نشود.

1- Reverse Polish Notation, RPN

2- Infix notation

3- Lukasiewicz

4- Prefix (or polish notation)

5- Postfix notation (or reverse polish notation)

در مثال فوق عملگر * را پس از A و B می بینیم. عمل $A * B$ را اجرا و A و B و * را با حاصلضرب جایگزین می کنیم

$$(A * B) CD * +$$

که در آن $(A * B)$ حاصلضرب است. عملگر بعدی هم * است و دو عملگر قبلی آن C و D می باشند. بنابراین عمل $C * D$ اجرا و حاصل عبارتی است با دو عملوند و یک عملگر

$$(A * B) (C * D) +$$

عمل بعدی + است و عملوندهایی که باید با هم جمع شوند دو حاصلضرب قبلی می باشند، بنابراین دو کمیت را جمع می کنیم تا نتیجه کلی بدست آید.

در تبدیل از نمایش میانوندی به نمایش لهستانی معکوس باید سلسله مراتب اولویت عمل ها مورد استفاده در نمایش میانوندی در نظر گرفته شود. این اولویت بندی حکم می کند که ابتدا همه عمل های داخل پرانتز درونی را انجام دهیم، سپس داخل بقیه پرانتزها را انجام دهیم، آنگاه اعمال ضرب و تقسیم و بالاخره جمع و تفریق اولویت های بعدی را دارند. عبارت زیر را در نظر بگیرید.

$$(A + B) * [C * (D + E) + F]$$

برای محاسبه این عبارت ما ابتدا باید داخل پرانتزهای $(A + B)$ و $(D + E)$ را ارزیابی کنیم. سپس باید عبارت داخل کروشه بدست آید. ضرب $C * (D + E)$ بر جمع با F اولویت دارد چون ضرب بر جمع مقدم است. آخرین عمل ضرب بین پرانتز و کروشه است. عبارت فوق را می توان با توجه به روش لهستانی معکوس به عبارتی بدون پرانتز تبدیل کرد. عبارت تبدیل شده چنین است:

$$AB + DE + C * F + *$$

با پیشروی از چپ به راست ابتدا A و B را جمع می کنیم، سپس حاصل جمع D و E را بدست می آوریم. در این حالت داریم

$$(A + B) (D + E) C * F + *$$

که در آن $(A + B)$ و $(D + E)$ هر کدام یک کمیت تک مقداری حاصل از جمع می باشند. دو عملوند مربوط به * بعدی C و $(D + E)$ هستند. این دو عدد در هم ضرب شده و حاصل با F جمع می شود. آخرین * نیز ضرب دو جمله $(A + B)$ و حاصل ضرب قبلی را انجام می دهد.

ارزیابی عبارات حسابی

روش RPN به همراه یک مجموعه پشته ثباتی بهترین روش شناخته شده برای ارزیابی عبارات

حسابی است. این روش در بسیاری از ماشین های حساب و نیز برخی از کامپیوترها رایج است. پشته خصوصاً برای حل مسائل طولانی و پیچیده از جمله محاسبات زنجیره ای مفید است. زیرا عبارات حسابی می توانند بصورت RPN بدون پرانتز بیان گردند.

روش مذکور بدین ترتیب انجام می شود که ابتدا عبارات حسابی به فرم معادل RPN در می آیند عملوندها بداخل پشته به همان ترتیبی که نشان داده شده اند هدایت می شوند. آغاز هر عمل بستگی به این دارد که ماشین حساب یا کامپیوتر بکار رفته باشد. در ماشین حساب، عملگرها از طریق صفحه کلید وارد می شوند. در کامپیوتر اعمال توسط دستوراتی که دارای بخش کد عملیاتی هستند و آدرس هم نیاز ندارند شروع می گردد. ریز عمل های زیر با پشته هرگاه که یک عملگر در ماشین حساب وارد شود و یا عملگری در کامپیوتر توسط کنترل تأیید گردد اجرا می گردد: (۱) دو عملوند که در بالاترین مکان پشته قرار دارند برای اجرای عمل مورد نظر انتخاب می شوند، و (۲) پشته پاپ می شود و نتیجه عمل بجای عملوند پائین تر می نشیند. با پوش کردن متوالی عملوندها بداخل پشته و انجام اعمال تعریف شده فوق، عبارت بطور صحیح ارزیابی شده و نتیجه نهایی در بالای پشته باقی می ماند.

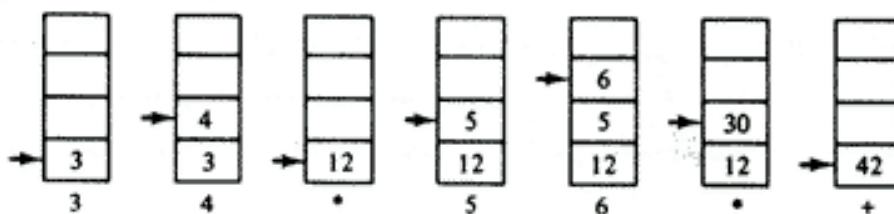
مثال عددی زیر می تواند این روش را واضح تر بیان نماید. عبارت حسابی زیر را در نظر بگیرید.

$$(3 * 4) + (5 * 6)$$

در RPN عبارت بصورت زیر است

$$34 * 56 * +$$

حال عملیات پشته طبق آنچه در شکل ۵-۸ آمده است را ملاحظه نمائید. هر قسمت از شکل یک مرحله از کار پشته را نشان می دهد و فلش همواره به بالای پشته اشاره می کند. با مرور عبارت از چپ به راست با دو عملوند مواجه می شویم. ابتدا عدد ۳ و سپس ۴ بداخل پشته فرستاده می شوند. نماد بعدی عملگر ضرب * است. این نماد موجب ضرب دو عدد واقع در بالاترین مکان پشته می شود. سپس پشته پاپ شده و حاصل ضرب در بالای پشته قرار می گیرد، و در واقع جایگزین دو عملوند می گردد. آنگاه در عبارت فوق با دو عملوند دیگر یعنی ۵ و ۶ مواجه می شویم و آنها را بداخل پشته پوش می کنیم. نتیجه عمل پشته که حاصل از * بعدی است دو عدد اخیر را با حاصل ضرب جایگزین می سازد. آخرین عمل، جمع بر روی دو عدد بالای پشته است تا عدد نهایی ۴۲ حاصل گردد.



شکل ۵-۸ عمل هایی که برای محاسبه $3 \times 4 + 5 \times 6$ روی پشته انجام می شود

ماشین حساب علمی^۱ که از پشته داخلی استفاده می کنند نیاز به تبدیل عبارات به RPN توسط کاربر را دارند. کامپیوترهایی که CPU آنها دارای پشته است نیاز به برنامه ای دارند تا تبدیل را برای کاربر انجام دهد. اغلب کامپایلرها، بدون توجه به سازمان CPU، تمام عبارات حسابی را به RPN تبدیل می کنند زیرا این بهترین روش برای تبدیل عبارات حسابی به دستورالعمل زبان ماشین است. بنابراین یک CPU با سازمان پشته مسلماً در برخی کاربردها مفیدتر از CPU بدون پشته است.

۴-۸ قالب دستورالعمل ها

ساختار فیزیکی و منطقی کامپیوترها معمولاً در کتابچه راهنمایشان که به همراه سیستم است تشریح می شود. چنین کتابچه های ساختمان داخلی CPU، از جمله ثبات های موجود و قابلیت های منطقی آنها را توضیح می دهد. آنها فهرست همه دستورات پیاده سازی شده با سخت افزار، قالب کد آنها و نهایتاً تعریف دقیقی برای هر دستور را فراهم می آورند. یک کامپیوتر معمولاً انواع متنوعی از قالب دستورات را داراست. وظیفه واحد کنترل در CPU این است تا هر کد دستورالعمل را تفسیر و تابع کنترل لازم برای پردازش دستورالعمل را تولید کند.

قالب یک دستورالعمل معمولاً در یک چهارگوشه کشیده می شود که در آن بیت های دستورالعمل آن طور که در کلمه حافظه یا ثبات کنترل ظاهر می شوند آمده است. بیت های دستورالعمل به گروه هایی بنام میدان^۲ تقسیم می شود. متداول ترین میدان هایی که در قالب دستورالعمل ها دیده می شوند، عبارتند از:

۱- میدان کد عملیات که عمل را مشخص می کند

۲- میدان آدرس که آدرس حافظه یا ثبات پردازشگر را معین می نماید.

۳- میدان روش^۳ (شیوه) که نحوه تعیین آدرس مؤثر یا عملوند را مشخص می نماید.

تحت شرایط خاصی، میدان های خاص دیگری نیز گاهی اوقات بکار گرفته می شوند، مثلاً میدانی که تعداد شیفت های یک دستورالعمل از نوع شیفت را معین می کند.

میدان کد عملیات یک دستورالعمل عبارتست از گروهی از بیت ها که انواع اعمال پردازشگر مانند جمع، تفریق، مکمل سازی و شیفت را تعریف می نمایند. متداول ترین اعمال موجود در کامپیوترها در بخش ۶-۸ بحث شده است. بیت هایی که روش را برای دستور تعریف می کنند راه های مختلفی را برای انتخاب عملوند از آدرس مفروض معین می نمایند. روشهای مختلف آدرس دهی که برای کامپیوترهای دیجیتال می توان ردیف کرد در بخش ۵-۸ ارائه شده اند. در این بخش درباره میدان آدرس یک دستورالعمل بحث خواهیم نمود و اثر میدان های چند آدرس در یک دستور مورد بررسی قرار خواهند گرفت.

اعمال مشخص شده توسط دستورات کامپیوتر بر روی داده های ذخیره شده در حافظه یا ثبات های پردازنده اجرا می شود. عملوندهای موجود در حافظه توسط آدرس حافظه مشخص می شوند. عملوندهایی

1- Scientific Calculator

2- Field

3- Mode

که در ثبات‌های پردازشگر قرار دارند توسط آدرس ثبات معین می‌گردند. یک آدرس ثبات یک عدد k بیتی است که 2^k ثبات را در CPU تعریف می‌کند. بنابراین یک CPU با 16 ثبات R_0 تا R_{15} دارای میدان آدرس ثبات چهاربیتی است. مثلاً عدد دودویی 0101 ثبات R_5 را مشخص می‌کند. کامپیوترها ممکن است دستوراتی با طول‌های مختلف داشته و تعداد آدرس‌های آن‌ها متفاوت باشد. تعداد میدان‌های آدرس در قالب دستور به سازمان داخلی ثبات‌های آن بستگی دارد. اغلب کامپیوترها یکی از سه نوع CPU زیر را استفاده می‌کنند:

۱- سازمان تک انباره‌ای

۲- سازمان چند ثباتی

۳- سازمان پشته‌ای

مثالی از نوع تک انباره‌ای همان کامپیوتر فصل ۵ است. تمام اعمال بکمک یک انباره صورت می‌گیرد. قالب دستور در این نوع کامپیوترها از یک آدرس استفاده می‌نماید. مثلاً، دستوری که یک جمع را تعریف می‌کند بصورت زبان اسمبلی زیر می‌باشد.

ADD X

که X آدرس عملوند است. دستور ADD در این حالت طبق عبارت زیر عمل می‌نماید

$AC \leftarrow AC + M[X]$

AC ثبات انباره و $M[X]$ کلمه حافظه واقع در آدرس X را مشخص می‌نماید.

مثالی از نوع چند ثباتی در شکل ۷-۱ نشان داده شد. قالب دستور در این نوع کامپیوترها به سه میدان آدرس ثبات نیاز دارد. بنابراین دستور برای یک عمل جمع بفرم اسمبلی بشکل زیر می‌تواند نوشته شود

ADD R_1, R_2, R_3

که عمل $R_1 \leftarrow R_2 + R_3$ را تداعی می‌کند. تعداد میدان‌های آدرس در دستور از سه به دو قابل تقلیل است بشرطی که ثبات مقصد یکی از دو ثبات مبدأ باشد. بنابراین دستور

ADD R_1, R_2

عمل $R_1 \leftarrow R_2 + R_1$ را تداعی خواهد کرد. در این دستور فقط آدرس ثبات‌های R_1 و R_2 مورد نیازند. کامپیوترهایی که ثبات‌های متعددی را دارا هستند از دستور انتقال (move) با نماد خلاصه MOV استفاده می‌کنند. با این ترتیب دستور

MOV R_1, R_2

بیانگر عمل $R_1 \leftarrow R_2$ (یا $R_2 \leftarrow R_1$ که به نوع کامپیوتر بستگی دارد) می‌باشد. بنابراین این دستورات به دو میدان آدرس برای تعیین مبدأ و مقصد نیاز دارند.

کامپیوترهای چند ثباتی دو یا سه میدان آدرس را در ساختمان دستورالعمل بکار می‌برند. هر میدان

آدرس یک ثبات پردازنده یا آدرس حافظه را معین می نماید. دستور زیر :

ADD R₁, X

مربوط به عمل $R_1 \leftarrow R_1 + M[X]$ است. این دستور دو میدان آدرس یکی برای R_1 و دیگری برای آدرس حافظه X دارد.

CPU سازمان یافته با پشته در شکل ۴-۸ ارائه شد. کامپیوترهایی از این نوع دارای دستورات پوش و پاپ می باشند که تنها یک میدان آدرس برای آنها نیاز است. بنابراین دستور

PUSH X

کلمه ای در آدرس X را در بالای پشته می نشانند. اشاره گر پشته بطور اتوماتیک تصحیح می گردد. دستورات محاسباتی در کامپیوترهای سازمان یافته با پشته احتیاجی به آدرس ندارند زیرا اعمال مزبور بر روی دو کمیت که در بالای پشته قرار دارند اجرا می شود.

دستورالعمل **ADD** در کامپیوتر پشته ای متشکل از یک کد عمل ولی بدون آدرس است. این عمل وظیفه بازگرفتن^۱ دو عدد در بالای پشته را بعهده داشته و پس از جمع آنها، حاصل را در پشته می نشانند^۲ در این دستور نیازی به عملونندی همراه با میدان آدرس نیست زیرا تمام عملوندها در پشته واقعند.

همانطور که اشاره شد اکثر کامپیوترها در یکی از سه سازمان فوق قرار می گیرند. برخی کامپیوترها ترکیبی از چند سازمان را دارا هستند. مثلاً ریزپردازنده 8080 از Intel دارای هفت ثبات در CPU است که یکی از آنها انباره است. در نتیجه پردازشگر دارای برخی خواص چند ثباتی و بعضی خواص نوع انباره ای است. تمام دستورات حساب و منطق به همراه دستورات ذخیره و بار کردن از ثبات انباره استفاده می نمایند. بنابراین دستورات مذکور دارای یک میدان آدرس هستند. از طرف دیگر دستوراتی که داده ها را بین ثبات ها نقل و انتقال می دهند دارای دو میدان آدرس برای ثبات هستند. بعلاوه پردازنده Intel 8080 دارای یک اشاره گر پشته بوده و دستوراتی برای نشانندن و بازگرفتن داده از پشته حافظه ای دارد. با این وجود پردازشگر دارای دستورات نوع آدرس صفر^۳ که از مشخصه های CPU با سازمان پشته است نمی باشد. برای تشریح تأثیر مقدار آدرس ها در برنامه کامپیوتر عبارت زیر را ارزیابی می کنیم

$$X = (A + B) * (C + D)$$

در این بحث دستورات صفر، یک، دو و سه آدرسی بکار خواهند رفت. ما دستورات **ADD**، **SUB**، **MUL** و **DIV** را برای چهار عمل اصلی بکار خواهیم برد و از دستور **MOV** برای انتقال و **LOAD** و **STORE** برای بار و ذخیره کردن بین حافظه و انباره **AC** استفاده می نمایم. همچنین فرض می کنیم عملوندها در آدرسهای A و B و C و D بوده و نتیجه در آدرس X قرار گیرد.

1- Pop

2- Push

3- Zero – address type instruction

دستورات سه آدرس

کامپیوترهایی که قالب دستورات آن‌ها سه آدرس است هر میدان آدرس را برای مشخص نمودن ثبات پردازشگر یا عملوندی در حافظه می‌توانند بکار برند. برنامه‌ای در زبان اسمبلی که عبارت $X = (A+B) * (C+D)$ را محاسبه کند به همراه توضیحات مربوط به انتقال از ثباتهای هر دستور در زیر آمده است.

ADD	R ₁ , A, B	$R_1 \leftarrow M[A] + M[B]$
ADD	R ₂ , C, D	$R_2 \leftarrow M[C] + M[D]$
MUL	X, R ₁ , R ₂	$M[X] \leftarrow R_1 * R_2$

فرض براین است که پردازشگر دارای دو ثبات R₁ و R₂ است. نماد M[A] بمعنی وجود عملوند در آدرس A است.

مزیت دستورات سه آدرس کوتاه‌تری برنامه در ارزیابی عبارات محاسباتی است. عیب این دستورات تخصیص تعداد زیاد بیت برای معین کردن سه آدرس است. نمونه‌ای از کامپیوترهای تجاری که اینگونه دستورات را بکار می‌برند Cyber 170 است. قالب دستورات در کامپیوتر Cyber محدود به سه آدرس ثبات یا دو آدرس ثبات و یک آدرس حافظه است.

دستورات دو آدرس

دستورات دو آدرس در کامپیوترهای تجاری بسیار مرسوم هستند. در اینجا نیز هر میدان آدرس یک ثبات پردازشگر یا یک کلمه حافظه را معین می‌کنند. برنامه ارزیابی عبارت $X=(A+B) * (C+D)$ بفرم زیر است.

MOV	R ₁ , A	$R_1 \leftarrow M[A]$
ADD	R ₁ , B	$R_1 \leftarrow R_1 + M[B]$
MOV	R ₂ , C	$R_2 \leftarrow M[C]$
ADD	R ₂ , D	$R_2 \leftarrow R_2 + M[D]$
MUL	R ₁ , R ₂	$R_1 \leftarrow R_1 * R_2$
MOV	X, R ₁	$M[X] \leftarrow R_1$

دستور MOV، عملوند را از او یا به حافظه و ثبات‌های پردازنده انتقال می‌دهد. اولین نمادی که در دستور آمده است هم یکی از منابع و هم مقصد مربوط به نتیجه عمل است.

دستورات یک آدرس

دستورات یک آدرس از انباره AC برای تمام دستکاری‌های روی داده استفاده می‌کنند. ضرب و تقسیم نیاز به یک ثبات دیگر نیز دارد. با این وجود، ما از این ثبات دوم صرف‌نظر کرده و فرض خواهیم کرد که AC

قادر است نتایج تمام عملیات را در خود جای دهد. برنامه ارزیابی $X=(A+B)*(C+D)$ بقرار زیر است:

LOAD	A	$AC \leftarrow M[A]$
ADD	B	$AC \leftarrow AC + M[B]$
STORE	T	$M[T] \leftarrow AC$
LOAD	C	$AC \leftarrow M[C]$
ADD	D	$AC \leftarrow AC + M[D]$
MUL	T	$AC \leftarrow AC * M[T]$
STORE	X	$M[X] \leftarrow AC$

دیده می شود که تمام عملیات بین انباره AC و عملوند حافظه صورت گرفته است. T آدرس یک مکان حافظه موقت است تا نتایج میانی را ذخیره نماید.

دستورات صفر آدرسه

یک کامپیوتر سازمان یافته با پشته، میدان آدرس را برای دستورات ADD و MUL بکار نمی برد. با این وجود دستورات پوش و پاپ برای تعیین عملوندی که با پشته تبادل اطلاعات می کند نیاز به آدرس دارند. برنامه زیر نشان می دهد که چگونه $X=(A+B) * (C+D)$ بصورت یک برنامه برای این نوع کامپیوتر درآمده است (TOS بمعنی بالای پشته^۱ است)

PUSH	A	$TOS \leftarrow A$
PUSH	B	$TOS \leftarrow B$
ADD		$TOS \leftarrow (A + B)$
PUSH	C	$TOS \leftarrow C$
PUSH	D	$TOS \leftarrow D$
ADD		$TOS \leftarrow (C + D)$
MUL		$TOS \leftarrow (C+D)*(A+B)$
Pop	X	$M[X] \leftarrow TOS$

برای ارزیابی عبارات حسابی در کامپیوتر پشته ای، لازم است تا عبارات بشکل RPN در آیند. بعلت عدم وجود میدان آدرس در این نوع کامپیوتر نام صفر آدرسه به آنها اتلاق شده است.

دستورات RISC

مزایای یک معماری کامپیوتر کم دستور^۲ (RISC) در بخش ۸-۸ تشریح شده است. مجموعه دستورات پردازشگر RISC به هنگام تبادل اطلاعات CPU با حافظه به LOAD و STORE محدود

1-Top of stack

2- Reduced Instruction Set Computer

می شود. سایر دستورات در داخل CPU و بدون ارجاع به حافظه انجام می شود. همانطور که اشاره شد برنامه ها برای CPU های نوع RISC از دستورات LOAD و STORE تشکیل یافته اند که دارای یک آدرس حافظه و یک آدرس ثبات اند ولی دستورات نوع محاسباتی سه آدرس ثبات از کل ثبات های پردازشگر را مشخص می نمایند. برنامه زیر عبارت $X = (A+B) * (C+D)$ را مشخص می نماید.

LOAD	R ₁ , A	R ₁ ← M[A]
LOAD	R ₂ , B	R ₂ ← M[B]
LOAD	R ₃ , C	R ₃ ← M[C]
LOAD	R ₄ , D	R ₄ ← M[D]
ADD	R ₁ , R ₁ ,R ₂	R ₁ ← R ₁ + R ₂
ADD	R ₃ , R ₃ , R ₄	R ₃ ← R ₃ + R ₄
MUL	R ₁ , R ₁ ,R ₃	R ₁ ← R ₁ * R ₃
STORE	X , R ₁	M[X] ← R ₁

دستور LOAD عملوندها را از حافظه به ثبات های CPU انتقال می دهد. دستورات ADD و MOV هم داده های درون ثباتها را بدون دستیابی به حافظه اجرا می نمایند. سپس نتیجه محاسبات با استفاده از دستور STORE در حافظه ذخیره می گردد.

۸-۵ روشهای آدرس دهی^۱

میدان عمل یک دستور، عملی را که قرار است انجام شود مشخص می نماید. این عمل بر روی داده هایی که در ثبات ها و یا حافظه ها ذخیره شده اند اجرا می شود. روش انتخاب عملوندها در حین اجرای برنامه به روش آدرس دهی دستور وابسته است. روش آدرس دهی، قاعده ای را برای تفسیر و تصحیح آدرس دستورالعمل مشخص می کند. کامپیوترها تکنیک روشهای آدرس دهی را به یک یا هر دو دلیل زیر بکار می برند.

۱- به کاربر، بدلیل در اختیار داشتن اشاره گر، شمارنده های حلقه، اندیس دهی داده ها و تغییر مکان برنامه انعطاف پذیری لازم را می دهند.

۲- تعداد بیت های میدان آدرس دستورالعمل را کاهش می دهند.

در اختیار داشتن روشهای آدرس دهی به برنامه نویسان زبان اسمبلی با تجربه، انعطاف لازم را ارائه می دهد تا از نظر تعداد دستورات و زمان اجرا برنامه های بهتری را تهیه نمایند.

1- Addressing modes

برای شناخت انواع روشهای آدرس دهی که در این بخش ارائه خواهد شد، بهتر است ابتدا سیکل عملیات یک کامپیوتر را بدانیم. واحد کنترل یک کامپیوتر به گونه‌ای طراحی شده تا وارد یک سیکل دستورالعمل متشکل از سه فاز عمده زیر گردد.

۱- برداشت^۱ دستورالعمل از حافظه

۲- دیکد کردن دستور

۳- اجرای^۲ دستور

در کامپیوتر ثباتی بنام شمارنده برنامه^۳ (یا PC) قرار دارد که دستورات یک برنامه ذخیره شده در حافظه را دنبال می‌کند. PC آدرس دستوری را که قرار است بعداً اجرا شود نگهداشته و هر وقت دستوری از حافظه برداشته شد یک واحد به محتوای آن اضافه می‌شود. دیکد شدن در مرحله ۲، عملی را که باید انجام شود، روش آدرس دهی دستور و مکان عملوند را مشخص می‌نماید. سپس کامپیوتر دستور را اجرا نموده و به مرحله ۱ برای برداشت دستور بعدی باز می‌گردد.

در برخی از کامپیوترها روش آدرس دهی دستور توسط یک کد دودویی مجزا، درست مثل کد عمل مشخص می‌شود. دسته‌ای دیگر از کامپیوترها از یک کد برای تعیین و روش آدرس دهی دستور استفاده می‌نمایند. دستورات ممکن است با انواع روشهای آدرس دهی تعریف شوند و گاهی اوقات دو یا سه روش آدرس دهی در یک دستور بصورت ترکیب بکار می‌روند.

مثالی از قالب دستورالعمل با میدان روش آدرس دهی مجزا در شکل ۶-۸ نشان داده شده است. کد عمل، عملی را که قرار است اجرا شود مشخص می‌نماید. میدان روش برای تعیین مکان عملوند بکار می‌رود. در دستورالعمل ممکن است میدان آدرس وجود داشته و یا نداشته باشد. اگر میدان آدرس وجود داشته باشد متعلق به آدرس حافظه و یا ثبات پردازشگر است. بعلاوه همانطور که در بخش قبل گفته شد، دستور ممکن است بیش از یک میدان آدرس داشته باشد و هر میدان می‌تواند روش آدرس دهی خاص خود را دارا باشد.

روش ضمنی: در این روش آدرس دهی عملوندها در تعریف خود دستور مستترند. مثلاً دستور "مکمل کردن انباره" یک دستور از نوع ضمنی است زیرا عملوند موجود در انباره در تعریف دستور آمده است. در واقع تمام دستورات ارجاع به ثبات که از ثبات انباره استفاده می‌کنند از این دسته‌اند.

آدرس	روش	کد عمل
------	-----	--------

شکل ۶-۸ قالب دستورالعمل با میدان روش (شیوه)

1- Fetch

2-Execute

3- Program Counter

دستورات صفر آدرسه در کامپیوترهای پشته‌ای نیز از نوع روش ضمنی هستند زیرا عملوندها در بالای پشته قرار دارند و برای آنها نیازی به آدرس نیست.

روش بلافصل: در این روش عملوند جزئی از دستورالعمل است. به بیان دیگر یک دستورالعمل از این نوع بجای میدان آدرس دارای میدان عملوند است. میدان عملوند، خود عملوندی است که باید همراه با کد عمل در دستور مورد استفاده قرار گیرد. دستورات بلافصل برای مقداردهی اولیه ثبات‌ها با یک عدد ثابت مناسبند.

قبلاً ذکر شده که میدان آدرس دستورالعمل ممکن است یک حافظه یا یک ثبات پردازشگر را مشخص نماید. اگر میدان آدرس، یک ثبات را معین کند گوئیم دستور از نوع روش ثباتی است.

روش ثباتی: در این روش عملوندها در ثبات‌های درون CPU قرار گرفته‌اند. ثبات مورد نظر از میدان آدرس دستور انتخاب می‌شود. یک میدان k بیت می‌تواند 2^k ثبات را مشخص نماید.

روش غیرمستقیم ثباتی: در این روش دستورالعمل، ثباتی را در CPU مشخص می‌سازد که محتوای آن آدرس عملوند در حافظه می‌باشد. به بیان دیگر ثبات انتخابی بجای عملوند، حاوی آدرس عملوند می‌باشد. قبل از استفاده از این نوع دستورالعمل، برنامه‌نویس باید مطمئن شود که آدرس حافظه قبلاً در ثبات قرار گرفته است. مزیت این روش این است که میدان آدرس ثبات، بیت‌های کمتری را در مقایسه با روش آدرس دهی مستقیم در برنامه بکار می‌برد.

روش خود افزایش یا خودکاهشی: این روش مشابه روش قبلی است با این تفاوت که پس از هر بار استفاده از آدرس حافظه مقدار آن در ثبات بطور خودکار افزایش یا کاهش می‌یابد. وقتی که آدرس ذخیره شده در ثبات به جدولی از داده‌ها در حافظه ارجاع داده شود، لازم است تا ثبات پس از هر دستیابی یکبار افزایش یا کاهش یابد. این تصحیح با بکارگیری دستورات افزایش یا کاهش میسر است. با این وجود بعلت استفاده‌های مکرر، برخی کامپیوترها از امکان خاصی سود می‌برند که توسط آن پس از هر دستیابی محتوای ثبات را بطور خودکار افزایش یا کاهش می‌دهند.

میدان آدرس یک دستور توسط واحد کنترل استفاده می‌شود تا عملوند را از حافظه بدست آورد. گاهی اوقات مقدار داده شده در میدان آدرس، آدرس عملوند است و گاهی نیز آدرسی است که با استفاده از آن آدرس عملوند محاسبه می‌گردد. برای تفکیک انواع روشهای آدرس دهی لازم است تا بخش آدرس دستورالعمل مورد تفحص قرار گرفته و آدرس مؤثر که بوسیله کنترل در زمان اجرای دستور استفاده می‌شود معین می‌گردد. آدرس مؤثر^۱ آدرس حافظه‌ای است که از محاسبه‌ای که روش آدرس دهی دیکته

1- Effective Address

می‌کند حاصل می‌شود. در دستورات محاسباتی آدرس مؤثر همان آدرس عملوند است. در دستورات انشعابی این آدرس، آدرسی است که کنترل به آن انشعاب می‌نماید. ما در فصل ۵ دو روش آدرس دهی را معرفی کردیم. در اینجا مجدداً آنها را بطور خلاصه یادآور می‌شویم.

روش آدرس دهی مستقیم^۱: در این روش آدرس مؤثر همان بخش آدرس دستورالعمل است. عملوند واقع در حافظه مستقیماً توسط این آدرس مشخص می‌شود. در دستورات نوع انشعاب، میدان آدرس، آدرس واقعی انشعاب را تعیین می‌نماید.

روش آدرس دهی غیرمستقیم^۲: در این روش میدان آدرس دستورالعمل مکان آدرس مؤثر را که در حافظه است مشخص می‌نماید. کنترل دستور را از حافظه برداشت و بخش آدرس آنرا برای دستیابی به حافظه و لذا یافتن آدرس مؤثر می‌خواند. روش آدرس دهی غیرمستقیم قبلاً در بخش ۱-۵ تشریح شد. در چند روش آدرس دهی لازم است تا میدان آدرس دستور به محتوای ثبات خاصی در CPU اضافه شود. در این روشها آدرس مؤثر از طریق زیر حاصل می‌گردد.

$$\text{آدرس مؤثر} = \text{بخش آدرس دستور} + \text{محتوای ثبات در CPU}$$

ثبات بکار رفته در محاسبه فوق می‌تواند شمارنده برنامه، یک ثبات شاخص (یا اندیس) یا یک ثبات پایه باشد. در هر حالت ما روش آدرس دهی مختلفی خواهیم داشت که در کاربردهای مختلف مورد استفاده قرار گیرد.

روش آدرس دهی نسبی^۳: در این روش معمولاً شمارنده برنامه به بخش آدرس دستور اضافه می‌شود تا آدرس مؤثر بدست آید. بخش آدرس دستور معمولاً یک عدد علامت دار است (بفرم مکمل 2) که می‌تواند مثبت یا منفی باشد. وقتی که این عدد به محتوای شمارنده برنامه اضافه شود حاصل آدرس مؤثری است که مکانی را در حافظه نسبت به دستورالعمل بعدی معین می‌سازد. بعنوان مثال، فرض کنید که شمارنده برنامه دارای عدد 825 باشد و بخش آدرس دستور 24 در نظر گرفته شود. دستور واقع در مکان 825 در طول فاز برداشت از مکان 825 خوانده شده و شمارنده سپس به 826 افزایش می‌یابد. آدرس مؤثر در این روش آدرس دهی برابرست $825 + 24 = 850$. این مکان 24 خانه حافظه بعد از دستور بعدی است. آدرس دهی نسبی اغلب همراه دستورات نوع انشعابی و بهنگام قرار گرفتن آدرس انشعاب در حول و حوش خود دستورالعمل بکار می‌رود. این روش میدان آدرس کوچکتری را در ساختار دستورالعمل می‌طلبد زیرا آدرس نسبی به تعداد بیت‌های کمتری در مقایسه با کل آدرس حافظه نیاز دارد.

1- Direct Address Mode

2- Indirect Address Mode

3- Relative Address Mode

روش آدرس دهی شاخص دار^۱ در این روش محتوای یک ثبات شاخص^۲ به بخش آدرس دستور اضافه می شود تا آدرس موثر بدست آید. ثبات شاخص، ثبات خاصی از CPU است که مقدار شاخص برای کمیت ها را در خود دارد. میدان آدرس دستورالعمل آدرس شروع یک آرایه داده را در حافظه تعریف می کند. هر عملوند در آرایه در حافظه ای نسبت به آدرس شروع ذخیره می گردد. فاصله بین آدرس شروع آرایه و آدرس عملوند، مقدار آدرس ذخیره شده در ثبات شاخص است. هر عملوندی در آرایه با یک دستور قابل دسترسی است بشرطی که ثبات شاخص دارای مقدار صحیح شاخص باشد. ثبات شاخص می تواند افزایش یابد تا عملوندهای متوالی را در دسترس قرار دهد. توجه داشته باشید که اگر این دسته دستورات دارای میدان آدرس در قالب دستورالعمل نباشند، دستور به آدرس دهی غیرمستقیم ثبات تبدیل می شود.

برخی از کامپیوترها یک ثبات بتنهایی برای شاخص اختصاص می دهند. این ثبات بهنگام استفاده از دستورات آدرس دهی شاخص دار در عملیات دخالت می نماید. در کامپیوترهایی که چندین ثبات پردازنده دارند هر یک از ثبات ها می توانند حاوی عدد شاخص باشند. در چنین حالتی ثبات را باید در داخل قالب دستور مشخص کرد.

روش آدرس دهی با ثبات پایه: در این روش محتوای یک ثبات پایه به بخش آدرس دستور اضافه می شود تا آدرس موثر بدست آید. این روش مشابه روش آدرس دهی شاخص دار است با این اختلاف که ثبات در اینجا ثبات پایه نام دارد. اختلاف بین دو روش بیشتر روش استفاده از آنهاست تا روش محاسبه آنها. فرض بر این است که ثبات شاخص، عدد شاخص را نسبت به بخش آدرس دستور درخود نگه می دارد. یک ثبات پایه آدرس مبنایی را نگه می دارد و میدان آدرس دستور جابجایی نسبت به این آدرس را معین می نماید. روش آدرس دهی ثبات پایه در کامپیوتر برای تسهیل در تغییر مکان برنامه در حافظه مورد استفاده است. وقتی که برنامه ها و داده ها از یک قسمت از حافظه به قسمت دیگری از حافظه جابجا می شوند، مثل سیستم های چند برنامه ای^۳، مقادیر آدرس دستورات باید این تغییر را انعکاس دهند. با یک ثبات پایه، مقادیر جابجایی دستورات نیاز به تغییر ندارند. و این فقط مقدار ثبات پایه است که نیاز به تصحیح دارد تا منعکس کننده ابتدای قطعه جدیدی از حافظه باشد.

مثال عددی

برای نشان دادن تفاوت های بین روشهای مختلف آدرس دهی، ما تأثیر آدرس دهی را روی دستورات معرفی شده در شکل ۷-۸ نشان می دهیم. دستور دو کلمه ای واقع در آدرس های 200 و 201 یک دستور "بارکردن AC" با میدان آدرس 500 می باشد. اولین کلمه دستور کدعمل و روش آدرس دهی و دومین

1- Indexed Addressing Mode

2- Index Register

3- Multiprogramming

$$PC = 200$$

$$R_1 = 400$$

$$XR = 100$$

$$AC$$

شکل ۷-۸ مثال عددی برای شیوه‌های آدرس‌دهی

آدرس	حافظه	شیوه
200	باردهی در AC	
201		$AC = 500$
202	دستورالعمل بعدی	
399		450
400		700
500		800
600		900
702		325
800		300

کلمه آدرس می‌باشد. PC دارای عدد 200 برای دریافت این دستور است. محتوای ثبات پردازشگر R_1 برابر 400 و محتوای ثبات اندیس XR نیز 100 است. AC عملوند را پس از اجرای دستور برداشت می‌نماید. در شکل مزبور چند آدرس مرتبط با مثال لیست شده و محتوای حافظه هر یک از آدرس‌های مذکور را نشان می‌دهد.

میدان روش^۱ دستور می‌تواند هر یک از روش‌ها را مشخص کند. برای هر روش ممکن ما آدرس موثر را محاسبه خواهیم کرد و عملوندی را که باید در AC بار شود معین می‌نمائیم. در روش آدرس دهی مستقیم، آدرس موثر همان بخش آدرس دستور است و عملوندی که باید در AC بار شود 800 است. در روش بلافصل، بخش دوم دستور بجای اینکه آدرس باشد عملوند تلقی می‌شود. بنابراین 500 در AC بار می‌گردد. (آدرس موثر در این حالت 201 است). در روش غیرمستقیم آدرس موثر در آدرس 500 ذخیره شده است. بنابراین آدرس موثر 800 و عملوند 300 می‌باشد. در روش نسبی آدرس موثر برابر است با $500 + 202 = 702$ و عملوند 325 می‌باشد. (توجه کنید که مقدار داخل PC بعد از فاز برداشت و در طول فاز اجرا 202 است) در روش آدرس دهی شاخص دار، آدرس موثر برابرست با $XR + 500 = 100 + 500 = 600$ و عملوند هم 900 است. در روش ثبات عملوند در R_1 و عدد 400 و برابر با محتوای R_1 است و عملوندی که در AC بار می‌شود 700 می‌باشد. روش خود افزایش مشابه روش ثباتی غیرمستقیم است با این اختلاف که پس

1- Mode field

جدول ۴-۸ جدول مقادیر مثال عددی

محتوای AC	آدرس موثر	آدرس دهی
800	500	آدرس مستقیم
500	201	عملوند بلافصل
300	800	آدرس غیر مستقیم
325	702	آدرس نسبی
900	600	آدرس شاخص دار
400	—	ثباتی
700	400	ثباتی غیر مستقیم
700	400	خودافزایش
450	399	خودکاهش

از اجراء R_1 افزایش و به 401 تغییر یافته است. در روش خود کاهش، قبل از اجرای دستور، R_1 را به 399 تقلیل می دهد. عملوند بارشده در AC برابر 450 است. جدول ۴-۸ مقادیر آدرس موثر و عملوند بارشده در AC را برای 9 روش آدرس دهی لیست نموده است.

۶-۸ انتقال و دستکاری داده ها

کامپیوترها مجموعه دستورات قابل توجهی را برای ایجاد انعطاف پذیری اعمال محاسباتی در اختیار کاربر قرار می دهند. مجموعه دستورالعمل های کامپیوترها از نظر روش تعیین عملوندها با یکدیگر اختلاف دارند. ولی اعمال واقعی انجام شده بوسیله دستورات از یک کامپیوتر به کامپیوتر دیگر با هم تفاوت چندانی ندارند. اغلب دیده می شود کد دودویی تخصیص یافته به دستورات در کامپیوترها مختلفند و این حتی برای دستورات مشابه با هم نیز دیده می شود. همچنین ممکن است نام اسمبلیک بکار رفته در زبان اسمبلی در کامپیوترهای مختلف متفاوت باشد. در هر صورت در بیشتر کامپیوترها مجموعه ای از دستورات دیده می شوند که اعمال ساده و پایه ای را انجام می دهند. مجموعه دستورات اصلی در نمونه ای از کامپیوترها موضوع مورد بحث در این بخش و بخش بعدی است.

اکثر دستورالعمل های کامپیوتر را می توان به سه دسته زیر تقسیم کرد:

۱- دستورالعمل های انتقال داده^۱

۲- دستورات عمل های دستکاری داده ها^۲

۳- دستورات کنترل برنامه

دستورالعمل های انتقال داده موجب انتقال داده ها از یک مکان به مکان دیگر بدون هر گونه تغییر در محتوای اطلاعات دودویی آنها می گردند. دستورات عمل های دستکاری داده ها دستوراتی هستند که اعمال حسابی، منطقی و شیفت را انجام می دهند. دستورات عمل های کنترل برنامه قابلیت های تصمیم گیری را

1- Data Transfer

2- Data Manipulation

فراهم می آورند و مسیر برنامه را در حین اجرای برنامه تغییر می دهند. مجموعه دستورالعمل های یک کامپیوتر بخصوص، اعمال انتقال ثبات و تصمیمات کنترلی موجود را برای استفاده کننده معین می نمایند.

دستورالعمل های انتقال داده

دستورالعمل های انتقال داده، انتقال داده ها را از یک مکان در کامپیوتر به مکانی دیگر بدون هر گونه تغییری در آنها انجام می دهد. متداولترین انتقال ها بین حافظه و ثبات های پردازشگر صورت می گیرد. جدول ۵-۸ لیست هشت دستورالعمل انتقال داده را که در بسیاری از کامپیوترها بکار می رود نشان می دهد. همراه با هر دستور یک سمبل اختصاری ارائه شده است. باید دانست که کامپیوترهای مختلف سمبل های اختصاری مختلفی را استفاده می کنند.

دستور LD به معنی بارکردن^۱ اکثراً به هنگام انتقال داده از حافظه به یک ثبات پردازنده که معمولاً انباره است به کار می رود. دستور ST به معنی ذخیره کردن^۲، انتقال از یک ثبات پردازنده به حافظه را مشخص می کند. دستورالعمل MOV به معنی انتقال^۳، در کامپیوتر چند ثباتی برای انتقال داده از ثباتی به ثبات دیگر مورد استفاده قرار می گیرد. این دستور برای انتقال داده ها بین ثبات های CPU و حافظه و نیز بین دو مکان حافظه هم بکار می رود. دستور XCH به معنی تعویض^۴، اطلاعات را بین دو ثبات یا بین یک ثبات و یک حافظه عوض می کند. دستورات IN و OUT به معنی ورودی^۵ و خروجی^۶، داده ها را بین ثبات ها و پایانه های ورودی یا خروجی جابجا می کنند. دستورات PUSH (پوش) و POP (پاپ) به ترتیب به معنی فشردن (نشاندن) و کشیدن (بازگرفتن)، داده ها را بین ثبات های پردازشگر و پشته ای در حافظه انتقال می دهند.

باید دانست که دستورات جدول ۵-۸ و نیز جداول بعدی در این بخش، اغلب با انواع روشهای آدرس دهی همراه اند. گاهی در زبان اسمبلی بطور قراردادی تغییری در سمبل های اختصاری داده می شود که منعکس کننده شیوه آدرس دهی می باشد. مثلاً سمبل اختصاری دستور load immediate به معنی

جدول ۵-۸ دستورالعمل های انتقال داده ها

نام	سمبل اختصاری
بارکردن	LD
ذخیره	ST
انتقال	MOV
تبادل	XCH
ورودی	IN
خروجی	OUT
پوش (فشردن)	PUSH
پاپ (برداشتن)	POP

1- Load

2- Store

3- Move

4- Exchange

5- Input

6- Output

بار کردن بلافاصل بصورت LDI درمی آید. گاهی نیز قرار داد زبان اسمبلی این است که از کاراکتر خاصی برای مشخص کردن روش آدرس دهی استفاده شود. مثلاً روش بلافاصل باقراردادن علامت # قبل از عملوند مشخص می شود. در هر حال نکته مهمی که باید به آن توجه کرد این است که هر دستور می تواند با انواع روش های آدرس دهی همراه باشد. مثلاً دستور بارکردن انباره رابا هشت روش آدرس دهی مختلف ملاحظه کنید. جدول ۶-۸ زبان اسمبلی پیشنهادی و انتقال واقعی داده را که در هر مورد صورت می گیرد نشان می دهد. ADR به معنی آدرس، NBR بمعنی یک عدد یا عملوند، X یک ثبات شاخص، R1 یک ثبات پردازنده و AC ثبات انباره است. کاراکتر @ سمبل آدرس غیرمستقیم است. کاراکتر \$ قبل از آدرس، آنرا به درون PC نسبت می دهد. کاراکتر # در دستورات بلافاصل قبل از عملوند قراردادده می شود. دستورات شاخص دار با ثباتی در درون پرانتز که پس از آدرس قرار می گیرد شناخته می شود. در آدرس دهی ثباتی نام ثبات حاوی عملوند جلو نام اختصاری دستور قرار می گیرد. در آدرس دهی ثباتی غیرمستقیم نام ثباتی که حاوی آدرس است داخل پرانتز قرار داده می شود. در روش خود افزایشی با قرار دادن علامت + بعد از پرانتز محصورکننده ثبات از روش ثباتی غیرمستقیم متمایز می شود. در روش خود کاهششی هم از علامت - استفاده می گردد. برای نوشتن برنامه هایی بزبان اسمبلی در کامپیوتر، لازم است که انواع دستورات عمل های موجود دانسته شوند و با شیوه ها و روش های آدرس دهی بکار رفته در آن آشنا باشیم.

دستورات عمل های دستکاری داده ها

دستورات عمل های دستکاری داده اعمال راروی داده ها انجام می دهند و قابلیت های محاسباتی کامپیوتر را فراهم می آورند. این دستورات عمل ها به سه نوع اساسی تقسیم می شوند.

۱- دستورات عمل های حسابی

۲- دستورات عمل های منطقی و دستکاری های بیتی

۳- دستورات عمل های شیفت

لیست دستورات دستکاری داده ها شبیه به لیست ریزعمل ها در فصل ۴ است. با این وجود باید دانست

جدول ۶-۸ هشت روش آدرس دهی برای دستورات عمل بارکردن

روش	قرارداد اسمبلی	انتقال ثباتی
آدرس مستقیم	LD ADR	$AC \leftarrow M[ADR]$
آدرس غیرمستقیم	LD @ADR	$AC \leftarrow M[M[ADR]]$
آدرس نسبی	LD \$ADR	$AC \leftarrow M[PC + ADR]$
عملوند بلافاصل	LD #NBR	$AC \leftarrow NBR$
آدرس دهی شاخص دار	LD ADR(X)	$AC \leftarrow M[ADR + XR]$
ثباتی	LD R1	$AC \leftarrow R1$
ثباتی غیرمستقیم	LD (R1)	$AC \leftarrow M[R1]$
خودافزایش	LD (R1) +	$AC \leftarrow M[R1], R1 \leftarrow R1 + 1$

که هر دستور ضمن اجرا باید وارد فاز برداشت شده و طی آن مقدار کد را از حافظه بخواند. همچنین طبق روش آدرس دهی، عملوندها نیز باید به ثبات‌های پردازشگر منتقل شوند. آخرین مرحله اجرای دستور در پردازشگر است و همین مرحله است که طبق بحث فصل ۴ توسط ریزعمل‌ها در ALU و یا در شیفت‌دهنده طبق شکل ۲-۸ اجرا شد. برخی دستورات حسابی هم نیاز به مدارات خاصی جهت پیاده‌سازی دارند.

دستورالعمل‌های حسابی

چهار عمل اصلی حساب عبارتند از جمع و تفریق، ضرب و تقسیم. بسیاری از کامپیوترها دستورالعمل‌های لازم را برای هر چهار عمل دارا هستند. برخی کامپیوترهای کوچک فقط جمع و احتمالاً دستورات تفریق را دارند. اعمال ضرب، تقسیم باید با نوشتن زیر روال‌های نرم افزاری تولید گردند. چهار عمل اصلی حسابی فوق برای حل فرمول‌ها و مسائل علمی پس از بیان آن‌ها براساس روش‌های محاسبات عددی کافی هستند.

لیستی از دستورات حسابی در جدول ۷-۸ دیده می‌شود. دستورافزایش، یک واحد به مقدار ذخیره شده در ثبات می‌افزاید. یکی از ویژگیهای دستورافزایش پس از اجرا این است که یک عدد دودویی متشکل از 1 در تمام بیت‌ها وقتی افزایش یابد به عددی تشکیل شده از تماماً 0 بدل می‌شود. بالعکس عددی با فقط 0 پس از کاهش به عددی تماماً 1 تبدیل می‌گردد.

دستورات جمع، تفریق، ضرب و تقسیم ممکن است برای انواعی از داده‌ها موجود باشند. فرض براین است که نوع داده در ثبات‌های پردازشگر در طول اجرای این دستورات حسابی در تعریف کد عمل گنجانده شده باشد. یک دستورالعمل حسابی ممکن است یک داده ممیز ثبات، ممیز شناور، دودویی و یا اعشاری، با دقت کم یا با دقت مضاعف باشد. انواع داده‌ها در فصل ۳ معرفی شدند.

جدول ۷-۸ دستورالعمل‌های حسابی

نام	سمبل اختصاری
افزایش	INC
کاهش	DEC
جمع	ADD
تفریق	SUB
ضرب	MUL
تقسیم	DIV
جمع با نقلی	ADDC
تفریق با قرض	SUBB
نفی (منم 2)	NEG

معمولاً نمی توان کامپیوتری برای هر سه نوع دستور جمع پیدا کرد: یکی برای اعداد صحیح دودویی، یکی برای عملوندها با ممیز شناور و یکی هم برای اعداد دهدهی. علامت اختصاری^۱ هر سه نوع دستور جمع که انواع مختلف داده ها را بکار می برند در زیر نشان داده شده است.

ADD I	جمع دو عدد صحیح دودویی
ADD F	جمع دو عدد ممیز شناور
ADD D	جمع دو عدد دهدهی بفرم BCD ^۲

الگوریتم های اعداد صحیح، ممیز شناور و اعمال حسابی دهدهی در فصل ۱۰ ارائه شده اند. تعداد بیت های هر ثبات محدود است بنابراین دقت نتایج اعمال حسابی محدودیت خواهد داشت. در برخی از کامپیوترها، مدارات سخت افزاری، اعمال محاسباتی با دقت مضاعف را ممکن می سازند بطوری که طول عملوند تا دو کلمه حافظه می رسد. بعضی کامپیوترهای کوچک نیز نرم افزار خاصی را برای محاسبات با دقت مضاعف ارائه می نمایند. فلیپ فلاپ خاصی برای ذخیره کردن رقم نقلی یک عمل جمع در نظر گرفته می شود. دستور "جمع با رقم نقلی"^۳ عمل جمع بر روی دو عملوند بعلاوه رقم نقلی حاصل از عمل قبلی را انجام می دهد. مشابهاً دستور "تفریق با قرض"^۴ دو عملوند را با توجه به رقم قرضی احتمالی از عمل قبل، از یکدیگر کم می کند. دستور نفی سازی^۵، مکمل 2 یک عدد را مشخص می نماید که عبارت است از معکوس کردن علامت یک عدد صحیح وقتی که بشکل مکمل 2 علامت دار در نظر گرفته می شود.

دستورالعمل های منطقی و دستکاری بیتی

دستورات منطقی اعمال دودویی را بر روی رشته ای از بیت های ذخیره شده در ثبات ها انجام می دهند. این دستورات برای بیت های جدا از هم و یا گروهی از بیت ها که نمایانگر اطلاعات کد شده دودویی هستند مفیدند. دستورات منطقی هر بیت از عملوند منطقی را جداگانه در نظر گرفته و آنرا بصورت یک متغیر بول تصور می کند. با کاربرد صحیح دستورات منطقی امکان تغییر بیت ها، امکان پاک کردن گروهی از بیت ها یا وارد نمودن مقادیری در بیت های عملوند ذخیره شده در ثبات یا کلمات حافظه وجود دارد.

نمونه هایی از دستورات منطقی و دستکاری بیتی در جدول ۸-۸ آورده شده اند. دستورالعمل پاک کردن^۶ سبب می شود عملوند خاصی با 0ها جایگزین شود. دستور مکمل سازی^۷، مکمل 1 عملوند را با معکوس کردن تمام بیت ها تولید می نماید. دستورات AND، OR و XOR اعمال منطقی مربوطه

1- Mnemonic	2- Binary Coded Decimal	3- Add With Carry
4- Subtract With Borrow	5- Negate	6- Clear Instruction
7- Complement Instruction		

جدول ۸-۸ دستورالعمل‌های منطقی و دستکاری بیت‌ها

نام	سبل اختصاری
پاک کردن	CLR
متمم کردن	COM
AND	AND
OR	OR
OR انحصاری	XOR
صفر کردن نقلی	CLRC
1 کردن نقلی	SETC
متمم کردن رقم نقلی	COMC
فعال کردن وقفه	EI
غیرفعال کردن وقفه	DI

را روی بیت‌های عملوند اجرا می‌نمایند. هرچند این دستورالعمل‌ها اعمال بولی را انجام می‌دهند ولی به آنها بصورت دستکاری کننده بیت‌ها باید نگریست. سه نوع دستکاری بیتی امکان پذیر است. بیت‌هایی که مورد نظر هستند می‌توانند 0 شوند، 1 شوند و یا مکمل شوند. سه دستور منطقی فوق دقیقاً برای انجام همین اهداف مورد استفاده قرار می‌گیرند.

دستور AND برای پاک کردن یک بیت یا گزیده‌ای از بیت‌های عملوند یک دستورالعمل است. برای هر متغیر بولی x رابطه‌های $x \wedge 0 = 0$ و $x \wedge 1 = x$ بیان می‌دارد که هر متغیری که با 0 AND شود، تولید 0 می‌نماید ولی اگر با 1 AND شود متغیر تغییری نمی‌یابد. بنابراین دستور AND می‌تواند برای پاک کردن بیت‌های یک عملوند، یا AND آن عملوند با عملوند دیگری که در مکان‌های مورد نظر 0هایی برای پاک نمودن بیت‌های متناظر در عملوند اول دارد بکار رود. دستور AND ماسک^۱ (پوشش) هم خوانده می‌شود زیرا در بخش انتخاب شده‌ای از عملوند بیت‌ها را ماسک (صفر) می‌کند.

دستور OR برای نشان دادن^۲ (برقرار کردن) یک بیت یا گروهی از بیت‌های عملوند مورد استفاده قرار می‌گیرد. برای هر متغیر بولی مانند x رابطه $x + 0 = x$ و $x + 1 = 1$ بیان می‌دارد که هر متغیری با 1، OR شود کمیت 1 را تولید می‌کند ولی اگر با 0، OR شود تغییری در متغیر وجود نخواهد داشت. بنابراین دستور OR می‌تواند بر روی گزیده‌ای از بیت‌های یک عملوند و با استفاده از عملوند دیگری که در بیت‌های مورد نظری از آن 1 قرار گرفته عمل کرده و بیت‌های متناظر در عملوند اول را 1 نماید.

بطور مشابه، دستور XOR بر روی بیت‌های منتخبی از عملوند مورد استفاده است. این دستور نیز با توجه به رابطه بولی $x \oplus 1 = x'$ و $x \oplus 0 = x$ عمل می‌نماید. بنابراین یک متغیر دودویی وقتی با 1، XOR شود مکمل می‌شود. مثالهای عددی که نشان‌دهنده سه عمل منطقی است در بخش (۴-۵) داده شد. چند دستور دیگر دستکاری بیت‌ها در جدول ۸-۸ دیده می‌شوند. بیت‌های خاصی همچون بیت رقم نقلی می‌توانند با دستورات مناسبی، پاک، نشانده و یا مکمل شوند. مثالی دیگر فلیپ فلاپ کنترل

1- Mask

2- Set

وقفه است که توسط دستورات دستکاری بیت‌ها توانا (فعال) و یا ناتوان (غیرفعال) می‌شود.

دستورات شیفت

دستورات شیفت، یک عملوند از جمله دستورات بسیار مفید بوده و بفرم‌های متنوعی وجود دارند. شیفت عملی است که تحت آن بیت‌های یک کلمه به چپ و یا به راست حرکت داده می‌شوند. بیتی که از انتهای کلمه وارد می‌شود نوع شیفت بکار رفته را مشخص می‌کند. دستورات شیفت ممکن است منطقی، یا حسابی و یا چرخشی باشند. در هر صورت شیفت ممکن است به راست و یا به چپ انجام شود. جدول ۸-۹ چهار نوع دستور شیفت را نشان می‌دهد. شیفت منطقی، 0 را وارد آخرین بیت می‌نماید. آخرین بیت برای شیفت به راست، سمت چپ‌ترین و برای شیفت به چپ، راست‌ترین بیت است. شیفت حسابی معمولاً با رعایت قواعد اعداد مکمل 2 صورت می‌گیرد. این قواعد در بخش ۶-۴ آورده شدند. دستور شیفت به راست حسابی باید بیت علامت در سمت چپ‌ترین مکان را حفظ کند. لذا بیت علامت همراه با بقیه عدد به سمت راست شیفت داده می‌شود، ولی خود بیت علامت تغییری نمی‌کند. این یک شیفت به راست با حفظ بیت علامت است. شیفت به چپ حسابی یک 0 را به آخرین مکان وارد می‌کند و در واقع با شیفت به چپ منطقی یکی هستند. بهمین دلیل بسیاری از کامپیوترها دستور شیفت به چپ جداگانه‌ای به هنگام وجود نوع منطقی ارائه نمی‌نمایند.

دستورات چرخش یک شیفت چرخشی را تولید می‌کنند. بیتی که از یک انتها از کلمه خارج می‌شود مفقود نشده و مجدداً از انتهای دیگر وارد می‌شود. در چرخش از طریق بیت نقلی، آنرا به منزله بیت اضافی برای ثباتی که محتوای آن چرخش داده می‌شود در نظر می‌گیرد. بنابراین چرخش به چپ همراه با رقم نقلی، بیت نقلی را به سمت راست‌ترین بیت در ثبات، و سمت چپ‌ترین بیت ثبات را به رقم نقلی انتقال می‌دهد، و همزمان با آنها کل ثبات را به چپ منتقل می‌سازد.

برخی کامپیوترها دارای قالب چند میدانی برای دستورات شیفت هستند. یک میدان مربوط به کد عمل، و بقیه نوع و تعداد شیفت‌ها را معین می‌سازد. نمونه‌ای از این دستورات می‌تواند تا پنج میدان مطابق زیر

جدول ۸-۹ دستورات شیفت

نام	سمبل
شیفت منطقی به راست	SHR
شیفت منطقی به چپ	SHL
شیفت حسابی به راست	SHRA
شیفت حسابی به چپ	SHLA
چرخش به راست	ROR
چرخش به چپ	ROL
چرخش به چپ با نقلی	RORC
چرخش به راست با نقلی	ROLC

را داشته باشد.

OP REG TYPE RL COUNT

که در آن OP عبارتست از میدان کد عمل؛ REG آدرس ثباتی است که مکان عملوند را مشخص می‌کند؛ TYPE یک میدان دوییتی برای معرفی یکی از چهارنوع شیفت است؛ RL یک میدان یک بیتی برای چپ و یاراست بودن شیفت می‌باشد و COUNT یک میدان k بیتی برای $2^k - 1$ مرتبه شیفت می‌باشد. با چنین قالبی، می‌توان نوع شیفت، جهت و تعداد شیفت را در یک دستور معین کرد.

۸-۷ کنترل برنامه

دستورالعمل‌ها همواره در مکانهای متوالی حافظه ذخیره می‌شوند. بهنگام پردازش در CPU، دستورات از حافظه‌ها متوالیاً برداشت و اجرا می‌گردند. هر بار که دستوری از حافظه برداشته شد، شمارنده برنامه افزایش می‌یابد بطوری که محتوای آن آدرس دستور بعدی خواهد بود. پس از اجرای یک عمل انتقال یا دستکاری، کنترل با کمک شمارنده برنامه که آدرس بعدی را داراست برای برداشت دستور بعدی به سیکل برداشت باز می‌گردد. از طرف دیگر، دستوراتی که از نوع کنترل برنامه هستند پس از اجرا ممکن است مقدار آدرس شمارنده برنامه را تعویض و جریان کنترل را عوض نمایند. به بیان دیگر، دستورات کنترل برنامه شرایط تعویض محتوای شمارنده برنامه را مشخص می‌نمایند، در حالیکه دستورات انتقال و دستکاری داده شرایطی برای پردازش داده را فراهم می‌کنند. تغییر در مقدار شمارنده برنامه به علت اجرای دستور کنترل برنامه سبب قطع در توالی اجرای دستورات می‌گردد. این یک ویژگی بسیار مهم در کامپیوترهای دیجیتال است که آنرا قادر می‌سازد تا کنترلی بر اجرای برنامه و انشعاب به قسمت‌های مختلف برنامه وجود داشته باشد. برخی از انواع دستورات کنترل برنامه در جدول ۸-۱۰ آورده شده‌اند. دستورات انشعاب^۱ و پرش^۲ عموماً قابل تعویض بوده و یک مفهوم دارند ولی گاهی از هر دو آنها به منظور تفکیک روشهای

جدول ۸-۱۰ دستورالعمل‌های کنترل برنامه

نام	نیم
انشعاب	BR
پرش	JMP
گذر	SKP
فراخوانی	CALL
بازگشت	RET
مقایسه (با تفریق)	CMP
آزمون (تست) با AND کردن	TST

1- Branch

2- Jump

آدرس دهی استفاده می شود. انشعاب معمولاً دستور یک آدرسی است و در زبان اسمبلی شکل BR ADR نوشته می شود که در آن ADR نام سمبلیک برای آدرس است. وقتی این دستور اجرا شود ADR به شمارنده برنامه منتقل می گردد. چون شمارنده برنامه آدرس دستور بعدی را در خود دارد دستور بعدی از ADR برداشته خواهد شد.

دستورات انشعاب و پرش ممکن است شرطی و یا غیرشرطی باشند. دستور انشعاب غیرشرطی سبب انشعاب به آدرسی دیگر بدون هرگونه شرطی می گردد. دستورات شرطی انشعاب با شرط برقراری شرایطی مانند مثبت یا صفر بودن نتیجه عملی، انشعاب ایجاد می نمایند. اگر شرط برقرار بود، شمارنده برنامه با آدرس انشعاب بارشده و دستور بعدی از این آدرس برداشته می شود. اگر شرط برقرار نشد شمارنده برنامه تغییر نکرده و دستور بعدی از مکان بعدی برداشته می شود.

دستور گذر^۱ نیازی به میدان آدرس ندارد و بنابراین یک دستور صفر آدرسه است. یک دستور شرطی گذر دستور بعدی را بشرط برقراری یک شرط گذر می کند. این عمل بدین ترتیب صورت می گیرد که شمارنده برنامه علاوه بر افزایش در فاز دریافت، در فاز اجرا هم افزایش می یابد. در صورتی که شرط برقرار نباشد کنترل دستور بعدی را اجرا خواهد کرد که برنامه نویس معمولاً یک دستور انشعاب غیرشرطی را در آنجا جای می دهد. بنابراین یک جفت دستور انشعاب-گذر بشرطی انشعاب انجام می دهد که شرط برقرار نباشد در حالیکه یک انشعاب شرطی، بشرطی خواهد کرد که شرط برقرار باشد. دستورات فراخوانی^۲ و بازگشت^۳ بهنگام استفاده از زیرروال^۴ بکار می روند. در مورد این دستورات بعداً در این بخش صحبت خواهد شد. دستورات مقایسه^۵ و تست^۶ مستقیماً توالی اجرا برنامه را عوض نمی کنند. این دستورات هم بعلاوه کاربردشان در برقراری شرایط برای دستور بعدی انشعاب در جدول ۱۰-۸ آورده شده اند. دستور مقایسه تفریقی را بین دو عملوند انجام می دهد ولی نتیجه عمل نگهداری نمی گردد. با این وجود بیت های وضعیت معینی بعنوان نتیجه مقدار می گیرند. بطور مشابه دستور تست عمل منطقی AND دو عملوند را اجرا می نماید و تصحیح بیت های وضعیت را بدون نگهداری نتیجه یا تغییر عملوند انجام می دهد. بیت های وضعیت معمولاً بیت نقلی، علامت، صفر و سرریز می باشند. تولید این بیت های وضعیت ابتدا بحث شده و سپس چگونگی کاربرد آنها در دستورات انشعاب نشان داده خواهد شد.

بیت های وضعیتی^۷

گاهی اوقات مناسب تر است تا مدار ALU در CPU را با یک ثبات وضعیت همراه کرد تا حالات بیت های وضعیت برای تحلیل بعدی در آن ذخیره شود. بیت های وضعیت گاهی اوقات بیت های کد

1- Skip

2- Call

3- Return

4- Subroutine

5- Compare

6- Test

7- Status Bit

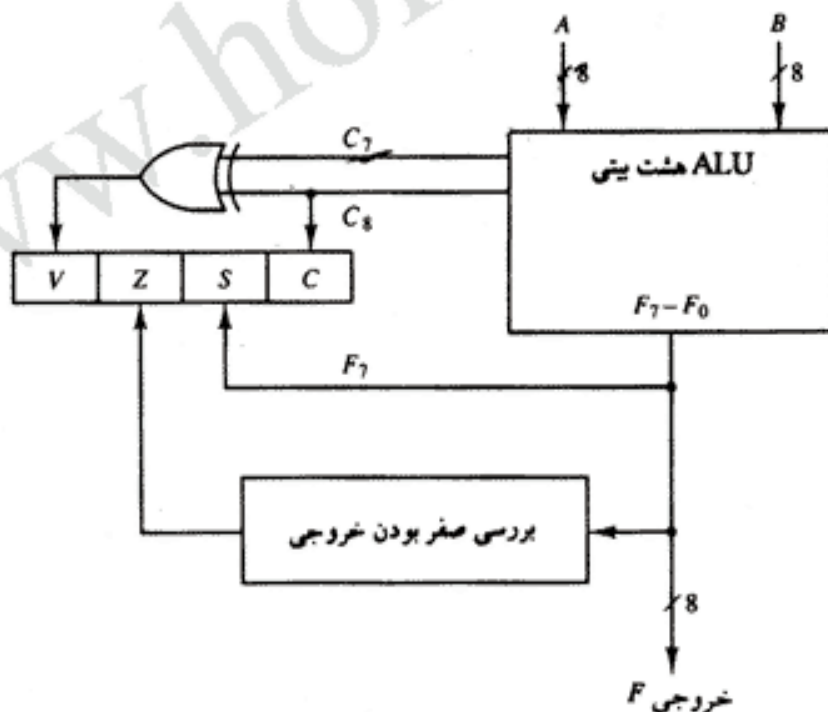
شرایط^۱ یا بیت‌های پرچم^۲ نیز خوانده می‌شوند. شکل ۸-۸ دیاگرام یک ALU هشت بیتی به همراه یک ثبات چهاربیتی وضعیت را نشان می‌دهد. این چهار بیت وضعیت با C، S، Z و V نشان داده شده‌اند. این بیت‌ها بعنوان نتیجه یک عمل انجام شده در ALU نشانده شده (1) و یا پاک می‌شوند (0).
 ۱- بیت C (نقلی) اگر رقم نقلی انتهایی $C_8=1$ شود نشانده می‌شود (1). اگر $C_8=0$ شود این بیت 0 می‌گردد.

۲- بیت S (علامت) نشانده می‌شود بشرطی که بالا رتبه‌ترین بیت یعنی F_7 در ALU برابر 1 شود. اگر بیت مذکور 0 گردد S هم 0 شود.

۳- بیت Z (صفر) هنگامی 1 می‌شود که تمام خروجی‌های ALU برابر 0 باشند. در غیر اینصورت این بیت پاک می‌شود. به بیان دیگر $Z=1$ است اگر خروجی صفر و $Z=0$ اگر خروجی غیرصفر باشد.

۴- بیت V (سرریز) نشانده می‌شود اگر، XOR دو رقم نقلی آخر برابر 1 شود و در غیر اینصورت 0 است. این شرط یک سرریز وقتی اعداد منفی بفرم مکمل 2 می‌باشند است. (بخش ۳-۳ را ببینید). برای یک ALU با هشت بیت اگر خروجی بیش از 127+ یا کمتر از 128- باشد، $V=1$ می‌گردد.

بیت‌های وضعیت پس از هر عمل در ALU می‌تواند چک شوند تا رابطه موجود بین مقادیر A و B



شکل ۸-۸ بیت‌های ثبات وضعیت

1- Condition Code Bit

2- Flag Bit

3- Zero Bit

معین گردد. اگر V پس از جمع دو عدد علامت دار نشانده شد، یک سرریز رخ داده است. اگر Z پس از یک عمل XOR نشانده شد یعنی $A=B$ است. این بدلیل وجود رابطه $x \oplus x = 0$ است که بیان می‌دارد XOR دو عدد مساوی 0 بوده و این نیز $Z=1$ را سبب می‌گردد. هر بیت در A نیز می‌تواند بوسیله پوشش دادن¹ سایر بیت‌ها چک شود و سپس بیت وضعیت Z ملاحظه گردد. مثلاً فرض کنید $A=101x1100$ باشد که در آن x بیت مورد آزمایش است. اگر $x=0$ باشد بیت وضعیت $Z=1$ می‌گردد، ولی اگر $x=1$ باشد $Z=0$ خواهد شد زیرا نتیجه غیر صفر است. عمل AND می‌تواند توسط دستور TEST، اگر حفاظت از محتویات اصلی A مورد نظر باشد، صورت گیرد.

دستورات انشعاب شرطی

جدول ۸-۱۱ لیستی از دستورات انشعابی متداول را ارائه می‌دهد. هر سمبل اختصاری تشکیل شده است از حرف B (برای انشعاب) و یک مخفف از کلمه شرط. وقتی حالت مخالف بکار رود، حرف N

جدول ۸-۱۱ دستورات عمل‌های انشعاب شرطی

سمبل	شرط انشعاب	شرط مورد تست
BZ	انشعاب به شرط صفر	$Z = 1$
BNZ	انشعاب به شرط غیر صفر	$Z = 0$
BC	انشعاب به شرط نقلی	$C = 1$
BNC	انشعاب به شرط عدم نقلی	$C = 0$
BP	انشعاب به شرط مثبت	$S = 0$
BM	انشعاب به شرط منفی	$S = 1$
BV	انشعاب اگر سرریز	$V = 1$
BNV	انشعاب اگر عدم سرریز	$V = 0$
	شرایط مقایسه بی علامت ($A-B$)	
BHI	انشعاب اگر بالاتر	$A > B$
BHE	انشعاب اگر بالاتر یا مساوی	$A \geq B$
BLO	انشعاب اگر پائین تر	$A < B$
BLOE	انشعاب اگر پائین تر یا مساوی	$A \leq B$
BE	انشعاب اگر مساوی	$A = B$
BNE	انشعاب اگر نامساوی	$A \neq B$
	شرایط مقایسه علامت دار ($A-B$)	
BGT	انشعاب اگر بزرگتر	$A > B$
BGE	انشعاب اگر بزرگتر یا مساوی	$A \geq B$
BLT	انشعاب اگر کوچکتر	$A < B$
BLE	انشعاب اگر کوچکتر یا مساوی	$A \leq B$
BE	انشعاب اگر مساوی	$A = B$
BNE	انشعاب اگر نامساوی	$A \neq B$

1- Mask

(برای نه) وارد می شود تا وضعیت 0 را بیان نماید. بنابراین BC بمعنی انشعاب بشرط $C=1$ و BNC بشرط $C=0$ است. اگر شرط بیان شده برقرار باشد کنترل به آدرسی که توسط دستور مشخص شده منتقل می گردد. در غیراینصورت کنترل با دستوری که بدنبال آمده ادامه خواهد داد. دستورات شرطی می توانند به همراه دستورات کنترل پرش¹، گذر، فراخوانی و بازگشت بکار روند.

بیت وضعیت صفر (Z) برای تست نتیجه در ALU و صفر بودن یا نبودن آن بکار می رود. بیت نقلی (C) برای وجود رقم نقلی از بالا رتبه ترین بیت در ALU استفاده می شود. این بیت به همراه دستور چرخش نیز مورد استفاده است تا بدین وسیله بیت شیفت داده شده از انتهای چرخش در آن مورد آزمایش قرار گیرد. بیت علامت (S) وضعیت بالا رتبه ترین بیت خروجی ALU را معین می نماید. $S=0$ به منزله علامت مثبت و $S=1$ به معنی علامت منفی است. بنابراین یک انشعاب بشرط مثبت بودن، بیت S را برای یافتن 0 چک می کند و انشعاب بشرط منفی بودن، S را برای یافتن 1 چک می نماید. با این وجود باید توجه داشت که این دو دستور بالاترته ترین بیت را بدون توجه به بیت علامت بودن یا نبودن چک می کنند. بیت سرریز (V) به همراه اعمال حسابی انجام شده روی اعداد علامت دار مکمل 2 مورد استفاده قرار می گیرد.

همانطور که قبلاً بیان شد دستورات مقایسه، عمل تفریق دو عملوند را از هم انجام می دهند، یعنی $A-B$ ولی نتیجه عمل به ثبات مقصد منتقل نمی گردد. در عوض بیت های وضعیت تحت تأثیر قرار می گیرند. ثبات وضعیت اطلاعات نسبی را درباره اندازه A و B فراهم می نماید. برخی کامپیوترها دستورات شرطی انشعاب را که بلافاصله بعد از دستور مقایسه قابل استفاده اند در اختیار برنامه نویس می گذارند. شرایط مورد تست بستگی به علامت دار بودن یا نبودن اعداد A و B دارد. جدول ۸-۱۱ لیستی از دستورات انشعاب شرطی را نشان می دهد. دقت کنید که ما اصطلاحات بالاتر^۲ و پائین تر^۳ را برای اعداد بدون علامت و بزرگتر^۴ و کوچکتر^۵ را برای اعداد علامت دار استفاده خواهیم کرد. بنظر می رسد نسبت اندازه ها در زیر ستون سمت راست جدول برای اعداد علامت دار و بدون علامت یکی است. با این وجود این صحیح نیست زیرا هر یک باید بطور جداگانه طبق آنچه در زیر آمده است مورد بررسی قرار گیرد.

یک ALU هشت بیتی را مطابق شکل ۸-۸ در نظر بگیرید. بزرگترین عدد بدون علامتی که می تواند در آن جای گیرد 255 است. محدوده عدد علامت دار بین $+127$ و -128 است. تفریق دو عدد چه علامت دار مکمل 2 و چه بدون علامت باشند یکی است (فصل ۳ ملاحظه شود). اجازه بدهید $A=11110000$ و $B=00010100$ باشند. در عمل $A-B$ ، ALU مکمل 2 کمیت B را با A جمع می کند.

$$\begin{array}{r} A: 11110000 \\ \bar{B} + 1: +11101100 \\ \hline A - B: 11011100 \end{array} \quad C = 1 \quad S = 1 \quad V = 0 \quad Z = 0$$

1- Jump

2- Higher

3- Lower

4- greater

5- Less Than

دستور مقایسه^۱ بیت‌های وضعیت را طبق آنچه در فوق دیده می‌شود تصحیح می‌نماید. $C=1$ می‌شود زیرا یک رقم نقلی از بالاترین رتبه خارج می‌گردد. $S=1$ است زیرا سمت چپ‌ترین بیت 1 می‌باشد. $V=0$ است چون آخرین دو بیت نقلی هر دو 1 هستند و $Z=0$ می‌باشد چون نتیجه حاصل 0 نیست. اگر ما اعداد را بدون علامت تصور کنیم، عدد اعشاری در A معادل 240 و B برابر 20 است. تفریق در مبنای ده $240-20=220$ می‌باشد. نتیجه دودویی حاصل نیز معادل 220 است. چون $240>20$ است، داریم $A>B$ و $A\neq B$. این دو رابطه از این واقعیت که $C=1$ و $Z=0$ است نیز استنتاج می‌گردد. دستوراتی که پس از این مقایسه‌ها می‌توانند انشعابی ایجاد نمایند، BHI (انشعاب اگر بزرگتر)، BHE (انشعاب اگر بزرگتر یا مساوی) و BNE (انشعاب اگر غیرمساوی) می‌باشند.

اگر ما اعداد را علامت دار تصور کنیم عدد موجود در A برابر 16- است. زیرا علامت A منفی و 11110000 مکمل 2 عدد 0001 0000 می‌باشد که معادل 16+ دهمی است. عدد دهمی B هم 20+ است. تفریق دهمی دو عدد، $(+20) - (-16)$ خواهد بود. نتیجه دودویی حاصل هم 11011100 (مکمل 2 عدد 00100100) می‌باشد که معادل دهمی 36- است. چون $(+20) < (-16)$ است داریم $A<B$ و $A\neq B$. این دو رابطه از $S=1$ (منفی) و $V=0$ (نبودن سرریز) و $Z=0$ قابل استنتاج است. دستوراتی که پس از مقایسه می‌توانند انشعابی را سبب شوند عبارتند از BLT (انشعاب اگر کوچکتر از)، BLE (انشعاب اگر کوچکتر یا مساوی) و BNE (انشعاب اگر غیرمساوی). دقت کنید که دستورات BNE و BNZ یکسانند. بطور مشابه BE و BZ نیز یکی هستند. هر یک از آنها بخاطر وضوح بیشتر مطلب سه بار در جدول ذکر شده‌اند.

با توجه به مثال فوق واضح است که نسبت اندازه‌های دو عدد بدون علامت می‌تواند از مقادیر C و Z (مسئله ۲۶-۸ ملاحظه شود) پس از دستور مقایسه معین شود. نسبت اندازه‌های اعداد علامت دار با توجه به مقادیر S و V و Z (مسئله ۲۷-۸) مشخص می‌گردد.

بعضی کامپیوترها بیت C را بیت قرض کردن پس از تفریق $A-B$ فرض می‌کنند. اگر $A\geq B$ باشد هیچ فرضی رخ نمی‌دهد ولی اگر $A<B$ باشد از بیت بالا رتبه‌تر عدد قرض گرفته خواهد شد. شرط قرض کردن این است که پس از انجام تفریق با توجه به مکمل 2 عدد B، مکمل بیت نقلی بدست آید. به این دلیل کامپیوتری که پس از یک تفریق قرضی انجام می‌دهد، پس از جمع مکمل 2 مفروق منه بیت C را مکمل نموده و آنرا قرض تلقی می‌نماید.

فراخوانی^۲ و بازگشت^۳ از زیرروال

یک زیرروال مجموعه‌ای مستقل از دستورات است که کار محاسباتی خاصی را انجام می‌دهد. در طول اجرای یک برنامه ممکن است چند مرتبه اجرای یک زیرروال تکرار گردد. هر بار که زیرروال احضار گردد یک انشعاب به شروع زیرروال انجام شده و مجموعه دستورات آن اجرا می‌گردد. پس از

1- Compare

2- Call

3- Return

اجرای زیرروال، انشعابی به برنامه اصلی صورت خواهد گرفت. دستوری که انتقال کنترل برنامه را به زیرروال بعهده دارد با نام‌های مختلفی شناخته می‌شود. معمول‌ترین نام عبارتند از: فراخوانی زیرروال، پرش به زیرروال، انشعاب به زیرروال، یا انشعاب و ذخیره آدرس می‌باشند. دستور فراخوانی زیرروال عبارتست از یک کد عمل به همراه آدرسی که شروع زیرروال را تعیین می‌نماید. دستور طی دو مرحله اجرا می‌شود: (۱) آدرس دستور بعدی موجود در شمارنده برنامه (آدرس برگشت) در مکان موقتی ذخیره می‌گردد و لذا کنترل محل بازگشت را خواهد دانست، و (۲) کنترل به شروع زیرروال منتقل می‌گردد. آخرین دستور هر زیرروال که بازگشت از زیرروال نام دارد، آدرس بازگشت را از مکان موقت به شمارنده برنامه باز می‌گرداند. نتیجه این عمل انتقال کنترل برنامه به آدرسی است که قبلاً در محل موقت ذخیره شده بود.

کامپیوترهای مختلف، مکان‌های موقت متفاوتی را برای ذخیره کردن آدرس برگشت بکار می‌برند. برخی آنرا در اولین مکان از حافظه مربوط به زیر برنامه ذخیره می‌کنند، و بعضی هم آنرا در مکان ثابتی نگه می‌دارند، دسته‌ای هم در یک ثبات پردازشگر آنرا حفظ می‌کنند و بالاخره عده‌ای هم آنرا در پشته ذخیره می‌نمایند، مزیت استفاده از پشته برای آدرس بازگشت این است که می‌توان زیرروال‌های متعددی را متوالیاً فراخواند که در این حالت آدرس‌های بازگشت می‌توانند بداخل پشته بطور متوالی ارسال گردند. دستور بازگشت از زیرروال سبب می‌شود تا محتوای حافظه فوقانی در پشته فراخوانی و به شمارنده برنامه منتقل گردد. بدین ترتیب، بازگشت همیشه به آخرین زیرروال خواهد بود. احضار یک زیرروال توسط دو ریزعمل زیر صورت می‌گیرد.

$$SP \leftarrow SP - 1$$

کاهش اشاره گر پشته

$$M[SP] \leftarrow PC$$

انتقال محتوای PC را به پشته (پوش)

$$PC \leftarrow \text{effective address}$$

انتقال کنترل به زیرروال

اگر زیرروال دیگری توسط زیرروال جاری احضار شود، آدرس بازگشت جدید بداخل پشته پوش داده می‌شود و والی آخر. دستوری که بازگشت از آخرین زیرروال را انجام می‌دهد توسط ریزعمل‌های زیرپیاده‌سازی می‌شود.

$$PC \leftarrow M[SP]$$

انتقال آدرس از پشته به PC

$$SP \leftarrow SP + 1$$

افزایش اشاره گر پشته

با استفاده از پشته برای زیرروال، تمام آدرس‌های بازگشت بصورت اتوماتیک توسط سخت افزار در یک واحد ذخیره می‌شوند و برنامه نویس لزومی ندارد مکان آدرس‌های ذخیره شده را بخاطر بسپارد. یک زیرروال بازگشتی^۱ زیرروالی است که خودش را فرا می‌خواند. اگر تنها یک ثبات یا حافظه برای

1- Recursive

ذخیره کردن آدرس برگشت بکاررود، و زیرروال بازگشتی خودش را فراخوانی نماید، آدرس ذخیره شده قبلی را تخریب خواهد کرد. معمولاً این اتفاق مطلوب نیست. برای رفع این مشکل مکان‌های مختلفی برای هر بار مورد استفاده قرار گیرد هر آدرس بازگشتی می‌تواند در پشته ذخیره شود بدون آنکه مقادیر قبلی تخریب گردند. این انتخاب، مسئله زیرروال بازگشتی را نیز حل می‌کند زیرا اولین زیرروالی که ترک می‌شود آخرین زیرروالی است که احضار شده است.

وقفه برنامه

وقفه برنامه به منظور برخورد و حل مشکلات ناشی از مسائلی است که خارج از روال معمول برنامه ایجاد می‌شوند. منظور از وقفه برنامه انتقال کنترل برنامه از برنامه جاری به برنامه دیگری بنام سرویس‌دهی وقفه است که پس از یک تقاضای داخلی یا خارجی صورت می‌گیرد. پس از اجرای برنامه سرویس‌دهی وقفه، کنترل مجدداً به برنامه اصلی باز می‌گردد.

رویه^۱ وقفه در اصل مشابه با یک فراخوانی زیرروال است، بجز در سه مورد زیر: (۱) وقفه معمولاً توسط یک سیگنال داخلی یا خارجی رخ می‌دهد و نه با اجرای یک دستورالعمل (بجز برای وقفه نرم افزاری که بعداً توضیح داده شده است)؛ (۲) آدرس برنامه سرویس‌دهی وقفه توسط سخت افزار معین می‌شود و نه بوسیله میدان آدرس یک دستور؛ (۳) رویه وقفه معمولاً تمام اطلاعات لازم را برای تعیین وضعیت CPU در آینده ذخیره می‌کند و تنها به ذخیره شمارنده برنامه اکتفا نمی‌نماید. در زیر مفاهیم سه گانه فوق را تشریح می‌نمائیم.

پس از هر وقفه در برنامه اصلی و اجرای برنامه سرویس وقفه، CPU باید دقیقاً به مکانی که به بهنگام ایجاد وقفه بود بازگردد. تنها در این صورت است که اجرای برنامه متوقف شده قادر خواهد بود بگونه‌ای که هیچ اتفاقی نیفتاده ادامه یابد. وضعیت CPU در انتهای سیکل اجرا (پس از تشخیص تقاضای وقفه) از موارد زیر معین می‌گردد.

۱- محتوای شمارنده برنامه

۲- محتوای تمام ثبات‌های پردازشگر

۳- محتوای بعضی از بیت‌های وضعیت

مجموعه کلیه بیت‌های وضعیت در CPU گاهی اوقات کلمه وضعیت برنامه^۲، PSW خوانده می‌شود. PSW در یک ثبات سخت افزاری جداگانه ذخیره می‌شود و حاوی اطلاعات وضعیتی CPU است. معمولاً این اطلاعات شامل بیت‌های وضعیت از آخرین عمل در ALU، اجازه به وقوع یا عدم وقوع وقفه و اینکه آیا CPU در وضعیت ناظری (سوپروایزری) است و یا در اختیار کاربر^۳، می‌باشد. بسیاری از کامپیوترها دارای یک سیستم عامل مقیمی هستند که تمام برنامه‌های دیگر را کنترل و نظارت می‌کند.

1- Procedure

2- Program Status Word

3- User Mode

وقتی که CPU بخشی از برنامه سیستم عامل را اجرا می‌کند در وضعیت نظارت است. تنها دستورات معینی در این وضعیت ارجحیت داشته و قابل اجرا هستند. CPU معمولاً بهنگام اجرای برنامه کاربر در وضعیت کاربر است. وضعیت کاری CPU در هر لحظه از بیت‌های خاصی از PSW قابل تشخیص است. دسته‌ای از کامپیوترها بهنگام پاسخ به وقفه شمارنده برنامه را ذخیره می‌کنند. در نتیجه برنامه سرویس به وقفه، قبل از استفاده از بیت‌های وضعیت و ثباتها باید توسط دستوراتی آنها را ذخیره کند. فقط چند کامپیوتر بهنگام پاسخ به وقفه هم شمارنده و هم تمام بیت‌های وضعیت و ثباتها را ذخیره می‌کنند. بسیاری از کامپیوترها نیز فقط شمارنده برنامه و PSW را ذخیره می‌نمایند. در بعضی موارد، برای هر یک از دونوع عملکرد نظارت و کاربری مجموعه‌های جداگانه‌ای از ثبات وجود دارد. با این ترتیب، وقتی که برنامه از وضعیت کاربری به نظارت سوئیچ می‌شود (یا بالعکس)، نیازی به ذخیره کردن محتوای ثباتها وجود نداشته و هر یک از مجموعه ثبات‌های خود استفاده می‌نمایند.

گفتم که روش یا رویه سخت افزاری پردازش وقفه مشابه با فراخوانی یک زیرروال است. در هنگام وقوع وقفه وضعیت CPU به داخل حافظه پشته پوش می‌شود و آدرس شروع روال سرویس دهی به شمارنده برنامه منتقل می‌گردد. آدرس شروع روال سرویس بجای تعیین توسط میدان آدرس یک دستور، بطور سخت افزاری معین می‌گردد. برخی کامپیوترها بهنگام وقفه به آدرس ثابتی مراجعه می‌نمایند. در این صورت برنامه سرویس باید علت وقفه را مشخص و آنرا سرویس دهد. بعضی کامپیوترها هم برای وقفه یک مکان حافظه جداگانه را تخصیص می‌دهند. گاهی اوقات هم وقفه سخت افزاری آدرس خود را برای هدایت CPU در اختیار می‌گذارد. در هر حال، CPU بایستی نوعی رویه سخت‌افزاری را برای انتخاب آدرس انشعاب به منظور سرویس دهی به وقفه داشته باشد.

CPU تا فرارسیدن انتهای اجرای یک دستور به وقفه پاسخ نمی‌دهد. کنترل، درست قبل از فاز دریافت بعدی، وجود سیگنال وقفه را چک می‌کند. اگر وقفه‌ای تقاضا شده باشد، کنترل وارد سیکل وقفه سخت افزاری می‌شود. در طول این سیکل، محتویات PC و PSW به داخل پشته پوش داده می‌شوند. سپس آدرس انشعاب مربوط به وقفه خاص موردنظر به PC منتقل می‌شود و PSW جدیدی در ثبات وضعیت بار می‌شود. بدین ترتیب، برنامه سرویس دهی می‌تواند با شروع از آدرس انشعاب و وضعیتی که PSW جدید مشخص می‌کند اجرا شود.

آخرین دستور در برنامه سرویس وقفه بازگشت از وقفه^۱ می‌باشد. وقتی که این دستور اجرا شود، پشته پاپ می‌شود تا PSW قبلی و آدرس برگشت بازیابی گردند. PSW به ثبات وضعیت و آدرس برگشت به شمارنده برنامه انتقال می‌یابد. لذا وضعیت CPU به حالت اولیه بازگشت و اجرا برنامه اصلی ادامه می‌یابد.

انواع وقفه

سه نوع وقفه عمده موجود که موجب توقف اجرای عادی یک برنامه می‌شوند در زیر طبقه‌بندی

1- Return from Interrupt

شده‌اند.

۱- وقفه خارجی

۲- وقفه داخلی

۳- وقفه نرم افزاری

وقفه خارجی از وسایل ورودی - خروجی^۱ (I/O)، از وسایل زمانبندی، از مدارهای کنترل کننده منابع تغذیه یا هر منبع خارجی دیگر تولید می‌شوند. مثالهایی از عوامل تولید وقفه خارجی عبارتند از وسایل I/O متقاضی انتقال داده، وسایل I/O اعلام کننده پایان انتقال داده، انقضای زمان یک رخداد، یا از کار افتادن منابع تغذیه است. وقفه‌های اعلان اتمام وقت^۲ از قرار گرفتن احتمالی برنامه در یک حلقه بی‌پایان ناشی می‌شوند. در نوعی وقفه حاصل از وجود اشکال در منبع انرژی، روال سرویس دهی می‌باید ظرف چند میلی ثانیه وضعیت کامل CPU را به یک حافظه انتقال دهد.

وقفه‌های داخلی از کاربرد غیرمجاز و یا اشتباه یک دستور و یا داده ناشی می‌شود. این وقفه را تله^۳ یا هم‌هم می‌خوانند. مثال‌هایی از این نوع وقفه‌ها عبارتند از، سرریز^۴ ثبات، تقسیم بر صفر، کد عمل غیرمعتبر، سرریزپشته و نادیده گرفتن مقررات حفاظتی. این خطاها معمولاً بهنگام اجرای ناقص دستور رخ می‌دهد. برنامه سرویس وقفه که وقفه داخلی را پردازش می‌کند اقدامات اصلاحی را که باید صورت گیرد تعیین می‌کند.

اختلاف بین وقفه‌های داخلی و خارجی این است که وقفه داخلی از بعضی وضعیت‌های استثنایی حاصل از خود برنامه ناشی می‌شود و نه از یک پدیده خارجی. وقفه‌های داخلی با برنامه همگام هستند در حالیکه وقفه‌های خارجی حالت غیرهمگام را با برنامه دارند. اگر برنامه مجدداً اجرا شود وقفه داخلی هر بار در همان مکان تکرار می‌شود. وقفه‌های خارجی به شرایط خارجی بستگی داشته و مستقل از برنامه در حال اجراست.

وقفه‌های خارجی و داخلی توسط سیگنالهایی که در سخت افزار CPU بوجود می‌آیند اتفاق می‌افتند. وقفه نرم افزاری با اجرای یک دستور ایجاد می‌شود. این وقفه نوع خاصی از دستور فراخوانی است که بیشتر مانند یک وقفه عمل می‌کند و نه مانند فراخوانی زیرروال. این دستور می‌تواند توسط کاربر برای ایجاد یک وقفه نرم افزاری در هر نقطه مطلوب از برنامه بکار رود. متداول‌ترین کاربرد وقفه نرم افزاری به دستورالعمل فراخوانی ناظر مربوط می‌شود. اجرای این دستور CPU را از حالت کاربری به حالت ناظر منتقل می‌نماید. در کامپیوتر اعمال معینی مانند انتقال ورودی و خروجی‌های پیچیده ممکن است به وضعیت ناظری محول شوند. برنامه نوشته توسط کاربر باید در وضعیت کاربری اجرا شود. وقتی که یک انتقال ورودی یا خروجی لازم شود، وضعیت ناظری توسط یک دستور فراخوانی

1- Input / Output

2- Time Out

3- Trap

4- Overflow

ناظر تقاضا می‌گردد. این دستور یک وقفه نرم افزاری را تولید می‌کند و در نتیجه وضعیت فعلی CPU را ذخیره و PSW جدیدی را که به وضعیت ناظر متعلق است بار می‌کند. برنامه فراخواننده باید اطلاعات را به سیستم عامل جهت انجام کار خاص تحویل نماید.

۸-۸ کامپیوتر کم دستور^۱ RISC

یکی از اهداف مهم آرشیکتک یا معماری کامپیوتر، طراحی مجموعه دستورات برای پردازنده است. مجموعه دستورات انتخابی برای یک کامپیوتر خاص تعیین کننده ساختار برنامه‌ها در زبان ماشین آن است. کامپیوترهای اولیه مجموعه دستورات ساده و کوچکی را داشتند که این خود بیشتر بعلت اجبار در کاهش سخت افزار بود. به مرور که سخت افزار دیجیتال با ابداع مدارهای مجتمع ارزان شد، دستورات کامپیوتر از نظر پیچیدگی و تعداد رو به افزایش گذاشت. بسیاری از کامپیوترها در حال حاضر بیش 100 و حتی 200 دستورالعمل را دارا هستند. بعلاوه این کامپیوترها از انواع داده‌ها و تعداد زیادی روشهای آدرس دهی سود می‌برند. سیر حرکت در پیچیدگی سخت افزار بعلت فاکتورهای مختلفی مانند ارتقای مدل‌های موجود برای افزایش زمینه‌های کاربردی، افزایش دستوراتی که تبدیل زبانهای سطح بالا را به زبان ماشین تسهیل نمایند و تلاش در طراحی ماشین‌هایی که توابع را از فرم نرم افزاری به سخت افزاری انتقال دهند، بود. یک کامپیوتر با تعداد زیادی دستور را کامپیوتر با مجموعه دستورات پیچیده^۲ (CISC) یا پردستور می‌نامند.

در اوایل 1980، تعدادی از سازندگان، طراحی کامپیوتری با دستورات کمتر و ساختاری ساده‌تر را پیشنهاد نمودند بطوری که بتوانند سریع‌تر در CPU اجرا شوند و ضمناً به استفاده مکرر از حافظه هم نیازی نداشته باشند. این نوع کامپیوترها با نام کامپیوتر با مجموعه دستورات کاهش یافته (RISC) یا کم دستور طبقه‌بندی می‌شوند. در این بخش مشخصات مهم معماری CISC و RISC را معرفی و سپس مجموعه و قالب دستورات را نشان خواهیم داد.

مشخصه‌های CISC

در طراحی یک مجموعه دستورالعمل نه تنها ساختار زبان ماشین بلکه نیازهای زبانهای سطح بالا را نیز باید در نظر گرفت. ترجمه برنامه‌های زبانهای سطح بالا به زبان ماشین، توسط کامپایلر (یا مترجم) انجام می‌شود. یکی از دلایل روی آوردن سازندگان به مجموعه دستورات CISC ساده کردن کار ترجمه فوق و ارتقای عملکرد کلی کامپیوتر است. وظیفه کامپایلر تولید مجموعه دستورات زبان ماشین برای هر عبارت در زبان سطح بالاست. اگر دستوراتی بزبان ماشین برای تبدیل مستقیم وجود داشته باشند کار ساده‌تر خواهد بود. هدف اصلی معماری CISC تهیه یک دستورالعمل برای هر عبارتی است که در زبان سطح بالا نوشته

1- Rednced Instruction Set Computer

2- Complex Instruction Set Computer

شده باشد. مثالهایی از CISC عبارتند از کامپیوتر VAX متعلق به Digital Equipment Cooperation و کامپیوتر IBM 370.

مشخصه دیگری از ساختار CISC استفاده از قالب دستورات با طول متغیر است. دستوراتی که به ثبات احتیاج دارند ممکن است دو بیتی باشند ولی دستوراتی که دو آدرس حافظه را لازم دارند می توانند برای تمام کد دستور تا پنج بایت را استفاده نمایند. اگر کلمات کامپیوتر ۳۲ بیتی (چهاربایت) باشند، اولین دستور نیمی از کلمه را اشغال می کند، در حالیکه دومین دستور به یک کلمه و یک بایت از کلمه بعدی احتیاج دارد. قرار دادن قالبهای متغیر دستورات عمل در کلمه های حافظه، با طول ثابت، نیاز به مدار دیکدر خاصی دارند که قادر است تعداد بایت را در کلمه شمرده و دستورات را برطبق تعداد بایت آنها دسته بندی نماید.

دستورات عمل های یک نمونه پردازشگر CISC امکان استفاده مستقیم از عملوندها را در حافظه فراهم می آورند. مثلاً، یک دستور ADD می تواند دارای یک عملوند در حافظه با روش آدرس دهی شاخص دار و یک عملوند در حافظه از طریق آدرس دهی مستقیم باشد. حافظه دیگری را هم می توان در دستور برای ذخیره کردن نتیجه منظور کرد. در نتیجه در طول اجرای دستور سه بار ارجاع به حافظه وجود خواهد داشت. هر چند پردازشگرهای CISC دارای دستوراتی هستند که فقط از ثباتها استفاده می نمایند. وجود روش های آدرس دهی دیگر در عملیات، کامپایل کردن زبان های سطح بالا را ساده می نماید. با این وجود، هر چه تعداد دستورات و روش های آدرس دهی در کامپیوتر بیشتر شود، مدارهای سخت افزاری بیشتری برای پیاده کردن و پشتیبانی آنها لازم است و این سبب می گردد تا سرعت محاسبات کاهش یابد. بطور خلاصه مهمترین مشخصه های یک معماری CISC عبارتند از:

- ۱- تعداد زیاد دستورات عمل ها - معمولاً بین ۱۰۰ تا ۲۵۰ دستور
- ۲- دستورات عمل هایی که کارهای خاصی انجام می دهند ولی بندرت بکار می روند.
- ۳- انواع متنوعی از روشهای آدرس دهی موجود است - معمولاً بین ۵ الی ۲۰ روش مختلف
- ۴- قالب دستورات با طول متغیر
- ۵- دستوراتی که عملوندها را در حافظه دستکاری می کند.

مشخصه های RISC

هدف از معماری RISC کوتاه کردن زمان اجرا، با کاهش مجموعه دستورات در کامپیوتر است. مهمترین مشخصه های پردازشگر RISC بقرار زیرند.

- ۱- دستورات نسبتاً کم
- ۲- روش های آدرس دهی نسبتاً کم
- ۳- دستیابی به حافظه منحصر است به دستورات بارکردن^۱ و ذخیره سازی^۲
- ۴- تمام اعمال در داخل ثبات های CPU انجام می شوند.

۵- دستورات باطول ثابت که بسادگی دیکد می شوند.

۶- اجرای دستورات در یک سیکل

۷- کنترل سخت افزاری بجای ریزبرنامه نویسی

مجموعه کوچک دستورات یک پردازشگر RISC عمدتاً از عملیات ثابت - ثابت تشکیل شده و برای دستیابی به حافظه نیز دستورات ساده بارکردن و ذخیره سازی نیز در نظر گرفته شده است. بنابراین هر عملوندی ابتدا توسط دستور بارکردن به یک ثابت داخلی منتقل می شود. تمام محاسبات روی داده های ذخیره شده در ثابت صورت می گیرد. نتایج هم بکمک دستور ذخیره سازی به حافظه بازگردانده می شود. این امکانات در معماری سبب ساده شدن مجموعه دستورات و موجب تشویق بهینه سازی عملیات دستکاری ثباتها می گردد. استفاده از تنها چند روش آدرس دهی بعلت بکارگیری آدرس دهی ثباتی برای تمام دستورات عمل هاست. سایر روشهای آدرس دهی مانند بلافصل یا نسبی نیز گاهاً بکار گرفته می شوند.

با استفاده از قالب نسبتاً ساده برای دستورات، طول دستور می تواند ثابت و برمحدوده کلمات منطبق گردد. یکی از خصوصیات مهم دستورات RISC، سادگی در دیکد شدن است. بنابراین کنترل می تواند به کد عمل و میدان ثابت بطور همزمان دست یابد. با ساده کردن دستورات و قالب آنها، مدار کنترل را می توان ساده کرد و برای بالا بردن سرعت عملیات، یک کنترل سخت افزاری برنوع ریزبرنامه نویسی شده ارجحیت دارد. مثالی از کنترل برای کامپیوتر پایه در فصل ۵ آورده شد. مثالهایی از کنترل ریزبرنامه ای نیز در فصل ۷ ارائه گردید.

از مشخصه های بارز پردازشگرهای RISC توانایی اجرای هر دستور در یک پالس ساعت است. این عمل با همزمانی انجام فازهای برداشت، دیکد و اجرای دو یا سه دستورالعمل با استفاده از روشی موسوم به پردازش خط لوله^۱ است. دستورات بارکردن و ذخیره ممکن است دو سیکل ساعت نیاز داشته باشند. زیرا دستیابی به حافظه نیاز به زمان بیشتری نسبت به اعمال ثباتی دارد. خط لوله کارا به همراه چند مشخصه دیگر اغلب به RISC منتسب اند، هرچند این ویژگی ها در معماری های غیر RISC نیز وجود دارند. دیگر مشخصه های مربوط به معماری RISC عبارتند از:

۱- تعداد قابل توجهی ثابت در واحد پردازشگر

۲- استفاده از دریچه های ثابت هم پوش^۲ برای بالا بردن سرعت دستورات فراخوانی و بازگشت از رویه

۳- خط لوله کارا در ثباتها

۴- پشتیبانی کامپایلر برای انتقال کارآی برنامه های زبان های سطح بالا به برنامه های زبان ماشین

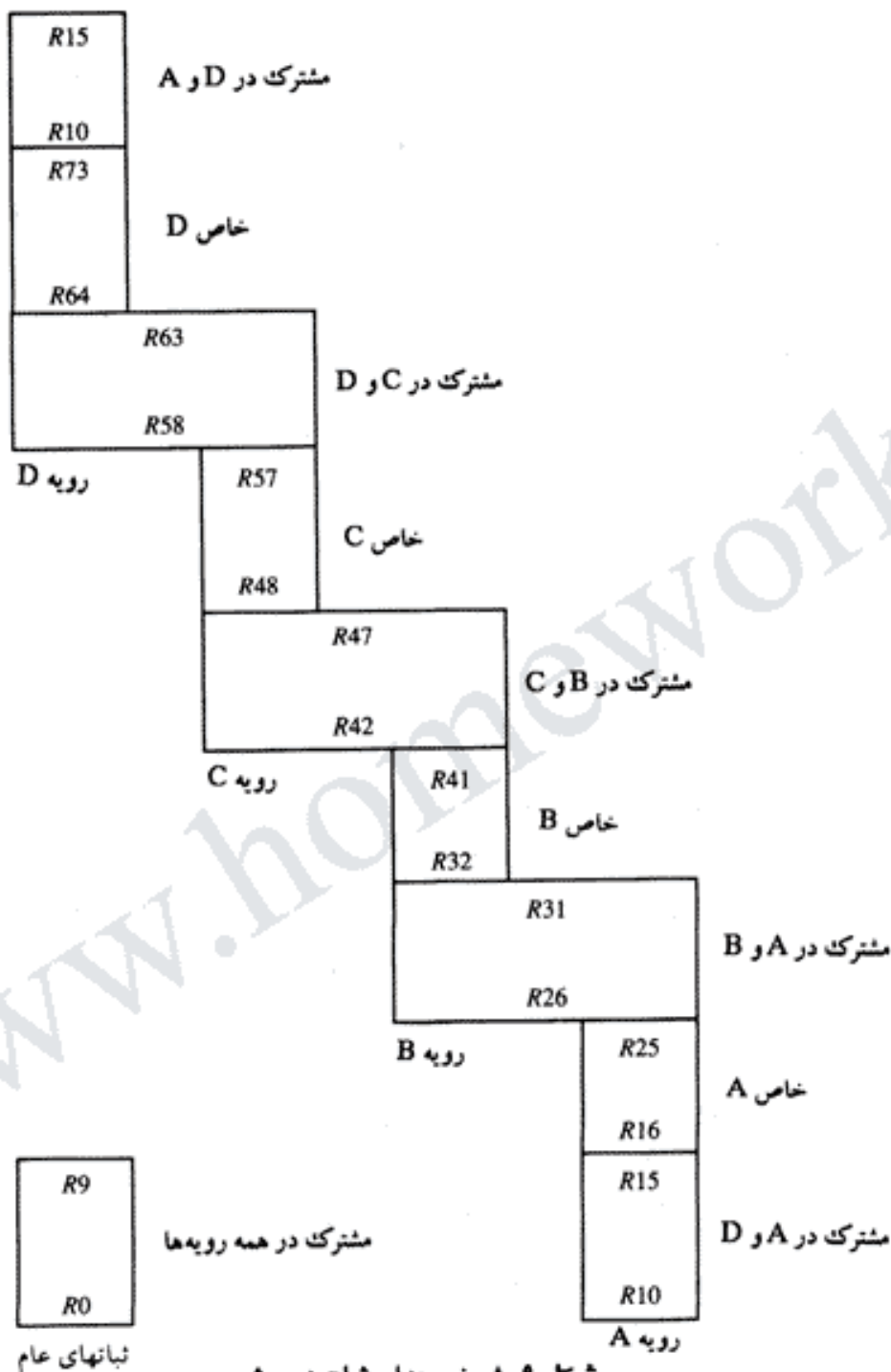
بالا بردن تعداد ثبات‌ها برای ذخیره کردن نتایج میانی و کاهش تعداد ارجاعات به عملوندها مفید است. مزیت مکان‌های ذخیره از نوع ثبات در مقابل حافظه این است که عمل انتقال در ثبات‌ها بسیار سریعتر از حافظه‌هاست. بنابراین عمل انتقال ثبات به حافظه را با نگهداری عملوند پرمصرف‌تر در ثبات می‌توان سرعت بخشید. مطالعات نشان می‌دهد که اصطلاحات در عملکرد معماری توأمأً بعلت هر دو عامل ناشی از مجموعه دستورات کاهش یافته و نیز ناشی از تعداد ثبات‌ها بوده است. به این دلیل گاهی اوقات تعداد زیادی ثبات در واحد پردازش RISC در نظر گرفته می‌شوند. در ادامه این بخش، استفاده از درجه‌های (پنجره) ثباتی هم پوش هنگام انتقال کنترل برنامه پس از یک فراخوانی رویه توضیح داده شده است. خط لوله دستورات در RISC در بخش ۵-۹ پس از تشریح مفهوم خط لوله توضیح داده خواهد شد.

دریچه‌های ثباتی همپوش

فراخوانی رویه‌ها و بازگشت، اغلب در زبانهای برنامه نویسی سطح بالا استفاده می‌شوند. وقتی که یک رویه بزبان ماشین ترجمه شود رشته‌ای از دستورات را تولید می‌کند که توسط آنها مقادیر ثبات‌ها ذخیره می‌شوند، پارامترهای لازم را برای رویه‌ها تحویل می‌نمایند و سپس زیرروالی را برای اجرای رویه فرامی‌خوانند. پس از بازگشت از رویه، برنامه مقادیر قبلی ثباتها را در مکان‌هایشان می‌نشانند، نتایج را به برنامه فرا خواننده تحویل و از زیرروال باز می‌گردد. ذخیره و بازیابی ثباتها و تحویل پارامترها و نتایج، عملیات وقتی گیری را بدنبال دارد. برخی کامپیوترها مجموعه‌هایی از ثبات‌ها را در اختیار دارند و لذا هر رویه مجموعه ثبات خاص خود را استفاده می‌نماید. این امکان سبب می‌شود تا از ذخیره و بازیابی مقادیر در ثبات‌ها خودداری شود. بعضی کامپیوترها هم از پشته حافظه‌ای برای ذخیره کردن پارامترهای لازم برای رویه‌ها استفاده می‌کنند ولی این خود نیاز به دستیابی حافظه در هر فراخوانی پشته دارد.

یکی از ویژگی‌های برخی از پردازشگرهای RISC استفاده از دریچه‌های ثباتی هم پوش برای تحویل پارامترها و اجتناب از ذخیره و بازیابی مقادیر ثبات‌هاست. فراخوانی هر رویه منجر به تخصیص یک دریچه جدید متشکل از مجموع ثبات‌ها از فایلی ثبات‌ها برای استفاده رویه جدید می‌شود. هر فراخوانی رویه، دریچه ثبات جدید را با افزایش یک اشاره‌گر فعال می‌سازد. و بالعکس یک عبارت بازگشت اشاره‌گر را کاهش و سبب فعال شدن دریچه قبلی می‌گردد. دریچه‌ها برای رویه‌های مجاور دارای ثبات‌های هم پوشند تا برای تبادل پارامترها و نتایج مورد استفاده مشترک قرار گیرند.

مفهوم دریچه‌های ثباتی هم پوش در شکل ۹-۸ تشریح شده است. سیستم دارای ۷۴ ثبات است. ثبات‌های R_0 تا R_9 ، از نوع عام هستند و پارامترهای مشترک بین همه رویه را نگه می‌دارند. بقیه ۶۴ ثبات به چهار دریچه برای جای دهی به رویه‌های A، B و C و D تخصیص یافته‌اند. هر دریچه ثبات متشکل از ده ثبات محلی (درونی) و دو مجموعه شش ثباتی مشترک با دریچه‌های مجاور است. ثبات‌های محلی برای متغیرهای محلی استفاده می‌شوند. ثبات‌های مشترک برای تعویض پارامترها و



شکل ۹-۸ پنجره‌های ثبات همپوش

نتایج با رویه مجاور مورد استفاده قرار می‌گیرند. ثبات‌های مشترک هم پوش به پارامترها اجازه می‌دهند تا بدون حرکت واقعی داده‌ها جابجا شوند. در هر لحظه تنها یک درجه ثبات فعال می‌شود و اشاره گر نیز درجه فعال را نشان خواهد داد. هر فراخوانی رویه یک درجه ثبات جدید را با افزایش اشاره گر فعال

می‌نماید. ثبات‌های فوقانی فراخواننده بر روی ثبات‌های تحتانی فراخواننده شده قرار دارند (هم پوشند) و لذا پارامترها بطور اتوماتیک از رویه فراخواننده به رویه فراخواننده منتقل می‌شوند. بعنوان مثال، فرض کنید که رویه A، رویه B را فرا بخواند. ثبات‌های R₂₆ تا R₃₁ بین هر دو رویه مشترکند، و بنابراین رویه A پارامترها را برای رویه B در این ثبات‌ها ذخیره می‌کند. رویه B از ثبات‌های محلی R₃₂ تا R₄₂ برای ذخیره متغیرهای محلی استفاده می‌کند. اگر رویه B، رویه C را فرا بخواند، پارامترها را از طریق R₄₂ تا R₄₇ به آن تحویل می‌نماید. وقتی که در انتهای محاسبات رویه B آماده بازگشت به رویه A باشد، برنامه نتایج محاسبات را در ثبات‌های R₂₆ تا R₃₁ ذخیره کرده و به دریچه ثبات‌های رویه A باز می‌گردد. توجه کنید که ثبات‌های R₁₀ تا R₁₅ بین رویه A و رویه D مشترک است زیرا چهار دریچه ساختار حلقوی دارند و لذا A با D مجاور است.

همانطور که قبلاً اشاره شد، ده ثبات عام R₀ تا R₉ مورد استفاده همه رویه‌ها هستند. هر رویه در صورت فعال بودن 32 ثبات در دسترس دارد. این تعداد شامل ده ثبات عام، ده محلی، و شش ثبات همپوش پائین رتبه و شش همپوش بالا رتبه می‌باشند. انواع دیگری از ساختارهای دریچه‌های ثباتی با ابعاد ثابت نیز امکان پذیر است، و هر یک ممکن است از نظر سایز دریچه‌ها و سایز کل فایل ثبات‌ها با دیگری اختلاف داشته باشند. بطور کلی سازمان دریچه‌های ثبات روابط زیر را دارا هستند.

۱- تعداد ثبات‌های عام G

۲- تعداد ثبات‌های محلی در هر دریچه L

۳- تعداد ثبات‌های مشترک بین دو دریچه C

۴- تعداد دریچه‌ها W

تعداد ثبات‌های قابل دسترسی توسط هر دریچه از رابطه زیر محاسبه می‌شود

$$\text{سایز دریچه} = L + 2C + G$$

تعداد ثبات‌های مورد نیاز در پردازشگر برابرست با

$$\text{فایل ثبات} = (L + C) W + G$$

در مثال شکل ۹-۸ داریم $G=10$ ، $L=10$ ، $C=6$ و $W=4$. سایز دریچه $10+12+10=32$ و فایل ثبات متشکل از 74 ثبات خواهد بود زیرا $(10+6) \times 4 + 10 = 74$.

Berkeley RISC I

یکی از پروژه‌هایی که مزایای معماری RISC را نشان داد در دانشگاه برکلی در ایالت کالیفرنیا انجام شد. Berkeley RISC I یک CPU مدار مجتمع 32 بیتی است. این CPU دارای 32 بیت آدرس بوده و

داده‌های 8، 16 و 32 بیتی را پشتیبانی می‌کند. قالب دستورات نیز 32 بیتی است و کلاً 31 دستورالعمل دارد. سه روش آدرس دهی اساسی را داراست: آدرس دهی ثباتی، عملوندهای بلافصل و آدرس دهی نسبت به PC برای دستورات انشعاب. فایل ثبات آن دارای 138 ثبات دسته‌بندی شده به ده ثبات عام و هشت درجه 32 بیتی در هر کدام است. تعداد 32 ثبات در هر درجه، سازمانی مشابه با شکل 9-8 دارد. چون در هر زمان معین فقط 32 ثبات در هر درجه قابل دستیابی هستند، قالب دستور می‌تواند دارای پنج بیت برای میدان ثبات باشد.

شکل 10-8 قالب دستورات بکار رفته برای دستورات ثبات - ثبات و دستیابی به حافظه را نشان می‌دهد. هفت بیت موجود کد عمل، یک عمل خاص را مشخص می‌کنند و بیت هشتم معین می‌کند که آیا بیت‌های وضعیت پس از عمل ALU باید بهنگام¹ شوند یا خیر. برای دستورات ثبات - ثبات میدان پنج بیتی Rd یکی از 32 ثبات را بعنوان مقصد² برای نتیجه عمل انتخاب می‌کند. عمل توسط داده‌های موجود در میدان‌های Rs و S2 معین می‌گردد. Rs یکی از ثبات‌های مبدأ³ (منبع) است. اگر بیت 13 از دستورالعمل 0 باشد پنج بیت پائین رتبه S2 ثبات مبدأ دیگر را معین می‌کند. اگر بیت 13 از دستورالعمل 1 باشد، S2 مشخص کننده یک مقدار ثابت 13 بیتی با تعمیم علامت⁴ است. بنابراین دستورالعمل دارای سه قالب آدرس است، ولی دومین مبدأ ممکن است یک ثبات و یا یک عملوند بلافصل باشد.

31	24	23	19	18	14	13	12	5	4	0
کد عمل		Rd	Rs	0	بدون استفاده			S2		
8		5	5	1	8			5		

(الف) شیوه ثباتی: (S2 یک ثبات را مشخص می‌کند)

31	24	23	19	18	14	13	12	0
کد عمل		Rd	Rs	1	S2			
8		5	5	1	13			

(ب) شیوه ثباتی بلافصل: (S2 یک عملوند را معین می‌نماید)

31	24	23	19	18	0
کد عمل		COND	Y		
8		5	19		

(ج) شیوه نسبت به PC

شکل 10-8 قالب‌های دستورالعمل‌ها در Berkeley RISC I

1- Update

2- Destination

3- Source

4- Sign Extended

دستورات دستیابی به حافظه از Rs برای تعیین آدرس 32 بیتی در ثبات استفاده می کنند و S2 یک فاصله از مبدأ است. محتوای ثبات R0 تماماً 0 است بنابراین در هر حال بعنوان کمیت صفر می تواند مورد استفاده قرارگیرد. سومین قالب دستور سه میدان آخر را با هم ترکیب کرده و یک آدرس نسبی ۱۹ بیتی Y را تشکیل داده است و عمدتاً با دستورات فراخوانی و پرش بکار می رود. میدان COND که بجای میدان Rd در دستورات عملی آمده است برای تعیین یکی از 16 شرط انشعاب استفاده می شود.

31 دستور RISC I در جدول (۱۳-۸) لیست و به سه گروه تقسیم شده اند. دستورات دستکاری داده اعمال حسابی، منطقی و شیفت را انجام می دهند. سمبل زیر ستون کد عمل و عملوند بهنگام نوشتن برنامه هایی بزیان اسمبلی بکار می روند. ستون های انتقال ثبات و توضیحات، عمل دستورات را بصورت علائم و نیز تشریحی بیان می دارند. دقت کنید که تمام دستورات دارای سه عملوند هستند. مبدأ S2 می تواند یک ثبات و یا یک عملوند بلافصل باشد که با علامت # مشخص شده است. مثلاً دستور ADD را در نظر بگیرید و ملاحظه کنید که چگونه می توان انواع اعمال را توسط آن انجام داد.

ADD R22, R21, R23	$R23 \leftarrow R22 + R21$	
ADD R22, #150, R23	$R23 \leftarrow R22 + 150$	
ADD R0, R21, R22	$R22 \leftarrow R21$	(انتقال)
ADD R0, # 150, R22	$R22 \leftarrow 150$	باردهی بلافصل
ADD R22, # 1, R22	$R22 \leftarrow R22+1$	(افزایش)

با استفاده از ثبات R0، که همواره محتوای آن 0 است، می توان محتوای یک ثبات یا یک مقدار ثابت به ثبات دیگری منتقل کرد. عمل افزایش هم با جمع ثابت 1 با یک ثبات امکان پذیر است. دستورات انتقال داده متشکل از شش دستور بارکردن، سه دستور ذخیره کردن و دو دستور انتقال کلمه PSW می باشد. محتویات ثبات PSW وضعیت CPU را نشان می دهد که عبارتست از: شمارنده برنامه، بیت های وضعیت از ALU، اشاره گرهای بکار رفته با دریچه های ثباتی و سایر اطلاعاتی که وضعیت CPU را مشخص می نماید.

دستورات بارکردن و ذخیره سازی، داده ها را بین ثبات و حافظه حرکت می دهند. دستورات بارکردن داده های علامت دار یا بدون علامت هشت بیتی (بایت) و یا 16 بیتی (کلمه کوتاه) را دارا هستند. دستورات کلمات بلند روی داده های 32 بیتی عمل می نمایند. هر چند بنظر می رسد که روش آدرس دهی ثباتی بعلاوه جابجایی در دستورات انتقال داده وجود دارد، روشهای آدرس دهی غیرمستقیم ثبات و مستقیم نیز امکان پذیر است. مثالهای زیر دستورات بارکردن با کلمات بلند را با روشهای مختلف نشان می دهد.

LDL (R22) # 150, R5	$R5 \leftarrow M[R22] + 150$
LDL (R22) # 0, R5	$R5 \leftarrow M[R22]$
LDL (R0) # 500, R5	$R5 \leftarrow M[500]$

جدول ۸-۱۲ مجموعه دستورالعمل‌های Berkeley RISC I

کد عمل	عملوندها	انتقال ثبات	شرح
دستورالعمل‌های دستکاری داده‌ها			
ADD	Rs, S2, Rd	$Rd \leftarrow Rs + S2$	جمع صحیح
ADDC	Rs, S2, Rd	$Rd \leftarrow Rs + S2 + \text{carry}$	جمع با نقلی
SUB	Rs, S2, Rd	$Rd \leftarrow Rs - S2$	تفریق صحیح
SUBC	Rs, S2, Rd	$Rd \leftarrow Rs - S2 - \text{carry}$	تفریق با نقلی
SUBR	Rs, S2, Rd	$Rd \leftarrow S2 - Rs$	تفریق معکوس
SUBCR	Rs, S2, Rd	$Rd \leftarrow S2 - Rs - \text{carry}$	تفریق با نقلی
AND	Rs, S2, Rd	$Rd \leftarrow Rs \wedge S2$	AND
OR	Rs, S2, Rd	$Rd \leftarrow Rs \vee S2$	OR
XOR	Rs, S2, Rd	$Rd \leftarrow Rs \oplus S2$	OR انحصاری
SLL	Rs, S2, Rd	$Rd \leftarrow Rs \text{ shifted by } S$	شیفت به چپ
SRL	Rs, S2, Rd	$Rd \leftarrow Rs \text{ shifted by } S$	شیفت منطقی به راست
SRA	Rs, S2, Rd	$Rd \leftarrow Rs \text{ shifted by } S$	شیفت حسابی به راست
دستورالعمل‌های انتقال داده			
LDL	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	بارکردن عملوند بلند
LDSU	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	بارکردن عملوند کوتاه بی علامت
LDSS	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	بارکردن عملوند کوتاه علامت‌دار
LDBU	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	بارکردن بایت بی علامت
LDBS	(Rs)S2, Rd	$Rd \leftarrow M[Rs + S2]$	بارکردن بایت علامت‌دار
LDHI	Rd, Y	$Rd \leftarrow Y$	بارکردن کمیت بلافاصل در بایت بالا
STL	Rd, (Rs)S2	$M[Rs + S2] \leftarrow Rd$	ذخیره عملوند بلند
STS	Rd, (Rs)S2	$M[Rs + S2] \leftarrow Rd$	ذخیره عملوند کوتاه
STB	Rd, (Rs)S2	$M[Rs + S2] \leftarrow Rd$	ذخیره بایت
GETPSW	Rd	$Rd \leftarrow PSW$	بارکردن کلمه وضعیت
PUTPSW	Rd	$PSW \leftarrow Rd$	مقداردهی به کلمه وضعیت
دستورالعمل‌های برنامه			
JMP	COND, S2(Rs)	$PC \leftarrow Rs + S2$	پرش شرطی
JMPR	COND, Y	$PC \leftarrow PC + Y$	پرش نسبی
CALL	Rd, S2(Rs)	$Rd \leftarrow PC$ $PC \leftarrow Rs + S2$	فراخوانی زیرروال
CALLR	Rd, Y	$CWP \leftarrow CWP - 1$ $Rd \leftarrow PC$ $PC \leftarrow PC + Y$	تغییر پنجره فراخوانی نسبی
RET	Rd, S2	$CWP \leftarrow CWP - 1$ $PC \leftarrow Rd + S2$	تغییر پنجره بازگشت و
CALLINT	Rd	$CWP - CWP + 1$ $Rd \leftarrow PC$	تغییر پنجره غیرفعال کردن وقفه‌ها
RETINT	Rd, S2	$CWP \leftarrow CWP - 1$ $PC \leftarrow Rd + S2$	فعال کردن وقفه‌ها
GTLPC	Rd	$CWP \leftarrow CWP + 1$ $Rd \leftarrow PC$	دریافت آخرین وضعیت PC

آدرس موثر در اولین دستور با توجه به محتویات R22 بعلاوه یک مقدار جابجایی 150 محاسبه شده است. دستور دوم دارای فاصله از مبدأ 0 است و در نتیجه دستور تبدیل به یک دستور ثباتی غیرمستقیم می‌گردد. دستور سوم از صفرهای R0 استفاده می‌نماید و بدین ترتیب دستور از انواع آدرس مستقیم خواهد بود.

دستورات کنترل برنامه با شمارنده برنامه PC همکاری می‌نمایند تا رشته عملیات برنامه را کنترل کنند. دو دستور فراخوانی (CALL) و دو دستور پرش (JUMP) وجود دارد. یکی از آنها از روش آدرس دهی شاخص بعلاوه فاصله از مبدأ استفاده می‌کند، و دومی با استفاده از 19 بیت مقدار Y بعنوان آدرس نسبی، روش نسبی نسبت به PC را بکار می‌گیرد. دستورات فراخوانی و بازگشت از یک ثبات 3 بیتی اشاره‌گر دریچه جاری (CWP)¹ استفاده کرده که به دریچه ثبات فعال جاری اشاره می‌کند. هرگاه برنامه یک رویه جدید را فراخوانی کند، CWP یک واحد کاهش یافته و به دریچه ثبات مرتبه پائین‌تر اشاره می‌نماید. پس از بازگشت، CWP یک واحد افزایش یافته و به دریچه ثبات قبلی بازمی‌گردد.

مسائل

۸-۱ یک CPU با سازمان گذرگاه مشابه با شکل ۲-۸ دارای 16 ثبات 32 بیتی، یک ALU و یک دیکدر مقصد یاب است.

الف - چند مولتی پلکسر در گذرگاه A وجود دارد و سائز هر مولتی پلکسر چقدر است.

ب - چند انتخاب کننده ورودی برای MUXA و MUXB لازم است.

پ - در دیکدر چند ورودی و چند خروجی وجود دارد.

ت - چند ورودی و خروجی داده، منجمله ورودی و خروجی رقم نقلی، در ALU وجود دارد.

ث - با فرض وجود 35 عمل در ALU، کلمه کنترل را برای سیستم فرموله کنید.

۸-۲ سیستم شکل ۲-۸ دارای تأخیرهای زمانی زیر است: 30 ns برای انتشار در MUX، 80ns برای

عمل جمع در ALU، 20ns تأخیر در دیکدر مقصدیاب و 10ns برای ورود داده به ثبات مقصد.

حداقل سیکل زمانی برای پالس ساعت چقدر است؟

۸-۳ کلمه کنترلی را که باید به پردازشگر شکل ۲-۸ اعمال شود تا اعمال زیر را انجام دهد مشخص کنید.

$$\text{الف - } R1 \leftarrow R2 + R3$$

$$\text{ب - } R4 \leftarrow R4$$

$$\text{ج - } R5 \leftarrow R5 - 1$$

$$\text{د - } R6 \leftarrow \text{Shl}R1$$

$$\text{ه - } R7 \leftarrow \text{input}$$

1- Current Window Pointer

۸-۴ عملیات جزئی قابل اجرا در پردازشگر شکل ۲-۸ را وقتی که کلمات کنترل 14 بیتی زیر اعمال می شود معین کنید.

الف) 00101001100101

ب) 00000000000000

ج) 01001001001100

د) 00000100000010

ه) 11110001110000

۸-۵ اجازه بدهید $SP = 000000$ در پشته شکل ۳-۸ باشد. چند مقدار در پشته وجود دارد اگر:

الف) $FULL = 1, EMTY = 0$

ب) $FULL = 0, EMTY = 1$

۸-۶ پشته طوری سازمان یافته است که همیشه SP به مکانی خالی در پشته اشاره می نماید. این بدان معنی است که مقدار اولیه SP در شکل ۴-۸ می تواند 4000 بوده و اولین کمیت در مکان 4000 از پشته ذخیره شود. برای اعمال پوش و پاپ عملیات جزئی را لیست نمائید.

۸-۷ عبارات محاسباتی را از میانوندی به RPN تبدیل کنید.

الف) $A * B + C * D + E * F$

ب) $A * B + A * (B * D + C * E)$

ج) $A + B * [C * D + E * (F + G)]$

د) $\frac{A * [B + C * (D + E)]}{F * (G + H)}$

۸-۸ عبارات محاسباتی زیر را از RPN به میانوندی تبدیل کنید.

الف) $A B C D E + * - /$

ب) $A B C D E * / - +$

ج) $A B C * / D - E F / +$

د) $A B C D E F G + * + * + *$

۸-۹ عبارت محاسباتی عددی زیر را به RPN تبدیل و عملیات پشته را برای بدست آوردن نتیجه نشان دهید.

$$[3+4 [10(2+6)+8]]$$

۸-۱۰ سازمان یک حافظه بشکل FIFO است. نشان دهید چگونه یک حافظه FIFO با سه شمارنده کار می کند. یک شمارنده نوشتن WC ، آدرس نوشتن در حافظه را نگه می دارد. یک شمارنده

خواندن^۱ RC آدرس خواندن در حافظه را حفظ می نماید. یک شمارنده فضای موجود در حافظه^۲، ASC، تعداد کلمات ذخیره شده در FIFO را معین می نماید. ASC برای هر کلمه ذخیره شده یک واحد اضافه می شود و برعکس برای کلمات دریافت شده یک واحد کم می گردد.

۸-۱۱ برنامه ای بنویسید که عبارت ریاضی زیر را ارزیابی کند.

$$X = \frac{A - B + C * (D * E - F)}{G + H * k}$$

الف) با استفاده از کامپیوتر با ثبات های عمومی و دستورات سه آدرس

ب) با استفاده از کامپیوتر با ثبات های عمومی و دستورات دو آدرس

ج) با استفاده از کامپیوتر نوع انباره ای با دستورات یک آدرس

د) با استفاده از کامپیوتر با سازمان پشته و دستورات صفر آدرس

۸-۱۳ واحد حافظه یک کامپیوتر 256k کلمه 32 بیتی دارد. کامپیوتر دارای قالب دستورات چهار

میدانی است: میدان کد عملیات، میدان روش آدرس دهی برای هفت روش موجود، میدان آدرس ثبات برای انتخاب یکی از 60 ثبات پردازشگر، و آدرس حافظه. قالب دستور و تعداد بیت ها در هر میدان را اگر دستور در یک کلمه حافظه باشد معین کنید.

۸-۱۴ یک دستور دو کلمه ای در حافظه و در آدرسی که با W مشخص شده ذخیره شده است. میدان

آدرس دستور (ذخیره شده در W+1) با Y معین شده است. عملوند بکار رفته در هنگام اجرای دستور در آدرس Z می باشد. یک ثبات شاخص دارای مقدار X است. نشان دهید که چگونه Z از سایر آدرس ها محاسبه می شود اگر که روش آدرس دهی دستور برابر زیر باشد.

الف) مستقیم
ب) غیرمستقیم

ج) نسبی
د) شاخص دار

۸-۱۵ یک دستور انشعاب از نوع نسبی در آدرس دهی 750 از یک حافظه ذخیره شده است. انشعاب به آدرس 500 صورت گرفته است.

الف) مقدار میدان آدرس نسبی دستور (به اعشاری) چقدر است.

ب) مقدار میدان آدرس نسبی را به دودویی در 12 بیت نشان دهید (چرا باید عدد بفرم مکمل 2 باشد).

ج) مقدار دودویی در PC را پس از فاز برداشت (واکشی) بدست آورید و مقدار 500 را محاسبه کنید. سپس نشان دهید که مقدار دودویی در PC بعلاوه آدرس نسبی محاسبه شده در قسمت ب) برابر عدد دودویی 500 است.

۸-۱۶ واحد کنترل وقتی که یک دستور با آدرس غیرمستقیم را برداشت و اجرا می کند چند بار به

- حافظه مراجعه می نماید اگر: (الف) دستور از نوع محاسباتی بوده و یک عملوند را از حافظه لازم داشته باشد. (ب) دستور از نوع انشعاب باشد.
- ۸-۱۷ میدان آدرس یک دستور با آدرس شاخص دار چه باشد تا مشابه یک دستور غیرمستقیم ثباتی گردد.
- ۸-۱۸ یک دستور در مکان 300 با میدانی در مکان 301 ذخیره شده است. میدان آدرس 400 است. یک ثبات پردازشگر R1 نیز حاوی عدد 200 است. آدرس موثر را اگر روش آدرس دهی دستور (الف) مستقیم؛ (ب) بلافصل (ج) نسبی؛ (د) غیرمستقیم ثباتی؛ (ه) شاخص دار با R1 بعنوان ثبات شاخص باشد بدست آورید.
- ۸-۱۹ با فرض داشتن یک کامپیوتر 8 بیتی، یک جمع با دقت مضاعف برای اعداد بدون علامت 32 بیتی زیر با استفاده از دستور جمع با رقم نقلی را انجام دهید. هر بایت بصورت عدد دو رقمی مبنای 16 می باشد.
- $$(6E\ C3\ 56\ 78) + (13\ 55\ 6B\ 8F)$$
- ۸-۲۰ برای دو رشته اعداد دودویی 10011100 و 10101010 اعمال AND، OR و XOR را انجام دهید.
- ۸-۲۱ عدد 16 بیتی 1001101011001101 مفروض است. چه عملی باید انجام داد تا:
- (الف) هشت بیت اول 0 شود
(ب) هشت بیت آخر 1 شود
(ج) هشت بیت وسط مکمل شود
- ۸-۲۲ یک ثبات 8 بیتی دارای مقدار 01111011 است و بیت نقلی نیز 1 است. هشت عمل شیفت موجود در جدول ۸-۹ را روی آنها پیاده کنید. هر بار مقدار اولیه، عدد فوق باشد.
- ۸-۲۳ اعداد علامت دار زیر را به دودویی بصورت 8 بیت نمایش دهید، 83-، 63+، 63-، الف- جمع 63+ + 83- را به دودویی انجام و نتیجه را تفسیر کنید.
ب- تفریق 83+ - 68- را به دودویی انجام و نشان دهید که سرریز رخ می دهد.
ج- عدد دودویی 68- را یکبار به راست شیفت دهید و مقدار شیفت داده شده را به دهدهی بدست آورید.
د - عدد دودویی 83- را یکبار به چپ شیفت دهید و ببینید آیا سرریز وجود دارد یا نه؟
- ۸-۲۴ نشان دهید مداری که خروجی 0 را در شکل ۸-۸ چک می کند یک گیت NOR است.
- ۸-۲۵ یک کامپیوتر 8 بیت دارای یک ثبات R است. مقادیر بیت های وضعیت C، S و Z و V (شکل ۸-۸) را پس از اجرای هر یک از دستورات زیر معین کنید. مقدار اولیه R در هر مورد 72 در مبنای شانزده است. اعداد زیر نیز در مبنای شانزده هستند.
- (الف) عملوند بلافصل C6 را با R جمع کنید.
(ب) عملوند بلافصل 1E را با R جمع نمایید.

ج) عملوند بلا فصل $9A$ را از R کم کنید.

د) R را با R, XOR نمائید.

۸-۲۶ دو عدد بدون علامت توسط رابطه $A - B$ با هم مقایسه شده اند. بیت نقلی بعنوان بیت قرض پس از مقایسه در بیشتر کامپیوترهای تجاری در نظر گرفته می شود، بطوری که اگر $A < B$ باشد $C=1$ است. نشان دهید که نسبت اندازه A نسبت به B از وضعیت C و Z طبق جدول مربوط به این مسئله و (جدول ۸-۱۱) قابل استنتاج است.

جدول مسئله ۸-۲۶

رابطه	حالت بیت های وضعیت
$A > B$	$C = 0$ and $Z = 0$
$A \geq B$	$C = 0$
$A < B$	$C = 1$
$A \leq B$	$C = 1$ or $Z = 1$
$A = B$	$Z = 1$
$A \neq B$	$Z = 0$

۸-۲۷ دو عدد علامت دار بشکل مکمل ۲ با هم طبق رابطه $A - B$ مقایسه شده اند. با توجه به نتیجه مقایسه بیت های وضعیت S و Z و V ، ۰ یا ۱ خواهند شد. (دقت کنید که اگر سرریز رخ دهد تغییر علامتی وجود خواهد داشت). نشان دهید که اندازه نسبی A نسبت به B با توجه به بیت های وضعیت در جدول مربوط به این مسئله (و جدول ۸-۱۱) قابل استنتاج است.

جدول مسئله ۸-۲۷

رابطه	حالت بیت های وضعیت
$A > B$	$(S \oplus V) = 0$ and $Z = 0$
$A \geq B$	$(S \oplus V) = 0$
$A < B$	$(S \oplus V) = 1$
$A \leq B$	$(S \oplus V) = 1$ or $Z = 1$
$A = B$	$Z = 1$
$A \neq B$	$Z = 0$

۸-۲۸ قرار است یک سیستم دیجیتال با چهار ورودی C و S و Z و V با ۱۰ خروجی طراحی شود، که هر یک مربوط به یک شرط انشعاب در مسئله ۸-۲۶ و ۸-۲۷ است. (حالات مساوی و نامساوی در هر دو جدول یکی است). دیاگرام منطقی مدار با استفاده از دو گیت OR ، یک XOR و پنج معکوس کننده را رسم کنید.

- ۸-۲۹ دو عدد 8 بیتی $A=01000001$ و $B=10000100$ مفروضند.
الف- معادل دهمی هر عدد را با فرض اینکه (۱) آنها بدون علامت هستند و (۲) آنها علامت دارند بدست آورید.
ب - دو عدد دودویی را جمع کنید و نتیجه را تفسیر کنید با این فرض که (۱) آنها بدون علامتند و (۲) آنها علامت دارند.
- ج- مقادیر بیت های C و Z و S و V را پس از جمع مشخص کنید.
د- دستورات انشعابی شرطی که شرط آنها برقرار است را از جدول ۸-۱۱ لیست کنید.
- ۸-۳۰ برنامه ای در یک کامپیوتر دو عدد بدون علامت A و B را با انجام عمل $A-B$ و تصحیح بیت های وضعیت مقایسه می کند. اگر $A=01000001$ و $B=10000100$ باشد،
الف- اختلاف را محاسبه و نتیجه دودویی را تفسیر کنید.
ب - مقادیر بیت های وضعیت C (قرض) و Z را معین کنید.
ج- دستورات انشعاب شرطی را از جدول ۸-۱۱ که اتفاق می افتند^۱ لیست کنید
- ۸-۳۱ برنامه ای در یک کامپیوتر دو عدد علامت دار A و B را با انجام عمل $A-B$ و تصحیح بیت های وضعیت مقایسه می کند. اگر $A=01000001$ و $B=10000100$ باشند،
الف- اختلاف را محاسبه و نتیجه دودویی را تفسیر کنید
ب - مقادیر بیت های S و Z و V را معین کنید.
ج- دستورات انشعاب شرطی را از جدول ۸-۱۱ که اتفاق می افتند لیست کنید
- ۸-۳۲ محتویات بالاترین مکان حافظه یک پشته 5320 است. محتوای اشاره گر پشته SP نیز 3560 می باشد. یک دستور فراخوانی دو کلمه ای در آدرس 1120 حافظه قرار دارد و در آدرس 1121 هم میدان آدرس آن یعنی 6720 ذخیره شده است. محتوای PC، SP و بالاترین مکان پشته چیست؟
الف- قبل از برداشت دستور فراخوانی از حافظه
ب- پس از اجرای دستور فراخوانی
ج- پس از بازگشت از زیرروال
- ۸-۳۳ اختلافات عمده بین دستور انشعاب، فراخوانی و وقفه برنامه چیست؟
۸-۳۴ پنج مثال از وقفه خارجی و پنج مثال از وقفه داخلی را ارائه نمایید. فرق بین وقفه نرم افزاری و فراخوانی چیست؟
- ۸-۳۵ کامپیوتری با انتقال PC و PSW به یک وقفه پاسخ می دهد. سپس PSW جدیدی را از مکان IAD که توسط آدرس وقفه مشخص می شود می خواند. اولین آدرس برنامه سرویس وقفه از $IAD + 1$ خوانده می شود.
الف- رشته عملیات جزئی را برای سیکل وقفه لیست می کند.

- ب - رشته عملیات را برای بازگشت از زیرروال وقفه لیست کنید.
- ۸-۳۶ مثالهایی از کامپیوترهای دارای قالب دستورات متغیر عبارتند از IBM 370 و VAX11 و Intel 386. قالب دستورات یکی از آنها را با دستورات با طول ثابت RISCی مقایسه کنید.
- ۸-۳۷ سه کامپیوتر از درجه‌های ثابت زیر استفاده می‌کنند. سایز درجه‌ها و تعداد کل ثابت در هر ثابت چقدر است؟

	کامپیوتر ۱	کامپیوتر ۲	کامپیوتر ۳
ثبات‌های عام	10	8	16
ثبات‌های محلی	10	8	16
ثبات‌های مشترک	6	8	16
تعداد درجه‌ها	8	4	16

- ۸-۳۸ مثالی از یک دستور کامپیوتر RISCی که اعمال زیر را انجام دهید ارائه کنید.
- الف - یک ثابت را کاهش دهد.
- ب - یک ثابت را مکمل کند.
- ج - علامت محتوای ثابت را معکوس می‌کند
- د - یک ثابت را 0 کند.
- ه - یک عدد علامت‌دار را به 4 تقسیم کند.
- و - هیچ کاری انجام ندهد.
- ۸-۳۹ دستوری از RISCی را به زبان اسمبلی بنویسید که سبب پرش به آدرس 3200 گردد بشرطی که بیت وضعیت $Z=1$ باشد.
- الف - با استفاده از روش بلافصل
- ب - با استفاده از روش نسبی ($PC = 3400$ فرض شود)

۹

پردازش خط لوله‌ای و برداری

۹-۵ خط لوله RISC

۹-۶ پردازش برداری

۹-۷ پردازشگر آرایه

۹-۱ پردازش موازی

۹-۲ خط لوله

۹-۳ خط لوله حسابی

۹-۴ خط لوله دستورالعمل

۹-۱ پردازش موازی

پردازش موازی به معنی بکارگیری تکنیک‌های متنوعی در پردازش همزمان داده‌ها است که به منظور افزایش سرعت محاسبات سیستم‌های کامپیوتری مورد استفاده قرار گرفته‌اند. یک سیستم پردازش موازی به جای پردازش متوالی دستورات، قادر است پردازش همزمان داده‌ها را برای رسیدن به سرعت پردازش بیشتر انجام دهد. مثلاً، ضمن اجرای دستوری در ALU، دستور بعدی قابل خواندن از حافظه است. سیستم ممکن است دارای دو یا چند ALU باشد و لذا قادر خواهد بود دو یا چند دستور را بطور همزمان اجرا نماید. بعلاوه سیستم ممکن است دو یا چند پردازنده داشته باشد که همزمان با هم عمل نمایند. در هر صورت همانطور که اشاره شد هدف از پردازش موازی بالا بردن سرعت پردازش کامپیوتر و افزایش دفعات پردازش در طول بازه معینی از زمان است. با بکارگیری تکنیک پردازش موازی حجم سخت افزاری افزایش می‌یابد و همراه با آن قیمت سیستم نیز بالا می‌رود. با این وجود توسعه تکنولوژی، بهای سخت‌افزاری را تا حدی پائین آورده است که در پردازش موازی کاملاً محسوس است. پردازش موازی را می‌توان از زوایای مختلفی بررسی کرد. ما در پائین‌ترین سطح، پردازش موازی و سری را از نظر ثبات‌های بکاررفته بررسی می‌کنیم. ثبات‌های شیفت‌دهنده بطور سری و به مقدار یک بیت در هر بار عمل مربوط به خود را انجام می‌دهند، در حالیکه ثبات‌ها با بارشدن موازی در هر لحظه قادرند روی تمام بیت‌های کلمه عمل کنند. در سطح بالاتر، پردازش موازی دارای واحدهای عملیاتی^۱

1- Functional unit

چندگانه است که اعمال یکسان یا متفاوتی را بطور همزمان انجام می دهند. در این حال پردازش موازی با توزیع داده ها در میان این واحدها صورت می گیرد. مثلاً واحد حساب، منطق و شیفت قابل تفکیک به سه واحد بوده و عملوندها تحت نظر واحد کنترل به هر واحد رانده می شوند.

شکل ۱-۹ یک راه تفکیک واحد اجرایی به هشت واحد تابعی (عملیاتی) که بطور موازی با یکدیگر کار می کنند را نشان می دهد. عملوند در ثباتها بسته به دستورالعمل به یکی از واحدها تحویل می گردد. عملی که در هر واحد انجام می شود در داخل چهارگوشه های دیاگرام مشخص شده اند. جمع کننده و ضرب کننده عدد صحیح روی اعداد صحیح عمل می کنند. اعمال ممیز شناور بین سه مدار که موازی با هم عمل می کنند تقسیم شده اند. اعمال منطقی، شیفت و افزایش در روی داده های مختلف بطور



شکل ۱-۹ پردازنده با واحدهای عملیاتی چندگانه

همزمان عمل می نمایند. کلیه واحدها از یکدیگر مستقل بوده و لذا یک عدد، شیفت و دیگری افزایش داده می شود. سازمان چند تابعی معمولاً به یک واحد کنترل پیچیده نیاز دارد تا کلیه فعالیتها را در میان قسمت های مختلف هماهنگ نماید.

روش های مختلف برای طبقه بندی پردازش موازی وجود دارد. می توان از دیدگاه سازمان داخلی یا از نظر اتصالات درونی بین پردازنده ها و یا از نظر جریان اطلاعات درون سیستم آنها را طبقه بندی کرد. یکی از دسته بندی ها توسط M. y. Flynn پیشنهاد گردید که طی آن سازمان یک کامپیوتر را با توجه به تعداد دستورات و داده های دستکاری شده در یک زمان مورد بررسی قرار می دهد. روال معمول یک کامپیوتر برداشت¹ دستورات از حافظه و اجرای آنها در پردازنده است. دستورات خوانده شده متوالی از حافظه رشته دستورات² را تشکیل می دهند. داده هایی نیز که رشته اعمال روی آنها انجام می شود رشته داده ها را تشکیل می دهند. پردازش موازی ممکن است روی رشته دستورات یا روی رشته داده ها یا هر دو باشد. دسته بندی Flynn کامپیوتر را به چهار گروه مهم مطابق زیر تقسیم می کند:

رشته تک دستوری، رشته تک داده ای³ (SISD)

رشته تک دستوری، رشته چند داده ای⁴ (SIMD)

رشته چند دستوری، رشته تک داده ای⁵ (MISD)

رشته چند دستوری، رشته چند داده ای⁶ (MIMD)

منظور از SISD سازمانی از یک کامپیوتر است که در آن یک واحد کنترل، یک واحد پردازنده و یک واحد حافظه وجود دارد. دستورات بترتیب اجرا می شوند و سیستم ممکن است توانایی پردازش موازی داخلی داشته و یا نداشته باشد. در این حالت پردازش موازی می تواند توسط واحدهای چندتابعی یا پردازش خط لوله صورت گیرد.

SIMD سازمانی را نشان می دهد که دارای چند واحد پردازش تحت یک واحد کنترل مشترک است. همه پردازنده ها دستورالعمل واحدی را توسط واحد کنترل دریافت می کنند ولی روی داده های مختلفی عمل می نمایند. واحد مشترک حافظه باید از چند ماژول⁷ (بخش) تشکیل شده باشد تا در یک زمان بتواند با تمام پردازنده ها ارتباط برقرار کند. MISD فقط از نظر تئوری حائز اهمیت است چون عملاً هیچ سیستمی با چنین سازمانی ساخته نشده است. سازمان MIMD به کامپیوترهایی اختصاص دارد که توان پردازش چندین برنامه را در یک زمان دارند. اغلب سیستم های چند پردازنده ای و سیستم های چند

1- Fetch

2- Instruction Stream

3- Single Instruction Stream Single Data Stream

4- Single Instruction Stream Multiple Data Stream

5-Multiple Instruction Stream Single Data Stream

6-Multiple Instruction Stream Multiple Data Stream

7- Module

کامپیوتری در این دسته طبقه بندی می شوند.

دسته بندی Flynn بستگی به اختلافات بین عمل واحد کنترل و واحد پردازش داده دارد. این دسته بندی بیشتر به مشخصات رفتاری سیستم کامپیوتر می پردازد تا به اتصالات درونی آن. یک نوع پردازش موازی که منطبق بر تقسیم بندی Flynn نیست روش خط لوله^۱ است. دو نمونه موجود از این طبقه بندی یکی پردازنده های آرایه ای^۲ SIMD بحث شده در بخش ۷-۹ و دیگری چند پردازنده MIMD ارائه شده در فصل ۱۳ است.

در این فصل ما پردازش موازی را تحت عنوان های اصلی زیر بررسی خواهیم کرد:

۱- پردازش خط لوله

۲- پردازش برداری

۳- پردازنده های آرایه ای

پردازش خط لوله یک تکنیک پیاده سازی است که در آن جزء (زیر) اعمال^۳ محاسبه ها یا فازهای سیکل دستورالعمل بطور هم پوش^۴ (همزمان) اجرا می شوند. پردازش برداری درباره محاسبات بردارها و ماتریس های بزرگ است. پردازنده های آرایه ای هم بر روی آرایه های بزرگی از داده ها عمل می نمایند.

۹-۲ خط لوله

خط لوله تکنیکی است که یک پردازش سری را به عملیات جزئی تفکیک می نماید و هر عمل جزئی در مقطع خاصی همزمان با سایر مقاطع اجرا می گردد. خط لوله را می توان بعنوان مجموعه ای از قطعه پردازش کننده هایی تصور کرد که در آنها اطلاعات دودویی جریان دارد. هر قطعه بخشی از پردازش را که به آن دیکته شده انجام می دهد. نتیجه حاصل از محاسبات در هر قطعه به قطعه بعدی در خط لوله منتقل می گردد. کلمه "خط لوله" بر جریانی از اطلاعات در سیستم، مشابه با خط تولید در کارخانجات اتلاق می شود. از مشخصه های خط لوله این است که چندین محاسبه در یک زمان در قطعات مختلف در حال پیشروی است. هم پوشی یا باصطلاح همزمانی اجرا بدین ترتیب تحقق می یابد که به هر قطعه در خط لوله ثابتی تعلق می گیرد. این ثابتها نوعی جداسازی را در هر قطعه ایجاد می نمایند بطوری که هر قطعه می تواند روی داده های جداگانه ای بصورت همزمان عمل نماید.

شاید ساده ترین راه در شناخت یک ساختار خط لوله این باشد که تصور کنیم هر قطعه از یک ثبات ورودی و بدنبال آن یک مدار ترکیبی ساخته شده است. ثبات داده را نگه میدارد و مدار ترکیبی جزء عملی را در ارتباط با قطعه انجام می دهد. خروجی مدار ترکیبی در یک قطعه مفروض به ثبات ورودی قطعه دیگر تحویل می گردد. به تمام ثبات ها پالس ساعتی اعمال می شود تا زمان کافی برای

1- Pipeline

2- Array Processors

3- Suboperation

4- overlap

فعالیت تمام قطعات تأمین گردد.

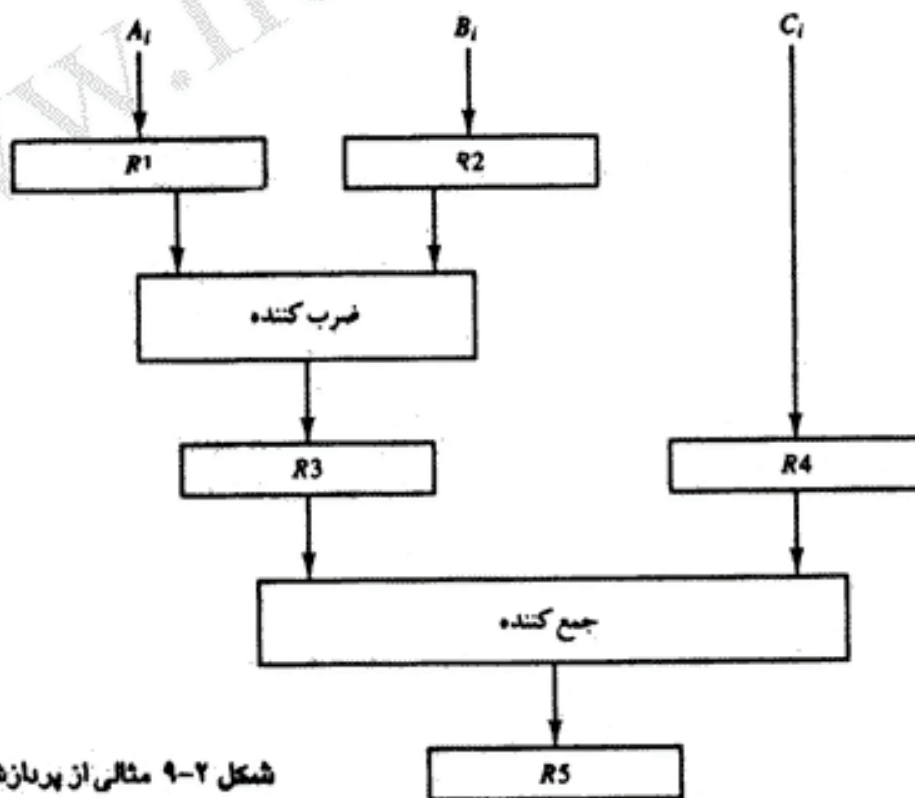
سازمان خط لوله توسط مثال ساده‌ای بیان می‌گردد. فرض کنید که می‌خواهیم یک عمل ترکیبی ضرب و جمع را با رشته‌ای از اعداد انجام دهیم.

$$A_i * B_i + C_i \quad \text{for } i = 1, 2, 3, \dots, 7$$

هر جزء (زیر) عمل در یک قطعه از خط لوله رخ می‌دهد. هر قطعه دارای یک یا دو ثبات و یک مدار ترکیبی طبق شکل ۹-۲ است. R_5 تا R_1 ثبات‌هایی هستند که داده‌های جدید را با هر پالس ساعت دریافت می‌کنند. ضرب کننده و جمع کننده مدارات ترکیبی هستند. زیر عملی که در هر قطعه از خط لوله اجرا می‌شود مطابق زیر است:

$$\begin{array}{ll} R1 \leftarrow A_i & \text{و} \quad R2 \leftarrow B_i & \text{دریافت } A_i \text{ و } B_i \\ R3 \leftarrow R1 * R2 & \text{و} \quad R4 \leftarrow C_i & \text{ضرب و دریافت } C_i \\ R5 \leftarrow R3 + R4 & & \text{جمع } C_i \text{ با حاصلضرب} \end{array}$$

پنج ثبات بازا هر پالس با داده‌های جدیدی بار می‌شوند. اثر هر پالس در جدول ۹-۱ دیده می‌شود. اولین پالس ساعت $A1$ و $B1$ را به داخل $R1$ و $R2$ انتقال می‌دهند. پالس ساعت دوم حاصلضرب $R1$ در $R2$ را به $R3$ و $C1$ را به $R4$ منتقل می‌کند. همان پالس ساعت $A2$ و $B2$ را به $R1$ و $R2$ انتقال می‌دهد. سومین



شکل ۹-۲ مثالی از پردازش لوله‌ای

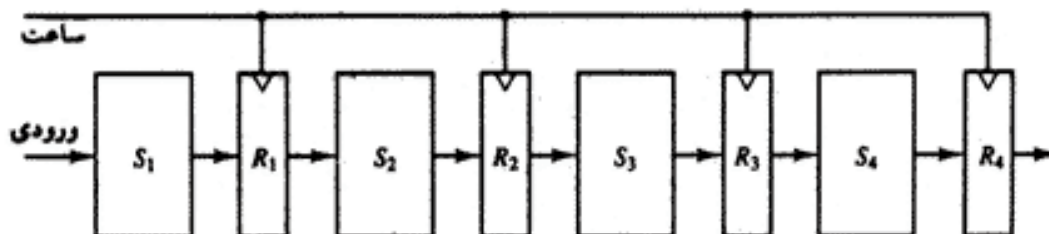
جدول ۹-۱ محتویات ثابت در مثال خط لوله

شماره	قطعه 1		قطعه 2		قطعه 3
	R1	R2	R3	R4	R5
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 + B_1$	C_1	—
3	A_3	B_3	$A_2 + B_2$	C_2	$A_1 + B_1 + C_1$
4	A_4	B_4	$A_3 + B_3$	C_3	$A_2 + B_2 + C_2$
5	A_5	B_5	$A_4 + B_4$	C_4	$A_3 + B_3 + C_3$
6	A_6	B_6	$A_5 + B_5$	C_5	$A_4 + B_4 + C_4$
7	A_7	B_7	$A_6 + B_6$	C_6	$A_5 + B_5 + C_5$
8	—	—	$A_7 + B_7$	C_7	$A_6 + B_6 + C_6$
9	—	—	—	—	$A_7 + B_7 + C_7$

پالس ساعت بطور همزمان بر روی هر سه قطعه عمل می‌کند. بدین ترتیب که A_3 و B_3 را در R_1 و R_2 می‌گذارد، حاصلضرب R_1 و R_2 را به R_3 منتقل و C_2 را در R_4 و بعلاوه حاصل جمع R_3 و R_4 را در R_5 قرار می‌دهد. بنابراین برای پرکردن خط و دریافت اولین خروجی از R_5 ، سه پالس ساعت زمان لازم است. از این پس یک خروجی جدید تولید و داده را یک قدم به پائین خط لوله می‌راند. این اعمال مادامی که داده‌های ورودی جدید بداخل سیستم جریان داشته باشند ادامه دارد. وقتی دیگر داده‌ای وجود نداشت، اعمال پالس باید تا خروج آخرین خروجی از خط لوله ادامه یابد.

بررسی‌های کلی

هر عملی که بتواند به رشته‌ای از جزء عمل‌ها با پیچیدگی یکسانی تجزیه شود می‌تواند توسط پردازنده خط لوله‌ای پیاده‌سازی شود. این تکنیک برای آن دسته از کاربردها مفید است که نیاز به تکرار عملیات یکسانی بر روی مجموعه‌ای از داده‌های مختلف دارند. ساختار عمومی یک خط لوله چهار قطعه‌ای در شکل ۳-۹ نشان داده شده است. عملوندها از تمام قطعه‌ها با ترتیب ثابتی عبور می‌کنند. هر قطعه از یک مدار ترکیبی S_i تشکیل شده که جزء عمل خاصی را روی جریان داده درون لوله انجام می‌دهد.



شکل ۳-۹ مثالی از پردازش لوله‌ای

قطعات توسط ثبات های R_i که نتایج میانی قطعات را نگه میدارند از هم جدا شده اند. اطلاعات بین قطعات توسط پالس ساعت مشترکی که به تمام ثبات ها بطور همزمان اعمال می شود جریان می یابد. ما یک تکلیف¹ را به عنوان کل یک عمل که با طی شدن تمامی قطعات خط لوله انجام می شود تعریف می کنیم. رفتار خط لوله را می توان توسط یک نمودار فاصله- زمان² تشریح کرد. این نمودار استفاده از قطعه را بصورت تابعی از زمان نشان می دهد. نمودار فاصله- زمان یک خط لوله چهار قطعه ای در شکل ۴-۹ دیده می شود. محور افقی نمودار زمان را برحسب پالس های ساعت و محور عمودی شماره قطعه را در آن نشان می دهد. نمودار، شش تکلیف T_1 الی T_6 را نشان می دهد که در چهار قطعه اجرا شده است. در ابتدا تکلیف T_1 توسط قطعه 1 انجام می شود. پس از پالس ساعت اول، قطعه 2 مشغول اجرای T_1 است، و در همان زمان قطعه 1، T_2 را انجام می دهد. به همین ترتیب، تکلیف T_1 پس از چهار پالس ساعت تکمیل شده است. از این پس، خط لوله در هر پالس یک تکلیف را کامل می کند. تعداد قطعات در سیستم اهمیتی ندارد، به محض اینکه خط لوله پر شود در هر پالس ساعت یک خروجی بدست خواهد آمد. حال خط لوله ای متشکل از k قطعه را در نظر بگیرید که در آن پالس ساعت t_p برای انجام n تکلیف بکار رفته است. اولین تکلیف T_1 زمانی برابر با kt_p را نیاز دارد تا پایان یابد زیرا در لوله تعداد k قطعه وجود دارد. $(n-1)$ تکلیف باقیمانده در لوله با سرعت یک تکلیف در هر پالس ساعت خارج می شوند و پس از زمانی برابر با $(n-1)t_p$ کلیه تکلیف ها پایان می یابند. بنابراین برای تکمیل n تکلیف با k قطعه در لوله، $k+(n-1)$ پالس ساعت لازم است. بعنوان مثال دیاگرام ۴-۹ چهار قطعه و شش تکلیف را نشان می دهد. می بینید که زمان لازم برای تکمیل تمام عملیات $9 = (6-1) + 4$ پالس ساعت است. حال یک واحد غیر خط لوله را در نظر بگیرید که همان اعمال را انجام دهد. اگر زمان تکلیف، t_n باشد زمان کل لازم برای n تکلیف برابر nt_n خواهد بود. افزایش سرعت پردازش خط لوله نسبت به یک پردازش غیر خط لوله ای با رابطه زیر تعریف می شود.

$$S = \frac{nt_n}{(K + n - 1) t_p}$$

	1	2	3	4	5	6	7	8	9	جرعه های ساعت
قطعه: 1	T_1	T_2	T_3	T_4	T_5	T_6				
2		T_1	T_2	T_3	T_4	T_5	T_6			
3			T_1	T_2	T_3	T_4	T_5	T_6		
4				T_1	T_2	T_3	T_4	T_5	T_6	

شکل ۴-۹ دیاگرام فاصله- زمان برای خط لوله

هرچه تعداد تکلیف ها اضافه شوند، n از $k - 1$ بیشتر می شود و $k + n - 1$ بسمت n میل می کند. تحت این شرط، رابطه فوق برابر است با

$$S = \frac{t_n}{t_p}$$

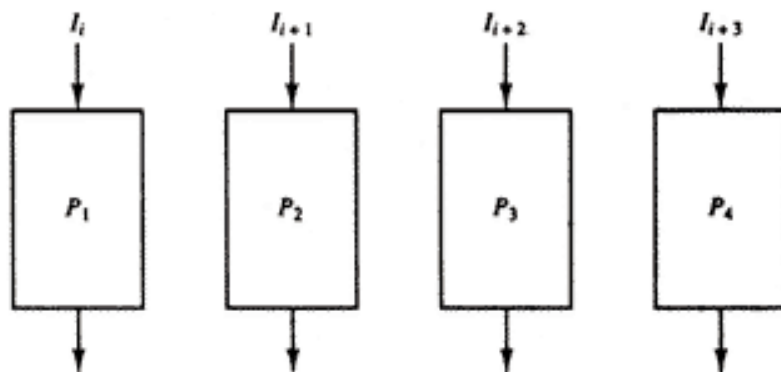
اگر فرض کنیم که زمان لازم برای غیرخط لوله همان زمان در خط لوله باشد داریم $t_n = kt_p$. لذا رابطه نسبت سرعت برابر خواهد بود با:

$$S = \frac{kt_p}{t_p} = k$$

رابطه فوق نشان می دهد که حداکثر نسبت سرعت تئوری در خط لوله می تواند k باشد که در آن k تعداد قطعه در خط لوله است.

برای درک بهتر نسبت افزایش سرعت مثال عددی زیر را در نظر بگیرید. فرض کنید زمان پردازش در هر قطعه $t_p = 20\text{ns}$ باشد. اگر خط لوله دارای $k=4$ قطعه باشد و کلاً 100 تکلیف پردازش گردد، سیستم خط لوله زمانی برابر $t_p = (k + n - 1) t_p = (4 + 99) \times 20 = 2060\text{ns}$ برای تکمیل پردازش نیاز دارد. با این فرض که $t_n = kt_p = 4 \times 20 = 80\text{ns}$ باشد یک سیستم غیرخط لوله ای زمانی برابر با $nkt_p = 100 \times 80 = 8000\text{ns}$ برای تکمیل 100 تکلیف نیاز خواهد داشت. نسبت سرعت برابرست با $8000 / 2060 = 3.88$. با افزایش تعداد تکلیف ها، افزایش سرعت بسمت 4 میل می کند که با تعداد قطعات در خط لوله برابرست. اگر $t_p = 60\text{ns}$ در نظر گرفته شود، نسبت افزایش سرعت $3 = 60 / 20$ خواهد شد.

برای تأکید بر مزیت سرعت تئوری یک پردازش خط لوله توسط یک واحد چند تابعی، لازم است که k واحد مشابه را ساخت تا به موازات عمل کنند. بدین ترتیب که می توان انتظار داشت تا k قطعه خط لوله، همانند k مدار همسان غیر خط لوله ای تحت شرایط عملکرد یکسان کار کنند. این مطلب در شکل ۹-۵ نشان داده شده است که در آن چهار مدار یکسان به موازات بسته شده اند. هر مدار P معادل با یک مدار خط لوله کار می کند. در عوض دریافت داده های متوالی، همچون مدار خط لوله، مدارات



شکل ۹-۵ واحدهای چند تابعی بصورت موازی

سوازی چهار داده ورودی همزمان را دریافت می‌کنند و چهار کار را در یک زمان انجام می‌دهند. از نظر سرعت این عملکرد معادل با عملکرد یک خط لوله چهار قطعه‌ای است. توجه داشته باشید چهار مدار شکل ۵-۹ یک سازمان SIMD را تشکیل می‌دهند چون دستورات یکسانی برای عمل روی داده‌های مختلف بطور موازی عمل می‌کنند.

دلایل متعددی برای علت عمل نکردن خط لوله در وضعیت حداکثر سرعت تئوری اش وجود دارد. قطعات مختلف ممکن است دارای زمان‌های اجرای مختلفی باشند. سیکل ساعت باید با تأخیر زمانی قطعه‌ای که حداکثر تأخیر در انتشار را دارد برابر باشد. در نتیجه سایر قطعات مجبورند وقت خود را تا پالس ساعت بعدی بیهوده تلف کنند. بعلاوه صحیح نیست تصور کنیم که خط لوله دارای زمان تأخیر مشابه با مدار غیر خط لوله است. بسیاری از ثبات‌های واسطه در مدار تک واحدی که مداری ترکیبی است لازم نیستند. با این وجود تکنیک خط لوله امکان پردازش سریعتری را در مقایسه با یک پردازش سری فراهم می‌آورد، حتی اگر سرعت ماکزیمم تئوری هرگز بدست نیاید.

در طراحی کامپیوتر دو زمینه برای کاربرد خط لوله وجود دارد. خط لوله حسابی اعمال حسابی را بصورت جزء عمل‌ها^۱ برای اجرا در قطعات خط لوله تقسیم می‌کند. خط لوله دستورالعمل روی رشته‌ای از دستورات عمل می‌کند و ضمن آن فازهای برداشت، دیکد و اجرای سیکل دستورالعمل را تماماً اجرا می‌نمایند. دو نوع خط لوله فوق در زیر توضیح داده شده‌اند:

۹-۳ خط لوله حسابی

خط لوله حسابی معمولاً در کامپیوترهای بسیار سریع یافت می‌شوند و برای پیاده‌سازی اعمال ممیز شناور، ضرب اعداد با ممیز ثابت و سایر محاسباتی از این قبیل که در مسائل علمی یافت می‌شوند بکار می‌روند. یک ضرب کننده خط لوله‌ای در اصل یک ضرب کننده آرایه‌ای مانند شکل ۱۰-۱۰ می‌باشد که در آن جمع کننده‌های خاصی برای کاهش زمان تأخیر انتشار در محاسبه حاصلضرب‌های جزیی طراحی شده‌اند. اعمال ممیز شناور بسادگی به جزء عمل‌ها طبق آنچه در بخش ۵-۱۰ آمد تفکیک می‌گردد. در این جا مثالی از یک خط لوله برای جمع و تفریق شناور را نشان می‌دهیم.

ورودی‌های خط لوله جمع کننده ممیز شناور دو عدد دودویی ممیز شناور نرمالیزه هستند

$$X = A \times 2^a$$

$$Y = B \times 2^b$$

A و B دو عدد کوچکتر از یک هستند که ماتیس‌ها را تشکیل می‌دهند و a و b نماهای آنها می‌باشند. جمع و تفریق ممیز شناور را می‌توان در چهار قطعه انجام داد، شکل ۶-۹. ثبات‌هایی که با R مشخص شده‌اند بین قطعات قرار گرفته‌اند تا نتایج میانی را ذخیره کنند. جزء عمل‌های اجرا شده در



شکل ۶-۹ خط لوله برای جمع و تفریق ممیز شناور

قطعات عبارتند از:

۱- مقایسه نماها

۲- همردیف کردن^۱ مانتیسها^۲

۳- جمع یا تفریق مانتیسها

۴- نرمالیزه کردن نتیجه

این اعمال در شکل ۱۵-۱۰ با کمی تغییر به منظور کاهش زمان اجرای زیرعملها آورده شده است. نماها توسط تفریق مقایسه شده تا اختلاف آنها پیدا شود. نمای بزرگتر بعنوان نمای نتیجه انتخاب می شود. تفاضل نماها نشان می دهد که چندبار مانتیس مربوط به نمای کوچکتر باید به راست شیفت داده شود. به این ترتیب دو مانتیس هم ردیف می شوند. باید توجه داشت که برای کاهش زمان شیفت، مدار شیفت دهنده بصورت مدار ترکیبی تهیه می شود. دو مانتیس در قطعه سوم جمع یا تفریق می شوند. نتیجه در قطعه چهار نرمالیزه می شود. وقتی که یک سرریز^۳ رخ دهد مانتیس مجموع یا تفاضل یکبار به راست شیفت یافته و نما یک واحد افزایش می یابد. اگر یک فروریز^۴ رخ دهد تعداد صفرهای جلو مانتیس بیانگر تعداد شیفت به چپ در مانتیس بوده و عددی که باید از نما کسر شود را معین می کند. مثال عددی زیر می تواند ریز عملهای انجام شده در هر قطعه را مشخص نماید. به منظور سادگی، هرچند که در شکل ۶-۹ اعداد دودویی بکار رفته اند، از اعداد دهدهی استفاده می کنیم. دو عدد ممیز شناور نرمالیزه زیر را در نظر بگیرید.

$$X = 0.9504 \times 10^3$$

$$Y = 0.8200 \times 10^2$$

دو نما در اولین قطعه از یکدیگر کسر می شوند تا $1 = 2 - 3$ بدست آید. نمای بزرگتر یعنی 3، بعنوان نمای نتیجه انتخاب شده است. قطعه بعدی مانتیس Y را به راست شیفت می دهد تا روابط زیر بدست آیند:

$$X = 0.9504 \times 10^3$$

$$Y = 0.0820 \times 10^3$$

باین ترتیب دو مانتیس هم ردیف و دارای نمای یکسان می شوند. جمع دو مانتیس در قطعه 3 حاصل جمع زیر را تولید می نماید

$$Z = 1.0324 \times 10^3$$

حاصل جمع با نرمالیزه کردن آن تنظیم می گردد بطوری که دارای اولین رقم غیر صفر در قسمت اعشاری باشد. این عمل توسط شیفت به راست و افزایش نما بدست می آید

$$Z = 0.10324 \times 10^4$$

مقایسه گر، شیفت دهنده، جمع کننده، تفریق گر، افزایش دهنده، و کاهش دهنده در خط لوله ممیز شناور

1- Align

2- Mantissas

3- Overflow

4- underflow

بروسيله مدارات ترکیبی ساخته می شوند. فرض کنید که تأخیر در چهار قطعه $t_2 = 70ns$ ، $t_1 = 60ns$ ، $t_3 = 100ns$ و $t_4 = 80ns$ و ثبات مدار واسطه $t_r = 10ns$ باشد. پالس ساعت $t_p = t_3 + t_r = 110ns$ انتخاب می شود. یک مدار غیرخط لوله ممیز شناور جمع و تفریق معادل، زمان تأخیری برابر $t_n = t_1 + t_2 + t_3 + t_4 + t_r = 320$ دارد. در این حال جمع کننده خط لوله سرعتی برابر با $320 / 110 = 2.9$ برابر نسبت به جمع کننده غیرخط لوله خواهد داشت.

۹-۴ خط لوله دستورالعمل

پردازش خط لوله نه تنها بر روی رشته ای از داده ها بلکه بر روی رشته ای از دستورات نیز می تواند اجرا شود. یک خط لوله دستورالعمل دستورات متوالی را از حافظه می خواند و در همان زمان دستور قبلی را در قطعه دیگری اجرا می نماید. این سبب می شود فازهای برداشت و اجرای دستورات بر هم منطبق و همزمان گردند. یکی از اشکالات چنین طرحی این است که یک دستور ممکن است موجب انشعاب و برهم خوردن توالی گردد. در این صورت، خط لوله باید تخلیه شده و تمام دستوراتی که بعد از دستور انشعاب قرار دارند و از حافظه خوانده شده اند باید نادیده گرفته شود.

کامپیوتری را با یک واحد برداشت و یک واحد اجرای دستور در نظر بگیرید که از خط لوله دو قطعه ای طراحی شده باشد. قطعه برداشت دستور می تواند به روش بافر FIFO^۱ پیاده سازی شود. این واحد بیشتر تشکیل یک صف را می دهد تا یک پشته. هر وقت واحد اجرا در حال استفاده حافظه نباشد کنترل شمارنده برنامه را افزایش داده و مقدار آنرا برای خواندن دستورات بعدی از حافظه بکار می برد. دستورات وارد بافر FIFO شده و بعداً بر اساس ترتیب اجرا می گردند. بنابراین رشته دستورات می توانند بصورت صف درآمد و منتظر دیکد شدن و سپس توسط واحد مربوطه اجرا گردند. مکانیزم صف بندی رشته دستورات روش بهینه ای را برای کاهش زمان متوسط دستیابی به حافظه فراهم می آورد. هر وقت که در بافر FIFO فضائی وجود داشته باشد، واحد کنترل فاز برداشت دستور بعدی را آغاز می کند. بافر همانند یک صف عمل نموده و کنترل، دستورات را از آن برای واحد اجرا استخراج می کند.

کامپیوترهایی که دارای دستورات پیچیده ای هستند (پر دستور)^۲ علاوه بر فازهای برداشت و اجرا نیاز به فازهای دیگری برای تکمیل پردازش دستورات دارند. در حالات کلی تر، کامپیوتر هر دستور را با توجه به مراحل زیر پردازش می کند.

۱- برداشت دستور از حافظه

۲- دیکد دستور

۳- محاسبه آدرس موثر

۴- برداشت عملوند از حافظه

۵- اجرای دستور

1- First IN First OUT

2- RISC

۶- ذخیره نتیجه در مکان مناسب

مشکلات عدیده‌ای وجود دارند که مانع اجرای دستورات خط لوله در حداکثر سرعتشان می‌گردند. قطعات مختلف ممکن است زمان‌های متفاوتی را برای اطلاعات وارده صرف نمایند. از برخی قطعات در عملیات خاصی ممکن است پرش شوند. مثلاً یک دستور با روش آدرس ثبات نیازی به محاسبه آدرس موثر ندارد. دو یا سه قطعه ممکن است بطور همزمان نیاز به حافظه داشته باشند که در نتیجه یک قطعه مجبور است تا پایان کار دیگری با حافظه صبر کند. مشکلات و پیچیدگی‌های دستیابی به حافظه گاهی اوقات با بکارگیری دو گذرگاه حافظه برای دستیابی به دستورات و داده‌ها در دو بخش مجزا حل می‌شود. در این روش، یک کلمه دستور و یک کلمه داده بطور جداگانه از دو واحد مجزا خوانده می‌شوند. طراحی یک خط لوله دستورالعمل هنگامی بهینه خواهد بود که سیکل دستور به قطعات زمانی برابر تقسیم گردد. زمانی که هر مرحله برای تکمیل کار خود نیاز دارد بستگی به دستور و روش اجرای آن دارد.

مثال: خط لوله دستور چهار قطعه‌ای

فرض کنید که دیکد یک دستور با محاسبه آدرس موثر بتواند در یک قطعه انجام شود. بعلاوه فرض کنید که اغلب دستورات نتایج خود را در داخل یک ثبات پردازنده قرار دهند. بنابراین اجرای دستور و ذخیره سازی نتایج بتواند در یک قطعه صورت گیرد. این فرضیات خط لوله دستور را به چهار قطعه کاهش می‌دهد. شکل ۷-۹ چگونگی پردازش دستور در CPU را با یک خط لوله چهار قطعه‌ای نشان می‌دهد. ضمن اینکه یک دستور در حال اجرا در قطعه ۴ است، دستور بعدی مشغول دریافت عملوند از حافظه در قطعه ۳ می‌باشد. آدرس موثر نیز ممکن است در یک مدار محاسباتی جداگانه برای سومین دستور محاسبه گردد، و هر وقت حافظه در دسترس باشد دستورات چهارم و بعد از آن قابل برداشت و جایگزینی در FIFO خواهند بود. بنابراین بطور همزمان تا چهار زیرعمل در سیکل دستور می‌توانند با هم انجام شوند و تا چهار دستور مختلف نیز در حال پیشروی برای اجرای همزمان می‌باشند.

گاهی اوقات یکی از دستورات ممکن است از نوع کنترل برنامه باشد که منجر به انشعاب و خروج عادی از ترتیب معمولی برنامه گردد. در این حال عمل نیمه تمام^۱ در دو قطعه آخر تکمیل و کلیه اطلاعات ذخیره شده در بافر دستورالعمل پاک می‌شوند. خط لوله سپس، از آدرس جدید ذخیره شده در شمارنده برنامه آغاز بکار می‌کند. بطور مشابه یک تقاضای وقفه هم وقتی تصدیق^۲ (تأیید) شد، موجب تخلیه خط لوله و آغاز از آدرس جدید می‌گردد.

شکل ۸-۹ طرز کار یک خط لوله دستورالعمل را نشان می‌دهد. زمان در روی محور افقی به قسمت‌های مساوی تقسیم شده است. چهار قطعه در دیاگرام هم با سمبل‌های اختصاری نشان داده شده‌اند.

۱- FI قطعه‌ای است که دستور را برداشت می‌کند.

۲- DA قطعه‌ای است که دستور را دیکد و آدرس موثر را محاسبه می‌نماید.



شکل ۷-۹ خط لوله چهار قطعه‌ای

مرحله:	1	2	3	4	5	6	7	8	9	10	11	12	13	
دستورالعمل:	1	FI	DA	FO	EX									
(انشعابها)	2		FI	DA	FO	EX								
	3			FI	DA	FO	EX							
	4				FI	-	-	FI	DA	FO	EX			
	5					-	-	-	FI	DA	FO	EX		
	6									FI	DA	FO	EX	
	7										FI	DA	FO	EX

شکل ۸-۹ زمانبندی خط لوله دستورالعمل

۳- FO قطعه‌ای است که عملوند را برداشت می‌کند.

۴- EX قطعه‌ای است که دستور را اجرا می‌کند.

فرض بر این است که پردازنده دارای حافظه‌های جداگانه برای دستورات و داده‌ها می‌باشد بطوری که عملیات در FI و FO بطور همزمان اجرای گردند. در صورت غیاب یک دستور انشعاب هر قطعه بر روی دستور جداگانه‌ای عمل می‌کند. بنابراین در مرحله چهارم، دستور 1 در قطعه EX اجرا می‌گردد؛ عملوند دستور 2 در قطعه FO برداشت می‌شود، دستور 3 در قطعه DA دیکد می‌شود و دستور 4 از حافظه هم در قطعه FI برداشت می‌گردد.

حال فرض کنید که دستور 3 یک دستور انشعاب باشد، به محض دیکد شدن این دستور در مرحله 4 و در قطعه DA، انتقال دستورات دیگر از FI به DA متوقف می‌شود تا دستورالعمل انشعاب در مرحله 6 اجرا شود. اگر انشعاب صورت گیرد، دستور جدیدی در مرحله 7 برداشت می‌گردد. اگر انشعابی صورت نگیرد، دستور قبلی برداشته شده در مرحله 4 می‌تواند بکار رود. سپس خط لوله تا رسیدن دستور انشعاب جدید بکار خود ادامه می‌دهد.

تأخیر دیگری که ممکن است در خط لوله رخ دهد این است که ضمن نیاز قطعه FO به حافظه برای برداشت داده، EX نیز نیاز به حافظه برای ذخیره کردن نتیجه خود داشته باشد. در این حال FO تا پایان کار EX منتظر می‌ماند.

بطور کلی سه مشکل عمده باعث انحراف دستور خط لوله از روند معمولی می‌گردند:

- 1- مشکل ناشی از منابع اطلاعات که بعلت دستیابی همزمان دو قطعه به حافظه می‌گردد. این مشکلات با بکاربردن حافظه‌های داده و دستور جداگانه حل می‌شود.
- 2- مشکل ناشی از وابستگی داده‌ها¹ نیز هنگامی رخ می‌دهد که دستوری به نتیجه دستور قبلی دیگری وابستگی داشته باشد و آن نتیجه هنوز بدست نیامده باشد.
- 3- مشکلات انشعاب از دستور انشعاب و سایر دستوراتی که PC را عوض می‌کنند ناشی می‌شود.

وابستگی به داده‌ها

مشکلی که ممکن است سبب افت سرعت در یک خط لوله دستور گردد بعلت احتمال برخورد داده‌ها یا آدرس‌ها می‌باشد. برخورد هنگامی رخ می‌دهد که دستور نمی‌تواند بعلت دستور قبلی پیشروی کند چرا که دستور قبلی کار خود را تمام نکرده است. وابستگی داده هنگامی رخ می‌دهد که دستوری، داده‌ای را که هنوز آماده نیست لازم دارد. مثلاً اگر یک دستور در قطعه FO بخواهد عملوندی را برداشت کند که در همان لحظه توسط دستور قبلی در قطعه EX در حال تولید باشد، دستور دوم باید تا در دسترس قرار گرفتن عملوند تأمل نماید. بطور مشابه نوعی وابستگی آدرس هنگامی رخ می‌دهد که آدرس عملوند، بعلت موجود نبودن اطلاعات لازم در روش آدرس‌دهی، قابل محاسبه نیست. مثلاً دستوری با روش

1- Data Dependency

آدرس دهی غیرمستقیم، اگر که دستور قبلی در حال بارکردن آدرس مزبور در ثبات باشد، نمی تواند برای دریافت عملوند جلو برود. بنابراین دستیابی به عملوند باید تا در دسترس قرار گرفتن آدرس مورد نیاز بتعویق بیفتد. کامپیوترهای خط لوله با چنین مشکلات در وابستگی داده ها به روشهای مختلفی برخورد می کنند مستقیم ترین روش ایجاد همبندهای سخت افزاری¹ می باشد. همبند مداری است که وظیفه اش شناسایی دستوراتی است که عملوندهای مبدأ² آنها مقصد عملوندهای دستورات بعدی در خط لوله است. شناسایی این دستورات سبب می شود تا دستوراتی که منابع آنها در دسترس نیستند با تأخیری کافی از پالس های ساعت بموقع اجرا شوند و بدین ترتیب مشکل مرتفع گردد. این روش ترتیب برنامه را با اضافه کردن سخت افزار و نگهداری تأخیر لازم در آن حفظ می کند.

روش دیگری که ارسال عملوند³ نام دارد سخت افزار خاصی را برای شناسایی مشکل بکار برده و سپس با جهت دهی به داده از طریق مسیرهای خاص بین قطعات از بروز مشکل جلوگیری می کند. مثلاً در عوض انتقال نتیجه حاصل از ALU به ثبات مقصد، استفاده از آن را در دستور بعدی بعنوان منبع چک می کند و در صورت لزوم نتیجه را مستقیماً به ورودی ALU فرستاده و بدین ترتیب فایلهای ثباتها را کنار می گذارد. این روش به مسیرهای سخت افزاری اضافی از طریق مولتی پلکسرها، علاوه بر مدار شناسایی اشکال، نیاز دارد.

روشی که در برخی کامپیوترها بکار برده می شود سپردن مسئولیت رفع مشکل به کامپایلر می باشد که وظیفه تبدیل زبانهای سطح بالا را به زبان ماشین بعهده دارد. کامپایلر برای چنین کامپیوترهایی چنان طراحی شده تا وجود یک مشکل در داده را شناسایی کرده و با اجرای دستورات "هیج کار"⁴ (NO-OP) و تغییر ترتیب اجرای دستورالعملها، تأخیر لازم را در بار کردن داده ها بوجود آورد. این روش بنام بار کردن با تأخیر⁵ خوانده می شود. مثالی از بارکردن با تأخیر در بخش بعدی آورده شده است.

دستکاری دستورالعمل های انشعاب

یکی از مسائل مهم در عملکرد خط لوله دستورالعمل، وقوع یک دستور انشعاب است. دستور انشعاب می تواند شرطی و یا غیر شرطی باشد. یک انشعاب بدون شرط همواره ترتیب اجرای برنامه را با بارکردن آدرس هدف در شمارنده برنامه عوض می کند. در یک انشعاب شرطی، کنترل دستور مقصد را اگر شرط برقرار شده باشد انتخاب می نمایند در غیر اینصورت دستور بعد از انشعاب اجرا خواهد شد. همانطور که قبلاً اشاره شد، دستورات انشعاب ترتیب اجرای رشته دستورات را شکسته و مشکلاتی را در کار خط لوله دستور ایجاد می نماید. کامپیوترهای خط لوله ای تکنیک های سخت افزاری مختلفی را برای حداقل کردن افت سرعت عملکرد حاصل از دستورات انشعاب بکار می برند.

یکی از روشهای دستکاری دستورات انشعاب دریافت زود هنگام دستور مقصد علاوه بر دستور بعد

1- Hardware interlock

2- Source operand

3- Operand forwarding

4- NO Operation, NO-OP

5- Delayed Load

از انشعاب است. هر دو دستور تا ختم اجرای دستور انشعاب ذخیره می‌شوند. اگر شرط انشعاب برقرار باشد خط لوله از دستور مقصد انشعاب ادامه کار می‌دهد. روش کامل‌ترین است که از هر دو جهت عمل برداشت دستورات ادامه یابد تا تصمیم انشعاب اتخاذ گردد. در آن زمان کنترل درباره رشته صحیح دستورات عمل‌ها برای ادامه برنامه تصمیم می‌گیرد.

امکان دیگر بکارگیری بافر مقصد انشعاب^۱ یا BTB می‌باشد. BTB یک حافظه اشتراکی^۲ است (بخش ۴-۱۲ ملاحظه شود) که در قطعه برداشت خط لوله کار گذاشته می‌شود. هر ورودی به BTB متشکل از آدرس انشعاب اجرا شده قبلی و دستور مقصد برای آن انشعاب است. این بافر چند دستور بعد از انشعاب را نیز ذخیره می‌نماید. وقتی که خط لوله دستور انشعاب را دیکد می‌کند BTB را برای آدرس دستورات عمل جستجو می‌نماید. اگر آدرس را در BTB بیابد، دستور مستقیماً در دسترس است و برداشت پیش‌هنگام دستور برای مسیر جدید ادامه می‌یابد. اگر دستور در BTB نباشد، خط لوله به رشته دستور جدیدی شیفت نموده و دستور مقصد را در BTB ذخیره می‌نماید. مزیت این روش این است که دستورات انشعابی که قبلاً اتفاق افتاده‌اند بدون توقف در خط لوله موجودند.

روش دیگر با کمی تغییر با BTB، روش بافر حلقه^۳ است. این بافر یک فایل ثبات کوچک و سریع است که تحت نظر قطعه برداشت دستور خط لوله می‌باشد. وقتی که یک حلقه در برنامه شناسایی شود در یک بافر حلقه به طور کامل همراه با انشعابات ذخیره می‌شود. بنابراین حلقه برنامه می‌تواند مستقیماً و بدون دستیابی به حافظه اجرا شود تا اینکه آخرین انشعاب موجب خروج از حلقه شود.

روش بعدی بکاررفته توسط برخی کامپیوترها، پیش‌بینی انشعاب^۴ است. یک خط لوله با پیش‌بینی انشعاب نتیجه حاصل از انشعاب را قبل از اجرای آن بکمک مدارهای منطقی اضافی پیش‌بینی می‌نماید. خط لوله سپس شروع به دریافت پیش‌هنگام رشته و دستورات مسیر پیش‌بینی شده می‌نماید. یک پیش‌بینی صحیح زمان تلف شده بوسیله انشعاب را از بین می‌برد.

روشی که توسط اغلب پردازنده‌های RISC مورد استفاده است انشعاب تأخیر داده شده^۵ می‌باشد. در این روش، کامپایلر دستورات انشعاب را شناسایی و کدهای زبان ماشین را جابجا نموده و دستورات مفیدی که خط لوله را در حال کار نگهدارد داخل آن می‌نماید. مثالی از این نوع وارد کردن دستورات "هیچ‌کار" پس از دستور انشعاب است. این عمل سبب می‌شود تا کامپیوتر در طول اجرای دستورات عمل فوق دستور مقصد را برداشت کند و در نتیجه خط لوله پیوسته در حال کار نگهداشته شود. مثالی از این انشعاب تأخیر یافته در بخش بعدی ارائه شده است.

۹-۵ خط لوله RISC

کامپیوتر کم دستور (RISC) در بخش ۸-۸ معرفی شد. از جمله مشخصات مربوط به RISC

- | | | |
|-------------------------|-----------------------|----------------|
| 1- Branch Target Buffer | 2- Associative memory | 3- Loop Buffer |
| 4- Branch Prediction | 5- Delayed Branch | |

توانایی آن در بکارگیری یک خط لوله دستورالعمل کارآمد است. سادگی مجموعه دستورات می تواند با بکارگیری تعداد محدودی زیرعمل که هر کدام در یک سیکل ساعت اجرا می شود در پیاده سازی خط لوله دستورالعمل مورد استفاده قرار گیرد. به دلیل ثابت بودن طول قالب دستورالعمل ها عمل دیکد، همزمان با انتخاب ثبات انجام می شود. تمام دستورات دستکاری داده از نوع ثبات به ثبات هستند. چون تمام داده ها در ثبات ها قرار دارند نیازی به محاسبه آدرس موثر یا برداشت عملوند از حافظه نیست. بنابراین خط لوله دستورالعمل با دو یا سه قطعه قابل پیاده شدن است. یک قطعه دستور را از حافظه برنامه برداشت می کند، و دیگری دستور را در ALU اجرا می کند. می توان قطعه سوم را هم برای ذخیره کردن نتیجه عمل ALU در یک ثبات مقصد بکار برد.

دستورات انتقال داده در RISC به دستورات بارکردن^۱ و ذخیره کردن^۲ محدودند. این دستورات از آدرس دهی غیرمستقیم ثباتی استفاده می کنند و معمولاً به سه یا چهار مرحله در خط لوله نیاز دارند. برای پیشگیری برخورد بین دستیابی به حافظه برای برداشت دستور و ذخیره کردن یک عملوند، اغلب ماشینهای RISC دو گذرگاه مجزا را بکار می برند که به دو حافظه جدا مرتبتند: یکی برای ذخیره کردن نتایج و دیگر برای ذخیره داده ها. بعضی اوقات دو حافظه قادرند با سرعت یکسان و برابر با سرعت پالس CPU عمل کنند و در این مواقع به نام حافظه های نهان^۳ خوانده می شوند (بخش ۶-۱۲).

همانطور که در بخش ۸-۸ گفته شد یکی از مزایای مهم RISC توانایی اجرای دستورات با سرعت یک دستور در هر پالس ساعت است. البته نباید تصور کرد که هر دستور در یک پالس از حافظه برداشت و اجرا شود. در واقع آنچه عملاً اتفاق می افتد این است که آغاز هر دستور با هر پالس بوده و استفاده از پردازنده خط لوله در رسیدن به هدف، بمعنی نیل به اجرای دستورات در یک پالس است. مزیت RISC در مقابل CISC این است که RISC می تواند با استفاده از قطعات خط لوله که فقط یک سیکل ساعت نیاز دارند به هدف برسد، درحالی که CISC چندین قطعه را در خط لوله بکار می برد و طولانی ترین قطعه دو یا چند پالس ساعت نیاز دارد.

مشخصه دیگر RISC پشتیبانی حاصل از کامپایلر است که زبانهای سطح بالا را به زبان ماشین ترجمه می کنند. پردازنده های RISC بعضی اوقات بر طراحی های سخت افزاری که با مشکلات ناشی از داده و تأخیرهای انشعاب مقابله می کنند، بر توانایی کامپایلر در شناخت و حداقل نمودن تأخیرات این برنامه ها متکی هستند. مثال های زیر نشان می دهد که چگونه یک کامپایلر می تواند برنامه های زبان ماشین را بهینه نماید تا مشکلات خط لوله جبران شود.

مثال: خط لوله دستورالعمل سه قطعه ای

نمونه ای از دستورات پردازنده RISC در جدول ۸-۱۲ لیست شدند. سه نوع دستورالعمل در این جدول دیده می شود. دستورات دستکاری داده روی داده های درون ثبات های پردازنده عمل می کنند.

1- Load

2- Store

3- Cache

دستورات انتقال داده همان دستورات بارکردن و ذخیره کردن می باشند که آدرس موثر بکار رفته را از افزایش محتویات دو ثبات یا ثبات و ثابت جابجایی (تغییر مکان) حاصل از دستور بدست می آورند. دستورات کنترل برنامه مقادیر ثبات و یک ثابت را برای ارزیابی آدرس انشعاب جهت انتقال به شمارنده برنامه PC یا یک ثبات بکار می برند.

اکنون عملیات سخت افزاری را برای چنین کامپیوتری ملاحظه کنید. بخش کنترل، دستور را از حافظه برنامه به یک ثبات دستورالعمل منتقل می کند. دستورالعمل، همزمان با انتخاب ثبات های لازم بهنگام اجرا دیکد می شود. واحد پردازنده متشکل از تعدادی ثبات و یک واحد محاسباتی منطقی ALU است که اعمال محاسباتی، منطقی و شیفت را انجام می دهد. یک حافظه داده برای ذخیره یا بارکردن داده در ثبات واقع در فایل ثبات ها لازم است. سیکل دستور می تواند به سه عمل جزئی که در سه قطعه پیاده شده است تقسیم شود

I - برداشت دستور

A - عمل ALU

E - اجرای دستور

قطعه I دستور را از حافظه برنامه برداشت می کند. دستور، دیکد شده و یک عمل ALU در قطعه A انجام می شود. ALU، بسته به دستور دیکد شده برای انجام سه وظیفه بکار می رود؛ یا عملی را برای دستورالعمل های دستکاری داده ها انجام می دهد، یا آدرس موثر یک دستور بارکردن و یا ذخیره را محاسبه می نماید و یا آدرس انشعاب را برای دستورالعمل های کنترل برنامه محاسبه می کند. قطعه E خروجی ALU، را به یکی از سه مقصد، بسته به نوع دستور، هدایت می نماید. یا نتیجه عمل ALU را به ثبات مقصدی واقع در فایل ثبات ها ارسال می دارد، یا آدرس موثر را جهت بارکردن یا ذخیره کردن به حافظه داده ها می فرستد و یا آدرس انشعاب را به شمارنده برنامه منتقل می کند.

بارکردن با تأخیر

اکنون عملکرد چهار دستور زیر را در نظر بگیرید:

1- LOAD: $R1 \leftarrow M[\text{address1}]$

2- LOAD: $R2 \leftarrow M[\text{address2}]$

3- ADD: $R3 \leftarrow R1 + R2$

4- STORE: $M[\text{address3}] \leftarrow R3$

اگر خط لوله سه قطعه ای بدون توقف بکار خود ادامه دهد در دستورالعمل 3 یک مشکل ناشی از داده پیش خواهد آمد زیرا عملوند R2 هنوز در قطعه A موجود نیست. این مطلب از زمانبندی خط لوله در شکل ۹-۹ (الف) قابل ملاحظه است. قطعه E در سیکل ساعت 4 در حال قرار دادن داده حافظه در R2

سیکل های ساعت :	1	2	3	4	5	6
1. Load R1	I	A	E			
2. Load R2		I	A	E		
3. Add R1 + R2			I	A	E	
4. Store R3				I	A	E

(الف) زمانبندی خط لوله با مشکل غیاب داده

سیکل های ساعت :	1	2	3	4	5	6	7
1. Load R1	I	A	E				
2. Load R2		I	A	E			
3. No-operation			I	A	E		
4. Add R1 + R2				I	A	E	
5. Store R3					I	A	E

(ب) زمانبندی خط لوله با بارکردن تأخیری

شکل ۹-۹ زمانبندی خط لوله سه قطعه‌ای

است. قطعه A در سیکل ساعت 4 داده‌ای را از R2 بکار می‌برد، ولی کمیت موجود در R2 معتبر نیست زیرا مقدار صحیح، هنوز از حافظه به آن منتقل نشده است. در اینجا کامپایلر باید مطمئن شود که دستور بعدی قادر به استفاده داده از حافظه هست. اگر موفق به این کار نشود و دستوری را نتواند پس از بارکردن قرار دهد یک دستور "هیچ‌کار" (no-op) در برنامه می‌گنجاند. این دستورالعمل از حافظه برداشت می‌شود ولی هیچ‌کاری را انجام نداده و تنها یک سیکل ساعت وقت را تلف می‌کند. ایجاد تأخیر در استفاده از داده حافظه، بارکردن با تأخیر نام دارد.

شکل ۹-۹ (ب) برنامه قبل را همراه با یک دستور هیچ‌کار که پس از بارکردن R2 آمده است نشان می‌دهد. داده در سیکل ساعت 4 در R2 بار می‌شود. دستور ADD مقدار R2 را در مرحله 5 استفاده می‌کند. بنابراین دستور هیچ‌کار اجرا را به اندازه یک سیکل ساعت بتأخیر می‌اندازد تا جبران زمانی برای داده در خط لوله انجام شود. (توجه کنید که عمل "هیچ‌کار" در سیکل 4 در قطعه A و در سیکل 5 در قطعه E انجام می‌شود). مزیت بارکردن با تأخیر این است که موضوع وابستگی داده توسط کامپایلر مورد توجه قرار می‌گیرد و نه توسط سخت‌افزار. این روش منجر به ساخت مدارات سخت‌افزاری ساده‌تر می‌گردد زیرا قطعه نیازی به کنترل ثبات جهت معتبر یافتن و یا نیافتن محتوای آن ندارد.

1- No - Operation

انشعاب با تأخیر

در شکل ۸-۹ دیدیم که یک دستور انشعاب عملکرد خط لوله را تا برداشت آدرس انشعاب با تأخیر مواجه می‌سازد. در مورد کاهش زمان تلف شده ناشی از انشعاب در بخش قبل بحث کردیم. روشی که در اغلب پردازنده‌های RISC معمول است تکیه بر کامپایلر می‌باشد که توسط آن انشعاب‌ها در زمان مناسب در خط لوله قرار می‌گیرند. این روش انشعاب با تأخیر نامیده شده است.

کامپایلر مربوط به پردازنده‌ای که انشعاب با تأخیر را استفاده می‌کند طوری طراحی شده است که دستورالعمل‌های قبل و بعد از انشعاب را تحلیل نموده و توالی برنامه را با وارد کردن دستورات مفید در آن تغییر دهد. مثلاً کامپایلر با توجه به وابستگی‌های برنامه قادر است یک یا دو دستور قبل از انشعاب را به مرحله تأخیر بعد از انشعاب منتقل کند. سپس این دستورات از حافظه برداشت شده و در خط لوله، همزمان با اجرای انشعاب در سایر قطعات، اجرا می‌گردد. اثر کلی بمانند اجرای دستورات در فرم طبیعی آنهاست با این تفاوت که تأخیر انشعاب حذف گردیده است. این دیگر بعهده کامپایلر است تا دستورالعمل‌های مفیدی را یافته و پس از دستورالعمل انشعاب قرار دهد. در صورت ناموفق بودن، کامپایلر دستورات هیچ‌کار را وارد خواهد کرد.

مثالی از انشعاب با تأخیر در شکل ۱۰-۹ دیده می‌شود. برنامه این مثال دارای پنج دستورالعمل است.

- بار کردن حافظه در R1

- افزایش R2

- جمع R3 با R4

- تفریق R5 از R6

- انشعاب به آدرس X

در شکل ۱۰-۹ (الف) کامپایلر دو دستور هیچ‌کار را بعد از انشعاب قرار می‌دهد. آدرس انشعاب، X، در سیکل ساعت 7 به PC منتقل می‌شود. برداشت دستور در X با تأخیری برابر با دو سیکل ساعت توسط دستور هیچ‌کار (no-op) صورت می‌گیرد. دستور واقع در آدرس X، فاز برداشت خود را در سیکل 8، یعنی پس از بهنگام شدن شمارنده PC، انجام می‌دهد.

آرایش برنامه شکل ۱۰-۹ (ب) با قرار گرفتن دستورالعمل‌های جمع و تفریق در بعد از انشعاب، بجای قبل از آن در برنامه اصلی، تغییر یافته است. بررسی زمانبندی خط لوله نشان می‌دهد که PC در سیکل ساعت 5 با مقدار X بهنگام می‌شود، ولی دستورات جمع و تفریق به ترتیب صحیح از حافظه برداشت و اجرا می‌شوند. به بیان دیگر، اگر دستور بارکردن در آدرس 101 و نیز X برابر 350 باشد، دستورالعمل انشعاب از مکان 103 برداشت می‌شود. دستورالعمل جمع از مکان 104 برداشت و در سیکل 6 اجرا می‌شود. دستورالعمل تفریق از آدرس 105 برداشت و در سیکل ساعت 7 اجرا می‌گردد. چون مقدار X در سیکل ساعت 5 در قطعه E به PC منتقل می‌شود، دستورالعمل برداشت شده از حافظه در سیکل ساعت 6 از آدرس 350، دستور موجود در آدرس انشعاب است.

سیکل های ساعت :

	1	2	3	4	5	6	7	8	9	10
1. Load	I	A	E							
2. Increment		I	A	E						
3. Add			I	A	E					
4. Subtract				I	A	E				
5. Branch to X					I	A	E			
6. No-operation						I	A	E		
7. No-operation							I	A	E	
8. Instruction in X								I	A	E

(الف) استفاده از دستورالعمل هیچکار

سیکل های ساعت :

	1	2	3	4	5	6	7	8
1. Load	I	A	E					
2. Increment		I	A	E				
3. Branch to X			I	A	E			
4. Add				I	A	E		
5. Subtract					I	A	E	
6. Instruction in X						I	A	E

(ب) تغییر آرایش دستورالعمل ها

شکل ۹-۱۰ مثال انشعاب با تأخیر

۹-۶ پردازش برداری

رده ای از مسائل محاسباتی وجود دارد که حل آن ورای توانایی های کامپیوترهای معمولی است. از مشخصه های این مسائل نیاز آنها به تعداد بسیار زیادی از محاسبات است که کامپیوترهای معمولی روزها حتی هفته ها وقت نیاز دارند. در بسیاری از کاربردهای علمی و مهندسی این مسائل قابل فرموله شدن برحسب جملات برداری و یا ماتریسی هستند که برای پردازش موازی مناسبند. کامپیوترهایی که قابلیت پردازش برداری را دارند در کارهای تخصصی کاربرد دارند. موارد زیر نمونه هایی از زمینه های کاربردی با اهمیت در پردازش موازی می باشند.

- پیش بینی درازمدت وضع هوا

- اکتشافات نفت

- تحلیل داده های زلزله نگاری

- تشخیص پزشکی

- آثرو دینامیک و شبیه سازی پروازهای فضایی
- هوش مصنوعی و سیستم های خبره
- نقشه برداری از ژن انسان
- پردازش تصویر

بسیاری از اعمال محاسباتی بدون استفاده از کامپیوترهای پیشرفته و پیچیده در زمان معقول تکمیل نمی شوند. برای دستیابی به سطح بالا و مطلوبی از عملکرد، لازم است تا از سریع ترین و مطمئن ترین سخت افزار استفاده گردد و رویه های ابتکاری از پردازش برداری و موازی به کار گرفته شوند.

عملیات برداری

بسیاری از مسائل علمی نیاز به عملیات محاسباتی بر روی آرایه های بزرگی از اعداد دارند. این اعداد معمولاً بصورت بردارها و ماتریس هایی از اعداد ممیز شناور فرموله می شوند. یک بردار، مجموعه مرتبی از یک آرایه یک بعدی از داده هاست. برداری مانند V بطول n بصورت یک بردار سطری $V = [V_1 \ V_2 \ V_3 \ \dots \ V_n]$ نشان داده می شود. با عمودی چیدن ارقام داده ها می توان این بردار را بصورت عمودی هم نشان داد. یک کامپیوتر ترتیبی معمولی تنها قادر است پردازش عملوندها را یک به یک پردازش نماید. در نتیجه، اعمال روی بردارها باید به محاسبات منفرد با متغیرهای شاخص دار تقسیم شود. عنصر V_i از بردار V بصورت $V(I)$ نوشته می شود که در آن شاخص (یا اندیس) I به آدرس حافظه یا ثباتی که در آن عدد ذخیره شده است اشاره می نماید. برای بررسی اختلاف بین یک پردازنده معمولی اسکالر و یک پردازنده برداری، حلقه DO فرتن را که در زیر آورده شده است در نظر بگیرید:

```
DO 20 I=1, 100
```

```
20 C(I) = B(I) + A(I)
```

این برنامه مربوط به جمع دو بردار A و B بطول 100 برای تولید بردار C می باشد. برنامه فوق توسط رشته اعمال زیر بزبان ماشین نوشته شده است:

```
Initialize I = 0
20 Read A(I)
Read B(I)
Store C(I) = A(I) + B(I)
Increment I = I + 1
If I ≤ 100 go to 20
Continue
```

رشته عملیات فوق حلقه‌ای از برنامه را ایجاد می‌کند که جفت عملوندها را از آرایه‌های A و B خوانده و جمع ممیز شناور را انجام می‌دهد. سپس متغیر کنترل حلقه بهنگام^۱ شده و مراحل 100 بار تکرار می‌گردند. کامپیوتری که قادر به پردازش برداری است اتلاف وقت مربوط به برداشت و اجرای دستورالعمل‌ها را در حلقه برنامه حذف می‌کند. در این کامپیوترها اعمال توسط دستورالعمل‌های برداری واحدی مانند آنچه در زیر آمده انجام می‌شود.

$$C(1:100) = A(1:100) + B(1:100)$$

در این دستور برداری، آدرس اول عملوندها، طول بردارها و عملیات مورد نظر، همگی منظور شده‌اند. جمع توسط یک جمع‌کننده ممیز شناور خط لوله‌ای مشابه با آنچه که در شکل ۶-۹ دیدیم انجام می‌شود. یک قالب ممکن برای دستورالعمل برداری در شکل ۱۱-۹ نشان داده شده است. این قالب اساساً یک دستور سه آدرس با سه میدان مشخص‌کننده آدرس پایه عملوندها و یک میدان اضافی است که طول اقلام داده را در بردار بدست می‌دهد. در اینجا فرض بر این است که عملوندهای برداری در حافظه قرار دارند. همچنین می‌توان پردازنده‌ای همراه با تعداد زیادی ثبات طراحی کرده و تمام عملوندها را قبل از عمل جمع در آنها ذخیره کنیم. در این حالت آدرس پایه و طول دستورالعمل گروهی از ثبات‌ها را در CPU مشخص می‌نمایند.

ضرب ماتریسی

ضرب ماتریسی یکی از اعمال محاسباتی سنگین است که در کامپیوترهای پردازش برداری انجام می‌شود. ضرب دو ماتریس $n \times n$ از n^2 ضرب داخلی یا n^3 عمل ضرب - جمع تشکیل شده است. یک ماتریس $n \times m$ از اعداد، دارای n سطر و m ستون بوده و می‌توان فرض کرد که از مجموعه n بردار سطری یا مجموعه‌ای از m بردار ستونی تشکیل گردیده است. بعنوان مثال ضرب دو ماتریس A و B، 3×3 را در نظر بگیرید.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

طول بردار	آدرس شروع مقصد	آدرس شروع مبدأ ۲	آدرس شروع مبدأ ۱	کد عمل
-----------	----------------	------------------	------------------	--------

شکل ۱۱-۹ قالب دستور برای پردازنده برداری

ماتریس حاصلضرب C یک ماتریس 3×3 بوده و توسط ضرب داخلی زیر به عناصر ماتریس های A و B مرتبط است

$$c_{ij} = \sum_{k=1}^3 a_{ik} \times b_{kj}$$

مثلاً، عددی در سطر و ستون اول ماتریس C توسط $J = 1$ و $i = 1$ محاسبه می شود

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$$

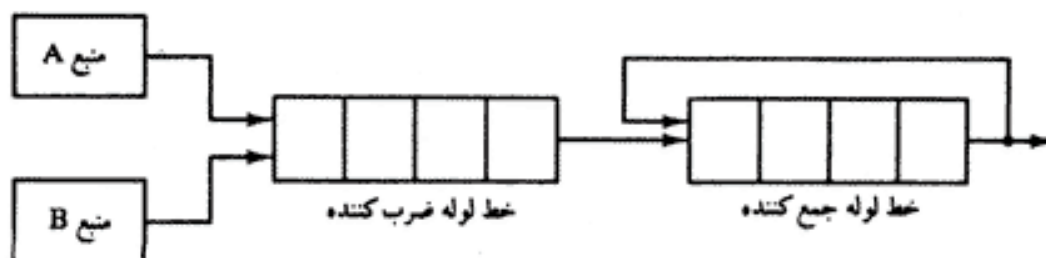
لازمه بدست آمدن این عنصر سه ضرب (پس از مقداردهی اولیه $C_{11} = 0$) و سه جمع می باشد. تعداد کل ضرب ها و جمع های لازم برای محاسبه ماتریس حاصلضرب $9 \times 3 = 27$ است.

اگر ما اعمال ضرب، جمع های مرتبط با هم $ab + c$ را یک عمل تجمعی تصور کنیم، ضرب دو ماتریس $n \times n$ نیاز به n^3 عمل ضرب - جمع دارد. محاسبات شامل n^2 ضرب داخلی است که هر ضرب داخلی هم n عمل ضرب و جمع نیاز دارد، البته فرض بر این است که قبل از محاسبه هر عنصر در ماتریس به C مقدار اولیه 0 داده شود.

بطور کلی، ضرب داخلی از مجموع K جمله ضرب مطابق زیر تشکیل یافته است

$$C = A_1 B_1 + A_2 B_2 + A_3 B_3 + A_4 B_4 + \dots + A_k B_k$$

در یک کاربرد نوعی، k ممکن است برابر 100 و حتی 1000 باشد. محاسبه ضرب داخلی در یک خط لوله با پردازش برداری در شکل ۹-۱۲ نشان داده شده است. مقادیر A و B در حافظه و یا در ثبات های پردازنده تصور می شوند. همچنین فرض بر این است که خط لوله ضرب ممیز شناور و نیز جمع ممیز شناور هر یک دارای چهار قطعه باشند. تمام ثبات های قطعات در ضرب کننده و جمع کننده با 0 مقداردهی اولیه شده اند. بنابراین، خروجی جمع کننده در هشت سیکل اولیه 0 است تا اینکه هر دو لوله پر شوند. زوج های A_i و B_i با سرعت یک زوج در هر سیکل بداخل آورده شده و ضرب می شوند. پس از چهار سیکل اول، حاصلضرب شروع به جمع شدن با خروجی جمع کننده می نماید. در طول چهار



شکل ۹-۱۲ خط لوله برای محاسبه یک ضرب داخلی

سیکل بعدی به حاصلضرب هایی که وارد خط لوله جمع کننده می شوند، 0 اضافه می شود. در پایان سیکل هشتم، چهار حاصلضرب اول یعنی A_1B_1 تا A_4B_4 در چهار قطعه جمع کننده و چهار حاصلضرب بعدی یعنی A_5B_5 تا A_8B_8 در قطعه ضرب کننده می باشند. در آغاز سیکل نهم در خروجی جمع کننده A_1B_1 و در خروجی ضرب کننده A_5B_5 وجود دارد. به این ترتیب سیکل نهم جمع $A_1B_1 + A_5B_5$ را در خط لوله جمع آغاز می نماید. سیکل دهم جمع $A_2B_2 + A_6B_6$ را آغاز می کند و عمل بهمین ترتیب ادامه دارد. با این الگو جمع به چهار بخش زیر تقسیم می شود

$$C = A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} + \dots \\ + A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} + \dots \\ + A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} + \dots \\ + A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} + \dots$$

وقتی که دیگر جمله ضربی برای جمع شدن وجود نداشته باشد سیستم چهار صفر بداخل خط لوله ضرب ارسال می کند. سپس در هر یک از چهار قطعه تنها یک ضرب جزئی وجود خواهد داشت که مربوط به چهار جمع در چهار سطر معادله فوق است. چهار جمع جزئی در نهایت با هم جمع می شوند تا حاصل جمع نهایی حاصل گردد.

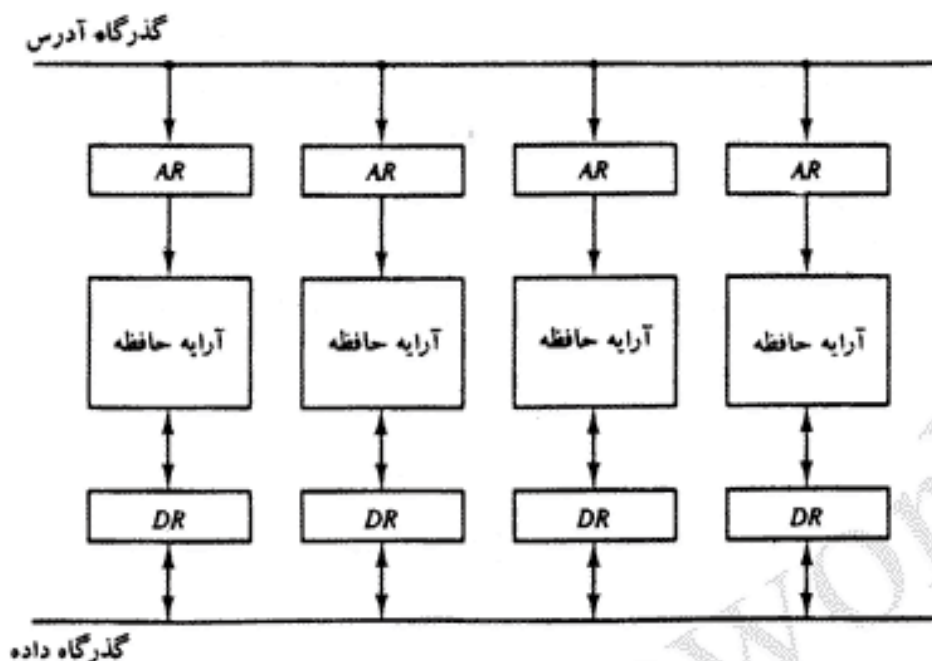
برگی کردن حافظه^۱

پردازنده های خط لوله و برداری اغلب نیاز به دستیابی همزمان به دو یا سه منبع (مبدأ) دارند. در این راستا ممکن است خط لوله حسابی هم معمولاً بطور همزمان نیاز به برداشت دو یا چند عملوند و ورود آن به خط لوله داشته باشد. در عوض استفاده از دو گذرگاه حافظه جهت دستیابی همزمان، حافظه را می توان به تعدادی ماژول^۲ که به گذرگاه های آدرس و داده مشترکی وصلند تقسیم کرد. یک ماژول حافظه را آرایه ای از حافظه به همراه ثبات های داده و آدرس مربوطه اش تشکیل می دهد. شکل ۱۳-۹ یک واحد حافظه چهار ماژولی را نشان می دهد. همانطور که دیده می شود هر آرایه دارای ثبات آدرس AR و ثبات داده DR است. ثبات های آدرس، اطلاعات را از گذرگاه آدرس مشترک دریافت و ثبات های داده هم با گذرگاه داده دوطرفه تبادل اطلاعات می نمایند. دو بیت کم ارزش تر آدرس می تواند برای تفکیک چهار ماژول بکار روند. سیستم ماژولی اجازه می دهد تا یک ماژول، دستیابی به حافظه را آغاز نماید در حالیکه سایر ماژول ها در حال خواندن یا نوشتن یک کلمه بوده و در واقع هر ماژول مستقل از وضعیت ماژول های دیگر به یک درخواست حافظه پاسخ می دهد.

مزیت حافظه ماژولی این است که امکان استفاده از تکنیکی بنام برگی کردن حافظه را فراهم می آورد. در یک حافظه برگی شده به ماژول های مختلف حافظه، آدرس های متفاوتی تخصیص یافته است. مثلاً،

1- Intertwaving

2- Module



شکل ۹-۱۳ سازمان حافظه چند ماژولی

در یک سیستم با دو ماژول برای حافظه، ممکن است آدرس های زوج در یک ماژول و آدرس های فرد در دیگری باشد، وقتی تعداد ماژول ها توانی از 2 است بیت های کم ارزش تر آدرس، ماژول حافظه را انتخاب می نمایند و بقیه بیت ها مکان خاصی را برای دستیابی در ماژول انتخاب شده مشخص می نمایند. حافظه ماژولی در سیستم های خط لوله ای و پردازش برداری مفیدند. یک پردازنده برداری که از یک حافظه n برگی استفاده می کند می تواند n عملوند را از n ماژول مختلف برداشت نماید. با توزیع دستیابی به حافظه، زمان موثر سیکل حافظه می تواند با فاکتوری در حدود تعداد ماژول ها کاهش یابد. یک CPU با خط لوله دستورالعمل می تواند از تعدد ماژول های حافظه بهره گیری نماید بطوری که هر قطعه در خط لوله می تواند مستقل از دستیابی قطعات دیگر، به حافظه دسترسی یابد.

سوپر کامپیوتر

یک کامپیوتر تجاری با دستورالعمل های برداری به همراه عملیات حسابی ممیز شناور خط لوله ای، سوپر کامپیوتر نامیده می شود. این کامپیوترها ماشین هایی قوی با قدرت اجرایی بالا بوده و اغلب برای محاسبات علمی بکار می روند. برای افزایش سرعت، قطعات بسیار نزدیک بهم نصب شده اند تا بدینوسیله فواصلی را که سیگنال ها طی می کنند حداقل نمایند. برای خارج ساختن گرما از مدارات و به منظور جلوگیری از سوختن آنها بعلت نزدیکی شان از تکنیک های خاصی استفاده می گردد.

مجموعه دستورالعمل های سوپر کامپیوترها شامل دستورات استاندارد انتقال داده ها، دستکاری داده ها و دستورات کنترل برنامه در کامپیوترهای معمولی است. علاوه بر آنها، دستوراتی وجود دارند که

بردارها و ترکیبی از اسکالرها و بردارها را پردازش می نمایند. سوپر کامپیوتر، یک سیستم کامپیوتری است که بیشتر بلحاظ سرعت بالایش در محاسبات، سیستم های حافظه سریع و حجیم و حجم بالای پردازش در آن شناخته شده است. این کامپیوتر با واحدهای چندتایی (چندکاره) تجهیز شده و هر واحد دارای پیکربندی لوله ای خاص خود می باشد. هرچند سوپر کامپیوتر قادر به اجرای برنامه های همه منظوره مربوط به کامپیوترهای معمولی می باشد، ولی در اصل برای انواع محاسبات عددی مربوط به بردارها و ماتریس های اعداد ممیز شناور بهینه سازی شده است.

سوپر کامپیوترها برای پردازش معمولی روزمره یک تشکیلات کامپیوتر مناسب نیستند. کاربرد این کامپیوترها به کاربردهای علمی همچون پیش بینی عدد وضع هوا، تحلیل موج های زلزله نگاری، و تحقیقات فضایی محدود است. ضمناً بعلت قیمت گران کاربرد و بازار محدودی نیز دارد.

یکی از معیارهای بکار رفته برای ارزیابی کامپیوترها، انجام تعداد اعمال روی اعداد ممیز شناور در هر ثانیه است که به آن فلاپس^۱ می گویند. هر مگافلاپس برابر یک میلیون فلاپس و یک گیگافلاپس^۲ نیز یک میلیارد فلاپس است. یک نمونه سوپر کامپیوتر از این رده دارای سیکل زمانی پایه ای برابر با 4 تا 20 نانوثانیه می باشد. اگر پردازنده قادر به اجرای هر عمل ممیز شناور در یک سیکل باشد، بین 50 تا 250 مگافلاپس قابلیت اجرایی دارد. این سرعت از زمانی که اولین پاسخ تولید می شود برقرار خواهد بود و شامل زمان راه اندازی اولیه خط لوله نیست.

اولین سوپر کامپیوتر بنام Cray-1 در سال ۱۹۷۱ ساخته شد. این کامپیوتر از پردازش برداری همراه با 12 واحد عملیاتی موازی جداگانه استفاده می کند. هر واحد عملیاتی به قطعاتی تقسیم شده است تا داده های ورودی را از طریق خط لوله پردازش نماید. همه واحدها می توانند بطور موازی با عملوندهایی که در ثبات های متعدد CPU ذخیره شده اند (بیش از ۱۵۰ ثبات) بطور همزمان کار کنند. یک عمل ممیز شناور می تواند روی دو مجموعه عملوند 64 بیتی در طول یک سیکل ساعت 12.5 نانوثانیه ای انجام شود. در نتیجه سرعت پردازش در خط لوله 80 مگافلاپس در حین پردازش داده خواهد بود. ظرفیت حافظه این کامپیوتر 4 میلیون کلمه 64 بیتی است. این حافظه به 16 بانک (بخش) تقسیم شده است و هر کدام دارای زمان دسترسی 50 نانوثانیه می باشند. بنابراین هرگاه همه بانک ها با هم دستیابی شوند سرعت انتقال حافظه 320 میلیون کلمه در ثانیه است. تحقیقات سازندگان Cray موجب گسترش این کامپیوتر گردید و در نتیجه Cary X-MP و Cary Y-MP ساخته شدند. سوپر کامپیوتر جدید cary-2، دوازده برابر قوی تر از cary-1 است.

یکی دیگر از سوپر کامپیوترهای اولیه Fujitsu VP-200 است. این کامپیوتر دارای یک پردازنده اسکالر و یک پردازنده برداری است که می توانند بطور همزمان کار کنند. در این کامپیوتر هم، مانند سوپر کامپیوتر cary از ثبات ها و واحدهای عملیاتی متعددی استفاده شده است تا عملیات ثبات- ثبات امکان پذیر باشد. در پردازنده برداری چهار خط لوله اجرای دستورالعمل ها وجود دارند که هرگاه همزمان

1- Flops

2- gigaflops

کار کنند به سرعتی حدود 300 مگافلاپس دست خواهند یافت. حافظه اصلی 32 میلیون کلمه داشته که از طریق خط لوله‌های بارکردن و ذخیره کردن به ثبات‌های برداری متصلند. VP200 دارای 83 دستورالعمل برداری و 195 دستور اسکالر است. مدل جدیدتر VP 2600 از سیکل ساعت 3.2 ns استفاده نموده و مدعی اوج سرعتی برابر با 5 گیگافلاپس است.

۷-۹ پردازنده آرایه^۱

یک پردازنده آرایه، پردازنده‌ای است که محاسبات را بر روی آرایه‌های بزرگی از داده‌ها انجام می‌دهد. این اصطلاح برای نام‌گذاری دو نوع پردازنده بکاررفته است. آرایه پرداز الحاقی^۲ نوعی پردازنده کمکی^۳ است و به یک کامپیوتر همه منظوره متصل می‌گردد. هدف از بکارگیری اینگونه پردازنده‌ها بالا بردن عملکرد کامپیوتر اصلی در محاسبات عددی خاص است. آرایه پرداز SIMD پردازنده‌ای است که سازمان تک دستورالعمل - چند داده‌ای را داراست. این پردازنده دستورالعمل‌های برداری را با استفاده از واحدهای علمياتی متعددی در پاسخ به یک دستور انجام می‌دهد. هرچند هر دو پردازنده بردارها را پردازش می‌کنند ولی سازمان داخلی آنها با هم متفاوت است.

آرایه پرداز الحاقی

یک آرایه پرداز الحاقی بعنوان یک دستگاه جانبی برای یک کامپیوتر معمولی طراحی شده و هدف از آن بالا بردن توان اجرایی کامپیوتر توسط پردازش برداری در کاربردهای علمی پیچیده است. افزایش توان اجرایی از طریق پردازش موازی در واحدهای چندتابعی^۴ (عملیاتی) است. این واحدها عبارتند از واحد حساب با یک یا چند جمع و ضرب‌کننده ممیز شناور خط لوله‌ای. آرایه پرداز قادر است انواع مسائل ریاضی پیچیده را بکمک برنامه‌های کاربر حل نماید.

شکل ۱۴-۹ اتصالات درونی یک آرایه پرداز الحاقی را به یک کامپیوتر میزبان^۵ نشان می‌دهد. کامپیوتر میزبان یک کامپیوتر همه منظوره تجاری است و آرایه پرداز مانند یک ماشین در انتهای خط^۶ است که توسط کامپیوتر میزبان هدایت می‌شود. آرایه پرداز از طریق یک کنترلر ورودی - خروجی به کامپیوتر متصل است و کامپیوتر همانند یک وسیله جانبی با آن برخورد می‌نماید. داده‌های مربوط به پردازنده الحاقی از طریق یک گذرگاه سریع از حافظه اصلی به یک حافظه محلی منتقل می‌گردد. کامپیوتر همه منظوره در غیاب آرایه پرداز، کاربرانی را که نیاز به پردازش معمولی داده‌ها دارند سرویس می‌دهد. سیستم به همراه آرایه پرداز نیازهای کاربردهای حسابی پیچیده را برآورده می‌سازد.

برخی از سازندگان آرایه پردازهای الحاقی مدلی را پیشنهاد می‌نمایند که قابل اتصال به انواع مختلفی

1- Array Processor

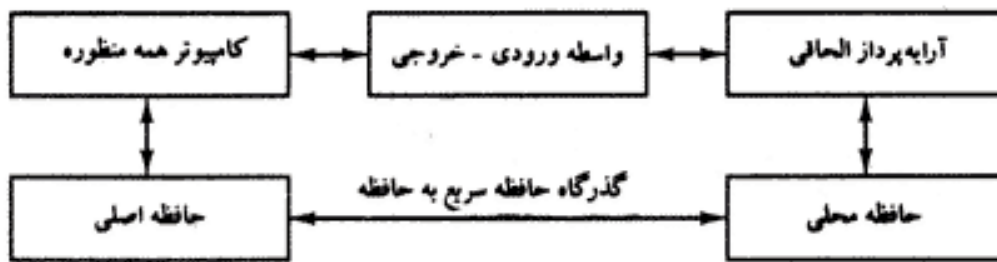
2- Attached Array Processor

3- Auxiliary Processor

4- Multiple Functional Unit

5- Host

6- Back-end machine



شکل ۹-۱۴ آرایه پرداز الحاقی با کامپیوتر میزبان

از کامپیوترهای میزبان می باشد. مثلاً اگر FSP-164/Max که یک سیستم متمیز شناور است به VAX11 متصل شود قدرت محاسبه VAX را تا 100 مگافلاپس افزایش می دهد. هدف از آرایه پرداز الحاقی ایجاد توانایی پردازش برداری کامپیوترهای معمولی با کسری از هزینه لازم برای سوپر کامپیوتر است.

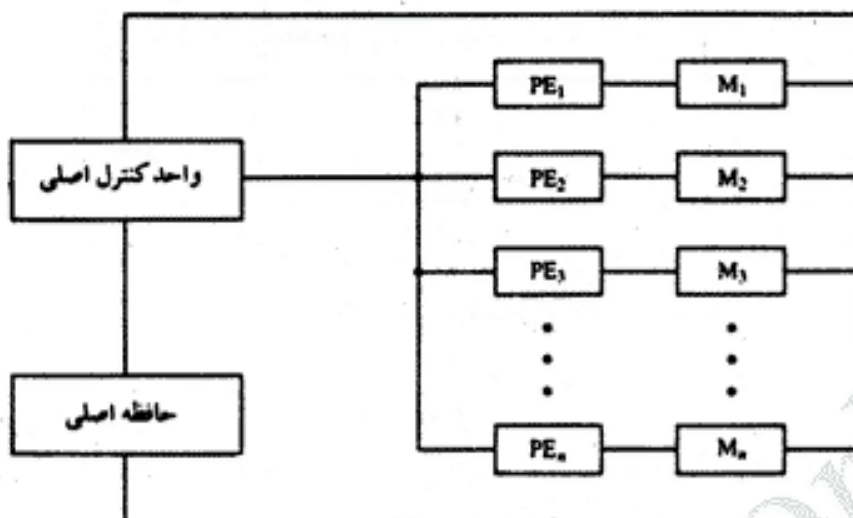
آرایه پرداز SIMD

یک آرایه پرداز SIMD کامپیوتری است با واحدهای پردازشی متعدد که بطور موازی کار می کنند. واحدهای پردازش در این کامپیوتر برای انجام کارهای یکسان تحت واحد کنترل مشترک همگام شده اند، و به این ترتیب سازمانی با رشته دستورالعمل های واحد ولی داده های متعدد (SIMD) حاصل می گردد. بلاک دیاگرام شامل مجموعه ای از عناصر پردازشی یکسان (PE ها) است که هر کدام یک حافظه محلی M را دارا هستند (شکل ۹-۱۵). در هر عنصر پردازشی یک ALU، یک واحد حساب متمیز شناور و یک تعداد ثابت های عملیاتی وجود دارند و کنترل اصلی اعمال را در عناصر پردازشی کنترل می نماید. برنامه اصلی برای ذخیره برنامه بکار می رود. وظیفه کنترل اصلی دیکد کردن دستورالعمل ها و تعیین چگونگی اجرای دستورالعمل است. دستورات اسکالر و کنترل برنامه مستقیماً در داخل واحد کنترل اصلی اجرا می شوند. دستورالعمل های برداری بطور همزمان به تمام PE ها اعلام می شوند. هر PE از عملوندهای ذخیره شده در حافظه محلی اش استفاده می کند. عملوندهای برداری قبل از اجرای موازی دستورات در حافظه ها توزیع شده اند.

به عنوان مثال، جمع برداری $C = A + B$ را در نظر بگیرید. کنترل اصلی ابتدا i امین مولفه های A و B یعنی a_i و b_i در حافظه محلی M_i در ازای $i = 1, 2, 3, \dots, n$ را ذخیره می کند. سپس دستور جمع متمیز شناور $c_i = a_i + b_i$ بین تمام PE ها منتشر می شود تا بدین ترتیب جمع بطور همزمان صورت گیرد. مولفه c_i در مکان های ثابتی در هر حافظه محلی ذخیره می گردد. در نتیجه مجموع برداری مورد نظر در یک سیکل جمع تولید می گردد.

برای کنترل وضعیت در هر PE در طول اجرای دستورالعمل های برداری از روشهای ماسک کردن^۱

1- Masking



شکل ۹-۱۵ سازمان آرایه پرداز SIMD

(پوشش دادن) استفاده می شود. هر PE دارای پرچمی است که هرگاه PE فعال باشد 1 می شود و هر وقت PE غیرفعال باشد 0 می گردد. بدین ترتیب تضمین می شود که فقط PE هایی که باید در حین اجرای دستور مشارکت داشته باشند فعال شوند. مثلاً فرض کنید که آرایه پرداز دارای 64 واحد PE باشد. اگر قرار باشد برداری با طول کمتر از 64 قلم داده پردازش شود، واحد کنترل تعداد مناسبی PE را فعال خواهد نمود. بردارهای با طول بیش از 64 باید به قسمت های 64 کلمه ای تقسیم شوند. معروف ترین پردازنده آرایه ای SIMD، کامپیوتر ILLIAC IV است که در دانشگاه ایلینویز^۱ طراحی و توسط شرکت باروز^۲ ساخته شد. از این کامپیوتر دیگر استفاده نمی شود. کامپیوترهای SIMD کامپیوترهای بسیار تخصصی هستند. این کامپیوترها عمدتاً برای مسائل عددی که قابل ارائه بشکل برداری یا ماتریسی باشند مناسبند. با این وجود در سایر انواع محاسبات یا اجرای برنامه های داده پردازشی کارایی چندانی ندارند.

مسائل

۹-۱ در برخی از محاسبات علمی لازم است تا رابطه حسابی $(C_i + D_i)$ $(A_i + B_i)$ با رشته ای از اعداد انجام شود. یک پیکربندی خط لوله را برای انجام آن مشخص کنید. محتوای تمام ثبات های خط لوله را بازا 6 تا $i = 1$ ایست نمایید.

۹-۲ یک دیاگرام فاصله- زمان برای یک خط لوله شش قطعه ای رسم کنید که زمان پردازش هشت

1- Illinois

2- Burroughs

تکلیف را نشان دهد.

- ۹-۳ تعداد سیکل هایی ساعتی که برای 200 تکلیف در یک خط لوله شش قطعه ای لازم است را معین کنید.
- ۹-۴ یک سیستم غیرخط لوله ای برای پردازش یک تکلیف 50 ns زمان نیاز دارد. همان تکلیف در یک خط لوله شش قطعه ای به یک سیکل ساعت 10 نانوثانیه ای نیازمنده است. نسبت افزایش سرعت خط لوله برای 100 تکلیف را معین کنید. حداکثر تسریع قابل دسترسی چقدر است؟
- ۹-۵ خط لوله شکل ۲-۹ دارای زمانهای انتشار زیر است: 40 ns برای خواندن عملوند از حافظه به ثبات R_1 و R_2 ، 45 ns برای انتشار سیگنال در ضرب کننده، 5 ns برای انتقال به R_3 و 15 ns برای جمع دو عدد و قرار دادن آن در R_5 .
- (الف) حداقل سیکل ساعتی که می توان بکار برد چقدر است؟
- (ب) یک سیستم غیرخط لوله ای می تواند همین عمل را با حذف R_3 و R_4 انجام دهد. ضرب و جمع عملوندها بدون استفاده از خط لوله چقدر طول می کشد؟
- (ج) تسریع عملکرد خط لوله را برای 10 تکلیف و نیز برای 100 تکلیف محاسبه کنید.
- (د) حداکثر افزایش سرعت قابل دستیابی چقدر است؟
- ۹-۶ می خواهیم یک خط لوله ضرب ممیز شناور که دو عدد صحیح دودویی هشت بیتی را ضرب می کند طراحی کنیم. هر قطعه از تعدادی گیت AND و یک جمع کننده دودویی شبیه به ضرب کننده آرایه در شکل ۱۰-۱۰ تشکیل شده است.
- (الف) در هر قطعه چند گیت AND وجود دارد و سائز جمع کننده لازم چیست؟
- (ب) چند قطعه در خط لوله است؟
- (ج) اگر تأخیر انتشار در هر قطعه 30 ns باشد، زمان متوسط لازم برای ضرب دو عدد ممیز ثابت در خط لوله چقدر است؟
- ۹-۷ زمان تأخیر چهار قطعه در خط لوله شکل ۹-۶ بقرار زیر است:
 $t_1 = 50 \text{ ns}$ ، $t_2 = 30 \text{ ns}$ ، $t_3 = 95 \text{ ns}$ و $t_4 = 45 \text{ ns}$. تأخیر ثبات های واسط $t_r = 5 \text{ ns}$.
- (الف) جمع 100 جفت عدد در خط لوله چقدر طول می کشد؟
- (ب) چگونه می توان زمان کل را به نصف زمان محاسبه شده در قسمت (الف) رساند؟
- ۹-۸ چگونه می توانید جمع کننده ممیز شناور خط لوله ای شکل ۹-۶ را برای جمع 100 عدد ممیز شناور $X_1 + X_2 + X_3 + \dots + X_{100}$ بکار برید؟
- ۹-۹ یک خط لوله دستورالعمل شش قطعه ای را برای یک کامپیوتر فرموله کنید. عملیات هر قطعه را نیز مشخص نمایید.
- ۹-۱۰ چهار روش سخت افزاری ممکن را که می توان در خط لوله دستورالعمل برای به حداقل رساندن افت عملکرد ناشی از دستورالعمل بکار برد، شرح دهید.
- ۹-۱۱ چهار دستور را در برنامه زیر در نظر بگیرید. فرض کنید که اولین دستورالعمل از مرحله ۱ در خط

لوله شکل ۹-۸ آغاز شود. عملیات انجام شده در چهار قطعه در چهار مرحله را مشخص کنید

```
Load   R1 ← M[312]
ADD    R2 ← R2 + M[313]
INC    R3 ← R3 + 1
STORE  M[314] ← R3
```

- ۹-۱۲ برنامه‌ای را مثال بزنید که موجب اغتشاش داده در خط لوله سه قطعه بخش ۵-۹ گردد.
- ۹-۱۳ مثالی ارائه نمایید که در آن در خط لوله سه قطعه‌ای بخش ۵-۹ با تأخیر عمل بازکردن را صورت دهد.
- ۹-۱۴ برنامه‌ای مثال بزنید که موجب اتلاف زمان ناشی از انشعاب در خط لوله سه قطعه‌ای بخش ۵-۹ گردد.
- ۹-۱۵ مثالی ارائه کنید که در آن از انشعاب با تأخیر در خط لوله سه قطعه‌ای بخش ۵-۹ استفاده شود.
- ۹-۱۶ ضرب دو ماتریس 40×40 را در یک پردازنده برداری در نظر بگیرید.
(الف) چند جمله ضرب در هر حاصلضرب داخلی وجود دارد و چند حاصلضرب داخلی باید محاسبه شود.
- (ب) چند عمل ضرب - جمع برای محاسبه ماتریس لازم است.
- ۹-۱۷ در شکل ۹-۱۲ وقتی که ماتریس‌های 60×60 درهم ضرب شوند چند سیکل ساعت برای پردازش حاصلضرب داخلی در خط لوله لازم است؟ چند حاصلضرب داخلی وجود دارد و چند سیکل ساعت برای محاسبه ماتریس حاصلضرب لازم است؟
- ۹-۱۸ آدرس‌های مربوط به یک آرایه 1024 کلمه، که قرار است در حافظه شکل ۹-۱۳ ذخیره شوند را مشخص نمایید.
- ۹-۱۹ محاسبه‌ای برای پیش‌بینی وضع هوا به 250 میلیارد عمل ممیز شناور نیاز دارد. مسئله در یک سوپر کامپیوتر با سرعت اجرایی 100 مگافلاپس پردازش می‌شود. زمان اجرا چقدر است؟
- ۹-۲۰ کامپیوتر را با چهار پردازنده لوله‌ای ممیز شناور در نظر بگیرید. فرض کنید که هر پردازنده از سیکل زمان 40 ns استفاده می‌کند. برای اجرای 400 عمل ممیز شناور چقدر وقت لازم است؟ اگر همین عمل با استفاده از یک پردازنده لوله‌ای تکی با سیکل ساعت 10 ns انجام شود چه تفاوتی خواهد داشت؟



معماری کامپیوتر

- ۱۰-۵ عمل های حسابی ممیز شناور
- ۱۰-۶ واحد حساب دهمی
- ۱۰-۷ اعمال حسابی دهمی

- ۱۰-۱ مقدمه
- ۱۰-۲ جمع و تفریق
- ۱۰-۳ الگوریتم ضرب
- ۱۰-۴ الگوریتم های تقسیم

۱۰-۱ مقدمه

دستورالعمل های حسابی در کامپیوترهای دیجیتال با دستکاری داده ها، نتایج لازم برای حل مسائل محاسباتی را تولید می کنند. این دستورالعمل ها محاسبات ریاضی را انجام می دهند و مسئول مجموعه فعالیت های عمده ای در پردازش داده ها در کامپیوترند. چهار عمل اصلی حساب عبارتند از جمع، تفریق، ضرب و تقسیم. با استفاده از این چهار عمل، می توان توابع حسابی دیگر را تنظیم کرده و مسائل علمی را با استفاده از روشهای محاسبات عددی حل کرد.

پردازنده حسابی بخشی از یک واحد پردازنده است که اعمال حسابی را اجرا می کند. نوع داده ای که در طول اجرای دستور در داخل ثبات های پردازنده قرار می گیرند در تعریف دستورالعمل مشخص می گردد. یک دستورالعمل حسابی ممکن است داده های دودویی یا دهمی داشته باشد، و در هر حال داده ها ممکن است بفرم ممیز ثابت یا ممیز شناور باشند. اعداد ممیز ثابت ممکن است صحیح یا کسری باشند. اعداد منفی می تواند بصورت اندازه علامت دار یا متمم علامت دار نشان داده شوند. اگر فقط دستورالعمل جمع برای عددهای دودویی ممیز ثابت در نظر گرفته شود، پردازش حسابی بسیار ساده خواهد بود. اما اگر در پردازنده حسابی هر چهار عمل اصلی برای داده های دودویی و دهمی به شکل ممیز ثابت و ممیز شناور در نظر گرفته شوند، موضوع پیچیده تر می شود.

در سنین کودکی آموختیم که چگونه چهار عمل اصلی را در نمایش اندازه علامت دار انجام دهیم. این دانش هنگام پیاده سازی عمل ها با سخت افزار ارزشمند است. با این وجود طراح باید واقعاً با سلسله مراحل لازم برای انجام عمل و حصول نتیجه صحیح آشنا باشد. راه حل هر مسئله که بوسیله تعداد محدودی از رویه های متوالی بطور مناسب تعریف شده باشد الگوریتم¹ خوانده می شود. برای جمع دو عدد با ممیز ثابت و اعداد منفی در نمایش متمم 2 علامت دار، الگوریتمی در بخش 3-3 بیان شد. این الگوریتم ساده است زیرا کلاً به یک جمع کننده دودویی موازی نیاز دارد. وقتی اعداد منفی بشکل اندازه علامت دار باشند، الگوریتم کمی پیچیده تر است و پیاده سازی آن به مدارهای جمع و تفریق، و مقایسه علامت ها و اندازه های اعداد نیاز دارد. معمولاً یک الگوریتم شامل تعدادی مراحل مرتبط باهم است که هر مرحله به نتایج مراحل قبلی وابسته است. روش مناسبی برای ارائه الگوریتم ها فلوچارت است. مراحل محاسباتی در فلوچارت در داخل کادرهای مستطیلی مشخص می شوند. مراحل تصمیم گیری در داخل کادرهای لوزی شکل نشان داده می شوند که از آن ها دو یا چند مسیر مختلف جدا می شود. در این فصل الگوریتم های حسابی متعددی را بدست می آوریم و رویه ای را برای پیاده سازی آنها با سخت افزار دیجیتال نشان خواهیم داد. ما برای داده های زیر، جمع، تفریق، ضرب و تقسیم را بررسی خواهیم کرد:

۱- داده دودویی ممیز ثابت با نمایش مقدار علامت دار.

۲- داده دودویی ممیز ثابت با نمایش متمم 2 علامت دار.

۳- داده ممیز شناور دودویی

۴- داده دهمی کد شده با دودویی

۲-۱۰ جمع و تفریق

همانطور که در بخش 3-3 گفته شد، سه راه برای نمایش اعداد دودویی ممیز ثابت منفی وجود دارد: اندازه علامت دار، متمم 1 علامت دار، یا متمم 2 علامت دار. بسیاری از کامپیوترها وقتی اعمال حسابی با اعداد صحیح را انجام می دهند از نمایش متمم 2 استفاده می کنند. برای اعمال ممیز شناور، اکثر کامپیوترها نمایش اندازه علامت دار را برای ماننسیس بکار می برند. در این بخش ما الگوریتم های جمع و تفریق را برای داده های نمایش داده شده با مقدار علامت دار و مجدداً برای نمایش متمم 2 علامت دار بدست می آوریم. توجه به این نکته مهم است که بدانیم نمایش انتخابی برای اعداد منفی به نمایش عددها در ثبات ها، قبل و بعد از اجرای عمل حسابی مربوط است. این بدان معنی نیست که از حساب متمم نمی توان در مرحله میانی استفاده کرد. مثلاً در تفریق با اعداد در نمایش اندازه علامت دار، استفاده از حساب متمم ساده تر است. مادامی که مفروق، مفروق منه و باقیمانده بصورت اندازه علامت دار باشند استفاده از متمم در یک مرحله تغییری در این واقعیت که نمایش بصورت مقدار علامت دار است نمی دهد.

1- Algorithm

جمع و تفریق با داده‌های اندازه علامت دار

نمایش اعداد بصورت اندازه علامت دار آشناست زیرا در محاسبات ریاضی روزمره بکار می‌رود. رویه جمع یا تفریق دو عدد دودویی علامت دار با قلم و کاغذ ساده و سراسر است. مروری بر این رویه در بدست آوردن الگوریتم سخت افزاری می‌تواند مفید باشد.

مانندازه دو عدد را با A و B نشان می‌دهیم. وقتی که اعداد علامت دار، جمع یا تفریق شوند، بسته به علامت و عمل انجام شده، هشت حالت برای بررسی وجود دارد. این حالات در اولین ستون جدول ۱۰-۱ لیست شده‌اند. سایر ستون‌ها در جدول، اعمالی را که باید بر روی اندازه اعداد انجام شود نشان می‌دهد. آخرین ستون برای جلوگیری از صفر منفی لازم است. به بیان دیگر وقتی دو عدد مساوی از یکدیگر تفریق شوند، نتیجه باید $+0$ باشد نه -0 .

الگوریتم جمع و تفریق از جدول بدست می‌آیند و می‌توان آنها را به صورت زیر بیان کرد (کلمات داخل پرانتز باید برای الگوریتم تفریق بکار روند):

الگوریتم جمع (تفریق): وقتی که علامت A و B یکسان (مختلف) باشند، دو اندازه را باهم جمع کرده و علامت A را به حاصل ملحق کنید. اگر علامت A و B مختلف (یکسان) باشند، اندازه‌ها را باهم مقایسه و عدد کوچکتر را از بزرگتر تفریق کنید. اگر $A > B$ است، علامت A را برای جواب برگزینید ولی اگر $A < B$ است علامت A را مکمل نمائید. اگر دو اندازه باهم مساویند، B را از A تفریق کرده و علامت حاصل را مثبت بگذارید.

این دو الگوریتم جز در مقایسه علامت‌ها مشابهند. رویه‌ای که باید برای علامت‌های یکسان در رویه جمع دنبال شوند مشابه رویه برای علامت‌های مختلف در تفریق است و بالعکس.

پیاده سازی سخت افزاری

برای پیاده سازی دو عمل حسابی با سخت افزار ابتدا لازم است هر دو عدد در ثبات‌ها ذخیره شوند.

جدول ۱۰-۱ جمع و تفریق اعداد اندازه علامت دار

عمل	جمع اندازه‌ها	تفریق اندازه‌ها		
		وقتی $A > B$	وقتی $A < B$	وقتی $A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

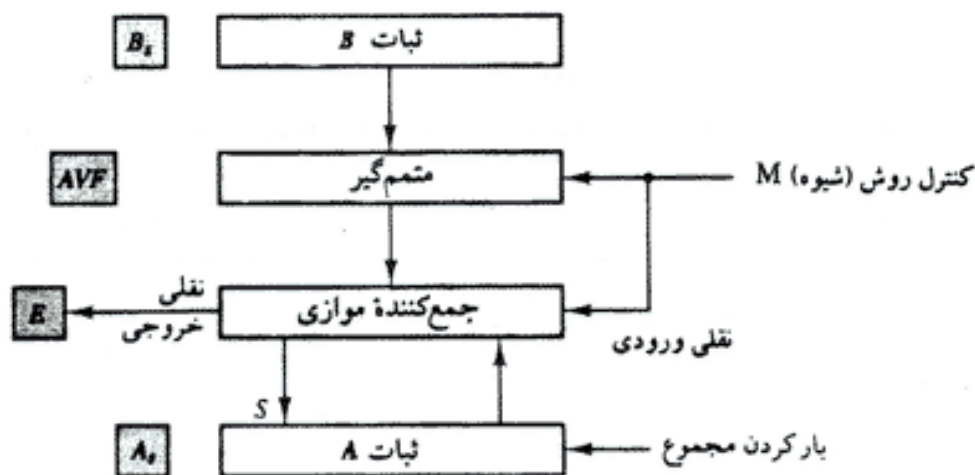
اجازه بدهید تا ثبات های A و B مقدار اعداد را ذخیره کنند و A_8 و B_8 فلیپ فلاپ های نگهدارنده علامت آنها باشند. نتیجه عمل ممکن است به ثبات سومی انتقال یابد: با این وجود، اگر نتیجه به A_8 منتقل شود صرفه جویی انجام شده است. بنابراین A و A_8 بالاتفاق انباره را می سازند.

اکنون پیاده سازی سخت افزار الگوریتم های فوق را ملاحظه کنید. اول، یک جمع کننده موازی لازم است تا ریز عمل $A+B$ انجام شود. دوم، یک مدار مقایسه گر لازم است تا $A > B$ ، $A = B$ یا $A < B$ را مشخص نماید. سوم، دو مدار تفریق گر موازی برای انجام $A-B$ یا $B-A$ لازم است. رابطه علامت ها را می توان با استفاده از OR انحصاری با ورودی های A_8 و B_8 معین کرد.

رویه به یک مقایسه گر، یک جمع کننده و دو تفریق گر نیاز دارد. با این وجود رویه متفاوتی می توان یافت که امکانات کمتری نیاز داشته باشد. اول، می دانیم که تفریق را می توان با جمع و متمم انجام داد. دوم، نتیجه مقایسه می تواند از رقم نقلی انتهایی، پس از تفریق مشخص گردد. بررسی های دقیقتر روش های دیگر نشان می دهد که استفاده از متمم 2 برای تفریق و مقایسه رویه بهتری است زیرا فقط یک جمع کننده و یک متمم کننده نیاز دارد.

شکل ۱-۱۰ بلاک دیاگرام سخت افزار مربوط به پیاده سازی اعمال جمع و تفریق می باشد. این دیاگرام از ثبات های A و B و فلیپ فلاپ های A_8 و B_8 برای علامت تشکیل شده است. تفریق بوسیله جمع A با متمم 2 عدد B انجام می شود. رقم نقلی خروجی به فلیپ فلاپ E منتقل می گردد و در آن جا برای تعیین اندازه های نسبی اعداد چک می شود. فلیپ فلاپ جمع - سرریز، AVF ، بیت سرریز را وقتی که A و B جمع شوند نگه می دارد. ثبات A سایر ریزاعمال را که ممکن است هنگام مشخص کردن دنباله مراحل در الگوریتم نیاز داشته باشیم فراهم می سازد.

جمع A بعلاوه B بوسیله جمع کننده موازی انجام می شود. خروجی S (جمع) جمع کننده به ورودی ثبات A اعمال می شود. متمم ساز، خروجی B یا متمم B را، بسته به وضعیت



شکل ۱-۱۰ سخت افزار جمع و تفریق اندازه علامت دار

کنترل روش (شیوه) M^1 در اختیار می‌گذارد. متمم ساز از گیت‌های OR انحصاری و جمع‌کننده موازی متشکل از مدارات تمام جمع‌کننده مطابق شکل ۷-۴ در فصل ۴ ساخته شده است. سیگنال M به ورودی نقلی جمع‌کننده هم متصل شده است. وقتی $M=0$ باشد، خروجی B به جمع‌کننده منتقل می‌شود، رقم نقلی ورودی 0، و خروجی جمع‌کننده برابر مجموع $A+B$ می‌گردد. هنگامی که $M=1$ باشد، متمم 1 عدد B به جمع‌کننده اعمال، رقم نقلی ورودی 1 و خروجی $S=A+\bar{B}-1$ خواهد شد. این عمل برابر با جمع A با متمم 2 عدد B است، که معادل تفریق $A-B$ است.

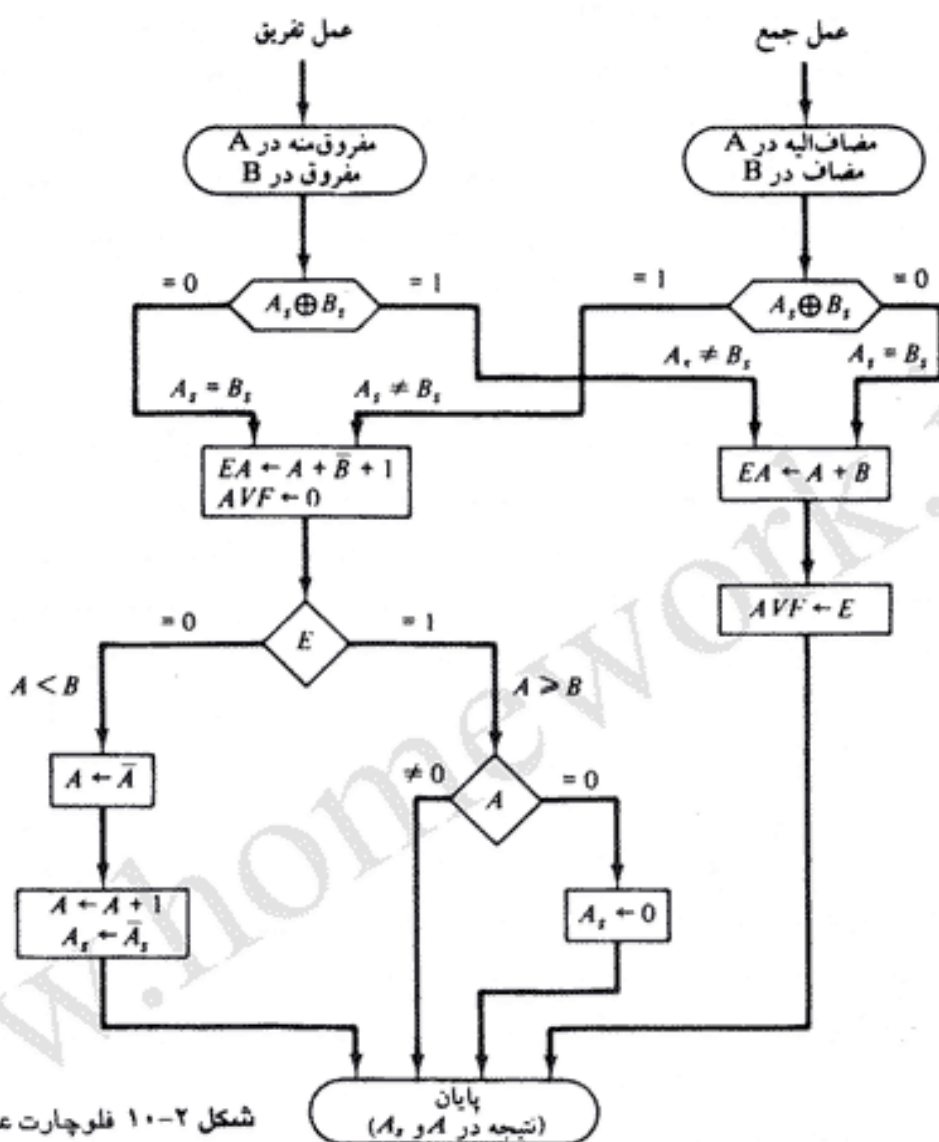
الگوریتم سخت افزاری

فلوچارت الگوریتم سخت افزاری در شکل ۲-۱۰ نشان داده شده است. دو علامت A_8 و B_8 بوسیله گیت OR انحصاری مقایسه می‌شوند. اگر خروجی گیت 0 باشد، علامت‌ها یکی هستند؛ اگر 1 باشد علامت‌ها مختلفند. برای یک عمل جمع، علائم یکسان جمع اندازه‌ها را حکم می‌کند. برای عمل تفریق علائم متفاوت جمع اندازه‌ها را حکم می‌نماید. اندازه‌ها بوسیله ریزعمل $A+B \leftarrow EA$ جمع می‌شوند، که EA ثباتی است که از ترکیب E و A بدست می‌آید. اگر رقم نقلی در E ، پس از جمع برابر 1 باشد سرریز رخ داده است. مقدار E به فلیپ فلاپ جمع سرریز AVF منتقل می‌گردد.

اگر علامت‌ها در جمع مختلف باشند و یا در تفریق یکسان باشند عمل تفریق انجام می‌شود. اندازه‌ها با جمع A و متمم 2 عدد B از یکدیگر تفریق می‌شوند. اگر اعداد از هم تفریق شوند هیچ سرریزی رخ نمی‌دهد بنابراین AVF برابر 0 خواهد شد. اگر $A \geq B$ باشد E برابر 1 است و عدد درون A نتیجه صحیحی است. اگر A برابر 0 باشد، علامت A_8 باید مثبت گردد تا از داشتن صفر منفی جلوگیری شود. اگر E برابر 0 باشد $A < B$ است. برای این حالت لازم است متمم 2 عدد A بدست آید. این عمل باریزعمل $A \leftarrow \bar{A} + 1$ انجام می‌گیرد. با این وجود ما فرض خواهیم کرد که ثبات A مدارهای لازم برای متمم سازی و افزایش را داشته باشد، بنابراین متمم 2 از این دو عمل حاصل خواهد شد. در مسیرهای دیگر فلوچارت، علامت نتیجه همان علامت A است و لذا هیچ تغییری در A_8 لازم نیست. معهذاً، وقتی $A < B$ باشد، علامت نتیجه متمم علامت اولیه A می‌باشد. در نتیجه لازم است A_8 را برای بدست آوردن علامت صحیح متمم کنیم. نتیجه نهائی در A و علامت آن در A_8 خواهد بود. مقدار در AVF بمعنی وقوع یک سرریز است، مقدار نهایی در E اهمیتی ندارد.

جمع و تفریق با داده‌های متمم 2 علامت دار

نمایش متمم 2 اعداد همراه با الگوریتم‌های حسابی برای جمع و تفریق در بخش ۳-۳ معرفی شدند. در اینجا آنها را برای سهولت مراجعه بطور خلاصه تکرار کنیم. سمت چپ‌ترین بیت عدد، علامت را نشان می‌دهد: 0 برای مثبت و 1 برای منفی. اگر بیت علامت 1 باشد، تمام عدد به فرم متمم 2 نمایش



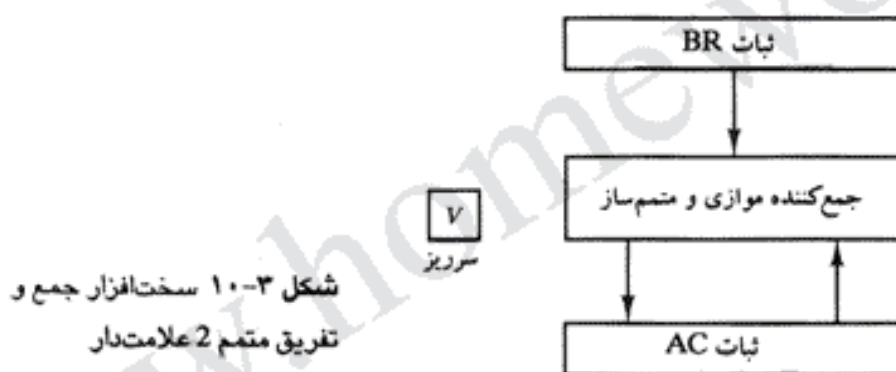
شکل ۲-۱۰ فلوجارت عمل های جمع و تفریق

داده می شود. بنابراین $+33$ بصورت 00100001 و -33 بصورت 11011111 خواهند بود. توجه کنید که 11011111 متمم 2 عدد 00100001 می باشد و بالعکس.

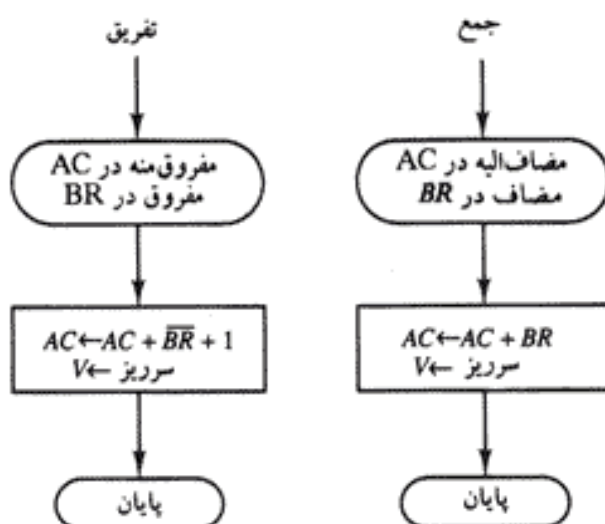
در جمع دو عدد در فرم متمم 2 علامت دار، آن دو را بصورت عادی با هم جمع می کنیم و با بیت های علامت آنها نیز مانند سایر بیت های عدد رفتار می نماییم. هر رقم نقلی حاصل از بیت علامت چشم پوشی می شود. برای تفریق ابتدا متمم 2 مفروق را بدست آورده و سپس آن را با مفروق منته جمع می کنیم. وقتی دو عدد n رقمی جمع شوند حاصل جمع $n+1$ رقم خواهد داشت و گوئیم که سرریز رخ داده است. اثر سرریز روی حاصل جمع اعداد متمم 2 علامت دار در بخش 3-3 توضیح داده شد. وقوع سرریز را می توان با بررسی دو نقلی آخر بوجود آمده در جمع تحقیق کرد. هرگاه این دو نقلی به یک گیت OR انحصاری اعمال شوند، در صورتی که خروجی این گیت 1 باشد سرریز رخ داده است.

آرایش ثبات‌ها برای پیاده‌سازی سخت افزار در شکل ۳-۱۰ نشان داده شده است. این شکل شبیه شکل ۱-۱۰ است جز اینکه بیت‌های علامت از بقیه ثبات‌ها جدا نشده‌اند. ما ثبات A را AC (انباره) و ثبات B را BR می‌نامیم. سمت چپ‌ترین بیت در AC و BR بیت علامت اعداد را نشان می‌دهند. دو بیت علامت همراه با سایر بیت‌ها در متمم ساز و جمع کننده موازی، جمع یا تفریق می‌شوند. فلاپ سرریز V پس از رخداد سرریز برابر 1 می‌شود. در این حالت رقم نقلی خروجی در نظر گرفته نشده است.

الگوریتم جمع و تفریق دو عدد دودویی با نمایش متمم 2 علامت‌دار در فلوچارت شکل ۴-۱۰ نشان داده شده است. حاصل جمع از جمع محتوای AC و BR (همراه با بیت‌های علامت) بدست می‌آید. بیت سرریز V هنگامی 1 است که OR انحصاری دو رقم نقلی آخر 1 باشند، در غیراینصورت 0 است. عمل تفریق از جمع محتوای AC با متمم 2 محتوای BR حاصل می‌گردد. گرفتن متمم 2 عدد BR بمعنی تبدیل عدد مثبت به منفی است، و بالعکس. در حین این عمل سرریز باید واریسی گردد زیرا



شکل ۳-۱۰ سخت‌افزار جمع و تفریق متمم 2 علامت‌دار



شکل ۴-۱۰ الگوریتم جمع و تفریق اعداد متمم 2 علامت‌دار

دو عددی که با هم جمع شده‌اند ممکن است هم علامت باشند. برنامه نویس باید بداند که اگر سرریز رخ دهد، نتیجه غلطی در AC بدست خواهد آمد.

با مقایسه این الگوریتم با معادل آن برای اعدادی که به شکل مقدار (اندازه) علامت دار نشان داده شده‌اند متوجه می‌شویم که اگر اعداد منفی را بفرم متمم 2 علامت دار نگهداریم عمل جمع و تفریق اعداد ساده‌تر خواهد بود. با این دلیل در اکثر کامپیوترها این روش بجای روش آشنا تر مقدار علامت دار بکار می‌رود.

۳-۱۰ الگوریتم ضرب

ضرب دو عدد ممیز ثابت دودویی با نمایش مقدار علامت دار با قلم و کاغذ، توسط فرآیند شیفت‌های متوالی و جمع انجام می‌شود. این روش را با مثالی می‌توان بهتر تشریح نمود.

$$\begin{array}{r}
 \text{مضروب} \quad 23 \quad 10111 \\
 \text{مضروب فیه} \quad 19 \quad \times 10011 \\
 \hline
 10111 \\
 00000 \\
 00000 \\
 10111 \\
 \hline
 \text{حاصلضرب} \quad 437 \quad 110110101
 \end{array}$$

فرآیند بدین صورت است که بیت‌های مضروب فیه متوالیاً، با شروع از کم ارزشترین بیت نظاره شوند. اگر بیت مضروب فیه 1 باشد، مضروب در پائین کپی می‌شود، در غیر این صورت صفرها در پائین کپی می‌گردند. اعدادی که در سطرها متوالی کپی می‌شوند نسبت به سطر قبلی یک بیت به چپ شیفت داده می‌شوند. نهایتاً اعداد باهم جمع شده و حاصل جمع، نتیجه ضرب خواهد بود.

علامت حاصلضرب با توجه به علامت‌های مضروب و مضروب فیه معین می‌شود. اگر آنها یکی باشند علامت حاصلضرب مثبت و در غیر این صورت منفی خواهد بود.

پیاده سازی سخت افزاری برای داده‌های علامت دار

هنگام پیاده سازی ضرب در یک کامپیوتر دیجیتال، بهتر است فرآیند کمی تغییر یابد. اولاً، در عوض تهیه ثبات‌هایی برای ذخیره و جمع همزمان چند عدد دودویی به تعداد بیت‌های مضروب فیه، بهتر است جمع کننده‌ای برای جمع فقط دو عدد دودویی در نظر گرفته شود و مرتباً حاصلضرب‌های جزئی^۱ را در یک ثبات نگهداری نمائیم. ثانیاً بجای شیفت مضروب به چپ، حاصلضرب جزئی به راست شیفت داده می‌شود که در نتیجه موقعیت نسبی حاصلضرب جزئی و مضروب همان موقعیت مطلوب خواهد بود. ثالثاً، وقتی که بیتی از مضروب فیه 0 باشد، لزومی ندارد که صفرها را با حاصلضرب جزئی

1- Partial Product

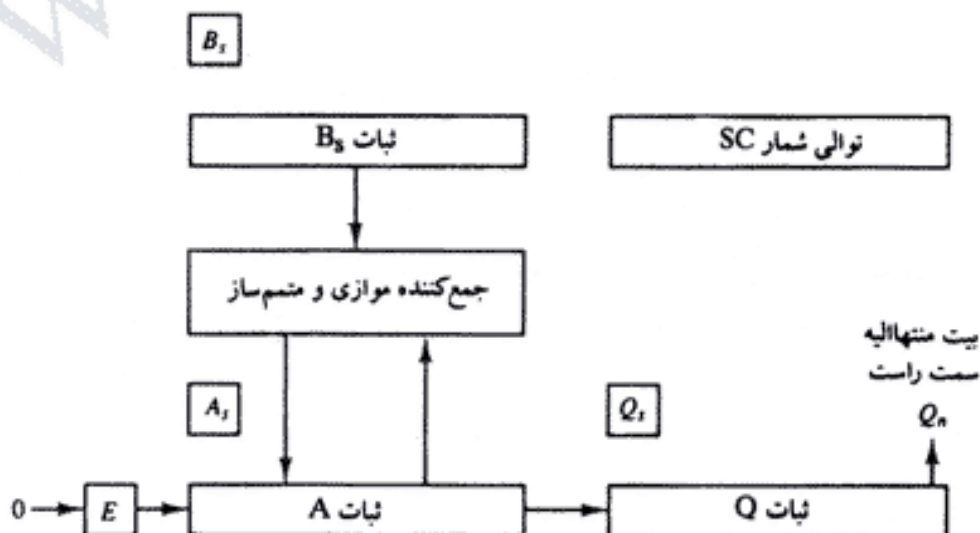
جمع کنیم زیرا مقدار آن را عوض نمی کند.

سخت افزار ضرب از امکانات شکل ۱-۱۰ بعلاوه دو ثبات اضافی دیگر تشکیل شده است. این ثبات به همراه ثبات های A و B در شکل ۵-۱۰ نشان داده شده اند. مضروب فیه در ثبات Q و علامتش در Q_8 ذخیره می شود. توالی شمار SC ابتدا با عددی برابر با تعداد بیت های مضروب فیه مقداردهی می شود. شمارنده پس از هر بار تشکیل حاصلضرب جزئی یک واحد کم می شود. وقتی که محتوای شمارنده به صفر برسد، حاصلضرب تکمیل و فرآیند متوقف می گردد.

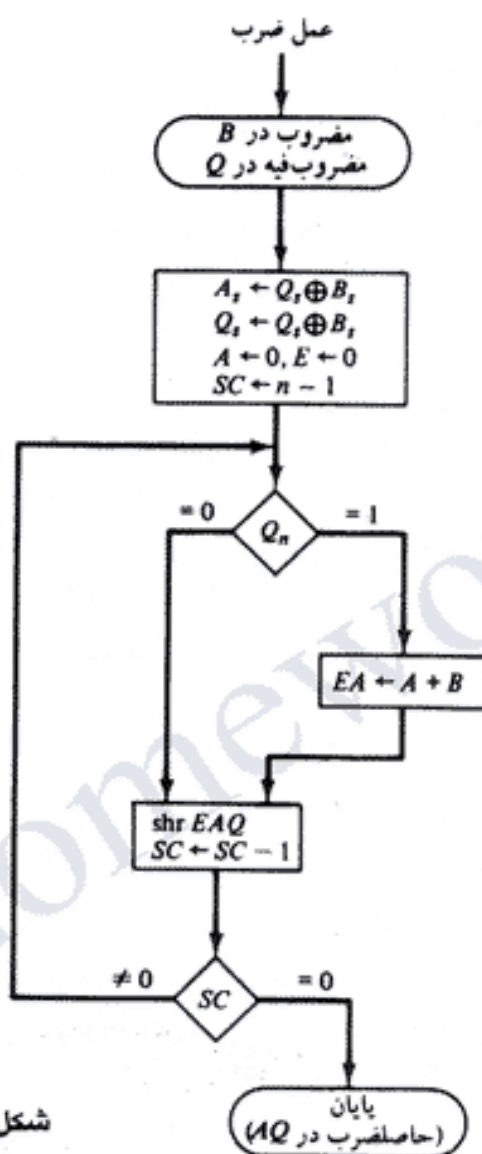
در آغاز مضروب در B و مضروب فیه در Q قرار دارد. حاصل جمع A و B تشکیل حاصلضرب جزئی را می دهند که به ثبات EA منتقل می گردد. حاصلضرب جزئی و مضروب فیه توأمأ به راست شیفت داده می شوند. این شیفت توسط عبارت shift EAQ نشان داده شده است که شیفت به راست را در شکل ۵-۱۰ مشخص می نماید. بیت کم ارزشتر A به با ارزشترین بیت Q منتقل می شود، بیت E به با ارزشترین بیت A نقل مکان می یابد، و یک 0 ولرد E می شود، یک بیت از حاصلضرب جزئی به داخل Q شیفت کرده و بیت های مضروب فیه را یک واحد به سمت راست می راند. باین ترتیب، سمت راست ترین فلیپ فلاپ در ثبات Q، که با Q_n مشخص شده است، حاوی بیتی از مضروب فیه خواهد بود که در مرحله بعدی باید بررسی شود.

الگوریتم سخت افزاری

شکل ۶-۱۰ فلوچارت الگوریتم سخت افزاری ضرب را نشان می دهد. ابتدا مضروب در B و مضروب فیه در Q قرار دارد. علامت های مربوطه به ترتیب در B_8 و Q_8 می باشند. علامت ها با هم مقایسه شده و علامت هر دو ثبات A و Q برابر علامت حاصلضرب می شود، زیرا حاصلضرب نهایی با طول دو برابر در



شکل ۵-۱۰ سخت افزار عمل ضرب



شکل ۶-۱۰ فلوجارت عمل ضرب

ثباتهای A و Q قرار خواهد گرفت. ثباتهای A و E پاک شده و توالی شمار SC برابر با تعداد بیت‌های مضروب فیه می‌شود. در اینجا فرض می‌کنیم که عملوندها از حافظه‌های n بیتی به ثبات‌ها مستقل شده‌اند. چون هر عملوند باید همراه با علامتش ذخیره شود لذا یک بیت از کلمه توسط علامت اشغال شده و مقدار علموند n-1 بیتی خواهد بود.

پس از دادن مقدار اولیه، بیت کم ارزشتر مضروب فیه که در Q_n قرار دارد تست می‌شود. اگر این بیت برابر 1 باشد، مضروب فیه موجود در B با حاصلضرب جزئی موجود در A جمع می‌شود. اگر 0 باشد، کاری انجام نمی‌شود. سپس ثبات EAQ یکبار بسمت راست شیفت داده می‌شود تا حاصلضرب جزئی را تشکیل دهد. توالی شمار 1 واحد کم شده و مقدار جدید آن چک می‌شود. اگر برابر صفر نباشد، فرآیند

تکرار شده و حاصلضرب جزئی جدید تشکیل می‌گردد. وقتی که $SC=0$ شود فرآیند متوقف می‌گردد. توجه کنید که حاصلضرب جزئی حاصل در A هر بار یک بیت به Q منتقل می‌گردد تا نهایتاً جای مضروب فیه را می‌گیرد. حاصلضرب نهایی در هر دو ثبات A و Q واقع است بدین ترتیب که A بیت‌های با ارزشتر و Q بیت‌های کم ارزشتر را نگه می‌دارند. مثال عددی قبلی در جدول ۲-۱۰ برای روشن شدن فرآیند ضرب سخت افزاری تکرار شده است. روند، مراحل مشخص شده در فلوجارت را دنبال می‌کند.

الگوریتم ضرب بوت

الگوریتم بوت رویه‌ای را برای ضرب اعداد دودویی در نمایش متمم ۲ علامتدار ارائه می‌نماید. مبنای کار الگوریتم بر این اساس استوار است که رشته‌های ۰ در مضروب فیه نیازی به جمع ندارند بلکه فقط جابجایی (شیفت) لازم دارند و رشته‌های ۱ در مضروب فیه از بیت مرتبه 2^k تا بیت 2^m را می‌توان معادل $2^{k+1}-2^m$ تلقی کرد. مثلاً عدد دودویی 001110 (+14) دارای رشته‌های ۱ از 2^1 تا 2^3 است، $(m=1, k=3)$. این عدد را می‌توان بصورت $2^4-2^1=16-2=14$ نوشت. بنابراین ضرب $M \times 14$ را، که در آن M مضروب و ۱۴ مضروب فیه است، می‌توان به صورت $M \times 2^4 - M \times 2^1$ انجام داد. لذا حاصلضرب با چهار بار شیفت مضروب به چپ و تفریق M که یکبار به چپ شیفت داده شده است بدست می‌آید.

همانند همه روشهای ضرب، الگوریتم بوت نیز به بررسی بیت‌های مضروب فیه و شیفت حاصلضرب جزئی نیاز دارد. قبل از شیفت، ممکن است مضروب طبق قواعد زیر با حاصلضرب جزئی

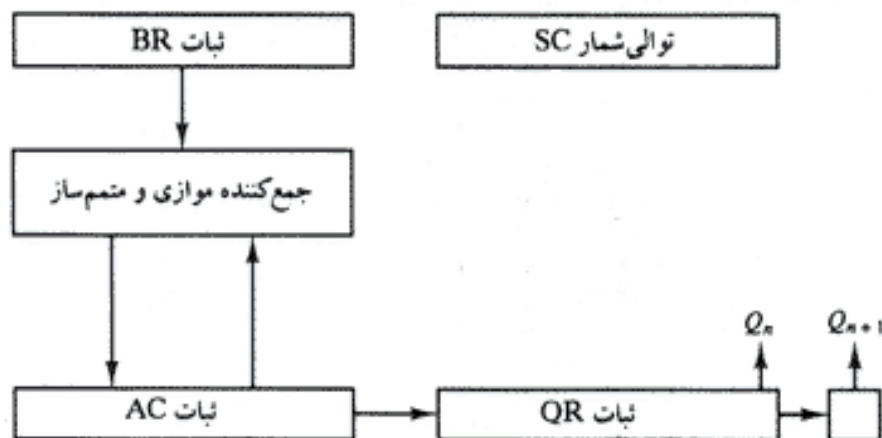
جدول ۲-۱۰ مثال عددی برای ضرب کننده دودویی

مضروب فیه $B = 10111$	E	A	Q	SC
مضروب فیه در Q	0	00000	10011	101
$Q_n=1$ جمع B		10111		
اولین حاصلضرب جزئی	0	10111		
شیفت E AQ به راست	0	01011	11001	100
$Q_n=1$ جمع B		10111		
دومین حاصلضرب جزئی	1	00010		
شیفت E AQ به راست	0	10001	01100	011
$Q_n=0$ شیفت E AQ به راست	0	01000	10110	010
$Q_n=0$ شیفت E AQ به راست	0	00100	01011	001
$Q_n=1$ جمع B		10111		
پنجمین حاصلضرب جزئی	0	11011		
شیفت E AQ به راست	0	01101	10101	000
حاصلضرب نهایی در AQ برابر 10110110101 است				

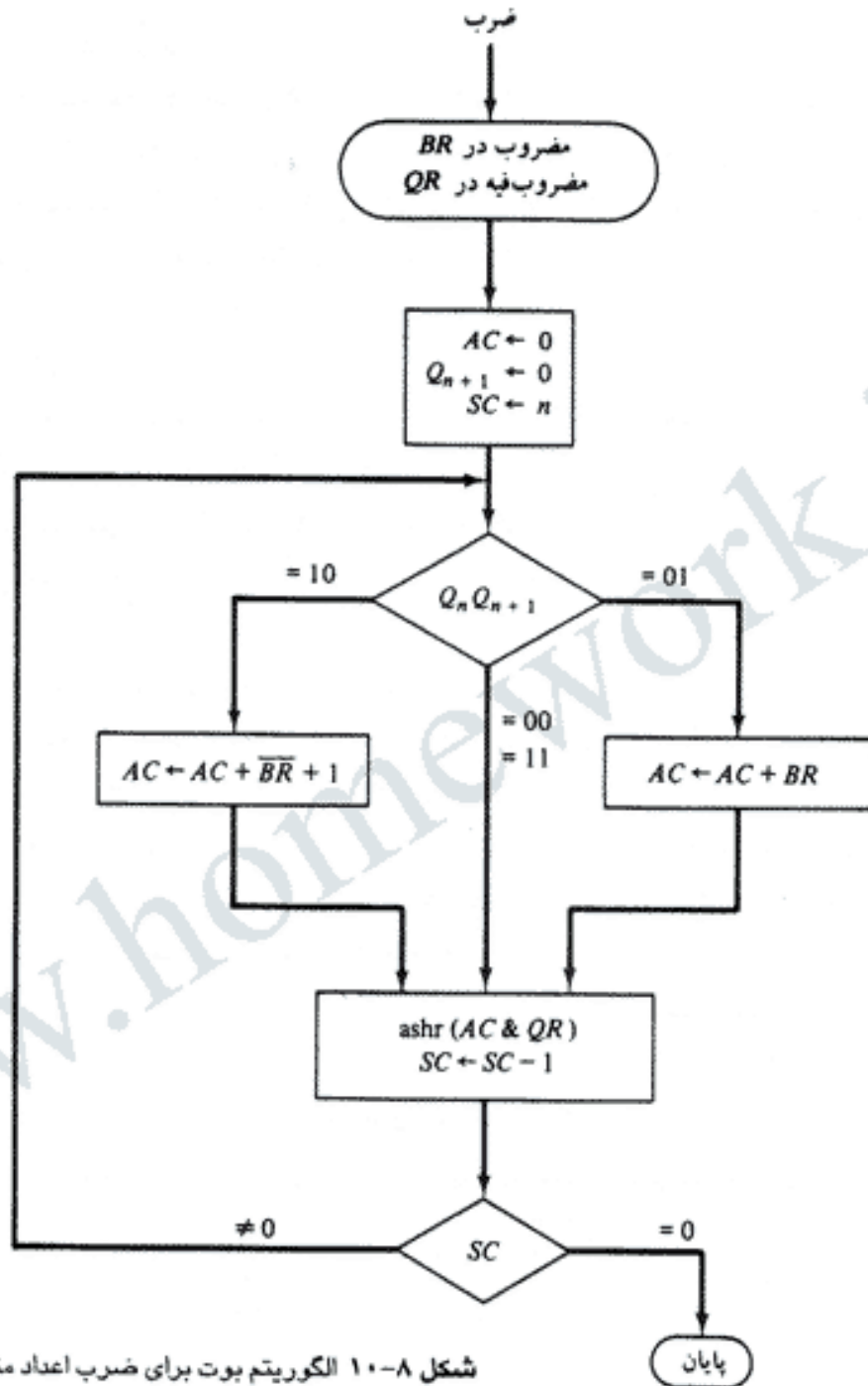
- جمع می‌شود، از آن تفریق شود و یا حاصلضرب جزئی بلا تغییر باقی بماند.
- ۱- به محض برخورد با اولین 1 کم ارزش در رشته 1 ها در مضروب فیه، مضروب از حاصلضرب جزئی کم می‌شود.
 - ۲- به محض برخورد با اولین 0 (بشرطی که قبل از آن 1 باشد) در رشته‌ای از 0 ها در مضروب فیه، مضروب با حاصلضرب جزئی جمع می‌شود.
 - ۳- وقتی که بیت جاری مضروب فیه همانند بیت قبلی باشد، حاصلضرب جزئی تغییری نمی‌کند.

الگوریتم فوق برای مضروب فیه‌های مثبت و یا منفی بفرم متمم 2 قابل استفاده است. این بدان علت است که مضروب فیه منفی به رشته‌ای از 1 ها خاتمه می‌یابد و آخرین عمل تفریق با وزن مناسب خواهد بود. مثلاً مضروب فیه -14 در نمایش متمم 2 عبارتست از 110010 و بصورت $-2^4 + 2^2 - 2^1 = -14$ با آن رفتار می‌شود.

پیاده‌سازی سخت افزار الگوریتم بوت آرایش ثبات شکل ۷-۱۰ را نیاز دارد. این شکل مشابه شکل ۵-۱۰ است جز اینکه بیت‌های علامت از بقیه ثبات‌ها تفکیک نشده‌اند. برای ملاحظه این اختلاف، ما ثبات‌های A و B و Q را بترتیب AC، BR و QR می‌نامیم. Q_n کم ارزشترین بیت مفروق منه در ثبات QR است. یک فلیپ فلاپ اضافی برای بررسی همزمان دو بیت از مضروب فیه به QR ملحق شده است. فلوچارت الگوریتم بوت در شکل ۸-۱۰ نشان داده شده است. AC و بیت الحاقی ابتدا پاک شده‌اند و توالی شمار SC برابر تعداد بیت‌های مضروب فیه یعنی n ، می‌شود. دو بیت از مضروب فیه که در Q_n و Q_{n+1} قرار دارند مورد بررسی قرار می‌گیرند. اگر دو بیت برابر 10 باشند بدان معنی است که اولین 1 در رشته 1 ها فرا رسیده است. این حالت، تفریق مضروب از حاصلضرب جزئی موجود در AC را لازم می‌دارد. اگر دو بیت 01 باشد، مفهوم این است که با اولین 0 در رشته 0 ها برخورد شده است. این حالت جمع مضروب با حاصلضرب جزئی موجود در AC را لازم می‌دارد. هرگاه دو بیت برابر باشند



شکل ۷-۱۰ سخت‌افزار الگوریتم بوت



شکل ۸-۱۰ الگوریتم بوت برای ضرب اعداد متمم ۲ علامت‌دار

حاصلضرب جزئی تغییری نمی‌کند. سرریز نمی‌تواند رخ دهد زیرا جمع و تفریق مضروب یک در میان انجام می‌شود. در نتیجه دو عددی که جمع می‌شوند، همواره علامت‌های مخالف دارند، و باین ترتیب امکان وقوع سرریز وجود نخواهد داشت. قدم بعدی شیفت حاصلضرب جزئی و مضروب فیه (شامل بیت Q_{n+1}) به راست است. این یک عمل شیفت حسابی به راست است که AC و QR را به راست جابجا کرده و بیت علامت در AC بلا تغییر می‌ماند (بخش ۶-۴ ملاحظه شود). توالی شمار کاهش یافته

و حلقه محاسبه n بار تکرار می‌گردد.

مثال عددی الگوریتم بوت در جدول ۳-۱۰ برای $n=5$ نشان داده شده است. این مثال، ضرب $117 = (-9) \times (-13)$ را نشان می‌دهد. توجه کنید که مضروب فیه در QR منفی و مضروب نیز در BR منفی است. حاصلضرب 10 بیتی در AC و QR قرار داشته و مثبت است. مقدار نهایی Q_{n+1} علامت اولیه مضروب فیه است و نباید بخشی از حاصلضرب تلقی شود.

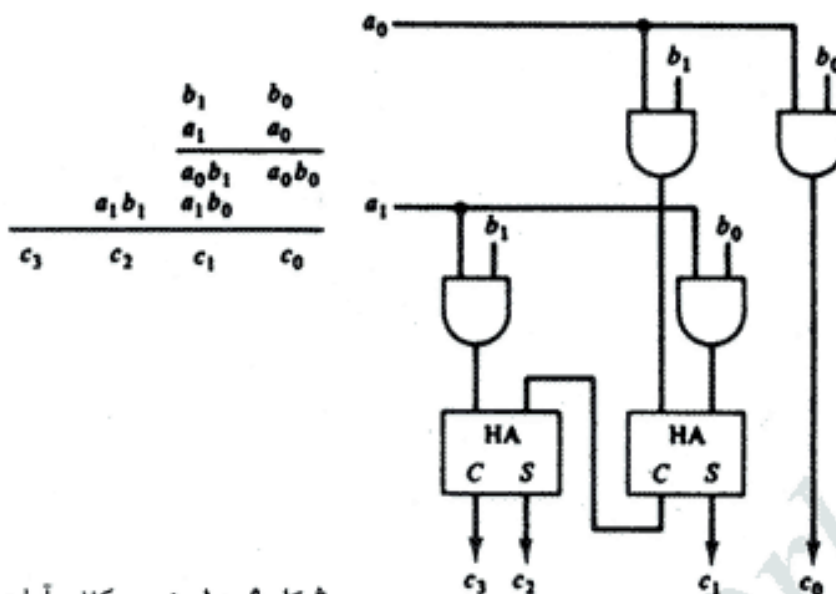
ضرب کننده آرایه‌ای

تست یک به یک بیت‌های مضروب فیه و تشکیل حاصلضرب جزئی یک عمل ترتیبی است که دنباله‌ای از جزء اعمال جمع و شیف‌ت را نیاز دارد. ضرب دو عدد دودویی بوسیله یک ریزعمل و یک مدار ترکیبی که بیت‌های ضرب را یکبار ایجاد کند تشکیل می‌گردد. این روش راه سریعی برای ضرب دو عدد است زیرا کل زمانی که صرف می‌شود زمان انتشار سیگنال‌ها از گیت‌هایی است که آرایه ضرب را تشکیل می‌دهند. با این وجود، یک ضرب کننده آرایه‌ای تعداد زیادی گیت لازم دارد و با این علت تا پیدایش مدارهای مجتمع اقتصادی تشخیص داده نشد.

برای اینکه بینیم چگونه یک ضرب کننده آرایه‌ای با یک مدار ترکیبی پیاده سازی می‌شود، ضرب دو عدد 2 بیت را مطابق شکل ۹-۱۰ در نظر بگیرید. بیت‌های مضروب b_1 و b_0 و بیت‌های مضروب فیه a_1 و a_0 و حاصلضرب $c_3c_2c_1c_0$ می‌باشند. اولین حاصلضرب جزئی با ضرب a_0 در b_1b_0 بدست می‌آید. ضرب دو بیت مانند a_0 و b_0 برابر 1 خواهد بود بشرطی که هر دو 1 باشند، در غیر این صورت 0 است. این با عمل AND یکسان است و می‌تواند با یک گیت AND پیاده سازی شود. همانطور که در شکل دیده می‌شود، اولین حاصلضرب جزئی با دو گیت AND حاصل می‌شود. دومین حاصلضرب جزئی از ضرب

جدول ۳-۱۰ مثال ضرب با الگوریتم بوت

$Q_n Q_{n+1}$	$BR = 10111$ $BR + 1 = 01001$	AC	QR	Q_{n+1}	SC
1 0	مقدار اولیه تفریق BR	00000 01001 01001	10011	0	101
1 1	ashr	00100	11001	1	100
0 1	ashr جمع BR	00010 10111 11001	01100	1	011
0 0	ashr	11100	10110	0	010
1 0	ashr تفریق BR	11110 01001 00111	01011	0	001
	ashr	00011	10101	1	000



شکل ۹-۱۰ ضرب کننده آرایه ۲ بیت در ۲ بیت

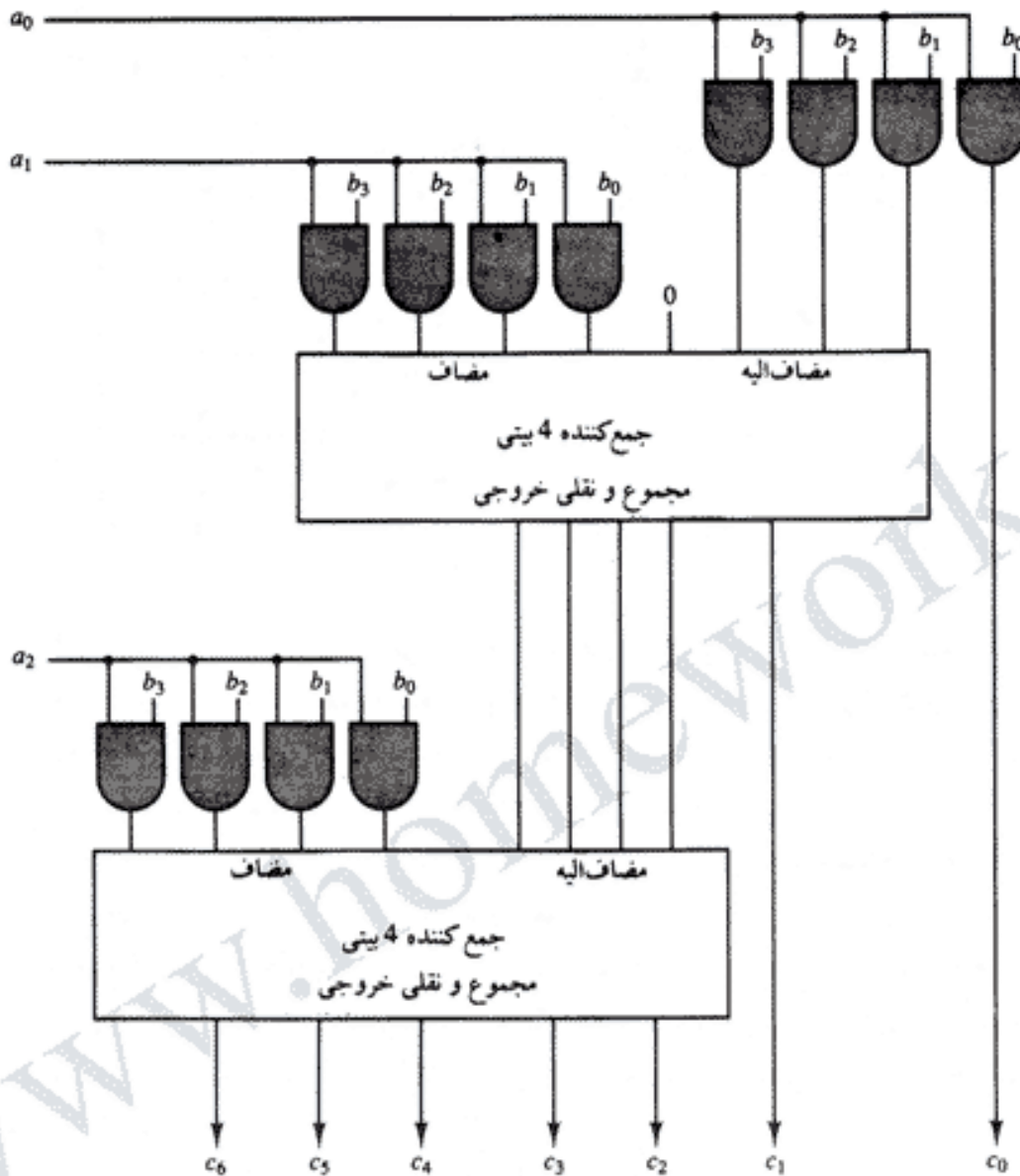
در $a_1 b_0$ و شیفت آن به چپ بدست می آید. دو حاصلضرب جزئی بوسیله دو مدار نیم جمع کننده (HA) با یکدیگر جمع می شوند. معمولاً، تعداد بیت های حاصلضرب جزئی بیشتر است و لازم خواهد بود تا از تمام جمع کننده استفاده شود. دقت داشته باشید که کم ارزشترین بیت حاصلضرب لازم نیست وارد جمع کننده شود زیرا توسط خروجی گیت AND تشکیل می گردد.

یک مدار ضرب کننده دودویی با بیت های بیشتر را می توان بروشی مشابه ساخت. هر بیت از مضروب فیه با هر بیت از مضروب، بتعداد بیت های مضروب فیه، AND می شود. خروجی دودویی در هر یک از سطوح گیت های AND بطور موازی با حاصلضرب جزئی قبلی جمع می شود و حاصلضرب جزئی جدیدی را تشکیل می دهد. آخرین سطح، حاصلضرب را ایجاد می نماید. برای مضروب فیه j بیتی و مضروب k بیتی نیاز به $j \times k$ گیت AND و $(j-1)k$ جمع کننده k بیتی برای تولید حاصلضرب $j+k$ بیتی لازم است.

به عنوان مثال دوم، یک مدار ضرب کننده را در نظر بگیرید که یک عدد دودویی چهاربیتی را در یک عدد سه بیتی ضرب می کند. فرض کنید مضروب را با $b_3 b_2 b_1 b_0$ و مضروب فیه را با $a_2 a_1 a_0$ نشان می دهیم. چون $k=4$ و $j=3$ است، ۱۲ گیت AND و ۲ جمع کننده ۴ بیتی مورد نیاز است تا حاصلضرب هفت بیتی را ایجاد کند. نمودار منطقی این ضرب کننده در شکل ۱۰-۱۰ نشان داده شده است.

۱۰-۴ الگوریتم های تقسیم

در تقسیم دو عدد ممیز ثابت در نمایش مقدار علامت دار با قلم و کاغذ، از اعمال مقایسه، شیفت و تفریق استفاده می شود. تقسیم دودویی از تقسیم دهمی ساده تر است زیرا ارقام خارج قسمت ۰ یا ۱ هستند و نیازی نیست تا بدانیم مقسوم یا باقیمانده چند بار در مقسوم علیه جای می گیرد. فرآیند تقسیم با



شکل ۱۰-۱۰ ضرب آرایه 4 بیت در 3 بیت

مثال عددی شکل ۱۰-۱۱ توضیح داده شده است. مقسوم علیه B از پنج بیت و مقسوم A از ده بیت تشکیل شده است. پنج بیت با ارزشتر مقسوم با مقسوم علیه مقایسه می شوند. چون این پنج بیت از B کوچکتر است، مقایسه B را با شش بیت با ارزشتر A آزمایش می کنیم. این عدد شش بیتی بزرگتر از B است، لذا رقم 1 را برای خارج قسمت در بالای بیت ششم مقسوم می نویسیم. سپس مقسوم علیه را یک بار به راست شیفت می دهیم و آن را از مقسوم کم می کنیم. تفاضل بدست آمده باقیمانده جزئی¹ خوانده می شود زیرا تقسیم ممکن است در همین مرحله پایان یابد و خارج قسمت 1 و باقیمانده ای برابر با این باقیمانده جزئی را داشته

1- Partial Reminder

Divisor: $B = 10001$	11010 $\overline{)0111000000}$ 01110 011100 -10001 -010110 $--10001$ $--001010$ $---010100$ $----10001$ $----000110$ $-----00110$	خارج قسمت $Q =$ مقسوم $A =$ 5 بیت از $A < B$ ، خارج قسمت 5 بیت دارد 6 بیت از $A \geq B$ هستند جا بجایی B به راست و تفریق، قرار دادن 1 در Q 7 بیت از باقیمانده بزرگتر یا مساوی B هستند جا بجایی B به راست و تفریق، قرار دادن 1 در Q باقیمانده کوچکتر از B ، قرار دادن 0 در Q ؛ شیفت B به راست باقیمانده بزرگتر یا مساوی B شیفت B به راست و تفریق، قرار دادن 1 در Q باقیمانده کوچکتر از B ، قرار دادن 0 در Q باقیمانده نهایی
-------------------------	--	---

شکل ۱۱-۱۰ مثال تقسیم دودویی

باشیم. روند تقسیم با مقایسه باقیمانده جزئی با مقسوم علیه ادامه می یابد. اگر باقیمانده جزئی بزرگتر یا مساوی مقسوم علیه باشد، بیت خارج قسمت برابر 1 خواهد بود. سپس مقسوم علیه یک واحد به راست شیفت داده شده و از باقیمانده جزئی کم می شود. اگر باقیمانده جزئی کوچکتر از مقسوم علیه باشد، بیت خارج قسمت 0 خواهد بود و تفریق لازم نیست. در هر صورت مقسوم علیه یک مرتبه به راست شیفت داده می شود. توجه کنید که در نتیجه این روند هم خارج قسمت و هم باقی مانده بدست خواهد آمد.

پیاده سازی سخت افزاری برای داده های بانمایش مقدار علامت دار

هنگام پیاده سازی تقسیم در یک کامپیوتر دیجیتال، بهتر است روند کمی تغییر یابد. در عوض شیفت مقسوم علیه به سمت راست، مقسوم یا باقیمانده جزئی را به چپ شیفت می دهیم و به این ترتیب موقعیت نسبی دو عدد همان موقعیت مورد نظر خواهد بود. تفریق را می توان با جمع A و متمم 2 عدد B انجام داد. نسبت اندازه ها از رقم نقلی انتهایی مشخص می گردد.

سخت افزار مربوط به پیاده سازی تقسیم همان سخت افزار ضرب است و از قطعات شکل ۵-۱۰ تشکیل شده است. در اینجا ثبات EAQ به چپ شیفت داده شده و 0 در Q_n وارد شده و مقدار قبلی E هم از دست می رود. مثال عددی مجدداً در شکل ۱۲-۱۰ برای روشن کردن روند تقسیم نشان داده شده است. مقسوم علیه در ثبات B و مقسوم که طولی دو برابر دارد در ثبات های A و Q ذخیره می شوند. مقسوم به سمت چپ شیفت داده می شود و متمم 2 مقسوم علیه با آن جمع شده و لذا عمل تفریق تحقق می یابد. اطلاعات مربوط به نسبت اندازه ها در E موجود است. اگر $E=1$ باشد بمعنی $A \geq B$ است. برای خارج قسمت، بیت 1 وارد Q_n می گردد و باقیمانده جزئی به سمت چپ شیفت پیدا می کند تا روند تکرار گردد. اگر $E=0$ باشد، یعنی $A < B$ است و بنابراین خارج قسمت در Q_n برابر 0 (که هنگام شیفت وارد آن شده است) می ماند. سپس مقدار B مجدداً جمع می شود تا باقیمانده جزئی در A به مقدار قبلی اش بازگردانده شود. باقیمانده جزئی به چپ شیفت داده شده و روند مجدداً تکرار می گردد تا اینکه

	E	A	Q	SC
مقسوم : shl EAQ	0	01110	00000	5
B+1 جمع		11100	00000	
		<u>01111</u>		
E = 1	1	01011		
قرار دهید $Q_n=1$	1	01011	00001	4
shl EAQ	0	10110	00010	
B+1 جمع		<u>01111</u>		
E = 1	1	00101		
قرار دهید $Q_n=1$	1	00101	00011	3
shl EAQ	0	01010	00110	
B+1 جمع		<u>01111</u>		
$Q_n=0$, E=0 بگذارید	0	11001	00110	
B جمع		<u>10001</u>		
بازیابی باقیمانده	1	01010		2
shl EAQ	0	10100	01100	
B+1 جمع		<u>01111</u>		
E = 1	1	00011		
قرار دهید $Q_n=1$	1	00011	01101	1
shl EAQ	0	00110	11010	
B+1 جمع		<u>01111</u>		
$Q_n=0$, E=0 بگذارید	0	10101	11010	
B جمع		<u>10001</u>		
بازیابی باقیمانده	1	00110	11010	0
E را نادیده بگیرید		00110		
باقیمانده در A			11010	
خارج قسمت در Q				

شکل ۱۲-۱۰ مثال تقسیم دودویی با سخت افزار دیجیتال

هر پنج بیت خارج قسمت ایجاد شود. توجه کنید که ضمن شیفت باقیمانده جزیی به چپ، بیت های خارج قسمت نیز شیفت داده می شود و پس از پنج شیفت، خارج قسمت در Q و باقیمانده در A خواهد بود. قبل از نمایش الگوریتم بصورت فلوجارت، باید علامت نتیجه و حالت سرریز احتمالی را در نظر داشت. علامت خارج قسمت از علامت های مقسوم و مقسوم علیه تعیین می شود. اگر علامت های این دو یکسان باشند، علامت خارج قسمت مثبت خواهد بود و اگر مخالف باشند، علامت منفی است. علامت باقیمانده همانند علامت مقسوم است.

سرریز در تقسیم

عمل تقسیم ممکن است منجر به سرریز در خارج قسمت شود. هنگام استفاده از کاغذ و قلم، این امر مشکلی نیست ولی هنگام پیاده سازی عمل با سخت افزار مسئله ساز خواهد بود. دلیل این است که طول ثبات ها محدود است و نمی توانند عددی را که از طول استاندارد تجاوز کند نگه دارند. برای درک موضوع، سیستمی را که دارای ثبات های پنج بیتی است در نظر بگیرید. مازیک ثبات برای نگهداری

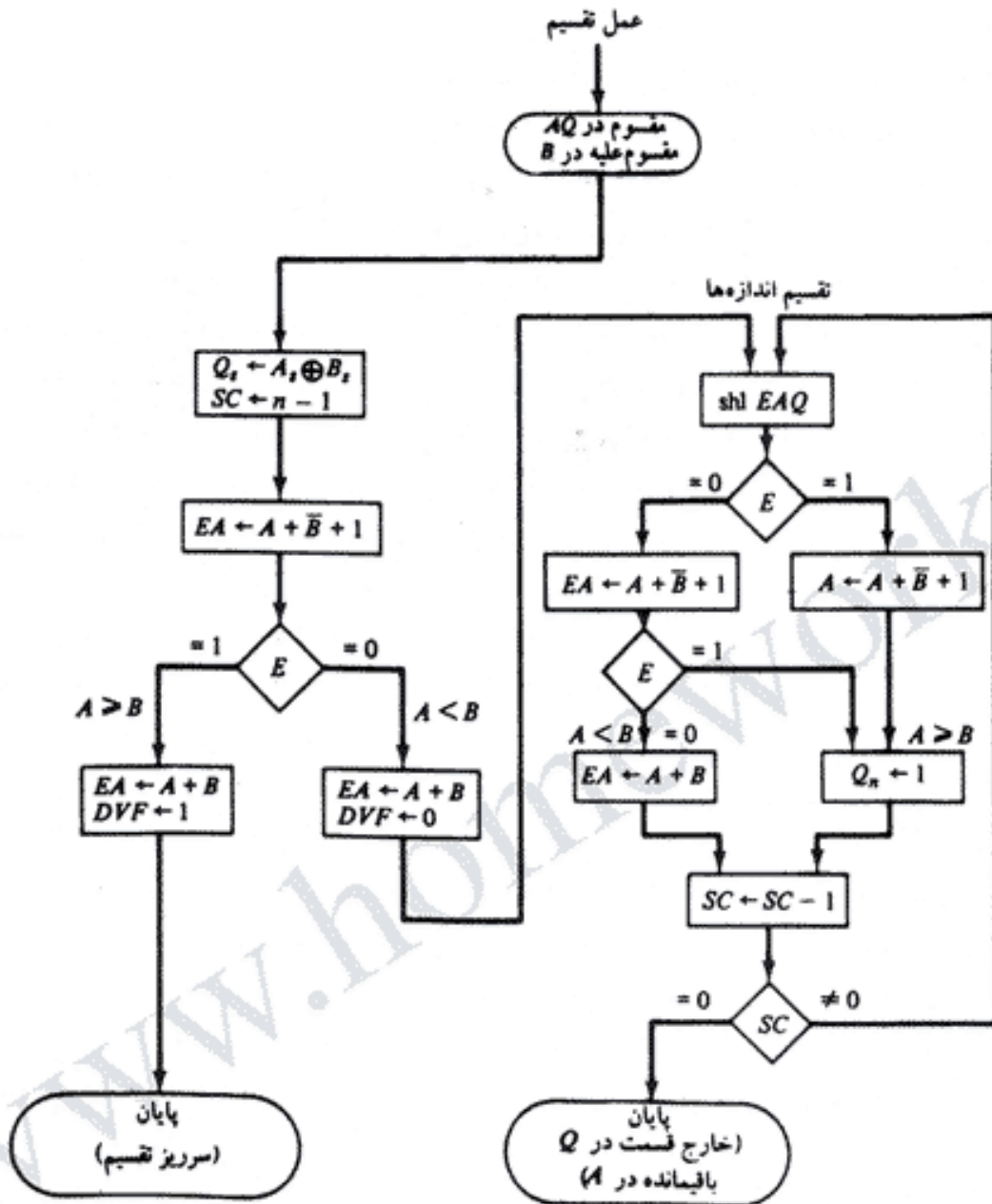
مقسوم علیه و دو ثبات برای نگهداری مقسوم استفاده می‌کنیم. با توجه به مثال شکل ۱۱-۱۰ می‌بینیم که خارج قسمت شش بیتی است بشرطی که پنج بیت باارزشتر مقسوم، عددی بزرگتر از مقسوم علیه باشد. خارج قسمت می‌باید در یک ثبات پنج بیتی استاندارد ذخیره شود، بنابراین بیت سرریز مستلزم یک فلیپ‌فلاپ اضافی برای ذخیره بیت ششم خواهد بود. در عملیات عادی کامپیوتر باید از بکارگیری این بیت سرریز خودداری شود زیرا در این صورت طول گُل خارج قسمت بیش از مقداری خواهد بود که بتوان آن را به واحدهای حافظه‌ای که طول استاندارد را دارند، یعنی ثبات‌ها، منتقل کرد. در سخت افزار و یا نرم‌افزار کامپیوتر و یا ترکیبی از هر دو باید امکاناتی را جهت آشکارسازی این وضعیت فراهم کرد.

اگر طول مقسوم دو برابر مقسوم علیه باشد، حالت سرریز را می‌توان بطریق زیر بیان کرد: حالت سرریز در تقسیم هنگامی اتفاق می‌افتد که بیت‌های نیمه پرارزشتر مقسوم، عددی بزرگتر یا مساوی مقسوم علیه را تشکیل دهند. مسئله دیگری که باید در تقسیم در نظر داشت این است که از تقسیم بر صفر احتراز شود. حالت سرریز در تقسیم مراقبت از این وضعیت را بعهده دارد. این بدان علت است که اگر مقسوم علیه صفر باشد، مقسوم قطعاً از مقسوم علیه بزرگتر یا مساوی آن است. حالت سرریز معمولاً با ۱ شدن فلیپ فلاب خاصی مشخص می‌گردد و آنرا فلیپ فلاب سرریز خوانده و با DVF نشان می‌دهیم. در قبال وقوع سرریز تقسیم بطرق مختلف می‌توان عکس العمل نشان داد. در بعضی کامپیوترها مسئولیت با برنامه نویس است تا پس از هر دستورالعمل تقسیم DVF را چک کند. در این صورت می‌توان به زیرروالی انشعاب کرده و اقدامی اصلاحی همچون تغییر مقیاس داده‌ها را برای جلوگیری از سرریز انجام داد. در برخی کامپیوترهای قدیمی‌تر، وقوع سرریز تقسیم سبب توقف کامپیوتر می‌شد و این حالت، توقف تقسیم نامیده می‌شد. امروزه متوقف کردن کار کامپیوتر توصیه نمی‌شود زیرا اتلاف کننده وقت است. روشی که در اکثر کامپیوترها انجام می‌شود تولید درخواست وقفه^۱ بهنگام ۱ شدن DVF است. این وقفه موجب می‌شود تا کامپیوتر اجرای برنامه جاری را بتعویق انداخته و به روال سرویس دهی برای اقدام اصلاحی مورد تقاضا انشعاب نماید. معمول‌ترین اقدام اصلاحی خارج شدن از برنامه و چاپ پیغام خطایی است که دلیل عدم امکان اتمام برنامه را شرح می‌دهد. از این پس، وظیفه کاربری که برنامه را نوشته این است که تا مقیاس داده‌ها را تغییر دهد و یا هر اقدام اصلاحی دیگری را بعمل آورد. بهترین راه جلوگیری از سرریز تقسیم استفاده از داده‌های ممیز شناور است. ما در بخش ۵-۱۰ خواهیم دید که سرریز تقسیم بسادگی بهنگام استفاده از اعداد ممیز شناور قابل پیش‌گیری است.

الگوریتم سخت افزاری

الگوریتم سخت افزاری سرریز در فلوچارت شکل ۱۳-۱۰ نشان داده شده است. مقسوم در A و Q و مقسوم علیه در B قرار دارد. علامت نتیجه به عنوان قسمتی از خارج قسمت به Q_8 منتقل می‌شود. برای مشخص کردن تعداد بیت‌های خارج قسمت، عدد ثابتی در توالی شمار SC قرار داده می‌شود. همانند

1- Interrupt Request



شکل ۱۳-۱۰ فلوجارت عمل تقسیم

فرض می‌کنیم که عملوندها از واحد حافظه n بیتی به ثبات‌ها منتقل شوند. چون عملوند همراه با علامتش ذخیره می‌شود، یک بیت از کلمه حافظه بوسیله علامت و $n-1$ بیت بوسیله اندازه آن اشغال می‌گردد. حالت سرریز تقسیم با تقریب مقسوم علیه در B از نیمی از بیت‌های ذخیره شده در A تست می‌شود. اگر $A \geq B$ باشد، فلیپ سرریز تقسیم، DVF برابر 1 شده و عمل بطور ناقص متوقف می‌شود. اگر $A < B$ باشد، سرریزی رخ نداده و مقدار مقسوم با جمع B با A بازیافته می‌شود. تقسیم اندازه‌های دو عدد با جابجا کردن مقسوم موجود در AQ به چپ شروع می‌شود که در این جابجایی بیت پرارزتر به E

منتقل می شود اگر بیت انتقال یافته به E برابر 1 باشد، درمی یابیم که $EA > B$ است زیرا EA از رقم 1 و بدنبال آن $n-1$ بیت تشکیل شده است، در حالیکه B فقط $n-1$ بیت است. در این حالت، باید B از EA تفریق شود و مقدار 1 بعنوان یک بیت خارج قسمت در Q_n قرار می گیرد. چون ثبات A فاقد بیت بالا رتبه مقسوم است (که در E قرار دارد) مقدار آن $EA - 2^{n-1}$ خواهد بود. از جمع متمم دو B با این مقدار داریم:

$$(EA - 2^{n-1}) + (2^{n-1} - B) = EA - B$$

اگر بخواهیم E برابر 1 بماند رقم نقلی عبارت فوق به E منتقل نمی شود. اگر عمل شیفت به چپ مقدار 0 را وارد E کند، مقسوم علیه با جمع متمم 2 آن تفریق می شود و مقدار رقم نقلی به E انتقال می یابد. اگر $E=1$ شود، $A \geq B$ است؛ بنابراین به Q_n ، 1 داده خواهد شد. اگر $E=0$ باشد، $A < B$ و عدد اولیه با جمع B با A بازیابی می شود. در مورد اخیر مقدار 0 را در Q_n باقی می گذاریم (مقدار 0 هنگام شیفت وارد آن شده است).

این روند مجدداً در حالی که ثبات A حاوی باقیمانده است تکرار می گردد. پس از $n-1$ تکرار، اندازه خارج قسمت در ثبات Q و باقیمانده در ثبات A خواهد بود. علامت خارج قسمت در Q_8 و علامت باقیمانده نیز که همان علامت مقسوم است در A_8 قرار دارد.

سایر الگوریتم ها

روش سخت افزاری که در فوق شرح داده شد روش بازیافتی^۱ خوانده می شود. دلیل این نامگذاری این است که باقیمانده جزئی از جمع مقسوم علیه با تفاضل منفی بازیافته می شود. در روش دیگر برای تقسیم اعداد وجود دارد، روش مقایسه ای و روش غیربازیافتی^۲. در روش مقایسه ای، A و B قبل از تفریق مقایسه می شوند. سپس اگر $A \geq B$ باشد B از A تفریق می شود. اگر $A < B$ باشد کاری انجام نمی شود. باقیمانده جزئی به چپ شیفت داده شده و مجدداً مقایسه تکرار می گردد. می توان مقایسه را قبل از تفریق با واریسی رقم نقلی انتهایی جمع کننده موازی، قبل از انتقال آن به ثبات E انجام داد.

در روش غیربازیافتی، اگر اختلاف منفی باشد B جمع نمی شود، و در عوض تفاضل منفی به چپ شیفت داده شده و سپس با B جمع می گردد. برای اینکه ببینیم چگونه این ممکن است، حالتی را در نظر بگیرید که در آن $A < B$ است. با توجه به فلوچارت ۹-۱۱ اعمال انجام شده عبارتند از $A - B + B$ ؛ یعنی B ابتدا تفریق شده و مجدداً جمع می شود تا A بازیافته شود. در دور بعدی اجرای حلقه، این عدد به چپ شیفت داده شده (یا در 2 ضرب می شود) و دوباره B از آن کم می شود. نتیجه این اعمال $2(A - B) - B = 2A - 3B$ است. این نتیجه در روش غیربازیافتی بارها کردن $A - B$ بهمان گونه که هست، یعنی $A - B$ ، بدست می آید. در مرتبه بعدی اجرای حلقه، عدد به چپ شیفت داده شده و B به آن اضافه می شود تا $2(A - B) + B = 2A - B$ بدست آید که همان نتیجه قبلی است. بنابراین، در روش غیربازیافتی، اگر مقدار

1- Restoring Method

2- Nonrestoring Method

قبلی Q_n برابر 1 باشد B تفریق می‌گردد، اما اگر مقدار قبلی Q_n برابر 0 باشد B جمع می‌شود و بازایی باقیمانده جزئی لازم نیست. باین ترتیب یک مرحله، یعنی جمع کردن مقسوم علیه اگر A کوچکتر از B باشد، صرفه‌جویی می‌شود ولی نیاز به کنترل خاصی برای بخاطر سپردن نتیجه قبلی دارد. اولین باری که مقسوم شیفتم می‌یابد، B باید تفریق شود. همچنین، اگر آخرین بیت خارج قسمت 0 باشد، باقیمانده جزئی باید بازایی شود تا باقیمانده نهایی صحیح باشد.

۵-۱۰ عمل‌های حسابی ممیز شناور

بسیاری از زبان‌های برنامه‌نویسی امکاناتی برای مشخص کردن عددهای ممیز شناور دارند. متداول‌ترین روش، مشخص کردن آنها با عبارت real (حقیقی) در برابر اعداد ممیز ثابت است که با عبارت Integer (صحیح) مشخص می‌گردند. هر کامپیوتری که دارای کامپایلر برای اینگونه زبانهای سطح بالا باشد باید امکاناتی برای انجام عمل‌های حسابی ممیز شناور داشته باشد. این عملیات غالباً در سخت افزار کامپیوتر پیاده سازی می‌شود. اگر سخت افزاری برای این اعمال وجود نداشته باشد، کامپایلری باید طراحی شود که دارای مجموعه‌ای از زیرروال‌های نرم افزاری ممیز شناور باشد. هرچند روش سخت افزاری گرانتر است، ولی کارایی آن بقدری از روش نرم افزاری بیشتر است که در اکثر کامپیوترها، سخت افزار ممیز شناور کار گذاشته می‌شود و فقط در کامپیوترهای خیلی کوچک از آن صرف نظر می‌شود.

ملاحظات اساسی

نمایش ممیز شناور داده در بخش ۴-۳ معرفی شد. عدد ممیز در ثباتهای کامپیوتر از دو بخش تشکیل می‌شود: مانتیس m و نمای e. این دو بخش نماینده عددی است که از ضرب m در 10^e به نمای e بدست می‌آید. یعنی

$$m \times 10^e$$

مانتیس ممکن است که عددی کسری یا صحیح باشد. محل ممیز و مقدار پایه 10^e فرض تصور شده‌اند و در ثبات‌ها وارد نشده‌اند. مثلاً نمایش کسری و پایه 10 را در نظر بگیرید. عدد دهدهی 537.25 در یک ثبات با $m=53725$ و $e=3$ نمایش داده می‌شود و چنین تفسیر می‌گردد که نماینده عدد ممیز شناور زیر است:

$$0.53725 \times 10^3$$

اگر با ارزشترین رقم مانتیس یک عدد ممیز شناور و غیرصفر باشد آن را نرمالیزه می‌کنیم. در نتیجه مانتیس دارای بیشترین تعداد ممکن رقم‌های معنی‌دار خواهد بود. صفر را نمی‌توان نرمالیزه کرد زیرا رقم غیرصفر ندارد. در نمایش ممیز شناور، صفر را با مقدار تمام 0 در مانتیس و نما نشان می‌دهیم. نمایش ممیز شناور محدوده اعدادی را که می‌توان در یک ثبات جای داد مشخص می‌کند. کامپیوتری با کلمه‌های 48 بیتی را در نظر بگیرید. چون یک بیت برای علامت رزرو شده است محدوده اعداد صحیح

ممیز ثابت $(1-2^{47}) \pm$ خواهد بود که تقریباً $10^{14} \pm$ می باشد. از این 48 بیت می توان برای نمایش یک عدد میز شناور با 36 بیت برای مانتیس و 12 بیت برای نمائستفاده کرد. با فرض نمایش کسری برای مانتیس و با احتساب دوبیت برای علامت ها، محدوده اعدادی که می توان جای داد برابرست با

$$\pm (1-2^{-35}) \times 2^{2047}$$

این عدد از کسری که حاوی 35 رقم 1، نمای 11 بیتی (منهای علامت آن) و اینکه $2^{11} - 1 = 2047$ می باشد، بدست می آید. بزرگترین عددی که می توان آن را جای داد 10^{615} است، که عدد بسیار بزرگی است. مانتیس می تواند 35 بیتی باشد (جدا از علامت) و اگر عدد صحیح تلقی شود می تواند تا $(1-2^{35})$ را ذخیره نماید. این تقریباً برابر با 10^{10} می باشد، که معادل یک عدد دهدهی 10 رقمی است. کامپیوترهایی که دارای طول کلمه های کوتاه تری هستند برای نمایش اعداد ممیز شناور از دو یا چند کلمه استفاده می نمایند. یک میکروکامپیوتر 8 بیتی ممکن است از چهار کلمه برای نمایش یک عدد ممیز شناور استفاده کند. یکی از کلمات 8 بیتی برای نما و 24 بیت سه کلمه دیگر برای مانتیس در نظر گرفته شده اند.

اعمال حسابی با اعداد ممیز شناور بسیار پیچیده تر از اعداد ممیز ثابت است و اجرای آنها زمان بیشتری می برد و سخت افزار پیچیده تری نیاز دارد. جمع یا تفریق دو عدد مستلزم معین کردن محل ممیز است زیرا قبل از جمع یا تفریق مانتیس ها، باید بخش نمای دو عدد با هم مساوی شود. هم ردیف کردن با جابجایی یکی از مانتیس ها و تنظیم نمای آن تا جایی که با دیگری برابر شود انجام می شود. جمع اعداد ممیز شناور زیر را در نظر بگیرید.

$$0.5372400 \times 10^2 \\ + 0.1580000 \times 10^{-1}$$

لازم است که دو نما قبل از جمع مانتیس ها با هم مساوی شوند. می توانیم عدد اول را سه مکان به چپ، یا عدد دوم را سه مکان به راست شیفت دهیم. وقتی که مانتیس ها در ثبات ها ذخیره شوند، شیفت به چپ سبب از دست دادن ارقام با ارزش تر می شود. روش دوم ترجیح دارد چون میزان دقت را کاهش می دهد در حالی که روش اول ممکن است موجب بروز خطا شود. رویه هم ردیف کردن معمولاً با این ترتیب است که مانتیسی را که نمای کوچکتر دارد باندازه اختلاف بین نماها به راست شیفت دهیم. پس از این عمل مانتیس ها می توانند با هم جمع شوند.

$$0.5372400 \times 10^2 \\ + 0.0001580 \times 10^2 \\ \hline 0.5373980 \times 10^2$$

هنگامی که دو مانتیس نرمالیزه شده با هم جمع شوند حاصل جمع ممکن است رقم سرریز داشته

باشد. سرریز را به سادگی می توان با شیفت حاصل جمع به راست و افزایش نما تصحیح کرد. در تفریق مثل مثال زیر نتیجه ممکن است دارای صفرهایی در مکان های بالارته باشد.

$$\begin{array}{r} 0.56780 \times 10^5 \\ - 0.56430 \times 10^5 \\ \hline 0.00350 \times 10^5 \end{array}$$

وقتی عدد ممیز شناوری دارای 0 در مکان های با ارزشتر باشد گوئیم فروریز¹ دارد. برای نرمالیزه کردن اعداد فروریز دار، لازم است تا مانتیس به چپ شیفت داده شود و از نما یک واحد کسر گردد تا رقم غیرصفر در اولین مکان ظاهر شود. در مثال فوق، لازم است مانتیس را دوبار به چپ شیفت دهیم تا 0.35000×10^3 حاصل شود. در بسیاری از کامپیوترها رویه نرمالیزه کردن پس از هر عمل صورت می گیرد تا از نرمالیزه شدن نتایج اطمینان حاصل شود.

ضرب و تقسیم ممیز شناور نیازی به هم ردیف کردن مانتیس ها ندارند. حاصلضرب با ضرب دو مانتیس و جمع نماها شکل می گیرد. عمل تقسیم از تقسیم دو مانتیس و تفریق نماها بدست می آید. اعمال اجرا شده با مانتیس ها مشابه اعداد ممیز ثابت است، بنابراین هر دو نوع می توانند از ثبات ها و مدارهای مشترکی استفاده کنند. عملیات مربوط به نما مقایسه و افزایش (برای هم ردیفی مانتیس ها)، جمع و تفریق (برای ضرب و تقسیم)، و کاهش (برای نرمالیزه کردن نتیجه) می باشد. نما را می توان به یکی از سه روش ممکن نمایش داد: مقدار علامت دار، متمم 2 علامت دار و متمم 1 علامت دار. چهارمین روشی که در بسیاری از کامپیوترها بکار گرفته می شود نمای خورانده شده یا بایاس است. در این نمایش، بیت علامت بعنوان یک عضو جدا در نظر گرفته نمی شود. بایاس عدد مثبتی است که بهنگام تشکیل عدد ممیز شناور به هر نما اضافه می شود. در نتیجه تمام نماها در درون مثبت خواهند بود. مثال زیر می تواند این نوع نمایش را روشن سازد. فرض کنید نماها از 50- تا 49+ متغیر باشند. دروناً نما پس از جمع با بایاس با دو رقم نشان داده می شود. (بدون علامت). ثبات نما حاوی عدد 50+ خواهند بود، که e نما می واقعی است. لذا نماها در ثبات بصورت اعداد مثبتی از 00 تا 99 نمایش داده خواهند شد. نماهای مثبت در ثبات ها، محدوده اعداد 99 تا 50 را دارند. تفریق 50 از آنها مقادیر مثبت 0 تا 49 را می دهد. نماهای منفی دو ثباتها محدوده 49 تا 00 را دارند. تفریق 50 مقدار منفی 1- تا 50- را می دهد.

مزیت استفاده از نماهای بایاس شده این است که فقط اعداد مثبت را شامل می شوند. در نتیجه مقایسه نسبت اندازه های آنها ساده تر بوده و لازم نیست به علامت آنها توجه کنیم. لذا هنگام هم ردیف کردن مانتیس ها می توان از مقایسه گر مقدار (اندازه) برای مقایسه اندازه های آنها استفاده کرد. مزیت دیگر آنها این است که کوچکترین نمای پایین ممکن، تمام 0 است. در این صورت نمایش ممیز شناور 0

1- Underflow

متشکل از مانتیس 0 و کوچکترین نمای ممکن خواهد بود.

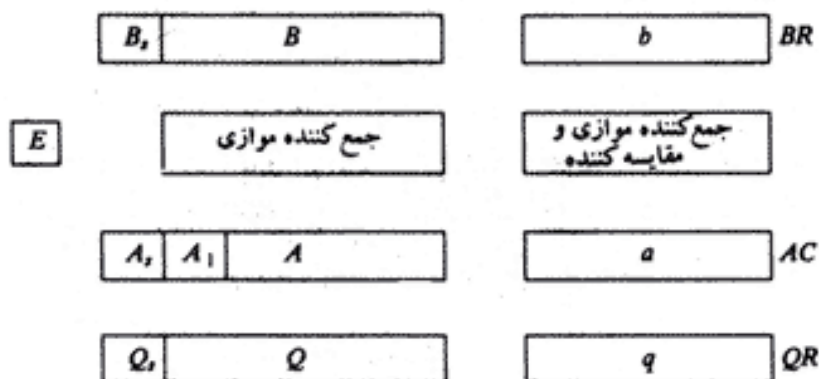
در مثال‌های بالا، از عددهای دهمی برای نشان دادن برخی از مفاهیمی که هنگام کار با اعداد ممیز شناور باید آن‌ها را دانست استفاده کردیم. واضح است که مفاهیم مشابهی در مورد اعداد دودویی نیز صادق است. الگوریتم‌های حاصل در این بخش برای اعداد دودویی است. در مورد حساب دهمی در کامپیوتر در بخش بعدی صحبت خواهیم کرد.

آرایش ثبات‌ها

آرایش ثبات‌ها برای اعمال ممیز شناور کاملاً مشابه اعمال ممیز ثابت است. قاعده کلی این است که همان ثبات‌ها و جمع‌کننده برای پردازش مانتیس بکار می‌روند. اختلاف موجود بر سر کار با نماهاست. سازمان ثبات‌ها برای اعمال ممیز شناور در شکل ۱۴-۱۰ نشان داده شده است. در این شکل سه ثبات BR ، AC و QR وجود دارد. هر ثبات به دو زیر بخش تقسیم شده است. بخش مانتیس همان سمبل‌های حروف بزرگ نمایش ممیز ثابت را دارد. بخش نما از سمبل حروف کوچک متناظر استفاده می‌نماید.

فرض بر این است که هر عدد ممیز شناور دارای یک مانتیس با نمایش مقدار علامت‌دار و یک نمای بایاس شده باشد. بنابراین AC دارای مانتیسی است که علامتش در A_8 و اندازه‌اش در A است. نما هم در بخشی از ثبات قرار دارد که با حرف کوچک a مشخص شده است. نمودار با ارزشترین بیت A_1 که A_1 می‌باشد، را بطور جداگانه نشان می‌دهد. برای نرمالیزه شدن این عدد بیت مذکور باید 1 باشد. دقت کنید که سمبل AC نماینده کل ثبات است، یعنی زنجیره a ، A و A_8 .

بطور مشابه، ثبات BR به B_8 ، B و b و QR به Q_8 ، Q و q تقسیم می‌شوند. یک جمع‌کننده موازی دو مانتیس را جمع می‌کند و مجموع را به A و نقلی را به E انتقال می‌دهد. جمع‌کننده موازی جداگانه‌ای برای نماها بکار رفته است. چون نماها بایاس شده‌اند، علامت متمایزی ندارند ولی بصورت یک کمیت مثبت بایاس شده نشان داده شده‌اند. فرض بر این است که اعداد ممیز شناور آنقدر بزرگ باشند که شانسی



شکل ۱۴-۱۰ ثبات‌های مربوط به عملیات حسابی ممیز-شناور

وقوع سرریز بسیار کم باشد، و به این دلیل سرریز نما چشم پوشی می‌گردد. نماها به یک مقایسه گر مقدار نیز متصل اند تا اندازه نسبی آنها را بصورت سه خروجی دودویی نشان دهند.

عدد مانتیس بصورت کسری در نظر گرفته می‌شود، لذا نقطه ممیز دودویی در سمت چپ بخش اندازه فرض خواهد شد. نمایش صحیح برای اعداد ممیز شناور موجب بروز مشکلات خاصی در مقیاس اعداد در حین ضرب و تقسیم می‌گردد. برای احتراز از این مشکلات، نمایش کسری بکار می‌رود.

اعداد درون ثبات‌ها از آغاز نرمالیزه فرض می‌شوند. پس از هر عمل حسابی، نتیجه نرمالیزه خواهد شد. بنابراین تمام عملوندهای ممیز شناور که از حافظه بیایند و یا به حافظه بروند همواره نرمالیزه هستند.

جمع و تفریق

در حین جمع یا تفریق، دو عملوند ممیز شناور در AC و BR قرار خواهند داشت. مجموع یا تفاضل در AC شکل می‌گیرد. الگوریتم می‌تواند در چهار بخش متوالی تقسیم گردد.

۱- تست صفر بودن عملوندها

۲- هم ردیف کردن مانتیس‌ها

۳- جمع یا تفریق مانتیس‌ها

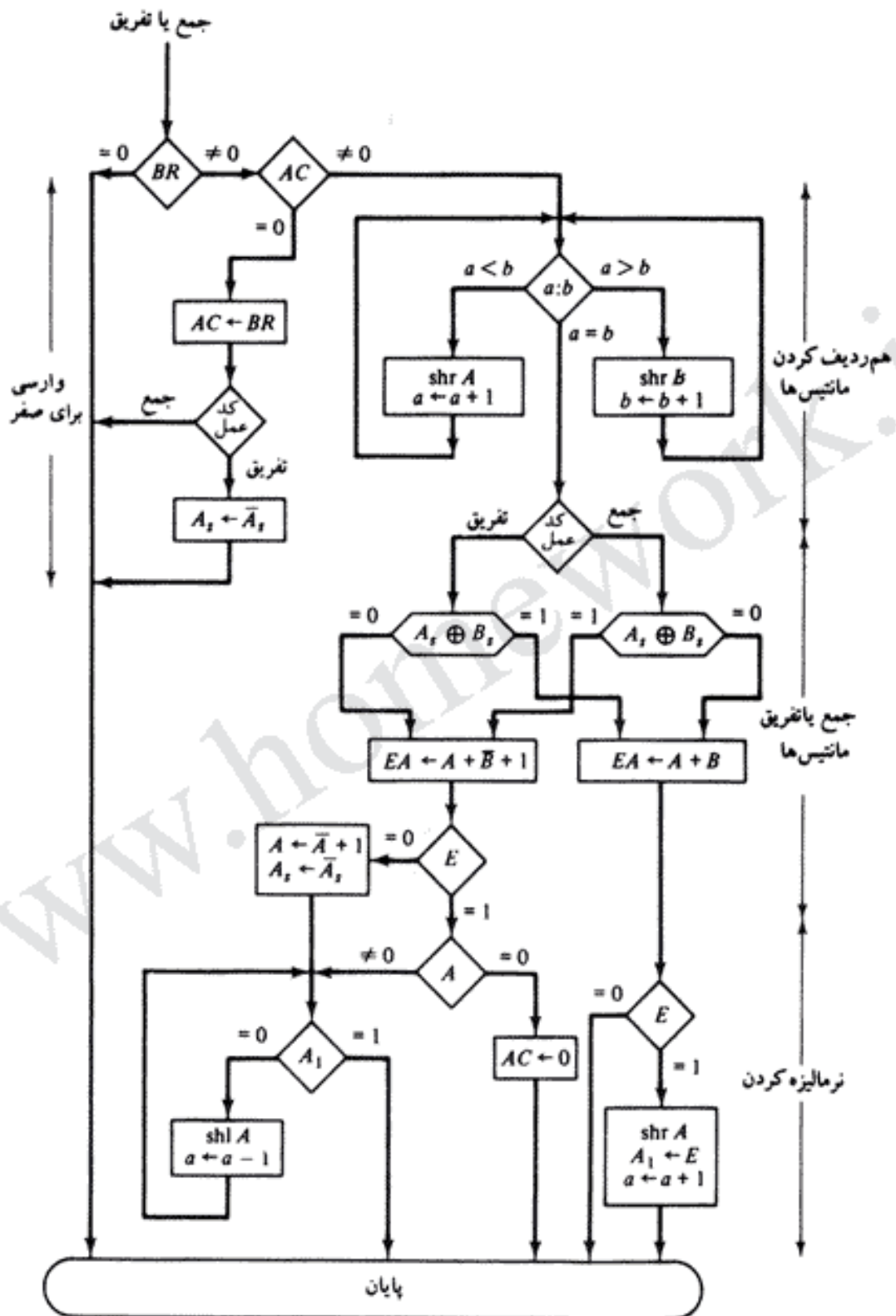
۴- نرمالیزه کردن نتیجه

عدد ممیز شناوری که صفر باشد قابل نرمالیزه شدن نیست. اگر این عدد طی محاسبات مورد استفاده قرار گیرد، نتیجه نیز ممکن است صفر شود. بجای تست صفر حین نرمالیزه کردن، در ابتدای کار صفر را چک می‌کنیم و در صورت لزوم پردازش را خاتمه می‌دهیم. هم ردیف کردن مانتیس‌ها باید قبل از اجرای اعمال جمع و تفریق باشد. پس از جمع یا تفریق مانتیس‌ها، نتیجه ممکن است غیر نرمالیزه باشد. با بکارگیری رویه نرمالیزه کردن، نتیجه قبل از انتقال به حافظه نرمالیزه می‌شود.

فلوچارت جمع یا تفریق دو عدد دودویی ممیز شناور در شکل ۱۵-۱۰ نشان داده شده است. اگر BR برابر صفر باشد، عمل متوقف می‌شود، و مقدار موجود در AC نتیجه عمل خواهد بود. اگر AC صفر باشد، محتوای BR را به AC منتقل می‌کنیم و اگر قرار باشد اعداد از هم تفریق شوند علامت آنرا متمم می‌کنیم. اگر هیچیک از اعداد صفر نباشند اقدام به هم ردیف کردن مانتیس‌ها می‌نمائیم.

مقایسه گر مقدار که به نماهای a و b مربوط است سه خروجی تولید می‌کند تا نسبت اندازه‌ها را مشخص نماید. اگر دو نما مساوی باشند، اجرای عملیات حسابی صورت می‌گیرد. اگر نماها مساوی نباشند مانتیسی که نمای کوچکتری دارد به راست شیفت داده شده و نمای آن افزایش می‌یابد. این روند تا مساوی شدن نماها ادامه می‌یابد.

جمع یا تفریق مانتیس‌ها شبیه الگوریتم جمع و تفریق اعداد ممیز ثابت ارائه شده در شکل ۲-۱۰ است. بخش اندازه بسته به عمل مورد نظر و علامت‌های دو مانتیس جمع یا تفریق می‌شوند. اگر در جمع



شکل ۱۰-۱۵ جمع و تفریق اعداد ممیز شناور

مانتیس‌ها سرریز رخ دهد، به فلیپ فلاپ E منتقل می‌شود. اگر E برابر 1 باشد، بیت به A_1 منتقل می‌گردد و تمام بیت‌های دیگر A به راست شیفت می‌یابند. برای حفظ عدد صحیح، نما باید افزایش یابد. در این حالت هیچ فروریزی صورت نمی‌گیرد زیرا مانتیس اولیه که در حین نرمالیزه کردن شیفت داده نشده است از قبل نرمالیزه بوده است.

اگر اندازه‌ها از هم تفریق شوند، نتیجه ممکن است صفر یا فروریز داشته باشد. اگر مانتیس صفر باشد. کل عدد ممیز شناور در AC صفر می‌شود. در غیراینصورت مانتیس باید حداقل دارای یک بیت 1 باشد. اگر با ارزشترین بیت در A_1 برابر 0 باشد مانتیس دارای فروریز است. در این حال، مانتیس به چپ شیفت یافته و نما کاهش می‌یابد. بیت A_1 مجدداً چک شده و روند تا 1 شدن A_1 ادامه می‌یابد. وقتی که $A_1=1$ باشد، مانتیس نرمالیزه شده و عملیات خاتمه یافته است.

ضرب

ضرب دو عدد ممیز شناور با ضرب مانتیس‌ها و جمع نماها انجام می‌شود. در این عمل مقایسه نماها و هم ردیف کردن مانتیس‌ها لزومی ندارد. ضرب مانتیس‌ها مشابه ضرب اعداد ممیز ثابت است و حاصلضرب با دقتی دو برابر حاصل می‌شود. پاسخ حاصل با دقت مضاعف^۱ در ممیز ثابت برای افزایش دقت حاصلضرب بکار می‌رود. در نمایش ممیز شناور، محدوده دقت معمولی^۲ همراه با نما به اندازه کافی دقت دارد و بنابراین اعداد بصورت غیرمضاعف یا معمولی نگهداری می‌شوند. لذا نیمه با ارزشتر حاصلضرب مانتیس‌ها و نماها همراه با هم برداشته شده و حاصلضربی با دقت معمولی را برای ممیز شناور بوجود می‌آورد.

الگوریتم ضرب به چهار بخش زیر تقسیم می‌شود

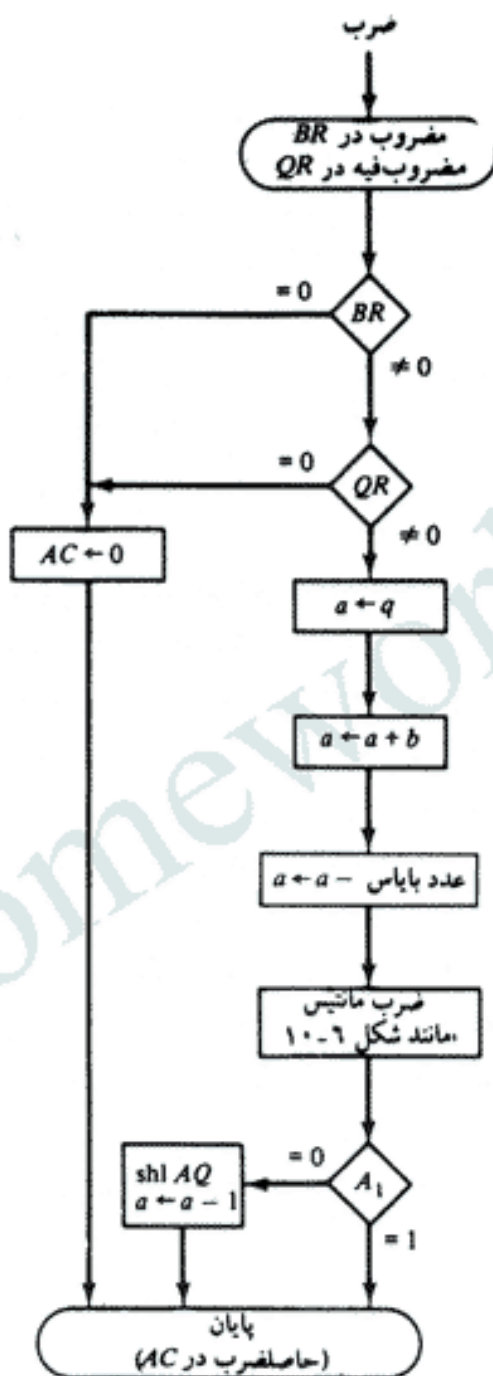
- ۱- چک کردن صفر
- ۲- جمع کردن نماها
- ۳- ضرب مانتیس‌ها
- ۴- نرمالیزه کردن حاصلضرب

مراحل ۲ و ۳ می‌توانند بطور همزمان انجام شوند مشروط بر این که برای مانتیس‌ها و نماها، جمع‌کننده‌های جداگانه‌ای وجود داشته باشند.

فلوچارت برای ضرب ممیز شناور در شکل ۱۶-۱۰ نشان داده است. دو عملوند برای تعیین اینکه آیا حاوی صفرند یا خیر چک می‌شوند. اگر هر یک از عملوندها برابر صفر باشند، حاصلضرب در AC برابر صفر می‌شود و عمل خاتمه می‌یابد. اما اگر هیچیک از عملوندها صفر نباشد، روند با جمع نماها ادامه می‌یابد.

1- Double - Precision

2- Single - Precision



شکل ۱۰-۱۶ ضرب اعداد ممیز شناور

نمای مضروب فیه در q است و جمع کننده a و b را جمع می‌کند. لازم است نما را از q به a انتقال دهیم و سپس دو نما را با هم جمع کنیم، و مجموع را به a منتقل نمائیم. چون هر دو نما بعلت جمع با یک مقدار ثابت بایاس شده‌اند، نمای مجموع دو برابر بایاس خواهد بود. نمای بایاس شده صحیح پس از تفریق عدد بایاس از مجموع بدست می‌آید.

ضرب مانتیس‌ها مشابه حالت ممیز ثابت انجام می‌شود و حاصلضرب در A و Q قرار می‌گیرد. در اینجا سرریز حین ضرب نمی‌تواند رخ دهد، لذا نیازی به چک کردن آن نیست.

حاصلضرب ممکن است فروریز داشته باشد، بنابراین با ارزشترین بیت A چک می‌شود. اگر این بیت 1 باشد ضرب قبلا نرمالیزه شده است. اگر 0 باشد مانتیس موجود در AQ به چپ شیفت پیدا کرده و نما کاهش داده می‌شود. توجه کنید که تنها یکبار شیفت برای نرمالیزه کردن کافی است. کوچکترین عملوند نرمالیزه کردن 0.1 است، لذا کوچکترین حاصلضرب ممکن 0.01 می‌باشد. بنابراین، تنها یک صفر ممکن است در انتها الیه سمت چپ وجود داشته باشد.

هرچند نیمه کم ارزشتر مانتیس در Q است، ولی ما آنرا برای ضرب ممیز شناور بکار نمی‌بریم و تنها مقدار داخل AC بعنوان حاصلضرب در نظر گرفته می‌شود.

تقسیم

در تقسیم ممیز شناور نماها تفریق و مانتیس‌ها تقسیم می‌شوند. تقسیم مانتیس مشابه ممیز ثابت است جز اینکه مقسوم مانتیسی با دقت معمولی دارد و در AC قرار داده می‌شود. بخاطر بسپارید که مانتیس مقسوم کسری است و نه عدد صحیح. برای نمایش، مقسومی با دقت معمولی باید در ثبات Q قرار گیرد و ثبات A پاک شود. صفرهای داخل A در سمت چپ ممیز دودویی قرار داشته و اهمیتی ندارند. در نمایش کسری، مقسوم با دقت معمولی در ثبات A قرار می‌گیرد و ثبات Q پاک می‌شود. صفرهای موجود در Q در سمت راست ممیز دودویی بوده و اهمیتی ندارند.

تست سرریز تقسیم همانند نمایش ممیز ثابت است. با این وجود، سرریز تقسیم در ممیز شناور مشکلی ایجاد نمی‌کند. اگر مقسوم بزرگتر یا مساوی مقسوم علیه باشد، کسر مقسوم به راست شیفت داده می‌شود و نمای آن یک واحد افزایش می‌یابد. برای عملوندهای نرمالیزه شده، این عمل برای تضمین عدم وقوع سرریز تقسیم کافی است. عمل فوق هم ردیف کردن مقسوم¹ خوانده می‌شود.

تقسیم دو عدد ممیز شناور نرمالیزه شده همواره خارج قسمت نرمالیزه شده ای را بدست می‌دهد بشرطی که هم ردیف کردن مقسوم قبل از تقسیم انجام شده باشد. بنابراین برخلاف سایر اعمال، خارج قسمت حاصل از تقسیم نیازی به نرمالیزه شدن ندارد.

الگوریتم تقسیم می‌تواند به پنج بخش زیر تقسیم شود.

۱- چک کردن برای وجود صفر

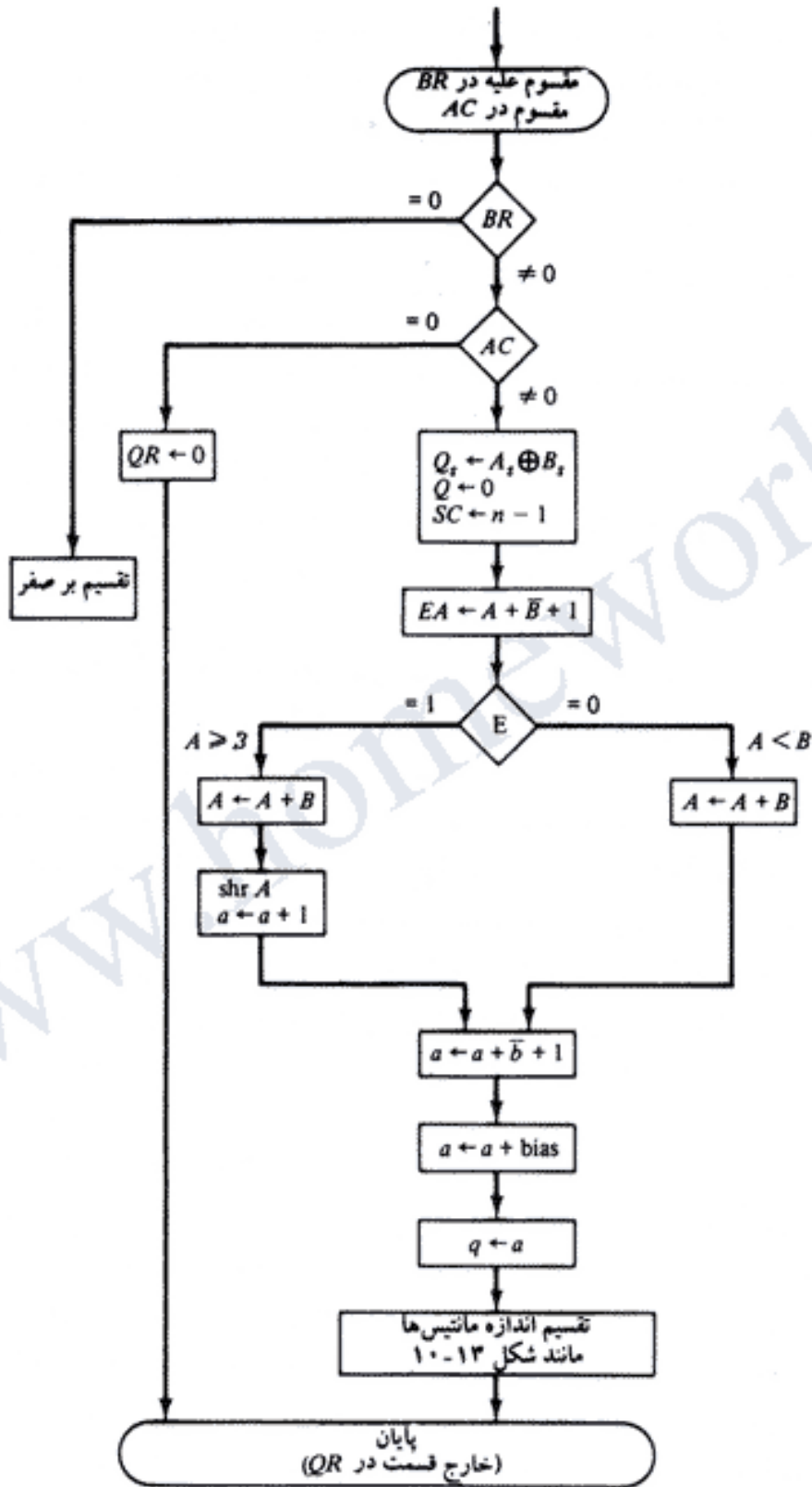
۲- مقدار دهی اولیه به ثباتها و تعیین علامت

۳- هم ردیف کردن مقسوم

۴- تفریق نمادها

۵- تقسیم مانتیس‌ها

فلوچارت برای تقسیم ممیز شناور در شکل ۱۷-۱۰ نشان داده شده است. دو عملوند برای وجود



شکل ۱۰-۱۷ تقسیم اعداد ممیز شناور

صفر چک شده‌اند. اگر مقسوم علیه صفر باشد نشاندهنده اقدام به تقسیم بر صفر است، که عملی غیر مجاز است. عملیات با اعلام خطایی متوقف می‌شود. راه دیگر این است که خارج قسمت واقع در QR را با مثبت‌ترین عدد ممکن (اگر مقسوم مثبت باشد) یا منفی‌ترین عدد ممکن (اگر مقسوم منفی) برابر کنیم. اگر مقسوم موجود در AC صفر باشد خارج قسمت موجود در QR صفر می‌شود و عمل خاتمه می‌یابد. اگر عملوندها صفر نباشند، در مرحله بعدی علامت خارج قسمت را تعیین و آن را در Q ذخیره می‌کنیم. علامت مقسوم در A به تغییر رها می‌شود تا علامت باقیمانده نیز باشد. ثبات Q پاک شده و توالی شمار SC با تعداد بیت‌های خارج قسمت بار می‌شود.

هم ردیف کردن مقسوم مشابه چک کردن سرریز تقسیم در تقسیم ممیز ثابت است. هم ردیف کردن صحیح مقسوم هنگامی است که مقسوم کسر کوچکی از مقسوم علیه باشد. این دو کسر توسط تفریق مقایسه می‌شوند. رقم نقلی E نسبت اندازه‌های آنها را معین می‌کند. کسر مقسوم با جمع کردن با مقسوم علیه به مقدار اولیه‌اش بازگردانده می‌شود. اگر $A \geq B$ باشد لازم است A یکبار به راست شیفت داده شود و نمای مقسوم افزایش یابد. چون هر دو عملوند نرمالیزه هستند، با هم ردیف کردن $A < B$ تضمین خواهد شد.

در عمل بعدی مقسوم علیه از نمای مقسوم تفریق می‌شود. چون هر دو نما در ابتدا بایاس شده بودند، با عمل تفریق تفاضل بایاس نشده آنها بدست می‌آید. سپس عدد بایاس جمع شده و نتیجه به q منتقل می‌گردد زیرا خارج قسمت در QR تشکیل می‌شود.

اندازه مانتیس‌ها همچون در حالت ممیز ثابت تقسیم می‌شوند. پس از عمل، خارج قسمت مانتیس‌ها در Q و باقیمانده در A قرار خواهد گرفت. خارج قسمت ممیز شناور قبلاً نرمالیزه شده و در QR قرار دارد. نمای باقیمانده باید مانند نمای مقسوم باشد. ممیز دودویی برای مانتیس باقیمانده $(n-1)$ مکان در سمت چپ A_1 قرار دارد. می‌توان باقیمانده را به کسر نرمالیزه شده با تفریق $n-1$ از نمای مقسوم و شیفت و کاهش تا رسیدن به 1 در A_1 تبدیل کرد. این عمل در فلوچارت نشان داده نشده است و بعنوان تمرین واگذار می‌شود.

۶-۱۰ واحد حساب دهمی

کاربر یک کامپیوتر داده‌ها را بصورت اعداد دهمی فراهم نموده و نتایج را هم بصورت دهمی دریافت می‌نماید. یک CPU با یک واحد حساب و منطق می‌تواند ریزاعمال را با اعداد دودویی انجام دهد. برای اجرای عملیات حسابی با داده‌های دهمی، لازم است داده‌های دهمی ورودی به اعداد دودویی تبدیل شوند، تا محاسبات با اعداد دودویی انجام شده، و نتایج به دهمی تبدیل شوند. این روش ممکن است در کاربردهایی که محاسبات زیادی دارند ولی مقدار داده‌های ورودی و خروجی آنها نسبتاً کم است، کاراً باشد. در مواردی که حجم داده‌های ورودی و خروجی زیاد است ولی محاسبات نسبتاً کم می‌باشد، بهتر است محاسبات درونی را مستقیماً با اعداد دهمی انجام دهیم. کامپیوترهایی که

قادرند محاسبات دهدی را اجرا نمایند باید داده‌های دهدی را به فرم کدهای دودویی ذخیره کنند. اعداد دهدی سپس به واحد محاسبات دهدی که قادر است ریزعمل‌های حساب دهدی را اجرا نماید اعمال می‌شود.

ماشین‌های حساب الکترونیکی بلااستثناء دارای واحد محاسبات دهدی داخلی می‌باشند زیرا حجم ورودی و خروجی آنها زیاد است. به نظر نمی‌رسد که برای تبدیل اعداد ورودی از طریق صفحه کلید به دودویی و بالعکس و سپس تبدیل مجدد نتایجی که باید نمایش داده شود به دهدی، دلیلی وجود داشته باشد. چون این روند نیاز به مدار خاصی داشته و زمان اجرای بیشتری هم نیاز دارد. اغلب کامپیوترها برای محاسبات حسابی با هر دو نوع داده‌های دودویی و دهدی دارای سخت افزار خاصی هستند. کاربرها بکمک دستورات برنامه ریزی شده می‌توانند خواسته خود، مبنی بر محاسبات دودویی یا دهدی توسط کامپیوتر را مشخص نمایند.

یک واحد محاسبات دهدی مدار دیجیتالی است که ریزعملیات دهدی را انجام می‌دهد. این مدار می‌تواند اعداد دهدی را جمع یا تفریق نماید، که این عمل اخیر معمولاً بکمک متمم 9 و یا 10 مفروق انجام می‌شود؛ مدار اعداد کد شده به دهدی را دریافت و نتایج را با همان کد دودویی تولید می‌نماید. یک واحد محاسبات دهدی یک مرحله‌ای، از نه متغیر ورودی دودویی تشکیل شده و پنج متغیر خروجی دودویی نیز دارد، چون حداقل چهار بیت برای نمایش هر رقم کد شده به دهدی لازم است. هر مرحله باید چهار ورودی برای رقم مضاعف، چهار رقم برای مضاف الیه، و یک رقم نقلی ورودی داشته باشد. خروجی‌ها شامل چهارپایانه برای رقم حاصل جمع و یک رقم نقلی خروجی است. البته، انواع متفاوتی از مدارات امکان پذیر است که به کد مورد استفاده برای نمایش دهدی بستگی دارد.

جمع کننده BCD

جمع حسابی دو رقم دهدی به شکل BCD، همراه با رقم نقلی احتمالی از طبقه قبلی را در نظر بگیرید. چون هر رقم ورودی از 9 تجاوز نمی‌کند، حاصل جمع خروجی $9+9+1=19$ تجاوز نخواهد کرد، که عدد 1 در جمع، رقم نقلی ورودی است. فرض کنید که دو رقم BCD را به یک جمع کننده دودویی چهاربیتی اعمال کنیم. جمع کننده حاصل جمع را بصورت دودویی تشکیل می‌دهد و نتیجه حاصل بین 0 تا 19 خواهد بود. این اعداد دودویی در جدول ۴-۱۰ لیست شده و با سمبل‌های K و Z_1, Z_2, Z_4, Z_8 نام گذاری شده‌اند. K، رقم نقلی و اندیس‌های زیر حروف Z نشان دهنده وزن‌های 8 و 4 و 2 و 1 بوده و به چهاربیت در کد BCD اختصاص دارند. اولین ستون در جدول حاصل جمع‌های دودویی را آنچنانکه در خروجی‌های یک جمع کننده دودویی چهاربیتی ظاهر می‌شوند نشان می‌دهد. خروجی حاصل جمع دو عدد دهدی بایستی بصورت BCD نشان داده شود. مسئله در اینجا یافتن قاعده ساده‌ای است که بوسیله آن عدد دودویی در ستون اول به رقم BCD صحیحی که در ستون دوم نمایش داده شده تبدیل شود.

جدول ۴-۱۰ بدست آوردن جمع کننده BCD

K	مجموع دودویی				مجموع BCD					دهمی
	Z ₈	Z ₄	Z ₂	Z ₁	C	S ₈	S ₄	S ₂	S ₁	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

با بررسی محتوای جدول، واضح است که وقتی حاصل جمع دودویی برابر یا کمتر از 1001 باشد، عدد BCD متناظر، مانند خود آن عدد است و لذا تبدیلی لازم نیست. وقتی که حاصل جمع بزرگتر از 1001 باشد نمایش نامعتبری از BCD را خواهیم داشت. جمع عدد دودویی 6 (0110) با حاصل جمع دودویی، نمایش BCD را تصحیح نموده و یک رقم نقلی خروجی لازم را نیز تولید می کند.

یکی از روش های جمع اعداد دهمی در BCD استفاده از یک جمع کننده دودویی چهاربیتی و اجرای عمل حسابی بصورت هر بار یک رقم خواهد بود. ابتدا جفت ارقام کم ارزشتر BCD جمع شده و یک حاصل جمع دودویی تولید می نماید. اگر نتیجه مساوی یا بزرگتر از 1010 باشد، با جمع 0110 با حاصل جمع اصلاح می شود. این عمل دوم بطور خودکار یک رقم نقلی را برای جفت رقم با ارزش بعدی تولید می کند. جفت ارقام با ارزشتر بعدی، همراه با نقلی ورودی با هم جمع شده و مجموع مربوطه را تولید می نمایند. اگر این نتیجه مساوی یا بزرگتر از 1010 می باشد، با جمع آن با 0110 تصحیح می گردد. رویه تا جایی که کلیه ارقام جمع شوند، ادامه می یابد.

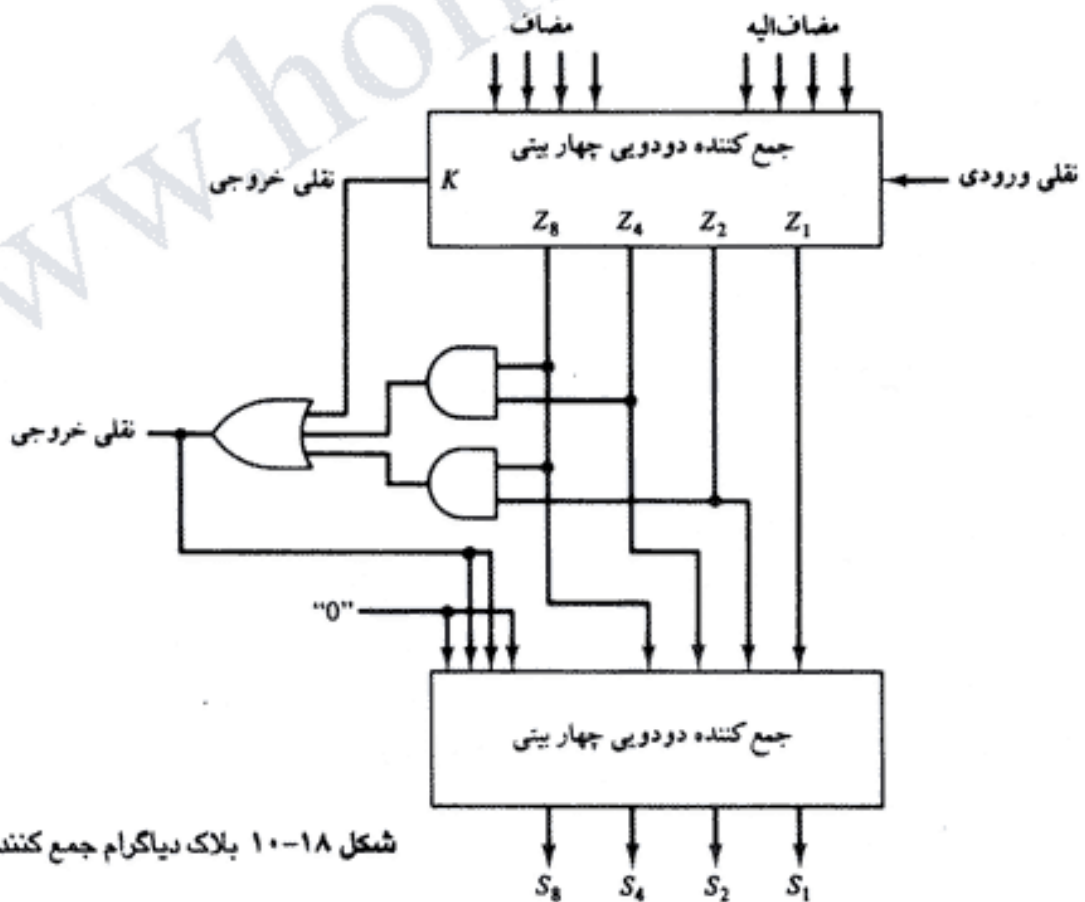
مداری که که ضرورت تصحیح را مشخص می کند از واردهای جدول بدست می آید. واضح است

که هرگاه مجموع دودویی دارای نقلی خروجی مساوی 1 باشد تصحیح لازم است. شش ترکیب دیگر از 1010 تا 1111 که اصلاح نیاز دارند دارای 1 در مکان Z_8 هستند، Z_4 یا Z_2 هم باید 1 داشته باشند، شرایط تصحیح و رقم نقلی خروجی با تابع بولی زیر بیان می شود.

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

هنگامی که $C=1$ باشد، لازم است به مجموع دودویی مقدار 0110 را اضافه کنیم و یک رقم نقلی برای طبقه بندی تولید نمائیم.

یک جمع کننده BCD مداری است که دو رقم BCD را بصورت موازی جمع کرده و ارقام مجموع را نیز بفرم BCD تولید می کند. یک جمع کننده BCD باید دارای مدار تصحیح کننده در ساختمان داخلی خود باشد. برای جمع 0110 با مجموع دودویی، از یک جمع کننده دودویی چهار بیتی دومی که در شکل ۱۰-۱۸ نشان داده شده است استفاده می کنیم. دو رقم دهدهی، همراه با رقم نقلی ورودی، ابتدا در جمع کننده فوقانی با هم جمع شده و مجموع دودویی را تولید می کنند. مادامی که رقم نقلی خروجی برابر 0 باشد، چیزی به مجموع دودویی اضافه نمی شود. اگر رقم نقلی خروجی 1 باشد، عدد دودویی 0110 به مجموع دودویی از طریق جمع کننده تحتانی اضافه می شود. رقم نقلی خروجی تولید شده بوسیله جمع کننده دودویی تحتانی می تواند چشم پوشی شود، زیرا اطلاعاتی را که قبلاً در خروجی رقم



شکل ۱۰-۱۸ بلاک دیاگرام جمع کننده BCD

نقلی موجود است تولید نموده است.

یک جمع کننده موازی دهدهی که n رقم دهدهی را جمع می‌کند به n طبقه جمع کننده BCD نیاز دارد که در آن هر خروجی نقلی از یک طبقه به ورودی نقلی طبقه با ارزشتر بعدی متصل است. برای دستیابی به تاخیرهای انتشار کوتاهتر، جمع کننده‌های BCD حاوی مدارات لازم برای تولید نقلی پیش بینی شده می‌باشند. بعلاوه، مدار جمع کننده برای اجرای عمل تصحیح نیازی به هر چهار تمام جمع کننده ندارد و لذا مدار می‌تواند بهینه شود.

تفریق BCD

تفریق مستقیم دو عدد دهدهی به یک مدار تفریق‌گر که از برخی لحاظ با جمع کننده BCD متفاوت است نیاز دارد. اگر تفریق را بکمک متمم 9 یا 10 مفروق و جمع آن با مفروق منه انجام دهیم اقتصادی تر خواهد بود. چون BCD یک کد خود متمم¹ نیست، متمم 9 آن را نمی‌توان با متمم کردن هر بیت در کد بدست آورد. این متمم را با تشکیل مداری که هر رقم BCD را از 9 تفریق می‌کند باید بدست آورد. متمم 9 یک رقم دهدهی که بصورت BCD نشان داده شده باشد را می‌توان با متمم کردن بیت‌ها در نمایش کد شده رقم بدست آورد بشرط اینکه تصحیحی صورت گیرد. دو روش تصحیح وجود دارد. در روش اول، عدد دودویی 1010 (10 دهدهی) به هر رقم متمم اضافه شده و رقم نقلی بعد از هر جمع چشم پوشی می‌شود. در روش دوم، عدد دودویی 0110 (6 دهدهی) قبل از متمم شدن رقم اضافه می‌شود. به عنوان یک مثال عددی، برای بدست آوردن متمم 9 عدد 0111 (7 دهدهی) BCD بدین ترتیب محاسبه می‌شود که ابتدا هر بیت متمم می‌شود تا 1000 بدست آید. با جمع 1010 با آن و چشم پوشی از رقم نقلی عدد 0010 (2 دهدهی) را خواهیم داشت. با استفاده از روش دوم، ما عدد 0110 را به 0111 اضافه می‌کنیم تا 1101 بدست آید. با متمم کردن هر بیت نتیجه مورد نظر 0010 بدست می‌آید. متمم کردن عدد چهاربیتی N مشابه تفریق آن از 1111 (15 دهدهی) می‌باشد. با اضافه کردن معادل دودویی عدد 10 به آن داریم $16 - N + 10 = 9 - N + 15$. از طرفی 16 بمعنی رقم نقلی بوده و نادیده گرفته می‌شود، بنابراین نتیجه $9 - N$ خواهد شد. با افزودن معادل دودویی عدد دهدهی 16 و متمم کردن آن نیز داریم $9 - N = 15 - (N + 6)$.

متمم 9 عدد BCD از طریق یک مدار ترکیبی نیز بدست می‌آید. وقتی که این مدار به یک جمع کننده BCD متصل گردد، یک جمع یا تفریق کننده BCD بدست می‌آید. اجازه بدهید تا رقم مفروق (یا مضاف) با چهار متغیر دودویی B_1, B_2, B_4, B_8 مشخص شده باشد. همچنین M بیت مد (شیوه) باشد که جمع یا تفریق را کنترل می‌کند. هنگامی که $M=0$ باشد، دو رقم جمع می‌شوند؛ اگر $M=1$ باشد، ارقام از یکدیگر تفریق می‌گردند. همچنین، فرض کنید متغیرهای دودویی x_1, x_2, x_4, x_8 خروجی‌های مدار متمم ساز (متمم 9) باشند. با بررسی جدول درستی این مدار مشاهده می‌شود (مسئله ۳۰-۱۰) که B_1 همواره باید متمم شود؛ B_2 همیشه در متمم 9 مثل رقم اصلی است؛ x_4 برابر 1 است اگر OR

1- Self Complement

انحصاری B_2 و B_4 برابر 1 باشند؛ x_8 هم، اگر $B_8 B_4 B_2 = 000$ باشد، برابر 1 می باشد. توابع بولی تولیدکننده متمم 9 عبارتند از:

$$x_1 = B_1 M' + B'_1 M$$

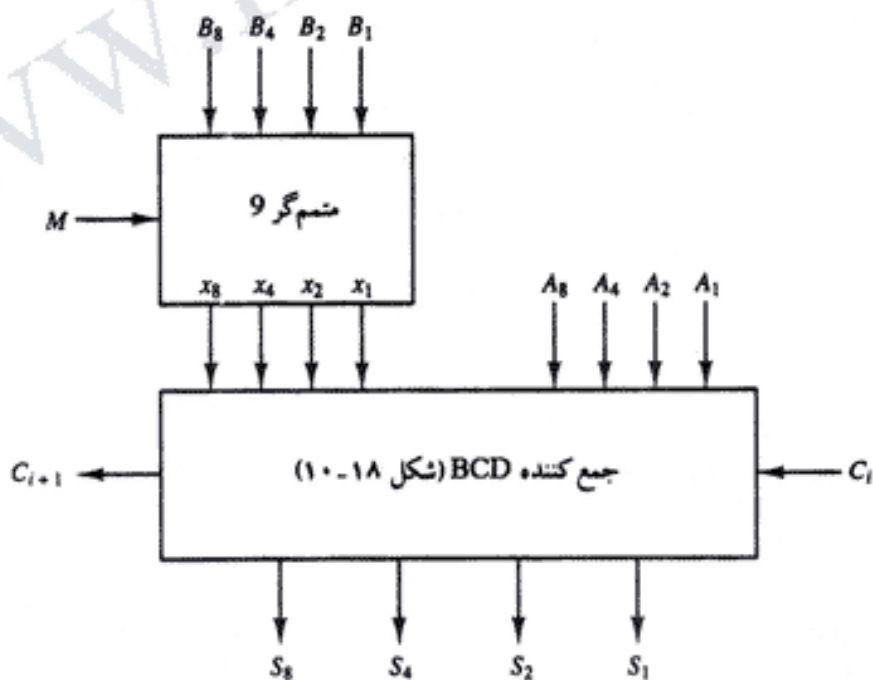
$$x_2 = B_2$$

$$x_4 = B_4 M' + (B'_4 B_2 + B_4 B'_2) M$$

$$x_8 = B_8 M' + B'_8 B'_4 M$$

باتوجه به معادلات فوق در می یابیم که وقتی $M=0$ است، $x=B$ می باشد. هنگامی که $M=1$ باشد خروجی های x متمم B را تولید می کنند.

یک طبقه از مدار محاسبات دهمی که قادر است دو رقم و BCD را جمع و یا تفریق کند در شکل ۱۰-۱۹ نشان داده شده است. این مدار متشکل از یک جمع کننده را BCD یک متمم ساز است. ورودی M کار مدار را کنترل می کند. اگر $M=0$ باشد خروجی های S مجموع A و B را تولید می کنند. هنگامی که $M=1$ باشد خروجی های S مجموع A بعلاوه متمم 9 عدد B را تولید می کنند. برای اعداد n رقمی به n طبقه از این نوع احتیاج است. خروجی نقلی C_{i+1} از یک طبقه باید به ورودی C_i طبقه با ارزشتر متصل شود. بهترین راه تفریق دو عدد دهمی این است که $M=1$ باشد و ورودی C_1 در اولین طبقه را هم به 1 متصل کنیم. خروجی ها مجموع A بعلاوه متمم 10 عدد B را تولید می کنند، که معادل با عمل تفریق است بشرط اینکه رقم نقلی خروجی آخرین طبقه نادیده گرفته شود.



شکل ۱۰-۱۹ یک طبقه از یک واحد حساب دهمی

۷-۱۰ اعمال حسابی دهدهی

الگوریتم‌های اعمال حسابی با داده‌های مشابه الگوریتم‌های همان اعمال با داده‌های دودویی است. در واقع بجز کمی تصحیح در الگوریتم ضرب و تقسیم، فلوچارت‌های یکسانی را برای دو نوع داده می‌توان بکاربرد با این شرط که سمبل ریز عمل‌ها را بطور صحیح تفسیر کنیم. اعداد دهدهی با نمایش BCD گروه‌های چهاربیتی در ثبات‌های کامپیوتر ذخیره می‌شوند. هر گروه چهاربیتی یک رقم دهدهی را نمایش می‌دهد و باید هنگام اجرای ریزعمل‌های دهدهی بصورت واحد در نظر گرفته شود.

برای سهولت، ما سمبل‌های یکسانی را برای ریزعمل حسابی دودویی و دهدهی بکار خواهیم برد ولی بصورت متفاوتی آنها را تعبیر خواهیم کرد. همان طور که در جدول ۵-۱۰ دیده می‌شود، یک خط روی سمبل حرف بمعنی متمم 9 یک عدد دهدهی ذخیره شده در ثبات است. جمع 1 با متمم 9 متمم 10 را تولید می‌کند. بنابراین برای اعداد دهدهی، سمبل $A \leftarrow A + \bar{B} + 1$ بمعنی انتقال مجموع دهدهی تشکیل شده از جمع مقدار اولیه A با متمم 10 عدد B است. استفاده از سمبل‌های یکسان برای متمم 9 و متمم 1، اگر هر دو نوع داده در یک سیستم بکار رود موجب سردرگمی می‌گردد. در این حالت بهتر خواهد بود سمبل‌های متفاوتی برای متمم 9 بکار برده شود. اگر یک نوع داده مورد استفاده قرار گیرد نماد مورد استفاده متعلق به همان نوع داده است.

افزایش یا کاهش ثبات برای اعداد دودویی و دهدهی، بجز تعداد حالات مجاز برای ثبات یکسان است. افزایش یک شمارنده دودویی موجب طی کردن 16 حالت از 0000 تا 1111 می‌گردد. یک شمارنده دهدهی 10 حالت از 0000 تا 1001 را می‌شمارد و دوباره به 0000 باز می‌گردد، زیرا عدد 9 آخرین عدد ممکن در شمارش دهدهی است. بطور مشابه، کاهش یک شمارنده دودویی رشته 1111 تا 0000 را طی می‌کند. شمارنده دهدهی در این فرم از 1001 تا 0000 را می‌شمارد.

در جلو شیفت به راست یا چپ دهدهی حرف d بکار رفته است تا مشخص کننده شیفت بر روی چهاربیتی باشد که ارقام دهدهی را دارا هستند. بعنوان مثال عددی، ثبات A که عدد دهدهی 7860 را بشکل BCD در خود دارد در نظر بگیرید. الگوی 12 فلیپ فلاپ بقرار زیر است.

0111 1000 0110 0000

جدول ۵-۱۰ سمبل‌های ریزعمل‌های حسابی دهدهی

نمایش سمبلیک	شرح
$A \leftarrow A + B$	جمع عددهای دهدهی و انتقال مجموع به A
\bar{B}	متمم 9 از B
$A \leftarrow A + \bar{B} + 1$	محتوای A به علاوه متمم 10 عدد B در A قرار گیرد
$Q_i \leftarrow Q_i + 1$	افزایش عدد BCD موجود در Q_i
dshr A	شیفت دهدهی ثبات A به راست
dshl A	شیفت دهدهی ثبات A به چپ

ریز عمل dshr عدد دهدهی را یک رقم به راست شیفت می دهد و لذا 0786 بدست می آید. این شیفت روی دسته های 4 بیتی انجام شده و محتوای ثبات را بصورت زیر تغییر می دهد.

0000 0111 1000 0110

جمع و تفریق

الگوریتم جمع و تفریق اعداد دودویی با نمایش مقدار علامت دار برای اعداد دهدهی با نمایش مقدار علامت دار نیز صادق است با این شرط که سبمل ها را بطور صحیحی تعبیر کنیم. بطور مشابه الگوریتم اعداد دودویی با نمایش متمم 2 علامت دار برای اعداد دهدهی متمم 10 علامت دار نیز صادق است. برای داده های دودویی از یک جمع کننده دودویی و یک متمم ساز استفاده می شود. برای داده های دهدهی از واحد محاسباتی که قادر به جمع اعداد BCD بوده و بتواند متمم 9 مفروق را طبق شکل ۱۹-۱۰ تشکیل دهد استفاده می شود.

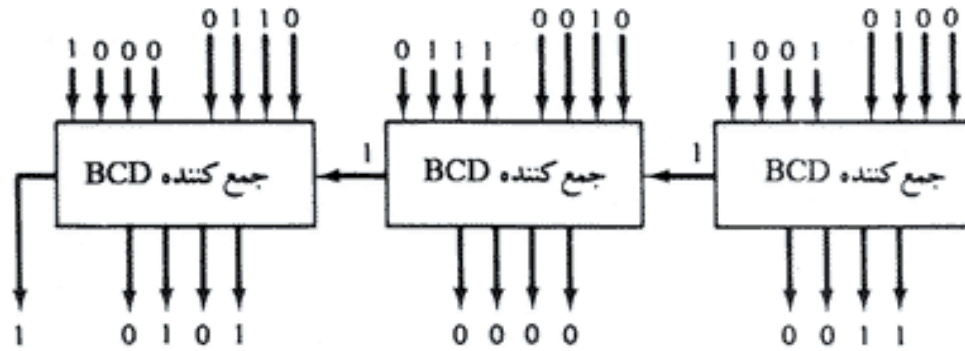
داده های دهدهی به سه طریق مختلف باهم جمع می شوند، شکل ۲۰-۱۰ روش موازی از یک واحد محاسبه دهدهی متشکل از جمع کننده های BCD بتعداد ارقام عدد استفاده می کند. مجموع بطور موازی تشکیل و فقط به یک ریز عمل نیاز دارد. در روش رقم- سری، بیت- موازی^۱، ارقام فقط به یک جمع کننده BCD بصورت سری اعمال می شوند، در حالیکه بیت های هر کد بصورت موازی انتقال می یابند. حاصل جمع با شیفت اعداد دهدهی به جمع کننده BCD، یکی پس از دیگری تشکیل می گردد. برای k رقم دهدهی، این آرایش به k ریز عمل نیاز دارد، که هر یک متعلق به یک شیفت دهدهی است. در تمام جمع کننده های سری، بیت ها یکی پس از دیگری بداخل تمام جمع کننده شیفت پیدا می کنند. مجموع دودویی حاصل از چهار بار شیفت باید تصحیح شود تا به صورت یک رقم BCD معتبر درآید. برای این تصحیح که در بخش ۶-۱۰ در مورد آن بحث شد، مجموع دودویی واریسی می شود. اگر این مجموع بزرگتر یا مساوی 1010 باشد، جمع آن با 0110 آن را تصحیح می کند و یک رقم نقلی را نیز برای جفت رقم بعدی تولید می نماید.

روش موازی سریع است ولی به تعداد زیادی جمع کننده نیاز دارد. روش رقم- سری، بیت- موازی تنها به یک جمع کننده BCD نیاز دارد که مشترکاً بوسیله کلیه ارقام مورد استفاده قرار می گیرد. این روش کندتر از روش موازی است زیرا شیفت ارقام به زمان احتیاج دارند. تمام روش های سری به حداقل امکانات نیاز دارند ولی خیلی کند عمل می کنند.

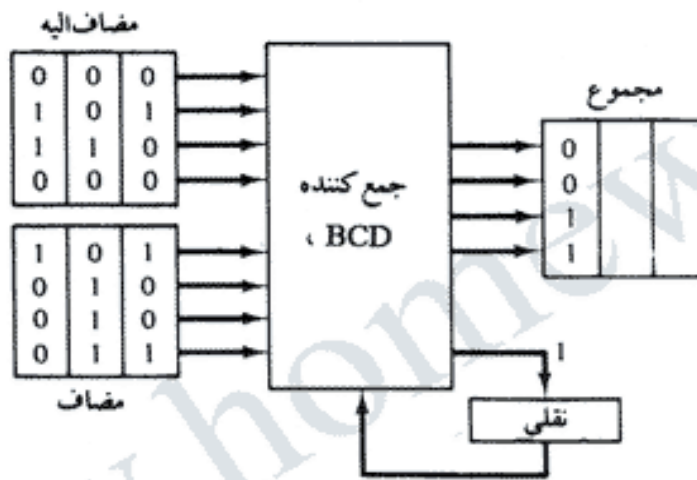
ضرب

ضرب اعداد دهدهی ممیز ثابت، بجز در تولید حاصلضرب جزئی، با روش دودویی مشابه است. در

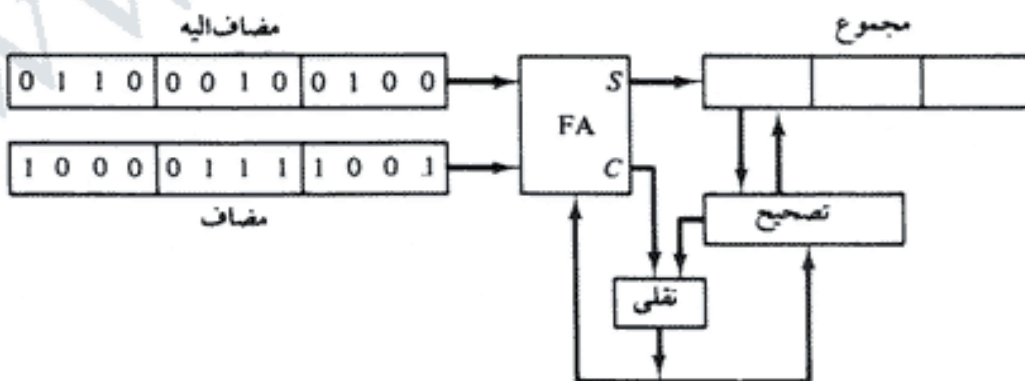
1- Digit-Serial Bit-Parallel



(الف) جمع دهدهی موازی $624+879=1503$



(ب) جمع دهدهی رقم سری، بیت-موازی



(ج) جمع دهدهی تمام سری

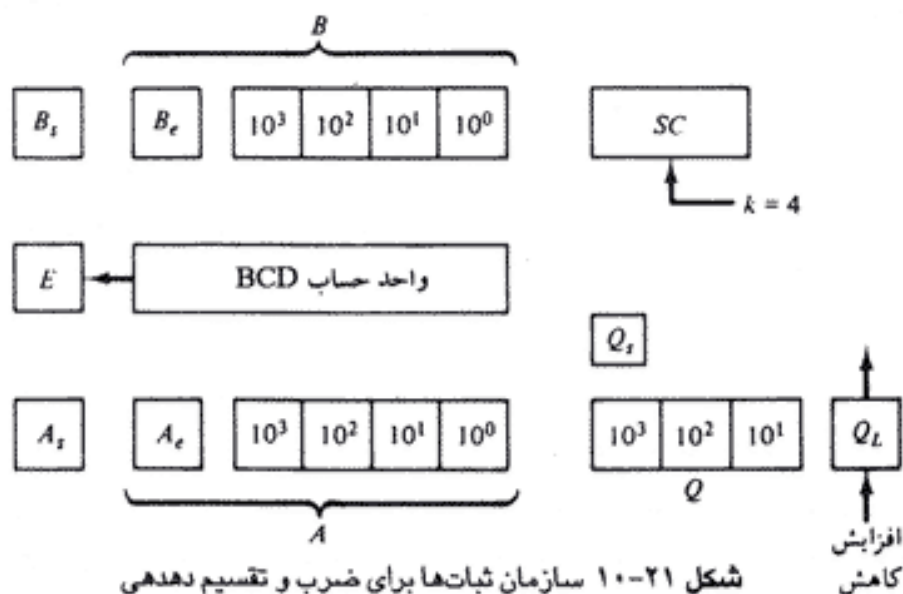
شکل ۲۰-۱۰ سه روش جمع اعداد دهدهی

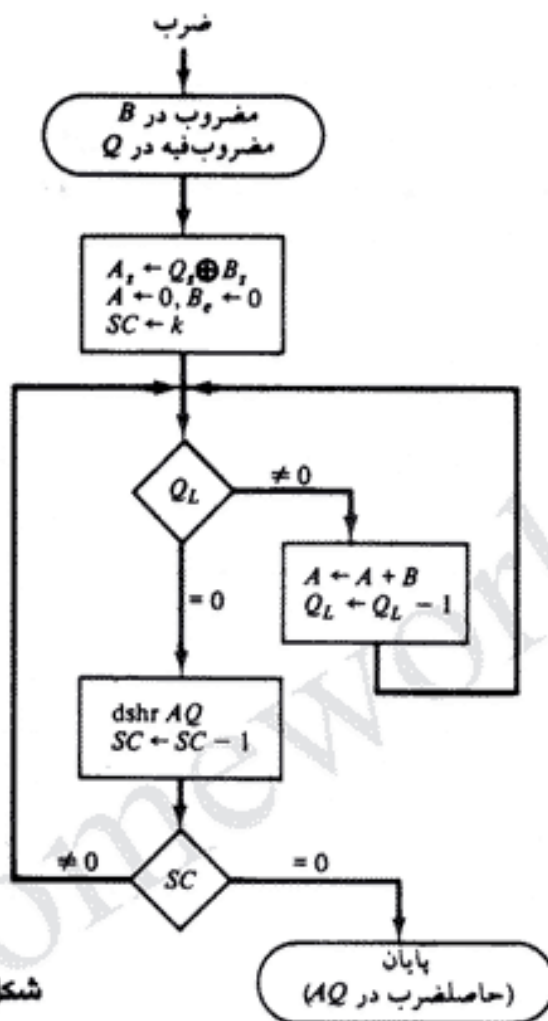
ضرب دهمی ارقام در محدوده 0 تا 9 قرار دارند، در صورتی که در ضرب دودویی فقط دو رقم 0 و 1 وجود دارد. در حالت دودویی، مضروب، بشرطی که مضروب فیه 1 باشد، با حاصلضرب جزئی جمع می شود. در حالت دهمی مضروب باید در رقم مضروب فیه ضرب شده و سپس به حاصلضرب جزئی اضافه شود. این عمل را می توان با جمع مضروب با حاصلضرب جزئی به تعداد دفعاتی برابر با مقدار رقم مضروب فیه انجام داد.

سازمان ثبات ها برای ضرب دهمی در شکل ۲۱-۱۰ نشان داده شده است. در اینجا فرض می کنیم که اعداد چهار رقمی باشند و هر رقم چهار بیت را اشغال نماید، در نتیجه هر عدد 16 بیتی خواهد بود. سه ثبات A و B و Q وجود دارند که هر یک فلیپ فلاپ علامت مربوط به خود یعنی A_e , B_e , Q_e را دارا هستند. ثبات های A و B دارای چهار بیت اضافی دیگریند که با A_e و B_e نشان داده شده و به منظور گسترش ثبات ها به میزان یک رقم اضافه شده است. واحد محاسبه BCD پنج رقم را بصورت موازی جمع و حاصل جمع را در ثبات پنج رقمی A قرار می دهد. رقم نقلی نهایی به فلیپ فلاپ E می رود. هدف از رقم A_e جای دادن سرریز احتمالی حاصل از جمع مضروب به حاصل ضرب جزئی در حین عمل ضرب است. هدف از B_e تشکیل متمم 9 مقسوم علیه هنگام تفریق آن از باقیمانده جزئی در حین عمل تقسیم است. کم ارزشترین رقم ثبات Q با Q_L مشخص می شود. این رقم را می توان افزایش یا کاهش داد.

یک عملوند Q دهمی برداشته شده از حافظه 17 بیتی است. یک بیت (علامت) به B_e مستقل می شود و اندازه عملوند در 16 بیت پائین تر B قرار می گیرد. در آغاز هر دو بیت A_e و B_e پاک می شوند. نتیجه عمل نیز 17 بیتی است و در آن از بخش A_e ثبات A استفاده نمی شود.

الگوریتم ضرب دهمی در شکل ۲۲-۱۰ دیده می شود. در آغاز، کل ثبات A و B_e پاک شده و توالی شمار SC با عددی مانند k که برابر تعداد ارقام مضروب فیه است پر می شود. کم ارزشترین رقم مضروب فیه در Q_L و آرسی می شود. اگر برابر 0 نباشد، مضروب در B با حاصلضرب جزئی در A یکبار جمع شده و





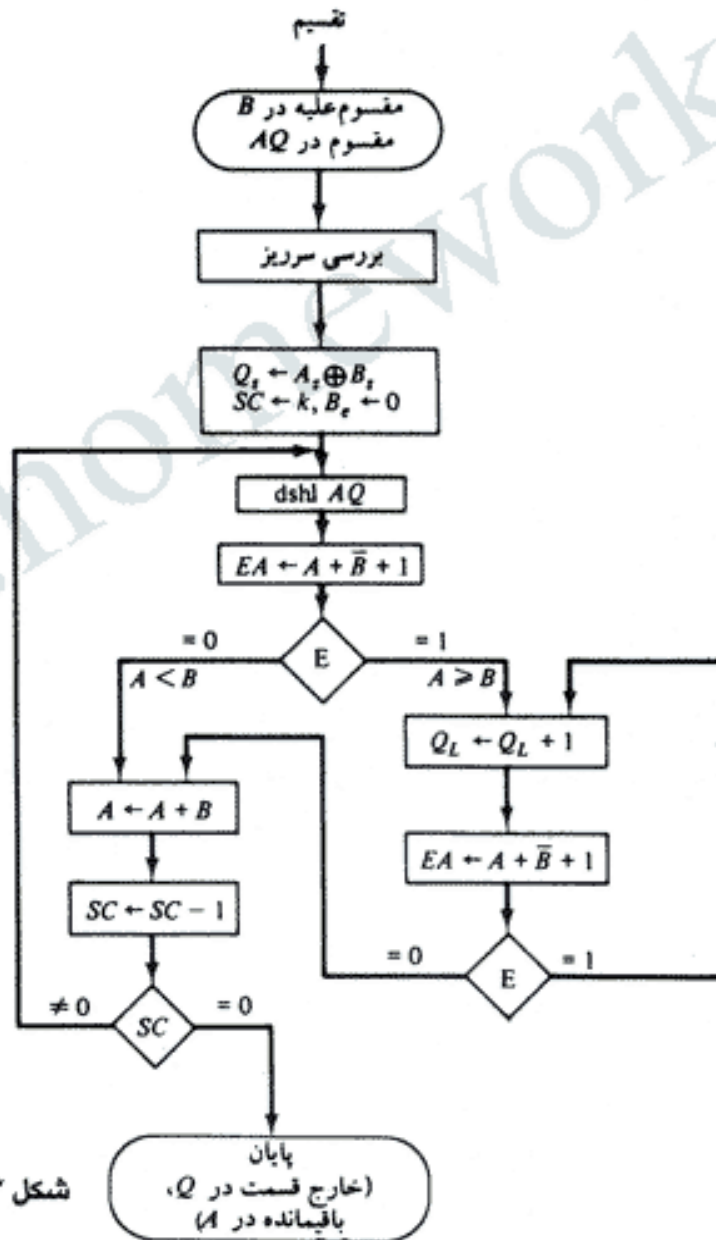
شکل ۲۲-۱۰ فلوجارت ضرب دهدهی

Q_L یک واحد کم می شود. مجدداً Q_L چک می شود و روند تا صفر شدن آن ادامه می یابد. بدین ترتیب مضروب موجود در B به تعداد دفعاتی برابر با رقم مضروب فیه با حاصلضرب جزئی جمع می شود. هر رقم سرریز موقت در A قرار گرفته و می تواند بین 0 تا 9 باشد. سپس حاصلضرب جزئی و مضروب فیه یکبار به راست شیفت داده می شود. این عمل یک 0 را در A قرار می دهد و رقم بعدی مضروب فیه به Q_L منتقل می شود. نهایتاً روند k مرتبه تکرار می شود تا حاصلضرب با طول دو برابر در AQ تشکیل شود.

تقسیم

تقسیم دهدهی مشابه تقسیم دودویی است با این تفاوت که ارقام خارج قسمت ممکن است هر یک مقادیر 0 تا 9 را داشته باشند. در روش تقسیم بازیافتی، مقسوم علیه به تعداد دفعات لازم از مقسوم یا باقیمانده جزئی تفریق می شود تا اینکه باقیمانده منفی بدست آید. سپس با جمع کردن مقسوم علیه، باقیمانده صحیح بازیابی می شود. رقم موجود در خارج قسمت منعکس کننده تعداد تفریق های انجام

شده منهای تفریقی است که موجب تفاضل منفی شده است. الگوریتم تقسیم دهدهی در شکل ۱۰-۲۳ نشان داده شده است. این الگوریتم بجز در مورد نحوه تشکیل بیت های خارج قسمت مشابه الگوریتم تقسیم داده های دودویی است. مقسوم (یا باقیمانده جزئی) به چپ شیفت می یابد و با ارزشترین رقم در A_e قرار می گیرد. سپس متمم 10 مقسوم علیه با مقسوم جمع شده و بدیت ترتیب عمل تفریق مقسوم علیه از مقسوم انجام می گیرد. چون در ابتدا پاک شده است، متمم آن 9 است. رقم نقلی در E نسبت اندازه های A و B را معین می کند. اگر $E=0$ باشد، نشان دهنده $A < B$ است. در این حالت مقسوم علیه جمع می شود تا باقیمانده جزئی بازیابی شود و Q_L



شکل ۱۰-۲۳ فلوجارت تقسیم دهدهی

در 0 باقی می ماند (درحین شیفت صفر وارد شده است). اگر $E=1$ باشد نشاننده $A \geq B$ است. رقم خارج قسمت موجود در Q_L یکبار افزایش یافته و مقسوم علیه مجدداً تفریق می شود. این روند تا رسیدن به تفاضل که با $E=0$ قابل تشخیص است ادامه می یابد. هرگاه این وضعیت رخ دهد. رقم خارج قسمت افزایش نیافته ولی مقسوم علیه جمع می شود تا باقیمانده مثبت بدست آید، در این وضعیت رقم خارج قسمت برابر با تعداد دفعاتی است که باقیمانده جزئی می تواند مقسوم علیه را در خود جای دهد. بیت های باقیمانده جزئی و خارج قسمت یکبار به چپ شیفت داده می شوند و روند k بار تکرار می شود تا k رقم خارج قسمت تشکیل گردد. سپس باقیمانده در ثبات A و خارج قسمت در ثبات Q خواهند بود. مقدار E صرف نظر می شود.

اعمال ممیز شناور

اعمال حسابی دهدهی ممیز شناور روند مشابهی با اعمال دودویی دارند. الگوریتم های بخش ۱۰-۵ می تواند برای داده های دهدهی بکار رود بشرطی که سمبل های ریز اعمال بطور صحیحی تعبیر شوند. ضرب تقسیم مانتیس ها باید با استفاده از روش های فوق صورت گیرد.

مسائل

۱۰-۱ اگر در شکل ۱۰-۱ بجای $A + \bar{B} + 1$ عمل $B + \bar{A}$ (متمم 1 عدد A بعلاوه B) را انجام دهیم متمم ساز لازم نیست. الگوریتمی به شکل فلوجارت برای جمع و تفریق اعداد دودویی ممیز ثابت با نمایش اندازه علامت دار بدست آورید که در آن اندازه ها با دو ریز عمل $A \leftarrow \bar{A}$ و $\bar{E} \leftarrow E$ $A+B$ تفریق شوند.

۱۰-۲ هر یک از مسیرهای فلوجارت شکل ۱۰-۲ را شماره گذاری کنید و سپس کل مسیری که الگوریتم هنگام محاسبه اعداد با اندازه علامت دار زیر طی می کند را مشخص نمایید. در هر مورد مقدار AVF را بنویسید. بیت متتھالیه سمت چپ در عددهای زیر نماینده بیت علامت است.

$$\text{الف) } 0\ 101101 + 0\ 011111$$

$$\text{ب) } 1\ 011111 + 1\ 101101$$

$$\text{ج) } 0\ 101101 - 0\ 011111$$

$$\text{د) } 0\ 101101 - 0\ 101101$$

$$\text{ه) } 1\ 011111 - 0\ 101101$$

۱۰-۳ اعمال حسابی زیر را با اعداد دودویی و اعداد منفی در نمایش متمم 2 علامت دار انجام دهید. برای جای دادن هر عدد همراه با علامتش از 7 بیت استفاده کنید. در هر حالت، با واری نقلی

ورودی و خروجی از بیت علامت وجود سرریز را واریسی کنید.

(الف) $(+40) + (+35)$

(ب) $(-40) + (-35)$

(ج) $(+40) - (-35)$

۱۰-۴ اعداد دودویی را با نمایش متمم 2 علامت دار در نظر بگیرید. هر عدد n بیت دارد: یک بیت برای علامت و $k=n-1$ بیت برای اندازه (یا مقدار). عددی منفی مانند $-x$ بصورت $2^k + (2^k - x)$ نمایش داده می شود که در آن اولین 2^k نشان دهنده بیت علامت و $(2^k - x)$ متمم 2 عدد x است. عدد مثبت بصورت $0+x$ نمایش داده می شود، که 0 بیت علامت و x مقدار k بیتی آن است. با استفاده از این سبب های کلی، ثابت کنید که مجموع $(\pm X) + (\pm Y)$ را می توان با جمع عددهایی شامل بیت های علامت آنها و حذف نقلی خروجی از بیت علامت تشکیل داد. به عبارت دیگر الگوریتم جمع دو عدد دودویی با نمایش متمم 2 علامت دار را ثابت کنید.

۱۰-۵ یک رویه سخت افزاری را برای آشکار سازی سرریز با مقایسه علامت مجموع و علامت های مضاف و مضاف الیه بسازید. عددها به شکل متمم 2 علامت دار هستند.

۱۰-۶ (الف) عمل $-15 = (-6) + (-9)$ را با اعداد دودویی با نمایش متمم 1 علامت دار فقط پنج بیتی برای هر عدد (با بیت علامت) انجام دهید. نشان دهید که رویه آشکار سازی سرریز با واریسی دونقلی آخر در این مورد به نتیجه منجر نمی شود.

(ب) رویه اصلاح شده ای را برای آشکار سازی یک سرریز وقتی که اعداد متمم 1 استفاده می شوند پیشنهاد کنید.

۱۰-۷ الگوریتمی بفرم فلوچارت برای جمع و تفریق دو عدد دودویی ممیز ثابت وقتی که اعداد منفی بصورت متمم 1 هستند بدست آورید.

۱۰-۸ ثابت کنید ضرب دو عدد n رقمی در پایه ۲ حاصلضربی بزرگتر از $2n$ رقم تولید نمی کند. نشان دهید که این عبارت بدان معنی است که سرریزی در عمل ضرب رخ نمی دهد.

۱۰-۹ محتوای ثبات های E, A, Q و SC (مطابق جدول ۱۰-۲) را در حین روند ضرب دو عدد دودویی 11111 (مضروب) و 10101 (مضروب فیه) نشان دهید. علامت ها در نظر گرفته نشده اند.

۱۰-۱۰ محتوای ثبات های E, A, Q و SC (مطابق شکل ۱۰-۱۲) را در حین روند تقسیم (الف) 10100011 بر 1011؛ (ب) 00001111 بر 0011 نشان دهید. (از مقسوم هشت بیتی استفاده کنید).

۱۰-۱۱ نشان دهید که جمع B پس از عمل $A + \bar{B} + 1$ مقدار اولیه A را باز می گرداند. با رقم نقلی نهایی چه باید کرد؟

۱۰-۱۲ چرا علامت باقیمانده پس از تقسیم باید با علامت مقسوم یکی باشد؟

۱۳-۱۰ یک ضرب کننده آرایه ای طراحی کنید که دو عدد چهاربیتی را ضرب کند. از گیت AND برای جمع کننده دودویی استفاده کنید.

۱۴-۱۰ روند قدم به قدم ضرب با بکارگیری الگوریتم بوت را (مثل جدول ۳-۱۰) هنگامی که اعداد دودویی زیر در هم ضرب می شوند نشان دهید. فرض کنید ثبات های پنج بیتی اعداد علامت دار را نگه می دارند. مضروب در هر دو حالت $+15$ است.

$$\text{الف) } (+13) \times (+15)$$

$$\text{ب) } (+15) \times (-13)$$

۱۵-۱۰ الگوریتمی را بفرم فلوچارت برای روش غیر بازیافتی تقسیم دودویی ممیز ثابت بدست آورید.

۱۶-۱۰ الگوریتمی برای محاسبه ریشه دوم عدد دودویی ممیز ثابت بدست آورید.

۱۷-۱۰ یک عدد دودویی ممیز شناور هفت بیت برای نمای بایاس شده اش دارد. مقدار ثابت بایاس 64 است.

الف) لیست بایاس شده تمام نماها را از -64 تا $+63$ بدست آورید.

ب) نشان دهید که یک مقایسه گر مقدار هفت بیت می تواند برای مقایسه اندازه نسبی دو نما بکار رود.

ج) نشان دهید که پس از جمع دو نمای بایاس شده لازم است 64 را تفریق کنیم تا مجموع نماها به شکل بایاس شده بدست آید. چگونه می توان 64 را با جمع کردن مقدار متمم 2 آن تفریق کرد؟
د) نشان دهید که پس از تفریق دو نمای بایاس شده لازم است 64 را اضافه کنیم تا تفاضل بایاس شده نماها را داشته باشیم

۱۸-۱۰ الگوریتمی بشکل فلوچارت برای مقایسه دو عدد دودویی علامت دار وقتی که اعداد منفی به

فرم متمم 2 علامت دار می باشند بدست آورید

الف) با استفاده از عمل تفریق با عددهای متمم 2 علامت دار

ب) با مرور و مقایسه جفت بیت ها از چپ به راست

۱۹-۱۰ مسئله ۱۸-۱۰ را برای اعداد دودویی مقدار علامت دار تکرار کنید.

۲۰-۱۰ فرض کنید n تعداد بیت های مانتیس در یک عدد دودویی ممیز شناور است. هنگامی که

مانتیس ها در حین جمع یا تفریق هم ردیف می شوند، اختلاف نماها ممکن است بزرگتر از $n-1$ باشد. اگر این حالت اتفاق بیفتد ماتریس با نمای کوچکتر کلاً از ثبات شیفته داده می شود. هم ردیف کردن مانتیس رادر شکل ۱۵-۱۰ با اضافه کردن یک توالی شماری که تعداد شیفته ها را می شمارد اصلاح کنید. اگر تعداد شیفته ها بزرگتر از $n-1$ باشد، بزرگترین عدد برای تعیین نتیجه بکار خواهد رفت.

۲۱-۱۰ رویه هم ردیف کردن مانتیس ها در جمع یا تفریق اعداد ممیز شناور بشرح زیر می تواند بیان شود: نمای کوچکتر را از بزرگتر تفریق کنید و مانتیسی را که دارای نمای کوچکتر است به تعداد

تفاضل نماها به راست شیفت دهید. نمای مجموع (یا تفاضل) برابر با نمای بزرگتر است. بدون استفاده از مقایسه کننده مقادیرها، با فرض نماهای بایاس شده، و در نظر گرفتن اینکه فقط AC را می توان جابجا کرد، الگوریتمی را به شکل فلوچارت برای هم ردیف کردن مانتیس ها و قرار دادن نمای بزرگتر در AC بدست آورید.

۱۰-۲۲ نشان دهید که سرریز مانتیس بعد از عمل ضرب ممکن نیست.

۱۰-۲۳ نشان دهید که تقسیم دو عدد نرمالیزه ممیز شناور با مانتیس کسری همواره خارج قسمت نرمالیزه خواهد داد مشروط بر اینکه قبل از عمل تقسیم، هم ردیف کردن مقسوم انجام شود.

۱۰-۲۴ فلوچارت شکل ۱۷-۱۰ را برای تهیه یک باقیمانده نرمالیزه ممیز شناور در AC تعمیم دهید. مانتیس باید یک کسر باشد.

۱۰-۲۵ در الگوریتم های اعمال حسابی ممیز شناور در بخش ۵-۱۰ امکان سرریز یا فروریز در نما نادیده گرفته می شود.

الف) سه فلوچارت را مرور کنید و محل هایی را که سرریز نما ممکن است رخ دهد پیدا کنید.
ب) قسمت الف) را برای فروریز تکرار کنید. فروریز نما هنگامی رخ می دهد که نما از کوچکترین عددی که بتواند در ثبات جای گیرد کمتر باشد.

ج) نشان دهید که سرریز یا فروریز نما بوسیله سخت افزار می تواند آشکار شود.

۱۰-۲۶ اگر نمایش عدد صحیح را برای مانتیس اعداد ممیز شناور تصور کنیم، درحین ضرب و تقسیم با مشکلات خاصی روبرو می شویم. اجازه بدهید تعداد بیت های بخش مقدار مانتیس $(n-1)$ باشد. برای نمایش صحیح:

الف) نشان دهید که اگر حاصلضرب با دقت معمولی بکار رود، باید $(n-1)$ به نمای حاصلضرب در AC اضافه شود.

ب) نشان دهید که اگر از مقسوم با مانتیس دقت معمولی استفاده شود، باید هنگامی که Q برابر صفر می شود $(n-1)$ از نمای مقسوم تفریق شود.

۱۰-۲۷ سخت افزاری، که برای جمع و تفریق دو عدد دهدهی در نمایش اندازه علامت دار بکار می رود را نشان دهید. چگونگی آشکارسازی سرریز را نشان دهید.

۱۰-۲۸ نشان دهید که $356-673$ با جمع 673 با متمم 10 عدد 356 و چشم پوشی از نقلی نهایی انجام می شود. بلاک دیاگرام سه مرحله از واحد حسابی دهدهی را رسم کنید و نشان دهید چگونه این عمل پیاده سازی می شود. تمام بیت های ورودی و خروجی واحد حساب را لیست کنید.

۱۰-۲۹ نشان دهید که به جای جمع کننده دودویی چهاربیتی پائین رتبه در شکل ۱-۱۰ می توان یک تمام جمع کننده و دو نیم جمع کننده قرار داد

۱۰-۳۰ با استفاده از تکنیک های طراحی مدارهای ترکیبی، توابع بولی متمم ساز 9 اعداد BCD شکل ۱۹-۱۰ را بدست آورید. دیاگرام منطقی را رسم کنید.

- ۱۰-۳۱ لازم است تا برای دو رقم دهدهی در کد افزونی 3 (جدول ۶-۳) یک جمع کننده طراحی شود. نشان دهید که تصحیح پس از جمع دو رقم با جمع کننده چهاربیتی بصورت زیر است.
- الف) رقم نقلی خروجی برابر نقلی تصحیح نشده است
ب) اگر نقلی خروجی برای 1 باشد، 0011 اضافه می شود
ج) اگر نقلی خروجی برابر 0 باشد، 1101 اضافه می شود و نقلی حاصل از این جمع نادیده گرفته می شود. نشان دهید که جمع کننده افزونی 3 می تواند با هفت تمام جمع کننده و دو معکوس کننده ساخته شود.
- ۱۰-۳۲ مداری را برای یک متمم ساز 9 هنگامی که ارقام دهدهی به فرم کد افزونی 3 نمایش داده می شوند، بدست آورید یک ورودی کنترل مد¹ تعیین می کند آیا رقم متمم شده است یا خیر. مزیت استفاده از این کد نسبت به کد BCD چیست؟
- ۱۰-۳۳ سخت افزار بکار رفته برای جمع و تفریق دو عدد دهدهی با نمایش متمم 10 علامت دار اعداد منفی را نشان دهید. نشان دهید که چگونه سرریز آشکار می شود. الگوریتمی بشکل فلوجارت تهیه کنید و خود را از صحت تولید پاسخ صحیح بوسیله آن مطمئن سازید.
- ۱۰-۳۴ محتوای ثبات های A و B و Q و SC را در حین ضرب دهدهی (شکل ۲۲-۱۰)، (الف) 470×152 و (ب) 999×199 بدست آورید. ثبات ها را سه رقمی تصور کنید و دومین عدد را مضروب فیه فرض نمایید.
- ۱۰-۳۵ محتوای ثبات های A، E و Q و SC را در حین تقسیم دهدهی (شکل ۲۳-۱۰) $1680/32$ نشان دهید. ثبات ها را دو رقمی فرض کنید
- ۱۰-۳۶ نشان دهید که زیر ثبات A در شکل ۲۱-۱۰ در پایان (الف) ضرب دهدهی چنانکه در شکل ۲۲-۱۰ مشخص شده است، و (ب) در تقسیم شکل ۲۳-۱۰، صفر است.
- ۱۰-۳۷ الگوریتم های حساب ممیز شناور در بخش های ۵-۱۰ را از داده های دودویی به داده های دهدهی تغییر دهید. در یک جدول نحوه تفسیر هر یک از نمادهای ریز عمل ها را بنویسید.

۱۱

سازمان ورودی - خروجی

- | | |
|----------------------------|------------------------------------|
| ۱۱-۱ وسایل جانبی | ۱۱-۵ وقفه اولویت دار |
| ۱۱-۲ واسطه ورودی - خروجی | ۱۱-۶ دستیابی مستقیم به حافظه (DMA) |
| ۱۱-۳ انتقال غیرهمزمان داده | ۱۱-۷ پردازنده ورودی و خروجی IOP |
| ۱۱-۴ شیوه‌های انتقال | ۱۱-۸ تبادل اطلاعات سری |

۱۱-۱ وسایل جانبی^۱

زیر سیستم ورودی - خروجی یک کامپیوتر که I/O نامیده می‌شود، روش تبادل اطلاعات کارآیی را بین سیستم مرکزی و محیط خارجی فراهم می‌کند. برنامه‌ها و داده‌ها باید برای پردازش وارد حافظه کامپیوتر شوند و نتایج حاصل از محاسبات نیز باید برای کاربر ضبط و یا نمایش داده شوند. کامپیوتر بدون امکان دریافت اطلاعات از یک منبع خارجی و یا انتقال نتایج بشکل مفهوم بی‌معنی است. آشناترین راه ورود اطلاعات بداخل یک کامپیوتر استفاده از صفحه کلیدهایی شبیه ماشین‌های تحریر است که به شخص اجازه می‌دهد تا اطلاعات الفبا عددی را مستقیماً وارد نماید. هر بار که کلیدی فشرده شود، پایانه کاراکتر کد شده بفرم دودویی را به کامپیوتر می‌فرستد. بالاترین سرعت ممکن برای ورود اطلاعات با این روش به سرعت ماشین نویس بستگی دارد. از طرف دیگر، واحد پردازش مرکزی وسیله فوق‌العاده سریعی است که می‌تواند عملیات را با سرعت بسیار بالا انجام دهد. وقتی که اطلاعات ورودی از طریق یک صفحه کلید کند به پردازنده ارسال شود، پردازنده باید مادامی که منتظر رسیدن اطلاعات است بیکار بماند. برای استفاده کارا از یک کامپیوتر مقدار زیادی از برنامه‌ها و داده‌ها از قبل آماده و روی یک محیط ذخیره سازی مانند نوار یا دیسک مغناطیسی قرار داده می‌شود. سپس اطلاعات موجود در دیسک با سرعت بالایی به حافظه کامپیوتر منتقل می‌شود. نتایج محاسبات نیز به یک محیط

1- Peripheral Devices

ذخیره سازی سریع، مانند دیسک منتقل می‌گردد و بعداً از آنجا می‌تواند به یک چاپگر ارسال و لذا نتایج بصورت خروجی چاپ شده در اختیار قرار می‌گیرد.

وسایلی که تحت کنترل مستقیم کامپیوتر می‌باشند، گفته می‌شود روی خط^۱ هستند. این وسیله‌ها بگونه‌ای طراحی می‌شوند که پس از دریافت فرمان از CPU اطلاعات را از حافظه بخوانند یا در آن بنویسند و در واقع بخشی از کل سیستم کامپیوتری محسوب می‌شوند. وسایل ورودی یا خروجی متصل به کامپیوتر وسایل جانبی هم خوانده می‌شوند. معمول‌ترین وسایل جانبی عبارتند از صفحه کلیدها، واحدهای نمایش^۲ و چاپگرها. وسایلی که ذخیره‌سازی کمکی را برای سیستم فراهم می‌نمایند عبارتند از دیسک‌ها و نوارهای مغناطیسی. وسایل جانبی وسیله‌های الکترومکانیکی و الکترومغناطیسی با پیچیدگی‌های خاص خود هستند. ما در اینجا بحث مختصری را از کار آنها را بدون جزئیات ساختمان داخلی آنها ارائه می‌نمائیم.

مونیتورهای تصویری^۳ متداولترین وسیله‌های جانبی هستند. این وسایل از یک صفحه کلید بعنوان وسیله ورودی و یک واحد نمایش تصویری بعنوان وسیله خروجی تشکیل شده‌اند. انواع متنوعی از نمایشگرهای تصویر موجودند، ولی معمول‌ترین آنها از لامپ اشعه کاتودی^۴ (CRT) استفاده می‌نمایند. CRT حاوی یک تفنگ الکترونی است که یک پرتو الکترونی را به صفحه فسفرسانی که در جلو لامپ قرار دارد می‌فرستد. این پرتو را می‌توان بصورت افقی و عمودی منحرف کرد. برای ایجاد الگویی بر روی صفحه نمایش، یک شبکه فلزی که در داخل CRT کار گذاشته شده و ولتاژ متغیری را دریافت کرده و موجب می‌شود تا پرتو در نقاط موردنظر بر صفحه بتابد و آن مکان را روشن نماید. سیگنالهای افقی و عمودی پرتو را هدایت می‌کنند و موجب حرکت جاروبی آن در امتداد عرض صفحه نمایش می‌شوند و لذا الگوی تصویری موردنظر بر روی صفحه ظاهر می‌گردد. یکی از ویژگیهای وسایل نمایش، مکان‌نمایی^۵ است که محل قرار گرفتن کاراکتر بر روی صفحه نمایش را مشخص می‌کند. مکان نما را می‌توان به هر جایی در صفحه، برای هر کاراکتر، آغاز هر کلمه، یا هر خط انتقال داد. کلیدهای ویرایشی^۶ براساس محل مکان‌نما اطلاعات را اضافه یا حذف می‌نمایند. پایانه نمایش^۷ می‌تواند در روش تک‌نویسی^۸ کار کند که در آن همه کاراکترها همزمان با ورود بوسیله صفحه کلید و نمایش بر روی صفحه به کامپیوتر ارسال می‌شوند. در روش بلاکی^۹، متن ویرایش شده ابتدا در یک حافظه محلی در داخل پایانه ذخیره می‌شود. این متن بعداً بصورت بلاکی از داده به کامپیوتر منتقل می‌گردد.

چاپگرها، داده‌ها یا متن خروجی کامپیوتر را بطور دائمی روی کاغذ ثبت می‌کنند. سه نوع عمده از چاپگرهای کاراکتری وجود دارند: چرخ‌گردان^{۱۰}، ماتریسی^{۱۱} و لیزری. چاپگر چرخ‌گردان دارای چرخ

1- ON-Line

2- Display Unit

3- Video Monitors

4- Cathode Ray Tube

5- Cursor

6- Edit

7- Display Terminal

8- Single -Character Mode

9- Block Mode

10- Daisywheel

است که کاراکترها حول محیط آن حک شده‌اند. برای چاپ یک کاراکتر، چرخ چرخیده و در موقعیت مناسبی قرار گرفته و سپس یک آهن ربای الکتریکی حرف موردنظر را روی نوار فشار می‌دهد. چاپگر ماتریسی حاوی مجموعه‌ای از نقطه‌ها در امتداد مکانیزم چاپ است. مثلاً یک چاپگر ماتریسی 5×7 قادر است 80 کاراکتر را در هر سطر چاپ کند که هر سطر نیز از هفت خط افقی تشکیل شده است. بنابراین هر خط $5 \times 80 = 400$ نقطه دارد. هر نقطه می‌تواند بسته به کاراکتر خاصی که روی خط چاپ می‌شوند، چاپ بشود یا نشود. چاپگر لیزری از یک استوانه عکاسی گردان استفاده می‌کند که با استفاده از آن تصاویر کاراکتر نقش می‌شوند. سپس الگو مشابه ماشین فتوکپی روی کاغذ منتقل می‌گردد.

غالباً نوارهای مغناطیسی برای ذخیره فایل‌های داده، مثل وضعیت حقوقی کارکنان یک شرکت بکار می‌رود. نحوه دستیابی به اطلاعات سری بوده و هر نوار از رکوردهایی تشکیل شده است و با عبور نوار از مقابل یک مکانیزم خواندن - نوشتن ثابت، می‌توان به رکوردها یکی پس از دیگری دست یافت. این روش یکی از ارزاترین و کندترین روشهای ذخیره‌سازی است ولی این مزیت را داراست که هنگام بکار گرفتن می‌توان نوار را خارج کرد. دیسک‌های مغناطیسی سطوح دوار سریعی دارند که با مواد مغناطیسی پوشانده شده‌اند. دستیابی اطلاعات با حرکت دادن یک مکانیزم متحرک خواندن و نوشتن و قرار دادن آن بر روی یکی از شیارها سطح مغناطیسی صورت می‌گیرد. از دیسک اکثراً برای ذخیره انبوه برنامه‌ها و داده‌ها استفاده می‌شود. در رابطه با نقش نوارها و دیسک‌ها به عنوان حافظه کمکی در بخش ۱-۱۲ بیشتر بحث خواهیم کرد.

وسایل ورودی و خروجی دیگری که در سیستم‌های کامپیوتری با آنها مواجه می‌شویم پلاترهای دیجیتال (رسام)، کاراکتر خوان نوری و مغناطیسی، مبدل‌های آنالوگ به دیجیتال و انواع تجهیزات دریافت^{۱۲} (اکتساب) داده می‌باشند. از کامپیوترها برای کنترل فرآیندهای مختلف به صورت بلادرنگ^{۱۳}، مانند ماشینهای ابزار، عملیات خط مونتاژ و فرآیندهای شیمیایی و صنعتی استفاده می‌شود. برای چنین کاربردهایی، باید روشی برای درک شرایط وضعیتی در فرآیند و ارسال سیگنال‌های کنترلی به فرآیند تحت کنترل فراهم شود.

سازمان ورودی - خروجی یک کامپیوتر تابعی از اندازه کامپیوتر و وسیله‌های متصل به آن است. تفاوت یک سیستم کوچک یا بزرگ عمدتاً به مقدار سخت افزاری که کامپیوتر برای ارتباط با وسایل جانبی در اختیار دارد و تعداد این وسایل بستگی دارد. چون هر وسیله جانبی رفتار متفاوتی با وسیله دیگر دارد، بحث درباره جزئیات اتصال بین کامپیوتر و هر دستگاه جانبی مشکل خواهد بود. در این فصل تکنیک‌های معینی را که در اکثر دستگاه‌های جانبی مشترک است ارائه می‌نمائیم.

11- Dot Matrix

12- Data Acquisition Equipment

13- Real - Time System

کاراکترهای الفبا عددی آسکی^۱

وسایل ورودی و خروجی که بین انسان‌ها و کامپیوتر ارتباط برقرار می‌کنند معمولاً اطلاعات الفبا عددی را با کامپیوتر تبادل می‌نمایند. کد دودویی کاراکترهای الفبا عددی آسکی است. این کد، همانطور که در جدول ۱-۱۱ نشان داده شده است از هفت بیت برای کد کردن 128 کاراکتر استفاده می‌کند. هفت بیت کد با b_1 الی b_7 نشان داده شده‌اند، که در آن b_7 با ارزشترین بیت است. حرف A، مثلاً در ASCII بصورت 1000001 (ستون 100 و سطر 0001) نمایش داده می‌شود. کد آسکی شامل 94 کاراکتر چاپ شدنی و 34 کاراکتر غیرچاپ شدنی است که برای مقاصد کنترلی بکار می‌رود. کاراکترهای چاپ شدنی عبارتند از 26 حرف بزرگ الفبا از A تا Z، 26 حرف کوچک الفبا، 10 عدد از 0 تا 9 و 32 کاراکتر قابل چاپ خاص مانند %، * و \$.

34 کاراکتر کنترل در جدول ASCII با نام‌های خاص بکار رفته‌اند. این کاراکترها مجدداً در قسمت پائین جدول همراه با نام عملیاتی‌شان آورده شده‌اند. کاراکترهای کنترلی برای مسيردهی داده‌ها و آرایش مسير متن چاپ به شکلی از پیش تعیین شده به کار می‌روند. سه نوع کاراکتر کنترل موجود است: شکل دهنده‌ها، جداکننده‌های اطلاعات و کاراکترهای کنترل ارتباطات. شکل دهنده‌ها کاراکترهایی هستند که آرایش و نظم چاپ را کنترل می‌نمایند. این کاراکترها شامل کنترل‌های شناخته شده ماشین‌های تحریر، مانند کلید پس‌بَر^۲ (BS)، جدول بندی افقی^۳ (HT)، برگشت نورد^۴ (CR) هستند. جداکننده‌های اطلاعات برای جداسازی اطلاعات به بخش‌هایی مانند پاراگراف و صفحه بکار می‌روند. این دسته شامل کاراکترهایی مانند جداکننده رکوردها^۵ (RS) و جداکننده فایل‌ها^۶ (FS) است. کاراکترهای کنترل ارتباطات هنگام ارسال متن بین پایانه‌های دور از هم مفیدند. مثال‌هایی از این نوع عبارتند از آغاز متن^۷ STX و پایان متن^۸ ETX، که برای کادربندی یک متن پیام هنگام ارسال از طریق یک رسانه ارتباطی بکار می‌روند.

آسکی یک کد هفت بیتی است، ولی در اکثر کامپیوترها داده‌ها بصورت کمیت هشت بیتی بنام بایت دستکاری می‌شوند. بنابراین کاراکترهای ASCII اغلب بصورت یک کاراکتر در هر بایت ذخیره می‌شوند. بیت اضافی بسته به کار برد گاهی برای مقاصد دیگر بکار می‌رود. مثلاً، بعضی چاپگرها کاراکترهای آسکی را بصورت هشت بیتی با مقدار 0 در بیت هشتم تشخیص می‌دهند. از 128 کاراکتر هشت بیتی با مقدار 1 در بیت هشتم برای دیگر سمبل‌ها مانند الفبای یونانی یا فونت‌های ایتالیک استفاده می‌شود. هنگام استفاده در انتقال داده‌ها، بیت هشتم ممکن است برای نشان دادن توازن کاراکتر کد شده با دودویی هم بکار رود.

1- American Standard Code for Information Interchange (ASCII)

2- Backspace

3- Horizontal Tabulation

4- Carriage Return

5- Record Separator

6- File Separator

7- Start of Text

8- End of Text

جدول ۱۱-۱ کد استاندارد آمریکایی برای تبادل اطلاعات (اسکی)

$b_4 b_3 b_2 b_1$	$b_7 b_6 b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	'	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	\$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(8	H	X	h	x
1001	HT	EM)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[k	{
1100	FF	FS	,	<	L	\	l	
1101	CR	GS	-	=	M]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

NUL	هیچ	DLE	کاراکترهای کنترل
SOH	شروع بتر	DC1	خروج از ارتباط داده
STX	آغاز متن	DC2	کنترل وسیله 1
ETX	پایان متن	DC3	کنترل وسیله 2
EOT	استعلام	DC4	کنترل وسیله 3
ENQ	تصدیق	NAK	کنترل وسیله 4
ACK	بوق	SYN	تصدیق منفی
BEL	پستر	ETB	همگامی در حالت بیکاری
BS	توقف افقی	CAN	پایان بلاک ارسال
HT	تعویض سطر	EM	لغو
LF	توقف عمودی	SUB	پایان محیط
VT	تعویض صفحه	ESC	جایگزین
FF	برگشت نورد	FS	خروج
CR	انتقال به خارج	GS	جداکننده فایل ها
SO	انتقال به داخل	RS	جداکننده گروه ها
SI	انتقال به داخل	US	جداکننده رکوردها
SP	فضای خالی	DEL	جداکننده واحدها
			پاک کننده

۱۱-۲ واسطه ورودی - خروجی

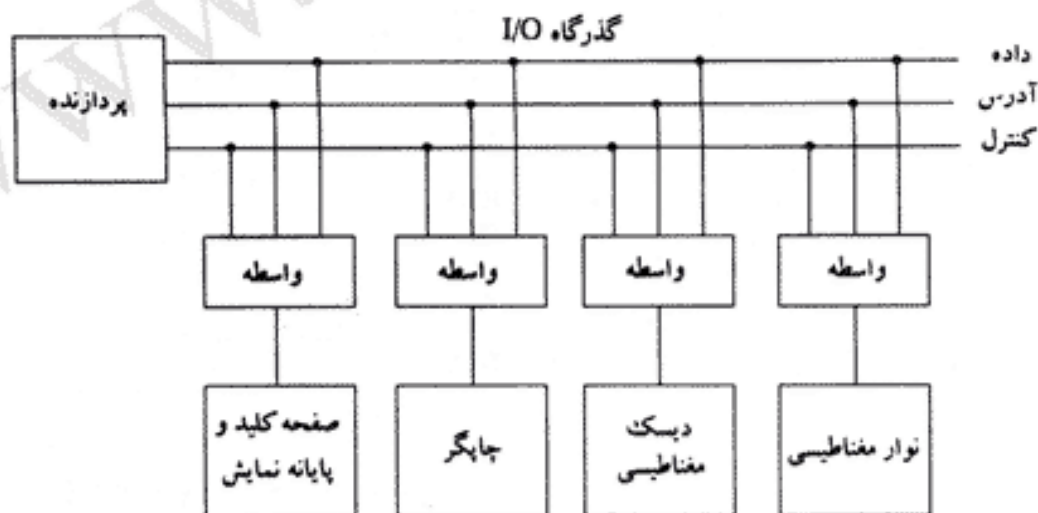
واسطه ورودی - خروجی روشی را برای انتقال اطلاعات بین محیطهای ذخیره سازی داخلی و وسیله های I/O خارجی در اختیار می گذارند. وسایل جانبی متصل به یک کامپیوتر اتصالات ارتباطی خاصی برای وصل به واحد پردازش نیاز دارند. منظور از اتصالات ارتباطی از میان برداشتن تفاوت های

- موجود بین کامپیوتر مرکزی و هر وسیله جانبی است. تفاوت های مهم موجود عبارتند از:
- ۱- وسایل جانبی وسایلی الکترومکانیکی یا الکترومغناطیسی هستند و نحوه عملکرد آنها با CPU و حافظه که وسایلی الکترونیکی می باشند متفاوت است. بنابراین ممکن است تبدیل سیگنال ها لازم باشد.
 - ۲- سرعت انتقال داده در وسایل جانبی معمولاً کمتر از سرعت انتقال CPU است، و در نتیجه، مکانیزم همزمانی ممکن است لازم باشد.
 - ۳- کدها و قالب های داده ها در وسایل جانبی با قالب کلمات در CPU و حافظه تفاوت دارند.
 - ۴- روش های عملکرد دستگاه های جانبی با یکدیگر متفاوت بوده و هر یک باید به نوعی کنترل شوند به نحوی که عملکرد دیگر دستگاه های متصل به CPU مختل نگردد.

برای برطرف کردن این تفاوت، سیستم های کامپیوتری قطعات و مدارهای خاصی بین CPU و دستگاه های جانبی برای مدیریت بر همه انتقال های ورودی و خروجی و همزمان سازی بین آنها را دارا هستند. این مدارات واحدهای واسطه^۱ نامیده می شوند زیرا واسطه بین گذرگاه پردازنده و دستگاه جانبی می باشند. بعلاوه، هر وسیله جانبی ممکن است کنترل کننده خاصی برای خود داشته باشد که بر عملیات مکانیزم خاص در آن وسیله نظارت می کند.

گذرگاه I/O و ماژول های واسطه

نمونه ای از اتصالات ارتباطی بین پردازنده و چند وسیله جانبی در شکل ۱-۱۱ نشان داده شده است. گذرگاه I/O از خطوط داده، خطوط آدرس و خطوط کنترل تشکیل شده است. دیسک مغناطیسی، چاپگر



شکل ۱-۱۱ اتصال گذرگاه I/O به وسیله های ورودی-خروجی

و پایانه عملاً در هر کامپیوتر چند منظوره بکار می‌روند. نوار مغناطیسی در بعضی از کامپیوترها بعنوان پشتیبان ذخیره سازی بکار می‌رود. برای هر وسیله جانبی، یک واحد واسطه وجود دارد. هر واحد واسطه، آدرس و کنترل دریافتی از گذرگاه I/O را دیکد کرده، سپس آنها را برای وسایل جانبی تفسیر، و سیگنال‌های لازم را برای کنترل کننده‌های وسایل جانبی فراهم می‌آورد. واحد واسطه انتقال داده را هماهنگ نموده و انتقال بین وسیله جانبی و پردازنده را مدیریت می‌کند. هر وسیله جانبی کنترل کننده خاص خود را داراست و توسط آن وسیله الکترومکانیکی خاصی را بکار می‌اندازد. مثلاً، یک کنترل کننده خاص حرکت کاغذ، زمانبندی چاپگر، و انتخاب کاراکتر چاپ شونده را کنترل می‌کند. یک کنترل کننده را می‌توان بطور جداگانه یا به همراه وسیله جانبی در مجموعه جاسازی کرد.

گذرگاه I/O پردازنده به تمام واسطه‌های وسایل جانبی متصل است. برای ارتباط با یک وسیله خاص، پردازنده آدرس یک وسیله را روی خطوط آدرس قرار می‌دهد. وقتی که واسطه آدرس خود را در می‌یابد، مسیر بین خطوط گذرگاه و وسیله‌ای را که کنترل می‌نماید فعال می‌سازد. کلیه وسایل جانبی که آدرسشان با آدرس روی گذرگاه مربوط نمی‌شود بوسیله مدار واسطه غیرفعال می‌گردند.

همزمان با وجود آدرس روی خطوط آدرس، پردازنده یک کد عملیات را روی خطوط کنترل قرار می‌دهد. واسطه انتخاب شده به کد عملیات پاسخ داده و آنرا اجرا می‌نماید. کد عملیات برای I/O یک فرمان تلقی شده و در واقع دستورالعملی است که در مدار واسطه و واحد جانبی اجرا می‌گردد. تفسیر فرمان، به وسیله جانبی که پردازنده آنرا احضار می‌کند بستگی دارد. یک واحد واسطه ممکن است چهار نوع فرمان دریافت کند. این فرمان‌ها عبارتند از کنترل، وضعیت، داده خروجی، داده ورودی.

یک فرمان کنترل برای فعال کردن وسیله جانبی و اینکه چه کاری را انجام دهد صادر می‌گردد. مثلاً، به یک واحد نوار مغناطیسی ممکن است فرمان داده شود تا باندازه یک رکورد بعقب برگردد، نوار به ابتدای آن برگردانده شود، و یا از ابتدا به سمت جلو حرکت داده شود. فرمان کنترل خاص صادره به وسیله جانبی بستگی داشته و هر وسیله جانبی رشته فرمان‌های متوالی خاص خود را بسته به نوع عملکرد، تشخیص می‌دهد.

یک فرمان وضعیت برای آزمایش شرایط وضعیتی مختلف در مدار واسطه و وسیله جانبی بکار می‌رود. مثلاً، کامپیوتر ممکن است مایل به چک کردن وضعیت یک وسیله جانبی قبل از آغاز انتقال داده باشد. در هنگام انتقال ممکن است مدار واسطه وقوع یک یا چند خطا را تشخیص دهد. این خطاها با نشاندن¹ (1 شدن) بیت‌ها در ثبات وضعیت مشخص شده و پردازنده می‌تواند با فواصل زمانی معینی آنها را بخواند.

فرمان خروجی داده سبب می‌شود تا مدار واسطه داده‌ها را از گذرگاه به یکی از ثبات‌های خود انتقال دهد. بعنوان مثال واحد نوار را در نظر بگیرید. کامپیوتر نوار را با صدور یک فرمان کنترل به حرکت در می‌آورد. آنگاه پردازنده، وضعیت نوار را با یک فرمان وضعیت ملاحظه می‌کند و وقتی که نوار در محل

1- Set

صحیحی قرار گرفته باشد، پردازنده یک فرمان خروجی داده را صادر می نماید. واحد واسطه به آدرس و فرمان پاسخ داده و اطلاعات را از خطوط داده در گذرگاه به ثبات بافر منتقل می کند. سپس واحد واسطه با کنترل کننده نوار ارتباط برقرار کرده و داده ای را که قرار است روی نوار ذخیره شود ارسال می نماید. فرمان ورودی داده عکس خروجی داده است. در این حالت مدار واسطه یک قلم داده را از وسیله جانبی دریافت و آنرا در ثبات بافر قرار می دهد. پردازنده وجود داده را توسط فرمان وضعیت چک کرده و سپس یک فرمان ورودی داده را صادر می کند. واحد واسطه داده را روی خطوط داده گذارده و پردازنده آن را می پذیرد.

گذرگاه I/O در برابر گذرگاه حافظه

- علاوه بر ارتباط با I/O، پردازنده باید با واحد حافظه هم ارتباط برقرار کند. همانند گذرگاه I/O، گذرگاه حافظه هم حاوی داده، آدرس و خطوط کنترل نوشتن و خواندن است. برای ارتباط گذرگاه های کامپیوتر با حافظه و I/O سه روش وجود دارد.
- ۱- از دو گذرگاه جداگانه، یکی برای حافظه و دیگری برای I/O استفاده شود.
 - ۲- برای هر دو بخش حافظه و I/O از یک گذرگاه مشترک استفاده شود ولی هر کدام خطوط کنترل جداگانه داشته باشند.
 - ۳- هر دو بخش حافظه و I/O از گذرگاه و خطوط کنترل مشترک استفاده نمایند.

در اولین روش، کامپیوتر دارای مجموعه گذرگاه جداگانه داده، آدرس و کنترل است، یک مجموعه برای دستیابی به حافظه و دیگری برای دستیابی به I/O. عمل دستیابی در کامپیوترهایی انجام می شود که دارای یک پردازنده I/O (IOP) علاوه بر واحد پردازش مرکزی (CPU) می باشند. حافظه با هر دو واحد CPU و IOP از طریق گذرگاه خود ارتباط برقرار می کند. IOP هم با وسایل ورودی و خروجی از طریق گذرگاه I/O جداگانه ای با خطوط خاص آدرس، داده و کنترل ارتباط برقرار می کند. هدف از IOP فراهم کردن مسیر مستقلی برای انتقال اطلاعات بین وسایل خارجی و حافظه درونی است. پردازنده I/O گاهی اوقات کانال داده^۱ هم خوانده می شود. در بخش ۷-۱۱ ماکار IOP را مفصل تر بحث خواهیم کرد.

I/O ایزوله در مقابل تگاشت یافته در حافظه^۲

بسیاری از کامپیوترها برای انتقال اطلاعات بین حافظه و I/O از یک گذرگاه مشترک استفاده می کنند. تمایز بین انتقال حافظه و I/O بلحاظ استفاده از خطوط خواندن و نوشتن جداگانه است. CPU با تواناسازی یکی از دو خط خواندن و نوشتن مشخص می کند که آدرس روی خطوط آدرس مربوط به کلمه حافظه است یا به ثبات واسطه مربوط می باشد. خطوط کنترل خواندن I/O یا نوشتن I/O بهنگام

1- Data Channel

2- Isolated I/O

انتقال حافظه فعال می شوند. این پیکربندی آدرس تمام واسطه های I/O را از آدرس های تخصیص یافته به حافظه، جدا کرده و روش I/O ایزوله (جدا شده) برای تخصیص آدرس در گذرگاه مشترک نامیده می شود. در آرایش I/O ایزوله، CPU دارای دستورالعمل های جداگانه ورودی و خروجی است، و هر یک از این دستورات همراه با آدرس یک ثبات واسطه است. وقتی که CPU کد عملیات یک دستورالعمل ورودی یا خروجی را برداشت می کند، آدرس مربوط به دستورالعمل را روی خطوط آدرس مشترک می گذارد. در همان زمان، خط کنترل خواندن I/O (برای ورودی) یا نوشتن I/O (برای خروجی) را فعال می سازد. این عمل قطعات خارجی را که به گذرگاه مشترک متصلند مطلع می سازد که آدرس روی خطوط آدرس متعلق به ثبات واسطه است و نه به یک کلمه حافظه. از طرف دیگر، وقتی که CPU دستورالعمل یا عملوندی را از حافظه برداشت می کند، آدرس حافظه را روی خطوط آدرس قرار داده و خط کنترل خواندن یا نوشتن حافظه را فعال می نماید. این عمل قطعات خارجی را مطلع می سازد که آدرس مربوط به کلمه حافظه است و نه متعلق به یک واسطه I/O.

روش I/O ایزوله آدرس های حافظه و I/O را طوری از هم جدا می کند که حافظه ها بوسیله آدرس های تخصیص یافته به مدار واسطه تحت تأثیر قرار نمی گیرند زیرا هر کدام فضای آدرس خاص خود را دارند. روش دیگر استفاده از فضای آدرس یکسان برای حافظه و I/O است. این روش در کامپیوترهایی بکار می رود که فقط از یک مجموعه از سیگنالهای خواندن و نوشتن استفاده می کنند و تمایزی بین آدرس حافظه و I/O وجود ندارد. این آرایش، I/O نگاشت یافته در حافظه¹ نامیده می شود. کامپیوتر با ثبات های مدارات واسطه مانند بخشی از حافظه سیستم رفتار می کند. در این روش نمی توان از آدرس های تخصیص یافته به ثبات های واسطه، برای کلمات حافظه استفاده کرد. بدین ترتیب محدوده آدرس های ممکن برای حافظه کاهش می یابد.

در سازمان I/O نگاشت یافته در حافظه، دستورالعمل خاصی برای ورودی و خروجی وجود ندارد. CPU قادر است داده های I/O واقع در ثبات واسطه را با همان دستوراتی که برای حافظه بکار می روند دستکاری کند. هر واسطه به صورت مجموعه ای از ثبات ها سازماندهی شده است و به درخواست های خواندن و نوشتن در فضای عادی آدرس پاسخ می دهد. نوعاً، قطعه ای از کل فضای آدرس برای ثبات های واسطه رزرو می شود، اما در حالت کلی، می توانند در هر آدرسی واقع شوند، بشرطی که کلمه ای از حافظه به همان آدرس پاسخ ندهد.

کامپیوترهایی که دارای سازمان I/O نگاشت یافته در حافظه هستند می توانند با استفاده از دستورالعمل های حافظه ای به داده های I/O دست یابند. این سازمان به کامپیوتر اجازه می دهد تا از دستورات یکسانی برای انتقال ورودی - خروجی و حافظه ای استفاده کنند. مزیت این است که دستورات بارکردن² و ذخیره کردن³ که برای خواندن و نوشتن از حافظه بکار می روند برای ورود و یا خروج داده از ثبات های I/O نیز مورد استفاده قرار گیرند. معمولاً در کامپیوترها دستورات حافظه ای

1- Memory Mapped I/O

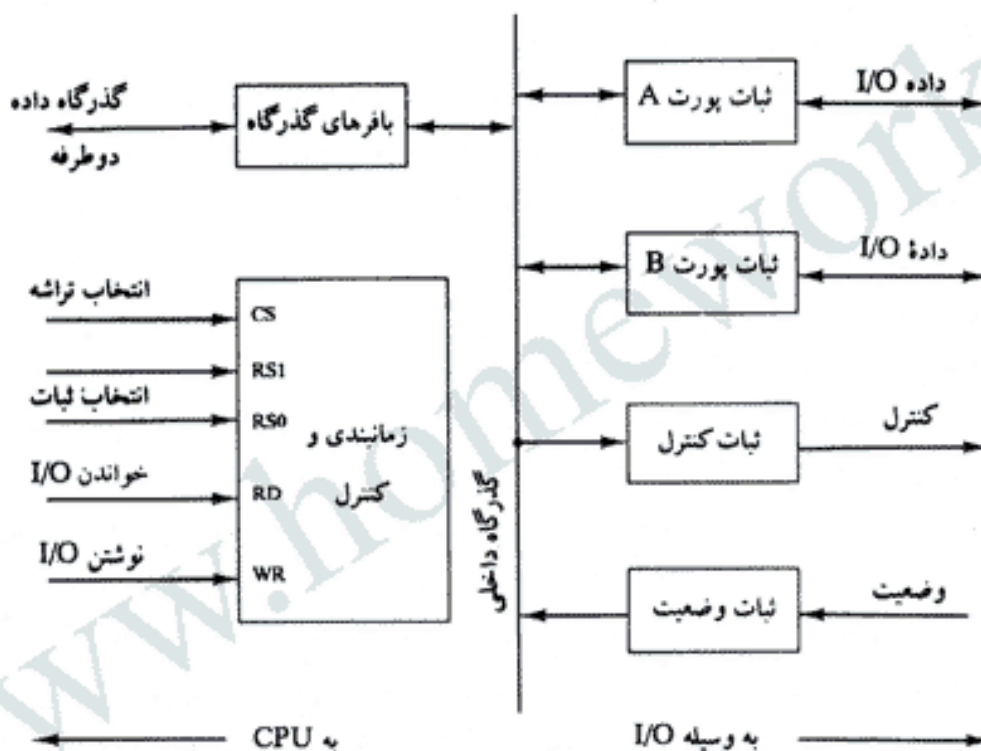
2- Load

3- Store

بیشتر از دستورات I/O است. با روش I/O نگاشت یافته در حافظه کلیه دستورات عمل‌های حافظه‌ای برای I/O هم قابل استفاده‌اند.

مثالی از واسطه I/O

مثالی از واحد واسطه I/O در شکل ۲-۱۱ بصورت بلاک دیاگرام نشان داده شده است. این واسطه متشکل از دو ثبت داده بنام پورت^۱ (درگاه)، یک ثبت کنترل، یک ثبت وضعیت، بافرهای گذرگاه و مدارات زمانبندی و کنترل است. مدار واسطه با CPU از طریق گذرگاه داده ارتباط برقرار می‌کند.



CS	RS1	RS0	ثباتی که انتخاب می‌شود
0	x	x	هیچ گذرگاه داده در امدانس بالا
1	0	0	ثبات پورت A
1	0	1	ثبات پورت B
1	1	0	ثبات کنترل
1	1	1	ثبات وضعیت

شکل ۲-۱۱ مثالی از واحد واسطه I/O

1- Port

ورودی‌های انتخاب تراشه و انتخاب ثبات، آدرس تخصیص یافته به مدار واسطه را معین می‌نمایند. خواندن و نوشتن I/O دو خط کنترل هستند که بترتیب ورودی یا خروجی را مشخص می‌کنند. چهار ثبات مستقیماً با وسایل I/O متصل به مدار واسطه ارتباط دارند.

داده I/O به واز وسایل می‌تواند به پورت A یا B منتقل گردد. مدار واسطه می‌تواند با یک وسیله ورودی، یا یک وسیله خروجی و یا وسیله‌ای که هم ورودی و هم خروجی است کار کند. اگر واسطه به یک چاپگر وصل باشد، فقط داده را خارج می‌کند، و اگر به یک کاراکترخوان سرویس دهد، فقط داده را وارد می‌کند. یک واحد دیسک مغناطیسی داده را در هر دو جهت بطور غیرهمزمان منتقل می‌نماید، بنابراین مدار واسطه می‌تواند از خطوط دو جهته استفاده کند. صدور فرمان به یک وسیله I/O با ارسال یک کلمه به یک ثبات مناسبی صورت می‌گیرد. در چنین سیستمی، کد تابع در گذرگاه I/O لازم نیست زیرا کنترل به ثبات کنترل ارسال می‌شود، اطلاعات وضعیت از ثبات وضعیت، و داده‌ها به و یا از پورت‌های A و B انتقال می‌یابند. بنابراین انتقال اطلاعات داده، کنترل و وضعیت همواره از طریق گذرگاه مشترک داده انجام می‌شود. تمایز بین اطلاعات داده، کنترل یا وضعیت با توجه به ثبات خاص در مدار واسطه‌ای که CPU با آن ارتباط برقرار می‌کند صورت می‌گیرد.

ثبات کنترل، اطلاعات کنترل را از CPU دریافت می‌نماید. با بارکردن بیت‌های معینی در ثبات کنترل، مدار واسطه و وسیله I/O متصل به آن می‌تواند در وضعیت‌های عملیاتی متنوعی قرار داده شود. مثلاً، پورت A ممکن است بعنوان ورودی و پورت B بعنوان خروجی تعریف شوند. به یک واحد نوار مغناطیسی ممکن است فرمان برگشت به عقب داده شود یا آن را بسمت جلو به حرکت درآورد. بیت‌های ثبات وضعیت برای شرایط وضعیتی و ضبط خطاهایی که در حین انتقال داده‌ها ممکن است رخ دهد بکار می‌رود. مثلاً، یک بیت وضعیت ممکن است نشان دهد که یک قلم داده جدید از وسیله I/O وارد شده است. بیت دیگری در ثبات وضعیت ممکن است وقوع یک خطای توازن را در حین انتقال معین نماید.

ثبات‌های واسطه از طریق گذرگاه دو جهته داده با CPU ارتباط برقرار می‌کنند. گذرگاه آدرس واحد واسطه را از طریق انتخاب کننده تراشه و دو ورودی انتخاب ثبات انتخاب می‌کند. برای تشخیص آدرس اختصاص یافته به ثبات‌های واسطه، یک مدار خارجی (معمولاً یک دیکدر) بکار گرفته می‌شود. این مدار ورودی انتخاب تراشه¹ (CS) را وقتی که واسطه بوسیله گذرگاه آدرس انتخاب شود فعال می‌سازد. دو ورودی انتخاب ثبات RS0 و RS1 معمولاً به دو خط کم ارزشتر گذرگاه آدرس وصل می‌شوند. این دو ورودی یکی از چهار ثبات را طبق جدول همراه دیاگرام انتخاب می‌نمایند. محتوای ثبات انتخاب شده از طریق گذرگاه داده، هنگامی که سیگنال خواندن I/O فعال شود، به CPU انتقال می‌یابد. CPU اطلاعات دودویی را از طریق گذرگاه داده وقتی که ورودی نوشتن I/O فعال شود، به ثبات انتخاب شده منتقل می‌سازند.

1- Chip Select

۳-۱۱ انتقال غیرهمزمان داده

عملیات درونی در یک سیستم دیجیتال بوسیله پالس های ساعت که بوسیله یک مولد پالس تهیه می شود همزمان می شوند. پالس های ساعت به تمام ثبات های درون یک واحد اعمال شده و کلیه انتقال داده ها بین ثبات های درونی همزمان با وقوع پالس ساعت رخ می دهند. معمولاً هر دو واحد CPU و واسطه I/O مستقل از یکدیگر طراحی می شوند. اگر ثبات های داخلی مدار واسطه از ساعت مشترکی با ثبات های CPU استفاده کنند، انتقال بین دو واحد همزمان^۱ (یا همگام) خوانده می شود. در اکثر موارد، زمانبندی درونی در هر واحد از دیگری مستقل است زیرا هر یک از ساعت اختصاصی خود برای ثبات های خودشان استفاده می کنند. در این صورت دو واحد را غیرهمزمان^۲ یا ناهمگام با هم خوانیم. این روند بطور گسترده ای در بیشتر سیستم های کامپیوتری بکار گرفته شده اند.

انتقال غیرهمزمان بین دو واحد مستقل مستلزم ارسال سیگنال های کنترل بین واحدهای ارتباط یافته است تا زمانی را که داده باید انتقال یابد مشخص کنند. یکی از راه های دستیابی به این هدف تهیه پالس فعال ساز^۳ (استروب) بوسیله یکی از واحدها، برای مشخص کردن زمان وقوع انتقال به واحد دیگر است. روش دیگر که عموماً بکار می رود همراهی اقلام داده با یک سیگنال کنترل است که حضور داده را در گذرگاه مشخص می نماید. واحدی که داده را دریافت می کند با یک سیگنال کنترل دیگر دریافت آنرا تأیید می نماید. این نوع توافق بین دو واحد مستقل، دست دهمی^۴ خوانده می شود.

روش پالس فعال ساز^۵ و دست دهمی در انتقال غیرهمزمان داده ها منحصر به انتقال I/O نیست. در واقع این روش ها بطور گسترده ای و در فرصت های متعددی که انتقال داده بین دو واحد صورت می گیرد بکار می روند. در حالت کلی ما واحد انتقال دهنده را بعنوان منبع^۶ و واحد دریافت کننده را مقصد^۷ می خوانیم. مثلاً ضمن عمل خروج داده و یا نوشتن، CPU یک منبع است ولی بهنگام ورود یا خواندن نقش یک مقصد را بعهده دارد. مرسوم است که انتقال غیرهمزمان بین دو واحد مستقل را بوسیله دیاگرام زمانبندی مشخص نمایند تا بدینوسیله رابطه زمانی موجود بین سیگنال های کنترل و داده ها در گذرگاه مشخص شود. رشته سیگنال های کنترل در یک انتقال غیرهمزمان به این بستگی دارد که آغازگر انتقال، واحد منبع یا مقصد باشد.

کنترل بوسیله استروب

روش کنترل انتقال داده غیرهمزمان بوسیله سیگنال استروب از یک خط کنترل برای هر انتقال استفاده می کند. استروب ممکن است بوسیله واحد منبع و یا مقصد فعال گردد. شکل ۳-۱۱ (الف) یک انتقال با آغازگر منبع را نشان می دهد. گذرگاه داده اطلاعات دودویی را از واحد منبع به مقصد حمل

1- Synchronous

2- Asynchronous

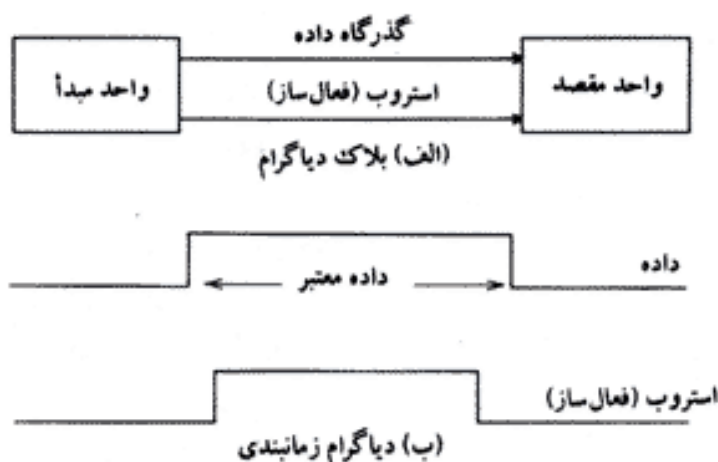
3- Strobe

4- Handshaking

5- Strobe

6- Source

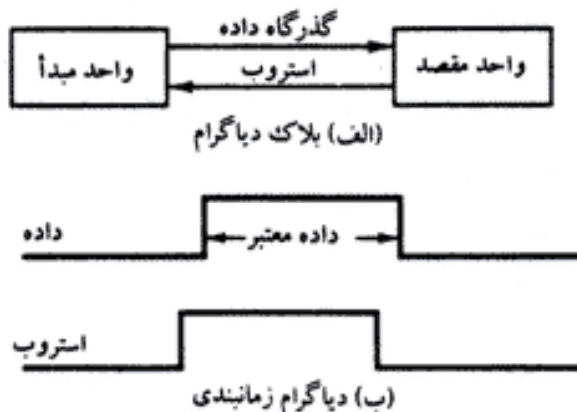
7- Destination



شکل ۱۱-۳ فعال سازی از سوی مبدأ برای انتقال

می کند. معمولاً گذرگاه برای انتقال تمام بایت یا کلمه چند خط دارد. استروب یک خط تک واحدی است که واحد مقصد را بهنگام قرار گرفتن یک داده معتبر بر روی گذرگاه، مطلع می سازد. همانطور که در دیاگرام ۱۱-۳ (ب) دیده می شود، واحد منبع ابتدا داده ها را روی گذرگاه داده قرار می دهد. پس از یک تأخیر مختصر که برای اطمینان از تثبیت مقدار داده، در یک مقدار ثابت است منبع استروب را فعال می سازد. اطلاعات روی گذرگاه داده و سیگنال استروب برای زمانی کافی در حالت فعال باقی می مانند تا به واحد مقصد اجازه دریافت داده را بدهند. اغلب واحد مقصد لبه پائین رونده پالس استروب را برای انتقال محتوای گذرگاه داده به یکی از ثبات های درونی بکار می برد. واحد منبع به فاصله کوتاهی پس از غیرفعال کردن پالس استروب، داده را از گذرگاه برمی دارد. در واقع، منبع نیازی به تغییر داده روی گذرگاه ندارد. غیرفعال شدن سیگنال استروب بیانگر این واقعیت است که گذرگاه داده دارای داده معتبر نیست. داده جدید معتبر پس از فعال شدن مجدد استروب وجود خواهد داشت.

شکل ۱۱-۴ انتقال داده ای را که توسط واحد مقصد آغاز می شود نشان می دهد. در این حالت، واحد مقصد پالس استروب را فعال نموده و منبع را برای در اختیار گذاشتن داده مطلع می نماید. واحد منبع با قرار دادن اطلاعات دودویی مورد نظر بر روی گذرگاه داده به آن پاسخ می دهد. داده باید به مدت کافی بر



شکل ۱۱-۴ استروب از سوی مقصد برای انتقال

روی گذرگاه داده قرار گیرد تا واحد مقصد آن را بپذیرد. لبه پایین رونده پالس استروب مجدداً برای تحریک ثبات مقصد بکار می‌رود. سپس واحد مقصد، منبع را غیرفعال می‌نماید. منبع نیز داده را پس از گذشت زمان از پیش تعیین شده‌ای از روی گذرگاه حذف می‌کند.

در اکثر کامپیوترها پالس استروب در واقع بوسیله پالس‌های ساعت در CPU کنترل می‌شود. CPU همواره کنترل گذرگاه‌ها را در دست دارد و به واحدهای خارجی اطلاع می‌دهد که چگونه داده‌های خود را منتقل سازند. مثلاً، استروب شکل ۳-۱۱ می‌توانست یک سیگنال کنترل نوشتن در حافظه از CPU به واحد حافظه باشد. منبع، که CPU است، کلمه‌ای را روی گذرگاه داده قرار داده و واحد حافظه را، که مقصد است مطلع می‌نماید، که این خود یک عمل نوشتن است. بطور مشابه استروب شکل ۴-۱۱ می‌توانست یک سیگنال کنترل خواندن حافظه از CPU به یک حافظه باشد. مقصد، یعنی CPU، عمل خواندن را آغاز می‌کند تا حافظه را که منبع است برای قرار دادن کلمه انتخابی روی گذرگاه داده مطلع سازد. انتقال داده بین CPU و یک واحد واسطه مشابه فوق می‌باشد. انتقال داده بین یک واسطه و یک وسیله I/O معمولاً بوسیله مجموعه‌ای از خطوط دست‌دهی کنترل می‌گردد.

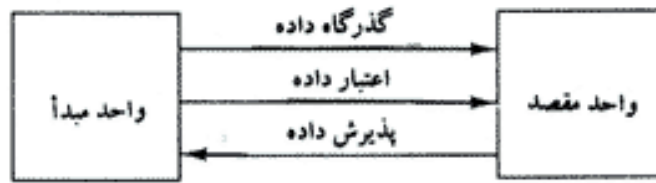
دست‌دهی

عیب روش استروب این است که واحد منبع که انتقال را شروع می‌کند قادر نیست بداند که آیا واقعاً مقصد داده را که بر روی گذرگاه داده قرار گرفته دریافت کرده است یا خیر. بطور مشابه، یک واحد مقصد که انتقال را آغاز می‌کند نمی‌داند که آیا منبع داده را روی گذرگاه قرار داده است یا نه. روش دست‌دهی این مسئله را با معرفی یک سیگنال کنترل دیگر که پاسخی برای واحد آغازگر انتقال است، حل می‌کند. اساس روش دست‌دهی دوسیمی انتقال داده بشرح زیر است. یک خط کنترل در جهت ارسال داده در گذرگاه، یعنی از منبع به مقصد است. این خط بوسیله منبع برای مطلع ساختن مقصد از وجود داده معتبر بر روی گذرگاه بکار می‌رود. خط دیگر در جهت دیگر، یعنی از مقصد به منبع می‌باشد. این خط بوسیله واحد مقصد بکار رفته و منبع را مطلع می‌سازد که آیا می‌تواند داده بپذیرد یا خیر. دنباله سیگنال‌های کنترل در طول انتقال به واحدی بستگی دارد که انتقال را آغاز می‌نماید.

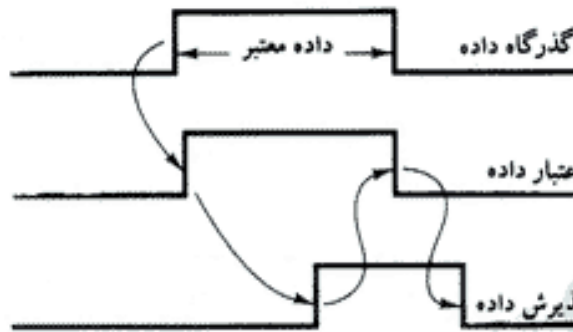
شکل ۵-۱۱ رویه انتقال داده را وقتی که بوسیله منبع آغاز شود نشان می‌دهد. دو خط دست‌دهی عبارتند از "داده معتبر"^۱ که بوسیله واحد منبع، و "پذیرش داده"^۲ که توسط واحد مقصد تولید شده‌اند. دیگرام زمانی تبادل سیگنال‌ها را بین دو واحد نشان می‌دهد. رشته وقایع ارائه شده در بخش (ج) چهار حالت ممکنه که سیستم می‌تواند در هر لحظه باشد را نشان می‌دهد. واحد منبع انتقال را با قرار دادن داده بر روی گذرگاه آغاز می‌کند و سیگنال داده معتبر را فعال می‌سازد. سیگنال پذیرش داده بوسیله واحد مقصد پس از پذیرش داده از گذرگاه فعال می‌گردد. سپس واحد منبع سیگنال داده معتبر را غیرفعال ساخته و داده را بر روی گذرگاه غیرمعتبر می‌نماید. پس از آن واحد مقصد سیگنال پذیرش داده را

1- Data valid

2- Data accepted



(الف) بلاک دیاگرام



(ب) دیاگرام زمانبندی

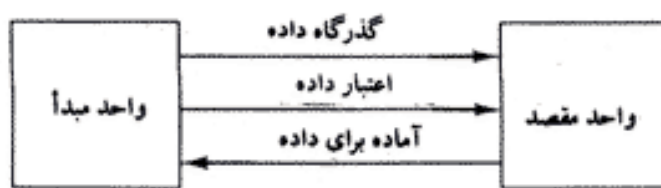


(ج) دنباله وقایع

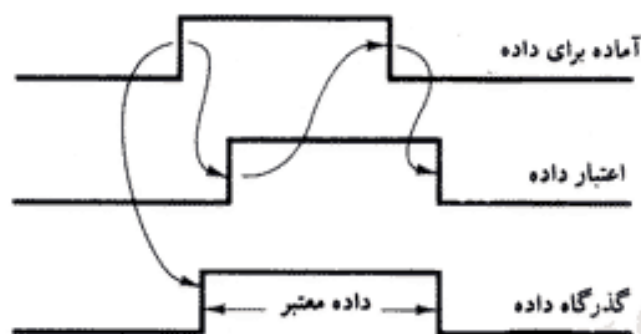
شکل ۵-۱۱ انتقال از سوی مبدأ با استفاده از دست‌دهی

غیرفعال نموده، و سیستم به حالت اولیه خود باز می‌گردد. منبع، تا هنگامی که مقصد آمادگی پذیرش داده جدید را با غیرفعال کردن سیگنال پذیرش داده انجام ندهد، داده بعدی را ارسال نمی‌کند. در این روش هر زمان تاخیری از یک حالت به حالت بعدی مجاز است و اجازه می‌دهد تا هر واحد با سرعت انتقال داده خاص خود کار کند. سرعت انتقال را کندترین واحد معین می‌کند.

انتقال آزاد شده بوسیله مقصد که بسمت روش دست‌دهی انجام می‌شود در شکل ۶-۱۱ نشان داده شده است. توجه کنید نام سیگنال تولید شده بوسیله واحد مقصد به "آماده برای داده" تغییر یافته است تا



(الف) بلاک دیاگرام



(ب) دیاگرام زمانبندی



(ج) دنباله وقایع

شکل ۶-۱۱ انتقال از سوی مقصد با استفاده از دست‌دهی

منعکس کننده مفهوم جدید باشد. واحد منبع در این حالت تا دریافت سیگنال "آماده برای داده" از واحد مقصد، داده را روی گذرگاه قرار نمی‌دهد. از این به بعد روند دستی‌گویی یکسانی را مانند آنچه در حالت آغاز از سوی منبع دنبال می‌کند. توجه کنید که اگر سیگنال آمادگی برای داده را بعنوان پذیرش داده در نظر بگیریم، رشته وقایع در هر دو مورد یکسان خواهد بود. در واقع تنها اختلاف بین انتقال آغاز با منبع و آغاز با مقصد در انتخاب حالت اولیه آنهاست. روش دستی‌دهی درجه بالایی از انعطاف و قابلیت اطمینان را فراهم می‌آورد زیرا اتمام موفق هر

انتقال داده به شرکت فعال هر دو واحد بستگی دارد. اگر یک واحد دچار اشکال باشد، انتقال داده به انجام نمی‌رسد. چنین خطایی با مکانیزمی بنام "تمام وقت"^۱ قابل تشخیص است، و طی آن اگر انتقال داده در مدت زمان از پیش تعیین شده‌ای پایان نپذیرد آلام صادر خواهد کرد. مکانیزم اتمام وقت با استفاده از یک ساعت داخلی هنگامی که واحد، یکی از سیگنال‌های کنترل دست دهی را فعال سازد اجرا می‌شود. اگر سیگنال بازگشت دست دهی در مدت زمان معینی بازنگردد، واحد رخداد خطایی را فرض خواهد کرد. سیگنال اتمام وقت برای وقفه پردازنده نیز می‌تواند مورد استفاده قرار می‌گیرد و لذا روال سرویس‌دهی مناسبی را برای رفع خطا اجرا نماید.

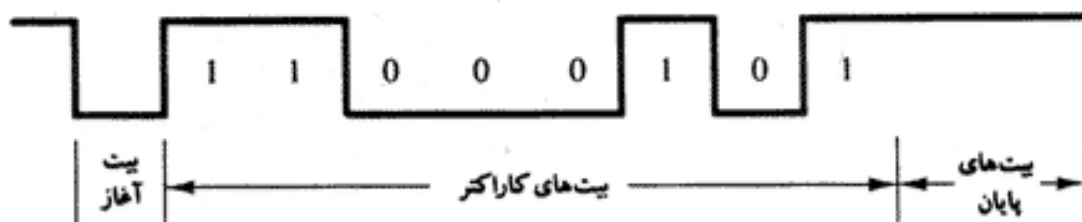
انتقال سری غیرهمزمان

انتقال داده بین دو واحد ممکن است موازی و یا سری باشد. در انتقال موازی داده، هر بیت از پیام مسیرخاص خود را داشته و کل پیام بطور همزمان انتقال می‌یابد. این بدان معنی است که یک پیام n بیتی باید از طریق n مسیرهادی جداگانه منتقل شود. در انتقال سری، هر بیت در پیام یکی پس از دیگری ارسال می‌شود. این روش مستلزم استفاده از یک جفت رسانا یا یک مسیر رسانا و یک زمین مشترک است. انتقال موازی سریعتر است ولی نیاز به چندین سیم دارد. این روش برای مسیرهای کوتاه و مواقعی که سرعت اهمیت دارد بکار می‌رود. انتقال سری کندتر است ولی ارزانتر می‌باشد زیرا فقط یک جفت رسانا نیاز دارد.

انتقال سری می‌تواند همزمان یا غیر همزمان باشد. در انتقال همزمان هر دو واحد یک فرکانس ساعت مشترک داشته و بیت‌ها مرتباً با سرعتی که بوسیله پالس‌های ساعت دیکته می‌شود انتقال می‌یابند. در انتقال سری راه دور، هر واحد بوسیله یک پالس ساعت جداگانه با فرکانس یکسان راه‌اندازی می‌شود. سیگنال‌های همزمان سازی بطور متناوب بین دو واحد ارسال می‌شوند تا ساعت‌های آنها را همگام با یکدیگر نگهدارد. در ارسال غیرهمزمان، اطلاعات دودویی فقط زمانی ارسال می‌شود که موجود باشد و هرگاه اطلاعاتی برای ارسال وجود نداشته باشد خط بی‌کار می‌ماند این برخلاف انتقال همزمان است که در آن بیتها باید بطور پیوسته ارسال شوند تا فرکانس ساعت در هر دو واحد با یکدیگر همزمان باقی بمانند. انتقال سری همزمان در بخش ۸-۱۱ بیشتر بحث شده است.

تکنیک انتقال سری غیرهمزمان داده که در اکثر پایانه‌های محاوره‌ای بکار می‌رود از بیت‌های خاصی که در دو طرف کد کاراکتر قرار داده می‌شود استفاده می‌کند. با این تفکیک، هر کاراکتر متشکل از سه بخش است: بیت شروع، بیت‌های کاراکتر، و بیت‌های توقف. قرار داد این است که وقتی کاراکتری انتقال نیابد، فرستنده در وضعیت 1 منطقی قرار می‌گیرد. اولین بیت، بیت شروع نام دارد و در وضعیت 0 قرار داشته و برای مشخص کردن شروع کاراکتر بکار می‌رود. بیت آخر که بیت توقف نامیده می‌شود همواره 1 است. مثالی از این قالب در شکل ۷-۱۱ نشان داده شده است.

1- Timeout



شکل ۷-۱۱ ارسال متوالی ناهمگام (غیرهمزمان)

کاراکتر انتقال یافته طبق قانون انتقال مشخص می شود:

- ۱- وقتی کاراکتر در حال ارسال نیست خط در 1 منطقی نگهداشته می شود.
- ۲- آغاز انتقال کاراکتر از بیت شروع که همواره 0 است مشخص می گردد.
- ۳- بیت های کاراکتر همواره بدنبال بیت شروع ارسال می گردد.
- ۴- پس از ارسال آخرین بیت کاراکتر، بیت توقف، وقتی که خط حداقل به مدت زمان یک بیت در سطح 1 باقی بماند، مشخص می گردد.

با استفاده از این قوانین، گیرنده قادر به تشخیص بیت شروع پس از انتقال خط از 1 به 0 است. ساعتی در گیرنده، خط را در زمان های مناسب واری می کند. گیرنده سرعت انتقال بیت ها و تعداد بیت های کاراکترها مورد پذیرش را می داند. پس از انتقال بیت های کاراکتر، یک یا دو بیت توقف ارسال می گردد. بیت های توقف همواره در وضعیت 1 منطقی هستند و انتهای کاراکتر را برای مشخص نمودن حالت انتظار یا بیکاری کادر بندی می نمایند.

در انتهای کاراکتر خط بمدت زمان یک یا دو بیت در حالت 1 نگهداشته می شود بطوری که فرستنده یا گیرنده می توانند مجدداً با یکدیگر همزمان شوند. طول زمان بقاء سیستم در این حالت به مقدار زمانی که برای دستگاه ها لازم است بستگی دارد. برخی پایانه های الکترومکانیکی قدیمی دو بیت توقف بکار می برند ولی پایانه های جدیدتر از یک بیت توقف استفاده می کنند. خط، تا انتقال کاراکتر بعدی در حالت 1 باقی می ماند. زمان توقف تضمین می کند که کاراکتر جدید به مدت یک یا دو برابر مدت زمان یک بیت روی خط ظاهر نخواهد شد.

بعنوان مثال، انتقال سریال به یک پایانه کم سرعت را در نظر بگیرید که سرعت انتقال آن 10 کاراکتر در ثانیه است. هر کاراکتر انتقال یافته متشکل از یک بیت شروع، هشت بیت اطلاعات، و دو بیت توقف است که جمعاً 11 بیت می شود. 10 کاراکتر در ثانیه به معنی سرعت انتقال هر کاراکتر اطلاعات در 0.1 ثانیه است. چون 11 بیت برای انتقال وجود دارد، زمان انتقال 9.09 میلی ثانیه خواهد بود. سرعت ارسال^۱ (باود) بعنوان سرعت انتقال اطلاعات تعریف شده و معادل با انتقال داده بر حسب تعداد بیت ها

1- Baud Rate

در ثانیه است. 10 کاراکتر در ثانیه با قالب 11 بیتی دارای سرعت ارسال 110 با ود است. پایانه دارای صفحه کلید و چاپگر است. هر بار کلیدی فشار داده شود، پایانه 11 بیت را بطور سری در طول مسیر ارسال می‌دارد. برای هر چاپ یک کاراکتر در یک چاپگر، یک پیام 11 بیتی بر روی سیم دیگری باید دریافت شود. واسطه پایانه از یک فرستنده و یک گیرنده تشکیل شده است. فرستنده یک کاراکتر 8 بیتی را از کامپیوتر دریافت کرده و سپس یک پیام 11 بیتی سری را به خط چاپگر می‌فرستد. گیرنده یک پیام 11 بیتی را از خط صفحه کلید دریافت و کد یک کاراکتر 8 بیتی را به کامپیوتر ارسال می‌دارد. برای ایجاد واسطه بین کامپیوتر و پایانه‌های محاوره‌ای مشابه، مدارات مجتمع خاصی طراحی شده‌اند. چنین مدارهایی مدار واسطه ارتباطی غیرهمزمان¹ یا گیرنده فرستنده عام غیرهمزمان (UART)² خوانده می‌شود.

واسطه ارتباطی غیرهمزمان³

بلاک دیاگرام مدار واسطه ارتباطی غیرهمزمان در شکل ۸-۱۱ نشان داده شده است. این واسطه هم بصورت فرستنده و هم گیرنده کار می‌کند. واسطه با استفاده از یک بایت کنترل که در ثبات کنترل بار می‌شود برای نوع خاصی از عمل انتقال راه‌اندازی اولیه می‌شود. ثبات فرستنده یک بایت داده را از طریق گذرگاه داده از CPU می‌پذیرد. این بایت به یک ثبات شیفت جهت ارسال سری منتقل می‌شود. بخش گیرنده، اطلاعات سری را با یک ثبات شیفت دیگر دریافت می‌کند و هنگامی که یک بایت کامل داده در این ثبات جای شد، آن را به ثبات گیرنده منتقل می‌کند. CPU می‌تواند ثبات گیرنده را برای خواندن بایت از طریق گذرگاه داده انتخاب کند. بیت‌های ثبات وضعیت برای پرچم‌های ورودی و خروجی و ضبط خطاهای معینی که ممکن است در حین ارسال رخ دهد بکار رود. CPU قادر است با خواندن ثبات وضعیت بیت‌های پرچم را واریسی کرده و معلوم نماید که خطایی رخ داده است یا خیر. ورودی انتخاب تراشه⁴ (CS) و کنترل‌های خواندن و نوشتن با CPU در ارتباطند. ورودی انتخاب تراشه برای انتخاب مدار واسطه از طریق گذرگاه آدرس به کار می‌رود. ورودی انتخاب ثبات (RS) به کنترل‌های خواندن (RD) و نوشتن (WR) مربوط است. دو ثبات از نوع فقط نوشتن و دو ثبات دیگر از نوع فقط خواندن هستند. همانطور که در جدول همراه دیاگرام مشخص شده است، ثبات انتخابی تابعی از مقدار RS و وضعیت RD و WR است.

عملکرد واسطه ارتباطی غیرهمزمان با ارسال یک بایت به ثبات کنترل بوسیله CPU آغاز می‌شود. روند مقداردهی اولیه، با تعریف پارامترهای معینی از قبیل سرعت ارسال⁵ مورد استفاده، تعداد بیت‌های توقف هرکاراکتر، آیا بیت توازن باید ایجاد و چک شود یا خیر، و تعداد بیت‌های هرکاراکتر، واسطه را در وضعیت کاری خاص قرار می‌دهد. دو بیت در ثبات وضعیت به عنوان پرچم بکار می‌روند. یکی از این بیت‌ها برای

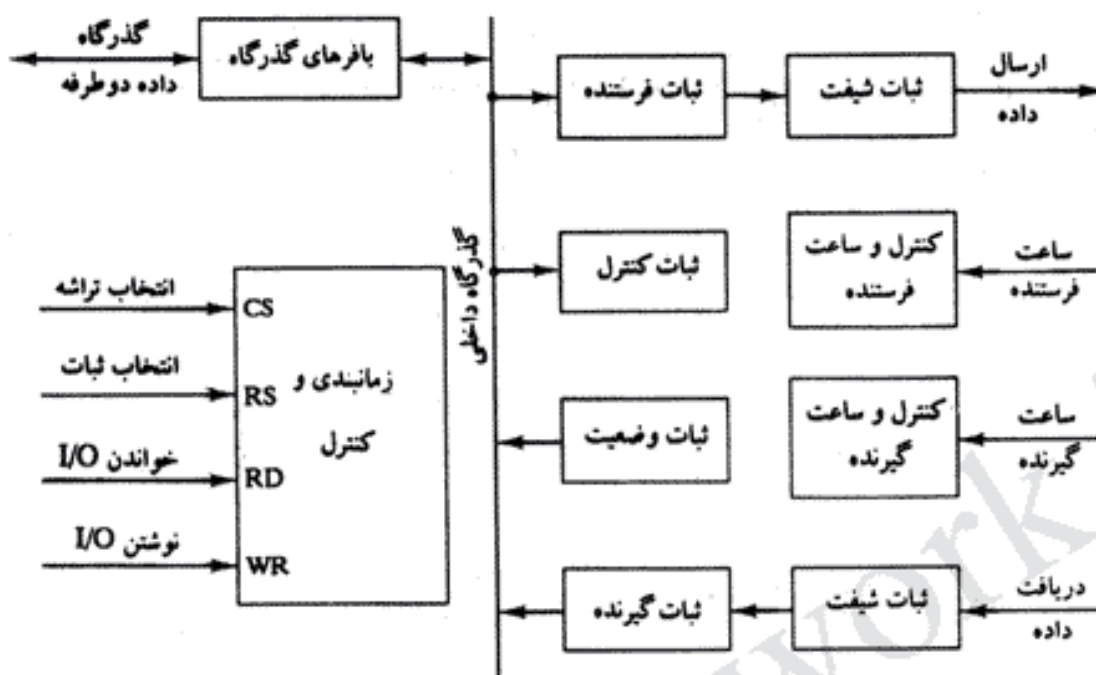
1- Asynchronous Communication Interface

2- Universal Asynchronous Receiver - Transmitter

3- Asynchronous

4- Chip Select

5- Baud rate



CS	RS	عمل	ثباتی که انتخاب می شود
0	x	x	هیچ گذرگاه داده در اهدانس بالا
1	0	WR	ثبات فرستنده
1	1	WR	ثبات کنترل
1	0	RD	ثبات گیرنده
1	1	RD	ثبات وضعیت

شکل ۸-۱۱ بلاک دیاگرام یک نوع واسطه ارتباطات غیرهمزمان (غیرهمگام)

نشان دادن خالی بودن ثبات فرستنده ثبات بودن و دیگری برای نشان دادن پر بودن ثبات گیرنده بکار می رود. عملکرد بخش فرستنده واسطه بشرح زیر است. CPU ثبات وضعیت را می خواند و پرچم ها را برای خالی بودن ثبات فرستنده چک می کند. اگر این ثبات خالی باشد، CPU یک کاراکتر را به ثبات فرستنده ارسال می نماید و مدار واسطه پرچم را صفر می کند تا نشان دهد که ثبات پر است. اولین بیت ثبات فرستنده 0 می شود تا شروع را ایجاد کند. کاراکتر بطور موازی از ثبات فرستنده به ثبات شیفت منتقل می شود و تعداد مناسبی بیت های توقف که از پیش تعیین شده است به ثبات شیفت ملحق می گردد. سپس ثبات فرستنده بعنوان یک ثبات خالی تلقی می شود. حال کاراکتر می تواند بصورت هر بار یک بیت با جایجایی داده های موجود در ثبات شیفت با سرعت انتقال مشخص شده ارسال شود. CPU می تواند پس از واریسی پرچم در ثبات وضعیت، کاراکتر دیگری را به ثبات فرستنده انتقال دهد. در این

حالت گوئیم مدار واسطه بصورت دوبله بافرشده¹ است. زیرا به محض اینکه کاراکتر قبلی شروع به ارسال شود کاراکتر جدید می تواند در ثبات فرستنده بار شود.

عملکرد بخش گیرنده مدار واسطه مشابه فوق است. وقتی که خط بیکار است ورودی دریافت داده در وضعیت 1 منطقی قرار می گیرد. واحد کنترل گیرنده بر خط دریافت داده نظارت می کند تا با مشاهده سیگنال 0 وقوع بیت شروع را شناسایی کند. پس از مشاهده بیت شروع، بیت های کاراکتر که متعاقباً فرا می رسند با سرعت ارسال از پیش تعیین شده وارد ثبات شیفت گیرنده می شوند. پس از دریافت بیت های داده، مدار واسطه، توازن و بیت های توقف را چک می کند. سپس بیت های کاراکتر بدون بیت های شروع و توقف بطور موازی از ثبات شیفت به ثبات گیرنده انتقال می یابند. پرچم مربوطه در ثبات وضعیت 1 می شود تا نشان دهنده پر بودن ثبات گیرنده باشد. CPU ثبات وضعیت را خوانده و پرچم را واریسی می کند و در صورت 1 بودن آن، داده را از ثبات گیرنده می خواند.

در حین انتقال، مدار واسطه هر خطای ممکن را چک نموده و بیت مربوطه را در ثبات وضعیت 1 می کند. در هر زمان که خطا رخ می دهد CPU می تواند ثبات وضعیت را واریسی کند. سه خطای ممکن که مدار واسطه در طول ارسال واریسی می کند عبارتند از خطای توازن²، خطای کادربندی³ و خطای تزام. خطای توازن هنگامی رخ می دهد که تعداد 1 در داده های رسیده دارای توازن صحیحی نیست. خطای کادربندی هنگامی بوقوع می پیوندد که تعداد بیت های توقف در انتهای کاراکتر رسیده صحیح نباشد. خطای تزام در صورتی رخ می دهد که قبل از خواندن کاراکتر از ثبات گیرنده بوسیله CPU، کاراکتر بعدی در ثبات شیفت قرار گیرد. این خطا موجب از دست رفتن کاراکتر از رشته داده های دریافتی می شود.

بافر اولین ورودی - اولین خروجی

بافر اولین ورودی - اولین خروجی (FIFO) واحد حافظه ای است که اطلاعات را به گونه ای ذخیره می کند که اولین داده وارد شده اولین داده ای خواهد بود که خارج شود. بافر FIFO پایانه های ورودی - خروجی جداگانه ای دارد. ویژگی مهم این نوع بافر اینست که می تواند ورود و خروج داده ها را با دو سرعت متفاوت انجام دهد و داده های خروجی همیشه بهمان ترتیبی که وارد می شوند خواهند بود. هر گاه FIFO بین دو واحد قرار گیرد، قادر خواهد بود داده ها را با سرعتی از واحد مبدأ دریافت و با سرعتی دیگر به واحد مقصد تحویل دهد. اگر واحد منبع کندتر از واحد مقصد باشد، می توان بافر را با سرعت کم در بافر پر کرد و بعداً با سرعت بیشتری آن را تخلیه نمود. اگر منبع سریعتر از مقصد باشد، FIFO برای مواقعی مفید خواهد بود که داده های ورودی بیکباره فرا برسند و میانگیر را پر کنند ولی بین هر دو بار دریافت داده زمان به حد کافی برای واحد مقصد وجود دارد تا همه یا قسمتی از اطلاعات را از بافر تخلیه کند. بنابراین بافر FIFO می تواند در برخی از کاربردها مفید باشد و آن هنگامی است که داده ها بطور غیرهمزمان انتقال یابند. این بافر داده ها را که به تدریج فرا می رسند انباشته کرده و آنها را هنگام لزوم با

1- Double Buffered

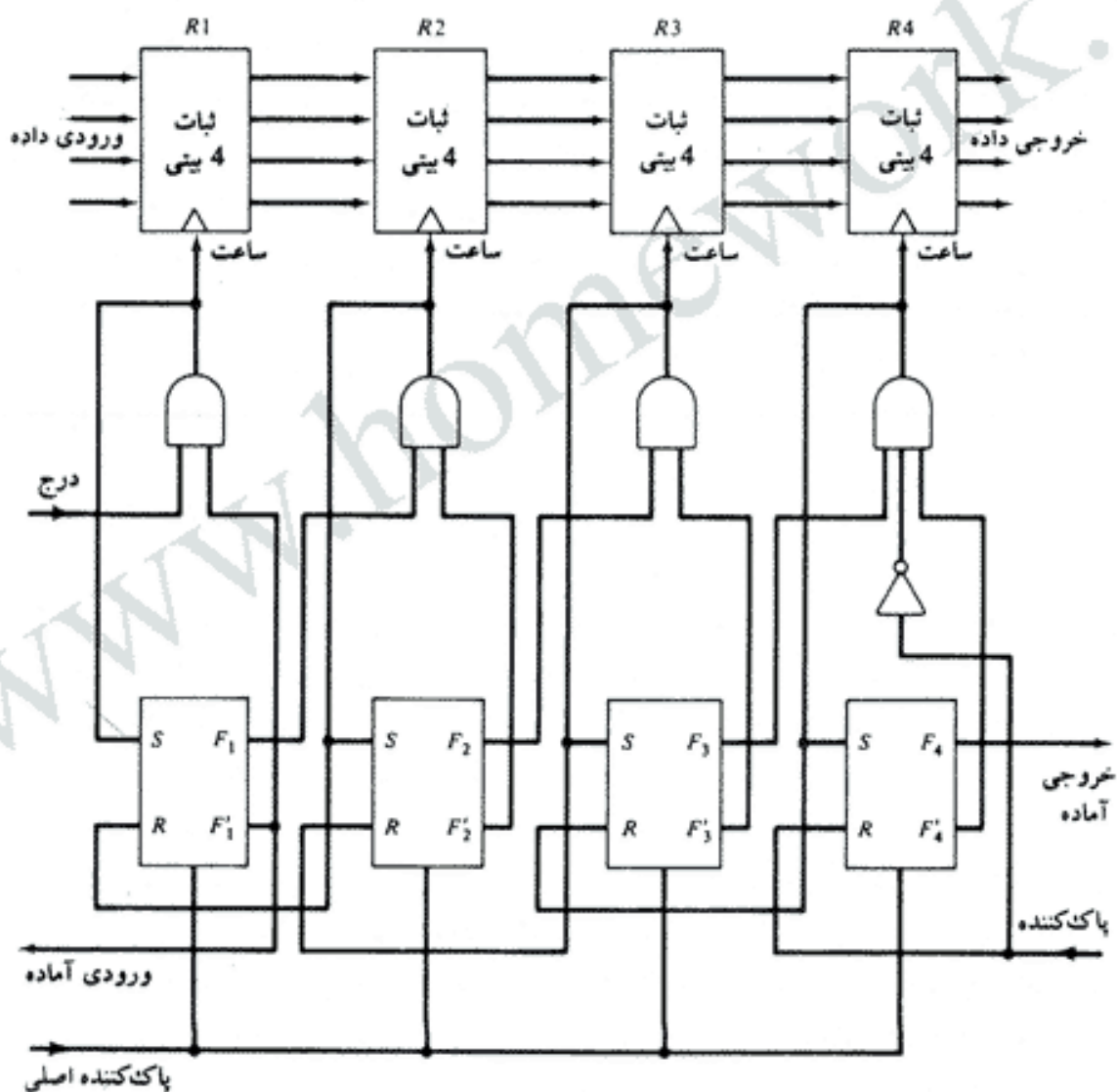
2- Parity

3- Framing error

همان ترتیبی که رسیده اند تحویل می دهد.

دیاگرام منطقی یک بافر FIFO نمونه با ظرفیت 4×4 در شکل ۹-۱۱ نشان داده شده است. این بافر متشکل از چهار ثبت چهار بیت RI با $I=1,2,3,4$ و یک ثبت کنترل با فلیپ فلاپ های F_i ، $i=1,2,3,4$ یعنی هر فلیپ فلاپ برای یک ثبت، می باشد. FIFO می تواند چهار کلمه چهار بیتی را ذخیره کند. تعداد بیت ها در هر کلمه می تواند با افزایش تعداد بیت ها در هر ثبت افزایش یابد و تعداد کلمات نیز با افزایش تعداد ثبت ها اضافه می گردد.

فلیپ فلاپ F_i در ثبت کنترل که با 1 بار شده است نشان می دهد که یک کلمه چهار بیتی در ثبت



شکل ۹-۱۱ دیاگرام مدار بافر FIFO با اندازه 4×4

مربوطه اش RI ذخیره شده است. وجود 0 در F_i بیانگر این حقیقت است که داده معتبری در ثبات مربوطه وجود ندارد. ثبات کنترل حرکت داده ها را در طول ثبات ها هدایت می نماید. هر وقت بیت F_i در ثبات کنترل 1 باشد ($F_i=1$) و بیت F_{i+1} پاک شده باشد ($F'_{i+1}=1$) یک پالس ساعت تولید شده و موجب می شود که ثبات $R(I+1)$ داده را از ثبات RI بپذیرد. همان گذرپالس ساعت F_{i+1} را F_i و F_i را 0 می کند. این وضعیت موجب می شود که پرچم کنترل یک مکان به راست همراه با داده حرکت کند. داده های موجود در ثبات ها تا جایی که فضای خالی در جلو آنها وجود داشته باشد به پائین FIFO و به سمت خروجی می روند. این حرکت موجی هنگامی که داده به ثباتی مانند RI برسد و فلیپ فلاپ بعدی آن یعنی F_{i+1} مقدار 1 را داشته باشد، یا به آخرین ثبات R_4 برسد، متوقف می گردد. یک پاک کننده اصلی برای پاک کردن کلیه فلیپ فلاپ های ثبات کنترل بکار می رود.

داده ها بشرطی به داخل بافر وارد می شوند که سیگنال "ورودی آماده" فعال شده باشد. این هنگامی رخ می دهد که اولین فلیپ فلاپ کنترل، F_1 پاک شده باشد که بیانگر خالی بودن R_1 است. داده ها از طریق خطوط ورودی با فعال کردن ساعت در پایه درج^۲ بار می شوند. همان پالس ساعت F_1 را 1 می نماید که این خود کنترل ورودی آماده^۳ را غیرفعال نموده و این بنوبه خود بیانگر آنست که FIFO در حال حاضر مشغول و آماده پذیرش داده بیشتر نیست. فرآیند حرکت موجی بشرطی آغاز می شود که R_2 خالی باشد. داده از R_1 به R_2 منتقل و F_1 پاک می شود. این امر باعث فعال شدن ورودی آماده گشته و بیانگر آنست که ورودی ها برای کلمه داده دیگر آماده اند. اگر FIFO پریاشد، F_1 در حالت 1 باقی مانده ولی خط ورودی آماده در حالت 0 باقی می ماند. توجه کنید که دو خط کنترل ورودی آماده و درج یک جفت خط دست دهی را برای حالتی که مقصد آغازگر است تشکیل می دهد.

داده هایی که بداخل ثبات ها می افتند در سمت خروجی جمع می شوند. خط کنترل "خروجی آماده"^۴ هنگامی که آخرین فلیپ فلاپ F_4 با 1 بار شود فعال می گردد که این مشخص کننده وجود داده معتبر در ثبات خروجی R_4 است. داده خروجی از R_4 بوسیله واحد مقصد پذیرفته شده که این سیگنال کنترل پاک کننده^۵ را فعال می سازد. این بنوبه خود F_4 را پاک کرده و سبب غیرفعال شدن "خروجی آماده" می گردد. اگر FIFO خالی باشد، داده ای از R_3 وجود نداشته و F_4 در وضعیت 0 باقی می ماند. توجه کنید که دو خط کنترل "خروجی آماده" و "پاک کننده" تشکیل خطوط دست دهی برای منبع آغازگر را می دهند.

۴-۱۱ شیوه های انتقال

اطلاعات دودویی دریافتی از وسایل خارجی معمولاً برای پردازش بعدی در حافظه قرار می گیرند و اطلاعات منتقله از کامپیوتر مرکزی به وسیله خارجی در واقع از حافظه سرچشمه می گیرند. CPU صرفاً

1- Clock Transition

2- Insert

3- Input ready

4- Output ready

5- Delete

دستورالعمل های I/O را اجرا نموده و ممکن است داده را بطور موقت بپذیرد، ولی مبدا یا مقصد نهایی حافظه است. انتقال داده بین کامپیوتر مرکزی و وسایل I/O می تواند با شیوه های مختلفی انجام شود. برخی روشها از CPU بعنوان مسیر میانی استفاده می کنند، بعضی مستقیماً داده را به و یا از حافظه منتقل می سازند. تبادل داده با وسایل جانبی به سه طریق زیر امکان پذیر است.

۱- I/O برنامه نویسی شده

۲- I/O بطریقه وقفه

۳- دستیابی مستقیم به حافظه (DMA)

عملیات I/O برنامه نویسی شده نتیجه دستورالعمل های I/O نوشته شده در برنامه کامپیوتر است. انتقال هر قلم داده از سوی دستورالعملی در برنامه آغاز می شود. معمولاً تبادل داده بین یک ثبات CPU و وسیله جانبی است. دستورات دیگری هم برای تبادل داده بین CPU و حافظه مورد نیازند. انتقال داده تحت کنترل برنامه مستلزم نظارت مداوم CPU بر دستگاه جانبی است. به محض آغاز انتقال، CPU باید بر مدار واسطه نظارت کند تا ببیند چه وقت می تواند دوباره انتقال را انجام دهد. در واقع این دیگر بعهدہ دستورات اجرا شده برنامه در CPU است تا دقیقاً بر هر اتفاقی که در واحد واسطه و وسیله I/O رخ می دهد نظارت کند.

در روش I/O برنامه نویسی شده، CPU در یک حلقه از برنامه باقی می ماند تا واحد I/O مشخص نماید که برای انتقال داده آماده است. این فرآیند موجب اتلاف قابل توجه وقت می شود زیرا پردازنده را بی آنکه لازم باشد مشغول نگه می دارد. برای احتراز از اتلاف وقت می توان از امکانات وقفه^۱ و فرمانهای خاصی برای اطلاع به مدار واسطه استفاده کرده و از آن خواست تا بهنگام وجود داده از طرف وسیله جانبی یک سیگنال تقاضای وقفه^۲ را صادر کند. در خلال این مدت مدار واسطه وسیله جانبی را زیر نظر دارد. وقتی مدار واسطه مشخص کند که وسیله جانبی آماده انتقال داده است، یک سیگنال تقاضای وقفه را به کامپیوتر صادر می کند. به محض تشخیص سیگنال وقفه خارجی، CPU موقتاً کار خود را متوقف کرده، به برنامه سرویس دهی برای پردازش انتقال I/O انشعاب نموده، و سپس به کاری که در اصل در حال انجام بود باز می گردد.

انتقال داده بروش I/O برنامه نویسی شده بین CPU و دستگاه جانبی انجام می شود. در دستیابی مستقیم به حافظه^۳ (DMA)، مدار واسطه داده را از طریق گذرگاه حافظه تبادل می کند. CPU انتقال را با تهیه آدرس شروع برای واسطه و تعداد کلمات لازم برای انتقال آغاز نموده و سپس برای انجام سایر کارها پیشروی می نماید. وقتی که انتقال پایان یافت، DMA سیکل حافظه را از طریق گذرگاه حافظه تقاضا کند. وقتی که تقاضا بوسیله کنترل کننده حافظه پذیرفته شد، DMA داده ها را مستقیماً به حافظه می فرستد. CPU فقط عمل مراجعه به حافظه خود را به تأخیر می اندازد تا انتقال مستقیم I/O به حافظه

1- Interrupt

2- Interrupt request signal

3- Direct Memory Access

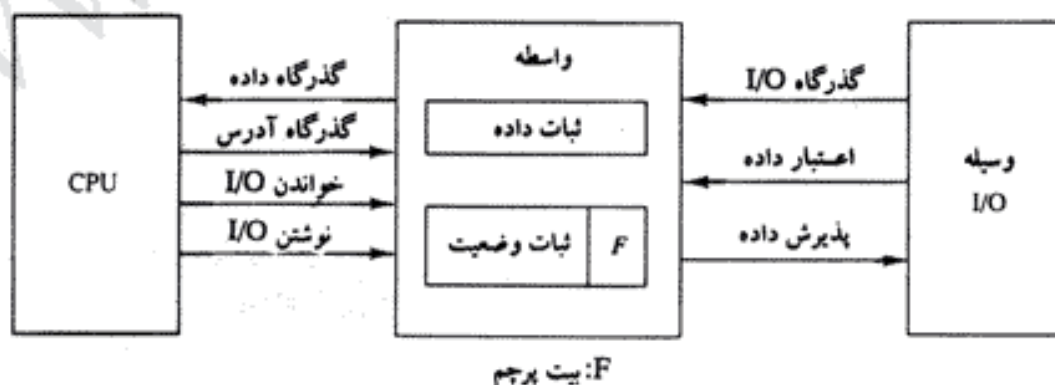
اجرا شود. چون معمولاً سرعت وسیله جانبی کم تر از سرعت پردازنده است، انتقال I/O - حافظه در مقایسه با مراجعات پردازنده به حافظه بندرت رخ می دهد. انتقال DMA در بخش ۶-۱۱ مفصل تر بحث شده است.

در بسیاری از کامپیوترها، مدار واسطه لازم برای ارجاع مستقیم به حافظه را در یک واحد که آن را پردازنده I/O (یا IOP) می نامند قرار می دهد. IOP می تواند بسیاری از وسایل جانبی را از طریق یک DMA و امکان وقفه کنترل نماید. در یک چنین سیستمی، کامپیوتر به سه ماژول جداگانه تقسیم می شود: واحد حافظه، CPU، و IOP. پردازشگر I/O در بخش ۷-۱۱ ارائه شده است.

مثال I/O برنامه نویسی شده

در روش I/O برنامه نویسی شده، وسیله I/O دسترسی مستقیمی به حافظه ندارد. انتقال از یک وسیله I/O به حافظه مستلزم اجرای چندین دستور بوسیله CPU، از جمله یک دستور ورودی برای انتقال داده از وسیله به CPU و یک دستور ذخیره سازی برای انتقال داده از CPU به حافظه است. دستورات عمل های دیگری نیز ممکن است برای تحقیق اینکه وسیله جانبی داده را در اختیار گذاشته یا نه و شمارش تعداد کلمات انتقال یافته لازم باشد.

مثالی از یک وسیله I/O به CPU از طریق واسطه در شکل ۱۰-۱۱ نشان داده شده است. وسیله بایت های داده را یک به یک همانطور که در اختیار قرار می گیرند انتقال می دهد. وقتی که یک بایت در اختیار قرار گرفت، وسیله آن را روی گذرگاه I/O قرار می دهد و خط داده معتبر^۲ را فعال می سازد. مدار واسطه بایت را پذیرفته و در ثبات داده خود قرار می دهد و خط پذیرش داده^۳ را فعال می نماید. مدار واسطه بیتی را در ثبات وضعیت 1 می نماید که ما آن را بیت F یا پرچم^۴ می خوانیم. در این لحظه وسیله جانبی می تواند خط داده معتبر را غیر فعال نماید ولی بایت دیگری را، تا غیر فعال شدن خط پذیرش داده



شکل ۱۰-۱۱ انتقال داده از وسیله I/O به CPU

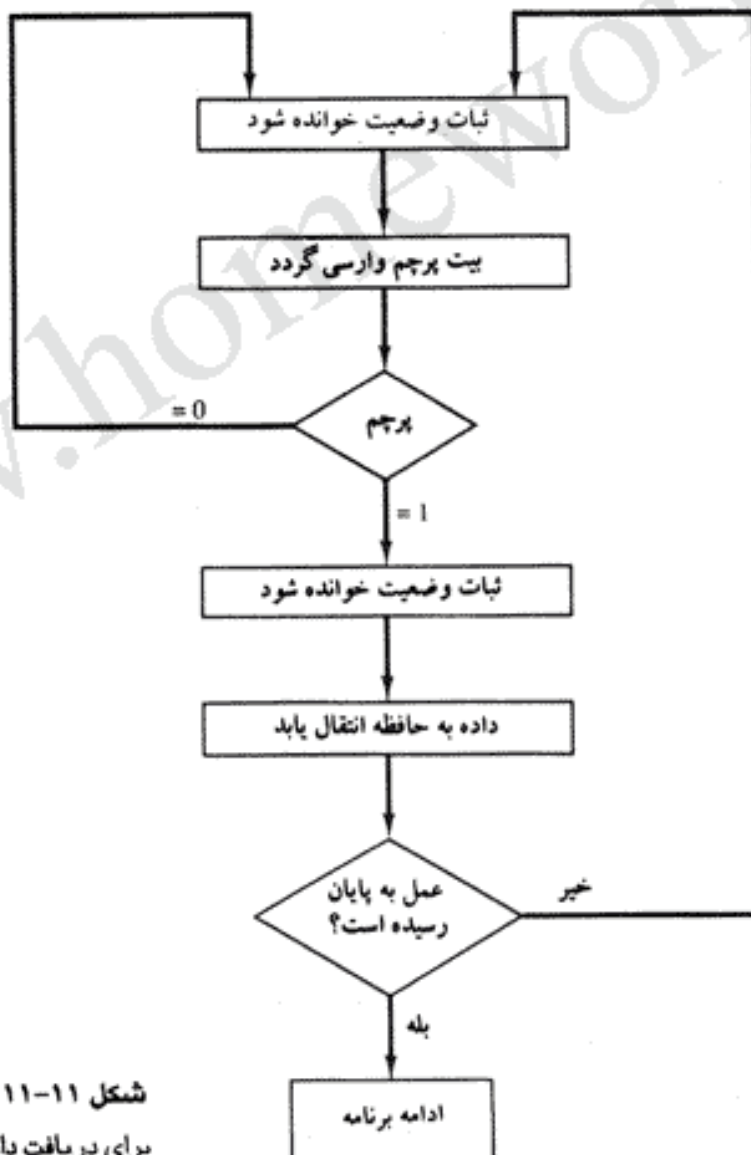
1- I/O Processor

2- Data Valid

3- Data Accepted

4- Flag

بوسیله مدار واسطه، منتقل نخواهد کرد. این عملیات بر طبق روند دست دهی مبتنی بر شکل ۱۱-۵ است. برنامه‌ای برای کامپیوتر نوشته شده است تا پرچم را در ثبات وضعیت واریسی نموده و تعیین کند که آیا وسیله I/O بایستی در ثبات داده قرار داده است یا خیر. این عمل با خواندن ثبات وضعیت و قرار دادن آن در یک ثبات CPU و واریسی بیت پرچم صورت می‌گیرد. اگر پرچم برابر 1 باشد، CPU داده را از ثبات داده می‌خواند. سپس بیت پرچم با 0 شدن بوسیله CPU و یا مدار واسطه، بسته به اینکه مدار واسطه چگونه طراحی شده باشد، پاک می‌شود. به محض پاک شدن پرچم، مدار واسطه خط پذیرش داده را غیرفعال کرده و بنابراین وسیله جانبی می‌تواند بایت داده بعدی را منتقل سازد. فلوچارت برنامه‌ای که باید برای CPU نوشته شود. در شکل ۱۱-۱۱ کشیده شده است. در این چارت فرض شده است که وسیله جانبی رشته‌ای از بایت‌هایی که باید در حافظه ذخیره شوند را ارسال



شکل ۱۱-۱۱ فلوچارت برنامه CPU
برای دریافت داده‌های ورودی

می نماید. انتقال هر بایت نیاز به سه دستورالعمل زیر دارد.

- ۱- خواندن ثبات وضعیت
- ۲- چک کردن بیت پرچم در ثبات وضعیت و انشعاب به مرحله 1 اگر این بیت 1 نباشد، و یا به مرحله 3 اگر بیت مذکور 1 باشد.
- ۳- خواندن ثبات داده.

هر بایت بداخل یک ثبات پردازنده خوانده شده و سپس با یک دستورالعمل ذخیره به حافظه منتقل می شود. یک کار متداول در برنامه نویسی I/O انتقال بلاکی از کلمات از وسیله I/O و ذخیره آنها در بافر حافظه است. برنامه ای که کاراکترهای ورودی را در یک بافر حافظه با استفاده از دستورات فصل ۶ تعریف می نماید در جدول ۲۱-۶ لیست شده است.

روش I/O برنامه نویسی شده خصوصاً در کامپیوترهای کوچک با سرعت کم یا در سیستم هایی مختص نظارت مداوم بریک وسیله مفید است. این نوع انتقال بدلیل تفاوت سرعت انتقال اطلاعات بین CPU و وسیله I/O روش کارآیی نیست. برای اثبات این عدم کارآیی، یک کامپیوتر نوعی را در نظر بگیرید که می تواند دو دستورالعملی که ثبات وضعیت را می خواند و پرچم را واری می کند در یک میکروثانه اجرا نماید. فرض کنید که وسیله ورودی داده های خود را با سرعت 100 بایت در ثانیه منتقل نماید. این سرعت معادل یک بایت در هر 10 هزار میکروثانه است. در نتیجه CPU بین هر دو بار انتقال 10 هزار بار پرچم را واری می کند. CPU هنگام واری پرچم بجای انجام یک کار پردازشی مفید دیگر، در واقع در حال اتلاف وقت است.

I/O بکممک وقفه

راه دیگری علاوه بر نظارت مداوم CPU بر پرچم، ایجاد این امکان است که هر گاه واسطه آماده انتقال داده بود به کامپیوتر اطلاع دهد. این روش از انتقال از امکان وقفه استفاده می کند. CPU مادامی که مشغول اجرای یک برنامه است پرچم را واری نمی کند. با این وجود، هنگامی که پرچم 1 شده باشد، یک وقفه کامپیوتر را لحظه ای از اجرای برنامه جاری متوقف ساخته و 1 شدن پرچم را به آن اطلاع می دهد. CPU از آنچه در حال انجام آن است منصرف شده و به انتقال ورودی یا خروجی می پردازد. پس از تکمیل انتقال، کامپیوتر به برنامه اصلی برمی گردد تا آنچه را که قبل از وقفه انجام می داد ادامه دهد. CPU در پاسخ به سیگنال وقفه، آدرس بازگشت را از شمارنده در پشته ای در حافظه ذخیره می کند و سپس کنترل به یک روال سرویس که انتقال I/O مطلوب را پردازش می کند، انشعاب می نماید.

طریقی که CPU برای انتخاب آدرس انشعاب روال سرویس برمی گزیند از یک نوع به نوعی دیگر فرق می کند. در اصل برای انجام این عمل دو راه وجود دارد. یک راه وقفه برداری^۱ و دیگری وقفه

1- Vectored Interrupt

غیربرداری^۱ نام دارد. در وقفه غیربرداری، آدرس انشعاب به مکان ثابتی از حافظه تخصیص می‌یابد. در وقفه برداری منبعی که وقفه را ایجاد می‌کند اطلاعات انشعاب را نیز برای کامپیوتر فراهم می‌آورد. این اطلاعات بردار وقفه^۲ نامیده می‌شود. در بعضی از کامپیوترها بردار وقفه آدرسی است که به مکانی از حافظه که روال سرویس I/O است اشاره می‌کند. سیستمی با وقفه برداری در بخش ۵-۱۱ آورده شده است.

ملاحظات نرم افزاری

بحث قبلی به سخت افزار پایه مورد نیاز برای ارتباط و وسایل I/O به کامپیوتر مربوط بود. کامپیوتر باید یک سری روال‌های نرم افزاری هم برای کنترل وسایل جانبی و انتقال داده‌ها بین پردازنده و وسایل جانبی داشته باشد. روال‌های I/O باید فرمان‌های کنترل را برای فعال کردن وسایل جانبی برای واری و وضعیت و آگاهی از لحظه‌ای که آماده انتقال داده هستند صادر کنند. به محض آمادگی، اطلاعات قلم به قلم تا انتقال کامل آنها منتقل می‌شود. در بعضی موارد یک فرمان کنترلی صادر می‌شود تا عملی مثل توقف نوار یا چاپ کاراکترها انجام گیرد. غالباً همراه با انتقال، چک کردن خطاها و سایر مراحل مفید دیگر نیز انجام می‌شود. در انتقال‌های کنترل شده با وقفه، نرم افزار I/O باید فرمانهایی را به دستگاه جانبی ارسال کند تا هر وقت آماده بود وقفه را صادر نماید و هر وقت که وقفه صادر شده خود وقفه را سرویس دهد. در انتقال DMA، نرم افزار I/O باید کانال DMA را برای آغاز عمل آن راه اندازی کند.

کنترل نرم افزاری تجهیزات ورودی - خروجی کاری پیچیده است. باین دلیل روال‌های I/O برای دستگاه‌های جانبی استاندارد توسط سازنده بعنوان بخشی از سیستم کامپیوتری در اختیار گذاشته می‌شود. این روال‌ها معمولاً همراه با سیستم عرضه می‌شود. اکثر سیستم‌های عامل انواع متنوعی از برنامه‌های I/O برای پشتیبانی خط خاص دستگاه‌های جانبی برای کامپیوتر را به همراه دارند. روال‌های I/O معمولاً بصورت رویه‌های سیستم عامل در اختیار قرار دارند و کاربر بدون اینکه وارد برنامه‌های زبان ماشین شود با ارجاع به روال‌های مربوطه نوع انتقال مطلوب را مشخص می‌کند.

۵-۱۱ وقفه اولویت دار

انتقال داده بین CPU و وسیله I/O بوسیله CPU شروع می‌شود. با این وجود CPU نمی‌تواند تا آمادگی وسیله برای ارتباط، انتقال را شروع نماید. آمادگی وسیله جانبی توسط یک سیگنال وقفه معین می‌شود. CPU به درخواست وقفه با ذخیره کردن آدرس بازگشت از PC بداخل حافظه‌ای در پشته^۳ پاسخ داده و سپس برنامه به روال سرویس که انتقال مورد نظر را انجام می‌دهد انشعاب می‌کند. همانطور که در بخش ۷-۸ بحث شد، بعضی از پردازنده‌ها PSW^۴ جاری را به پشته منتقل می‌کنند و یک PSW جدیدی را نیز برای روال سرویس پار می‌نمایند. در اینجا ما از PSW چشم‌پوشی می‌کنیم تا بحث وقفه

1- Nonvectored Interrupt

2- Interrupt vector

3- Stack

4- Program Status Word (PSW)

I/O پیچیده نشود.

در یک کاربرد نوعی، تعدادی از وسایل I/O به کامپیوتر متصلند، و هر یک نیز قادر است یک تقاضای وقفه را ارسال دارد. اولین کار سیستم وقفه شناخت منبع وقفه است. همچنین امکان درخواست وقفه همزمان از طرف چندین وسیله نیز وجود دارد. در این حالت سیستم باید تصمیم بگیرد که ابتدا به کدام وسیله سرویس دهد.

وقفه اولویت دار سیستمی است که اولویت را برای انواع منابع ایجاد می کند تا بدین ترتیب معین کند کدام یک از چند تقاضای همزمان ابتدا سرویس دهی شود. سیستم ممکن است معین کند که وقتی کامپیوتر مشغول سرویس دادن به وقفه است کدام وقفه دیگر می تواند کامپیوتر در حال اجرا را متوقف نماید. سطوح بالاتر وقفه معمولاً به تقاضاهایی اختصاص می یابد که اگر تأخیر یا وقفه ای در آنها ایجاد شود موجب عواقب وخیمی گردد. به وسایلی که دارای سرعت انتقال بالایی هستند، مانند دیسک های مغناطیسی، اولویت بالاتری داده می شود، و وسایل کند مانند صفحه کلید اولویت پائین تری را دارا می باشند. وقتی دو وسیله بطور همزمان کامپیوتر را متوقف کنند، کامپیوتر اول به اولویت بالاتر پاسخ می دهد.

ایجاد اولویت به وقفه های همزمان می تواند بوسیله نرم افزار یا سخت افزار انجام گیرد. برای شناخت منبعی با بالاترین اولویت بطریقه نرم افزاری، از رویه همه پرسی¹ استفاده می شود. در این روش، یک آدرس انشعاب مشترک برای تمام وقفه ها وجود دارد. برنامه ای که مراقبت از وقفه را بعهده دارد از این آدرس شروع شده و به همه منابع بطور متوالی مراجعه می کند. ترتیبی که این وقفه ها تست می شوند تعیین کننده اولویت هر وقفه است. منبعی که بالاترین اولویت را دارد اول تست می شود، و اگر سیگنال وقفه آن فعال باشد کنترل به روال سرویس این منبع انشعاب می کند. در غیر این صورت منبعی که اولویت پائین تر را دارد تست می شود و الی آخر. بنابراین روال سرویس دهی اولیه متشکل از برنامه ای است که منابع وقفه را بترتیب تست می کند و به یکی از چند روال سرویس موجود انشعاب می نماید. روال سرویس دهی که اجرا می شود متعلق به وسیله ای است که بالاترین اولویت را در بین همه وسیله هائی که به کامپیوتر وقفه ارسال داشته اند داراست. عیب روش نرم افزاری این است که اگر وقفه های بسیاری وجود داشته باشد، ممکن است زمان لازم برای مراجعه و همه پرسی از آنها از زمان سرویس دهی به وسیله I/O تجاوز کند. در این شرایط می توان از یک دستگاه وقفه اولویت دار سخت افزاری برای تسریع عمل استفاده کرد.

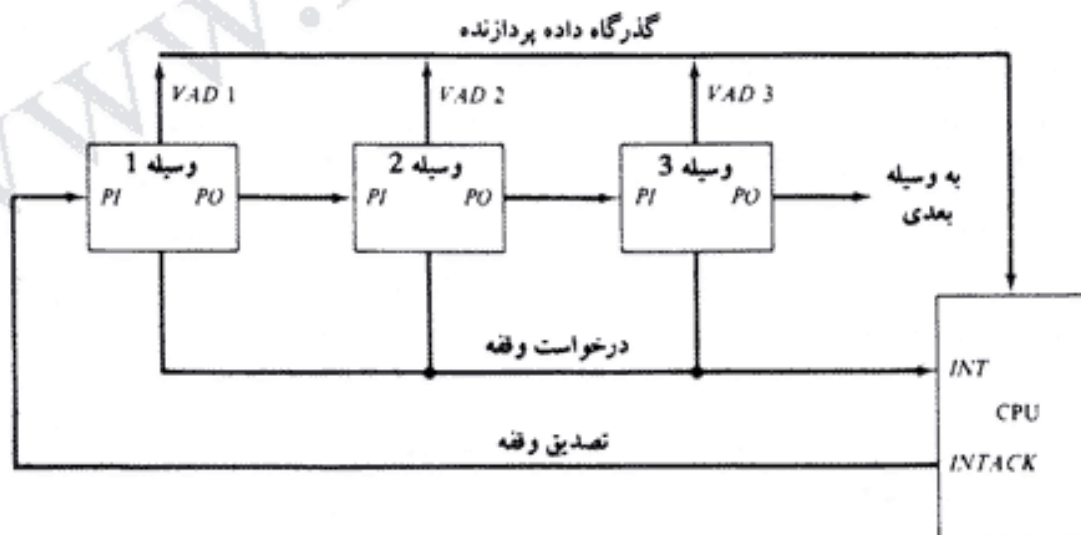
دستگاه وقفه اولویت دار سخت افزاری بصورت مدیر کل در سیستم وقفه عمل می کند. این دستگاه درخواست های وقفه را از چندین منبع دریافت و سپس تعیین می کند که کدام تقاضا دارای بالاترین اولویت است. آنگاه بر مبنای آن یک درخواست وقفه را به کامپیوتر صادر می نماید. برای تسریع عمل، هر منبع وقفه بردار وقفه خاص خود را برای دستیابی مستقیم به روال سرویس مخصوص آن داراست. به

1- Polling Procedure

این ترتیب همه پرسى لازم نیست زیرا همه تصمیمها توسط دستگاه وقفه اولویت دار سخت افزاری اتخاذ می‌گردد. عمل اولویت دهی سخت افزاری را می‌توان با اتصال سری یا موازی خطوط وقفه انجام داد. روش اتصال سری، اتصال زنجیره‌ای^۱ نیز خوانده می‌شود.

اولویت زنجیره‌ای

روش زنجیره‌ای برای اولویت دهی بر اتصال سری وسیله‌هایی که درخواست وقفه ارسال می‌دارند مبتنی است. وسیله‌ای که بالاترین اولویت را داراست در اولین مکان قرار داده می‌شود، و بدنبال آن وسیله‌های دارای اولویت پائین‌تر، تا وسیله‌ای که پائین‌ترین اولویت را دارد و در انتهای زنجیر قرار گرفته، آمده‌اند. این روش اتصال در شکل ۱۱-۱۲ برای سه وسیله و CPU نشان داده شده است. خط درخواست وقفه برای همه وسیله‌ها غیرمشترک بوده و یک اتصال منطقی سیمی^۲ را تشکیل می‌دهد. اگر هر وسیله سیگنال وقفه خود را در سطح پائین ببرد، خط وقفه به سطح پائین رفته و ورودی وقفه را در CPU فعال می‌سازد. وقتی که هیچ وقفه‌ای رخ ندهد، خط وقفه در سطح بالا باقی مانده و هیچ وقفه‌ای بوسیله CPU تشخیص داده نمی‌شود. این وضعیت معادل OR با منطق منفی است. CPU با فعال کردن خط تصدیق^۳ وقفه به تقاضای وقفه پاسخ می‌دهد. این سیگنال را وسیله شماره ۱ در ورودی خود، PI، (ورودی اولویت) دریافت می‌کند. در صورتی که وسیله ۱ درخواست وقفه نداده باشد سیگنال تصدیق از خروجی PO (خروجی اولویت) به وسیله بعدی منتقل می‌گردد. اگر وسیله ۱ درخواست وقفه داده باشد، با قرار دادن مقدار ۰ روی خروجی PO مانع از رسیدن سیگنال تصدیق به وسیله بعدی می‌شود.



شکل ۱۱-۱۲ سیستم وقفه اولویت‌دار زنجیره‌ای

1- Daisy channing

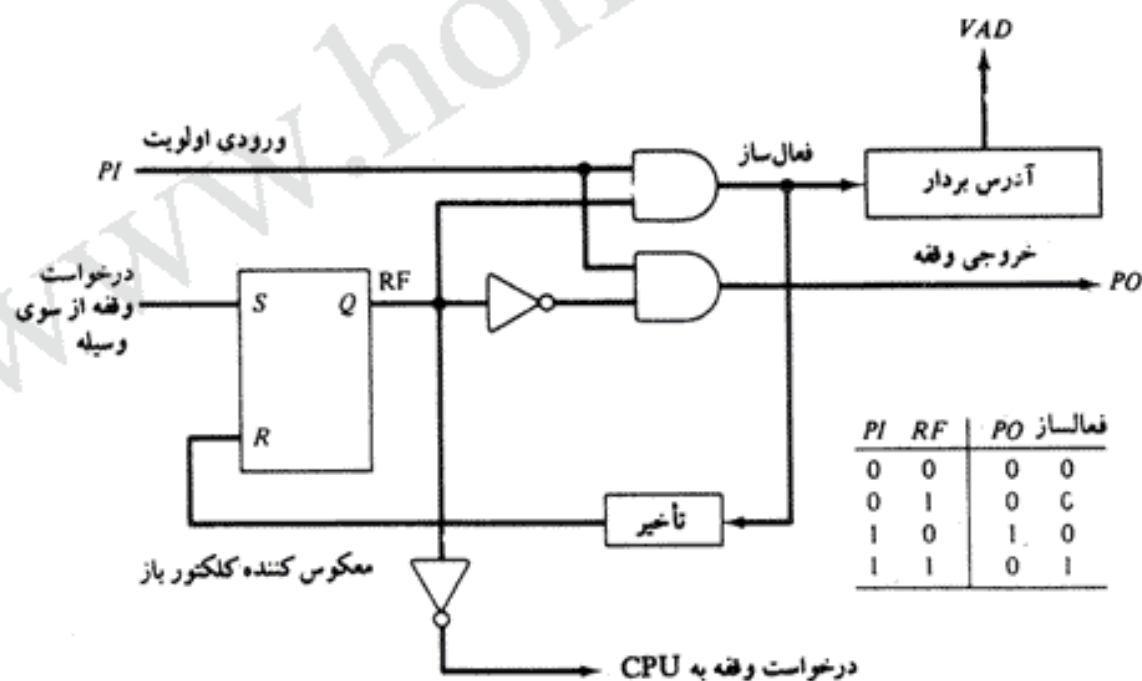
2- Wired logic

3- Acknowledge

سپس این وسیله آدرس بردار وقفه^۱ خود را (VAD) روی گذرگاه قرار می دهد تا CPU در سیکل وقفه از آن استفاده نماید.

وسیله ای که در ورودی PI دارای 0 باشد در خروجی PO هم 0 تولید می کند تا به وسیله ای که دارای اولویت وقفه پائین تر بعدی است اطلاع دهد که سیگنال تصدیق ممانعت شده است. وسیله ای که تقاضای وقفه نموده باشد و دارای 1 در ورودی PI باشد سیگنال تصدیق را با قرار دادن 0 در خروجی PO می بندد. اگر وسیله جانبی وقفه سرویس دهی شونده ای نداشته باشد، سیگنال تصدیق را با قرار دادن 1 در خروجی PO خود به وسیله بعدی می فرستد. بنابراین وسیله ای که دارای $PI=1$ و $PO=0$ باشد دارای بالاترین اولویت در تقاضای وقفه است، و این وسیله VAD خود را روی گذرگاه داده قرار می دهد. آرایش زنجیره ای بالاترین اولویت رابه وسیله ای که سیگنال تصدیق وقفه را از CPU دریافت کند تخصیص می دهد. هر چقدر وسیله از اولین مکان دورتر باشد دارای اولویت پائین تری است.

شکل ۱۱-۱۳ مدار منطقی درونی برای هر وسیله بهنگام اتصال به سیستم زنجیره ای را نشان می دهد. هنگامی که وسیله بخواهد CPU را متوقف کند فلیپ فلاپ RF خود را 1 می کند. خروجی فلیپ فلاپ RF از طریق یک معکوس کننده کلکتور باز^۲، یعنی مداری که منطق سیمی برای خط وقفه مشترک فراهم می سازد، می گذرد. اگر $PI=0$ باشد، هر دو خط PO و فعال ساز VAD، صرفنظر از مقدار RF، برابر صفرند. اگر $PI=1$ باشد و $RF=0$ باشد، سپس $RO=1$ و بردار آدرس غیرفعال



شکل ۱۱-۱۳ یک طبقه از آرایش اولویت دهی زنجیره ای

1- vector

2- Open - collector

می‌گردد. این وضعیت سیگنال تصدیق را از طریق PO به وسیله بعدی منتقل می‌سازد. وسیله جانبی هنگامی فعال است که $PI=1$ و $RF=1$ باشد. این وضعیت مقدار 0 را در PO قرار داده و بردار آدرس را برای گذرگاه داده فعال می‌سازد. فرض بر این است که هر وسیله دارای بردار آدرس متمایز خاص خود است. مدتی پس از دریافت بردار آدرس توسط CPU، و اطمینان از انجام آن، فلیپ فلاپ RF، 0 می‌شود.

وقفه اولویت دار موازی

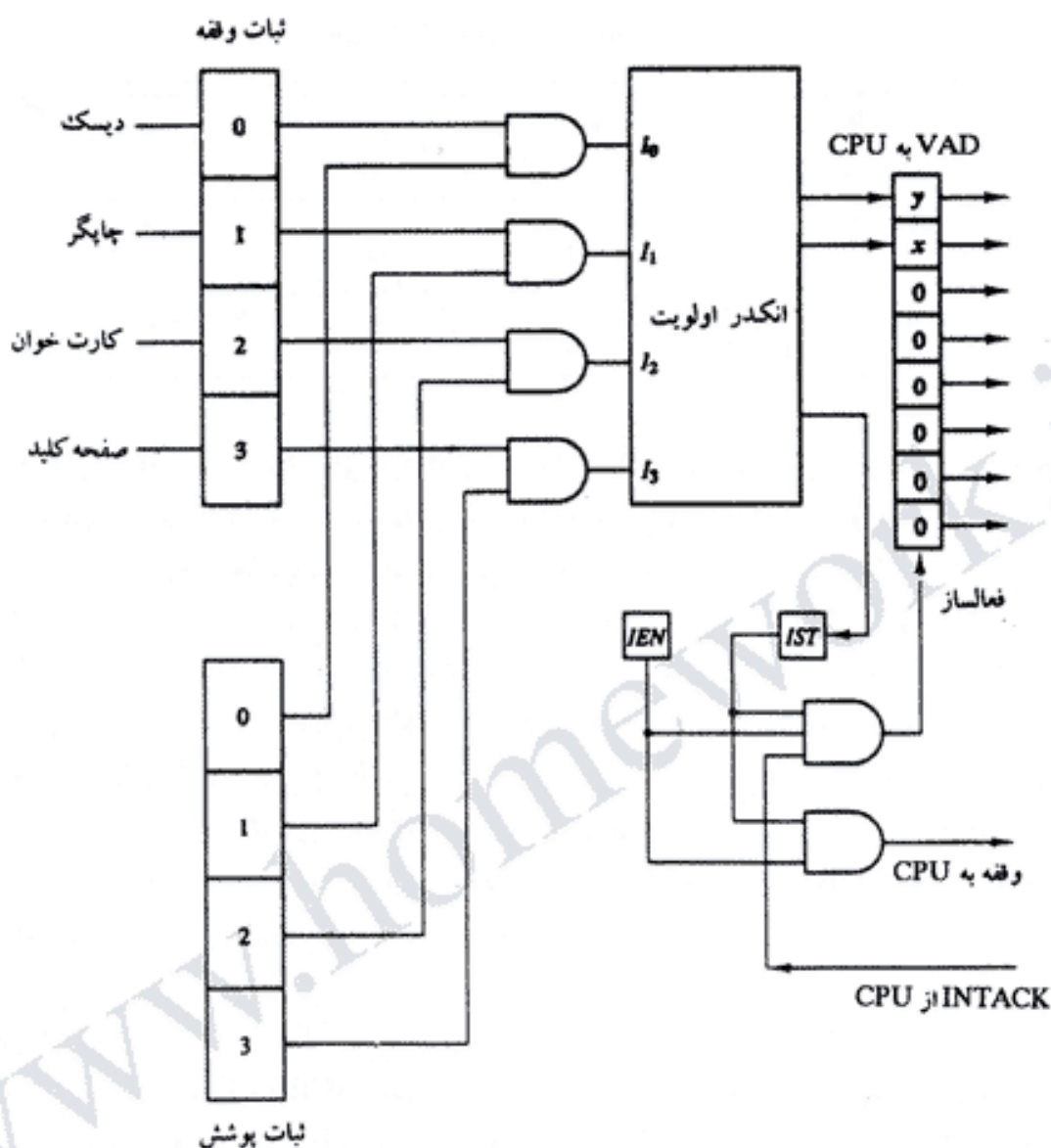
روش وقفه اولویت دار موازی از ثباتی استفاده می‌کند که بیت‌های آن بطور جداگانه توسط سیگنال وقفه هر وسیله 1 می‌شود. اولویت براساس محل بیت‌ها در ثبات تعیین می‌گردد. علاوه بر ثبات وقفه، مدار ممکن است دارای یک ثبات پوشش¹ باشد که هدف از آن کنترل وضعیت هر تقاضای وقفه است. ثبات پوشش (ماسک) را می‌توان برای غیرفعال شدن وقفه‌های پائین رتبه‌تر، در حین سرویس دهی به وقفه‌هایی با اولویت بالاتر، برنامه‌دهی کرد. این ثبات همچنین می‌تواند این امکان را فراهم سازد که در حین سرویس دهی به وسیله‌ای با اولویت پائینتر، وسیله‌ای با اولویت بالاتر بتواند به CPU وقفه دهد. مدار منطقی اولویت برای یک سیستم با چهار منبع وقفه در شکل ۱۴-۱۱ نشان داده شده است. این مدار شامل یک ثبات وقفه است که تک تک بیت‌های آن تحت شرایط خارجی 1 و توسط دستورات برنامه 0 می‌شود. به یک دیسک مغناطیسی، چون یک وسیله سریع است، بالاترین اولویت داده می‌شود. چاپگر اولویت بعدی را داراست و بدنبال آن دستگاه کاراکتر خوان و صفحه کلید می‌آیند. ثبات پوشش دارای تعداد بیت‌های مساوی با ثبات وقفه است. توسط دستورات برنامه، می‌توان هر بیتی را در ثبات پوشش 0 یا 1 کرد. هر بیت وقفه و بیت پوشش متناظر آن به یک گیت AND برای تولید چهار ورودی انکدر اولویت، اعمال می‌شوند. بدین ترتیب، وقفه تنها در صورتی قابل تشخیص است که بیت پوشش متناظر آن بوسیله برنامه 1 شده باشد. انکدر اولویت دو بیت آدرس بردار را ایجاد می‌کند که به CPU منتقل می‌گردد.

خروجی دیگر انکدر، فلیپ فلاپ وضعیت وقفه IST را، وقتی یک وقفه پوشش نیافته رخ دهد، 1 می‌کند. فلیپ فلاپ توانا ساز وقفه IEN می‌تواند بوسیله برنامه 1 یا 0 شود تا کنترل کلی را روی سیستم وقفه فراهم آورد. خروجی IST با AND شدن با IEN، سیگنال وقفه مشترکی را برای CPU فراهم می‌کند. سیگنال تصدیق وقفه INTACK از CPU، بافر گذرگاه در ثبات خروجی را فعال کرده و لذا یک آدرس بردار VAD روی گذرگاه داده قرار می‌گیرد. حال ما مدار انکدر اولویت را توضیح داده و سپس عکس‌العمل متقابل بین کنترل کننده وقفه و CPU را بحث خواهیم کرد.

انکدر اولویت

انکدر اولویت مداری است که عمل اولویت دهی را تحقق می‌بخشد. منطق انکدر اولویت طوری

1- Mask Register



شکل ۱۴-۱۱ ساختار افزار وقفه اولویت دار

است که اگر دو یا چند ورودی در یک لحظه وارد آن شوند، ورودی که بالاترین اولویت را دارا باشد تقدم خواهد داشت. جدول درستی انکدر اولویت چهار ورودی در جدول ۱۱-۲ ارائه شده است. x ها در جدول حالت بی اهمیت را تداعی می کنند. ورودی I_0 بالاترین اولویت را دارد، لذا صرف نظر از مقادیر دیگر ورودی ها، وقتی که این ورودی 1 باشد، خروجی تولید $xy = 00$ می نماید. I_1 اولویت بعدی را دارد. اگر $I_1 = 1$ باشد، مشروط بر اینکه $I_0 = 0$ باشد، صرف نظر از مقادیر دو ورودی دیگر که اولویت پائینتر را دارند خروجی 01 خواهد بود.

جدول ۲-۱۱ جدول ارزش کنگذار اولویت

ورودی‌ها				خروجیها			توابع بولی
I_0	I_1	I_2	I_3	x	y	IST	
1	x	x	x	0	0	1	
0	1	x	x	0	1	1	$x = I'_0 I'_1$
0	0	1	x	1	0	1	$y = I'_0 I'_1 + I'_0 I'_2$
0	0	0	1	1	1	1	$(IST) = I_0 + I_1 + I_2 + I_3$
0	0	0	0	x	x	0	

خروجی برای I_2 هنگامی تولید می‌شود که ورودی‌های با اولویت بالاتر 0 باشند و به همین منوال اولویت‌های پائین‌تر. فلیپ فلاپ وضعیت وقفه IST فقط وقتی که یک یا چند ورودی برابر 1 باشند، 1 می‌شود. اگر تمام ورودی‌ها 0 باشند، IST پاک می‌شود و سایر خروجی‌های آنکدر بکار نمی‌روند، لذا با x علامت زده شده‌اند. این بدان علت است که آدرس بردار وقتی که $IST = 0$ است به CPU منتقل نمی‌شود. توابع بول لیست شده در جدول منطق درونی آنکدر را مشخص می‌کند. معمولاً یک کامپیوتر بیش از چهار منبع وقفه دارد. مثلاً یک آنکدر با هشت ورودی، یک خروجی سه بیتی تولید می‌نماید. خروجی آنکدر اولویت، برای تشکیل بخشی از آدرس بردار هر منبع وقفه بکار می‌رود. به سایر بیت‌های بردار آدرس هر مقداری را می‌توان اختصاص داد. مثلاً آدرس بردار را می‌توان با الحاق شش صفر به خروجی‌های x و y آنکدر تشکیل داد. با این انتخاب، شماره‌های دودویی 0، 1، 2، 3 به بردارهای وقفه مربوط به چهار وسیله I/O اختصاص می‌یابد.

سیکل وقفه

فلیپ فلاپ تواناساز وقفه IEN که در شکل ۱۴-۱۱ نشان داده شده است می‌تواند بوسیله دستورالعمل‌های برنامه 1 یا 0 شود. وقتی که IEN پاک شود، تقاضای وقفه‌ای که از IST می‌رسد بوسیله CPU نادیده گرفته می‌شود. بیت IEN کنترل شده بوسیله برنامه به برنامه نویس اجازه می‌دهد تا در مورد استفاده یا عدم استفاده از امکانات وقفه تصمیم بگیرد. اگر دستورالعملی برای پاک کردن IEN در برنامه گنجانیده شود، بدان معنی است که برنامه نویس مایل نیست برنامه‌اش متوقف شود. دستورالعملی برای 1 کردن IEN نشانه این است که در حین اجرای برنامه جاری از قابلیت وقفه استفاده خواهد شد. اکثر کامپیوترها دارای سخت افزار برای پاک کردن IEN ، در مواقعی که وقفه بوسیله پردازنده تصدیق شوند، هستند. در پایان هر سیکل دستورالعمل، CPU بیت IEN و سیگنال وقفه را از IST و ارسی می‌کند. اگر هر یک از آنها 0 باشد، کنترل با دستور بعدی به کار ادامه می‌دهد. اگر هر دو بیت IEN و IST برابر 1 باشند، CPU وارد سیکل وقفه می‌شود. در حین سیکل وقفه، CPU رشته ریزاعمال زیر را انجام می‌دهد.

اشاره‌گر پشته را کاهش بده $SP \leftarrow SP-1$

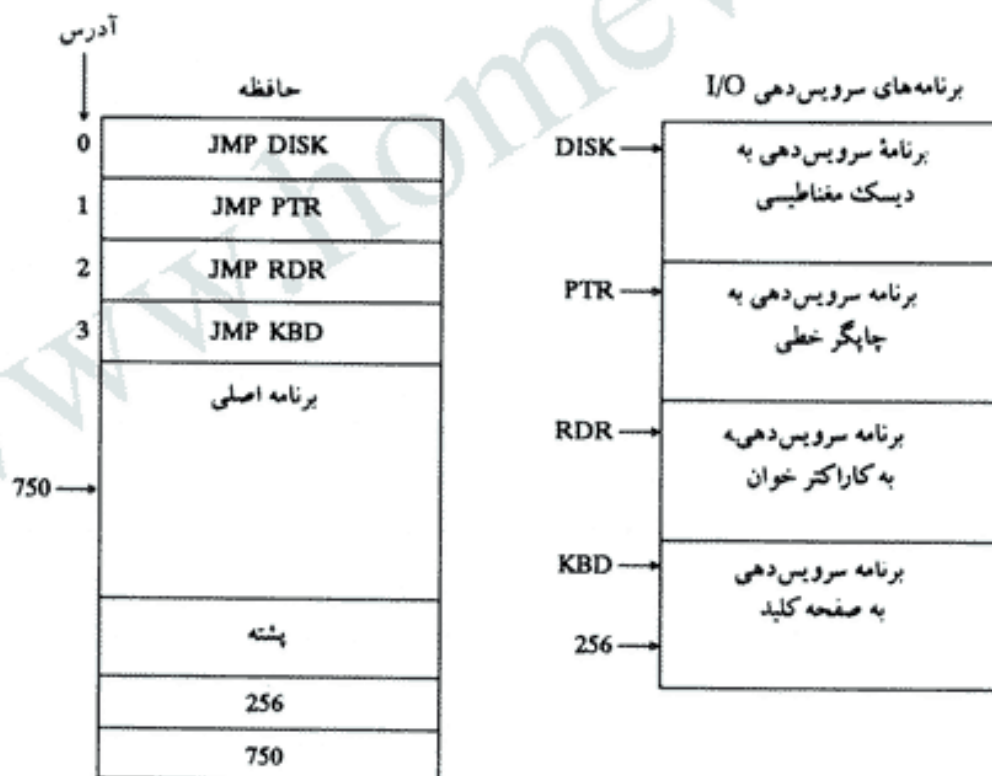
M [SP] ← PC	پوش کردن PC به داخل پشته
INTACK ← 1	تصدیق وقفه را فعال کن
PC ← VAD	بردار آدرس را به PC منتقل کن
IEN ← 0	وقفه بعدی را غیرفعال کن

برو برای برداشت دستورالعمل بعدی

CPU آدرس برگشت را از PC به پشته پوش می‌کند سپس با فعال کردن خط INTACK وقفه را فعال می‌کند. واحد اولویت وقفه با قرار دادن یک بردار وقفه منحصر بفرد در گذرگاه داده به وقفه پاسخ می‌دهد. CPU سپس با انتقال بردار آدرس به PC قبل از رفتن به فاز برداشت^۱ بعدی IEN را پاک می‌کند. دستورالعملی که در حین فاز برداشت بعدی از حافظه خوانده می‌شود در بردار آدرس قرار خواهد داشت.

روال های نرم افزاری

یک سیستم اولویت وقفه ترکیبی از تکنیک‌های سخت افزاری و نرم افزاری است. ماتاکنون جنبه‌های سخت افزاری اولویت وقفه را بحث کردیم. کامپیوتر باید برای سرویس دهی به تقاضاهای



شکل ۱۱-۱۵ برنامه‌های نخیره شده در حافظه برای سرویس دهی به وقفه‌ها

1- Fetch

وقفه و کنترل ثبات های سخت افزار وقفه روال های نرم افزاری نیز داشته باشد. شکل ۱۵-۱۱ برنامه هایی را که باید برای اداره سیستم وقفه در حافظه قرار داشته باشد نشان می دهد. هر وسیله برنامه سرویس دهی خاص خود را دارد و از طریق یک دستورالعمل پرش (JMP)، که در آدرس بردار قرار دارد قابل دسترسی است. نام سمبلیک هر روال نشان دهنده آدرس شروع برنامه سرویس است. پشته نشان داده شده برای ذخیره کردن آدرس بازگشت پس از هر وقفه است.

برای روشن کردن مطلب با یک مثال فرض کنید که صفحه کلید بیت وقفه را مادامی که CPU در حال اجرای دستورالعملی در مکان 749 برنامه اصلی است، 1 کند. در پایان سیکل دستورالعمل، کامپیوتر به یک سیکل وقفه می رود. ابتدا آدرس بازگشت 750 را در پشته ذخیره کرده و سپس آدرس بردار 00000011 را از گذرگاه پذیرفته و آنرا به PC انتقال می دهد. آنگاه دستورالعمل واقع در مکان 3 اجرا شده، و در نتیجه کنترل به روال KBD منتقل می شود. حال فرض کنید که دیسک بیت وقفه اش را، هنگامی که CPU در حال اجرای دستورالعمل واقع در مکان 255 در برنامه KBD است، 1 کند. آدرس 256 بداخل پشته پوش داده شده و کنترل به برنامه سرویس DISK منتقل می شود. آخرین دستور در هر برنامه، دستورالعمل بازگشت از وقفه است. وقتی که برنامه سرویس دیسک کامل شود، دستور بازگشت پشته را پاپ کرده و عدد 256 را در PC می گذارد. این عمل کنترل را به روال KBD برای ادامه روال سرویس صفحه کلید منتقل می سازد. در پایان برنامه KBD، آخرین دستورالعمل پشته را پاپ کرده و کنترل برنامه را به برنامه اصلی برمی گرداند. بنابراین، وسیله ای که دارای اولویت بالاتر باشد می تواند وسیله ای که اولویت پائین تر را دارد وقفه دهد. فرض بر این است که زمان صرف شده در سرویس به وسیله ای که وقفه اولویت بالا دارد نسبت به وسیله ای که اولویت پائین را داراست کوتاه تر باشد و بنابراین مشکل از دست دادن اطلاعات رخ نمی دهد.

عملیات آغازین و پایانی

هر روال سرویس وقفه باید مجموعه ای از عملیات آغازین و پایانی را برای کنترل ثبات ها سیستم وقفه سخت افزاری داشته باشد. بخاطر بیاورید که فعال ساز وقفه IEN در پایان سیکل وقفه پاک می شود. این فلیپ فلاپ مجدداً باید برای فعال کردن تقاضاهای وقفه با اولویت بالاتر، 1 شود، ولی نه قبل از غیرفعال شدن وقفه های با اولویت پائین تر. رشته اعمال آغازین هر روال سرویس وقفه باید دارای دستورالعمل هایی برای کنترل سخت افزار وقفه به شکل زیر باشد.

۱- صفر کردن بیت های سطح پائین تر ثبات پوشش

۲- پاک کردن بیت وضعیت وقفه IST

۳- ذخیره کردن محتوای ثبات های پردازنده

۴- 1 کردن بیت فعال ساز وقفه IEN

۵- اجرای روال سرویس

بیت‌های سطح پائین‌تر ثبات پوشش (از جمله بیت منبعی که وقفه داده است) پاک شده تا مانع فعال شدن وقفه تحت این شرایط گردد. اگرچه منابع وقفه‌ای که اولویت پائینتر دارند به بیت‌های بالا رتبه ثبات پوشش تخصیص یافته‌اند، می‌توان اولویت را در صورت تمایل عوض کرد زیرا برنامه نویسی می‌تواند هر آرایشی را برای ثبات پوشش اختیار کند. بیت وضعیت وقفه باید پاک شود تا بتواند هنگام رخداد وقفه با اولویت بالاتر، برابر با 1 شود. محتوای ثبات پردازنده ذخیره می‌شوند زیرا ممکن است بوسیله برنامه‌ای که به آن وقفه داده شده است پس از بازگشت کنترل به آن مورد نیاز واقع شود. سپس فعال ساز وقفه IEN برابر با 1 شده و اجازه می‌دهد سایر وقفه‌ها (با اولویت بالاتر) رخ دهند و کامپیوتر تقاضاهای وقفه را سرویس می‌دهد.

در پایان هر روال سرویس دهی باید دستورالعمل‌هایی برای کنترل سخت افزار وقفه بشکل زیر موجود باشد.

- ۱- پاک کردن بیت فعال ساز وقفه IEN
- ۲- بازگرداندن محتوای ثبات‌های پردازنده
- ۳- پاک کردن بیت متعلق به منبعی که سرویس داده شده، در ثبات وقفه
- ۴- 1 کردن بیت‌های با اولویت پائینتر در ثبات پوشش
- ۵- بازگرداندن آدرس بازگشت در PC، و 1 کردن IEN

بیتی در ثبات وقفه که به منبع وقفه متعلق است باید 0 شود تا این منبع بتواند مجدداً وقفه را تقاضا کند. بیت‌هایی دارای اولویت پائین‌تر در ثبات پوشش (از جمله بیت مربوط به منبعی که به آن سرویس داده شد) 1 می‌شوند که بتوانند وقفه بدهند. بازگشت به برنامه متوقف شده با بازگرداندن آدرس برگشت به PC صورت می‌گیرد. توجه کنید که سخت افزار باید طوری طراحی شود که هیچ وقفه‌ای ضمن اجرای مراحل ۲ تا ۵ رخ ندهد؛ در غیراینصورت آدرس بازگشت ممکن است مفقود شده و اطلاعات در ثبات‌های پوشش و پردازنده ممکن است از دست برود. با این دلیل IEN ابتدا پاک شده و پس از انتقال آدرس بازگشت در ثبات PC دوباره 1 می‌شود.

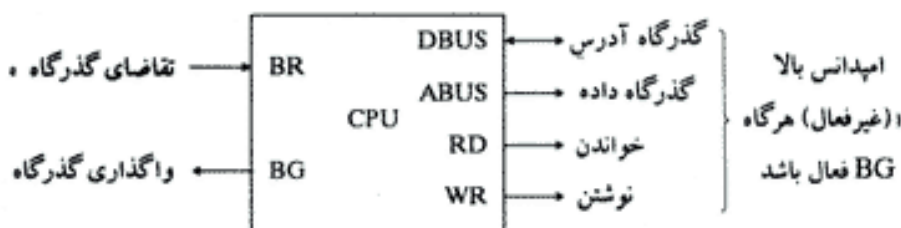
عملیات آغازین و پایانی لیست شده فوق عملیات سریار^۱ نامیده می‌شوند. این عملیات بخشی از برنامه اصلی سرویس نیستند ولی برای پردازش وقفه ضرورت دارند. همه عملیات سریار را می‌توان با نرم افزار پیاده‌سازی کرد. این عمل با گنجانیدن دستورالعمل‌های مناسب در ابتدا و انتهای هر زیرروال انجام می‌شود. برخی از اعمال سریار بوسیله سخت افزار بطور اتوماتیک انجام می‌گردد. محتوای ثبات‌های پردازنده می‌تواند قبل از انشعاب به روال سرویس بداخل پشته پوش داده شود. سایر عملیات آغازین و پایانی را نیز می‌توان به عهده سخت افزار گذاشت. به این ترتیب می‌توان زمان بین دریافت یک وقفه و اجرای دستورالعمل‌های لازم برای سرویس دادن به منبع وقفه را کاهش داد.

1- Overhead or Housekeeping Chores

۶-۱۱ دستیابی مستقیم به حافظه (DMA)^۱

انتقال داده بین یک وسیله ذخیره سازی سریع مانند دیسک مغناطیسی و حافظه اغلب بوسیله سرعت CPU محدود می شود. حذف CPU از مسیر و ایجاد امکان کنترل مستقیم گذرگاه حافظه توسط وسیله جانبی سرعت انتقال را بهبود می بخشد. این تکنیک انتقال دستیابی مستقیم به حافظه (DMA) نام دارد. در حین انتقال بروش DMA، CPU بیکار است و کنترلی بر گذرگاه های حافظه ندارد. یک کنترل کننده DMA به منظور اداره مستقیم امر انتقال و بین وسیله I/O و حافظه، کنترل گذرگاه ها را بدست می گیرد. CPU را می توان بطرق مختلف در حالت بیکاری قرارداد. یک روش معمول که بطور گسترده ای در ریزپردازنده ها از آن استفاده می شود غیرفعال کردن گذرگاه ها بوسیله سیگنال های کنترل خاصی است. در شکل ۱۶-۱۱ دو سیگنال کنترل در CPU که امکان انتقال DMA را فراهم می کنند نشان داده شده است. ورودی تقاضای گذرگاه^۲ (BR) بوسیله کنترل کننده DMA برای تقاضای وقفه از CPU بکار می رود تا کنترل گذرگاه ها را رها نماید. وقتی این ورودی فعال است، CPU اجرای دستورالعمل جاری را پایان داده و گذرگاه آدرس، گذرگاه داده، و خطوط خواندن و نوشتن را در حالت امپدانس بالا قرار می دهد. حالت امپدانس بالا مانند مدار باز عمل می کند، یعنی خروج قطع شده و معنی منطقی نخواهد داشت (۴-۳ را ببینید). CPU خروجی واگذاری گذرگاه^۳ (BG) را فعال می کند تا به DMA اطلاع دهد که گذرگاه ها در حالت امپدانس - بالا هستند. اکنون DMA ای که تقاضای گذرگاه را داده است می تواند کنترل گذرگاه ها را برای هدایت انتقال های حافظه بدون دخالت پردازنده به دست بگیرد. وقتی که DMA انتقال را به انجام رساند، خط تقاضای گذرگاه را غیرفعال می کند. CPU خط واگذاری گذرگاه را غیرفعال می کند، کنترل گذرگاه را بعهده می گیرد، و آنرا به عملکرد عادی خود باز می گرداند.

وقتی که DMA کنترل سیستم گذرگاه را بعهده می گیرد، مستقیماً با حافظه ارتباط برقرار می کند. انتقال می تواند به چندین طریق انجام شود. در انتقال یکباره^۴ DMA، یک رشته بلاک متشکل از یک تعداد کلمات حافظه در حالی که DMA بر گذرگاه حافظه حاکم است، بطور پیوسته بیکباره مستقل می گردد. این شیوه انتقال برای وسایل سریع مانند دیسک های مغناطیسی ضروری است. در این سیستم ها



شکل ۱۶-۱۱ سیگنال های گذرگاه CPU برای انتقال DMA

1- Direct Memory Access

2- Bus Request

3- Bus Grant

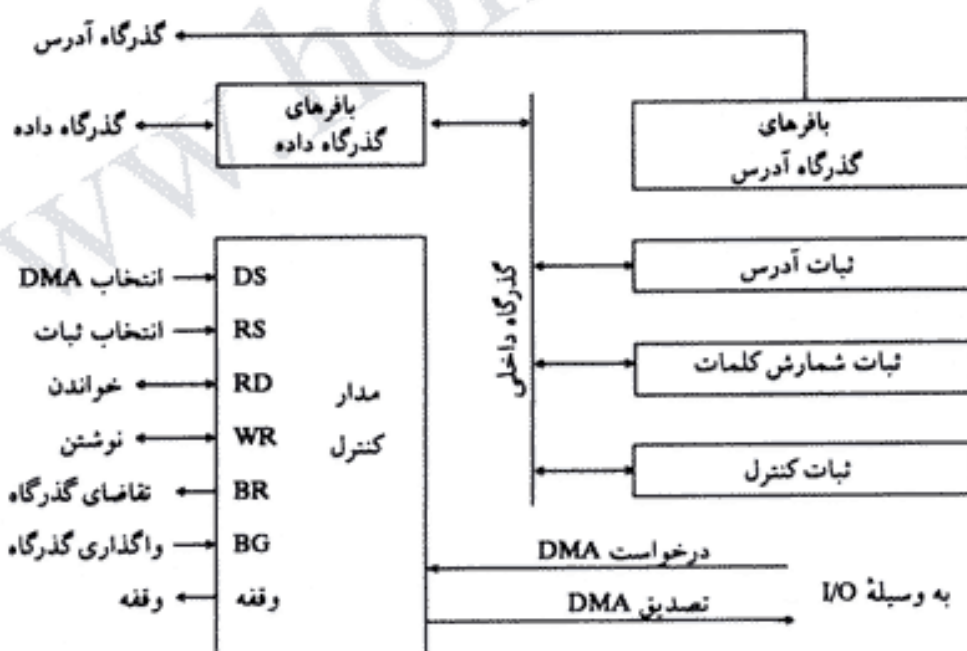
4- Burst Transfer

نمی توان تا انتقال کامل بلاک داده ها، انتقال را متوقف یا کند کرد. تکنیک دیگری که سرقت سیکل^۱ نامیده می شود به کنترل کننده DMA اجازه می دهد که هر بار یک کلمه داده را انتقال دهد و پس از آن کنترل گذرگاه ها را به CPU بازگرداند. CPU صرفاً عملکرد خود را به اندازه یک سیکل حافظه به تأخیر می اندازد تا عمل انتقال حافظه I/O بطور مستقیم صورت گرفته و یک سیکل حافظه را بریابد.

کنترل کننده DMA

کنترل کننده DMA برای ارتباط با CPU و وسیله جانبی به مدارات معمول واسطه نیاز دارد. به علاوه یک ثبات آدرس و یک ثبات شمارش کلمه^۲ و یک مجموعه خطوط آدرس هم مورد نیاز است. ثبات آدرس و خطوط آدرس برای ارتباط مستقیم با حافظه بکار می روند. ثبات شمارش کلمه تعداد کلماتی را که باید منتقل شود مشخص می کند. انتقال داده ها می تواند مستقیماً تحت کنترل DMA بین وسیله جانبی و حافظه صورت گیرد.

شکل ۱۷-۱۱ بلاک دیاگرام یک نمونه کنترل کننده DMA را نشان می دهد. کنترل کننده با CPU از طریق گذرگاه داده و خطوط کنترلی ارتباط برقرار می کند. ثبات های DMA بوسیله CPU از مسیر گذرگاه آدرس و با فعال کردن ورودی های DS (انتخاب DMA)^۳ و RS (انتخاب ثبات)^۴ انتخاب می شوند. ورودی های RD (خواندن) و WR (نوشتن) دوجهته هستند. وقتی که BG (واگذاری گذرگاه) صفر باشد،



شکل ۱۷-۱۱ بلاک دیاگرام کنترل کننده DMA

1- Cycle stealing

2- Word Count Register

3- DMA Select

4- Register select

CPU می تواند از طریق گذرگاه داده برای خواندن یا نوشتن در ثبات های DMA، با مشخص کردن آدرس و فعال کردن کنترل های RD و WR، مستقیماً با آنها ارتباط برقرار کند. DMA با وسایل خارجی از طریق خطوط تقاضا و تصدیق و استفاده از رویه از قبل تعیین شده دست دهی ارتباط برقرار می کند.

کنترل کننده DMA سه ثبات دارد: ثبات آدرس، ثبات شمارش کلمه و ثبات کنترل. ثبات آدرس حاوی یک آدرس برای مشخص کردن مکان مورد نظر در حافظه است. بیت های آدرس از طریق بافرها به گذرگاه آدرس وارد می شوند. ثبات آدرس پس از انتقال هر کلمه به حافظه یک واحد اضافه می شود. ثبات شمارش کلمه تعداد کلماتی که باید منتقل شوند را نگه می دارد. این ثبات پس از هر بار انتقال یک واحد کسر می شود. ثبات شمارش کلمه تعداد کلمات انتقالی را نگه می دارد. ثبات پس از هر انتقال یک واحد کم شده و در آغاز هم برای صفر بودن و ارسی می شود. ثبات کنترل شیوه انتقال را معین می کند. تمام ثبات ها در DMA از دید CPU مانند ثبات های I/O هستند. بنابراین CPU از ثبات DMA تحت کنترل برنامه و از طریق گذرگاه، داده را خوانده و یا در آن می نویسد.

ابتدا DMA بوسیله CPU مقداردهی اولیه می شود. پس از آن، DMA شروع به کار کرده و انتقال داده بین حافظه و واحد جانبی را تا انتقال کامل بلاک انجام می دهد. روند مقداردهی اولیه اساساً برنامه ای است متشکل از دستورالعمل های I/O که شامل آدرس انتخاب ثبات های خاص DMA است. CPU با ارسال اطلاعات زیر از طریق گذرگاه داده DMA را مقداردهی اولیه می کند.

۱- آدرس شروع بلاکی از حافظه که داده ها در آن قرار دارد (برای خواندن) و یا داده ها باید در آن ذخیره شود (برای نوشتن)

۲- شمارش کلمات، یعنی تعداد کلمات بلاک حافظه

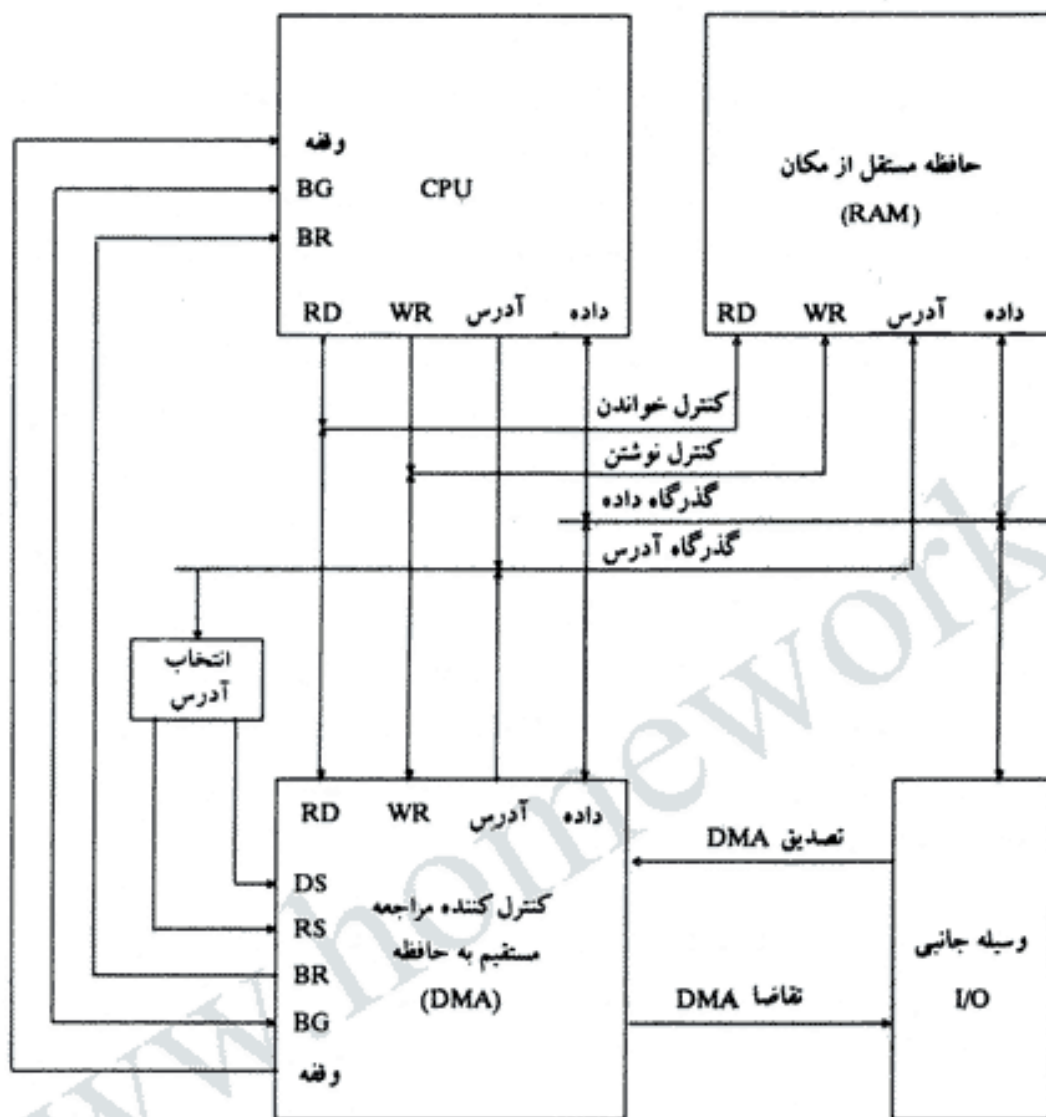
۳- کلمه کنترل برای مشخص کردن اطلاعات مربوط به شیوه انتقال مانند خواندن یا نوشتن

۴- کلمه کنترلی برای شروع انتقال DMA

آدرس شروع در ثبات آدرس ذخیره می شود. تعداد کلمه در ثبات شمارش و کلمه اطلاعات کنترلی در ثبات کنترل ذخیره می گردد. وقتی که DMA مقداردهی اولیه شد، CPU از ارتباط با DMA دست می کشد مگر اینکه سیگنال وقفه ای دریافت کند و یا بخواهد تعداد کلمات انتقال یافته را ورسی نماید.

انتقال DMA

موقعیت کنترل کننده DMA در میان سایر قسمت های کامپیوتر در شکل ۱۸-۱۱ نشان داده شده است. CPU با DMA از طریق گذرگاه های آدرس و داده، مانند سایر واسطه ها، ارتباط برقرار می کند. DMA آدرس خاص خود را دارد، که خطوط DS و RS را فعال می کند. CPU مقداردهی اولیه به DMA را از طریق گذرگاه داده انجام می دهد. وقتی که DMA فرمان شروع کنترل را دریافت می کند، انتقال وسیله جانبی و حافظه آغاز می شود.



شکل ۱۱-۱۸ انتقال DMA در یک سیستم کامپیوتری

وقتی که وسیله جانبی درخواست DMA را ارسال می‌کند، کنترل کننده DMA خط BR را فعال می‌نماید، و به این طریق به CPU اطلاع می‌دهد که گذرگاه‌ها را رها کند. CPU با فعال کردن خط BG به DMA اطلاع می‌دهد که گذرگاه‌های آن غیر فعالند. سپس DMA مقدار فعلی ثبات آدرس خود را روی گذرگاه آدرس قرار داده، سیگنال RD یا WR را فعال می‌کند، و سیگنال تصدیق DMA را به وسیله جانبی می‌فرستد. توجه کنید که خطوط RD و WR در کنترل کننده DMA دو جهت هستند. جهت انتقال بستگی به وضعیت خط BG دارد. وقتی که $BG=0$ باشد، خطوط RD و WR برای DMA ورودی بوده و این امکان را برای CPU فراهم می‌کنند که با ثبات‌های DMA ارتباط برقرار کند. وقتی که $BG=1$ است، خطوط خروجی از کنترل کننده DMA به RAM بوده و عمل خواندن یا نوشتن را مشخص می‌کنند.

وقتی که وسیله جانبی یک سیگنال تصدیق را از DMA دریافت کند، کلمه‌ای را در گذرگاه داده قرار می‌دهد (برای نوشتن) یا کلمه را از گذرگاه داده دریافت می‌کند (برای خواندن). بنابراین DMA عمل خواندن یا نوشتن را کنترل می‌کند و آدرس حافظه را فراهم می‌کند. واحد جانبی سپس با حافظه از طریق گذرگاه داده برای انتقال مستقیم بین دو واحد ارتباط برقرار می‌نماید و این مدت CPU موقتاً غیرفعال است. برای هر کلمه‌ای که انتقال یابد، DMA ثبات آدرس را افزایش داده و ثبات شمارش کلمه را کاهش می‌دهد. اگر شمارش کلمه به صفر نرسد، DMA خط تقاضای^۱ واصله از وسیله جانبی را چک می‌کند. برای یک وسیله سریع، خط به محض خاتمه انتقال قبلی فعال خواهد شد. سپس دومین انتقال آغاز می‌شود، و روند تا انتقال کامل بلاک ادامه می‌یابد. اگر سرعت وسیله جانبی کندتر باشد، خط تقاضای DMA ممکن است قدری دیرتر برسد. در این صورت DMA خط تقاضای گذرگاه را غیرفعال می‌کند تا CPU بتواند به اجرای برنامه خود ادامه دهد. وقتی که دستگاه جانبی تقاضای انتقال کرد، DMA مجدداً گذرگاه‌ها را تقاضا می‌نماید.

اگر ثبات شمارش به صفر برسد، DMA از ارسال بیشتر انتقال خودداری می‌نماید و تقاضای گذرگاه خود را حذف می‌کند. همچنین CPU را از اتمام کار بوسیله وقفه مطلع می‌سازد. وقتی که CPU به وقفه پاسخ دهد، محتوای ثبات شمارش کلمه را می‌خواند. مقدار صفر در این ثبات بمعنی انتقال موفقیت آمیز کلیه کلمات است. CPU در هر لحظه قادر به خواندن این ثبات می‌باشد و بدینوسیله کلماتی را که قبلاً منتقل شده‌اند و ارسی می‌کند.

یک کنترل کننده DMA ممکن است بیش از یک کانال داشته باشد. در این حال، هر کانال دارای جفت سیگنال‌های کنترل تقاضا و تصدیق می‌باشد که بطور جداگانه به وسایل جانبی متصل است. هر کانال، همچنین دارای ثبات آدرس و ثبات شمارش کلمه خود در کنترل کننده DMA است. میان کانال‌ها می‌توان اولویت هم برپا کرد بطوری که کانال با اولویت بالاتر قبل از کانال‌های با اولویت پایین‌تر سرویس داده شود.

انتقال DMA در بسیاری از کاربردها بسیار مفید است. این روش در انتقال سریع اطلاعات بین دیسک‌های مغناطیسی و حافظه بکار می‌رود. همچنین برای بهنگام کردن صفحه نمایش در پایانه محاوره‌ای نیز مفید است. معمولاً، تصویری از نمایش صفحه پایانه^۲ در حافظه نگهداری می‌شود که قابل بهنگام شدن تحت کنترل برنامه است. محتوای حافظه می‌تواند بطور دوره‌ای با استفاده از انتقال DMA به صفحه ارسال شود.

۷-۱۱ پردازنده ورودی و خروجی IOP^۳

بعوض ارتباط هر مدار واسطه با CPU، یک کامپیوتر می‌تواند یک یا چند پردازنده خارجی داشته باشد که به هر یک از آنها کاری که مستقیماً ارتباط با وسایل I/O را فراهم می‌سازد تخصیص داد. یک

1- Request Line

2- Terminal

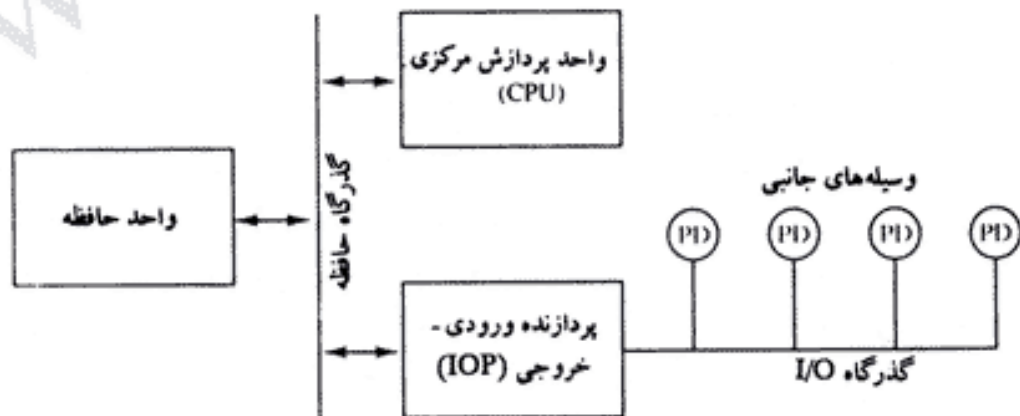
3- Input - Output Processor

پردازنده ورودی - خروجی (IOP) ممکن است بعنوان پردازنده ای با مراجعه مستقیم به حافظه طبقه بندی شود که با وسیله های I/O تبادل اطلاعات می کند. در این پیکربندی، سیستم کامپیوتر می تواند به یک واحد حافظه و تعدادی پردازنده از CPU و یک یا چند IOP تقسیم شود. هر IOP از یک کار ورودی یا خروجی مراقبت می کند، و CPU را از علمیات سریار موجود در انتقال آزاد می سازد. یک پردازنده که با پایانه های دوردست از طریق تلفن و سایر وسایل مخابراتی بصورت سری تبادل اطلاعات می کند، پردازنده تبادل داده¹ (DCP) نامیده می شود.

IOP مشابه CPU است جز اینکه برای انجام جزئیات پردازش I/O طراحی شده است. برخلاف کنترل کننده DMA که باید کلاً بوسیله CPU تنظیم شود، IOP می تواند دستورالعمل های خود را برداشت و اجرا کند. دستورالعمل های IOP بخصوص برای انتقال IOP طراحی شده اند. بعلاوه IOP می تواند پردازش دیگری مانند محاسبات، منطق، انشعاب و ترجمه کد را نیز انجام دهد.

بلاک دیاگرام یک کامپیوتر با دو پردازنده در شکل ۱۹-۱۱ دیده می شود. واحد حافظه یک موقعیت مرکزی را اشغال کرده است و بصورت دستیابی مستقیم به حافظه می تواند با هر پردازنده ارتباط برقرار کند. CPU مسئول پردازش داده لازم در حل کارهای محاسباتی است. پردازنده IOP مسیری برای انتقال داده بین وسایل مختلف جانبی و واحد حافظه است. CPU معمولاً کار مقداردهی اولیه به برنامه I/O را برعهده دارد. از این پس IOP بطور مستقل از CPU کار کرده و انتقال داده را از وسیله خارجی به حافظه ادامه می دهد.

قالب های داده و وسایل جانبی با قالب های داده حافظه و قالب داده CPU تفاوت دارد. IOP باید به کلمات داده دریافتی از منابع متعدد و مختلف ساختار بدهد. مثلاً، ممکن است لازم باشد تا چهار بایت از یک وسیله ورودی دریافت شده و قبل از انتقال به حافظه بصورت کلمه 32 بیتی درآید. در حالی که



شکل ۱۹-۱۱ بلاک دیاگرام کامپیوتر با پردازنده I/O

1- Data Communication Processor

CPU برنامه خود را اجرا می کند داده ها در IOP با سرعت و ظرفیت بیت وسیله ورودی جمع می شوند. پس از اینکه داده ها در کلمه حافظه کنار هم قرار گرفتند^۱، مستقیماً از IOP با سرعت یک سیکل حافظه از CPU به حافظه منتقل می شوند. بطور مشابه یک کلمه حافظه انتقال یافته از حافظه به IOP از IOP به وسیله خروجی با سرعت و ظرفیت وسیله منتقل می گردد.

ارتباط بین IOP و وسایل متصله به آن مشابه روش کنترل برنامه در انتقال است. ارتباط با حافظه مشابه دستیابی مستقیم به حافظه است. روشی که بوسیله آن CPU و IOP با هم تبادل داده می کند بستگی به سطح پیچیدگی در سیستم دارد. در کامپیوترهای با مقیاس بزرگ (VLS)^۲، هر پردازنده از دیگری مستقل است و هر پردازشگر می تواند عملی را آغاز کند. در اکثر سیستم های کامپیوتری، CPU حاکم و IOP یک پردازنده تابع است. آغاز کلیه عملیات به CPU واگذار شده است، ولی دستورالعمل های I/O در IOP اجرا می شود. دستورالعمل های CPU امکان شروع انتقال I/O و نیز امکان تست وضعیت لازم برای تصمیم گیری بر انواع فعالیت های I/O را فراهم می کنند. IOP به نوبه خود با استفاده از وقفه توجه CPU را جلب می کند. همچنین به تقاضای CPU با قراردادن کلمه وضعیت در مکان از پیش تعیین شده که بعداً توسط برنامه CPU واریسی می شود پاسخ می دهد. وقتی که یک عمل I/O لازم شود، CPU به پردازنده IOP اطلاع می دهد که درجه مکانی برنامه I/O را بیابد و سپس جزئیات انتقال را به IOP واگذار می کند.

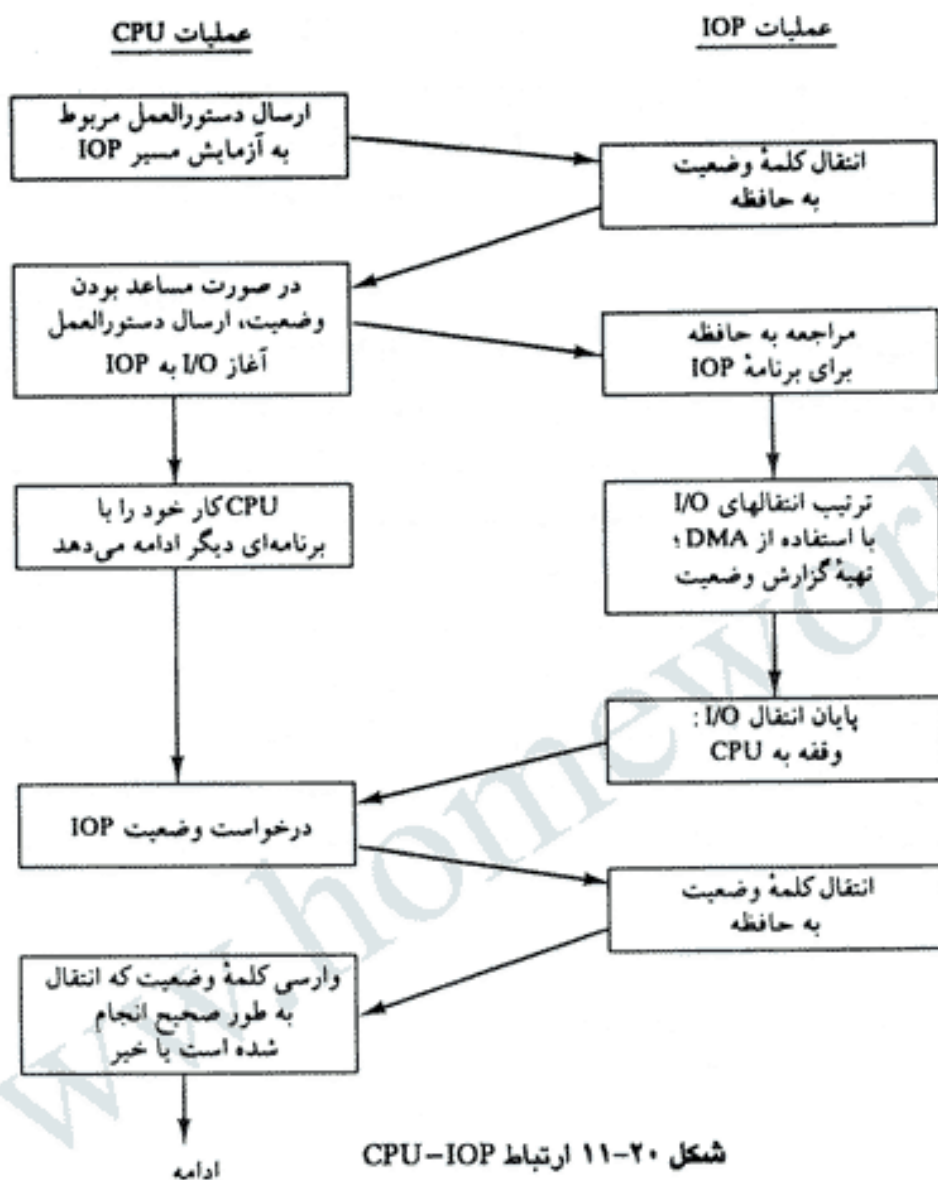
ارتباط CPU - IOP

تبادل اطلاعات بین CPU و IOP ممکن است بسته به کامپیوتر خاص مورد استفاده، به شکل های مختلفی صورت گیرد. در بسیاری از حالات حافظه نقش مرکز پیامی را بازی می کند که هر پردازنده اطلاعات را برای دیگری در آن قرار می دهد. برای درک بهتر کار یک IOP، ما روشی را که بوسیله آن CPU و IOP تبادل اطلاعات می کنند با مثالی تشریح خواهیم کرد. این مثال که بسیار ساده است از جزئیات بسیاری از اعمال صرفنظر می کند تا ایده ای کلی از مفاهیم اساسی را فراهم نماید.

رشته اعمال می تواند توسط فلوجارت شکل ۲۰-۱۱ نشان داده شود. CPU دستورالعملی را برای تست مسیر IOP ارسال می نماید. IOP با درج یک کلمه وضعیت در حافظه که CPU آن را واریسی می نماید، به آن پاسخ می دهد. بیت های کلمه وضعیت شرایط IOP و وسیله I/O را مانند، بار پیش از حد، وسیله با انتقالی دیگر مشغول است، یا وسیله برای انتقال I/O آماده است، مشخص می کند. CPU به یک کلمه وضعیت واقع در حافظه مراجعه می کند تا ببیند در مرحله بعدی باید چه کاری را انجام دهد. اگر همه چیز مرتب باشد، CPU دستورالعمل را برای شروع انتقال I/O ارسال می کند. آدرس حافظه ای که بوسیله این دستور دریافت می شود به IOP اطلاع می دهد که کجا برنامه خود را بیابد. اکنون CPU می تواند با برنامه دیگری کار خود را دنبال کند در حالیکه IOP با برنامه I/O مشغول است.

1- Assemble

2- Very - Large - Scale Computers



هر دو برنامه توسط انتقال DMA به حافظه مراجعه می‌کنند. وقتی که I/O اجرای برنامه خود را پایان می‌دهد، یک تقاضای وقفه به CPU ارسال می‌کند. CPU با صدور یک دستورالعمل جهت خواندن گزارش وضعیت واصله از I/O به وقفه پاسخ می‌دهد. کلمه وضعیت مشخص می‌کند که انتقال تکمیل شده است و یا اینکه خطایی در حین انتقال اتفاق افتاده است. CPU با بررسی بیت‌های کلمه وضعیت تشخیص می‌دهد که عملیات I/O بطور موفقیت آمیز و بدون خطا پایان یافته است.

I/O از کلیه انتقال داده‌ها بین چند واحد I/O و حافظه مراقبت می‌کند در حالیکه CPU برنامه دیگری را پردازش می‌نماید. CPU و I/O، هر دو برای استفاده از حافظه به آن مراجعه می‌کنند، بنابراین تعداد وسایلی که می‌توانند همزمان کار کنند بوسیله زمان دستیابی به حافظه محدود می‌شود. البته

امکان اشباع حافظه توسط وسایل جانبی I/O ضعیف است، زیرا سرعت وسایل خیلی کمتر از CPU است. با این وجود، برخی واحدهای بسیار سریع، مانند دیسک‌های مغناطیسی، می‌توانند تعداد قابل ملاحظه‌ای از سیکل موجود حافظه استفاده کنند. در این حالت سرعت CPU ممکن است افت کند زیرا در اغلب اوقات مجبور خواهد بود منتظر انجام انتقال‌های حافظه توسط IOP بماند.

کانال I/O در IBM 370

پردازنده I/O در کامپیوتر IBM370، کانال^۱ خوانده می‌شود. پیکربندی این کامپیوتر شامل تعدادی کانال است و هر کانال به یک یا چند وسیله I/O متصل است. سه نوع کانال وجود دارد: مولتی پلکسری، سلکتوری و بلاکی - مولتی پلکسری. کانال مولتی پلکسری را می‌توان به تعدادی وسایل با سرعت پائین یا متوسط وصل کرد و قادر است بطور همزمان با چند وسیله I/O کار کند. کانال سلکتور بصورتی طراحی شده که در هر زمان یک عمل I/O انجام دهد و معمولاً برای کنترل یک وسیله سریع بکار می‌رود. کانال بلاکی - مولتی پلکسری ترکیبی از ویژگی‌های هر دو کانال مولتی پلکسری و سلکتوری را داراست. این کانال با چندین وسیله سریع ارتباط برقرار می‌کند، ولی همه انتقال‌های I/O با همه بلاک داده صورت می‌گیرد، در صورتی که کانال مولتی پلکسری فقط هر لحظه یک بایت را منتقل می‌نماید. CPU مستقیماً از طریق خطوط کنترل اختصاصی با کانال‌ها و از طریق نواحی رزرو شده ذخیره‌سازی در حافظه بطور غیرمستقیم تبادل اطلاعات می‌کند. شکل ۲۱-۱۱ قالب کلمات مربوط به عملکرد کانال را نشان می‌دهد.

آدرس وسیله	آدرس کانال	کد عمل
------------	------------	--------

(الف) قالب دستورالعمل‌های I/O

شمارش	وضعیت	آدرس	کلید
-------	-------	------	------

(ب) قالب کلمه وضعیت کانال

شمارش	پرچم‌ها	آدرس داده	کد فرمان
-------	---------	-----------	----------

(ج) قالب کلمه فرمان کانال

شکل ۲۱-۱۱ قالب‌های کلمات مربوط به I/O در IBM 370

1- Channel

قالب دستورالعمل I/O سه میدان دارد: کد عملیات، آدرس کانال، و آدرس وسیله. سیستم کامپیوتر ممکن است دارای تعدادی کانال باشد، که به هریک آدرسی اختصاص یافته است. بطور مشابه، هر کانال ممکن است به چند وسیله وصل شده باشد که به هر وسیله آدرسی اختصاص می یابد. کد عملیات یکی از هشت دستورالعمل I/O را مشخص می کند: شروع I/O، شروع سریع I/O، آزمایش I/O، پاک کردن I/O توقف I/O، توقف وسیله، آزمایش کانال، و ذخیره شماره شناسایی کانال. کانالی که آدرس دهی شده به هریک از دستورالعمل های I/O پاسخ می دهد و آن را اجرا می کند. همچنین یکی از چهار کد وضعیتی ممکن را در یک ثبات پردازنده بنام PSW (کلمه وضعیت پردازنده) قرار می دهد. CPU می تواند با واریسی کد وضعیت موجود در PSW نتیجه عمل I/O را معین کند. معنای چهار کد وضعیتی برای هر دستورالعمل I/O متفاوت است. اما در حالت کلی مشخص می کنند که کانال مشغول است یاخیر، عملیاتی است یاخیر.

قالب کلمه وضعیت کانال در شکل ۲۱-۱۱ (ب) نشان داده شده است. این کلمه همیشه در مکان 64 حافظه ذخیره می شود. میدان کلید^۱، یک مکانیزم حفاظتی است که برای جلوگیری از مراجعه غیرمجاز از سوی یک کاربر به اطلاعات متعلق به کاربر دیگر یا سیستم عامل بکار می رود. میدان آدرس در کلمه وضعیت، آدرس آخرین کلمه استفاده شده بوسیله کانال را نشان می دهد. میدان شمارش، تعداد باقیمانده را پس از پایان انتقال نشان می دهد. اگر انتقال با موفقیت انجام شود میدان شمارش حاوی مقدار صفر خواهد بود. میدان وضعیت شرایط وسیله و کانال و هرگونه خطایی را که در حین انتقال رخ داده باشد نشان می دهد.

تفاوت بین دستورالعمل های شروع I/O و شروع سریع I/O این است که اجرای دومی به صرف زمان کمتری از سوی CPU نیاز دارد. وقتی که کانال یکی از این دو دستورالعمل را دریافت کند، به خانه حافظه 72 برای برداشت اولین کلمه فرمان کانال (CCW) مراجعه می کند. قالب کلمه فرمان کانال در شکل ۲۱-۱۱ (ج) نشان داده شده است. میدان آدرس داده اولین آدرس بافر حافظه را مشخص می کند و میدان شمارش تعداد بایت هایی که در انتقال شرکت دارند را معین می نماید. میدان فرمان مشخص کننده یک عمل I/O و بیت های پرچم اطلاعات اضافی دیگر را برای کانال فراهم می کنند. میدان فرمان متعلق به کد عملی است که یکی از شش نوع کار اصلی I/O زیر را مشخص می نماید.

- ۱- نوشتن - انتقال داده از حافظه به وسیله I/O
- ۲- خواندن - انتقال داده از وسیله I/O به حافظه
- ۳- خواندن معکوس - خواندن نوار مغناطیسی بهنگام حرکت معکوس نوار
- ۴- کنترل - برای آغاز عملی که مستلزم انتقال داده نیست به کار می رود، مانند به عقب برگرداندن نوار، یا تغییر مکان مکانیزم دستیابی به دیسک^۲
- ۵- درک - به کانال اطلاع می دهد تا کلمه وضعیت کانال خود را به مکان 64 حافظه منتقل کند.

1- key

2- Channel Command word

۶- انتقال در کانال - در عوض دستورالعمل پرش بکار می‌رود. در اینجا میدان آدرس داده، آدرس کلمه فرمان بعدی را که کانال باید اجرا کند مشخص می‌نماید.

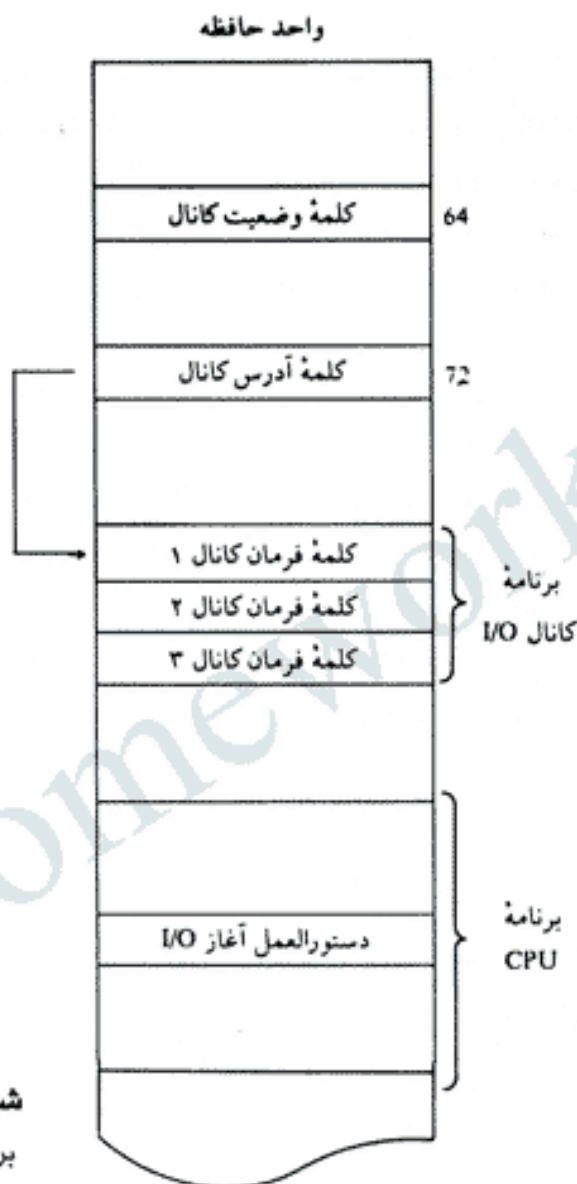
مثالی از برنامه کانال در جدول ۳-۱۱ نشان داده شده است. برنامه از سه کلمه فرمان تشکیل شده است. اولین کلمه موجب انتقال 60 بایت از حافظه با آدرس شروع 4000 به یک نوار مغناطیسی می‌گردد. دو کلمه فرمان بعدی عمل مشابهی را با بخش متفاوتی از حافظه و بایت شمارش انجام می‌دهند. شش پرچم در هر کلمه کنترل روابط دو جانبه‌ای را بین کلمات فرمان معین می‌کنند.

پرچم اول در اولین کلمه فرمان برابر 1 قرار داده می‌شود تا زنجیره‌ای شدن داده‌ها را مشخص کند. این امر منجر به ترکیب 60 بایت فرمان اول با 20 بایت فرمان بعدی آن و تشکیل یک رکورد 80 بایتی می‌شوند. این 80 بایت حتی اگر از دو بخش حافظه استفاده شده باشند بدون هرگونه جداسازی یا فاصله بر روی نوار نوشته می‌شوند. پرچم دوم در فرمان دوم 1 می‌گردد تا مشخص کننده زنجیره‌ای شدن فرمانها^۱ باشد. این پرچم کانال را مطلع می‌سازد که کلمه فرمان بعدی از همان وسیله I/O، یعنی در اینجا نوار، استفاده خواهد شد. کانال به دستگاه نوار اطلاع می‌دهد که شروع به وارد کردن یک فاصله رکورد روی نوار کند و سپس به خواندن فرمان بعدی از حافظه پردازد. سپس 40 بایت از سومین فرمان بصورت یک رکورد جداگانه بر روی نوار نوشته می‌شود. هر گاه همه پرچم‌ها صفر باشند، نشان دهنده پایان عملیات I/O برای وسیله خاص مورد نظر است.

نقشه‌ای از حافظه که همه اطلاعات مربوط به پردازش I/O را نشان می‌دهد در شکل ۲۲-۱۱ رسم شده است. عملیات وقتی آغاز می‌شود که برنامه CPU با یک دستورالعمل شروع I/O مواجه شود. سپس IOP به مکان 72 حافظه مراجعه کرده و کلمه آدرس کانال را از آنجا برمی‌دارد. این کلمه حاوی آدرس شروع برنامه کانال I/O است. سپس کانال کار خود را با اجرای برنامه مشخص شده با کلمات فرمان کانال ادامه می‌دهد. کانال در حین انتقال یک کلمه وضعیت را تشکیل و آن را در مکان 64 ذخیره می‌کند. به محض وقوع وقفه CPU می‌تواند برای کلمه وضعیت به مکان 64 حافظه مراجعه نماید.

جدول ۳-۱۱ مثالی از برنامه کانال برای IBM 370

شمارش	پرچم‌ها	آدرس	وقفه
60	100000	4000	نوشتن نوار
20	010000	6000	نوشتن نوار
40	000000	3000	نوشتن نوار



شکل ۱۱-۲۲ محل اطلاعات در حافظه
برای عملیات I/O در IBM 370

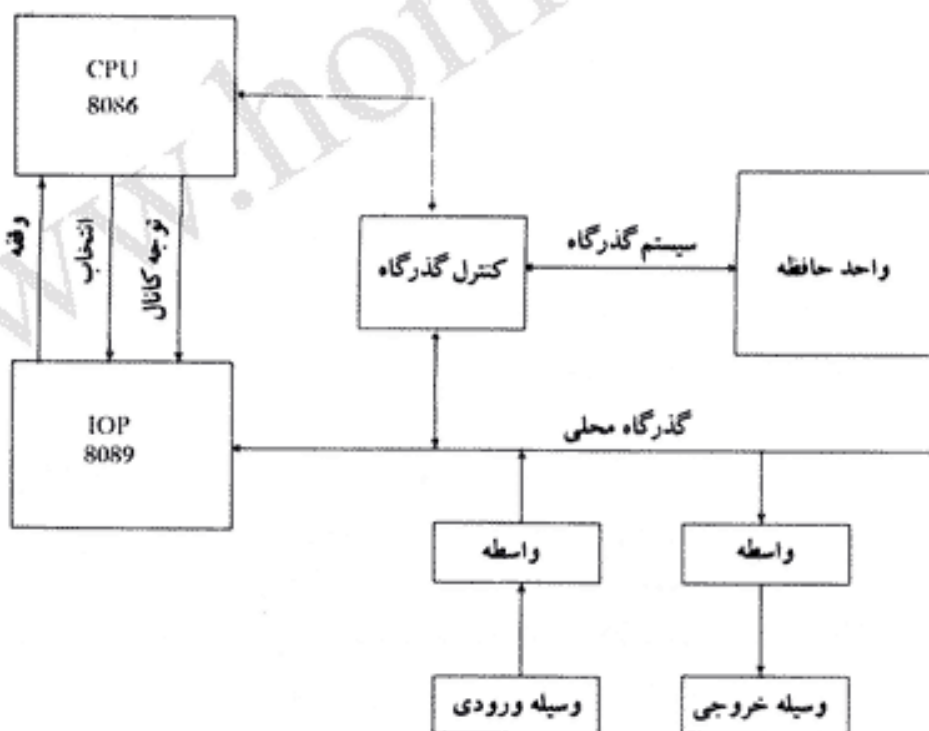
IOP مدل 8089 Intel

پردازنده I/O مدل 8089 در یک بسته مدار مجتمع 40 پایه ای قرار دارد. در داخل 8089 دو واحد مستقل به نام کانال وجود دارد. هر کانال مشخصات و ویژگی های عمومی یک واحد پردازنده را همراه با امکان یک کنترل کننده با دستیابی مستقیم به حافظه^۱ (DMAC) یکجا داراست. 8089 به صورتی طراحی شده است که در یک سیستم کامپیوتری، با ریزپردازنده Intel 8086 که نقش CPU را دارد، به عنوان IOP عمل کند. پردازنده 8086 با ایجاد یک پیام در حافظه که انجام کار مورد نظر را معین می نماید، اقدام به یک عمل I/O می کند. پردازنده 8089 این پیام را از حافظه می خواند، عمل را انجام می دهد، و

1- Direct Memory Access Controller

پس از اتمام کار به CPU اطلاع می دهد.

بر خلاف کانال IBM 370، که فقط دارای شش فرمان اصلی I/O است، پردازنده IOP 8089، پنجاه دستورالعمل اصلی دارد که می توانند روی بیت ها بطور جداگانه، روی بایت ها، یا روی کلمات 16 بیتی عمل کنند. IOP می تواند برنامه ها را مشابه با یک CPU اجرا نماید، جز اینکه مجموعه دستورالعمل ها خصوصاً برای انجام یک عمل ورود و خروج کارا انتخاب شده اند. این مجموعه دستورالعمل ها شامل دستورات عمومی انتقال داده، عملیات پایه حسابی و منطقی، اعمال انشعاب شرطی و غیرشرطی، و قابلیت فراخوانی و بازگشت از زیرروال است. این مجموعه، دستورالعمل های خاصی هم برای مقدار دهی اولیه برای انتقال DMA و صدور تقاضای وقفه برای CPU دارد. همچنین برای انتقال داده بین هر دو وسیله متصل به گذرگاه سیستم، مانند I/O به حافظه، حافظه به حافظه، یا I/O به I/O کارآیی خوبی دارد. یک سیستم ریزکامپیوتری که از جفت مدار مجتمع 8086/8089 استفاده می کند در شکل ۲۳-۱۱ نشان داده شده است. 8086 مثل یک CPU و 8089 مانند یک IOP عمل می کنند. هر دو واحد از طریق سیستم گذرگاهی که بوسیله اینتل گذرگاه چندگانه^۱ نامیده می شود، از حافظه مشترکی استفاده می کنند. IOP با استفاده از یک گذرگاه محلی با واحدهای واسطه که به وسایل I/O متصلند تبادل اطلاعات می کنند. CPU با فعال کردن خط توجه کانال^۲ ارتباط برقرار می کند خط انتخاب که بوسیله CPU بکار



شکل ۲۳-۱۱ بلاک دیاگرام سیستم ریزکامپیوتر 8086/8089 اینتل

1- Multibus

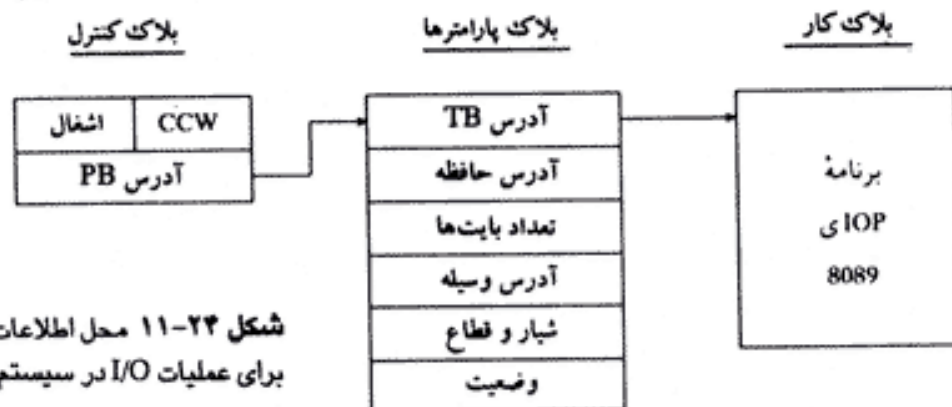
2- Channel Attention

می رود برای انتخاب یکی از دو کانال در 8089 بکار می رود. IOP توجه CPU را با ارسال یک تقاضای وقفه جلب می کند.

CPU و IOP، با نوشتن پیام برای یکدیگر در سیستم حافظه، باهم ارتباط برقرار می کنند. CPU مکان پیام را آماده کرده و با فعال کردن خط توجه کانال IOP را آگاه می سازد. IOP پیام را خوانده، عملیات I/O درخواستی را انجام می دهد، و برنامه کانال مناسب را اجرا می نماید. وقتی که کانال برنامه خود را پایان رساند، یک تقاضا وقفه برای CPU ارسال می دارد.

این روش ارتباط، متشکل از برنامه های تقسیم شده ای است که بلاک^۱ نامیده شده و طبق شکل ۱۱-۲۴ در حافظه ذخیره می شوند. هر بلاک حاوی اطلاعات کنترل و پارامترها و آدرس اشاره گر به بلاک بعدی است. آدرس بلاک کنترل در آغاز به هر کانال IOP داده می شود. پرچم اشغال بودن مشخص می کند که آیا IOP مشغول است یا آماده انجام یک عمل I/O جدید می باشد. CCW (کلمه فرمان کانال)^۲ توسط CPU برای مشخص کردن نوع عملی که IOP باید انجام دهد معین می شود. در 8089 در CCW مفهوم مشابهی با کلمه فرمان در کانال IBM ندارد. در اینجا CCW بیشتر شبیه یک دستورالعمل I/O است که یک عمل را برای IOP مشخص می کند، بلاک پارامتر حاوی داده های متغیری است که برنامه IOP برای اجرای کار خود از آن استفاده می کند و بلاک کار^۳ (تکلیف) حاوی برنامه واقعی است که باید در IOP اجرا شود.

CPU و IOP از طریق کنترل بلاک های پارامترها با یکدیگر کار می کنند. CPU پس از تست پرچم اشغال و اطمینان از در اختیار بودن IOP، استفاده از حافظه مشترک را بدست می گیرد. سپس اطلاعات لازم را در بلاک پارامترها پرمی کند و یک فرمان "شروع عمل" در CCW می نویسد. پس از برپایی بلاک تبادل اطلاعات^۴، CPU سیگنال "توجه کانال" را فعال می کند تا IOP را برای آغاز عمل I/O مطلع سازد. سپس CPU با برنامه دیگری بکار ادامه می دهد. IOP با قراردادن آدرس بلاک کنترل در شمارنده



شکل ۱۱-۲۴ محل اطلاعات در حافظه برای عملیات I/O در سیستم کامپیوتری 8086/8089 اینتل

- | | |
|---------------|-------------------------|
| 1- Block | 2- Channel Command Word |
| 3- Task Block | 4- Communication Block |

برنامه‌اش، به سیگنال توجه کانال پاسخ می‌دهد. I/O به بلاک کنترل مراجعه کرده و پرچم اشغال را 1 می‌کند. پس از آن، عمل موجود در CCW را واری می‌نماید. سپس آدرس PB (بلاک پارامتر) و آدرس TB (بلاک تکلیف)¹ را به ثبات داخلی I/O منتقل می‌سازد. I/O با استفاده از اطلاعات موجود در بلاک پارامتر شروع به اجرای برنامه در بلاک تکلیف می‌کند. واردهای موجود در بلاک پارامترها به وسیله I/O بستگی دارد. پارامترهای لیست شده در شکل ۲۴-۱۱ برای انتقال داده به دیسک مغناطیسی و یا از آن، مناسبند. آدرس حافظه، آدرس شروع یک بافر حافظه است. شمارش بایت تعداد بایت‌های انتقال یافته را معین می‌نماید. آدرس وسیله، وسیله I/O خاصی که قرار است بکار گرفته شود را مشخص می‌کند. شماره‌های شیار^۲ و قطاع داده را روی دیسک قرار می‌دهد. وقتی که عمل I/O پایان یابد، I/O بیت‌های وضعیت خود را در مکان کلمه وضعیت بلاک پارامتر قرار می‌دهد و به CPU سیگنال وقفه می‌فرستد. CPU به کلمه وضعیت رجوع کرده و انتقال موفقیت آمیز را واری می‌نماید.

۸-۱۱ تبادل اطلاعات سری

یک پردازنده تبادل داده‌ها یک پردازنده I/O است که داده را از پایانه‌های متعدد دور از هم که از طریق خطوط تلفن و دیگر خطوط ارتباطی به یکدیگر متصلند توزیع و یا جمع‌آوری می‌کند. این پردازنده نوعی پردازنده خاص I/O است که برای ارتباط مستقیم با شبکه‌های ارتباطی طراحی شده است. یک شبکه ارتباطی ممکن است از هر نوع وسیله متنوعی مانند چاپگر، وسایل نمایش محاوره‌ای، سنسورهای دیجیتال یا یک مرکز کامپیوتر دوردست متشکل باشد. با استفاده از یک پردازنده انتقال داده^۳، کامپیوتر می‌تواند بطور متناوب به بخشی از هر تقاضای شبکه سرویس بدهد و بنابراین رفتار ظاهری آن بصورتی باشد که به کاربران متعدد بطور همزمان سرویس می‌دهد. با این ترتیب کامپیوتر قادر خواهد بود تا بطور موثری در محیط‌های تقسیم زمانی^۴ بکار رود.

مهمترین اختلاف بین یک پردازنده I/O و یک پردازنده انتقال داده، نحوه ارتباط پردازنده با وسیله I/O است. یک پردازنده I/O از طریق یک گذرگاه مشترک I/O که از چندین خط داده و کنترل تشکیل شده است با وسایل I/O ارتباط برقرار می‌کند. تمام وسایل جانبی از گذرگاه بطور مشترک استفاده می‌کنند و از آن برای انتقال اطلاعات به و یا از پردازنده I/O استفاده می‌کنند. یک پردازنده ارتباطی (انتقال داده) از طریق یک جفت سیم با هر پایانه ارتباط برقرار می‌نماید. هر دو نوع اطلاعات داده و کنترل به فرم سری در آمده که این موجب سرعت انتقال بسیار کندتری می‌شود. وظیفه پردازنده انتقال داده، ارسال و جمع‌آوری اطلاعات دیجیتال به و یا از هر پایانه، تعیین اینکه اطلاعات از نوع داده است یا کنترلی و پاسخ به تمام تقاضاها طبق رویه‌های از پیش تعیین شده است. مسلماً پردازنده باید با CPU و حافظه هم بطور مشابه به هر پردازنده I/O ارتباط برقرار کند.

1- Task Block

2- Track

3- Data Communication Processor

4- Time - sharing

روش اتصال پایانه‌های دوردست به پردازنده انتقال داده از طریق خطوط تلفن یا سایر وسایل ارتباط خصوصی یا عمومی است. چون خطوط تلفن اصولاً برای تبادل صوت طراحی شده و کامپیوترها با سیگنال‌های دیجیتال با هم تبادل اطلاعات می‌کنند، نوعی تبدیل باید صورت گیرد. مبدل‌ها، دستگاه داده^۱، کوپل‌کننده صوتی^۲ و یا مودم^۳ (مخفف مودولاتور و دمودولاتور) خوانده می‌شوند. مودم سیگنال دیجیتال را به علائم صوتی برای ارسال بر روی خط تلفن تبدیل می‌کند. روش‌های مدولاسیون مورد استفاده و نیز درجه کیفیت رسانه‌های ارتباطی و سرعت‌های انتقال در بین آنها متفاوت است. بسته به روش ارسال پایانه دوردست، خط ارتباطی ممکن است به یک واسطه همزمان یا غیرهمزمان متصل باشد. واسطه غیرهمزمان، داده‌های سری را با بیت‌های شروع و توقف در هر کاراکتر مطابق شکل ۷-۱۱ دریافت می‌کند. این نوع واسطه مشابه واحد واسطه ارتباطی غیرهمزمان در شکل ۸-۱۱ است.

انتقال همزمان (همگام) بیت شروع و توقف^۴ را برای کادربندی کاراکترها استفاده نمی‌کند، و بنابراین از ارتباط جهت تبادل داده‌ها بصورت کاراکتری استفاده می‌شود. وسایلی که سرعت بالا دارند از انتقال همزمان برای رسیدن به این کارایی استفاده می‌کنند. مودم‌هایی که در انتقال همزمان بکار می‌روند، دارای ساعت درونی بوده و این ساعت‌ها با فرکانس انتقال در خط ارتباط تنظیم می‌شوند. برای عملکرد صحیح مودم، لازم است ساعت در فرستنده و گیرنده در تمام مدت همگام باشند. با این وجود خط ارتباطی فقط حاوی بیت‌های داده است که از آنها اطلاعات ساعت باید استخراج شود. همگامی فرکانس توسط مودم گیرنده از گذارهای سیگنال که در داده‌های دریافتی رخ می‌دهد حاصل می‌گردد. هر شیفت فرکانسی که ممکن است بین ساعت‌های گیرنده و فرستنده رخ دهد بطور مداوم با نگهداشتن ساعت فرستنده در فرکانس رشته بیت‌های رسیده تنظیم می‌شود. مودم داده دریافتی را همراه با ساعت فرستنده به واحد واسطه می‌فرستد. واسطه یا پایانه در سمت فرستنده هم از اطلاعات ساعت مودم خود استفاده می‌کند. به این ترتیب در فرستنده و گیرنده سرعت بیت یکسانی برقرار می‌شود.

برخلاف ارسال غیرهمگام که در آن هر کاراکتر را می‌توان جداگانه همراه با بیت‌های شروع و توقف آنها ارسال کرد، انتقال همگام برای حفظ همگامی باید پیام پیوسته‌ای را ارسال نماید. پیام شامل گروهی از بیت‌های ارسال شده سری مانند یک بلاک از داده است. کل بلاک همراه با کاراکترهای کنترلی خاص در ابتدا و انتهای بلاک ارسال می‌شود. کاراکترهای کنترل در ابتدای بلاک اطلاعات لازم را برای تفکیک بیت‌های دریافتی به صورت کاراکترهای مستقل فراهم می‌کنند.

یکی از وظایف پردازنده انتقال داده و ارسی خط‌های ارسال است. یک نوع خط را با ورسی بیت توازن در هر کاراکتر رسیده می‌توان تشخیص داد. روش دیگر که در پایانه‌های غیرهمگام با اپراتوری انسانی بکار می‌رود انعکاس کاراکتر است. کاراکتری که از صفحه کلید به کامپیوتر ارسال می‌شود بوسیله پردازنده تشخیص داده شده و سپس به پایانه چاپگر منعکس می‌شود. اگر کاراکتر چاپ شده همان

1- Data set

2- Acoustion Coupler

3- Modem

4- Start-Stop

کاراکتری که از فشار کلید حاصل شده است نباشد، اپراتور درمی یابد که در حین ارسال خطایی رخ داده است. در ارسال همگام، که در آن کل بلاک کاراکترها ارسال می شود، هر کاراکتر دارای یک بیت توازن برای وراسی توسط گیرنده است. پس از ارسال تمام بلاک، فرستنده یک کاراکتر اضافی که توازن را در طول پیام تشکیل می دهد ارسال می دارد. این کاراکتر وراسی تجمعی طولی¹ (LRC) نامیده شده و در واقع تجمع OR انحصاری تمام کاراکترهای ارسال شده است. ایستگاه گیرنده LRC را ضمن دریافت کاراکترها محاسبه می کند و آنرا با LRC ارسال شده مقایسه می نماید. برای پیام های بدون خطا، LRC محاسبه شده و دریافت شده با هم مساویند. اگر گیرنده در بلاک ارسال شده خطایی را بیابد، به فرستنده اطلاع می دهد تا یکبار دیگر همان اطلاعات را ارسال کند. روش بکار رفته دیگر برای وراسی خطاها در ارسال اطلاعات، وراسی تجمعی چرخشی² (CRC) است. این یک کد چند جمله های حاصل از بیت های پیام است که با پاس دادن آنها از یک ثبات شیفت حاوی تعدادی بیت های OR انحصاری به فرستنده پسخورد (برگشت³) حاصل می شود. این نوع کد برای آشکار سازی خطاهای پیاپی در کانال ارتباطی مناسب است.

انتقال داده بین دو نقطه می تواند به سه طریق انجام شود. یک طرفه⁴ نیمه دو طرفه و تمام دو طرفه. یک خط یک طرفه اطلاعات را فقط در یک جهت حمل می کند. این روش بندرت در خطوط ارتباطی بکار می رود زیرا گیرنده نمی تواند در صورت بروز خطا با فرستنده ارتباط برقرار کند. مثال هایی از این نوع ارتباط پیام های رادیوی و تلویزیونی است.

سیستم ارسال نیمه دو طرفه سیستمی است که قادر است اطلاعات را در هر دو جهت ارسال کند ولی در هر لحظه اطلاعات فقط در یک جهت منتقل می گردد. برای این روش یک جفت سیم لازم است. یک نوع متداول از این روش، استفاده از یک مودم بعنوان فرستنده و مودم دیگر بعنوان گیرنده است. وقتی که ارسال پیام در یک جهت پایان یابد، نقش مودم ها برای معکوس کردن جهت انتقال تعویض می گردد. زمان لازم برای تعویض خط سیستم نیمه دو طرفه⁵ از یک جهت به جهت دیگر زمان تحویل خواننده می شود. سیستم تمام دو طرفه قادر است که بطور همزمان داده را ارسال و دریافت کند. این عمل توسط خط چهار سیمی صورت می گیرد که در آن هر دو خط برای یکی از دو جهت در نظر گرفته شده است. راه دیگر، استفاده از یک مدار دو سیمه برای ارسال تمام دو طرفه است مشروط براین که طیف فرکانسی به دو بخش باند فرکانس غیرهمپوش⁶ تقسیم شود تا در یک جفت سیم بتوان کانال های ارسال و دریافت جداگانه بوجود آورد.

خطوط ارتباطی، مودم ها و سایر تجهیزات بکار رفته در ارسال داده ها بین دو یا چند ایستگاه خطوط

1- Longitudinal Redundancy Check

2- Cyclic Redundancy Check

3- Feedback

4- simplex

5- Turnaround

6- Nonoverlap

ارتباط داده^۱ نامیده می شود. ارسال منظم اطلاعات در یک خط ارتباط داده توسط یک قرارداد^۲ انجام می شود. قرارداد کنترل یک خط ارتباط داده مجموعه ای از قوانین است که کامپیوترها و پایانه های متصل بهم برای اطمینان از یک انتقال صحیح اطلاعات از آن پیروی می کنند. هدف از قرارداد ارتباط داده برقراری و اتمام ارتباط بین دو ایستگاه، شناسایی فرستنده و گیرنده، اطمینان از ارسال صحیح و بدون خطای همه پیام ها، و انجام همه عملیات کنترلی لازم در انتقال رشته ای از داده هاست. قراردادها برحسب روش کادربندی پیام ها^۳ به دو گروه عمده تقسیم می شوند. این دو گروه عبارتند از قرارداد مبتنی بر کاراکتر و قرارداد مبتنی بر بیت.

قرارداد مبتنی بر کاراکتر

قرارداد مبتنی بر کاراکتر بر اساس کد دودویی مجموعه کاراکترها ایجاد شده است. متداول ترین کد مورد استفاده کد اسکی ASCII (کد استاندارد آمریکایی برای تبادل اطلاعات) است. این کد 7 بیتی است و ۱۲۸ کاراکتر است که از ۹۵ کد گرافیکی و ۳۳ کاراکتر کنترلی تشکیل شده است. کاراکترهای گرافیکی شامل حروف بزرگ، کوچک، ده رقم اعداد و انواع سمبل های خاص است. لیستی از کاراکترهای ASCII در جدول ۱-۱۱ قابل ملاحظه است. کاراکترهای کنترل برای مسيردهی داده ها، آرایش متن با قالب مورد نظر، و صفحه آرایی^۴ متن چاپی بکار می روند. کاراکترهایی که ارسال داده ها را کنترل می کنند، کاراکترهای کنترل ارتباطات^۵ خوانده می شوند. این کاراکترها در جدول ۴-۱۱ لیست شده اند. هر کاراکتر 7 بیتی است و با یک سمبل سه حرفی معرفی شده است. نقش هر کاراکتر در کنترل ارسال داده بطور خلاصه در ستون نقش در جدول آمده است.

جدول ۴-۱۱ کاراکترهای کنترل ارتباطات اسکی

کد	سمبل	عملکرد	معنی
0010110	SYN	در حالت بیکار، همگامی تولید می کند	بیکار و همگام
0000001	SOH	سرآیند بلاک پیام	آغاز سرآیند
0000010	STX	فیل از بلاک متن قرار گیرد	آغاز متن
0000011	ETX	در پایان بلاک متن قرار گیرد	پایان متن
0000100	EOT	عمل ارسال را پایان می دهد	پایان ارسال
0000110	ACK	تصدیق مثبت	تصدیق
0010101	NAK	تصدیق منفی	تصدیق منفی
0000101	ENQ	استعلام روشن بودن پایانه	استعلام
0010111	ETB	پایان بلاک داده	پایان بلاک ارسال
0010000	DLE	کاراکتر کنترلی خاص	کنترل رابط داده

1- Data Link

2- Protocol

3- Message – Framing

4- Layout

5- Communication Control Character

کاراکتر SYN نقش همگام‌ساز را بین فرستنده و گیرنده بعهدده دارد. وقتی که از کد 7 بیتی اسکی همراه با بیت توازن فرد در با ارزشترین مکان استفاده شود، کاراکتر SYN دارای کد هشت بیتی 00010110 خواهد بود و این خاصیت را دارد که با استفاده از شیفت حلقوی، فقط پس از هشت بار چرخش خود را تکرار می‌کند. وقتی که فرستنده ارسال کاراکتر 8 بیتی را آغاز می‌کند، ابتدا چند کاراکتر را فرستاده و سپس اقدام به فرستادن پیام واقعی می‌کند. رشته بیت‌های متوالی اولیه توسط گیرنده پذیرفته شده و به منظور یافتن کاراکتر SYN و ارسی می‌شوند. به بیان دیگر، با هر پالس ساعت، گیرنده هشت بیت آخر دریافتی را ورسی می‌کند. اگر این هشت بیت با کاراکتر SYN مطابقت نکند، گیرنده بیت بعدی را می‌پذیرد، بیت بالا رتبه قبلی را حذف می‌نماید، و مجدداً هشت بیت آخر را برای یافتن کاراکتر SYN چک می‌کند. این عمل پس از هر پالس ساعت تکرار شده و بیت‌ها تا یافتن کاراکتر SYN دریافت می‌شوند. به محض تشخیص کاراکتر SYN، گیرنده کاراکتر را کادربندی می‌کند. از این پس گیرنده هر هشت بیت را شمارش کرده و آنرا بعنوان یک کاراکتر در نظر می‌گیرد. معمولاً گیرنده دو کاراکتر SYN متوالی را چک می‌کند تا هرگونه تردیدی در این مورد، که مبادا اولی در نتیجه سیگنال نویز روی خط باشد، را رفع کند. بعلاوه، وقتی که فرستنده بیکار است و هیچ پیام یا کاراکتری را ارسال نمی‌کند. رشته پیوسته‌ای از کاراکتر SYN را ارسال می‌نماید. گیرنده این کاراکترها را بعنوان شرایطی برای همگام سازی خطا می‌شناسد و به یک حالت بیکار و همگام می‌رود. در این حالت، هر دو واحد، همزمانی بیت و کاراکتر را حتی اگر اطلاعات معنی داری مخایره نشود حفظ می‌کنند.

پیام‌ها از طریق خط ارتباط داده همراه با قالبی متشکل از میدان سرآیند¹، میدان متن، میدان ورسی خط ارسال می‌شوند. نمونه‌ای از قالب پیام برای قرارداد مبتنی بر کاراکتر در شکل ۱۱-۲۵ نشان داده شده است. دو کاراکتر SYN همگامی صحیح را در آغاز پیام تضمین می‌کنند. بدنبال کاراکترهای SYN، سرآیند قرارداد، که با کاراکتر SOH² شروع می‌شود (شروع سرآیند). سرآیند از اطلاعات آدرس و کنترلی تشکیل شده است. کاراکتر STX سرآیند را پایان داده و شروع متن را معین می‌نماید. بخش متن پیام از نظر طول متغیر بوده و می‌تواند حاوی هر کاراکتر اسکی بجز کاراکترهای کنترلی باشد. میدان متن با ETX پایان می‌پذیرد. آخرین میدان، یک کاراکتر ورسی بلاک³ (BCC) است که برای ورسی خطا بکار می‌رود. این میدان معمولاً ورسی تجمعی طولی (LRC) یا ورسی تجمعی چرخشی (CRC) است. گیرنده پیام را دریافت کرده و BCC خود را محاسبه می‌کند. اگر BCC ارسالی با BCC محاسبه شود

SYN	SYN	SOH	سرآیند	STX	متن	ETX	BCC
-----	-----	-----	--------	-----	-----	-----	-----

شکل ۱۱-۲۵ نوعی قالب پیام در قرارداد مبتنی بر کاراکتر

1- Header field

2- Start of Header

3- Block Check Character

بوسیله گیرنده همخوانی نداشته باشد، گیرنده با ارسال یک کاراکتر تصدیق منفی^۱ (NAK) واکنش نشان می دهد. پیام، سپس مجدداً ارسال و واریسی می شود. معمولاً قبل از اینکه خط اشکال دار فرض شود، ارسال مجدد پیام چند بار تکرار می شود. هنگامی که BCC ارسالی؛ محاسبه شده بوسیله گیرنده مطابقت نماید، پاسخ تصدیق مثبت بوده و کاراکتر ACK استفاده می شود.

مثال ارسال داده

برای درک عملکرد یک پردازنده انتقال داده، اجازه بدهید روشی را که یک پایانه و پردازنده با هم ارتباط برقرار می کنند مثال بزنیم. تبادل اطلاعات بین واحد حافظه و CPU مشابه با هر پردازنده I/O است. نمونه پیامی که ممکن است از یک پایانه به پردازنده ارسال شود در جدول ۵-۱۱ دیده می شود. نگاهی به این پیام مشخص می نماید که تعدادی کاراکترهای کنترلی برای شکل دادن به پیام بکار رفته است. هر کاراکتر، از جمله کاراکترهای کنترلی بصورت کد دودویی هشت بیتی شامل ۷ بیت کد اسکی بعلاوه یک بیت توازن فرد در با ارزشترین مکان، بطور سری ارسال می شوند. برای همگامی فرستنده و گیرنده دو کاراکتر SYN بکار رفته است. سرآیند با کاراکتر SOH شروع می شود و به دو کاراکتر که آدرس پایانه را معین می کنند ختم می گردد. در این مثال خاص، آدرس T₄ است، ولی عموماً می تواند مجموعه ای از دو یا چند کاراکتر گرافیکی باشد. کاراکتر STX سرآیند را ختم کرده و شروع ارسال متن را معین می نماید. داده مستن در اینجا "request of balance number 1234" است. تک تک کاراکترهای این پیام

جدول ۵-۱۱ نمونه پیام ارسالی از یک پایانه به پردازنده

کد	نماد	شرح
0001 0110	SYN	اولین کاراکتر Sync
0001 0110	SYN	دومین کاراکتر Sync
0000 0001	SOH	آغاز سرآیند
0101 0100	T	آدرس پایانه T ₄ است
0011 0100	4	
0000 0010	STX	آغاز ارسال متن
0101 0010		
0100 0101	request	متن ارسالی درخواست اعلام موجودی
.	balance	شماره حساب ۱۲۳۴
.	of account	
.	No. 1234	
1011 0011		
0011 0100		
1000 0011	ETX	پایان ارسال متن
0111 0000	LRC	کاراکتر توازن طولی

1- Negative Acknowledge

در این جدول لیست نشده اند زیرا فضای بسیاری را اشغال می کنند. با این وجود، باید توجه داشت که هر کاراکتر در پیام دارای کد 8 بیتی است و هر بیت بطور سری ارسال می شود. کاراکتر کنترل EXT پایان کاراکترهای متن را مشخص می نماید. کاراکتر بعدی که بدنبال ETX آمده است یک کاراکتر واریسی طولی، LRC، است هر بیت در این کاراکتر بیت توازنی است که از محاسبه تمام بیت های واقع در یک ستون در بخش کد از جدول محاسبه می شود.

پردازنده انتقال داده، این پیام را دریافت کرده و به تحلیل آن می پردازد. پردازنده تشخیص می دهد که پایانه T4 است و متن پیام را ذخیره می کند. در حین دریافت کاراکترها، پردازنده توازن را در هر کاراکتر واریسی کرده و توازن طولی را نیز محاسبه می کند. LRC محاسبه شده با LRC کاراکتر دریافتی مقایسه می شود. اگر هر دو با هم مطابقت داشته باشند، یک تصدیق مثبت (ACK) به پایانه بازگردانده می شود. اگر عدم تطابق وجود داشته باشد، یک تصدیق منفی به پایانه برگردانده می شود، که یک ارسال مجدد از همان بلاک را آغاز خواهد کرد. اگر پردازنده پیام را بدون خطا بیابد، آنرا به حافظه انتقال داده و CPU را وقفه می دهد. هنگامی که CPU وقفه را تصدیق نماید، پیام را تحلیل کرده و متن پیام را برای پاسخ به تقاضا آماده می کند. CPU یک دستورالعمل را برای پردازنده انتقال داده می فرستد تا پیام را به پایانه ارسال دارد. نمونه پاسخ از پردازنده به پایانه در جدول ۶-۱۱ لیست شده است. پس از دو کاراکتر SYN، پردازنده پیام قبلی را با یک کاراکتر ACK تصدیق می کند. خط با کاراکتر SYN در حالت بیکاری قرار

جدول ۶-۱۱ نمونه بیان ارسالی از پردازنده به پایانه

کد	نماد	شرح
0001 0110	SYN	اولین کاراکتر Sync
0001 0110	SYN	دومین کاراکتر Sync
1000 0110	ACK	پردازنده دریافت پیام قبلی را تصدیق می کند
0001 0110	SYN	خط بیکار است
.	.	.
0001 0110	SYN	خط بیکار است
0000 0001	SOH	آغاز سرآیند
0101 0100	T	آدرس پایانه T4 است
0011 0100	4	
0000 0010	STX	آغاز ارسال متن
1100 0010		
1100 0001	balance	متن ارسالی پاسخی از سوی کامپیوتر
.	is	است که موجودی حساب را می دهد
.	\$100.00	
.		
1011 0000		
1000 0011	ETX	پایان ارسال متن
1101 0101	LRC	کاراکتر توازن طولی

می‌گیرد و منتظر دریافت پاسخ می‌ماند. پیام دریافتی از CPU توسط پردازنده با گنجاندن کاراکترهای کنترلی لازم در ابتدا و انتهای متن به قالب صحیحی درمی‌آید. پیام دارای سرآیند SOH و آدرس پایانه T4 برای پایانه است. متن پیام پایانه را مطلع می‌سازد که بالانس \$100 است. یک کاراکتر LRC محاسبه شده و به پایانه ارسال می‌شود. اگر پایانه با کاراکتر NAK پاسخ دهد، پردازنده پیام را مجدداً ارسال می‌کند. مادامی که پردازنده مراقب این پایانه است پردازش دیگر ترمینال‌ها را هم انجام می‌دهد. چون کاراکترها بصورت سری دریافت می‌شوند، جمع‌آوری 8 بیت از یک کاراکتر زمان معینی لازم دارد. در حین این جمع‌آوری پردازنده سایر خطوط ارتباطی را مولتی‌پلکس می‌نماید و هر یک را به نوبت سرویس می‌دهد. سرعت دورترین ترمینال‌ها در مقایسه با سرعت پردازنده فوق‌العاده کم است. این خصوصیت اجازه می‌دهد تا چند کاربر، مولتی‌پلکس شده و به کارایی بهتری در سیستم اشتراک زمانی^۱ دستیابی شود. این خصوصیت همچنین اجازه می‌دهد تا چندین کاربر بطور همزمان کار کنند و هر یک با سرعت واکنش معمولی انسان نمونه‌برداری شوند.

شفافیت داده‌ها^۲

قرارداد مبتنی بر کاراکتر در واقع برای ارتباط با صفحه کلید، چاپگر و وسایل نمایشی که کاراکترهای الفبا عددی را بکار می‌برند تهیه شد. با توسعه زمینه ارتباطات، لازم شد تا اطلاعات دودویی که متن اسکی نبودند ارسال شوند. این نیاز، بعنوان مثال وقتی که دو کامپیوتر دور از هم برنامه‌ها و داده‌هایی را به یکدیگر با یک کانال ارتباطی ارسال کنند، وجود دارد. در قرارداد مبتنی بر کاراکتر، الگوی بیتی در متن پیام مشکل‌ساز می‌شود. این بدان علت است که هر الگوی 8 بیتی متعلق به کاراکتر کنترل انتقال ممکن است اشتباه تفسیر شود. بعنوان مثال، اگر داده دودویی در بخش متن پیام دارای الگوی هشت بیتی 10000011 باشد، گیرنده آنرا بعنوان یک کاراکتر EXT تفسیر کرده و فرض خواهد کرد که انتهای میدان متن فرارسیده است. وقتی که بخش متن پیام از نظر طول متغیر و حاوی بیت‌هایی باشد که باید بدون ارجاع به هر کد خاصی پذیرفته شوند گوئیم که حاوی داده‌های شفاف^۳ است. این ویژگی مستلزم این است که مدار منطقی تشخیص کاراکتر گیرنده غیرفعال شود تا الگوی داده‌های موجود در میدان متن تصادفاً بعنوان اطلاعات کنترل ارتباطات تعبیر نگردد.

شفافیت داده در قرارداد مبتنی بر کاراکتر بوسیله گنجاندن کاراکتر چشم‌پوشی از ارتباط داده^۴ DLE قبل از هر کاراکتر کنترل ارتباط حاصل می‌شود. بنابراین، سرآیند با دو کاراکتر DLE SOH شروع خواهد شد، و میدان متن هم با دو کاراکتر DLE ETX خاتمه می‌یابد. اگر الگوی بیتی DLE برابر 00010000 در بخش متن پیام ظاهر شود، فرستنده یک DLE دیگر بدنبال آن می‌گنجانند. گیرنده تمام DLE‌ها را حذف و الگویی هشت بیتی بعدی را واریسی می‌کند. اگر این الگو یک DLE دیگر باشد، گیرنده آن را

1- Time – Sharing

2- Data Transparency

3- Transparent data

4- Data link escape

بخشی از متن تصور کرده و به دریافت متن ادامه می‌دهد. در غیر اینصورت گیرنده الگوی هشت بیتی بعدی را یک کاراکتر کنترل ارتباطی تلقی می‌نماید.

کارایی دستیابی به شفافیت داده بوسیله کاراکتر DLE پایین و پیاده‌سازی آن هم تا حدی پیچیده است. بنابراین قراردادهای دیگری برای کارآتر کردن ارسال داده‌های شفاف تهیه شده است. یکی از این قراردادها که بوسیله کمپانی Digital Equipment بکار رفته است، بکارگیری میدان شمارش بایت است که تعداد بایت‌های پیامی را که بدنبال آن ارسال خواهد شد مشخص می‌کند. باین ترتیب گیرنده باید تعداد بایت‌های دریافتی را تا پایان میدان متن بشمرد. قراردادی که بیش از همه برای حل مشکل شفافیت داده و سایر مشکلات مربوط به قرارداد مبتنی بر کاراکتر بکار رفته است، قرارداد مبتنی بر بیت است.

قرارداد مبتنی بر بیت^۱

قرارداد مبتنی بر بیت در میدان کنترل خود از کاراکترها استفاده نمی‌کند و مستقل از هر کد بخصوص است. این قرارداد امکان ارسال رشته‌ای از بیت‌های سری را با هر طول، بدون در نظر گرفتن مرزهایی برای کاراکترها فراهم می‌سازد. پیام‌ها با قالب خاصی بنام کادر سازماندهی می‌شوند. یک کادر علاوه بر میدان اطلاعات حاوی میدان‌های آدرس، کنترل و ارسی خطاست. مرزهایی کادر از عدد هشت بیتی خاصی که پرچم نامیده می‌شود معین می‌گردد. مثالی‌هایی از قرارداد مبتنی بر بیت کنترل، ارتباط داده همگام^۲، SDLC، است که بوسیله IBM، قرارداد کنترل ارتباط داده سطح بالا^۳ HDLC بکار رفته بوسیله سازمان بین‌المللی استاندارد و رویه کنترل پیشرفته مخایره داده، ADCCP، مورد استفاده انستیتو ملی استانداردهای آمریکا می‌باشد.

هر خط انتقال داده حداقل از دو ایستگاه که در انتقال شرکت دارند تشکیل شده است. ایستگاهی که مسئولیت ارتباط داده را بعهده دارد و زمانی را برای کنترل ارتباط صادر می‌کند ایستگاه اصلی^۴ خوانده می‌شود. ایستگاه دیگر ایستگاه فرضی^۵ یا ثانویه نامیده می‌شود. قراردادهای مبتنی بر بیت وجود یک ایستگاه اصلی و یک یا چند ایستگاه فرعی را فرض می‌کند. کلیه انتقال‌های داده از ایستگاه اصلی به ایستگاه فرعی، یا از یک ایستگاه فرعی به ایستگاه اصلی است.

قالب کادر برای قرارداد مبتنی بر بیت در شکل ۲۶-۱۱ نشان داده شده است. کادر با پرچم هشت بیتی 01111110 و بدنبال آن رشته آدرس و کنترل است. میدان اطلاعات محدودیتی در قالب ندارد و هر طولی می‌تواند داشته باشد. میدان و ارسی کادر یک رشته CRC (وارسی چرخشی^۶) مورد استفاده در کشف خطا در ارسال است. پرچم انتهایی به ایستگاه گیرنده اطلاع می‌دهد که اطلاعات رسیده تا آن لحظه بیت‌های CRC است. کادر انتهایی ممکن است بوسیله کادری دیگر، پرچمی دیگر، یا یک

1- Bit-Oriented protocol 2- Synchronous Data link Control

3- High-Level Data Link Control

4- Primary station

5- Secondary station

6- Cyclic Redundancy Check

پرچم 01111110	آدرس 8 بیت	کنترل 8 بیت	اطلاعات هر تعداد بیت	وارسی کادر 16 بیتی	پرچم 01111110
------------------	---------------	----------------	-------------------------	-----------------------	------------------

شکل ۲۶-۱۱ قالب کادر برای قرارداد مبتنی بر بیت

رشته‌های متوالی از 1 ها دنبال شود. وقتی دو کادر دنبال هم بیایند پرچم فی مابین دو کادر، هم پرچم انتهایی کادر اول و هم پرچم شروع کادر بعدی است. اگر اطلاعاتی تبادل نشود، فرستنده یک سری پرچم را برای حفظ خط در وضعیت فعال خود ارسال می‌کند. با وقوع 15 بیت متوالی 1، یا بیشتر، خط فعال تصور می‌شود. کادرهایی که دارای پیام‌های کنترل معین نیستند بدون میدان اطلاعات ارسال می‌گردند. یک کادر باید حداقل 32 بیت بین دو پرچم برای میدان‌های آدرس، کنترل، و وارسی کادر داشته باشد. طول حداکثر به شرایط کانال ارتباطی و قابلیت‌های آن در ارسال پیام‌های طولانی بدون خطا وابسته است. برای جلوگیری از وقوع یک پرچم در میانه یک کادر، در قرارداد مبتنی بر بیت از روشی بنام درج صفر^۱ استفاده می‌شود. در این روش ایستگاه فرستنده پس از هر پنج 1 متوالی، یک صفر را ارسال می‌کند. گیرنده همواره هر صفری را که پس از پنج 1 آمده باشند حذف می‌کند. بنابراین الگوی بیتی 01111111 بصورت 01111101 ارسال گشته و بوسیله گیرنده پس از حذف 0 درج شده، بصورت اولیه‌اش ذخیره می‌گردد. در نتیجه به هیچ وجه الگوی 01111110 بین ابتدا و انتهای خط ارسال نمی‌گردد.

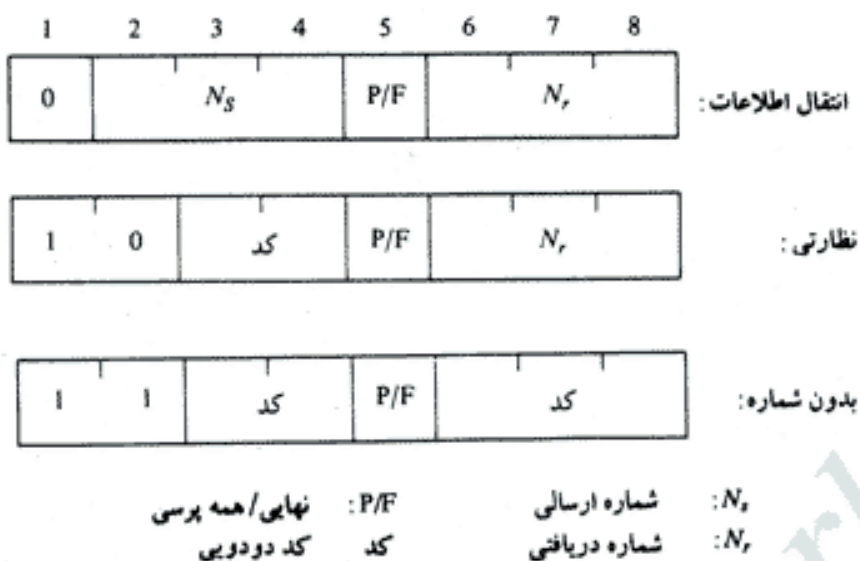
پس از پرچم، آدرس قرارداد، که بوسیله ایستگاه اصلی برای تشخیص آدرس ایستگاه ثانویه بکار می‌رود. وقتی که یک ایستگاه فرعی، یا ثانویه، کادری را ارسال می‌نماید، آدرس به کادر اصلی اطلاع می‌دهد که کدام ایستگاه فرعی کادر را ایجاد کرده است. یک میدان آدرس هشت بیتی می‌تواند تا 256 آدرس را مشخص کند. برخی قراردادهای مبتنی بر بیت امکان استفاده از میدان آدرس وسیعتری را در اختیار می‌گذارند. برای حصول به این امکان، کم‌ارزشترین بیت بایت آدرس فعلی، در صورت وجود بایت بعدی، صفر می‌گردد. تشخیص آخرین بایت آدرس با یافتن یک 1 در کم‌ارزشترین بیت آن آدرس امکان‌پذیر می‌باشد.

پس از میدان آدرس، میدان کنترل قرارداد. میدان کنترل می‌تواند در سه قالب مختلف مطابق شکل ۲۷-۱۱ وجود داشته باشد. قالب انتقال اطلاعات برای ارسال داده‌های معمولی بکار می‌رود. هر کادری که با این قالب ارسال گردد حاوی شمارش‌های ارسال و دریافت است. ایستگاهی که کادرهای متوالی را می‌فرستد هر کادر را شمارش و شماره‌گذاری می‌کند. این شمارش بوسیله ارسال شمار^۲ N_s شمرده می‌شود. ایستگاهی که کادرهای متوالی را دریافت می‌نماید، هر یک از کادرهای بدون خطای دریافتی را می‌شمارد. تعداد این کادرهای دریافتی را دریافت شمار^۳ N_r می‌شمارد. هر وقت کادری وارسی می‌شود و مشخص شود خطایی رخ نداده است، N_r افزایش می‌یابد. گیرنده، کادرهای اطلاعاتی شماره‌دار

1- Zero Insertion

2- Send Count

3- Recieve Count



شکل ۲۷-۱۱ قالب میدان کنترل در قرارداد مبتنی بر بیت

پذیرفته شده را با برگرداندن شمار N_r به ایستگاه فرستنده تأیید می‌کند. بیت P/F توسط ایستگاه اصلی برای همه پرسى از یک ایستگاه برای تقاضای اقدام به ارسال به کار می‌رود. این بیت بوسیله ایستگاه فرعی برای شناسایی آخرین کادر ارسالی مورد استفاده قرار می‌گیرد. بنابراین میدان P/F وقتی که ایستگاه اصلی در حال ارسال باشد، P (بمعنی همه پرسى^۱) و وقتی که ایستگاه فرعی در حال ارسال باشد، F (بمعنی نهایی^۲) خوانده می‌شود. هر کادر ارسالی از ایستگاه اصلی به فرعی دارای بیت P برابر با 0 است. وقتی که ایستگاه اصلی بکار خود پایان دهد و آماده پاسخ ایستگاه فرعی گردد، P برابر 1 می‌گردد. سپس ایستگاه فرعی شروع به ارسال تعدادی کادر می‌کند که در آنها بیت F برابر 0 است. وقتی که ایستگاه دوم آخرین کادر را ارسال نمود، بیت F را 1 می‌کند. بنابراین، بیت P/F برای تعیین ختم ارسال داده بکار می‌رود.

قالب نظارتی میدان کنترل با 0 یا 1 بودن دو بیت اول مشخص می‌شود. دو بیت بعدی نوع فرمان را مشخص می‌نماید. بدنبال آن بیت P/F و کادر شمار رشته دریافتی^۳ آمده است. کادرهای قالب نظارت حامل میدان اطلاعات نیستند. این کادرها برای کمک در انتقال اطلاعات استفاده می‌شوند، بدین ترتیب که آنها پذیرش کادرهای قبلی حاوی اطلاعات را تأیید می‌کنند، شرایط آمادگی یا مشغولیت را حمل می‌نمایند، و خطاهای شمارش کادر را گزارش می‌نمایند.

قالب بدون شماره با 11 بودن دو بیت اول مشخص می‌شود. پنج بیت موجود در این قالب می‌تواند تا 32 فرمان و پاسخ را معین نمایند. ایستگاه اصلی از میدان کنترل برای مشخص کردن یک فرمان برای

-
- 1- Poll
 - 2- Final
 - 3- Receive sequence frame Count

ایستگاه فرعی استفاده می‌کند. ایستگاه فرعی میدان را برای ارسال یک پاسخ به ایستگاه اصلی بکار می‌گیرد. کادرهای قالبی که شماره ندارند برای شروع عملیات ارتباط^۱، گزارش خطاهای رویه‌ای^۲، قرار دادن ایستگاه‌ها در حالت قطع، و سایر عملیات کنترلی ارتباط داده بکار می‌روند.

مسائل

۱۱-۱ آدرس‌های اختصاص یافته به چهار ثبات واسطه I/O در شکل ۲-۱۱ برابر با معادل دودویی اعداد 12، 13، 14 و 15 است. مدار خارجی لازم که باید بین یک آدرس I/O هشت بیتی از CPU و در ورودی‌های CS، RS1 و RSO از مدار واسطه وصل شوند را نشان دهید.

۱۱-۲ شش واحد واسطه از نوعی که در شکل ۲-۱۱ نشان داده شده است به یک CPU که از یک آدرس I/O هشت بیتی استفاده می‌کند متصل‌اند. هر یک از شش ورودی انتخاب تراشه^۳ CS به خط آدرس متفاوتی متصل است، بنابراین خط با ارزشتر آدرس به ورودی CS اولین واحد واسطه و ششمین خط آدرس به ششمین واحد واسطه متصل است. دو خط آدرس کم‌ارزشتر به RS1 و RSO از هر شش واحد واسطه متصل‌اند. آدرس هشت بیتی هر یک از ثبات‌های هر واسطه را معین کنید.

۱۱-۳ چهار وسیله جانبی که خروجی قابل درکی برای انسان داشته باشد نام ببرید.

۱۱-۴ نام کامل خود را به ASCII با استفاده از هشت بیت برای هر کاراکتر و یک 0 در سمت چپ‌ترین مکان بنویسید. بین قسمت‌های مختلف نام یک فاصله بگذارید و نیز در صورت استفاده از مخفف، پس از آن نقطه بگذارید.

۱۱-۵ اختلاف بین I/O مجزا با I/O نگاهشت چیست؟ مزایا و معایب هر یک را بگویید.

۱۱-۶ مشخص کنید که کدام یک از موارد زیر یک فرمان کنترل، وضعیت یا انتقال داده می‌باشد.

الف) گذر از دستورالعمل بعدی اگر پرچم 1 باشد.

ب) جستجوی یک رکورد خاص بر روی یک دیسک مغناطیسی

ج) واریسی آماده بودن یا نبودن وسیله I/O

د) حرکت کاغذ چاپگر به ابتدای صفحه بعد

ه) خواندن ثبات وضعیت واسطه

۱۱-۷ یک واحد واسطه تجاری برای خطوط دست‌دهی مربوط به انتقال داده‌ها از وسیله I/O به واحد واسطه از نام‌های مختلفی استفاده می‌کند. خط دست‌دهی ورودی واسطه STB (مخفف

1- Link Function

2- Reporting Procedural Errors

3- Chip Select

استروب یا فعال‌ساز^۱) و خط دست‌دهی خروجی واسطه IBF (مخفف پر بودن بافر ورودی^۲) نام‌گذاری شده است. وجود یک سیگنال سطح بالا روی IBF نشان می‌دهد که داده توسط واسطه پذیرفته شده است. پس از سیگنال خواندن I/O از طرف CPU از طریق ثبات داده، IBF به سطح پایین می‌رود.

الف) یک بلاک دیاگرام که CPU، واسطه و وسیله I/O همراه با اتصال‌های لازم بین آنها را نشان دهد رسم کنید.

ب) یک دیاگرام زمانی برای انتقال دست‌دهی رسم کنید.

ج) برای انتقال از وسیله به واسطه و از واسطه به CPU یک چارت دنباله رخدادها^۳ را رسم کنید. ۱۱-۸ یک CPU با ساعت 20MHz به یک واحد حافظه که زمان دسترسی آن 40ns است متصل است. دیاگرام‌های زمانی خواندن و نوشتن آنها را با استفاده از فعال‌کننده‌های READ و WRITE تهیه کنید. در دیاگرام زمانی آدرس را هم نشان دهید.

۱۱-۹ مدار واسطه ارتباطی غیرهمگام شکل ۱۱-۸ بین CPU و یک چاپگر متصل شده است. فلوجارتی رسم کنید که دنباله عملیات ارسال کاراکتر از CPU به چاپگر را نشان دهد.

۱۱-۱۰ حداقل شش وضعیت مختلف را برای 1 کردن بیت‌های ثبات وضعیت در یک واسطه ارتباطی غیرهمگام ارائه دهید.

۱۱-۱۱ اگر واسطه به پایانه‌ای متصل باشد که به یک بیت توقف نیاز دارد، چند بیت در ثبات شیفت فرستنده شکل ۱۱-۸ وجود دارد؟ با استفاده از کد اسکی توازن زوج بیت‌های ثبات شیفت را وقتی که حرف W ارسال شود مشخص کنید.

۱۱-۱۲ بر روی یک خط 1200-Baud با هر یک از روش‌های زیر چند کاراکتر در هر ثانیه ارسال می‌شود (کد کاراکترها را هشت بیتی فرض کنید).

الف) ارسال سری همگام (همزمان)

ب) ارسال سری غیرهمگام (با دو بیت توقف)

ج) ارسال سری غیرهمگام با یک بیت توقف

۱۱-۱۳ اطلاعاتی با سرعت m بایت در ثانیه به یک بافر FIFO وارد می‌شود. اطلاعات پاک شده در آن نیز n بایت در ثانیه است. حداکثر ظرفیت بافر ۱۲ بایت است.

الف) اگر $m > n$ باشد چه مدت طول می‌کشد تا بافر خالی، پر شود.

ب) اگر $m < n$ باشد چقدر طول می‌کشد تا بافر پر، خالی شود.

ج) اگر $m = n$ باشد آیا بافر FIFO لازم است یا خیر؟

۱۱-۱۴ بیت‌های ثبات کنترل FIFO در شکل ۱۱-۹ عبارتند از $F_1F_2F_3F_4 = 0011$. رشته عملیات داخلی را هرگاه یک قلم از داده از FIFO حذف و سپس داده جدیدی وارد شود معین کنید.

1- Strobe

2- Input Buffer Full

3- Sequence-of events Chart

۱۱-۱۵ مقادیر "ورودی آماده" و "خروجی آماده" و بیت‌های کنترل F_1 تا F_4 در شکل ۹-۱۱ در وضعیت‌های زیر چیست؟

الف) وقتی بافر خالی است.

ب) وقتی بافر پر است.

ج) وقتی بافر حاوی دو قلم داده است.

۱۱-۱۶ یک بلاک دیاگرام مشابه شکل ۱۰-۱۱ را برای انتقال داده از CPU به یک واسطه و سپس به یک وسیله I/O رسم کنید. رویه‌ای برای 1 کردن و 0 کردن بیت پرچم ارائه دهید.

۱۱-۱۷ با استفاده از پیکربندی شکل مسئله ۱۱-۱۶ فلوچارتی (مشابه با شکل ۱۱-۱۱) برای برنامه CPU جهت خروجی داده بدست آورید.

۱۱-۱۸ مزیت اصلی استفاده از انتقال داده به کمک وقفه در برابر انتقال تحت کنترل برنامه بدون استفاده از وقفه چیست؟

۱۱-۱۹ در بیشتر کامپیوترها وقفه فقط پس از اجرای دستورالعمل جاری تشخیص داده می‌شود. امکان پذیرش وقفه را در هر زمانی در حین اجرای دستورالعمل بررسی کنید. در مورد مشکلاتی که ممکن است بروز کند بحث کنید.

۱۱-۲۰ در سیستم وقفه اولویت‌دار چرخشی شکل ۱۲-۱۱، هرگاه پس از درخواست وقفه وسیله 2 به CPU ولی قبل از تصدیق وقفه به وسیله CPU، وسیله 1 درخواست وقفه کند چه اتفاقی می‌افتد.

۱۱-۲۱ کامپیوتری را بدون سخت‌افزار وقفه اولویت‌دار در نظر بگیرید. هر یک از چند منبع می‌توانند به کامپیوتر وقفه بدهند، در هر درخواست وقفه منجر به ذخیره آدرس برگشت و انشعاب به یک روال مشترک وقفه می‌گردد. توضیح دهید که چگونه می‌توان در برنامه سرویس‌دهی وقفه، اولویت برقرار کرد.

۱۱-۲۲ با استفاده از روش‌های طراحی مدارهای ترکیبی، عبارات بولی ارائه شده در جدول ۲-۱۱ را برای کد گذار اولویت بدست آورید. نمودار منطقی مدار را رسم کنید.

۱۱-۲۳ سخت‌افزار وقفه اولویت‌دار موازی را برای سیستمی با هشت منبع وقفه طراحی کنید.

۱۱-۲۴ جدول درستی 8×3 اولویت را بدست آورید. فرض کنید که سه خروجی xyz از انکدر اولویت برای تهیه یک آدرس بردار به شکل 101xyz00 بکار روند. هشت آدرس بردار را با شروع از آن که بالاترین اولویت را دارد بنویسید.

۱۱-۲۵ در شکل ۱۴-۱۱ چه کاری باید انجام داد تا چهار مقدار VAD برابر با 76، 77، 78، 79 گردند؟

۱۱-۲۶ برای واریسی زمانی که منبعی به کامپیوتر در حال سرویس‌دهی به وقفه قبلی بوسیله همان منبع، وقفه می‌دهد، چه برنامه‌ای لازم است.

۱۱-۲۷ چرا خطوط کنترل خواندن و نوشتن در کنترل کننده DMA دو طرفه است؟ تحت چه شرایطی و

به چه منظوری از آنها بعنوان ورودی استفاده می شود؟ تحت چه شرایطی و برای چه منظوری از آنها بعنوان خروجی استفاده می شود.

۱۱-۲۸ می خواهیم 256 کلمه را از یک دیسک مغناطیسی به بخشی از حافظه که از آدرس 1230 شروع می شود انتقال دهیم. انتقال با استفاده از DMA طبق شکل ۱۸-۱۱ صورت می گیرد.

۱۱-۲۹ یک کنترل کننده DMA کلمه های 16 بیتی را با استفاده از سرعت سیکل به حافظه منتقل می کند. این کلمه ها از وسیله ای دریافت می شود که کاراکترها را با سرعت 2400 کاراکتر در ثانیه ارسال می نماید. CPU دستورالعمل ها را با سرعت یک میلیون دستورالعمل در ثانیه برداشت و اجرا می کند. CPU در اثر انتقال به شیوه DMA تا چه حد کند می شود؟

۱۱-۳۰ چرا DMA بهنگام تقاضای انتقال به حافظه نسبت به CPU بهنگام تقاضای انتقال به حافظه اولویت دارد.

۱۱-۳۱ فلوجارتی مشابه ۱۱-۲۰ برای IBM 370 رسم کنید و ارتباط CPU با کانال انتقال I/O را توضیح دهید.

۱۱-۳۲ آدرس یک پایانه متصل به یک پردازنده انتقال داده از دو حرف الفبای انگلیسی یا یک حرف و بدنبال آن یکی از ده رقم تشکیل شده است. چند آدرس مختلف می توان ایجاد کرد.

۱۱-۳۳ رویه ای را برای خط انتقال و دنباله کاراکترها، به منظور ارتباط بین یک پردازنده انتقال داده و یک پایانه دوردست تنظیم کنید. پردازنده سوال می کند که پایانه در حال کار است یا خیر. پایانه با بله و خیر به آن پاسخ می دهد. اگر پاسخ بلی باشد پردازنده بلاکسی از متن برای آن ارسال می نماید.

۱۱-۳۴ یک ارتباط انتقال داده از قرارداد مبتنی بر کاراکتر با شفافیت داده ها با بکارگیری کاراکتر DLE استفاده می کند. پیام متنی که فرستنده بین STX و ETX می فرستد بصورت زیر است:

DLE STX DLE DLE ETX DLE DLE ETX DLE ETX

داده های شفاف متنی را بصورت دودویی بنویسید.

۱۱-۳۵ حداقل تعداد بیت هایی که یک کادر در قرارداد مبتنی بر بیت دارد چقدر است؟

۱۱-۳۶ نشان دهید که چگونه روش درج صفر در قرارداد مبتنی بر بیت، وقتی که یک صفر و بدنبال آن ده بیت معادل دودویی 1023 ارسال می شود عمل می کند.

۱۲

سازمان حافظه

- | | |
|------|------------------------|
| ۱۲-۱ | سلسله مراتب حافظه |
| ۱۲-۲ | حافظه اصلی |
| ۱۲-۳ | حافظه کمکی |
| ۱۲-۴ | حافظه تداعیگر |
| ۱۲-۵ | حافظه کش |
| ۱۲-۶ | حافظه مجازی |
| ۱۲-۷ | سخت افزار مدیریت حافظه |

۱۲-۱ سلسله مراتب حافظه

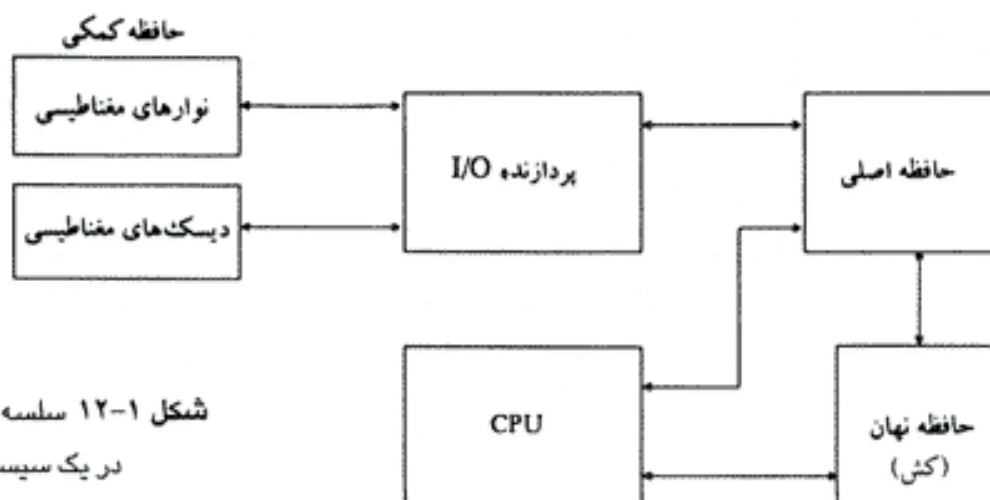
واحد حافظه در هر کامپیوتر دیجیتال یکی از بخش های اصلی است زیرا برای ذخیره برنامه ها و داده ها مورد نیاز می باشد. یک کامپیوتر بسیار کوچک با کاربرد محدود ممکن است بتواند کار معینی را بدون نیاز به افزایش ظرفیت ذخیره سازی انجام دهد. اما کامپیوترهای همه منظوره اگر علاوه بر ظرفیت حافظه اصلی، به امکانات ذخیره سازی اضافی مجهز باشند کارها را موثرتر اجرا خواهند نمود. در یک کامپیوتر معمولاً فضای واحد حافظه برای جای دادن همه برنامه های مورد استفاده کافی نیست. بعلاوه، اکثر کاربران کامپیوتر مقادیر قابل توجهی از نرم افزارهای پردازش داده را جمع آوری نموده و به این عمل همچنان ادامه می دهند. البته کلیه اطلاعات جمع آوری شده بطور همزمان مورد نیاز پردازنده نیست. بنابراین، مقرون به صرفه تر است از وسایل ذخیره سازی ارزان قیمت بعنوان پشتیبانی، اطلاعاتی که در حال حاضر مورد نیاز CPU نیستند، استفاده شود. واحد حافظه ای که مستقیماً با CPU ارتباط دارد حافظه اصلی^۱ نامیده می شود. وسایلی که نقش پشتیبانی ذخیره سازی را بعهده دارند حافظه کمکی^۲ خوانده می شوند. معمول ترین وسیله حافظه کمکی بکار رفته در سیستم های کامپیوتری دیسک ها و نوارهای مغناطیسی هستند. این وسایل برای ذخیره برنامه های سیستم، فایل های بزرگ داده، و سایر اطلاعات پشتیبانی مورد استفاده اند. در حافظه اصلی فقط برنامه ها و داده های مورد استفاده جاری پردازنده جای می گیرند. کلیه اطلاعات دیگر در حافظه کمکی ذخیره می شود و هنگام لزوم به حافظه اصلی منتقل می گردند.

1- Main Memory

2- Auxiliary Memory

ظرفیت کل یک کامپیوتر را می توان بصورت یک سیستم سلسله مراتبی از اجزاء تصور کرد. این سیستم حافظه سلسله مراتبی متشکل است از کلیه وسایل ذخیره سازی در یک سیستم کامپیوتری، از حافظه کمکی گنند ولی با ظرفیت بالا گرفته تا حافظه اصلی نسبتاً سریع تر، و حتی حافظه کوچکتر ولی سریع تر گنش (نهان) که بوسیله مدارهای منطقی پردازشگر سریع قابل دسترسی هستند. شکل ۱-۱۲ اجزاء سلسله مراتبی حافظه ای نوعی را نشان می دهد. در پایین سیستم سلسله مراتبی نوارهای مغناطیسی نسبتاً گنند قرار دارد که برای ذخیره فایل های جابجایی بکار می رود. پس از آن دیسک های مغناطیسی واقعند که بعنوان ذخیره سازی پشتیبانی بکار می روند. حافظه اصلی مکان میانی را دارد. و قادر است مستقیماً با CPU و وسایل کمکی از طریق پردازنده I/O ارتباط داشته باشد. وقتی برنامه هایی که در حافظه اصلی قرار ندارند بوسیله CPU مورد نیاز باشند، از حافظه کمکی بداخل آورده می شوند. برنامه هایی که در حال حاضر در حافظه اصلی قرار دارند ولی مورد نیاز نیستند به حافظه کمکی منتقل می گردند تا فضای کافی برای برنامه ها و داده های جاری بکار رفته ایجاد شود.

نوعی حافظه خاص بسیار سریع که گنش نامیده می شود گاهی اوقات برای افزایش سرعت پردازش با در اختیار گذاشتن برنامه ها و داده های جاری برای CPU با سرعتی بالا، بکار می رود. حافظه گنش در کامپیوترها برای جبران اختلاف سرعت بین دستیابی حافظه اصلی و مدار پردازنده مورد استفاده قرار می گیرد. مدار CPU معمولاً سریعتر از زمان دستیابی (سرعت) حافظه اصلی است، لذا سرعت پردازش را عمدتاً سرعت حافظه اصلی محدود می کند. تکنیکی که برای جبران عدم تطابق سرعت های عملیاتی بکار می رود استفاده از یک حافظه کش کوچک و فوق العاده سریع بین CPU و حافظه اصلی است که زمان دستیابی آن نزدیک به پیوند ساعت پردازنده می باشد. حافظه کش برای ذخیره قطعه هایی از برنامه ها که در CPU در دست اجرا هستند و داده های موقتی که در محاسبات جاری مکرراً بکار می روند مورد استفاده قرار می گیرد. با فراهم نمودن سریع برنامه ها و داده ها، امکان افزایش سرعت عملکرد وجود خواهد داشت. مادامی که پردازنده I/O انتقال داده را بین حافظه کمکی و حافظه اصلی اداره می کند، سازمان کش به



شکل ۱-۱۲ سلسله مراتب حافظه در یک سیستم کامپیوتری

انتقال اطلاعات بین حافظه اصلی و CPU می پردازد. بنابراین هر یک از آنها با سطح متفاوتی از سلسله مراتبی حافظه درگیر است. دلیل داشتن دو یا سه سطح در سلسله مراتب حافظه، مسائل اقتصادی است. با افزایش ظرفیت حافظه، ارزش ذخیره اطلاعات در هر بیت کاهش می یابد و زمان دستیابی حافظه طولانی تر می گردد. حافظه کمکی ظرفیت ذخیره سازی بالایی را داراست، نسبتاً ارزان است، ولی در مقایسه با حافظه اصلی سرعت دستیابی پایینی را دارد. حافظه کش خیلی کوچک است، نسبتاً گران است، و سرعت دستیابی بالایی دارد. بنابراین با افزایش دستیابی حافظه، ارزش نسبی آن اضافه می گردد. هدف نهایی از بکارگیری سلسله مراتب حافظه ای، فراهم آوردن بالاترین سرعت دستیابی متوسط ممکن با حداقل ارزش کل تمام حافظه سیستم است.

حافظه های کمکی و کش برای مقاصد مختلفی بکار می روند. کش آن بخش از برنامه و داده هایی را که بیشتر مورد استفاده اند ذخیره می کند، در حالی که حافظه کمکی بخش هایی را که در حال حاضر مورد استفاده CPU نیست نگهدارند. بعلاوه CPU بر روی حافظه کش و حافظه اصلی دسترسی مستقیم دارد در صورتی که بر روی حافظه کمکی این چنین نیست. انتقال از حافظه کمکی به حافظه اصلی معمولاً بصورت دسترسی مستقیم به بلاک های داده است. نسبت زمان دستیابی بین کش و حافظه اصلی 1 به 7 است. مثلاً یک حافظه کش نوعی ممکن است زمان دستیابی 100ns را داشته باشد، در حالیکه زمان دستیابی حافظه اصلی 700ns است. زمان دستیابی متوسط حافظه کمکی معمولاً 1000 برابر حافظه اصلی است. سایز بلاک در حافظه کمکی بین 256 تا 2048 کلمه است، در حالی که سایز بلاک در حافظه کش از 1 تا 16 کلمه می باشد. بسیاری از سیستم های عامل بصورتی طراحی می شوند تا CPU بتواند بطور همزمان تعدادی از برنامه های مستقل را اجرا نماید. این مفهوم که چندبرنامگی¹ خوانده می شود، به وجود دو یا چند برنامه در بخش های متفاوت سلسله مراتب حافظه در یک زمان اشاره دارد. بدین ترتیب می توان تمام بخش های کامپیوتر را با کار بر روی چندین برنامه بطور سری، مشغول نگهداشت. مثلاً فرض کنید برنامه ای در CPU در دست اجراست و یک انتقال I/O لازم شود. CPU، پردازنده I/O را برای شروع اجرای انتقال مقداردهی اولیه می نماید. این عمل به CPU اجازه می دهد تا برنامه دیگری را اجرا کند. در یک سیستم چندبرنامگی، وقتی یک برنامه برای انتقال ورودی - خروجی منتظر می ماند، برنامه دیگری آماده استفاده از CPU است. در چندبرنامگی، نیاز به اجرای برنامه ها بصورت قطعه به قطعه، تغییر اندازه حافظه اصلی در حال استفاده توسط یک برنامه مفروض، و جابجایی برنامه ها به بخش های مختلف سلسله مراتب حافظه وجود دارد. برنامه های کامپیوتر گاهی طولانی تر از آن هستند که در کل فضای موجود در حافظه اصلی جا بگیرند. بعلاوه یک سیستم کامپیوتر چندین برنامه را بکار می برد و همه برنامه ها نمی توانند در یک حافظه اصلی در تمام مدت قرار بگیرند. معمولاً یک برنامه با داده های مربوطه اش در حافظه کمکی قرار می گیرند. وقتی که برنامه یا بخشی از برنامه قرار است اجرا شود، ابتدا به حافظه اصلی انتقال می یابد تا بوسیله CPU اجرا شود. بنابراین هر کس می تواند تصور کند که حافظه کمکی حاوی همه اطلاعاتی است که در یک

1- Multiprogramming

سیستم ذخیره شده است. کار سیستم عامل این است که بخشی از این اطلاعات را که در حال حاضر فعال است در حافظه اصلی نگهدارد. بخشی از سیستم کامپیوتر که بر جریان اطلاعات، بین حافظه کمکی و حافظه اصلی نظارت می‌کند سیستم مدیریت حافظه^۱ نامیده می‌شود. سخت‌افزار یک سیستم مدیریت حافظه در بخش ۷-۱۲ ارائه شده است.

۱۲-۲ حافظه اصلی

حافظه اصلی واحد ذخیره‌سازی مرکزی در یک سیستم کامپیوتری است. این واحد، حافظه نسبتاً بزرگ و سریعی است که برای ذخیره برنامه‌ها و داده‌ها در حین عملکرد کامپیوتر مورد استفاده قرار می‌گیرد. تکنولوژی اصلی بکاررفته برای حافظه اصلی بر مدارهای مجتمع نیمه هادی مبتنی است. تراشه‌های RAM مدار مجتمع با دو شیوه عملکرد مختلف استاتیک^۲ و دینامیک^۳ در دسترس قرار دارند. RAM استاتیک اساساً از فلیپ‌فلاپها ساخته شده که اطلاعات دودویی را ذخیره می‌کنند. اطلاعات ذخیره شده مادامی که توان الکتریکی به واحد اعمال شود معتبر است. RAM دینامیکی اطلاعات دودویی را بصورت بارهای الکتریکی که به خازن‌ها اعمال می‌شوند، ذخیره می‌نماید. خازن‌ها در داخل تراشه با ترانزیستورهای MOS ایجاد می‌شوند. بارهای ذخیره شده در خازن‌ها به مرور تخلیه می‌شوند و لذا حافظه دینامیکی باید بصورت دوره‌ای بارگذاری شوند. تجدید بارگذاری، یا تازه‌سازی، با چرخش در بین کلمات در هر چند میلی‌ثانیه یکبار و احیای بار در حال تخلیه انجام می‌شود. در RAM دینامیکی مصرف توان الکتریکی کمتر و ظرفیت ذخیره‌سازی در یک حافظه بیشتر است. RAM استاتیکی از نظر کاربرد ساده‌تر بوده و سیکل خواندن و نوشتن در آن کوتاهتر است.

بیشتر حافظه اصلی در یک کامپیوتر همه منظوره از تراشه‌های مدار مجتمع RAM تشکیل می‌شود، اما بخشی از این حافظه ممکن است از تراشه ROM ساخته شود. در ابتدا RAM برای ارجاع به حافظه با دستیابی تصادفی^۴ بکار رفت، اما اکنون از آن برای مشخص کردن حافظه خواندنی و نوشتنی^۵ در مقابل حافظه فقط خواندنی^۶ استفاده می‌شود، هرچند که ROM هم نوعی حافظه با دستیابی تصادفی است. RAM برای ذخیره کردن برنامه‌ها و داده‌هایی که قرار است تغییر یابند بکار می‌رود ولی ROM برای ذخیره برنامه‌هایی را که بطور دائم در کامپیوتر جای داده می‌شوند و یا محتوای جداول ثابتی که پس از تکمیل ساخت کامپیوتر تغییر نمی‌کنند مورد استفاده قرار می‌گیرند.

در میان سایر امکانات در کامپیوتر، بخش ROM حافظه اصلی برای ذخیره برنامه آغازینی بنام بارکننده راه‌انداز^۷ بکار می‌رود. بارکننده راه‌انداز برنامه‌ای است که کار آن آغاز کار نرم‌افزار کامپیوتر پس از روشن کردن دستگاه است. چون RAM فرار است، وقتی که برق دستگاه قطع می‌شود محتوای آن از بین

1- Memory Management System

2- Static

3- Dynamic

4- Random Access Memory

5- Read/Write Memory

6- Read Only Memory

7- Bootstrap Loader

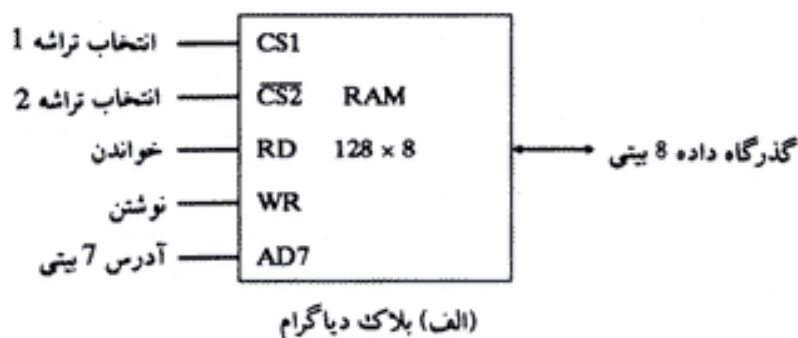
می رود. محتوای ROM پس از قطع و وصل توان الکتریکی تغییری نمی کند. آغاز بکار کامپیوتر متشکل است از روشن شدن دستگاه و شروع اجرای یک برنامه اولیه. بنابراین وقتی که دستگاه روشن شد، سخت افزار کامپیوتر شمارنده برنامه را با اولین آدرس بارکننده راه انداز بار می کند. برنامه راه انداز بخشی از سیستم عامل را از دیسک به حافظه اصلی بار می کند و سپس کنترل به سیستم عامل انتقال می یابد، که کامپیوتر را برای استفاده عمومی آماده می نماید.

تراشه های RAM و ROM در اندازه های مختلف موجودند. اگر حافظه مورد نیاز برای کامپیوتر بزرگتر از ظرفیت یک تراشه باشد، لازم است تعدادی تراشه با هم ترکیب شوند تا اندازه حافظه لازم حاصل شود. برای نشان دادن اتصالات بین تراشه ها، ساخت یک تراشه 8×1024 را با تراشه های RAM با اندازه 8×128 و تراشه های ROM با اندازه 8×512 نشان می دهیم.

تراشه های RAM و ROM

یک تراشه RAM اگر یک یا چند ورودی کنترل داشته باشد تا فقط در صورت نیاز این تراشه را انتخاب کنند به نحو مناسبتری قادر به ارتباط با CPU است. ویژه گی متداول دیگر، گذرگاه داده دو طرفه ای است که امکان انتقال داده از حافظه به CPU را در عمل خواندن، یا از CPU به حافظه را در حین عمل نوشتن فراهم می سازد. گذرگاه دو طرفه را می توان با بافرهای سه حالته ساخت. خروجی یک بافر سه حالته می تواند در یکی از سه حالت مختلف قرار گیرد: سیگنال معادل با 1 منطقی، سیگنال معادل با 0 منطقی، یا یک حالت امپدانس بالا. 1 و 0 منطقی سیگنال های معمولی منطقی هستند. حالت امپدانس بالا مانند یک مدار باز عمل می کند، و بدان معنی است که خروجی سیگنالی حمل نمی کند و معنای منطقی ندارد.

بلاک دیاگرام یک تراشه RAM در شکل ۲-۱۲ نشان داده شده است. ظرفیت حافظه 128 کلمه هشت بیتی است (یک بایت). این حافظه دارای 7 خط آدرس و یک گذرگاه داده هشت بیتی دو طرفه است. ورودی های خواندن و نوشتن عمل مورد نظر را مشخص می کنند و دو ورودی کنترل انتخاب تراشه (CS) برای فعال کردن تراشه بهنگام انتخاب توسط ریزپردازنده، بکار می روند. وجود بیش از یک ورودی کنترل برای انتخاب تراشه، عمل دیکد شدن آدرس را هنگامی که چندین تراشه در ریزپردازنده وجود داشته باشد تسهیل می نماید. ورودی های خواندن و نوشتن گاهی بصورت یک خط R/W در می آیند. هر وقت تراشه انتخاب شود، دو حالت دودویی این خط دو عمل خواندن یا نوشتن را مشخص می کند. جدول کار لیست شده در شکل ۲-۱۲ (ب) طرز کار تراشه RAM را مشخص می نماید. واحد حافظه فقط هنگامی که $CS1=1$ و $CS2=0$ باشد فعال می شود. خط بالای متغیر انتخاب دوم نشان می دهد که این ورودی وقتی که برابر 0 باشد فعال است. اگر ورودی های انتخاب تراشه فعال نباشند، و یا اگر فعال باشند ولی ورودی های خواندن یا نوشتن فعال نباشند، حافظه مسدود و گذرگاه داده آن در حالت



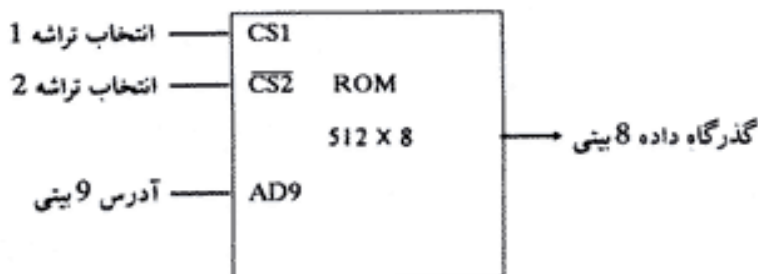
CS1	$\overline{CS2}$	RD	WR	عمل حافظه	وضعیت گذرگاه داده
0	0	x	x	مسدود	امپدانس بالا
0	1	x	x	مسدود	امپدانس بالا
1	0	0	0	مسدود	امپدانس بالا
1	0	0	1	نوشتن	ورودی داده به RAM
1	0	1	x	خواندن	خروجی داده به RAM
1	1	x	x	مسدود	امپدانس بالا

(ب) جدول تابع

شکل ۲-۱۲ تراشهای از نوع RAM

امپدانس بالا خواهد بود. اگر $CS1=1$ و $\overline{CS2}=0$ باشد، حافظه می تواند در شیوه عملکرد خواندن یا نوشتن قرار گیرد. اگر ورودی WR فعال شود، حافظه یک بایت را از گذرگاه داده در مکانی که خطوط آدرس تعیین می کند ذخیره می نماید. اگر ورودی RD فعال شود، محتوای بایت انتخاب شده روی گذرگاه داده قرار می گیرد. سیگنالهای RD و WR عملکرد حافظه و بافرهای گذرگاه مربوط به گذرگاه داده دو جهت را کنترل می نماید.

یک تراشه ROM هم سازمان خارجی مشابهی دارد. با این وجود، چون ROM تنها خوانده می شود، گذرگاه داده تنها در حالت خروجی قرار می گیرد. بلاک دیاگرام یک تراشه ROM در شکل ۳-۱۲ دیده می شود. در مقایسه دو تراشه RAM و ROM با ساینز برابر، ROM می تواند تعداد بیت های بیشتری نسبت به RAM داشته باشد، زیرا سلولهای دودویی ROM فضای کمتری را نسبت به



شکل ۳-۱۲ تراشهای از نوع ROM

RAM اشغال می‌کنند. به همین دلیل، دیباگرام ROM دارای 512 بایت و RAM دارای 128 بایت می‌باشد. نه خط آدرس تراشه ROM یکی از 512 بایت ذخیره شده در آن را مشخص می‌کند. دو ورودی انتخاب تراشه باید $CS1=1$ و $CS2=0$ باشند تا واحد حافظه کار کند. در غیراینصورت، گذرگاه داده در حالت امپدانس بالاست. در ROM به کنترل خواندن یا نوشتن نیازی نیست زیرا این مدار فقط می‌تواند خوانده شود. بنابراین وقتی تراشه بوسیله دو ورودی انتخاب فعال شود بایت انتخابی بوسیله خطوط آدرس روی گذرگاه داده ظاهر می‌شود.

نقشه آدرس‌های حافظه

طراح یک سیستم کامپیوتری باید مقدار حافظه لازم را برای کاربرد خاص محاسبه و به آن RAM یا ROM لازم را اختصاص دهد. سپس ارتباطات بین حافظه و پردازنده با دانستن اندازه حافظه و نوع تراشهای RAM یا ROM موجود تعیین می‌شود. آدرس دهی حافظه را می‌توان با استفاده از جدولی تنظیم کرد که آدرس حافظه اختصاص یافته به هر تراشه را مشخص می‌کند. این جدول که نقشه آدرس حافظه نامیده می‌شود، نمایش تصویری از فضای آدرس برای هر تراشه در سیستم است.

برای تشریح مطلب فوق با یک مثال فرض کنید که یک سیستم کامپیوتری به یک RAM با ظرفیت 512 بایت و یک ROM با ظرفیت 512 بایت نیاز دارد. تراشه های RAM و ROM بکار رفته در شکل ۱۲-۲ و ۱۲-۳ نشان داده شده‌اند. نقشه آدرس‌های حافظه برای این پیکربندی در جدول ۱-۱۲ آمده است. ستون قطعات نشان می‌دهد که آیا ROM بکار رفته است یا RAM. ستون آدرس شانزده شانزدهمی به هر تراشه محدوده‌ای از آدرس‌های معادل شانزده شانزدهمی را مشخص می‌کند. خطوط گذرگاه آدرس در ستون سوم ارائه شده است. هر چند گذرگاه آدرس 16 خط دارد، جدول فقط 10 خط را نشان می‌دهد زیرا 6 خط دیگر در این مثال بکار نرفته است و فرض می‌شود صفر باشند. X های کوچک در زیر ستون گذرگاه آدرس خطوطی را مشخص می‌کنند که باید به ورودی‌های آدرس هر تراشه وصل شوند. تراشه‌های RAM دارای 128 بایت بوده و به هفت خط آدرس نیاز دارند. تراشه ROM دارای 512 بایت بوده و به 9 خط آدرس نیاز دارد. X ها همیشه به خطوط پائین رتبه گذرگاه اختصاص می‌یابند: خطوط 1

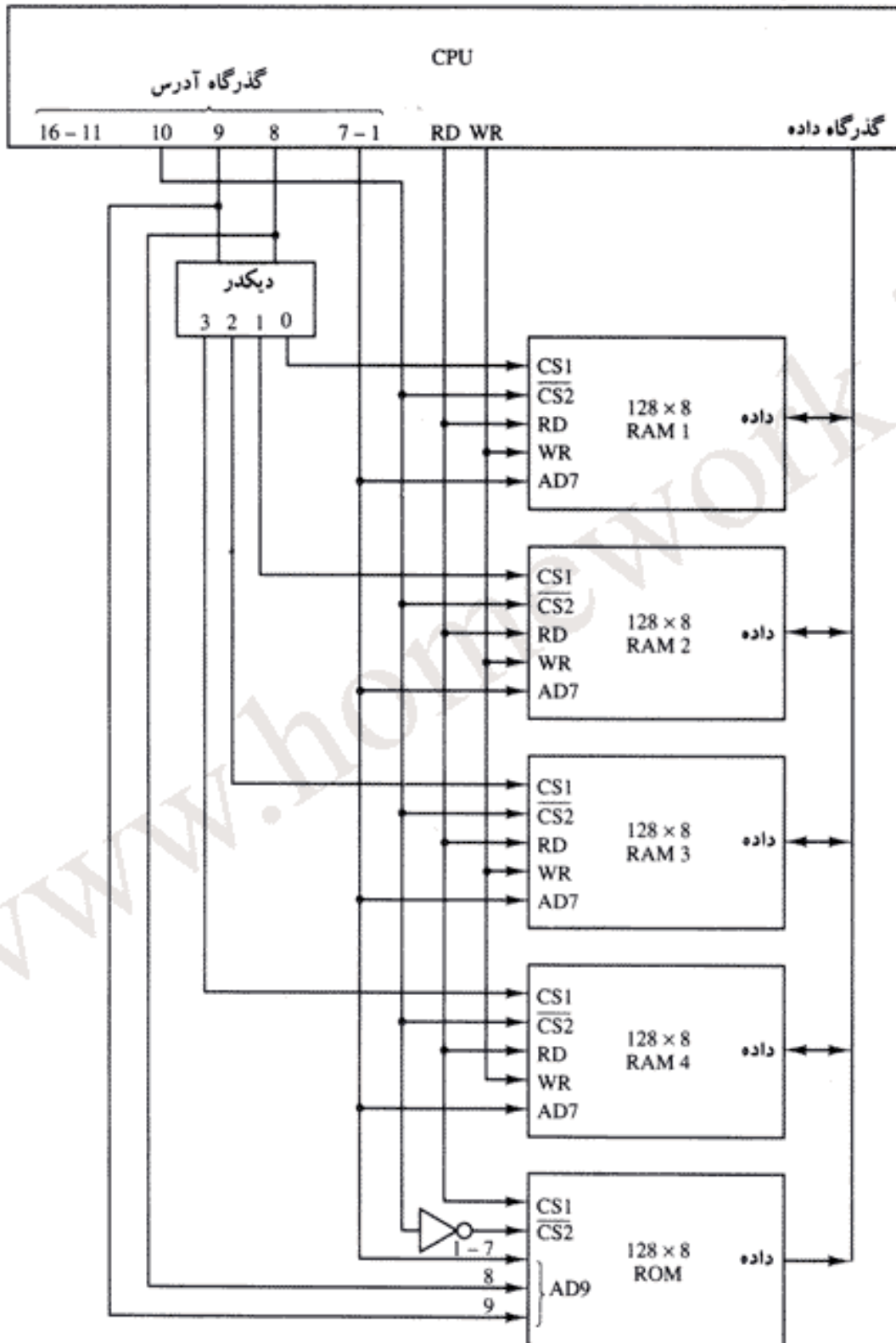
جدول ۱-۱۲ نقشه آدرس‌های حافظه برای ریزکامپیوتر

قطعه	آدرس شانزده شانزدهمی	گذرگاه آدرس									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000-007F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080-00FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100-017F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180-01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200-03FF	1	x	x	x	x	x	x	x	x	x

تا 7 به RAM و خطوط 1 الی 9 به ROM. حال لازم است تا چهار تراشه RAM را با تخصیص آدرس متفاوتی به هریک از هم متمایز کنیم. برای این مثال خاص، ما خطوط گذرگاه 8 و 9 را برای نمایش چهار ترکیب دودویی متمایز انتخاب می‌کنیم. دقت کنید هر جفت دیگری از خطوط گذرگاه که بکار نرفته باشند را می‌توان برای این منظور انتخاب کرد. جدول بخوبی نشان می‌دهد که 9 خط پائین رتبه آدرس فضای حافظه‌ای برابر با $2^9=512$ بایت را ایجاد می‌کند. تمایز بین آدرس‌های RAM و ROM به یک خط دیگر گذرگاه انجام می‌شود. در اینجا فقط خط 10 را برای این منظور انتخاب می‌کنیم. هنگامی که خط 10 صفر باشد، CPU یک تراشه RAM را انتخاب می‌کند. و اگر این خط 1 باشد، ROM انتخاب می‌شود. آدرس معادل شانزده شانزدهی تراشه از اطلاعات زیر ستون گذرگاه آدرس فراهم می‌شود. خطوط گذرگاه آدرس به گروه‌های چهار بیتی تقسیم شده است بطوری هر گروه با یک رقم شانزده شانزدهی نمایش داده می‌شود. اولین رقم شانزده شانزدهی خطوط 13 تا 16 را نشان می‌دهد و همواره 0 است. رقم شانزده شانزدهی بعدی خطوط 9 الی 12 را نشان می‌دهد ولی خطوط 11 و 12 همیشه صفرند. محدوده آدرس‌های شانزده شانزدهی برای هر قطعه از x های مربوط به آن تعیین می‌شود. این x ها یک عدد دودویی را از تماماً 0 تا تماماً 1 نمایش می‌دهند.

اتصالات حافظه به CPU

تراشه‌های RAM و ROM از طریق گذرگاه‌های داده و آدرس به CPU متصل می‌شوند. خطوط پائین رتبه در گذرگاه آدرس بایت درون تراشه را انتخاب می‌کنند و سایر خطوط در گذرگاه آدرس تراشه را از طریق ورودی‌های انتخاب برمی‌گزینند. اتصال تراشه‌های حافظه به CPU در شکل ۴-۱۲ نشان داده شده است. این پیکربندی 512 بایت را برای RAM و 512 بایت را هم برای ROM فراهم می‌آورد. پیکربندی مذکور با توجه به جدول ۱-۱۲ پیاده سازی شده است. هر RAM هفت بیت پائین رتبه از گذرگاه آدرس را برای انتخاب یکی از 128 بایت ممکن دریافت می‌کند. ROM بخصوصی که انتخاب می‌شود با توجه به خطوط 8 و 9 گذرگاه است. این عمل از طریق یک دیکدر 2×4 که خروجی‌های آن به ورودی‌های CS1 در هر تراشه RAM متصل است صورت می‌گیرد. بنابراین وقتی که خطوط آدرس 8 و 9 برابر 00 شوند، اولین تراشه RAM انتخاب می‌شود. وقتی که 01 باشد، دومین تراشه انتخاب می‌گردد، و الی آخر. خروجی‌های RD و WR از ریزپردازنده به ورودی‌های همه تراشه‌های RAM وصل می‌شوند. انتخاب بین RAM و ROM از طریق خط شماره 10 گذرگاه حاصل می‌گردد. RAM ها هنگامی انتخاب می‌شوند که بیت واقع بر روی این خط 0 باشد، و اگر مقدار بیت 1 شود ROM انتخاب خواهد شد. ورودی انتخاب تراشه دیگر در ROM به کنترل RD متصل است تا بهنگام خواندن موجب فعال شدن ROM گردد. خطوط گذرگاه آدرس 1 تا 9 بدون عبور از دیکدر مستقیماً به ورودی‌های آدرس ROM اعمال شده‌اند. این اتصالات آدرس‌ها 0 تا 511 را به RAM و 512 تا 1023 را به ROM اختصاص می‌دهند. گذرگاه داده ROM تنها امکان خروجی شدن را دارد، در صورتیکه گذرگاه داده متصل



شکل ۴-۱۲ اتصال حافظه به CPU

به RAM می تواند اطلاعات را در هر دو جهت هدایت کند. مثالی که ارائه شد پیچیدگی اتصالات بین تراشه های حافظه و CPU را نمایان می سازد. هر چه تعداد تراشه های متصله بیشتر شود، دیکدرهای بیشتری بین حافظه ها مورد نیاز است طراح باید نقشه حافظه ای را تهیه کرده و آدرس ها را به تراشه تخصیص دهد تا با استفاده از آن اتصالات لازم معین شد.

۳-۱۲ حافظه کمکی

متداولترین دستگاه حافظه کمکی بکار رفته در سیستم های کامپیوتری دیسک ها و نوارهای مغناطیسی هستند. وسایل دیگری هم که نه کراراً ولی بکار می روند عبارتند از درام های مغناطیسی^۱، حافظه حباب مغناطیسی^۲، و دیسک های نوری می باشند. برای درک فیزیکی مکانیزم دستگاه های حافظه کمکی باید اطلاعاتی در مورد سیستم های مغناطیسی، الکترونیکی و الکترومکانیکی داشت. هر چند خواص فیزیکی این وسایل ذخیره سازی می تواند بسیار پیچیده باشد، ولی خصوصیات منطقی آنها قابل تشخیص و مقایسه است. مشخصه های مهم هر وسیله شیوه دستیابی، زمان دستیابی، سرعت انتقال، ظرفیت و قیمت آن است.

زمان متوسط لازم برای رسیدن به یک مکان ذخیره سازی در حافظه و دریافت محتوای آن، زمان دستیابی^۳ (یا دسترسی) خواننده می شود. در وسایل الکترومکانیکی با قطعات متحرک مانند دیسک ها و نوارها، زمان دستیابی متشکل است از زمان جستجوی^۴ لازم برای قراردادن هد خواندن - نوشتن بر روی یک مکان و زمان انتقال^۵ لازم برای انتقال داده به و یا از وسیله. چون زمان جستجو معمولاً خیلی طولانی تر از زمان انتقال است، حافظه کمکی بصورت مجموعه ای از رکوردها یا بلاک ها سازمان یافته است. یک بلاک تعداد مشخصی از کاراکترها یا کلمات است. معمولاً خواندن یا نوشتن بر روی تمام رکورد انجام می شود. سرعت انتقال تعداد کلمات یا کاراکترهایی است که وسیله می تواند در هر ثانیه، پس از قرارگرفتن در آغاز رکورد، انتقال دهد.

درام ها و دیسک های مغناطیسی از نظر عملکرد کاملاً مشابهند. هر دو از سطوح چرخان با سرعت بالا که از ماده مغناطیسی ضبط کننده پوشش یافته اند تشکیل شده اند. سطح چرخان درام، سیلندری شکل و دیسک سطح گرد مسطحی است. سطح ضبط کننده با سرعت یکنواختی چرخیده و در زمان دستیابی هرگز راه اندازی یا متوقف نمی شود. بیت ها بصورت نقطه های مغناطیسی با عبور سطح از زیر یک مکانیزم ساکن بنام هد نوشتن^۶، بر روی آن ضبط می شوند. بیت های ذخیره شده با تغییر میدان حاصل از نقطه های ثبت شده در روی سطح که با عبور از کنار هد خواندن^۷ حاصل می شود آشکار می گردد. مقدار سطح موجود برای ضبط در یک دیسک بزرگتر از درامی است که از نظر ابعاد با آن یکسان

1- Magnetic Drum

2- Magnetic Bubble Memory

3- Access time

4- Seek time

5- Transfer time

6- Write Head

7- Read Head

باشد. بنابراین اطلاعات بیشتری را می توان بر روی یک دیسک هم اندازه با درام ذخیره کرده باین دلیل، دیسک ها در اکثر کامپیوترها جایگزین درام شده اند.

دیسک های مغناطیسی

یک دیسک مغناطیسی صفحه دواری است که از فلز یا پلاستیک پوشش یافته با مواد مغناطیسی ساخته شده است اغلب هر دو طرف دیسک بکار رفته و چندین دیسک ممکن است روی یک محور نصب شده و روی هر سطح هم هد خواندن - نوشتن وجود داشته باشد. تمام دیسک ها با هم و با سرعت زیادی می چرخند و به منظور دستیابی راه اندازی یا متوقف نمی شوند. بیت ها در سطح مغناطیس شده بصورت نقاطی بر روی دوار هم مرکزی بنام شیار^۱ ذخیره می شوند. شیارها معمولاً به بخش هایی به نام قطاع^۲ تقسیم می شوند. در بیشتر سیستم ها، حداقل اطلاعات که می تواند انتقال یابد یک قطاع است. تقسیم بندی یک دیسک به شیارها و قطاع ها در شکل ۵-۱۲ نشان داده شده است. برخی از دستگاه ها از یک هد خواندن و نوشتن برای هر سطح دیسک استفاده می کنند. در این دستگاه ها قبل از خواندن و نوشتن، بیت های آدرس شیارها برای حرکت دادن هد خواندن یا نوشتن بر روی مکان مورد نظر بکار می روند. در نوعی دیگر از سیستم های دیسک، هدهای خواندن - نوشتن جداگانه برای هر شیار در هر سطح تهیه شده است. سپس بیت های آدرس، شیار خاصی را بطور الکترونیکی از طریق یک دیکدر انتخاب می کنند. این نوع دستگاه ها گرانتر بوده و فقط بر روی سیستم های کامپیوتری بسیار بزرگ یافت می شوند.



شکل ۵-۱۲ دیسک مغناطیسی

1- Track

2- Sector

در دیسک‌ها از شیارهای زمانبندی دائمی برای همگام کردن بیت‌ها و تشخیص قطعات‌ها استفاده می‌شود. یک سیستم دیسک با بیت‌های آدرسی که شماره دیسک، سطح دیسک، شماره قطعه، و شیار موردنظر در قطعه مربوطه را مشخص می‌کنند آدرس دهی می‌شود. پس از قرار گرفتن هد خواندن - نوشتن در شیار موردنظر، سیستم باید با چرخش دیسک تا رسیدن قطعه مشخص شده به زیر هد خواندن و نوشتن صبر کند. پس از رسیدن به ابتدای قطعه، انتقال اطلاعات با سرعت صورت می‌گیرد. دیسک‌ها ممکن است هدهای متعدد داشته باشند و بیت‌های چندین شیار بطور همزمان انتقال یابند.

در هر قطعه مفروض، شیاری که در نزدیکی لبه خارجی قرار دارد طولانی‌تر از شیاری است که در نزدیکی مرکز دیسک می‌باشد. اگر بیت‌ها با تراکم مساوی ذخیره شوند، برخی از شیارها دارای بیت‌های ضبط شده بیشتری خواهند بود. برای هم طول کردن همه شیارهای یک قطعه، برخی از دیسک‌ها از تراکم ضبط متغیر استفاده می‌کنند، بطوری که شیارهای نزدیکتر به مرکز نسبت به شیارهای نزدیک به لبه خارجی تراکم بیشتری دارند. باین ترتیب تعداد بیت‌های موجود بر روی همه شیارهای یک قطعه، معین یکسان خواهند بود.

دیسک‌هایی که بطور دائمی به مجموعه دستگاه متصل‌اند و آنها را هرکاربری نمی‌تواند از مجموعه خارج کند، دیسک سخت¹ می‌خوانند. دیسک خوانی که دیسک‌های قابل جابجایی داشته باشد دیسک نرم² خوانده می‌شود. دیسک‌هایی که در یک دیسک خوان نرم بکار می‌روند دیسک‌های قابل جابجایی کوچکی هستند که از پلاستیک ساخته شده و با ماده مغناطیسی ضبط‌کننده پوشش داده شده‌اند. دو سایز برای این دیسک‌ها متداول است، قطر 5.25 و قطر 3.5 اینچ. دیسک‌های 3.5 اینچی کوچکترند و می‌توانند داده‌های بیشتری را نسبت به دیسک‌های 5.25 اینچی ذخیره کنند. دیسک‌های نرم گسترده‌ای در کامپیوترهای شخصی بعنوان وسیله‌ای برای توزیع نرم افزار بین کاربران کامپیوتر بکار می‌روند.

نوار مغناطیسی

حامل نوار مغناطیسی متشکل است از اجزاء الکتریکی، مکانیکی و الکترونیکی که قطعات و مکانیزم کنترل را برای دستگاه نوار مغناطیسی فراهم می‌نمایند. خود نوار باریکه‌ای از پلاستیک پوشیده از ماده مغناطیسی ضبط‌کننده است. بیت‌ها بشکل نقاط مغناطیسی روی نوار در طول چند شیار ضبط می‌شوند. معمولاً هفت یا نه رکورد بطور همزمان ضبط می‌شوند تا به همراه بیت توازن تشکیل یک رکورد را بدهند. هدهای خواندن و نوشتن هر کدام بر روی یک شیار قرار می‌گیرند بطوری که داده‌ها بصورت رشته‌ای از کاراکترها می‌توانند ضبط و یا خوانده شوند.

دستگاه‌های نوار مغناطیسی را می‌توان متوقف کرد، بسمت جلو یا عقب راه اندازی کرد، یا آن را به عقب برگرداند. با وجود این نمی‌توان آنها را از بین دو کاراکتر شروع و یا بین آنها متوقف کرد. بهمین دلیل، اطلاعات بصورت بلاک‌هایی که به آنها رکورد می‌گویند ضبط می‌شود. فواصلی از نوار ضبط

1- Hard Disk

2- Floppy Disk

نشده بین رکوردها گنجانیده می شود تا در آنجا بتوان نوار را متوقف نمود. نوار هنگامی که روی یکی از فواصل قرار دارد شروع به حرکت می کند و تا زمانی که به رکورد بعدی برسد سرعت ثابت خود را بدست می آورد. هر رکورد واقع در روی نوار یک الگوی شناسایی بیتی در ابتدا و انتهای خود دارند. با خواندن الگوی بیتی واقع در ابتدای رکورد، کنترل نوار شماره رکورد را تشخیص می دهد. با خواندن الگوی بیتی واقع در انتها، واحد کنترل شروع یک فاصله را درک می کند. دستگاه نوار با مشخص کردن شماره رکورد و تعداد کاراکترهای رکورد آدرس دهی می شود. رکوردها ممکن است دارای طول ثابت یا متغیر باشند.

۴-۱۲ حافظه تداعیگر

بسیاری از کاربردهای داده پردازی به جستجو اقلامی که در یک جدول حافظه ذخیره شده اند نیاز است. یک برنامه اسمبلر جدول آدرس های سمبل را برای استخراج معادل دودویی جستجو می کند. ممکن است برای تعیین نام وضعیت حساب یکی از مشتریان بانک، شماره حساب وی در فایل جستجو شود، روش متداول در جستجوی جدول، ذخیره اقلام آن در جایی است که بتواند بصورت متوالی آدرس دهی شود. رویه جستجو سیاستی است برای انتخاب رشته ای از آدرسها، خواندن محتوای حافظه در هر آدرس، و مقایسه اطلاعات خوانده شده با داده تحت جستجو است تا اینکه انطباق رخ دهد. تعداد دفعات مراجعه با دسترسی به حافظه به مکان داده و کارایی الگوریتم بستگی دارد. برای به حداقل رساندن تعداد مراجعات برای یافتن یک داده، الگوریتم های جستجوی متعددی ابداع شده است. اگر بتوان داده های ذخیره شده در حافظه را به منظور دستیابی به آنها براساس محتوای خود داده و نه براساس یک آدرس شناسایی کرد، زمان لازم برای یافتن یک قلم داده تا حد قابل توجهی کاهش می یابد. واحد حافظه ای که دستیابی به آن براساس محتوا انجام می شود، حافظه تداعیگر^۱ یا حافظه قابل آدرس دهی با محتوا (CAM) خوانده می شود دستیابی به این نوع حافظه بطور همزمان و موازی و براساس محتوای داده، بجای آدرس یا مکان خاص، انجام می شود. حافظه می تواند یک مکان خالی بدون استفاده را برای ذخیره کلمه مورد نظر بیابد. وقتی که قرار باشد کلمه ای از حافظه تداعیگر خوانده شود، محتوای کلمه، یا بخشی از کلمه مشخص می شود. حافظه تمام کلماتی را که برای محتوای مشخص شده منطبق شوند برای خواندن علامت گذاری می کند.

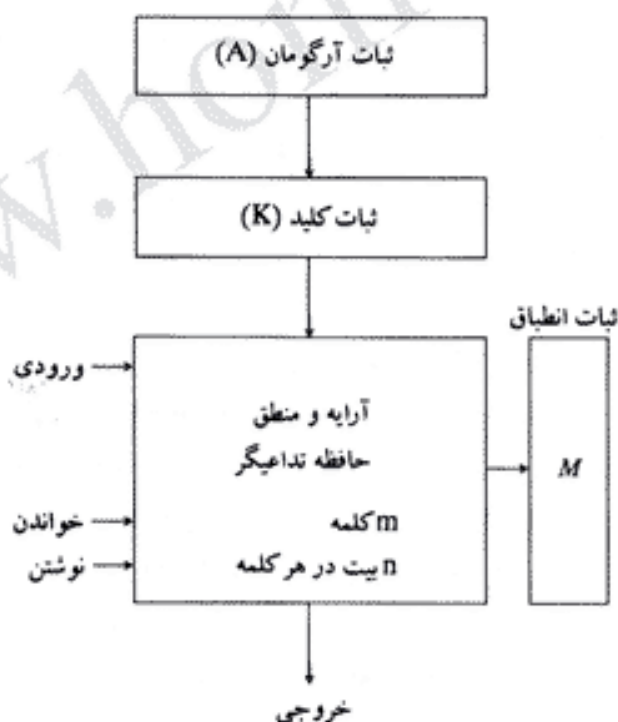
حافظه تداعیگر بعلت سازمان داخلی اش خصوصاً برای جستجوهای موازی براساس داده ها مناسب است. بعلاوه جستجو می تواند بر روی کل کلمه و یا روی میدان خاصی در یک کلمه انجام شود. حافظه تداعیگر گرانتر از حافظه با دستیابی تصادفی است زیرا هر سلول آن علاوه بر قابلیت ذخیره سازی باید دارای مدارهای منطقی برای انطباق محتوای آن با یک آرگومان خارجی نیز باشد. به همین دلیل، حافظه های تداعیگر در کاربردهایی که در آنها زمان جستجو بسیار اهمیت دارد و باید کوتاه باشد استفاده می شود.

1- Associative Memory

سازمان سخت افزاری

بلاک دیاگرام یک حافظه تداعیگر در شکل ۶-۱۲ داده شده است. این حافظه متشکل از یک آرایه حافظه و منطق لازم برای m کلمه n بیتی در هر کلمه است. ثبات آرگومان 1 و ثبات کلید k هر یک n بیت دارند، یعنی یک بیت به ازای هر یک از بیت‌های کلمات. ثبات انطباق M ، m بیتی است، یعنی یک بیت بازاء هر بیت از کلمه حافظه. هر کلمه در حافظه بطور موازی با محتوای ثبات آرگومان مقایسه می‌شود. کلماتی که با بیت‌های ثبات آرگومان تطبیق داشته باشند بیت متناظری را در ثبات انطباق 1 می‌کنند. پس از روند تطبیق، بیت‌هایی در ثبات انطباق که 1 شده‌اند نشان می‌دهند که کلمات متناظر آنها با کلمه مورد جستجو مطابقت دارند. عمل خواندن با مراجعه متوالی به کلماتی از حافظه که بیت‌های متناظر آنها در ثبات انطباق 1 شده‌اند انجام می‌شود.

ثبات کلید پوششی را برای انتخاب یک میدان خاص یا کلید آرگومان فراهم می‌آورد. اگر ثبات کلید تماماً 1 باشد، کل آرگومان با هر یک از کلمات حافظه مقایسه می‌شود. در غیراینصورت، فقط بیت‌هایی در آرگومان که محل متناظر آنها در ثبات کلید مقدار 1 دارد مقایسه می‌شوند. بنابراین ثبات کلید پوشش و یا قطعه اطلاعات شناسایی را برای ارجاع به حافظه فراهم می‌کند. برای تشریح مطلب توسط یک مثال عددی، فرض کنید که ثبات آرگومان A و ثبات کلید K دارای آرایش بیتی زیر، باشند. تنها سه بیت



شکل ۶-۱۲ بلاک دیاگرام حافظه تداعیگر

1- Argument Register

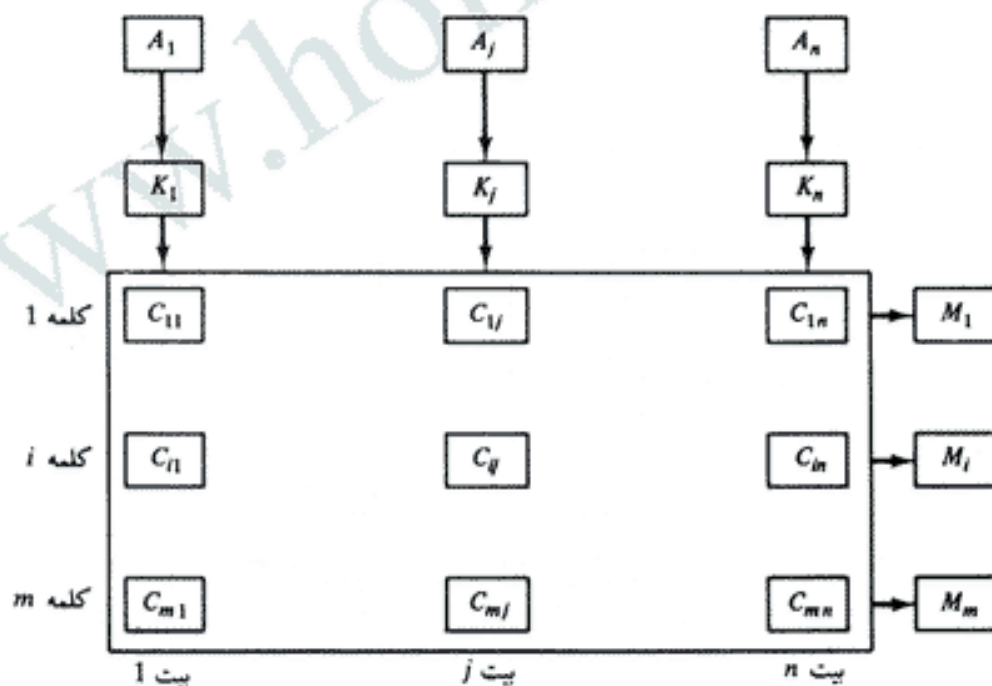
2- Switch Register

سمت چپ A با کلمات مقایسه زیرا K در این مکانها دارای 1 است.

A	101	111100	
K	111	000000	
کلمه 1	100	111100	انطباق ندارد
کلمه 2	101	000001	انطباق دارد

کلمه 2 با میدان آرگومان پوشش نیافته تطبیق دارد زیرا سه بیت سمت چپ آرگومان و کلمه برابرند. رابطه بین آرایه حافظه و ثباتهای خارجی در حافظه تداعیگر در شکل ۷-۱۲ نشان داده شده است. سلولهای آرایه با حرف C و دو زیر نویس مشخص شدهاند. اولین زیر نویس شماره کلمه و دومین زیر نویس مکان بیت را در کلمه مشخص می کند. باین ترتیب سلول C_{ij} مربوط به بیت j در کلمه i است. بیت A_j در ثبات آرگومان با همه بیتهای ستون j از آرایه مقایسه می شود بشرطی که $K_j=1$ باشد. این عمل برای همه ستونهای $n, \dots, 2, 1$ انجام می شود.

اگر بین همه بیتهای پوشش نیافته آرگومان و بیتهای کلمه، تطابق وجود داشته باشد، بیت متناظر در ثبات انطباق یعنی M_i برابر با 1 می شود. اگر یک یا چند بیت پوشش نیافته و کلمه مطابقت نداشته باشند M_i برابر با 0 خواهد بود.



شکل ۷-۱۲ حافظه تداعیگر به m کلمه و n بیت بازاء هر کلمه

1- Mask

سازمان درونی یک سلول نوعی C_{ij} در شکل ۸-۱۲ نشان داده شده است. این سلول از یک المان ذخیره سازی فلیپ فلاپ F_{ij} و مدارات لازم برای خواندن و نوشتن و انطباق سلولها تشکیل شده است. بیت ورودی در عمل نوشتن به سلول ذخیره سازی انتقال می یابد. در حین عمل خواندن بیت ذخیره شده خوانده می شود. مدار منطقی محتوای سلول ذخیره سازی را با بیت متناظر پوشش نیافته آرگومان مقایسه کرده و یک خروجی را در اختیار مدار تصمیم گیری قرار می دهد تا بیت M_i را 1 کند.

مدار انطباق

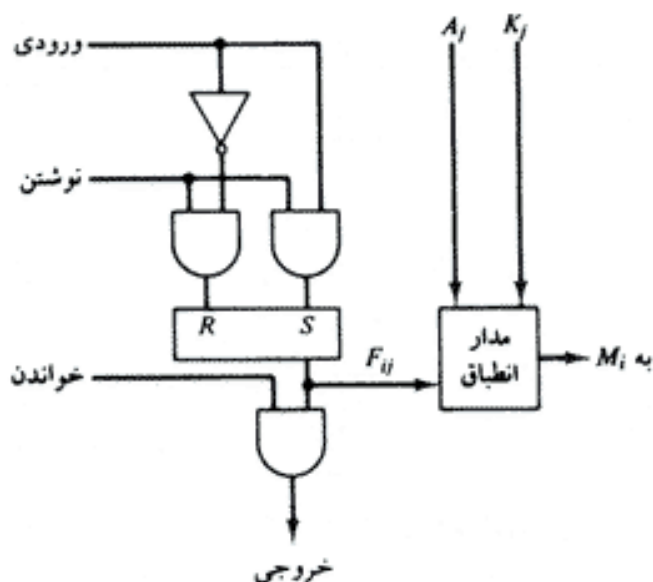
مدار منطقی برای هر کلمه را می توان از الگوریتم مقایسه دو عدد دودویی بدست آورد. ابتدا بیت های کلید را نادیده می گیریم و آرگومان A را با بیت های ذخیره شده در سلول های کلمه مقایسه می کنیم. کلمه i برابر با آرگومان در A است بشرطی که $A_j = F_{ij}$ با $j=1,2,\dots,n$ باشد. اگر هر دو بیت مساوی 1، یا 0 باشند، تساوی دو بیت بصورت تابع بول با عبارت زیر بیان می شود.

$$x_j = A_j F_{ij} + A'_j F'_{ij}$$

که در آن $x_j=1$ است اگر هر دو جفت بیت در مکان j مساوی باشد، در غیر این صورت $x_j=0$ خواهد بود. برای اینکه کلمه i برابر آرگومان A شود، تمام متغیرهای x_j باید برابر 1 باشند. این وضعیت شرط 1 شدن بیت انطباق متناظر یعنی M_i است، تابع بول برای این شرط عبارت است از

$$M_i = x_1 x_2 x_3 \dots x_n$$

که بمعنی انجام عمل AND بر روی همه جفت بیت های انطباق یافته در یک کلمه است.



شکل ۸-۱۲ یک سلول از حافظه تداعیگر

اکنون بیت کلید K_j را در مدار مقایسه می‌کنیم. اگر $K_j=0$ باشد، بیت های متناظر در A_j و F_j نیازی به مقایسه ندارند. فقط اگر $K_j=1$ باشد آنها باید مقایسه شوند. این شرط با OR کردن هر جمله با K'_j حاصل می‌شود، بنابراین:

$$x_j + K'_j = \begin{cases} x_j & \text{اگر } K_j = 1 \\ 1 & \text{اگر } K_j = 0 \end{cases}$$

هر گاه $K_j=1$ باشد داریم $k'_j=0$ و $x_j+0=x_j$ هر وقت هم $K=0$ باشد، آنگاه $K'_j=1$ و $x_j+1=1$ می‌گردد. جمله $(x_j + K'_j)$ در وضعیت 1 خواهد بود اگر جفت بیت های آن مقایسه نشوند. این مطلب الزامی است زیرا هر جمله با همه جملات دیگر AND می‌شود و خروجی 1 اثری روی حاصل نخواهد داشت. فقط هنگامی که $K_j=1$ باشد مقایسه بیت‌ها تأثیر خواهد داشت. اکنون می‌توان مدار انطباق برای کلمه i در یک حافظه تداعیگر را با تابع بولی زیر بیان کرد:

$$M_i = (x_1 + K'_1)(x_2 + K'_2)(x_3 + K'_3) \cdots (x_n + K'_n)$$

هر جمله در عبارت بشرطی 1 است که K_j متناظر آن 0 باشد. اگر $K_j=1$ باشد، جمله بسته به مقدار x_j برابر با 0 یا 1 خواهد بود. اگر همه جملات 1 باشد انطباق صورت گرفته و M_i برابر 1 خواهد بود. اگر تعریف اولیه x_j را بجای آن قرار دهیم، تابع بول فوق بصورت زیر بیان می‌شود:

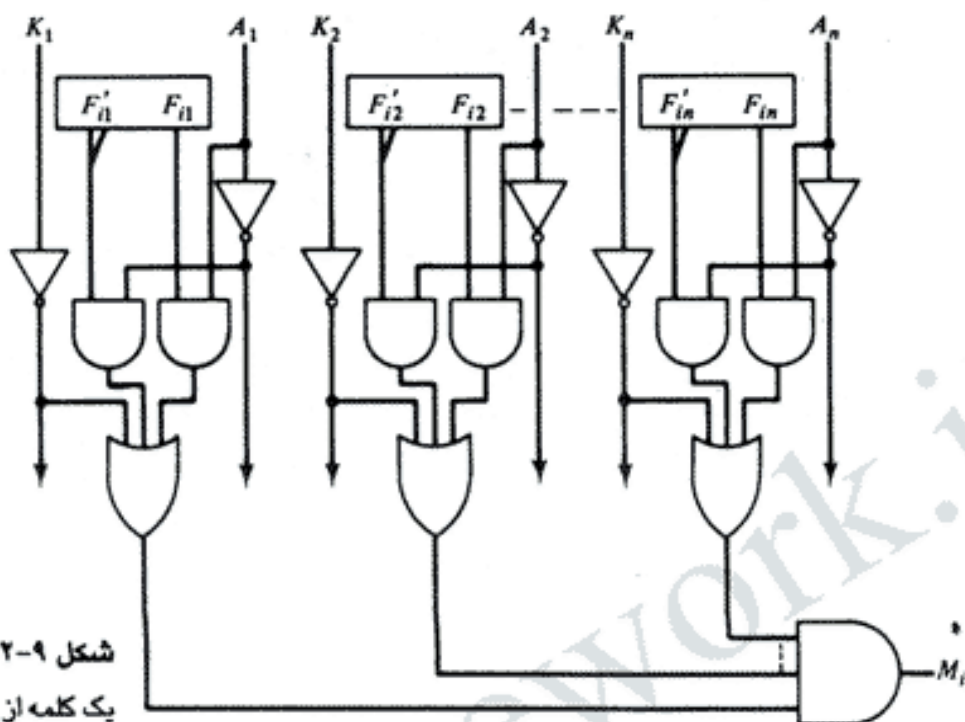
$$M_i = \prod_{j=1}^n (A_j F_j + A'_j F'_j + K'_j)$$

که در آن Π سمبل ضرب و مشخص کننده AND تمام n جمله است. ما به m تابع این چنینی نیاز داریم که هر یک مربوط به یک کلمه m و ... و 3 و 2 و 1 است.

مدار انطباق برای هر کلمه در شکل ۹-۱۲ دیده می‌شود. هر سلول دو گیت AND و یک گیت OR نیاز دارد. معکوس کننده‌های A_j و K_j یک بار برای هر ستون لازم‌اند و برای همه بیت‌های ستون بکار می‌روند. خروجی تمام گیت‌های OR در سلول‌های یک کلمه به ورودی یک گیت AND مشترک می‌روند سیگنال انطباق M_i را ایجاد کنند. اگر انطباقی رخ دهد M_i برابر 1 منطقی خواهد بود و اگر انطباقی رخ ندهد 0 است. توجه کنید که اگر ثبات کلید کلاً 0 باشد، خروجی M_i صرف نظر از مقدار A یا کلمه 1 خواهد بود. در طول عملکرد عادی باید از چنین وضعی اجتناب شود.

عمل خواندن

اگر بیش از یک کلمه با میدان آرگومان پوشش نیافته تطبیق نماید، بیت های متناظر با همه کلمات در ثبات انطباق مقدار 1 خواهند داشت. پس از آن لازم است تا بیت های ثبات انطباق یک به یک مرور



شکل ۹-۱۲ مدار انطباق برای یک کلمه از حافظه تداعیگر

شوند. کلمات انطباق یافته با اعمال سیگنال خواندن به ورودی خواندن هر کلمه‌ای که بیت M_i متناظر آن 1 است بطور متوالی خوانده می‌شود.

در اغلب کاربردها، حافظه تداعیگر جدولی که در آن هیچ دو داده یکسانی تحت یک کلید ندارد را ذخیره می‌نماید. در این حالت، فقط یک کلمه ممکن است با میدان آرگومان پوشش نیافته مطابقت داشته باشد. با اتصال مستقیم M_i به خط خواندن که در موقعیت همان کلمه (بجای ثبات M)، محتوای کلمه منطبق بطور اتوماتیک در خروجی ظاهر می‌شود و هیچ سیگنال فرمان خواندن بخصوصی لازم نیست. بعلاوه، اگر ما کلماتی را که محتوای صفر دارند مستثنی کنیم، یک خروجی تمام صفر بیانگر عدم تطبیق بوده و داده مورد جستجو در حافظه موجود نیست.

عمل نوشتن

یک حافظه تداعیگر باید قابلیت ذخیره سازی اطلاعات مورد جستجو را داشته باشد، نوشتن در یک حافظه تداعیگر می‌تواند، بسته به کاربرد اشکال مختلفی داشته باشد. اگر کل حافظه از قبل با اطلاعات جدیدی بار شده باشد، سپس نوشتن با آدرس دهی متوالی هر مکان میسر است. به این ترتیب مدار مورد استفاده به یک حافظه با دستیابی اتفاقی (RAM) برای نوشتن و به یک حافظه قابل آدرس دهی با محتوا برای خواندن تبدیل خواهد شد. در اینجا مزیت این است که آدرس برای ورودی مانند یک حافظه RAM دیکد می‌شود. در نتیجه بعوض داشتن m خط آدرس یعنی یک خط برای هر کلمه در حافظه.

تعداد خطوط آدرس بوسیله دیکدر به d خط کاهش می یابد که در آن $m=2^d$ است. اگر که قرار باشد کلمه ناخواسته ای پاک شود و کلمات جدیدی یک به یک وارد شوند، به ثبات خاصی برای تمایز بین کلمات فعال و غیر فعال نیاز است. تعداد بیت های این ثبات که گاهی اوقات ثبات نشانه نامیده می شود، به تعداد کلمات در حافظه خواهد بود. به ازاء هر کلمه فعال ذخیره شده در حافظه، بیت متناظری در ثبات نشانه 1 می شود. با 0 کردن بیت نشانه کلمه متناظر آن در حافظه پاک می شود. برای ذخیره کلمات در حافظه، ثبات نشانه مرور می شود تا اینکه اولین 0 دیده شود. باین ترتیب اولین کلمه غیر فعال و نیز محل برای نوشتن یک کلمه جدید مشخص می شود. پس از ذخیره کلمه جدید در حافظه، این کلمه با 1 کردن بیت نشانه فعال می شود. اگر قرار باشد یک مکان خالی با تماماً 0 نشان داده شود می توان پس از حذف کلمه ناخواسته از حافظه، آن را پاک کرد. بعلاوه کلماتی که دارای بیت نشانه 0 هستند باید با آرگومان کلمه پوشش یابند (علاوه بر پوششی که بابت های K انجام می شود) تا فقط کلمات فعال مقایسه شوند.

۵-۱۲ حافظه کش

تحلیل تعداد زیادی از برنامه نشان داده است که ارجاعات به حافظه در هر فاصله زمانی معینی غالباً محدود به چند ناحیه موضعی در حافظه است. این پدیده، خاصیت محلّیت ارجاع خواننده می شود. دلیل این خاصیت را می توان با بررسی این موضوع که یک برنامه کامپیوتر نوعاً بشکل مستقیم الخط پیش می رود و ضمن پیشروی با حلقه ها و زیر روال ها مکرراً برخورد می شود دریافت. وقتی که یک حلقه برنامه اجرا می شود، CPU متوالیاً به مجموعه ای از دستورات که حلقه را در حافظه تشکیل می دهند رجوع می نماید. هر بار یک زیر روال فراخوانی شود، مجموعه دستورات عمل های آن از حافظه برداشت می شود. بنابراین حلقه ها و زیر روال ها تمایل به محلی کردن ارجاع به حافظه برای برداشت دستورات عمل ها دارند. ارجاع داده به حافظه نیز، تا حدودی میل به محلی بودن دارد. رویه های جستجوی جدول مکرراً به بخشی از حافظه که جدول در آن است مراجعه می کنند. رویه های تکراری به مکان های حافظه مشترکی مراجعه می کنند و آرایه های اعداد به بخشی محلی از حافظه محدودند. نتیجه همه این ملاحظات خاصیت محلّیت ارجاع است، که می گوید در فواصل زمانی کوتاه، آدرس های تولید شده توسط یک برنامه مکرراً به چند ناحیه محلی از حافظه اشاره می کنند، در حالی که بقیه برنامه بدفعات کمتری مورد دستیابی قرار می گیرند.

اگر بخش های فعال برنامه و داده ها در یک حافظه کوچک و سریع جای داده شوند، زمان دستیابی متوسط می تواند کاهش یافته و باین ترتیب زمان کل اجرای برنامه تقلیل می یابد. چنین حافظه سریع و کوچک، حافظه کش^۱ خوانده می شود. این حافظه همان طور که در شکل ۱-۱۲ دیده شد بین CPU و حافظه اصلی قرار می گیرد. زمان دستیابی حافظه کش به نسبت ۵ تا ۱۰ برابر کمتر از زمان دستیابی حافظه اصلی

1- Cache

است. حافظه کش سریع ترین جزء در سلسله مراتب حافظه بوده و به حوالی سرعت CPU نزدیک است. ایده اصلی سازمان کش این است که با نگهداری دستورالعمل ها و داده هایی که بیشتر دستیابی می شوند در یک حافظه کش اصلی سریع، زمان متوسط دستیابی حافظه به زمان دستیابی کش نزدیک شود. گرچه کش تنها بخش کوچکی از حافظه اصلی است، بدلیل خاصیت محلّیت ارجاع به حافظه، بخش عمده اطلاعات یا در واقع نیازهای حافظه ای در حافظه سریع کش، یافت خواهد شد.

عملکرد اساسی سازمان حافظه بدین شرح است. وقتی که CPU نیاز به دستیابی به حافظه دارد، حافظه کش بررسی می شود. اگر کلمه موردنظر در کش یافت شود، از این حافظه سریع خوانده می شود. اگر کلمه ای که بوسیله CPU آدرس دهی داده شده در کش یافته نشود، حافظه اصلی برای خواندن کلمه آدرس دهی می شود. سپس بلاکی از کلمات از جمله کلمه ای که در آن لحظه دستیابی شد از حافظه اصلی به حافظه کش انتقال می یابد. اندازه بلاک می تواند از یک کلمه (کلمه ای که در لحظه اول دستیابی می شود) و 16 کلمه مجاور آن باشد. بدین ترتیب، مقداری داده به کش انتقال می یابد و لذا ارجاعات آینده به حافظه، کلمات مورد نیاز را در حافظه کش سریع بیابد.

عملکرد حافظه کش غالباً براساس کمیتی که نسبت برد¹ نامیده می شود سنجیده می گردد. هنگامی که CPU به حافظه مراجعه کند و کلمه موردنظر را در حافظه کش بیابد گفته می شود برد صورت گرفته است. اگر کلمه موردنظر در حافظه کش یافته نشود، مسلماً در حافظه اصلی است و باخت² صورت گرفته است. نسبت تعداد بردها تقسیم بر تعداد کل ارجاعات CPU به حافظه (برد بعلاوه باخت) نسبت برد است. بهترین نسبت برد اندازه گیری شده تجربی، اجرای برنامه های نمونه در کامپیوتر و سنجش تعداد بردها و باخت ها در یک فاصله زمانی معین است. در این رابطه نسبت بردهای 0.9 و بالاتر گزارش شده است. این نسبت برد بالا اعتبار خاصیت محلّیت ارجاع را تأیید می کند.

زمان متوسط دستیابی به حافظه برای یک سیستم کامپیوتر با استفاده از حافظه کش می تواند بمقدار قابل ملاحظه ای اصلاح شود. اگر نسبت برد نسبتاً بالا باشد بطوری که در بیشتر اوقات CPU به جای حافظه اصلی به حافظه کش مراجعه کند، زمان میانگین دستیابی به زمان دستیابی حافظه سریع کش نزدیکتر خواهد بود. مثلاً کامپیوتری با زمان 100ns برای حافظه کش، و زمان دستیابی 1000ns برای حافظه اصلی، و نسبت برد 0.9 زمان میانگین دستیابی 200ns را خواهد داشت. این نتیجه یک اصلاح قابل توجه نسبت به کامپیوتر بدون حافظه کش است که دارای زمان دستیابی 1000ns می باشد.

مشخصه اصلی حافظه کش زمان دستیابی سریع آنست. بنابراین، وقتی جستجویی برای کلمه در کش صورت گیرد اتلاف زمانی کوتاه بوده یا اصلاً زمانی تلف نمی شود. انتقال داده از حافظه اصلی به حافظه کش فرآیند نگاهت³ نام دارد. سه نوع رویه نگاهت هنگام بررسی سازمان حافظه کش عملاً مورد توجه است.

۱- نگاهت تداعیگر⁴

1- Hit ratio

2- Miss

3- Mapping

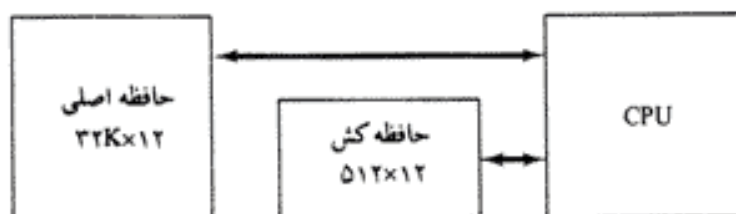
4- Associative Mapping

۲- نگاشت مستقیم^۱۳- نگاشت تداعیگر مجموعه‌ای^۲

برای کمک به درک بحث این سه رویه نگاشت ما از مثال خاصی برای سازمان حافظه که در شکل ۱۰-۱۲ نشان داده شده است استفاده می‌کنیم. حافظه اصلی می‌تواند 32k کلمه 12 بیتی را ذخیره کند. حافظه کش قادر است 512 عدد از این کلمات را در هر زمان ذخیره نماید. برای هر کلمه‌ای که در کش ذخیره شود، یک نسخه کپی در حافظه اصلی وجود دارد. CPU با هر دو حافظه ارتباط دارد. ابتدا آدرس 15 بیتی را به حافظه کش ارسال می‌نماید. اگر بر دی صورت گیرد، CPU داده 12 بیتی را از کش می‌پذیرد. اگر باخت صورت گیرد، CPU کلمه را از حافظه اصلی خوانده و سپس آن کلمه به حافظه کش منتقل می‌شود.

نگاشت تداعیگر

در سریع‌ترین و انعطاف‌پذیرترین حافظه کش از یک حافظه تداعیگر استفاده می‌شود. این سازمان در شکل ۱۱-۱۳ نشان داده شده است. حافظه تداعیگر هم آدرس و هم محتوای (داده) کلمه حافظه را ذخیره می‌کند. این توانایی اجازه می‌دهد تا هر مکان در حافظه کش بتواند هر یک از کلمات حافظه اصلی را ذخیره کند. دیاگرام، سه کلمه را که در حال حاضر در حافظه کش ذخیره شده نشان می‌دهد. مقدار آدرس 15 بیتی بصورت یک عدد هشت هشتی پنچ رقمی و کلمه 12 بیتی متناظر آن بصورت یک عدد هشت هشتی چهار رقمی نشان داده شده است. یک آدرس 15 بیتی بوسیله CPU در ثبات آرگومان قرار داده شده و حافظه تداعیگر برای تطبیق با آن آدرس جستجو می‌شود. اگر آدرس مورد نظر پیدا شود، داده 12 بیتی متناظر آن خوانده شده و به CPU ارسال می‌شود. اگر تطبیق صورت نگیرد، حافظه اصلی برای یافتن کلمه دستیابی می‌شود. سپس جفت کمیت آدرس - داده به حافظه کش تداعیگر منتقل می‌شود. اگر کش پر باشد، باید یک جفت آدرس - داده از آن خارج شود تا جا برای زوج مورد نظر که در حافظه کش قرار ندارد باز شود. تصمیم در مورد زوجی که باید حذف شود براساس الگوریتمی است که طراح برای کش انتخاب می‌کند. یک رویه ساده این است که هر وقت که یک کلمه جدید از حافظه اصلی تقاضا شد



شکل ۱۰-۱۲ مثالی از حافظه کش

1- Direct Mapping

2- Set - Associative Mapping

آدرس CPU (15 بیت)

↓

ثبات آرگومان

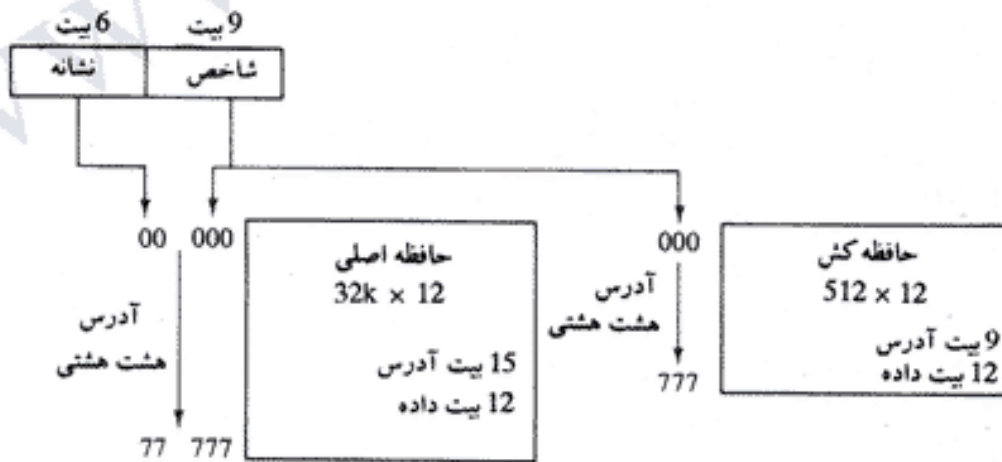
آدرس	داده
0 1 0 0 0	3 4 5 0
0 2 7 7 7	6 7 1 0
2 2 3 4 5	1 2 3 4

شکل ۱۱-۱۲ حافظه کش با نگاشت تداعیگر
(همه اعداد در مبنای هشت هستند)

سلول های موجود در حافظه کش بصورت چرخشی^۱ جایگزین شوند. این رویه، سیاست اولین ورودی - اولین خروجی (FIFO) نام دارد.

نگاشت مستقیم

حافظه های تداعیگر در مقایسه با حافظه های دستیابی تصادفی بدلیل مدار اضافی همراه هر سلول، گرانتر است. امکان استفاده از حافظه با دستیابی تصادفی برای حافظه کش در شکل ۱۲-۱۲ مورد بررسی



شکل ۱۲-۱۲ روابط آدرس دهی بین حافظه های کش و اصلی

قرار گرفته است. آدرس 15 بیتی CPU به دو میدان تقسیم شده است. نُه بیت کم ارزشتر میدان شاخص^۱ و شش بیت باقیمانده میدان نشانه^۲ را تشکیل می دهند. شکل نشان می دهد که حافظه اصلی نیاز به آدرسی دارد که هم شامل بیت های شاخص و هم شامل بیت های نشانه می باشد. تعداد بیت های میدان شاخص برابر با تعداد بیت های آدرس لازم برای دستیابی به حافظه کش است.

در حالت کلی، 2^k کلمه در حافظه کش و 2^n کلمه در حافظه اصلی وجود دارد. آدرس حافظه n بیتی به دو میدان تقسیم می شود: k بیت برای میدان شاخص و $n-k$ بیت برای میدان نشانه. سازمان حافظه کش با نگاشت مستقیم، از آدرس n بیتی برای دستیابی به حافظه اصلی و k بیت شاخص برای دستیابی به حافظه کش استفاده می کند. سازمان داخلی کلمات در حافظه کش در شکل ۱۳-۱۲ (ب) نشان داده شده است. هر کلمه در کش متشکل از کلمه داده و نشانه مربوطه آنست. وقتی که کلمه جدیدی برای اولین بار به حافظه کش آورده شود، بیت های نشانه همراه با بیت های داده ذخیره می شوند. وقتی که CPU یک سیگنال تقاضای حافظه تولید می کند، میدان شاخص برای دستیابی به حافظه کش بکار می رود. میدان نشانه آدرس CPU، با نشانه موجود در کلمه خوانده شده از حافظه کش مقایسه می شود. اگر دو نشانه با هم تطبیق کنند یک برد صورت می گیرد و کلمه داده مورد نظر در حافظه کش قرار دارد. اگر تطبیق صورت نگیرد، یک باخت وجود داشته و کلمه مطلوب از حافظه اصلی خوانده می شود. این کلمه سپس همراه با نشانه جدید در حافظه کش ذخیره شده و جایگزین مقدار قبلی می شود. عیب نگاشت مستقیم

آدرس شاخص	نشانه	داده
000	00	1 2 2 0
777	02	6 7 1 0

(ب) حافظه کش

شکل ۱۳-۱۲ سازمان حافظه کش مبتنی بر نگاشت مستقیم

آدرس حافظه	داده های حافظه
00000	1 2 2 0
00777	2 3 4 0
01000	3 4 5 0
01777	4 5 6 0
02000	5 6 7 0
02777	6 7 1 0

(الف) حافظه اصلی

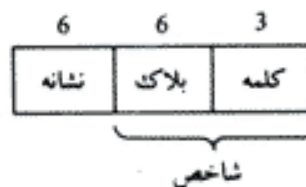
1- Index Field

2- Tag Field

این است که اگر دو یا چند کلمه که آدرسهایشان دارای شاخص یکسانی ولی نشانه متفاوت اند مرتباً دستیابی شوند نسبت برد بمیزان قابل توجهی افت نماید. با این وجود این امکان باین ترتیب به حداقل رسانده شده است که این چنین کلمات در محدوده آدرسهای نسبتاً دور از هم قرار دارند (در این مثال مضرب 512 مکان). برای درک نحوه عملکرد سازمان نگاشت مستقیم مثال عددی شکل ۱۳-۱۲ را در نظر بگیرید. کلمه واقع در آدرس صفر اخیراً در حافظه کش ذخیره شده است (شاخص = 000، نشانه = 00، داده = 1220). فرض کنید که CPU اکنون بخواهد به کلمه‌ای واقع در آدرس 02000 دستیابی کند. آدرس شاخص 000 است، بنابراین این آدرس برای دستیابی به کش استفاده می‌شود. سپس دو نشانه مقایسه می‌شوند. نشانه کش 00 است ولی نشانه آدرس 02 است، که حالت تطابق را ایجاد نمی‌کند. بنابراین حافظه اصلی دستیابی شده و کلمه داده 5670 به CPU انتقال داده می‌شود. سپس کلمه کش در اندیس آدرس 000 با نشانه 02 و داده 5670 جایگزین می‌شود.

مثال نگاشت مستقیم که در بالا توصیف شد از بلاک‌هایی با اندازه یک کلمه استفاده می‌کند. سازمان مشابهی اما با استفاده از هشت کلمه در شکل ۱۴-۱۲ نشان داده شده است. میدان شاخص اکنون به دو بخش تقسیم شده است: میدان بلاک^۱ و میدان کلمه^۲. در یک حافظه کش 512 کلمه‌ای، 64 بلاک 8 کلمه‌ای وجود دارد زیرا $512 = 8 \times 64$ است. شماره بلاک با یک میدان 6 بیتی و محل کلمه در بلاک با یک میدان سه بیتی مشخص می‌شود. میدان نشانه که در حافظه کش ذخیره شده برای هر هشت کلمه یک بلاک یکسان است. هر بار یک باخت اتفاق بیفتد، تمام بلاک هشت کلمه‌ای باید از حافظه اصلی به حافظه کش انتقال یابد. هر چند این عمل وقتی اضافی را صرف می‌کند ولی نسبت برد بدلیل ماهیت ترتیبی برنامه‌های کامپیوتر با احتمال بسیار افزایش می‌یابد.

	شاخص	نشانه	داده
بلاک 0	000	0 1	3 4 5 0
	007	0 1	6 5 7 8
بلاک 1	010		
	017		
بلاک 63	770	0 2	
	777	0 2	6 7 1 0



شکل ۱۳-۱۲ حافظه کش با نگاشت مستقیم و اندازه بلاک 8 کلمه

1- Block Field

2- Word Field

نگاشت تداعیگر مجموعه‌ای

قبلاً اشاره شد که عیب نگاشت مستقیم این است که دو کلمه با شاخص یکسان در بیت آدرس ولی با مقادیر نشانه متفاوت نمی‌توانند بطور همزمان در حافظه کش قرار گیرند. نوع سومی از سازمان حافظه کش، که نسبت به نگاشت مستقیم از لحاظ اشکال فوق اصلاح شده است و نگاشت تداعیگر مجموعه‌ای خوانده می‌شود، می‌تواند دو یا چند کلمه با شاخص یکسان در آدرس را نگهداری کند. هر کلمه داده همراه با نشانه آن ذخیره می‌شود و تعداد ارقام نشانه - داده در هر کلمه از حافظه کش یک مجموعه نامیده می‌شود. مثالی از سازمان حافظه کش تداعیگر مجموعه‌ای برای مجموعه‌ای با اندازه 2 در شکل ۱۲-۱۵ نشان داده شده است. هر آدرس شاخص به دو کلمه داده و نشانه مربوطه‌اش اشاره می‌کند. هر نشانه به شش بیت نیاز دارد و هر کلمه داده 12 بیتی است. بنابراین طول کلمه $2(6+12)=36$ بیت خواهد بود. هر آدرس شاخص 9 بیتی می‌تواند برای 512 کلمه جا مشخص کند. بنابراین اندازه حافظه کش 512×36 بیت است. این حافظه می‌تواند 1024 کلمه از حافظه اصلی را جای داد زیرا محتوای هر کلمه از حافظه کش دو کلمه داده است. بطور کلی، یک حافظه تداعیگر مجموعه‌ای با اندازه مجموعه k قادر است k کلمه از حافظه اصلی را در هر کلمه حافظه کش جای دهد.

اعداد هشت هشتی در شکل ۱۲-۱۵ براساس محتویات حافظه اصلی در شکل ۱۳-۱۲ (الف) لیست شده‌اند. کلمات در آدرس‌های 01000 و 02000 از حافظه اصلی در حافظه کش با آدرس شاخص 000 ذخیره شده‌اند. بطور مشابه، کلمات در آدرس‌های 02777 و 00777 در آدرس شاخص 777 در حافظه کش ذخیره شده‌اند. هنگامی که CPU یک تقاضای حافظه تولید می‌کند، مقدار شاخص آدرس برای دستیابی به کش مورد استفاده قرار می‌گیرد. سپس میدان نشانه آدرس CPU با هر دو نشانه در حافظه کش مقایسه می‌شود تا وقوع انطباق معین گردد. منطق مقایسه با جستجوی نشانه‌ها در مجموعه، به شیوه‌ای مشابه با جستجوی حافظه تداعیگر انجام می‌شود. نام "تداعیگر مجموعه‌ای" نیز از همینجا گرفته شده

شاخص	نشانه	داده	نشانه	داده
000	01	3450	02	5670
777	02	6710	00	2340

شکل ۱۲-۱۵ حافظه کش با نگاشت تداعیگر در مجموعه دوتایی (اعداد در مبنای هشت هستند)

است. نسبت برد با افزایش اندازه مجموعه افزایش می یابد زیرا در اینصورت کلمات بیشتری با شاخص یکسان ولی نشانه های متفاوت می تواند در حافظه کش قرار داشته باشند. اما افزایش اندازه مجموعه تعداد بیت های کلمات کش را افزایش داده و مدارهای مقایسه پیچیده تری را نیاز خواهند داشت. وقتی که باختی در حافظه کش تداعیگر مجموعه ای رخ دهد و مجموعه پر باشد، لازم است تا به جای یکی از اقلام نشانه - داده، مقدار جدیدی جایگزین شود. متداولترین الگوریتم جایگزینی بکار رفته عبارتند از: جایگزینی تصادفی، اولین ورودی - اولین خروجی (FIFO)، و قدیمترین مورد استفاده^۱ (LRU). با سیاست جایگزینی تصادفی کنترل یک قلم نشانه - داده را برای جایگزین تصادفی برمیگزیند. رویه FIFO موردی که بیش از همه در مجموعه بوده است را انتخاب می کند. الگوریتم LRU، اطلاعات Least Recently Used بوسیله CPU را برمیگزیند. هر دو روش FIFO و LRU می توانند با افزایش چند بیت در هر کلمه کش پیاده سازی شوند.

نوشتن در حافظه کش

یکی از جنبه های مهم سازمان کش به درخواست های نوشتن مربوط است. وقتی که CPU در حین عمل خواندن کلمه ای را در کش می یابد، حافظه اصلی در عمل انتقال دخالتی نمی کند. با این وجود اگر عمل یک نوشتن باشد، دو راه برای عملکرد سیستم وجود دارد.

ساده ترین و متداولترین رویه های مورد استفاده بهنگام کردن حافظه اصلی در قبال هر نوشتن حافظه است، که حافظه کش هم به موازات آن، اگر دارای کلمه موجود در آدرس مشخص شده باشد، بهنگام می گردد. این روش رویه کامل نویسی^۲ خوانده می شود. مزیت این روش اینست که حافظه اصلی همواره داده یکسانی با حافظه کش دارد. این مشخصه در سیستم هایی که از روش انتقال با دستیابی مستقیم حافظه استفاده می کنند اهمیت دارد. روش مذکور اعتبار داده هایی را که در تمام مدت در حافظه اصلی اند تضمین می کند، بنابراین وسیله I/O که از طریق DMA ارتباط دارد جدیدترین اطلاعات را دریافت خواهد کرد.

روش دوم، روش پس نویسی^۳ پس نویسی خوانده می شود. در این روش فقط مکان کش در حین عمل نوشتن بهنگام می شود. سپس مکان مزبور با یک پرچم مشخص می شود تا بعداً که کلمه موجود در آن از حافظه کش خارج می شود بداخل حافظه اصلی کپی شود. دلیل بکارگیری روش پس نویسی این است که در زمان قرار گرفتن یک کلمه در حافظه کش، ممکن است چندین بار بهنگام شود؛ با این وجود مادامی که کلمه در کش قرار می گیرد، قدیمی بودن کپی واقع در حافظه اصلی مهم نیست، زیرا تقاضاهای کلمه از حافظه کش تأمین می شوند. فقط هنگامی که کلمه از حافظه کش خارج می شود لازم است نسخه صحیحی در حافظه اصلی نوشته شود. نتایج تحلیلی نشان می دهد که تعداد نوشتن ها در حافظه در یک برنامه، بین ۱۰ تا ۳۰ درصد کل ارجاعات به حافظه است.

1- Least Recently Used

2- Write Through

3- Write - Back

مقدار دهی اولیه در حافظه کش

جنبه دیگری از سازمان کش که باید مورد توجه قرار گیرد مقداردهی اولیه آنست. حافظه کش هنگامی که کامپیوتر روشن می شود یا وقتی که حافظه اصلی با مجموعه کاملی از برنامه ها از حافظه کمکی بار می شود مقداردهی می گردد. پس از مقدار دهی اولیه حافظه کش خالی تصور خواهد شد، ولی در اصل حاوی مقداری داده های بی اعتبار است. مرسوم است تا به هر کلمه در کش یک بیت اعتبار^۱ اختصاص یابد تا مشخص شود آیا کلمه حاوی داده معتبر است یا خیر.

حافظه کش با صفر کردن تمام بیت های معتبرش مقداردهی اولیه می شود. اولین باری که یک کلمه خاص از حافظه اصلی بار شود بیت اعتبار آن به مقدار 1 تغییر می یابد و تا مقداردهی اولیه بعدی همچنان برابر 1 باقی می ماند. معرفی بیت اعتبار بمعنی این است که آن کلمه با هیچ کلمه ای جایگزین نشود مگر اینکه بیت اعتبار 1 باشد و عدم تطابق نشانه ها رخ دهد. اگر بیت اعتبار 0 باشد، کلمه جدید بطور اتوماتیک جایگزین داده بی اعتبار می گردد. بنابراین شرط مقداردهی اولین این اثر را دارد که تا زمانی که حافظه نهان از داده های معتبر پر شود موجب باخت می شود.

۶-۱۲ حافظه مجازی

در یک سیستم سلسه مراتبی حافظه، برنامه ها و داده ها ابتدا در حافظه کمکی ذخیره می شوند. قسمت هایی از یک برنامه یا داده اگر قرار باشد مورد استفاده CPU واقع شوند به حافظه اصلی آورده می شوند. حافظه مجازی^۲، مفهومی مورد استفاده در برخی از سیستم های کامپیوتری بزرگ است که این امکان را به کاربر می دهد تا برنامه را بگونه ای بنویسد که گوئی فضای حافظه بزرگی باندازه تمام حافظه کمکی را در اختیار دارد. هر آدرسی که بوسیله CPU ارجاع شود از یک مکانیزم تبدیل آدرس عبور نموده و از یک آدرس مجازی به یک آدرس فیزیکی در حافظه اصلی بدل می شود. حافظه مجازی باین خاطر بکار می رود تا در برنامه نویسی تصور را ایجاد کند که علیرغم داشتن یک حافظه نسبتاً کوچک، حافظه زیادی را در اختیار دارد. یک سیستم حافظه مجازی مکانیزمی برای ترجمه آدرس های تولید شده بوسیله برنامه، به آدرس های صحیح در حافظه اصلی را فراهم می کند. این عمل بصورت دینامیکی ضمن اجرای برنامه ها در CPU انجام می شود ترجمه یا نگاشت به صورت اتوماتیک بکمک جدول و بوسیله سخت افزار انجام می شود.

فضای آدرس و فضای حافظه

آدرسی که بوسیله برنامه نویس مورد استفاده قرار می گیرد و آدرس مجازی^۳ و مجموعه این آدرسها فضای آدرس^۴ خوانده می شود. یک آدرس در حافظه اصلی، مکان^۵ یا آدرس فیزیکی^۶ نامیده می شود.

1- Valid Bit
4- Address Space

2- Virtual Memory
5- Location

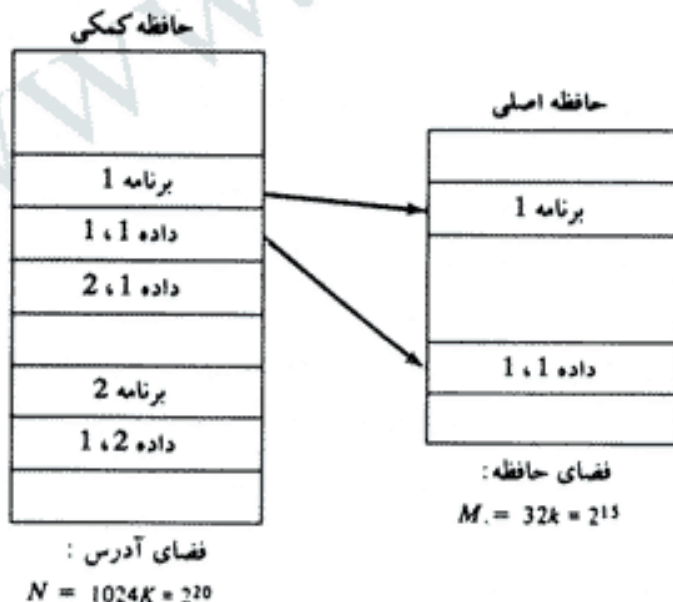
3- Virtual Address
6- Physical Address

مجموعه چنین مکان‌هایی فضای حافظه^۱ خوانده می‌شود. بنابراین فضای آدرس مجموعه‌ای از آدرس‌های تولید شده بوسیله برنامه‌ها می‌باشند، که به دستورالعمل‌ها و داده‌ها اشاره می‌کنند؛ و فضای حافظه هم متشکل از مکان‌های واقعی حافظه اصلی می‌باشند، که مستقیماً برای پردازش قابل آدرسی‌دهی هستند. در اکثر کامپیوترها فضاهای آدرس و حافظه یکی هستند. در کامپیوترهایی که حافظه مجازی دارند فضای آدرس بزرگتر از فضای حافظه است.

بعنوان تشریحی بر مطلب فوق، یک کامپیوتر با حافظه اصلی 32k کلمه ($K=1024$) را در نظر بگیرید. برای مشخص کردن آدرس فیزیکی در حافظه به پانزده بیت نیاز است چون $32K = 2^{15}$ است. فرض کنید که کامپیوتر دارای حافظه‌ای مجازی برای ذخیره $1024K = 2^{20}$ کلمه باشد. بنابراین حافظه کمکی دارای ظرفیت ذخیره سازی اطلاعات معادل با 32 برابر حافظه اصلی است. هرگاه فضای آدرس را با N و فضای حافظه را با M نشان دهیم، برای مثال فوق $N = 1024K$ و $M = 32K$ خواهد بود.

در سیستم‌های کامپیوتر چند برنامه‌ای با برنامه‌ها و داده‌ها براساس تقاضای CPU از حافظه اصلی به حافظه کمکی و یا از آن به حافظه اصلی انتقال می‌یابد. فرض کنید که برنامه 1 در حال حاضر در CPU در حال اجراست. برنامه 1 و بخشی از داده مربوط به آن از حافظه کمکی به حافظه اصلی طبق شکل ۱۶-۱۲ انتقال یابند. بخشهای مختلف برنامه‌ها و داده‌ها ضرورتاً لازم نیست در مکان‌های متوالی قرار داشته باشند، زیرا اطلاعات بطور مداوم به داخل و خارج حافظه انتقال می‌یابد و نیز ممکن است فضاهای خالی در مکان‌هایی پراکنده حافظه موجود باشند.

در یک سیستم با حافظه مجازی به برنامه نویس چنین گفته می‌شود که آنها تمام فضای آدرس را در اختیار دارند. بعلاوه، میدان آدرس کد دستورالعمل دارای تعداد بیت کافی برای مشخص کردن تمام



شکل ۱۶-۱۲ رابطه بین فضای آدرس و حافظه در یک سیستم حافظه مجازی

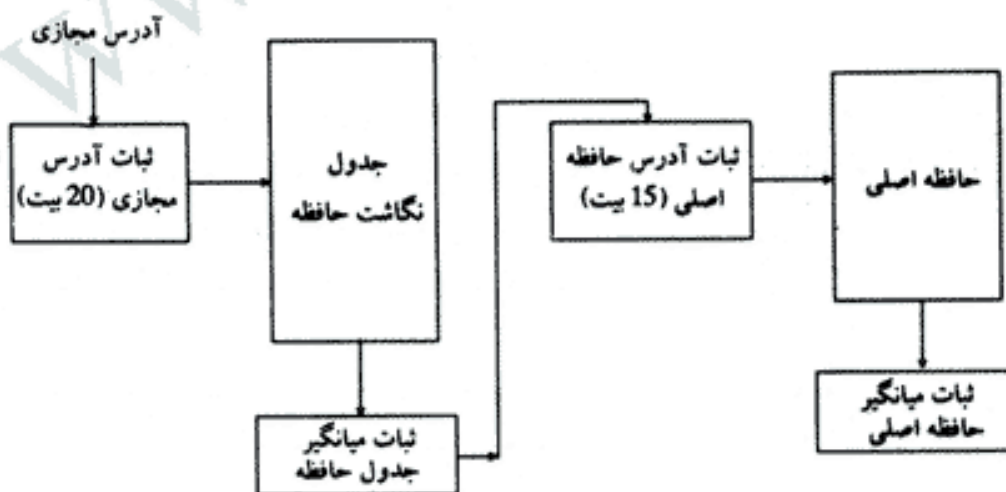
1- Memory Space

حافظه های مجازی است. در مثال ما، میدان آدرس یک کد دستورالعمل، از 20 بیت تشکیل شده است ولی آدرس های حافظه فیزیکی باید بوسیله 15 بیت معین شوند. بنابراین CPU دستورالعمل ها و داده ها را با آدرس 20 بیتی فرا می خواند، ولی اطلاعات در این آدرس بایستی از حافظه فیزیکی برداشت شود زیرا دستیابی به حافظه مجازی برای تک تک کلمات بیش از حد وقت گیر خواهد بود. (به یاد داشته باشید که برای کارایی عمل انتقال، حافظه کمکی یک رکورد کامل را به حافظه اصلی منتقل می کند. سپس همانطور که در شکل ۱۷-۱۲ دیده می شود، جدولی برای نگاشت آدرس مجازی 20 بیتی به آدرس فیزیکی 15 بیتی لازم خواهد بود. این تبدیل آدرس یک عمل دینامیکی است، یعنی هر آدرس به محض ارجاع CPU به یک کلمه تبدیل می شود.

جدول تبدیل^۱ می تواند در یک حافظه جداگانه طبق شکل ۱۷-۱۲ و یا در حافظه اصلی باشد. در حالت اول، یک واحد حافظه اضافی و نیز زمان دستیابی به حافظه اضافی لازم است. در حالت دوم، جدول حافظه اصلی را اشغال می کند و دو بار دستیابی به حافظه لازم می باشد ضمن اینکه برنامه با نصف سرعت عادی اجرا می شود. روش سوم استفاده از حافظه تداپیگر است که در ذیل تشریح می شود.

نگاشت آدرس با استفاده از صفحات

پیاده سازی جدول تبدیل یا نگاشت آدرس بشرطی ساده می شود که اطلاعات در فضای آدرس و فضای حافظه هر یک به گروه هایی با اندازه ثابت تقسیم شوند. حافظه فیزیکی به گروه هایی با اندازه مساوی که بلاک خوانده می شود و هر یک می تواند از 64 تا 4096 کلمه داشته باشد شکسته می شود. اصطلاح صفحه^۲ اشاره به گروه هایی از فضای حافظه هم اندازه دارد. مثلاً، اگر یک صفحه یا بلاک از 1K



شکل ۱۷-۱۲ جدول حافظه برای نگاشت آدرس های مجازی

کلمه تشکیل شده باشد، سپس با استفاده از مثال قبل، فضای آدرس به 1024 صفحه و حافظه اصلی به 32 بلاک تقسیم می شود. هر چند یک صفحه و بلاک هر دو به گروهی از کلمات 1K تقسیم شده اند، یک صفحه به سازمان فضای آدرس و یک بلاک به سازمان فضای حافظه اشاره دارد. برنامه ها نیز منقسم شده به صفحات تصور می شوند. انتقال بخش هایی از برنامه ها از حافظه کمکی به حافظه اصلی بصورت رکوردهایی با اندازه مساوی با یک صفحه صورت می گیرد. گاهی از اصطلاح "کادر صفحه" برای اشاره به بلاک استفاده می شود. یک کامپیوتر با فضای آدرس 8K و فضای حافظه 4K را در نظر بگیرید. اگر ما هر یک را به گروه های 1K کلمه تقسیم کنیم هشت صفحه و چهار بلاک مطابق شکل ۱۸-۱۲ خواهیم داشت. در هر لحظه، تا چهار صفحه از فضای آدرس می تواند در هر یک از چهار بلاک از حافظه اصلی قرار گیرد.

نگاشت از فضای آدرس به فضای حافظه می تواند ساده شود بشرطی که هر آدرس مجازی با دو عدد نشان داده شود: یک آدرس شماره صفحه و یک خط در صفحه. در کامپیوتری با 2^p کلمه در هر صفحه، p برای مشخص کردن آدرس خط و بقیه بیت های مرتبه بالاتر از حافظه مجازی برای مشخص کردن شماره صفحه مورد استفاده قرار می گیرد. در مثال شکل ۱۸-۱۲، آدرس مجازی دارای 13 بیت است. چون هر صفحه از $1024 = 2^{10}$ کلمه تشکیل شده است، سه بیت با ارزشتر آدرس مجازی یکی از هشت صفحه را مشخص خواهد کرد و 10 بیت کم ارزشتر آدرس سطر را در صفحه فراهم می کند. توجه کنید که آدرس سطر در فضای آدرس و فضای حافظه یکسان است؛ تنها تبدیل لازم از شماره صفحه به شماره بلاک می باشد. سازمان جدول نگاشت یا تبدیل حافظه در یک سیستم صفحه بندی شده در شکل ۱۹-۱۲ نشان داده شده است. جدول صفحات حافظه از هشت کلمه تشکیل می شود، یعنی یک کلمه بازاء هر صفحه آدرس جدول

صفحه 0
صفحه 1
صفحه 2
صفحه 3
صفحه 4
صفحه 5
صفحه 6
صفحه 7

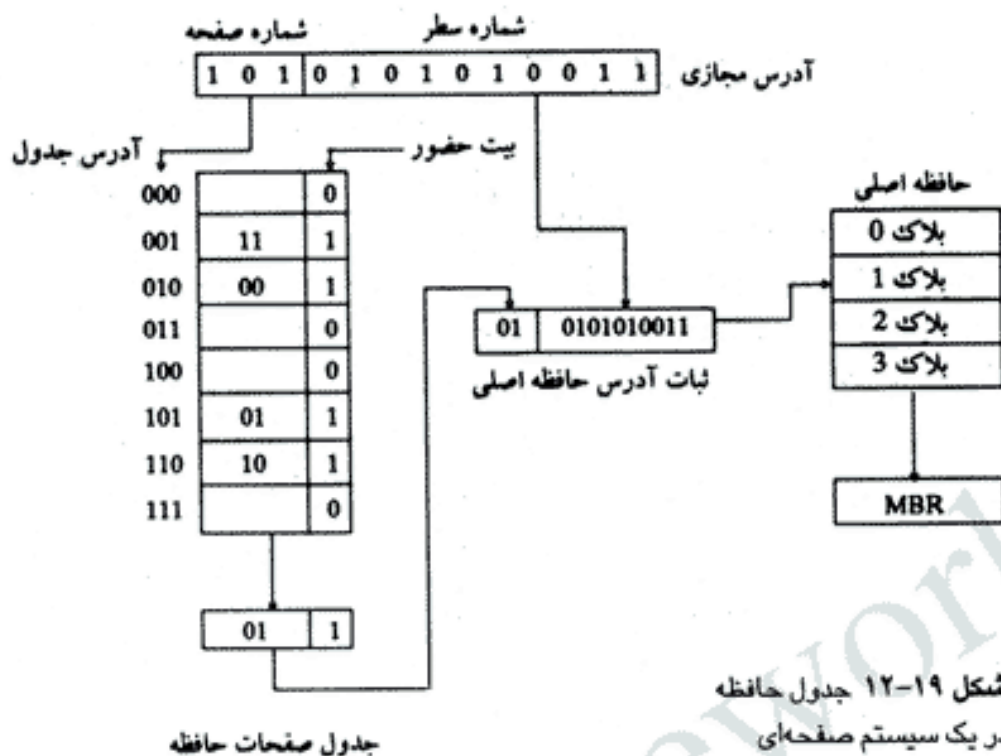
فضای آدرس:
 $N = 8K = 2^{13}$

بلاک 0
بلاک 1
بلاک 2
بلاک 3

فضای حافظه:
 $M = 4K = 2^{12}$

شکل ۱۸-۱۲ تقسیم فضای آدرس و فضای حافظه به دسته های 1K کلمه ای

صفحات شماره صفحه را نشان می دهد و محتوای کلمه شماره بلاکی را که صفحه مورد نظر در حافظه اصلی در آن قرار دارد مشخص می نماید. جدول نشان می دهد که صفحات 1، 2، 5 و 6 هم اکنون در حافظه اصلی بترتیب بلوک های 3، 0، 1 و 2 قرار دارند. یک بیت حضور در هر خانه مشخص می کند که صفحه از حافظه کمکی به حافظه اصلی منتقل شده است یا خیر. مقدار 0 در هر بیت حضور نشان می دهد که این صفحه در حافظه اصلی موجود نیست. CPU به هر کلمه واقع در حافظه با یک آدرس مجازی 13 بیتی رجوع می کند. سه بیت پرارزشتر آدرس مجازی شماره صفحه و نیز آدرسی را برای جدول صفحات حافظه مشخص می نماید. محتوای کلمه در جدول صفحات حافظه در آدرس شماره صفحه خوانده شده و در ثبات میانگیر جدول حافظه قرار داده می شود. اگر بیت حضور مقدار 1 را داشته



باشد شماره بلاکی که باین ترتیب خوانده می‌شود به دو بیت پرارزشتر ثبات آدرس حافظه اصلی منتقل می‌گردد. شماره سطر از آدرس مجازی به 10 بیت کم ارزشتر ثبات آدرس منتقل می‌شود. یک سیگنال خواندن که به حافظه اصلی متصل است محتوای کلمه را به ثبات میانگیر حافظه منتقل و در آنجا آماده استفاده CPU خواهد بود. اگر بیت حضور در کلمه خوانده شده از جدول صفحه 0 باشد، بمعنی این است که محتوای کلمه‌ای که بوسیله آدرس مجازی به آن اشاره شده است در حافظه اصلی نیست. سپس قبل از ادامه محاسبات، بافراخوانی سیستم عامل صفحه مطلوب از حافظه کمکی برداشت و در حافظه اصلی قرار داده می‌شود.

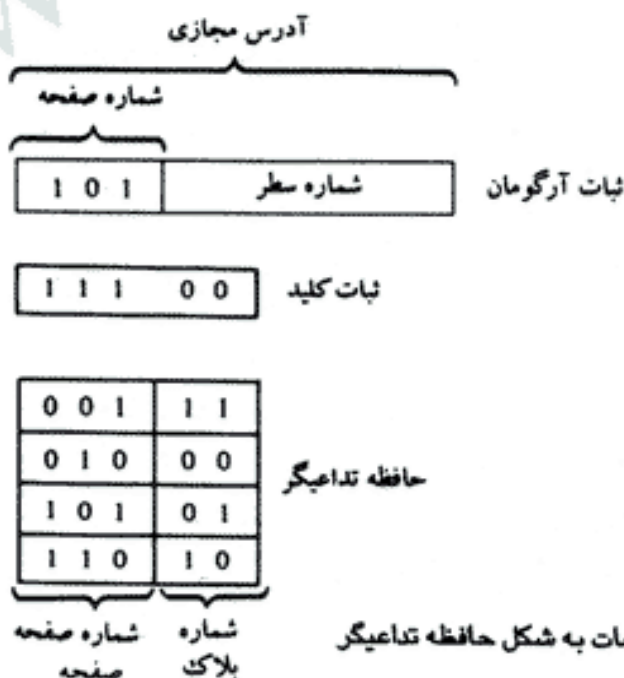
جدول صفحات از حافظه تداغیگر

یک جدول صفحات حافظه با دستیابی تصادفی کارآیی لازم را از نظر بهره‌گیری دارا نیست. در مثال شکل ۱۲-۱۹ دیدیم که حداقل به هشت کلمه حافظه نیاز است، که هر یک به یک صفحه تعلق داشتند، ولی همواره چهار کلمه علامت خالی خواهند داشت زیرا حافظه اصلی گنجایش بیش از چهار بلاک را ندارد. بطور کلی، سیستمی با n صفحه و m بلاک یک جدول حافظه - صفحه n مکانی را لازم دارد که در آن حداکثر m بلاک حاوی شماره بلاک و بقیه خالی خواهد بود. بعنوان یک مثال عددی یک فضای آدرس 1024K کلمه در یک فضای حافظه 32K کلمه را در نظر بگیرید. اگر هر صفحه یا بلاک 1K کلمه داشته باشد، تعداد صفحات 1024 و تعداد بلاک‌ها 32 خواهد بود. ظرفیت جدول حافظه - صفحه باید 1024 کلمه و فقط 32 مکان دارای بیت حضور 1 می‌باشند. در هر لحظه حداقل 992 مکان خالی و بلااستفاده است. راه کارآتری برای سازمان دهی جدول صفحات این است که آن را با تعدادی از کلمات برابر با بلاک‌ها در حافظه اصلی بسازیم. بدین ترتیب اندازه حافظه کاهش یافته و هر مکان بطور کامل مورد استفاده قرار

می‌گیرد. این روش را می‌توان بکمک حافظه تداعیگر به انجام رساند که هر کلمه در حافظه حاوی یک شماره صفحه به همراه شماره بلاک متناظر آن باشد. میدان صفحه در هر کلمه با شماره صفحه در آدرس مجازی مقایسه می‌شود. اگر انطباق رخ دهد کلمه از حافظه خوانده می‌شود و شماره بلاک آن استخراج می‌گردد. مجدداً حالت هشت صفحه و چهار بلاک را در نظر بگیرید. همانطور که در شکل ۱۹-۱۲ ملاحظه شد به جای جدول حافظه صفحه از نوع با دستیابی تصادفی از آرایه حافظه تداعیگر استفاده می‌کنیم که هر یک از واردهای حافظه تداعیگر از دو میدان ساخته شده است (شکل ۲۰-۱۲). سه بیت اول میدانی را برای ذخیره شماره صفحه مشخص می‌کند. دو بیت آخر میدانی را برای ذخیره شماره بلاک معین می‌نماید. آدرس مجازی در ثبات آرگومان قرار گرفته است. بیت‌های متعلق به شماره صفحه در ثبات آرگومان با تمام شماره صفحات در میدان صفحه حافظه تداعیگر مقایسه می‌شوند. اگر شماره صفحه پیدا شود، کلمه ۵ بیتی از حافظه خوانده می‌شود. شماره بلاک متناظر، که در همان کلمه است، به ثبات آدرس حافظه اصلی منتقل می‌گردد. اگر تطبیق صورت نگیرد، برای آوردن صفحه از حافظه کمکی یک فراخوانی سیستم عامل صورت می‌گیرد.

جایگزینی صفحات

یک سیستم حافظه مجازی ترکیبی از تکنیک‌های سخت‌افزاری و نرم‌افزاری است. سیستم مدیریت حافظه نرم‌افزاری تمام عملیات نرم‌افزاری را برای بهره‌برداری موثر از فضای حافظه انجام می‌دهد. این سیستم باید در این موارد تصمیم بگیرد: (۱) کدام صفحه از حافظه اصلی باید برای ایجاد فضای لازم برای یک صفحه جدید حذف شود، (۲) چه زمانی باید یک صفحه جدید از حافظه کمکی به حافظه اصلی منتقل شود، و (۳) صفحه مورد نظر باید در چه محلی از حافظه اصلی قرار داده شود. مکانیزم نگاشت سخت‌افزاری و نرم‌افزاری حافظه همراه با هم معماری حافظه مجازی را تشکیل می‌دهند.



شکل ۲۰-۱۲ جدول صفحات به شکل حافظه تداعیگر

وقتی که برنامه‌ای شروع به اجرا می‌کند، یک یا چند صفحه به حافظه اصلی انتقال یافته و جدول صفحات مقدار می‌گیرد تا محل آنها را نشان دهد. برنامه از حافظه اصلی اجرا می‌شود تا اینکه اقدام به ارجاع به صفحه‌ای کند که هنوز در حافظه کمکی قرار دارد. این وضعیت فقدان صفحه^۱ نامیده می‌شود. هنگامی که فقدان صفحه رخ دهد اجرای برنامه جاری متوقف می‌شود تا اینکه صفحه لازم به حافظه اصلی منتقل گردد. چون بارکردن یک صفحه از حافظه کمکی به حافظه اصلی اصولاً یک عمل I/O است، سیستم عامل این کار را به پردازنده I/O واگذار می‌کند. در این حال، کنترل به برنامه‌بندی در حافظه که منتظر پردازش در CPU است منتقل می‌شود. پس از اختصاص بلاک حافظه و اتمام انتقال، برنامه قبلی می‌تواند عملکرد خود را از سر بگیرد.

هرگاه، در یک سیستم حافظه فقدان صفحه رخ دهد، بدان معنی است که صفحه ارجاع شده بوسیله CPU در حافظه اصلی نیست. سپس یک صفحه جدید از حافظه کمکی به حافظه اصلی انتقال می‌یابد. اگر حافظه اصلی پر باشد، لازم است یک صفحه از بلاک حافظه حذف گردد تا فضای کافی برای صفحه جدید ایجاد گردد. سیاست انتخاب صفحات برای حذف توسط الگوریتم جایگزینی بکار رفته معین می‌شود. هدف از سیاست جایگزینی تلاش برای حذف صفحه است که کمترین احتمال ارجاع مجدد در آینده نزدیک را دارد.

دو الگوریتم جایگزینی متداول تر مورد استفاده عبارتند از اولین ورودی - اولین خروجی (FIFO) و قدیمی‌ترین مورد استفاده (LRU). الگوریتم FIFO صفحه‌ای را که بیشتر از همه در حافظه وجود داشته است برای جایگزینی انتخاب می‌کند. هر بار که صفحه‌ای به حافظه بار شود، شماره شناسایی آن در یک پشته FIFO پوش می‌شود. هر وقت در حافظه هیچ بلاک خالی وجود نداشته باشد FIFO پر است. وقتی که قرار باشد صفحه جدیدی بار شود، قدیمی‌ترین صفحه وارده حذف می‌شود. تعیین صفحه‌ای که قرار است حذف شود آسان است زیرا شماره شناسایی آن در بالای پشته FIFO قرار دارد. سیاست جایگزینی FIFO این مزیت را دارد که از نظر پیاده‌سازی ساده است. عیب این روش این است که در شرایط معینی تعداد ورود و خروج صفحات از حافظه اصلی خیلی زیاد است.

سیاست LRU از نظر پیاده‌سازی مشکل‌تر است ولی بلحاظ مناسب‌تر بودن قدیمی‌ترین صفحه مورد استفاده بعنوان کاندید حذف، نسبت به قدیمی‌ترین صفحه بار شده FIFO، جذاب‌تر می‌باشد. الگوریتم LRU را می‌توان با اختصاص یک شمارنده برای هر صفحه‌ای که در حافظه اصلی قرار دارد پیاده‌سازی کرد. هر وقت صفحه‌ای مورد ارجاع قرار گیرد، شمارنده مربوطه اش صفر می‌شود در فواصل زمانی ثابت شمارنده‌های مربوط به تمام صفحات موجود در حافظه افزایش می‌یابند. قدیمی‌ترین صفحه مورد استفاده صفحه‌ای است که شمارنده‌اش بیشترین مقدار را دارد. این شمارنده‌ها اغلب ثبات‌های سن^۲ خوانده می‌شوند، زیرا شمارش، سن آنها را معین می‌کند، یعنی صفحه متناظر با آنها چه مدت قبل مورد ارجاع قرار گرفته است.

1- Page Fault

2- Aging Register

۱۲-۷ سخت افزار مدیریت حافظه

در یک محیط چند برنامه‌ای که برنامه‌های زیادی در حافظه مقیم است، گاهی لازم می‌شود تا برنامه‌ها و داده‌ها در حافظه به این سو و آن سو جابجا شوند، تا مقدار حافظه مورد استفاده یک برنامه معین تغییر نماید و از برنامه‌های دیگر توسط یک برنامه جلوگیری گردد. مقتضیات ناشی از چند برنامه‌گی در رابطه با حافظه کامپیوتر نیاز به یک سیستم مدیریت را بوجود آورده است. سیستم مدیریت حافظه مجموعه‌ای از رویه‌های سخت‌افزاری و نرم‌افزاری برای مدیریت انواع برنامه‌های مقیم در حافظه است. نرم‌افزار مدیریت حافظه بخشی از سیستم عامل کلی موجود در اکثر کامپیوترهاست. در اینجا ما به واحد سخت‌افزاری مربوط به سیستم مدیریت حافظه می‌پردازیم.

اجزای اصلی یک واحد مدیریت حافظه عبارتند از:

۱- امکاناتی برای جابجایی دینامیک در حافظه که ارجاع‌های منطقی حافظه را به آدرسهای فیزیکی حافظه تبدیل کند.

۲- امکان برای استفاده مشترک کاربران مختلف از برنامه‌های ذخیره شده در حافظه

۳- حفاظت اطلاعات در مقابل دستیابی غیرمجاز بین کاربران و جلوگیری از تغییر توابع سیستم عامل توسط کاربران

سخت افزار جابجایی دینامیک در حافظه یک روند تبدیل مشابه به سیستم صفحه‌بندی شرح داده شده در بخش ۶-۱۲ است. ثابت بودن اندازه صفحه مورد استفاده در سیستم حافظه مجازی موجب اشکالات معینی در رابطه با اندازه برنامه و ساختار منطقی برنامه‌ها می‌شود. بهتر است تا برنامه‌ها را به بخش‌های منطقی بنام قطعه^۱ تقسیم می‌کرد. یک قطعه مجموعه‌ای از دستورات منطقی مرتبط با هم یا عناصر تحت یک نام خاص است. قطعات ممکن است بوسیله برنامه نویس یا سیستم عامل تولید شود. مثال‌هایی از قطعات عبارتند از، زیرروال‌ها، آرایه‌ای از داده‌ها، جدول سمبل‌ها، یا برنامه یک کاربر.

استفاده مشترک از برنامه‌ها بخش لاینفکی از یک سیستم چند برنامه‌ای است. مثلاً چندین کاربر که مایلند برنامه‌های فوترن خود را ترجمه^۲ کنند، باید بتوانند از یک نسخه واحد از برنامه مترجم استفاده کنند نه اینکه هر کدام نسخه جداگانه‌ای از آن را در حافظه داشته باشند. در یک سیستم چند برنامه‌ای، همه کاربران از دیگر برنامه‌های سیستم موجود در حافظه نیز بطور مشترک استفاده می‌کنند و لازم نیست نسخه‌های متعددی از آنها ایجاد شود.

سومین مسئله در سیستم چند برنامه‌ای حفاظت از یک برنامه از تداخل دیگر برنامه‌هاست. مثالی از اینگونه دخالت‌های ناخواسته اینست که یک کاربر بطور غیرمجاز برنامه یک کاربر دیگر را کپی کند. جنبه دیگری از حفاظت، جلوگیری از انجام عملیات سیستم عامل و لذا ایجاد وقفه در ترتیب منظم عملیات

در یک سیستم کامپیوتری توسط کاربران عادی است. برنامه‌های معینی که جنبه سری دارند باید از دسترسی پرسنل غیرمجاز حفاظت شوند تا از سوء استفاده در فعالیت‌های محرمانه یک سازمان جلوگیری شود. آدرس تولید شده بوسیله یک برنامه قطعه‌ای آدرس منطقی خوانده می‌شود. این آدرس مشابه آدرس مجازی است جز اینکه فضای آدرس منطقی در عوض تقسیم به صفحات با طول ثابت، به قطعه‌هایی با طول متغیر تقسیم می‌گردد. در اینجا نیز همانند سیستم حافظه مجازی، ممکن است آدرس منطقی بزرگتر از آدرس حافظه فیزیکی باشد. هر قطعه علاوه بر اطلاعات مربوط به تغییر مکان، اطلاعات حفاظتی را نیز همراه خود دارد. برنامه‌های مشترک، در یک قطعه منحصر بفرد در فضای آدرس منطقی هر کاربر قرار داده می‌شود، تا بتوان از یک نسخه فیزیکی واحد بطور مشترک استفاده کرد. وظیفه واحد مدیریت حافظه تبدیل آدرس‌های منطقی به آدرس‌های فیزیکی مانند تبدیل در حافظه مجازی است.

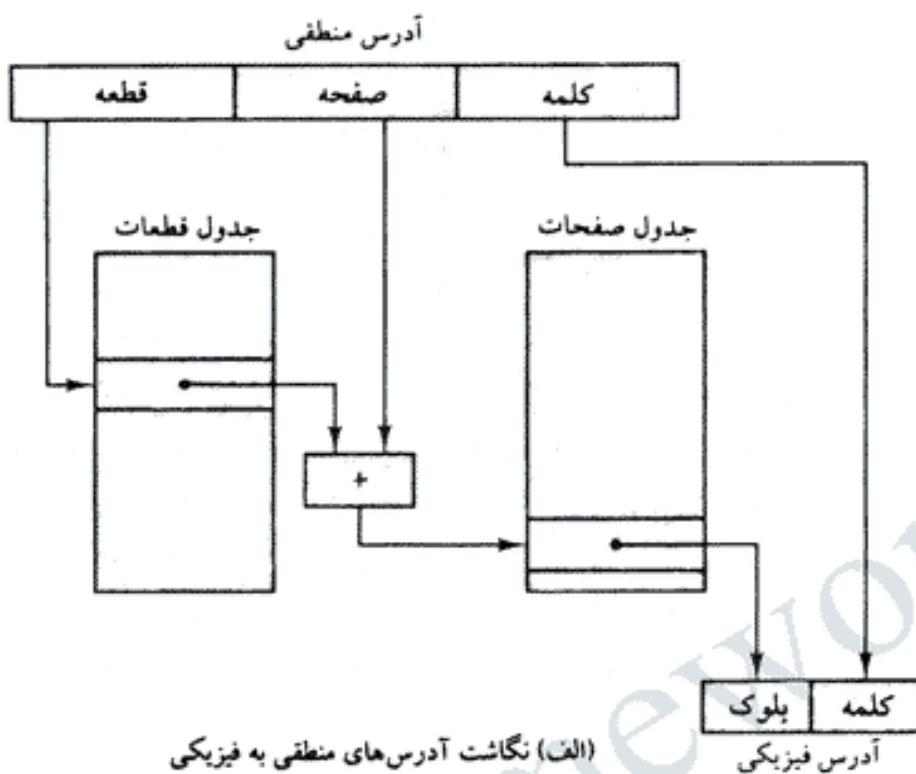
نگاشت قطعه - صفحه‌ای

قبلاً گفته شده که خاصیت فضای منطقی این است که از قطعات با طول متغیر استفاده می‌کند. طول هر قطعه می‌تواند براساس نیازهای برنامه جاری افزایش یا کاهش یابد. یکی از راه‌های مشخص کردن طول هر قطعه، متناظر ساختن آن با تعدادی از صفحات هم‌اندازه است. برای آگاهی از چگونگی انجام آن، آدرس منطقی شکل ۲۱-۱۲ را در نظر بگیرید. آدرس منطقی به سه میدان تقسیم شده است. میدان قطعه یک شماره قطعه را مشخص می‌کند. میدان صفحه، آن را در داخل قطعه و میدان کلمه نیز کلمه را در صفحه مشخص می‌کند. یک میدان صفحه با k بیت قادر است تا 2^k صفحه را معین کند. یک شماره قطعه می‌تواند، فقط یک صفحه یا تا 2^k صفحه متناظر باشد. بنابراین طول یک قطعه برطبق تعداد صفحاتی که به آن اختصاص داده می‌شود متغیر است.

تبدیل آدرس منطقی به آدرس فیزیکی بوسیله دو جدول انجام می‌شود، شکل ۲۱-۱۲ (الف). شماره قطعه آدرس منطقی، آدرس جدول قطعه را مشخص می‌کند. واردها در جدول قطعه، اشاره‌گر^۱ آدرس برای پایه^۲ (شروع) جدول صفحه می‌باشد. پایه جدول صفحه به شماره صفحه که در آدرس منطقی داده می‌شود اضافه می‌شود. مجموع دو کمیت اشاره‌گر آدرسی را برای یک وارده در جدول صفحه تولید می‌کند. مقداری که از جدول صفحات استخراج می‌شود شماره بلاک حافظه فیزیکی را مشخص می‌نماید. با کنار هم قراردادن میدان بلاک و میدان کلمه، آدرس تبدیل شده فیزیکی نهایی بدست می‌آید. دو جدول تبدیل ممکن است در دو حافظه جدا از هم کوچک یا در حافظه اصلی ذخیره شود. در هر صورت، یک ارجاع حافظه بوسیله CPU سه دستیابی به حافظه را نیاز دارد: یکی به جدول قطعات، یکی به جدول صفحه و بار سوم به حافظه اصلی. این امر سیستم را در مقایسه با سیستم معمولی که فقط یک ارجاع به حافظه را نیاز دارد بسیار کند می‌نماید. برای اجتناب از این کاهش سرعت، یک حافظه تداعیگر سریع، برای نگهداری آخرین واردهای ارجاع شده جدول بکار می‌رود. (این نوع حافظه گاهی با فرم دستی ترجمه یا TLB خوانده می‌شود). اولین باری که یک بلاک ارجاع داده شود، مقدار آن همراه با

1- Pointer

2- Base



ثبات آرگومان		
قطعه	صفحه	بلاک

(ب) بافر دم دستی ترجمه (TLB) در حافظه تداعیگر

شکل ۲۱-۱۲ نگاشت در واحد مدیریت حافظه قطعه-صفحه ای

شماره های قطعه و صفحه طبق شکل ۲۱-۱۲ (ب) وارد یک حافظه تداعیگر می شود. بنابراین روند تبدیل ابتدا با جستجوی تداعیگر با شماره های قطعه و صفحه آزمایش می شود. اگر نتیجه موفقیت آمیز باشد، تأخیر تبدیل تنها متعلق به حافظه تداعیگر است. اگر انطباقی رخ ندهد، جدول تبدیل شکل ۲۱-۱۲ (الف) که آهسته تر است بکار رفته و نتیجه تبدیل بداخل حافظه تداعیگر برای کاربرهای آینده انتقال می یابند.

مثال عددی

یک مثال عددی می تواند عملکرد یک واحد مدیریت حافظه را روشن تر کند. آدرس منطقی 20 بیتی شکل ۱۲-۲۲ (الف) را در نظر بگیرید. شماره قطعه 4 بیتی یکی از شانزده قطعه موجود را مشخص می کند. شماره صفحه 8 بیتی می تواند تا 256 صفحه را مشخص می نماید، و میدان کلمه هشت بیتی دلالت بر وجود صفحات با ظرفیت 256 کلمه دارد. این پیکربندی اجازه می دهد که هر قطعه هر تعداد صفحه تا 256 صفحه داشته باشد. کوچکترین قطعه ممکن یک صفحه یا 256 کلمه خواهد داشت. بزرگترین قطعه 256 صفحه و در کل $256 \times 256 = 64K$ کلمه خواهد داشت.

حافظه فیزیکی نشان داده شده در شکل ۱۲-۲۲ (ب) از 2^{20} کلمه 32 بیتی تشکیل شده است. آدرس 20 بیتی به دو میدان تقسیم شده است: یک شماره بلاک 12 بیتی و یک شماره کلمه 8 بیتی. بنابراین، آدرس فیزیکی به 4096 بلاک 256 بیتی تقسیم شده است. یک صفحه در آدرس منطقی دارای بلاک متناظری در حافظه فیزیکی می باشد. توجه کنید که آدرس های منطقی و فیزیکی 20 بیتی هستند. در غیاب یک واحد مدیریت حافظه، آدرس 20 بیتی از CPU می تواند مستقیماً برای دستیابی به حافظه بکار رود. برنامه ای را در نظر بگیرید که برای بار شدن در حافظه به 5 صفحه نیاز دارد. همانطور که در شکل ۱۲-۲۳ (الف) دیده می شود سیستم عامل می تواند به این برنامه قطعه 6 و صفحات 0 تا 4 را اختصاص دهد. محدوده کل آدرس منطقی برای برنامه از شانزده شانزدهی 60000 تا 604FF است. وقتی که برنامه در حافظه فیزیکی بار شود در میان پنج بلاک حافظه فیزیکی که توسط سیستم عامل خالی تشخیص داده شود توزیع می گردد. سپس واسطه های بین هر بلاک حافظه و شماره صفحه منطقی طبق شکل ۱۲-۲۳ (ب) وارد جدول می شوند. اطلاعات حاصل از این جدول مطابق شکل ۱۲-۲۴ (الف) وارد جداول قطعات و صفحات می گردد.

اکنون آدرس منطقی خاص نشان داده شده در شکل ۱۲-۲۴ را در نظر بگیرید. آدرس 20 بیتی

4	8	8
قطعه	صفحه	کلمه

(الف) قالب آدرس منطقی: 16 قطعه 256 صفحه ای که در هر صفحه 256 کلمه قرار دارد

حافظه فیزیکی $2^{20} \times 32$	12	8
	بلاک	کلمه

(ب) قالب آدرس فیزیکی: 4096 بلاک 256 کلمه ای که هر کلمه 32 بیت دارد

شکل ۱۲-۲۲ مثال از آدرس های منطقی و فیزیکی

بلاک			شماره صفحه		آدرس شانزده شانزدهمی
صفحه	قطعه		صفحه 0	صفحه 1	60000
00	6	012	صفحه 2	صفحه 3	60100
01	6	000	صفحه 4	صفحه 5	60200
02	6	019	صفحه 6	صفحه 7	60300
03	6	053	صفحه 8	صفحه 9	60400
04	6	A61	صفحه 10		604FF

(ب) بلاک های حافظه متناظر با قطعه - صفحه ها

(الف) تخصیص آدرس های حافظه منطقی

شکل ۲۳-۱۲ مثالی از تخصیص آدرسهای حافظه منطقی و فیزیکی

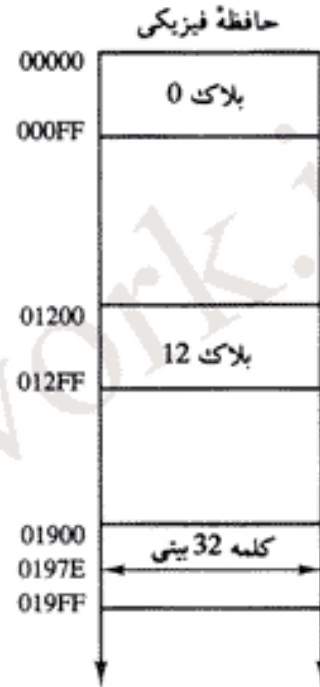
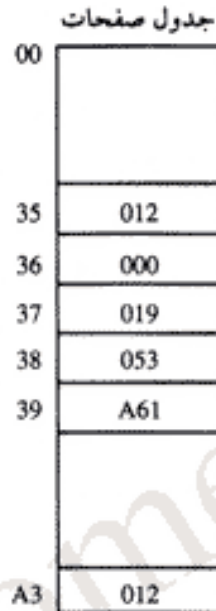
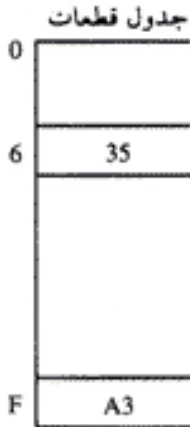
بصورت یک عدد پنج رقمی شانزده شانزدهمی نشان داده شده است. این آدرس به شماره کلمه 7E از صفحه 2 در قطعه 6 اشاره می کند. شروع قطعه 6 در جدول صفحه در آدرس 35 است. قطعه 6 پنج صفحه دارد، که در جدول صفحات در آدرسهای 35 تا 39 نشان داده شده اند. صفحه 2 از قطعه 6 در آدرس $35+2=37$ قرار دارد. از جدول صفحات دیده می شود که بلاک حافظه فیزیکی در 019 است. کلمه 7E در بلاک 19 آدرس فیزیکی 20 بیته 0197E را بدست می دهد. توجه کنید که صفحه 0 از قطعه 6 به بلاک 12 و صفحه 1 به بلاک 0 نگاشت می شود. حافظه تداعیگر شکل ۲۴-۱۲ (ب) نشان می دهد که صفحات 2 و 4 از قطعه 6 قبلاً مورد ارجاع قرار گرفته اند و بنابراین شماره بلاک متناظر آنها در حافظه تداعیگر ذخیره شده است. با توجه به مثال فوق آشکار است که سیستم مدیریت حافظه قادر است هر تعداد صفحات را به هر قطعه اختصاص دهد. هر صفحه منطقی را می توان به هر یک از بلاک های حافظه فیزیکی نگاشت کرد. صفحات را می توان بسته به نیاز فضای لازم برای حافظه، به بلاک های مختلفی در حافظه جابجا کرد. تنها بهنگام سازی مورد نیاز تغییر شماره بلاک در جدول صفحه است. صفحات می توانند بدون تأثیر بر یکدیگر کوچک یا بزرگ شوند. اگر قرار باشد برنامه مشترکی بوسیله چند کاربر مورد استفاده قرار گیرد قطعات مختلف می توانند از یک بلاک استفاده کنند. مثلاً به بلاک شماره 12 در حافظه فیزیکی آدرس منطقی دومی نیز می توان از F0000 تا F00FF اختصاص داد. این محدوده شماره قطعه 15 و شماره صفحه 0 را داراست، و همانطور که در شکل ۲۴-۱۲ (الف) دیده می شود به بلاک 12 نگاشت می شود.

حفاظت حافظه

حفاظت حافظه می تواند از طریق آدرس فیزیکی یا آدرس منطقی انجام شود. حفاظت حافظه از طریق آدرس فیزیکی بدین ترتیب صورت می گیرد که به هر بلاک در حافظه تعدادی بیت حفاظت که معین کننده نوع دستیابی به بلاک مربوطه اش می باشد اختصاص می یابد. هر بار صفحه ای از یک بلاک به بلاک دیگر جابجا شود لازم است بیت حفاظت بهنگام شود. اعمال حفاظت در فضای آدرس منطقی بهتر از فضای

آدرس منطقی (در مبنای 16)

6	02	7E
---	----	----



(الف) نگاشت با استفاده از جداول قطعات و صفحات

قطعه	صفحه	بلاک
6	02	019
6	04	A61

(ب) حافظه تداعیگر (TLB)

شکل ۱۲-۲۴ مثال نگاشت حافظه منطقی به فیزیکی (همه عددها در مبنای شانزده)

آدرس فیزیکی است. این کار را می توان با گنجانیدن اطلاعات حفاظتی در جدول قطعات یا ثبات قطعه سخت افزار مدیریت حافظه انجام داد.

محتوای هر وارده در جدول قطعه یا یک ثبات قطعه توصیفگر^۱ خواننده می شود. یک توصیفگر نوعی علاوه بر میدان آدرس پایه دارای یک یا دو میدان اضافی برای اهداف حفاظتی است. نوعی قالب برای توصیفگر قطعه در شکل ۲۵-۱۲ نشان داده شده است. آدرس پایه آدرس شروع جدول صفحه را در سازمان قطعه - صفحه و یا در آدرس پایه بلاک در سازمان ثبات قطعه فراهم می سازد. این آدرس در تبدیل یک آدرس منطقی به فیزیکی بسیار موثر است.

میدان طول^۲، اندازه قطعه را با مشخص کردن حداکثر تعداد صفحات قابل اختصاص به قطعه معین می کند. اگر شماره صفحه خارج از مرز قطعه باشد. تخطی از اندازه صورت گرفته است. به این ترتیب یک برنامه معین و داده های آن نمی توانند به حافظه ای که سیستم عامل به آن اختصاص نداده است دست یابد.

میدان حفاظت در توصیفگر قطعه، حقوق دستیابی مجاز به قطعه مورد نظر را مشخص می کند. در سازمان قطعه - صفحه، هر یک از وارده ها ممکن است میدان حفاظت خود را برای توصیف حقوق دستیابی هر صفحه داشته باشد. اطلاعات حفاظت توسط برنامه کنترل اصلی سیستم عامل در توصیفگر قرار می گیرد. بعضی از حقوق دستیابی مورد نظر مقیم در حافظه برای حفاظت برنامه ها عبارتند از:

- ۱- حقوق کامل خواندن و نوشتن
- ۲- فقط خواندن (حفاظت در برابر نوشتن)
- ۳- فقط اجرا (حفاظت از برنامه ها)
- ۴- فقط سیستم (حفاظت از سیستم عامل)

حقوق کامل خواندن و نوشتن هنگامی که یک برنامه در حال اجرای دستورالعمل های خودش باشد به آن داده می شود. حفاظت در برابر نوشتن برای استفاده مشترک از برنامه های سیستم از قبیل برنامه های ابزاری^۳ و روال های کتابخانه ای^۴ دیگر مفید است. این برنامه های سیستم در ناحیه ای از حافظه ذخیره می شوند تا بوسیله کار بران متعدد مورد استفاده مشترک قرار گیرند. این برنامه ها را سایر برنامه ها می توانند بخوانند، اما نوشتن در آنها مجاز نیست. این خصوصیت آنها را از هر گونه تغییر توسط برنامه های دیگر حفاظت می کند.

شرط فقط - اجرا برنامه ها را از کپی شدن محافظت می نماید. این شرط قطعه را محدود می کند تا فقط در حین فاز برداشت مورد ارجاع قرار گیرد و نه در حین فاز اجرا. بنابراین به کاربران اجازه می دهد

حفاظت	طول	آدرس مبنا
-------	-----	-----------

شکل ۲۵-۱۲ قالب نوعی توصیفگر قطعه

1- Descriptor

2- Access Leights

3- Utilitys Programs

4- Library Routine

تادستورالعمل های برنامه در قطعه را اجرا کند اما از خواندن دستورالعمل ها بعنوان داده به قصد کپی کردن آنها ممانعت می کند.

در هر لحظه از زمان بخش هایی از سیستم عامل در حافظه واقع است. این برنامه های سیستم با قرار نگرفتن در دسترس کاربران غیرمجاز محافظت شود. شرط حفاظت سیستم عامل برای تمام برنامه های سیستم عامل در توصیفگر قرار می گیرد تا مانع دستیابی کاربران موقت به قطعات سیستم عامل گردد.

مسائل

۱۲-۱ الف) چند تراشه RAM، 128×8 برای تهیه یک حافظه با ظرفیت 2048 لازم است؟
ب) چند خط گذرگاه آدرس باید بکاربرد تا 2048 بایت از حافظه دستیابی شود؟ چند خط از این خطوط برای همه تراشه ها مشترک است.

پ) چند خط برای انتخاب تراشه باید دیکدر شود؟ اندازه دیکدر را مشخص کنید.

۱۲-۲ کامپیوتری از یک RAM با ظرفیت 1024×1 استفاده می کند.

الف) چند تراشه لازم است، و چگونه خطوط آنها باید متصل شود تا 1024 بایت حافظه حاصل شود؟

ب) چند تراشه برای تهیه 16K بایت حافظه لازم است؟ شفاهاً توضیح دهید که چگونه تراشه ها باید به گذرگاه حافظه وصل شوند.

۱۲-۳ یک تراشه ROM با 1024×8 بیت دارای چهار ورودی انتخاب بوده و با منبع تغذیه 5 ولتی کار می کند. برای بسته IC چند پایه لازم است. بلاک دیاگرام را رسم کرده و پایانه های ورودی و خروجی را در ROM نام گذاری کنید.

۱۲-۴ سیستم حافظه شکل ۱۲-۴ را به 4096 بایت RAM و 4096 بایت ROM گسترش دهید. نقشه حافظه - آدرس را رسم کنید. مشخص کنید که اندازه دیکدر مورد استفاده باید چقدر باشد.

۱۲-۵ کامپیوتری از تراشه های RAM با 256×8 و ROM با 1024×8 استفاده می کند. سیستم کامپیوتری به 2K بایت از RAM، 4K بایت از ROM، و چهار واحد واسطه، که هر یک چهار ثبات دارد، نیاز دارد. از پیکربندی I/O نگاشت حافظه استفاده شده است. به دو بیت با ارزشتر گذرگاه آدرس 00 برای RAM، 01 برای ROM و 10 برای ثبات های واسطه اختصاص داده می شود.

الف) چند تراشه RAM و ROM لازم است

ب) نقشه حافظه - آدرس را برای سیستم رسم کنید.

پ) محدوده آدرس را برحسب شانزده شانزدهی برای RAM، ROM و مدار واسطه معین کنید.

۱۲-۶ یک کامپیوتر دارای گذرگاه آدرس 16 بیتی است. 15 خط اول آدرس ها برای انتخاب یک بانک 32K بایتی حافظه استفاده شده است. بیت با ارزشتر آدرس برای انتخاب یک ثبات که محتوای گذرگاه داده

- را دریافت می کند بکار رفته است. توضیح دهید چگونه می توان از این پیکربندی استفاده کرده و ظرفیت حافظه را به هشت بانک 32K بایتی افزایش داد تا یک بانک 256K بایتی حاصل شود.
- ۱۲-۷ یک سیستم دیسک مغناطیسی دارای پارامترهای زیر است،
- $$T_s = \text{زمان متوسط لازم برای قرار گرفتن هد مغناطیسی روی یک شیار}$$
- $$R = \text{سرعت چرخش دیسک برحسب دور بر ثانیه}$$
- $$N_t = \text{تعداد بیت ها در شیار}$$
- $$N_s = \text{تعداد بیت ها در قطاع}$$
- زمان متوسط T_s که برای خواندن یک قطاع لازم است چقدر می باشد.
- ۱۲-۸ سرعت انتقال یک نوار مغناطیسی هشت شیاره که سرعتش 120 اینچ در ثانیه و تراکم آن 1600 بیت در اینچ است چقدر باشد.
- ۱۲-۹ تابع متمم منطق انطباق یک کلمه را در حافظه تداعیگر بدست آورید. به بیان دیگر، نشان دهید که M'_i مجموع توابع OR انحصاری است. دیاگرام منطقی را برای M'_i رسم نموده و با یک معکوس کننده آنرا پایان دهید تا M_i بدست آید.
- ۱۲-۱۰ تابع بول را برای منطق انطباق یک کلمه را با یک حافظه تداعیگر با در نظر گرفتن یک بیت نشانه که مشخص کند کلمه فعال است یا غیرفعال، بدست آورید.
- ۱۲-۱۱ چه مدار منطقی اضافی لازم است تا نتیجه عدم تطابق را برای یک کلمه در حافظه تداعیگر وقتی که همه بیت های کلید صفر است مشخص کند.
- ۱۲-۱۲ الف) دیاگرام منطقی تمام سلول های یک کلمه در حافظه تداعیگر را رسم کنید. منطق خواندن و نوشتن شکل ۱۲-۸ و منطق انطباق شکل ۱۲-۹ را نیز اضافه کنید.
- ب) دیاگرام منطقی تمام سلول های عمودی یک ستون (ستون l) را در حافظه تداعیگر رسم کنید. یک خط خروجی مشترک را برای همه بیت هایی که در یک ستون قرار دارند اضافه کنید.
- ج) با استفاده از دیاگرام های (الف) و (ب) نشان دهید که اگر خروجی M_i به خط خواندن همان کلمه وصل شود، کلمه منطبق به خارج خوانده خواهد شد، مشروط بر اینکه فقط یک کلمه با آرگومان پوشانده شده مطابقت داشته باشد.
- ۱۲-۱۳ با استفاده از بلاک دیاگرام شرح دهید چگونه می توان کلمات منطبق متعدد را از حافظه تداعیگر به خارج خواند.
- ۱۲-۱۴ منطق یک سلول و نیز یک کلمه کامل را برای یک حافظه تداعیگر، همراه با نشانگری برای مواقعی که آرگومان پوشش نیافته بزرگتر از (اما نه مساوی با) کلمه موجود در حافظه تداعیگر باشد، بدست آورید.
- ۱۲-۱۵ یک حافظه کش تداعیگر دوتایی از بلاک های چهار کلمه ای استفاده می کند. حافظه کش می تواند مجموعاً 2048 کلمه از حافظه اصلی را در خود جای دهد. اندازه حافظه اصلی 128×32 است.
- الف) همه اطلاعات لازم برای ساختن این حافظه کش را فرموله کنید.

ب) اندازه حافظه کش چقدر است.

۱۶-۱۲ زمان دستیابی یک حافظه کش 100ns و حافظه اصلی 1000ns است. پیش بینی می شود 80 درصد تقاضاهای حافظه برای خواندن و 20 درصد بقیه برای نوشتن می باشند. نسبت برد برای دستیابی های خواندن فقط 0.9 است. رویه کامل نویسی استفاده می شود.

الف) زمان دستیابی متوسط سیستم فقط با در نظر گرفتن سیکل خواندن چقدر است.

ب) زمان دستیابی متوسط سیستم برای تقاضای خواندن و نوشتن چقدر است.

ج) نسبت برد با در نظر گرفتن سیکل های نوشتن چقدر است.

۱۷-۱۲ یک مجموعه حافظه کش تداعیگر چهارتایی در هر مجموعه چهار کلمه دارد. یک رویه جایگزینی

مبتنی بر الگوریتم بر قدیمی ترین مورد استفاده (LRU) باشمارنده های دویینی متناظر با هر یک از

کلمات مجموعه پیاده سازی می شود. بنابراین مقداری در محدوده 0 تا 3 برای هر کلمه ثبت می شود.

وقتی بردی رخ دهد، شمارنده متناظر با کلمه مورد ارجاع 0 می گردد. شمارنده هایی که مقادیر قبلی آنها

کمتر از مقدار شمارنده مورد ارجاع است 1 واحد افزایش می یابند و بقیه تغییر نمی کنند. اگر باخت

رخ دهد، کلمه ای که مقدار شمارنده آن 3 است حذف می شود، کلمه جدید در محل آن قرار داده

می شود، و شمارنده آن 0 می گردد. سه شمارنده دیگر 1 واحد افزایش می یابند. نشان دهید که این

رویه برای دنباله ارجاعات زیر درست عمل می کنند: A, B, C, D, E, B, D, C, A, D, E, C, A, E, C, E, B, A

(A, B, C و D بعنوان مقادیر اولیه شروع کنید که در آنها A قدیمی ترین مورد استفاده است).

۱۸-۱۲ یک کامپیوتر دیجیتال دارای واحد حافظه 64Kx16 و یک حافظه کش 1K کلمه ای است. کش از

نگاشت مستقیم استفاده می کند و سایز بلاک چهار کلمه است.

الف) در میدان های نشانه، شاخص (اندیس)، بلاک، و کلمه قالب آدرس چند بیت وجود دارد؟

ب) در هر کلمه کش چند بیت وجود دارد، و آنها چگونه به توابع مختلف کاری تقسیم شده اند.

ج) حافظه کش چند بلاک را در خود جای می دهد.

۱۹-۱۲ یک فضای آدرس با 24 بیت و فضای حافظه متناظر آن با 16 بیت مشخص می شود

الف) چند کلمه در فضای آدرس وجود دارد. ب) چند کلمه در فضای حافظه وجود دارد.

ج) اگر یک صفحه از 2K کلمه ساخته شده باشد، چند صفحه و بلاک در سیستم وجود دارد.

۲۰-۱۲ یک حافظه مجازی دارای یک صفحه 1K کلمه ای است. در این حافظه هشت صفحه و چهار

بلاک وجود دارد. جدول صفحات حافظه تداعیگر حاوی وارده های زیر است:

بلاک	صفحه
3	0
1	1
2	4
0	6

لیستی از آدرس های مجازی (به دهدهی) تهیه کنید که در صورت استفاده CPU از آن موجب فقدان صفحه شوند.

۱۲-۲۱ یک سیستم حافظه مجازی دارای فضای آدرس 8K کلمه یک فضای حافظه 4K کلمه، و صفحات و بلاک های 1K کلمه است (شکل ۱۸-۱۲). تغییرات ارجاع صفحه زیر در طول یک فاصله زمانی معین رخ می دهد. (فقط تغییرات صفحات نوشته شده است. اگر یک صفحه مجدداً مورد ارجاع قرار گرفته باشد دوبار نوشته نشده است).

4 2 0 1 2 6 1 4 0 1 0 2 3 5 7

چهار صفحه مقیم در حافظه اصلی را پس از هر تغییر ارجاع صفحات تعیین کنید بشرطی که الگوریتم جایگزینی مورد استفاده (الف) FIFO، (ب) LRU باشد.

۱۲-۲۲ دو آدرس منطقی را از شکل ۱۲-۲۲ (الف) که سبب دستیابی به حافظه فیزیکی در آدرس فیزیکی شانزده شانزدهمی 012AF می گردد معین کنید.

۱۲-۲۳ فضای آدرس منطقی در یک سیستم کامپیوتری متشکل از 128 قطعه است. هر قطعه می تواند تا 32 صفحه 4K کلمه ای داشته باشد. حافظه فیزیکی از 4K بلاک 4K کلمه در هر کدام است. قالب آدرس های فیزیکی و منطقی را تنظیم کنید.

۱۲-۲۴ معادل دودویی آدرس منطقی تنظیم شده در مسئله ۱۲-۲۳ را برای قطعه 36 و کلمه 2000 در صفحه 15 بنویسید.

۱۳

چند پردازنده‌ها

- ۱۳-۱ مشخصات چند پردازنده‌ها
- ۱۳-۲ ساختارهای اتصالات متقابل
- ۱۳-۳ داوری بین پردازنده‌ها
- ۱۳-۴ ارتباط و همگامی بین پردازنده‌ها
- ۱۳-۵ همبستگی حافظه کش

۱۳-۱ مشخصات چند پردازنده‌ها

یک سیستم چند پردازنده‌ای از اتصال دو یا چند CPU به همراه حافظه و تجهیزات ورودی - خروجی تشکیل می‌شود. اصطلاح "پردازنده" در چند پردازنده^۱ می‌تواند بمعنی یک واحد پردازش و یا یک پردازنده ورودی - خروجی (IOP) باشد. با این وجود، سیستمی با یک CPU و چند IOP در تعریف سیستم چند پردازنده نمی‌گنجد مگر اینکه IOP دارای تسهیلات محاسباتی قابل مقایسه با CPU باشد. با توجه به متداولترین تعریف، مفهوم یک سیستم چند پردازنده دلالت بر وجود چند CPU دارد، گرچه معمولاً یک یا چند IOP هم ممکن است موجود باشد. همان طور که در بخش ۱-۹ ذکر شد، چند پردازنده‌ها بعنوان سیستم‌هایی با رشته‌های چند دستوری^۲ و رشته‌های چند داده‌ای^۳ دسته‌بندی می‌شوند.

بین سیستم‌های چند پردازنده و چند کامپیوتری شباهت‌هایی وجود دارد زیرا هر دو آنها عملیات همزمان را پشتیبانی می‌کنند. معهذاً اختلاف مهمی بین سیستمی با چند کامپیوتر و سیستمی با چند پردازنده وجود دارد. کامپیوترها با اتصالات بین خود بهم متصل شده و یک شبکه کامپیوتری را تشکیل می‌دهند. شبکه از چندین کامپیوتر مستقل تشکیل می‌شود که می‌توانند با هم ارتباط داشته یا نداشته باشند. یک سیستم چند پردازنده توسط یک سیستم عامل کنترل می‌شود تا اتصالات بین پردازشگرها برقرار شده و در نتیجه تمام اجزاء سیستم در حل یک مسئله همکاری نمایند.

گرچه بعضی کامپیوترهای بزرگ دو یا چند CPU را در سیستم کلی خود دارا هستند، انگیزه عمده

1- Multiprocessor

2- Multiple Instruction

3- Multiple Data

برای سیستم های چند پردازنده، ظهور ریزپردازنده بوده است. این واقعیت که ریز پردازنده فضای فیزیکی کوچکی را اشغال کرده و بسیار ارزان است، اتصال تعداد زیادی از ریز پردازنده ها و تشکیل یک سیستم مرکب را امکان پذیر می سازد. تکنولوژی مدارهای مجتمع با مقیاس بزرگ قیمت اجزاء کامپیوتر را به سطح پایینی برده است که مفهوم استفاده از چند پردازنده برای دستیابی به نیازمندی های عملکردی یک سیستم، تبدیل به امکان طراحی جذابی شده است.

چند پردازشی قابلیت اطمینان سیستم را طوری بالا می برد که اشکال یا خطایی در یک بخش اثر محدودی بر بقیه سیستم دارد. اگر اشکالی موجب از کار افتادگی یک پردازنده شود، می توان پردازنده دیگری را برای انجام اعمال پردازنده غیرفعال اختصاص داد. سیستم در کل، می تواند بطور صحیح، با احتمالاً مقداری افت بازدهی، بکار خود ادامه دهد.

مزیت حاصل از یک سازمان چند پردازنده، ارتقای عملکرد سیستم است. عملکرد بالای سیستم ناشی از حقیقت است که محاسبات می تواند بطور موازی به یکی از دو روش زیر انجام شود.

- ۱- چند عمل^۱ مستقل می توانند بطور موازی انجام شوند.
- ۲- یک کار مستقل می تواند به چند کار کوچکتر^۲ موازی تقسیم شود.

یک کار^۳ کلی را می توان به تعدادی کار کوچکتر (یا تکلیف) تقسیم کرد بطوری که هر پردازنده بتواند مستقلاً یکی از آنها را انجام دهد. کارهای کوچک سیستم را می توان به پردازنده های کاربرد ویژه ای^۴ محول کرد که طراحی آنها برای اجرای انواع معینی از پردازش بهینه سازی شده است. به عنوان مثال می توان سیستمی را در نظر گرفت که در آن یک پردازنده محاسبات مربوط به کنترل یک فرآیند صنعتی را انجام می دهد در حالی که پردازنده دیگری پارامترهای مختلف از قبیل دما، سرعت جریان سیال را نظارت و کنترل می کند. مثال دیگر کامپیوتری است که در آن یک پردازنده محاسبات ریاضی ممیز - شناور سرعت های بالا را انجام می دهد و کامپیوتر دیگری بر کارهای کوچک یک داده پردازشی نظارت می کند. چند پردازنده ای می تواند با تفکیک یک برنامه به چند کار کوچک قابل اجرا بصورت موازی، عملکرد را بهبود ببخشد. این امر به یکی از دو روش زیر امکان پذیر است. کاربر می تواند بطور صحیح تصمیم بگیرد تا کارهای کوچک معینی از برنامه بطور موازی اجرا شوند. این کار باید قبل از بار کردن برنامه با مشخص کردن قطعه های قابل اجرای موازی انجام شود. اکثر سازندگان چند پردازنده ها سیستم عاملی با ساخت هایی در زبانهای برنامه نویسی که برای پردازش موازی مناسبند را در اختیار می گذارند. روش کارآتر دیگر عبارتست از تهیه مترجمی^۵ با نرم افزار چند پردازنده ای که می تواند بطور اتوماتیک امکان پردازش موازی را در برنامه کاربر کشف کند. مترجم برنامه را برای یافتن وابستگی داده ها و ارسی می کند. اگر بخشی از برنامه به داده های تولید شده در بخشی دیگر وابستگی داشته باشد، بخش

1- Job

2- Task

3- Function

4- Special Purpose CPU

5- Compiler

تولیدکننده داده باید زودتر اجرا شود. مع الوصف دو بخش از برنامه که از داده‌های تولید شده یکدیگر استفاده نمی‌کنند می‌توانند همزمان اجرا گردند. مترجم موازی کننده تمام برنامه را به منظور یافتن هر گونه وابستگی‌های احتمالی بین داده‌ها چک می‌کند. سپس بخش‌هایی که هیچگونه وابستگی نداشته باشند برای اجرای همزمان روی پردازنده‌های مختلف بررسی می‌شوند.

چند پردازنده‌ها با توجه به نحوه سازماندهی حافظه خود طبقه‌بندی می‌شوند. سیستم چند پردازنده‌ای با حافظه اشتراکی^۱ در طبقه چند پردازنده‌های دارای حافظه اشتراکی یا چند پردازنده‌های با کوپل محکم^۲ قرار می‌گیرد. این امر مانع نمی‌شود تا هر پردازنده حافظه محلی خاص برای خود داشته باشد. در واقع، اغلب چند پردازنده‌های با کوپل محکم یک حافظه کش را برای هر CPU در اختیار می‌گذارند. بعلاوه، یک حافظه مشترک کلی وجود دارد که همه CPUها به آن دسترسی دارند. بنابراین اطلاعات می‌تواند با قرار گرفتن در حافظه مشترک کلی بین CPUها مشترک باشد.

مدل دیگری برای چند پردازنده‌ها، سیستم دارای حافظه توزیع شده^۳ یا کوپل سست^۴ است. در یک سیستم با کوپل سست، هر عنصر پردازنده دارای حافظه محلی مربوط به خود است. پردازنده‌ها بوسیله یک طرح سوئیچینگ که برای راه‌دهی اطلاعات از یک پردازنده به پردازنده دیگر از طریق یک طرح پیام‌رسانی طراحی شده‌اند، بهم متصل می‌شوند. پردازنده‌ها برنامه و داده‌ها را به شکل بسته به پردازنده‌های دیگر می‌فرستند. هر بسته از یک آدرس، محتوای داده و برخی کدهای کشف خطا تشکیل شده است. بسته‌ها بر اساس سیستم ارتباطی بکاررفته، به آدرس یک پردازنده خاص ارسال می‌شوند یا بوسیله اولین پردازنده آزاد دریافت می‌گردند. سیستم‌های با کوپل سست، وقتی عمل و عکس‌العمل متقابل بین کارهای کوچک حداقل است کاراثر می‌باشد، در صورتی که سیستم‌های با کوپل محکم می‌توانند درجه بالاتری از اینگونه اعمال و عکس‌العمل‌ها را بین تکالیف تحمل کنند.

۱۳-۲ ساختارهای اتصالات متقابل

اجزایی که یک سیستم چند پردازنده‌ای را تشکیل می‌دهند عبارتند از CPUها، IOPها که به وسایل ورودی-خروجی متصلند، و یک واحد حافظه که ممکن است به چند ماژول جداگانه تقسیم شده باشد. اتصالات فی مابین این اجزاء بسته به تعداد مسیرهای انتقال موجود بین پردازنده‌ها و حافظه در یک سیستم حافظه مشترک یا میان عناصر پردازش در سیستم‌های با کوپل سست، می‌تواند آرایش‌های مختلف فیزیکی داشته باشد. برای ایجاد شبکه فی مابین آنها چند شکل فیزیکی وجود دارد. بعضی از این انواع در این بخش ارائه شده است.

۱- گذرگاه اشتراکی زمان مشترک

1- Shared Memory

2- Tightly Coupled Multiprocessor

3- Distributed - Memory System

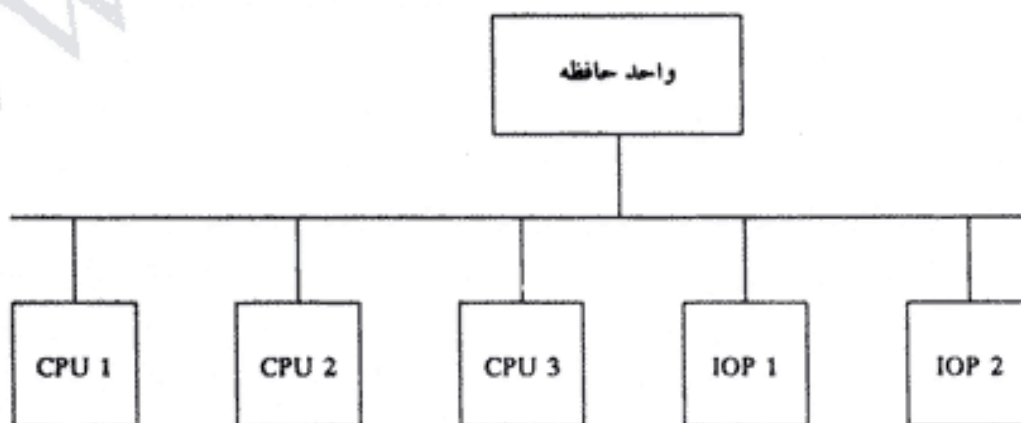
4- Loosely Coupled System

- ۲- حافظه چند پورته^۱
- ۳- سویچ های تقاطعی
- ۴- شبکه سوئیچینگ چند طبقه
- ۵- سیستم فوق مکعبی

گذرگاه مشترک با تسهیم زمانی

یک چند پردازنده گذرگاه مشترک متشکل است از تعدادی پردازنده که از طریق مسیر های مشترک به یک واحد حافظه متصل شده اند. یک گذرگاه مشترک با تسهیم زمانی^۲ برای پنج پردازنده در شکل ۱-۱۳ نشان داده است. در هر لحظه از زمان تنها یک پردازنده می تواند با حافظه یا پردازنده دیگر ارتباط برقرار کند. عملیات انتقال توسط پردازنده ای انجام می شود که از طریق یک مسیر مشترک به یک واحد حافظه متصل است. هر پردازنده دیگری که بخواهد یک انتقال را آغاز کند باید ابتدا وضعیت گذرگاه را از نظر آزاد بودن آزمایش کند، و پس از در اختیار بودن گذرگاه پردازنده می تواند واحد مقصد را برای آغاز انتقال آدرس دهی نماید. برای آگاه کردن واحد مقصد از نوع عمل، دستوری به آن صادر می شود. واحد گیرنده آدرس آن را در گذرگاه تشخیص داده و به سیگنال های کنترل که از فرستنده پس از آغاز انتقال می رسند پاسخ می دهد. سیستم بعلت اشتراک گذرگاه ممکن است دچار مشکلاتی در انتقال گردد زیرا یک گذرگاه بوسیله همه پردازنده ها به اشتراک گرفته شده است. این مشکلات باید با استفاده از یک کنترل کننده گذرگاه که واحدهای درخواست کننده را اولویت بندی می کند حل گردد.

یک سیستمی با یک گذرگاه مشترک به یک انتقال در هر لحظه محدود است. این بدان معنی است که وقتی یک پردازنده با حافظه تبادل اطلاعات می کند، کلیه پردازنده های دیگر یا مشغول کار داخلی می باشند یا باید بانتظار استفاده از گذرگاه بیکار بمانند. در نتیجه، سرعت انتقال کل در سیستم توسط سرعت یک گذرگاه

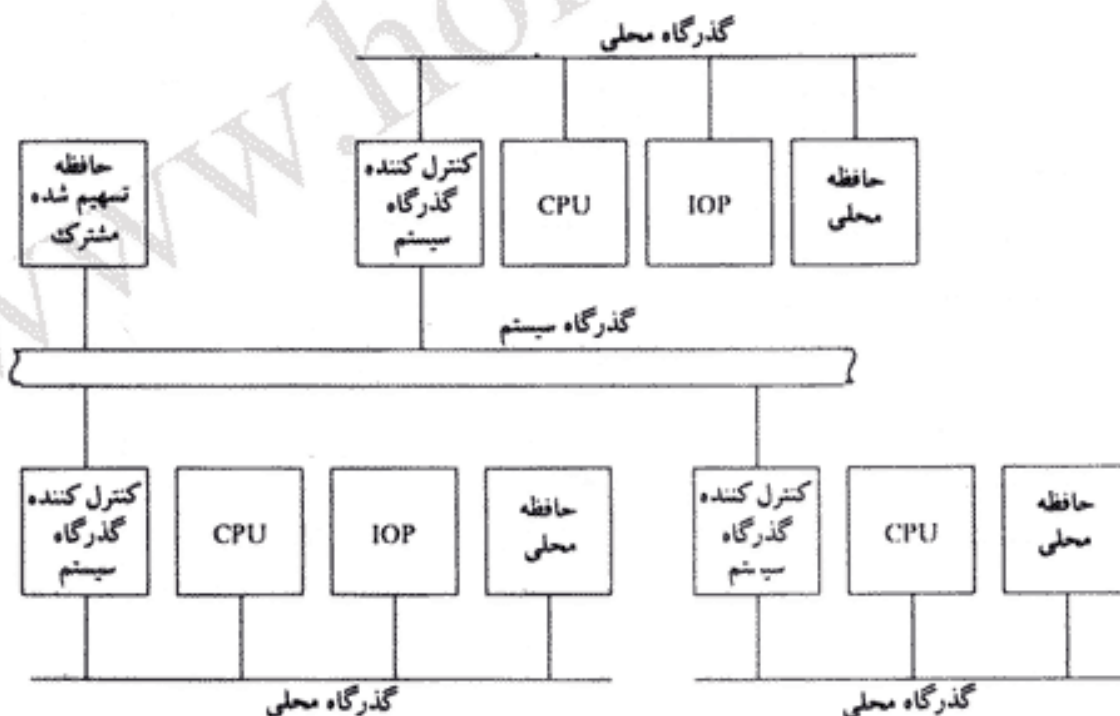


شکل ۱-۱۳ سازمان گذرگاه اشتراکی زمان مشترک

محدود می شود. پردازنده ها در سیستم، اغلب از طریق بکارگیری دو یا چند گذرگاه برای انتقال های همزمان متعدد مشغول نگهداشته می شوند. با این وجود این روش هزینه و پیچیدگی سیستم را افزایش می دهد. پیاده سازی مقرون به صرفه تری برای یک ساختار دو گذرگاهی در شکل ۲-۱۳ نشان داده شده است. در این شکل تعداد گذرگاه های محلی وجود دارد که هر کدام به حافظه محلی خود و یک یا دو پردازنده وصل شده اند. هر گذرگاه محلی ممکن است به یک CPU، یا IOP، یا هر ترکیبی از پردازنده ها متصل گردد. یک سیستم کنترل کننده گذرگاه، هر گذرگاه محلی را به سیستم گذرگاه مشترک ارتباط می دهد. وسایل I/O متصل به IOP محلی و حافظه محلی در دسترس پردازنده های محلی هستند. حافظه متصل به سیستم گذرگاه مشترک بصورت اشتراکی در اختیار همه پردازنده ها قرار دارد. اگر یک IOP مستقیماً به گذرگاه سیستم وصل شده باشد، وسایل I/O متصل به آن قابل دسترسی برای پردازنده ها خواهند بود. در هر لحظه از زمان تنها یک پردازنده می تواند با حافظه اشتراکی یا هر منبع مشترک دیگر از طریق گذرگاه سیستم ارتباط برقرار نماید. سایر پردازنده ها با حافظه محلی خود و نیز وسایل I/O مشغول خواهند بود. بخشی از حافظه محلی ممکن است بصورت حافظه کش متصل به CPU طراحی گردد (بخش ۶-۱۲). بدین ترتیب زمان متوسط دستیابی حافظه محلی به سمت سیکل زمانی CPU متصل به آن سوق داده می شود.

حافظه چند پورتی

یک سیستم حافظه چند پورتی از گذرگاه های تفکیک شده بین هر ماژول حافظه^۱ و هر CPU استفاده

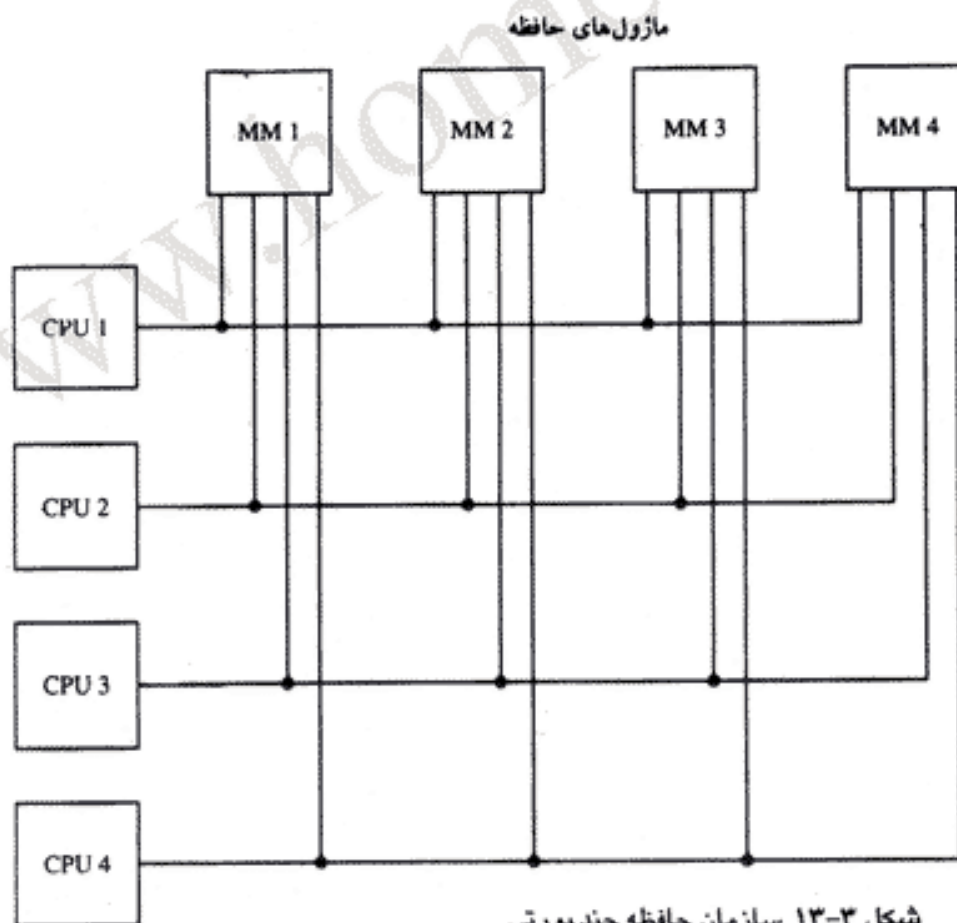


شکل ۲-۱۳ ساختار گذرگاه سیستم برای چند پردازنده ها

1- Memory Modules

می‌کند. این آرایش برای چهار CPU و چهار ماژول حافظه (MMS) در شکل ۳-۱۳ نشان داده شده است. گذرگاه هر پردازنده به هر ماژول حافظه متصل شده است. یک گذرگاه پردازنده از خطوط لازم آدرس، داده و کنترل تشکیل شده است تا با حافظه ارتباط برقرار کند. در این آرایش گوئیم ماژول حافظه چهار پورت دارد و هر پورت یکی از گذرگاه‌ها را سرویس می‌دهد. ماژول حافظه باید دارای مدارهای منطقی کنترل داخلی برای تعیین پورتی که باید به حافظه در هر لحظه دستیابی نماید باشد. پیچیدگی دستیابی به حافظه با اختصاص اولویت‌های ثابت به هر پورت حافظه حل می‌شود. اولویت در دستیابی به حافظه متناظر با هر پردازنده را می‌توان با استفاده از انتخاب محل فیزیکی پورتی که گذرگاه آن در هر ماژول اشغال می‌کند تعیین کرد. با این ترتیب CPU1 بر CPU2 اولویت خواهد داشت، و CPU2 بر CPU3 اولویت دارد، و نهایتاً CPU4 پائین‌ترین اولویت را دارد.

مزیت سازمان حافظه چند پورته عبارتست از سرعت انتقال بالا که بعلمت مسیرهای چندگانه بین پردازنده و حافظه امکان پذیر است. عیب این سازمان این است که نیاز به مدار منطقی کنترل حافظه گرانقیمتی دارد و مقدار زیادی کابل و رابط لازم دارد. در نتیجه ساختار اتصالات فی‌مابین معمولاً برای سیستم‌هایی مناسب است که تعداد کمی پردازنده داشته باشد.



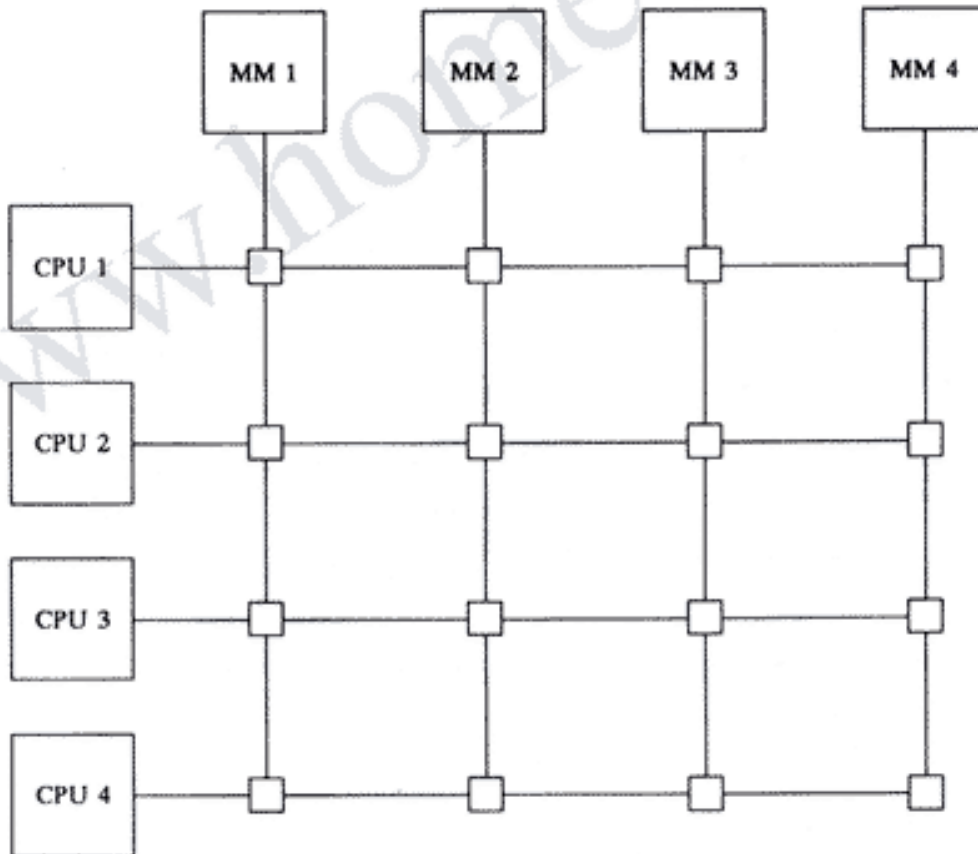
شکل ۳-۱۳ سازمان حافظه چند پورته

سوئیچ تقاطعی

سازمان یک سوئیچ تقاطعی از تعدادی نقطه تقاطعی تشکیل می شود که در محل تقاطع گذرگاه های بین پردازنده ها و مسیرهای ماژول حافظه قرار داده می شود. شکل ۴-۱۳ اتصالات بین چهار CPU و چهار ماژول حافظه را با سوئیچ تقاطعی نشان می دهد. مربع کوچک در هر نقطه تقاطع یک کلید است که مسیر را از پردازنده به ماژول حافظه معین می کند. هر نقطه سوئیچ دارای مدار منطقی کنترل برای ایجاد مسیر بین پردازنده و حافظه است. این سوئیچ آدرس روی گذرگاه را برای تعیین آدرس یک ماژول خاص بررسی می کند. همچنین مشکل درخواست های متعدد برای دستیابی به یک ماژول آدرس دهی شده را بر مبنای اولویت قبلی حل می کند.

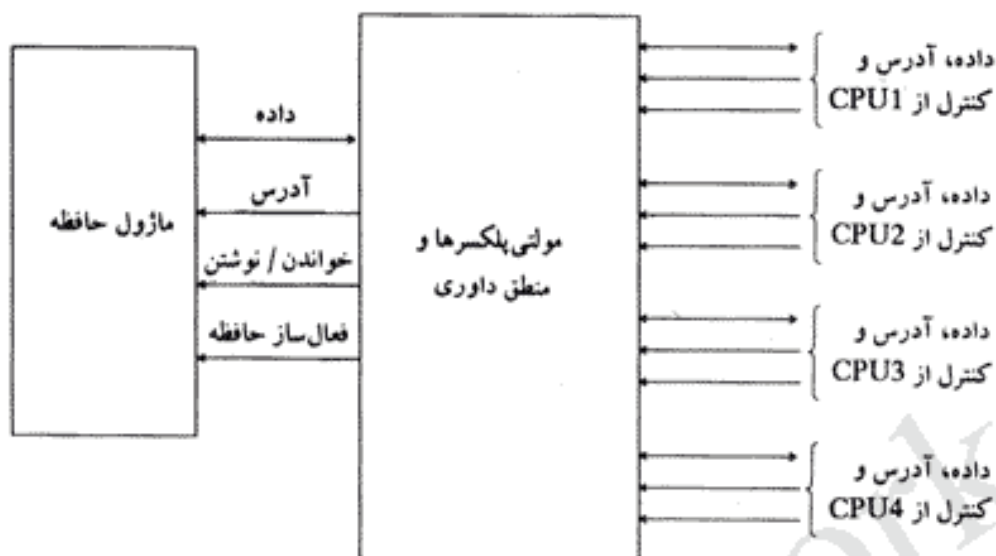
شکل ۵-۱۳ طرح عملیاتی یک سوئیچ تقاطعی متصل به یک ماژول حافظه را نشان می دهد. این مدار از مولتی پلکسرهایی تشکیل می شود که داده ها، آدرس، و کنترل را از یک CPU برای ارتباط با ماژول حافظه انتخاب می کنند. سطوح اولیه توسط منطق داوری^۱ برقرار می شود تا هر وقت دو

ماژول های حافظه



شکل ۴-۱۳ سوئیچ تقاطعی

1- Arbitration Logic

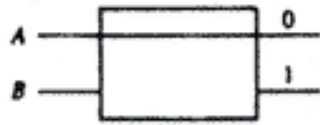


شکل ۵-۱۳ بلاک دیاگرام سوئیچ تقاطعی

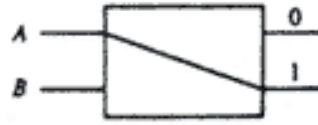
یا چند CPU اقدام به مراجعه به یک حافظه کنند یکی از آنها انتخاب شود. مولتی پلکسرها با کد دودویی که توسط یک گذرگاه اولویت در داخل منطق داوری ایجاد می شود کنترل می گردند. سازمان سوئیچ تقاطعی انتقال های همزمان از همه ماژول های حافظه را پشتیبانی می کند زیرا برای هر ماژول یک مسیر جداگانه موجود است. با این وجود سخت افزار لازم برای پیاده سازی سوئیچ می تواند بسیار بزرگ و پیچیده باشد.

شبکه سوئیچینگ چند طبقه

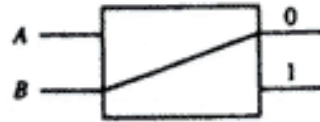
جزء اصلی یک شبکه چند طبقه، یک سوئیچ تبادلی دو ورودی - دو خروجی است. همانطور که در شکل ۶-۱۳ دیده می شود، سوئیچ 2×2 دارای دو ورودی A و B، و دو خروجی 0 و 1 است. در رابطه با هر سوئیچ سیگنال های کنترلی (در اینجا نشان داده نشده اند) که این ارتباط بین پایانه های ورودی - و خروجی را برقرار کنند وجود دارند. سوئیچ قابلیت اتصال A را به هر یک از خروجی ها دارد. پایانه B در سوئیچ نقش مشابهی دارد. همچنین سوئیچ داوری بین درخواست های مشکل ساز را دارد. اگر ورودی های A و B، هر دو یک پایانه خروجی را تقاضا کنند، تنها یکی از آنها متصل شده و دیگری بلوکه می شود. با استفاده از سوئیچ 2×2 بعنوان بلاک ساختمانی، می توان یک شبکه چند طبقه برای کنترل ارتباط بین چند منبع و مقصد ساخت. برای ملاحظه نحوه انجام این کار، درخت دودویی شکل ۷-۱۳ را در نظر بگیرید. دو پردازنده P_1 و P_2 از طریق سوئیچ ها به هشت ماژول حافظه که بصورت 000 تا 111 شماره گذاری شده اند متصل شده اند. هر مسیر یک منبع به یک مقصد از بیت های شماره مقصد تعیین می شود. اولین بیت شماره مقصد خروجی سوئیچ را در اولین طبقه تعیین می کند. بیت دوم خروجی



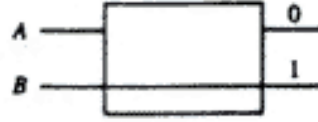
A به 0 وصل شده است



A به 1 وصل شده است

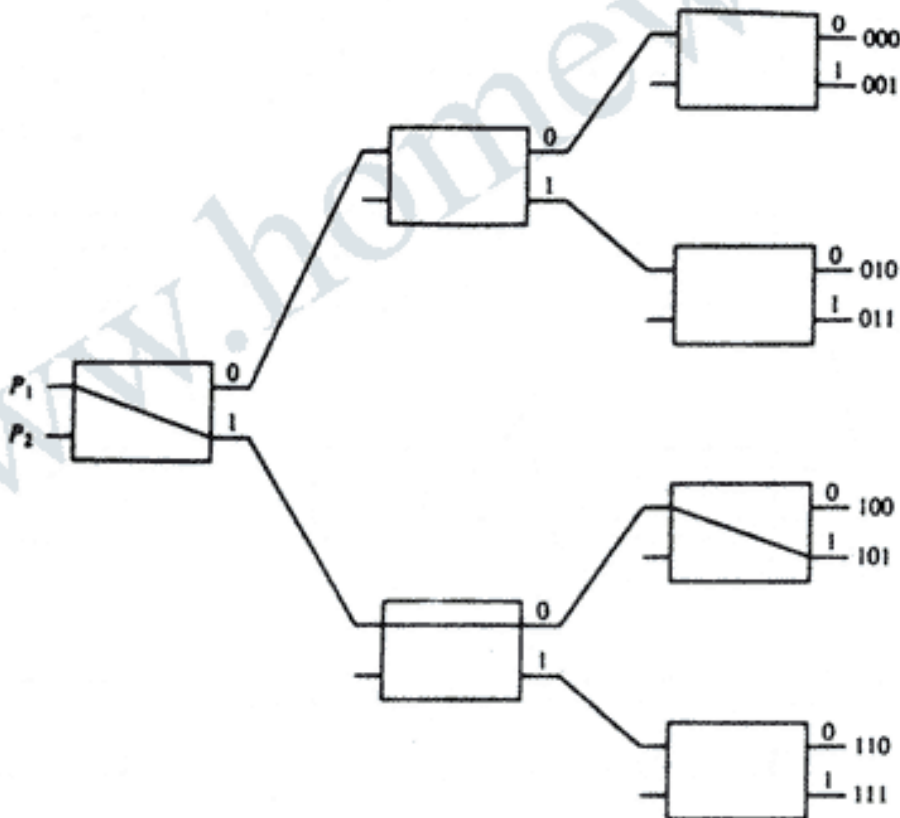


B به 0 وصل شده است



B به 1 وصل شده است

شکل ۶-۱۳ عملکرد یک سوئیچ متقابل 2×2

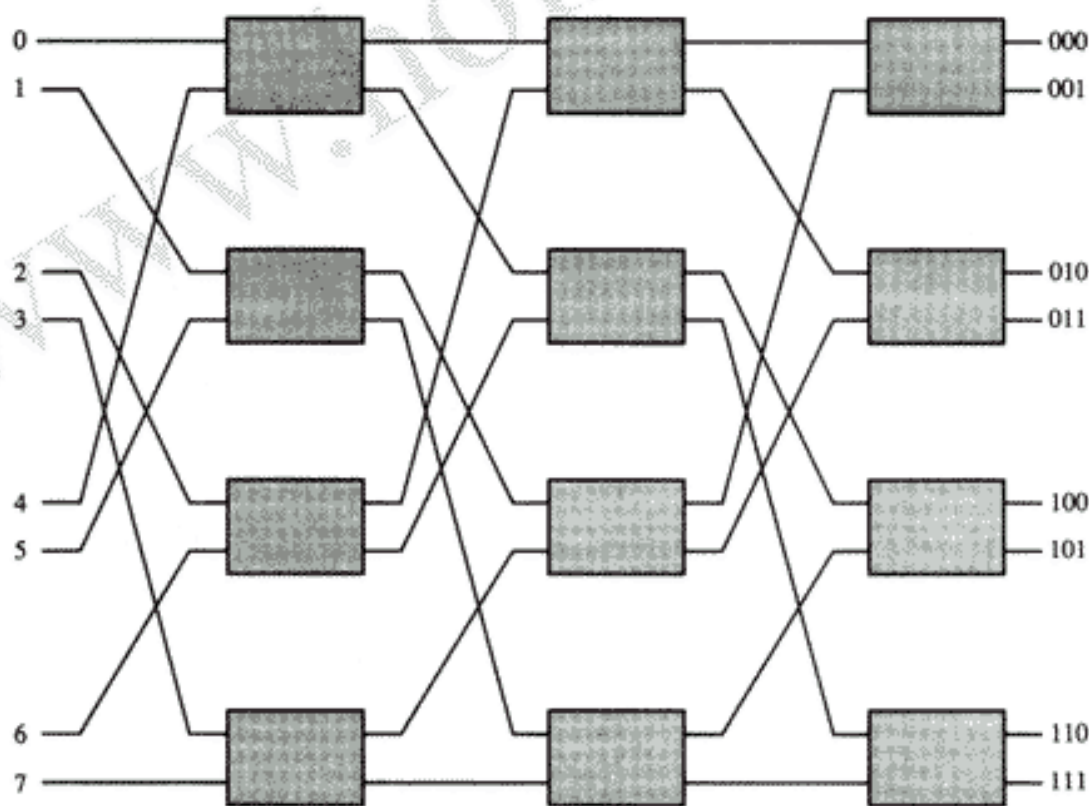


شکل ۷-۱۳ درخت دودویی با سوئیچهای 2×2

سوئیچ طبقه دوم را مشخص می‌سازد، و سومین بیت خروجی سوئیچ را در طبقه سوم معین می‌نماید. مثلاً، برای اتصال P_1 به حافظه 101، لازم است مسیری از P_1 به خروجی 1 در سطح اول سوئیچ، به

خروجی 0 در سطح دوم سوئیچ، و به خروجی 1 در سطح سوم برقرار شود. واضح است که هر یک از دو ورودی P_1 یا P_2 قابل اتصال به هریک از هشت حافظه است. با این وجود الگوهای تقاضای معینی برای ارتباط نمی تواند بطور همزمان ایجاد شود. مثلاً، اگر P_1 به یکی از مقصدهای 000 تا 011 وصل شود، P_2 فقط به یکی از مقصدهای 100 تا 111 وصل می گردد.

توپولوژی های متعددی برای شبکه های سوئیچینگ چند طبقه جهت کنترل ارتباط پردازنده - حافظه در سیستم چند پردازنده با کوپل محکم یا کنترل ارتباط بین عناصر پردازنده در سیستم با کوپل سست پیشنهاد شده است. نمونه ای از این توپولوژی ها شبکه سوئیچینگ شکل ۸-۱۳ می باشد. در این آرایش، دقیقاً یک مسیر از هر منبع به هر مقصد بخصوص وجود دارد. با این وجود برخی الگوهای مورد نظر را نمی توان بطور همزمان برقرار کرد. مثلاً هر دو منبع نمی توانند به مقصدهای 000 و 001 وصل شوند. یک درخواست بخصوص در شبکه سوئیچینگ توسط مبداء آغاز می شود، و ضمن آن یک الگوی ۳ بیتی نماینده شماره مقصد را ارسال می کند. با حرکت الگوی دودویی در طول شبکه، هر یک از سطوح بیت متفاوتی را بررسی می کنند تا نحوه تنظیم سوئیچ 2×2 تعیین شود. سطح 1 پرارزش ترین بیت را بررسی می کند، سطح 2 بیت وسط، و سطح 3 کم ارزشترین بیت را چک می نماید. وقتی که تقاضا به هر یک از ورودی های سوئیچ 2×2 می رسد، اگر بیت مشخص شده 0 باشد به خروجی بالایی و اگر 1 باشد به خروجی پائینی هدایت می شود.



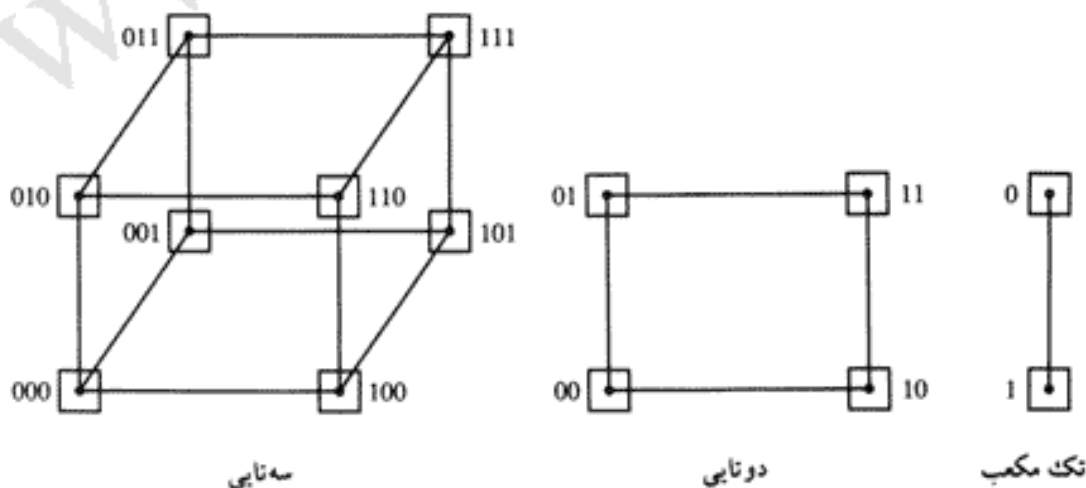
شکل ۸-۱۳ شبکه سوئیچینگ 8×8

در یک سیستم چند پردازنده با کوپل محکم، منبع یک پردازنده و مقصد ماژول حافظه است. اولین مرور در شبکه مسیر را برقرار می‌کند. مرورهای بعدی برای انتقال آدرس به حافظه و سپس انتقال داده‌ها در یکی از دو جهت، بسته به اینکه درخواست خواندن یا نوشتن باشد بکار می‌روند. در سیستم چند پردازنده با کوپل سست، هر دو عنصر منبع و مقصد عناصر پردازنده هستند. پس از برقراری مسیر، پردازنده منبع پیام را به پردازنده مقصد انتقال می‌دهد.

اتصال فوق مکعبی^۱

ساختار چند پردازنده فوق مکعبی یا چند پردازنده n بعدی، یک سیستم با کوپل سست متشکل از 2^n پردازنده است که بشکل یک مکعب دودویی n بعدی بهم متصل شده‌اند. هر پردازنده یک گره را برای مکعب تشکیل می‌دهد. گرچه معمولاً گفته می‌شود که هر گره دارای یک پردازنده است ولی در واقع نه فقط یک CPU، بلکه یک حافظه محلی و یک واسطه I/O نیز بهمراه آن است. هر پردازنده ارتباط مستقیمی با پردازنده‌های مجاور خود دارد. این مسیرها اضلاع مکعب را تشکیل می‌دهند. 2^n آدرس دودویی متمایز وجود دارد که می‌توانند به پردازنده‌ها اختصاص یابند. هر آدرس پردازنده با آدرس هر یک از n پردازنده مجاور آن دقیقاً در یک بیت اختلاف دارند.

شکل ۹-۱۳ ساختار یک فوق مکعب را برای ۳ و ۲ و ۱ نشان می‌دهد. در ساختار مکعب یک تایی داریم $n=1$ و $2^n=2$ این ساختار حاوی دو پردازنده متصل بهم توسط یک مسیر است. ساختار مکعب دوتایی دارای $n=2$ و $2^n=4$ است. این ساختار دارای چهار گره می‌باشد که بشکل مستطیل بهم متصلند. یک ساختار مکعب سه تایی دارای هشت گره متصل بهم می‌باشد و تشکیل مکعب را می‌دهد. یک ساختار مکعب



شکل ۹-۱۳ ساختارهای فوق مکعبی برای $n = 1, 2, 3$

1- Hypercube Interconnection

n تایی دارای 2^n گره بوده و هر پردازنده در یک گره قرار گرفته است. به هر گره یک آدرس دودویی آنچنان اختصاص یافته است که آدرس دو گره مجاور دقیقاً در یک بیت اختلاف دارند. مثلاً سه گره همسایه با آدرس 100 در یک ساختار مکعب سه تایی عبارتند از 000، 110 و 101. هر یک از این اعداد دودویی با آدرس 100 تنها در یک بیت اختلاف دارند.

راهدی پیامها در ساختار مکعب n تایی ممکن است از یک تا n مسیر متصل، از هر گره منبع به گره مقصد باشد. مثلاً در ساختار مکعب سه تایی، گره 000 می تواند مستقیماً با گره 001 ارتباط برقرار کند. برای ارتباط با 011 باید دست کم از دو ضلع عبور کند (از 000 به 001 به 011 یا از 000 به 010 به 011). برای ارتباط گره 000 با گره 111 باید دست کم از سه ضلع عبور کرد. یکی از روش های مسیریابی محاسبه OR انحصاری آدرس گره مبدا با آدرس گره مقصد است. در عدد دودویی حاصل، بیت های متناظر با محورهایی که روی آنها دو گره با هم متفاوتند مقدار 1 خواهند داشت. سپس پیام در طول این محورها ارسال می شود. مثلاً در ساختار مکعب سه تایی، برای ارسال پیامی از 010 به 001، OR انحصاری دو آدرس محاسبه می شود و نتیجه 011 را می دهد. پیام می تواند در طول محور دوم به 000 و سپس در طول محور سوم به 001 ارسال شود.

مثالی از معماری فوق مکعبی، مجموعه کامپیوتری iPSC اینتل است. این سیستم از 128 ($n=7$) ریز کامپیوتر، که از طریق کانال های ارتباطی بهم متصل اند، تشکیل شده است. هر گره از یک CPU، یک پردازنده ممیز شناور، حافظه محلی، و واحدهای واسطه ارتباطات سری تشکیل می شود. هر یک از گره ها بر اساس برنامه های مقیم بطور مستقل بر روی داده های ذخیره شده در حافظه محلی عمل می کنند. داده ها و برنامه های هر گره از طریق یک سیستم پیام رسانی از گره های دیگر یا از مدیر مکعب تامین می شود. برنامه های کاربردی روی مدیر مکعب تهیه، ترجمه و سپس در گره های مختلف بار می شوند. محاسبات در سیستم توزیع و بطور موازی اجرا می گردد.

۳-۱۳ دآوری بین پردازنده ها^۱

سیستم های کامپیوتری برای سهولت انتقال اطلاعات بین اجزاء مختلف دارای تعدادی گذرگاه در سطوح مختلف هستند. CPU حاوی گذرگاه های داخلی برای انتقال اطلاعات بین ثبات های پردازنده و ALU است. گذرگاه حافظه از خط انتقال داده، آدرس و اطلاعات خواندن و نوشتن تشکیل شده است. یک گذرگاه I/O برای انتقال اطلاعات به وسایل ورودی و خروجی و بالعکس استفاده می شود. گذرگاهی که تعداد قابل توجهی از اجزاء مانند CPU ها و IOP را در یک سیستم چند پردازنده ای بهم وصل می کنند گذرگاه سیستم^۲ نامیده می شوند. مدارهای فیزیکی یک گذرگاه سیستم در تعدادی صفحات مدار چاپی^۳ مشابه قرار داده می شود. هر یک از این صفحات سیستم به یک ماژول خاص تعلق

1- Interprocessor Arbitration

2- System bus

3- Printed Circuit Board

دارد. صفحات از طریق کانکتورهایی بطور موازی با هم نصب می شوند. هر پایه از هر کانکتور^۱ یک مدار با سیم به پایه های مربوط به خود بر روی سایر بردها وصل می شود. بنابراین هر مدار را می توان در یک شکاف^۲ در صفحه نگهدارنده ای^۳ که گذرگاه سیستم را تشکیل می دهد قرار داد.

پردازنده ها در یک سیستم چند پردازنده ای نیاز به حافظه مشترک یا سایر منابع مشترک از طریق گذرگاه سیستم دارند. اگر هیچیک از پردازنده ها از گذرگاه سیستم استفاده نکنند، تقاضای دستیابی یک پردازنده ممکن است بلافاصله پاسخ داده شود. با این وجود، در صورت استفاده گذرگاه سیستم بوسیله پردازنده دیگر، پردازنده متقاضی باید صبر کند. علاوه، دیگر پردازنده ها هم ممکن است بطور همزمان گذرگاه سیستم را تقاضا کنند. برای حل این رقابت چند جانبه بوسیله منابع نوعی داوری باید صورت گیرد. منطق داوری بخشی از کنترل کننده گذرگاه سیستم است که بین گذرگاه سیستم، طبق شکل ۲-۱۳، قرار دارد.

گذرگاه سیستم

یک گذرگاه سیستم از تقریباً 100 خط سیگنال تشکیل می شود. این خطوط به سه گروه عملیاتی تقسیم می شوند: خطوط داده، آدرس و کنترل. علاوه خطوط توزیع نیروی الکتریکی نیز برای تأمین توان مورد نیاز قطعات لازم است. مثلاً سیستم چند گذرگاهی 796 استاندارد IEEE دارای 16 خط داده، 24 خط آدرس، 26 خط کنترل و 20 خط توان و کلاً 86 خط می باشد.

خطوط داده مسیری برای انتقال داده بین پردازنده و حافظه مشترک فراهم می نمایند. تعداد خطوط داده معمولاً مضربی از 8 و عمدتاً 16 و یا 32 می باشد. خطوط آدرس برای شناسایی آدرس یک حافظه یا هر منبع یا مقصد دیگر مانند پورت های ورودی یا خروجی می باشد. تعداد خطوط آدرس حداکثر ممکن ظرفیت حافظه را در سیستم معین می کند. مثلاً یک آدرس 24 خطی می تواند تا به 2^{24} (16 مگا) کلمه حافظه دستیابی نماید. خطوط داده و آدرس به بافرهای سه حالته منتهی می شود (شکل ۵-۴). بافرهای آدرس یک جهته بوده و از پردازنده بسمت حافظه می باشند. خطوط داده دو جهته بوده (شکل ۳-۱۲) و انتقال داده را در هر دو جهت ممکن می سازند.

انتقال داده ها در گذرگاه سیستم می تواند همگام و یا غیرهمگام باشد. در یک گذرگاه همگام، هر قلم داده در طول یک بازه زمانی که در ابتدا برای هر دو واحد منبع و مقصد مشخص است انتقال می یابد. عمل همگامی با بکار انداختن هر دو واحد توسط یک منبع ساعت مشترک صورت می گیرد. روش دیگر این است که ساعتهای جداگانه ای با فرکانس تقریباً یکسان در هر واحد داشته باشیم. سیگنال های همگام سازی بطور متناوب ارسال می شوند تا همه ساعتهای سیستم همگام با یکدیگر باقی بمانند. در یک سیستم غیرهمگام هر قلم داده ای که انتقال می یابد با سیگنال کنترل دست دهی (شکل ۹-۱۱) برای نشان دادن زمان انتقال داده ها از مبدأ و دریافت آنها توسط مقصد همراه است.

خطوط کنترل سیگنال هایی را برای کنترل انتقال اطلاعات بین واحدها فراهم می کنند. سیگنال های

1- Connector

2- Slot

3- Backplane

زمانبندی، اعتبار اطلاعات داده و آدرس را مشخص می‌کنند. سیگنال‌های فرمان، عملیاتی را که باید انجام شود مشخص می‌کنند. خطوط کنترل شامل سیگنال‌های انتقال از قبیل خواندن و نوشتن حافظه، تصدیق یک انتقال، درخواست‌های وقفه، و سیگنال‌های کنترل گذرگاه از قبیل درخواست گذرگاه و واگذاری گذرگاه و سیگنال‌های مربوط به رویه‌های داوری هستند.

در جدول ۱-۱۳، ۸۶ خطی که در گذرگاه چندتایی ۷۹۶ استاندارد IEEE وجود دارد فهرست شده است. این سیستم شامل ۱۶ خط داده و ۲۴ خط آدرس است. همه سیگنال‌های این گذرگاه در سطح پائین خود فعال می‌شوند. از جمله سیگنال‌های کنترل انتقال داده عبارتند از خواندن و نوشتن حافظه و خواندن و نوشتن I/O. در نتیجه خطوط آدرس می‌توانند فضاهای حافظه و I/O را جداگانه آدرس‌دهی کنند. وقتی که انتقال به پایان رسید، حافظه یا I/O با سیگنال تصدیق انتقال پاسخ می‌دهد. هر پردازنده متصل به چند پورت تا هشت خروجی درخواست وقفه و یک خط ورودی تصدیق وقفه دارد. این خروجی‌ها معمولاً به یک کنترل کننده وقفه اولویت دار مشابه با کنترل کننده‌ای که در بخش ۲۱-۱۱

جدول ۱-۱۳ سیگنال‌های چند گذرگاه ۷۹۶ استاندارد IEEE

نام سیگنال	
داده آدرس	DATA0-DATA15
خطوط داده (۱۶ خط)	ADRS0-ADRS23
خطوط آدرس (۲۴ خط)	
انتقال داده	
خواندن حافظه	MRDC
نوشتن حافظه	MWTC
خواندن I/O	IORC
نوشتن I/O	IOWC
تصدیق انتقال	TACK
کنترل وقفه	
درخواست وقفه (۸ خط)	INT0-INT7
تصدیق وقفه	INTA
سیگنال‌های کنترل متفرقه	
ساعت اصلی	CCLK
مقداردهی اولیه سیستم	INIT
فعال‌ساز بایت بالا	BHEN
بازدارنده حافظه (۲ خط)	INH1-INH2
فصل گذرگاه	LOCK
داوری گذرگاه	
درخواست گذرگاه	BREQ
درخواست گذرگاه مشترک	CBRQ
اشغال	BUSY
ساعت گذرگاه	BCLK
ورودی اولویت گذرگاه	BPRN
خروجی اولویت گذرگاه	BPRO
منبع تغذیه و زمین (۲۰ خط)	

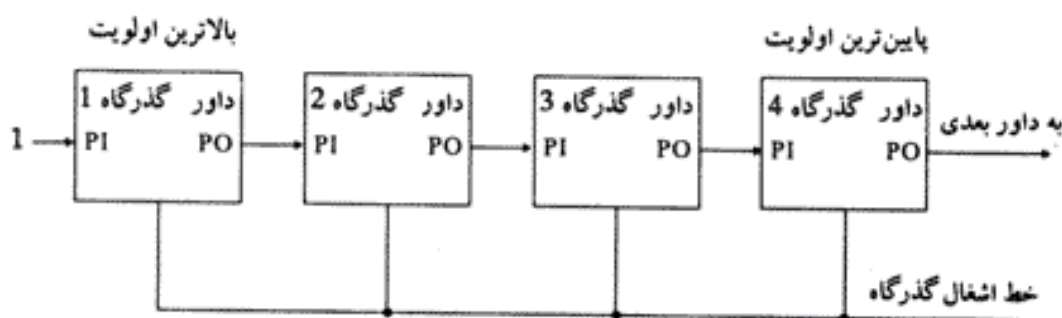
شرح داده شد اعمال می شوند. سیگنال های کنترل متفرقه قابلیت زمانبندی و مقداردهی اولیه را در اختیار می گذارند. خصوصاً سیگنال قفل کردن گذرگاه برای کاربردهای مربوط به چند پردازنده ها ضروری است. این سیگنال که توسط پردازنده فعال می شود برای جلوگیری از در اختیار قرار گرفتن گذرگاه توسط پردازنده های دیگر در حال اجرای دستورالعمل های تست و نشانیدن به کار می رود. این دستورالعمل ها برای همگام سازی صحیح پردازنده لازم اند (بخش ۴-۱۳).

شش سیگنال دآوری گذرگاه برای دآوری بین پردازنده بکار می روند. این سیگنال ها را بعداً پس از بحثی درباره رویه های دآوری سری و موازی شرح می دهیم.

رویه دآوری سری

رویه های دآوری به تمام تقاضاهای پردازنده ها براساس اولویت های مقرر شده سرویس می دهند. تکنیک سخت افزاری اولویت بندی درخواست های گذرگاه را با استفاده از اتصال سری یا موازی واحدهای درخواست کننده کنترل گذرگاه سیستم می توان برقرار کرد. تکنیک اولویت دهی سری از یک اتصال زنجیره دوار^۱ بیان شده در بخش ۵-۱۱ حاصل می گردد. اولویت به پردازنده های متصل به گذرگاه سیستم براساس موقعیت آنها در طول خط کنترل اولویت حاصل می شود. نزدیکترین وسیله به خط اولویت دارای بالاترین اولویت است. وقتی که وسیله های متعددی بطور همزمان استفاده از گذرگاه را درخواست کنند، به وسیله ای که بالاترین اولویت را دارد امکان دستیابی داده می شود.

شکل ۱۰-۱۳ متصل زنجیره دوار چهار داور را نشان می دهد. فرض بر این است که هر پردازنده مدار داور گذرگاه خاص خود را با خطوط ورودی و خروجی اولویت دار است. خروجی اولویت PO از هر داور به ورودی اولویت مدار داور اولویت طبقه پائین تر متصل است. PI مربوط به واحد دارای اولویت بالاتر در مقدار منطقی 1 نگهداشته می شود. اگر واحدی که بالاترین اولویت را در سیستم دارد گذرگاه سیستم را تقاضا کند همیشه موفق به دستیابی به آن خواهد بود. اگر ورودی یک داور خاص PI برابر 1 باشد و پردازنده مربوطه اش تقاضای کنترل گذرگاه را نکند در اینصورت خروجی PO آن برابر 1 است. به این



شکل ۱۰-۱۳ دآوری سری (زنجیره ای)

1- daisy chain

ترتیب است که اولویت به واحد بعدی در زنجیره منتقل می شود. اگر پردازنده درخواست کنترل گذرگاه کند و داور مربوطه اش ورودی PI متناظرش را برابر 1 بیابد خروجی PO را 0 می کند. داور الویت پائین تر یک 0 در PI دریافت کرده و یک 0 در PO تولید می کند. بنابراین به پردازنده ای که دارای $PI=1$ و $PO=0$ باشد کنترل گذرگاه داده می شود.

ممکن است وقتی که پردازنده ای در میانه یک عمل با گذرگاه است، پردازنده ای با الویت بالاتر گذرگاه را درخواست کند. در این حال پردازنده با اولویت پائین تر باید قبل از رها کردن کنترل گذرگاه، عمل خود را روی آن به پایان برساند. خط اشغال گذرگاه^۱ که در شکل ۱۰-۱۳ نشان داده شده است، مکانیزمی را برای کنترل بترتیب انتقال فراهم می سازد. خط اشغال از مدارهای کلکتور باز در هر مدار گرفته شده و اتصال OR سیمی^۲ را فراهم می کند. وقتی که یک داور کنترل گذرگاهی را دریافت می کند (زیرا در آن $PI=1$ و $PO=0$ است) خط اشغال را بررسی می کند. اگر خط غیرفعال باشد، بدان معنی است که پردازنده دیگری در حال استفاده از گذرگاه است. داور خط اشغال را فعال کرده و پردازنده مربوطه آن، اختیار گذرگاه را بدست می گیرد. با این وجود اگر داور گذرگاه را اشغال بیابد مفهوم این است که پردازنده دیگری در حال استفاده از گذرگاه است. داور به بررسی خود از خط اشغال ادامه می دهد در حالی که پردازنده با اولویت پائین تر که کنترل گذرگاه را از دست داده است عمل خود را به پایان می رساند. وقتی که خط اشغال گذرگاه به حالت غیرفعال خود باز گردد، داور با اولویت بالاتر خط اشغال را فعال می نماید و از آن به بعد پردازنده مربوطه اش قادر خواهد بود تا انتقال خود را روی گذرگاه انجام دهد.

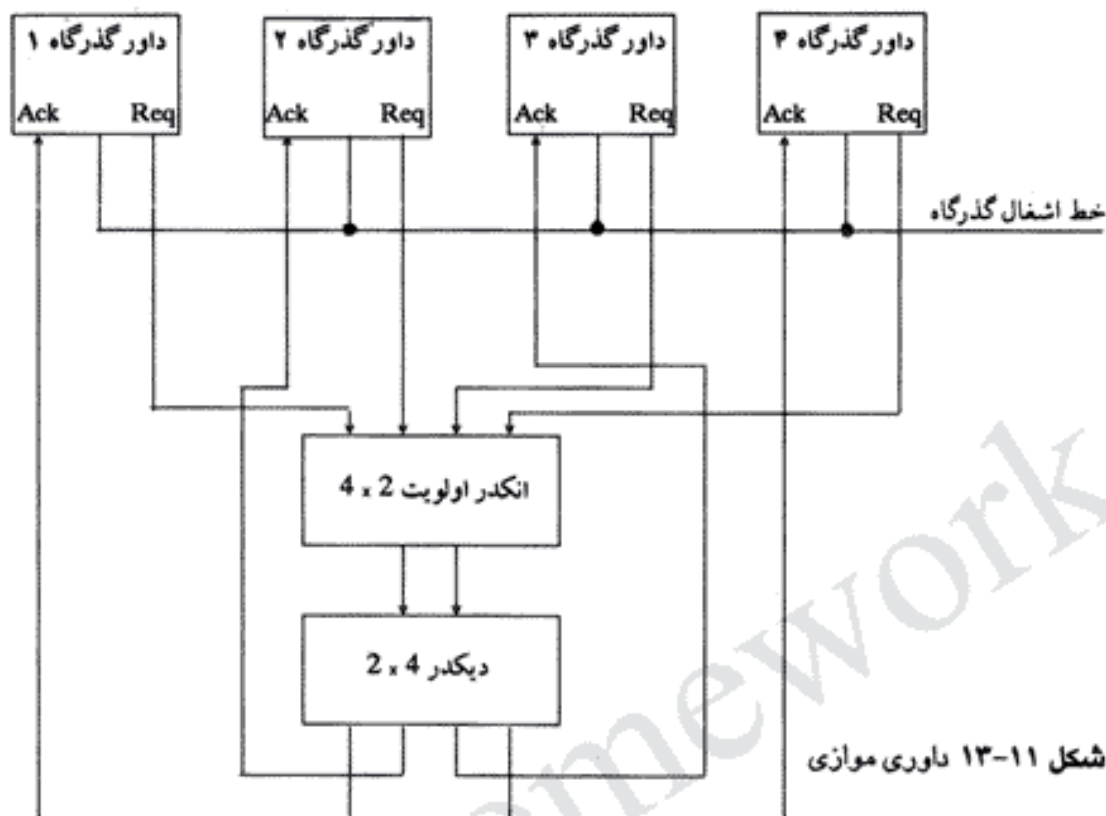
منطق داوری موازی

در تکنیک داوری موازی گذرگاه، از یک آنکدر اولویت خارجی و یک دیگر که طبق شکل ۱۱-۱۳ نشان داده شده است استفاده می شود. در روش موازی، هر داور گذرگاه یک خط خروجی تقاضای گذرگاه و یک خط ورودی تصدیق گذرگاه دارد. هر داور وقتی که پردازنده دستیابی به گذرگاه سیستم را درخواست کند، خط تقاضا را فعال می نماید. پردازنده بشرطی کنترل گذرگاه را بدست می گیرد که خط ورودی تصدیق فعال شده باشد، خط اشغال گذرگاه همچون روشن زنجیره ای، کنترل بترتیب انتقال را انجام می دهد.

شکل ۱۱-۱۳ خطوط تقاضا را که از چهار داور به یک آنکدر اولویت می روند نشان می دهد. خروجی آنکدر یک کد دو بیتی را که نمایش دهنده واحدی است که بالاترین اولویت را در میان متقاضیان گذرگاه دارد تولید می کند. جدول درستی آنکدر اولویت را می توان در جدول ۲-۱۱ (بخش ۵-۱۱) یافت. کد دو بیتی از خروجی دیگر، 2×4 که خط تصدیق مشخصی را فعال می نمایند، دستیابی به گذرگاه را برای واحدی که بالاترین اولویت را دارد امکان پذیر می سازد. اکنون ما عملکرد و سیگنال های داوری اولویت لیست شده در جدول ۱-۱۳ را توضیح می دهیم.

1- Bus Busy Line

2- Wired OR



ورودی اولویت گذرگاه^۱ BPRN و خروجی اولویت گذرگاه^۲ BPRO برای اتصال زنجیره‌ای مدارهای دآوری گذرگاه بکار رفته‌اند. سیگنال اشغال گذرگاه BUSY یک خروجی کلکتور باز است که برای اعلام مشغول بودن گذرگاه به همه داورها به کار می‌رود. درخواست گذرگاه مشترک CBRQ نیز یک خروجی کلکتور باز است که برای اعلام این موضوع به داور بکار می‌رود که داورهای دیگری با اولویت پائین تر در حال درخواست استفاده از گذرگاه سیستم هستند یا خیر. سیگنال‌های مورد استفاده برای پیاده‌سازی روبه دآوری موازی، سیگنال درخواست گذرگاه BREQ و ورودی اولویت BPRN متناظر با سیگنال‌های درخواست و تصدیق شکل ۱۱-۱۳ هستند. ساعت گذرگاه BCLK برای همگام کردن همه همزمانی همه انتقال‌های گذرگاه بکار می‌روند.

الگوریتم‌های دآوری دینامیک

دوروش دآوری که در فوق توصیف شد از الگوریتم اولویت استاتیک استفاده می‌نماید زیرا اولویت هر وسیله توسط ارتباطش با گذرگاه تثبیت می‌گردد. درمقابل، یک الگوریتم دآوری دینامیک به سیستم

1- Bus Priority - in

2- Bus Priority - out

توانایی تغییر اولویت وسایل را در حین عمل فراهم می‌کند. اکنون درباره چند رویه داوری که از الگوریتم‌های داوری دینامیک استفاده می‌کنند بحث خواهیم کرد.

الگوریتم برش زمانی^۱، برشی با طول ثابت از زمان گذرگاه را بترتیب و بصورت چرخشی^۲ به هر پردازنده اختصاص می‌دهد. سرویس ارائه شده به هر سیستم با این روش مستقل از مکان آن در طول گذرگاه است. به هیچ وسیله خاصی ارجحیتی داده نمی‌شود چون به هر یک مقدار زمان مساوی برای ارتباط با گذرگاه داده می‌شود.

در یک سیستم گذرگاه که از همه پرس‌های^۳ استفاده می‌کند، بجای سیگنال واگذاری گذرگاه مجموعه‌ای از خطوط را بعنوان خطوط همه پرس داریم که به همه واحدها متصلند. این خطوط بوسیله کنترل کننده گذرگاه برای تعیین یک آدرس برای هر وسیله متصل به گذرگاه استفاده می‌شوند. کنترل کننده گذرگاه رشته آدرس‌ها را بصورت از قبل تعیین شده‌ای مرور می‌کند. اگر پردازنده‌ای که درخواست دستیابی نموده آدرس خود را تشخیص دهد، خط اشغال گذرگاه را فعال می‌کند و سپس به گذرگاه دست می‌یابد. پس از چند سیکل گذرگاه، فرآیند همه پرس با انتخاب یک پردازنده دیگر ادامه می‌یابد. دنباله همه پرس معمولاً قابل برنامه ریزی است و لذا انتخاب اولویت را می‌توان تحت کنترل برنامه تغییر داد.

الگوریتم قدیمی‌ترین مورد استفاده (LRU) بالاترین اولویت را به وسیله درخواست کننده می‌دهد که مدت عدم استفاده آن از گذرگاه بیش از بقیه باشد. اولویت‌ها پس از چند سیکل ساعت طبق الگوریتم LRU تنظیم می‌شود. با این رویه به هیچ پردازنده‌ای نسبت به دیگری ارجحیت داده نمی‌شود زیرا اولویت‌ها بصورت دینامیکی تغییر می‌کنند تا هر وسیله فرصت دستیابی به گذرگاه را داشته باشد.

در روش اولین سرویس - اولین درخواست^۴، تقاضاها بترتیبی که دریافت می‌شوند سرویس دهی می‌گردند. برای پیاده کردن این الگوریتم، کنترل کننده گذرگاه، صفی را براساس زمان دریافت تقاضاهای گذرگاه مرتب شده ایجاد می‌کند. هر پردازنده باید برای رسیدن نوبت استفاده از گذرگاه براساس اولین ورودی - اولین خروجی (FIFO)، صبر کند.

روشی زنجیره‌ای چرخان^۵ نوع تعمیم یافته دینامیکی از الگوریتم زنجیره‌ای است. در این روش کنترل کننده مرکزی گذرگاه موجود نیست، و خط اولویت از خروجی اولویت^۶ آخرین وسیله به ورودی اولویت^۷ اولین وسیله بازگشته و یک حلقه بسته را تشکیل می‌دهد. این نحوه اتصال مشابه به شکل ۱۰-۱۳ است جز اینکه خروجی PO داور 4 به ورودی PI داور 1 متصل شده است. هر وسیله‌ای که به گذرگاه دست یابد بعنوان کنترل کننده داور بعدی نیز عمل می‌کند. اولویت هر داور برای هر سیکل گذرگاه بر اساس مکانش در طول خط اولویت گذرگاه و با شروع از داوری که پردازنده آن در حال حاضر کنترل گذرگاه را در دست دارد معین می‌شود. وقتی که داوری گذرگاه را رها می‌کند، کمترین اولویت را خواهد داشت.

1- Time Slite

2- Round - Robin Fashhion

3- Polling

4- First Come - First Serve

5- Rotating Daisy - Chain

6- Priority - Out

7- Priority - in

۴-۱۳ ارتباط و همگامی بین پردازنده‌ها

در یک سیستم چند پردازنده‌ای باید پردازنده‌های متفاوتی با امکان ارتباط با یکدیگر فراهم شوند. مسیر ارتباط می‌تواند از طریق کانال‌های مشترک ورودی - خروجی ایجاد شود. در یک سیستم چند پردازنده با حافظه اشتراکی، متداولترین رویه، کنارگذاشتن بخشی از حافظه است که برای همه پردازنده‌ها قابل دسترسی است. هدف اصلی از بکارگیری این حافظه مشترک ایفای نقش یک مرکز پیام مشابه با یک صندوق پستی است که هر پردازنده می‌تواند پیامی برای سایر پردازنده‌ها باقی‌گذارده و پیام‌های موجود در آن را بردارد.

پردازنده فرستنده یک درخواست، پیام، یا یک رویه را ساخته و آن را در صندوق پستی حافظه قرار می‌دهد. بیت وضعیت واقع در حافظه مشترک معمولاً برای مشترک کردن حالت صندوق پستی، اینکه حاوی اطلاعات معنی دار است یا خیر، و اینکه پیام برای کدام پردازنده قرار داده شده است استفاده می‌شود. پردازنده گیرنده می‌تواند بصورت دوره‌ای صندوق پستی را برای یافتن پیام‌های معتبر و آرسی کند. زمان پاسخ این رویه می‌تواند وقت گیر باشد، چون یک پردازنده فقط با همه پرس‌پیام‌ها می‌تواند تقاضایی را شناسایی کند. رویه کارآکتر این است که پردازنده فرستنده با کمک سیگنال وقفه پردازنده گیرنده را آگاه کند. این عمل را می‌توان توسط یک وقفه بین پردازنده‌ای که بوسیله نرم‌افزار و با استفاده از دستورالعملی در برنامه پردازنده‌ها تولید می‌شود، انجام داد و وقتی اجرا شود وضعیتی مشابه با وقفه خارجی در پردازنده دوم بوجود می‌آورد. باین ترتیب پردازنده متوقف شده آگاه می‌شود که پیام جدیدی توسط پردازنده وقفه دهنده در صندوق پستی قرار داده شده است.

یک سیستم چند پردازنده علاوه بر حافظه مشترک ممکن است منابع مشترک دیگری هم داشته باشد. مثلاً یک دیسک مغناطیسی متصل به یک IOP ممکن است در اختیار همه CPUها قرار گیرد. با این ترتیب امکان استفاده مشترک از برنامه‌های سیستم ذخیره شده بر روی دیسک بوجود می‌آید. مسیر ارتباطی بین دو CPU ممکن است از طریق رابطی بین دو IOP متناظر با دو CPU ایجاد شده باشد. این نوع ارتباط اجازه می‌دهد تا هر CPU دیگری را بعنوان یک وسیله I/O در نظر گرفته و لذا پیام‌ها قابل انتقال از طریق مسیر I/O خواهد بود.

برای جلوگیری از ایجاد مزاحمت ناشی از استفاده منبع مشترک به وسیله چندین پردازنده، باید اولویتی در تخصیص منابع به پردازنده‌ها وجود داشته باشد. این وظیفه به سیستم عامل محول شده است. در طراحی سیستم عامل برای چند پردازنده‌ها، از سه سازمان استفاده شده است. آرایش حاکم و تابع^۱، سیستم عامل جداگانه و سیستم عامل توزیع شده.

در شیوه حاکم و تابع، همیشه یک پردازنده، که حاکم نامیده می‌شود، عملیات سیستم عامل را اجرا می‌کند. پردازنده‌های دیگر بمانند یک تابع عمل کرده و عملیات سیستم عامل را اجرا نمی‌کنند. اگر پردازنده تابع به سرویس سیستم عامل نیاز داشته باشد باید آن را با وقفه دادن به حاکم تقاضا کند و تا

1- Master Slave Mode

وقفه برنامه جاری منتظر بماند.

در سازمان مبتنی بر سیستم عامل جداگانه، هر پردازنده می تواند روالهایی از سیستم عامل را که به آنها نیاز دارد اجرا کند. این سازمان برای سیستم های با کوپل سست که هر پردازنده ممکن است نسخه کامل از سیستم عامل خود را داشته باشد بیشتر مناسب است.

در سازمان مبتنی بر سیستم عامل توزیع شده، روال های سیستم عامل بین پردازنده های موجود توزیع شده اند. با این وجود، هر تابع خاص از سیستم عامل در هر زمان فقط به یک پردازنده اختصاص داده می شود. این نوع سازمان دهی، سیستم عامل شناور¹ خوانده می شود زیرا روال ها از یک پردازنده به پردازنده دیگر شناورند و ممکن است اجرای روال ها در زمان های مختلف به عهده پردازنده های مختلف باشد.

در یک سیستم چند پردازنده با کوپل سست حافظه بین پردازنده توزیع شده و حافظه مشترکی برای تبادل اطلاعات وجود ندارد، تبادل ارتباطات بین پردازنده ها از طریق کانال های I/O صورت می گیرد. این تبادل اطلاعات بوسیله یک پردازنده آغاز می شود و طی آن یک رویه واقع در حافظه، پردازنده ای که مایل به این ارتباط است را فرا می خواند. وقتی که پردازشگر فرستنده و پردازشگر گیرنده یکدیگر را بعنوان منبع و مقصد صدا کنند، یک کانال ارتباط برقرار می شود. سپس پیامی با سرآیند و داده های مختلف مورد استفاده برای ارتباط بین گره ها ارسال می شود. ممکن است مسیرهای متعددی برای ارسال پیام بین دو گره وجود داشته باشد. سیستم عامل موجود در هر گره حاوی اطلاعات مسیریابی که مسیرهای مختلف دیگری که می تواند برای ارسال پیام به سایر گره ها بکار رود را مشخص می کند. کارآرایی تبادل اطلاعات یک شبکه بین پردازنده ای به قرارداد مسیریابی ارتباطی، سرعت پردازنده، سرعت رابط داده ها، و توپولوژی شبکه بستگی دارد.

همگامی بین پردازنده ها

مجموعه دستورالعمل های یک چند پردازنده حاوی دستورالعمل اصلی است که برای پیاده سازی ارتباط و همگامی بین پردازنده های همکار مورد استفاده قرار می گیرد. منظور از ارتباط، تبادل داده ها بین پردازنده هاست. مثلاً، پارامترهایی که به یک رویه در پردازنده دیگری انتقال می یابد یک تبادل بین پردازنده ای را می طلبد. همگامی به حالت خاصی که در آن داده های مورد استفاده در ارتباط بین پردازنده ها اطلاعات کنترلی هستند اتلاق می شود. همگامی برای ایجاد توالی صحیح در پردازش ها و نیز تضمین جلوگیری از مراجعه همزمان به داده های مشترک قابل نوشتن، مورد نیاز است.

سیستم های چند پردازنده ای معمولاً شامل مکانیزم های متعددی برای همگام سازی منابع هستند. عملیات پیش پا افتاده مستقیماً بوسیله سخت افزار پیاده سازی می شوند. این عملیات اساس مکانیزم هایی است که جدایی متقابل را برای مکانیزم های پیچیده تری که توسط نرم افزار پیاده سازی

1-Floating Operating System

می شوند ایجاد می کنند. یکی از متداولترین روش ها استفاده از سمافور دودویی¹ است.

جداسازی متقابل با استفاده از سمافور

برای اینکه یک سیستم چند پردازنده درست کار کند باید مکانیزمی را فراهم سازد تا دستیابی متوالی و منظم به حافظه اشتراکی و دیگر منابع مشترک تضمین شود. این امر برای حفاظت داده ها از تغییر همزمان توسط دو یا چند پردازنده لازم است. این مکانیزم جداسازی متقابل² نامیده می شود. جداسازی متقابل باید در سیستم های چند پردازنده ای برای فعال کردن یک پردازنده فراهم شود تا وقتی که آن پردازنده با یکی از منابع مشترک در یک بخش بحرانی مشغول انجام کار است مانع دستیابی دیگر پردازنده ها به آن منبع گردد. بخش بحرانی، رشته ای از دستورات برنامه است که اگر اجرای آن شروع شود نباید قبل از اتمام اجرای آن هیچ پردازنده دیگری به آن منبع مشترک دست یابد.

یک متغیر دودویی بنام سمافور اغلب برای مشخص کردن اینکه پردازنده ای یک بخش بحرانی را اجرا می کند یا نه بکار می رود. سمافور یک پرچم کنترل شده با نرم افزار است که در مکانی از حافظه که قابل دسترسی بوسیله تمام پردازنده ها است قرار داده می شود. اگر سمافور 1 باشد، مفهوم این است که پردازنده ای یک بخش بحرانی را اجرا می کند و لذا حافظه مشترک برای پردازنده های دیگر مهیا نیست. وقتی که سمافور 0 باشد، حافظه مشترک برای هر پردازنده متقاضی در دسترس است. پردازنده هایی که از یک قطعه حافظه بطور مشترک استفاده می کنند بنا به قرارداد توافق دارند که از آن قطعه استفاده نکنند مگر اینکه سمافور برابر 0 باشد، که نشان دهنده این است که حافظه قابل دسترسی است. همچنین، پردازنده ها توافق دارند تا بهنگام اجرای یک بخش بحرانی سمافور را 1 و در پایان آن را 0 کنند.

تست و 1 کردن سمافور خود یک عمل بحرانی است و باید به صورت یک عمل منفرد غیر قابل تقسیم انجام شود. در غیر این صورت، دو یا چند پردازنده ممکن است بطور همزمان سمافور را تست کرده و هر یک جداگانه آن را 1 کنند و اجازه یابند تا باتفاق وارد بخش بحرانی گردند. این عمل سبب می شود تا اجرای همزمان بخش بحرانی بوجود آید که می تواند منجر به مقداردومی غلط پارامترهای کنترلی و از دست رفتن اطلاعات اساسی گردد.

سمافور را می توان با استفاده از دستورالعمل تست و یک مکانیزم قفل سخت افزاری مقداردومی اولیه نمود. قفل سخت افزاری یک سیگنال تولید شده بوسیله پردازنده است که از بکارگیری گذرگاه توسط پردازنده های دیگر تا زمانی که این سیگنال فعال است ممانعت می کند. دستورالعمل تست و یک کننده سمافور را 1 می کند و مکانیزم قفل را در حین اجرای دستورالعمل فعال می سازد. این عمل سبب می شود تا سایر پردازنده ها در فاصله زمانی اجرای عمل تست و 1 کردن سمافور قادر به تغییر آن نباشند. فرض کنید که سمافور بیت کم ارزشتر مکانی از یک کلمه حافظه است که آدرس آن با SEM نشان داده شده باشد. اجازه بدهید سمبل TSL بمعنی عمل "تست و 1 شدن در حین قفل شدن" باشد.

1- Binary Semaphore

2-Mutual Exclusion

دستورالعمل

TSL SEM

در دو سیکل حافظه بدون تداخل اجرا می شود (اولین سیکل برای خواندن و دوم برای نوشتن)

$$R \leftarrow M[SEM] \quad \text{تست سمافور}$$

$$M[SEM] \leftarrow 1 \quad \text{1 کردن سمافور}$$

سمافور با انتقال مقدارش به ثبات R ، در پردازنده تست شده و سپس مقدار 1 را می گیرد. مقدار واقع در R کار بعدی را معین می سازد. اگر پردازنده $R=1$ را بیابد، خواهد دانست که R سمافور از قبل 1 بوده است. (دقت کنید که دوباره 1 شدن R مقدار سمافور را عوض نمی کند). این بدان معنی است که پردازنده دیگری مشغول اجرای بخش بحرانی است، بنابراین پردازنده ای که سمافور را چک کرده است به حافظه مشترک دست نمی یابد. اگر $R=0$ باشد، یعنی حافظه مشترک مهیا است (یا منبع مشترکی که سمافور نمایندگی آن است). سمافور برابر 1 می شود تا از دستیابی پردازنده های دیگر به حافظه جلوگیری شود. اکنون پردازنده می تواند بخش بحرانی را اجرا کند. آخرین دستورالعمل برنامه باید خانه SEM را 0 کند تا منبع مشترک قابل دسترسی پردازنده های دیگر گردد.

توجه کنید که در حین اجرای دستورالعمل تست و 1 کردن، سیگنال قفل باید فعال باشد. وقتی که سمافور 1 شد دیگر لازم نیست قفل فعال باقی بماند. باین ترتیب مکانیزم قفل از دستیابی پردازنده های دیگر به حافظه در حالی که سمافور در حال 1 شدن است جلوگیری می کند. خود سمافور، وقتی که 1 شود، از دستیابی پردازنده های دیگر به حافظه مشترک در حالی که پردازنده ای مشغول اجرای یک بخش بحرانی است جلوگیری می کند.

۱۳-۵ همبستگی حافظه کش

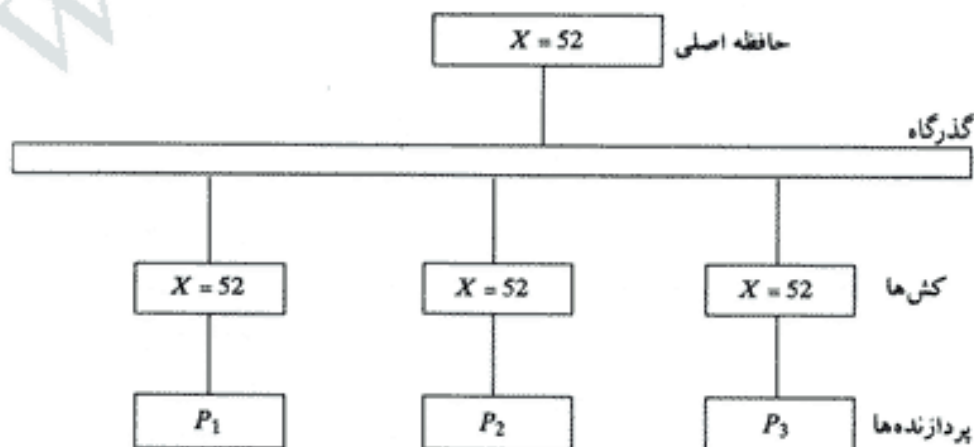
عملکرد حافظه کش در بخش ۶-۱۲ تشریح شد. مزیت عمده حافظه کش قابلیت کاهش میانگین زمان دستیابی در تک پردازنده هاست. هر گاه پردازنده ای در عمل خواندن، کلمه مورد نظری را در کش بیابد، حافظه اصلی در عمل انتقال دخالت نمی کند. اگر عمل نوشتن باشد، دو رویه متداول برای بهنگام کردن حافظه وجود دارد. در سیاست کامل نویسی، هر دو بخش حافظه کش و حافظه اصلی با هر عمل نوشتن بهنگام می شود. در سیاست پس نویسی، فقط حافظه کش بهنگام می شود و خانه موردنظر علامت گذاری می شود تا بعداً در حافظه اصلی کپی شود.

در یک سیستم چند پردازنده با حافظه مشترک، همه پردازنده ها از حافظه مشترک استفاده می کنند. بعلاوه هر پردازنده ممکن است یک حافظه محلی هم داشته باشد، که یک بخش یا تمام آن می تواند کش باشد. دلیل داشتن اجباری حافظه کش جداگانه برای هر پردازنده، کاهش میانگین زمان دستیابی در آن

است. البته اطلاعات یکسانی ممکن است در نسخه های متعدد در برخی حافظه کش و حافظه اصلی قرار داده شود. برای اطمینان از توانایی در اجرای صحیح عملیات حافظه، نسخه های متعدد باید یکسان نگهداشته شوند. این نیاز، مسئله همبستگی حافظه کش^۱ را بوجود می آورد. یک سیستم حافظه همبسته است اگر مقداری که یک دستورالعمل باردهی برمی گرداند همیشه مقداری باشد که با آخرین دستور ذخیره با همان آدرس داده شده است. بدون راه حل مناسبی برای مسئله همبستگی در حافظه کش نمی توان از این نوع حافظه در چند پردازنده ها مبتنی بر گذرگاه با دو یا چند پردازنده استفاده کرد.

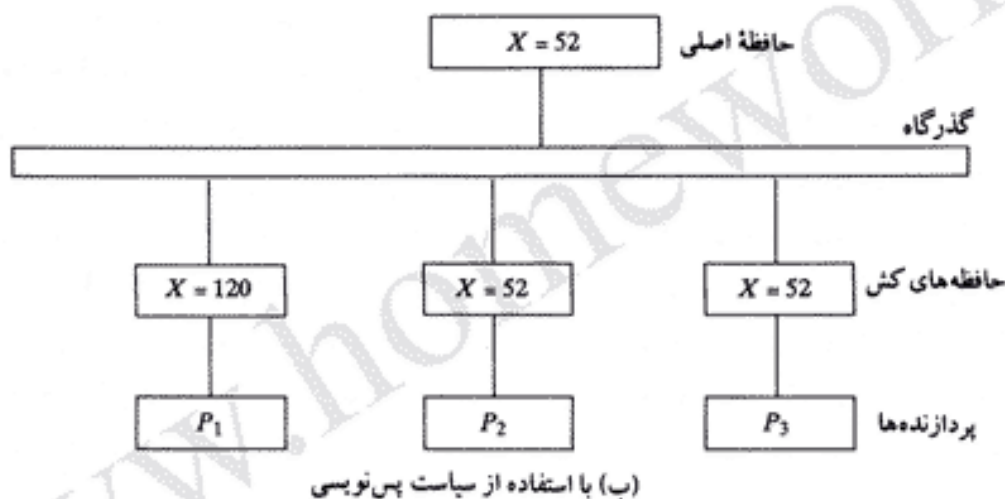
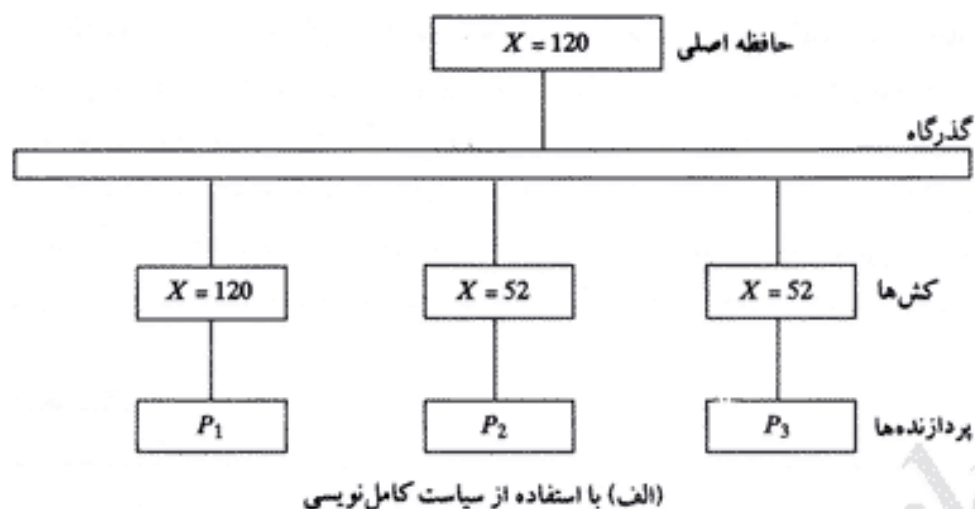
شرایط عدم بستگی

مشکلات همبستگی حافظه کش در چند پردازنده هایی که بدلیل نیاز به داده های مشترک قابل نوشتن دارای حافظه های اختصاصی می باشند بوجود می آید. داده های فقط خواندنی را می توان با اطمینان و بدون مکانیزم های ایجاد کننده همبستگی حافظه کش تکثیر کرد. برای روشن شدن مسئله، آرایش سه پردازنده با حافظه هایی که اختصاصی در شکل ۱۲-۱۳ را در نظر بگیرید. گاهی در حین کار سیستم، عنصری مثل X از حافظه اصلی در سه پردازنده P_1 ، P_2 ، P_3 بار می شود. در نتیجه، این عنصر در حافظه های اختصاصی سه پردازنده نیز کپی می شود. برای سادگی، ما فرض خواهیم کرد که X حاوی مقدار 52 باشد، بار شدن X در سه پردازنده منجر به نسخه های یکسان در حافظه های کش و حافظه اصلی می گردد. اگر یکی از پردازنده عمل ذخیره در X را انجام دهد، نسخه های X در حافظه های کش تغییر می کنند. بارکردن بوسیله پردازنده های دیگر آخرین مقدار را بر نمی گرداند. بسته به سیاست بهنگام سازی حافظه در سیستم حافظه کش، ممکن است محتوای حافظه اصلی هم با حافظه کش یکی نباشد. این مطلب در شکل ۱۳-۱۳ نشان داده شده است. یک عمل ذخیره در X با مقدار 120 در حافظه کش پردازنده P_1 در



شکل ۱۲-۱۳ آرایش کش پس از بار کردن در X

1- Cache Coherence



شکل ۱۳-۱۳ آرایش کش پس از ذخیره کردن در X بوسیله پردازنده P₁

سیاست کامل نویسی، حافظه را به مقدار جدید بهنگام می‌کند. سیاست کامل نویسی هماهنگی بین حافظه اصلی و حافظه کش مبداء تغییر را حفظ می‌کند، اما دو حافظه کش دیگر ناهماهنگ خواهند بود زیرا هنوز حاوی مقدار قبلی هستند. در سیاست پس نویسی، حافظه اصلی در هنگام ذخیره بهنگام نمی‌شود. نسخه‌های موجود در دو حافظه کش دیگر و حافظه اصلی متفاوت خواهند بود. در نهایت حافظه اصلی وقتی که داده‌های تغییر داده شده موجود در حافظه کش در آن کپی شوند بهنگام می‌شود. یک پیکربندی دیگر که ممکن است موجب مشکلات مربوط به ناهماهنگی شود عمل دستیابی مستقیم به حافظه DMA همراه با یک IOP متصل به گذرگاه سیستم است. در حالت ورودی، DMA ممکن است خانه‌هایی از حافظه اصلی را که در حافظه کش نیز مقیم‌اند تغییر دهد بدون آنکه حافظه کش را بهنگام نماید. در حین عمل خروجی DMA، در صورت استفاده از سیاست پس نویسی، ممکن است مکان‌هایی از حافظه اصلی قبل از اینکه روی حافظه کش بهنگام گردند، خوانده شوند. با شرکت دادن

IOP در حل مشکل همبستگی حافظه کش که در سیستم بکار گرفته می شود می توان برنا همبستگی مبتنی بر I/O غلبه کرد.

راه حل هایی برای حل مسئله همبستگی حافظه کش

روش های متعددی برای حل مسئله همبستگی حافظه کش در چند پردازنده های با حافظه مشترک پیشنهاد شده است. ما در اینجا برخی از این روش ها را بطور مختصر بحث می کنیم. برای بحث های مشروح تر به مراجع 3 و 10 مراجعه نمایید.

یک روش ساده عبارتست از عدم اجازه به وجود حافظه های کش اختصاصی برای هر پردازنده و داشتن یک حافظه کش مشترک همراه با حافظه اصلی است. هر دستیابی داده به حافظه کش اشتراکی صورت می گیرد. این روش اصل نزدیک بودن CPU را به حافظه کش نقض می کند و میانگین زمان دستیابی را افزایش می دهد. در واقع این روش مشکل را با اجتناب از آن حل می نماید.

از دیدگاه ملاحظات مربوط به عملکرد، بهتر است که یک حافظه کش در هر پردازنده وجود داشته باشد. یکی از روشهای بکار رفته فقط اجازه ذخیره داده های غیرمشترک و فقط خواندنی را در حافظه های کش می دهد. چنین داده هایی قابل کش شدن¹ خوانده می شوند. داده های قابل نوشتن مشترک قابلیت ذخیره شدن در کش را ندارند.² کامپایلر باید داده ها را از نظر قابل ذخیره شدن در کش و نشدن عنوان گذاری کند، و سخت افزار سیستم طوری عمل خواهد کرد که فقط داده های قابل ذخیره در کش ذخیره شوند. داده هایی که در کش ذخیره نمی شوند در حافظه اصلی باقی می مانند. این روش نوع داده های ذخیره شده در حافظه ای کش را محدود کرده و یک سریار اضافی را که ممکن است موجب افت عملکرد گردد ایجاد می نماید.

روشی که اجازه وجود داده های قابل نوشتن در حداقل یک حافظه کش را می دهد، روشی است که از یک جدول سراسری متمرکز در برنامه کامپایلر خود استفاده می کند. وضعیت بلاک های حافظه در جدول سراسری مرکزی ذخیره می شوند. هر بلاک به عنوان فقط خواندنی (RO) یا خواندنی - نوشتنی (RW) مشخص می گردد.

تمام حافظه های کش می توانند یک کپی از بلاک ها را که به عنوان RO مشخص شده اند داشته باشند. بنابراین اگر داده ها در حافظه کشی که حاوی بلاک RW است بهنگام شود، دیگر حافظه های کش بعلت نداشتن کپی هایی از این بلاک تحت تأثیر قرار نمی گیرند. مسئله همبستگی حافظه کش می تواند با استفاده از ترکیبی از نرم افزار، و سخت افزار یا با استفاده از روشهای صرفاً سخت افزاری حل شود. دو روشی که قبلاً ذکر شد از رویه های مبتنی بر نرم افزار استفاده می نمایند که قابلیت نشانه گذاری برای جلوگیری از قرارگرفتن داده های قابل نوشتن در حافظه کش دارند. در راه حل های سخت افزاری، کنترل کننده کش بطور خاصی طراحی می شود تا بتواند بر همه درخواست های گذرگاه از سوی CPU ها و IOP ها

1- Cachable

2- Nonchachable

نظارت داشته باشد. همه حافظه کش متصل به گذرگاه بطور مستمر بر شبکه برای یافتن یک عمل احتمالی نوشتن نظارت می کنند. این حافظه های کش باید در صورت کشف انطباق، بسته به روش مورد استفاده کپی های موجود خود را بهنگام و یا غیر معتبر نمایند. کنترل کننده گذرگاهی که بر این عمل نظارت می کند کنترل کننده مداخله گر^۱ نامیده می شود. این کنترل کننده اساساً یک واحد سخت افزاری است که برای برقراری یک مکانیزم نظارت بر گذرگاه در همه حافظه های نهان متصل به گذرگاه طراحی می شود. روش های مختلفی برای حل مسئله همبستگی حافظه نهان با استفاده از قرارداد کنترل کننده مداخله گر حافظه کش پیشنهاد شده است ساده ترین متد استفاده از سیاست کامل نویسی و استفاده از رویه زیر است. تمام کنترل کننده های مداخله گر برای یافتن عملیات ذخیره حافظه بر گذرگاه نظارت می کنند. هرگاه کلمه ای با نوشته شدن در یک حافظه کش بهنگام شود، مکان متناظر آن در حافظه اصلی نیز بهنگام می شود. کنترل کننده های مداخله گر محلی مربوط به کلیه حافظه های کش دیگر، حافظه خود را واریسی می کنند تا مشخص کنند کپی کلمه ای را که روی آن نوشته اند دارند یا خیر. اگر کپی در حافظه کش دیگر وجود داشته باشد، آن مکان بعنوان نامعتبر علامت گذاری می شود. چون همه حافظه کش همه عمل های نوشتن گذرگاه را زیر نظر دارند، هرگاه کلمه ای نوشته شود، اثر خالص این عمل بهنگام شدن آن در حافظه کش و حافظه اصلی و حذف آن از همه حافظه های کش دیگر خواهد بود. اگر در آینده، پردازنده ای به داده نامعتبر واقع در حافظه کش خود مراجعه نماید، پاسخی که دریافت می کند باعث در حافظه کش خواهد بود، و شکل بهنگام آن داده از حافظه اصلی آورده می شود. باین ترتیب از ایجاد کپی های متفاوت با کپی های جدید داده جلوگیری می گردد.

مسائل

- ۱-۱۳ اختلاف بین چند پردازنده با کوپل محکم و چند پردازنده با کوپل سست را از نقطه نظر سازمان سخت افزار و تکنیک های برنامه نویسی بحث کنید.
- ۲-۱۳ هدف از کنترل کننده گذرگاه سیستم در شکل ۲-۱۳ چیست؟ توضیح دهید که چگونه می توان سیستم را به صورتی طراحی کرد که ارجاع به حافظه محلی و ارجاع به حافظه مشترک را از هم تشخیص دهد.
- ۳-۱۳ در یک شبکه سوئیچ تقاطعی که p پردازنده را به m مازول حافظه متصل می کند چند نقطه سوئیچ وجود دارد.
- ۴-۱۳ شبکه سوئیچینگ امگای 8×8 در شکل ۸-۱۳ سه طبقه و چهار سوئیچ در هر طبقه و جمعاً ۱۲ سوئیچ دارد. برای یک شبکه سوئیچینگ $m \times m$ چند طبقه و چند سوئیچ در هر طبقه لازم است.
- ۵-۱۳ فرض کنید که در شبکه سوئیچینگ امگای شکل ۱۳-۸ سیم بین سوئیچ واقع در ردیف اول و ستون دوم و سوئیچ واقع در ردیف دوم ستون سوم قطع شود. چه مسیریابی مسدود می شود؟

1- Snoopy Cache Controller

- ۱۳-۶ نموداری را برای یک شبکه سوئیچینگ امگای 4×4 بسازید. تنظیم های سوئیچ ها را برای اتصال ورودی 3 به خروجی 1 نشان دهید.
- ۱۳-۷ برای طراحی یک شبکه ارتباطات چند طبقه از سه نوع سوئیچ استفاده شده است: سوئیچ تبدیلی^۱ با دو ورودی و دو خروجی طبق شکل ۱۳-۶، یک سوئیچ داوری با دو ورودی و یک خروجی، و سوئیچ توزیع با یک ورودی و دو خروجی
(الف) نحوه عملکرد سوئیچ های داوری و توزیع را نشان دهید.
(ب) با استفاده از سوئیچ های داوری و تبدیلی، یک شبکه 8×4 با یک مسیر منحصر بفرد بین هر مبدأ و هر مقصد بسازید.
(ج) با استفاده از سوئیچ های توزیع و تبدیلی، یک شبکه 4×8 با یک مسیر منحصر بفرد بین هر مبدأ و هر مقصد بسازید.
- ۱۳-۸ دیاگرامی رسم کنید که ساختار یک شبکه فوق مکعبی چهار بعدی را نشان دهد. همه مسیرهای موجود از گره 7 به گره 9 را که از حداقل تعداد گره های میانی استفاده می کنند مشخص کنید.
- ۱۳-۹ یک نمودار منطقی با استفاده از گیت و فلیپ فلاپ رسم کنید که مدار یک طبقه داور گذرگاه رادر طرح داوری زنجیره ای شکل ۱۳-۱۰ نشان دهد.
- ۱۳-۱۰ مدار تحت کنترل مدار داوری موازی در شکل ۱۳-۱۱ در ابتدا بکار است. سپس وسیله های 2 و 3 بطور همزمان گذرگاه را درخواست می کنند. مفادیر دودویی ورودی و خروجی را در آنکدر و دیگر مشخص نمایند و مشخص کنید به کدام داور گذرگاه سیگنال تصدیق داده می شود.
- ۱۳-۱۱ نشان دهید که چگونه مدار داوری شکل ۱۳-۱۰ را می توان برای فراهم نمودن یک رویه داوری زنجیره ای بسته اصلاح کرد. توضیح دهید که وقتی خط گذرگاه غیرفعال است چگونه اولویت تعیین می شود.
- ۱۳-۱۲ یک توپولوژی گذرگاه که در آن دو پردازنده از طریق یک بافر در حافظه مشترک با هم تبادل اطلاعات می کنند را در نظر بگیرید. اگر یکی از پردازنده ها بخواهد با پردازنده دیگری ارتباط برقرار کند، اطلاعات را در بافر حافظه قرار می دهد و پرچمی را 1 می کند. پردازنده دیگر بطور متناوب پرچم ها را واری می کند تا معلوم کند که اطلاعاتی برای دریافت دارد یا خیر. برای اطمینان از همگامی صحیح و به حداقل رساندن زمان بین ارسال و دریافت اطلاعات چه کاری می توان انجام داد؟
- ۱۳-۱۳ اصطلاحات زیر را در رابطه با چند پردازنده شرح دهید. (الف) جداسازی متقابل، (ب) بخش بحرانی، (ج) قفل سخت افزاری، (د) سمافور، (ه) دستورالعمل تست و 1 نمودن.
- ۱۳-۱۴ همبستگی حافظه کش چیست، و چرا در سیستم های چند پردازنده با حافظه مشترک مهم است؟ چگونه می توان این مسئله را با یک کنترل کننده مداخله گر حافظه نهان حل کرد؟

1- Interchange Switch