

# معماری کامپیوتر

نویسنده

موریس مانو

## فهرست مطالب

<b>۱.....</b>	<b>فصل ۱: مدارهای منطقی دیجیتال .....</b>
	۱-۱ کامپیوترهای دیجیتال ۱
	۱-۲ گیت‌های 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
<b>۴۱.....</b>	<b>فصل ۲: قطعات دیجیتال</b>
	۲-۱ مدارهای مجتمع ۴۱
	۲-۲ دیکدرها ۴۴
	۲-۳ مولتی پلکسراها ۴۸
	۲-۴ ثبات‌ها ۵۰
	۲-۵ شیفت رجیسترها ۵۴
	۲-۶ شمارنده‌های دودویی ۵۷
	۲-۷ واحد حافظه ۶۱
	مسائل ۶۵
<b>۶۸.....</b>	<b>فصل ۳: نمایش داده‌ها</b>
	۳-۱ انواع داده‌ها ۶۸
	۳-۲ متهم‌ها ۷۶
	۳-۳ نمایش معیز - ثابت ۷۹
	۳-۴ نمایش معیز شناور ۸۵
	۳-۵ انواع دیگر کدهای دودویی ۸۶
	۳-۶ کدهای آشکارسازی خط ۸۹
	مسائل ۹۲
<b>۹۴.....</b>	<b>فصل ۴: انتقال ثبات‌ها و ریز عمل‌ها.....</b>
	۴-۱ زیان انتقال ثبات ۹۴
	۴-۲ انتقال ثبات ۹۶

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

## فصل ۵: سازمان و طراحی یک کامپیووتر پایه ..... ۱۲۴

۵-۱ کدهای دستورالعمل ها	۱۲۴
۵-۲ ثبات های کامپیووتر	۱۲۸
۵-۳ دستورالعمل های کامپیووتر	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 کامپیووتر	۱۵۷
۵-۹ طراحی کامپیووتر پایه	۱۵۹
۵-۱۰ طراحی مدار منطقی انباره	۱۶۴
مسائل	۱۶۸

## فصل ۶: برنامه نویسی کامپیووتر پایه ..... ۱۷۳

۶-۱ مقدمه	۱۷۳
۶-۲ زبان ماشین	۱۷۴
۶-۳ زبان اسembly	۱۷۹
۶-۴ اسembler	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	۲۲۱
	مسائل	۲۲۶

## فصل ۸: واحد مرکزی پردازش ..... ۲۴۰

۸-۱	مقدمه	۲۴۰
۸-۲	سازمان ثبات‌های عمومی	۲۴۱
۸-۳	سازمان پشتہ	۲۴۶
۸-۴	قالب دستورالعمل‌ها	۲۵۴
۸-۵	روش‌های آدرس دهنی	۲۵۹
۸-۶	انتقال و دستکاری داده‌ها	۲۶۵
۸-۷	کنترل برنامه	۲۷۲
۸-۸	RISC کامپیووتر کم دستور	۲۸۲
	مسائل	۲۹۱

## فصل ۹: پردازش خط لوله‌ای 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 کامپیووتر ..... ۳۳۱

۱۰-۱	مقدمه	۳۳۱
۱۰-۲	جمع و تفریق	۳۳۲
۱۰-۳	الگوریتم ضرب	۳۳۸
۱۰-۴	الگوریتم‌های تقسیم	۳۴۵
۱۰-۵	عمل‌های حسابی ممیز شناور	۳۵۲
۱۰-۶	واحد حساب دهدهنی	۳۶۲
۱۰-۷	اعمال حسابی دهدهنی	۳۶۸
	مسائل	۳۷۴

**فصل ۱۱: سازمان ورودی - خروجی..... ۳۷۹**

۱۱-۱	وسایل جانبی	۳۷۹
۱۱-۲	واسطه ورودی - خروجی	۳۸۲
۱۱-۳	انتقال غیرهمزمان داده	۳۹۰
۱۱-۴	شبوهای 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	داوری بین پردازنده ها	۵۰۰
۱۳-۴	ارتباط و همگامی بین پردازنده ها	۵۰۷
۱۳-۵	همبستگی حافظه کش	۵۱۰
	مسائل	۵۱۴

# مدارهای منطقی دیجیتال

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

## ۱-۱ کامپیوترهای دیجیتال

کامپیوتر دیجیتال سیستمی دیجیتالی<sup>۱</sup> (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 گستره را بخود اختصاص می‌دهند نمایش داده می‌شوند. این مقادیر در داخل کامپیوتر بوسیله اجزائی که می‌توانند تعداد محدودی از حالت‌های گستره را ذخیره حفظ کنند پردازش می‌شوند. مثلًاً ارقام دهدی ۰، ۱، ۲، ...، ۹، ده مقدار گستره را فراهم می‌آورند. اولین کامپیوترهای الکترونیک دیجیتال که در اواخر دهه ۱۹۴۰ ساخته شدند اساساً برای محاسبات عددی بکار می‌رفتند. در این مورد، عناصر گستره 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 منطق دودویی (یعنی بکارگیری

---

1- Digital

عبارات صحیح و غلط، بله یا خیر) قطعات دیجیتال که مقید به نگهداری مقادیر گسته هستند مقید به نگهداری و حفظ فقط دو مقدار که دودویی<sup>۱</sup> نامیده می شوند نیز می باشند.

کامپیوترهای دیجیتال از سیستم اعداد دودویی<sup>۲</sup> استفاده می کنند که دو رقم بیشتر ندارد: ۰ و ۱. یک رقم دودویی بیت خوانده می شود. اطلاعات در کامپیوترهای دیجیتال بوسیله گروههایی از بیت‌ها نشان داده می شوند. با استفاده از تکنیک‌های کدگذاری، گروههای بیت‌ها نه تنها برای نمایش اعداد دودویی بلکه برای سایر سمبول‌های گسته، همچون ارقام دهدی یا حروف الفبا نیز بکار بردۀ می شوند. با استفاده صحیح از مجموعه‌های دودویی و بکارگیری روش‌های مختلف کدگذاری می‌توان گروههای بیت‌ها را برای ساخت مجموعه‌های کامل دستورالعمل<sup>۳</sup>‌ها جهت انجام محاسبات مختلف بکار برد.

برخلاف اعداد دهدی متداول که سیستم مبنای ۱۰ را بکار می‌برند، اعداد دودویی سیستم مبنای ۲ با دو رقم ۰ و ۱ را استفاده می‌نمایند. معادل دهدی یک عدد دودویی را می‌توان با بسط آن به صورت یک سری از توان‌های ۲ بدست آورد. مثلاً عدد دودویی 10010111 کمیتی را نشان می‌دهد که قابل تبدیل به یک عدد دهدی است و از ضرب هر بیت در توان صحیحی از ۲ 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 دهدی آن ۷۵ 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 چیزهایی است. 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 نرم‌افزاری<sup>۴</sup> و سخت افزاری<sup>۵</sup> تقسیم می‌گردد. سخت افزار کامپیوتر شامل تمام قطعات الکترونیک و الکترومکانیک تشکیل دهنده ماهیت فیزیکی آن است. نرم‌افزار کامپیوتر از دستورالعمل‌ها و داده‌هایی<sup>۶</sup> تشکیل شده که کامپیوتر برای انجام کارهای مختلف داده پردازی از آنها استفاده می‌نماید. رشته‌ای از دستورات، برنامه<sup>۷</sup> خوانده می‌شود. داده‌هایی که برنامه با آنها کار می‌کند پایگاه داده‌ها<sup>۸</sup> را تشکیل می‌دهد.

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

---

1- Binary

2- Binary Number System

3- Instruction

4- Software

5- Hardware

6- Data

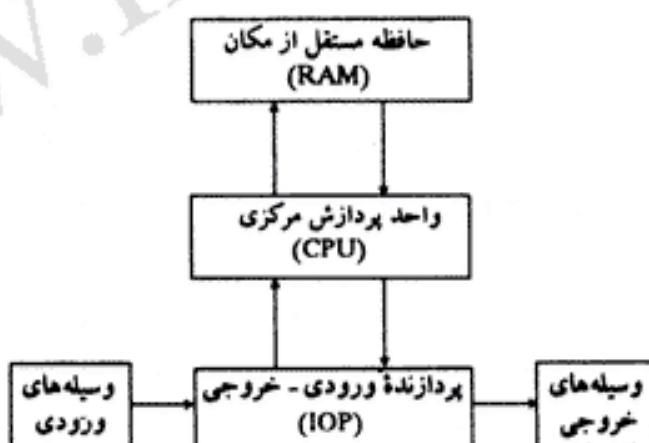
7- Program

8- Data Base

می باشد. برنامه هایی که بصورت یک بسته نرم افزاری سیستم درآمده اند سیستم عامل<sup>۱</sup> خوانده می شوند. این برنامه ها از برنامه هایی که کاربر برای حل مسائل خاصی می نویسد متمایز است. بعنوان مثال، یک برنامه به زبان سطح بالا که توسط کاربر برای حل یک داده پردازی خاص نوشته شده است یک برنامه کاربردی است ولی یک کامپایلر<sup>۲</sup> که برنامه زبان سطح بالا را به برنامه های به زبان ماشین تبدیل می کند یک برنامه سیستم است. خریداری کامپیوتر را می خرد علاوه بر سخت افزار به انواع نرم افزار های موجود برای بکارگیری موثر تر کامپیوتر هم نیاز دارد. نرم افزار سیستم بخش لاینک یک سیستم کامپیوتری است. نقش آن جبران اختلافات موجود بین نیازهای کاربر و توانایی سخت افزار است.

سخت افزار کامپیوتر معمولاً به سه بخش عمده مطابق شکل ۱-۱ تقسیم می شود. واحد پردازش مرکزی<sup>۳</sup> CPU حاوی یک واحد حساب و منطق<sup>۴</sup> ALU برای کار روی داده ها، تعدادی ثبات برای ذخیره کردن داده ها، واحد های کنترل برای برداشت<sup>۵</sup> و اجرای<sup>۶</sup> دستور العمل هاست. حافظه یک کامپیوتر محل نگهداری دستور العمل ها و داده هاست. این حافظه، حافظه با دستیابی تصادفی<sup>۷</sup> خوانده می شود 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<sup>۸</sup> ورودی و خروجی (IOP) حاوی مدارات الکترونیک برای کنترل تبادل و انتقال اطلاعات بین کامپیوتر و دنیای خارج است. وسایل<sup>۹</sup> ورودی - خروجی متصل به کامپیوتر عبارتند از صفحه کلیدها، چاپگرهای پرینتر، پایانه ها، راه اندازهای دیسک مغناطیسی و سایر وسایل تبادل اطلاعات و ارتباطی دیگر است.

این کتاب دانش ابتدایی لازم را برای درک عملیات سخت افزاری 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<sup>۱</sup>، طراحی کامپیوتر<sup>۲</sup> و معماری کامپیوتر<sup>۳</sup> تمايز قابل شویم.

سازمان کامپیوتر به نحوه عملکرد اجزاء قطعات سخت افزاری و چگونگی اتصالات آنها در تشکیل سیستم کامپیوتر اشاره دارد. فرض براین 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 گردد. این جنبه از سخت افزار کامپیوتر را گاهی پیاده سازی<sup>۴</sup> کامپیوتر می خوانند.

معماری کامپیوتر به ساختار و رفتار کامپیوتر از دیدگاه کاربر مربوط می شود. این بحث شامل مطالبی درباره قالب اطلاعات، مجموعه دستور العمل ها، و تکنیک های آدرس دهی حافظه است. طراحی معماری یک سیستم کامپیوتری درباره مشخصات انواع مازول های<sup>۵</sup> 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 مراحلی 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 ولتاژ های 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 ۱ یا ۰ تصور شود. مثلاً یک کامپیوتر دیجیتال ممکن است از یک سیگنال ۳ ولت برای نمایش ۱ دودویی و ۰.۵ ولت برای نشان دادن ۰ دودویی استفاده نماید. پایانه های ورودی مدارهای دیجیتال سیگنال های ۳ و ۰.۵ را پذیرفته و پایانه های خروجی هم با سیگنالهای ۳ و ۰.۵ به ورودی ها پاسخ می دهند. ورودی ها و خروجی های ۳ و ۰.۵ بترتیب متناظر با مقادیر دودویی ۱ و ۰ 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

استفاده می شود. دستکاری اطلاعات دودویی توسط مدارهای منطقی بنام گیت<sup>۱</sup> انجام می شود. گیت ها بلاک های سخت افزاری هستند که سیگنال های دودویی ۰ یا ۱ را در خروجی، بسته به شرایط ورودی هایشان تولید می کنند. امروزه انواع گیت های منطقی در سیستم های کامپیووتر دیجیتال بکار برده می شوند. هر گیت سمبول 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 درستی<sup>۲</sup> نشان داد.

نام، سمبول 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 ۱ است اگر هر دو ورودی A و B برابر ۱ باشد؛ در غیر اینصورت خروجی آن ۰ است. این شرایط برای گیت AND بصورت جدول مشخص شده است. جدول همچنین نشان می دهد که X برابر ۱ است اگر هر دو ورودی A و B برابر ۱ باشند. سمبول جبری عملکرد تابع AND شبیه سمبول 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 ۱ است که تمام ورودی ها ۱ باشند. گیت OR تابع OR غیر انحصاری را تولید می کند، یعنی خروجی ۱ است اگر ورودی A یا هر B یا هر دو ۱ باشند؛ در غیر اینصورت خروجی ۰ است. سمبول 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 ۱ است که هر یک از ورودی ها ۱ باشند.

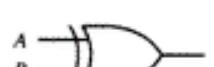
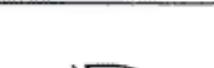
مدار معکوس کننده<sup>۳</sup> وضعیت منطقی هر سیگنال دودویی را معکوس می کند. این مدار تابع NOT یا متمم را تولید می نماید. سمبول جبری مورد استفاده برای متمم منطقی، علامت پریم یا خط روی سمبول متغیر مورد نظر است. در این کتاب ما از علامت پریم برای متمم منطقی یک متغیر دودویی استفاده می کنیم و خط روی حرف را برای مشخص کردن زیر عمل متمم که در فصل ۴ تعریف خواهد شد نگه میداریم. دایره کوچک در خروجی سمبول گرافیکی معکوس کننده، مشخص کننده متمم منطقی است. یک سمبول مثلثی شکل، مربوط به یک بافر<sup>۴</sup> (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 ۳ ولت را برای عدد دودویی ۱ مورد استفاده قرار می دهد. هرگاه در ورودی ۳ ولت را دریافت کند خروجی ۳ 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 هاست که توان قابل ملاحظه ای را نیاز دارند.

1- Gate

2- Truth table

3- Inverter

4- Buffer

جدول درستی	تابع جبری	سیمبل گرافیکی	نام
$\begin{array}{ c c c } \hline A & B & x \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ \hline \end{array}$	$x = A \cdot B$ یا $x = AB$		AND
$\begin{array}{ c c c } \hline A & B & x \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ \hline \end{array}$	$x = A + B$		OR
$\begin{array}{ c c c } \hline A & x \\ \hline 0 & 1 \\ 1 & 0 \\ \hline \end{array}$	$x = A'$		معکوس کننده
$\begin{array}{ c c c } \hline A & x \\ \hline 0 & 0 \\ 1 & 1 \\ \hline \end{array}$	$x = A$		مانگیر
$\begin{array}{ c c c } \hline A & B & x \\ \hline 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ \hline \end{array}$	$x = (AB)'$		NAND
$\begin{array}{ c c c } \hline A & B & x \\ \hline 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ \hline \end{array}$	$x = (A+B)'$		NOR
$\begin{array}{ c c c } \hline A & B & x \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ \hline \end{array}$	$x = A \oplus B$ یا $x = A'B + AB'$		یا انحصاری (XOR)
$\begin{array}{ c c c } \hline A & B & x \\ \hline 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ \hline \end{array}$	$x = (A \oplus B)'$ یا $x = A'B' + AB$		انحصاری NOR یا تابع برابری

شکل ۱-۲ گیت های منطقی دیجیتال

تابع NAND متمم تابع AND است و همانطور که از سمبول گرافیکی آن مشخص است از یک سمبول NOT-AND NAND از اختصار گرفته شده است. کلمه OR متمم NOR است و سمبول گرافیکی OR که بدنیال آن دایره کوچکی تشکیل شده است. گیت NOR ممکن است بیش از دو ورودی داشته باشند، و خروجی آنها همیشه بترتیب متمم توابع AND و OR است.

گیت OR انحصاری<sup>1</sup> دارای سمبول 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 توامان یک باشند. تابع OR انحصاری (XOR یا EOR) می تواند توسط سمبول گرافیکی خاص خودش و یا بر حسب گیتهای AND و OR و متمم طبق شکل ۱-۲ نشان داد. خروجی آنها همیشه بترتیب متمم OR انحصاری است. شود. NOR انحصاری<sup>2</sup> همانطور که از سمبول گرافیکی آن مشخص است متمم OR انحصاری است. خروجی این گیت فقط هنگامی 1 است که هر دوی ورودی 1 و یا هر دو ورودی 0 باشند. نام مناسبتری برای گیت OR انحصاری تابع فرد<sup>3</sup> است، یعنی خروجی 1 است اگر تعداد فردی از ورودی ها 1 باشد. 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. گیتهای ENOR و EOR 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 z مشخص می شوند. سه عمل اصلی منطقی عبارتند از AND، OR و متمم. یک تابع بول را می توان با استفاده از متغیرهای دودویی، سمبول های عملیات منطقی، پرانتزها و یک علامت تساوی نشان داد. تابع بول برای هر مقدار مفروض از متغیرها می تواند 1 یا 0 باشد. یعنوان 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 گوییم  $y'z = 1$  معادل است با اینکه بگوییم  $y = 0$ ،  $z = 1$  است. بنابراین ما ممکن است بگوییم که F هنگامی 1 است که  $x = 1$  یا  $y = 0$  یا  $z = 1$  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 ترکیب هایی 1 است که در آنها  $x = 1$ ،  $y = 0$  و  $z = 1$  باشد؛ این تابع برای سایر ترکیب ها صفر است.

یک تابع بول را می توان از یک عبارت جبری به یک دیاگرام منطقی مشکل از گیت های AND، OR و یا معکوس کننده تبدیل کرد. دیاگرام منطقی  $F$  در شکل ۱-۳ (ب) نشان داده شده است برای ورودی  $z$ ، معکوس کننده ای وجود دارد که متمم  $z$  را تولید می نماید. برای جمله  $z'$  یک گیت AND وجود دارد و از یک گیت OR برای ترکیب دو جمله استفاده شده است. در یک دیاگرام منطقی، متغیرهای تابع ورودی های مدار و سمبول تابع، خروجی آن را تشکیل می دهد.

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 رابطه ای بین ورودی - خروجی یک دیاگرام منطقی بصورت عبارت جبری

۳- یافتن مدار ساده تر 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 می توان فرم ساده تری که گیت های کمتری نیاز دارد را برای آن بدست آورد. برای دیدن چگونگی انجام این اعمال، ابتدا توانایی های دستکاری جبر بول را می بینیم.

جدول ۱-۱ ابتدایی ترین اتحادهای<sup>۱</sup> جبر بول را نشان می دهد. تمام اتحادهای جدول بواسیله جدول درستی قابل اثباتند. هشت اتحاد اول رابطه اساسی بین یک متغیر و خودش، یا با ثابت های ۰ و ۱ را نشان می دهد. پنج اتحاد بعدی (از معمولی ۹ تا ۱۳) شبیه به روابط در جبر معمولی هستند. اتحاد ۱۴ در جبر معمولی صادق نیست ولی در دستکاری<sup>۲</sup> عبارات بولی مفید است. اتحادهای ۱۵ و ۱۶ تثویر های دومورگان نامیده می شوند و در زیر بحث شده اند. آخرین اتحاد بیان می دارد که اگر متغیری دوبار متمم شود مقدار اولیه آن بدست می آید.

$z$	$y$	$x$	$F$
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



(ب) دیاگرام منطقی

(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) $(x')' = x$	

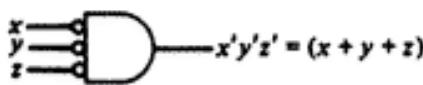
اتحادهایی که در جدول لیست شده‌اند هم در مورد متغیرهای تکی و هم در مورد توابع بولی که بر حسب مجموعه‌ای از متغیرهای دو دویی بیان شده‌اند صادق‌اند. مثلاً عبارت جبری بولی زیر را در نظر بگیرید:

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

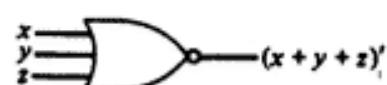
با فرض  $x = AB' + C'B$  رابطه فرق برابر است با  $x = x + x$ . با استفاده از اتحاد ۵ در جدول ۱-۱ می‌بینیم که  $x + x = x$  است. بنابراین عبارت فوق بصورت زیر کاهاش می‌یابد

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

شوری دومورگان در کار با گیت‌های NOR و NAND بسیار با اهمیت است. این شوری بیان می‌کند که یک گیت NOR که عمل  $'(x+y)$  را انجام می‌دهد معادل با  $'(x'y')$  است. بطور مشابه یک تابع NAND می‌تواند به یکی از دو صورت  $'(xy)$  یا  $'(x'y')$  بیان شود. باین دلیل همان‌طور که در شکل‌های ۱-۴ و ۱-۵ دیده می‌شود گیت‌های NOR و NAND دو سمبول نمایشی متمایز دارند. بعوض نمایش گیت NOR با یک OR و دایره کوچکی بدنبال آن، می‌توان آن را با یک گیت AND با دایره‌های کوچکی در تمام ورودی‌هایش نشان داد. سمبول معکوس‌کننده-AND برای گیت NOR از شوری دومورگان حاصل می‌شود و در آن دایره کوچک به معنی متمم سازی است. بطور مشابه، گیت NAND دارای دو سمبول متمایز است، شکل ۱-۵.

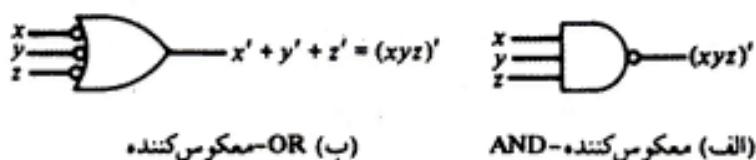


(ب) AND-معکوس‌کننده

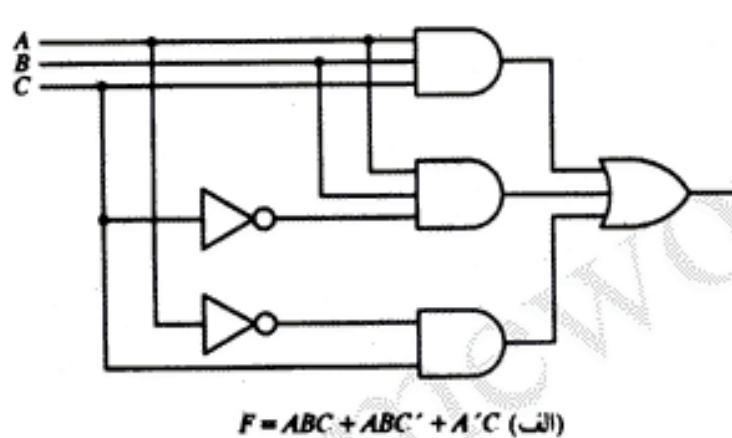


(الف) معکوس‌کننده-OR

شکل ۱-۴ دو سمبول گرافیکی برای گیت NOR



شکل ۱-۵ دو سعیل گرافیک، برای کیت NAND.



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

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

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

هر جمله متعلق به یک گیت AND است و گیت OR جمله سه جمله را تشکیل می‌دهد. برای متهم‌های 'A' و 'C' دو معکوس کننده<sup>۱</sup> لازم است. عبارت فوق می‌تواند بصورت زیر ساده شود.

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

توجه کنید طبق رابطه ۷ که در جدول ۱-۱  $(C + C') = 1$  و  $AB = 1$  است، دیاگرام منطقی عبارت ساده شده در شکل ۱-۶ (ب) کشیده شده است. این دیاگرام تنها به چهار گیت در عرض شش گیت شکل ۱-۶ (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.

### متتم یک تابع

متتم یک تابع مانند F را، وقتی که بصورت جدول درستی نشان داده شده باشد، می توان از تعویض ۱ ها به ۰ ها بدست آورد. بهنگام نمایش تابع بصورت عبارت جبری، متتم تابع با توجه به تشوری ڈمورگان حاصل می شود. فرم کلی تصوری ڈمورگان 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 تصوری کلی ڈمورگان می توانیم رویه ساده ای برای بدست آوردن متتم یک تابع نتیجه بگیریم. این کار با تبدیل تمام گیت های AND به OR و تمام OR ها به AND و سپس متتم سازی هر متغیر بتنها یعنی، حاصل می شود. بعنوان مثال رابطه زیر و متتم آن را در نظر بگیرید

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

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

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

### ۱-۴ 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 بصور 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 میترم نام دارد. مثلاً، جدول درستی شکل ۱-۳ هشت میترم دارد. هنگامی که تابع  $n$  متغیره بوسیله نقشه کارنو نشان داده شود دارای  $2^n$  میترم خواهد بود که

معادل  $2^n$  عدد دودویی حاصل از  $n$  بیت است. تابع بولی به ازاء بعضی از میترم ها برابر ۱ و برخی دیگر ۰ است. با نوشتمن معادل دهدی میترم هایی که مقدار تابع را ۱ میکنند میتوان اطلاعات موجود در جدول درستی را بشکل فشرده ای درآورد. مثلاً جدول درستی شکل ۱-۳ بصورت زیر بیان میشود

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

حروف داخل پرانتز متغیرهای دودویی را بترتیبی که در جملات میترم ظاهر میشوند نشان میدهند. سمبول  $\sum$  نشاندهنده مجموع میترم هایی است که پس از آن در پرانتز آمدهاند. میترم هایی که مقدار ۱ را برای تابع تولید میکنند با معادل دهدی شان در پرانتز آمدهاند. میترم هایی که در پرانتز وجود ندارند مقدار ۰ را برای تابع ایجاد میکنند.

نقشه کارنو دیاگرامی است مشکل از تعدادی مربع که هر مربع نشاندهنده یک میترم است. در مربعات مربوط به میترم هایی که عدد ۱ را برای تابع تولید کنند عدد ۱ و در بقیه ۰ یا چیزی نمینویسد. با تشخیص الگوهای مختلف و ترکیب مربع هایی که در نقشه با ۱ ها مشخص شدهاند میتوان 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 شدهاند. تعداد مربع های تشکیل دهنده نقشه  $n$  متغیره  $2^n$  است. میترم  $2^n$  است. نقشه دو متغیره

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

$A$

$C$

(ب) نقشه سه متغیره

	$B$		$B$
	0	1	
0	0	1	
1	2	3	

$A$

(الف) نقشه دو متغیره

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

$AB$

$D$

(ج) نقشه چهار متغیره

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

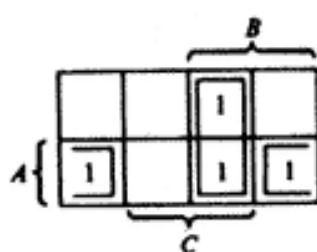
مشخص می شوند. شماره های میترم ها با ترتیب خاصی به خانه ها تخصیص داده شده اند بطوری که میترم های مربوط به دو مربع مجاور فقط در یک متغیر باهم اختلاف دارند. نام متغیرها در دو طرف یک خط مورب در گوش نوشته شده می شوند. ۰ ها و ۱ های نوشته شده در کنار سطرها و ستون ها مقدار متغیرها را مشخص می کنند. در زیر آنکه مربوط به هر متغیر، نیمی از خانه های نقشه مشخص شده که در آنها متغیر مربوطه بدون پریم است. در بقیه خانه های نقشه متغیر دارای پریم (متنم) می باشد. میترم متناظر با هر مربع از روی اعداد دودویی مربوط به متغیرها که در طول سمت چپ و بالای نقشه نوشته شده اند تعیین می شود. مثلاً میترم ۵ در نقشه سه متغیره در سیستم دودویی ۱۰۱ است، که می تواند از کنار هم قرار گرفتن ۱ در سطر دوم و ۰۱ از ستون دوم بدست آید. این میترم مقداری را برای متغیرهای A و B و C مشخص می کند که در آن A و C بدون پریم و B پریم دار است (یعنی  $AB'C$ ). از طرف دیگر میترم ۵ در یک نقشه چهار متغیره دارای چهار متغیر می باشد. عدد دودویی برای چهار بیت ۰۱۰۱ می باشد و میترم مربوطه هم  $A'BC'D$  می باشد.

میترم های مربوطات مجاور در نقشه بجز در یک متغیر یکسانند. این متغیر در یک مربع نسبت به مربع مجاور متمم است. براساس این تعریف از مجاورت، مربوطات انتهایی در سطرها نیز مجاور تصور خواهند شد. موضوع برای مربوطات انتهایی در ستون های نیز صادق است. در نتیجه، چهار مربع در گوش های نقشه نیز مجاور خواهند بود. یک تابع بول که بوسیله جدول درستی اش نمایش داده شده است به یک نقشه منتقل می گردد. بدین ترتیب که در مربع هایی که تابع در آنها ۱ است عدد ۱ نوشته می شود. مربع های مجاوری که در آنها ۱ وجود دارد بصورت گروهی باهم ترکیب می شوند. تعداد مربع های هر گروه باید توان صحیحی از ۲ باشد. هر گروه می تواند در یک یا چند مربع با یک یا چند گروه دیگر اشتراک داشته باشد. هر گروه از مربوطات یک عبارت جبری را نشان می دهد، و OR این جملات، عبارت ساده شده جبری را برای تابع فراهم می آورد. مثالهای زیر کاربرد نقشه را در ساده سازی تابع بول نشان می دهند.

در مثال اول تابع تولی زیر را ساده می کنیم:

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

نقشه سه متغیره این تابع در شکل ۱-۸ نشان داده شده است. در این نقشه چهار مربع با ۱ مشخص شده اند که هر کدام متعلق به یکی از میترم هایی است که برای تابع، ۱ را ایجاد می نمایند. این مربع ها متعلق به میترم های ۳، ۴، ۶ و ۷ بوده و از شکل ۱-۷ (ب) تشخیص داده می شوند. در ستون سوم دو



شکل ۱-۸ نقشه برای  $(3, 4, 6, 7)$

مربع مجاور با هم ترکیب می‌شوند. این ستون متعلق به  $B$  و  $C$  بوده و جمله  $BC$  را تولید می‌کنند. دو مربع باقیمانده حاوی ۱ در دو گوش سطر دوم، مجاورند و متعلق به سطر  $A$  و دو ستون  $C'$  می‌باشند. بنابراین این دو جمله  $AC'$  را تولید می‌کنند. عبارت جبری ساده شده برای تابع عبارتست از OR دو جمله:

$$F = BC + AC'$$

در مثال دوم تابع بولی زیر را ساده می‌کنیم:

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

پنج میترم در مربع‌های مربوطه خود در نقشه سه متغیره در شکل ۱-۹ با ۱ علامت زده شده‌اند. چهار مربع در اولین و چهارمین ستون مجاورند و جمله  $C'$  را بدست می‌دهند. تنها مربع باقیمانده که با ۱ مشخص شده متعلق به میترم ۵ بوده و قابل ترکیب با مربع میترم ۴ می‌باشد که جمله  $AB'$  از آن نتیجه می‌شود. تابع ساده شده عبارتست از:

$$F = C' + AB'$$

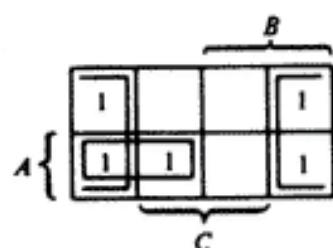
مثال سوم نیاز به یک نقشه چهار متغیره دارد

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

بخشی از نقشه که توسط این تابع چهار متغیره پوشش یافته است مربع‌هایی است که در شکل ۱-۱۰ با ۱ مشخص شده‌اند. تابع دارای ۱ هایی در چهار گوش نقشه است که اگر بصورت یک گروه با هم ترکیب شوند

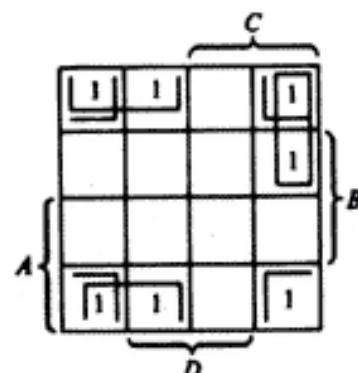
شکل ۱-۹ نقشه برای

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



شکل ۱-۱۰ نقشه برای

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



جمله  $B'D'$  را تولید می‌کنند. دلیل ترکیب این مربعات مجاور بودن آنها بعلت مجاور بودن لبه‌های بالا و پایین و یا چپ و راست است که جمله  $B'C'$  را بدست می‌دهند. تنها 1 باقیمانده در مربع مربوط به میترم 6 است که با میترم 2 ترکیب می‌شود و جمله  $A'CB$  را می‌سازد. تابع ساده شده عبارتست از

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

### ساده‌سازی با ضرب حاصل جمع‌ها

در تمام مثال‌های قبلی توابع بول حاصل از نقشه‌ها به فرم جمع حاصل‌ضرب‌ها 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 جمع‌ها بدست آوریم. مجموع، همان OR جملات و حاصل‌ضرب تیز AND آنها هستند. با کمی تصحیح، ضرب حاصل جمع‌ها را از نقشه می‌توان بدست آورد. روش بدست آوردن عبارت ضرب حاصل جمع‌ها، از خصوصیات اساسی جبر بول حاصل می‌گردد. 1 ها در نقشه نشان‌دهنده میترم‌هایی است که برای تابع تولید 1 می‌نمایند. مربع‌هایی که با 1 پر نشده‌اند، برای تابع 0 تولید می‌کنند. اگر ما مربع‌های خالی را با 0 پر کنیم و سپس آنها را با هم بر اساس مربع‌های مجاور ترکیب نمائیم، متمم تابع را خواهیم داشت. با متمم نمودن  $F'$ ، عبارتی برای  $F$  بصورت ضرب حاصل جمع‌ها بدست می‌آید. بهترین روش برای درک بهتر ارائه یک مثال است. می‌خواهیم تابع بولی زیر را به هر دو روش جمع حاصل‌ضرب‌ها و ضرب حاصل جمع‌ها 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 میترم‌هایی است که 1 را برای تابع تولید می‌کنند. مربع‌هایی که با 0 مشخص شده‌اند نشان دهنده میترم‌هایی هستند که در  $F$  وجود ندارند و بنابراین متمم  $F$  می‌باشند. ترکیب مربع‌های 1، فرم ساده شده تابع را بصورت مجموع حاصل‌ضرب‌ها فراهم می‌کند:

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

		C	
		0	1
A	0	0	0
	1	0	0
		D	
		0	1
		1	1

شکل ۱-۱۱ نقشه برای

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

اگر مربع های ۰ ترکیب شوند، طبق شکل، متمم ساده شده تابع بدست می آیند.

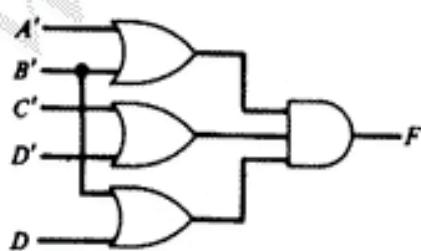
$$F' = AB + CD + BD'$$

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

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

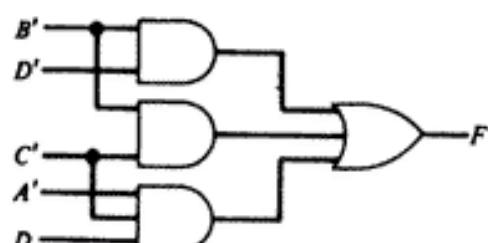
دیاگرام های منطقی دو عبارت ساده شده در شکل ۱-۱۲ نشان داده شده اند. عبارت جمع حاصل ضرب با تعدادی گیت AND در شکل ۱-۱۲ (الف) پیاده سازی شده است که هر عبارت مربوط به یک گیت AND می باشد. خروجی های گیت های AND به ورودی های یک گیت OR متصل است. همان تابع بكمک گیت های OR بصورت ضرب حاصل جمع ها در شکل ۱-۱۲ (ب) 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 ۱-۱۲ فرم کلی پیاده سازی هر تابع بولی است که به یکی از شکل های استانداره 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 ۱-۱۳ (الف) نشان داده شده است جمع حاصل ضرب ها می توانند 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

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

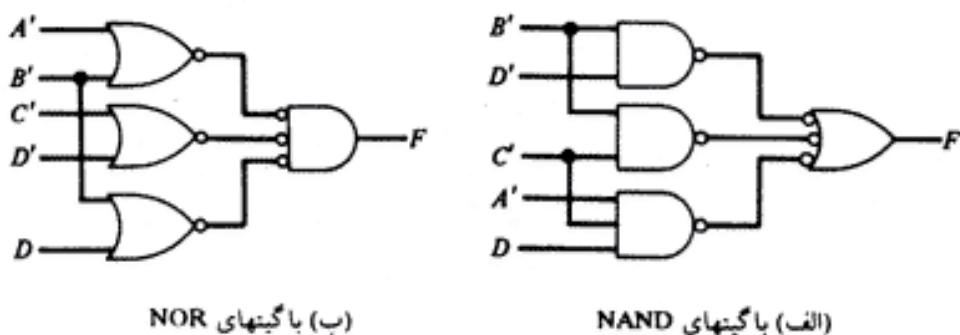


(الف) جمع حاصل ضرب ها

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

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

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



شکل ۱-۱۳ دیاگرام های منطقی با گیت های NOR و NAND

### حالات بی اهمیت

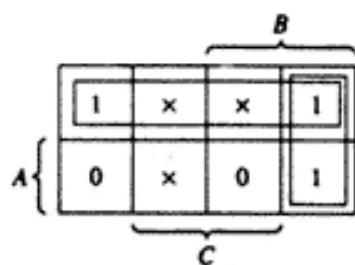
۱ها و ۰ها در نقشه میترم هایی را نشان می دهند که بترتیب تابع را ۱ یا ۰ می کنند. گاهی مهم نیست که تابع به ازاء یک میترم معین، ۰ ایجاد کند و ۱. چون تابع ممکن است تابع ۰ یا ۱ باشد، می گوییم اهمیتی ندارد که خروجی تابع برای این میترم چیست. میترم هایی که ۰ یا ۱ را برای یک تابع تولید می نمایند حالات بی اهمیت خوانده شده و با علامت X در نقشه مشخص می شوند. این حالات بی اهمیت را می توان برای ساده تر کردن عبارت جبری بکار برد.

وقتی که مربع های مجاور برای تابع در نقشه انتخاب می شوند، X ها ممکن است ۰ یا ۱ فرض شوند، بسته به اینکه کدام یک عبارت ساده تری 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 می آید فقط به حد ساده سازی حاصل بستگی دارد. بعنوان مثال، تابع بولی زیر را با جملات بی اهمیت آن در نظر بگیرید.

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

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

میترم هایی که در F آمده اند مقدار ۱ را برای تابع تولید می کنند. میترم های بی اهمیت که در F آمده اند هر یک از دو مقدار ۰ یا ۱ را می توانند تولید نمایند. باقیمانده میترم ها یعنی میترم های ۴ و ۷ مقدار ۰ را برای تابع تولید می کنند. نقشه در شکل ۱-۱۴ نشان داده شده است. میترم های F با ۱، میترم های d با X، و مربع های باقیمانده با ۰ مشخص شده اند. ۱ها و Xها به هر روش مناسبی با هم ترکیب می شوند.



شکل ۱-۱۴ مثالی از نقشه با حالات بی‌اهمیت

تا حد اکثر تعداد مربعهای مجاور را در بر بگیرند. لزومی ندارد که تمام یا هر یک از X ها را در این ترکیب‌ها در نظر بگیریم ولی تمام ۱ ها باید در نظر گرفته شوند. با جایگزینی X ها در میترم‌های ۱، ۳ با ۰ ها، جمله 'A' را از سطر اول بدست می‌آوریم. ۱ باقیمانده برای میترم ۶ هم با میترم ۲ ترکیب شده و از آن جمله 'BC' بدست می‌آید. عبارت ساده شده بصورت زیر است:

$$F = A' + BC'$$

توجه کنید که میترم بی‌اهمیت ۵ در ترکیب گنجانده نشد زیرا به ساده شدن عبارت کمکی نمی‌نماید. همچنین اگر میترم‌های بی‌اهمیت ۱ و ۳ با ۱ 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 ۰ یا ۱ 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 میترم‌های اصلی ۰، ۲ و ۶ و میترم بی‌اهمیت ۱ و ۳ است. میترم ۵ در تابع گنجانده نشده است. از آنجاکه میترم‌های ۱، ۳ و ۵ بعنوان حالات بی‌اهمیت در تابع در نظر گرفته شدند ما میترم‌های ۱ و ۳ را با مقدار ۱ و میترم ۵ را ۰ انتخاب کردیم. این انتخاب باین علت صورت گرفت تا ساده‌ترین فرم برای عبارت بولی حاصل شود.

## ۱-۵ مدارهای ترکیبی

یک مدار ترکیبی آرایشی از گیت‌های منطقی متصل بهم با مجموعه‌ای از ورودی‌ها و خروجی‌های است. در

هر لحظه از زمان، مقادیر دودویی خروجی‌ها تابعی از ترکیب دودویی ورودی‌هاست. بلاک دیاگرام مدار ترکیبی در شکل ۱-۱۵ دیده می‌شود.  $n$  متغیر دودویی ورودی از یک منبع خارجی سرچشمه گرفته و  $m$  متغیر دودویی خروجی هم به یک مقصد خارجی می‌روند و در بین این دو، اتصالات داخلی گیت‌های منطقی قرار دارد. یک مدار ترکیبی اطلاعات دودویی داده‌های ورودی را به داده‌های خروجی تبدیل می‌نماید. مدارهای ترکیبی در کامپیوتروهای دیجیتال به منظور تولید تضمینات کنترلی دودویی استفاده می‌شوند. همچنین بخش‌های لازم برای پردازش داده هم از این مدارات ساخته شده‌اند.

مدار ترکیبی بوسیله جدول درستی اش که رابطه دودویی بین  $n$  متغیر ورودی دودویی و  $m$  متغیر خروجی را نشان می‌دهد توصیف می‌شود. جدول درستی مقادیر دودویی خروجی را برای هر  $2^n$  ترکیب ورودی نشان می‌دهد. مدار ترکیبی می‌تواند بوسیله  $m$  تابع بولی، که هر یک متعلق به یک متغیر خروجی است، مشخص شود. هر تابع بر حسب  $n$  متغیر ورودی بیان می‌گردد.

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

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

۱- بیان مسئله

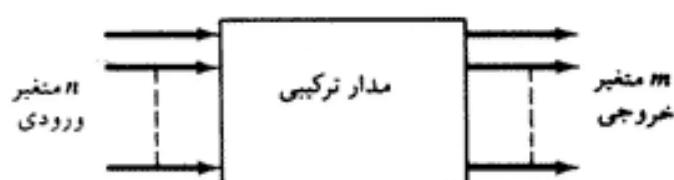
۲- اختصاص سمبول‌های حرفی به متغیرهای ورودی و خروجی

۳- بدست آوردن جدول درستی که رابطه بین ورودی‌ها و خروجی‌ها را تعریف می‌کند

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 می‌نماییم.



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

### نیم جمع کننده<sup>۱</sup>

اساسی‌ترین مدار محاسباتی دیجیتال، مدار جمع دو رقم دودویی است. مدار توکیبی 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<sup>۲</sup> 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 سمبول‌های  $x$  و  $y$  را به دو متغیر ورودی و  $S$  و  $C$ <sup>۳</sup> 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<sup>۴</sup> (پایین رتبه) مجموع می‌باشد. توابع بولی دو خروجی را می‌توان مستقیماً از جدول درستی بدست آورد.

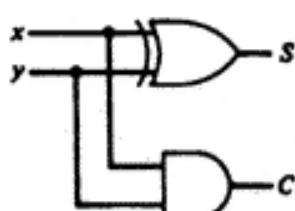
$$S = x'y + xy' = x \oplus y$$

$$C = xy$$

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 توکیبی 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 جمع کننده در جدول ۱-۲ نشان داده شده است. هشت سطر زیر متغیرهای خروجی از ترکیب‌های ممکنی را که متغیرها می‌توانند اختیار کنند مشخص می‌نمایند. مقادیر متغیرهای خروجی از حاصل جمع بیت‌های ورودی معین می‌شوند. وقتی همه بیت‌ها 0 باشند، خروجی  $S$  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 ۱-۱۷ برای یافتن عبارات جبری دو متغیر خروجی بکار می‌روند. ۱) ها در مربع‌های نقشه‌های  $S$  و  $C$  مستقیماً از میترم‌های جدول درستی تعیین می‌شوند. مربعات مربوط به خروجی  $S$  که در آن‌ها 1 ها 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 ۱-۲ ملاحظه شود). مربع‌هایی که برای  $C$  مقدار 1 دارند را می‌توان بطريق مختلف با هم ترکیب کرد. یک عبارت ممکن برای  $C$  بفرم زیر است:

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

نظر به رابطه  $y = x \oplus y$ ، دو عبارت بولی  $x'y + xy' = x \oplus y$  و  $S = x \oplus y \oplus z$

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

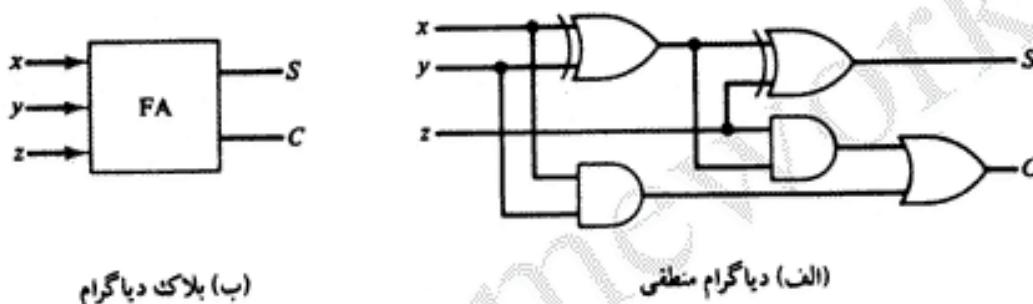
جدول ۱-۲ جدول درستی برای تمام جمع کننده

خرجی‌ها			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

$S = x'y'z' + x'yz' + xy'z' + xyz$   
 $= x \oplus y \oplus z$

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

شکل ۱-۱۷ نقشه های برای تمام جمع کننده



شکل ۱-۱۸ مدار تمام جمع کننده

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

## ۱-۶ فلیپ فلایپا<sup>۱</sup>

تاکنون مدارهای بررسی شده دیجیتال از نوع ترکیبی بودند، که در آنها خروجی ها در هر لحظه کلاً به ورودی های همان لحظه بستگی داشتند. هرچند هر سیستم دیجیتال احتمالاً یک مدار ترکیبی دارد، اغلب سیستم هایی که در عمل با آنها مواجه می شویم دارای عناصر حافظه نیز هستند و لذا سیستم باید در چهارچوب مدارهای ترتیبی مورد بررسی قرار گیرد. متداول ترین نوع مدار ترتیبی نوع همگام<sup>۲</sup> (همزمان یا سنکرون) آن است. مدارهای ترتیبی همگام در لحظه های گستته و معینی از زمان بر عناصر حافظه اثر می گذارند. همزمان سازی با یک وسیله زمانبندی یا یک مولد پالس ساعت<sup>۳</sup> حاصل می شود که رشته ای از پالس های ساعت را بطور پریودیک تولید می نماید. پالس های ساعت در سراسر سیستم توزیع می شوند و عناصر حافظه فقط با رسیدن پالس همگام کننده تأثیر می پذیرند. 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<sup>۱</sup> 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<sup>۲</sup> دودویی را در خود نگه میدارد 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.

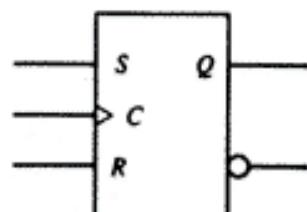
### فلیپ فلاپ SR

سمبل گرافیکی فلیپ فلاپ SR در شکل ۱-۱۹ (الف) دیده می شود. این فلیپ فلاپ دارای سه ورودی با نام های نشاندن<sup>۳</sup> (S) حذف کردن<sup>۴</sup> (R) و ساعت<sup>۵</sup> (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 به معنی دینامیک بودن (پویا بودن) ورودی است. سمبول دینامیکی نشانده شده 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 ۰ به ۱ تغییر نماید خروجی تحت تأثیر ورودی های S و R قرار خواهد گرفت. بهنگام تغییر C از ۰ به

S	R	$Q(t+1)$
0	0	$Q(t)$
0	1	0 پلاسخیر
1	0	1 پاک شدن، ۰
1	1	?

(ب) جدول مشخصه



(الف) سمبول گرافیکی

شکل ۱-۱۹ فلیپ فلاپ SR

- |          |          |        |
|----------|----------|--------|
| 1- Cell  | 2- State | 3- Set |
| 4- Reset | 5- Clock |        |

اگر  $S = 0$  و  $R = 0$  باشند در خروجی  $Q$  مقدار 1 نشانده می شود. اگر در تغییر  $C$  از 0 به 1 مقادیر  $S = 1$  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 ۱-۱۹ (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.  $(t) Q(t+1)$  نیز حالت خروجی  $Q$  در یک زمان معین است (که به آن حالت فعلی می گویند).  $(t+1) Q(t+1)$  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  $S = 1$  و  $R = 0$  باشد فلیپ فلاپ به حالت 0 (پاک شده) می رود. اگر  $S = 0$  و  $R = 1$  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.

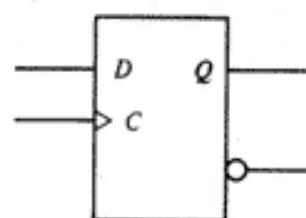
### فلیپ فلاپ $D^1$

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 سمبول  $D$  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 ۱-۲۰ نشان داده شده است. از جدول مزبور درمی یابیم که حالت بعدی  $Q(t+1) = Q(t) + D$  توسط ورودی  $D$  تعیین می شود. این ارتباط بوسیله معادله

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

(ب) جدول مشخصه



(الف) سمبل گرافیکی

شکل ۱-۲۰ فلیپ فلاپ  $D$

مشخصه زیر بیان می‌گردد

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

این بدان معنی است که خروجی  $Q$  فلیپ فلاب مقدار خود را، هر بار بهنگام انتقال سیگنال ساعت از ۰ به ۱ از ورودی  $D$  دریافت می‌کند.

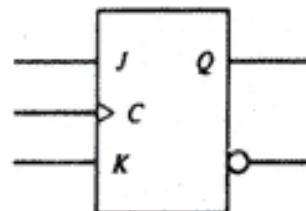
دقت کنید که حالتی از فلیپ فلاب وجود ندارد که خروجی را تغییر ندهد. هرچند فلیپ فلاب  $D$  از مزیت داشتن فقط یک ورودی (بدون احتساب  $C$ ) بهره می‌برد ولی عیب آن این است که جدول مشخصه آن فاقد حالت " بلا تغییر " $Q(t+1) = Q(t)$  می‌باشد. در این فلیپ فلاب حالت بلا تغییر بدین صورت انجام می‌شود که یا باید سیگنال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

### فلیپ فلاب JK

یک فلیپ فلاب JK نوع اصلاح شده فلیپ فلاب SR است که حالت "نامعین"<sup>۳</sup> در فلیپ فلاب SR در آن "معین" شده است. ورودی‌های  $J$  و  $K$  در نشاندن و یا پاک کردن فلیپ فلاب بترتیب مانند  $S$  و  $R$  عمل می‌کنند. وقتی هر دو ورودی  $J$  و  $K$  برابر ۱ باشند، انتقال یا گذار پالس ساعت خروجی‌های فلیپ فلاب را به حالت متمم‌شان سوییج می‌کنند.  
سمبل گرافیکی و جدول مشخصه فلیپ فلاب 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 فلاب SR است. در فلیپ فلاب JK اگر هر دو ورودی  $J$  و  $K$  برابر ۱ باشند، بجای حالت نامعین، حالت متمم  $Q(t+1) = Q'(t)$  را خواهیم داشت.

$J$	$K$	$Q(t+1)$
۰	۰	$Q(t)$ بلا تغییر
۰	۱	۰ پاک شدن، ۰
۱	۰	۱ نشانده شدن، ۱
۱	۱	$Q'(t)$ نم

(ب) جدول مشخصه



(الف) سمبل گرافیکی

شکل ۱-۲۱ فلیپ فلاب JK

1- Unchanged

2- Feedback

3- Indeterminate

**فلیپ فلاب T<sup>1</sup>**

نوع دیگری از فلیپ فلابها که در کتب درسی دیده می شود فلیپ فلاب T می باشد. این فلیپ فلاب، که در شکل ۱-۲۲ نشان داده شده است از فلیپ فلاب 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 فلاب T فقط دو حالت دارد. وقتی ۰ T = باشد (J = K = 0) یک گذار ساعت حالت فلیپ فلاب را تغییر نمی دهد. وقتی ۱ T = باشد (J = K = 1) انتقال ساعت حالت فلیپ فلاب را متمم می کند. این حالات را می توان با معادله مشخصه زیر بیان کرد

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

**فلیپ فلاب های تحریک شونده بالبه<sup>۲</sup>**

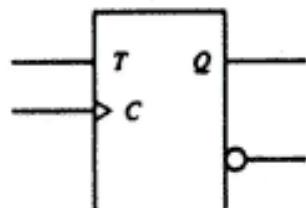
متداول ترین نوع فلیپ فلاب های مورد استفاده در همگام کردن تغییر حالت به هنگام تغییر وضعیت یا گذار پالس ساعت فلیپ فلاب تحریک شونده بالبه (یا فلیپ فلاب حساس به لبه) می باشد. در این نوع فلیپ فلاب ها، تغییرات در خروجی بازه سطح خاصی از ورودی پالس ساعت رخ می دهد. وقتی که سطح ورودی پالس ساعت از این سطح آستانه<sup>۳</sup> تجاوز کند ورودی ها قفل می شوند بطوری که فلیپ فلاب به تغییرات دیگر در ورودی، تا بازگشت پالس ساعت به ۰ و ظهرور پالس ساعتی دیگر، پاسخ نمی دهد. در بعضی از این نوع فلیپ فلاب 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 فلاب D حساس به لبه مثبت نشان

T	$Q(t+1)$
۰	$Q(t)$
۱	$Q'(t)$

بلا تغییر  
متمن

(ب) جدول مشخصه

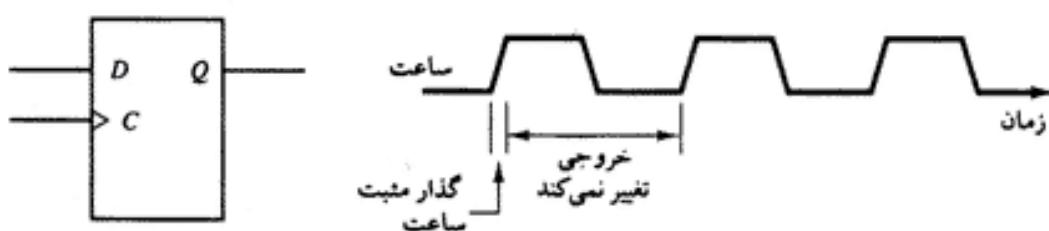


(الف) سیگنال گرافیکی

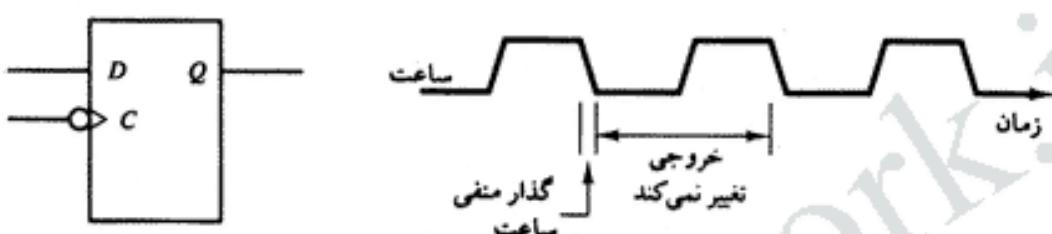
شکل ۱-۲۲ فلیپ فلاب T

- 1- Toggle  
3- Threshold level

- 2- Edge- Triggered Flip Flop



(الف) فلیپ فلاب D تحریک شونده باله مثبت



(ب) فلیپ فلاب D تحریک شونده باله منفی

شکل ۱-۲۳ (الف) فلیپ فلاب تحریک شونده باله

می‌دهد. مقدار ورودی  $D$  در لبه مثبت پالس ساعت به خروجی  $Q$  منتقل می‌شود. هنگامی که پالس ساعت در سطح ۰، در سطح ۱، در حال انتقال از سطح ۰ به ۱ باشد خروجی تغییری نمی‌کند. گذار یا انتقال مثبت پالس شامل دو فاصله زمانی است که یکی حداقل زمان آماده سازی<sup>۱</sup> می‌باشد 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<sup>۲</sup> 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) سمبل‌های گرافیکی و دیاگرام زمانی مربوط به فلیپ فلاب D حساس به لبه منفی را نشان می‌دهد. در سمبل گرافیکی، یک دایره کوچک منفی کننده<sup>۳</sup> در جلو علامت دینامیکی در ورودی  $C$  قرار دارد. این علامت بیانگر تحریک فلیپ فلاب در لبه منفی پالس است. در این حالت پاسخ فلیپ فلاب در قبال انتقال سطح سیگنال ساعت از سطح ۱ به ۰ می‌باشد.

نوع دیگری از فلیپ فلاب که در بعضی از سیستم‌ها بکار می‌رود فلیپ فلاب ماستر-اسلیو<sup>۴</sup> (یعنی حاکم و پیرو) است. این نوع مدار از دو فلیپ فلاب تشکیل شده است. اولین فلیپ فلاب ماستر نامیده می‌شود و به سطح مثبت پالس ساعت پاسخ می‌دهد، و دومین فلیپ فلاب اسلیو می‌باشد که به سطح منفی پالس ساعت واکنش نشان می‌دهد. در نتیجه خروجی بهنگام تغییر ۱ به ۰ 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 نشاندن<sup>۱</sup> و پاک کردن<sup>۲</sup> غیر همگام<sup>۳</sup> می باشند. این ورودی ها معمولاً نشاندن اولیه<sup>۴</sup> 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 بر حسب 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 جدول تحریک چهار نوع فلیپ فلاپ را نشان می دهد. هر جدول مشکل از دو ستون  $(t)$   $Q$  و  $(t+1)$   $Q$  و یک ستون برای هر ورودی است که چگونگی وقوع انتقال مورد نظر را نشان می دهد. انتقال از حالت فعلی  $(t)$   $Q$  به چهار طریق امکان پذیر است.

جدول ۱-۳ جدول تحریک برای چهار فلیپ فلاپ

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

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

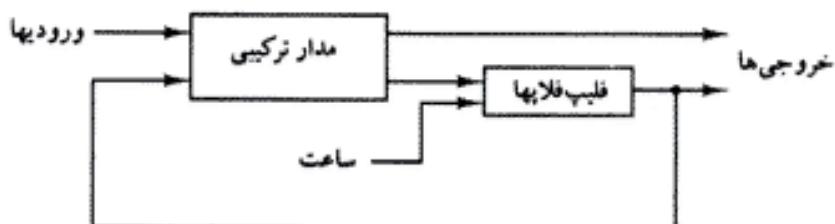
4- Preset

ورودی های مورد نیاز برای هر یک از چهار انتقال، از اطلاعات موجود در جدول مشخصه بدست آمده اند. سمبول X در جداول نشانده شده حالت بی اهمیت است و باین معنی است که 0 یا 1 بودن ورودی مهم نیست.

دلیل وجود حالات بی اهمیت 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 می توان با قرار دادن J و K در حالت 0 (برای حالت بلا تغییر) یا با قرار دادن  $0 = J = 1 = K$  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 علامت می زنیم و لذا طراح اجازه خواهد داشت مقدار K را با توجه به اینکه کدام مناسبتر است، 1 یا 0 انتخاب کنند.

## ۷-۱ مدارهای ترتیبی

مدار ترتیبی از اتصال مجموعه ای از فلیپ فلاپها و گیت ها بوجود می آید. گیت ها به تنها یی مدارهای ترکیبی را تشکیل می دهند ولی وقتی با فلیپ فلاپها همراه باشند مدار حاصل از رده مدارهای ترتیبی خواهد بود. بلوك دیاگرام یک مدار ترتیبی ساعت دار در شکل ۱-۲۴ نشان داده شده است. این مدار از یک مدار ترکیبی و تعدادی فلیپ فلاپ ساعت دار تشکیل شده است. بطور کلی، هر تعداد یا هر نوع فلیپ فلاپ می تواند در مدار بکار رود. همانطور که در دیاگرام نشان داده شده است، بلوك مدار ترکیبی سبکنالهای دودویی را از ورودی های خارجی و از خروجی های فلیپ فلاپها دریافت می کند. خروجی های مدار ترکیبی به خروجی های خارجی و به ورودی های فلیپ فلاپها می روند. گیت های تشکیل دهنده مدار ترکیبی مقادیر ذخیره شونده در فلیپ فلاپها را بعد از هر گذار پالس ساعت معین می کنند. خروجی فلیپ فلاپها به نوبه خود به ورودی های مدار ترکیبی اعمال شده و رفتار مدار را تعیین می کنند. این فرآیند نشان می دهد که خروجی های خارجی مدار ترتیبی تابعی از ورودی های خارجی و حالت فعلی<sup>۱</sup> فلیپ فلاپها است. بعلاوه، حالت بعدی<sup>۲</sup> فلیپ فلاپها نیز تابعی از حالت فعلی آنها و ورودی های خارجی



شکل ۱-۲۴ ۱- بلوك دیاگرام یک مدار ترتیبی همگام ساعت دار

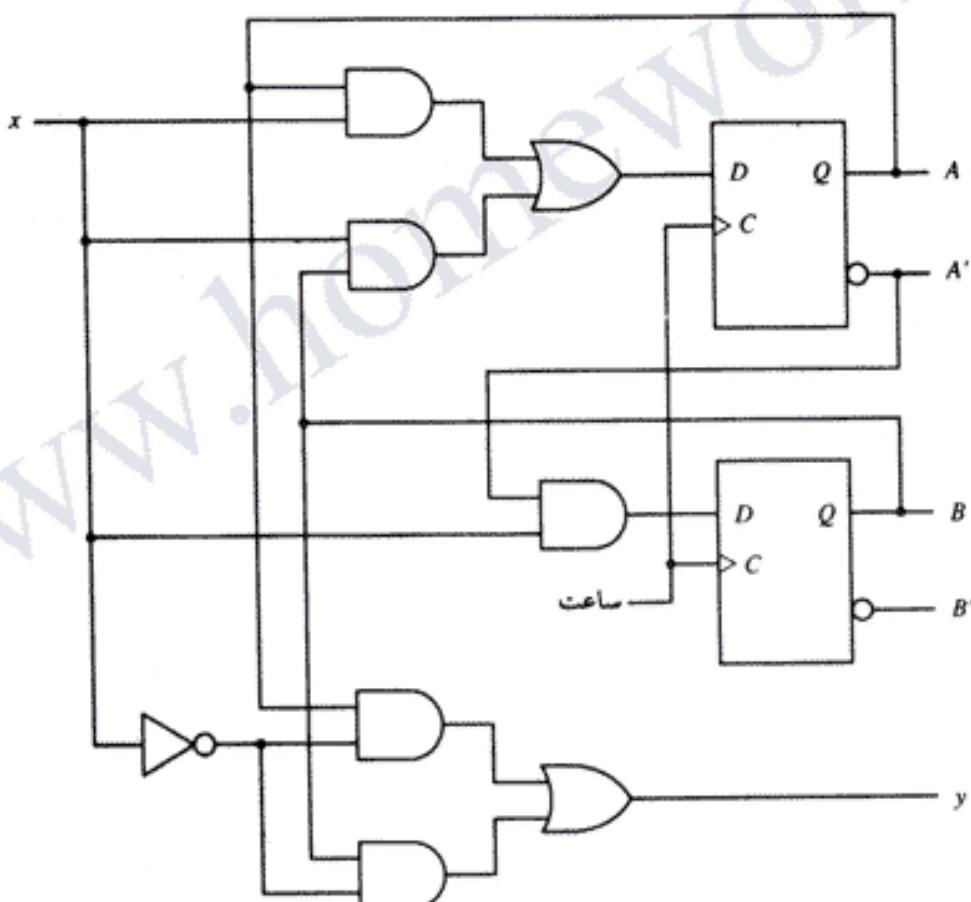
1- Present State

2- Next State

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

### معادلات ورودی فلیپ فلاپها<sup>۱</sup>

مثالی از یک مدار ترتیبی در شکل ۱-۲۵ دیده می شود. این مدار دارای یک متغیر ورودی  $X$ ، یک متغیر خروجی  $A$  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 توانند 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 سمبول ورودی هر فلیپ فلاپ را بعنوان متغیر معادله ورودی و اندیس را برای مشخص کردن سمبول



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  $D_A$  و  $D_B$  مشخص می‌شوند. اولین حرف در هر مورد نماینده ورودی D یک فلیپ فلاپ D و حرف اندیس نام فلیپ فلاپ مربوط 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 و x 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<sup>۱</sup>

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 ساعت دار، 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 عنوانین 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 ۱ نشان می‌دهد. بخش ورودی مقداری از x را در هر حالت فعلی معین می‌سازد. بخش حالت بعدی حالات فلیپ فلاپ‌ها را در پریود زمانی بعدی، ۱+۱، بدست می‌دهد. بخش خروجی نیز مقدار لازما در حالت فعلی و وضعیت ورودی می‌دهد.

#### ۱- State Table

جدول ۱-۴ جدول حالت برای مدار شکل ۱-۲۵

حالت فعلی		ورودی	حالت بعدی		خروجی
A	B		A	B	
		x			y
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 ۰۰۰ تا ۱۱۱ 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 می‌یابند. بنابراین هرگاه حالت فعلی و ورودی شرایط  $D_A = 1$  یا  $Bx = 1$  را ایجاد کنند  $D_A$  برابر ۱ و در نتیجه حالت بعدی A هم ۱ خواهد شد. این مطلب در جدول حالت با سه عدد ۱ 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 ۱ است که A برابر ۰ و x برابر ۱ باشد. ستون خروجی نیز از معادله خروجی زیر بدست می‌آید.

$$y = Ax' + Bx'$$

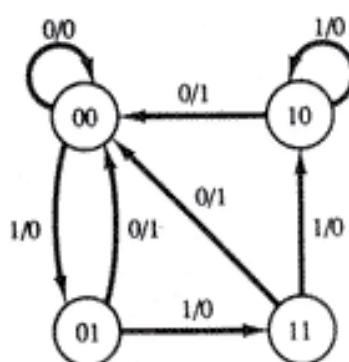
جدول حالت هر مدار ترتیبی با استفاده از روشی که در این مثال دیده شد بدست می‌آید. بطور کلی، یک مدار ترتیبی m فلیپ فلاپ، n متغیر ورودی و p متغیر خروجی دارای m ستون حالت فعلی، n ستون برای ورودی‌ها، m ستون برای حالت بعدی و p ستون برای خروجی‌هاست. ستون‌های حالت فعلی و ورودی با هم ترکیب شده و ما  $2^{m+n}$  ترکیب دودویی، از ۰ 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 می‌کنند بدست می‌آیند.

### دیاگرام حالت

اطلاعات موجود در جدول حالت را می‌توان 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 دایر را بهم وصل می‌کنند نشان داده می‌شود. دیاگرام حالت مدار ترتیبی شکل ۱-۲۵ در شکل ۱-۲۶ نشان داده شده است. دیاگرام حالت همان اطلاعات جدول حالت را در دسترس می‌گذارد. و مستقیماً از جدول ۱-۴ بدست می‌آید. عدد دودویی داخل هر دایره حالت فلیپ فلاپ‌ها را مشخص می‌کند. درکنار خطوط جهت دار دو عدد دودویی که با ممیز از هم جدا شده‌اند نوشته شده است. اولین عدد مقدار ورودی را در حالت فعلی و عدد بعد از ممیز، خروجی را در همان حالت نشان می‌دهد. مثلاً کنار خط جهت داری که حالت ۰۰ را به ۰۱ وصل می‌کند ۰/۱ نوشته شده است، و بدان معنی است که وقتی مدار ترتیبی در حالت ۰۰ است بازه ورودی ۱، خروجی ۰ خواهد بود. پس از اعمال یک پالس ساعت، مدار به حالت بعدی ۰۱ می‌رود. همین اعمال پالس ساعت ممکن است ورودی را عرض کند. اگر ورودی ۰ شود خروجی ۱ شده و لی اگر ورودی در ۱ باقی بماند خروجی در ۰ خواهد ماند. این اطلاعات از دیاگرام حالت و با دنبال کردن دو خط جهت داری که به دایره ۰۱ وصل شده‌اند حاصل می‌گردد. خط جهت داری که دایره را به خودش وصل می‌کند نشان می‌دهد 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 آسان‌تر است و نمودار حالت مستقیماً از جدول حالت بدست می‌آید. دیاگرام حالت یک دید مصور را از انتقال حالات فراهم می‌نماید و بشکلی است که برای تفسیر عملکرد مدار مناسب‌تر می‌باشد. مثلاً دیاگرام حالت شکل ۱-۲۶ بوضوح نشان می‌دهد که با شروع حالت از ۰۰ مدامی که ورودی ۱ است خروجی ۰ خواهد بود. اولین ورودی ۰ پس از رشته‌ای از ۱ ها، خروجی ۱ تولید خواهد کرد و مدار را به حالت اولیه ۰۰ بر می‌گردد.



شکل ۱-۲۶ دیاگرام حالت یک مدار ترتیبی

### مثال طراحی

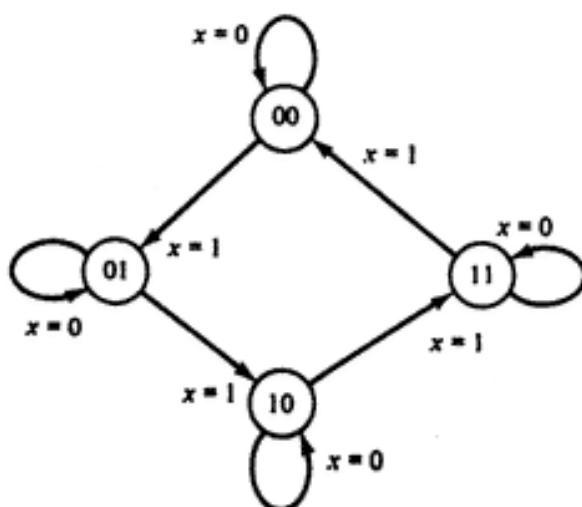
اکنون روش طراحی مدارهای ترتیبی بوسیله ارائه یک مثال خاص نشان داده می شود. در روش طراحی ابتدا مشخصات مدار را به یک دیاگرام حالت تبدیل می کنیم. سپس دیاگرام حالت به یک جدول حالت تبدیل می شود. بالاخره از جدول حالت برای تهیه دیاگرام مدار منطقی استفاده می شود.

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 0 =  $x$  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 مدامی 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 قبل اس梅بل  $x$  را به متغیر ورودی اختصاص دادیم. حال س梅بل 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 ۱-۵ جدول حالت را بوجود می آورند. واردہ های این جدول مستقیماً از دیاگرام حالت بدست می آیند.

جدول تحریک یک مدار ترتیبی نوع تعیین یافته یک جدول حالت است. این تعیین 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. ورودی های 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 ۰ به حالت بعدی ۰ در فلیپ فلاپ A داریم. در جدول ۱-۳ ۱ 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 ۰ 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 ۰ به حالت بعدی ۰ نشان می دهد، لذا مجدداً ۰ و X در سطر اول و در زیر  $J_B$  و  $K_B$  نوشته شده اند. دو میں سطر جدول ۱-۵ انتقال فلیپ فلاپ B را از حالت فعلی ۶ به حالت بعدی ۱ نشان می دهد. با توجه به جدول ۱-۳ ۱ می بینیم که انتقال از  $Q(t)=0$  به  $Q(t+1)=1$  مستلزم ورودی  $J=1$  و  $K=X$  است. بنابراین ۱ و ۰ 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 تشیکل شده است. با توجه به شکل ۱-۲۴ می بینیم که خروجی های مدار ترتیبی باید به چهار ورودی فلیپ فلاپها یعنی  $J_A$ ،  $J_B$ ،  $K_A$  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. وارد هایی که ورودی های مدار ترکیبی را

جدول ۱-۵ جدول تحریک برای شمارنده دودویی

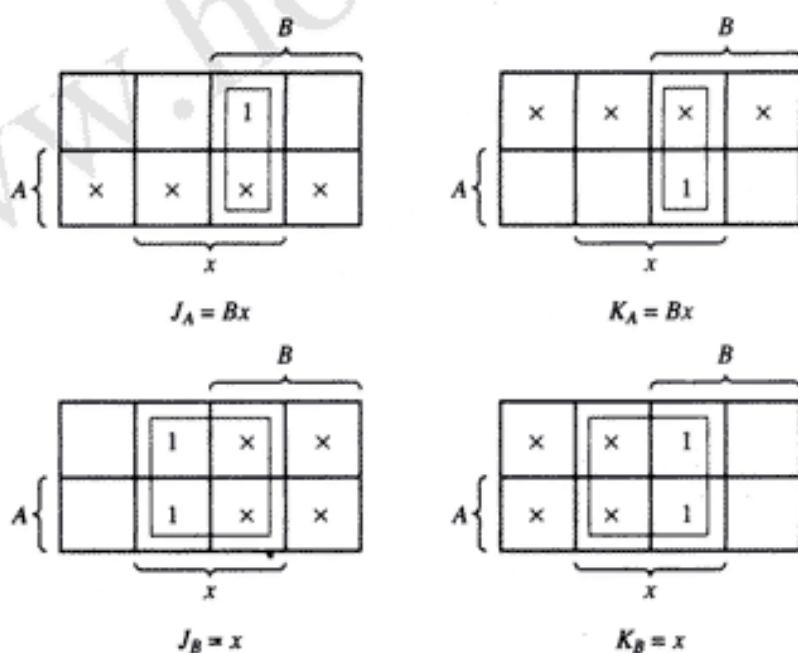
		حالت فعلی		ورودی		حالت بعدی		ورودی های فلیپ فلاپ			
A	B	x		A	B	$J_A$	$K_A$	$J_B$	$K_B$		
0	0	0		0	0	0	x	0	x		
0	0	1		0	1	0	x	1	x		
0	1	0		0	1	0	x	x	0		
0	1	1		1	0	1	x	x	1		
1	0	0		1	0	x	0	0	x		
1	0	1		1	1	x	0	1	x		
1	1	0		1	1	x	0	x	0		
1	1	1		0	0	x	1	x	1		

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

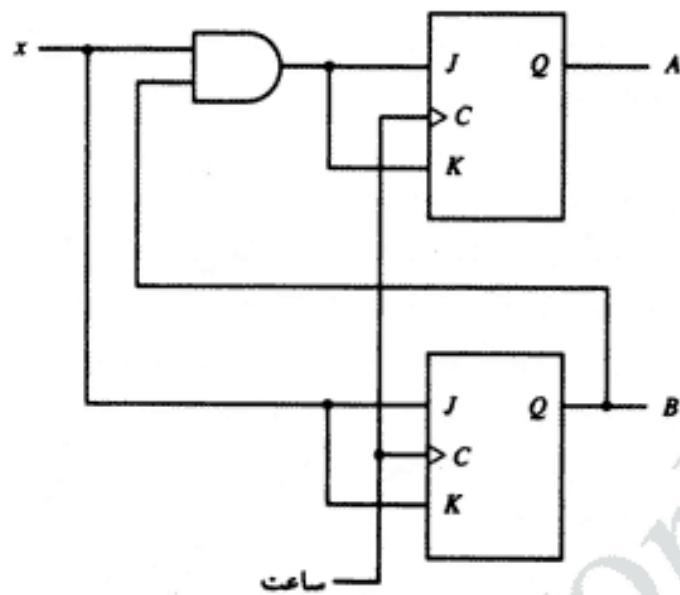
اکنون توابع بولی ساده شده در مدار ترکیبی را می‌توان بدست آورد. ورودیها متغیرهای  $x$ ،  $A$  و  $B$  می‌باشند. خروجی‌ها نیز متغیرهای  $J_A$ ،  $J_B$ ،  $K_A$  و  $K_B$  هستند. اطلاعات حاصل از جدول تحریک به نقشه‌های شکل ۱-۲۸ انتقال یافته و چهار معادله ورودی ساده شده فلیپ فلاپ‌ها از آن بدست آمده است

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

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 ورودی‌های  $J$  و  $K$  حالت بعدی شمارنده را پس از وقوع پالس ساعت معین می‌کنند. اگر هر دو ورودی  $J$  و  $K$  برابر ۰ باشند پالس ساعت تأثیری نخواهد داشت. یعنی حالت فلاپ‌ها تغییری نمی‌کند. بنابراین وقتی  $0 = x$  باشد، تمام ورودی‌های فلاپ فلاپ‌ها برابر ۰ و حالت فلاپ‌ها بالاتغیر باقی می‌مانند حتی اگر پالسهای ساعت مرتبأ به مدار اعمال شوند.



شکل ۱-۲۸ نقشه‌های مدار ترکیبی شمارنده



شکل ۱-۲۹ ۱ دیاگرام منطقی یک شمارنده دو بیتی

### روش طراحی

طراحی مدارهای ترتیبی رئوس 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 و اصل 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 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 ۱-۳ 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 نمائیم. 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 نمائیم.

### مسائل

۱-۱ با استفاده از جدول درستی صحت ثوری دمودرگان 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:

(ب)  $AB + AB'$

(الف)  $A + AB$

(د)  $A'B + ABC' + ABC$

(ج)  $A'BC + AC$

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

(ب)  $(BC' + A'D)(AB' + CD')$

(الف)  $AB + A(CD + CD')$

۱-۵ با استفاده از ثوری دمودرگان نشان دهید که:

(ب)  $A + A'B + A'B' = 1$

(الف)  $(A + B)'(A' + B')' = 0$

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

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

(ج) نشان دهید که  $F + F' = 1$

(ب) نشان دهید که  $FF' = 0$

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 گیت‌ها را با دیاگرام بخش (ب) مقایسه کنید.

تابع بولی زیر را با استفاده از نقشه سه متغیره ساده کنید. ۱-۸

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

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

تابع بولی زیر را با استفاده از نقشه چهار متغیره ساده کنید: ۱-۹

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

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

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

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

عبارات زیر را ۱) بصورت جمع حاصلضربها و ۲) بصورت ضرب حاصل جمعها ساده کنید: ۱-۱۰

$$(الف) x'z' + y'z' + yz' + xy$$

$$(ب) AC' + B'D + A'CD + ABCD$$

تابع بولی زیر را به شکل جمع حاصلضربها با استفاده از نقشه چهار متغیره ساده کنید. دیاگرام منطقی را، الف) با گیت‌های AND-OR؛ ب) با گیت‌های NAND رسم کنید. ۱-۱۱

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

تابع بولی زیر را به شکل ضرب حاصل جمعها با استفاده از نقشه چهار متغیره ساده کنید. دیاگرام منطقی را، الف) با گیت‌های OR-AND؛ ب) با گیت‌های NOR رسم کنید. ۱-۱۲

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

تابع بولی F را همراه با حالات بی اهمیت d در فرم ۱) جمع حاصلضربها و ۲) ضرب حاصل جمعها ساده کنید. ۱-۱۳

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

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

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 دستکاری های 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 ۱ بیشتر از ورودی های ۰ باشند خروجی ۱ 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 ۰، ۱، ۲ یا ۳ باشد خروجی یکی بیشتر از ورودی است. وقتی ورودی ها ۴، ۵، ۶ و ۷ 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  $z$  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 برباکنید.

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  $0 = x$  است حالت فلیپ فلاپ تغییر نمی کند. وقتی  $1 = x$  باشد، ترتیب حالات ۱۱، ۱۰، ۰۱، ۰۰ و ۱۱ و تکرار آن خواهد بود.

۱-۲۱ یک مدار ترتیبی با دو فلیپ فلاپ JK و دو ورودی E و X طراحی کنید. اگر  $0 = E$  باشد، مدار بدون توجه به مقدار X در همان حالت باقی می ماند. وقتی  $1 = E$  و  $0 = x$  است مدار به یک سری حالات از ۰۰ به ۰۱، به ۱۰، به ۱۱ و ۰۰ و تکراری از آن وارد می شود. وقتی  $1 = E$  و  $x = 0$  باشد مدار به حالات انتقالی ۰۰ به ۱۱، به ۱۰، به ۰۱ و ۰۰ و تکراری از آن وارد می شود.



## قطعات دیجیتال

۲-۱ مدارهای مجتمع

۲-۲ دیکدرها

۲-۳ مولتی پلکسراها

۲-۴ ثبات‌ها

۲-۵ شیفت رجیسترها

۲-۶ شمارندهای دودویی

۲-۷ واحد حافظه

### ۲-۱ مدارهای مجتمع<sup>۱</sup>

مدارهای دیجیتال با مدارهای مجتمع ساخته می‌شوند. یک مدار مجتمع (یا آی سی)، یک گریستال کوچک نیمه هادی بنام تراشه<sup>۲</sup> است که قطعات الکترونیکی را برای گیت‌های دیجیتال در خود دارد. اتصالات داخل تراشه، مدار مورد نیاز را بوجود می‌آورد. تراشه در داخل یک محفظه پلاستیک و یا سرامیک جاسازی می‌شود و اتصالات آن با سیم‌های طلایی نازک به پایه‌های خارجی جوش داده می‌شود تا مدار مجتمع بوجود آید. تعداد پایه‌ها ممکن است از ۱۴ پایه در بسته‌های کوچک تا ۱۰۰ پایه یا بیشتر در بسته‌های بزرگتر تغییر کند. هر مدار مجتمع یا آی سی دارای یک مشخصه عددی است که روی سطح بسته‌بندی آن برای شناسایی چاپ می‌شود. هر سازنده یک کتابچه راهنمای کاتالوگ با شرح دقیق و تمام اطلاعات لازم درباره آی سی های ساخت خود را چاپ می‌کند.

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<sup>۱</sup> (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<sup>۲</sup> (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<sup>۳</sup> (LSI) بین ۲۰۰ تا چند هزار گیت در هر بسته دارند. این بسته‌ها سیستم‌های دیجیتالی همچون پردازنده‌ها، تراشه‌های حافظه و مازول‌های قابل برنامه‌ریزی را شامل می‌شوند. قطعات مجتمع با مقیاس بسیار بزرگ<sup>۴</sup> (VLSI) حاوی هزاران گیت در یک بسته‌اند. مثالهایی از این گروه عبارتند از، آرایه‌های بزرگ حافظه و تراشه‌های پیچیده ریز کامپیوترها. بدلیل کوچکی و ارزانی انقلابی در تکنولوژی ساخت سیستم‌های کامپیوتری بوجود آورده و به طراحان امکان ساخت و ایجاد ساختارهایی را دادند که قبل 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<sup>۵</sup> می‌خوانند. هر خانواده منطقی، مدار الکترونیکی پایه خاصی را دارد که مدارها و توابع دیجیتال پیچیده‌تر براساس آن تهیه می‌شوند. مدار پایه در هر تکنولوژی یک گیت NOR، NAND و یا معکوس کننده است. در نام گذاری 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. متدائل‌ترین خانواده‌ها در زیر معرفی شده‌اند.

TTL - منطق ترانزیستور - ترانزیستور<sup>۶</sup>

ECL - منطق کوپل امپت<sup>۷</sup>

MOS - منطق فلز - اکسید - نیمه هادی<sup>۸</sup>

CMOS - منطق فلز - اکسید - نیمه هادی مکمل<sup>۹</sup>

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 می‌شود. ECL در سیستم‌هایی که به سرعت عمل بالا نیاز دارند ترجیح داده می‌شود. MOS برای مدارهایی که نیاز به تراکم بالا دارند مناسب است، و CMOS در سیستم‌های کم مصرف بکار می‌رود.

خانواده منطقی ترانزیستور - ترانزیستور گونه تکامل یافته تکنولوژی قدیمی‌تری است که در آن از دیود و ترانزیستور برای ساخت گیت پایه NAND استفاده می‌شده است. این تکنولوژی منطق دیود - ترانزیستور<sup>۱</sup> (DTL) خوانده می‌شده است. بعدها برای بهبود عملکرد مدار بجای دیود از ترانزیستور استفاده شد و نام خانواده جدید، منطق ترانزیستور - ترانزیستور گذاشته شد. بهمین 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،<sup>۲</sup> TTL توان پائین<sup>۳</sup> (یا کم مصرف)، TTL شوتکی<sup>۴</sup>، TTL شوتکی توان پائین<sup>۵</sup> و TTL شوتکی پیشرفته<sup>۶</sup>. منبع تغذیه مدارهای TTL، ۵ ولت است و دو سطح منطقی تقریباً ۰ و ۳/۵ ولت می‌باشد.

خانواده کوپل امپیتر سریع ترین مدارهای دیجیتال را بقلم مجتمع در اختیار می‌گذارد. ECL در مدارهایی مانند سوپر کامپیوترها و پردازنده‌های سیگنال که در آنها سرعت بالا ضرورت دارد، بکار می‌رود. ترانزیستورها در گیت‌های ECL در حالت غیر اشباع کار می‌کنند و رسیدن به تأخیرهای انتشاری در حد ۱ تا ۲ نانو ثانیه در آنها می‌سرّ است.

منطق فلز - اکسید - نیمه‌هادی (MOS) یک ترانزیستور تک قطبی است که به جریان یک نوع حامل الکتریکی وابسته است. این حامل‌ها ممکن است الکترون (در نوع کانال n) یا حفره (در نوع کانال p) باشند. این، برخلاف ترانزیستور دوقطبی بکار رفته در گیت‌های 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 می‌نامند. معمولاً در مدارهایی که فقط یک نوع ترانزیستور MOS وجود دارد از NMOS استفاده می‌شود. در تکنولوژی CMOS هردو نوع ترانزیستور PMOS و NMOS که بشکل مکمل در تمام مدارها بسته شده‌اند بکار رفته است. بزرگترین مزیت CMOS نسبت به دو قطبی تراکم بالای مدارها در بسته‌بندی، ساده بودن تکنیک ساخت و عملکرد مقرون به صرف آن بدلیل مصرف توان کم است.

بعثت مزایای بی‌شمار، مدارهای مجتمع انحصاراً در تهیه انواع قطعات لازم در طراحی سیستم‌های کامپیوتر به کار می‌رود. 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 مجموعه‌ای از واحدهای عملیاتی دیجیتال را فراهم می‌کنند که در طراحی کامپیوترهای دیجیتال بعنوان بلاک‌های ساختمانی اصلی (پایه) به کار می‌روند.

1- Diode - Transistor Logic

2- High Speed TTL

3- Low Power TTL

4- Schottky TTL

5- Low - Power Schottky TTL

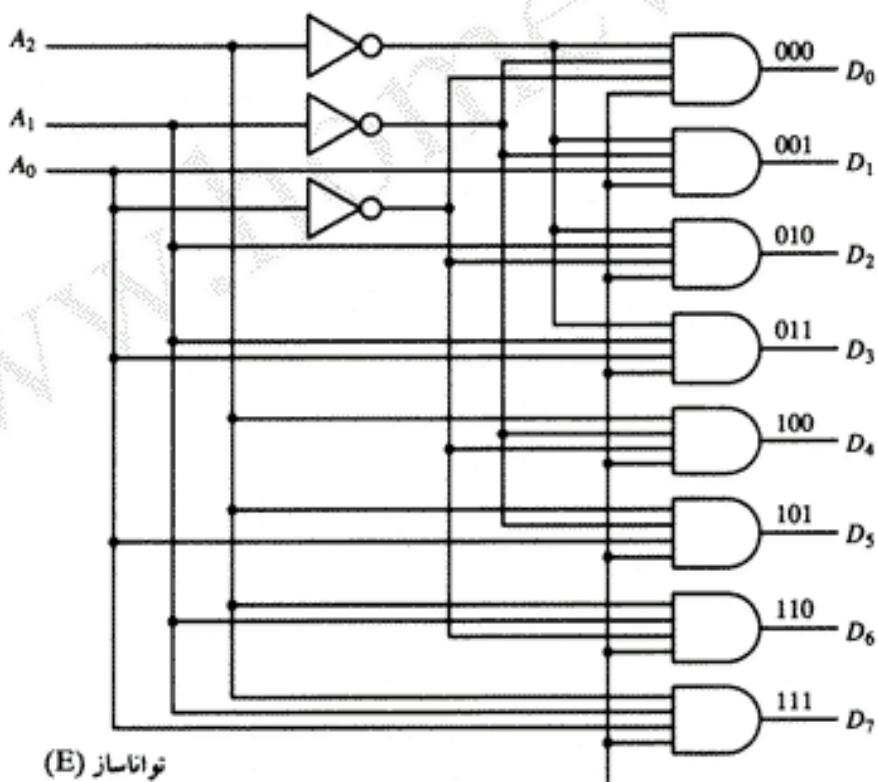
6- Advanced Schottky TTL.

## ۲-۲ دیکدرها<sup>۱</sup>

کمیت‌های اطلاعاتی گستته در سیستم‌های کامپیوترا دیجیتال با کدهای دودویی نشان داده می‌شوند. یک کد دودویی  $n$  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 حد اکثر  $2^n$  خط خروجی مستقل منحصر بفرد تبدیل می‌کند. اگر اطلاعات دیکد شده  $n$  بیتی ترکیبات بلااستفاده یا بی‌همیت داشته باشد در اینصورت دیکدر دارای خروجی‌های کمتر از  $2^n$  خواهد بود.

دیکدرهایی که در این بخش مورد بحث قرار گرفته‌اند 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 می‌شود.

دیاگرام منطقی یک دیکدر 3 به 8 در شکل ۲-۱ نشان داده شده است. سه ورودی داده  $A_0$ ,  $A_1$  و  $A_2$  به هشت خروجی داده تبدیل شده‌اند که هر خروجی یکی از ترکیبات سه متغیر ورودی دودویی را نشان



شکل ۲-۱ دیکدر 3 به 8 خطی

1- Decoder

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 هشتایی<sup>۱</sup> 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 ۳ به ۸ 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 تواناسازی<sup>۲</sup> (فعال سازی) برای کنترل کار مدار هستند. دیکدر شکل ۱-۲ دارای یک ورودی تواناساز، E می باشد. دیکدر هنگامی فعال شده است که E برابر ۱ باشد و اگر E مساوی ۰ بود دیکدر ناتوان (غیرفعال) شده است. عملکرد دیکدر با استفاده از یک جدول درستی که در جدول ۱-۲ دیده می شود آشکارتر خواهد شد. هر گاه ورودی تواناساز E برابر ۰ باشد، تمام خروجی ها، بدون توجه به مقادیر سه ورودی داده، ۰ خواهند شد. سه علامت × در جدول حالات بی اهمیت ورودی ها را نشان می دهند. اگر ورودی تواناساز، ۱ شود، دیکدر در وضع عادی خودکار خواهد کرد. برای هر ترکیب ممکن ورودی، هفت خروج ۰ و فقط یک خروجی ۱ وجود دارد که بیانگر عدد هشتایی معادل عدد دودویی در خطوط ورودی داده است.

### دیکدر با گیت NAND

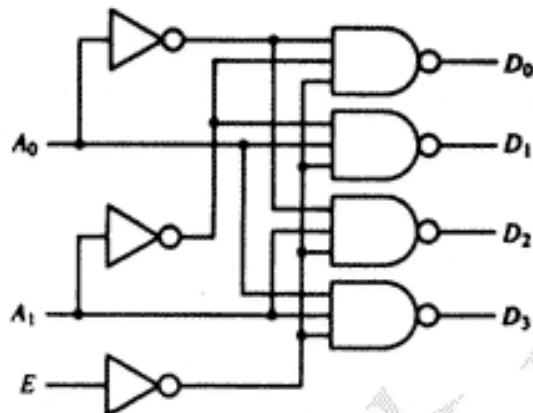
بعضی از دیکدرها بجای گیت AND با گیت NAND ساخته می شوند. چون گیت NAND عمل گیت AND را با خروجی معکوس شده انجام می دهد، تولید خروجی های معکوس شده با دیکدر مفروض به صرفه تر است. یک دیکدر ۲ به ۴ همراه با ورودی توانا ساز که از گیت های NAND ساخته شده است در شکل ۲-۲ نشان داده شده است. مدار دارای خروجی های متمم بوده و با ورودی تواناساز متمم شده کار می کند. دیکدر هنگامی تواناست که E برابر ۰ شود. همانطور که از جدول درستی

جدول ۱-۲ جدول درستی برای دیکدر ۳ به ۸ خطی

تواناساز	ورودی ها			خروجی ها								
	E	A <sub>2</sub>	A <sub>1</sub>	A <sub>0</sub>	D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
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	0	1	0
1	0	1	0	0	0	0	0	0	0	1	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

$E$	$A_1$	$A_0$	$D_0$	$D_1$	$D_2$	$D_3$
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

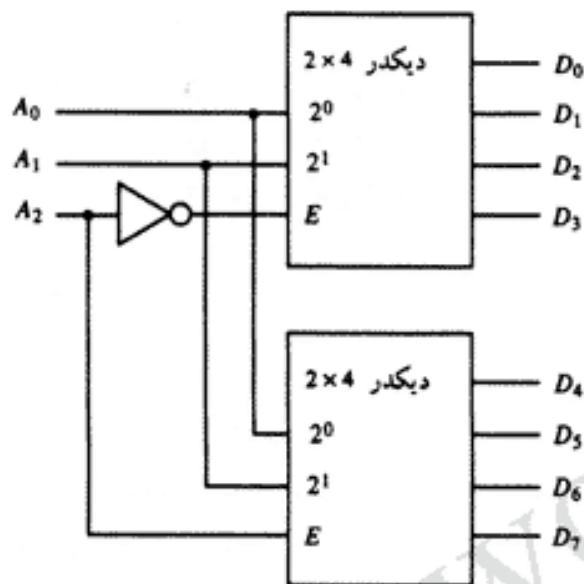
شکل ۲-۲ دیکدر ۲ به ۴ خطی با گیت‌های NAND

پیداست، تنهادر هر لحظه از زمان فقط یک خروجی برابر 1، و بقیه سه خروجی دیگر برابر 0 هستند. خروجی که نماینده عدد دودویی واقع در ورودی‌های  $A_0$  و  $A_1$  است دارای مقدار 0 است. هر گاه  $E$  برابر 1 باشد، مدار صرف نظر از مقدار دودویی ورودی‌های دیگر غیرفعال خواهد بود. وقتی که مدار غیرفعال است، هیچ یک از خروجی‌ها انتخاب نمی‌شود و تمام خروجی‌ها برابر 1 خواهند بود. بطور کلی، یک دیکدر ممکن است با خروجی‌های 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 می‌آید که دیکدری با اندازه مشخصی مورد نیاز است ولی تنها اندازه کوچکتر آن موجود است در چنین موقعی می‌توان دو یا چند دیکدر را با هم ترکیب نمود و با استفاده از ورودی‌های فعلی (تواناساز) دیکدر بزرگتری بوجود آورد. بنابراین اگر یک دیکدر 6 به 64 مورد نیاز باشد می‌توان آن را با چند دیکدر 4 به 16 تشکیل داد.

شکل ۲-۳ نشان می‌دهد که چگونه از اتصال دیکدرهای دارای ورودی‌های تواناساز می‌توان دیکدرهای بزرگتری ساخت. در این شکل دو دیکدر 2 به 4 با هم ترکیب شده‌اند تا یک دیکدر 3 به 8 تشکیل شود. دو بیت کم ارزشتر ورودی‌ها به هر دو دیکدر متصل شده‌اند. بیت پرازشتر به ورودی تواناساز یکی از دیکدرها و از طریق یک معکوس کننده به ورودی تواناساز دیکدر دیگر وصل است. فرض براین است که دیکدر وقتی که  $E=1$  است فعال شود. اگر  $E=0$  باشد دیکدر غیرفعال و تمام خروجی‌های آن در سطح 0 است. هرگاه  $A_2=0$  باشد، دیکدر بالایی فعال و دیکدر پائینی غیرفعال است. خروجی‌های

شکل ۲-۳ یک دیکدر  $3 \times 8$  (یا  $3$  به  $8$ ) ساخته شده با دو دیکدر  $2 \times 4$ 

دیکدر تھتائی همگی غیرفعال و در وضعیت ۰ می‌باشند. خروجی‌های دیکدر فوقانی خروجی‌های  $D_0$  الی  $D_3$  را، بسته به مقادیر  $A_1$  و  $A_0$ ، تولید می‌کنند (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 می‌کند زیرا این عددهای دودویی رقم ۱ را در مکان  $A_2$  دارا هستند.

مثال فوق فوق فواید ورودی فعال‌ساز (تواناساز) را در دیکدرها یا هرمدار منطقی ترکیبی نشان داد. ورودی‌های توواناساز ابزار مناسبی برای اتصال دو یا چند مدار به منظور گسترش ورودی‌ها و خروجی‌ها می‌باشند.

### انکدرها<sup>۱</sup>

انکدر (کدگذار) یک مدار دیجیتال است که عکس عمل دیکدر را انجام می‌دهد. یک انکدر دارای  $2^n$  (یا کمتر) خط ورودی و  $n$  خط خروجی است. خطوط خروجی کد دودویی متناظر مقدار ورودی را تولید می‌کنند. 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 می‌کنند، است. فرض برایمن است که فقط یک ورودی دارای مقدار ۱ در هر لحظه از زمان است؛ در غیر این صورت مدار مفهومی ندارد. انکدر را می‌توان با گیت‌های OR، که ورودی‌هایشان مستقیماً از جدول درستی تعیین می‌شوند،

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 برقار 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 پلکسراها<sup>۱</sup>

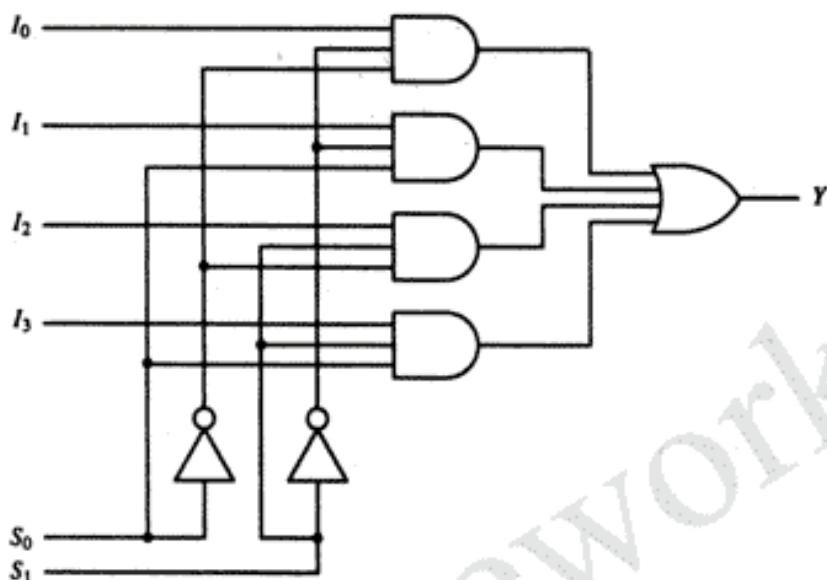
مولتی پلکسر یک مدار ترکیبی است که اطلاعات دودویی را از یکی از  $2^n$  ورودی دریافت و آن را به یک مسیر خروجی هدایت می نماید. مجموعه ای از خطوط ورودی انتخاب  $2^n$  خط داده خاصی را برای خروجی انتخاب می کنند. یک مولتی پلکسر  $2^n$  به 1 دارای  $2^n$  خط ورودی داده و  $n$  خط ورودی انتخاب می باشد. ترکیب 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  $S_1$  و  $S_0$  برای انتخاب یک گیت OR 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 مولتیپلکسر ۴ به ۱

گیت AND دیگر، هر یک حداقل یک ورودی ۰ دارند که در نتیجه آن خروجی شان ۰ می‌شود. بنابراین خروجی گیت OR برابر مقدار  $I_2$  است و باین ترتیب مسیری از ورودی‌های منتخب به خروجی ایجاد شده است.

مولتیپلکسر ۴ به ۱ شکل ۲-۴ دارای شش ورودی و یک خروجی است. جدول درستی که عملکرد مدار را نشان می‌دهد نیاز به ۶۴ 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 مناسب‌تری برای بیان عملکرد مولتیپلکسر استفاده از جدول تابع است. جدول تابع برای مولتیپلکسر در جدول ۲-۳ نشان داده شده است. این جدول رابطه بین چهار ورودی داده و تنها خروجی مدار را بصورت تابعی از ورودی‌های انتخاب  $S_0$  و  $S_1$  نشان می‌دهد. هرگاه ورودی‌های انتخاب برابر ۰۰ باشند، خروجی  $Y$  برابر  $I_0$  است. اگر ورودی‌های انتخاب ۰۱ شوند، ورودی  $I_1$  به خروجی  $Y$  متصل است، و برای دو ورودی دیگر نیز استدلال مشابه وجود دارد. مولتیپلکسر را انتخاب کننده داده<sup>۱</sup> هم می‌نامند، چون این مدار یکی از چند ورودی داده را برمی‌گزیند و اطلاعات دودویی را به خروجی هدایت می‌نماید.

گیت‌های AND و معکوس کننده‌ها در مولتیپلکسر مشابه یک مدار دیکدر می‌باشند و خطوط ورودی انتخاب را دیکد می‌کنند. بطور کلی، یک مولتیپلکسر  $2^n$  به ۱ خطی، از یک دیکدر  $n$  به

1- Data Selector (or data switch)

جدول ۲-۳ جدول تابع برای مولتیپلکسر ۴ به ۱ خطی

انتخاب		وروودی
$S_1$	$S_0$	$Y$
0	0	$I_0$
0	1	$I_1$
1	0	$I_2$
1	1	$I_3$

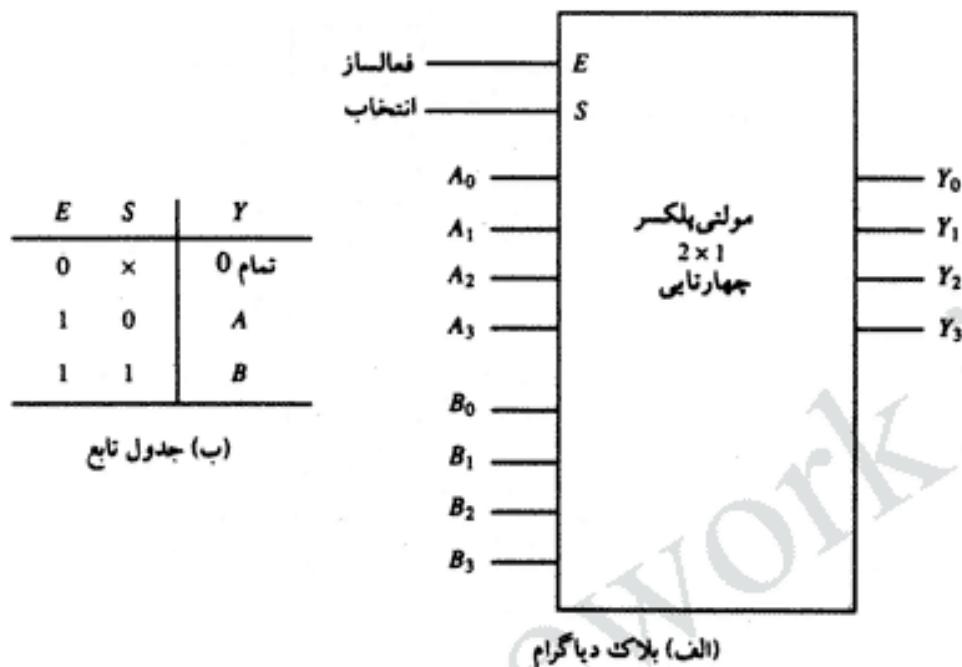
با اضافه کردن ۲ خط ورودی به آن، یعنی یک خط از هر ورودی داده، ساخته می شود. اندازه  $^1$  یک مولتیپلکسر بوسیله تعداد  $^2$  ورودی داده و تنها خروجی اش مشخص می شود. این اندازه بطور ضمنی بیانگر این واقعیت است که تعداد خطوط انتخاب نیز  $^2$  است. مولتیپلکسر را غالباً با نام اختصاری MUX می خوانند.

همانند دیکدرها، مولتیپلکسرا 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 افتد، و اگر در حالت فعال قرار گیرد، مدار به صورت یک مولتیپلکسر بطور عادی عمل می کند. ورودی تواناساز برای اتصال دو یا چند مولتیپلکسر و تشکیل یک مولتیپلکسر با تعداد ورودی های بیشتر مفید است.

در برخی از موارد دو یا چند مولتیپلکسر در یک بسته مدار مجتمع جای داده می شود. ورودی های انتخاب و تواناساز در ساختار چند واحدی، برای همه مولتیپلکسراها مشترک اند. بلاک دیاگرام یک مولتیپلکسر چهارتایی  $^2$  به  $^1$ ، بعنوان مثال در شکل ۲-۵ نشان داده شده است. مدار دارای چهار مولتیپلکسر 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 الى آخر. خط ورودی انتخاب  $S$  یکی از خطوط را در هر یک از چهار مولتیپلکسر انتخاب می کند. ورودی تواناساز  $E$  برای عملکرد عادی فعال می شود. اگرچه مدار دارای چهار مولتیپلکسر 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 خواهد 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 ۰ خواهند بود.

## ۲-۴ ثبات ها<sup>۲</sup>

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



شکل ۲-۵ مولتیپلکسر ۲ به ۱ چهارتایی

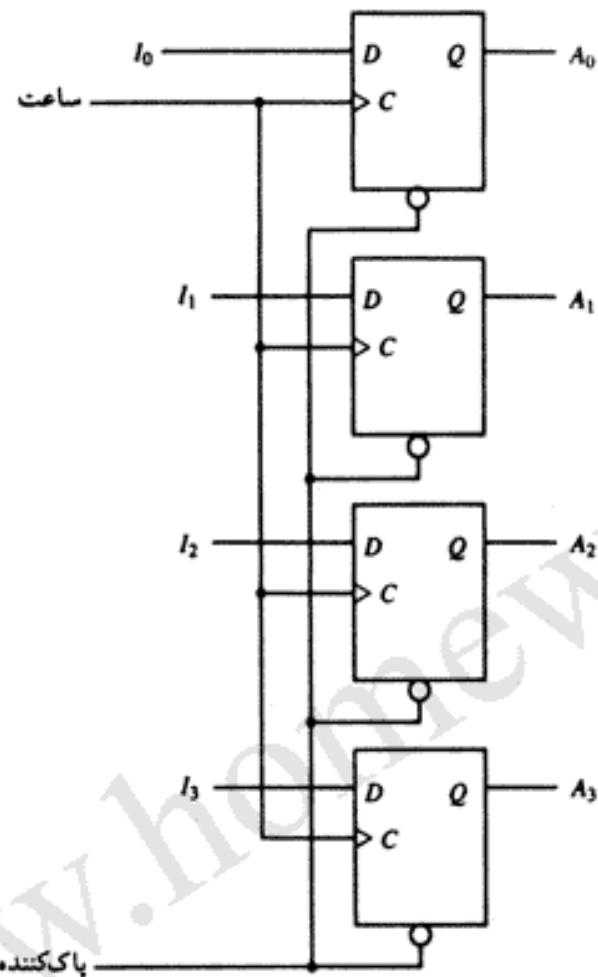
ذخیره کند. یک ثبات  $n$  بیتی  $n$  فلیپ فلاپ دارد و می‌تواند هر اطلاعات دودویی  $n$  بیتی را در خود ذخیره کند. یک ثبات، علاوه بر فلیپ فلاپ‌ها، ممکن است گیت‌های ترکیبی، که کارهای پردازش داده معینی را انجام می‌دهند، را هم در خود داشته باشد. یک ثبات در تعریف جامع تر خود متشکل از گروهی از فلیپ فلاپها و گیتهاست که بر تغییر حالت (خروچی) 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 قادر هر نوع گیت خارجی است. شکل ۲-۶ چنین ثباتی را که از چهار فلیپ فلاپ D ساخته شده نشان می‌دهد. ورودی پالس ساعت مشترک در لبه بالا رونده خود همه فلیپ فلاپ‌هارا فعال<sup>۱</sup> می‌سازد و داده دودویی موجود در چهار ورودی بداخل ثبات چهار بیتی منتقل می‌شود. چهار خروچی را می‌توان در هر زمانی نموده برداری کرد و اطلاعات ذخیره شده در ثبات را بدست آورد. ورودی پاک کردن<sup>۲</sup> به پایه خاصی در هر فلیپ فلاپ متصل است. هرگاه این ورودی 0 شود تمام فلیپ فلاپ‌ها بطور غیرهمزمان (آستکرون) و قبل از عملکرد پالس ساعت پاک<sup>۳</sup> 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 ۲-۶ ثبات ۴ بیت

انتقال اطلاعات جدید بداخل ثبات "بارکردن"<sup>۱</sup> ثبات نامیده می‌شود. اگر تمام بیت‌های 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 ورودی‌های C در ثبات شکل ۲-۶ تمام چهار ورودی I<sub>0</sub> تا I<sub>3</sub> 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.

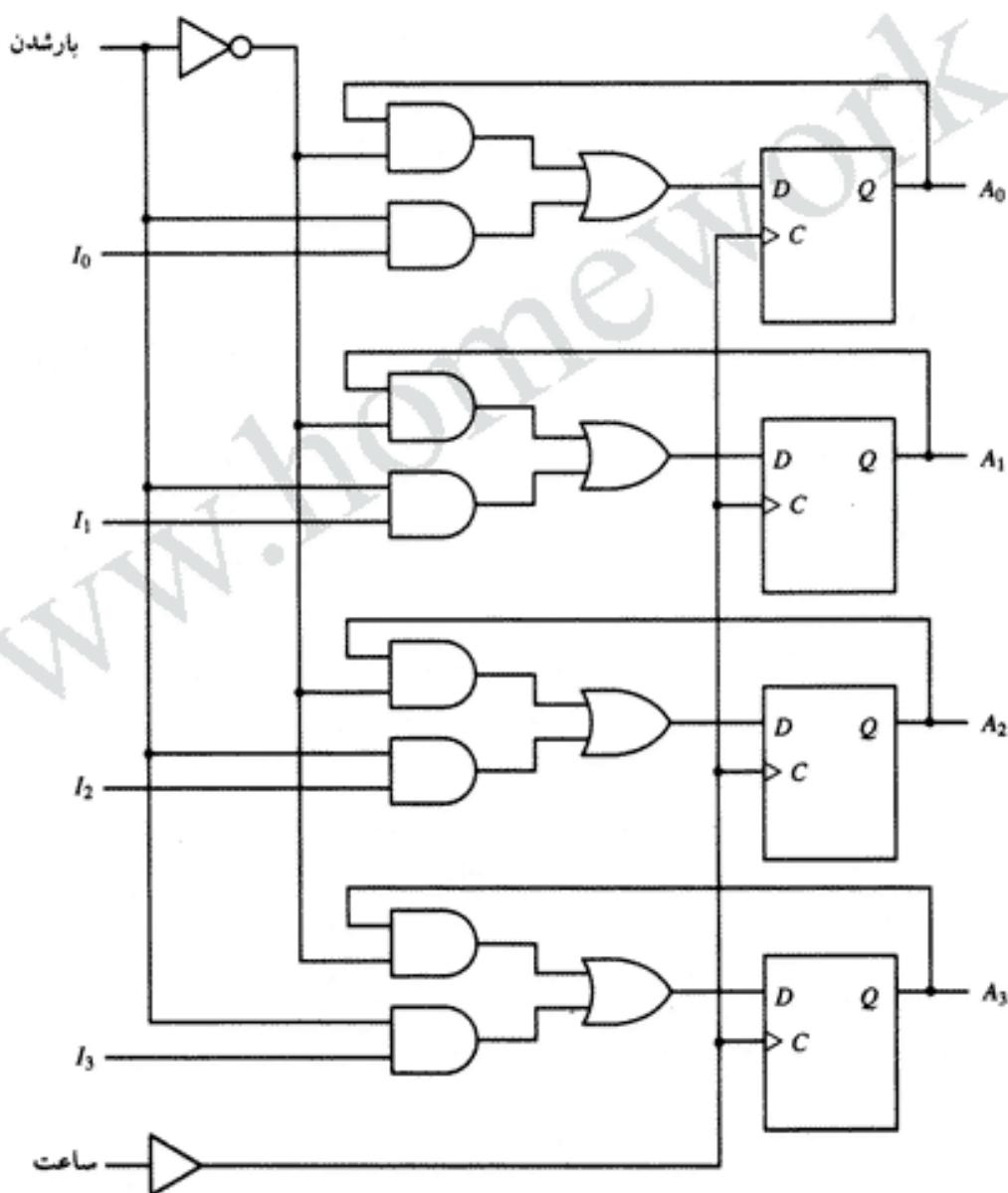
### ثبات با بارشدن موازی

بسیاری از سیستم‌های دیجیتال دارای یک مدار ساعت اصلی می‌باشند که رشته‌ای از پالس ساعت را

1- Loading

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 بارگردان که از طریق گیت‌ها به ورودی‌های D می‌رسند در شکل ۲-۷ 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 عرض چند گیت وصل شود توان کمتری لازم است ولی چنانچه این گیت بافر مورد



شکل ۲-۷ پک شیات ۴ بیت یا پارشدن موازی

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

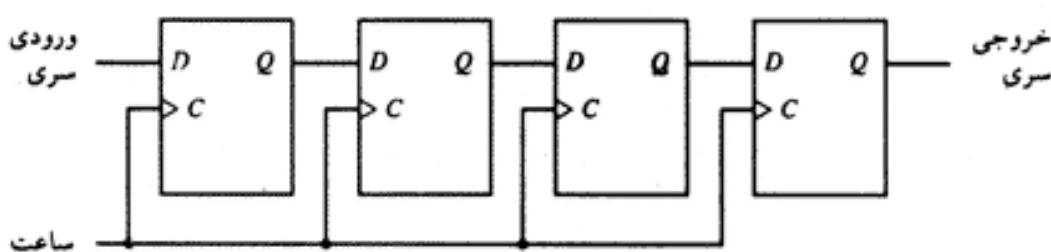
ورودی بارکردن عملی را که با هر پالس ساعت در ثبات انجام می شود تعیین می کند. هرگاه ورودی بارکردن برابر 1 باشد، داده های چهار ورودی بالبه مثبت پالس ساعت (گذار مثبت) به داخل ثبات انتقال می یابند. اگر ورودی باردهی 0 باشد، ورودی های داده ممانعت شده و ورودی های فلیپ فلاپ D به خروجی های آن وصل می شوند. اتصال برگشت<sup>۱</sup> (پسخون) 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.

## ۲-۵ شیفت رجیسترها<sup>۲</sup>

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 زنجیرهای از فلیپ فلاپهای متوالی تشکیل شده، که در آن خروجی يک فلیپ فلاپ به ورودی فلیپ فلاپ بعدی متصل شده است. تمام فلیپ فلاپ ها پالس ساعت مشترکی را که موجب جابجا یابی یک طبقه به طبقه بعدی است دریافت می کنند.

ساده ترین شیفت رجیستر ممکن، فقط از فلیپ فلاپ ها استفاده می کند، شکل ۲-۸. 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 ۲-۸ شیفت رجیستر ۴ بیت

1- Feedback

2- Shift Register

گاهی لازم است شیفت را چنان کنترل کنیم که با وقوع پالس های معینی و نه همه پالس ها بوقوع پیووند. این عمل با ممانعت از اعمال پالس ساعت به ورودی ثبات میسر است. اگر شیفت رجیستر شکل ۲-۸ 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.

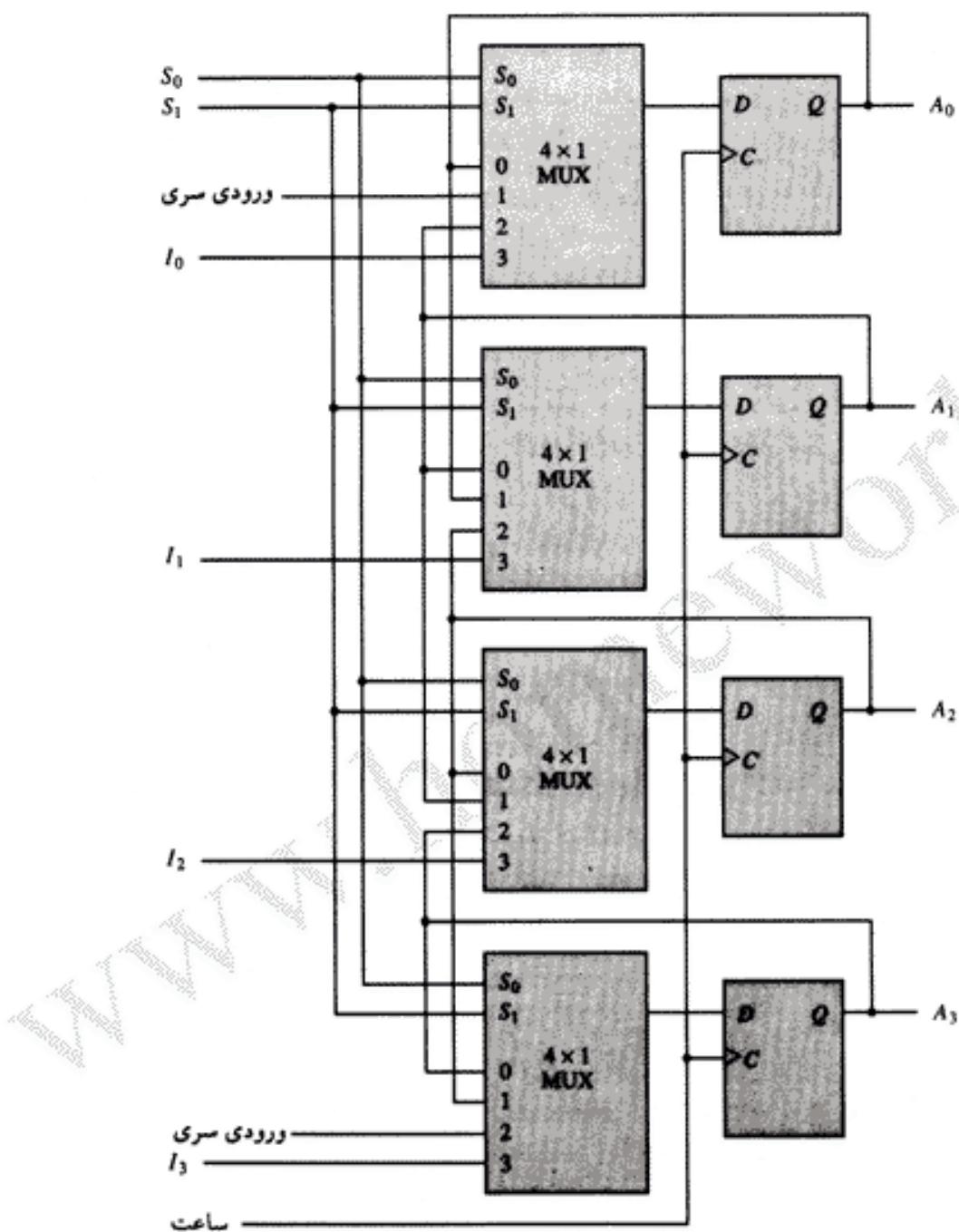
### شیفت رجیستر دو جهته با بارشدن موازی

ثباتی که بتواند عمل شیفت را فقط در یک جهت انجام دهد ثبات شیفت رجیستر یک طرفه<sup>۱</sup> 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 تمام توانایی های ذکر شده زیر را دارد. 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 ۲-۹ نشان داده شده است. هر طبقه از یک فلیپ فلاپ D و یک مولتی پلکس  $4 \times 1$  تشکیل شده است. دو ورودی انتخاب  $S_1$  و  $S_0$  یکی از ورودی های داده مولتی پلکس را برای فلیپ فلاپ D انتخاب می کند. خطوط انتخاب شده عملکرد ثبات را طبق جدول تابع ۲-۴ کنترل می کند. هرگاه ورودی های کنترل  $S_1 = S_0 = 00$  باشند ورودی داده 0 هر مولتی پلکس 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  $S_1 = S_0 = 01$  باشد، پایانه های در هر مولتی پلکس که با 1 علامت زده شده است به ورودی D فلیپ فلاپ مربوطه اش راه می یابد. این خود سبب یک شیفت

1- Unidirectional



شکل ۲-۹ شبکت 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  $A_i$  منتقال می‌باشد که در آن  $i=1,2,3$  می‌باشد. اگر  $S_1S_0=10$  باشد، اطلاعات دودویی از هر ورودی  $I_0$  تا  $I_3$  وارد فلیپ فلاپ مربوط به خود می‌شوند و در نتیجه یک ساعت

جدول ۲-۴ جدول تابع برای ثبات شکل ۲-۹

کنترل شیوه		عملکرد ثبات
$S_1$	$S_0$	
0	0	بلا تغییر
0	1	ثابت به راست (پائین)
1	0	ثابت به چپ (بالا)
1	1	بار شدن موازی

بار شدن موازی صورت گرفته است. وقت کنید که براساس شکل ترسیم شده، شیفت به راست محتوای ثبات‌ها را رویه پائین و شیفت به چپ جایجاوی 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 مبدأ و مقصد خیلی زیاد باشد، استفاده از  $n$  خط برای انتقال موازی آنها پر هزینه خواهد بود. ممکن است استفاده از یک خط واحد مفروض 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 پرداشت 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 شمارندهای دودویی

ثبتی 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 سیگнал‌های 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 سرراست‌ترین است. شمارنده‌ای که رشته اعداد دودویی را دنبال می‌کند شمارنده دودویی نام دارد. یک شمارنده دودویی  $n$  بیتی، ثباتی است که از  $n$  فلیپ فلáp و گیت‌های مربوطه‌اش تشکیل شده و رشته از حالات را براساس شمارش دودویی  $n$  بیت، از ۰ تا  $1^n-2$  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 ۰۰۰۰۱، ۰۰۰۱۰، ۰۰۱۱۰ و غیره، در می‌باییم 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 ۱ باشند متمم می‌شود. مثلاً شمارش دودویی از ۰۱۱۱ (۷) به ۱۰۰۰ (۸) با؛ الف) متمم کردن بیت کم ارزشتر، ب) متمم کردن بیت دوم، چون اولین بیت در ۰۱۱۱ برابر ۱ است، ج) متمم کردن بیت مرتبه سوم، چون دو بیت اول در ۰۱۱۱ برابر ۱ می‌باشد، ذ) متمم کردن بیت چهارم، چون سه بیت اول در ۰۱۱۱ همگی برابر ۱ می‌باشند، بدست می‌آید.

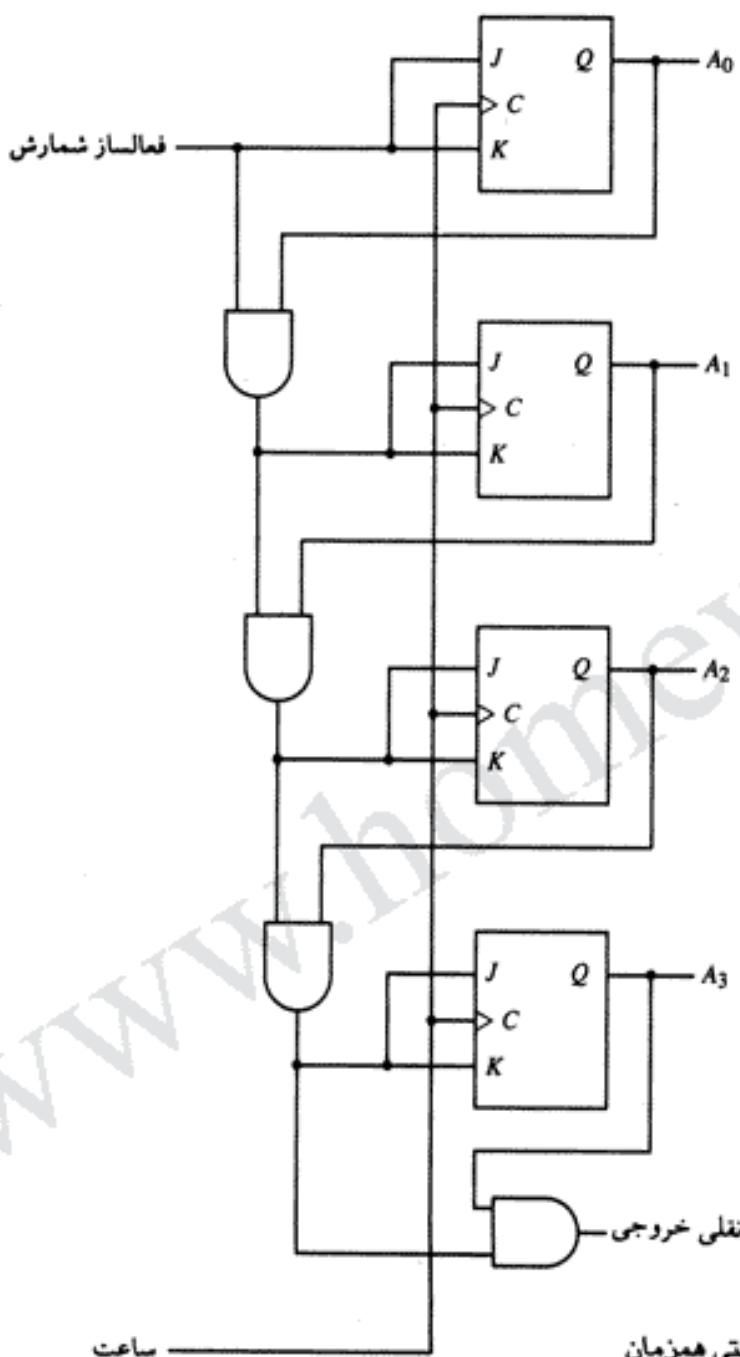
مدار شمارنده معمولاً از فلیپ فلاپهایی که قابلیت متمم شدن دارند استفاده می‌نماید. فلیپ فلاپ‌های T و JK این خاصیت را دارا هستند. بخاطر بیاورید که فلیپ فلاپ JK در صورتی که هر دو ورودی J و K برابر ۱ باشند و ساعت گذار مثبت انجام دهد متمم می‌شود. در  $J=K=0$  خروجی فلیپ فلاپ تغییر نمی‌کرد. بعلاوه، شمارنده 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 ۴ بیتی شکل ۲-۱۰ 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<sup>۱</sup> ۰ باشد تمام ورودی‌های J و K در ۰ باقی می‌مانند و خروجی‌های شمارنده تغییری نمی‌کنند. طبقه اول یعنی A۰ 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 ۱ و شمارش نیز فعال شود. زنجیره گیت‌های AND منطق لازم را برای ورودی‌های J و K تهیه می‌کنند. با استفاده از رقم نقلی خروجی می‌توان شمارنده رابه 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 ۲-۱۱ دیاگرام منطقی یک شمارنده دودویی که دارای امکان بارشدن موازی و نیز پاک شدن همزمان به ۰ می‌باشد را نشان می‌دهد. اگر ورودی پاک کننده ۱ باشد، همه ورودی‌های K را ۱ می‌کند و بنابراین با گذار بعدی ساعت همه فلیپ فلاپ‌ها را ۰ می‌نماید. اگر ورودی کنترل بارگردان برابر ۱ باشد، عمل شمارش را غیرفعال نموده و موجب می‌شود تا چهار ورودی داده بداخل چهار فلاپ انتقال یابند (شرطی که ورودی پاک کردن

1- Count Enable

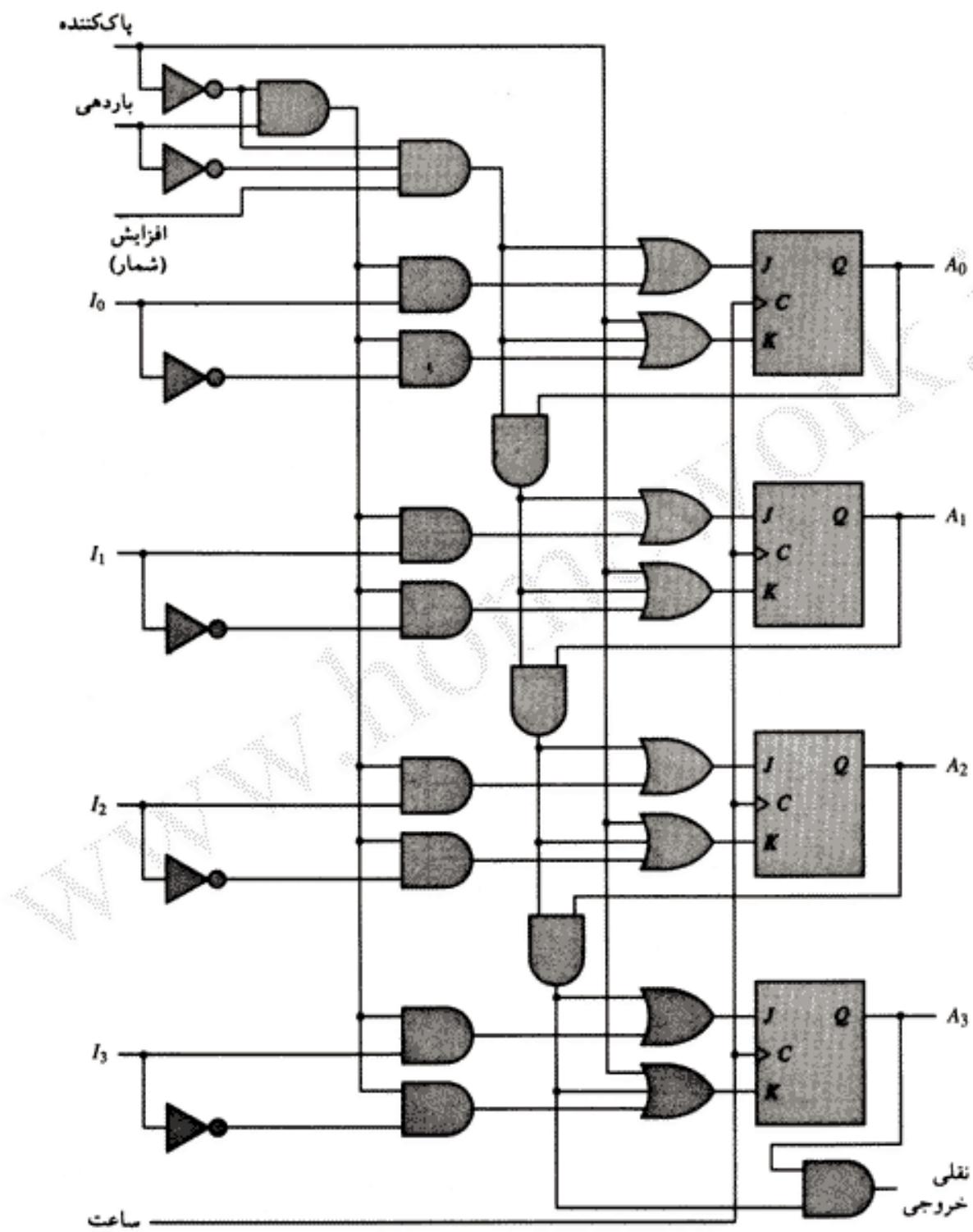


شکل ۲-۱۰ شمارنده همزمان ۴ بیتی همزمان

0 باشد). اگر هر دو ورودی پاک کردن و بارگردان 0 و ورودی افزایش<sup>۱</sup> 1 باشد مدار بصورت یک شمارنده دودویی عمل خواهد کرد.

---

#### 1- Increment



شکل ۲-۱۱ شمارنده دودویی ۴ بیتی با بارشدن موازی و پاک شدن همزمان

جدول ۲-۵ جدول تابع برای ثبات شکل ۲-۱۱

عمل	افزایش	بارشدن	پاک شدن	ساعت
بلا تغییر	0	0	↑	
1 واحد افزایش شمارش	1	0	↑	
بارگردان ورودی های ۰ تا ۱۳	×	1	↑	
پاک کردن خروجی های ۰ به ۱	×	1	↑	

عملکرد مدار در جدول ۲-۵ خلاصه شده است. اگر همه ورودی های پاک کردن، بارگردان و افزایش ۰ باشند هیچ تغییر در خروجی نخواهد بود حتی اگر پالس ها به ورودی C اعمال گردند. اگر ورودی های پاک و بارگردان در ۰ نگهداشته شود، ورودی افزایش عملکرد را کنترل نموده و خروجی ها بازه هر گذار پالس ساعت به عدد دو دویی بعدی در شمارش تغییر خواهند کرد. هر وقت ورودی بارگردان ۱ باشد، بشرطی که ورودی پاک کردن ناتوان (غیرفعال) ولی ورودی افزایش ۰ یا ۱ باشد، داده های ورودی بداخل فلیپ فلاپ ها بار می شوند. با اعمال ورودی پاک کردن، ثبات بدون توجه به مقادیر ورودی های بارگردان و افزایش ، ۰ خواهد شد.

شمارنده هایی که دارای قابلیت بارگردان 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 پریود 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 <sup>۱</sup> 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 <sup>۲</sup> گروهی از ۱ ها و ۰ هاست و ممکن است بعنوان یک عدد، یک کد دستور العمل، یک یا چند کاراکتر الفبا عددی یا هر اطلاعات دودویی کد شده دیگر باشد. یک گروه هشت بیتی، بایت <sup>۳</sup> خوانده می شود. در اغلب حافظه های کامپیوتر، تعداد بیت های هر کلمه مضربی از ۸ است. بنابراین یک کلمه ۱۶ بیتی حاوی دو بایت است، و یک کلمه ۳۲ 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. ساختمنان 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 آدرس<sup>۱</sup> نام دارند. به هر کلمه در حافظه یک شماره شناسایی تعلق می‌گیرد که به آن آدرس اطلاق می‌شود، و از ۰ آغاز و با ۱، ۲، ۳ تا  $1-k^2$  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. دیکدری 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)<sup>۲</sup>،  $M$  (مگا)<sup>۳</sup> یا  $G$  (گیگا)<sup>۴</sup> بیان کنند.  $K$  برابر با  $2^{10}$ ،  $M$  برابر با  $2^{20}$  و  $G$  برابر با  $2^{30}$  می‌باشد. بنابراین  $4G=2^{21}$ ،  $2M=2^{21}$  و  $64K=2^{16}$ .

دو نوع عمده حافظه که در سیستم‌های کامپیوتری بکار می‌روند عبارتند از: حافظه با دستیابی تصادفی<sup>۵</sup> (RAM) و حافظه فقط خواندنی<sup>۶</sup> (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 نشاندن 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. بلک دیاگرام یک واحد RAM در شکل ۲-۱۲ نشان داده شده است. خط ورودی داده اطلاعات را که باید در حافظه ذخیره شوند فراهم می‌آورند، و خط خروجی داده نیز اطلاعاتی را که از حافظه بیرون می‌آیند در اختیار می‌گذارند. خط آدرس یک عدد دودویی  $k$  بیتی را برای انتخاب کلمه مورد نظر از میان  $2^k$  کلمه، در داخل حافظه فراهم



شکل ۲-۱۲ بلک 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 مورد نظر را مشخص می کنند. در یک حافظه RAM می توان دو عمل نوشتن<sup>۱</sup> و خواندن<sup>۲</sup> را انجام داد. سیگنال نوشت، 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 دهد. 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 خواندن.

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 مورد نظر 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.

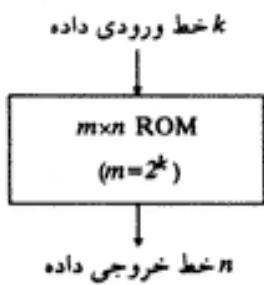
### حافظه فقط خواندنی

حافظه فقط خواندنی (ROM) همانطور که از نامش مشخص است، حافظه ای است که فقط عمل خواندن را انجام می دهد؛ یعنی دارای امکان نوشت نیست. این بدان معنی است که اطلاعات دودویی ذخیره شده در ROM حین تولید سخت افزار در آن بصورت دائمی درآمده است و نمی توان با نوشت 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 دهنی" کرد. به محض ایجاد الگوی مورد نظر، 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 ۲-۱۳ دیده می شود، ROM دارای  $k$  خط ورودی آدرس

1- Write

2- Read



شکل ۲-۱۳ 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 (تواناسان) 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 خود بخود  $n$  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 ثابت شده‌اند.

ROM‌ها 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 کامپیوتراً 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 کامپیوتراً دیجیتال نیز بکار گرفته می‌شود. در این حال، این واحد برای ذخیره اطلاعات کد شده‌ای که نماینده رشته متغیرهای کنترل داخلي لازم برای فعال کردن اعمال مختلف در کامپیوتراً است استفاده می‌شود. یک واحد کنترل که از ROM برای ذخیره سازی اطلاعات دودویی کنترل استفاده می‌کند یک واحد کنترل ریزبرنامه نویسی 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<sup>۱</sup>، در حین آخرین فرآیند ساخت توسط کمپانی سازنده نیمه رسانا، انجام می‌شود. در روش ساخت ROM، مشتری جدول درستی آنچه را که ROM باید برآن اساس کار کند پر می‌نماید. جدول درستی ممکن است بروش‌های مختلفی تهیه و یا با قالب خاصی در خروجی کامپیووتر ارائه شود. سازنده، ماسک مناسبی را برای مسیرها تهیه می‌کند تا ۱ها و ۰ها بر طبق جدول درستی مشتری تهیه شوند. این روش پرهزینه است زیرا فروشنده، مبلغ خاصی را بخاطر ساخت سفارشی ROM تقاضا می‌کند. 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،<sup>۲</sup> PROM نام دارد اقتصادی‌تر است. بهنگام سفارش تمام فیوزهای PROM دست نخورده هستند و لذا تمام بیت‌های ذخیره شده در کلمه ۱ خواهند بود. فیوزهای PROM را می‌توان با اعمال پالس‌های جریان از طریق پایانه‌های خروجی برای هر یک از آدرس‌ها سوزاند. یک فیوز سوخته حالت دودویی ۰ را بوجود می‌آورد و فیوز دست نخورده حالت دودویی ۱ را می‌دهد. این مطلب به کاربر این امکان را می‌دهد تا PROM مورد نظر خود را در آزمایشگاه خود برای بدست آوردن رابطه بین آدرس ورودی و کلمه ذخیره شده برنامه‌نویسی کند. دستگاه‌های خاصی بنام برنامه‌نویس PROM برای انجام این عمل بصورت تجاری موجود است. در هر صورت، روش‌های موجود در برنامه‌نویس ROM‌ها سخت افزاری می‌باشند هر چند که از کلمه "برنامه نویسی" هم در بیان آن استفاده شده باشد.

اعمال برنامه‌نویسی 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 بیت‌ها، آنها را تغییر دهیم، قطعه باید کلاً دور اندادخته شود. نوع سومی از ROM‌ها نیز وجود دارد که PROM پاک شونده یا EPROM خوانده می‌شود. EPROM را، حتی اگر فیوزهایش قبل از سوزانده شده باشد، می‌توان بصورت اولیه بازسازی کرد. وقتی EPROM در زیر اشعه ماوراء بنشان برای مدت معینی قرار گیرد، تابش طول موج کوتاه گیت‌های داخلی را که نقش فیوز را بعهده دارند تخلیه می‌نماید. پس از پاک کردن، EPROM به حالت اولیه‌اش باز می‌گردد و می‌تواند با مجموعه جدیدی از کلمات برنامه نویسی گردد. انواع بخصوصی از PROM را می‌توان با سیگنالهای الکتریکی، بجای نور ماوایر بنفس، پاک کرد. این PROM پاک شدنی الکتریکی یا EEPROM می‌نامند.

## مسائل

- ۲-۱ قطعات SSI TTL اغلب بصورت مدارات مجتمع 14 پایه عرضه می‌شوند. دو پایه برای منبع تغذیه در نظر گرفته شده و بقیه برای پایانه‌های ورودی و خروجی بکار می‌روند. چند مدار از انواع زیر را در یک چنین بسته‌ای می‌توان 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 فلاب JK ساعت دار با پاک کننده غیرهمزمان.
- ۲-۴ 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 راهنمای یافته و آنها را با قطعات متناظر ارائه شده در این فصل مقایسه کنید.
- الف) IC نوع 74155، دیکدر دوتایی ۲ به ۴
- ب) IC نوع 74157، مولتی پلکسر چهار تایی ۲ به ۱ خطی
- ج) IC نوع 74194، شیفت رجیستر چهار بیت دوطرفه با بارشدن موازی
- د) IC نوع 74163، شمارنده دودویی چهار بیت بارشدن موازی و پاک شدن همزمان ۴ یک دیکدر ۵ به ۳۲ را با چهار دیکدر ۳ به ۸ که دارای ورودی تواناساز است و یک دیکدر ۲ به ۴ بسازید. از بلاک دیاگرام مشابه شکل ۲-۳ استفاده کنید.
- ۲-۴ دیاگرام منطقی یک دیکدر ۲ به ۴ را فقط با گیت NOR ترسیم کنید. ورودی فعال ساز نیز در نظر گرفته شود.
- ۲-۵ دیکدر شکل ۲-۲ 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 انکدر با هشت ورودی و سه خروجی، که جدول درستی آن در جدول ۲-۲ آورده شد را ترسیم کنید. وقتی تمام ورودی ها ۰ باشند خروجی چیست؟ اگر فقط ورودی D برابر ۰ باشد خروجی چیست. روشی را پیشنهاد کنید که این دو حالت را از هم تمیز دهد.
- ۲-۷ یک مولتی پلکسر ۱۶ به ۱ را با دو مولتی پلکسر ۸ به ۱ و یک مولتی پلکسر ۲ به ۱ 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 فلاب 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 ۱۱۰۱ است. ثبات شش بار با استفاده از ورودی سری ۱۰۱۱۰۱ 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 دهد چگونه می‌توان ورودی سری را به خروجی موازی و بالعکس تبدیل کرد.
- ۲-۱۴ شمارنده حلقوی یک شیفت رجیستر مطالق شکل ۲-۸ می‌باشد که خروجی سری اش به ورودی سری آن وصل شده است. با شروع از حالت اولیه ۱۰۰۰، رشته حالات چهار فلیپ فلاپ را پس از هر شیفت لیست کنید.
- ۲-۱۵ یک شیفت رجیستر چهاربیتی دو طرفه با قابلیت بارشدن موازی طبق شکل ۲-۹ در یک بسته مدار مجتمع بسته‌بندی شده است.
- الف) بلاک دیاگرام مدار مجتمع را با تمام ورودی‌ها و خروجی‌هایش ترسیم کنید.
- ب) با استفاده از دو مدار مجتمع بلاک دیاگرام یک شیفت رجیستر ۸ بیت با قابلیت بارشدن موازی را رسم کنید.
- ۲-۱۶ در یک شمارنده دودویی ۶ بیت چند فلیپ فلاپ در شمارش بعدی متمم می‌شوند.
- الف) ۱۰۰۱۱۱۱۱۱۱۱  
ب) ۰۰۰۱۱۱۱۱۱۱۱
- ۲-۱۷ اتصالات لازم بین چهار شمارنده دودویی چهاربیتی با بارشدن موازی (شکل ۲-۱۱) برای ایجاد یک شمارنده دودویی ۱۶ بیت با بارشدن موازی را نشان دهد. برای هر شمارنده چهاربیتی از یک بلاک دیاگرام استفاده کنید.
- ۲-۱۸ نشان دهد که چگونه یک شمارنده دودویی با بارشدن موازی شکل ۲-۱۱ را می‌توان به یک شمارنده تقسیم بر N تبدیل کرد (یعنی شمارندهای که از ۰۰۰۰ تا N شمرده و به ۰۰۰۰ بازگردد).  
بعنوان یک حالت خاص مدار یک تقسیم بر ۱۰ 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 بیت‌ها مشخص شده‌اند. در هر مورد چند خط آدرس و چند خط داده ورودی - خروجی لازم است؟
- الف) 2Kx8  
ب) 64Kx8  
ج) 16Mx32  
د) 4Gx64
- ۲-۲۰ تعداد بایت‌های را که می‌توان در حافظه‌های مسئله ۲-۱۹ ذخیره کرد مشخص کنید.
- ۲-۲۱ چند تراشه حافظه  $128 \times 8$  برای ایجاد یک حافظه  $4096 \times 16$  لازم است؟
- ۲-۲۲ با فرض داشتن یک ROM به ظرفیت  $8 \times 32$  بیت و یک ورودی تواناساز، اتصالات خارجی لازم برای ساخت یک تراشه ROM با ظرفیت  $128 \times 8$  و یک دیکدر را نشان دهد.
- ۲-۲۳ یک تراشه ROM یا ظرفیت  $4096 \times 8$  دارای دو ورودی تواناساز بوده و با منبع تغذیه ۵ ولت کارمی‌کند. چند پایه برای بسته مدار مجتمع لازم است؟ بلاک دیاگرام آن را رسم و پایانه‌های ورودی و خروجی را نام‌گذاری کنید.

## نمایش داده ها



### ۱-۳ انواع داده ها

۲-۱ متممها

۲-۲ نمایش ممیز- ثابت

۲-۳ نمایش ممیز شناور

۲-۴ انواع دیگر کدهای دودویی

۲-۵ کدهای آشکارسازی خط

### ۱-۳ انواع داده ها

اطلاعات دودویی در کامپیوترهای دیجیتال در حافظه ها یا ثبات های پردازنده ذخیره می شوند. ثبات ها حاوی داده ها یا اطلاعات کنترل هستند. اطلاعات کنترل مشکل از یک بیت یا گروهی از بیت هاست که برای مشخص کردن سیگنال های فرمان لازم در دستکاری داده های سایر ثبات ها استفاده می شود. داده ها، اعداد و اطلاعات دودویی کد شده دیگری می باشند که برای حصول به نتایج محاسباتی مورد نظر روی آنها کار می شود. در این فصل متداول تر نوع داده ها را که در کامپیوترهای دیجیتال یافت می شوند معرفی می کنیم و چگونگی نمایش انواع مختلف داده ها، بشکل کد شده دودویی در ثبات های کامپیوتر را نشان خواهیم داد.

انواع داده های موجود در ثبات های کامپیوترهای دیجیتال را می توان به ۳ فرم زیر طبقه بندی کرد:

(۱) اعدادی که در عملیات حسابی بکار می روند، (۲) حروف الفبا مورد استفاده در پردازش داده، و (۳) سایر سمبول های گستته که برای اهداف خاصی بکار می روند. همه انواع داده ها، بجز اعداد دودویی، به فرم دودویی کد شده در ثبات های کامپیوتر نمایش داده می شوند. این بدان علت است که ثبات ها از

فلایپ فلاپ ساخته شده‌اند و فلایپ فلاپ‌ها وسیله دو حالتی هستند که قادرند فقط ۱ها و ۰ها را در خود ذخیره کنند. سیستم اعداد دودویی طبیعی ترین سیستم بکار رفته در یک کامپیوتر دیجیتال است. ولی گاهی مناسبتر است سیستم اعداد دیگری را بکار برد. به خصوص سیستم اعداد دهدی به این دلیل که بوسیله انسان جهت عملیات حسابی بکار می‌رond.

سیستم‌های عدد‌نویسی

یک سیستم عددنویسی با پایه<sup>۱</sup> ۲ مبنای<sup>۲</sup> سیستمی است که سمبول‌های متمایز از همی را برای نمایش ۲ رقم بکار می‌برد. اعداد بارشته‌ای از سمبول‌های ارقام نمایش داده می‌شوند. برای تعیین کمیتی که یک عدد نمایشگر آنست، لازم است تا هر رقم را در توان صحیحی از ۲ ضرب کرده و سپس مجموع همه این حاصلضرب‌ها محاسبه شود. مثلاً، سیستم اعداد دهدی که همه روزه مورد استفاده است سیستم مبنای ۱۰ را بکار می‌برد. ۱۰ سمبول عبارتند، ۰، ۱، ۲، ۳، ۴، ۵، ۶، ۷، ۸، ۹. رشته رقم ۷۲۴.۵ بیانگر کمیت زیر تلقی می‌شود.

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

یعنی 7 صد تایی بعلاوه 2 ده تایی بعلاوه 4 یکی بعلاوه 5 دهم. بطور مشابه هر عدد دهدھی را می‌توان بهمین ترتیب برای یافتن کمیت آن تفسیر کرد.  
سیستم عددنویسی دودویی مبنای 2 را بکار می‌برد. دو سمبول ارقام مورد استفاده، 0 و 1 هستند. رشته ارقام 101101 یانگر 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 مبنای مختلف، اعداد در پرانتزهایی قرار داده شده و مبنای عدد بصورت اندیس در زیر پرانتز قرار می‌گیرد. مثلاً برای نشان دادن تساوی بین چهل و پنج دودویی و دهدھی می‌نویسیم  $(101101)_2 = (45)_{10}$

علاوه بر سیستم‌های اعداد دودویی و دهدزی، سیستم‌های هشت‌هشتی (مبنای 8) و شانزده‌شانزده‌ی (مبنای 16) نیز در کامپیوترهای دیجیتال اهمیت دارند. هشت‌سنبیل اعداد هشت‌هشتی عبارتند از: 1, 2, 3, 4, 5, 6, 7. شانزده‌سنبیل اعداد شانزده‌شانزده‌ی عبارتند از: 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. شش سنبیل آخر، متأسفانه، 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 شانزده‌ی بکار روند به ترتیب متناظر با اعداد 10, 11, 12, 13, 14, 15 می‌باشند.

یک عدد در مینای ۲ را با تشکیل جمیع حاصلضرب‌های ارقام در وزن‌هایشان می‌توان به سیستم

دده‌هی تبدیل کرد. مثلاً عدد هشت هشتی 736.4 به صورت زیر به دده‌هی تبدیل می‌شود:

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

عدد دده‌هی معادل عدد F3 در مبنای 16 از محاسبه زیر بدست می‌آید

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

تبدیل از دده‌هی به معادل مبنای ۲ آن، با جداسازی بخش صحیح عدد از کسری آن و تبدیل هر بخش بطور جداگانه صورت می‌گیرد. تبدیل یک عدد صحیح دده‌هی به مبنای ۲ بوسیله تقسیمات متوالی بر ۲ و کنار هم گذاشتن باقیمانده‌ها انجام می‌شود. شکل ۳-۱ این عملیات را نشان می‌دهد.

تبدیل عدد دده‌هی 41.6875 به دودویی ابتدا با جداکردن عدد به دو قسمت صحیح 41 و کسری 0.6875 آغاز می‌شود. برای تبدیل بخش صحیح، 41 را بر 2 تقسیم می‌کنیم تا خارج قسمت صحیح 20 و باقیمانده 1 بدست آید. خارج قسمت را دوباره بر 2 تقسیم می‌کنیم تا خارج قسمت و باقیمانده جدیدی بدست آید. این روند تا 0 شدن خارج قسمت ادامه می‌یابد. ضرائب عدد دودویی از باقیمانده‌ها حاصل می‌گردد، بدین ترتیب که اولین باقیمانده بیت مرتبه پائین عدد دودویی تبدیل شده را تشکیل می‌دهد.

بخش کسری عدد با ضرب آن در 2 حاصل می‌گردد تا یک عدد صحیح همراه با کسری تولید کند. قسمت کسری جدید (بدون بخش صحیح آن) دوباره در 2 ضرب می‌شود تا عدد صحیح همراه با کسر دیگری بدست آید. این روند تا صفر شدن بخش کسری و یا تا جایی 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 می‌گردد. نهایتاً دو قسمت برای تبدیل کل با یکدیگر ترکیب می‌گردند.

بخش صحیح 41

= بخش کسری 0.6875

41	0.6875
20	2
10	1.3750
5	x 2
2	0.7500
1	x 2
0	1.5000
0	x 2
	1.0000

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

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

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

شکل ۳-۱ تبدیل 41.6875 دده‌هی به دودویی

## اعداد هشت هشتی و شانزده شانزده‌ی

تبديل متقابل دودویی، هشت هشتی و شانزده شانزدهی نقش مهمی در کامپیوترهای دیجیتال دارند. چون  $8 = 2^3$  و  $16 = 2^4$  می‌باشد، هر رقم هشت هشتی متعلق به سه رقم دودویی است و هر رقم مبنای شانزده مربوط به چهار رقم دودویی می‌باشد. تبدیل از دودویی به هشت هشتی بسادگی با تقسیم بندی عدد دودویی به گروههای سه بیتی انجام می‌شود. سپس رقم هشت هشتی مربوط به هر گروه از بیت‌ها به آن‌ها اختصاص می‌یابد و رشته ارقام حاصل، معادل هشت هشتی را برای عدد دودویی بدست می‌دهد. مثلاً یک ثبات 16 بیتی را در نظر بگیرید. بطور فیزیکی می‌توان تصور کرد که یک ثبات از 16 سلول ذخیره سازی دودویی تشکیل شده است، که هر سلول قادر است 1 یا 0 را در خود نگهدارد. فرض کنید آرایش بیت‌های ذخیره شده در ثبات طبق شکل ۳-۲ باشد. چون یک عدد دودویی رشته‌ای از ترکیبات ۰ و ۱ است، ثبات 16 بیتی برای ذخیره کردن هر عدد دودویی از ۰ تا  $1 - 2^{16}$  مناسب است. برای مثال خاص نشان داده شده، عدد دودویی ذخیره شده در ثبات معادل با ۴4899 می‌باشد. با شروع از بیت مرتبه پایین‌تر، ما ثبات را به گروههای سه بیتی تقسیم می‌کنیم (بیت شانزدهم به تنها بیت باقی می‌ماند). به هر گروه سه بیتی، یک رقم معادل هشت هشتی تخصیص می‌یابد و در بالای آن نوشته می‌شود. رشته ارقام هشت هشتی حاصل نمایش دهنده معادل هشت هشتی عدد دودویی است.

تبديل دودویی به شانزده شانزدهی مشابه فوق است بجز اینکه بیت‌ها به گروه‌های چهارتایی تقسیم می‌شوند. رقم شانزده شانزدهی مربوطه، برای هر گروه چهاربیتی در زیر ثبات در شکل ۳-۲ نوشته شده‌اند. رشتۀ ارقام شانزده شانزدهی حاصل، معادل عدد دودویی است. رقم هشت هشتی مربوط به هر گروه سه بیتی بسادگی از روی اولین هشت رقم در لیست جدول ۳-۱ بخاطر سپرده می‌شود. کد چهاربیتی معادل هر رقم شانزده شانزدهی را می‌توان در ۱۶ سطر اول جدول ۳-۲ پیدا کرد.

جدول ۳-۱ چند عدد هشت هشتی و نمایش آنها را در ثبات‌ها بشکل کد شده دودویی نشان می‌دهد. کد دودویی از روشنی که تشریح شد حاصل می‌شود. به هر رقم هشت هشتی یک کد سه بیتی مطابق آنچه در هشت رقم جدول مشخص شده تخصیص می‌یابد. بطور مشابه، جدول ۳-۲ چند عدد شانزده شانزده‌ی و نمایش آنها را در ثبات بفرم کد شده دودویی نشان می‌دهد. در اینجا کد دودویی بدین صورت حاصل می‌شود که به هر رقم شانزده شانزده‌ی یکی از شانزده وارد لیست شده در جدول اختصاص می‌یابد.

با مقایسه اعداد هشت هشتی و شانزده شانزدهی در فرم کد دودویی با معادل عدد دودویی آنها، می بینیم که ترکیب بیت‌ها در هر سه فرم یکسان است. مثلاً عدد ددهی 99 وقتی بصورت دودویی درآید بشکل

$$\begin{array}{cccccc}
 \underline{1} & \underline{2} & \underline{7} & \underline{5} & \underline{4} & \underline{3} \\
 \underline{1} & \underline{0} & \underline{1} & \underline{1} & \underline{0} & \underline{1} \\
 \hline
 A & F & 6 & 3
 \end{array}
 \quad \text{هشت هشتی} \\
 \quad \text{دو دو بی} \\
 \quad \text{شانزده شانزده بی}$$

**شکل ۳-۲ تبدیل دو دویی، هشت هشتی، شانزده شانزدهی**

جدول ۳-۱ اعداد هشت هشتم کد شده با دودویی

عدد	هشت هشتم	معادل دهدهی	هشت هشتم کد شده با دودویی
0	000	0	
1	001	1	
2	010	2	
3	011	3	کد یک رقم
4	100	4	هشت هشتم
5	101	5	
6	110	6	
7	111	7	
			↑
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 ۳-۲ اعداد شانزده شانزدهی کد شده با دودویی

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

1100011 خواهد بود. معادل هشت هشتی عدد دهدھی 99 برابر است با 011 001 100 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 شانزده مشخص شوند. بدین ترتیب تعداد ارقام به یک سوم در هشت هشتی، و به یک چهارم در شانزده شانزده تقلیل می یابد. مثلًا عدد دودویی 1111 1111 1111 1111 دارای 12 رقم است. این عدد در نمایش هشت هشتی بصورت 7777 (چهار رقم) و یا در نمایش شانزده شانزدهی بشكل 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 دهدھی

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

یک کد دودویی گروهی از ۱۱ بیت است که تا  $2^{11}$  ترکیب متمایز از ۰ها و ۱ها را بخود می گیرد. هر ترکیب نمایشگر یک عنصر از مجموعه ای است که قرار است بصورت کد درآید. مثلًا، مجموعه ای از چهار عنصر بوسیله یک کد دو بیتی بصورت کد درمی آیند که هر عنصر متعلق بیکی از ترکیبهای بیتی؛ ۰۰، ۰۱، ۱۰ و ۱۱ است. مجموعه ای از هشت عنصر نیازمند یک کد سه بیتی است و یک مجموعه شانزده عنصری به کد چهار بیتی احتیاج دارد و بهمین ترتیب. اگر تعداد عناصر در مجموعه توانی صحیح از ۲ نباشد، برخی از ترکیب بیت ها در کد دودویی به هیچ عنصری اختصاص نخواهد یافت. ارقام دهدھی

چنین مجموعه‌ای را تشکیل می‌دهند. یک کد دودویی که بین ده عنصر تمایز بوجود آورد باید حداقل چهاربیت داشته باشد، ولی شش ترکیب بدون استفاده خواهد ماند. کدهای متفاوت متعددی را با چهاربیت بصورت ده ترکیب مختلف و تمایز می‌توان بدست آورد. معمول‌ترین تخصیص بیت‌های بکار رفته برای ارقام ددهی همان تخصیص دودویی سراسرتی است که در ده وارده اول جدول ۳-۳ لیست شده است. این کد خاص، ددهی کد شده با دودویی<sup>۱</sup> نامیده می‌شود و غالباً با BCD نشان داده می‌شود. سایر کدهای ددهی گاهی اوقات بکار می‌روند و چند تا از آنها در بخش ۳-۵ آورده شده‌اند.

دانستن اختلاف بین تبدیل اعداد ددهی به دودویی و کد دودویی یک عدد ددهی بسیار مهم است. مثلاً، وقتی که عدد ددهی 99 به دودویی تبدیل شود حاصل 1100011 خواهد شد، ولی وقتی بصورت BCD درآید حاصل 1001 1001 خواهد بود. تنها اختلاف بین عدد ددهی که بصورت سمبلهای رقمه ۰، ۱، ۲، ...، ۹ نمایش داده شده است و سمبلهای BCD 0010، 0001، 0010، 0001 در نمادهای مورد استفاده برای نمایش ارقام است و در واقع خود عدد دقیقاً یکسان است. چند عدد ددهی 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 ددهی کد شده با دودویی (BCD)

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

الفبا و کاراکترهای خاصی را نیز شامل می‌شود. یک مجموعه از کاراکترهای الفبا عددی مشکل از مجموعه 10 عنصر ارقام دهدھی، 26 حرف الفبا و تعدادی کاراکترهای خاص مانند  $\text{æ}$  + 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<sup>۱</sup> ASCII است که از هفت بیت برای 128 کاراکتر استفاده می‌کند. کدهای دودویی برای حروف بزرگ، ارقام دهدھی، و چند کاراکتر خاص دیگر در جدول ۳-۴ لیست شده‌اند. توجه کنید که ارقام دهدھی در ASCII با حذف سه بیت مرتبه بالاتر ۰۱۱، قابل تبدیل به BCD هستند. لیست کامل کاراکترهای ASCII در جدول ۱۱-۱ 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 سمبول‌ها را تغییر می‌دهند و نه مفهوم عناصر گسته‌ای را که نمایش می‌دهند. در عملیات کامپیوترهای دیجیتال باید مفهوم بیت‌های ذخیره شده در ثبات‌ها در نظر گرفته شود تا اعمال روی عملوندی‌های 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 گسته 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 پنجم بحث خواهد شد.

### ۳-۲ مقام‌ها

در کامپیوترهای دیجیتال از متمم‌ها برای ساده‌کردن عمل تغیریق و عملیات منطقی استفاده می‌شود. در هر سیستم مبنای  $2$  دو نوع متمم وجود دارد: متمم  $2$  و متمم  $1$ . هرگاه به جای  $n$  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.

#### $r-1$ متمم

اگر عدد  $N$  در پایه  $2$  بصورت  $\bar{N}$  رقم موجود باشد، متمم  $1$  عدد  $N-1$  بصورت  $\bar{N}-1$  (تعریف می‌شود. برای اعداد دهدهی  $10=10-1=9$  و  $2=2-1=1$  است، بنابراین متمم  $9$  عدد  $N$  برابر است با  $\bar{N}-1$ ). 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  $10^4-1=9999$  و  $10000=10^4$ . نتیجه می‌شود که متمم  $9$  یک عدد دهدهی با تغیریق هر یک از ارقام آن از  $9$  بدست می‌آید. مثلاً متمم  $9$  عدد  $546700$  برابرست با  $546700-999999=453299$  و متمم  $9$  عدد  $12389$  برابرست با  $12389-99999=87610$ .

برای اعداد دودویی،  $2=1-1=0$  و  $1=1-0=1$  است، بنابراین متمم  $1$  عدد  $N$  برابر  $\bar{N}-1$  (خواهد بود. مجدداً،  $2^n$  بوسیله یک عدد دودویی نمایش داده می‌شود و مشکل است از یک  $1$  و  $n$  عدد  $0$  که بدبیال آن آمده است.  $2^n-1$  یک عدد دودویی است که بوسیله  $n$  عدد  $1$  نشان داده می‌شود. مثلاً برای  $n=4$  داریم  $2^4-1=(1111)_2$  و  $(10000)_2=2^4-1$ . بنابراین متمم  $1$  یک عدد دودویی از تغیریق هر رقم از  $1$  بدست می‌آید. با این وجود تغیریق 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  $1$ ‌ها به  $0$  و  $0$ ‌ها به  $1$  است می‌آید. مثلاً متمم  $1$  عدد  $1011001$  برابر است با  $0100110$ ، و متمم  $1$  عدد  $0001111$  عبارتست از  $1110000$ .

متمم (۱-۲) اعداد هشت هشتی یا شانزده شانزدهی بترتیب از تفریق هر رقم از ۷ یا F (۱۵ دهده) بدست می آید.

### متمم ۲

متمم ۲ یک عدد n رقمی N در پایه ۲ بازه N<sup>n</sup>-۱ و بازه 0 برابر ۰ است. می بینیم که متمم ۲ در مقایسه با متمم ۱، از جمع ۱ با متمم ۱-۲ حاصل می شود زیرا  $(r^n - 1) - N = [(r^n - 1) + 1] - N = 2389$  باشد. بنابراین متمم ۱۰ عدد دهده ۷۶۱۱+۱=۶۷۱۱ است با  $101100$  برابرست با که از جمع ۱ با مقدار متمم ۹ بدست می آید. متمم ۲ عدد دو دویی  $010011+1=010100$  که از جمع ۱ با مقدار متمم ۱ بدست آمده است.

چون  $10^n$  عددی است که با یک ۱ و بدنیال آن با  $n$  عدد ۰ نشان داده می شود، لذا  $N^{10^n}$  که متمم ۱۰ عدد N است، نیز می تواند بهمین صورت تعریف شود. بدین ترتیب که تمام ۰های کم ارزشتر را دست نخورده باقی گذاشته و اولین رقم غیر ۰ را از ۱۰ کم می کنیم، و سپس تمام رقم های مرتبه بالاتر را از ۹ کسر می نمائیم. متمم ۱۰ عدد ۲۴۶۷۰۰ برابر است با ۷۵۳۳۰۰ و بدین ترتیب حاصل می شود که دو رقم ۰۰ را باقی گذاشته، ۷ را از ۱۰ کم می کنیم، و سه رقم باقیمانده را از ۹ کسر می کنیم. بطور مشابه، برای بدست آوردن متمم ۲ می توان همه ۰های سمت راست و اولین رقم ۱ را دست نخورده باقی گذاشته، و سپس ۰ ها بجای ۱ و ۱ ها بجای ۰ در بقیه مکان های با ارزشتر گذاشته می شوند. متمم ۲ عدد ۱۱۰۱۱۰۰ برابر است با ۰۰۱۰۱۰۰ و با باقی گذاشتن دو ۰ مرتبه پائین تر و اولین ۱، و سپس با جایگزینی ۱ها با ۰ها و ۰ها با ۱ها در چهار بیت مرتبه بالاتر بدست می آید.

در تعریف فوق فرض شد که اعداد دارای معیز نیستند. اگر عدد اولیه N دارای نقطه معیز باشد، باید بطور موقت حذف گردد تا بتوان متمم های ۲ و ۱-۲ را تشکیل داد. سپس نقطه معیز 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 ۲ عدد N برابر  $N^{r^n} - 1$  است. متمم این متمم  $N^{(r^n - 1)} - 1$  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 "فرض کردن" استفاده شد. در این روش هرگاه رقمی از مفروق منه کوچکتر از رقم متناظر 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 کارآیی کمتری برخوردار است.

تفریق دو عدد  $n$  رقمی بی علامت ( $N \neq 0$ ) در مبنای ۱۰ طبقه زیر انجام می شود

۱- مفروق منه  $M$  را با متمم ۱۰ مفروق  $N$  جمع می کنیم. نتیجه این عمل عبارتست از:

$$M + (10^n - N) = M - N + 10^n$$

۲- اگر  $M \geq N$  باشد، مجموع، یک رقم نقلی  $10^n$  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  $(N - M) - 10^n$ ، که متمم ۱۰ عدد است. برای بدست آوردن جواب بصورت قابل درک، متمم ۱۰ مجموع را بدست آورده و در جلو آن یک علامت منفی می گذاریم

برای مثال، تفریق  $72532 - 13250 = 59282$  را در نظر بگیرید. متمم ۱۰ عدد ۱۳۲۵۰ برابرست با ۸۶۷۵۰ بنابراین:

$$\begin{array}{rcl} M & = & 72532 \\ N & = & \text{متمم ۱۰ عدد} \\ & = & 86750 \\ & = & \text{مجموع} \\ 10^5 & = & \frac{159282}{-100000} \\ & = & \text{با حذف رقم نقلی} \\ & = & 59282 \\ & = & \text{جواب} \end{array}$$

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

$$\begin{array}{rcl} M & = & 13252 \\ N & = & \text{متمم ۱۰ عدد} \\ & = & +27468 \\ & = & \text{جمع} \\ & = & 40718 \end{array}$$

در این مثال رقم نقلی وجود ندارد جواب نهایی  $59282$  است که برابر با متمم ۱۰ 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 دو دویی نیز مشابه روش فوق است. با بکارگیری دو عدد

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

$$\begin{array}{r}
 X = 1010100 \\
 Y = \text{متمم 2 عدد } Y \\
 = \text{مجموع} \\
 = \text{حذف رقم نقلی } 2^7 \\
 \hline
 X - Y = 0010001 \\
 Y = 1000011 \\
 \text{متمم 2 عدد } X \\
 = \text{مجموع} \\
 = 1101111
 \end{array}$$

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 بین علامت نمایش داده می شوند. معهدا 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، کامپیووترها باید هر چیزی، حتی علامت اعداد را، با ۰ها و ۱ها نمایش دهند. در نتیجه، مرسوم است که علامت را در منتهای ایه سمت چپ عدد قرار دهند. بنابراین قرارداد بیت علامت را برای اعداد مثبت برابر ۰ و برای اعداد منفی برابر ۱ در نظر می گیرند.

یک عدد، علاوه بر علامت، معکن 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. دو مکانی که بیشتر مورد استفاده اند عبارتند از: (۱) ممیز دودویی در منتهای ایه سمت چپ ثبات قرار گیرد و در نتیجه عدد ذخیره شده به یک کسر تبدیل شود؛ (۲) ممیز دودویی در منتهای ایه 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 برحورد 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 دو دویی صحیح مثبت باشد، علامت آن با ۰ و اندازه آن با یک عدد دو دویی مثبت نشان داده می شود. هرگاه عدد منفی باشد، علامت توسط ۱ ولی بقیه عدد ممکن است به یکی از سه روش زیر نمایش داده شود.

۱- نمایش مقدار (اندازه) علامت دار<sup>۱</sup>

۲- نمایش متمم ۱ علامت دار<sup>۲</sup>

۳- نمایش متمم ۲ علامت دار<sup>۳</sup>

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 ۱ و متمم ۲ مقدار مثبت عدد نمایش داده می شود. مثلاً، عدد علامت دار ۱۴ را در یک ثبات در نظر بگیرید.  $+14$  + بوسیله بیت علامت ۰ که به دنبال آن مقدار معادل دو دویی آن آمده است نشان داده می شود،  $00001110 : 14$ . توجه کنید که هر یک از هشت بیت ثبات باید مقداری داشته باشند و لذا ۰ها باید در مکان های پارز شتری که بدنبال بیت علامت آمده اند قرار گیرند. هرچند تنها یک راه برای نمایش  $+14$  وجود دارد، ولی برای نمایش ۱۴ - سه روش موجود است.

در نمایش مقدار علامت دار  $10001110$

در نمایش متمم ۱ علامت دار  $1110001$

در نمایش متمم ۲ علامت دار  $11110010$

نمایش مقدار علامت دار برای ۱۴ - فقط از متمم کردن بیت علامت بدست می آید. نمایش متمم ۱ علامت دار ۱۴ - با متمم کردن تمام بیت های  $+14$  از جمله بیت علامت، فراهم می شود. نمایش متمم ۲ علامت دار با بدست آوردن متمم ۲ عدد مثبت، از جمله بیت علامت حاصل می شود.

سیستم مقدار علامت دار در محاسبات معمولی بکار می رود، اما اگر در محاسبات کامپیوتروی بکار رود کارها گند و پیچیده خواهد شد. بنابراین معمولاً متمم علامت دار مورد استفاده قرار می گیرد. از متمم ۱ به این علت که در آن دو نحوه نمایش  $0$  وجود دارد ( $+0$  ،  $-0$ )، به ندرت، بجز در عملیات حسابی در برخی کامپیوتروهای قدیمی استفاده می شود. متمم ۱ 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 دو دویی علامت دار خواهیم کرد برای عده های منفی منحصر از نمایش متمم ۲ علامت دار استفاده خواهد شد.

1- signed – magnitude

2- signed – 1's complement

3- Signed – 2's Complement

### 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  $(+25) - (-37) = (+25) + (-37) = 62$  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 ۱۰-۲ 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 (متنها لیه 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.

$$\begin{array}{r}
 + 6 & 00000110 \\
 + 13 & 00001101 \\
 \hline
 + 19 & 00010011
 \end{array}
 \quad
 \begin{array}{r}
 - 6 & 11111010 \\
 + 13 & 00001101 \\
 \hline
 + 7 & 00000111
 \end{array}$$
  

$$\begin{array}{r}
 + 6 & 00000110 \\
 - 13 & 11110011 \\
 \hline
 - 7 & 11111001
 \end{array}
 \quad
 \begin{array}{r}
 - 6 & 11111010 \\
 - 13 & 11110011 \\
 \hline
 - 19 & 11101101
 \end{array}$$

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 نا آشناست . برای تعیین مقدار یک عدد منفی وقتی که بشکل متمم 2 علامت دار باشد، لازم است به عدد مثبتی تبدیل شود تا به فرم آشنایی درآید. مثلاً عدد دودویی علامت دار 11111001 منفی است زیرا سمت چپ ترین بیت 1 است. متمم 2 آن 11111000 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 بگیرید (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 نمایید. این رویه از این واقعیت نتیجه شده است که یک عمل تفریق می‌تواند به یک جمع بدل شود بشرطی که علامت مفروق عوض شود. این مطلب با رابطه زیر نشان داده شده است.

$$\begin{aligned} (\pm A) - (+B) &= (\pm A) + (-B) \\ (\pm A) - (-B) &= (\pm A) + (+B) \end{aligned}$$

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 می‌دهد. تفریق  $= +7 - (-6)$  را در نظر بگیرید. این تفریق بصورت هشت بیت دودویی بفرم  $11110011 - 11110101 = 11110111$  نوشته می‌شود. با گرفتن متمم 2 عدد  $(-13)$  که  $(+13) - (-13)$  را می‌دهد تفریق به جمع تبدیل می‌شود و در نتیجه داریم  $11110111 + 00001101 = 100000111$ . با حذف رقم نقلی انتهایی، پاسخ صحیح  $00000111$  یعنی  $(+7)$  بدست می‌آید.

اشارة به این نکته لازم به نظر می‌رسد که بگوئیم جمع و تفریق اعداد متمم 2 علامت دار با همان قواعد جمع و تفریق اعداد بی‌علامت صورت می‌گیرد. 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 بی‌علامت بگونه‌ای متفاوت تفسیر کند.

### سرریز<sup>۱</sup>

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 مسئله‌ای را ایجاد نمی‌کند زیرا محدودیتی در عرض صفحه برای نوشتن حاصل جمع وجود ندارد. سرریز در کامپیوترهای دیجیتال ایجاد اشکال می‌کند زیرا عرض ثبات‌ها محدودیت دارد. نتیجه‌ای که  $n+1$  بیتی باشد را نمی‌توان در ثباتی که دارای عرض استاندارد  $n$  بیت است جای داد. با این دلیل اغلب کامپیوترها رخداد یک سرریز را ردیابی می‌کنند، و وقتی که اتفاق بیفتد، فلایپ فلاپ بخصوصی 1 می‌شود که کاربر می‌تواند 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 باهم جمع شوند، سرریز هنگامی آشکار می‌شود که رقم نقلی از با ارزش ترین مکان خارج شود. در حالت علامت دار بودن عدد، سمت چپ ترین بیت همیشه علامت را نمایش می‌دهد 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 نتیجه‌ای را تولید می‌کند 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 شده‌اند دامنه اعدادی که هر ثبات می‌تواند در خود جای دهد از عدد دودویی  $127 + 128 = 255$  است. چون مجموع  $150 + 150 = 300$  است، از ظرفیت ثبات هشت بیتی تجاوز می‌کند. این وضعیت هنگامی رخ می‌دهد که اعداد هر دو مثبت یا منفی باشند. دو جمع دودویی در زیر همراه با دو رقم نقلی آخر نشان داده شده‌اند.

ارقام نقلی: 01	ارقام نقلی: 10
$+70 \quad 0 \ 1000110$	$-70 \quad 1 \ 0111010$
$+80 \quad 0 \ 1010000$	$-80 \quad 1 \ 0110000$
$+150 \quad \underline{1 \ 0010110}$	$-150 \quad \underline{0 \ 1101010}$

توجه کنید که نتیجه هشت بیتی که می‌باید مثبت باشد، دارای بیت علامت منفی است و نتیجه هشت بیتی که باید منفی می‌بود 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 نمی‌تواند در هشت بیت جای گیرد گروئیم سرریز رخ داده است:

حالت سرریز را با مشاهده رقم نقلی واردہ به بیت علامت و نقلی خارج شدن از بیت علامت می‌توان درک کرد. اگر این دو نقلی با هم مساوی نباشد، یک وضعیت سرریز تولید شده است. این مطلب در مثال‌های فوق که هر دو رقم نقلی نشان داده شده‌اند دیده می‌شود. چنانچه دو رقم نقلی به یک گیت OR انحصاری اعمال شوند، سرریز هنگامی رخ داده است که خروجی این گیت 1 شود.

### نمایش مهمیز - ثابت دهد

نمایش اعداد دهدی در ثبات‌ها تابعی از کد دودویی مورد استفاده برای نمایش رقم دهدی است. یک کد دهدی چهاربیتی نیاز به چهار فلیپ‌فلاب برای هر رقم دهدی دارد. نمایش 4385 در BCD به 16 فلیپ‌فلاب نیاز دارد. این عدد در یک ثبات با 16 فلیپ‌فلاب نمایش داده می‌شود.

0100 0011 1000 0101

با نمایش اعداد به شکل دهدی مقدار قابل توجهی از فضای ذخیره‌سازی تلف می‌شود زیرا تعداد بیت‌های لازم برای ذخیره سازی یک عدد دهدی در کد دودویی بزرگتر از تعداد بیت‌های لازم در نمایش معادل دودویی آن است. همچنین، مدارهای لازم برای انجام محاسبات دهدی پیچیده‌تر هستند. با این

وجود، استفاده از نمایش دهدۀ دارای مزایایی است زیرا داده‌های ورودی و خروجی کامپیوتر بوسیله انسانها که سیستم دهدۀ 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 دهدۀ کم است. به همین دلیل برخی از کامپیوتر و ماشین حساب‌ها اعمال حسابی را مستقیماً با داده‌های دهدۀ کم است. ( بصورت یک کد دودویی) انجام می‌دهند، و بنابراین در آنها نیازی به تبدیل به دودویی و تبدیل مجدد به دهدۀ وجود ندارد. بعضی از سیستم‌های کامپیوتری دارای بخش سخت افزاری برای هر دو نوع محاسبه حسابی بروی داده‌های دودویی و دهدۀ می‌باشند.

نمایش اعداد دهدۀ 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 دهدۀ معمولاً با چهار بیت نشان داده می‌شود تا با کد 4 بیتی ارقام دهدۀ مطابقت داشته باشد. معمول است که علامت مثبت را با چهار ۰ و علامت منفی را با معادل BCD عدد ۹ که ۱۰۰۱ است نشان دهند. بکارگیری سیستم اعداد مقدار علامت دار در کامپیوترها مشکل است. سیستم متمم علامت دار هم در فرم متمم ۹ و یا متمم ۱۰ می‌باشد. ولی سیستم متمم ۱۰ علامت دار بیشتر استفاده می‌شود. برای بدست آوردن متمم ۱۰ یک عدد BCD، ابتدا متمم ۹ را بدست آورده و سپس به کم ارزش‌ترین بیت یک واحد اضافه می‌کنیم. متمم ۹ نیز از تفریق هر رقم از ۹ حاصل می‌گردد.

روشهای بدست آمده برای سیستم متمم ۲ علامت دار در مورد سیستم متمم ۱۰ علامت دار برای اعداد دهدۀ 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 براین است که همه اعداد منفی به شکل متمم ۱۰ هستند. به جمع  $+135 = +375 - (-240)$  که در سیستم متمم ۱۰ علامت دار انجام شده توجه کنید.

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

عدد ۹ در سمت چپ ترین مکان عدد دوم نشان دهنده منفی بودن آنست. عدد ۹۷۶۰ متمم ۱۰ عدد ۰۲۴۰ می‌باشد. دو عدد با یکدیگر جمع شده و رقم نقلی نیز چشم پوشی شده است تا  $+135 + 9760 = 11018$  حاصل گردد. البته اعداد دهدۀ ارقام علامت، در داخل کامپیوتر باید بشكل BCD باشند. عمل جمع با جمع کننده‌های BCD صورت می‌گیرد (شکل ۱۸-۱۰ را ملاحظه کنید).

تفریق اعداد دهدۀ بدون علامت یا سیستم متمم ۱۰ علامت دار همانند حالت دودویی است. متمم ۱۰ مفروق را با مفروق منه جمع می‌کنیم. بسیاری از کامپیوترها سخت افزار خاصی را برای انجام مستقیم محاسبات حسابی با اعداد دهدۀ دارند. کاربر کامپیوتر می‌تواند با کمک دستورالعمل‌هایی در برنامه مشخص کند که اعمال حسابی مستقیماً با اعداد دهدۀ دهدۀ و بدون تبدیل آنها به دودویی انجام شود.

### ۳-۴ نمایش ممیز شناور<sup>۱</sup>

نمایش ممیز شناور یک عدد دارای دو بخش است. بخش اول یک عدد ممیز ثابت علامت دار است که مانتیس<sup>۲</sup> خوانده می شود. بخش دوم محل ممیز دهدہ (یا دودویی) را معین می کند و نما<sup>۳</sup> نام دارد. مانتیس ممیز ثابت، ممکن است یک عدد کسری و یا صحیح باشد. مثلاً عدد دهدہ 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<sup>۴</sup> و محل ممیز در مانتیس همیشه بطور ضمنی مساوی با مقدار معینی فرض می شوند. مدارهایی که برروی 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  $1001.11_2 +$  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 منتهای 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 ۴+ است. عدد ممیز شناور معادل است با

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

یک عدد ممیز شناور، در صورتی که پرازشترین رقم مانتیس غیر صفر باشد، نرمالیزه<sup>۵</sup> (یا بهنجار)

1- Floating point  
4- Radix

2- Mantissa  
5- Normalized

3- Exponent

خوانده می شود. مثلاً عدد دهدھی 350 نرمالیزه است ولی 00035 نرمالیزه نیست. صرف نظر از اینکه محل ممیز در کجای ماتریس فرض شود، عدد هنگامی نرمالیزه است که رقم منتهای سمت چپ آن غیرصفر باشد. مثلاً عدد دودویی 00011010 به دلیل داشتن سه ۰ در ابتدای آن نرمالیزه نیست. این عدد با سه بار شیفت به چپ و چشم پوشی از علای ابتدای آن نرمالیزه می شود و عدد حاصل 11010000 خواهد بود. سه بار عمل شیفت عدد را در  $8^3$  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 غیرصفر نیست. این عدد در نمایش ممیز شناور با ماتریس و نمای تمام ۰ 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 قادر ساخت 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 ۱۰-۵ بحث شده اند.

### ۳-۵ انواع دیگر کدهای دودویی

در بخش های قبلی متدائل ترین نوع داده های کد شده دودویی در کامپیوترهای دیجیتال را معرفی کردیم. گاهی اوقات از کدهای دیگری هم برای اعداد دهدھی 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.

#### کدگوی<sup>۱</sup>

سیستم های دیجیتال فقط قادرند داده ها را بشكل گسته 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<sup>۲</sup> بکمک مبدل های آنالوگ به دیجیتال به کمیت های دیجیتال بدل می شوند. از کد انعکاسی<sup>۳</sup> 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 و یا از 1 به 0 است. یکی از کاربردهای کدگری هنگامی است که داده‌های آنالوگ نشان دهنده تغییر مکان پیوسته یک محور دوران<sup>۱</sup> باشد. محور به قسمت‌هایی تقسیم بندی می‌شود و به هر قسمت عددی تخصیص می‌یابد. اگر قطعات مجاور، متناظر با عدددهای مجاور در کدگری باشند، ابهام در موقعیت شفت هنگامی که در وضعیت بین دو نقطه قرار گیرد کمتر خواهد شد.

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

### کدهای دهدۀ دیگر

کدهای دودویی برای ارقام دهدۀ حداقل به چهار بیت نیاز دارند. کدهای متعدد مختلفی را با کمک چهار بیت 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 سمبول‌های ارقام دهدۀ مفهوم ندارد. مثلاً بیان اینکه 1101 1110 یک عدد دهدۀ BCD است مانند این است که بگوئیم 9H یک عدد دهدۀ است که با سمبول‌های مرسوم نوشته شده باشد. در هر دوی اینها یک سمبول 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 دهدی

رقم دهدی	BCD 8421	2421	افزونی ۳	افزونی ۳ گری
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 پلاستفاده بیت ها				
	1010	0101	0000	0000
	1011	0110	0001	0001
	1100	0111	0010	0011
	1101	1000	1101	1000
	1110	1001	1110	1001
	1111	1010	1111	1011

عدد بی معنی را نشان می دهدند.

یکی از معاوی استفاده از BCD مشکلات ناشی از محاسبه متمم ۹ عدد است. از طرف دیگر، متمم ۹ در کدهای 2421 و افزونی ۳<sup>۱</sup> که در جدول ۳-۶ نشان داده شده اند، به آسانی قابل محاسبه است. این دو کد دارای خاصیت خود متممی<sup>۲</sup> هستند. بدآن معنی که متمم ۹ عدد دهدی هنگامی که بیکی از این دو نوع تبدیل شود براحتی با تعویض ۰ ها به ۱ و ۱ ها به ۰ بدست می آید. این خاصیت هنگام انجام عملیات حسابی در نمایش متمم علامت دار مفیدند.

2421 مثالی از کدهای وزن دار<sup>۳</sup> است. در یک کد وزن دار بیتها در وزنهای مشخص شده ضرب می شوند، و مجموع بیتها وزن دار شده رقم دهدی را می دهد. مثلاً ترکیب بیتی 1101 وقتی که در رقمهای مربوطه در 2421 ضرب شوند، معادل دهدی این ترکیب یعنی  $2 \times 1 + 4 \times 1 + 2 \times 0 + 1 \times 1 = 7$  بددست می آید. به کد BCD می توان وزنهای 8421 را اختصاص داد و بهمین دلیل کد 8421 خوانده می شود

کد افزونی ۳ یک کد دهدی است که در کامپیوترهای قدیمی بکار می رفته است. این یک کد غیر وزن دار است. کد دودویی اختصاص یافته این کد از عدد دودویی BCD معادل آن پس از جمع با عدد (0011)3 بدست می آید.

با توجه به جدول ۳-۵ می بینیم که کدگری برای ۱۰ واردہ اول جدول، کد دهدھی مناسبی نیست. این بدان علت است که انتقال از ۹ به ۰ مستلزم تغییر سه بیت می باشد (از ۱۱۰۱ به ۰۰۰۰). برای غلبه براین مشکل، ما ۱۰ عددرا، که از سومین واردہ ۰۰۱۰ شروع و به ۱۰۱۰ ختم می شود، انتخاب می کنیم. حال گذر از ۱۰۱۰ به ۰۰۱۰ تنها تغییر یک بیت را موجب می شود. چون کد به میزان سه عدد به بالا شیفت داده شده است کدگری افزونی ۳ خوانده می شود. این کد به همراه سایر کدهای دهدھی در جدول ۳-۶ آورده شده است.

### کدهای الفباعدی دیگر

کد ASCII (جدول ۳-۴) کد استانداری 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 ۳-۶ ملاحظه شود). کد دارای ۱۲۸ کاراکتر می باشد. نود و پنج کاراکتر سمبول های گرافیکی<sup>۱</sup> را نمایش می دهند که شامل حروف بزرگ و کوچک، ارقام ۰ تا ۹، علامت نقطه گذاری و سمبول های خاص هستند. بیست و سه کاراکتر نماینده شکل دهنده ها<sup>۲</sup> می باشند که شامل کاراکتر های عملیاتی برای کنترل دستگاه های چاپ و یا نمایشگرها از قبیل برگشت نورد<sup>۳</sup>، تعویض سطر<sup>۴</sup>، جدول بندی افقی<sup>۵</sup>، و پسبر<sup>۶</sup> می باشند. ۱۰ 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<sup>۷</sup> می باشد. در این کد از هشت بیت برای هر کاراکتر استفاده می شود (بیت نهم متعلق به توازن است). کاراکتر های EBCDIC همان کاراکتر های ASCII می باشند ولی بیت های اختصاص یافته به آنها متفاوت است.

هرگاه کاراکتر های الفباء عددی برای پردازش داخلی در کامپیوتر بکار روند (ونه به منظور ارسال داده ها) بهتر است از یک کد ۶ بیتی برای نمایش ۶۴ کاراکتر استفاده شود. با یک کد ۶ بیتی می توان ۲۶ حرف بزرگ، ارقام ۰ تا ۹، و تا ۲۸ 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 الفباء کاهش می یابد.

### ۳-۶ کدهای آشکارسازی خطأ

اطلاعات دودویی انتقالی از طریق برخی از وسایل ارتباطی در معرض پارازیت های<sup>۸</sup> خارجی قرار دارند که ممکن است بیت ها را از ۰ به ۱ و بالعکس تبدیل کند. کد کشف خطأ 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 رقمنی را در طول ارسال کشف می‌کند. خطاهای درک شده را نمی‌توان اصلاح کرد و فقط وجود آنها معلوم می‌شود. معمول این است که تواتر خطاهای مشاهده شود. اگر خطاهای غیرمتوالی و تصادفی رخ دهند، اطلاعات خاصی که خطای دارد مجدداً ارسال می‌گردد. اگر خطای بیش حد معقول رخ دهد سیستم از نظر وجود اشکال چک می‌شود.

متداول‌ترین کد کشف خطای بیت توازن<sup>۱</sup> است. بیت توازن بیتی است اضافی که به یک پیام دو دویی اضافه می‌شود تا تعداد ۱ها 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 ۱ها را فرد نماید (در تمام چهاریت). بیت زوج P چنان اختیار می‌شود تا مجموع تمام ۱ها را زوج نماید. در هر صورت مجموع ۱ها 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 ۰ خواهد داشت، در حالی که در توازن فرد همیشه یکی از بیت‌ها (از چهاریت) که پیام و P را تشکیل می‌دهند) ۱ خواهد بود. وقت 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<sup>۲</sup> 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 چهاریت) به یک چک کننده توازن<sup>۳</sup> اعمال می‌شوند تا توازن انتخابی را چک کند (فرد یا زوج). اگر توازن بیت‌ها با توازن انتخابی مغایرت داشته باشد، خطای اعلام می‌شود. روش توازن می‌تواند وجود یک، سه یا هر تعداد فردی از خطای را کشف کند. اگر تعداد خطاهای زوج باشند کشف نمی‌شوند.

شبکه‌های مولد و چک کننده توازن مدارهایی منطقی هستند که با توابع OR انحصاری ساخته می‌شوند. دلیل این امر این است که، همانطور که در بخش ۱-۲ 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

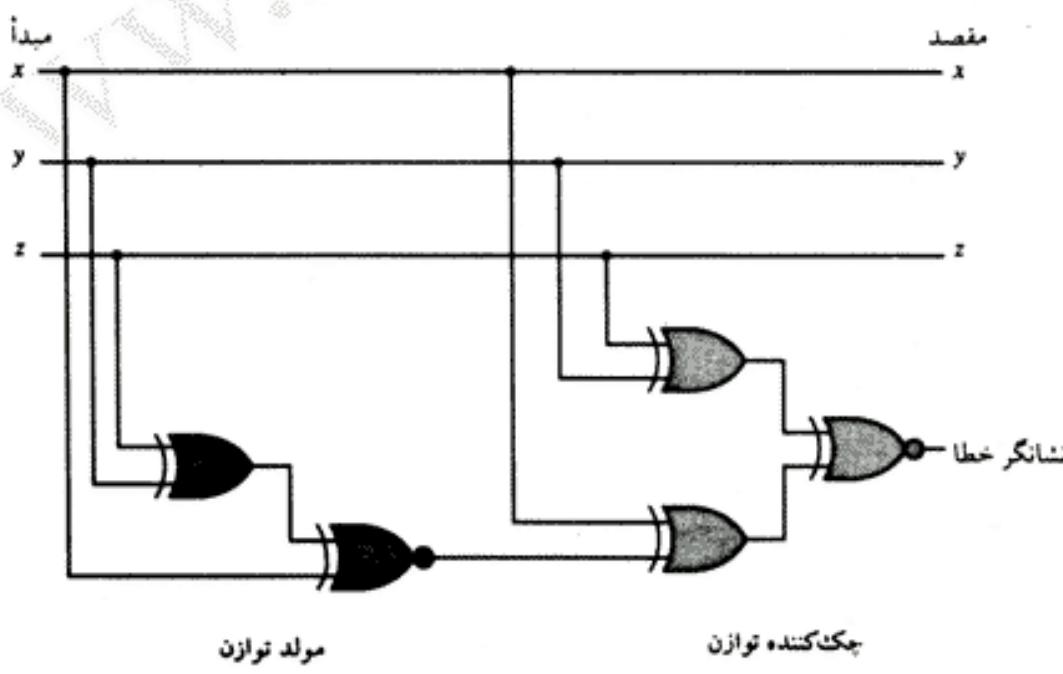
3- Parity Checker

چند متغیر بنا به تعریف یک تابع فرد است. تابع فرد تابعی منطقی است که مقدار آن ۱ دودویی است، اگر و فقط اگر تعداد فردی از متغیرها ۱ باشند. طبق این تعریف، تابع زوج P عبارتست از OR انحصاری  $x, y$  و  $z$ ، زیرا هر گاه یکی از این سه متغیر ۱ یا هر سه آنها ۱ باشند، تابع برابر ۱ خواهد بود (جدول ۳-۷). تابع فرد متمم تابع زوج است.

عنوان 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 می‌شود. این مدار، همانطور که در شکل ۳-۳ دیده می‌شود، 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 می‌شوند. در حین ارسال پیام اگر توازن چهاربیت رسیده زوج باشد خطایی رخ داده است، زیرا اطلاعات ارسالی ابتدأ فرد بوده است. اگر خطایی رخ داده باشد خروجی چک کننده توازن ۱ خواهد شد، یعنی تعداد ۱‌های چهار ورودی زوج است. چون تابع OR انحصاری چهارورو دی یک تابع فرد است، ما می‌باید مجدداً با استفاده از یک گیت NOR انحصاری، خروجی را متمم کنیم.

ذکر این نکته مفید است که بدانیم مولد توازن می‌تواند همان مدار چک کننده توازن را داشته باشد بشرطی که ورودی چهارم دائماً در ۰ منطقی نگهداشته شود. مزیت این خاصیت این است که برای تولید و چک کردن توازن می‌توان از یک مدار استفاده کرد.

با توجه به مثال فرق واضح است که مولدها و چک کننده‌های توازن با توابع OR انحصاری پیاده سازی می‌شوند. مدارات توازن فرد برای متمم کردن خروجی تابع به یک NOR انحصاری نیاز دارند.



شکل ۳-۲ کشف خطا بوسیله بیت توازن فرد

## مسائل

۳-۱ اعداد دو دویی زیر را به دهدی تبدیل کنید: 110110100; 1110101; 101110;

۳-۲ اعداد زیر را که در آنها پایه هایشان مشخص شده است را به دهدی تبدیل کنید: 3(12121); 5(4310); 7(50); 12(198).

۳-۳ اعداد دهدی زیر را به دو دویی تبدیل کنید: 1231; 673 و 1998.

۳-۴ اعداد دهدی زیر را به مبنای های مشخص شده تبدیل کنید

الف) 7562 به هشت هشتی      ب) 1938 به شانزده شانزدهی      ج) 175 به دو دویی

۳-۵ عدد شانزده شانزدهی F3A7C2 را به دو دویی و هشت هشتی تبدیل نمایید.

۳-۶ اگر جواب معادله  $0 = 31 - 10x - x^2$  برابر  $x=5$  باشد. پایه عدد  $x$  چیست؟

۳-۷ مقدار همه بیت های یک ثبات 12 بیت که عددی معادل 215 دهدی را نگه میدارد:

(الف) به دو دویی؛ (ب) هشت هشتی کد شده با دو دویی، (ج) شانزده شانزدهی کد شده با دو دویی

(د) دهدی کد شده با دو دویی (BCD)، نشان دهید.

۳-۸ آرایش یک ثبات 24 بیتی را که محتوا یش معادل یک عدد دهدی 295 است؛ (الف) به دو دویی؛

(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 نویش و بیت سمت چپ منتهایی را در

نظر نگیرید. 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 دیگر کنید

1001010 1001111 1001000 1001110 0100000 1000100 1001111 1000101

۳-۱۱ متمم ۹ اعداد دهدی هشت رقمی زیر را بدست آورید.

12349876 ; 00980100 ' 90009951 ; , 00000000

۳-۱۲ متمم 10 اعداد دهدی شش رقمی زیر را بدست آورید.

123900; 090657 ; 100000; , 000000

۳-۱۳ متمم های 1 و 2 اعداد دو دویی هشت رقمی زیر را بدست آورید

10101110 ; 10000001 ; 10000000; 00000001; 00000000

۳-۱۴ تفریق را با اعداد دهدی بدون علامت زیر و با استفاده از متمم 10 مفروق انجام دهید.

الف) 5250-1321      ب) 1753-8640

د) 1200-250      ج) 20-100

۳-۱۵ تفریق را با اعداد دو دویی بدون علامت زیر و با استفاده از متمم 2 مفروق انجام دهید.

الف) 11010-10000      ب) 11010-1101

د) 1010100-1010100      ج) 100-110000

- ۳-۱۶ اعمال حسابی  $(-13) + (+42)$  و  $(-42) - (-13)$  را در دو دویسی و با استفاده از متمم 2 علامت دار برای اعداد منفی انجام دهید.
- ۳-۱۷ اعمال حسابی  $(+80) + (+70)$  و  $(-70) + (-80)$  را با نمایش اعداد دو دویسی 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 سریز رخ می دهد، دو نقلی آخر نامساویند، و علامت معکوس می شود.
- ۳-۱۸ اعمال حسابی زیر را برای اعداد دهدی مشخص شده با استفاده از نمایش متمم 10 علامت دار برای اعداد منفی انجام دهید
- الف)  $(+785) + (-638)$   
ب)  $(-638) - (+785)$
- ۳-۱۹ یک عدد دو دویسی ممیز شناور 36 بیتی دارای 8 بیت بعلاوه علامت برای نما و 26 بیت بعلاوه علامت برای مانیس می باشد. مانیس یک کسر نرمالیزه شده است. اعداد مانیس 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<sub>(+)46.5</sub> را بصورت یک عدد دو دویسی ممیز شناور 24 بیتی نشان دهید. مانیس که یک کسر نرمالیزه است 24 بیت و نما 8 بیت دارند.
- ۳-۲۱ کدگری گاهی اوقات کد انعکاسی خوانده می شود زیرا مقادیر بیت ها در دو طرف هر یک از توان های 2<sup>n</sup> 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 ۵-۳  
ب) کدگری افزونی 3 برای اعداد دهدی 10 تا 19 به عنوان ادامه جدول ۳-۶
- ۳-۲۲ عدد دهدی 8620 را به شکل (الف) BCD؛ (ب) افزونی 3؛ (ج) کد 2421؛ (د) عدد دو دویسی نمایش دهید.
- ۳-۲۳ ده رقم BCD را با توازن زوج در متها ایه سمت چپ آن (کلاً پنج بیت در هر رقم) نشان دهید. عمل را با توازن فرد تکرار کنید.
- ۳-۲۴ عدد دهدی 3984 را بصورت کد 2421 جدول ۳-۶ 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  $x = y \oplus z$  و  $z = C \oplus D$  را بدست آورده و سپس جدول درستی  $x = y \oplus z$  بدست آورید. نشان دهید که 1 است بشرطی که تعداد 1 ها در A، B و C و D فرد باشد.
- ۳-۲۶ مدارهای یک مولد توازن سه بیتی و چک کننده توازن چهار بیتی را با استفاده از بیت توازن زوج بدست آورید (مدارات شکل ۳-۳ 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 مازول‌های<sup>۱</sup> سخت افزاری متصل بهم است که کاری خاص را در زمینه پردازش اطلاعات انجام می‌دهند. سیستم‌های دیجیتال از نظر اندازه و پچیدگی از چند مدار مجتمع تا مجموعه‌ای از کامپیوترهای مرتبط متغیرند. در طراحی سیستم‌های دیجیتال از روش مازولی استفاده می‌شود. مازول‌ها از اجزایی<sup>۲</sup> چون ثبات‌ها، دیکدرها عناصر حسابی و کنترل منطقی ساخته می‌شوند. مازول‌های مختلف با مسیرهای مشترک داده و کنترل به هم متصل می‌شوند تا یک سیستم کامپیوترا دیجیتال بوجود آید.

مازول‌های 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<sup>۳</sup> 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<sup>۴</sup>، شمارش<sup>۵</sup>، پاک کردن<sup>۶</sup> و بارکردن<sup>۷</sup>

۱- Module  
۵- Count

2- Component  
6- Clear

۳- Microoperation  
7- Load

4- Shift

است. برخی از مؤلفه های دیجیتال که در فصل 2 معرفی شدند ثبات هایی هستند که ریز عمل ها را پیاده سازی می کنند. مثلاً، یک شمارنده با قابلیت بارشدن موازی می تواند ریز عمل های افزایش و بارشدن را اجرا کند. یک ثبات شیفت (جابجایی) دو طرفه قادر است ریز عمل های شیفت به راست و چپ را انجام دهد. سازمان داخلی یک کامپیوتر دیجیتال به بهترین نحو توسط موارد زیر مشخص می شود:

۱- مجموعه ثبات های آن و وظایف آنها

۲- رشته ریز عمل های انجام شده روی اطلاعات دودویی ذخیره شده در ثبات ها

۳- واحد کنترلی که موجب آغاز رشته ریز عمل ها می شود.

رشته ریز عمل ها در کامپیوتر را می توان با تشریح لفظی هر عمل مشخص کرد، ولی این روش معمولاً تشریحی طولانی خواهد بود. بهتر آنست تا روشی سمبولیک را برای توصیف رشته انتقال ها بین ثبات ها و ریز عمل های حسابی و منطقی مختلف مربوط به این انتقال ها، مشخص کنیم. استفاده از سمبول بجای توضیحات، روشی سازمان یافته و فشرده ای را برای نشان دادن رشته ریز عمل ها در ثبات ها و توابع کنترلی که موجب اجرای آنها می شوند فراهم می کند.

نحوه بیان سمبولیک مورد استفاده برای بیان انتقال های ریز عملی در بین ثبات ها، زیان انتقال ثباتها خواهد می شود. اصطلاح "انتقال ثباتها" بیانگر وجود مدارات منطقی سخت افزاری است که می تواند یک ریز عمل بیان شده را اجرا نماید و نتیجه عمل را به همان ثبات یا ثبات دیگر انتقال می دهد. کلمه "زیان" از برنامه نویسان اقتباس شده است، که این کلمه را برای زیان های برنامه نویسی بکار می بردند. زیان برنامه نویسی رویه ای برای نوشتمن سمبول هایی است که فرآیند محاسباتی خاصی را مشخص می کند. بطور مشابه، یک زیان طبیعی مانند انگلیسی، سیستمی است برای نوشتمن سمبول ها و ترکیب آنها بصورت کلمات و جملات برای ارتباط بین انسانها. زیان انتقال ثبات هم سیستمی است برای بیان رشته ریز عمل ها بین ثبات های یک مازول دیجیتال بصورت سمبولیک. چنین زیانی ابزار مناسبی برای توصیف فشرده و دقیق سازمان داخلی کامپیوترهای دیجیتال است. همچنین می توان از آن برای تسهیل روند طراحی سیستم های دیجیتال استفاده کرد.

اعتقاد براین است که زیان انتقال ثبات بکار رفته تا حد امکان ساده باشد، بنابراین بخارط سپردن آن طولی نخواهد کشید. ما همچنان به تعریف سمبول برای انواع ریز عمل ها ادامه خواهیم داد، و همزمان با آن سخت افزاری که ریز عمل بیان شده را پیاده سازی کند معرفی می کنیم. علائم سمبولیک معرفی شده در این فصل، در فصل های بعدی برای مشخص کردن انتقالات ثبات ها، ریز عمل ها، و توابع کنترلی که سازمان سخت افزاری داخلی کامپیوترهای دیجیتال را بیان می کنند، بکار می روند. به محض آشنایی با این زیان، سایر سمبول های بکار رفته بسادگی قابل آموختن است زیرا بیشتر تفاوت های بین زیان های انتقال ثبات در جزئیات آنهاست، نه در کلیات.

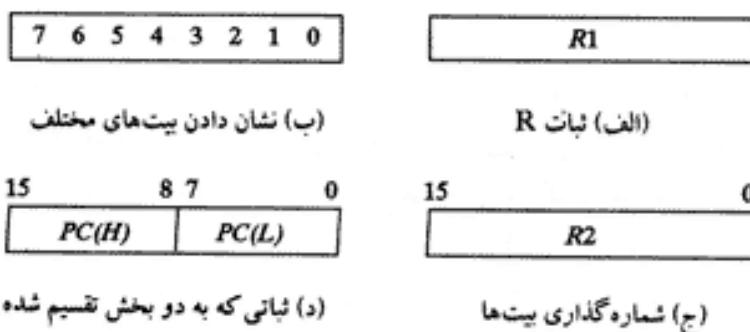
## ۴-۲ انتقال ثبات

ثبات‌های کامپیوتر با حروف بزرگ الفبای انگلیسی (وگاهی عددهایی به دنبال آنها) برای نشان دادن کارثبات مشخص می‌شوند. مثلاً، ثباتی که آدرس را برای یک واحد حافظه نگهدارد ثبات آدرس حافظه<sup>۱</sup> نام داشته و با MAR مشخص می‌شود. سایر نام‌ها برای ثبات‌ها عبارتند از PC (برای شمارنده برنامه)<sup>۲</sup>، IR (برای ثبات دستورالعمل)<sup>۳</sup>، و R1 (برای ثبات پردازنده). فلیپ فلاپ‌های تشکیل دهنده یک ثبات n بیتی به طور متوالی از ۰ الی n-1 راست به چه شماره‌گذاری می‌شوند. شکل ۴-۱ 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 ۴-۱ (الف). بیت‌های مختلف ثبات مانند قسمت (ب) در شکل متمایز می‌شوند. شماره‌گذاری 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 ۰ تا ۷ علامت L (برای بایت کم ارزشتر) و به بیت‌های ۸ تا ۱۵ سمبول H (برای بایت بالریزشتر) بکار می‌رود. نام ثبات 16 بیتی PC(0-7) است. سمبول (PC(0-7)) به بایت کم ارزشتر اطلاق می‌شود و PC(H) یا PC(L) به بایت بالریزشتر اشاره می‌کند.

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

$$R2 \leftarrow R1$$

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



شکل ۴-۱ 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.

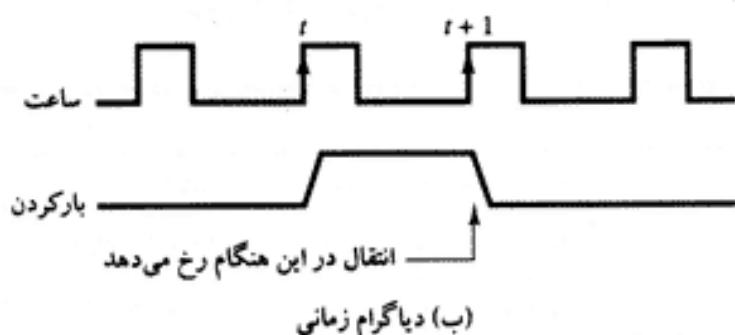
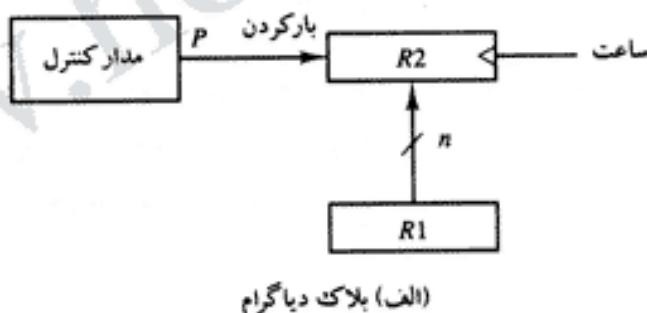
If ( $P = 1$ ) then ( $R2 \leftarrow R1$ )

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 ۱ یا ۰ است. تابع کنترل در عبارت بصورت زیر مشخص می شود

$P: P2 \leftarrow R1$

شرط کنترل با علامت نقل<sup>۱</sup>،:، خاتمه می باید. این سمبول بیان می دارد که لازمه عمل انتقال توسط سخت افزار این است که  $P=1$  باشد.

هر یک از عبارت های نوشته شده به زیان انتقال ثبات ها، دلالت بر وجود یک ساختمان سخت افزاری برای پیاده سازی انتقال دارد. شکل ۴-۲، انتقال از  $R1$  به  $R2$  را بصورت بلاک دیاگرام نمایش می دهد. 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



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

1- Colon

می شود. ثبات R2 دارای ورودی بارکردن است که بوسیله متغیر کنترل P فعال می شود. فرض براین است که متغیر کنترل با همان ساعتی که به ثبات اعمال می شود همگام<sup>۱</sup> است. همانطور که در دیاگرام زمانبندی نشان داده شده است، P در بخش کنترل با لبه بالا رونده پالس ساعت در زمان ۱ 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 ۰ بازگردد؛ در غیراین صورت، عمل انتقال با هر گذر پالس ساعت مدامی که P فعال است تکرار می شود.

توجه کنید که پالس ساعت بعنوان یک متغیر در عبارت انتقال ثبات آورده نشده است. فرض براین است که تمام انتقال ها در یک لبه گذر پالس ساعت رخ می دهند. هرچند که شرایط کنترلی مانند P درست پس از زمان ۱ 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 ۴-۱ 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<sup>۲</sup> برای جدا کردن دو یا چند عمل که همزمان انجام می شوند استفاده می شود. عبارت

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

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

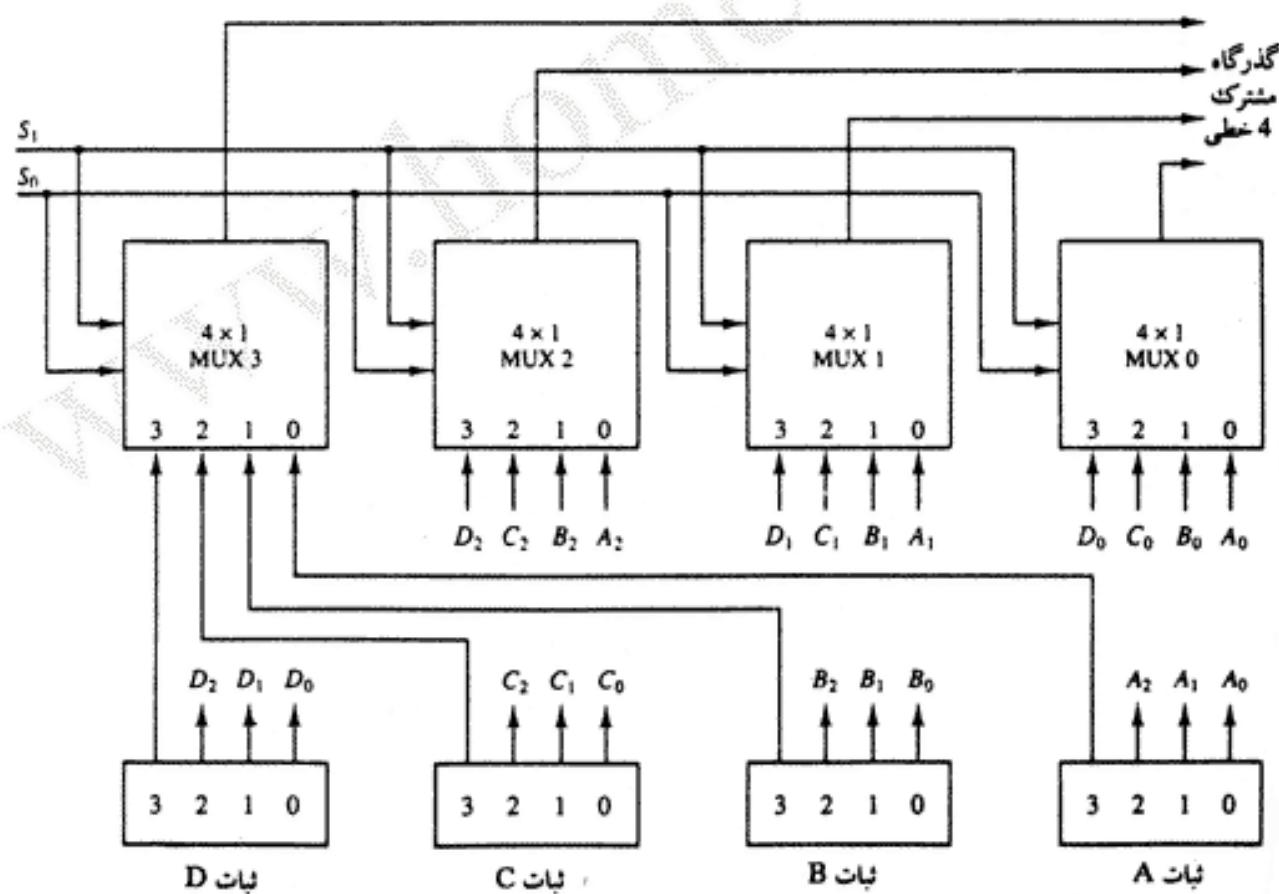
جدول ۴-۱ سمبل های اصلی انتقال ثبات ها

مثال	توضیح	سمبل
MAR R2	یک ثبات را مشخص می کند	حروف
R2(0-7) R2(L)	بخشی از یک ثبات را مشخص می کند	(وارقام) 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

### ۴-۳ انتقال های گذرگاهی و حافظه ای

یک کامپیوچر دیجیتال نوعاً ثبات های زیادی دارد و باید در آن مسیرهایی برای انتقال اطلاعات از یک ثبات به ثبات دیگر فراهم شود. اگر خطوط جداگانه ای بین هر ثبات و دیگر ثبات های سیستم به کار رود، تعداد سیم ها بیش از حد خواهد شد. روش کاراتر برای انتقال اطلاعات بین ثبات ها در یک آرایش چند ثباتی، سیستم گذرگاه مشترک است. ساختار یک گذرگاه از مجموعه ای از خطوط مشترک تشکیل می شود، که تعداد آنها، یک خط بازی هر یک از بیت های ثبات هاست، و از طریق آنها اطلاعات دودویی بکی یک انتقال می یابند. سیگنال کنترل، تعیین کننده ثبات انتخاب شده بوسیله گذرگاه هستند.

یک راه ساخت یک سیستم گذرگاه مشترک استفاده از مولتی پلکسر است. مولتی پلکسرها ثبات مبداء را انتخاب می کنند و سپس اطلاعات آنها روی گذرگاه قرار می گیرد. ساخت یک سیستم گذرگاه برای چهار ثبات در شکل ۴-۳ نشان داده شده است. هر ثبات دارای چهار بیت است، که از ۰ تا ۳ شماره گذاری شده اند. گذرگاه از چهار مولتی پلکسر  $1 \times 4$  تشکیل شده که هر یک چهار ورودی داده ۰ تا ۳ و دو ورودی انتخاب  $S_0$  و  $S_1$  دارند. برای جلوگیری از پیچیده شدن دیاگرام بعلت تقاطع ۱۶ خط منقطع،



شکل ۴-۳ سیستم گذرگاه برای چهار ثبات

برای نشان دادن اتصالات خروجی های ثبات ها به ورودی های مولتی پلکسراها از برقسب یا حروف استفاده می شود. مثلاً خروجی 1 در ثبات A به ورودی 0 از MUX1 متصل است زیرا این ورودی دارای برقسب A<sub>1</sub> است. با توجه به دیاگرام دیده می شود که بیت های هم ارزش در هر ثبات به ورودی های یک مولتی پلکسر متصل می شوند تا یک خط گذرگاه را تشکیل دهند. بنابراین MUX0 چهار بیت 0 ثباتها را راهده می کند، MUX1 چهار بیت 1 ثباتها را راهده می نماند، و برای دو بیت دیگر نیز بهمین ترتیب.

دو خط انتخاب S<sub>1</sub> و S<sub>0</sub> به ورودی های انتخاب هر چهار مولتی پلکسر متصل اند. خطوط انتخاب چهار بیت یک ثبات را انتخاب کرده و آنها را به گذرگاه مشترک منتقل می نمایند. اگر S<sub>1</sub>S<sub>0</sub>=00 باشد، ورودی داده 0 هرچهار مولتی پلکسر، انتخاب شده و به خروجی ها که گذرگاه را تشکیل می دهند اعمال می گرددند. این موجب می شود تا خطوط گذرگاه محتویات ثبات A را دریافت دارند زیرا خروجی این ثبات به ورودی داده 0 مولتی پلکسراها متصل شده است. بطور مشابه، ثبات B بشرطی انتخاب می شود که ورودی داده 1 باشد و الی آخر. جدول ۴-۲ ثباتی را نشان می دهد که بوسیله گذرگاه بازاء هر یک از چهار مقدار دو دویی ممکن خطوط انتخاب، برگزیده شده است.

بطور کلی، یک سیستم گذرگاه، k ثبات n بیتی را برای تشکیل یک گذرگاه مشترک n خطی راهده می نماید. تعداد مولتی پلکسراها مورد نیاز برای ساخت گذرگاه برابر n است که همان تعداد تعداد بیت ها در هر ثبات است. اندازه هر مولتی پلکسر باید k×1 باشد زیرا خط داده را راهده می نماید. مثلاً یک گذرگاه مشترک برای هشت بیت 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 راهده مجموعه بیت های هم ارزش در هشت ثبات باشد.

انتقال اطلاعات از یک گذرگاه بداخل یکی از چند ثبات مقصد با اتصال خطوط گذرگاه به ورودی های تمام ثبات های مقصد و فعال کردن کنترل بارکردن ثبات مقصد خاصی که انتخاب شده تحقق می باید. گذرگاه ممکن است در عبارت سمبولیک انتقال گذرگاه ذکر شود و یا وجود آن در عبارت بطور ضمینی فرض گردد. وقتی که گذرگاه در عبارت قید شود انتقال ثبات بصورت زیر نشان داده می شود. محتوای ثبات C روی گذرگاه قرار گرفته است و محتوای گذرگاه نیز با فعال کردن ورودی کنترل بارکردن

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

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

ثبتی که انتخاب می شود		
S <sub>1</sub>	S <sub>0</sub>	
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.

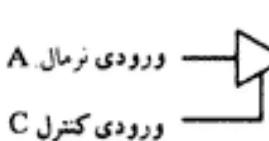
$$R1 \leftarrow C$$

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 پلکسراز گیت‌های سه حالتی ساخته شود. گیت‌های سه حالته یک مدار دیجیتال است که سه حالت را در خروجی خود ایجاد می‌کند. دو تا از این حالات سیگنال‌های معادل ۱ و ۰ 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 حالته می‌توانند هریک از انواع گیت‌های منطقی مانند AND یا NAND باشند. اما نوعی که بیش از همه در طراحی یک سیستم گذرگاه به کار می‌رود گیت بافر<sup>۱</sup> (یا میانگیر) می‌باشد.

سمبل گرافیکی یک گیت بافر سه حالته در شکل ۴-۴ نشان داده شده است. وجود ورودی کنترل، علاوه بر ورودی عادی، آن را از میانگیرهای عادی متایز می‌کند. اگر ورودی کنترل ۱ باشد، خروجی فعال می‌شود و گیت مانند هر بافر معمولی کار کرده و خروجی آن برابر ورودی نرمال<sup>\*</sup> است. وقتی ورودی کنترل ۰ 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. بعلت 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 قادر خروجی سه حالتی هستند امکان پذیر نیست). ورودی‌های کنترل بافرهای تعیین می‌کنند که کدام یک از چهار ورودی نرمال با خط گذرگاه ارتباط برقرار کند. در هر



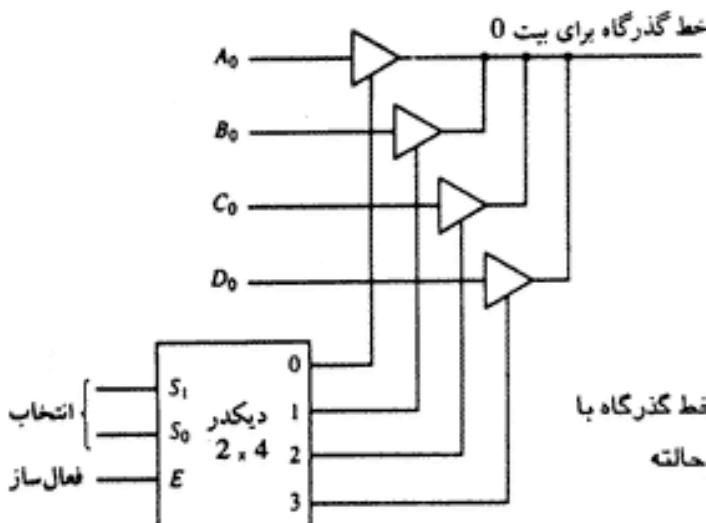
اگر  $C=1$  باشد خروجی  $Y$  برابر  $A$  می‌شود

اگر  $C=0$  باشد خروجی امپدانس بالا است

شکل ۴-۴ سمبل گرافیکی بافر سه حالته

### 1- Buffer

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



شکل ۴-۵ خط گذرگاه با  
بافرهای سه حالت

لحظه از زمان فقط یک بافر می‌تواند فعال باشد. بافرهای متصل به خط باید طوری کنترل شوند که فقط یک بافر سه حالت به گذرگاه دسترسی داشته و سایر بافرهای 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 می‌شود. وقتی که ورودی تواناساز دیکدر ۰ است، هر چهار خروجی آن ۰ بوده و خط گذرگاه در حالت امپدانس بالاست زیرا تمام چهار بافر ناتوان شده‌اند. وقتی که ورودی توانا ساز فعال باشد، یکی از بافرهای سه حالت، بسته به مقدار دودویی در ورودی‌های انتخاب دیکدر، فعال می‌شود. بررسی دقیق شکل ۴-۵ نشان می‌دهد که این روش راه دیگری برای ساختن یک مولتی پلکسر  $4 \times 1$  است زیرا این مدار می‌تواند جایگزین مولتی پلکسر شکل ۴-۳ گردد.

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

### انتقال‌های حافظه‌ای

عملکرد حافظه در بخش ۲-۷ شرح داده شد. انتقال اطلاعات از یک کلمه حافظه به محیط خارج، "خواندن"<sup>۱</sup> نامیده می‌شود. انتقال اطلاعات جدیدی که در حافظه ذخیره می‌شود، "نوشتن"<sup>۲</sup> 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 می‌گیرد. وقتی که عمل نوشتن انجام می‌شود لازم

است آدرس حافظه مشخص شود. این عمل با قرار دادن آدرس در کروشهای 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 نوشتمن 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 نوشتمن 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 ریز عمل‌های حسابی

ریز عمل<sup>۱</sup> یک عمل جزیی است که روی داده‌های ذخیره شده در ثبات‌ها انجام می‌شود. ریز عمل‌هایی که بیش از همه در کامپیوترهای دیجیتال با آن مواجه می‌شویم به چهار دسته زیر طبقه بندی می‌شوند:

- ۱- ریز عمل‌های انتقال ثبات، اطلاعات دودویی را از یک ثبات به ثبات دیگر منتقل می‌کنند.
- ۲- ریز عمل‌های حسابی عملیات محاسباتی را روی داده‌های عددی ذخیره شده در ثبات‌ها انجام می‌دهند.
- ۳- ریز عمل‌های منطقی عملیات دستکاری بیت‌ها را روی داده‌های غیر عددی ذخیره شده در ثبات‌ها انجام می‌دهند.
- ۴- ریز عمل‌های شیفت<sup>۲</sup> اعمال شیفت (جابجایی) را روی داده‌های ذخیره شده در ثبات‌ها اجرا می‌کنند.

ریز عمل انتقال ثبات، در بخش ۴-۲ معرفی شد. این نوع ریز عمل محتوای اطلاعاتی را، هنگامی که از ثبات‌های مبدأ به ثبات مقصد منتقل می‌شود، تغییر نمی‌دهد. سه نوع دیگر ریز عمل محتوای اطلاعاتی را حین انتقال تغییر می‌دهند. در این بخش ما مجموعه‌ای از ریز عمل‌های 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 دهنده نیاز داریم. سایر ریز عمل های اصلی حسابی در جدول ۴-۳ 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.

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

$\overline{R2}$  سمبول متمم ۱ ثبات  $R2$  است. با جمع ۱ با متمم ۲ تولید می شود. جمع محتوای  $R1$  با متمم ۲ محتوای ثبات  $R2$ ، برابر است با  $R1 - R2$ .

ریز اعمال افزایش و کاهش را بترتیب بوسیله عمل های جمع با ۱ و تفریق ۱ انجام می دهیم. این ریز اعمال با مدارهای ترکیبی یا با شمارنده های بالا - پایین<sup>۱</sup> پیاده سازی می شوند.

اعمال حسابی ضرب و تقسیم در جدول ۴-۳ 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 ۴-۳ ریز عمل های حسابی

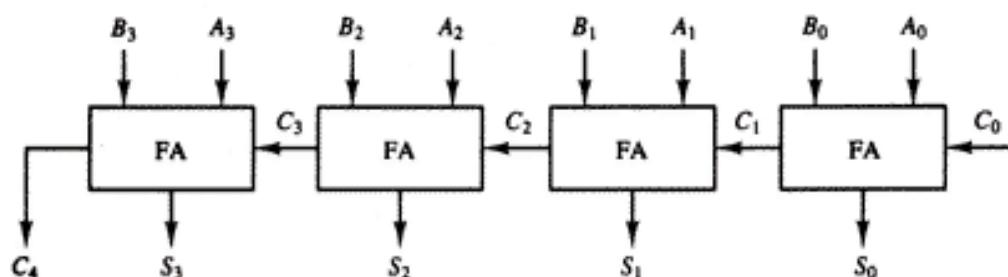
نمایش سمبولیک	شرح
$R3 \leftarrow R1 + R2$	محتوای $R1$ به $R3$ بعلاوه $R2$ می شود
$R3 \leftarrow R1 - R2$	محتوای $R1$ منهای $R2$ به $R3$ می گردد
$R2 \leftarrow \overline{R2}$	محتوای $R1$ متمم می شود (متمم ۱)
$R2 \leftarrow R2 + 1$	محتوای $R2$ متمم می شود (منفی می شود)
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ بعلاوه متمم ۲ از $R2$ (تفریق)
$R1 \leftarrow R1 + 1$	پک واحد افزایش در محتوای $R1$
$R1 \leftarrow R1 - 1$	پک واحد کاهش در محتوای $R1$

عمل می تواند 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 رشته ای از ریز عمل های جمع و شیفت اجرا می شود. تقسیم با رشته ای از ریز عمل های تفیریق و شیفت صورت می گیرد. مشخص کردن سخت افزار برای این موارد مستلزم استفاده از لیست عباراتی است که در آنها ریز عمل های جمع و تفیریق و شیفت بکار رفته اند.

### جمع کننده دودویی

برای پیاده سازی ریز عمل جمع بوسیله سخت افزار، ما به ثبات هایی که داده را نگهدارند و نیز قطعات یا مؤلفه های دیجیتالی که جمع حسابی را اجرا نمایند، نیاز داریم. مدار دیجیتالی که مجموع حسابی دو بیت و رقم نقلی قبلی را تشکیل می دهد تمام جمع کننده<sup>۱</sup> نامیده می شود (شکل ۱-۱۷). ملاحظه شود، مدار دیجیتالی که جمع دو عدد دودویی با هر طول را اجرا می نماید جمع کننده دودویی<sup>۲</sup> خوانده می شود. جمع کننده دودویی از مدارهای تمام جمع کننده که بصورت سری بهم متصلند ساخته می شود، با این ترتیب که رقم نقلی خروجی یک تام جمع کننده به ورودی رقم نقلی تمام جمع کننده بعدی متصل است. شکل ۴-۶ اتصالات درونی چهار تام جمع کننده (FA) را برای فراهم کردن یک جمع کننده دودویی چهار بیت نشان می دهد. بیت های مضاف A و مضاف الیه B بوسیله زیرنویس هایی از راست به چپ مشخص شده اند و در آن زیرنویس 0 متعلق به کم ارزشترین بیت است. ارقام نقلی، بصورت زنجیره ای بین تمام جمع کننده ها کشیده شده اند. رقم نقلی ورودی به جمع کننده دودویی C<sub>0</sub> و رقم نقلی خروجی C<sub>4</sub> است. خروجی S تمام جمع کننده ها بیت های حاصل جمع مورد نظر را تولید می نمایند.

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



شکل ۴-۶ جمع کننده دودویی 4 بیتی

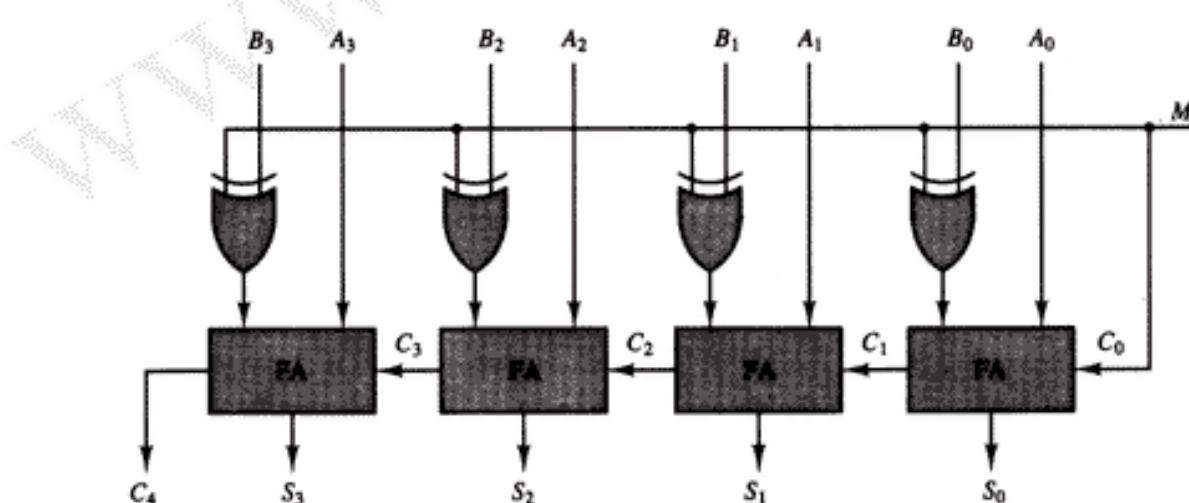
1- Full Adder

2- Binary Adder

می شوند. مجموع می تواند 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 می توان به بهترین وجه با استفاده از متمم طبق آنچه در بخش ۳-۲ بحث شده انجام داد. به خاطر بیاورید که تفریق  $A - B$  با بدست آوردن متمم  $B$  و جمع آن با  $A$  انجام می شود. متمم ۲، از متمم ۱ و جمع کم ارزشترین جفت بیت ها با ۱ حاصل می گردد. متمم ۱ را هم با گیت های معکوس کننده می توان ایجاد کرد و جمع کردن یک واحد را نیز می توان از طریق ورودی نقلی انجام داد. اعمال جمع و تفریق را می توان با افزودن یک گیت OR انحصراری 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 می کند. اگر  $M=0$  باشد، مدار یک جمع کننده و هر وقت  $M=1$  باشد مدار تبدیل به یک تفریق گر می شود. هر گیت OR انحصراری ورودی  $M$  و یکی از ورودی های  $B$  را دریافت می کند. اگر  $M=0$  باشد داریم  $B \oplus 0 = B$ . تمام جمع کننده ها مقدار  $B$  و رقم نقلی ۰ را دریافت کرده و مدار عمل  $A$  بعلاوه  $B$  را انجام می دهد. اگر  $M=1$  باشد، داریم  $B \oplus 1 = B'$  و  $C_0 = 1$ . ورودی های  $B$  کلاً متمم شده و مقدار ۱ از طریق نقلی ورودی با آن جمع می شود. مدار عمل  $A$  بعلاوه متمم ۲ عدد  $B$  را انجام می دهد. 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  $A - B$  می باشد، بشرطی که سرریزی وجود نداشته باشد.



شکل ۴-۷ جمع و تفریق کننده ۴ بیتی

1- Mode Input

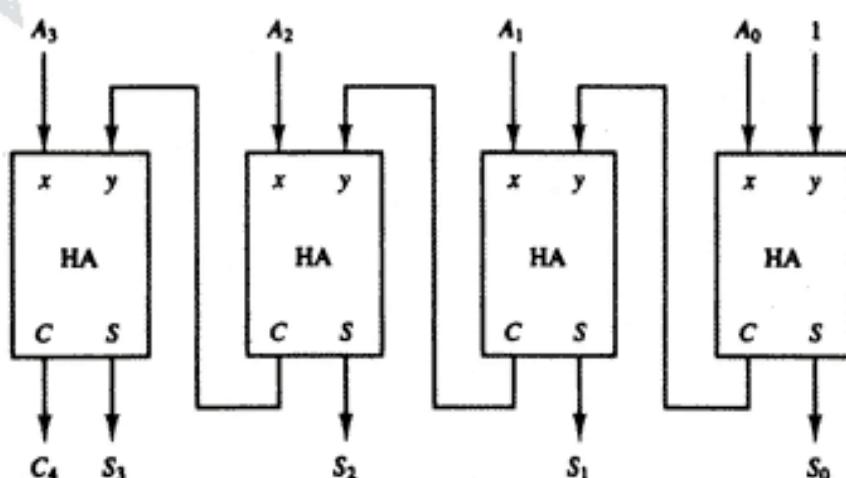
### افزایشگر دودویی

ریز عمل 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 ریز عمل بسادگی با یک شمارنده (شکل ۴-۱۰ ملاحظه شود) پیاده سازی می‌شود. هر دفعه که توانا ساز شمارش فعال شود، لبه پالس ساعت محتوای ثبات را یک واحد افزایش می‌دهد. مواردی هم پیش می‌آید که ریز عمل افزایش باید با یک مدار ترکیبی و مستقل از یک ثبات انجام شود. این کار با استفاده از نیم جمع کننده‌هایی که با هم متواالیاً بسته شده‌اند (شکل ۴-۱۶) امکان پذیر است.

دیاگرام مدار ترکیبی افزایشگر چهار بیتی در شکل ۴-۸ نشان داده شده است. یکی از ورودی‌های نیم جمع کننده<sup>۱</sup> با ارزش کمتر به 1 متصل شده است و ورودی دیگر به بیت کم ارزشتر عددی که قرار است افزایش باید وصل می‌شود. رقم نقلی خروجی از یک نیم جمع کننده به یکی از ورودی‌های نیم جمع کننده مرتبه بالاتر بعدی اتصال یافته است. مدار، چهار بیت A<sub>3</sub> تا A<sub>0</sub> را دریافت کرده، یک واحد به آن اضافه می‌کند، و خروجی افزایش یافته را در S<sub>0</sub> تا S<sub>3</sub> تولید می‌نماید. رقم نقلی خروجی C<sub>4</sub> فقط پس از حالت دودویی 1111 برابر 1 خواهد شد. در این حالت S<sub>0</sub> تا S<sub>3</sub> هم 0 می‌شوند. مدار شکل ۴-۸ می‌تواند به یک افزایشگر ۱۱ بیتی گسترش باید و این با گسترش دیاگرام توسط نیم جمع کننده امکان پذیر است. یکی از بیت‌های کم ارزشتر باید به 1 منطقی متصل گردد. سایر ورودی‌ها، عددی را که باید افزایش باید و یا رقم نقلی مرحله قبلی را دریافت می‌کنند.

### مدار حسابی

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



شکل ۴-۸ 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 ۴-۹ نشان داده شده است. در این دیاگرام چهار مدار تمام کننده وجود دارد که جمع چهار بیت را بوجود می آورد و چهار مولتی پلکسرا 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 پلکسرا متصلند. ورودی های داده مولتی پلکسرا 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 پلکسرا بوسیله دو خط انتخاب  $S_1$  و  $S_0$  کنترل می شوند. رقم نقلی  $C_{in}$  به ورودی 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

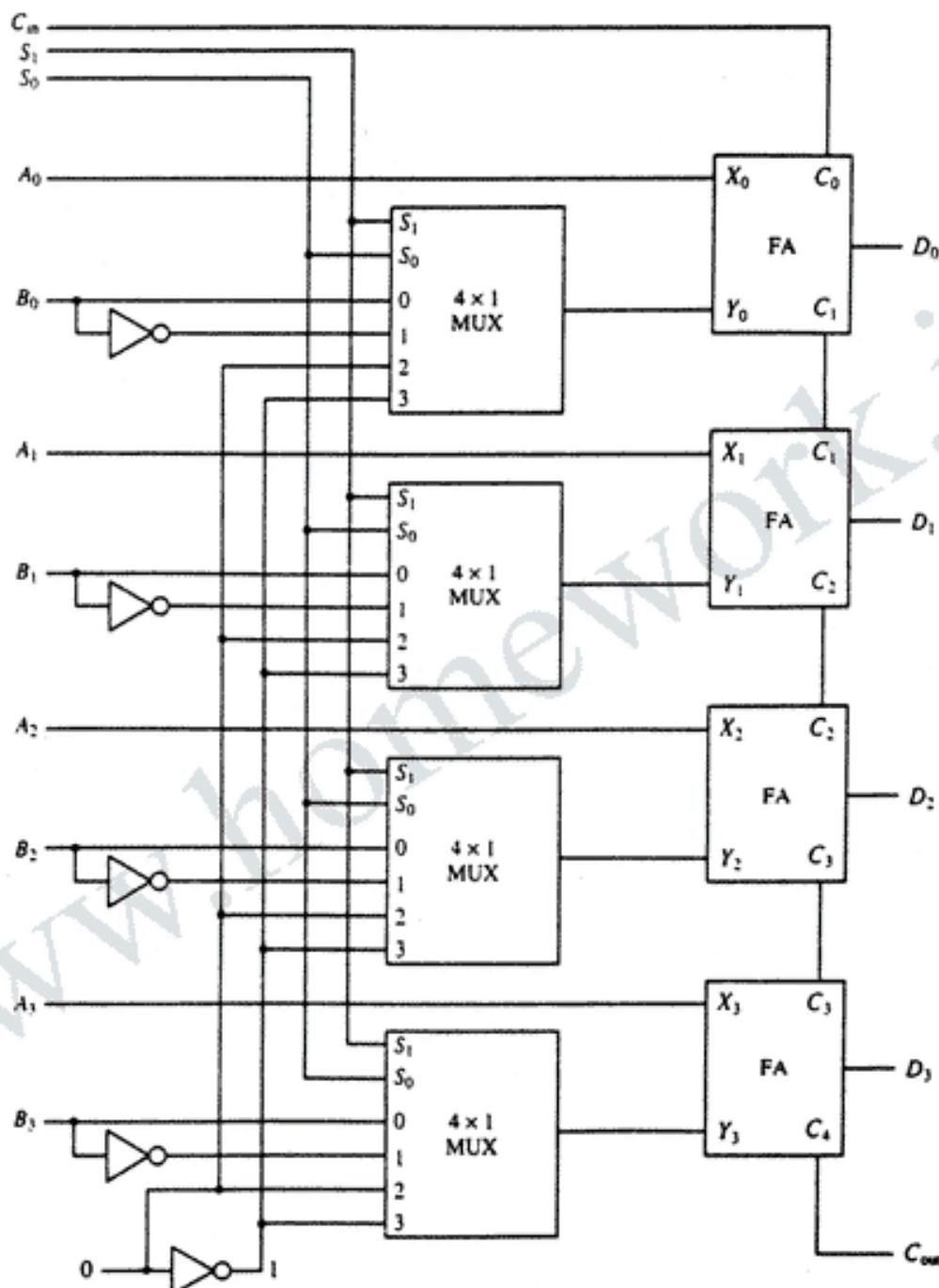
$$D = A + Y + C_{in}$$

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 0 یا 1 کردن  $C_{in}$  می توان هشت ریز عمل حسابی لیست شده در جدول ۴-۴ را تولید کرد. اگر  $00 = S_1S_0$  باشد، مقدار B به ورودی های Y جمع کننده اعمال می شود. اگر 0 باشد،  $00 = S_1S_0$  باشد خواهد بود. هر دو خروجی D = A + B + 1 باشد خروجی 1 باشد خروجی D = A + B است. اگر 1 باشد خروجی 1 باشد خواهد بود. هر دو حالت ریز عمل جمع را با و یا بدون نقلی ورودی انجام می دهند.

اگر  $01 = S_1S_0$  باشد، متمم B به ورودی های Y جمع کننده اعمال می گردد. اگر 1 باشد  $C_{in} = 1$  سپس  $D = A + \bar{B} + 1$  خواهد بود. این عمل A بعلاوه متمم 2 عدد B را که معادل تفیق A-B است تولید می نماید. وقتی که 0 باشد سپس  $D = A + \bar{B}$  خواهد شد. این عمل معادل با تفیق بدون قرض کردن می باشد، یعنی،  $A - B = 1$ .

در  $10 = S_1S_0$  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 0 باشد داریم  $C_{in} = 1$  و اگر 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  $11 = S_1S_0$  باشد، بداخل ورودی های Y جمع کننده تماماً 1 وارد می شود تا عمل کاهش 1 واحد، وقتی که 0 باشد،  $D = A - 1$ ، 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 ۴-۹ مدار حساب ۴ بیتی

منم 2 عدد 1 برابرست با  $A - 1$ . اگر  $C_{in} = 1$  خواهد شد، که انتقال مستقیمی را از ورودی A به خروجی D ایجاد می‌نماید. 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	خروجی	ریز عمل
$S_1$	$S_0$	$C_m$	$Y$	$D = A + Y + C_m$	
0	0	0	$B$	$D = A + B$	جمع
0	0	1	$B$	$D = A + B + 1$	جمع با نقلی
0	1	0	$\bar{B}$	$D = A + \bar{B}$	نفی با فرض
0	1	1	$\bar{B}$	$D = A + \bar{B} + 1$	نفی
1	0	0	0	$D = A$	انتقال
1	0	1	0	$D = A + 1$	افزایش
1	1	0	1	$D = A - 1$	کاهش
1	1	1	1	$D = A$	انتقال

## 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<sup>۱</sup> 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 بدهد متغایر R1 برابر 1010 و متغایر R2 برابر 1100 باشد. ریز عمل OR انحصاری مشخص شده بالا محاسبات منطقی زیر را انجام خواهد داد

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

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

برای تفکیک ریز عمل های OR و AND سمبول های خاصی بکار خواهند رفت. تا از سمبول های بکار رفته از توابع بول متمایز شوند. سمبول ۷ برای ریز عمل OR و سمبول ۸ برای ریز عمل AND بکار خواهد رفت. ریز عمل متمم سازی همان متمم ۱ بوده و با یک خط که روی نام ثبات قرار می گیرد

1- Exclusive OR (XOR, EXOR)

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

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

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

### لیست ریز عمل 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 آورده تعریف کرد. در این جدول هر یک از شانزده ستون F<sub>0</sub> تا F<sub>15</sub> یک جدول درستی را برای هر تابع بول دو متغیر x و y نمایش می دهد. توجه کنید که تابع از شانزده ترکیب ممکن که برای F می تواند تخصیص یابد بدست آمده است. ۱۶ تابع بول دو متغیر x و y بصورت جبری در جدول ۴-۶ آورده شده اند. ۱۶ ریز عمل 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 دهد، اهمیت دارد. ریز عمل 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 ریز عمل روی رشته ای از بیت های ذخیره شده در ثبات انجام می گیرد.

### پیاده سازی سخت افزاری

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

جدول ۴-۵ جدول های ارزش برای ۱۶ تابع دو متغیره

x	y	F <sub>0</sub>	F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	F <sub>4</sub>	F <sub>5</sub>	F <sub>6</sub>	F <sub>7</sub>	F <sub>8</sub>	F <sub>9</sub>	F <sub>10</sub>	F <sub>11</sub>	F <sub>12</sub>	F <sub>13</sub>	F <sub>14</sub>	F <sub>15</sub>
0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	1	0	0	0	0	1	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 ریز عمل منطقی

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

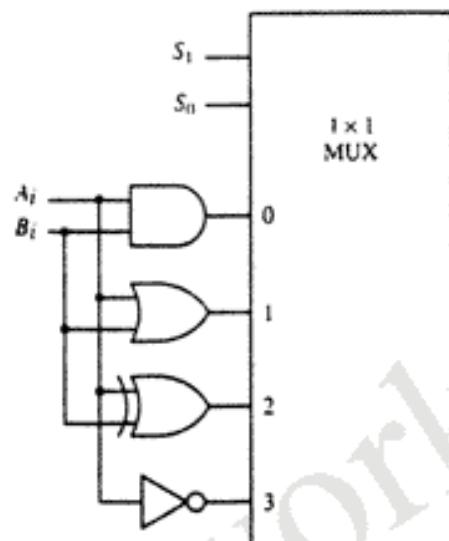
بیت های ثبات های منطقی گذاشته شود تا تابع منطقی مورد نظر را اجرا نماید. اگرچه ۱۶ ریز عمل منطقی وجود دارد اغلب کامپیوتراها فقط از چهار ریز عمل (OR, XOR, AND, OR) انحصاری) و متمن استفاده می کنند، و سایر ریز عمل ها را نیز می توان از این چهار عمل بدست آورد.

شکل ۴-۱۰ یک طبقه از مدار را که چهار ریز عمل اصلی منطقی را اجرا می نماید، نشان می دهد. این طبقه مشکل از چهار گیت و یک مولتی پلکسراست. 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 پلکسرا اعمال می شود. دو ورودی انتخاب  $S_1$  و  $S_0$  یکی از ورودی های داده به مولتی پلکسرا انتخاب کرده و مقدار آن را به خروجی هدایت می نماید. دیگر، نمونه ای از طبقات را که با اندیس  $i$  مشخص شده نشان می دهد. برای یک مدار منطقی  $n$  بیتی، دیگر باید  $n$  بار برای  $i=0, 1, 2, \dots, n-1$  تکرار شود. متغیر های انتخاب به تمام طبقات اعمال می شوند. جدول تابع شکل ۴-۱۰ (ب) ریز عمل های منطقی حاصل برای هر ترکیبی از متغیر های انتخاب را لیست نموده است.

## چند کاربرد

ریز عمل های منطقی در دستکاری بیت های متمایز یا بخشی از کلمات ذخیره شده در یک ثبات بسیار مفیدند. از این ریز عمل 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 (فرض A) را با ریز عملی که تابعی از بیت های ثبات دیگر است (فرض 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	$E = \bar{A}$	نمم



(ب) جدول تابع

(الف) نمودار منطقی

شکل ۴-۱۰ یک طبقه از مدار منطق

کاربرد، ثبات A می‌تواند ثبات یک پردازشگر و بیت‌های ثبات B عملوندی منطقی حاصل از حافظه باشد که در ثبات B قرار گرفته است.

عمل نشاندن انتخابی<sup>۱</sup> هایی را در بیت‌های ثبات A که متناظرشان در ثبات B برابر 1 هستند، می‌نشاند. این عمل مکان‌هایی را که متناظرشان در B برابر 0 باشد تغییر نمی‌دهد. مثال عددی زیر این عمل را واضح‌تر بیان می‌دارد.

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

دو بیت سمت چپ B ، 1 هستند، بنابراین بیت‌های متناظرشان در A ، 1 خواهند شد. یکی از این دو بیت قبل<sup>۱</sup> 0 بوده و دیگری از 0 به 1 تبدیل شده است. دو بیت A که نظیرشان در B برابر 0 می‌باشند بلا تغییر باقی مانده‌اند. مثال فوق مانند یک جدول درستی است زیرا تمام چهار بیت ترکیب دو متغیر دودویی را داراست. از جدول درستی در می‌یابیم که بیت A پس از عمل OR منطقی بیت‌ها در ثبات B و مقادیر قبلی A بدست می‌آیند. بنابراین، ریز عمل OR می‌توانند برای 1 کردن بیت‌های انتخاب شده یک ثبات بکار روند.

## 1- Selective set

عمل متمم سازی انتخابی<sup>۱</sup> بیت های A را که متناظر شان در B، برابر ۱ باشد متمم می نماید. این عمل مکان هایی را که نظری شان در B برابر ۰ باشد تغییر نمی دهد. مثلاً

$$\begin{array}{r} 1010 \\ 1100 \\ \hline 0110 \end{array} \quad \begin{array}{l} \text{قبل از عمل} \\ \text{عملوند منطقی} \\ \text{بعد از عمل} \end{array} \quad \begin{array}{l} A \\ \\ A \end{array}$$

مجددآ دو بیت متنهاالیه سمت چپ B، ۱ 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 انتخابی<sup>۲</sup> فقط بیت هایی از A را، که بیت های متناظر آنها در B مقدار ۱ دارد، ۰ می کند مثلاً

$$\begin{array}{r} 1010 \\ 1100 \\ \hline 0010 \end{array} \quad \begin{array}{l} \text{قبل از عمل} \\ \text{عملوند منطقی} \\ \text{بعد از عمل} \end{array} \quad \begin{array}{l} A \\ \\ A \end{array}$$

باز هم دو بیت متنهاالیه چپ B مقدار ۱ دارند، بنابراین بیت های متناظر آنها در A برابر ۰ شده اند. می توان نتیجه گرفت که عمل بولی انجام شده روی بیت ها  $\bar{A}$  است. ریز عمل منطقی مربوطه عبارتست از

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

عمل ماسک<sup>۳</sup> (پوشش) مشابه عمل پاک کردن انتخابی است بجز اینکه بیت های A فقط در صورتی که نظری شان در B مقدار ۰ باشد، پاک می شوند. عمل ماسک، همانطور که از مثال عددی زیر دیده می شود، یک ریز عمل AND است.

$$\begin{array}{r} 1010 \\ 1100 \\ \hline 1000 \end{array} \quad \begin{array}{l} \text{قبل از عمل} \\ \text{عملوند منطقی} \\ \text{پس از ماسک} \end{array} \quad \begin{array}{l} A \\ B \\ A \end{array}$$

دو بیت متنهاالیه سمت راست A پاک شده اند زیرا بیت های متناظر شان در B، ۰ هستند. دو بیت سمت چپ بلا تغییر مانده اند زیرا بیت های نظری شان در B، ۱ می باشند. عمل ماسک از عمل پاک کردن انتخابی مناسبتر است زیرا اکثر کامپیوترها دارای دستور AND می باشند، و آنها بیان که دستوراتی را برای اجرای ریز عمل پاک کردن انتخابی دارا هستند اندکند.

عمل درج<sup>۱</sup>، 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:

0110 1010	قبل از عمل A
0000 1111	(ماسک) B
<hr/>	
0000 1010	پس از عمل A

و سپس مقدار جدید را درج می‌نماییم

0000 1010	قبل از عمل A
1001 0000	(درج) B
<hr/>	
1001 1010	بعد از عمل A

عمل ماسک یک ریز عمل AND و عمل درج یک ریز عمل OR می‌باشد. عمل پاک کردن<sup>۲</sup>، 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 می‌باید و بوسیله مثال زیر نشان داده شده است.

1010	A
1010	B
<hr/>	
0000	A $\leftarrow$ A $\oplus$ B

اگر A و B مساوی باشند، بیت‌های نظیر به نظری هر دو یا 0 و 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 پردازی 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 چپ ترین 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<sup>۱</sup> مقدار ۰ را از طریق ورودی سری انتقال می‌دهد. ما سمبول `shl` و `shr` را بترتیب برای ریز عمل‌های شیفت به چپ و شیفت به راست برمی‌گزینیم. مثلاً

$$\begin{aligned} R1 &\leftarrow \text{shl } R1 \\ R2 &\leftarrow \text{shr } R2 \end{aligned}$$

دو نمونه ریز عمل می‌باشند که محتوای ثبات  $R1$  را یک بیت به راست شیفت می‌دهند. سمبول 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، ۰ باشد.

شیفت چرخشی (که عمل چرخش<sup>۲</sup> نیز خوانده می‌شود) بیت‌های ثبات را از طریق دواتها بدون از دست دادن هرگونه اطلاعات می‌چرخاند. این عمل با اتصال خروجی سری به ورودی سری ثبات تحقق می‌یابد. ما سمبول‌های `cil` و `cir` را بترتیب برای چرخش به چپ و چرخش به راست بکار خواهیم برد. سمبول‌های بکار رفته و ریز عمل‌های شیفت در جدول ۴-۷ نشان داده شده‌اند.

شیفت حسابی<sup>۳</sup> ریز عملی است که که یک عدد دودویی علامت دار را به چپ یا راست شیفت می‌دهد. شیفت حسابی به چپ، یک عدد دودویی علامت دار را در ۲ ضرب می‌کند. یک شیفت حسابی به راست نیز عدد را بر ۲ تقسیم می‌نماید. شیفت‌های حسابی باید بیت علامت را تغییر دهند زیرا وقتی عدد را در ۲ ضرب یا تقسیم می‌کنیم علامت همچنان باقی می‌ماند. سمت چپ‌ترین بیت در ثبات بیت علامت را نگه می‌دارد، و بقیه بیت‌ها عدد را حفظ می‌کنند. بیت علامت برای اعداد مثبت، ۰ و برای منفی، ۱ است. اعداد منفی در فرم متمم ۲ هستند. شکل ۴-۱۱ نمونه‌ای از یک ثبات  $n$  بیت را نشان می‌دهد. بیت  $-R_{n-1}$  در سمت چپ‌ترین مکان بیت علامت را نگه می‌دارد. بیت  $-R_{n-2}$ ، با ارزش‌ترین بیت  $R_0$  کم ارزش‌ترین بیت است. شیفت به راست حسابی بیت علامت را عوض نمی‌کند و همه بیت‌ها (از جمله علامت) را به راست شیفت می‌دهد. بنابراین  $R_{n-1} R_{n-2} \dots R_0$  بیت  $-R_{n-1}$  را دریافت می‌نماید و برای سایر بیت‌های ثبات نیز بهمین ترتیب. بیت واقع در  $R_0$  از دست می‌رود.

جدول ۴-۷ ریز عمل‌های شیفت

سمبول نشانده‌شده	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 چرخش ثبات $R$ به چپ
$R \leftarrow \text{cir } R$	شیفت چرخش 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 ۴-۱۱ شیفت حسابی به راست

شیفت حسابی به چپ یک  $R_0$  می‌نماید، و کلیه بیت‌های دیگر را به چپ شیفت می‌دهد. بیت اولیه  $R_{n-1}$  از دست رفته و با بیت  $-2R_{n-2}$  جایگزین می‌شود. اگر بیت واقع در  $R_{n-1}$  پس از شیفت عوض شود، علامت معکوس شده است. این هنگامی رخ می‌دهد که ضرب در 2 سبب سرریز گردد. سرریز در شیفت به چپ هنگامی رخ می‌دهد که قبل از شیفت  $R_{n-1}$  و  $-2R_{n-2}$  مساوی نباشد. یک فلیپ فلاب سرریز  $\oplus V_s$  برای کشف یک سرریز حاصل از شیفت حسابی به چپ می‌تواند مورد استفاده قرار گیرد.

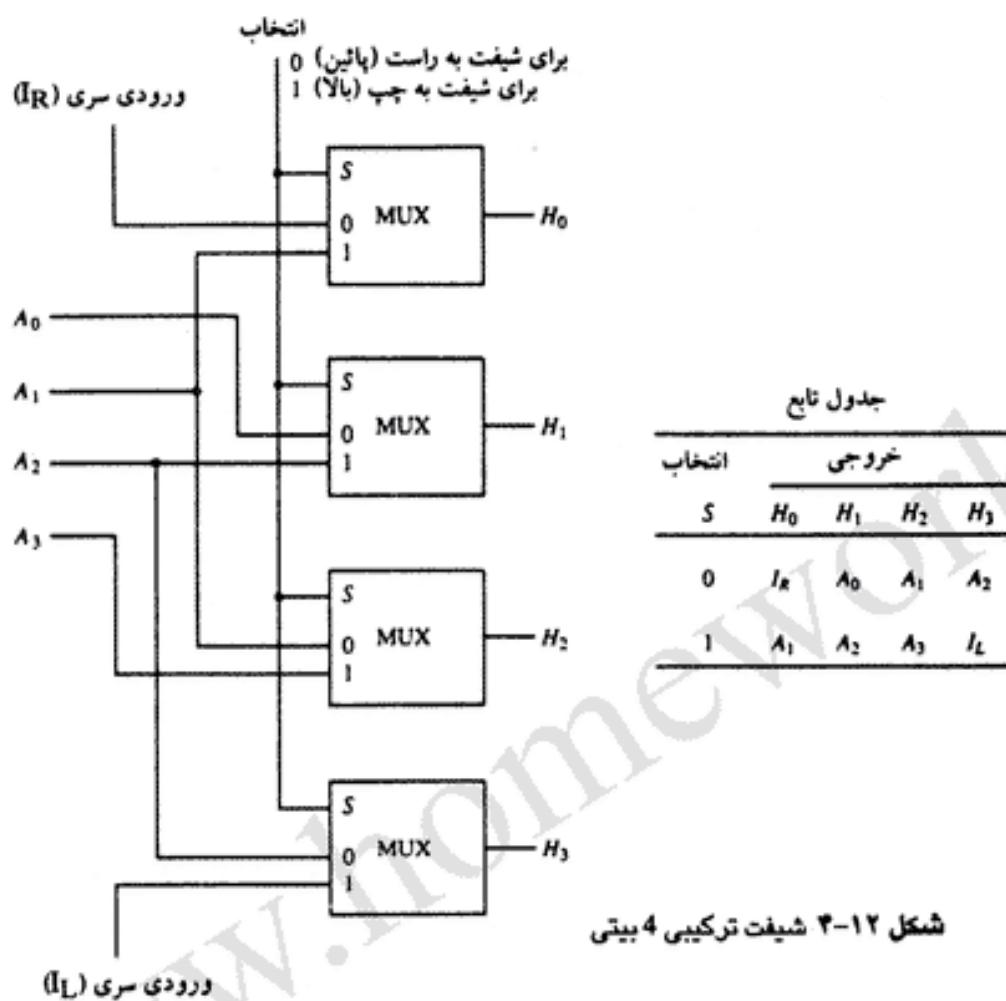
$$V_s = R_{n-1} \oplus R_{n-2}$$

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 فلاب سرریز گردد.

### پیاده سازی سخت افزاری

یکی از انتخاب‌های ممکن برای واحد شیفت، ثبات شیفت دو طرفه با بارشدن موازی است (شکل ۴-۹ ملاحظه شود). اطلاعات را می‌توان 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 موردنیاز است و پالس دیگری برای شیفت دادن لازم است. در یک واحد پردازنده با ثبات‌های 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 می‌شود، و سپس عدد شیفت داده شده به ثبات بازگردانده می‌شود. این نیز تنها یک پالس ساعت برای بارکردن مقدار شیفت یافته بداخل ثبات نیاز دارد.

یک شیفت دهنده ترکیبی را می‌توان مانند شکل ۴-۱۲ 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 ( $I_L$ ) و دیگری برای شیفت به راست ( $I_R$ ). وقتی که ورودی انتخاب  $S=0$  باشد، داده ورودی به راست شیفت داده می‌شود (در دیاگرام به سمت پائین). اگر  $S=1$  باشد، داده ورودی به چپ شیفت می‌پابد (به سمت بالای دیاگرام). با  $n$  ورودی و خروجی داده به  $n$  مولتی پلکسر نیاز دارد. دو ورودی سری بوسیله مولتی پلکسر دیگری قابل کنترل‌اند تا سه نوع شیفت ممکن را فراهم آورند.



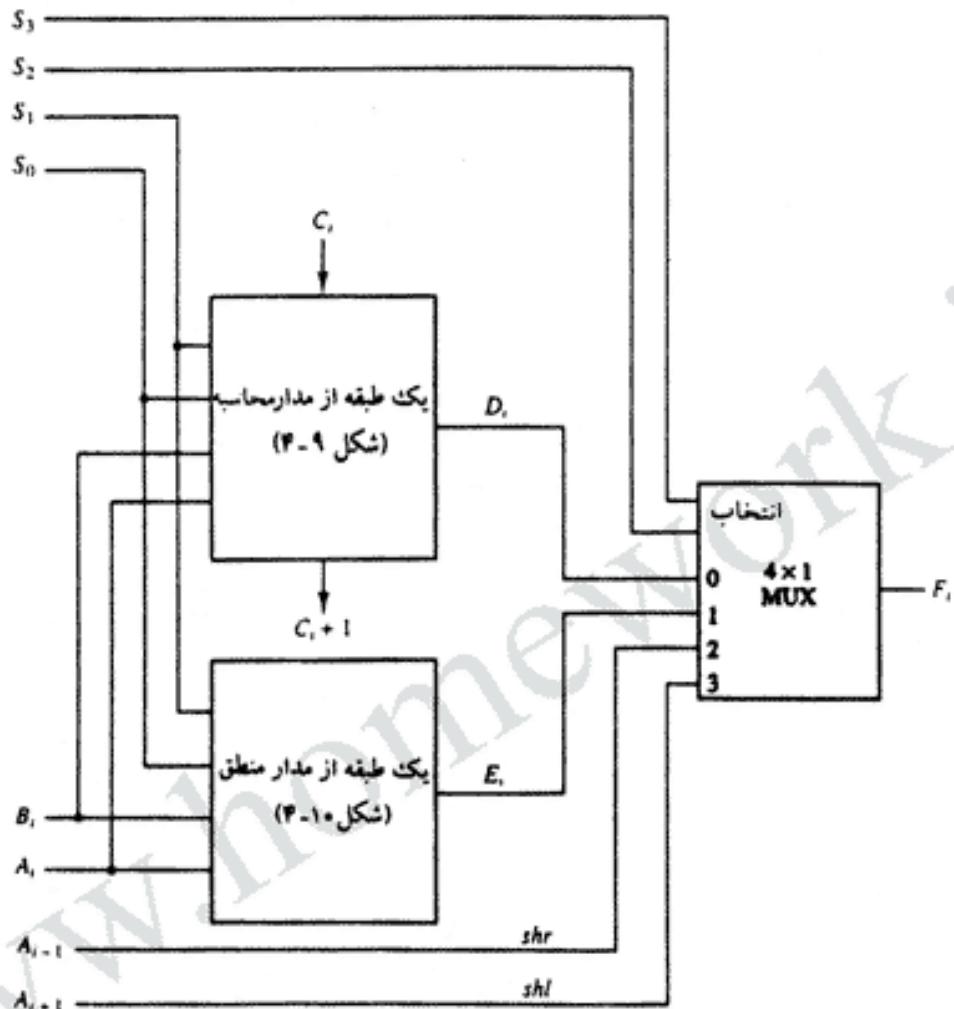
شکل ۴-۱۲ شیفت ترکیبی ۴ بیتی

#### ۴-۷ واحد حساب، منطق و شیفت

در سیستم‌های کامپیوترازی به جای اینکه ثبات‌های مختلف ریز عملها را مستقیماً اجزا کنند، از تعدادی ثبات ذخیره سازی استفاده می‌شود که به یک واحد عملیاتی مشترک بنام واحد حساب و منطق<sup>۱</sup> متصل‌اند. برای اجرای یک ریز عمل، 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 ۴-۱۳

1- Arithmetic and Logic Unit



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

دیده می‌شود. زیرنویس آنمايانگر یک طبقه نمونه است. ورودی‌های  $A_i$  و  $B_i$  به هر دو مدار حساب و منطق اعمال شده‌اند. یک ریز عمل خاصی با ورودی‌های  $S_1$  و  $S_0$  انتخاب می‌شوند. یک مولتی‌پلکسر  $4 \times 1$  در خروجی، خروجی حسابی  $E_i$  یا منطقی  $H_i$  را برمی‌گزیند. داده‌ها در این مولتی‌پلکسر با  $S_2$  و  $S_3$  انتخاب می‌شوند. دو ورودی دیگر داده به مولتی‌پلکسر،  $A_{i-1}$  را برای عمل شیفت به راست و  $A_{i+1}$  را برای عمل شیفت به چپ دریافت می‌کند. توجه کنید که دیاگرام تنها یک طبقه را نشان می‌دهد. برای یک ALU با ساختمان  $n$  بیتی مدار شکل ۴-۱۳  $n$  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 طبقه‌اش 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 ۴-۱۰ 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 ۱۱، ۱۰،  $S_3S_2=10$  انتخاب می شوند. سه ورودی انتخاب دیگر تاثیری بر روی شیفت ندارند.

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

انتخاب کننده عمل						تابع
$S_3$	$S_2$	$S_1$	$S_0$	$C_{in}$	عمل	
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	$F = A + \bar{B}$	تفريق با قرض
0	0	1	0	1	$F = A + \bar{B} + 1$	تفريق
0	0	1	1	0	$F = A - 1$	A کاهش
0	0	1	1	1	$F = A$	انتقال
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	$F = \bar{A}$	منس 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

۴-۲ 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 چهارثبات R0، R1، R2 و R3 از طریق یک مولتی پلکسر  $4 \times 1$  به ورودی های ثبات پنجم R5 وصل شده اند. هر ثبات هشت بیتی است. انتقالات لازم توسط چهار متغیر زمانبندی T0 تا T3 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. بلکه دیاگرام رسم کنید که نشان دهنده ساخت افزار پیاده سازی انتقال ثبات باشد. اتصالات لازم برای چهار متغیر زمانبندی به خطوط انتخاب مولتی پلکسر و ورودی بارکردن ثبات R5 را نیز نشان دهد.

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

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

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

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

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 دهد.

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

۴-۸ بلکه دیاگرام ساخت 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 دهد.

(باید داشته باشید که سمعبل + در بخش کنترل عمل OR و در ریز عمل جمع معنی بعلاوه است).

۴-۹ ساخت افزاری را نشان دهد که عبارت زیر را پیاده سازی کند. برای تابع کنترل گیت ها و برای شمارنده دودویی با ورودی فعال ساز (توان انسان) شمارش، بلکه 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 ۴ 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  $R_1 R_2$  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 ۴ بیتی، یک جمع کننده ۴ بیتی، و یک مولتی پلکسر چهارتایی ۲:۱ که ورودی ها  $x$  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 ۴ بیتی با بارشدن موازی مانند شکل ۲-۱۱ و یک جمع کننده ۴ بیتی مانند شکل ۴-۶، یک بلاک دیاگرام رسم کرده و چگونگی پیاده سازی عبارت های زیر را نشان دهید.

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

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

که  $R_1$  یک شمارنده با بارشدن موازی و  $R_2$  یک ثبات ۴ بیت است.

۴-۱۲ مدار جمع - تفاضل کننده شکل ۴-۷ دارای مقادیر زیر برای ورودی مدار  $M$ <sup>۱</sup> و ورودی های داده  $A$  و  $B$  می باشد. در هر حالت، مقادیر خروجی  $S_3 S_2 S_1 S_0$  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 کاوش گر چهار بیت با چهار مدار تمام جمع کننده بسازید.

۴-۱۴ فرض کنید که مدار چهار بیت شکل ۴-۹ در یک مدار مجتمع قرار گرفته باشد. اتصالات لازم برای دو IC از این نوع را برای ساختن یک مدار حسابی ۸ بیتی نشان دهید.

۴-۱۵ یک مدار حسابی با یک متغیر انتخاب  $s$  و دو خط ورودی  $A$  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)	$D = A + \bar{B} + 1$ (تفريق)

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

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

۴-۱۸ ثبات A عدد هشت بیتی ۱۱۰۱۱۰۰۱ را حفظ می‌کند. عملوند B و ریز عمل منطقی لازم را برای تغییر A به هر یک از حالات زیر معین کنید

(الف) ۱۱۱۱۱۱۰۱      (ب) ۰۱۱۰۱۱۰۱

۴-۱۹ ثبات های ۸ بیتی AR، BR، CR و DR ابتداً دارای مقادیر زیرند

$$AR = 11110010$$

$$BR = 11111111$$

$$CR = 10111001$$

$$DR = 11101010$$

محتوای هشت بیتی هر یک از ثبات‌ها را پس از اجرای هر یک از رشته ریز عمل‌ها مشخص کنید.

$AR \leftarrow AR + BR$  را با هم جمع کن  
BR

$CR \leftarrow CR \wedge DR$  ،  $BR \leftarrow BR + 1$  را افزایش بده DR کن، CR را از AR کم کن

$AR \leftarrow AR - CR$  را از AR کم کن

۴-۲۰ یک ثبات هشت بیتی حاوی عدد دودویی ۱۰۰۱۱۱۰۰ است. پس از یک شیفت به راست حسابی مقدار ثبات چقدر است؟ با شروع از مقدار اولیه ۱۰۰۱۱۱۰۰، 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=۱۱۰۱۱۱۰۱ رشته مقادیر دودویی را در R پس از یک شیفت به چپ منطقی و بدنیال آن یک شیفت به راست چرخشی، و سپس با یک شیفت به راست منطقی و نهاياناً یک شیفت به چپ چرخشی تعیین کنید.

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

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

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

الف)  $AR \leftarrow \overline{AR}$  ،  $AR \leftarrow 0$

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

# سازمان و طراحی یک کامپیوتر پایه



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

۵-۲ ثبات‌های کامپیوتر

۵-۳ دستورالعمل‌های کامپیوتر

۵-۴ زمانبندی و کنترل

۵-۵ سیکل دستورالعمل

## ۵-۱ کدهای دستورالعمل‌ها

در این فصل یک کامپیوتر پایه<sup>۱</sup> را معرفی می‌کنیم و نشان خواهیم داد که چگونه می‌توان عملکرد آن را توسط عبارات انتقال ثبات مشخص نمود. سازمان کامپیوتر بوسیله ثبات‌های 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 برخورde با اشکالات متعدد، قادر می‌سازد.

سازمان داخلی یک سیستم دیجیتال با رشتہ ریز عملهای<sup>۲</sup> که روی داده‌های ذخیره شده در ثبات‌هایش انجام می‌دهد تعریف می‌شود. کامپیوتر دیجیتال همه منظوره<sup>۳</sup> قادر به اجرای انواع ریز عملهای 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 دستورالعمل‌های است 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- Generol Purpose Digital Computer

دستورالعمل کامپیووتر یک کد دودویی است که رشته‌ای از ریزاعمال را برای کامپیووتر مشخص می‌کند. کدهای دستورات همراه با داده‌هایشان در حافظه ذخیره می‌شوند. کامپیووتر 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 بدنیال آن با صادر کردن رشته‌ای از ریزعمل‌ها آن را اجرا می‌نماید. هر کامپیووتر مجموعه دستورالعمل‌های خاص خود را دارد. توانایی در ذخیره و اجرای دستورات، یا مفهوم توانایی برنامه ذخیره شده، مهمترین خاصیت یک کامپیووتر همه منظوره است.

یک کد دستورالعمل مجموعه‌ای از بیت‌هاست که انجام یک عمل خاص را به کامپیووتر 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 دستورگروهی 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 کامپیووتر بستگی دارد. برای  $2^n$  عمل مستقل از هم (یا کمتر)، کد عمل حداقل  $n$  بیتی است. بعنوان توضیحی بیشتر در این مورد، کامپیووتری 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، کامپیووتر 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 کامپیووتر بدانیم. عمل یخسی از دستور ذخیره شده در حافظه است. این بخش که یک کد دودویی است انجام عملی خاص را به کامپیووتر 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 کامپیووتر 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 عمل<sup>1</sup> (کلان) هم می‌گویند زیرا مجموعه‌ای از ریزعمل‌ها را مشخص می‌نماید.

بخش عمل در یک کد دستورالعمل، کاری را که قراراست انجام بگیرد معین می‌کند. این عمل باید ببروی 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 کامپیووتر ساختار کد دستورالعمل خاص خود را دارد. قالب کدهای دستور توسط طراحان که معماری آن را مشخص می‌نمایند تعیین می‌شود. در این فصل ما قالب کدهای خاصی را برای نشريج سازمان و طراحی کامپیووترهای ديجيتال برمی‌گزئيم.

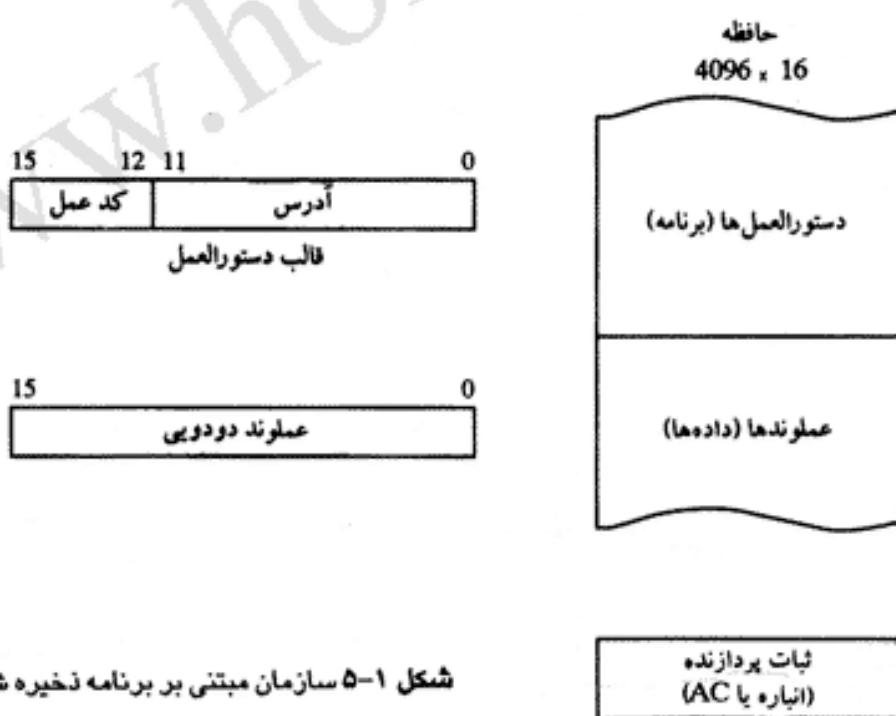
## 1- Macrooperation

### سازمان مبتنی بر برنامه ذخیره شده

ساده‌ترین راه سازماندهی یک کامپیوتر داشتن یک ثبات پردازنده و قالب کد دستورالعملی متشكل از دو بخش است. قسمت اول عملی را که انجام خواهد شد مشخص می‌کند و قسمت دوم آدرس را معین می‌نماید. آدرس به کنترل، محل عملوند<sup>۱</sup> را در حافظه نشان می‌دهد. این عملوند از حافظه خوانده شده و بر روی آن بعنوان داده ذخیره شده در ثبات پردازشگر عمل می‌شود.

شکل ۱-۵ این نوع سازماندهی را نشان می‌دهد. دستورالعمل‌ها در یک بخش دیگری ذخیره می‌شوند. برای واحد حافظه‌ای با  $4096 \times 16$  کلمه به ۱۲ بیت برای مشخص کردن آدرس نیاز است، زیرا  $2^{12} = 4096$  می‌باشد. اگر هر کد دستورالعمل در یک کلمه ۱۶ بیتی حافظه جای گیرد، چهار بیت برای کد عمل لازم است تا یکی از ۱۶ عمل ممکن را مشخص نماید و نیز ۱۲ بیت برای مشخص کردن آدرس عملوند خواهیم داشت. کنترل یک کد ۱۶ بیتی دستورالعمل را از آن قسمت از حافظه که برنامه در آن است می‌خواند. سپس از بخش ۱۲ بیتی آدرس در دستورالعمل برای خواندن یک عملوند ۱۶ بیتی از ناحیه داده‌ها استفاده می‌شود. آنگاه عملی را که کد عمل مشخص می‌نماید اجرا می‌کند. در کامپیوترهایی که فقط یک ثبات پردازش دارند نام اثباته<sup>۲</sup> را به آن اختصاص می‌دهند. اعمال در این کامپیوتر معمولاً بر روی عملوند و محتویات اثباته AC صورت می‌گیرد.

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



شکل ۱-۵ سازمان مبتنی بر برنامه ذخیره شده

دیگری بکار گرفته می شوند. مثلاً، اعمالی مانند پاک کردن<sup>۱</sup> AC، متتم 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 ۰ تا ۱۱) برای تعیین آدرس حافظه مورد نیاز نیستند و می توانند برای اعمال دیگری در کامپیوتر بکار روند.

### آدرس غیر مستقیم

گاهی اوقات مناسبتر است از بیت های آدرس دستورالعمل نه بعنوان آدرس بلکه عملوند واقعی استفاده شود. در اینصورت گوئیم دستورالعمل دارای عملوند بلافصل<sup>۲</sup> است. هر گاه این بخش، آدرس عملوند را مشخص کند گوئیم دستور دارای آدرس مستقیم<sup>۳</sup> است. دلیل این نام وجودامکان سومی است که آدرس غیر مستقیم<sup>۴</sup> نامیده می شود در آن بیت های 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 غیر مستقیم می توان بکار برد.

به منظور توضیح بیشتر، قالب کد دستورالعمل شکل ۵-۲ (الف) را در نظر بگیرید. این دستور متشکل است از سه بیت کد عملیات، ۱۲ بیت آدرس و یک بیت مربوط به روش آدرس دهی مستقیم یا غیر مستقیم، که با I نشان داده شده است. بیت I برای آدرس دهی مستقیم برابر ۰ و برای آدرس دهی غیر مستقیم ۱ می باشد. در شکل ۵-۲ (ب) دستورالعملی با آدرس مستقیم نشان داده شده است. این دستورالعمل در آدرس ۲۲ از حافظه قرار دارد. بیت I برابر ۰ است بنابراین دستور از نوع آدرس مستقیم است. کد عمل مشخص کننده دستور ADD است و بخش آدرس یک عدد دو دویی معادل با ۴۵۷ می باشد. واحد کنترل عملوند را در آدرس ۴۵۷ یافته و آن را با محتوای AC جمع می کند. دستورالعمل موجود در آدرس ۳۵ در شکل ۵-۲ (ج) دارای بیت I=۱ است. بنابراین این دستور بعنوان یک دستورالعمل با آدرس غیر مستقیم شناخته می شود. بخش آدرس عدد دو دویی معادل ۳۰۰ است. کنترل به آدرس ۳۰۰ مراجعه می نماید تا آدرس عملوند را بیابد. آدرس عملوند در این مورد ۱۳۵۰ است. سپس عملوند حاصل از آدرس ۱۳۵۰ با محتوای AC جمع می شود. لذا دستورالعمل ها با آدرس غیر مستقیم 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 مقصد<sup>۵</sup> تعریف می کنیم. بنابراین آدرس موثر در دستورالعمل شکل ۵-۲ (ب) عبارتست از ۴۵۷ و در دستور شکل ۵-۲ (ج) برابر ۱۳۵۰ می باشد.

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

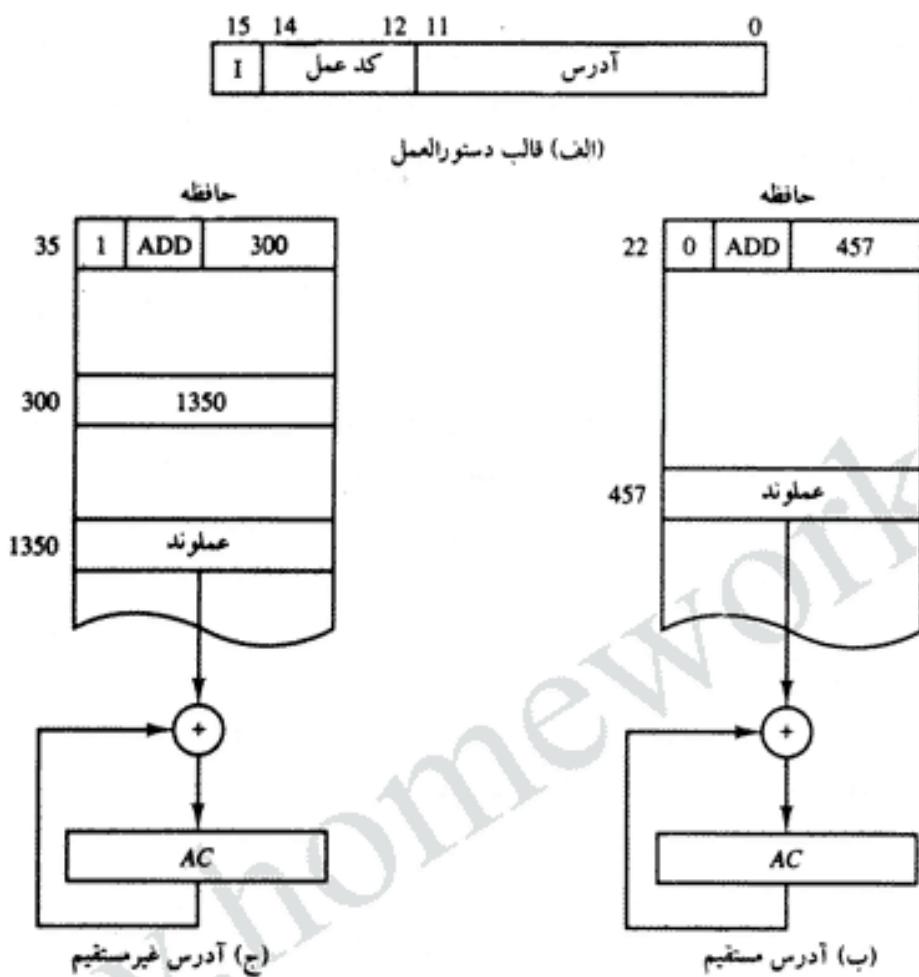
1- Clear

2- Immediate

3- Direct Address

4- Indirect Address

5- Destination



شکل ۵-۲ نمایش آدرس مستقیم و غیرمستقیم

اشارة گری به آرایه داده بکار می‌رود. همانطور که در کامپیووترهای تجاری مرسوم است این اشاره گر را می‌توان بجای قرار دادن در یک حافظه، در یک ثبات پردازشگر قرار داد.

## ۵-۲ ثبات‌های کامپیووتر

دستورالعمل‌های کامپیووتر معمولاً در مکان‌هایی از حافظه بطور متوالی ذخیره شده و هر یک بتوت اجرا می‌گردد. 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 ثبات در واحد کنترل برای ذخیره کد دستورالعمل، پس از خواندن آن از حافظه، لازم است. بعلاوه کامپیووتر به ثبات‌هایی در پردازشگر جهت دستکاری داده‌ها و نیز یک ثبات برای ذخیره‌سازی آدرس حافظه نیاز دارد. این نیازها، وجود ثبات‌های شکل ۵-۳ را در سیستم الزامی می‌دارد.

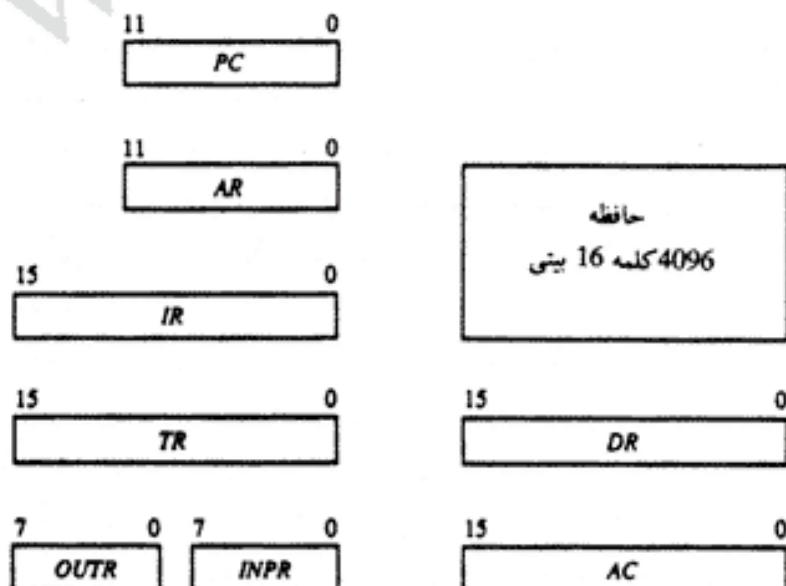
این ثبات‌ها در جدول ۱-۵ هم همراه با شرحی مختصر از کارشان و تعداد بیت‌های آنها آورده شده‌اند.

واحد حافظه 4096 کلمه 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. ثبات داده (DR) عملوند خوانده شده از حافظه را نگه می‌دارد. ثبات ابیاره (AC) یک ثبات همه منظوره در پردازش است. دستور خوانده شده از حافظه در ثبات دستورالعمل (IR) قرار می‌گیرد. ثبات موقت (TR) برای نگهداری موقت داده‌ها در حین پردازش به کار می‌رود.

ثبات 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 کامپیووتر ساده

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



شکل ۲-۳ ثبات‌ها و حافظه کامپیووتر پایه

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

دو ثبات برای ورودی و خروجی بکار رفته اند. ثبات ورودی (INPR) یک کاراکتر هشت بیتی را از یک وسیله ورودی دریافت می کند. ثبات خروجی (OUTR) یک کاراکتر هشت بیتی را برای یک وسیله خروجی نگه می دارد.

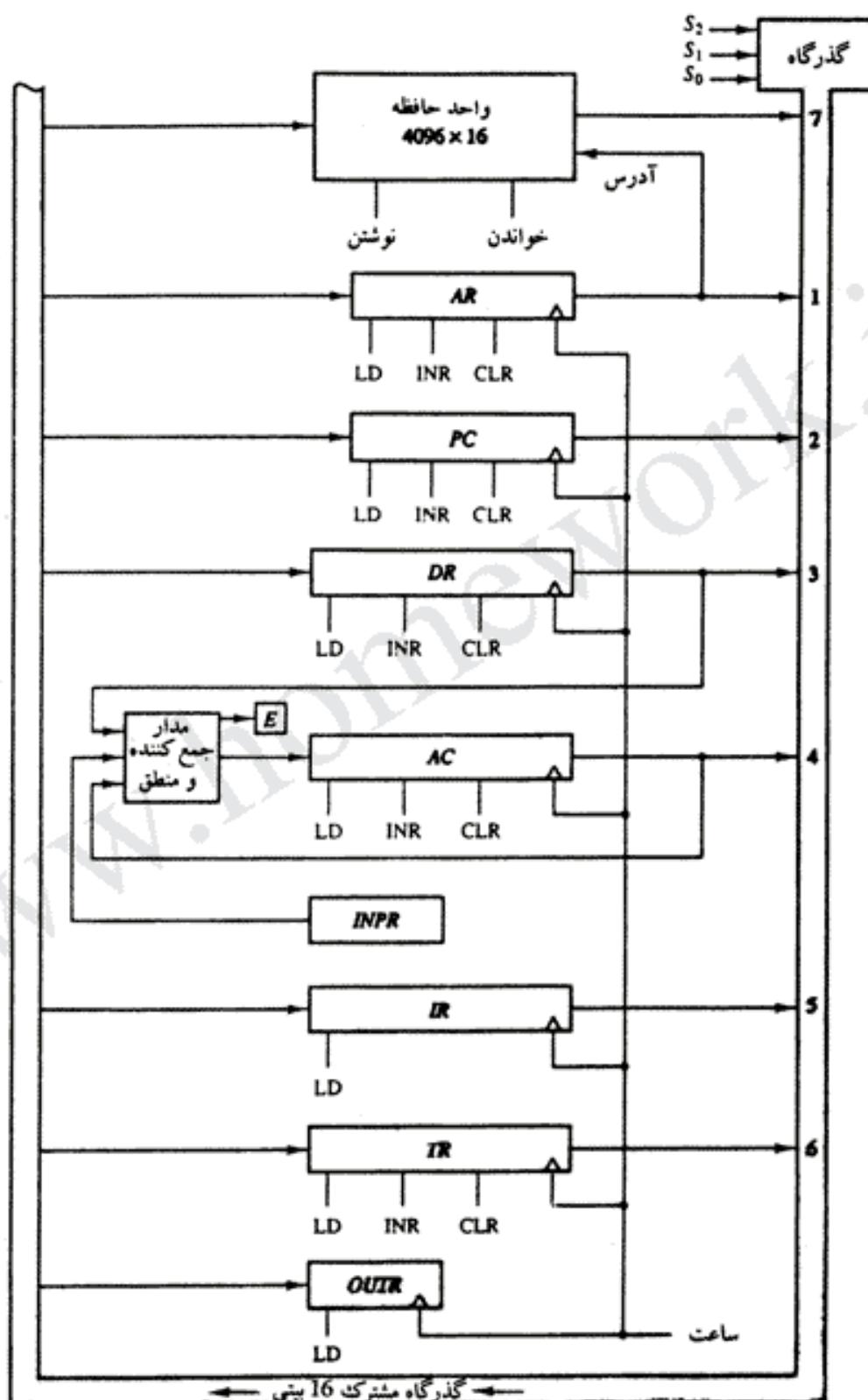
### سیستم گذرگاه مشترک

کامپیوتر مورد نظر دارای هشت ثبات، یک واحد حافظه و یک واحد کنترل، (که در بخش ۴-۵ ارائه خواهد شد) می باشد. برای انتقال اطلاعات از یک ثبات به ثبات دیگر و یا بین حافظه و ثبات ها، مسیرهایی باید ایجاد شود. اگر بین خروجی هر ثبات با ورودی ثبات دیگری اتصال برقرار شود، تعداد سیم ها 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 می توان یک گذرگاه سیستم ساخت. اتصالات ثبات ها و حافظه های کامپیوتر مورد نظر به گذرگاه مشترک فوق در شکل ۴-۵ نشان داده شده است.

خروچی های هفت ثبات و حافظه ها به گذرگاه مشترک متصل شده اند. در هر مقطع زمانی، خروجی خاصی که برای خطوط گذرگاه انتخاب می شود توسط مقادیر دودویسی انتخاب  $S_0, S_1, S_2$  معین می گردد. مقادیر دهدی که در امتداد هر خروجی نوشته شده معادل عدد دودویی لازم را برای انتخاب نشان می دهد. مثلاً در کنار خروجی DR عدد ۳ نوشته شده است، لذا ۱۶ بیت خروجی DR هنگامی روی گذرگاه قرار می گیرد که  $S_2S_1S_0 = 011$  باشد زیرا این مقدار معادل دودویی عدد ۳ است. خطوط گذرگاه مشترک به ورودی های همه ثبات ها و ورودی داده حافظه ها نیز وصل شده اند.

هر ثبات خاصی که ورودی LD آن فعال باشد، داده ها را با پالس بعدی از گذرگاه دریافت می کند. حافظه هم بهنگام فعال شده ورودی "نوشتن"<sup>۲</sup> محتوای گذرگاه را دریافت می نماید. هنگام فعال شدن ورودی "خواندن"<sup>۳</sup> حافظه، خروجی آن روی گذرگاه قرار می گیرد.

چهار ثبات DR، AC، IR و TR هر یک ۱۶ بیتی هستند. دو ثبات AR و PC هر یک ۱۲ بیت دارند زیرا آدرس حافظه را نگه می دارند. وقتی که محتوای AR یا PC به گذرگاه مشترک ۱۶ بیتی اعمال شود،



شکل ۵-۲ ثبات‌های کامپیووتر پایه متصل به یک گذرگاه مشترک

چهار بیت بالا رتبه ۰ می شوند. وقتی که AR یا PC اطلاعات را از گذرگاه دریافت می کنند، تنها ۱۲ بیت پائین 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 تحويل 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 به ورودی شش ثبات و حافظه متصلند. پنج ثبات دارای سه ورودی کنترل می باشند: LD (بارشدن)<sup>۱</sup> INR (افزایش)<sup>۲</sup> و CLR (پاک شدن)<sup>۳</sup>. این نوع ثبات معادل یک شمارنده دودویی با بارشدن موازی و نیز پاک کننده همزمان<sup>۴</sup> مطابق شکل ۱۱-۲ است. عمل افزایش با فعال شدن ورودی "شمارش"<sup>۵</sup> شمارنده میسر است. دو ثبات هر یک فقط دارای یک ورودی LD می باشند. این نوع ثبات در شکل ۱۱-۷ 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 ریز عمل هایی<sup>۶</sup> مانند متمم سازی AC و شیفت AC بکار می روند. مجموعه ورودی ۱۶ بیتی دیگری از ثبات DR می آید. ورودی هایی که از DR و AC می آیند برای ریز عمل های حسابی و منطقی مانند جمع AND با AC یا DR با AC کردن می روند. نتیجه جمع به AC و رقم نقلی نهایی<sup>۷</sup> به فلیپ فلاپ E (بیت گسترش<sup>۸</sup> AC) منتقل می شوند. سومین مجموعه هشت بیتی به ثبات ورودی INPR وصل است. عملکرد INPR و OUTR در بخش ۱۱-۷ 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. لب 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

$$DR \leftarrow AC \quad AC \leftarrow DR$$

می توانند هم زمان اجرا شوند. این اعمال با قرار گرفتن محتوای AC بر روی گذرگاه ( $S_2S_1S_0 = 1001$ )، 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.

### ۵-۳ دستورالعمل‌های کامپیووتر

کامپیووتر پایه مورد بحث دارای سه قالب دستورالعمل مطابق شکل ۵-۵ می باشد. این قالب‌ها ۱۶ بیتی هستند. بخش عمل کد دستور سه بیتی است و مفهوم بقیه بیت‌ها به نوع کد وابسته است. دستورالعمل‌های ارجاع به حافظه از ۱۲ بیت برای تعیین آدرس و ۱ بیت برای مشخص کردن روش آدرس دهی I استفاده می کنند. برای آدرس دهی مستقیم، ۱ برابر ۰ و برای آدرس غیرمستقیم I مساوی ۱ است، شکل ۵-۲. دستورالعمل‌های ارجاع به ثبات با کد عملی ۱۱۱ و یک بیت ۰ در منتها الیه سمت چپ آن (بیت ۱۵) قابل تشخیص‌اند. دستورالعمل‌های ثباتی عمل را برروی AC و یا هر تستی برروی آن را مشخص می نمایند. در این دستورات عملوندی از حافظه موردنیاز نیست؛ بنابراین از ۱۲ 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 ۱۱۱ و ۱ در بیت منتها الیه سمت چپ دستور قابل تشخیص‌اند. بقیه ۱۲ بیت نوع عمل ورودی - خروجی و یا تستی که باید انجام شود را مشخص می سازد. واحد کنترل کامپیووتر نوع دستورالعمل را با توجه به بیت‌های مکان‌های ۱۲ تا ۱۵ دستورالعمل تشخیص می دهد. اگر سه بیت کد عمل در مکان‌های ۱۲ تا ۱۴ برابر ۱۱۱ نباشد دستور از نوع ارجاع به حافظه است و بیت ۱۵ یعنوان بیت روش آدرس دهی I در نظر گرفته می شود. اگر سه بیت کد عمل ۱۱۱

15	14	12	11	0
I	کد عمل	آدرس		کد عمل از ۰۰۰ تا ۱۱۰

(الف) دستورالعمل‌های حافظه‌ای

15	12	11	0
0	۱	۱	۱

(ب) دستورالعمل‌های ثبات

15	12	11	0
1	۱	۱	۱

(ج) دستورالعمل‌های ورودی - خروجی

شکل ۵-۵ قالب دستورالعمل‌ها در کامپیووتر پایه

باشد، واحد کنترل بیت مکان 15 را بررسی می نماید. اگر این بیت 0 باشد، دستور ازنوع ارجاع به ثبات است. اگر بیت مذکور 1 باشد، دستور ازنوع ورودی - خروجی است. هنگامی که کد عمل 111 باشد بیت مکان 15 با 1 مشخص می شود ولی بعنوان بیت روش آدرسی دهی بکار نمی رود.

گفته شده که برای کد عمل تنها سه بیت به کار رفته است. لذا ممکن است به نظر برسد که کامپیوتر به حد اکثر 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 دستورالعمل ها از هشت تجاوز می کنند. در واقع در این حال، دستورالعمل های مورد استفاده در کامپیوتر 25 خواهد بود دستورالعمل های کامپیوتر در جدول ۵-۱ ملاحظه می شوند. سهیل مشخص کننده دستورالعمل ها از یک کلمه سه حرفی تشکیل شده و خلاصه عمل را برای برنامه نویس و کاربر مشخص می کنند. کد مبنای

جدول ۵-۲ دستورالعمل های کامپیوتر پایه

کد شانزده شانزده			
سبل	$I = 0$	$I = 1$	شرح
AND	0xxx	8xxx	AND کردن کلمه حافظه با
ADD	1xxx	9xxx	جمع کردن کلمه حافظه با
LDA	2xxx	Axxx	بارگردان کلمه حافظه در
STA	3xxx	Bxxx	ذخیره محتوای AC در حافظه
BUN	4xxx	Cxxx	انشعاب نامشروع
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		منس کردن AC
CME	7100		منس 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		پردازنی کاراکتر از AC و انتقال آن به خروجی
SKI	F200		گذر مبتنی بر بروم ورودی
SKO	F100		گذر مبتنی بر بروم خروجی
ION	F080		فعال کردن وقفه ها
IOF	F040		غیرفعال کردن وقفه ها

شانزده برابر با عدد معادل شانزده کد دو دویی بکار رفته در دستورالعمل است. با استفاده از معادل شانزده، ما 16 بیت کد دستورالعمل را به چهار رقم کاهش می دهیم که هر رقم معادل چهار بیت است. دستورالعمل ارجاع به حافظه دارای بخش آدرس 12 بیتی است. این بخش از دستور با سه X مشخص شده و به مفهوم سه رقم در مبنای شانزده است. آخرین بیت دستورالعمل با نعاد آ 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 می باشد.

دستورات ارجاع به ثبات برای مشخص کردن عملوند از 16 بیت استفاده می نمایند. چهار بیت سمت چپ همیشه 0111 هستند. که معادل 7 در مبنای شانزده می باشد. سه رقم دیگر معادل دو دویی 12 بیت باقیمانده را در مبنای شانزده نشان می دهند. دستورات ورودی - خروجی هم از تمام 16 بیت برای مشخص کردن عمل استفاده می کنند. چهار بیت آخر همیشه 1111 می باشند که معادل F در مبنای شانزده است.

### کامل بودن مجموعه دستورات

قبل از بررسی عمل هایی که دستورات انجام می دهند، اجازه بدھید درباره نوع دستوراتی که باید در کامپیوتر در نظر داشت بحث کنیم. یک کامپیوتر باید دستورالعمل هایی را داشته باشد که کاربر بتواند بکمک آنها برنامه هایی بزرگ ماین ماشین برای محاسبه توابعی که قابل محاسبه هستند بتواند. اگر کامپیوتر دارای دستورالعمل های کافی در هر یک از رسته های زیر باشد مجموعه دستورات آن کافی تلقی می گردد:

- ۱- دستورالعمل های حسابی، منطقی و شیفت
- ۲- دستورالعمل هایی برای تبادل اطلاعات با حافظه و ثبات های پردازشگر
- ۳- دستورات کنترل برنامه همراه با دستورالعمل هایی که شرایط وضعیتی<sup>۱</sup> راچک می کنند
- ۴- دستورات ورودی و خروجی

دستورات حسابی منطقی قابلیت های محاسباتی را برای پردازش داده هایی که کاربر مایل است بکار برد فراهم می آورند. بخش اعظم اطلاعات دو دویی در یک کامپیوتر دیجیتال در حافظه ذخیره می شود، ولی تمام محاسبات در ثبات های پردازشگر انجام می گردد. بنابراین کاربر باید امکان انتقال اطلاعات را بین دو واحد داشته باشد. تصمیم گیری، از جمله قابلیت های مهم در کامپیوتراهای دیجیتال است. بعنوان مثال دو عدد را می توان 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 دستورالعمل های کنترل برنامه مانند دستورات انشعاب برای تغییر ترتیب اجرای برنامه بکار می روند. دستورات ورودی - خروجی برای ارتباط بین کامپیوتر و کاربر لازمند. برنامه ها و داده ها باید به حافظه منتقل گردند و نتایج محاسبات باید به کاربر بازگردانده شود.

1- Status Condition

دستورات لیست شده در جدول ۵-۲ 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 (INC) و افزایش AC (CMA) وجود دارد. با این سه دستور می‌توان عده‌های دودویی را جمع و تفیریک کرد بشرطی که اعداد منفی به‌گل متمم ۲ علامت دار نشان داده شوند. با استفاده از دستورالعمل‌های چرخش CIR و CIL می‌توان شیفت‌های حسابی و یا هر گونه شیفت دیگر را انجام داد. ضرب و تقسیم با استفاده از جمع و تفیریک 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 ۴-۶).

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 انتساب BUN و BSA و ISZ همراه با چهار دستور گذر<sup>۱</sup> امکان کنترل برنامه و بررسی شرایط وضعیتی بوجود می‌آورند. دستورالعمل‌های ورودی INP و خروجی OUT موجب انتقال اطلاعات بین کامپیوتر و وسائل خارجی می‌گردند.

هر چند مجموعه دستورالعمل‌ها برای کامپیوتر پایه نسبتاً کافی است ولی کارا نیست زیرا دستوراتی که کراراً بکار می‌روند بسرعت اجرا نمی‌شوند. مجموعه کارایی از دستورات باید شامل دستورالعمل‌های تفیریق، ضرب، OR و XOR<sup>۲</sup> 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 ۵-۲ 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 پتأخیر می‌اندازیم.

## ۵-۴ زمانبندی و کنترل

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

1- Skip

2- Exclusive OR (XOR)

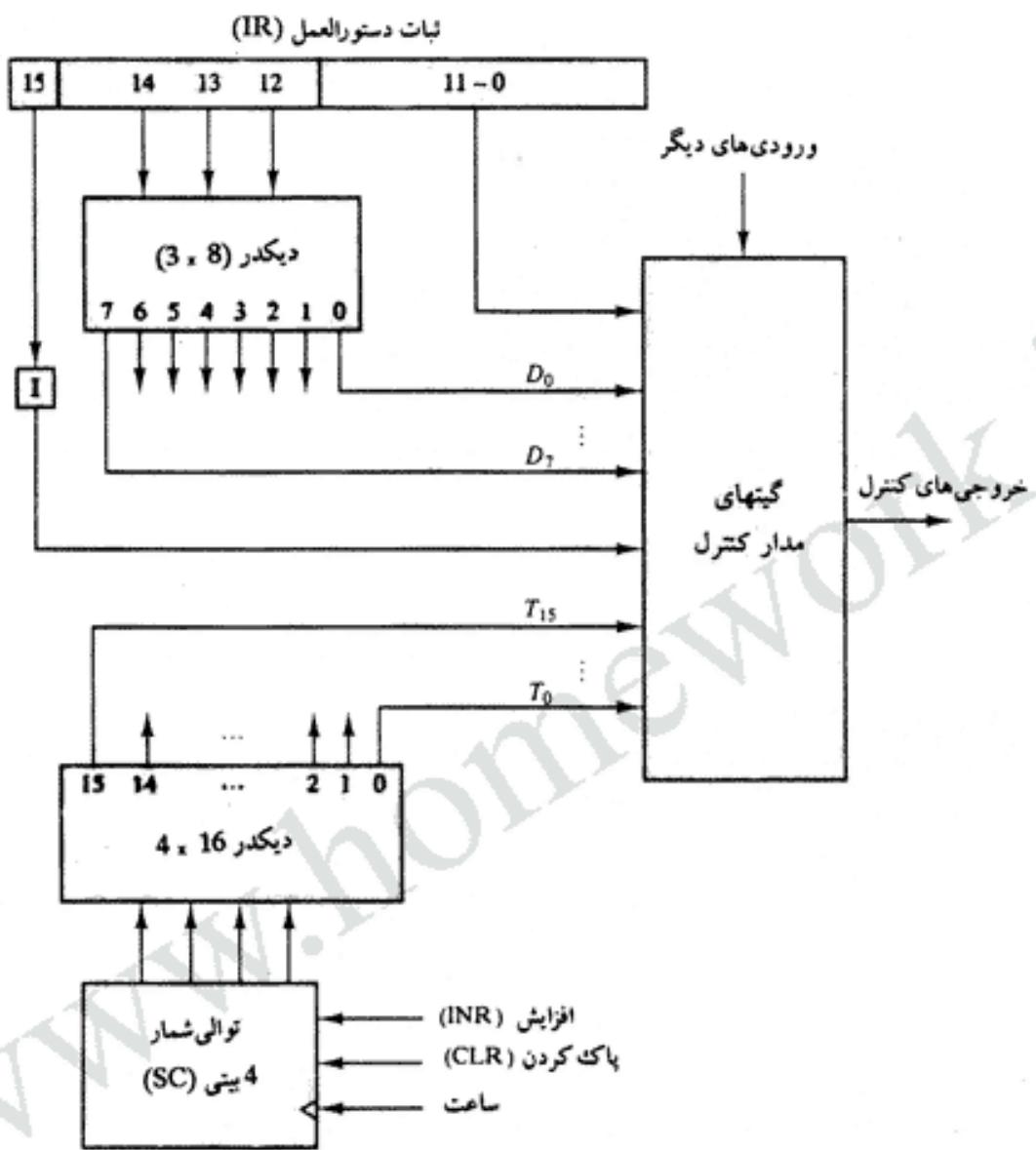
اعمال می شوند. پالس های ساعت حالت هیچ ثباتی را عوض نمی کنند مگر اینکه ثبات توسط یک سیگنال کنترل فعال شود. سیگنال های کنترل در واحد کنترل تولید می شوند و ورودی های کنترل را برای مولتی پلکسر در گذرگاه مشترک، ورودی های کنترل در ثبات های پردازشگر و ریز اعمال در انباره AC فراهم می آورند.

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 گیت ها، فلیپ فلاپ ها، دیکدرها و سایر مدارهای دیجیتال پیاده سازی می شود. حسن این روش داشتن سرعت بالا است که با بهینه کردن آن حاصل می شود. در سازمان ریز برنامه نویسی شده اطلاعات کنترل در یک حافظه کنترل ذخیره شده است. حافظه کنترل به صورتی برنامه نویسی می شود که رشته ریز عمل های لازم را اجرا کند. در کنترل سخت افزاری، 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 هر گونه 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 می کنیم. واحد کنترل ریز برنامه نویسی شده برای کامپیوتری مشابه در فصل ۷ آمده است.

بلاک دیاگرام واحد مزبور در شکل ۶-۵ دیده می شود. 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<sup>۱</sup> (IR) قرار می گیرد. مکان این ثبات در سیستم گذرگاه مشترک در شکل ۴-۵ مشخص شده است. ثبات دستورالعمل که مجددآ در شکل ۶-۶ آورده شده به سه بخش تقسیم شده است: بیت I، کد عمل، و بیت ۰ تا ۱۱. کد عمل واقع در بیت های ۱۲ تا ۱۴، توسط یک دیکدر  $8 \times 3$  دیکد می شود. هشت خروجی دیکدر با  $D_0$  تا  $D_7$  مشخص شده اند. زیر نویس دهدی می بین مقدار دودویی کد عمل است. بیت ۱۵ دستورالعمل به فلیپ فلاپ I منتقل می شود. بیت های ۰ تا ۱۱ به مدار کنترل اعمال می گردند. شمارنده چهار بیتی می تواند از ۰ تا ۱۵ متواالیاً بشمارد. خروجی های این شمارنده به صورت ۱۶ سیگنال زمانبندی  $T_0$  تا  $T_{15}$  دیکد می شود. مدار داخلی گیت های کنترل بعداً بهنگام بررسی مشروحتر طراحی کامپیوتر بدست خواهد آمد.

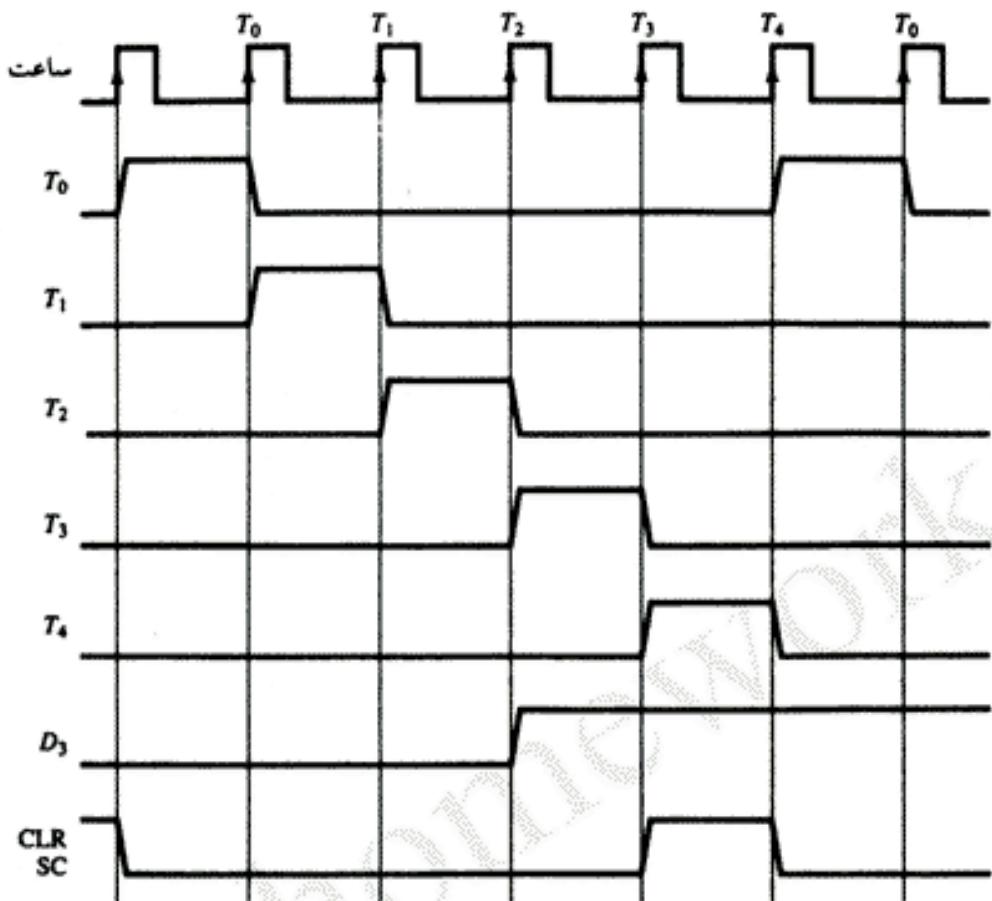
شمارنده SC را می توان بطور همزمان<sup>۲</sup> افزایش داد یا پاک کرد (شمارنده شکل ۲-۱۱ ملاحظه شود). اغلب اوقات، شمارنده افزایش می یابد تا رشته سیگنالهای زمانبندی متواالی را از خروجی دیکدر ایجاد کند. هر از گاه یکبار این شمارنده ۰ می شود و سبب می گردد تا سیگنال زمانبندی بعدی  $T_0$  شود. مثلاً حالتی را در نظر بگیرید که SC افزایش داده شود و تا سیگنال های  $T_0, T_1, T_2, T_3, T_4$  را بطور متواالی فراهم نماید. در  $T_4$ ، اگر خروجی دیکدر  $D_3$  فعال باشد، SC صفر می شود. این موضوع بصورت نمادی با عبارت زیر نشان داده می شود:

$$D_3 \quad T_4: \quad SC \leftarrow 0$$



شکل ۵-۶ واحد کنترل کامپیوتر

دیاگرام زمانی شکل ۵-۷ ارتباط زمانی سیگنال‌های کنترل را نشان می‌دهد. شمارنده SC بهنگام لبه مثبت پالس ساعت واکنش نشان داده و عمل می‌کند. در آغاز، ورودی CLR شمارنده SC فعال است. اولین لبه مثبت پالس ساعت، SC را ۰ می‌کند که این بنویه خود سیگنال زمانبندی  $T_0$  را که از دیکدر خارج می‌شود فعال می‌نماید.  $T_0$  در طول یک پالس ساعت فعال باقی می‌ماند. لبه مثبت پالس  $T_0$  فقط آن دسته از ثباتها را فعال می‌کند که ورودی کنترلشان به سیگنال زمانبندی  $T_0$  متصل باشد. SC با هر لبه مثبت پالس ساعت یک واحد افزایش می‌یابد، مگر اینکه CLR آن فعال باشد. این افزایش سبب تولید سیگنال‌های  $T_0, T_1, T_2, T_3, T_4$  و غیره مطابق شکل می‌گردد (به رابطه بین سیگنال



شکل ۵-۷ مثالی از سیگنال‌های زمانبندی واحد کنترل

زمانبندی و لبه مثبت پالس مربوطه توجه کنید). اگر SC صفر نشد، 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 عرض  $T_5$  که می‌باید با افزایش SC فعال می‌شد، سیگنال  $T_0$  بعلت پاک شدن شمارنده SC تولید شود.

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

انتظاری<sup>۱</sup> 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 لبه‌های پالس ساعت و سیگنال‌های 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 می‌کند.  $T_0$  در تمام مدت یک سیکل ساعت فعال است. در طول این مدت محتوای PC روی گذرگاه قرار می‌گیرد. (با  $S_2S_1S_0 = 010$ ) و ورودی بارشدن LD ثبات AR فعال می‌شود. انتقال واقعی تا پایان ساعت و در واقع تا آغاز لبه مثبت پالس بعدی رخ نمی‌دهد. همین لبه SC را از 0000 به 0001 افزایش می‌دهد. پالس ساعت بعدی  $T_1$  را فعال و  $T_0$  را غیرفعال می‌سازد.

## ۵-۵ سیکل دستورالعمل

برنامه‌ای که در واحد حافظه کامپیوتر ذخیره شده از دنباله‌ای از دستورالعمل‌ها تشکیل شده است. این برنامه طی سیکل‌های اجرایی دستورالعمل اجرا می‌شود. هر سیکل دستورالعمل نیز خود به مجموعه‌ای از سیکل‌های جزیی یا فازها تقسیم می‌شود. در کامپیوتر پایه هر سیکل دستور متتشکل 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<sup>۲</sup>، کنترل به مرحله<sup>۱</sup> بازگشته و برداشت<sup>۲</sup>، دیکد و اجرای<sup>۳</sup> دستور بعدی را انجام می‌دهد. این فرآیند بطور نامحدود ادامه می‌یابد مگر اینکه کامپیوتر با دستور HALT مواجه شود.

### برداشت و دیکد

در آغاز شمارنده برنامه، PC با آدرس اولین دستورالعمل برنامه بار می‌شود. شمارنده SC پاک شده و در نتیجه سیگنال  $T_0$  را تولید می‌نماید. پس از هر پالس ساعت، SC یک واحد افزایش می‌یابد تا رشته سیگنال‌های  $T_0$ ،  $T_1$ ،  $T_2$  و غیره متوالیاً تولید گردد. ریز عمل‌های مربوط به فاز برداشت و دیکد را می‌توان توسط عبارات انتقال ثبات زیر مشخص نمود.

$$T_0 : AR \leftarrow PC$$

---

1- Wait

2- Fetch

3- Execute

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

$$T_2 : D_0, \dots, D_7 \leftarrow IR(0-11), I \leftarrow IR(15)$$

از آنجا که فقط AR به ورودی‌های آدرس حافظه متصل می‌باشد، 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 می‌گیرد. همزمان با عمل اخیر، PC یک واحد افزایش می‌یابد تا حاوی دستورالعمل بعدی در برنامه باشد. در زمان  $T_2$ ، کد عمل موجود در IR، دیکد می‌شود و بیت غیرمستقیم به فلیپ فلاپ I منتقل می‌گردد و نهایتاً بخش آدرس دستورالعمل به AR منتقال می‌یابد. توجه داشته باشید که SC پس از هر پالس ساعت افزایش می‌یابد تا سیگنال‌های  $T_1, T_0$ ،  $T_2$  متوالیاً تولید گردند.

شکل ۵-۸ چگونگی پیاده سازی دو عبارت انتقال ثبات را در سیستم گذرگاه نشان می‌دهد. برای فراهم کردن مسیر داده‌ها 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 ورودی‌های انتخاب کننده  $S_2S_1S_0 = 010$

۲- انتقال محتوای گذرگاه به AR با تواناکردن ورودی LD مربوط به AR

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

دوم

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

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.

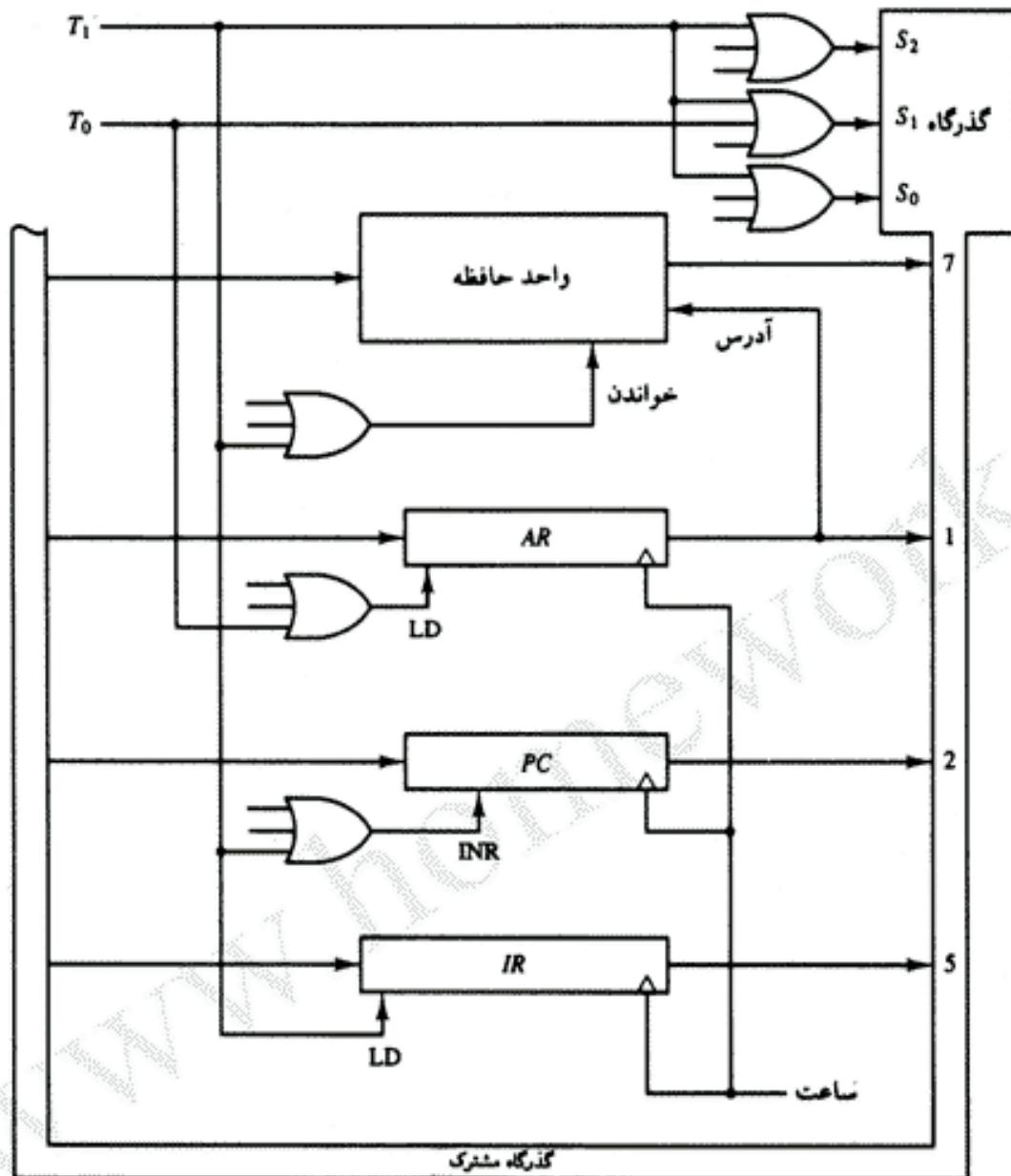
۳- محتوای گذرگاه با توانا شدن LD ثبات IR، به آن منتقل گردد.

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

شکل ۵-۸ بخشی از گذرگاه سیستم را همراه با اتصالات  $T_0$  و  $T_1$  به ورودی‌های کنترل ثبات‌ها، حافظه‌ها و ورودی‌های انتخاب کننده نشان می‌دهد. چون توابع کنترلی دیگری که اعمال مشابهی را انجام می‌دهند نیز وجود دارند لذا گیت‌های OR چند ورودی در دیاگرام آورده شده‌اند.

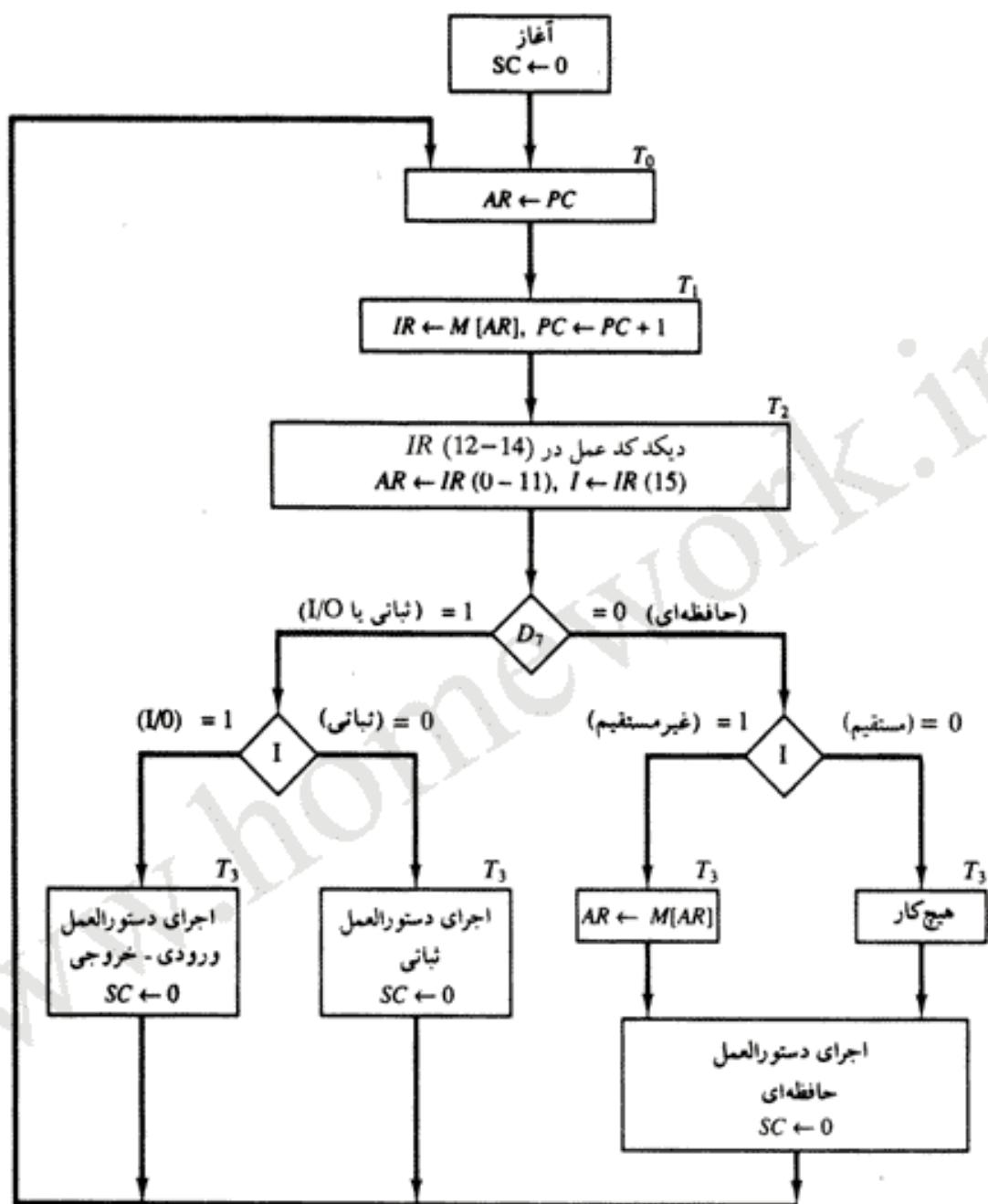
### تعیین نوع دستورالعمل

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



شکل ۵-۸ انتقال ثبات‌های مربوط به فاز برداشت

کنترل نوع دستوری را که درست چند لحظه قبل خوانده شده معین می‌کند. فلوچارت شکل ۵-۹ 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 فلوچارت سیکل دستورالعمل (آرایش اولیه)

بیت دستورالعمل را که در این لحظه در فلیپ فلاب I است چک می‌کند. اگر  $I = 0$  باشد دستور از نوع ارجاع به حافظه با آدرس غیرمستقیم است. در اینصورت لازم است که آدرس موقّت از حافظه خوانده شود. ریز عمل مربوط به حالت آدرس غیرمستقیم نوسط 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  $T_3$  فعال می‌گردد که برای چهار مسیر توسط روابط زیر مشخص می‌شود.

$$D_7'IT_3 : AR \leftarrow M[AR]$$

هیچ کار :

اجرای دستور ارجاع به ثبات :

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  $I=1 = D_7'T_3$  باشد، باید افزایش یابد تا اجرای دستورالعمل ارجاع به حافظه با متغیر زمانی  $T_4$  ادامه یابد. یک دستورالعمل ارجاع به ثبات یا ورودی - خروجی با پالس ساعت مربوط به  $T_3$  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 اینجا این قرارداد را می‌پذیریم که بازاء هر افزایش عبارت  $SC + SC \leftarrow SC$  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  $I=1$  و  $D_7=0$  است توسط کنترل قابل تشخیص‌اند. بیت‌های 0 تا 11 این دستورات برای تعیین کی از 12 دستورالعمل بکار می‌روند. این 12 بیت در (0-11) IR قرار دارند و در طول  $T_2$  نیز به AR منتقل شده‌اند. توابع کنترل و ریز عمل‌ها برای دستورات ارجاع به ثبات در جدول ۳-۵ لیست شده‌اند. این دستورالعمل‌ها با انتقال پالس ساعت در  $T_3$  اجرا می‌شوند. هر یک از توابع کنترل به یک  $D_7'I'T_3$  نیاز دارند که ما آن را به آن نشان می‌دهیم. توابع کنترل توسط یکی از بیت‌های (0-11) IR از هم متمایز می‌گردند. با تخصیص نماد یا سمبل  $B_i$  به بیت از IR، همه توابع کنترلی را می‌توان با  $B_i$  نشان داد. مثلاً دستورالعمل CLA دارای کد مبنای شانزده 7800 است (جدول ۳-۲).

## جدول ۵-۳ اجرای دستورالعمل‌های ثباتی

$D_7 I' T_3 = r$	مشترک در همه دستورالعمل‌های ثباتی
$IR(i) = B_i$	بیتی در $(0-11)$ که عمل را مشخص می‌کند.
$r: SC \leftarrow 0$	پاک کردن SC
$CLA \quad rB_{11}: AC \leftarrow 0$	پاک کردن AC
$CLE \quad rB_{10}: E \leftarrow 0$	پاک کردن E
$CMA \quad rB_9: AC \leftarrow \overline{AC}$	متضم کردن AC
$CME \quad rB_8: E \leftarrow \overline{E}$	متضم کردن E
$CIR \quad rB_7: AC \leftarrow shr AC, AC(15) \leftarrow E, E \leftarrow AC(0)$	شبفت چرخشی به راست
$CIL \quad rB_6: AC \leftarrow shl AC, AC(0) \leftarrow E, E \leftarrow AC(15)$	شبفت چرخشی به چپ
$INC \quad rB_5: AC \leftarrow AC + 1$	افزایش AC
$SPA \quad rB_4: If (AC(15) = 0) then (PC \leftarrow PC + 1)$	گذار در صورت مثبت بودن
$SNA \quad rB_3: If (AC(15) = 1) then (PC \leftarrow PC + 1)$	گذار در صورت منفی بودن
$SZA \quad rB_2: If (AC = 0) then (PC \leftarrow PC + 1)$	گذار در صورت صفر بودن
$SZE \quad rB_1: If (E = 0) then (PC \leftarrow PC + 1)$	گذار در صورت صفر بودن
$HLT \quad rB_0: S \leftarrow 0$	توقف کامپیوتر

ملاحظه شود)، که معادل عدد دودویی 0000 0000 1000 1011 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  $D_7 I' T_3 B_{11} = rB_{11}$  خواهد بود. اجرای دستورالعمل ارجاع به ثبات در زمان  $T_3$  پایان می‌پذیرد. شمارنده SC پاک شده و کنترل برای برداشت دستور بعدی با  $T_0$  به عقب بر می‌گردد.

هفت دستور اول ثباتی، اعمال پاک کردن، مکمل سازی، چرخش و افزایش روی ثبات‌های AC و E را انجام می‌دهند. چهار دستور بعدی موجب گذار<sup>۱</sup> (رد شدن) از دستور بعدی در صورت برقرار بودن یک شرط معین می‌شود. این گذار با افزایش PC صورت می‌گیرد (علاوه بر افزایش آن در فاز برداشت دستورالعمل در فاز  $T_1$ ). عبارت‌های کنترل شرطی را باید بخشنی از شرایط کنترلی تلقی کرد. هرگاه بیت علامت AC $=0$  شود محتوای آن یک عدد مثبت است و اگر AC $=1$  باشد منفی است. اگر تمام فلیپ‌فلاب‌های ثبات صفر باشند محتوای AC صفر است. دستور HLT فلیپ‌فلاب استارت - استارت<sup>۲</sup> را پاک می‌کند که موجب توقف شمارنده SC می‌گردد. برای بکار انداختن مجدد کامپیوتر فلیپ‌فلاب استارت - استارت - استارت باید بصورت دستی 1 شود.

## ۵-۶ دستورالعمل‌های ارجاع به حافظه

در تعیین ریز عملیاتی موردنیاز 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

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 مؤثر 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 سبیلیک هر دستور هم در جدول بر حسب علائم انتقال ثبات مشخص شده است. اجرای واقعی هر دستور در سیستم نیاز به یک رشته ریز عمل 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 ریز عمل‌های لازم را برای اجرای آنها لیست می‌کنیم. فلوچارتی که تمام این ریز اعمال را در اختیار می‌گذارد در آخر این بخش آمده است.

### AC با AND

این دستور عمل منطقی AND بر روی جفت بیت‌های AC و حافظه‌ای که توسط آدرس مؤثرش معین می‌شود را انجام می‌دهد. نتیجه عمل فوق در انتهای به AC منتقل می‌شود. ریز عمل‌هایی که این دستور العمل 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 دستور العمل‌های حافظه‌ای

عمل	دیکدر	توضیح سبیلک
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 SC 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 محتوا حافظه مشخص شده توسط آدرس موثر را با مقدار AC جمع می کند. حاصل جمع به AC و رقم نقلی خروجی  $C_{out}$  به فلیپفلاب E (بیت گسترش ابزاره)<sup>1</sup> منتقل می شود. ریز اعمال لازم برای این دستور عبارتست از

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

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

مشابه با دستور قبل دو سینگال زمانی  $T_4$  و  $T_5$  بکار رفته اند ولی خروجی  $D_1$  دیگر بجای  $D_0$  که در عمل AND بکار رفت فعال است. پس از برداشت و دیگر دستور تنها یک خروجی دیگر 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.

### AC با LDA

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

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

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

با بازنگری مجدد شکل ۵-۴ 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: انشاب بدون شرط**

این دستور برنامه را به دستور العملی که توسط آدرس موثرش مشخص شده منتقل می‌نماید. بخارط داشته باشد که PC آدرس دستوری که در سیکل بعدی خوانده می‌شود را در خود نگمیدارد. PC در زمان  $T_1$  افزایش می‌یابد تا برای دستور بعدی در برنامه آماده باشد. دستور العمل 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 انشاب یافته است. این دستور العمل 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.

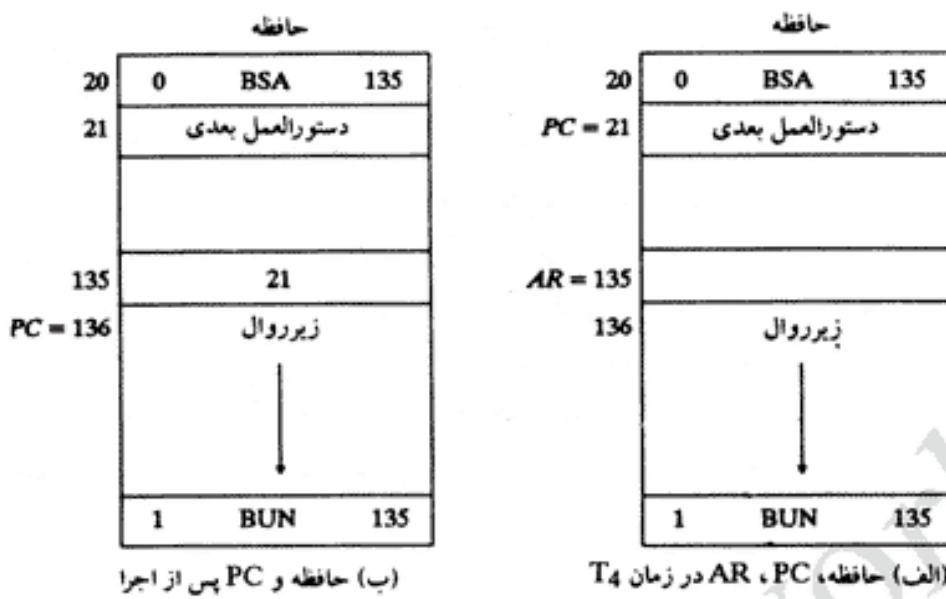
**BSA: انشاب با ذخیره آدرس برگشت**

این دستور برای انشاب به بخشی از برنامه که زیرروال<sup>۱</sup> 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 ۱ به PC منتقل می‌شود تا یعنوان آدرس اولین دستور در زیرروال مورد استفاده قرار گیرد. این عمل در جدول ۵-۴ 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 ۵-۱۰ دیده می‌شود فرض بر این است که دستور BSA در حافظه 20 قرار گرفته باشد. بیت I برابر ۰ و بخش آدرس دستور 135 تصویر 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. دستور العمل BSA عمل عددی زیر را اجرا می‌نماید.

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



شکل ۵-۱۰

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

دستورالعمل BSA عملی را انجام می‌دهد که معمولاً به آن فرآخونی<sup>۱</sup> زیرروال می‌گویند. دستور BUN در انتهای زیر روال عملی را که به آن بازگشت از زیرروال گویند انجام می‌دهد. در سیاری از کامپیوترهای تجاری آدرس برگشت از زیرروال در یکی از ثباتهای پردازشگر یا در بخشی از حافظه بنام پشتۀ<sup>۲</sup> ذخیره می‌شود. 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 ریز عمل زیر اجرا شود

$$D_5 \text{ T}_4: M[AR] \leftarrow PC, AR \leftarrow AR+1$$

$$D_5 \text{ T}_5: PC \leftarrow AR, SC \leftarrow 0$$

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

بعدی تکمیل می‌گردد. در  $T_5$  گذرگاه برای انتقال AR به PC مورد استفاده قرار می‌گیرد.

### ISZ: افزایش و گذراگر 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 تویس معمولاً یک عدد منفی را (بفرم نتیم 2) در حافظه قرار می‌دهد. این عدد منفی مرتبًا افزایش می‌یابد و نهایتاً به صفر می‌رسد. در این لحظه PC یک واحد اضافه می‌شود تا از دستور بعدی در برنامه گذر<sup>1</sup> 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 ریز عمل‌های زیر انجام می‌شود:

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

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

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

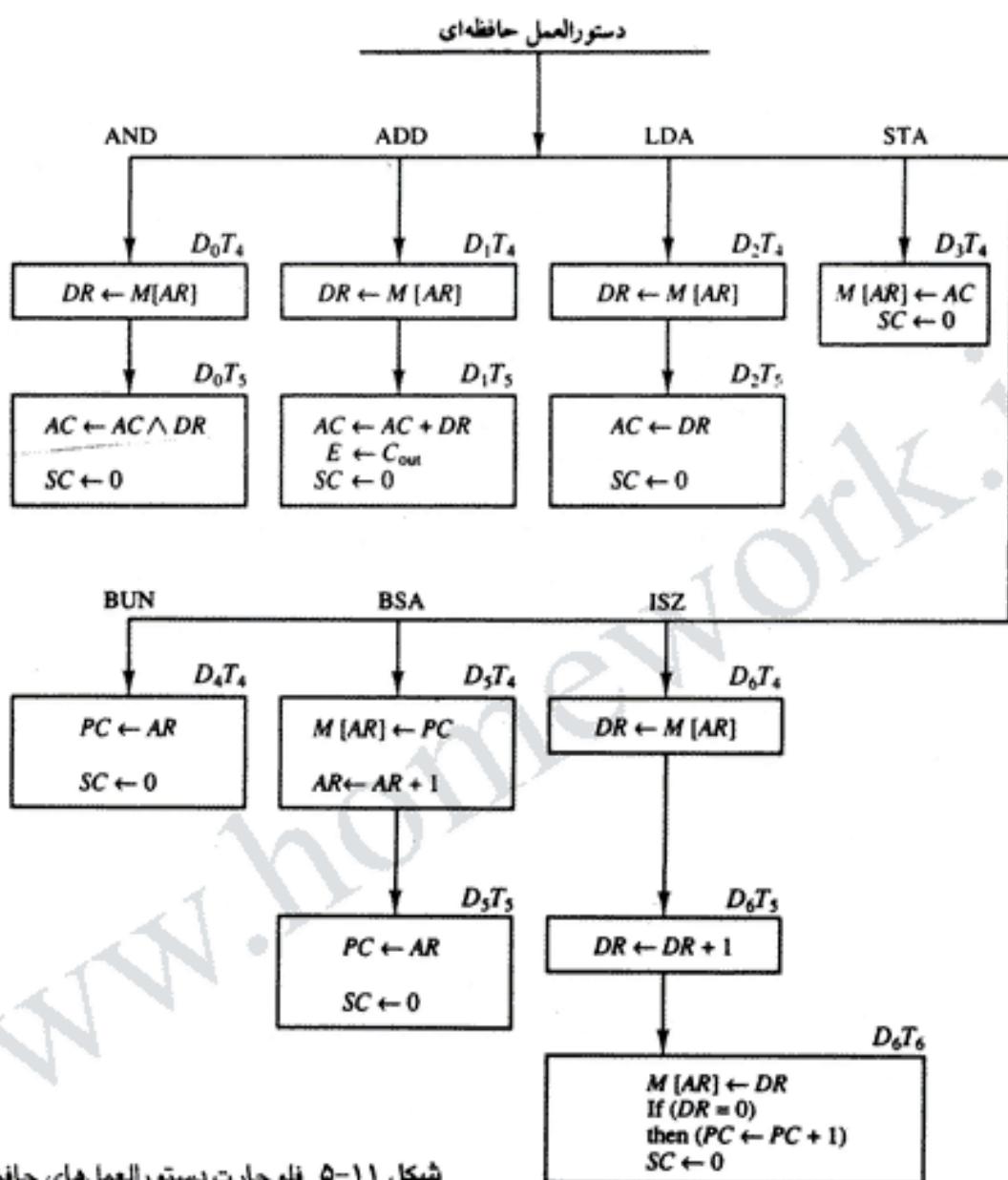
### فلوچارت کنترل

فلوچارت نمایش اجرای ریز عمل‌های مربوط به هفت دستور العمل ارجاع به حافظه در شکل ۵-۱۱ نشان داده شده است. توابع کنترل در بالای هر چهار گوشه آمده است. ریز عمل‌های اجرا شده در  $T_5$  و  $T_6$  به مقدار کد عمل بستگی دارد. این مطلب در فلوچارت توسط شش مسیر مختلف مشخص شده که هر یک از آنها پس از دیگد شدن دستور العمل 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  $T_0$  برای آغاز سیکل دستور العمل بعدی می‌گردد.

توجه کنید که برای اجرای طولانی ترین دستور العمل (ISZ) به هفت سیگنال زمانبندی نیاز داریم. لذا کامپیوتر را می‌توان با یک شمارنده سه بیتی SC طراحی کرد. دلیل بکارگیری چهار بیت در SC این است که سیگنال‌های زمانی لازم برای سایر دستورات در بخش تعمیرات میسر باشد.

### ۵-۷ ورودی - خروجی و وقفه<sup>۲</sup>

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 وسائل خارجی متصل به ورودی گرفته شوند. نتایج محاسبات هم باید از طریق نوعی وسیله خروجی به کاربر انتقال یابد. کامپیوترهای تجاری مجهز به انواع وسائل ورودی و خروجی هستند. برای نمایشی از اساسی ترین ملزمات ارتباطی ورودی و



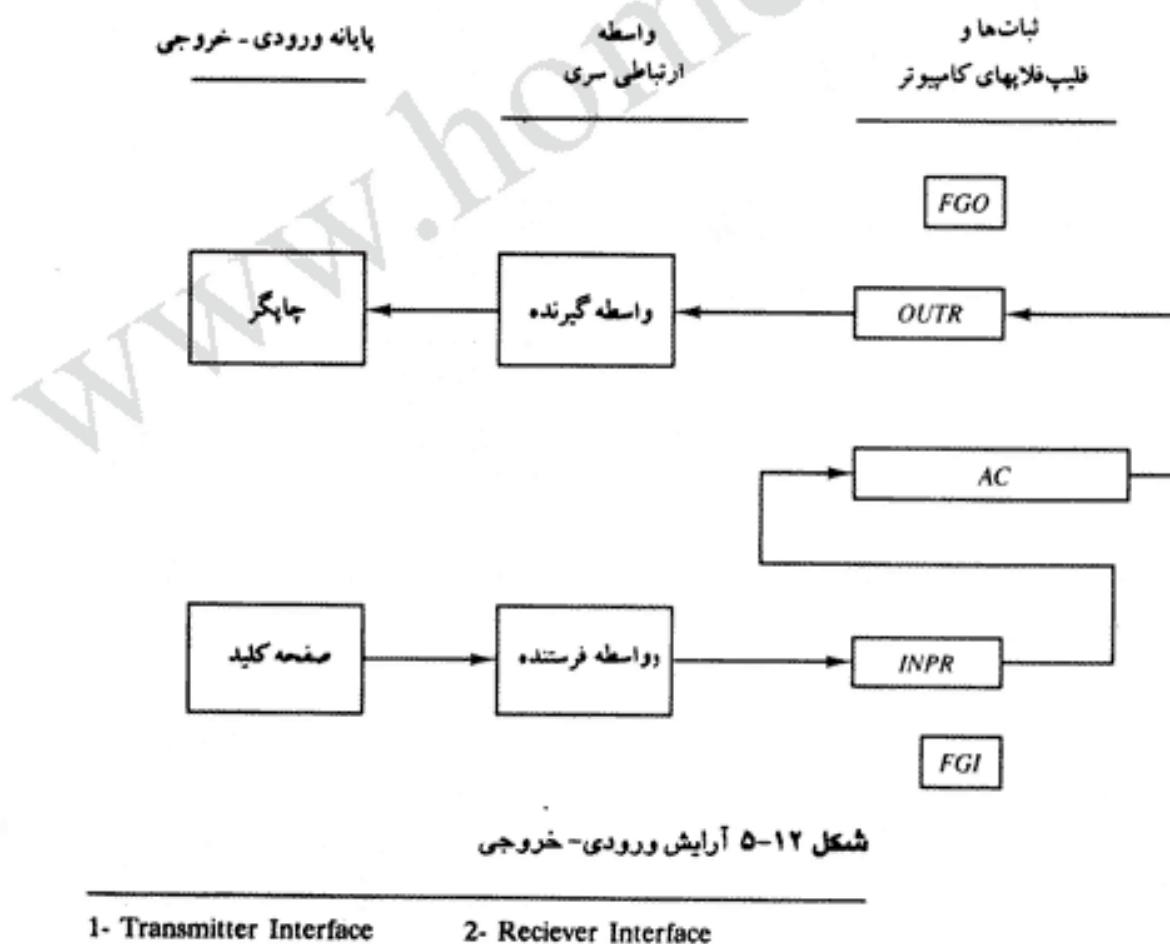
خروجی، ما واحد پایانه‌ای را با صفحه کلید و چاپگر نشان خواهیم داد. سازمان ورودی - خروجی در فصل ۱۱ بحث خواهد شد.

### آرایش ورودی - خروجی

پایانه، اطلاعات سری را ارسال و دریافت می‌کند. هر یک از کمبیت‌های اطلاعاتی حاوی یک کد الفباء عددی هشت بیتی است. اطلاعات سری وارد از صفحه کلید بداخل INPR شیفت داده می‌شود. اطلاعات سری چاپگر در OUTR ذخیره می‌گردد. این دو ثبات با یک مدار واسط وسیله جانبی بطور سری و با

ارتباط موازی دارد. آرایش ورودی - خروجی در شکل ۵-۱۲ دیده می‌شود. مدار واسطه وسیله AC ارسال کننده<sup>۱</sup> (فرستنده) اطلاعات سری را از صفحه کلید دریافت و به INPR می‌فرستد. مدار واسطه دریافت کننده<sup>۲</sup> (دریافت و به چاپگر ارسال می‌دارد. عملکرد مدار واسط در بخش ۱۱-۳ تشریح شده است.

ثبات INPR از هشت بیت تشکیل و اطلاعات الفبا عددی را در خود نگه می‌دارد. پرچم یک بیتی FGI یک فلیپ فلاپ کنترل است و این پرچم هر وقت اطلاعات جدیدی دروسیله ورودی موجود باشد برابر ۱ می‌شود و وقتی کامپیووتر این اطلاعات را پذیرفت برابر ۰ می‌گردد. این پرچم برای همزمان کردن سرعت جریان اطلاعات بین وسیله ورودی و کامپیووتر لازم است. فرآیند انتقال اطلاعات بقرار زیر است. در ابتدا به پرچم ورودی FGI مقدار ۰ داده می‌شود. وقتی کلیدی روی صفحه کلید فشار داده می‌شود، یک کد الفبا عددی هشت بیتی بطور سری به INPR شیفت پیدا کرده و پرچم FGI برابر ۱ می‌گردد. مادامی که این پرچم ۱ است محتوای INPR قابل تغییر بواسطه کلید دیگر نیست. کامپیووتر بیت پرچم را چک می‌نماید، اگر پرچم ۱ باشد، اطلاعات از INPR بطور موازی به AC منتقل و FGI پاک



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. کامپیوترو 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. کامپیوترو 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 دستور العمل خاص مورد نظر هستند. توابع کنترل و ریز عمل های ورودی - خروجی در جدول 5-5 لیست شده اند. اجرای این دستورات بالغه 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 آم IR با  $B_i$  تمام توابع کنترل را می توان با  $PB_i$  نشان داد که در آن 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. دستور العمل 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 6-8 آمده است). دو دستور آخر یک قلیپ فلاپ IEN وقفه را

جدول 5-5 دستور العمل های ورودی - خروجی

(مشترک در همه دستور العمل های ورودی - خروجی)  
[بیشتر در (0-11) IR که دستور العمل را مشخص می کند]

INP	$p: pB_{11}$	$SC \leftarrow 0$ $AC(0-7) \leftarrow INPR, FGI \leftarrow 0$	پاک کردن SC دریافت کاراکتر ورودی
OUT	$p: pB_{10}$	$OUTR \leftarrow AC(0-7), FGO \leftarrow 0$	ارسال کاراکتر خروجی
SKI	$p: pB_9$	If ( $FGI = 1$ ) then ( $PC \leftarrow PC + 1$ )	گذار بر اساس پرچم ورودی
SKO	$p: pB_8$	If ( $FGO = 1$ ) then ( $PC \leftarrow PC + 1$ )	گذار بر اساس پرچم خروجی
ION	$p: pB_7$	$IEN \leftarrow 1$	روشن کردن فعال ساز وقفه
IOF	$p: pB_6$	$IEN \leftarrow 0$	خاموش کردن فعال ساز وقفه

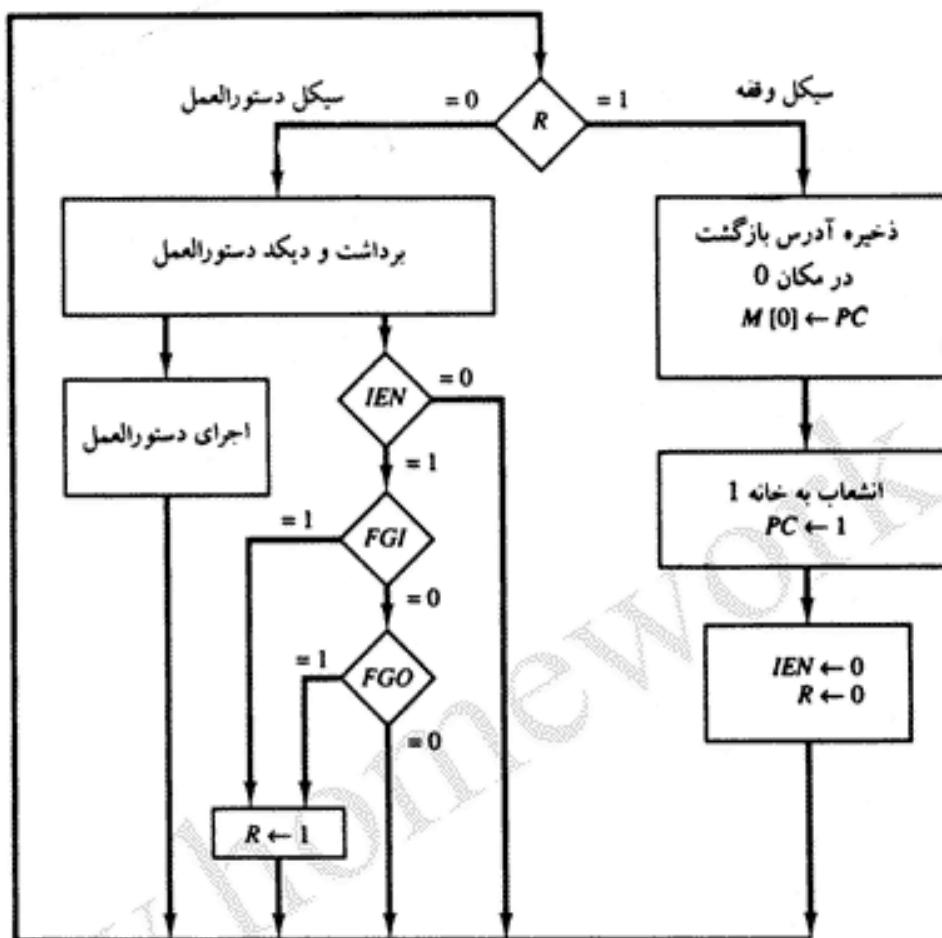
پاک و یا برقرار<sup>۱</sup> می نمایند (می نشاند). هدف از IEN به همراه بحث مربوط به وقفه بیان شده است.

### وقفه بر قامه

فرآیند ارتباطی بحث فوق بنام انتقال تحت کنترل برنامه خوانده می شود. کامپیوتر بطور مداوم بیت پرچم را چک کرده و وقتی آن را ۱ بیابد انتقال را آغاز می کند. بدلیل تفاوت سرعت جریان اطلاعات بین کامپیوتر و سیله ورودی - خروجی کارآیی این نوع انتقال پائین است. برای درک این مطلب، کامپیوتری را در نظر بگیرید که می تواند سیکل دستور را در ۱۴۵ نجات دهد. فرض کنید که سیله ورودی - خروجی می تواند اطلاعات را حداکثر با میزان ۱۰ کاراکتر در ثانیه منتقل نماید. این سرعت معادل با ۱۰۰,۰۰۰ میکروثانیه برای هر کاراکتر می باشد. وقتی که کامپیوتر بیت پرچم را چک می کند و تصمیم می گیرد اطلاعات را انتقال ندهد دو دستور العمل اجرا می شود. این بدان معنی است که در سرعت ماکزیمم، کامپیوتر بین هر دو انتقال ۵۰۰۰۰ بار پرچم را چک خواهد کرد. کامپیوتر مدامی که در حال چک کردن پرچم هاست بعوض انجام کار مفیدی وقت را تلف می کند.

روشی دیگر این است که سیله 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 ۱ شد کامپیوتر از پیش روی در برنامه جاری بازداشت شده و از ۱ 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 ۰ یا ۱ کرد. وقتی IEN پاک شود (با دستور IOF) پرچم ها قادر به ایجاد وقفه در کامپیوتر نیستند. وقتی IEN برقرار شود (۱ 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 ۵-۱۳ قابل تشریح است. یک فلیپ فلاپ R در کامپیوتر منظور شده است. وقتی  $R=0$  باشد کامپیوتر وارد سیکل دستور العمل می گردد. به موازات سیکل اجرای دستور العمل، IEN توسط کنترل چک می شود. ۰ بودن این فلیپ فلاپ بمعنی این است که برنامه نویس مایل به استفاده از وقفه نیست. لذا کنترل با دستور بعدی بکار اダメه می دهد. اگر  $IEN=1$  باشد کنترل بیت های پرچم را چک می نماید. اگر هر دو پرچم ۰ باشند نه ورودی و نه خروجی آماده انتقال اطلاعات نیستند. در این حالت کنترل با دستور بعدی به کار اダメه می دهد. اگر صمن  $IEN=1$  هر یک از دو پرچم ورودی یا خروجی ۱ شوند  $R=1$  می گردد. در انتهای فاز اجرای دستور، کنترل R را چک می نماید و چنانچه مقدار آن برابر ۱ 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

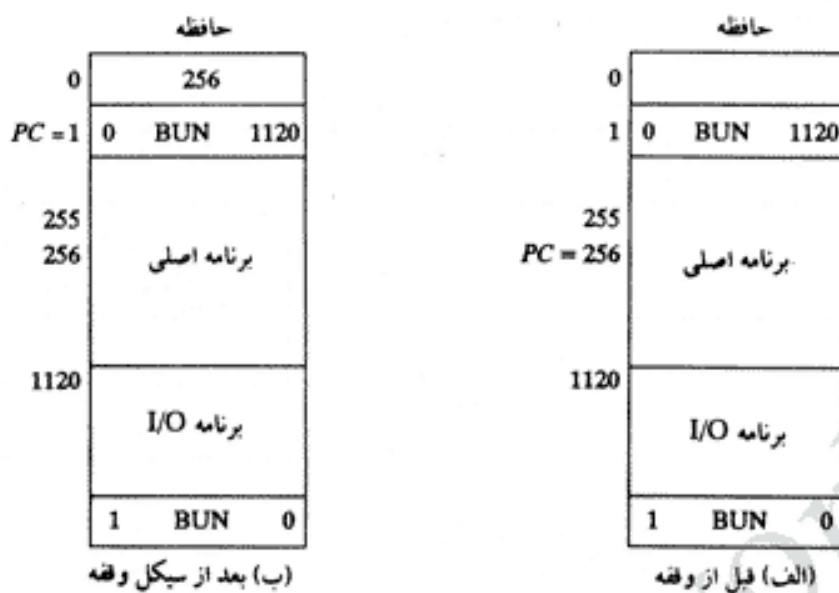


شکل ۵-۱۲ فلوچارت سیکل وقه

PC در یک مکان خاص ذخیره می‌شود تا یهندگام 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 می‌کنیم. کنترل سپس آدرس 1 را در PC وارد کرده و IEN و R را پاک می‌کنند در نتیجه هیچ وقفه‌ای، مادامی که تقاضای وقه قبلی سرویس نشده باشد اتفاق نمی‌افتد.

مثالی که نشان دهنده رخدادها در سیکل وقه است در شکل ۵-۱۴ دیده می‌شود. فرض کنید که وقفه‌ای رخ داده و  $R=1$  است و بموازات آن هم کنترل مشغول اجرای دستوری در آدرس 255 می‌باشد. در این زمان، آدرس برگشت در PC برابر با 256 است. برنامه نویس قبل از برنامه‌ای را برای سرویس دادن به ورودی و خروجی در حافظه و در آدرس 1120 قرار داده و یک 1120 BUN را در آدرس 1 قرار داده است. این مطلب در شکل ۵-۱۴ (الف) نشان داده شده است.

وقتی کنترل به  $T_0$  رسیده و  $R$  را برابر 1 بیابد، وارد سیکل وقه می‌گردد. محتویات (256) در



شکل ۱۴-۵ مثالی از سیکل وقه

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 وقه را ادامه دهد) و برنامه به مکانی که وقه در آن رخ داده بود باز می گردد. دستورالعملی که کامپیوتر را به مکان اولیه اش در برنامه اصلی باز می گرداند یک انشعاب غیرمستقیم است که بخش آدرس آن 0 است. این دستور در انتهای برنامه سرویس  $I/O^1$  قرار دارد. پس از اینکه این دستور از حافظه دریافت شده کنترل به فاز غیرمستقیم ( $I=1$ ) رفته و آدرس مؤثر را می خواند. آدرس مؤثر در مکان 0 است و همان آدرس برگشتی است که در لحظه بروز وقه ذخیره شده بود. اجرای دستور غیرمستقیم BUN نشاندن آدرس برگشت در PC است.

#### سیکل وقه

ما حالا آمده ایم تا عبارات انتقال ثبات را برای سیکل وقه بنویسیم. سیکل وقه پس از آخرین فاز اجراء، 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 FGO یا FGI برابر 1 باشد. این وضعیت در هر انتقال پالس ساعتی امکان پذیر است

جز وقتی که سیگنال های  $T_0$ ,  $T_1$  و یا  $T_2$  فعال باشند. شرایط 1 شدن فلیپ فلاپ R با عبارت انتقال ثبات زیر می تواند بیان شود:

$$T'_0 T'_1 T'_2 \text{ (IEN) (FGI+FGO): } R_1 \leftarrow 1$$

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 ۵-۹) سه سیگنال مذکور را با 'R', AND می نماییم در نتیجه فازهای برداشت و دیکد با سه تابع کنترل  $R' T_0$ ,  $R' T_1$ ,  $R' T_2$  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 وقfe خواهد رفت. سیکل وقfe آدرس برگشت (موجود در PC) را در آدرس 0 ذخیره ساخته و سپس به حافظه 1 انشعاب کرده و R و SC را هم 0 می کند. این اعمال با رشته ریز عمل های زیر امکان پذیر است.

$$RT_0: AR \leftarrow 0, TR \leftarrow PC$$

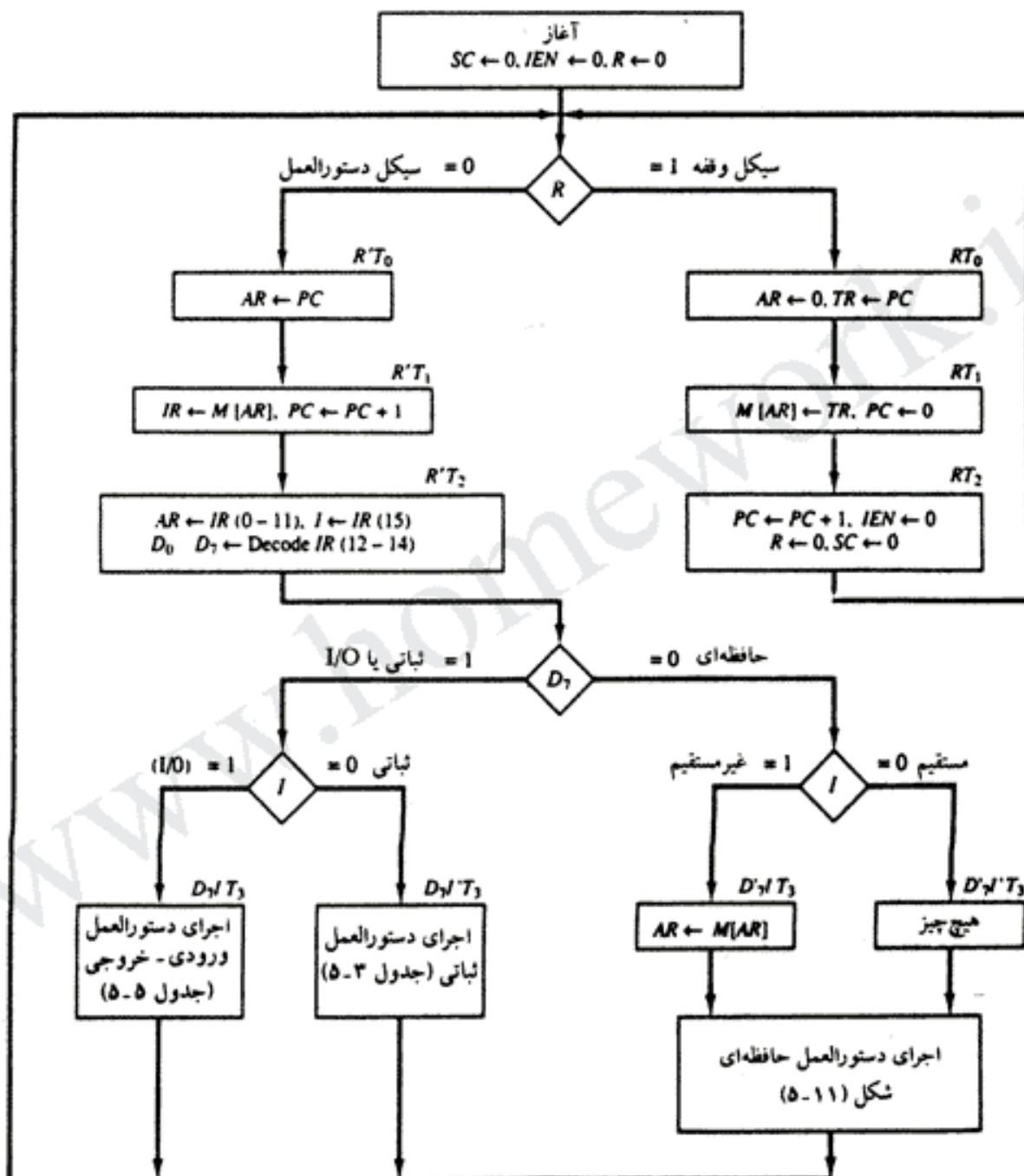
$$RT_1: M[A R] \leftarrow TR, PC \leftarrow 0$$

$$RT_2: PC \leftarrow PC+1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$$

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  $T_0$  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 کامپیووتر

فلوچارت نهایی سیکل دستورالعمل به همراه سیکل وقfe برای کامپیووتر پایه در شکل ۱۵-۵ نشان داده شده است. فلیپ فلاپ R هر لحظه ممکن است در طول فازهای غیرمستقیم<sup>۱</sup> یا اجرا<sup>۲</sup> برابر 1 شود. کنترل پس از  $SC=0$  به  $T_0$  بر می گردد. اگر  $R=1$  باشد کنترل وارد سیکل وقfe می گردد. اگر  $R=0$  باشد کامپیووتر وارد سیکل دستورالعمل می شود. اگر دستور ازنوع ارجاع به حافظه باشد، کامپیووتر در صورت وجود آدرس غیرمستقیم اجرای دستور دیکد شده را طبق شکل ۵-۱۱ ادامه می دهد. اگر دستورالعمل از نوع ارجاع به ثبات باشد، اجرای آن طبق جدول ۵-۳ خواهد بود و بالاخره اگر دستور ازنوع ورودی -



شکل ۵-۱۵ فلوچارت برای اعمال کامپیووتر

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 ۵-۱۱ و ۵-۱۵ و جداول ۵-۳ و ۵-۵ تکمیل گردیده است.

توابع کنترل و ریز اعمال برای کل کامپیوتر در جدول ۵-۶ 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 ۴۰۹۶ کلمه ۱۶ بیتی

۲- ۹ ثبات: SC, INPR, OUTR, TR, JR, AC, DR, PC, AR و

۳- ۷ فلیپ فلاپ: I, S, E, FGI, JEN, R, E و

۴- دو دیکدر: یک دیکدر عملیاتی  $3 \times 8$  و یک دیکدر زمانی  $16 \times 4$

۵- یک گذرگاه مشترک ۱۶ بیتی

۶- کنترل منطقی گیتی

۷- جمع کننده و مدار منطقی متصل به ورودی AC

واحد حافظه قطعه استانداردی است و بصورت آماده قابل تهیه از منابع تجاری است. ثبات ها از نوع شکل ۲-۱۱ و مشابه آی سی 74163 می باشند. فلیپ فلاپها می توانند از نوع D یا JK مطابق بخش ۱-۶ باشند. دو دیکدر نیز استاندارد و مشابه آنچه در بخش (۲-۲) گفته شد می باشند. سیستم گذرگاه مشترک می تواند با شانزده مولتی پلکسor  $8 \times 1$  با آرایش مشابه با شکل ۴-۳ 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

دیاگرام مدار کنترل در شکل ۵-۶ دیده می شود. ورودی های این مدار شامل دو دیکدر، فلیپ فلاپ I و بیت های ۰ تا ۱۱ ثبات IR می باشند. سایر ورودی ها به کنترل عبارتند از بیت های ۰ تا ۱۵ ثبات AC

## جدول ۵-۶ توابع کنترل و اعمال جزئی کامپیوتر پایه

برداشت	$R'T_0: AR \leftarrow PC$
دیکد	$R'T_1: IR \leftarrow M[AR], PC \leftarrow PC + 1$
غیرمستقیم	$D_0, \dots, D_7 \leftarrow \text{Decode } IR(12-14),$ $AR \leftarrow IR(0-11), I \leftarrow IR(15)$
وقت:	$D_5IT_5: AR \leftarrow M[AR]$
$T_0T_1T_2(IEN)(FGI + FGO):$	$R \leftarrow 1$
	$RT_0: AR \leftarrow 0, TR \leftarrow PC$
	$RT_1: M[AR] \leftarrow TR, PC \leftarrow 0$
	$RT_2: PC \leftarrow PC + 1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$
حافظه ای:	
AND	$D_0T_0: DR \leftarrow M[AR]$
	$D_0T_1: AC \leftarrow AC \wedge DR, SC \leftarrow 0$
ADD	$D_1T_0: DR \leftarrow M[AR]$
	$D_1T_1: AC \leftarrow AC + DR, E \leftarrow C_{\text{out}}, SC \leftarrow 0$
LDA	$D_2T_0: DR \leftarrow M[AR]$
	$D_2T_1: AC \leftarrow DR, SC \leftarrow 0$
STA	$D_3T_0: M[AR] \leftarrow AC, SC \leftarrow 0$
BUN	$D_4T_0: PC \leftarrow AR, SC \leftarrow 0$
BSA	$D_4T_1: M[AR] \leftarrow PC, AR \leftarrow AR + 1$
	$D_5T_0: PC \leftarrow AR, SC \leftarrow 0$
ISZ	$D_6T_0: DR \leftarrow M[AR]$
	$D_6T_1: DR \leftarrow DR + 1$
	$D_7T_0: M[AR] \leftarrow DR, \text{ if } (DR = 0) \text{ then } (PC \leftarrow PC + 1), SC \leftarrow 0$
پایانی:	
	$D_7IT_7 = r$ (مشترک در همه دستورالعمل های پایانی)
	$IR(i) = B_i (i = 0, 1, 2, \dots, 11)$
	$r: SC \leftarrow 0$
CLA	$rB_{01}: AC \leftarrow 0$
CLE	$rB_{02}: E \leftarrow 0$
CMA	$rB_{03}: AC \leftarrow \bar{AC}$
CME	$rB_{04}: E \leftarrow \bar{E}$
CIR	$rB_{05}: AC \leftarrow \text{shr } AC, AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	$rB_{06}: AC \leftarrow \text{shl } AC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	$rB_{07}: AC \leftarrow AC + 1$
SPA	$rB_{08}: \text{If } (AC(15) = 0) \text{ then } (PC \leftarrow PC + 1)$
SNA	$rB_{09}: \text{If } (AC(15) = 1) \text{ then } (PC \leftarrow PC + 1)$
SZA	$rB_{10}: \text{If } (AC = 0) \text{ then } PC \leftarrow PC + 1$
SZE	$rB_{11}: \text{If } (E = 0) \text{ then } (PC \leftarrow PC + 1)$
HLT	$rB_{12}: S \leftarrow 0$
ورودی - خروجی:	
	$D_7IT_7 = p$ (مشترک در همه دستورالعمل های ورودی خروجی)
	$IR(i) = B_i (i = 6, 7, 8, 9, 10, 11)$
	$p: SC \leftarrow 0$
INP	$pB_{13}: AC(0-7) \leftarrow INPR, FGI \leftarrow 0$
OUT	$pB_{14}: OUTR \leftarrow AC(0-7), FGO \leftarrow 0$
SKI	$pB_{15}: \text{If } (FGI = 1) \text{ then } (PC \leftarrow PC + 1)$
SKO	$pB_{16}: \text{If } (FGO = 1) \text{ then } (PC \leftarrow PC + 1)$
ION	$pB_{17}: IEN \leftarrow 1$
IOF	$pB_{18}: IEN \leftarrow 0$

برای کنترل  $AC=0$  و کنترل بیت علامت (AC15)؛ 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 نوشتן
- ۳- سیگنال های نشاندن<sup>۱</sup>، پاک کردن<sup>۲</sup> و مکمل سازی فلیپ فلاپها
- ۴- سیگنال S<sub>1</sub>, S<sub>2</sub> و S<sub>0</sub> 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.

$$\begin{aligned} R'T_0 : & AR \leftarrow PC \\ R'T_2 : & AR \leftarrow IR (0-11) \\ D'IT_3 : & AR \leftarrow M[AR] \\ RT_0 : & AR \leftarrow 0 \\ D_5T_4 : & AR \leftarrow AR+1 \end{aligned}$$

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.

$$LD(AR) = R'T_0 + R'T_2 + D'IT_3$$

$$CLR(AR) = RT_0$$

$$INR(AR) = D_5 T_4$$

که در آنها LD(AR) ورودی بارکردن AR (AR) ورودی پاک کردن AR و INR(AR) ورودی افزایش AR است. مدار منطقی کنترل گیتی AR در شکل ۵-۱۶ دیده می‌شود. بطريق مشابه می‌توان مدار کنترل سایر ثبات‌ها و نیز ورودی‌های کنترل حافظه را نیز بدست آورد. گیت‌های منطقی مربوط به ورودی خواندن با جستجو در جدول ۵-۶ برای یافتن عبارات خواندن نیز امکان پذیر است. نشانه عمل خواندن با نماد  $M[AR] \leftarrow$  قابل تشخیص است.

$$\text{Read: } R'T_1 + D'_7IT_3 + (D_0 + D_1 + D_2 + D_6) T_4$$

خروچی گیت‌های منطقی که عبارت فوق را پیاده‌سازی می‌کنند باید به ورودی خواندن حافظه متصل گردد.

#### کنترل فلیپ فلاپ های منفرد

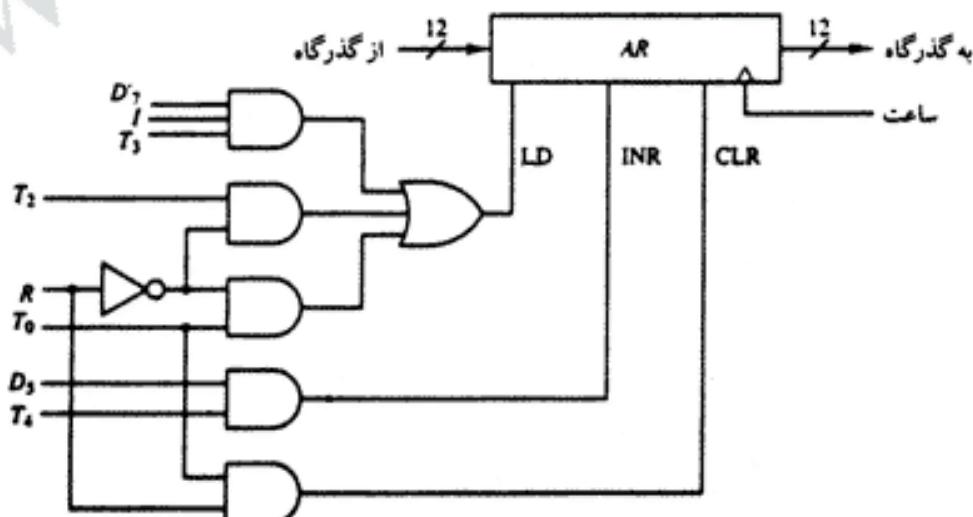
گیت‌های کنترل هفت فلیپ فلاپ بروش مشابهی تعیین می‌شود. عنوان مثال جدول ۵-۶ نشان می‌دهد که IEN توسط دو دستور العمل ممکن است تغییر بابد.

$$pB_7: IEN \leftarrow 1$$

$$pB_6: IEN \leftarrow 0$$

که در آن  $p = D_7IT_3$  و  $B_7 = B_6$  بیت‌های ۷ و ۶ از ثبات IR هستند. بعلاوه در انتهای سیکل وقفه هم IEN، پاک می‌شود.

$$RT_2: IEN \leftarrow 0$$



شکل ۵-۱۶ گیت‌های کنترل AR

چنانچه یک فلیپ فلاپ JK برای IEN بکار رود، مدار کنترل منطقی مطابق شکل ۱۷-۵ خواهد بود.

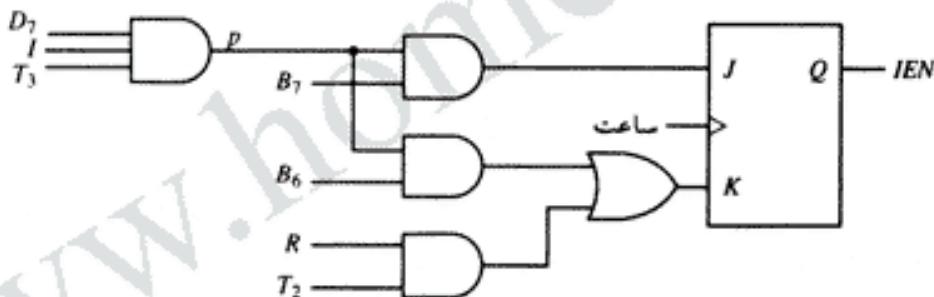
### کنترل گذرگاه مشترک

گذرگاه ۱۶ بیتی شکل ۴-۵ توسط ورودی‌ها انتخاب  $S_2$  و  $S_1$  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. یعنوان مثال وقتی  $x_1 = 1$  است مقدار  $S_2S_1S_0$  باید ۰۰۱ باشد و خروجی AR برای گذرگاه انتخاب می‌شود. جدول ۵-۷ می‌تواند یعنوان جدول درستی انکدر دودویی در نظر گرفته شود. مکان قرار گرفتن انکدر در ورودی‌های انتخاب گر در شکل ۱۸-۵ دیده می‌شود. توابع بول برای انکدر برابر است با:

$$S_0 = x_1 + x_3 + x_5 + x_7$$

$$S_1 = x_2 + x_3 + x_6 + x_7$$

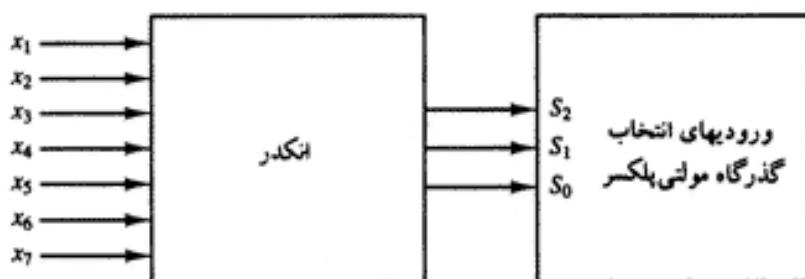
$$S_2 = x_4 + x_5 + x_6 + x_7$$



شکل ۱۷-۵ 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	میکدام		
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	0	1	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 ۵-۱۸ 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.

$$\begin{aligned} D_4 T_4: PC \leftarrow AR \\ D_5 T_5: PC \leftarrow AR \end{aligned}$$

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 قبلًا برای "خواندن" بدست آمد خواهد بود.

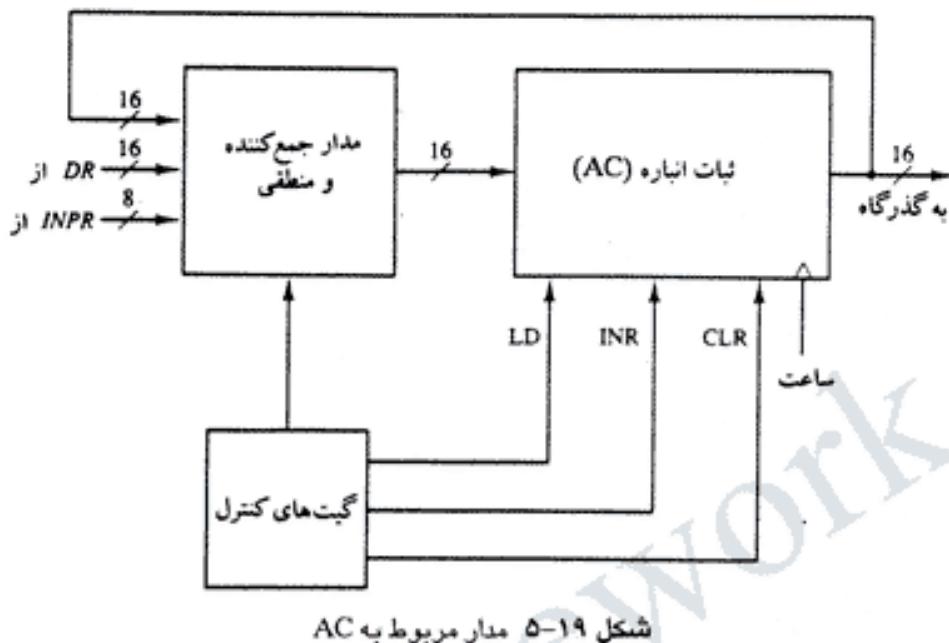
$$x_7 = R'T_1 + D'7IT_3 + (D_0 + D_1 + D_2 + D_6) T_4$$

بهروشی مشابه می‌توانیم مدار گیتی را برای سایر ثبات‌ها بدست آوریم

#### ۵-۱۰ طراحی مدار منطقی انباره<sup>۱</sup>

مدارهای مربوط به ثبات AC در شکل ۵-۱۹ دیده می‌شود. جمع کننده و مدار منطقی دارای سه مجموعه ورودی است. یک مجموعه ۱۶ ورودی از خروجی AC می‌آید. مجموعه ۱۶ ورودی دیگر از INPR DR می‌رسد. سومین مجموعه هشت ورودی از ثبات ورودی INPR گرفته شده است. خروجی جمع کننده و مدار منطقی ورودی‌های داده را برای ثبات AC فراهم می‌آورند. بعلاوه مدارهای منطقی لازم

۱- Accumulator



شکل ۵-۱۹ 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.

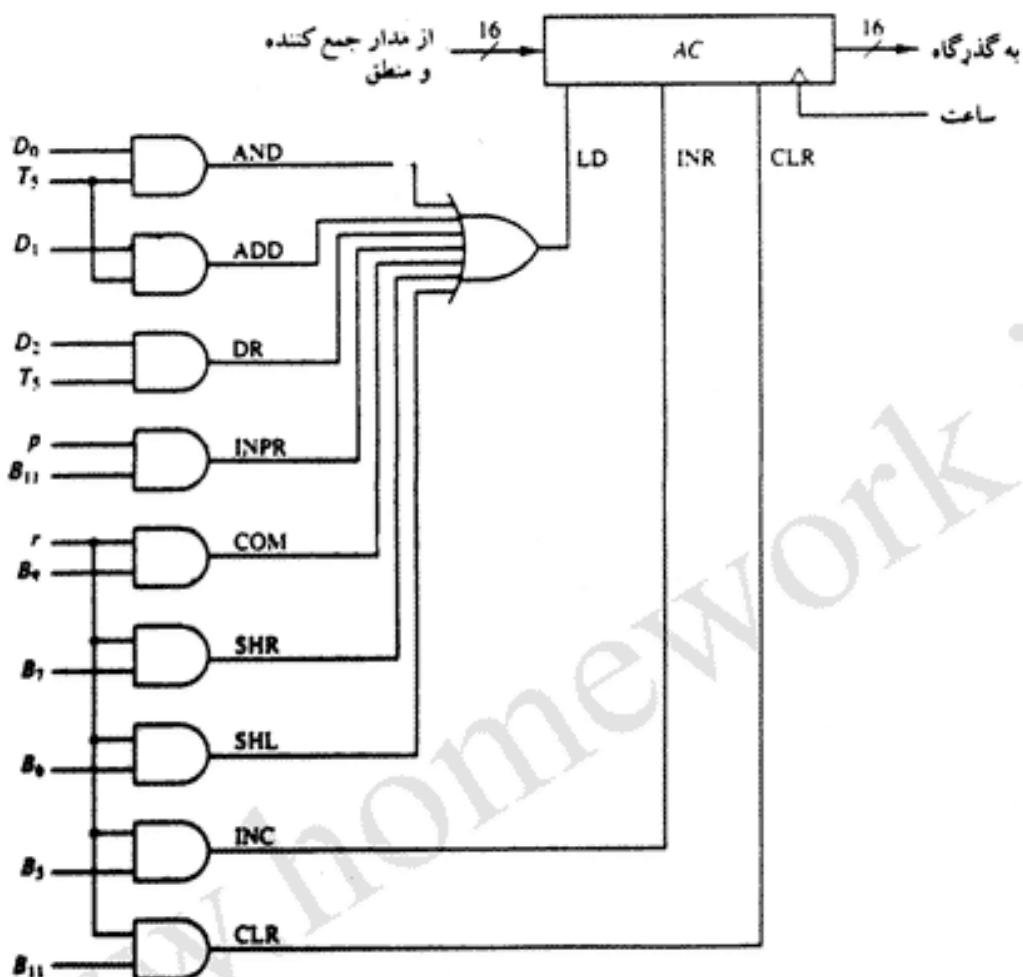
برای طراحی مدار منطقی مربوط به AC، باید به عبارات انتقال ثبات در جدول ۵-۶ مراجعه و تمام عباراتی که محتویات AC را تغییر می‌دهند استخراج شوند.

$D_0T_5:$	$AC \leftarrow AC \wedge DR$	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
$rB_9:$	$AC \leftarrow \bar{AC}$	شم
$rB_7:$	$AC \leftarrow shr AC, AC(15) \leftarrow E$	شیفت به راست
$rB_6:$	$AC \leftarrow shl AC, AC(0) \leftarrow E$	شیفت به چپ
$rB_{13}:$	$AC \leftarrow 0$	پاک
$rB_5:$	$AC \leftarrow AC + 1$	افزایش

از این لیست می‌توانیم مدار منطقی جمع کننده و گیت‌های منطقی آنرا بدست آوریم

### کنترل ثبات AC

ساختار گیتی کنترل کننده ورودی‌های AC یعنی LD، INR، CLR در شکل ۵-۲۰ نشان داده شده‌اند. پیکربندی گیت‌ها از توابع کنترل در لیست فوق بدست آمده‌اند. تابع کنترل برای ریز عمل‌های پاک



شکل ۵-۲۰ ساختار گیت‌های مربوط به کنترل در ورودی‌های LD، INR و CLR از AC

کردن<sup>۱</sup> ،  $B_{11} = IR(11)$  و  $B_7 = D_7T_3$  است که در آن  $T_3 = D_7I'T_1$  باشد. خروجی گیت AND که این تابع کنترل را تولید می‌کند به ورودی CLR ثبات متصل است. بطور مشابه خروجی گیت JNPR که ریز عمل افزایش<sup>۲</sup> را پیاده سازی می‌نماید به ورودی INR ثبات 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 سیمبلیک 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

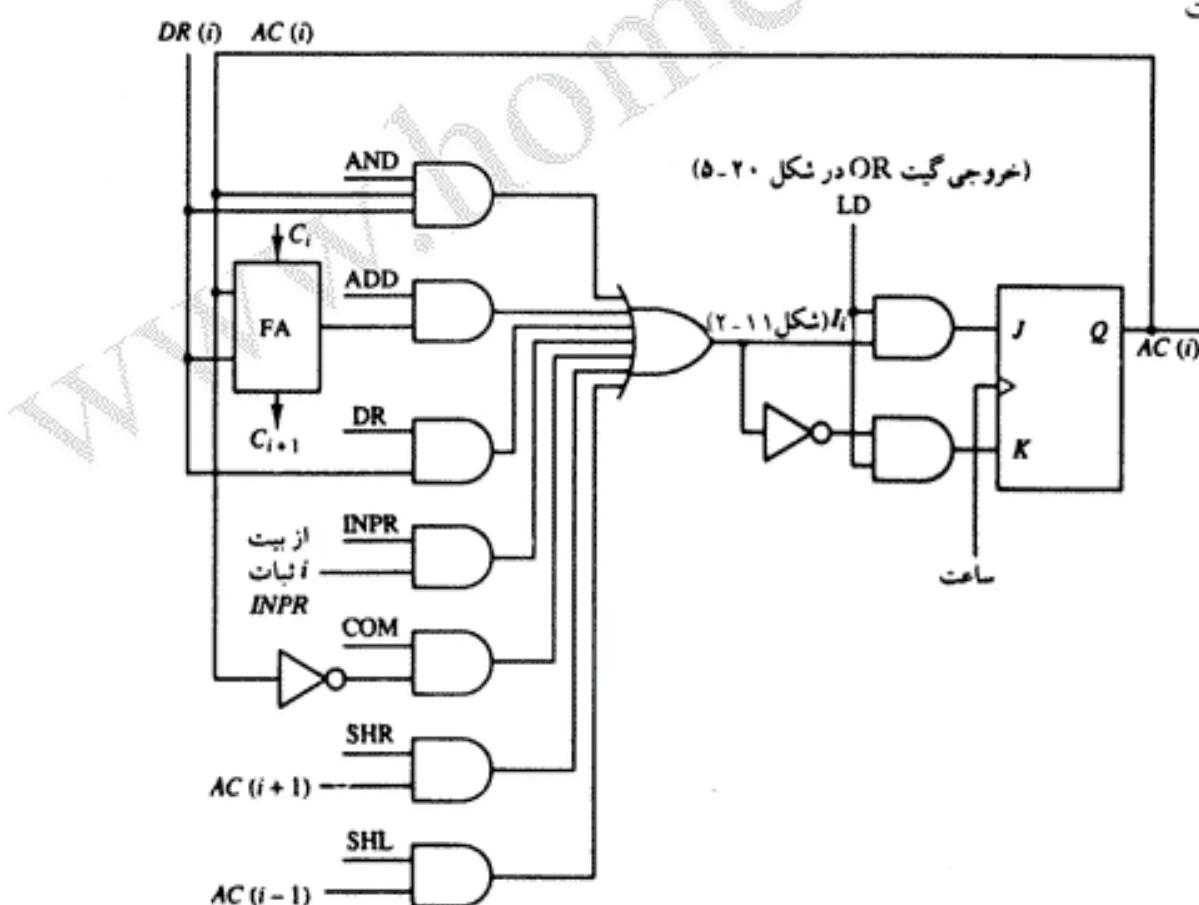
1- Clear

2- Increment

است. ساختمان درونی ثبات مطابق شکل ۲-۱۱ است. با بازگشت به این شکل می بینیم که هر گیت AND دارای یک فلیپ فلاب JK دو گیت OR و دو گیت LD است. ورودی بار (LD) به ورودی گیت های AND وصل است. شکل ۵-۲۱ 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 مطابق شکل ۵-۲۱ ساخته شده است. ورودی این گیت ها با نام هایی سمبولیکی از خروجی (FA) گیت ها با همان نام در شکل ۵-۲۰ آمده اند. بعنوان مثال، ورودی علامت زده با ADD در شکل ۵-۲۱ به خروجی مشابه با همان نام در شکل ۵-۲۰ وصل است.

عمل AND با کردن AND در بیت  $AC(i)$  با بیت متناظر شرکت داده  $DR(i)$  صورت می گیرد. عمل ADD هم با جمع کننده ای مشابه با شکل ۴-۶ انجام می شود. در طبقه جمع کننده از یک تمام جمع کننده با ورودی و خروجی نقلی استفاده شده است. انتقال از AC به INPR 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 ۵-۲۱ یک طبقه از مدار جمع کننده و منطقی

انتقال را از بیت (i) AC(i+1) به (i) AC و شیفت به چپ انتقال را از بیت (i-1) AC(i-1) به (i) AC انجام می‌دهد. مدار کامل جمع‌کننده و مدار منطقی از 16 بخش که بهم متصلند تشکیل شده‌اند.

### مسائل

۵-۱ یک کامپیوتر از حافظه‌ای با 256k کلمه 32 بیتی استفاده می‌کند. یک دستورالعمل دودویی در یک کلمه از حافظه ذخیره شده است. دستورالعمل چهار بخش 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 ورودی‌های داده و آدرس حافظه چند بیت وجود دارد.

۵-۲ اختلاف بین دستور با 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 ورودی‌های کنترل زیر در سیستم گذرگاه شکل ۵-۴ 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	جمع‌کننده	مجموع
(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 ثبات‌های زیر قرار است در سیستم شکل ۵-۴ اجرا شوند. برای هر انتقال: (۱) مقدار دودویی که باید به ورودی‌های انتخاب گذرگاه  $S_2$ ,  $S_1$  و  $S_0$  اعمال شوند را معین کنید؛ (۲) ثباتی که کنترل آن باید فعال شود (اگر وجود دارد)؛ (۳) عمل نوشتن یا خواندن حافظه (اگر لازم است)؛ و (۴) عمل در جمع‌کننده و مدار منطقی (اگر وجود دارد).

$$IR \leftarrow M[AR] \quad (ب)$$

$$AC \leftarrow DR, DR \leftarrow AC \quad (د)$$

$$AR \leftarrow PC \quad (الف)$$

$$M[AR] \leftarrow TR \quad (ج)$$

۵-۵ توضیح دهید چرا هیچک از ریز عمل‌های زیر در طول یک پالس ساعت در سیستم شکل ۵-۴ اجرا نمی‌شوند. رشته ریز اعمال لازم برای انجام عملیات را معین کنید

$$IR \leftarrow M[PC] \quad (الف)$$

$$AC \leftarrow AC + TR \quad (ب)$$

ج)  $DR \leftarrow DR + AC$  (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 ۱۶ بیتی، کد معادل چهاربیتی مبنای شانزده را نوشه و بزیان ساده بگویند این دستور چه کاری انجام می‌دهد.

الف)	0001	0000	0010	0100
ب)	1011	0001	0010	0100
ج)	0111	0000	0010	0000

۵-۷ کدام دو دستور برای ۱ 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 ۰ شده باشد بشرط اینکه سیگنال کنترل  $C_7$  فعال باشد.

$$C_7 T_3: SC \leftarrow 0$$

$C_7$  باله پالس مربوط به  $T_1$  فعال می‌شود.

۵-۹ محتویات AC در کامپیوتر پایه عدد مبنای شانزده A937 است و مقدار اولیه E برابر ۱ است.

۵-۱۰ محتویات AC، PC، E، AR، IR و CLA در مبنای ۱۶ پس از اجرای دستور CLA چیست. عمل قبل را ۱۱ بار با هر یک از دستورات العملها تکرار کنید. مقدار اولیه PC را عدد مبنای شانزده ۰۲۱ فرض کنید.

۵-۱۱ محتویات AC در آدرس ۰۲۱ کامپیوتر پایه دارای  $I=0$ ، کد عملیاتی AND و آدرس ۰۸۳ است (تمام ارقام در مبنای شانزده است). کلمه حافظه واقع در آدرس ۰۸۳ 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، IR. مسئله را شش بار دیگر برای دستورات العمل ارجاع به حافظه دیگری تکرار کنید.

۵-۱۲ 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 ۷FF در نظر بگیرید. محتویات حافظه در آدرس ۷FF EA9F برابر است. محتویات حافظه در آدرس ۰C35 A9F هم 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 زمان‌بندی تهیه کنید. محتویات هر ثبات را پس از لبه مثبت هر پالس ساعت نشان دهید.

۵-۱۳ محتویات PC در کامپیوتر پایه ۳AF است (تمام اعداد در مبنای شانزده). محتویات AC هم ۷BC3 است. محتویات حافظه آدرس ۳AF برابر با ۹32E می‌باشد. محتویات حافظه در آدرس ۳2E برابر ۰9AC و در آدرس ۸B9F هم ۹AC 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) محتويات ثبات های PC, AR, DR 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 ۵-۴ با جدول زیر تعویض شوند. EA آدرس موثر واقع در AR در T<sub>4</sub> است. فرض کنید که جمع کننده و مدار منطقی شکل ۵-۴ می تواند عمل XOR را انجام دهد  $AC \leftarrow AC \oplus DR$ . بعلاوه فرض کنید که جمع کننده و مدار منطقی نمی توانند مستقیماً تفريقي را انجام دهند. تفريقي باید بكمک مکمل 2 انجام شود. رشته عبارات انتقال ثبات لازم برای اجرای هر دستور لیست شده را از T<sub>4</sub> به بعد مشخص کنید. وقت کنید که هیچ تغییری در AC رخ نمی دهد مگر اينکه دستورالعمل تغییری را در آن معین کند. شما می توانید با استفاده از TR محتويات AC را موقتاً ذخیره و یا محتويات DR و AC را باهم عوض کنید.

توضیح	نمایش سمبولیک	کد عمل	سلم
انحصاری با OR	$AC \leftarrow AC \oplus M[EA]$	000	XOR
جمع AC با حافظه	$M[EA] \leftarrow M[EA] + AC$	001	ADM
تفريقي 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 000 = S<sub>2</sub>S<sub>1</sub>S<sub>0</sub>

انتخاب نمایید.

۲- ISZ را با دستوری که یک عدد را در CTR بار کند عوض کنید.

LDC Address      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 نمائید. مزیت این تغییر را بیان کنید.

۵-۱۵ واحد حافظه کامپیوتر پایه در شکل ۵-۱۳ را با یک حافظه 16 × 65536 عوض کنید. این

حافظه آدرس 16 بیتی نیاز دارد. قالب دستورالعمل ارجاع به حافظه در شکل (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 بدنیال 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 (هر یک شانزده بیت) و AC و DR و IR (هر یک هشت بیت) است. یک دستور ارجاع به حافظه مشکل 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 ۵-۳ 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 های 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 فوار گیرد؟

ب - دو دستور آخر برنامه خروجی چیست؟

5-19 عبارات انتقال ثبات برای ثبات R و حافظه در یک کامپیوتر مطابق زیر است ( $X_i$ ها 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 ۲-۱۲ 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 کرده اید، و ورودی های خواندن و نوشتن حافظه را انتخاب می کنند.

۵-۲۰ رشته اعمالی که باید توسط فلیپ فلاپ F انجام شوند با عبارت انتقال ثبات داده شده اند.

XT<sub>3</sub>:  $F \leftarrow 1$ 

۱ در F نشانده شود

yT<sub>1</sub>:  $F \leftarrow 0$ 

با ۰ پاک شود

ZT<sub>2</sub>:  $F \leftarrow \bar{F}$ 

مکمل

WT<sub>5</sub>:  $F \leftarrow G$ 

مقدار G را به F انتقال بده

در غیرابنصورت F نباید تغییر یابد. دیاگرام منطقی مربوط به اتصالات گیت ها که توابع کنترل و ورودی های فلیپ فلاپ F تشکیل می دهند را ترسیم نمائید. از یک فلیپ فلاپ JK استفاده کرده و تعداد گیت هارا حداقل کنید.

۵-۲۱ مدار کنترل گیتی مربوط به شمارنده برنامه PC را در کامپیوتر پایه بدهست آورید.

۵-۲۲ مدار کنترل گیتی برای ورودی خواندن یک حافظه را در کامپیوتر پایه بدهست آورید.

۵-۲۳ مدار کامل منطقی وقفه را در کامپیوتر پایه نشان دهید. از فلیپ فلاپ JK استفاده کرده و گیت ها را به حداقل برسانید.

۵-۲۴ عبارت بولی را برای  $x_2$  (جدول ۵-۷ را ملاحظه کنید) بدهست آورید. نشان دهید که  $x_2$  می تواند با یک گیت AND و یک OR تولید شود.

۵-۲۵ عبارت بولی برای یک مدار گیتی که شمارنده SC را پاک کند بدهست آورید. دیاگرام منطقی آنرا رسم نموده و نشان دهید که چگونه خروجی به ورودی های INR و CLR از شمارنده SC وصل می شود شکل ۵-۶. تعداد گیت ها را حداقل نمائید.



# برنامه نویسی کامپیوتر پایه

- ۶-۱ مقدمه
- ۶-۲ زبان ماشین
- ۶-۳ زبان اسمنبلی
- ۶-۴ اسمنبلر
- ۶-۵ حلقه در برنامه نویسی
- ۶-۶ برنامه نویسی اعمال حسابی و منطقی
- ۶-۷ زیرروالها
- 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 دستور العمل های ماشین، به طور مستقیم یا غیر مستقیم است. دستور العمل های ماشین در داخل کامپیوتر الگویی دودویی را تشکیل می دهند که کار با آن یا در ک آن، اگر غیر ممکن نباشد، مشکل هست. لذا نوشتن برنامه ها با مجموعه کاراکتر های آشناتر 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 های 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 های که بزبان فرترن استاندارد نوشته می شود مستقل از ماشین

است، زیرا اکثر کامپیووترها برنامه مترجمی دارند که فرترن استاندارد را به کد دودویی آن کامپیووتر تبدیل می‌کنند. اما خود برنامه مترجم وابسته به ماشین است زیرا باید برنامه فرترن را به کد دودویی قابل تشخیص بوسیله سخت افزار کامپیووتر خاص مورد استفاده ترجمه کند.

این فصل برخی از مفاهیم ابتدایی برنامه نویسی را معرفی می‌کند و رابطه آنها را با نمایش سخت افزاری دستورالعمل‌ها نشان می‌دهد. اولین بخش، عملکرد و ساختار اساسی برنامه‌ای که بزبان سمبولیک 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 سخت افزار آن در فصل ۵ بررسی شد، در این فصل به منظور تشریح بسیاری از تکنیک‌های مورد استفاده در برنامه‌نویسی یک کامپیووتر، پکار رفته است. به این ترتیب امکان خواهد داشت تا رابطه بین یک برنامه و عملیات سخت افزاری که دستورات را اجرا می‌کنند مورد تحقیق قرار گیرد.

۲۵ دستور پایه کامپیووتر در جدول ۱-۶ تکرار شده‌اند تا مراجعه به آنها بهنگام مثال‌های برنامه‌نویسی تسهیل گردد. به هر دستورالعمل یک سمبول سه حرفی اختصاص یافته است تا نوشتن برنامه سمبولیک 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<sup>۱</sup> (شیوه) سه بیت کد عمل، و دوازده بیت آدرس. اولین رقم شانزده شانزده‌ی یک دستورالعمل حافظه‌ای شامل بیت روش و کد عمل آنست. سه رقم دیگر آدرس را مشخص می‌کنند. در یک دستورالعمل با آدرس غیر مستقیم بیت روش برابر ۱ و اولین رقم شانزده شانزده‌ی بین ۸ تا E می‌باشد. در حالت مستقیم محدوده از ۰ تا ۶ است. هجده دستور دیگر دارای کد عمل شانزده بیتی هستند. کد هر دستورالعمل بصورت یک عدد چهار رقمی شانزده شانزده‌ی لیست شده است. اولین رقم یک دستورالعمل ثباتی همواره ۷ است. اولین رقم یک دستور ورودی- خروجی همیشه F است. سمبول M در ستون توضیح معنی آدرس موثر است. حرف M کلمه‌ای از حافظه را که در آدرس موثر قرار دارد نشان می‌دهد.

## ۶-۲ زیان‌های ماشین

برنامه مجموعه‌ای از دستورالعمل‌ها برای هدایت کامپیووتر در اجرای یک کار داده پردازی مورد نظر است. زیان‌های مختلفی برای نوشتن یک برنامه کامپیووتری وجود دارد. ولی کامپیووتر می‌تواند تنها برنامه‌هایی را اجرا<sup>۲</sup> کند که در داخل آن به شکل دودویی ارائه شده باشند. برنامه‌های نوشته شده به هر زیان دیگر، قبل از اجرا بوسیله کامپیووتر باید به دستورالعمل دودویی ترجمه شوند. برنامه‌های نوشته

1- Mode bit

2- Execute

## جدول ۱-۶ دستورالعمل های کامپیوتر

سبل	کد شانزده شانزدهی	شرح
AND	0 يا 8	AC با M AND
ADD	1 يا 9	جمع AC با M ، نقلی به E
LDA	2 يا A	بارگذرن M از AC
STA	3 يا B	ذخیره M در AC
BUN	4 يا C	انشعاب غیرشرطی به m
BSA	5 يا D	ذخیره آدرس برگشت در m و انشعاب به m+1
ISZ	6 يا E	افزایش M و گذار در صورت صفر بودن نتیجه
CLA	7800	صفر کردن AC
CLE	7400	صفر کردن E
CMA	7200	AC متمم کردن
CME	7100	E متمم کردن
CIR	7080	چرخش به راست E و AC
CIL	7040	چرخش به چپ E و AC
INC	7020	افزایش AC
SPA	7010	گذار در صورت مثبت بودن AC
SNA	7008	گذار در صورت منفی بودن AC
SZA	7004	گذار در صورت صفر بودن AC
SZE	7002	گذار در صورت صفر بودن E
HLT	7001	توقف کامپیوتر
INP	F800	دریافت اطلاعات ورودی و 0 کردن برجم
OUT	F400	ارسال اطلاعات خروجی و 0 کردن برجم
SKI	F200	گذار در صورت 1 بودن برجم ورودی
SKO	F100	گذار در صورت 1 بودن برجم خروجی
ION	F080	فعال کردن وقفه
IOF	F040	غیرفعال کردن وقفه

شده برای یک کامپیوتر می توانند یکی از رسته 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 شانزدهی. این کد معادل ترجمه شده کد دودویی به هشت هشتی یا شانزده شانزدهی است.

۳- کد سمبليک. در اين کد، کاربر از سبل (حروف، اعداد یا کاراکتر های خاص) برای بخش عملیاتی، بخش آدرس، و سایر قسمت های کد دستورالعمل استفاده می کند. هر دستورالعمل سمبليک را می توان به یک دستورالعمل کد شده با دودویی ترجمه کرد. اين ترجمه توسط برنامه خاصی بنام اسبلر<sup>۱</sup> انجام می شود. چون اسبلر سبل را ترجمه می کند، اين نوع برنامه سمبليک، برنامه بزبان اسبلی خوانده می شود.

## 1- Assemblers

۴- زیان های برنامه نویسی سطح بالا. این برنامه ها که بزیان های خاصی نوشته می شوند بخاطر دریافت تأثیر رویه هایی است که به منظور حل مسئله خاصی بکار می روند و نه صرفاً بخاطر تأثیر بر رفتار سخت افزار کامپیوتر. مثالی از یک برنامه بزیان سطح بالا فرتون<sup>۱</sup> است. برنامه بصورت رشته ای از عبارات بر اساس نحوه تفکر فرد بهنگام حل یک مسئله نوشته می شود. با این وجود، هر عبارت باید قبل از اجرا در کامپیوتر به رشته ای از دستورات دودویی تبدیل شود. برنامه ای که یک برنامه دیگر بزیان سطح بالا را به دودویی ترجمه می کند کامپایلر نامیده می شود.

دقیقتر بگوئیم، برنامه زیان ماشین یک برنامه دودویی از رسته ۱ است. بعلت شباهت بین نمایش های دودویی، هشت هشتی و شانزده شانزدهی، رسم بر این است که رسته ۲ عنوان زیان ماشین شناخته شود. بدلیل رابطه یک به یک بین دستور سمبیلیک و معادل دودویی اش، زیان اسمبلی عنوان زیان در سطح ماشین شناخته می شود.

اکنون با بکارگیری کامپیوتر پایه رابطه بین زیان های دودویی و اسمبلی را شرح می دهیم. برنامه لیست شده در جدول ۶-۲ را ملاحظه کنید. اولین ستون، مکان حافظه (به دودویی) را برای هر دستور یا عملوند می دهد. ستون دوم محتوای دودویی این مکان هاست. (مکان، آدرس کلمه حافظه ای است که دستور العمل در آن ذخیره شده است. شناخت تفاوت این آدرس، از بخش آدرس دستور العمل اهمیت دارد) برنامه در بخش مشخصی از حافظه ذخیره شده و سپس بوسیله کامپیوتر و با شروع از آدرس ۱۰ اجرا می شود. سخت افزار کامپیوتر این دستور العمل ها را اجرا کرده و کار مورد نظر را انجام می دهد. معهدها، فردی که به این برنامه نگاه می کند به سختی درخواهد یافت که پس از اجرای برنامه چه رخداد داد. با این حال، سخت افزار کامپیوتر تنها این نوع کدهای دستور العمل را تشخیص می دهد.

نوشتن ۱۶ بیت برای هر دستور العمل خسته کننده است زیرا رقم ها زیادند. ما می خواهیم تعداد ارقام برای هر دستور العمل را با بکاربردن کد هشت هشتی معادل کد دودویی کاهش دهیم. این کار نیاز به شش

جدول ۶-۲ برنامه دودویی برای جمع کردن دو عدد

		کد دستور العمل	خانه حافظه
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	

۱- 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 باید استفاده کنیم. نمایش شانزده باید در کاربرد ساده است؛ اما باید دانست که هر رقم شانزده باید وقتی که برنامه وارد کامپیوتر می‌شود باید تبدیل به معادل چهاربیتی اش شود. مزیت نوشتن برنامه‌های دودویی به معادل هشت بیتی یا شانزده باید از این مثال باید آشکار شده باشد.

برنامه جدول ۶-۴ از نام‌های سمبولیک دستورالعمل‌ها (این دستورالعمل‌ها در جدول ۱-۶ معرفی شدند) بجای معادل دودویی یا شانزده باید استفاده کرده است. بخش آدرس دستورات حافظه‌ای، همانند عملوند‌ها، به همان شکل شانزده باید کنید که در مکان ۰۰۵ مقدار عملوند منفی است زیرا بیت علامت در سمت چپ ترین مکان ۱ است. در نظر گرفتن ستونی برای توضیحات راهی را برای شرح عملکرد دستورات در اختیار می‌گذارد. کار با برنامه‌های سمبولیک آسانتر است، و در نتیجه، بهتر است برنامه‌ها با سمبول‌ها نوشته شوند. این برنامه‌ها قابل تبدیل به کد دودویی معادل برای تولید برنامه دودویی می‌باشند.

حال یک قدم جلوتر می‌رویم و هر آدرس شانزده باید را با یک آدرسی سمبولیک و نیز هر عملوند شانزده باید را با عملوند دهدی 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 باید برای جمع کردن دو عدد**

دستورالعمل	خانه حافظه
000	2004
001	1005
002	3006
003	7001
004	0053
005	FFE9
006	0000

**جدول ۴-۶ برنامه با کدهای عمل سمبولیک**

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 ۰۰۶
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 دهدھی آشنا تر از معادل شانزده شانزدهی شان است.

برنامه جدول ۶-۵ یک برنامه به زبان اسمبلی برای جمع دو عدد است. سمبول 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 سمبولیک (نمادین) هستند. مقدار هریک از سمبول‌ها از سطري که آن سمبول را بصورت عنوان در ستون اول خود دارد مشخص می‌شود. عملوندهای دهدھی با سمبول DEC مشخص می‌گردند. عددها می‌توانند مثبت یا منفی باشند، اما اگر منفی باشند باید به فرم متمم ۲ علامتدار دودویی درآیند. آخرین خط سمبول END را دارد که انتهای برنامه را نشان می‌دهد. سمبول‌های ORG، DEC و END، شبیه دستورالعمل<sup>۱</sup>‌ها خوانده می‌شوند و در بخش بعد تعریف شده‌اند. توجه کنید که قبل از تمام توضیحات علامت معیز (/) آمده است.

برنامه معادل بزیان 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 متشكل 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 سمبول‌های برنامه فرترن را به لیستی از برنامه جدول ۶-۶ که بشکل دودویی است ترجمه می‌کند.

#### جدول ۶-۵ برنامه به زبان اسمبلی برای جمع دو عدد

ORG 0	مبدأ برنامه خانه ۰ است /
LDA A	بارگردان عملوند از /A
ADD B	جمع کردن عملوند در /B
STA C	ذخیره مجموع در /C
HLT	توقف کامپیوترا /
A,	عملوند دهدھی /
DEC 83	عملوند دهدھی /
B,	عملوند دهدھی /
DEC -23	مجموع در خانه C ذخیره می‌شود /
C,	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
```

1- Preudomstruction

### ۶-۳ زبان اسembly

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 اسembly خاص خود است. قواعد نوشتن برنامه‌ها به زبان اسembly بصورت راهنمایها مستندسازی و چاپ می‌شود و معمولاً توسط سازندگان کامپیوتر در دسترس قرار می‌گیرد.

واحد پایه یک برنامه اسembly یک سطر کد است. یک زبان خاص بر اساس مجموعه قواعد خاصی تعریف می‌شود و این قواعد بیان می‌دارند که سمبیل‌ها چگونه می‌توانند استفاده شوند یا با یکدیگر ترکیب شوند تا یک خط از کد را بسازند. حال ما قواعد یک زبان اسembly را برای نوشتن یک برنامه سمبیلیک برای کامپیوتر پایه فرموله می‌کنیم.

#### قواعد زبان

هر خط از برنامه زبان اسembly به صورت سه ستون مرتب شده‌اند که هر یک میدان<sup>۱</sup> نامیده می‌شوند. میدان‌ها اطلاعات زیر را مشخص می‌کنند.

- ۱- میدان عنوان (برچسب) که می‌تواند خالی باشد یا یک آدرس سمبیلیک 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 سمبیلیک 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. سمبیل می‌تواند بدلخواه توسط برنامه‌نویس انتخاب شود. یک آدرس سمبیلیک در میدان عنوان به یک کاما ختم می‌شود تا اسمبیل تشخیص دهد که عنوان است. میدان دستورالعمل در یک برنامه زبان اسembly یکی از اقلام زیر را مشخص می‌نماید.

- ۱- دستورالعمل‌های حافظه‌ای MRI
- ۲- دستورالعمل‌های ثباتی<sup>۲</sup> یا ورودی خروجی از نوع غیر ارجاع به حافظه non - MRI
- ۳- شبه دستورالعمل‌ها با ، و بدون عملوند

یک دستورالعمل حافظه‌ای دو یا سه سمبیل را که با فواصلی از هم جدا شده‌اند استفاده می‌کند. سمبیل اول باید سه حرفی باشد که یکی از کدهای عمل MRI جدول ۶-۱ را تعریف می‌نماید. دومین سمبیل، آدرس سمبیلیک است. سومین سمبیل، که ممکن است وجود داشته 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 سمبول های سه حرفی در لیست جدول ۱-۶ برای دستورات ثباتی و ورودی تشخیص داده می شود.

مثالهای زیر نمونه ایی از سمبول هایی است که می توان در میدان دستورالعمل یک برنامه قرار داد.

CLA	non-MRI
ADD OPR	با آدرس مستقیم MRI
ADD PRT I	با آدرس غیرمستقیم MRI

سمبول سه حرفی هر سطر باید یکی از دستورالعمل های کامپیوتر در جدول ۱-۶ باشد. بدنبال یک دستور، مانند ADD، باید یک آدرس سمبولیک وجود داشته باشد. وجود حرف I اختیاری است. آدرس سمبولیک 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 اسمبولی را به یک برنامه دو دویی ترجمه کرد، ضروری است، هر آدرس سمبولیک که در میدان آدرس ذکر شده است مجدداً در میدان عنوان (برچسب) ظاهر گردد.

شبیه دستور، یک دستورالعمل ماشین نیست ولی دستورالعمل برای اسمبول است تا اطلاعاتی از بعضی فازهای ترجمه را در اختیار آن قرار دهد. چهار شبیه دستوری که اسمبول آنها را می شناسد در جدول ۷-۶ ارائه شده اند. (برنامه های اسمبولی دیگر شبیه دستورات متعدد دیگری دارند). شبیه دستور ORG به اسمبول 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. سمبول END در انتهای برنامه قرار گرفته است تا اسمبول را از ختم برنامه مطلع سازد. دو شبیه دستورالعمل دیگر پایه <sup>۱</sup> عملوند را مشخص کرده و به

#### جدول ۷-۶ تعریف شبیه دستورالعمل ها

سمبول	اطلاعاتی که به اسمبول داده می شود
ORG N	عدد شانزده شاخصی N مکانی از حافظه است که دستورالعمل با عملوند. واقع در سطر بعدی باید در آنجا قرار داده شود
END	انتهای برنامه سمبولیک
DEC N	عدد دهدی علامتدار N که باید به دو دویی تبدیل گردد
HEX N	عدد شانزده شاخصی N که باید به دو دویی 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 ۶-۸ مثالی از یک برنامه زبان اسمبلی است. در اولین سطر، شبه دستور ORG برای تعریف مبدأ برنامه در مکان حافظه ۱۰۰ (100) قرار دارد. شش خط بعد دستورات عمل‌های ماشین را تعریف می‌کنند و چهار سطر آخر هم شبه دستورات عمل می‌باشند. سه آدرس سمبولیک بكاررفته 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 مفروق منه با متتم ۲ مفروق انجام می‌شود. مفروق یک عدد منفی است. این عدد به یک عدد دودویی در نمایش متتم ۲ علامت دار بدل می‌شود زیرا خواسته‌ایم تا تمام اعداد دودویی بشکل متتم ۲ باشند. وقتی که متتم ۲ مفروق بدست آمد (با متتم کردن و افزایش دادن ، ۲۳ - به ۲۳ + تبدیل و اختلاف  $106 = 83 + 23 = 83 - (-23)$  بدست می‌آید.

#### ترجمه به دودویی

ترجمه برنامه سمبولیک 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 ۶-۸ برنامه زبان اسمبلی برای تفریق دو عدد

ORG 100	مبدأ برنامه مکان 100 است /
LDA SUB	بار کردن مفروق در AC /
CMA	متمن کردن / AC
INC	افزایش / AC
ADD MIN	جمع کردن مفروق منه با / AC
STA DIF	ذخیره تفاضل /
HLT	توقف کامپیوتر /
MIN,	مفروق منه /
SUB,	مفروق /
DIF,	محل ذخیره تفاضل /
DEC 83	پایان برنامه سمبولیک /
DEC -23	
HEX 0	
END	

جدول ۶-۸، به کد دودویی معادل آن را می‌توان با مرور برنامه و جایگزین کردن سمبول‌ها با معادل دودویی کد ماشین آنها انجام داد. با شروع از سطر اول، ابتدا با شبه دستورالعمل ORG مواجه می‌شویم. این دستور می‌گوید که برنامه دودویی را از آدرس 100 شانزده شانزدهی شروع کنیم. در سطر دوم دو سمبول بکار رفته است. این سطر باید یک دستورالعمل حافظه‌ای باشد که هر خانه 100 حافظه قرار دارد. چون حرف A وجود ندارد بیت اول کد دستورالعمل باید 0 باشد. نام سمبولیک دستورالعمل LDA است. با ملاحظه جدول ۶-۶ می‌بینیم که اولین رقم شانزده شانزدهی دستور باید 2 باشد، مقدار دودویی بخش آدرس باید از آدرس SUB بدست آید. ستون عنوان را مرور می‌کنیم و می‌بینیم که این سمبول در سطر ۹ واقع شده است. برای تعیین مقدار شانزده شانزدهی می‌بینیم که سطر ۲ حاوی دستورالعملی برای مکان 100 است و هر یک از سطرهای بعدی یک دستور ماشین یا یک عملوند را برای خانه‌های متوالی حافظه مشخص می‌کنند. با شمارش سطرهای، می‌بینیم که عنوان SUB در سطر ۹ مربوط به مکان 107 است. بنابراین آدرس شانزده شانزدهی دستور LDA باید 107 باشد. وقتی که دو بخش دستور مونتاژ شوند، کد شانزده شانزدهی 2107 حاصل می‌گردد. سطرهای دیگری که نمایش دهنده دستورات ماشین هستند به روش مشابهی ترجمه شده و کد شانزده شانزدهی آنها در جدول ۶-۹ لیست شده است.

دو سطر در برنامه سمبولیک که با شبه دستور DEC مشخص شده‌اند عملوند‌های دهدی را مشخص می‌نمایند. سومین سطر صفری را با شبه دستور HEX معین می‌کند (DEC نیز می‌تواند بکار رود). عدد دهدی 83 به دودویی تبدیل شده و در مکان 106 بقلم شانزده شانزدهی قرار داده شده است. عدد دهدی -23 - یک عدد منفی است و باید بقلم متمم 2 علامت دار دودویی تبدیل شود. معادل شانزده شانزدهی عدد دودویی در مکان 107 قرار دارد. سمبول END انتهای برنامه را خبر می‌دهد و می‌گوید که دیگر سطري برای ترجمه وجود ندارد.

جدول ۶-۹ لیست برنامه ترجمه شده جدول ۶-۸

کد شانزده شانزدهی		
عمل	محtra	برنامه سمبولیک
100	2107	ORG 100
101	7200	LDA SUB
102	7020	CMA
103	1106	INC
104	3108	ADD MIN
105	7001	STA DIF
106	0053	HLT
107	FFE9	DEC 83
108	0000	DEC -23
		HEX 0
		END

اگر ما دو مرور کامل بر روی برنامه سمبولیک داشته باشیم عمل ترجمه ساده می شود. در مرور اول ترجمه‌ای صورت نمی‌گیرد بلکه فقط مکانی برای هر دستورالعمل ماشین و عملوند در حافظه تعلق می‌گیرد. تخصیص مکان مقدار آدرس را برای عنوانین تعریف کرده و برای فرآیند ترجمه که در مرور دوم صورت می‌گیرد تسهیلات لازم را فراهم می‌آورد. بنابراین ما در جدول ۶-۹ مکان ۱۰۰ را به اولین دستور بعد ORG اختصاص می‌دهیم. سپس مکان‌های متوالی بعدی را تا انتهای برنامه برای هریک از سطرهای کد که حاوی دستور یا عملوند هستند، اختصاص می‌دهیم. (به END و ORG عددی اختصاص نمی‌باید زیرا دستور یا عملوند نیستند). وقتی که اولین مرور برنامه تکمیل شد، برای هر عنوان شماره مکان آن را مشخص می‌کنیم و جدولی را تشکیل می‌دهیم که مقدار شانزده شانزدهی هر آدرس سمبولیک را نشان می‌دهد. برای این برنامه جدول سمبول آدرس برابر زیر است:

آدرس شانزده شانزدهی	سمبل آدرس
MIN	106
SUB	107
DIF	108

در دوین مرور به برنامه سمبولیک، با مراجعه به جدول سمبول آدرس‌ها مقدار آدرس هر دستورالعمل حافظه‌ای را تعیین می‌کنیم. مثلاً سطر کد LDA SUB در مرور دوم با برداشتن مقادیر شانزده شانزدهی LDA از جدول ۶-۱ و SUB از لیست جدول آدرس ترجمه می‌شود. سپس دو بخش را بصورت یک دستورالعمل در مبنای شانزده کتار هم<sup>۱</sup> می‌گذاریم. اگر بخواهیم نحوه قرار گرفتن این برنامه را در کامپیوتر بدانیم می‌توانیم به آسانی کد شانزده شانزدهی حاصل را به دودویی تبدیل کنیم. وقتی که ترجمه از سمبول‌ها به دودویی بوسیله یک برنامه اسمبلر انجام شد، اولین مرور، اولین گذر خوانده شده و دوین مرور نیز دوین گذر نامیده می‌شود.

#### ۶-۴ اسمبلر

اسمبلر برنامه‌ای است که یک برنامه بزیان اسمبلی را دریافت و معادل زبان ماشین دودویی آن را تولید می‌کند. برنامه سمبولیک ورودی برنامه منبع<sup>۲</sup> و برنامه دودویی حاصل برنامه مقصد<sup>۳</sup> خوانده می‌شود. در واقع برنامه اسمبلر روی رشته‌های حرفی عمل کرده و معادل دودویی آنها را تولید می‌نماید.

### نمایش برنامه سمبولیک در حافظه

قبل از آغاز اس梅بل کردن (مونتاژ کردن)، برنامه سمبولیک باید در حافظه ذخیره شود. کاربر برنامه سمبولیک را از یک پایانه تایپ می‌کند. با استفاده از یک برنامه بارگذاری، کاراکترهای برنامه سمبولیک را در حافظه وارد می‌نماید. چون برنامه از مجموعه‌ای از سمبول‌ها 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 ۰ است و هفت بیت دیگر بوسیله ASCII مشخص می‌شود. معادل شانزده شانزدهی مجموعه کاراکترها در جدول ۶-۱۰ آورده شده است. به هر کاراکتر دو رقم مبنای شانزده تخصیص یافته است که بسادگی قابل تبدیل به کد معادل هشت بیتی آش می‌باشد. آخرین وارد در جدول، کاراکتری را تولید نمی‌کند بلکه مربوط به حرکت فیزیکی مکان نما<sup>۱</sup> بر روی صفحه پایانه است. کد مربوطه به CR هنگامی تولید می‌شود که کلید Return فشار داده شود. این باعث بازگشت "نورد"<sup>۲</sup> به مکان اولیه‌اش برای شروع سطر جدید می‌گردد. اس梅بل کد CR را عنوان 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 ۱۶ بیت ظرفیت است. هر سمبول عنوان به یک کاما ختم می‌شود. سمبول‌های عملیات و آدرس با یک فاصله<sup>۳</sup> و پایان هر سطر را کد CR مشخص می‌کند. مثلاً سطر کد

### PL3, LDA SUB I

جدول ۶-۱۰ 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
P	50	5	35		برگشت نورد

1- Cursor

2- Carriage

3- Space

در هفت کلمه حافظه متواالی، مطابق جدول ۱۱-۶، ذخیره شده است. عنوان PL3 دو کلمه را اشغال می‌کند و به یک کاما (2C) متنه می‌شود. میدان دستورالعمل در سطر کد ممکن است دارای یک یا چند سمبول باشد. هر سمبول به کد مریبوط به فاصله (20) ختم می‌شود، بجز آخرین سمبول که به کد برگشت نورد (0D) ختم می‌گردد. اگر سطر کد دارای توضیحات باشد، اسمبول آن را بوسیله ممیز یعنی 2F شناسایی می‌کند. اسمبول 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 سمبول سطر قرار می‌دهد. ورودی برنامه اسمبول برنامه بزیان سمبولیک در ASCII است. این ورودی بوسیله اسمبول دوبار مرور می‌شود تا برنامه معادل دودویی آن تولید شود. برنامه دودویی، خروجی تولید شده توسط اسمبول می‌باشد. اکنون ما بطور مختصر عده کارهای اسمبول را حین ترجمه تشریح می‌کنیم.

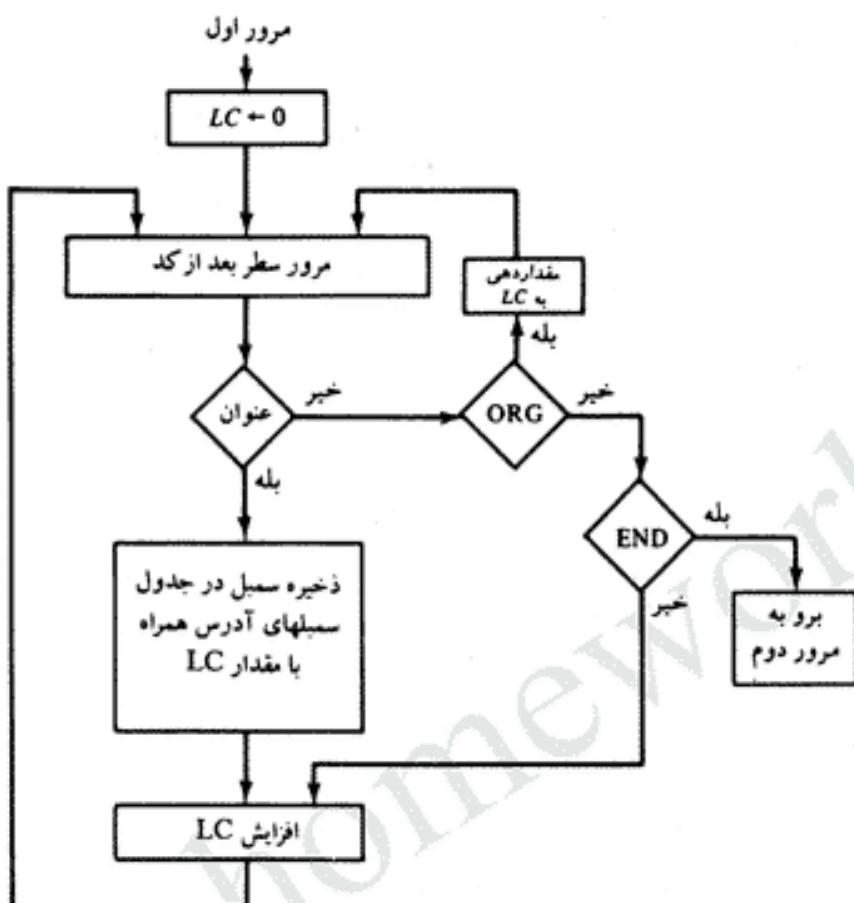
### مرور اول

یک اسمبول دو مروری سراسر برنامه سمبولیک را دوبار مرور می‌کند. در مرور اول، جدولی را تولید می‌کند که مقدار معادل دودویی همه سمبول‌های آدرس تعریف شده توسط کاربر را نشان می‌دهد. ترجمه به دودویی در مرور دوم صورت می‌گیرد. برای دنبال کردن محل دستورالعمل‌ها، اسمبول از کلمه‌ای در حافظه به نام مکان شمار<sup>۱</sup> (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، اسمبول در آغاز مقدار 0 را به LC می‌دهد. کارهایی که توسط اسمبول در اولین مرور صورت می‌گیرد در فلوچارت شکل ۶-۱ توصیف شده

جدول ۱۱-۶ نمایش سطر کد PL3, LDA SUB I در کامپیوتر

		کلمه حافظه	کد شانزده شاتزدهی	نمایش دودویی
1	P L	50 4C	0101 0000 0100 1100	
2	3 .	33 2C	0011 0011 0010 1100	
3	L D	4C 44	0100 1100 0100 0100	
4	A	41 20	0100 0001 0010 0000	
5	S U	53 55	0101 0011 0101 0101	
6	B	42 20	0100 0010 0010 0000	
7	I CR	49 · 0D	0100 1001 0000 1101	

1- Loation Counter



شکل ۶-۱ فلوچارت مرور اول اسمبلر

است. ابتدا ۰ می‌شود. خط اول کد سمبلیک تحلیل می‌شود تا معین شود آیا دارای عنوان است یا خیر (این کار با وجود کاما مشخص می‌گردد). اگر سطر فاقد عنوان باشد، اسمبلر سمبول را در میدان دستورالعمل چک می‌کند. اگر این میدان حاوی شبه دستور ORG باشد، اسمبلر LC را با مقداری که در جلوی ORG آمده است بار می‌کند و برای پردازش سطر بعدی باز می‌گردد. اگر سطر دارای شبه دستور END باشد، اسمبلر مرور اول را خاتمه می‌دهد و به سراغ مرور دوم می‌رود (دقت کنید سطر ORG و END دارای عنوان نیست). اگر سطر موردنظر دارای عنوان باشد، این عنوان همراه با معادل دو دویی اش که در LC قرار دارد در جدول سمبول آدرس قرار داده می‌شود. اگر عنوان وجود نداشته باشد چیزی در جدول ذخیره نمی‌شود. سپس یک واحد افزایش یافته و سطر جدیدی پردازش می‌گردد.

برای برنامه جدول ۶-۸، اسمبلر جدول سمبول آدرس در جدول ۶-۱۲ را تولید می‌کند. هر سمبول عنوان در دو مکان حافظه ذخیره می‌شود و بوسیله یک کاما خاتمه می‌یابد. اگر عنوان کمتر از سه کاراکتر داشته باشد، مکان‌های حافظه اضافی با کد فاصله پر می‌شود. مقداری که در LC در حین پردازش سطر وجود دارد در حافظه بعدی ذخیره می‌شود. این برنامه سه آدرس سمبلیک دارد: SUB، MIN و DIF.

## جدول ۶-۱۲ جدول سمبول‌های آدرس برای برنامه جدول ۶-۸

کلمه حافظه	سمبول (LC)*	کد شانزده‌شانزده	نایاب 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 شمارنده مکان را مشخص می‌کند (LC).

این سمبول‌ها بترتیب آدرس‌های 12 بیتی را که معادل 106، 107 و 108 می‌باشدند نمایش می‌دهند. جدول سمبول آدرس سه کلمه حافظه فرق را برای هر سمبول عنوان که با آن مواجه شود اشغال می‌کند و داده‌های خروجی را که اسمبول در حین مرور اول ایجاد می‌کند تشکیل می‌دهد.

## مرور دوم

دستورات ماشین در حین مرور دوم با استفاده از روش‌های نظاره به جدول<sup>۱</sup> ترجمه می‌شوند. روش نظاره به جدول جستجویی است بر واردۀ‌های جدول برای اینکه تعیین کنیم آیا یک نمونه مورد نظر با نمونه‌های ذخیره شده در جدول مطابقت دارد یا خیر. اسمبول چهار جدول را بکار می‌برد. هر سمبولی که در برنامه با آن مواجه شود باید بعنوان یکی از واردۀ‌ها در این جدول موجود باشد؛ در غیر اینصورت سمبول قابل تفسیر نیست. ما نام‌های زیر را به چهار جدول اختصاص می‌دهیم:

۱- جدول شبه دستورالعمل‌ها

۲- جدول MRI

۳- جدول non-MRI

۴- جدول سمبول آدرس

واردۀ‌های جدول شبه دستورالعمل‌ها چهار سمبول دارد: ORG، DEC، END و HEX. هر واردۀ هنگام برخورد با آن در برنامه، اسمبول را به زیروال شبه دستورMRI مربوطه، هدایت می‌کند. جدول MRI حاوی هفت سمبول دستورالعمل‌های حافظه‌ای و کد عمل معادل سه بیتی آنهاست. جدول non-MRI حاوی هجده دستورالعمل ثباتی و ورودی- خروجی همراه با شانزده کد دودویی معادل آنها می‌باشد.

## 1- Table look up

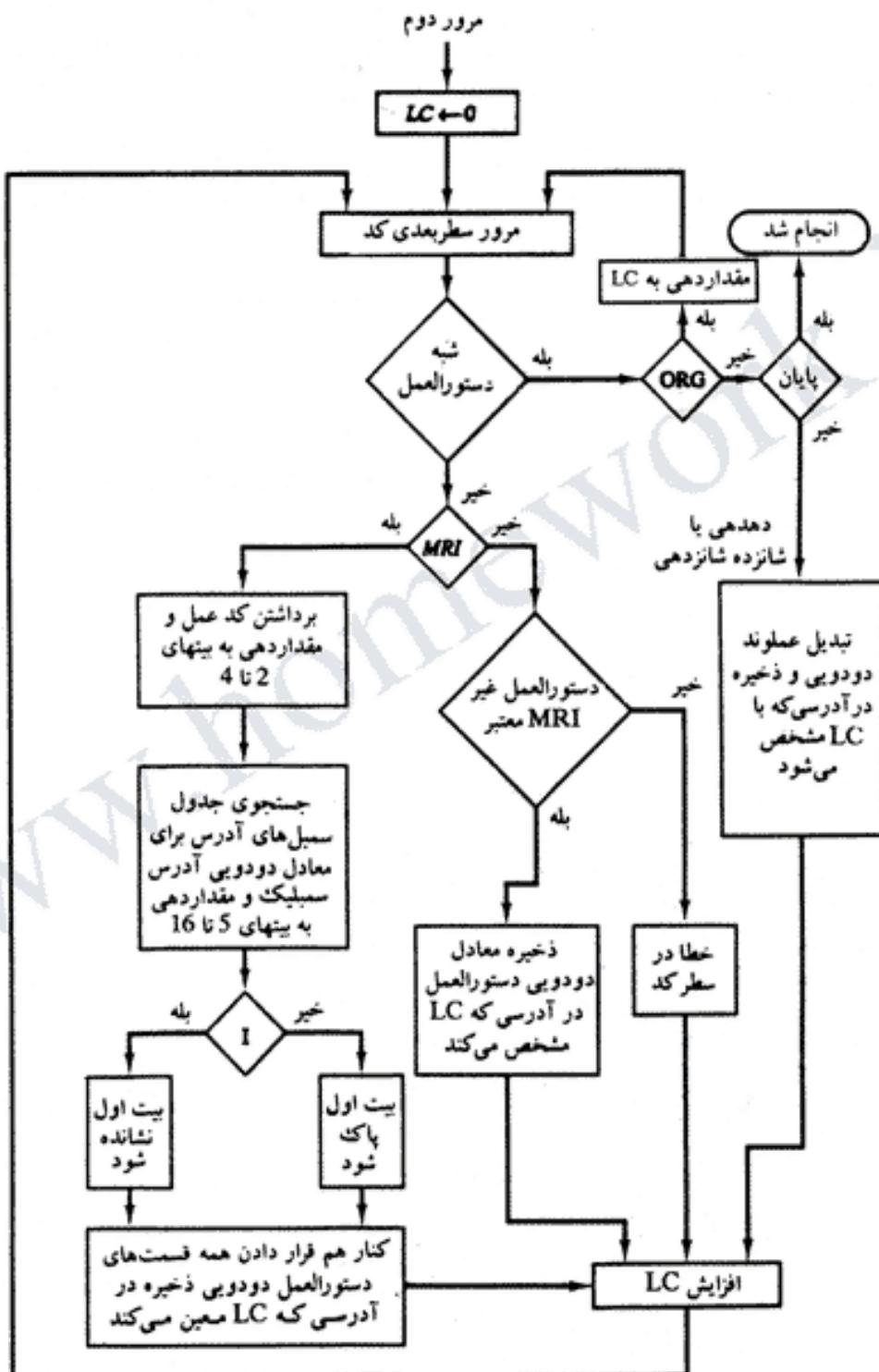
جدول سمبول آدرس در مرور اول پردازش اسمبلی تولید می‌گردد. اسمبلر این جداول را برای یافتن سمبولی که در حال پردازش است جستجو می‌کند تا مقدار دودویی آن را معین کند.

کاری که در حین دومین مرور توسط اسمبلر انجام می‌شود در شکل ۲-۶ آمده است. LC در آغاز با ۰ بار می‌شود. سپس سطرهای کد یک به یک تحلیل می‌شوند. عنوانین در مرور دوم چشم‌پوشی می‌شوند، لذا اسمبلر بلا فاصله به میدان آدرس رفته و اولین سمبولی را که با آن مواجه می‌شود وارسی 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 دودویی شان می‌گردد. این عملوند در خانه‌ای از حافظه که بوسیله LC مشخص می‌شود قرار می‌گیرد. سپس شمارنده مکان، LC، یک واحد افزایش می‌باید و اسمبلر به تحلیل کد سطر بعدی ادامه می‌دهد.

اگر سمبولی که با آن برخورد شده است شبه دستور نباشد، اسمبلر به جدول MRI مراجعه می‌کند. اگر سمبول در این جدول پیدا نشد، اسمبلر به جدول non-MRI می‌رود. سمبولی که در این جدول پیدا شود متعلق به دستورالعمل‌های ثباتی یا ورودی- خروجی است. اسمبلر کد شانزده بیتی دستورالعمل را در کلمه حافظه‌ای که بوسیله LC مشخص می‌گردد ذخیره می‌کند. مکان‌شمار، LC، افزایش یافته و تحلیل ادامه می‌باید.

هنگامی که سمبولی 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 دستورالعمل حافظه‌ای با دو یا سه سمبول مشخص می‌شود. سمبول دوم آدرس سمبولیک است و سومی، که ممکن است وجود داشته باشد و یا نداشته باشد حرف I است. آدرس سمبولیک با جستجوی جدول سمبول 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 دستورالعمل در کنار هم گذاشته شده<sup>۱</sup> (یا به بیان دیگر موتناز 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 سمبولیک است. این کار تشخیص خطأ<sup>۲</sup> خوانده می‌شود. چنین خطایی ممکن است سمبول نامعتبر کد ماشین باشد که نشانه آن عدم وجودش در جدول‌های MRI و non-MRI است. اسمبلر قادر به ترجمه چنین سمبولی نیست زیرا مقدار معادل دودویی آن را نمی‌داند. در چنین حالتی، اسمبلر پیام خطایی را چاپ می‌کند تا برنامه‌نویس را از وجود خطأ در سطر خاصی از کد برنامه سمبولیک مطلع سازد. خطای ممکن دیگر این است که برنامه دارای آدرس سمبولیکی باشد که در جدول وجود ندارد اسمبلر قادر نخواهد بود این سطر کد را بطور صحیحی ترجمه کند زیرا معادل دودویی آن سمبول را در جدول سمبول آدرس که در مرور اول تولید شده نمی‌باید. خطاهای دیگری هم ممکن است رخ دهند و یک اسمبلر قابل استفاده در عمل باید



شکل ۲-۲ فلوچارت مرور دوم اسمنیلر

همه این گونه خطاهای را کشف و برای هر یک پیغام خطایی را چاپ کند.

باید تأکید کرد که یک اسمبلر عملی بسیار پیچیده‌تر از آنچه در اینجا شرح داده‌ایم می‌باشد. اکثر کامپیووترها انعطاف زیادی را در نوشتن برنامه‌ها بزبان اسمبلی در اختیار برنامه‌نویس می‌گذارند. مثلاً ممکن است به کاربر اجازه داد تا برای مشخص کردن آدرس از عدد یا یک سمبول استفاده کند. بسیاری از اسمبلرها این امکان را به کاربر می‌دهند که آدرس را با یک عبارت ریاضی مشخص کند. شبیدستورالعمل‌های بسیار زیادی را نیز ممکن است برای تسهیل در کار برنامه‌نویس مشخص کرد. با کاراتر شدن و تکمیل زبان اسمبلی، اسمبلر پیچیده‌تر می‌شود.

## ۶-۵ حلقه در برنامه‌نویسی

حلقه برنامه رشته‌ای از دستورات است که چندین بار، و هر بار با مجموعه متفاوتی از داده‌ها، اجرا می‌شود. حلقه برنامه در فرترن با عبارت DO مشخص می‌شود. مثال زیر برنامه‌ای بزبان فرترن است که مجموع 100 عدد صحیح را تهیه می‌کند.

```
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 می‌شود و هر بار عملوند متفاوت بوده و مقادیر (J) که در آن 100, ..., 0, 1, ... می‌باشد بکار می‌روند.

برنامه سیستمی که یک برنامه بزبان سطح بالا را، مانند آنچه در بالا دیدیم به زبان ماشین ترجمه کند، کامپایلر<sup>۱</sup> نام دارد. کامپایلر در مقایسه با اسمبلر، برنامه پیچیده‌تری است و درک کامل عملکرده آن مستلزم دانش برنامه‌نویسی سیستم است. معهذا، می‌توانیم 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 می‌تواند از یک برنامه اسمبلی یعنوان 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 می‌دهد تا 100 کلمه حافظه را برای 100 عملوند رزرو نماید. مقدار عملوندها از یک عبارت ورودی<sup>۲</sup> بدست می‌آیند (دستور مزبور در برنامه بالا آورده نشده است). عبارت دوم به کامپایلر خبر می‌دهد که

اعداد صحیح هستند. اگر اعداد از نوع غیرصحیح<sup>۱</sup> باشند، کامپایلر باید مکانهایی را برای اعداد معیز شناور رزرو نموده و دستوراتی را تولید کند که عملیات محاسباتی بعدی را با داده‌های معیز شناور انجام دهند. این دو دستور همانند شبه دستور العمل‌ها در اسمبلی، غیرقابل اجرا هستند. فرض کنید کامپایلر مکانهای ۱۶(150) تا ۱۶(1B3) را برای ۱۰۰ عملوند رزرو نماید. این کلمات حافظه رزرو شده در سطرهای ۱۹ الی ۱۱۸ برنامه ترجمه شده جدول ۱۳-۶ لیست شده‌اند. این کار توسط شبه دستور ORG در سطر ۱۸ انجام می‌شود که ابتدای عملوند‌ها را مشخص می‌کند. اولین و آخرین عملوند‌ها توسط عدد دهدی 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 شده‌اند. آن‌دیس موجود در عبارت DO بصورت دستورالعمل‌های سطرهای ۲ الی ۵ و ثابت‌های سطرهای

جدول ۱۳-۶ برنامه سمبلیک برای جمع ۱۰۰ عدد

سطر			
1		ORG 100	مبدأ برنامه ۱۰۰ شانزدهی /
2		LDA ADS	بار کردن آدرس ابتدای عملوندها /
3		STA PTR	ذخیره در اشاره گر /
4		LDA NBR	بار کردن ۱۰۰ منی /
5		STA CTR	ذخیره در شمارنده /
6		CLA	هاک کردن ایشانگر /
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	مبدأ عملوندها ۱۵۰ در بنای شانزدهی است /
19		DEC 75	نهایی عملوند /
*			
*			
*			
118		DEC 23	آخرین عملوند /
119		END	انهای برنامه تمامی /

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 فرترن 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) حاصل از (J) 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  $0 = \text{SUM}$  را به یک دستور العمل ماشین که مکان SUM را بعنوان مقدار اولیه صفر مینماید ترجمه می کنند. سپس ارجاع به این مکان هر بار که سطر 3 برنامه فرترن اجرا شود صورت می گیرد. کامپایلر هوشمندتر می تواند چنین تشخیص دهد که جمع در اینباره<sup>۱</sup> صورت گرفته و فقط نتیجه نهایی در SUM قرار گیرد. کامپایلر دستور العملی در سطر 6 تولید خواهد کرد تا اینباره 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 دستور العمل های واقع در سطرهای 7 تا 10 ترجمه می شود. سطر 7 دستور العمل ADD غیر مستقیم است زیرا دارای سمبول I می باشد. آدرس عملوند جاری در خانه PTR ذخیره می شود. وقتی این مکان بطور غیرمستقیم آدرس دهنده شود کامپیوتر محتوای PTR را بعنوان آدرس عملوند تلقی می نماید. در نتیجه، عملوند در مکان 150 با اینباره جمع می شود. سپس PTR با دستور العمل 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 دستور العمل بعدی گذر نمی کند. دستور بعدی یک دستور العمل 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 گذر<sup>۲</sup> (صرفنظر) می شود و کامپیوتر دستور العمل سطر 11 و 12 را اجرا می نماید. جمع حاصل در اینباره در SUM ذخیره شده و کامپیوتر متوقف می شود. دستور العمل توقف<sup>۳</sup> 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 ثابت است هرگز گذری رخ نمی دهد.

برنامه جدول ۱۳-۶ برای نخستین بار مفاهیم اشاره گر، شمارنده همراه با عمل آدرس دهنده 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 اجزای حلقه برنامه را می شمارد. در این مثال ما دو وظیفه را برای این مقاصد استفاده کردیم. در کامپیوتراهایی که بیش از یک ثبات پردازنده دارند می توان از یک ثبات بعنوان

اشارة گر، از دیگری بعنوان شمارنده و از سومی بجای انباره استفاده کرد. وقتی که از ثبات‌های پردازنده بعنوان اشاره گر و شمارنده استفاده شود به آنها ثبات‌های شاخص<sup>۱</sup> گفته می‌شود. ثبات‌های شاخص در بخش ۸-۵ مورد بحث قرار گرفته‌اند.

## ۶-۶ برنامه‌نویسی اعمال حسابی و منطقی

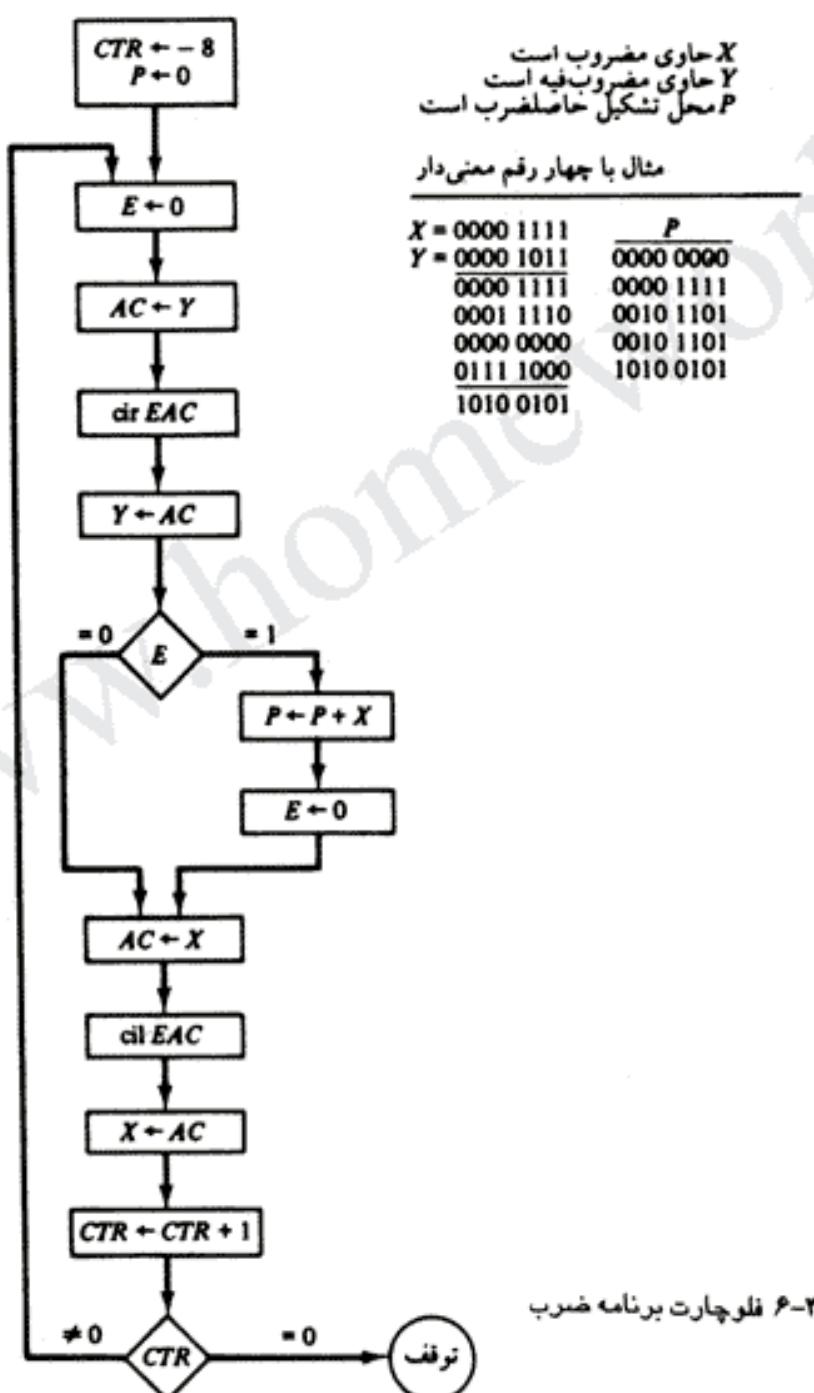
تعداد دستورالعمل‌های موجود در یک کامپیوتر از چند صد دستور در سیستم‌های بزرگ تا چند دوچین در کامپیوترهای کوچک متغیر است. برخی کامپیوترها عمل مفروضی را با یک دستورالعمل انجام می‌دهند و برخی دیگر برای انجام همان عمل ممکن است تعداد زیادی دستورات ماشین نیاز داشته باشند. مثلاً چهار عمل اصلی حساب را در نظر 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 دستورالعمل‌های ماشین گنجانده نشده باشد باید بوسیله یک برنامه پیاده سازی شود. ما در جدول ۶-۸ برنامه‌ای را برای تفریق دو عدد دیدیم. برنامه‌های مربوط به سایر عملیات حسابی به روشنی مشابه بودند.

اعمالی که در یک کامپیوتر توسط یک دستورالعمل ماشین پیاده سازی شده باشند گوئیم بطور سخت‌افزاری پیاده سازی شده‌اند. به اعمالی که بوسیله مجموعه‌ای از دستورالعمل‌ها پیاده سازی می‌شوند گوییم که بطور نرم‌افزاری پیاده سازی شده‌اند. برخی از کامپیوترها مجموعه گسترده‌ای از دستورات سخت‌افزاری را دارا هستند که برای بالا بردن سرعت انجام کارهای متداول طراحی شده‌اند. دسته‌ای دیگر دارای دستورالعمل‌های سخت‌افزاری کمتری هستند و بر پیاده سازی نرم‌افزاری اعمال متکی‌اند. پیاده سازی سخت‌افزاری بسیار پرهزینه است زیرا مدارهای اضافه‌تری برای پیاده سازی اعمال نیاز دارد. پیاده سازی نرم‌افزاری موجب ایجاد برنامه‌های طولانی هم از نظر تعداد دستورات و هم از نظر زمان اجرا می‌شود. این بخش پیاده سازی نرم‌افزاری چند عمل حسابی و منطقی را نشان می‌دهد. برنامه‌ها برای هر نوع عمل حسابی نه فقط برای داده‌های ممیز ثابت بلکه برای داده‌های دهدی و ممیز شناور نوشته شده‌اند. پیاده سازی سخت‌افزاری عملیات حسابی در فصل ۱۰ آورده شده است.

### برنامه ضرب

اینک برنامه‌ای را برای ضرب دو عدد می‌نویسیم. برای ساده کردن برنامه از بیت علامت صرفنظر کرده و فرض می‌کنیم اعداد مثبت باشند. همچنین فرض می‌کنیم که دو عدد دودویی بیش از هشت بیت ندارند، لذا حاصلضرب آنها از ۱۶ بیت تجاوز نمی‌کند. می‌توان برنامه را اصلاح کرد تا علامت اعداد نیز در نظر گرفته شود و یا از اعداد ۱۶ بیتی استفاده گردد. در این صورت حاصلضرب ممکن است تا ۳۱ بیت طول داشته و دو کلمه از حافظه را اشغال نماید.

برنامه ضرب دو عدد مبتنی بر روش ضرب اعداد با قلم و کاغذ است. همانطور که در مثال عددی شکل ۶-۳ نشان داده شده است، فرآیند ضرب از وارسی بیت‌های مضروب فیه  $Y$  و جمع مضروب  $X$  بتعادل ۱ های موجود در  $Y$  تشکیل می‌شود، پشرطی 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 ۶-۳ فلوچارت برنامه ضرب

حافظه را که با  $P$  نشان داده شده برای حفظ مجموعهای میانی<sup>۱</sup> در نظر می‌گیریم. مجموعهای میانی، حاصلضربهای جزیی خوانده می‌شوند زیرا حاوی بخشی از حاصلضرب هستند که تا آن زمان بدست آمده است. همانطورکه در مثال عددی در شکل نشان داده شده است، حاصلضرب جزیی با صفر شروع می‌شود، مضروب،  $X$ ، به محتوای  $P$ ، بازاء هر بیت از مضروب فیه  $Y$  که ۱ است، اضافه می‌شود. پس از وارسی هر یک از بیتهای مضروب فیه، مقدار  $X$  بسمت چپ شیفت داده می‌شود. مقدار نهايی حاصل در  $P$  حاصلضرب را تشکیل می‌دهد. وقتی که اعداد در هم ضرب شدند، حاصلضرب دارای چهار بیت معنی دار است. پس از عمل ضرب، حاصلضرب هشت بیت معنی دار خواهد داشت. کامپیوتر می‌تواند از اعداد با هشت بیت معنی دار استفاده کند و لذا حاصلضرب تا ۱۶ بیت طول خواهد داشت.

فلوچارت شکل ۶-۳ رویه برنامه‌نویسی عمل ضرب را قدم به قدم نشان می‌دهد. برنامه دارای حلقه‌ای است که هشت بار طی می‌شود، یعنی هر بار بازاء یک بیت معنی دار مضروب فیه. در ابتدا، مکان  $X$  حاوی مضروب و مکان  $Y$  حاوی مضروب فیه است. شمارنده CTR با ۸- بار می‌شود و مکان  $P$  به ۰ پاک می‌گردد. هر بیت مضروب فیه پس از انتقال به  $E$  وارسی می‌شود. این عمل با پاک کردن  $E$ ، بارگردان  $Y$  در  $AC$ ، چرخش  $E$  و  $AC$  به راست و ذخیره عدد شیفت یافته به  $Y$  انجام می‌گردد. بیت ذخیره شده در  $E$  کم ارزشترین بیت مضروب فیه است. اکنون مقدار  $E$  را وارسی می‌کنیم. اگر  $E$  برابر ۱ باشد، مضروب،  $X$ ، با حاصلضرب جزیی  $P$  جمع می‌شود. ولی اگر  $E$  برابر ۰ باشد حاصلضرب جزیی تغییری نمی‌کند. سپس مقدار  $X$  را با بارگردان 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 ۶-۱۴ دستورات ضرب دو عدد بدون علامت را نشان می‌دهد. مرحله مقداردهی اولیه<sup>۲</sup> آورده نشده است ولی بهنگام بارگردان برنامه در کامپیوتر باید انجام شود. مقداردهی اولیه شامل آوردن مضروب و مضروب فیه به ترتیب در خانه‌های  $X$  و  $Y$ ؛ دادن مقدار ۸- 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 کامپیوترا فاقد دستور ماشین برای کار مورد نظر می‌باشد، آن عمل را می‌توان با رشته‌ای از دستورالعمل‌ها برنامه‌نویسی کرد. بنابراین ما پیاده‌سازی نرم‌افزاری عمل ضرب را نشان دادیم. پیاده‌سازی سخت‌افزاری متناظر آن در بخش ۱۰-۳ آمده است.

### جمع با دقت مضاعف

هنگامی که دو عدد ۱۶ بیتی بی علامت درهم ضرب شوند، نتیجه حاصلضرب ۳۲ بیتی خواهد بود که باید در

## جدول ۶-۱۴ برنامه ضرب دو عدد مثبت

	ORG 100	
LOP,	CLE	پاک کردن /
	LDA Y	بارگردان مضروب فیه /
	CIR	انتقال بیت مضروب فیه به /
	STA Y	ذخیره مضروب فیه جایجا شده /
	SZE	چک بیت، صفر است یا نه /
	BUN ONE	بیت ۱ است، به ONE برو /
	BUN ZRO	بیت ۰ است، به ZRO برو /
ONE,	LDA X	بارگردان مضروب /
	ADD P	جمع با حاصلضرب جزیی /
	STA P	ذخیره حاصلضرب جزیی /
	CLE	پاک کردن E /
ZRO,	LDA X	بارگردان مضروب /
	CIL	چرشش 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 تیکل حاصلضرب /
	END	

دو کلمه حافظه ذخیره شود. هرگاه عددی در دو کلمه حافظه جای گیرد گوییم دقت مضاعف دارد. وقتی که یک حاصلضرب جزیی 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 ۶-۱۵ 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 چرشش داده می شود. سپس دو بخش پرازشتر 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، CLA و CMA 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	/ A	بار کردن بخش کم ارزشتر
ADD BL	/ E	جمع بخش کم ارزشتر B، نقلی در
STA CL	/ C	ذخیره در بخش کم ارزشتر
CLA	/ AC	پاک کردن
CIL	/ AC(0)	چرخش برای آوردن نقلی به درون
ADD AH	/	جمع کردن بخش بالارزشتر A و نقلی
ADD BH	/ B	جمع کردن بخش پرارزشتر
STA CH	/ C	ذخیره در بخش پرارزشتر
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 هرتابع 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 دسورگان رابطه  $x' + y = (x'y)'$  را می توان نتیجه گرفت. عبارت دوم فقط از اعمال AND و متمم سازی تشیکل 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 بارگیرن
CMA	Complement to get $\bar{A}$	با متمم سازی $\bar{A}$ را بدست یافته باور
STA TMP	Store in a temporary location	حاصل را در یک مکان مؤقت ذخیره کن
LDA B	Load Second operand B	عملوند دوم را در B بارگیرن
CMA	Complement to get $\bar{B}$	با متمم سازی $\bar{B}$ را بدست یافته باور
AND TMP	AND with $\bar{A}$ to get $\bar{A} \wedge \bar{B}$	حاصل را با برای بدست آوردن $\bar{A} \wedge \bar{B}$ کن AND
CMA	Complement again to get A $\wedge$ B	حاصل را مجددآ 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 کامپیوتراها 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 دستور العمل ها می توان برنامه نویسی کرد. شیفت منطقی مستلزم این است که رقم ۰ به یکی از دو انتهای اضافه شود. این بسادگی با پاک کردن E و

چرخش CLE و E میسر است. بنابراین برای یک عمل منطقی شیفت به راست مانیاز به دو دستور العمل 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 ۴-۶ 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 برایت حسابی مستلزم این است تا E را برابر مقدار بیت علامت کنیم و سپس چرخش به راست را انجام دهیم. بنابراین:

CLE	E را با 0 پاک کن
SPA	اگر AC مثبت است گذر کن
CME	AC منفی است، E برابر 1 شود
CIR	چرخش E و AC

برای شیفت به چپ حسابی لازم است تا بیت اضافه شده به متنهای ایه 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 صحیح انجام گرفته است، ولی اگر برابر نباشند، سریز رخ داده است. سریز 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

مکرراً اتفاق افتاده است که یک قطعه از کد در بخش‌های 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 دستورالعمل‌های مشترکی که در یک برنامه بتوان از آنها بدفعات زیاد استفاده کرد زیر روال نام دارد. هر باو که زیر روال در بخش اصلی برنامه بکار رود، انشعابی به ابتدای زیر روال صورت

می‌گیرد. پس از اجرای زیرروال، انشعابی دیگر موجب بازگشت به برنامه اصلی می‌گردد. زیر روال از رشته مستقلی از دستورالعمل‌ها تشکیل می‌شود که کار معینی را انجام می‌دهند. از هر جای برنامه اصلی می‌توان به زیر روال انشعاب کرد. در اینجا این سؤال مطرح می‌شود که زیرروال چگونه می‌فهمد که به چه محلی بازگردد، زیرا معنی است از مکان‌های متعددی در برنامه اصلی به زیرروال مذبور انشعاب گردد. بنابراین لازم است آدرس بازگشت را جایی در کامپیوتر ذخیره کنیم تا زیرروال بداند به کجا بازگردد. چون انشعاب به زیر روال و بازگشت به برنامه اصلی عمل متداولی است، تمام کامپیوترها دستورات خاصی را برای ورود به زیر روال و بازگشت از آن در اختیار می‌گذارند.

در کامپیوتر پایه، رابط بین برنامه اصلی و یک زیرروال دستورالعمل<sup>1</sup> BSA می‌باشد. برای شرح چگونگی استفاده از این دستور، زیر روالی می‌نویسیم که محتوای یک انباره را چهار بار به چه شیفت دهد. چهار بار شیفت یک کلمه برای پردازش اعداد دهدی کد شده با دو دویی، 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 ۶-۱۶ با بارگیردن X در AC شروع می‌شود. دستورالعمل بعدی که با آن مواجه می‌شویم BSA

جدول ۶-۱۶ برنامه‌ای برای نمایش طریقه استفاده از زیرروال‌ها

مکان			
100	ORG 100		برنامه اصلی /
	LDA X		باردهی X /
101	BSA SH4		انشعاب به زیرروال /
102	STA X		ذخیره عدد جایجا شده /
103	LDA Y		/
104	BSA SH4		انشعاب دوباره به زیرروال /
105	STA Y		ذخیره عدد جایجا شده /
106	HLT		
107	X,	HEX 1234	
108	Y,	HEX 4321	
			زیرروال چهار بار جایجا می‌شود
109	SH4.	HEX 0	عمل ذخیره آدرس بازگشت /
10A		CIL	یک بار چرخش به چپ /
10B		CIL	
10C		CIL	
10D		CIL	چرخش به چپ برای چهارمین بار /
10E		AND MSK	سفر کردن (16-13) / AC(13)
10F		BUN SH4 I	بازگشت به برنامه اصلی /
110	MSK,	HEX FFF0	عملوند پوشش /
		END	

## 1- Branch and Save Address

است. دستورالعمل BSA در مکان 101 است. زیر روای 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 سمبولیک SH4 تعریف می شود (که 109 است) ذخیره می کند. همچنین محتوای 1 SH4+1 را بداخل شمارنده برنامه منتقل می سازد. پس از اجرای این دستورالعمل، مکان حافظه 109 حاوی دودویی معادل با شانزده شانزده 102 می باشد و شمارنده برنامه حاوی دودویی معادل با شانزده شانزده 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 روای 10A 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 روای کامپیوتر را به برنامه اصلی باز می گرداند. این عمل با دستورالعمل انشعب غیرمستقیم و سمبول آدرس مورد استفاده برای نام زیر روای صورت می گیرد. آدرسی که کامپیوتر به آن انشعب 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 انشعب 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 روای به برنامه اصلی در مکان 105 باز می گردد.

از این مثال می بینیم که اولین مکان حافظه هر زیر روای نقش یک رابط بین برنامه اصلی و زیرروال را بعده دارد. رویه انشعب به زیرروال و بازگشت به برنامه اصلی پیوند<sup>1</sup> زیر روای نامیده می شود. عملی را که دستورالعمل BSA انجام می دهد معمولاً فراخوانی<sup>2</sup> زیرروال می نامند. آخرین دستورالعمل زیرروال نیز عملی را که بازگشت<sup>3</sup> از زیرروال می خوانند انجام می دهد.

روش بکار رفته در کامپیوتر پایه برای پیوند زیرروال عمدتاً در کامپیوتراهای یافت می شود که فقط یک ثبات پردازنده دارند. بسیاری از کامپیوتراها دارای چندین ثبات پردازنده هستند و به برخی از آنها نام ثبات اندیس<sup>4</sup> (شاخص) داده شده است. در چنین کامپیوتراهایی، یک ثبات اندیس برای پیوند زیرروال بکار می رود. دستورالعمل انشعب به زیر روای آدرس برگشت را در ثبات اندیس ذخیره می کند. دستورالعمل بازگشت از زیر روای، با انشعب 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 داده های پیوند

هنگامی که زیرروالی فراخوانده شود، برنامه اصلی باید داده هایی 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. زیر روal 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 داده های برنامه خواننده دسترسی داشته و تایج را به آن بازگرداند. از انباره می توان برای یک پارامتر ورودی و یک پارامتر خروجی استفاده کرد. در کامپیوتروهایی که ثبات های متعدد دارند تعداد بیشتری پارامتر بدین طریق قابل انتقال است. راه دیگری برای انتقال داده به یک زیر روal از طریق حافظه است. داده ها غالباً در مکان هایی از حافظه که بدنبال فراخوانی آمده اند قرار داده می شوند. همچنین می توان آنها را در بلاک هایی از حافظه نیز ذخیره کرد. سپس اولین آدرس بلاک در مکان بعد از فراخوانی قرار می گیرد. در هر حال، آدرس بازگشت همواره اطلاعات ارتباطی برای انتقال داده بین برنامه اصلی و زیرروال را فراهم می آورد.

بعنوان مثال، زیرروالی را در نظر بگیرید که عمل منطقی OR را انجام دهد. دو عملوند باید به زیر روal منتقل و زیر روal هم باید نتیجه عمل را بازگرداند. از انباره می توان برای انتقال یک عملوند و دریافت نتیجه استفاده کرد. عملوند دیگر در مکانی که بدنبال دستور العمل BSA می آید وارد می شود. این مطلب در برنامه جدول ۶-۱۷ نشان داده شده است. اولین عملوند در مکان X بداخل AC بار می شود. عملوند دوم در مکان 202 که بدنبال دستور BSA است ذخیره می شود. پس از انشعاب، اولین

جدول ۶-۱۷ برنامه ای برای نمایش اشتراک پارامترها

مکان			
		ORG 200	
200		LDA X	بارگردان 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	منم کردن عملوند اول /
209		STA TMP	ذخیره در مکان موقت /
20A		LDA OR I	بارگردان عملوند دوم /
20B		CMA	منم کردن عملوند دوم /
20C		AND TMP	کردن AND منم عملوند اول /
20D		CMA	منم مجدد برای بازن / 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. وقت 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 دستور العمل غیرمستقیم در مکان OR به AC بار می‌شود. بیاورد 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 AND، TMP می‌شود. متمم کردن نتیجه، حاصل عمل OR را می‌دهد.

بازگشت از زیر روال باید بگونه‌ای صورت گیرد که برنامه اصلی از خانه 203، که دستور العمل بعدی در آن قرار دارد ادامه یابد. این کار با افزایش آدرس OR توسط دستور العمل ISZ صورت می‌گیرد. اکنون مکان OR عدد 203 را نگه میدارد و یک دستور العمل BUN غیرمستقیم موجب بازگشت به محل صحیح می‌گردد. اعکان دارد بیش از یک عملوند بدبال دستور العمل 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 ۶-۱۸ آورده شده است. طول بلاک ۱۶ کلمه می‌باشد. اولین دستور العمل، انشعب به MVE است. اولین بخش زیر روال سه پارامتر 100، 200 و 16- را از برنامه اصلی منتقل و در مکان‌های مربوطه شان قرار می‌دهد. اقلام داده از بلاک‌های خود بکمک دو اشاره‌گر برداشته می‌شوند. شمارنده انتقال تنها ۱۶ 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 دستور العمل HLT است.

## ۶-۸ برنامه‌نویسی ورودی - خروجی

کاربران کامپیوتر برنامه‌ها را با استفاده از سمبول‌هایی می‌نویستند که بوسیله زبان برنامه‌نویسی بکار رفته تعریف شده‌اند. سمبول‌ها رشته‌هایی از کاراکترها هستند و به هر کاراکتر یک کد ۸ بیت اختصاص می‌یابد بطوری که می‌توان آنرا در حافظه کامپیوتر ذخیره کرد. یک کاراکتر کد شده به دودویی هنگامی وارد کامپیوتر می‌شود که یک دستور العمل INP<sup>۱</sup> (ورودی) اجرا شود. یک کاراکتر کد شده به دودویی وقتی به دستگاه خروجی<sup>۲</sup> منتقل می‌شود که یک دستور العمل OUT<sup>۳</sup> (اجرا گردد. وسیله خروجی کد دودویی را تشخیص داده و کاراکتر مربوط به آن را تایپ می‌کند.

جدول ۶-۱۹ (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 نیزروال انتقال بلاک دادها

		برنامه اصلی /
BSA MVE		اشعباب به زیر روال /
HEX 100		آدرس شروع مبدأ دادهها /
HEX 200		آدرس شروع مقصد دادهها /
DEC - 16		تعداد اقلامی که باید انتقال باید /
HLT		
MVE,	HEX 0	زیرروال / MVE
	LDA MVE I	بارگردان آدرس مبدأ /
	STA PT1	ذخیره در اشاره گر اول /
	ISZ MVE	افزایش آدرس بازگشت /
	LDA MVE I	بارگردان آدرس مقصد /
	STA PT2	ذخیره در اشاره گر دوم /
	ISZ MVE	افزایش آدرس برگشت /
	LDA MVE I	بارگردان تعداد اقلام /
	STA CTR	ذخیره در شمارنده /
	ISZ MVE	افزایش آدرس برگشت /
LOP,	LDA PT1 I	بارگردان یک قلم 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 ۰ است، انشعباب برای وارسی مجدد /
	INP	پرچم ۱ است، دریافت کاراکتر /
	OUT	چاپ کاراکتر /
	STA CHR	ذخیره کاراکتر /
	HLT	
CHR,	—	محل ذخیره کاراکتر /
(ب) خروجی یک کاراکتر		
COF,	LDA CHR	بارگردان کاراکتر در AC /
	SKO	چک کردن پرچم خروجی /
	BUN COF	پرچم ۰ است، انشعباب برای وارسی مجدد /
	OUT	پرچم برابر ۱ است، ارسال کاراکتر /
	HLT	
CHR,	HEX 0057	کاراکتر مورد نظر "W" است /

می‌دهد. دستورالعمل SKI پرچم<sup>۱</sup> ورودی را چک می‌کند تا وجود یک کاراکتر را برای انتقال دریابد. اگر پرچم ورودی ۱ باشد دستور بعدی گذر می‌شود. دستورالعمل INP کاراکتر کد شده به دودویی را به (0-7) AC می‌فرستد. سپس کاراکتر توسط دستورالعمل OUT چاپ می‌شود. واحد پایانه‌ای<sup>۲</sup> (ترمینال) که مستقیماً به کامپیوتر متصل است، با فشار دادن کلید روی صفحه کلید، کاراکتری را چاپ نمی‌کند. 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 ۰ بیابد، دستورالعمل متوالی بعدی را اجرا خواهد کرد. این دستورالعمل، انشعاب برای بازگشت و وارسی مجدد بیت پرچم است. چون دستگاه (وسیله) ورودی، خیلی آهسته‌تر از کامپیوتر است، دو دستورالعمل حلقه، قبل از انتقال یک کاراکتر به داخل انباره، چند بار اجرا خواهند شد.

جدول ۶-۱۹ (ب) لیست دستورات لازم برای چاپ یک کاراکتر را که قبلاً در حافظه ذخیره شده نشان می‌دهد. ابتدا کاراکتر در AC بار می‌شود. سپس پرچم خروجی چک می‌گردد. اگر پرچم ۰ باشد، کامپیوتر در حلقه مشکل از دو دستور باقی می‌ماند و بیت پرچم را چک می‌نماید. وقتی که پرچم به ۱ نغیری یابد، کاراکتر از انباره به چاپگر منتقل می‌گردد.

### دستکاری کاراکتر

یک کامپیوتر صرفاً یک ماشین حساب نیست بلکه دستکاری کننده سمبیل‌ها نیز هست. کاراکترهای کد شده به دودویی که سمبیل‌ها را می‌سازند می‌توانند بواسیله دستورالعمل‌های کامپیوتر برای انجام انواع کارهای داده پردازی دستکاری شوند. یکی از این کارها فشرده کردن دو کاراکتر در یک کلمه است. این کار ساده است زیرا هر کاراکتر 8 بیت را اشغال می‌کند و یک کلمه حافظه از 16 بیت تشکیل شده است. برنامه لیست شده در جدول ۶-۲۰ عبارتست از زیرروالی بنام IN2 که دو کاراکتر را وارد کرده و آنها را در یک کلمه فشرده می‌سازد. کلمه فشرده شده در انباره باقی می‌ماند. توجه کنید که زیر روال SH4 (جدول ۶-۱۶) برای هشت بار شیفت به چپ انباره دوبار فراخوانی شود.

در بحث اسمبیل فرض شد که برنامه سمبیلیک در بخشی از حافظه که گاهی بافر<sup>۳</sup> نامیده می‌شود، ذخیره می‌گردد. برنامه سمبیلیک تایپ شده از طریق وسایل ورودی داخل شده و در مکان‌های متوالی حافظه در بافر ذخیره می‌گردد. برنامه لیست شده در جدول ۶-۲۱ می‌تواند برای ورود برنامه سمبیلیک از صفحه کلید و فشرده کردن 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 می‌شود و سایر کاراکترها در مکان‌های متوالی ذخیره می‌شوند. برنامه از اشاره‌گر برای دنبال کردن مکان‌های خالی موجود در بافر استفاده می‌کند. در برنامه از شمارنده استفاده نشده است، بنابراین مادامی که کاراکترها موجود باشند و یا تا رسیدن بافر به 0 (پس از FFFF) خوانده می‌شوند. در شرایط عملی ممکن است محدود کردن ظرفیت بافر لازم بنظر برسد و از یک

جدول ۶-۲۰ زیرروالی برای دریافت و فشرده کردن دو کاراکتر

IN2,	—	محل ورود به زیرروال /
FST,	SKI	
	BUN FST	
	INP	دریافت اولین کاراکتر /
	OUT	
	BSA SH4	چهار بار شیفت به چپ /
	BSA SH4	چهار بار شیفت به چپ /
SCD,	SKI	
	BUN SCD	
	INP	دریافت دومین کاراکتر /
	OUT	
	BUN IN2 I	بازگشت /

جدول ۶-۲۱ برنامه ذخیره کاراکترهای ورودی در یک بافر

LOP,	LDA ADS	بارگذاردن آدرس ابتدای بافر /
	STA PTR	آغاز اشاره گر /
	BSA IN2	ہوش به زیرروال 2 (جدول ۶-۲۰) /
	STA PTR I	ذخیره کلمه دو کاراکتری در بافر /
	ISZ PTR	افزایش اشاره گر /
	BUN LOP	انشعاب برای دریافت کاراکترهای دیگر /
	HLT	
ADS,	HEX 500	آدرس ابتدای بافر /
PTR,	HEX 0	محل اشاره گر /

شمارنده برای این منظور استفاده گردد. توجه داشته باشید که زیر روال IN2 جدول ۶-۲۰ برای وارد کردن و فشرده کردن جفت کاراکترها فراخوانی می شود. در بحث مربوط به مرور دوم اسمابلر در بخش ۶-۴، گفته شد که یکی از متداول ترین عملیات یک اسمابلر جستجوی جدول<sup>۱</sup> است. این عملی است که طی آن یک جدول برای یافتن یک سمبول مورد نظر جستجو می شود. جستجو ممکن است، مقایسه سمبول مفروض با هر یک از سمبول های ذخیره شده در جدول باشد. وقتی که تطابقی رخ دهد و یا اینکه هیچ یک از سمبول ها با آن مطابقت نداشته باشند جستجو خاتمه می یابد. اگر تطابق اتفاق بیفتند، اسمابلر کد دودویی معادل را استخراج می کنند. برای مقایسه دو کلمه برنامه ای در جدول ۶-۲۲ 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 روالي در یک برنامه جستجوی جدول مورد استفاده قرار داد.

## 1- Table look up

## جدول ۶-۲۲ برنامه مقایسه دو کلمه

LDA WD1	بارگردان کلمه اول /
CMA	
INC	نشکل منم 2 /
ADD WD2	اضافه کردن کلمه دوم /
SZA	گذر اگر AC صفر باشد /
BUN UEQ	انتعاب به روال "اگر نامساوی " /
BUN EQL	انتعاب به روال "اگر مساوی " /
WD1, WD2,	—

برنامه وقفه<sup>۱</sup>

زمان اجرای برنامه های ورودی و خروجی عمدهاً مربوط به زمانی است که کامپیوتر با صرف آن، منتظر شدن پرچم وسیله خارجی می باشد. این اتلاف زمانی ناشی از حلقة انتظاری است که عمل چک کردن پرچم را انجام می دهد و لذا کامپیوتر را به کاری که جز اتلاف وقت زیادی نیست مشغول نگه میدارد. این زمان انتظار می تواند حذف شود بشرطی که برای اطلاع دادن ۱ شدن پرچم از قابلیت وقفه استفاده شود. مزیت استفاده از وقفه این است که انتقال اطلاعات با درخواست از سوی وسیله خارجی آغاز می شود. در این احوال کامپیوتر می تواند 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 می تواند همان چک کردن پرچم ها را انجام دهد. قابلیت وقفه در محیط های چند برنامه ای که دو یا چند برنامه در حافظه مقیم آند مفید است.

در هر لحظه از زمان، حتی اگر چند برنامه در حافظه مقیم باشد، تنها یک برنامه می تواند اجرا شود. برنامه ای که در حال حاضر در حال اجرا شدن است، برنامه جاری خوانده می شود. سایر برنامه ها معمولاً منتظر داده های ورودی یا خروجی هستند. کار وقفه این است که در حین اجرای برنامه، به انتقال داده های برنامه دیگر بپردازد. برنامه جاری باید شامل یک دستور ION برای فعال کردن وقفه ها 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 می اندازد).

قابلیت وقفه اجازه می دهد تا برنامه جاری به کار خود ادامه دهد تا اینکه وسیله داخلی یا خارجی پرچم خود را ۱ نماید. هر وقت پرچمی ۱ شد، کامپیوتر اجرای دستور جاری را تمام کرده و به وقفه پاسخ<sup>۲</sup> می دهد. نتیجه این عمل این است که آدرس بازگشت در مکان ۰ ذخیره می گردد. سپس دستور العمل در مکان ۱ اجرا می شود؛ باین ترتیب یک روال سرویس دهی برای انتقال ورودی یا خروجی آغاز می گردد. روال سرویس دهی در هر جا از حافظه می تواند ذخیره شود بشرطی که

- 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- وارسی اینکه کدام پرچم 1 است
  - ۳- سرویس دهی به وسیله‌ای 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 خطأ 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 انتهای نیز بازگردانده شوند. ترتیب وارسی پرچم‌ها اولویت هر وسیله را تعیین می‌کند. اگرچه ممکن است دو یا چند پرچم بطور همزمان نشانده<sup>۱</sup> شوند، ولی در هر زمان به یک وسیله سرویس داده می‌شود. ترتیب سرویس دهی بر اساس اولویت وسائل است.

وقوع یک وقفه مانع وقوع وقفه‌های دیگر می‌گردد. روال سرویس باید قبل از بازگشت به برنامه جاری وقفه‌ها را فعال کند. در نتیجه کامپیوتر حین اجرای برنامه جاری قادر خواهد بود وقفه‌های بعدی را پیذیرد. قابلیت وقفه تباید تا زمانی که آدرس بازگشت در شمارنده برنامه قرار نگرفته است فعال شود. مثالی از یک برنامه که به یک وقفه سرویس می‌دهد در جدول ۶-۲۳ آرائه شده است. مکان 0 برای آدرس بازگشت نگهداشته 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 وقفه‌ها را فعال می‌کند. فرض کنید در حالی که کامپیوتر دستورالعمل واقع در مکان 103 را اجرا می‌کند وقفه‌ای رخ دهد. سیکل وقفه معادل دو دویی عدد شانزده شانزده 104 را در مکان 0 ذخیره کرده و به مکان 1 انشعاب می‌نماید. دستورالعمل انشعاب در مکان 1، کامپیوتر را به روال SRV می‌فرستد.

روال سرویس دهی شش کار ذکر شده فوق را اجرا می‌نماید. محتوای AC و E در مکان‌های خاصی ذخیره می‌شوند. (این مکان‌ها در کامپیوتر پایه فقط محتوای ثبات های پردازنده هستند. پرچم‌ها متواالیاً وارسی می‌شوند 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 می‌شود. قبل از بازگشت به برنامه جاری محتوای قبلی E و AC بازیابی شده و قابلیت وقفه فعال می‌گردد. آخرین دستور، انشعابی را به آدرس ذخیره شده در مکان 0 موجب می‌شود. این آدرسی است که قبلاً در حین وقفه ذخیره شده است. از این‌رو برنامه جاری

1- Set

## جدول ۶-۲۳ برنامه سرویس دهن به یک وقه

مکان	
0	ZRO, —
1	BUN SRV
100	CLA
101	ION
102	LDA X
103	ADD Y
104	STA Z
•	•
•	•
200	SRV, STA SAC
	CIR
	STA SE
	SKI
	BUN NXT
	INP
	OUT
	STA PT1 I
	ISZ PT1
NXT,	SKO
	BUN EXT
	LDA PT2 I
	OUT
	ISZ PT2
EXT,	LDA SE
	CIL
	LDA SAC
	ION
	BUN ZRO I
SAC,	—
SE,	—
PT1,	—
PT2,	—

از مکان 104 به کار آدame خواهد داد، یعنی به همان جایی که در آن وقه رخ داده بود.

در یک نمونه کامپیوترا مکان دارد چندین وسیله ورودی یا خروجی به مدار وقه متصل شده باشد. بعلاوه، منابع وقه محدود به وسائل ورودی و خروجی محدود نیستند. وقہ ها را می توان برای اهداف دیگری همچون خطاهای پردازشی درونی یا اختوارهای ویژه بکار برد. بحث های بیشتر درباره وقہ ها و برخی مفاهیم پیشرفتی که اهمیت موضوع را بیان کنند در بخش ۱۱-۵ آورده شده است.

### مسائل

۶-۱ برنامه زیر در واحد حافظه کامپیوتر پایه ذخیره شده است. محتوای AC، PC و IR را به شائزده شائزدهی در پایان اجرای هر دستورالعمل نشان دهید. تمام اعداد در جدول به شائزده شائزدهی 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 شائزده است. کامپیوتر با شروع از آدرس 100 دستورالعمل‌ها را اجرا می‌کند. 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 اسمبلي تولید شده بوسیله یک کامپایلر را از برنامه فرتون زیر بنویسید. متغیرها را عدد صحیح تصویر کنید.

$$\text{SUM} = 0$$

$$\text{SUM} = \text{SUM} + A + B$$

$$\text{DIF} = \text{DIF} - C$$

$$\text{SUM} = \text{SUM} + \text{DIF}$$

۶-۴ آیا از حرف A می‌توان بعنوان یک آدرس سمبلیک در برنامه زبان اسمبلي برای کامپیوتر پایه استفاده کرد؟ توضیح دهید.

۶-۵ اگر سطری از کد که حاوی شبه دستورالعمل END ORG یا END است عنوان نیز داشته باشد در مرور اول اسمبیلر (شکل ۱-۶) چه رخداد دارد. فلوچارت را تغییر دهید تا در این صورت پیغام خطأ بدهد.

۶-۶ سطری از کد در یک برنامه زبان اس梅لی بصورت زیر است:

DEC – 35

(الف) نشان دهد که برای ذخیره سطر کد به چهار کلمه حافظه نیاز است و محتوای دودویی آنها را بنویسید.

(ب) نشان دهد که یک کلمه حافظه کد ترجمه شده دودویی را ذخیره می نماید. محتوای دودویی آنها را بنویسید.

۶-۷ (الف) جدول سمبول آدرس تولید شده برای برنامه جدول ۱۲-۶ در حین مرور اول بوسیله اس梅لر را بدست آورید.

(ب) برنامه ترجمه شده را به شانزده شانزدهی بنویسید.

۶-۸ شبیه دستورالعمل BSS (بلاک آغاز شده بوسیله سمبول) گاهی برای رزرو N کلمه حافظه برای گروهی از عملوندها به کار می رود. مثلًا سطر کد

A, BSS 10

به اس梅لر اطلاع می دهد که بلاکی از 10 (دهدهی) مکان را، با شروع از مکان متناظر با سمبول A خالی بگذارد. این مشابه جمله (10) DIMENSION A(10) در فرترن است. فلوچارت شکل ۱-۶ را برای پردازش این شبیه دستور تصحیح کنید.

۶-۹ فلوچارت شکل ۱-۶-۲ را تصحیح کنید بطوری که اگر یک دستور اس梅لیک با عنوان تعریف نشده باشد یک پیام خطاط تولید شود.

۶-۱۰ نشان دهد که چگونه جداول MRI و non-MRI می توانند در حافظه ذخیره شوند.

۶-۱۱ لیست برنامه زبان اس梅لی (معادل دودویی دستورالعمل ها) 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 120 اگر  $A - B = 0$  باشد؛ و یا به 130 اگر  $A - B > 0$  باشد.

۶-۱۲ (الف) توضیح دهد 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 سمبول های آدرس بدست آمده در مرور اول اسمبلر را بنویسید.

ج) کد شاتزده شانزدهی برنامه ترجمه شده را بنویسید.

۶-۱۳ یک حلقه برنامه با استفاده از اشاره گر و شمارنده بنویسید تا محتویات مکان شاتزده شانزدهی ۵FF را ۰۵۰ کند.

۶-۱۴ برنامه ای بنویسید تا دو عدد را با روش جمع تکراری، در هم ضرب کند. مثلاً برای ضرب  $5 \times 5$  برنامه ۵ را چهار بار با هم جمع کند، یعنی  $5 + 5 + 5 + 5$ .

۶-۱۵ برنامه ضرب جدول ۶-۱۴ مقدار دهنی اولیه نشده است. پس از یکبار اجرای برنامه، مکان CTR مقدار صفر می گیرد. نشان دهید که اگر برنامه مجدداً از مکان ۱۰۰ شروع شود، حلقه ۶۵۵۳۶ بار تکرار خواهد شد. دستورات لازم برای مقدار دهنی اولیه<sup>۱</sup> را برای برنامه اضافه کنید.

۶-۱۶ برنامه ای برای ضرب دو عدد مشتت بدون علامت ۱۶ بیتی بنویسید و حاصل ضرب 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 ۲ علامت دار نشان داده می شوند. حاصل ضرب باید با دقت منفرد باشد و در صورت منفی بودن هم بصورت متمم ۲ علامت دار نشان داده شود.

۶-۱۸ برنامه ای بنویسید که دو عدد با دقت مضاعف را از هم تفريق کند.

۶-۱۹ برنامه ای بنویسید که OR انحصاری منطقی دو عملوند منطقی را بدست آورد.

۶-۲۰ برنامه ای برای شيفت حسابی به چپ بنویسید که در صورت سرريز به OVF انشعاب کند.

۶-۲۱ برنامه ای برای تفريق دو عدد بنویسید. در برنامه فراخواننده، مفروق و مفروق منه بدنیال دستور العمل 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 يك بلاک متمم کند. در برنامه فراخواننده، دو پارامتر

#### 1- Initialize

- آدرس شروع بلاک و تعداد کلمات بلاک پس از دستور العمل BSA آمده‌اند.
- ۶-۲۳ برنامه‌ای بنویسید که E و AC را چهار بار به راست بچرخاند. اگر ابتدا محتوای AC برابر E باشد محتوای آنها پس از اجرای زیر روال چیست؟
- ۶-۲۴ برنامه‌ای بنویسید که کاراکترهای ورودی را پیذیرد، هر دو کاراکتر بصورت یک کلمه فشرده کند و متواالیاً در بافر حافظه ذخیره نماید. اولین آدرس بافر ۱۶(400) و اندازه آن ۵۱۲(16) 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 ۰ تا ۷ از مکان‌های CH1 و CH2 ذخیره کند. بیت‌های ۹ تا ۱۵ باید صفر باشد.
- ۶-۲۶ فلوچارتی 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 ۰ تا ۷ مکان LNE منتقل می‌سازد بدون آن که بیت‌های ۸ تا ۱۵ تغییر کنند.
- ۶-۲۷ روال سرویس دهی SRV را از جدول ۶-۲۳ به کد شانزده شانزدهی معادلش ترجمه کنید. فرض کنید که روال از مکان 200 شروع شده باشد.
- ۶-۲۸ یک روال سرویس دهی وقفه که تمام عملیات لازم را انجام دهد بنویسید ولی وسیله ورودی فقط اگر مکان MOD تماماً ۱ باشد سرویس دهی شود. وسیله خروجی هم فقط اگر مکان MOD تماماً ۰ باشد سرویس دهی 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 دناله هایی از ریز عمل هاست. تعداد انواع ریز عمل های موجود در یک سیستم معین محدود است. پیچیدگی سیستم دیجیتال به تعداد ریز عمل های متالی که اجرا می کند بستگی دارد. هنگامی که سیگنال های کنترلی توسط سخت افزار حاصل از تکنیک های مدار های منطقی رایج تولید شود، گوییم واحد کنترل سخت افزاری<sup>۱</sup> است. ریز برنامه نویسی روش دیگری برای طراحی واحد کنترل یک کامپیوتر دیجیتال است. اصل ریز برنامه نویسی رو شی شاخص و سیستماتیک در کنترل رشته ریز عمل ها در یک کامپیوتر دیجیتال است. تابع کنترلی که مشخص کننده یک ریز عمل است یک متغیر دودویی می باشد. وقتی که این متغیر در یکی از حالات دودویی باشد، ریز عمل متناظر آن اجرا می شود. اگر متغیر کنترلی که در وضعیت دودویی مخالف باشد وضعیت ثبات های سیستم را تغییر نمی دهد. وضعیت فعال یک متغیر کنترل ممکن است بسته به کاربرد هر یک از دو حالت ۰ یا ۱ باشد. در سیستم مبتنی بر گذرگاه<sup>۲</sup>، سیگنال های کنترلی که ریز عمل را مشخص می نمایند بصورت گروه هایی از بیت ها هستند که مسیرها را در مولتی پلکسرها، دیکدرها و واحد های حساب و منطق انتخاب می نمایند.

واحد کنترل موجب اجرای یک سری مراحل متوالی از ریز عمل ها می شود. در هر زمان مفروض،

1- Hardware = Hardwired      2- Bus Organized System

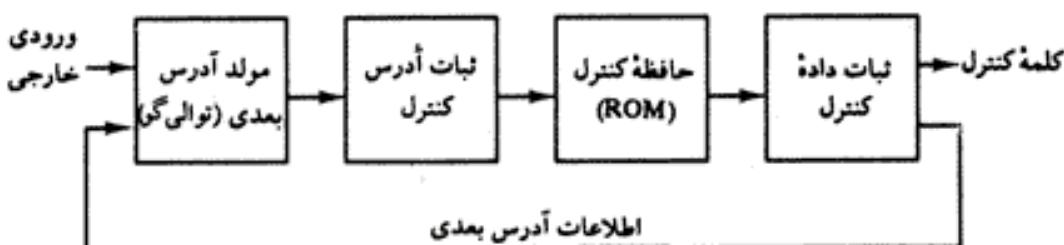
ریز عمل های معینی باید اجرا شوند، در حالیکه سایر ریز عمل ها را کد هستند. متغیر های کنترلی در هر لحظه از زمان می توانند با رشته ای از ۱ ها و ۰ ها که کلمه کنترل نامیده می شود نمایش داده شوند. باین ترتیب می توان با برنامه نویسی کلمات کنترلی، عملیات مختلف را روی اجزاء سیستم انجام داد. واحد کنترلی که متغیر های دودویی کنترل آن در حافظه ذخیره شده باشد واحد کنترل ریز برنامه نویسی شده نام دارد. هر کلمه در واحد کنترل حاوی یک ریز دستورالعمل است. ریز دستورالعمل مشخص کننده یک یا چند ریز عمل برای سیستم است. رشته ای از ریز دستورالعمل ها، ریز برنامه را تشکیل می دهند. با توجه به اینکه پس از نصب واحد کنترل در سیستم لازم نیست تغییری در ریز برنامه داده شود، حافظه کنترل می تواند از نوع فقط خواندنی<sup>۱</sup> (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 ریز برنامه بمعنی در اختیار قرار دادن تمام متغیر های کنترل در کلمات ROM برای استفاده واحد کنترل از طریق خواندن های متوالی است.

محتوای کلمه در ROM در یک آدرس مشخص یک ریز دستورالعمل را مشخص می کند.

در روش پیشرفته تری که برنامه نویسی دینامیکی نام دارد این امکان وجود دارد که ریز برنامه ابتدا از طریق یک حافظه کمکی مانند دیسک مغناطیسی به سیستم بار شود. واحد های 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 اکثرآ برای خواندن مورد استفاده قرار می گیرند. حافظه ای که بخشی از واحد کنترل باشد حافظه کنترل<sup>۲</sup> نامیده می شود.

کامپیوتري که از واحد کنترل ریز برنامه 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 ریز برنامه ثابتی را نگهداری می کند و بوسیله هر کاربری قابل تغییر نیست. ریز برنامه از ریز دستورالعمل هایی تشکیل شده که بنویه خود انواع سیگنال های کنترل را برای اجرای ریز اعمال روی ثبات ها معین می کنند. هر دستورالعمل ماشین موجب اجرای رشته ای از ریز دستورالعمل های موجود در حافظه کنترل می شود. وظیفه این ریز دستورالعمل ها برداشت<sup>۳</sup> 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 ریز برنامه 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 ریز دستورالعمل را مشخص می کند، و ثبات داده کنترل به



شکل ۱-۷ سازمان کنترل ریزبرنامه نویسی شده

ریزدستورالعمل خوانده شده از حافظه را نگه میدارد. ریزدستورالعمل حاوی یک کلمه کنترل است که یک یا چند ریز عمل را برای پردازشگر داده مشخص می‌نماید. 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. با این دلیل لازم است تعدادی از بیت‌های ریزدستورالعمل فعلی را برای کنترل ایجاد آدرس ریزدستورالعمل بعدی بکار برد. آدرس بعدی می‌تواند 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 می‌گردد تا ریزدستورالعمل بعدی را بخواند. بنابراین یک ریزدستورالعمل حاوی بیت‌هایی برای آغاز ریز اعمال در بخش پردازشگر داده و بیت‌هایی برای تعیین رشته آدرس‌ها برای حافظه کنترل است.

مولد آدرس بعدی گاهی اوقات توالی گر<sup>۱</sup> ریزبرنامه نامیده می‌شود، زیرا آدرس‌های متوالی کلماتی را که از حافظه کنترل خوانده می‌شوند معین می‌سازد. آدرس ریزدستورالعمل بعدی بسته به ورودی توالی گر به چند طریق معین می‌شود. عملیات توالی گر ریزبرنامه عمده‌تاً عبارتست از افزایش ثبات آدرس کنترل به میزان یک واحد، بارگردان آدرس حافظه کنترل در ثبات آدرس کنترل، انتقال آدرس خارجی، یا بارگردان یک آدرس اولیه برای آغاز اعمال کنترل.

ثبات داده کنترل در حالی که آدرس بعدی محاسبه و از حافظه خوانده می‌شود، حاوی دستورالعمل فعلی است. ثبات داده گاهی اوقات ثبات خط لوله<sup>۲</sup> هم خوانده می‌شود. این ثبات امکان اجرای ریزاعمال مشخص شده با کلمه کنترل، همزمان با تولید ریزدستورالعمل بعدی را فراهم می‌آورد. (برای توضیح بیشتر در مورد ثبات خط لوله به فصل‌های 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 می‌شود.

سیستم می‌تواند با اعمال یک ساعت تک‌فاز به ثبات آدرس، بدون ثبات داده کنترل کارکند. کلمه کنترل و اطلاعات آدرس بعدی مستقیماً از حافظه کنترل دریافت می‌شود. باید دانست که ROM بعناند یک مدار ترکیبی کار می‌کند، که در آن مقدار آدرس ورودی و کلمه مربوط به آن آدرس خروجی آن است.

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؛ توالی گر و حافظه کنترل مدارهای ترکیبی بوده و به پالس ساعت نیازی ندارند.

مزیت عمده کنترل ریزبرنامه نویسی شده این است که وقتی پیکربندی سخت‌افزاری تکمیل شد، سخت‌افزار اضافی و یا تغییر سیم‌بندی نخواهیم داشت. اگر بخواهیم 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 ریزبرنامه‌ای است که در حافظه کنترل جای داده می‌شود.

باید یادآوری کرد اکثر کامپیوترهای پایه‌ریزی شده بر مفهوم معماری کامپیوترهای کم‌دستور<sup>۱</sup> یا RISC (به بخش ۸-۸ مراجعه شود) از کنترل سخت‌افزاری استفاده می‌کنند نه از حافظه کنترل ریزبرنامه. مثالی از کنترل سخت‌افزاری برای یک کامپیوتر ساده در بخش ۵-۴ 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 ریزبرنامه 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 ریزبرنامه کنترل، اجازه بدھید مراحلی 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 ریزدستورالعمل است که روال برداشت<sup>۲</sup> دستورالعمل را فعال می‌سازد. اجرای روال برداشت می‌تواند با افزایش ثبات آدرس کنترل تا طی شدن کلیه ریزدستورالعمل‌های 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 انشعباب قابل دسترسی است، که این انشعبابات مشروط بر وضعیت بیت‌های حالت دستورالعمل صورت می‌گیرد. 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 می‌نمایند. مراحل ریزاعمالی که باید در ثبات‌های پردازنده تولیدشوند به کد عمل که بخشی از دستورالعمل است بستگی دارد. به هر دستورالعمل روال ریزبرنامه خاصی تعلق می‌گیرد که در مکان معینی از حافظه کنترل ذخیره شده است. تبدیل بیت‌های کد دستورالعمل به یک آدرس در حافظه کنترل که روال در آن قرار دارد نگاشت<sup>۱</sup> نامیده می‌شود. رویه نگاشت قاعده‌ای است که کد دستورالعمل را به آدرس حافظه کنترل تبدیل می‌کند. هنگامی که به روال مورد نظر دسترسی پیدا شد، ریزدستورالعمل‌هایی که دستورالعمل را اجرا می‌کند می‌توانند با افزایش ثبات آدرس کنترل طی شوند، اما گاهی هم توالی ریزدستورالعمل‌ها به مقادیر معینی از بیت‌های وضعیت در ثبات‌های پردازشگر بستگی دارد. ریزبرنامه‌هایی که از زیرروال‌ها استفاده می‌کنند به یک ثبات خارجی برای نگهداری آدرس بازگشت نیاز دارند. آدرس‌های بازگشت نمی‌توانند در ROM ذخیره شوند زیرا ROM دارای قابلیت نوشتند نیست.

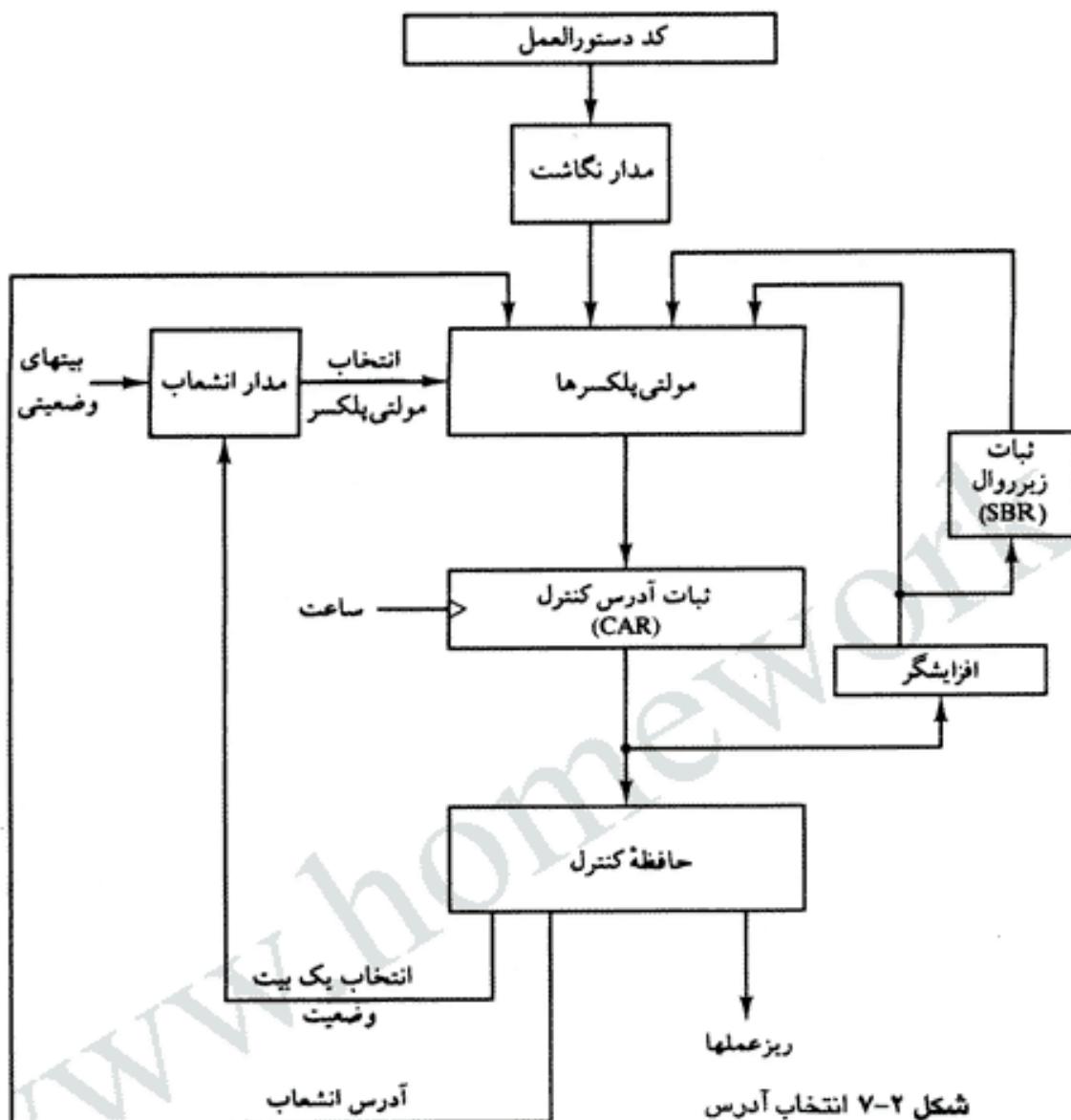
هنگامی که اجرای دستورالعملی پایان یابد، کنترل باید به روال برداشت (واکشی) بازگردد. این عمل با اجرای یک ریزدستورالعمل انشعباب غیرشرطی به ابتدای روال برداشت تحقق می‌یابد. بطور خلاصه، قابلیت‌های لازم برای توالی آدرس در حافظه کنترل عبارتند از:

- ۱- افزایش ثبات آدرس کنترل
- ۲- انشعباب غیرشرطی و یا شرطی، بسته به شرایط بیت‌های وضعیت
- ۳- فرآیند نگاشت (تبدیل) از بیت‌های دستورالعمل به آدرس حافظه کنترل
- ۴- امکاناتی برای فراخوانی زیرروال و بازگشت از آن

شکل ۷-۲ بلاک دیاگرام یک حافظه کنترل و سخت‌افزار لازم برای انتخاب آدرس ریزدستورالعمل بعدی را نشان می‌دهد. ریزدستورالعمل واقع در حافظه کنترل متشکل از مجموعه‌ای از بیت‌های است که ریزاعمال را در ثبات‌های کامپیوتر موجب می‌شود و نیز مجموعه‌ای از بیت‌های دیگر برای تهیه آدرس بعدی است. دیاگرام چهار مسیری که توسط آنها ثبات آدرس کنترل<sup>۲</sup> (CAR) آدرس را دریافت می‌نماید، نشان می‌دهد. افزایشگر محتوای ثبات آدرس کنترل را یک واحد افزایش می‌دهد تا ریزدستورالعمل بعدی را انتخاب نماید. عمل انشعباب پس از مشخص شدن آدرس انشعباب در یکی از میدان‌های ریزدستورالعمل انجام می‌شود. انشعباب شرطی با بکارگیری بخشی از ریزدستورالعمل بخاطر انتخاب بیت وضعیت معینی برای مشخص نمودن شرایط انشعباب، صورت می‌گیرد. آدرس خارجی (که بخشی از دستورالعمل است) از طریق مدار منطقی نگاشت به کنترل حافظه منتقل می‌شود. 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 می‌گیرد ذخیره می‌شود.

### انشعب شرطی

منطق انشعب شکل ۷-۲ قابلیت تصمیم‌گیری را در واحد کنترل فراهم می‌آورد. حالت وضعیت سیستم، بیت‌های خاصی هستند که اطلاعات پارامتری همچون، نقلی خروجی یک جمع کننده، بیت علامت یک عدد، بیت‌های روش<sup>۱</sup> (شیوه) در یک دستورالعمل، شرایط وضعیتی ورودی یا

1- Mode bit

خروجی<sup>۱</sup> را فراهم می‌کنند. اطلاعات موجود در این بیت‌ها قابل تست بوده و تصمیم اتخاذ شده می‌تواند بر اساس حالت آنها یعنی ۰ یا ۱ باشد. بیت‌های وضعیت، همراه با میدانی در ریزدستورالعمل که آدرس انشعباب را مشخص می‌نماید، تصمیمات انشعباب‌های شرطی را که در مدار انشعباب تولید می‌شود کنترل می‌نمایند.

سخت‌افزار مدار انشعباب بطرق مختلفی قابل پیاده‌سازی است. در ساده‌ترین راه، شرط مشخص شده و انشعباب 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 جماعت 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 تشیکل می‌دهند. اگر بیت وضعیت انتخاب شده در حالت ۱ باشد، خروجی مولتی‌پلکسر ۱ است؛ در غیر اینصورت ۰ است. خروجی ۱ در مولتی‌پلکسر یک سیگنال کنترل برای انتقال آدرس انشعباب از ریزدستورالعمل به ثبات آدرس کنترل تولید می‌کند. خروجی ۰ 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 انشعباب غیرشرطی را می‌توان با بارگردان آدرس انشعباب 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 ۱ انجام داد. ارجاع به این بیت توسط خطوط انتخاب از حافظه کنترل موجب می‌شود تا آدرس انشعباب در ثبات آدرس کنترل بدون هر شرطی بار شود.

### نگاشت<sup>۲</sup> دستورالعمل

نوع خاصی از انشعباب هنگامی است که یک ریزدستورالعمل حکم کند که به اولین کلمه در حافظه کنترل که روای ریزبرنامه مربوط به یک دستورالعمل در آنجا واقع است انشعباب شود. بیت‌های وضعیت برای این نوع انشعباب، بیت‌های بخش کد عملیات در دستورالعمل هستند. مثلاً، کامپیوترا با قالب

کد عمل		آدرس	دستورالعمل‌های کامپیوترا:
۱	۰ ۱ ۱		
۰	× × × ×	۰ ۰	نگاشت بیت‌ها:
۰	۱ ۰ ۱ ۱	۰ ۰	آدرس ریزدستورالعمل:

شکل ۷-۳ نگاشت (تبديل) 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 ۱۶ دستورالعمل متمایز را تشخیص دهد. فرض کنید حافظه کنترل ۱۲۸ 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 ۴ بیتی را به آدرس ۷ بیتی برای حافظه کنترل تبدیل می‌کند در شکل ۳-۷ نشان داده شده است. این نگاشت بدین صورت انجام می‌شود که یک ۰ در پارزشترین مکان آدرس قرار داده می‌شود، چهار بیت کد عملیات منتقل می‌گردد، و دو بیت کم ارزشتر 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 ۱۰۰۰۰۰۰ تا ۱۱۱۱۱۱۱ 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 قرار گیرد. 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 پذیر<sup>۱</sup> 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.

## زیر روال‌ها<sup>۲</sup>

زیر روال‌ها 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 مؤثر 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 مؤثر 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<sup>۱</sup> از نوع آخرین ورودی - اولین خروجی<sup>۲</sup> (LIFO) می‌باشد. استفاده از پشته در یک زیر روال فراخوانی و بازگشت از زیرروال‌ها در بخش ۷-۸ مفصل‌اً شرح داده شده است.

### ۷-۳ مثال ریزبرنامه

وقتی که آرایش یک کامپیووتر و واحد کنترل ریزبرنامه‌نویسی شده آن مسجل شد، وظیفه طراح، تولید ریز کد برای حافظه کنترل است. این تولید کد را ریز برنامه‌نویسی می‌نامند که مشابه برنامه‌نویسی معمولی بزبان ماشین است. برای درک بهتر این فرآیند، در اینجا یک کامپیووتر دیجیتال ساده را ارائه داده و نشان می‌دهیم. کامپیووتر مورد بحث در این مثال مشابه کامپیووتر پایه معرفی شده در فصل ۵ است ولی دقیقاً همانند آن نیست.

### آرایش کامپیووتر

بلک دیاگرام کامپیووتر مورد نظر در شکل ۷-۴ نشان داده شده است. این کامپیووتر 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<sup>۳</sup>، PC، ثبات آدرس AR، ثبات داده DR و ثبات انباره AC. کار این ثبات‌ها مشابه کامپیووتر پایه‌ای است که در فصل ۵ معرفی شد (شکل ۵-۳ ملاحظه شود). واحد کنترل دارای ثبات آدرس کنترل CAR و یک ثبات زیرروال ISBR است. حافظه کنترل و ثبات‌های متعلق به آن، همان‌طور که در شکل ۷-۲ دیده می‌شود، بصورت یک واحد کنترل ریزبرنامه‌نویسی شده شناخته می‌شوند.

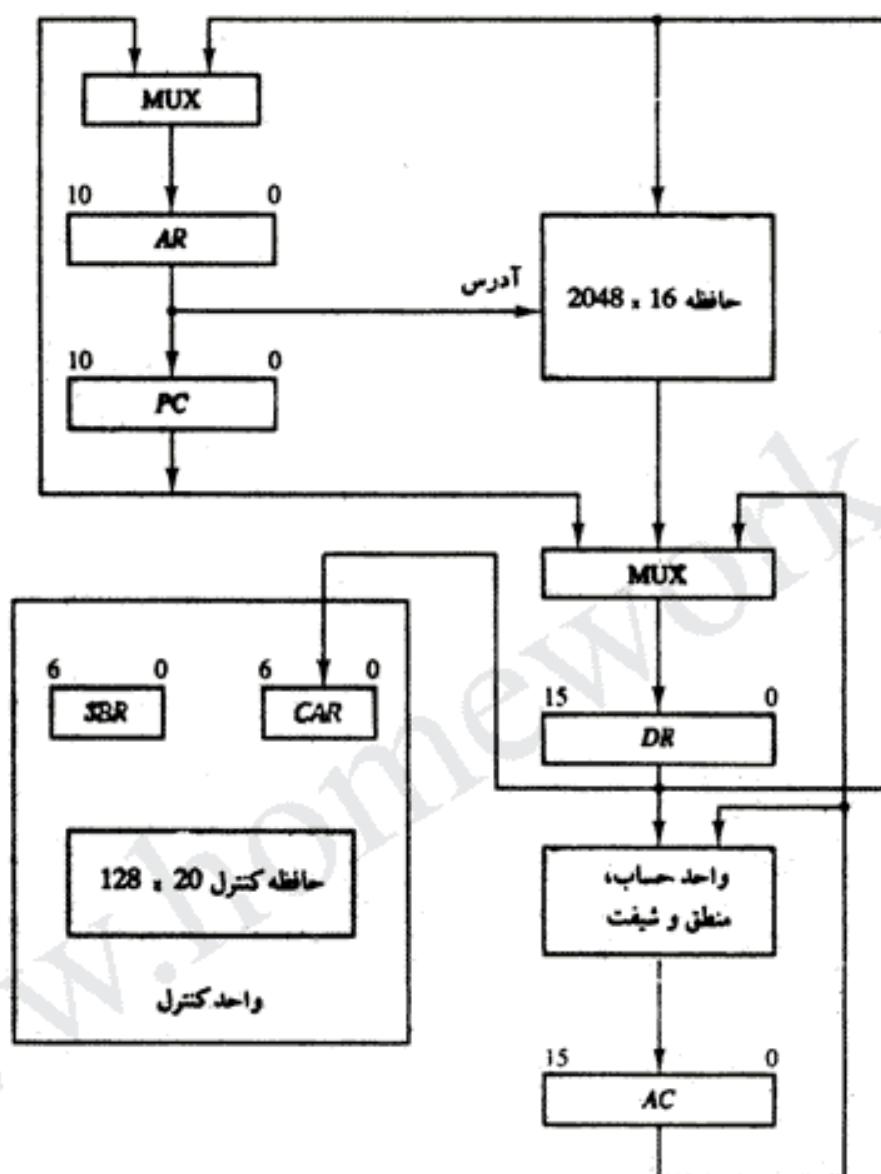
انتقال اطلاعات بین ثبات‌های پردازنده از طریق مولتی‌پلکسر صورت می‌گیرد و نه با استفاده از گذرگاه مشترک. DR می‌تواند اطلاعات را از AC، PC و یا حافظه دریافت کند. PC فقط از AR اطلاعات دریافت می‌کند. واحد حساب، منطق و شیفت ریزاعمال را برروی داده‌های دریافتی از AC و DR انجام داده و حاصل را در AC قرار می‌دهد. وقتی 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 کامپیووتر در شکل ۷-۵ (الف) نشان داده شده است. این قالب از سه میدان تشکیل شده است: یک میدان 1 بیتی برای آدرس دهنی غیرمستقیم که با I مشخص شده است، یک کد

1- stack

2- Last In First Out

3- Program Counter



شکل ۷-۴ پیکربندی سخت‌افزار کامپیووتر

عملیات ۱۴ بیتی (opcode) و یک میدان آدرس ۱۱ بیتی. شکل ۷-۵ (ب) چهار دستور از شانزده دستورالعمل‌های ارجاع به حافظه را نشان می‌دهد. دستورالعمل ADD، عملوند موجود در آدرس مؤثر را با محتوای AC جمع می‌کند. دستورالعمل BRANCH، انشعابی را به آدرس مؤثر موجب می‌شود بشرطی که عملوند درون AC منفی نباشد. اگر AC منفی نباشد برنامه با دستورالعمل بعدی ادامه می‌یابد. AC در صورتی منفی است که بیت علامت (بیت مسنه‌ایه سمت چپ ثبات) ۱ باشد. دستورالعمل STORE محتوای AC را به کلمه‌ای از حافظه که آدرس مؤثر آن را مشخص می‌کند انتقال

## 1- Operation Code

15	14	11	10	0
I	کد عمل			آدرس

(الف) قالب دستورالعمل

توضیح	کد عمل	سبل
$AC \leftarrow AC + M[EA]$	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

آدرس مؤثر است EA

(ب) چهار دستورالعمل کامپیووتر

شکل ۷-۵ دستورالعمل‌های کامپیووتر

می‌دهد. دستورالعمل EXCHANGE داده‌هایی را بین AC و کلمه حافظه مشخص شده بوسیله آدرس مؤثر تعویض می‌کند.

در دنباله بحث نشان داده خواهد شد که هر دستورالعمل کامپیووتر باستی ریزبرنامه‌نویسی شود. بخاطر پرهیز از پیچیده شدن مثال ریز برنامه‌نویسی، در اینجا فقط چهار دستورالعمل مورد بررسی قرار گرفته است. باید دانست که 12 دستورالعمل دیگر می‌تواند به مجموعه اضافه شود و هر دستورالعمل باید توسط روند شرح داده شده در ذیل ریز برنامه نویسی گردد.

### قالب ریزدستورالعمل‌ها

قالب ریزدستورالعمل‌ها 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 ریزاعمال را برای کامپیووتر مشخص می‌کنند. میدان CD بیت‌های شرطی وضعیت را انتخاب می‌نماید. میدان BR نوع

3	3	3	2	2	7
F1	F2	F3	CD	BR	AD

Mیدان‌های ریز عمل

CD : شرط انشتاب

BR : میدان انشتاب

AD : میدان آدرس

شکل ۷-۶ قالب کد ریزدستورالعمل (20 بیت)

انشعاب بکار رفته را معین می‌کند. میدان AD آدرس انشعاب را دارد. 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 ۰۰۰ معنی هیچ‌کار استفاده خواهد کرد. مثلاً یک ریزدستور العمل می‌تواند بطور همزمان دو ریزعمل را در F2 و F3 و هیچ عمل را در F1 انجام دهد.

$$DR \leftarrow M[AR] \quad F2 = 100$$

$$PC \leftarrow PC + 1 \quad F3 = 101$$

۹ بیت میدان ریز عملیات عبارت از ۱۰۱ ۱۰۰ ۰۰۰ خواهد بود. توجه به این نکته که نمی‌توان دو یا چند عمل متضاد را به طور همزمان اجرا کرد اهمیت دارد. مثلاً میدان ریز اعمال ۰۰۱ ۰۱۰ ۰۱۰ مفهومی ندارد زیرا این ریز اعمال، عملیاتی مانند پاک کردن AC و تفریق DR از AC را بطور همزمان نشان می‌دهد. هر ریز عمل در جدول ۷-۱ بوسیله یک عبارت انتقال ثبات تعریف شده و سمبولی جهت استفاده در برنامه‌های سمبولیک به آن اختصاص داده شده است. همه سمبول‌های ریز عمل‌های انتقالی از پنج حرف استفاده می‌کنند. دو حرف اول ثبات مبدأ را مشخص می‌کند، سومین حرف همواره T است، و دو حرف آخر ثبات مقصد را معین می‌نماید.

میدان شرط<sup>۲</sup> CD دو بیتی است که برای تعیین چهار بیت وضعیت شرطی طبق جدول ۷-۱ انکد شده است. اولین حالت همیشه ۱ است، ارجاع به ۰۰ = CD (یا سمبول 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 دستور العمل از حافظه می‌توان در بیت ۱۵ 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 ۱ خواهد شد. ما هنگامی که ریزبرنامه‌هایی را به فرم سمبولیک بنویسیم از چهار سمبول 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 ۰۰ = BR = باشد، کنترل عمل پرش (JMP) را اجرا می‌کند (که مشابه عمل انشعاب است)، و هنگامی که ۰۱ = BR = باشد عمل فراخوانی زیرروال (CALL) انجام خواهد شد. دو عمل یکسان هستند با این تفاوت که ریز عمل فراخوانی آدرس

جدول ۷-۱ سمبول‌ها و کدهای دودویی برای میدانهای ریزدستورالعمل‌ها

F1	ریز عمل	سبل
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	سبل
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	سبل
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	سبل	توضیح
00	$1 =$ همیشه	U	انشعاب غیرشرطی
01	$DR(15)$	I	یست آدرس غیرمستقیم
10	$AC(15)$	S	یست علامت AC
11	$AC = 0$	Z	مقدار صفر در AC

BR	سبل	عملکرد
00	JMP	اگر شرط برابر 1 باشد $CAR \leftarrow AD$ اگر شرط برابر صفر باشد 1 $CAR \leftarrow CAR + 1$
01	CALL	اگر شرط برابر 1 باشد $CAR \leftarrow AD, SBR \leftarrow CAR + 1$ اگر شرط برابر 0 باشد 1 $CAR \leftarrow CAR + 1$
10	RET	بازگشت از زیرروال time)
11	MAP	$CAR(2-5) \leftarrow DR(11-14), CAR(0,1,6) \leftarrow 0$

بازگشت را در ثبات زیروال SBR ذخیره می‌کند. اعمال پرش و فراخوانی به میدان CD وابسته هستند. اگر شرایط بیت وضعیت در میدان CD برابر ۱ باشد، آدرس بعدی واقع در میدان AD به ثبات آدرس کنترل CAR منتقل می‌گردد. در غیراینصورت CAR یک واحد اضافه می‌شود. بازگشت از زیروال در صورتی انجام می‌شود که میدان BR برابر ۱۰ 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 ۱۱ باشد. این نگاشت در شکل ۷-۳ 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 سمبولیک

سمبل‌های ارائه شده در جدول ۷-۱ می‌توانند برای مشخص کردن ریزدستورالعمل‌ها به فرم سمبولیک بکارروند. یک ریزبرنامه سمبولیک می‌تواند بکمک اسمبول به معادل دودویی اش تبدیل شود. اسمبول یک ریزبرنامه از نظر مفهوم مشابه اسمبول معمولی در کامپیوتر است که در بخش ۶-۳ تعریف شد. ساده‌ترین و سرراست‌ترین روش تنظیم یک زبان اسمبولی برای یک ریزبرنامه، تعریف سمبول‌ها برای هر میدان ریز دستورالعمل و امکان تعریف آدرس سمبولیک توسط خود کاربران است. هر سطر از ریزبرنامه زبان اسمبولی، یک ریزدستورالعمل سمبولیک را تعریف می‌کند. هر دستورالعمل سمبولیک 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 سمبولیک باشد. هر عنوان با یک علامت نقل قول (: ) خاتمه می‌یابد.
- ۲- میدان ریزعمل‌ها از یک، دو و یا سه سمبول تعریف شده در جدول ۷-۱ تعریف می‌شود و بوسیله کاما از هم جدا می‌شوند. نمی‌توان بیش از یک سمبول از هر میدان F داشت. هرگاه ریزدستورالعمل هیچ عملی نداشته باشد از سمبول NOP<sup>۱</sup> استفاده می‌شود. اسمبول این سمبول را به ۹ عدد ۰ ترجمه می‌کند.
- ۳- میدان CD دارای یک حرف U، I، S و یا Z است.
- ۴- میدان BR حاوی یکی از چهار سمبول تعریف شده در جدول ۷-۱ است.
- ۵- میدان AD مقدار میدان آدرس ریز دستورالعمل را به یکی از سه طریق زیر مشخص می‌کند.
  - (الف) با یک آدرس سمبولیک، که باید بصورت عنوان هم وجود داشته باشد.
  - (ب) با سمبول NEXT برای مشخص کردن آدرس متولی بعدی
  - (ج) اگر میدان BR حاوی سمبول RET، یا MAP باشد، میدان AD خالی گذاشته می‌شود و اسمبول آن را به هفت عدد صفر تبدیل می‌نماید.

1- No Operation

ما از شبه دستور ORG نیز برای تعریف مبداء، یا اولین آدرس یک روال ریزبرنامه استفاده می‌کنیم. مثلاً سمبول 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<sup>۱</sup>

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:

$AR \leftarrow PC$

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

$AR \leftarrow DR(0 - 10), CAR(2 - 5) \leftarrow DR(11 - 14), CAR(0, 1, 6) \leftarrow 0$

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 بخشن 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 زیان اسمبلی که قبلاً تعریف شد، ما می‌توانیم ریزبرنامه سمبولیک را برای

روال برداشت بنویسیم:

ORG 64					
FETCH:	PCTAR	U	JMP	NEXT	
	READ, INCPC	U	JMP	NEXT	
	DRTAR	U	MAP		

ترجمه ریزبرنامه سمبولیک 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 Routine

سه ریز دستورالعمل که روال برداشت را تشکیل می‌دهند به سه فرم مختلف نشان داده شدند. نمایش انتقال ثباتی عمل انتقال بین ثبات‌ها را که هر ریز دستورالعمل پیاده می‌کند نشان می‌دهد. نمایش سمبولیک برای نوشتمن ریز برنامه در قالب زیان اسمبلی مفید است. نمایش دودویی محتوای داخلی واقعی را که باید در حافظه کنترل ذخیره شود نشان می‌دهد. مرسوم است که ریز برنامه‌ها را به شکل سمبولیک بنویسند و سپس با استفاده از یک برنامه اسمبلر ترجمه دودویی آن را بدست آورند.

### ریز برنامه سمبولیک

اجرای سومین ریز دستورالعمل در روال برداشت (MAP) سبب می‌شود به آدرس 0xxxxx00 انشعباب شود که در آن xxxx چهار بیت کد عملیات است. مثلاً اگر دستورالعمل ADD باشد که کد عمل آن 0000 است، ریز دستورالعمل MAP آدرس 00000000 که آدرس شروع روال ADD در حافظه کنترل می‌باشد را به CAR منتقل می‌سازد. اولین آدرس برای روال‌های STORE و BRANCH بترتیب عبارتند از 00 0001 0 (دهدهی 4) و 00 0010 0 (دهدهی 8). اولین آدرس سیزده روال دیگر در آدرس‌های 12، 16، 20، ... و 60 می‌باشند. باین ترتیب چهار کلمه در حافظه کنترل به هر روال اختصاص می‌یابد.

در هر روال باید ریز دستورالعمل‌هایی برای محاسبه آدرس موثر و اجرای دستورالعمل در نظر بگیریم. روش آدرس دهنی غیر مستقیم در همه دستورالعمل‌های حافظه‌ای می‌تواند بکار رود. با ذخیره دستورالعمل‌های مربوط به آدرس دهنی غیر مستقیم به صورت یک روال، می‌توان در تعداد کلمه‌های حافظه کنترل صرفه‌جویی کرد. این زیر روال، که با INDRCT نشان داده شده، درست پس از روال برداشت قرار دارد، جدول ۷-۲. این جدول ریز برنامه سمبولیک را برای روال برداشت و روال‌های ریز دستورالعملی که چهار دستورالعمل کامپیوترا اجرا می‌کنند را نیز نشان می‌دهد.

برای اینکه ببینیم چگونه انتقال و برگشت از زیر روال غیر مستقیم INDRCT رخ می‌دهد، فرض کنید که ریز دستورالعمل MAP در انتهای روال برداشت موجب انشعباب به آدرس 0، یعنی مکانی که روال ADD در آن قرار دارد، شود. اولین ریز دستورالعمل در روال ADD زیر روال INDRCT را با شرط بیت وضعیت I، فرامی خواند. اگر I = 1 باشد انشعباب به INDRCT رخ داده و آدرس برگشت (آدرس 1 در این حالت) در ثبات زیر روال SBR ذخیره می‌شود. زیر روال INDRCT دو ریز دستورالعمل دارد.

INDRCT : READ U JUMP NEXT  
DRTAR U RET

بخاطر آورید که یک آدرس غیر مستقیم، بخش آدرس دستورالعمل را بعنوان آدرسی که آدرس موثر در آن قرار دارد تلقی می‌کند و آنرا عملوند تصور نمی‌نماید. بنابراین حافظه برای بدست آوردن آدرس موثر باید دستیابی شود و سپس به AR منتقل گردد. برگشت از زیر روال (RET) آدرس SBR را به CAR انتقال می‌دهد، و لذا به دومین ریز دستورالعمل روال ADD بازمی‌گردد.

## جدول ۷-۲ بخشی از برنامه سمبیک

عنوان	ریز عمل 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:	I	CALL	INDRCT
STORE:	ARTPC	U	JMP	FETCH
	ORG 8			
	NOP	I	CALL	INDRCT
	ACTDR	U	JMP	NEXT
EXCHANGE:	WRITE	U	JMP	FETCH
	ORG 12			
	NOP	I	CALL	INDRCT
	READ	U	JMP	NEXT
FETCH:	ACTDR, DRTAC	U	JMP	NEXT
	WRITE	U	JMP	FETCH
	ORG 64			
	PCTAR	U	JMP	NEXT
INDRCT:	READ, INCPC	U	JMP	NEXT
	DRTAR	U	MAP	
	READ	U	JMP	NEXT
	DRTAR	U	RET	

اجرای دستورالعمل ADD بوسیله ریزدستورالعمل واقع در آدرس های 1 و 2 انجام می‌گردد. اولین ریز دستورالعمل عملوند را از حافظه به DR می‌خواند. دومین ریز دستورالعمل ریزعمل جمع با محتوای DR و AC انجام داده و سپس به ابتدای روال برداشت بازمی‌گردد.

دستورالعمل BRANCH موجب انشعابی به آدرس موثر می‌گردد بشرطی که  $0 < AC$  باشد. AC هنگامی کوچکتر از صفر است که بیت علامت در آن منفی باشد که این از 1 بودن بیت وضعیت S آشکار می‌گردد. روال BRANCH در جدول ۷-۲ با وارسی مقدار S آغاز می‌شود. اگر S برابر 0 باشد، انشعابی رخ نمی‌دهد و ریزدستورالعمل بعدی موجب پرش به عقب به روال برداشت، بدون تغییر محتوای PC، می‌گردد. اگر S برابر 1 باشد، اولین ریزدستورالعمل JMP کنترل را به مکان OVER منتقل می‌سازد. ریزدستورالعمل واقع در این مکان زیرروال INDRCT را، اگر  $1 = I$  باشد، فرامی‌خواند. سپس آدرس

موثر از AR به PC منتقل و زیربرنامه به روال برداشت پوش می‌کند. روال STORE مجدداً زیرروال INDRCT را، اگر  $I = 1$  باشد، فرامی‌خواند. محتوای AC به DR منتقل می‌شود. یک عمل نوشتمن 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<sup>۱</sup> باشند (شکل ۱-۲۲). محتوای اصلی AC که حالا در DR است مجدداً در حافظه ذخیره می‌شود. توجه کنید که جدول ۷-۲ بخشی از ریزبرنامه را داراست. تنها چهار دستور از ۱۶ دستور کامپیوتر ریزبرنامه‌نویسی 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.

### ریزبرنامه دودویی

ریزبرنامه سمبیلیک فرم مناسبی برای نوشتمن ریزبرنامه‌هایی است که افراد می‌توانند بخوانند و بفهمند. ولی این روش ذخیره ریزبرنامه در حافظه نیست. برنامه سمبیلیک باید با یکی از دو روش برنامه اسمبیلی یا کاربر به دودویی ترجمه شود بشرطی که ریزبرنامه مانند این مثال ساده باشد.

فرم دودویی معادل ریزبرنامه در جدول ۷-۳ لیست شده است. آدرس‌های حافظه کنترل به هر دو صورت دهدی و دودویی داده شده است. همانطور که در جدول ۷-۱ دیده شد، محتوای دودویی هر ریزدستورالعمل از سمبیلها و معادلهای دودویی آنها بدست می‌آید.

دقت کنید که آدرس ۳ در ریزبرنامه سمبیلیک 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 هیچ وقت بکار نخواهد رفت. با این وجود اگر خطای پیش‌بینی نشده‌ای رخ دهد، یا اگر یک سیگنال پارازیت<sup>۲</sup> (نویز) مقدار ۳ را در CAR قرار دهد، پوش به آدرس ۶۴ که ابتدای روال برداشت است عاقلانه است.

ریزبرنامه دودویی لیست شده در جدول ۷-۳ 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 ۱ ها و ۰ ها را برای هر کلمه بوجود می‌آورد، تشکیل شده است. پس از آنکه اتصالات درونی در حین ساخت سوزانده شدند بیت‌های ROM ثبیت می‌گردند. ROM از بسته‌های مدار مجتمع، IC،

### جدول ۷-۳ بخشی از ریزبرنامه دودویی برای حافظه کنترل

ریزروال	آدرس			ریزدستورالعمل دودویی					
	ددهی	دودویی	ددهی	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	
INDRCT	66	1000010	101	000	000	00	11	0000000	
	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.

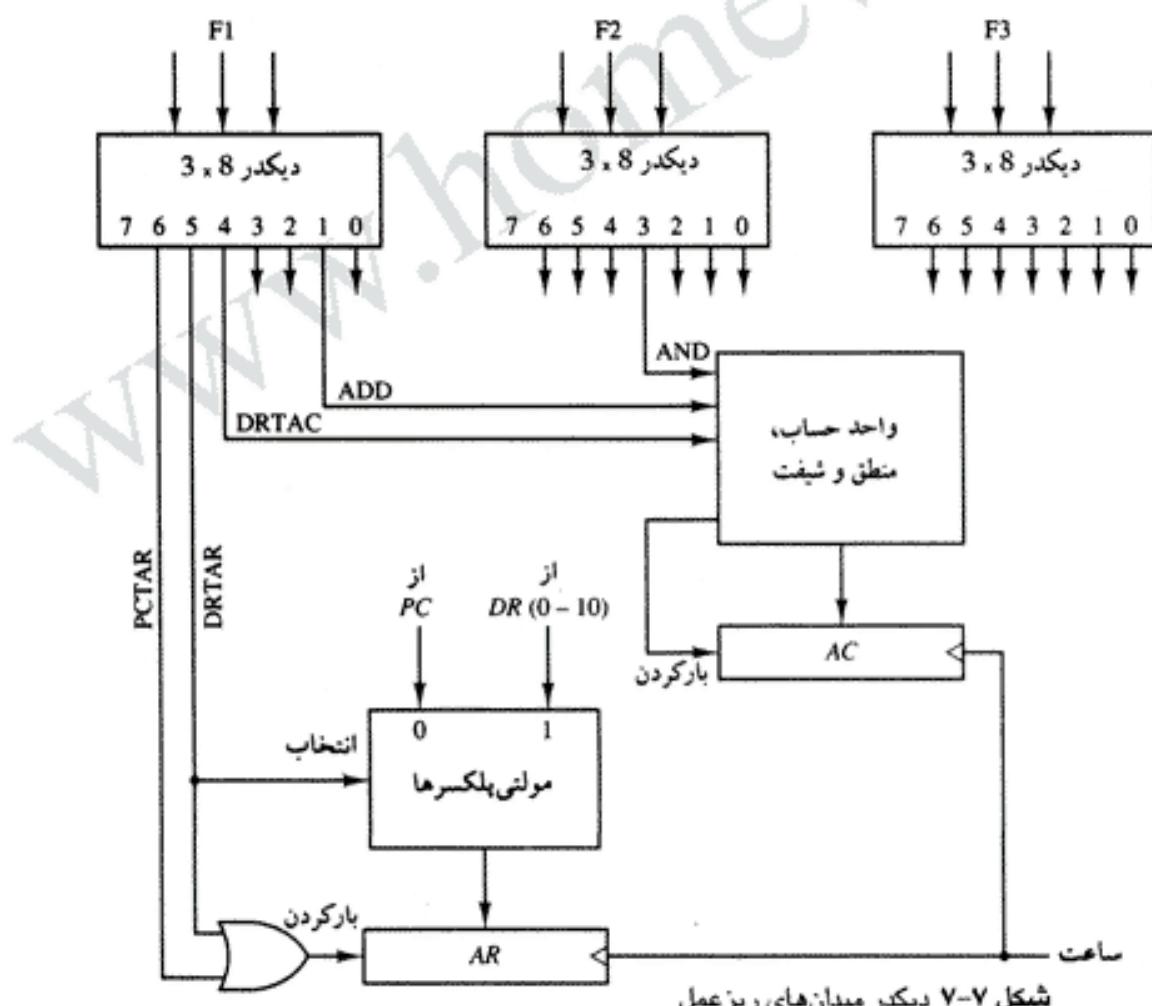
اگر بخواهیم حافظه کنترل قابل نوشتن باشد، ROM با RAM جایگزین می شود. مزیت بکارگیری RAM برای حافظه کنترل این است که ریزبرنامه بسادگی با نوشتن یک الگوی جدید از 1 ها و 0 ها بدون اعمال روش های سخت افزاری میسر است. قابلیت نوشتن در حافظه کنترل این انعطاف را ایجاد می کند که با تغییر ریزبرنامه تحت کنترل پردازنده 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 ارزانتر و سریعتر است و نیز هر کاربری نمی تواند معماری سیستم را تغییر دهد.

### ۷-۴ طراحی واحد کنترل

بیت های ریزدستورالعمل معمولاً به میدان 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 کد گذاری 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 دستور العمل را کاهش می دهد ولی مستلزم ساخت 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.

آنکه کردن بیت های کنترل در مثال برنامه نویسی بخش قبل نشان داده شد. 9 بیت میدان ریز عمل به سه میدان جزیی 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 دستور العمل موجود در خروجی حافظه کنترل با یک دیکدر  $8 \times 3$  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 ریز عمل مربوطه، مطابق جدول ۱-۷، متصل شود. مثلاً وقتی که  $F_1 = 101$  (دو دویی) باشد، گذرهای پالس ساعت بعدی محتوای  $(0 - 10)$  را به AR (که در جدول ۱-۷ با سمبل DRTAR نشان داده شده است) منتقل می‌نماید. بطور مشابه اگر  $F_1 = 110$  (دو دویی) باشد، انتقالی از PC به AR (که با سمبل PCTAR نشان داده شده است) صورت می‌گیرد. همانطور که در شکل ۱-۷ دیده می‌شود، خروجی‌های ۵ و ۶ دیکدر  $F_1$  به ورودی پارکردن AR وصل شده‌اند بطوری که وقتی هر یک از خروجی‌ها فعال گردد، اطلاعات از مولتی‌پلکسرها به AR منتقل می‌گردد. مولتی‌پلکسرها اطلاعات را، وقتی که خروجی ۵ فعال است از DR، و وقتی این خروجی غیر فعال باشد از PC دریافت می‌نمایند. انتقال به AR، با گذرهای پالس ساعت و هنگامی که خروجی ۵ یا ۶ 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 ۱-۱۹ و ۱-۲۰ طراحی کرد. به جای استفاده از گیت‌ها در تولید سیگنال‌های کنترلی AND، ADD و DR در شکل ۱-۱۹، این ورودی‌ها بترتیب از خروجی دیکدرهای متناظر AND، ADD و DRTAC می‌آیند، شکل ۱-۷. سایر خروجی‌های دیکدرها که مربوط به عمل روی AC است باید به نحوی مشابه به واحد حساب، منطق، شیفت متصل گردد.

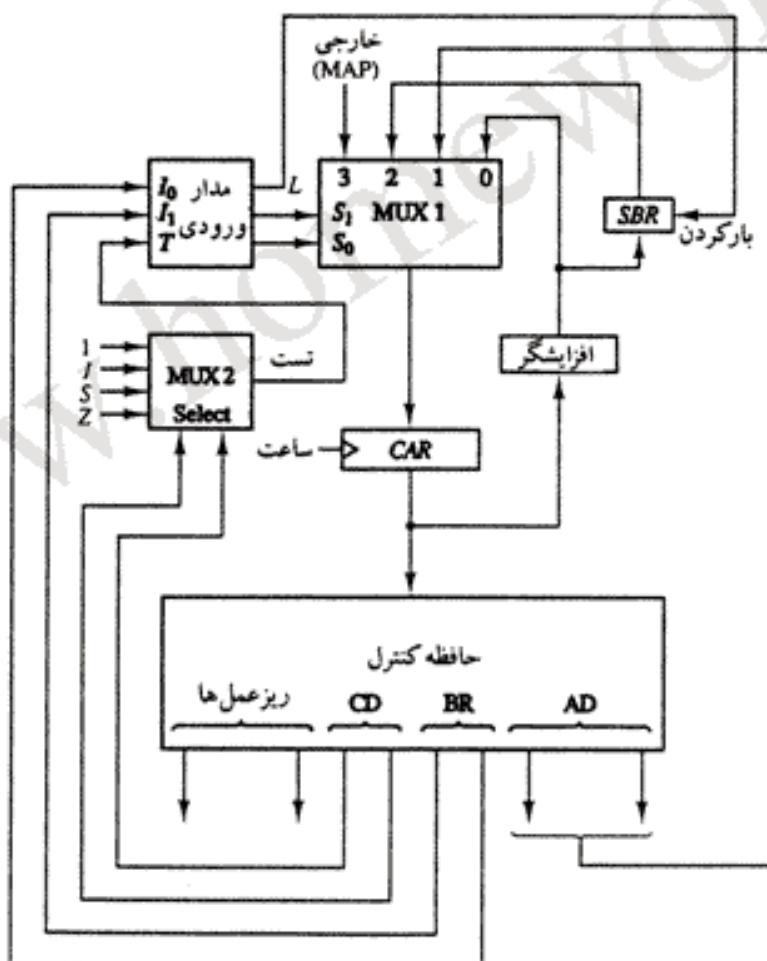
### توالی‌گر ریزبرنامه

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 پکارگیری توالی‌گر ریزبرنامه، ارائه آدرس به حافظه کنترل است بطوری که یک ریزدستور العمل 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 ریزدستور العمل جاری دریافت می‌کند هدایت می‌شود. توالی‌گرهای تجاری در داخل خود دارای مجموعه‌ای از ثبات‌های داخلی هستند که برای ذخیره موقع آدرسها هنگام اجرای حلقه‌های ریزبرنامه و فرآخوانی 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 ۱-۸ 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 مولتیپلکسر وجود دارد. اولین مولتیپلکسر آدرسی را از یکی از چهار منبع انتخاب کرده و آنرا به ثبات آدرس کنترل CAR هدایت می نماید. دومین مولتیپلکسر 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 مولتیپلکسر و نیز به ثبات زیر روال SBR اعمال می شود. سه ورودی دیگر مولتیپلکسر شماره ۱ از میدان آدرس ریز دستور العمل جاری، از خروجی SBR، و از منبع خارجی که دستور العمل را نگاشت می کند می آیند. اگر چه دیاگرام فقط یک ثبات زیر روال را نشان می دهد، یک توالی گر دارای پشتیاهی از ثباتها با عمق چهار تا هشت سطح می باشد. باین ترتیب، بطور همزمان چند زیر روال می توانند فعال باشند. اعمال پوش<sup>۱</sup> و پوب<sup>۲</sup>، به همراه اشاره گر



شکل ۸-۸ توالی گر ریز برنامه برای حافظه کنترل

1- Push

2- POP

پشت، 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 اینصورت برابر 0 است. مقدار T به همراه دو بیت از میدان BR (انشعب) 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، بارگردان آدرس خارجی، پوش یا پاپ پشت. 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 بارگردان SBR را فعال می نماید. مقادیر دودویی دو متغیر، انتخاب مسیر مورد نظر را در مولتی پلکسر معین می کند. مثلاً با  $10 = S_1 S_0$ ، ورودی شماره ۲ مولتی پلکسر انتخاب و مسیری بین CAR و SBR 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  $I_1$  و  $I_0$  همان بیت های میدان BR هستند. تابع مشخص شده برای هر واردہ از جدول ۷-۱ گرفته شده است. مقادیر بیت های  $S_1$  و  $S_0$  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 تابع 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 $I_1$ $I_0$ $T$	MUX 1 $S_1$ $S_0$	بارگردان $L$	SBR
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 شمارنده‌ای که با فلیپ فلاپ ساخته شده باشد نیست. بلکه مداری ترکیبی است که باگیت ساخته شده است. یک مدار افزایشگر ترکیبی را می‌توان با اتصالی سری مدارهای نیم جمع کننده (شکل ۴-۸) طراحی کرد. رقم نقلی خروجی از یک طبقه باید به ورودی طبقه بعدی وصل شود. یکی از ورودی‌های کم‌ارزشترین طبقه باید برابر ۱ 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. حداقل 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 ریزعمل ۱۶ بیتی است.

(الف) میدان آدرس انشعباب چند بیتی است

(ب) اگر ۱۶ بیت وضعیت وجود داشته باشد، چند بیت منطق انشعباب برای انتخاب یک بیت وضعیت لازم است.

(ج) چند بیت برای انتخاب ورودی مولتی‌پلکسر باقی‌مانده 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 بتویسید: (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 کترول 2048 کلمه دارد.

۷-۹ توضیح دهید که چگونه با استفاده از یک حافظه فقط خواندنی می‌توان 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 پیکریندی کامپیوتر شکل ۷-۴ به دو مولتیپلکسر نیاز داریم. آیا راه دیگری برای انتقال اطلاعات چند منبع به یک مقصد مشترک وجود دارد؟

۷-۱۱ با استفاده از جدول ۱-۷، برای ریزاعمال زیر، ۹ بیت میدان ریز عمل 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 ۱-۷، ریزاعمال سمبیلیک 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 می‌خواهیم 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 سمبیلیک 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 می‌شود را مشخص کنید.

(ب) چهار ریزدستورالعمل را به فرم معادل دودویی آنها تبدیل کنید.

۷-۱۵ کامپیوتر بخش ۳-۷ دارای ریزبرنامه دودویی زیر است.

آدرس	ریزبرنامه دودویی
60	0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 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 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 1 1 1 1 0 1 1 1 1 0 0

الف) آنرا به ریزبرنامه سمبولیک، 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 می‌آید بنویسید.  
۷-۱۶ دستورات زیر را به کامپیوتر بخش ۳-۷ اضافه کنید (EA آدرس موثر است): برنامه سمبولیک را برای هر روال مطابق جدول ۷-۲ بنویسید. (توجه کنید که مقدار AC نباید تغییر کند مگر اینکه دستور العمل تغییری در AC را مشخص نماید).

عمل	کد عمل	شکل سمبولیک	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 ( $PC \leftarrow EA$ )	انشعاب در صورت صفر بودن AC
SEQ	1001	If ( $AC = M[EA]$ ) then ( $PC \leftarrow PC + 1$ )	گذار در صورت برابری
BPNZ	1010	If ( $AC > 0$ ) then ( $PC \leftarrow EA$ )	انشعاب اگر مثبت و غیر صفر

۷-۱۷ یک روال ریزبرنامه برای دستور العمل ISZ (افزایش و گذار در صورت صفر بودن) که در فصل ۵ (جدول ۵-۴) تعریف شد بنویسید. از قالب دستور العمل بخش ۳-۷ استفاده کنید. توجه کنید

که شرط وضعیتی  $DR = 0$  در میدان CD کامپیوتر بخش ۳-۷ 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 سمبولیک را برای دستور العمل BSA (انشعاب و ذخیره آدرس) که در فصل ۵ تعریف شد بنویسید (جدول ۵-۴). از قالب ریز دستور العمل بخش ۳-۷ استفاده کنید. تعداد ریز دستور العمل را حداقل کنید.

۷-۱۹ نشان دهید که چگونه خروجی های ۵ و ۶ دیکدر F3 در شکل ۷-۷ به شمارنده برنامه PC وصل می شوند.

۷-۲۰ نشان دهید که چگونه یک میدان ۹ بیتی ریز عمل در ریز دستور العمل را می توان به زیر

میدانهایی تقسیم کرد تا 46 ریز عمل را مشخص نمایند؟ چند ریز عمل می‌توان در یک ریز دستور العمل مشخص کرد؟

۷-۲۱ کامپیوتروی دارای 16 ثبات، یک ALU<sup>۱</sup> با 32 عمل و یک شیفت دهنده با هشت عمل است که کلاً به یک سیستم گذرگاه مشترک متصل اند.

(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 ریز عمل  $R_6 + R_5 + R_4 \leftarrow R_4$  را نشان دهید.

۷-۲۲ فرض کنید که منطق ورودی توالی گر ریز برنامه شکل ۷-۸ چهار ورودی  $I_2, I_1, I_0$  و  $T$  (تست) و سه خروجی  $S_0, S_1$  و  $L$  را دارد. 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 $T=1$ ، پوش به AD اگر 0
x	0	1	پوش غیرشرطی به AD
1	0	0	افزایش غیرشرطی CAR
0	1	0	پوش به AD اگر $T=1$ ، افزایش CAR اگر 0
1	1	0	فراغواني زیرروال اگر $T=1$ ، افزایش CAR اگر 0
0	1	1	بازگشت غیرشرطی از زیرروال
1	1	1	نگاشت غیرشرطی آدرس خارجی

۷-۲۳ یک افزایشگر ترکیبی 7 بیتی برای توالی گر ریز برنامه شکل ۷-۸ طراحی کنید (شکل ۴-۸). با اضافه کردن یک ورودی کنترل D افزایشگر را تکمیل کنید. وقتی 0 = D است، مدار یک واحد افزایش می‌یابد، ولی وقتی 1 = D 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  $P^2$  (معنی قطبیت) در ریز دستور العمل حاصل از حافظه کنترل می‌آید. خروجی گیت به ورودی T در مدار منطقی ورودی متصل می‌شود. کنترل P چه کاری انجام می‌دهد.

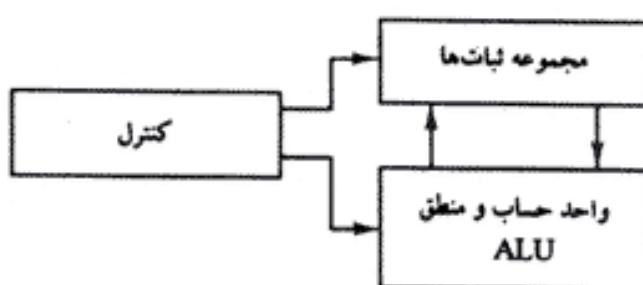
# واحد مرکزی پردازش



- |   |  |
|---|--|
| ۸-۱ مقدمه<br>۸-۲ سازمان ثبات های عمومی<br>۸-۳ سازمان پشتی<br>۸-۴ قالب دستورالعمل ها | ۸-۵ روش های آدرس دهنده<br>۸-۶ انتقال و دستکاری داده ها<br>۸-۷ کنترل برنامه<br>۸-۸ کامپیوتر کم دستور RISC |
|---|--|

## ۸-۱ مقدمه

بخشی از کامپیوتر که عملهای پردازش داده ها را انجام می دهد واحد مرکزی پردازش نام داشته و آنرا CPU نیز می خوانند. CPU از سه بخش عملهای تشکیل یافته است، شکل ۸-۱ مجموعه ثبات های داده های میانی که در حین اجرای دستورالعمل بکار می روند را ذخیره می نمایند. واحد حساب و منطق (ALU) ریزاعمال لازم برای اجرای دستورات را انجام می دهد. واحد کنترل مدیریت انتقال داده ها را درین ثبات بعده داشته و ALU را در انجام اعمال هدایت می کند.



شکل ۸-۱ اجزاء مهم یک CPU

CPU انواع متعددی از توابع<sup>۱</sup> را که توسط نوع خاصی دستور دیکته می‌شود اجرا می‌نماید. گاهی اوقات معماری<sup>۲</sup> کامپیوتر بوسیله برنامه نویس، که دستورات زبان ماشین را بکار می‌برد، عنوان ساختار و رفتار کامپیوتر تعریف می‌شود. این ساختار و رفتار قالب دستورالعمل‌ها، روش‌های آدرس دهی، مجموعه دستورالعمل‌ها و سازمان کلی ثبات‌های CPU است.

یکی از مزهای مشترک دیدگاه‌های طراح کامپیوتر و برنامه نویس از ماشین، بخشی از CPU است که به مجموعه دستورات مربوط می‌شود. از نظر طراحان، مجموعه دستورالعمل‌ها، مشخصات طراحی CPU را فراهم می‌آورند. بخش عده طراحی یک CPU انتخاب سخت افزار برای اجرای دستورات ماشین است. کاربری که کامپیوتر را در زبان ماشین یا اسembly برنامه ریزی می‌کند باید از مجموعه ثبات‌ها، ساختمان حافظه، نوع داده‌هایی که دستورالعمل‌ها می‌توانند با آنها کار می‌کنند و عملی که هر دستور اجرا می‌نماید آگاه باشد. مثال‌هایی از CPU‌های ساده در فصل ۵ و ۷ آورده شد. این فصل سازمان و معماری یک CPU را با تأکید به نظرات کاربر کامپیوتر توصیف می‌کند. ابتدا بطور مختصر چگونگی ارتباط ثبات‌ها را با ALU از طریق گذرگاه‌ها توضیح داده و طرز کار پشته حافظه را تشریح می‌کنیم. سپس انواع قالب‌های دستورات موجود، روش‌های آدرس دهی بکار رفته در بازیابی داده‌ها از حافظه و انواع دستوراتی که عموماً در کامپیوترها دیده می‌شوند را نشان خواهیم داد. آخرین بخش مفهوم کامپیوترهای کم دستور<sup>۳</sup> (RISC) را بیان دارد.

## ۲-۸ سازمان ثبات‌های عمومی

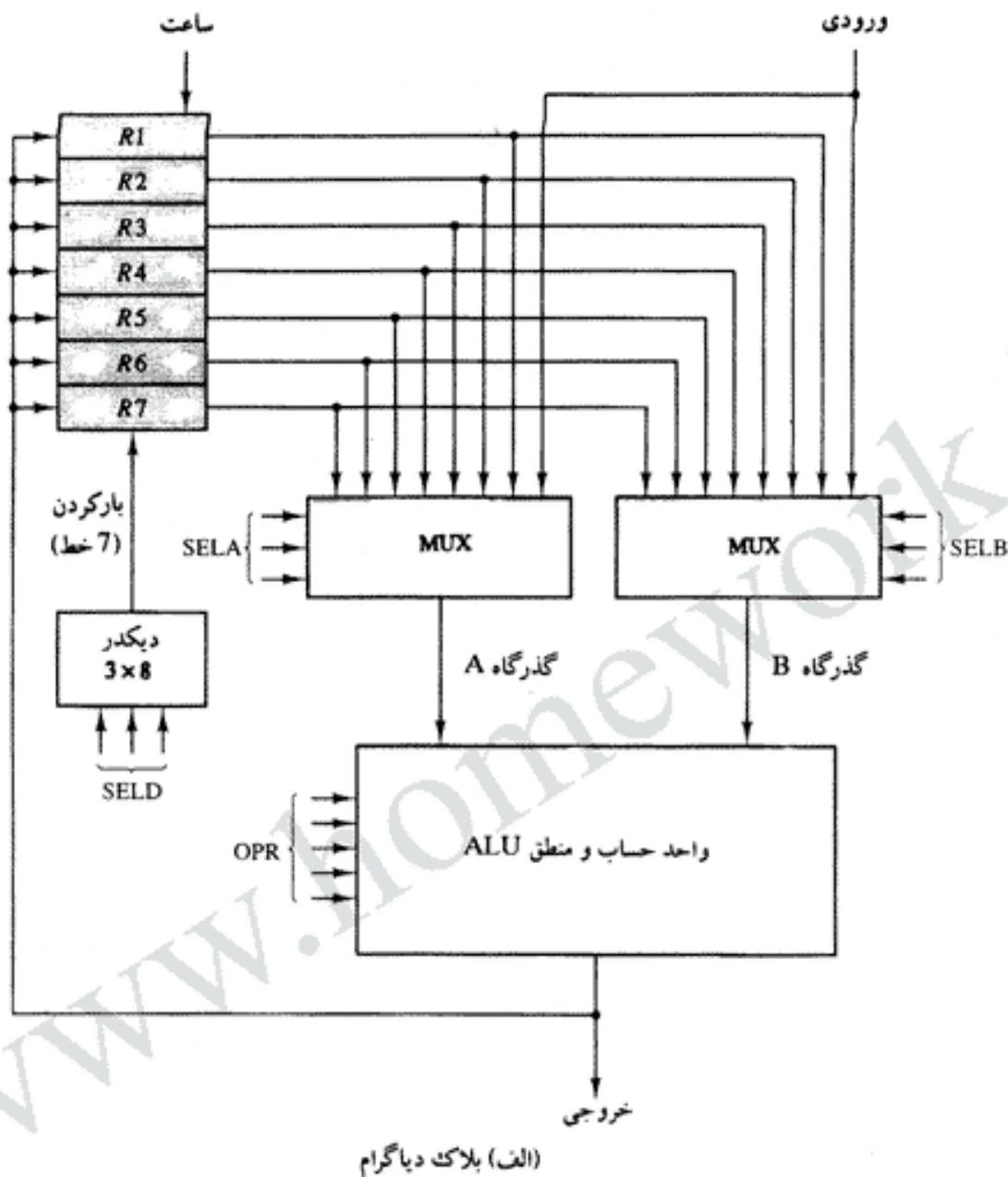
در مثال برنامه نویسی فصل ۶ دیدیم که مکان‌های حافظه برای ذخیره کردن اشاره‌گرهای، شمارنده‌ها، آدرس برگشت نتایج موقت و حاصلضرب جزیی<sup>۴</sup> در ضرب لازم هستند. ارجاع به مکان‌های حافظه برای اهداف فوق، چیزی جز اتلاف وقت نیست، زیرا دستیابی به حافظه وقت گیرترین عمل در یک کامپیوتر است. بهترین و مفیدترین راه برای نگهداری این مقادیر میانی ذخیره کردن آنها در ثبات‌های پردازشگر است. وقتی که تعداد زیادی ثبات در CPU در نظر گرفته شد، بهترین راه بهره‌گیری از آنها ارتباطشان از طریق یک سیستم گذرگاه مشترک است. این ثبات‌ها تنها در انتقال مستقیم داده‌ها با هم ارتباط برقرار نمی‌کنند بلکه در حین اجرای انواع ریز عمل‌ها هم بهم وصل می‌شوند. از این‌رو لازم است واحد مشترکی را فراهم آوریم تا تمام ریز اعمال 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 می‌نمایند. گذرگاه‌های A و B ورودی‌های واحد حساب - منطق (ALU) مشترک را تشکیل می‌دهند. عمل انتخاب شده در U

1- Function

2- Architecture

3- Reduced Instruction Set Computer

4- Partial Product



3	3	3	5
SEL A	SEL B	SEL D	OPR

(ب) کلمه کنترل

شکل ۸-۲ مجموعه ثبات‌ها با ALU مشترک

معین کننده 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 بارگردان، 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 پلکسرا A (SEL A): برای قرار دادن محتوای  $R_2$  روی گذرگاه A
- ۲- انتخاب کننده مولتی پلکسرا B (SEL B): برای قرار دادن محتوای  $R_3$  روی گذرگاه B
- ۳- انتخاب گر عملیات در ALU (OPR): برای انجام عمل جمع حسابی  $A + B$
- ۴- انتخاب گر دیکدر مقصود (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 پلکسراها به ALU سپس به گذرگاه خروجی و از آنجا به ورودی ثبات‌های مقصود 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 پلکسراها و گیت‌های سه حالته مطابق آنچه در بخش (۴-۳) دیده شد پیاده سازی شده‌اند.

### کلمه کنترل

در سیستم، چهارده ورودی انتخاب دودویی وجود دارد که ترکیب مقدار عددی آنها یک کلمه کنترل<sup>۱</sup> را تشکیل می‌دهد. کلمه کنترل 14 بیتی در شکل ۸-۲ (ب) تعریف شده است. این کلمه از چهار بخش تشکیل می‌شود. سه بخش از آن هر یک سه بیت و یک بخش پنج بیتی است. سه بیت SEL A یک ثبات مبدأ را برای ورودی A از واحد ALU انتخاب می‌کنند. سه بیت مربوط به SEL B هم یک ثبات دیگر را برای ALU از طریق ورودی B انتخاب می‌نمایند. سه بیت SELD ثبات مقصود را بکمک دیکدر و هفت خروجی مربوطه‌اش انتخاب می‌نمایند. پنج بیت OPR یکی از عملیات ALU را بر می‌گزینند. چهارده بیت کلمه کنترل فوق وقتی کلاً به ورودی‌های انتخاب اعمال شوند یک ریز عمل خاصی را مشخص خواهند کرد.

جدول (۸-۱) 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 SEL A و SEL B و SELD انتخاب می‌شود عدد دهدۀ مربوطه‌اش معادل با عدد دودویی کد مربوطه‌اش می‌باشد. هرگاه SEL A

1- Control Word

جدول A-1 تخصیص کد به میدانهای انتخاب ثباتها

کد دو دویی	SEL A	SEL B	SEL D
000	ورودی	ورودی	پنج
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

SELB برابر 000 باشد، مولتی پلکس مریبوطه داده را از ورودی خارجی می پذیرد. وقتی SELD=000 باشد هیچیک از ثباتها انتخاب نشده ولی محتوای گذرگاه روی خروجی خارجی قرار می گیرد. ALU اعمال حسابی و منطقی را انجام می دهد. بعلاوه CPU باید قابلیت انجام شیفت را داشته باشد. مدار شیفت دهنده می تواند در ورودی ALU قرار گرفته و امکان پیش شیفت<sup>۱</sup> را فراهم آورد و یا پس از ALU بوده و امکان پس شیفت<sup>۲</sup> را بوجود آورد. در برخی مواقع این عمل را خود ALU انجام می دهد. یک مدار حساب، منطق و شیفت در بخش (۴-۷) طراحی شد. جدول عملیات این ALU در جدول (۴-۸) دیده شد. کدگذاری<sup>۳</sup> 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 سمبولیک مشخص شده است.

جدول A-2 تخصیص کد به اعمال ALU

OPR انتخابگر	عمل	سبل
00000	A انتقال	TSFA
00001	A افزایش	INCA
00010	A+B جمع	ADD
00101	A-B تفریق	SUB
00110	A کاهش	DECA
01000	B و A بات های AND	AND
01010	B و A بات های OR	OR
01100	B و A بات های XOR	XOR
01110	A متضمن کردن	COMA
10000	A به راست شیفت	SHRA
11000	A به چپ شیفت	SHLA

1- Preshift

2- Postshift

3- Encode

### مثالهایی از ریز عملها

برای تعیین یک کلمه کنترل 14 بیتی نیاز است. کلمه کنترل مربوط به یک ریز عمل معین را می توان از متغیرهای انتخاب بدست آورد. مثلاً برای ریز عمل تفریق که با عبارت زیر مشخص می شود:

$$R_1 \leftarrow R_2 - R_3$$

را برای ورودی A از ALU، R<sub>3</sub> را برای ورودی B از ALU و R<sub>1</sub> را بعنوان ثبات مقصد و عمل تفریق A-B را در ALU مشخص می نماید. بنابراین کلمه کنترل از چهار میدان تشکیل و هر میدان از لیست کدهای جدول (۸-۱) و (۸-۲) بدست می آید. کلمه کنترل دو دویسی برای ریز عمل تفریق برایست با 010 011 001 00101 که بطريق زیر بدست می آید.

بخش:	SEL A	SEL B	SEL E	OPR
عمل:	R <sub>2</sub>	R <sub>3</sub>	R <sub>1</sub>	SUB
کلمه کنترل:	010	011	001	00101

کلمه کنترل برای تفریق و چند دستور دیگر در جدول (۸-۳) آورده شده است.

ریز عمل های افزایش و انتقال، ورودی B از ALU را بکار نمی برد. در این موارد برای میدان B خط تیره (-) قرار داده می شود. برای هر میدان بکار نرفته، مانند فوق، در کلمه کنترل عدد دو دویسی 000 را تخصیص خواهیم داد هر چند که هر عدد دو دویسی 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 ریز عمل های CPU

ریز عمل	SEL A	SEL B	SEL D	OPR	مشخص کردن سیبل	کلمه کنترل
R1 $\leftarrow$ R2 - R3	R2	R3	R1	SUB	010 011 001 00101	
R4 $\leftarrow$ R4 $\vee$ R5	R4	R5	R4	OR	100 101 100 01010	
R6 $\leftarrow$ R6 + 1	R6	—	R6	INCA	110 000 110 00001	
R7 $\leftarrow$ R1	R1	—	R7	TSFA	001 000 111 00000	
Output $\leftarrow$ R2	R2	—	None	TSFA	010 000 000 00000	
Output $\leftarrow$ Input	Input	—	None	TSFA	000 000 000 00000	
R4 $\leftarrow$ sh1 R4	R4	—	R4	SHLA	100 000 100 11000	
R5 $\leftarrow$ 0	R5	R5	R5	XOR	101 101 101 01100	

### $x \oplus x = 0$ انحصاری<sup>۱</sup> (XOR) صفرشود، زیرا

باتوجه به این مثالها واضح است که ریز عمل های 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 ریز عمل های دلخواه را برای CPU ایجاد کرد.

این نوع کنترل به کنترل ریزبرنامه نویسی<sup>۲</sup> شده معروف است. این نوع واحد کنترل در شکل (۷-۸) نشان داده شد. کلمه دو دویی کنترل برای CPU از خروجی هایی که در حافظه کنترل با "ریز عمل ها" مشخص شده می آیند.

### ۸-۳ سازمان پشته

یک ویژگی مفید که در CPU اغلب کامپیوترها وجود دارد پشته یا لیست اولین ورودی - آخرین خروجی<sup>۳</sup> (LIFO) است. پشته یک وسیله ذخیره سازی است که اطلاعات را بنحوی ذخیره می کند که همیشه آخرین کمیت ذخیره شده اولین کمیت بازیابی شونده 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 بارگرد). ثباتی که آدرس پشته را نگهداری می نماید اشاره گر پشته<sup>۴</sup> 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 پشتئ کرد، ثبات های فیزیکی پشته فقط همواره برای خواندن و نوشتمن 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<sup>۵</sup> (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<sup>۶</sup> (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 بالافزایش یا کاهش ثبات اشاره گر پشته شبیه سازی می گردند.

### پشته های ثباتی

یک پشته می تواند دریخشی 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

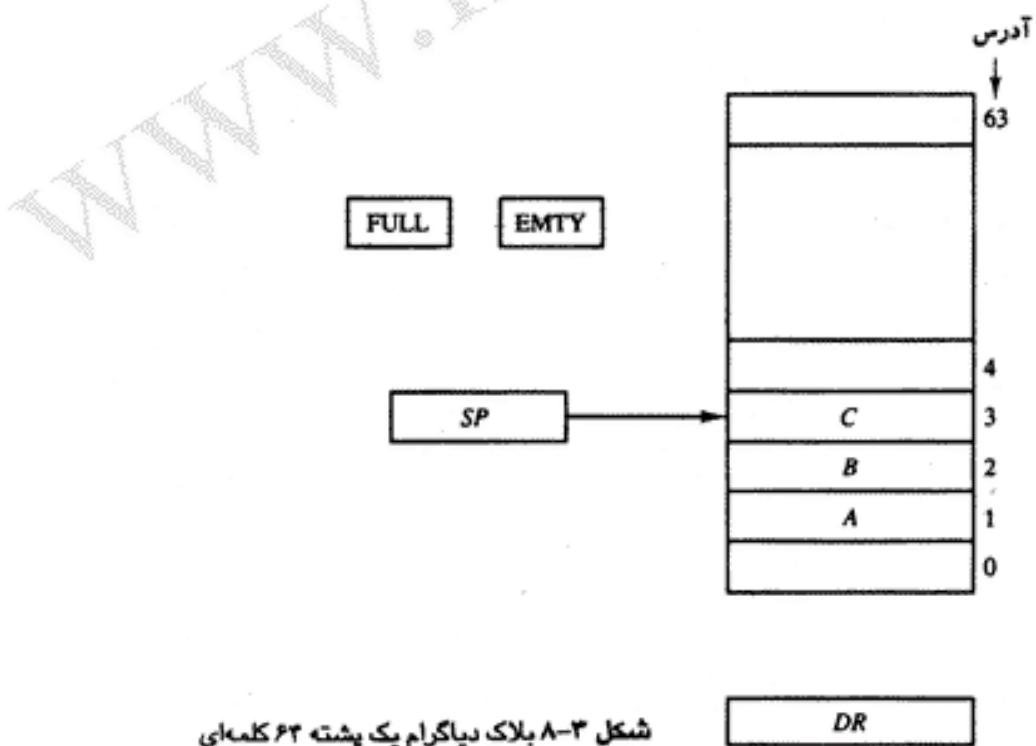
5- Push

6- Pop

ثبات قرار داشته باشد. شکل ۸-۳ سازمان یک پشته ثباتی 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 بترتیب 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  $64 = 2^6$  است. چون اشاره گر SP تنها دارای ۶ بیت است نمی‌تواند از ۶۳ (دودویی 111111) 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 ۰ 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=0) یک کمیت جدید با عمل پوش



شکل ۸-۳ بلاک دیاگرام 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( $SP = 0$ ) then ( $FULL \leftarrow 1$ )	آیا پشته پر است یا نه
$EMTY \leftarrow 0$	علامت خالی نبودن پشته

شاره 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  $EMTY = 0$  می گردد. اگر پشته خالی نباشد (یعنی  $EMTY = 0$ ) 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 ( $SP = 0$ ) then ( $EMTY \leftarrow 1$ )	آیا پشته خالی است
$FULL \leftarrow 0$	علامت پرنبودن 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  $EMTY = 1$  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،  $SP$  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 EMTY = 1 پشته پاپ شود خطای عملیاتی رخ می دهد.

### پشته حافظه ای

پشته می تواند بعنوان یک واحد مستقل طبق شکل ۸-۳ وجود داشته باشد و یا در یک حافظه

RAM متصل به CPU پیاده سازی شود. پیاده سازی پشته در CPU بدین ترتیب صورت می‌گیرد که بخشی از حافظه به کمک یک ثبات پردازشگر به پشته اختصاص می‌یابد. شکل ۸-۴ بخشی از حافظه کامپیوتر را که به سه قسمت شده نشان می‌دهد: این سه قسمت به برنامه، داده و پشته تخصیص یافته است. شمارنده برنامه PC به آدرس دستور بعدی اشاره می‌کند. ثبات آدرس AR به آرایه ای از داده‌ها اشاره می‌کند. اشاره‌گر پشته نیز به بالای پشته اشاره می‌نماید. سه ثبات به یک گذرگاه آدرس مشترک متصلند و همواره یکی از آنها می‌تواند آدرسی را برای حافظه فراهم آورد. PC در طول فاز برداشت<sup>۱</sup> برای خواندن یک دستورالعمل بکار می‌رود. AR در طول فاز اجرا برای خواندن عملوند بکار گرفته می‌شود. SP نیز بهنگام پوش یا پاپ کمیت‌ها را به پشته می‌فرستد یا از آن برگزیند.



شکل ۸-۴ حافظه کامپیوتر با  
قطعه‌های برنامه، داده و پشته

1- Fetch

همانطور که در شکل ۸-۴ ملاحظه می شود، مقدار اولیه SP برابر ۴۰۰۱ بوده و پشته با کاهش آدرس پرمی شود. بنابراین اولین کمیت در آدرس ۴۰۰۰ ذخیره می گردد، دومین کمیت در ۳۹۹۹ و بالاخره آخرین کمیت می تواند در آدرس ۳۰۰۰ 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.

$$\begin{aligned} SP &\leftarrow SP - 1 \\ M[SP] &\leftarrow DR \end{aligned}$$

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.

$$\begin{aligned} DR &\leftarrow M[SP] \\ SP &\leftarrow SP + 1 \end{aligned}$$

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 سرریزشدن<sup>۱</sup> پشته (پرترشدن از حداکثر) و یا فروریز<sup>۲</sup> (حالی ترشدن از حداقل) ندارند. حدود پشته می تواند با دو ثبات پردازشگر چک شود: یکی از آنها تا حد بالا (دراین حالت ۳۰۰۰) و دیگری حد پائین (دراین حالت ۴۰۰۰) رانگه 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 عبارتنداز (۱) دستیابی به حافظه توسط SP و (۲) بهنگام کردن<sup>۳</sup> 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 ۸-۴ گسترش پشته با کاهش آدرس حافظه همراه است. پشته می تواند با افزایش آدرس هم مطابق شکل ۸-۳ 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.

اشارة 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

### نمایش لهستانی معکوس<sup>۱</sup>

سازمان پشته برای ارزیابی عبارات حسابی بسیار مفید است. روش‌های ریاضی معمول در نوشتند عبارات حسابی وقتی توسط یک کامپیوتر ارزیابی شوند مشکلاتی را ایجاد می‌نمایند. عبارات حسابی معمول بصورت نمایش میانوندی<sup>۲</sup> نوشته شده و عملگر بین دو عملوند قرار می‌گیرد. عبارت ساده حسابی زیرا در نظر بگیرید

$$A * B + C * D$$

علامت ستاره (که در واقع ضرب می‌باشد) 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 می‌بینیم در محاسبه عبارات میانوندی لازم است جستجوهایی رو به جلو و عقب در عبارت صورت گیرد تا عمل بعدی انجام پذیرد.

ریاضی دان لهستانی لوکاسویچ<sup>۳</sup> نشان داد که عبارات ریاضی می‌توانند بصورت نمایش پیشوندی<sup>۴</sup> نشان داده شوند. این نمایش عبارات اغلب نمایش لهستانی خوانده می‌شوند و در آن عملگر را قبل از عملوندها قرار می‌دهند. نمایش پسوندی<sup>۵</sup> یا نمایش لهستانی معکوس (RPN) عملگر را بعداز عملوندها قرار می‌دهد. مثال‌های زیر سه فرم نمایش عبارات را نشان می‌دهد.

$$\text{نمایش میانوندی } B$$

$$\text{نمایش پیشوندی لهستانی } +AB$$

$$\text{نمایش پسوندی لهستانی } AB+$$

روش لهستانی معکوس برای عملیات پشته مناسب است، عبارت

$$A * B + C * D$$

در این روش بشکل زیر در می‌آید

$$AB * CD * +$$

و بطريق زیر ارزیابی می‌شود: 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 نمائید. این عمل ادامه می‌یابد 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 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 (A + B) \* (D + E) بر جمع با C \* 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 پیش روی 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 ماشین‌های حساب و نیز برخی از کامپیوترها رایج است. پشته خصوصاً برای حل مسائل طولانی و پیچیده از جمله محاسبات زنجیره‌ای مفید است. زیرا عبارات حسابی می‌توانند بصورت RPN بدون پرانتز بیان گردد.

روش مذکور بدین ترتیب انجام می‌شود که ابتدا عبارات حسابی به فرم معادل RPN در می‌آیند عملوندها بداخل پشته به همان ترتیبی که نشان داده شده‌اند هدایت می‌شوند. آغاز هر عمل بستگی به این دارد که ماشین حساب یا کامپیوتر بکار رفته باشد. در ماشین حساب، عملگرها از طریق صفحه کلید وارد می‌شوند. در کامپیوتر اعمال توسط دستوراتی که دارای بخش کد عملیاتی هستند و آدرس هم نیاز ندارند شروع می‌گردد. ریز عمل‌های زیر با پشته هرگاه که یک عملگر در ماشین حساب وارد شود و یا عملگری در کامپیوتر توسط کنترل تأیید گردد اجرا می‌گردد: (۱) دو عملوند که در بالاترین مکان پشته قرار دارند برای اجرای عمل مورد نظر انتخاب می‌شوند، و (۲) پشته پاپ می‌شود و نتیجه عمل بجای عملوند پائین‌تر می‌نشیند. با پوش کردن متواالی عملوندها بداخل پشته و انجام اعمال تعریف شده فوق، عبارت بطور صحیح ارزیابی شده و نتیجه نهایی در بالای پشته باقی می‌ماند.

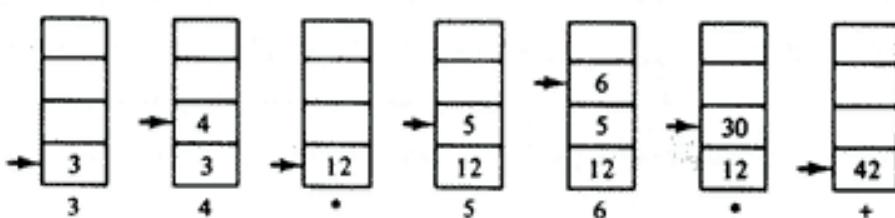
مثال عددی زیر می‌تواند این روش را واضح تر بیان نماید. عبارت حسابی زیر را در نظر بگیرید.

$$(3 * 4) + (5 * 6)$$

در RPN عبارت بصورت زیر است

$$34 * 56 * +$$

حال عملیات پشته طبق آنچه در شکل ۸-۵ آمده است را ملاحظه نمایید. هر قسمت از شکل یک مرحله از کار پشته را نشان می‌دهد و فلش همواره به بالای پشته اشاره می‌کند. با مرور عبارت از چپ به راست با دو عملوند مواجه می‌شویم. ابتدا عدد ۳ و سپس ۴ بداخل پشته فرستاده می‌شوند. نماد بعدی عملگر ضرب \* است. این نماد موجب ضرب دو عدد واقع در بالاترین مکان پشته می‌شود. سپس پشته پاپ شده و حاصل ضرب در بالای پشته قرار می‌گیرد، و در واقع جایگزین دو عملوند می‌گردد. آنگاه در عبارت فوق با دو عملوند دیگر یعنی ۵ و ۶ مواجه می‌شویم و آنها را بداخل پشته پوش می‌کنیم. نتیجه عمل پشته که حاصل از \* بعدی است دو عدد اخیر را با حاصل ضرب جایگزین می‌سازد. آخرین عمل، جمع بر روی دو عدد بالای پشته است تا عدد نهایی ۴۲ حاصل گردد.



شکل ۸-۵ عمل‌هایی که برای محاسبه  $6 \times 5 \times 4 + 3 \times 4$  روی پشته انجام می‌شود

ماشین حساب علمی<sup>۱</sup> که از پشته داخلی استفاده می‌کنند نیاز به تبدیل عبارات به RPN توسط کاربر را دارند. کامپیوترهایی که CPU آنها دارای پشته است نیاز به برنامه‌ای دارند تا تبدیل را برای کاربر انجام دهد. اغلب کامپایلرها، بدون توجه به سازمان CPU، تمام عبارات حسابی را به RPN تبدیل می‌کنند 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.

#### ۸-۴ قالب دستورالعمل‌ها

ساختار فیزیکی و منطقی کامپیوترها معمولاً در کتابچه راهنمایشان که به همراه سیستم است تشریح می‌شود. چنین کتابچه‌های ساختمان داخلی CPU، از جمله ثبات‌های موجود و قابلیت‌های منطقی آنها را توضیح می‌دهد. آن‌ها فهرست همه دستورات پیاده‌سازی شده با سخت افزار، قالب کد آنها و نهایتاً تعریف دقیقی برای هر دستور را فراهم می‌آورند. یک کامپیوتر معمولاً انواع متنوعی از قالب دستورات را دارد. 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 چهارگوش کشیده می‌شود که در آن بیت‌های دستورالعمل آن طور که در کلمه حافظه یا ثبات کنترل ظاهر می‌شوند آمده است. بیت‌های دستورالعمل به گروههایی بنام میدان<sup>۲</sup> تقسیم می‌شود. متداول‌ترین میدان‌هایی که در قالب دستورالعمل‌ها دیده می‌شوند، عبارتند از:

- ۱- میدان کد عملیات که عمل را مشخص می‌کند.
- ۲- میدان آدرس که آدرس حافظه یا ثبات پردازشگر را معین می‌نماید.
- ۳- میدان روش<sup>۳</sup> (شیوه) که نحوه تعیین آدرس مؤثر یا عملوند را مشخص می‌نماید.

تحت شرایط خاصی، میدان‌های خاص دیگری نیز گاهی اوقات بکار گرفته می‌شوند، مثلًاً میدانی که تعداد شیفت‌های یک دستورالعمل از نوع شیفت را معین می‌کند.

میدان کد عملیات یک دستورالعمل عبارتست از گروهی از بیت‌ها که انواع اعمال پردازشگر مانند جمع، تفریق، مکمل‌سازی و شیفت را تعریف می‌نمایند. متداول‌ترین اعمال موجود در کامپیوترها در بخش ۸-۶ بحث شده است. بیت‌هایی که روش را برای دستور تعریف می‌کنند راه‌های مختلفی را برای انتخاب عملوند از آدرس مفروض معین می‌نمایند. روش‌های مختلف آدرس دهنی که برای کامپیوترهای دیجیتال می‌توان ردیف کرد در بخش ۸-۵ آرائه شده‌اند. در این بخش درباره میدان آدرس یک دستورالعمل بحث خواهیم نمود و اثر میدان‌های چند آدرسه در یک دستور مورد بررسی قرار خواهد گرفت.

اعمال مشخص شده توسط دستورات کامپیوتر بر روی داده‌های ذخیره شده در حافظه یا ثبات‌های پردازنده اجرا می‌شود. عملوندهای موجود در حافظه توسط آدرس حافظه مشخص می‌شوند. عملوندهایی

که در ثبات‌های پردازشگر قرار دارند توسط آدرس ثبات معین می‌گردند. یک آدرس ثبات یک عدد  $k$  بیش است که  $2^k$  ثبات را در CPU تعریف می‌کند. بنابراین یک CPU با ۱۶ ثبات  $R_0$  تا  $R_{15}$  دارای میدان آدرس ثبات چهاربیتی است. مثلاً عدد دو دویی ۰۱۰۱ 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 پشتیای

مثالی از نوع تک انباره‌ای همان کامپیوتر فصل ۵ است. تمام اعمال یکمک 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

**$AC \leftarrow AC + M[X]$**

ثبات انباره و  $M[X]$  کلمه حافظه واقع در آدرس X را مشخص می‌نماید.

مثالی از نوع چند ثباتی در شکل ۱-۷ 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<sub>1</sub>, R<sub>2</sub>, R<sub>3</sub>**

که عمل  $R_3 \leftarrow R_1 + R_2$  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<sub>1</sub>, R<sub>2</sub>**

عمل  $R_1 \leftarrow R_1 + R_2$  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<sub>1</sub>, R<sub>2</sub>**

بیانگر عمل  $R_2 \leftarrow R_1$  (یا  $R_1 \leftarrow R_2$ ) 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 R<sub>1</sub>, X**

مریوط به عمل  $M[X] \leftarrow R_1 + R_1$  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 نشاند. اشاره گر پشته بطور اتوماتیک تصحیح می گردد. دستورات محاسباتی در کامپیوترهای سازمان یافته با پشته احتیاجی به آدرس ندارند زیرا اعمال مذبور بر روی دو کمیت که در بالای پشته قرار دارند اجرا می شود.

دستور العمل ADD در کامپیوتر بسته ای مشکل از یک کد عمل ولی بدون آدرس است. این عمل وظیفه بازگرفتن<sup>۱</sup> دو عدد در بالای پشته را بعده داشته و پس از جمع آنها، حاصل را در پشته می نشاند<sup>۲</sup>. در این دستور تیازی به عملوندی 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 بارگردان 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 8080 Intel دارای یک اشاره گر پشته بوده و دستوراتی برای نشاندن و بازگرفتن داده از پشته حافظه ای دارد. با این وجود پردازشگر دارای دستورات نوع آدرس صفر<sup>۳</sup> 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 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 نماییم. همچنین فرض می کنیم عملوندها در آدرس های 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 آدرسه

کامپیوترهایی که قالب دستورات آنها سه آدرسه است هر میدان آدرس را برای مشخص نمودن ثبات پردازشگر یا عملوندی در حافظه می‌توانند بکار بزنند. برنامه‌ای در زبان اسمبلی که عبارت  $(C+D) * (A+B)$  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 آست که پردازشگر دارای دو ثبات  $R_1$  و  $R_2$  است. نماد  $M[A]$  معنی وجود عملوند در آدرس  $A$  است.

مزیت دستورات سه آدرسه کوتاهی برنامه در ارزیابی عبارات محاسباتی است. عیب این دستورات هخصوص 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 آدرسه

دستورات دو آدرسه 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  $(C+D) * (A+B)$  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 ازو 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 آدرسه

دستورات یک آدرسه از اثماره 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 صفرآدرسه

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 معنی بالای پشته<sup>1</sup> 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		$TOS (C + D)$
MUL		$TOS (C+D)*(A+B)$
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 صفرآدرس به آنها اطلاق شده است.

### RISC

مزایای یک معماری کامپیوتر کم دستور<sup>2</sup> (RISC) در بخش ۸-۸ تشریح شده است. مجموعه دستورات پردازشگر RISC به هنگام تبادل اطلاعات CPU با حافظه به STORE و LOAD 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  $(A+B) * (C+D) = X$  را مشخص می نماید.

LOAD	$R_1, A$	$R_1 \leftarrow M[A]$
LOAD	$R_2, B$	$R_2 \leftarrow M[B]$
LOAD	$R_3, C$	$R_3 \leftarrow M[C]$
LOAD	$R_4, D$	$R_4 \leftarrow M[D]$
ADD	$R_1, R_1, R_2$	$R_1 \leftarrow R_1 + R_2$
ADD	$R_3, R_3, R_4$	$R_3 \leftarrow R_3 + R_4$
MUL	$R_1, R_1, R_3$	$R_1 \leftarrow R_1 * R_3$
STORE	$X, R_1$	$M[X] \leftarrow R_1$

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 <sup>۱</sup>

میدان عمل یک دستور، عملی را که قرار است انجام شود مشخص می نماید. این عمل بر روی داده هایی که در ثباتها 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 های را برای تفسیر و تصحیح آدرس دستور العمل مشخص می کند. کامپیووترها تکنیک روش های آدرس دهی را به یک یا هر دو دلیل زیر بکار می بردند.

- ۱- به کاربر، بدليل در اختیار داشتن اشاره گر، شمارنده های حلقه، اندیس دهی داده ها و تغییر مکان برنامه انعطاف پذیری لازم را می دهد.
- ۲- تعداد بیت های میدان آدرس دستور العمل را کاهش می دهد.

در اختیار داشتن روش های آدرس دهی به برنامه نویسان زبان اس梅بلی 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 دهنی 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 متخلص از سه فاز عمدۀ زیر گردد.

۱- برداشت<sup>۱</sup> دستورالعمل از حافظه

۲- دیکذکردن دستور

۳- اجرای<sup>۲</sup> دستور

در کامپیوتر ثباتی بنام شمارنده برنامه<sup>۳</sup> (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 دهنی 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 دهنی دستور توسط یک کددودویی مجلزا، درست مثل کد عمل مشخص می‌شود. دستهای دیگر از کامپیوترها از یک کد برای تعیین و روش آدرس دهنی دستور استفاده می‌نمایند. دستورات ممکن است با انواع روش‌های آدرس دهنی تعریف شوند و گاهی اوقات دو یا سه روش آدرس دهنی در یک دستور بصورت ترکیب بکار می‌روند.

مثالی از قالب دستورالعمل با میدان روش آدرس دهنی مجلزا در شکل ۸-۶ 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 دهنی خاص خود را دارا باشد.

روش ضمنی: در این روش آدرس دهنی 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 ۸-۶ قالب دستورالعمل با میدان روش (شیوه)

1- Fetch

2- Execute

3- Program Counter

دستورات صفر آدرسه در کامپیوترهای پشتیاهی نیز از نوع روش ضمنی هستند زیرا عملوندها در بالای پشتیه 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 گوییم 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 محتواهی 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 دهن مستقیم در برنامه بکار می‌برد.

**روش خود افزایش یا خود کاهشی:** 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 می‌باید. 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 باید. 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 محتواهی 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 دهن 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<sup>۱</sup> آدرس حافظه‌ای است که از محاسبه‌ای که روش آدرس دهن دیگر

#### 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 دهن را معرفی کردیم. در اینجا مجدداً آنها را بطور خلاصه یادآور می‌شویم.

**روش آدرس دهن مستقیم<sup>۱</sup>:** 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 دهن غیرمستقیم<sup>۲</sup>:** 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 دهن غیرمستقیم قبلاً در بخش ۱-۵ تشریح شد.  
در چند روش آدرس دهن 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 دهن مختلف خواهیم داشت که در کاربردهای مختلف مورد استفاده قرار گیرد.

**روش آدرس دهن نسبی:**<sup>۳</sup> 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 ۲) 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. یعنوان 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 دهن برابرست  $850 = 826 + 24$ . این مکان 24 خانه حافظه بعداز دستور بعدی است. آدرس دهن 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.

روش آدرس دهنده شاخص دار:<sup>۱</sup> در این روش محتوای یک ثبات شاخص<sup>۲</sup> 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 دهنده غیرمستقیم ثبات تبدیل می شود.

برخی از کامپیوترها یک ثبات بtentها برای شاخص اختصاص می دهند. این ثبات بهنگام استفاده از دستورات آدرس دهنده شاخص دار در عملیات دخالت می نماید. در کامپیوتراهایی 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 دهنده بالات 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 دهنده 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 در خود نگه می دارد. یک ثبات پایه آدرس مبنایی رانگه می دارد و میدان آدرس دستور جابجایی نسبت به این آدرس را معین می نماید. روش آدرس دهنده ثبات پایه در کامپیوترا 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<sup>۳</sup>، 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 دهنده، ما تأثیر آدرس دهنده را روی دستورات معرفی شده در شکل ۸-۷ نشان می دهیم. دستور دو کلمه ای واقع در آدرس های ۲۰۰ و ۲۰۱ یک دستور "بارگردان AC" با میدان آدرس ۵۰۰ می باشد. اولین کلمه دستور کد عمل و روش آدرس دهنده و دومین

1- Indexed Addressing Mode

2- Index Register

3- Multiprogramming

آدرس	حافظه
شیوه	باردهی در
200	AC = 200
201	$R_1 = 400$
202	$XR = 100$
399	
400	
500	
600	
702	
800	
	450
	700
	800
	900
	325
	300

شکل A-7 مثال عددی برای  
شیوه‌های آدرس‌دهی

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 مذبور چند آدرس مرتبط با مثال لیست شده و محتوای حافظه هر یک از آدرس‌های مذکور را نشان می‌دهد.

میدان روش<sup>۱</sup> 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 می‌نماییم. در روش آدرس‌دهی 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 اینحالت 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 بعداز فاز برداشت و در طول فاز اجرا 202 است) در روش آدرس دهی شاخص دار، آدرس موثر برابرست با  $600 = 100 + 500 = 100 + 500$  و  $XR + 500$  و عملوند هم 900 است. در روش ثبات عملوند در  $R_1$  و عدد 400 و برابر با محتوای R1 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 ۸-۴ جدول مقادیر مثال عددی

محتوای AC	آدرس موثر	آدرس دهن
500	800	آدرس مستقیم
500	500	عملوند بلافضل
300	800	آدرس غیرمستقیم
325	702	آدرس نسی
900	600	آدرس شاخص دار
400	—	ثباتی
700	400	ثباتی غیرمستقیم
700	400	خودافزایش
450	399	خودکاهش

از اجراء،  $R_1$  افزایش و به 401 تغییر یافته است. در روش خودکاهش، قبل از اجرای دستور،  $R_1$  را به 399 تقلیل می‌دهد. عملوند بارشده در AC برابر 450 است. جدول ۸-۴ مقادیر آدرس موثر و عملوند بارشده در AC را برای 9 روش آدرس دهن لیست نموده است.

## ۸-۶ انتقال و دستکاری داده‌ها

کامپیووترها مجموعه دستورات قابل توجهی را برای ایجاد انعطاف پذیری اعمال محاسباتی در اختیار کاربر قرار می‌دهند. مجموعه دستورالعمل‌های کامپیووترها از نظر روش تعیین عملوند‌ها با یکدیگر اختلاف دارند. ولی اعمال واقعی انجام شده بوسیله دستورات از یک کامپیووتر به کامپیووتر دیگر با هم تفاوت چندانی ندارند. اغلب دیده می‌شود که دودویی تخصیص یافته به دستورات در کامپیووترها مختلفند و این حتی برای دستورات مشابه با هم نیز دیده می‌شود. همچنین ممکن است نام سمبولیک بکار رفته در زبان اسمبولی در کامپیووترهای مختلف متفاوت باشد. در هر صورت در بیشتر کامپیووترها مجموعه‌ای از دستورات دیده می‌شوند که اعمال ساده و پایه‌ای را انجام می‌دهند. مجموعه دستورات اصلی در نمونه‌ای از کامپیووترها موضوع مورد بحث در این بخش و بخش بعدی است.

اکثر دستورالعمل‌های کامپیووتر را می‌توان به سه دسته زیر تقسیم کرد:

- ۱- دستورالعمل‌های انتقال داده<sup>۱</sup>
- ۲- دستورالعمل‌های دستکاری داده‌ها<sup>۲</sup>
- ۳- دستورات کنترل برنامه

دستورالعمل‌های انتقال داده موجب انتقال داده‌ها از یک مکان به مکان دیگر بدون هرگونه تغییر در محتوای اطلاعات دودویی آنها می‌گردند. دستورالعمل‌های دستکاری داده‌ها دستوراتی هستند که اعمال حسابی، منطقی و شیفت را انجام می‌دهند. دستورالعمل‌های کنترل برنامه قابلیت‌های تصمیم‌گیری را

فرام می‌آورند و مسیر برنامه را در حین اجرای برنامه تغییر می‌دهند. مجموعه دستورالعمل‌های یک کامپیوتر بخصوص، اعمال انتقال ثبات و تصمیمات کنترلی موجود را برای استفاده کننده معین می‌نمایند.

### دستورالعمل‌های انتقال داده

دستورالعمل‌های انتقال داده، انتقال داده‌ها را از یک مکان در کامپیوتر به مکانی دیگر بدون هرگونه تغییری در آنها انجام می‌دهد. متداول‌ترین انتقال‌ها بین حافظه و ثبات‌های پردازشگر صورت می‌گیرد. جدول ۸-۵ لیست هشت دستورالعمل انتقال داده را که در بسیاری از کامپیوتراها بکار می‌رود نشان می‌دهد. همراه باهر دستور یک سمبول اختصاری ارائه شده است. باید دانست که کامپیوتراها مختلف سمبول‌های اختصاری مختلفی را استفاده می‌کنند.

دستور LD به معنی بارگردان<sup>۱</sup> اکثراً به هنگام انتقال داده از حافظه به یک ثبات پردازنده که معمولاً انباره است به کار می‌رود. دستور ST به معنی ذخیره کردن<sup>۲</sup>، انتقال از یک ثبات پردازنده به حافظه را مشخص می‌کند. دستورالعمل MOV به معنی انتقال<sup>۳</sup>، در کامپیوتر چند ثباتی برای انتقال داده از ثباتی به ثبات دیگر مورد استفاده قرار می‌گیرد. این دستور برای انتقال داده‌ها بین ثبات‌های CPU و حافظه و نیز بین دو مکان حافظه هم بکار می‌رود. دستور XCH به معنی تعویض<sup>۴</sup>، اطلاعات را بین دو ثبات یا بین یک ثبات و یک حافظه عوض می‌کند. دستورات IN و OUT به معنی ورودی<sup>۵</sup> و خروجی<sup>۶</sup>، داده‌ها را بین ثبات‌ها و پایانه‌های ورودی یا خروجی جابجا می‌کنند. دستورات PUSH (پوش) و POP (پاپ) به ترتیب به معنی فشردن (نشاندن) و کشیدن (بازگرفتن)، داده‌ها را بین ثبات‌های پردازشگر و پشته‌ای در حافظه انتقال می‌دهند.

باید دانست که دستورات جدول ۸-۵ و نیز جداول بعدی در این بخش، اغلب با انواع روش‌های آدرس دهنی همراه‌اند. گاهی در زبان اسمبلي بطور قراردادی تغییری در سمبول‌های اختصاری داده می‌شود که منعکس کننده شیوه آدرس دهنی می‌باشد. مثلاً سمبول اختصاری دستور load immediate به معنی

جدول ۸-۵ دستورالعمل‌های انتقال داده‌ها

نام	سمبول اختصاری
بارگردان	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 بلافصل بصورت LDI در می آید. گاهی نیز قرار داد زیان اسمبلی این است که از کاراکتر خاصی برای مشخص کردن روش آدرس دهن استفاده شود. مثلاً روش بلافصل با قراردادن 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 دهن همراه باشد. مثلاً دستور بارکردن انباره رابا هشت روش آدرس دهن مختلف ملاحظه کنید. جدول ۸-۶ زیان اسمبلی پیشنهادی و انتقال واقعی داده را که در هر مورد صورت می گیرد نشان می دهد. ADR به معنی آدرس، NBR به معنی عدد یا عملوند، X یک ثبات شاخص، R1 یک ثبات پردازنده و AC ثبات انباره است. کاراکتر @ سمبول آدرس غیرمستقیم است. کاراکتر \$ قبل از آدرس، آنرا به درون PC نسبت می دهد. کاراکتر # در دستورات بلافصل قبل از عملوند فرارداده می شود. دستورات شاخص دار با ثباتی در درون پرانتز که پس از آدرس قرار می گیرد شناخته می شود. در آدرس دهن ثباتی نام ثبات حاوی عملوند جلو نام اختصاری دستور قرار می گیرد. در آدرس دهن 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 محصور گشته ثبات از روش ثباتی غیرمستقیم متمایز می شود. در روش خود کاهشی هم از علامت - استفاده می گردد. برای نوشتن برنامه هایی بزیان اسمبلی در کامپیوتر، لازم است که انواع دستور العمل های موجود دانسته شوند و با شیوه ها و روش های آدرس دهن بکار رفته در آن آشنا باشیم.

### دستور العمل های دستکاری داده ها

دستور العمل های دستکاری داده اعمال را روی داده ها انجام می دهند و قابلیت های محاسباتی کامپیوتر را فراهم می آورند. این دستور العمل ها به سه نوع اساسی تقسیم می شوند.

۱- دستور العمل های حسابی

۲- دستور العمل های منطقی و دستکاری های بیتی

۳- دستور العمل های شیفت

لیست دستورات دستکاری داده ها شبیه به لیست ریز عمل ها در فصل ۴ است. با این وجود باید دانست

### جدول ۸-۶ هشت روش آدرس دهن برای دستور العمل بارکردن

روش	فرارداد اسمبلی	انتقال ثباتی
آدرس مستقیم	LD ADR	$AC \leftarrow M[ADR]$
آدرس غیرمستقیم	LD @ADR	$AC \leftarrow M[M[ADR]]$
آدرس نسی	LD SADR	$AC \leftarrow M[PC + ADR]$
عملوند بلافصل	LD #NBR	$AC \leftarrow NBR$
آدرس دهن شاخص دار	LD ADR(X)	$AC \leftarrow M[ADR + XR]$
ثباتی	LD R1	$AC \leftarrow R1$
ثباتی غیرمستقیم	LD (R1)	$AC \leftarrow M[R1]$
خود افزایش	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 دهنی، عملوندها نیز باید به ثبات های پردازشگر منتقل شوند. آخرین مرحله اجرای دستور در پردازشگر است و همین مرحله است که طبق بحث فصل ۴ توسط ریز عمل ها در ALU و یا در شیفت دهنده طبق شکل ۸-۲ اجرا شد. برخی دستورات حسابی هم نیاز به مدارات خاصی جهت پیاده سازی دارند.

### دستورالعمل های حسابی

چهار عمل اصلی حساب عبارتند از جمع و تفریق، ضرب و تقسیم. بسیاری از کامپیووترها دستورالعمل های لازم را برای هر چهار عمل دارا هستند. برخی کامپیووترهای کوچک فقط جمع و احتمالاً دستورات تفریق را دارند. اعمال ضرب، تقسیم باید با نوشتن زیر روال های نرم افزاری تولید گردند. چهار عمل اصلی حسابی فوق برای حل فرمول ها و مسائل علمی پس از بیان آن ها براساس روش های محاسبات عددی کافی هستند.

لیستی از دستورات حسابی در جدول ۸-۷ دیده می شود. دستورافزایش، یک واحد به مقدار ذخیره شده در ثبات می افزاید. یکی از ویژگیهای دستور افزایش پس از اجرا این است که یک عدد دودویی مت Shankl از ۱ در تمام بیت ها وقتی افزایش یابد به عددی تشکیل شده از تماماً ۰ بدل می شود. بالعکس عددی با فقط ۰ پس از کاهش به عددی تماماً ۱ تبدیل می گردد.

دستورات جمع، تفریق، ضرب و تقسیم ممکن است برای انواعی از داده ها موجود باشند. فرض بر این است که نوع داده در ثبات های پردازشگر در طول اجرای این دستورات حسابی در تعریف کد عمل گنجانده شده باشد. یک دستورالعمل حسابی ممکن است یک داده معیز ثبات، معیز شناور، دودویی و یا اعشاری، با دقت کم یا با دقت مضاعف باشد. انواع داده ها در فصل ۳ معرفی شدند.

جدول ۸-۷ دستورالعمل های حسابی

نام	سیبل اختصاری
افزایش	INC
کاهش	DEC
جمع	ADD
تفریق	SUB
ضرب	MUL
تقسیم	DIV
جمع با عقلی	ADDC
تفریق با فرض	SUBB
نفی (منم ۲)	NEG

معمولًاً نمی‌توان کامپیوتروی برای هر سه نوع دستور جمع پیدا کرد؛ یکی برای اعداد صحیح دودویی، یکی برای عملوندها با معیز شناور و یکی هم برای اعداد دهددهی. علامت اختصاری<sup>۱</sup> هر سه نوع دستور جمع که انواع مختلف داده‌ها را بکار می‌برند در زیر نشان داده شده است.

ADD I	جمع دو عدد صحیح دودویی
ADD F	جمع دو عدد معیز شناور
ADD D	جمع دو عدد دهددهی بفرم <sup>۲</sup> BCD

الگوریتم‌های اعداد صحیح، معیز شناور و اعمال حسابی دهددهی در فصل ۱۰ ارائه شده‌اند. تعداد بیت‌های هر ثبات محدود است بنابراین دقت نتایج اعمال حسابی محدودیت خواهد داشت. در برخی از کامپیوتراها، مدارات سخت افزاری، اعمال محاسباتی با دقت مضاعف را ممکن می‌سازند بطوری که طول عملوند تا دو کلمه حافظه می‌رسد. بعضی کامپیوتراها کوچک نیز نرم افزار خاصی را برای محاسبات با دقت مضاعف ارائه می‌نمایند. فلیپ فلاپ خاصی برای ذخیره کردن رقم نقلی یک عمل جمع در نظر گرفته می‌شود. دستور "جمع با رقم نقلی"<sup>۳</sup> عمل جمع بروی دو عملوند بعلاوه رقم نقلی حاصل از عمل قبلی را انجام می‌دهد. مشابهًا دستور "تفريق بافرض"<sup>۴</sup> دو عملوند را با توجه به رقم قرضی احتمالی از عمل قبل، از یکدیگر کم می‌کند. دستور نفی سازی<sup>۵</sup>، مکمل ۲ یک عدد را مشخص می‌نماید که عبارت است از معکوس کردن علامت یک عدد صحیح وقتی که بشكل مکمل ۲ علامت دار در نظر گرفته می‌شود.

### دستورات منطقی و دستکاری بیتی

دستورات منطقی اعمال دودویی را بروی رشته‌ای از بیت‌های ذخیره شده در ثبات‌ها انجام می‌دهند. این دستورات برای بیت‌های جدا از هم و یا گروهی از بیت‌ها 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 می‌کند. با کاربرد صحیح دستورات منطقی امکان تغییر بیت‌ها، امکان پاک کردن گروهی از بیت‌ها یا وارد نمودن مقادیری در بیت‌های عملوند ذخیره شده در ثبات یا کلمات حافظه وجود دارد.

نمونه‌هایی از دستورات منطقی و دستکاری بیتی در جدول ۸-۸ آورده شده‌اند. دستورات العمل پاک کردن<sup>۶</sup> سبب می‌شود عملوند خاصی بالاها جایگزین شود. دستور مکمل سازی<sup>۷</sup>، مکمل ۱ عملوند را با معکوس کردن تمام بیت‌ها تولید می‌نماید. 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	سیبل اختصاری
پاک کردن	CLR
شم کردن	COM
AND	AND
OR	OR
انحصاری OR	XOR
صفر کردن نقلی	CLRC
۱ کردن نقلی	SETC
شم کردن رقم نقلی	COMC
فعال کردن وقه	EI
غیرفعال کردن وقه	DI

را روی بیتهاي عملوند اجرا مي نمایند. هرچند اين دستورالعملها اعمال بولی را انجام مي دهند ولی به آنها بصورت دستکاري کننده بیتها باید نگريست. سه نوع دستکاري بیش امکان پذيراست. بیتهاي که مورد نظر هستند می توانند ۰ شوند، ۱ شوند و یا مكمل شوند. سه دستور منطقی فوق دقیقاً برای انجام همین اهداف مورد استفاده قرار می گيرند.

دستور AND برای پاک کردن يك بیت یا گزیده‌ای از بیتهاي عملوند يك دستورالعمل است. برای هر متغير بولی  $x$  رابطه‌های  $x \wedge 0 = 0$  و  $x \wedge 1 = x$  بیان می دارد که هر متغيری که با ۰ AND شود، تولید ۰ می نماید ولی اگر با ۱ AND شود متغير تغييري نمی یابد. بنابراین دستور AND می تواند برای پاک کردن بیتهاي يك عملوند، يا AND آن عملوند با عملوند دیگري که در مكانهای مورد نظر  $0$ هايی برای پاک نمودن بیتهاي متناظر در عملوند اول دارد بکار رود. دستور AND ماسک<sup>۱</sup> (پوشش) هم خوانده می شود زира در بخش انتخاب شده‌ای از عملوند بیتها را ماسک (صفر) می کند.

دستور OR برای نشاندن<sup>۲</sup> (برقرارکردن) يك بیت یا گروهي از بیتهاي عملوند مورد استفاده قرار می گيرد. برای هر متغير بولی مانند  $x$  رابطه  $1 = 1 + 0 = 1$  و  $0 = 0 + x = x$  بیان می دارد که هر متغيری با ۱، OR شود کميت ۱ را توليد می کند ولی اگر با ۰ OR شود تغييري در متغير وجود نخواهد داشت. بنابراین دستور OR می تواند بروي گزیده‌ای از بیتهاي يك عملوند و با استفاده از عملوند دیگري که در بیتهاي موردنظری از آن ۱ قرار گرفته عمل کرده و بیتهاي متناظر در عملوند اول را ۱ نماید.

بطور مشابه، دستور XOR بر روی بیتهاي منتخبی از عملوند مورد استفاده است. اين دستور نيز با توجه به رابطه بولی  $x' = x \oplus 1$  عمل می نماید. بنابراین يك متغير دو دویی وقتي با ۱، XOR شود مكمل می شود. مثالهای عددی که نشانده‌اند سه عمل منطقی است در بخش (۴-۵) داده شد. چند دستور دیگر دستکاري بیتها در جدول ۸-۸ دیده می شوند. بیتهاي خاصی همچون بیت رقم نقلی می توانند با دستورات مناسبی، پاک، نشانده و یا مكمل شوند. مثالی دیگر فلیپ فلاپ کنترل

وقه 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 دارد. 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، ۰ 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 ۲ صورت می‌گیرد. این قواعد در بخش ۴-۶ آورده شدند. دستور شیفت به راست حسابی باید بیت علامت در سمت چپ ترین 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 ۰ 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. بیشی 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 متزله 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	سبل
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 دویتی برای معرفی یکی از چهار نوع شیفت است؛ RL یک میدان یک بیتی برای چپ و یا راست بودن شیفت می‌باشد و COUNT یک میدان کا بیتی برای  $1 - 2^k$  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 متولیاً برداشت و اجرا می‌گردد. هر بار که دستوری از حافظه برداشته شد، شمارنده برنامه افزایش می‌باید 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 برنامه بسب 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<sup>۱</sup> و پرش<sup>۲</sup> 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	سبل sic
انشعاب	BR
پرش	JMP
گذار	SKP
فراسخوانی	CALL
بازگشت	RET
مقایسه (با تغیریق)	CMP
آزمون (تست) با AND کردن	TST

آدرس دهنده استفاده می شود. انشعباب معمولاً دستور یک آدرسی است و در زبان اسمابلی شکل BR ADR نوشته می شود که در آن ADR نام سمابلیک 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 انشعباب و پرش ممکن است شرطی و یا غیرشرطی باشند. دستور انشعباب غیرشرطی سبب انشعباب به آدرسی دیگر بدون هرگونه شرطی می گردد. دستورات شرطی انشعباب با شرط برقراری شرایطی مانند مثبت یا صفر بودن نتیجه عملی، انشعباب ایجاد می نمایند. اگر شرط برقرار بود، شمارنده برنامه با آدرس انشعباب 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<sup>۱</sup> 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 انشعباب غیرشرطی را در آنجا جای می دهد. بنابراین یک جفت دستور انشعباب-گذر بشرطی انشعباب انجام می دهد که شرط برقرار نباشد در حالیکه یک انشعباب شرطی، بشرطی خواهد کرد که شرط برقرار باشد. دستورات فراخوانی<sup>۲</sup> و بازگشت<sup>۳</sup> بهنگام استفاده از زیرروال<sup>۴</sup> بکار می روند. در مورد این دستورات بعداً در این بخش صحبت خواهد شد. دستورات مقایسه<sup>۵</sup> و تست<sup>۶</sup> مستقیماً توالی اجرا برنامه را عرض نمی کنند. این دستورات هم بعلت کاربردشان در برقراری شرایط برای دستور بعدی انشعباب در جدول ۸-۱۰ آورده شده‌اند. دستور مقایسه تفريقي را بين دو عملوند انجام می دهد ولى نتیجه عمل نگهداري نمی گردد. با اين وجود بيت های وضعیت معینی بعنوان نتیجه مقدار می گيرند. 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 سریز می باشند. تولید این بیت های وضعیت ابتدا بحث شده و سپس چگونگی کاربرد آنها در دستورات انشعباب نشان داده خواهد شد.

#### بیت های وضعیتی<sup>۷</sup>

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<sup>۱</sup> یا بیت‌های پرچم<sup>۲</sup> 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 (۱) و یا پاک می‌شوند (۰).

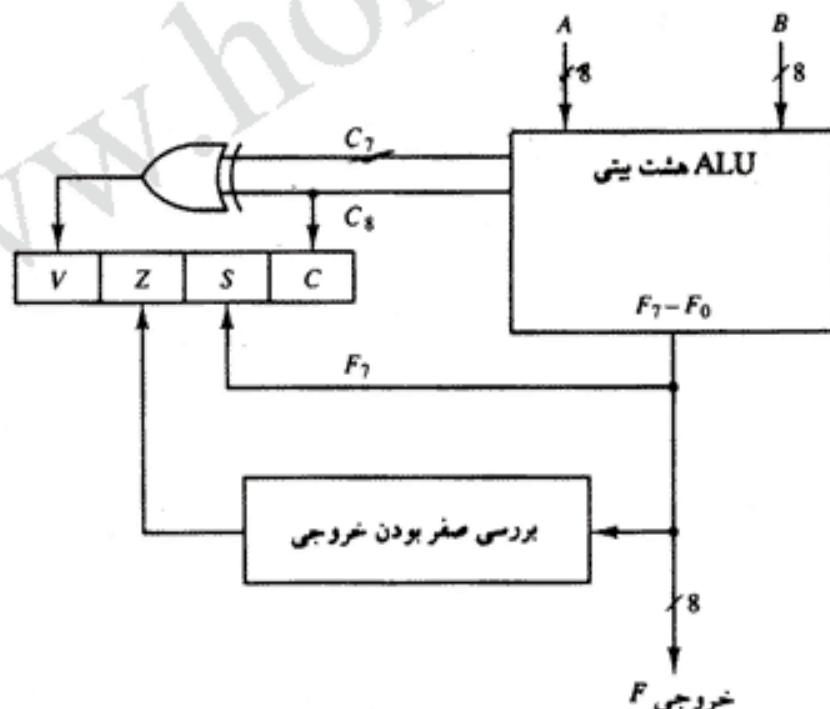
۱- بیت C (نقلی) اگر رقم نقلی انتهایی  $C_8 = 1$  شود نشانده می‌شود (۱). اگر  $C_8 = 0$  شود این بیت ۰ می‌گردد.

۲- بیت S (علامت) نشانده می‌شود بشرطی که بالا رتبه‌ترین بیت یعنی  $F_7$  در ALU برابر ۱ شود. اگر بیت مذکور ۰ گردد S هم ۰ شود.

۳- بیت Z<sup>۳</sup> (صفر) هنگامی ۱ می‌شود که تمام خروجی‌های ALU برابر ۰ باشند. در غیر اینصورت این بیت پاک می‌شود. به بیان دیگر Z=۱ است اگر خروجی صفر و Z=۰ اگر خروجی غیر صفر باشد.

۴- بیت V (سرریز) نشانده می‌شود اگر، XOR دو رقم نقلی آخر برابر ۱ شود و در غیر اینصورت ۰ است. این شرط یک سرریز وقتی اعداد منفی بقلم مکمل ۲ می‌باشد است. (بخش ۳-۳ را ببینید). برای یک ALU با هشت بیت اگر خروجی بیش از  $127 +$  یا کمتر از  $128 -$  باشد، V=۱ می‌گردد.

بیت‌های وضعیت پس از هر عمل در ALU می‌توانند 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  $0 \oplus x = 0$  است که بیان می دارد XOR دو عدد مساوی 0 بوده و این نیز  $Z=1$  را سبب می گردد. هر بیت در  $A$  نیز می تواند بوسیله پوشش  $A=101x1100$  دادن<sup>۱</sup> سایر بیت ها چک شود و سپس بیت وضعیت  $Z$  ملاحظه گردد. مثلاً فرض کنید  $x=1$  باشد که در آن  $x$  بیت مورد آزمایش است. اگر  $x=0$  باشد بیت وضعیت  $Z=1$  می گردد، ولی اگر  $x=1$  باشد  $Z=0$  خواهد شد زیرا نتیجه غیر صفر است. عمل AND می تواند توسعه دستور TEST، اگر حفاظت از محتویات اصلی  $A$  مورد نظر باشد، صورت گیرد.

### دستورات انشعب شرطی

جدول ۸-۱۱ لیستی از دستورات انشعبی متداول را ارائه می دهد. هر سمبول اختصاری تشکیل شده است از حرف  $B$  (برای انشعب) و یک مخفف از کلمه شرط. وقتی حالت مخالف بکار رود، حرف  $N$

جدول ۸-۱۱ دستور العمل های انشعب شرطی

سبل	شرط انشعب	شرط مورد نت
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	$V = 0$
شرط مقایسه 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	$A \neq B$
شرط 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  $BC$  معنی 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 بهمراه دستورات کنترل پوش<sup>۱</sup>، گذرا، 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 بالا رتبه 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 قبل ایجاد شد دستورات مقایسه، عمل تغیریق 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 در باره اندازه A و B فراهم می نماید. برخی کامپیوتراها دستورات شرطی انشعاب را که بلا فاصله 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. وقت کنید که ما اصطلاحات بالاتر<sup>۲</sup> و پائین تر<sup>۳</sup> را برای اعداد بدون علامت و بزرگتر<sup>۴</sup> و کوچکتر<sup>۵</sup> را 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. تغیریق دو عدد چه علامت دار مکمل 2 و چه بدون علامت باشند یکی است (فصل ۳ ملاحظه شود). اجازه بدھید  $A=11110000$  و  $B=00010100$  باشند. در عمل  $A-B$  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  
4- greater

2- Higher  
5- Less Than

3- Lower

دستور مقایسه<sup>۱</sup> بیت های وضعیت را طبق آنچه در فوق دیده می شود تصحیح می نماید.  $C=1$  می شود زیرا یک رقم نقلی از بالاترین رتبه خارج می گردد.  $S=1$  است زیرا سمت چپ ترین بیت ۱ می باشد.  $V=0$  است چون آخرین دو بیت نقلی هر دو ۰ هستند و  $Z=0$  می باشد چون نتیجه حاصل ۰ نیست. اگر ما اعداد را بدون علامت تصور کنیم، عدد اعشاری در A معادل ۲۴۰ و B برابر ۲۰ است. تفریق در مبنای ده  $220 - 240 = 20$  می باشد. نتیجه دودویی حاصل نیز معادل ۲۲۰ است. چون  $20 < 240$  است، داریم  $B > A$  و  $A \neq B$ . این دو رابطه از این واقعیت که  $C=1$  و  $Z=0$  است نیز استنتاج می گردد. دستوراتی که پس از این مقایسه ها می توانند انشعابی ایجاد نمایند، BHI (انشعاب اگر بزرگتر)، BHE (انشعاب اگر بزرگتر یا مساوی) و BNE (انشعاب اگر غیرمساوی) می باشند.

اگر ما اعداد را علامت دار تصور کنیم عدد موجود در A برابر ۱۶ است. زیرا علامت A منفی و ۱۱۱۱۰۰۰۰۰۱ عدد ۰۰۰۱ مکمل ۲ می باشد که معادل  $16 + 2^{15}$  دهدی است. عدد دهدی B هم  $2^{15} + 11011100$  است. تفریق دهدی دو عدد،  $(+20) - (-16)$  خواهد بود. نتیجه دودویی حاصل هم  $(+20) - (-16)$  است داریم  $(M) 00100100$  می باشد که معادل دهدی ۳۶ است. چون  $(+20) < (-16)$  است داریم  $A \neq B$  و  $A < B$ . این دو رابطه از  $S=1$  (منفی) و  $V=0$  (نبودن سرریز) و  $Z=0$  قابل استنتاج است. دستوراتی که پس از مقایسه می توانند 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 BNZ و BNE یکسانند. بطور مشابه و BZ نیز یکی هستند. هر یک از آنها بخاطر وضوح بیشتر مطلب سه بار در جدول ذکر شده اند.

با توجه به مثال فوق واضح است که نسبت اندازه های دو عدد بدون علامت می تواند از مقادیر C و Z (مسئله ۸-۲۶ ملاحظه شود) پس از دستور مقایسه معین شود. نسبت اندازه های اعداد علامت دار با توجه به مقادیر S و V و Z (مسئله ۸-۲۷) مشخص می گردد.

بعضی کامپیوترها بیت C را بیت قرض کردن پس از تفریق A-B فرض می کنند. اگر  $A \geq B$  باشد هیچ قرضی رخ نمی دهد ولی اگر  $A < B$  باشد از بیت بالا رتبه تر عدد قرض گرفته خواهد شد. شرط قرض کردن این است که پس از انجام تفریق با توجه به مکمل ۲ عدد B، مکمل بیت نقلی بدست آید. به این دلیل کامپیوتری که پس از یک تفریق قرضی انجام می دهد، پس از جمع مکمل ۲ مفروق منه بیت C را مکمل نموده و آنرا قرض تلقی می نماید.

## فراخوانی<sup>۲</sup> و بازگشت<sup>۳</sup> از زیرروال

یک زیرروال مجموعه ای مستقل از دستورات است که کار محاسباتی خاصی را انجام می دهد. 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 می گردد. پس از

اجرای زیرروال، انشعابی به برنامه اصلی صورت خواهد گرفت.

دستوری که انتقال کنترل برنامه را به زیرروال بعده دارد با نامهای مختلفی شناخته می‌شود. معمول‌ترین نام عبارتنداز: فراخوانی زیرروال، پرش به زیرروال، انشعاب به زیرروال، یا انشعاب و ذخیره آدرس می‌باشد. 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 موقعی ذخیره می‌گردد و لذا کنترل محل بازگشت را خواهد داشت، و (۲) کنترل به شروع زیرروال منتقل می‌گردد. آخرین دستور هر زیرروال که بازگشت از زیرروال نام دارد، آدرس بازگشت را از مکان موقعت به شمارنده برنامه باز می‌گرداند. نتیجه این عمل انتقال کنترل برنامه به آدرسی است که قبلاً در محل موقع ذخیره شده بود.

کامپیوترهای مختلف، مکانهای موقع 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 فرقانی در پشته فراخوانی و به شمارنده برنامه منتقل گردد. بدین ترتیب، بازگشت همیشه به آخرین زیرروال خواهد بود. احضار یک زیرروال توسط دو ریز عمل 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 والی آخر. دستوری که بازگشت از آخرین زیرروال را انجام می‌دهد توسط ریز عمل‌های 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 بازگشته<sup>۱</sup> زیروالی است که خودش را فرا می‌خواند. اگر تنها یک ثبات یا حافظه برای

ذخیره کردن آدرس برگشت بکار رود، 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 گردد. 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 سرویس دهنده وقفه است که پس از یک تقاضای داخلی یا خارجی صورت می‌گیرد. پس از اجرای برنامه سرویس دهنده وقفه، کنترل مجدداً به برنامه اصلی باز می‌گردد.

رویه<sup>۱</sup> وقفه در اصل مشابه با یک فراخوانی زیرروال است، بجز در سه مورد زیر: (۱) وقفه معمولاً توسط یک سیگنال داخلی یا خارجی رخدنده و نه با اجرای یک دستور العمل (بجز برای وقفه نرم افزاری که بعداً توضیح داده شده است); (۲) آدرس برنامه سرویس دهنده 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 می‌نماییم.

پس از هر وقفه در برنامه اصلی واجرای 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<sup>۲</sup>، PSW خوانده می‌شود. 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 (سوپر وایزری) است و یا در اختیار کاربر<sup>۳</sup> می‌باشد. بسیاری از کامپیوترها دارای یک سیستم عامل مقیمه هستند که تمام برنامه‌های دیگر را کنترل و نظارت می‌کنند.

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 محتواهای ثبات‌ها وجود نداشته و هر یک از مجموعه ثبات‌های خود استفاده می‌نمایند.

گفتنیم که روش یا رویه سخت افزاری پردازش وقفه مشابه با فراخوانی یک زیرروال است. در هنگام قوع وقفه وضعیت CPU به داخل حافظه پشته پوش می‌شود و آدرس شروع رووال سرویس دهی به شمارنده برنامه منتقل می‌گردد. آدرس شروع رووال 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 می‌گذارند. در هر حال، CPU باستنی نوعی رویه سخت افزاری را برای انتخاب آدرس انشعباب به منظور سرویس دهی به وقفه داشته باشد.

CPU تا فرارسیدن انتهای اجرای یک دستور به وقفه پاسخ نمی‌دهد. کترل، درست قبل از فاز دریافت بعدی، وجود سیگنال وقفه را چک می‌کند. اگر وقفه‌ای تقاضا شده باشد، کترول وارد سیکل وقفه سخت افزاری می‌شود. در طول این سیکل، محتویات PC و PSW به داخل پشته پوش داده می‌شوند. سپس آدرس انشعباب مربوط به وقفه خاص مورد نظر به PC منتقل می‌شود و PSW جدیدی در ثبات وضعیت بار می‌شود. بدین ترتیب، برنامه سرویس دهی می‌تواند با شروع از آدرس انشعباب و وضعیتی که PSW جدید مشخص می‌کند اجرا شود.

آخرین دستور در برنامه سرویس وقفه بازگشت از وقفه<sup>۱</sup> 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

۱- Return from Interrupt

شده‌اند.

- ۱- وقفه خارجی
- ۲- وقفه داخلی
- ۳- وقفه نرم افزاری

وقفه خارجی از وسایل ورودی - خروجی<sup>۱</sup> (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 متقارضی 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<sup>۲</sup> از قرار گرفتن احتمالی برنامه در یک حلقه بین پایان 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 یا اشتباه یک دستور و یا داده ناشی می‌شود. این وقفه را تله<sup>۳</sup> یا ھام هم می‌خوانند. مثالهایی از این نوع وقفه‌ها عبارتند از، سرریز<sup>۴</sup> ثبات، تقسیم بر صفر، کد عمل غیرمعتبر، سرریزپشته و نادیده گرفتن مقررات حفاظتی. این خطاهای 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 هریار 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 می‌افتد. 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 بیای ایجاد یک وقفه نرم افزاری در هر نقطه مطلوب از برنامه بکار رود. متدائل‌ترین کاربرد وقفه نرم افزاری به دستور العمل 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 نمود. 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 فراخوانده باید اطلاعات را به سیستم عامل جهت انجام کار خاص تحويل نماید.

## ۸-۸ کامپیوتر کم دستور<sup>۱</sup> RISC

یکی از اهداف مهم آرشیتکت 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. کامپیوتراهای اولیه مجموعه دستورات ساده و کوچکی را داشتند که این خود بیشتر بعلت اجراء 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 کامپیوتراها در حال حاضر بیش 100 و حتی 200 دستورالعمل را دارا هستند. بعلاوه این کامپیوتراها از انواع داده‌ها و تعداد زیادی روش‌های آدرس دهنی سود می‌برند. سیرحرکت 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<sup>۲</sup> (CISC) یا پردازور می‌نامند.

در اوایل 1980، تعدادی از سازندگان، طراحی کامپیوترا را 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 کامپیوتراها با نام کامپیوترا 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.

### CISC مشخصه‌های

در طراحی یک مجموعه دستورالعمل نه تنها ساختار زبان ماشین بلکه نیازهای زبان‌های سطح بالا را نیز باید در نظر گرفت. ترجمه برنامه با زبانهای سطح بالا به زبان ماشین، توسط کامپایلر (یا مترجم) انجام می‌شود. یکی از دلایل روی آوردن سازندگان به مجموعه دستورات CISC ساده‌کردن کار ترجمه فوق و ارتقای عملکرد کلی کامپیوترا 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 نیهی یک دستورالعمل برای هر عبارتی است که در زبان سطح بالا نوشته

1- Reduced 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 بایتی باشند ولی دستوراتی که دو آدرس حافظه را لازم دارند می‌توانند برای تمام کد دستور تا پنج بایت را استفاده نمایند. اگر کلمات کامپیوتر ۳۲ بیتی (چهار بایت) باشند، اولین دستور نیمی از کلمه را اشغال می‌کند، در حالیکه دومین دستور به یک کلمه و یک بایت از کلمه بعدی احتیاج دارد. قرار دادن قالبهای متغیر دستورالعمل در کلمه‌های حافظه، با طول ثابت، نیاز به مدار دیکدر خاصی دارند که قادر است تعداد بایت را در کلمه شمرده و دستورات را بربطی تعداد بایت آنها دسته‌بندی نماید.

دستورالعمل‌های یک نمونه پردازشگر CISC امکان استفاده مستقیم از عملوندها را در حافظه فراهم می‌آورند. مثلاً، یک دستور ADD می‌تواند دارای یک عملوند در حافظه با روشن آدرس دهی شاخص دار و یک عملوند در حافظه از طریق آدرس دهی مستقیم باشد. حافظه دیگری را هم می‌توان در دستور برای ذخیره کردن نتیجه منظور کرد. درنتیجه در طول اجرای دستور سه بار ارجاع به حافظه وجود خواهد داشت. هر چند پردازشگرهای CISC دارای دستوراتی هستند که فقط از ثبات‌ها استفاده می‌نمایند. وجود روش‌های آدرس دهی دیگر در عملیات، کامپایل کردن زبان‌های سطح بالا را ساده می‌نماید. با این وجود، هر چه تعداد دستورات و روش‌های آدرس دهی در کامپیوتر بیشتر شود، مدارهای سخت افزاری بیشتری برای پیاده کردن و پشتیابی آنها لازم است و این سبب می‌گردد تا سرعت محاسبات کاهش یابد.

بطور خلاصه مهمترین مشخصه‌های یک معماری CISC عبارتند از:

- ۱- تعداد زیاد دستورالعمل‌ها - معمولاً بین ۱۰۰ تا ۲۵۰ دستور
- ۲- دستورالعمل‌هایی که کارهای خاصی انجام می‌دهند ولی بندرت بکار می‌روند.
- 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 می‌کند.

### RISC مشخصه‌های

هدف از معماری RISC کوتاه کردن زمان اجرا، با کاهش مجموعه دستورات در کامپیوتر است. مهمترین مشخصه‌های پردازشگر RISC بقرار زیرند.

- ۱- دستورات نسبتاً کم
- ۲- روش‌های آدرس دهی نسبتاً کم
- ۳- دستیابی به حافظه منحصر است به دستورات بارکردن<sup>۱</sup> و ذخیره سازی<sup>۲</sup>
- ۴- تمام اعمال در داخل ثبات‌های CPU انجام می‌شوند.

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 ثبات - ثبات تشکیل شده و برای دستیابی به حافظه نیز دستورات ساده بارگردان و ذخیره سازی نیز در نظر گرفته شده است. بنابراین هر عملوندی ابتدا توسط دستور بارگردان به یک ثبات داخلی منتقل می شود. تمام محاسبات روی داده های ذخیره شده در ثبات صورت می گیرد. نتایج هم بكمک 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 دستورالعمل هاست. سایر روش های 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 دیکشدن است. بنابراین کنترل می تواند به کد عمل و میدان ثبات بطور همزمان دست یابد. با ساده کردن دستورات و قالب آن ها، مدار کنترل را می توان ساده کرد و برای بالا بردن سرعت عملیات، یک کنترل سخت افزاری بر نوع 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<sup>۱</sup> است. دستورات بارگردان 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 ثبات در واحد پردازشگر

۲- استفاده از دریچه های ثبات هم پوش<sup>۲</sup> برای بالا بردن سرعت دستورات فراخوانی و بازگشت از رویه

۳- خط لوله کارا در ثبات ها

۴- پشتیانی کامپایلر برای انتقال کارآی برنامه های زبان های سطح بالا به برنامه های زبان ماشین

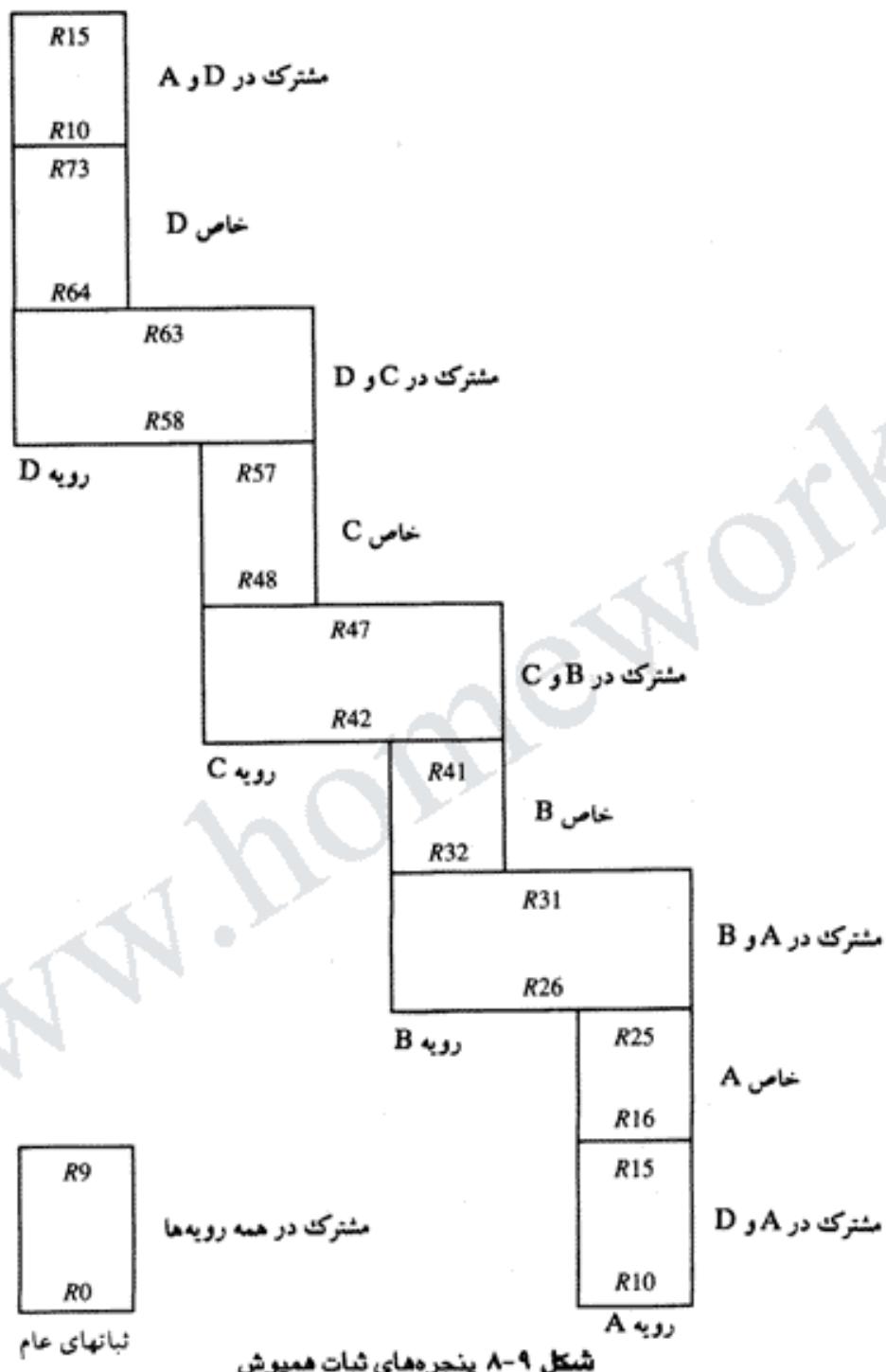
بالا بردن تعداد ثبات‌ها برای ذخیره کردن نتایج میانی و کاهش تعداد ارجاعات به عملوندها مفید است. مزیت مکان‌های ذخیره از نوع ثبات در مقابل حافظه این است که عمل انتقال در ثبات‌ها بسیار سریعتر از حافظه‌های است. بنابراین عمل انتقال ثبات به حافظه را با نگهداری عملوند پرمصرف‌تر در ثبات می‌توان سرعت بخشد. مطالعات نشان می‌دهد که اصطلاحات در عملکرد معماری توأمًا 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 دریچه‌های (پنجه) ثباتی هم پوش هنگام انتقال کنترل برنامه پس از یک فراخوانی رویه توضیح داده شده است. خط لوله دستورات در RISC در بخش ۹-۵ 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 تحويل می‌نمایند و سپس زیرروالی را برای اجرای رویه فرامی‌خوانند. پس از بازگشت از رویه، برنامه مقادیر قبلی ثبات‌ها را در مکان‌هایشان می‌نشاند، نتایج را به برنامه فراخوانده تحويل و از زیرروال باز می‌گردد. ذخیره و بازیابی ثبات‌ها و تحويل 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 تحويل پارامترها و اجتناب از ذخیره و بازیابی مقادیر ثبات‌های است. فراخوانی هر رویه منجر به تخصیص یک دریچه جدید مشکل از مجموع ثبات‌ها از فایل 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 ۸-۹ تشریح شده است. سیستم دارای ۷۴ ثبات است. ثبات‌های R<sub>0</sub> تا R<sub>9</sub>، 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 مشکل از ده ثبات محلی (درونوی) و دو مجموعه شش ثباتی مشترک با دریچه‌های مجاور است. ثبات‌های محلی برای متغیرهای محلی استفاده می‌شوند. ثبات‌های مشترک برای تعویض پارامترها و



شکل ۹-۹ پنجره های 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 دریچه ثبات فعال می‌شود و اشاره‌گر نیز دریچه فعال را نشان خواهد داد. هر فراخوانی رویه یک دریچه ثبات جدید را با افزایش اشاره‌گر فعال

می نماید. ثبات های فوقانی فراخوانده بروی ثبات های تحتانی فراخوانده شده قرار دارند (هم پوشند) و لذا پارامترها بطور اتوماتیک از رویه فراخوانده به رویه فراخوانده منتقل می شوند.

بعنوان مثال، فرض کنید که رویه A، رویه B را فرا بخواند. ثبات های R<sub>26</sub> تا R<sub>31</sub> بین هر دو رویه مشترکند، و بنابراین رویه A پارامترها را برای رویه B در این ثبات ها ذخیره می کند. رویه B از ثبات های محلی R<sub>32</sub> تا R<sub>42</sub> برای ذخیره متغیرهای محلی استفاده می کند. اگر رویه B، رویه C را فرا بخواند، پارامترها را از طریق R<sub>42</sub> تا R<sub>47</sub> به آن تحویل می نماید. وقتی که در انتهای محاسبات رویه B آماده بازگشت به رویه A باشد، برنامه نتایج محاسبات را در ثبات های R<sub>31</sub> تا R<sub>32</sub> ذخیره کرده و به دریچه ثبات های رویه A بازمی گردد. توجه کنید که ثبات های R<sub>10</sub> تا R<sub>15</sub> بین رویه A و رویه D مشترک است زیرا چهار دریچه ساختار حلقوی دارند و لذا A با D مجاور است.

همانطور که قبل اشاره شد، ده ثبات عام R<sub>0</sub> تا R<sub>9</sub> مورد استفاده همه رویه ها 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 می باشند. انواع دیگری از ساختارهای دریچه های ثباتی با ابعاد ثابت نیز امکان پذیر است، و هر یک ممکن است از نظر سایز دریچه ها و سایز کل فایل ثبات ها با دیگری اختلاف داشته باشند. بطور کلی سازمان دریچه های ثبات روابط زیر را دارا هستند.

- ۱- تعداد ثبات های عام G
- ۲- تعداد ثبات های محلی در هر دریچه L
- ۳- تعداد ثبات های مشترک بین دو دریچه C
- ۴- تعداد دریچه ها W

تعداد ثبات های قابل دسترسی توسط هر دریچه از رابطه زیر محاسبه می شود

$$\text{سایز دریچه} = L + 2C + G$$

تعداد ثبات های مورد نیاز در پردازشگر برابرست با

$$\text{فایل ثبات} = (L + C) W + G$$

در مثال شکل ۹-۸ داریم  $L = 10$ ,  $G = 10$ ,  $C = 6$ ,  $W = 4$ . سایز دریچه  $32 = 10 + 12 + 10 = 32$  و فایل ثبات منشکل از 74 ثبات خواهد بود زیرا  $(10 + 6) \times 4 + 10 = 74$ .

### Berkeley RISC I

یکی از پژوههایی که مزایای معماری RISC را نشان داد در دانشگاه برکلی در ایالت کالیفرنیا انجام شد. Berkeley RISCI یک CPU مدار مجتمع 32 بیتی است. این CPU دارای 32 بیت آدرس بوده و

داده های 8، 16 و 32 بیتی را پشتیبانی می کند. قالب دستورات نیز 32 بیتی است و کلاً 31 دستورالعمل دارد. سه روش آدرس دهنی اساسی را داراست: آدرس دهنی ثباتی، عملوندهای بلافصل و آدرس دهنی نسبت به PC برای دستورات انشعاب. فایل ثبات آن دارای 138 ثبات دسته بندی شده به ده ثبات عام و هشت دریچه 32 ثباتی در هر کدام است. تعداد 32 ثبات در هر دریچه، سازمانی مشابه با شکل ۸-۹ دارد. چون در هر زمان معین فقط 32 ثبات در هر دریچه قابل دستیابی هستند، قالب دستور می تواند دارای پنج بیت برای میدان ثبات باشد.

شکل ۸-۱۰ قالب دستورات بکار رفته برای دستورات ثبات - ثبات و دستیابی به حافظه را نشان می دهد. هفت بیت موجود کد عمل، یک عمل خاص را مشخص می کنند و بیت هشتم معین می کند که آیا بیت های وضعیت پس از عمل ALU باید بهنگام<sup>۱</sup> شوند یا خیر. برای دستورات ثبات - ثبات میدان پنج بیتی Rd یکی از 32 ثبات را عنوان مقصد<sup>۲</sup> برای نتیجه عمل انتخاب می کند. عمل توسط داده های موجود در میدان های Rs و S2 معین می گردد. Rs یکی از ثبات های مبدأ<sup>۳</sup> (منبع) است. اگر بیت 13 از دستورالعمل 0 باشد پنج بیت پائین رتبه 2 ثبات مبدأ دیگر را معین می کند. اگر بیت 13 از دستورالعمل 1 باشد، S2 مشخص کننده یک مقدار ثابت 13 بیتی با تعمیم علامت<sup>۴</sup> است. بنابراین دستورالعمل دارای سه قالب آدرس است، ولی دومین مبدأ معکن 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 می کند

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 می نماید

31	24 23	19 18	0
کد عمل	COND		Y
8	5		19

(ج) شیوه نسبت به PC

شکل ۸-۱۰ قالب های دستورالعملها در I Berkeley RISC

1- Update

2- Destination

3- Source

4- Sign Extended

دستورات دستیابی به حافظه از RS برای تعیین آدرس 32 بیتی در ثبات استفاده می‌کنند و S2 یک فاصله از مبدأ است. محتوای ثبات R0 تماماً 0 است بنابراین در هر حال بعنوان کمیت صفر می‌تواند مورد استفاده قرار گیرد. سومین قالب دستور سه میدان آخر را با هم ترکیب کرده و یک آدرس نسبی ۱۹ بیتی Y را تشکیل داده است و عمدها با دستورات فراخوانی و پرش بکار می‌رود. میدان COND که بجای میدان Rd در دستور العملها آمده است برای تعیین یکی از 16 شرط انتساب استفاده می‌شود.

31 دستور RISC در جدول (۸-۱۳) لیست و به سه گروه تقسیم شده‌اند. دستورات دستکاری داده اعمال حسابی، منطقی و شیفت را انجام می‌دهند. سمبول زیر ستون کد عمل و عملوند بهنگام نوشتن برنامه‌هایی بزیان اسمبلی بکار می‌روند. ستون‌های انتقال ثبات و توضیحات، عمل دستورات را بصورت علائم و نیز تشریحی بیان می‌دارند. دقت کنید که تمام دستورات دارای سه عملوند هستند. مبدأ S2 می‌تواند 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 می‌توان 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، می‌توان محتوای یک ثبات یا یک مقدار ثابت را به ثبات دیگری منتقل کرد. عمل افزایش هم با جمع ثبات 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 می‌باشد. محتويات ثبات PSW وضعیت CPU را نشان می‌دهد که عبارتست از: شمارنده برنامه، بیت‌های وضعیت از ALU، اشاره‌گرهای بکار رفته با دریچه‌های ثباتی و سایر اطلاعاتی که وضعیت CPU را مشخص می‌نماید.

دستورات بارکردن و ذخیره سازی، داده‌ها را بین ثبات و حافظه حرکت می‌دهند. دستورات بارکردن داده‌های علامت دار یا بدون علامت هشت بیتی (بایت) و یا 16 بیتی (کلمه کوتاه) را دارا هستند. دستورات کلمات بلند روی داده‌های 32 بیتی عمل می‌نمایند. هر چند بمنظور می‌رسد که روش آدرس دهنی ثباتی بعلاوه جابجایی در دستورات انتقال داده وجود دارد، روشهای آدرس دهنی غیرمستقیم ثبات و مستقیم نیز امکان پذیر است. مثالهای زیر دستورات بارکردن با کلمات بلند را با روشهای مختلف نشان می‌دهد.

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
<b>دستورالعمل‌های دستکاری داده‌ها</b>			
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	$Rd \leftarrow Rs - S2$	تفاضل صحیح
SUBC	Rs,S2,Rd	$Rd \leftarrow Rs - S2 - \text{carry}$	تفاضل با نقلی
SUBR	Rs,S2,Rd	$Rd \leftarrow S2 - Rs$	تفاضل معکوس
SUBCR	Rs,S2,Rd	$Rd \leftarrow S2 - Rs - \text{carry}$	تفاضل 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	$Rd \leftarrow Rs \oplus S2$	انحصاری OR
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 منطبقی به راست
SRA	Rs,S2,Rd	$Rd \leftarrow Rs \text{ shifted by } S$	شیفت حسابی به راست
<b>دستورالعمل‌های انتقال داده</b>			
LDL	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	بارگردان عملوند بلند
LDSU	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	بارگردان عملوند کوتاه بی علامت
LDSS	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	بارگردان عملوند کوتاه علامت دار
LDBU	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	بارگردان بایت بی علامت
LDBS	(Rs)S2,Rd	$Rd \leftarrow M[Rs + S2]$	بارگردان بایت علامت دار
LDHI	Rd,Y	$Rd \leftarrow Y$	بارگردان کمیت بلافضل 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	$Rd \leftarrow PSW$	بارگردان کلمه وضعیت
PUTPSW	Rd	$PSW \leftarrow Rd$	مقداردهی به کلمه وضعیت
<b>دستورالعمل‌های برنامه</b>			
JMP	COND, S2(Rs)	$PC \leftarrow Rs + S2$	پرش شرطی
JMPR	COND,Y	$PC \leftarrow PC + Y$	پرش نسبی
CALL	Rd,S2(Rs)	$Rd \leftarrow PC$ $PC \leftarrow Rs + S2$ $CWP \leftarrow CWP - 1$	فراخوانی زیرروال و تغییر پنجره
CALLR	Rd,Y	$Rd \leftarrow PC$ $PC \leftarrow PC + Y$ $CWP \leftarrow CWP - 1$	فراخوانی نسبی و تغییر پنجره
RET	Rd,S2	$PC \leftarrow Rd + S2$ $CWP \leftarrow CWP + 1$	بازگشت و تغییر پنجره
CALLINT	Rd	$Rd \leftarrow PC$ $CWP \leftarrow CWP - 1$	غیرفعال کردن وقتهای
RETINT	Rd,S2	$PC \leftarrow Rd + S2$ $CWP \leftarrow CWP + 1$	فعال کردن وقتهای
GTLPC	Rd	$Rd \leftarrow PC$	دربافت آخرين 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 دهن 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 اشاره گر دریچه جاری (CWP)<sup>۱</sup> 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 پائین تر 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 ۸-۲ دارای 16 ثبات 32 بیتی، یک ALU و یک دیکدر مقصد یا ب است.

- الف - چند مولتی پلکسر در گذرگاه A وجود دارد و سایز 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 ۸-۲ دارای تأخیرهای زمانی زیر است: 30 ns برای انتشار در MUX، 80ns برای عمل جمع در ALU. 20ns تأخیر در دیکدر مقصد یا ب 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 ۸-۲ اعمال شود تا اعمال زیر را انجام دهد مشخص کنید.

$$\text{الف} - \text{R1} \leftarrow \text{R2} + \text{R3}$$

$$\text{ب} - \text{R4} \leftarrow \text{R4} - \text{R5}$$

$$\text{ج} - \text{R5} \leftarrow \text{R5} - 1$$

$$\text{د} - \text{R6} \leftarrow \text{ShlR1}$$

$$\text{ه} - \text{R7} \leftarrow \text{input}$$

۸-۴ عملیات جزیی قابل اجرا در پردازشگر شکل ۲-۸ را وقتی که کلمات کنترل ۱۴ بیتی زیر اعمال می شود معین کنید.

(الف) 00101001100101

(ب) 0000000000000000

(ج) 01001001001100

(د) 00000100000010

(ه) 11110001110000

۸-۵ اجازه بدید  $SP = 000000$  در پشته شکل ۸-۳ 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 ۸-۴ می تواند 4000 بوده و اولین کمیت در مکان 4000 از پشته ذخیره شود. برای اعمال پوش و پاپ عملیات جزیی را لیست نمائید.

۸-۷ عبارت 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 عبارت 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 پشتکل FIF0 است. نشان دهید چگونه یک حافظه FIF0 با سه شمارنده کار می کند. یک شمارنده نوشتن<sup>۱</sup>، WC، آدرس نوشتمن در حافظه رانگه می دارد. یک شمارنده

خواندن<sup>۱</sup> RC آدرس خواندن در حافظه را حفظ می‌نماید. یک شمارنده فضای موجود در حافظه<sup>۲</sup>، تعداد کلمات ذخیره شده در FIFO را معین می‌نماید. برای هر کلمه ذخیره شده یک واحد اضافه می‌شود و بر عکس برای کلمات دریافت شده یک واحد کم می‌گردد.

برنامه‌ای 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 ثبات‌های عمومی و دستورات سه آدرسه  
 (ب) با استفاده از کامپیوتر با ثبات‌های عمومی و دستورات دو آدرسه  
 (ج) با استفاده از کامپیوتر نوع انباره‌ای با دستورات یک آدرسه  
 (د) با استفاده از کامپیوتر با سازمان پشته و دستورات صفر آدرسه

۸-۱۳ واحد حافظه یک کامپیوتر  $k=256$  کلمه 32 بیتی دارد. کامپیوتر دارای قالب دستورات چهار میدانی است: میدان کد عملیات، میدان روش آدرس دهنده برای هفت روش موجود، میدان آدرس ثبات برای انتخاب یکی از 60 ثبات پردازشگر، و آدرس حافظه. قالب دستور و تعداد بیت‌ها در هر میدان را اگر دستور در یک کلمه حافظه باشد معین کنید.

۸-۱۴ یک دستور دو کلمه‌ای 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  $Z$  می‌باشد. یک ثبات شاخص دارای مقدار  $X$  است. نشان دهید که چگونه  $Z$  از سایر آدرس‌ها محاسبه می‌شود اگر که روش آدرس دهنده دستور برابر زیر باشد.

- (الف) مستقیم  
 (ج) نسبی  
 (ب) غیرمستقیم  
 (د) شاخص دار

۸-۱۵ یک دستور انشعاب از نوع نسبی در آدرس دهدی 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 دو دویی در 12 بیت نشان دهید (چرا باید عدد بفرم مکمل 2 باشد).

(ج) مقدار دو دویی در PC را پس از فاز برداشت (واکشی) بدست آورید و مقدار 500 را محاسبه کنید. سپس نشان دهید که مقدار دو دویی در PC بعلاوه آدرس نسبی محاسبه شده در قسمت (ب) برابر عدد دو دویی 500 است.

۸-۱۶ واحد کنترل وقتی که یک دستور با آدرس غیرمستقیم را برداشت و اجرا می‌کند 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 انتساب 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 مؤثر را اگر روش آدرس دهنی 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 فرق باشد.

۸-۲۳ اعداد علامت دار زیر را به دودویی بصورت 8 بیت نمایش دهید، +63، -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 دهدی بدمست آورید.

د- عدد دودویی 83 را یکبار به چپ شیفت دهید و بینید 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، Z، S 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 نماید.

ج) عملوند بلافصل 9A را از R کم کنید.

د) R را با  $\text{R} \oplus \text{XOR}$  نمائید.

A-۲۶ دو عدد بدون علامت توسط رابطه  $A - B$  باهم مقایسه شده‌اند. بیت نقلی یعنوان بیت قرض پس از مقایسه در بیشتر کامپیوترهای تجاری در نظر گرفته می‌شود، بطوری که اگر  $A < B$  باشد است. نشان دهید که نسبت اندازه A نسبت به B از وضعیت C و Z طبق جدول مربوط به این مسئله و (جدول A-۱۱) قابل استنتاج است.

جدول مسئله A-۲۶

رابطه	حالت بیت‌های وضعیت
$A > B$	$C = 0$ and $Z = 0$
$A \geq B$	$C = 0$
$A < B$	$C = 1$
$A \leq B$	$C = 1$ or $Z = 1$
$A = B$	$Z = 1$
$A \neq B$	$Z = 0$

A-۲۷ دو عدد علامت دار بشکل مکمل 2 با هم طبق رابطه  $A-B$  مقایسه شده‌اند. با توجه به نتیجه مقایسه بیت‌های وضعیت S و Z و V یا 1 خواهد شد. (دقیت کنید که اگر سریز رخ دهد تغییر علامتی وجود خواهد داشت). نشان دهید که اندازه نسبی A نسبت به B با توجه به بیت‌های وضعیت در جدول مربوط به این مسئله (جدول A-۱۱) قابل استنتاج است.

جدول مسئله A-۲۷

رابطه	حالت بیت‌های 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 $Z = 1$
$A = B$	$Z = 1$
$A \neq B$	$Z = 0$

A-۲۸ قرار است یک سیستم دیجیتال با چهار ورودی C و S و Z و V با 10 خروجی طراحی شود، که هر یک مربوط به یک شرط انشعاب در مسئله A-۲۶ و A-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  $B=10000100$  مفروض است.

الف - معادل دهدی هر عدد را با فرض اینکه (۱) آنها بدون علامت هستند و (۲) آنها علامت دارند بدست آورید.

ب - دو عدد دو دویی را جمع کنید و نتیجه را تفسیر کنید با این فرض که (۱) آنها بدون علامت هستند (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 دو دویی را تفسیر کنید.

ب - مقادیر بیت های وضعیت C (فرض) و Z را معین کنید.

ج - دستورات انشعاب شرطی را از جدول ۸-۱۱ که اتفاق می افتد<sup>۱</sup> 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  $B=10000100$  باشد،

الف - اختلاف را محاسبه و نتیجه دو دویی 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 افتد 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 و SP 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 نمائید. 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 ۳۷۰ IBM و VAX11 و Intel 386. قالب دستورات یکی از آنها را با دستورات با طول ثابت RISCI مقایسه کنید.

۸-۳۷ سه کامپیوتر از دریچه های ثبات زیر استفاده می کنند. سایز دریچه ها و تعداد کل ثبات در هر ثبات چقدر است؟

	کامپیوتر ۱	کامپیuter ۲	کامپیوتر ۳
ثبات های عام	10	8	16
ثبات های محلی	10	8	16
ثبات های مشترک	6	8	16
تعداد دریچه ها	8	4	16

۸-۳۸ مثالی از یک دستور کامپیوتر RISCI که اعمال زیر را انجام دهد ارائه کنید.

الف - یک ثبات را کاهش دهد.

ب - یک ثبات را مکمل کند.

ج - علامت محتوای ثبات را معکوس می کند

د - یک ثبات را ۰ کند.

ه - یک عدد علامت دار را به ۴ تقسیم کند.

و - هیچ کاری انجام ندهد.

۸-۳۹ دستوری از RISCI را به زبان اسمنبلی بنویسید که سبب پرسش به آدرس 3200 گردد بشرطی که بیت وضعیت  $Z=1$  باشد.

الف - با استفاده از روش بلافصل

ب - با استفاده از روش نسبی ( $3400 = PC$  فرض شود)

# پردازش خط لوله‌ای و برداری



- ۹-۱ پردازش موازی
- ۹-۲ خط لوله
- ۹-۳ خط لوله حسابی
- ۹-۴ خط لوله دستورالعمل

- ۹-۵ خط لوله RISC
- ۹-۶ پردازش برداری
- ۹-۷ پردازشگر آرایه

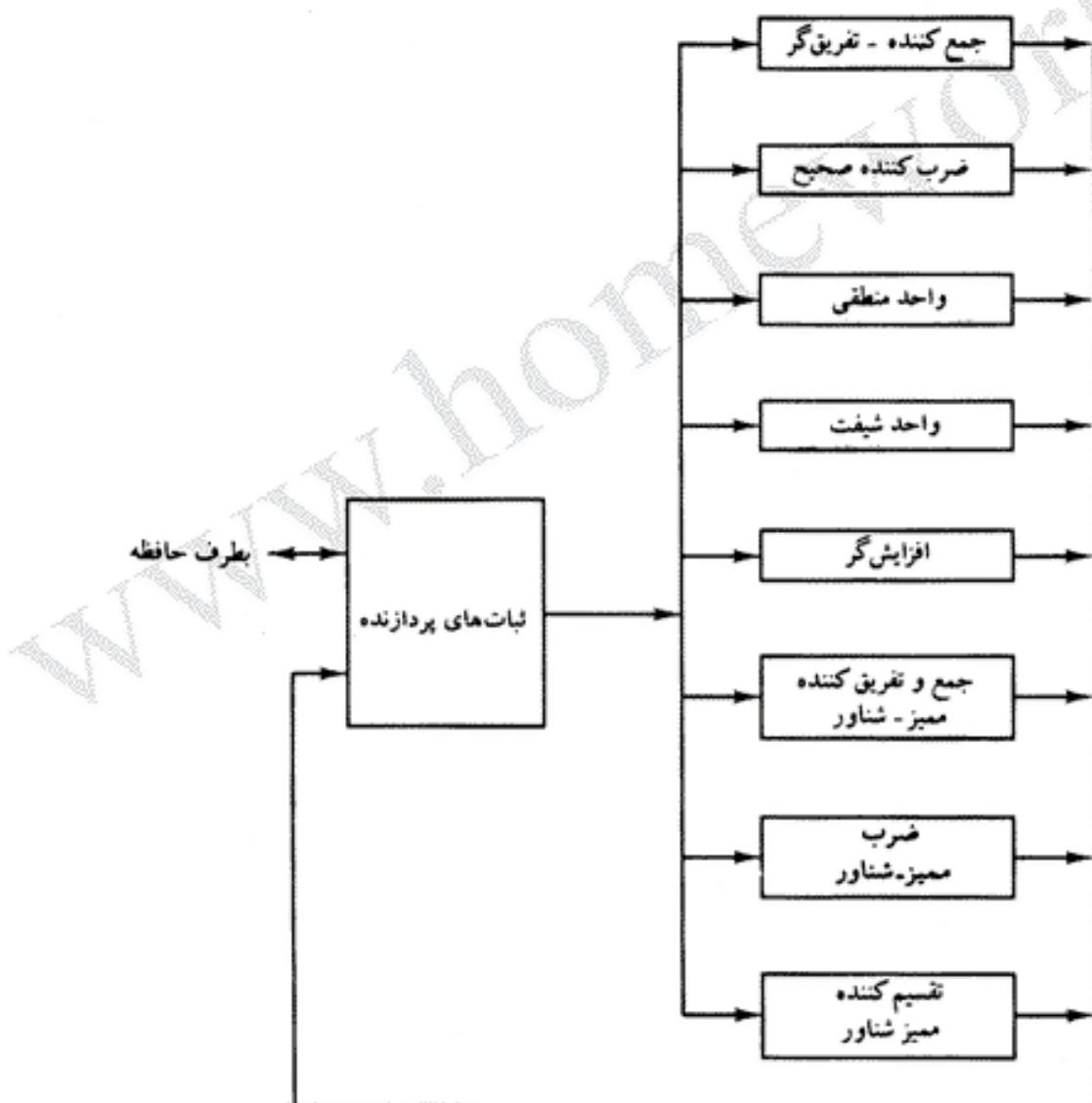
## ۹-۱ پردازش موازی

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 ALL، دستور بعدی قابل خواندن از حافظه است. سیستم ممکن است دارای دو یا چند ALL 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 سخت افزاری را تا حدی پائین آورده است که در پردازش موازی کاملاً محسوس است. پردازش موازی را می‌توان از زوایای مختلفی بررسی کرد. ما در پائین 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<sup>۱</sup>

1- Functional unit

چندگانه است که اعمال یکسان یا متفاوتی را بطور همزمان انجام می‌دهند. در این حال پردازش موازی با توزیع داده‌ها در میان این واحدها صورت می‌گیرد. مثلاً واحد حساب، منطق و شیفت قابل تفکیک به سه واحد بوده و عملوندتها تحت نظر واحد کنترل به هر واحد رانده می‌شوند.

شکل ۹-۱ یک راه تفکیک واحد اجرایی به هشت واحد تابعی (عملیاتی) که بطور موازی با یکدیگر کار می‌کنند را نشان می‌دهد. عملوند در ثبات‌ها بسته به دستورالعمل به یکی از واحدها تحویل می‌گردد. عملی که در هر واحد انجام می‌شود در داخل چهارگوش‌های دیاگرام مشخص شده‌اند. جمع کننده و ضرب کننده عدد صحیح روی اعداد صحیح عمل می‌کنند. اعمال معیز شناور بین سه مدار که موازی با هم عمل می‌کنند تقسیم شده‌اند. اعمال منطقی، شیفت و افزایش در روی داده‌های مختلف بطور



شکل ۹-۱ 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 فعالیت ها 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. رووال معمول یک کامپیوتر برداشت<sup>۱</sup> دستورات از حافظه و اجرای آنها در پردازنده است. دستورات خوانده شده متوالی از حافظه رشته دستورات<sup>۲</sup> 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 میهم مطابق زیر تقسیم می کند:

رشته تک دستوری، رشته تک داده ای<sup>۳</sup> (SISD)

رشته تک دستوری، رشته چند داده ای<sup>۴</sup> (SIMD)

رشته چند دستوری، رشته تک داده ای<sup>۵</sup> (MISD)

رشته چند دستوری، رشته چند داده ای<sup>۶</sup> (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 چند تابعی 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 دستور العمل واحدی را توسط واحد کنترل دریافت می کنند ولی روی داده های مختلفی عمل می نمایند. واحد مشترک حافظه باید از چند مأذول<sup>۷</sup> (بخش) تشکیل شده باشد تا در یک زمان بتواند با تمام پردازنده ها ارتباط برقرار کند. MISD فقط از نظر شوری حائز اهمیت است چون عمل می هیج سیستمی با چنین سازمانی ساخته نشده است. سازمان MIMD به کامپیوترهای 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

کامپیوتروی در این دسته طبقه‌بندی می‌شوند.

دسته‌بندی Flynn بستگی به اختلافات بین عمل واحد کنترل و واحد پردازش داده دارد. این دسته‌بندی بیشتر به مشخصات رفتاری سیستم کامپیوترویی پردازد تا به اتصالات درونی آن. یک نوع پردازش موازی که منطبق بر تقسیم‌بندی Flynn نیست روش خط لوله<sup>۱</sup> است. دو نمونه موجود از این طبقه‌بندی یکی پردازنده‌های آرایه‌ای<sup>۲</sup> SIMD بحث شده در بخش ۹-۷ و دیگری چند پردازنده MIMD ارائه شده در فصل ۱۳ است.

در این فصل ما پردازش موازی را تحت عنوان‌های اصلی زیر بررسی خواهیم کرد:

- ۱- پردازش خط لوله
- ۲- پردازش برداری
- ۳- پردازنده‌های آرایه‌ای

پردازش خط لوله یک تکنیک پیاده‌سازی است که در آن جزء (زیر) اعمال<sup>۳</sup> محاسبه‌ها یا فازهای سیکل دستورالعمل بطور هم پوش<sup>۴</sup> (همزمان) اجرا می‌شوند. پردازش برداری درباره محاسبات بردارها و ماتریس‌های بزرگ است. پردازنده‌های آرایه‌ای هم بر روی آرایه‌های بزرگی از داده‌ها عمل می‌نمایند.

## ۹-۲ خط لوله

خط لوله تکنیکی است که یک پردازش سری را به عملیات جزیی تفکیک می‌نماید و هر عمل جزیی در مقطع خاصی همزمان با سایر مقاطع اجرا می‌گردد. خط لوله را می‌توان بعنوان مجموعه‌ای از قطعه پردازش کننده‌هایی تصور کرد که در آنها اطلاعات دودویی جریان دارد. هر قطعه بخشی از پردازش را که به آن دیکته شده انجام می‌دهد. نتیجه حاصل از محاسبات در هر قطعه به قطعه بعدی در خط لوله منتقل می‌گردد. کلمه "خط لوله" بر جریانی از اطلاعات در سیستم، مشابه با خط تولید در کارخانجات اطلاق می‌شود. از مشخصه‌های خط لوله این است که چندین محاسبه در یک زمان در قطعات مختلف در حال پیشروی است. هم‌پوشی یا باصطلاح همزمانی اجرا بدین ترتیب تحقق می‌یابد که به هر قطعه در خط لوله ثباتی تعلق می‌گیرد. این ثبات‌ها نوعی جداسازی را در هر قطعه ایجاد می‌نمایند بطوری که هر قطعه می‌تواند روی داده‌های جداگانه‌ای بصورت همزمان عمل نماید.

شاید ساده‌ترین راه در شناخت یک ساختار خط لوله این باشد که تصور کنیم هر قطعه از یک ثبات ورودی و بدنیال آن یک مدار ترکیبی ساخته شده است. ثبات داده را نگه‌میدارد و مدار ترکیبی جزء عملی را در ارتباط با قطعه انجام می‌دهد. خروجی مدار ترکیبی در یک قطعه مفروض به ثبات ورودی قطعه دیگر تحويل می‌گردد. به تمام ثبات‌ها پالس ساعتی اعمال می‌شود تا زمان کافی برای

1- Pipeline

2- Array Processors

3- Suboperation

4- overlap

فعالیت تمام قطعات تأمین گردد.

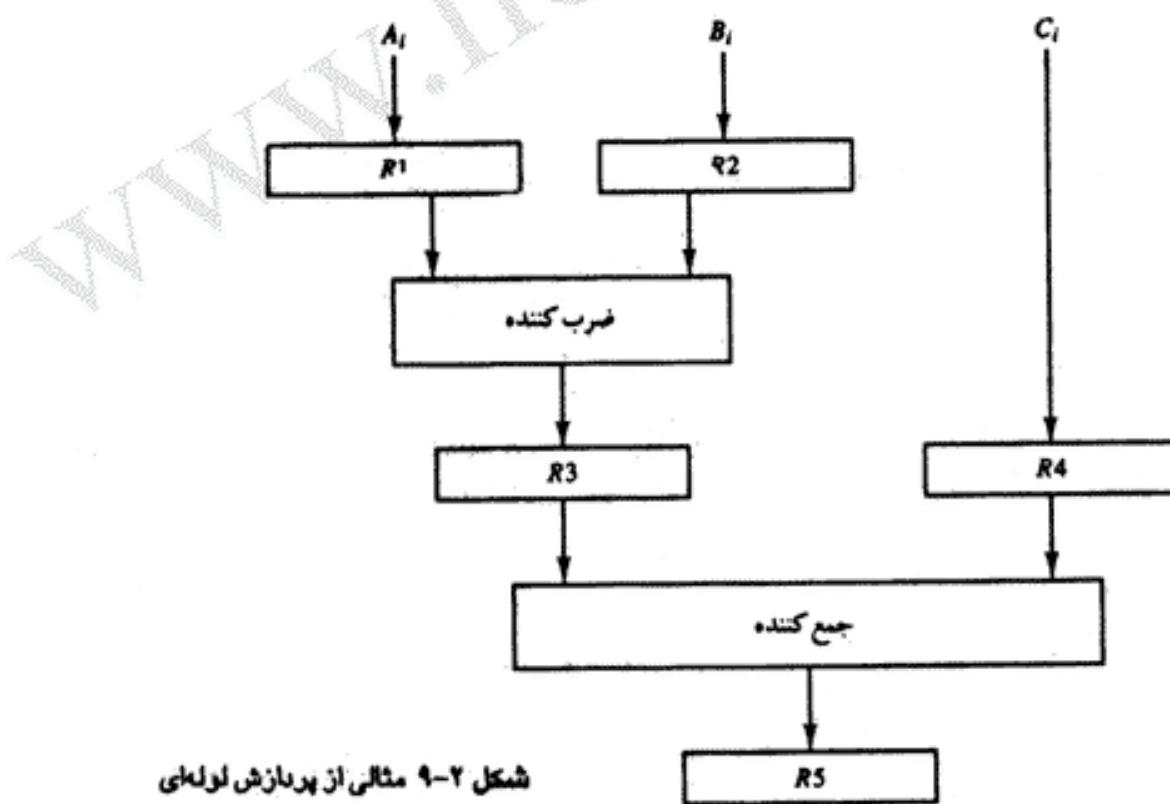
سازمان خط لوله توسط مثال ساده‌ای بیان می‌گردد. فرض کنید که می‌خواهیم یک عمل ترکیبی ضرب و جمع را با رشته‌ای از اعداد انجام دهیم.

$$A_i * B_i + C_i \quad \text{for } i = 1, 2, 3, \dots, 7$$

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.  $R_1$  تا  $R_5$  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:

$R1 \leftarrow A_i$	$R2 \leftarrow B_i$	دریافت $A_i$ و $B_i$
$R3 \leftarrow R1 * R2$	$R4 \leftarrow C_i$	ضرب و دریافت $C_i$
$R5 \leftarrow R3 + R4$		جمع $C_i$ با حاصل ضرب

پنج ثبات بازاء هر پالس با داده‌های جدیدی بار می‌شوند. اثر هر پالس در جدول ۹-۱ دیده می‌شود. اولین پالس ساعت  $A_1$  و  $B_1$  را به داخل  $R1$  و  $R2$  انتقال می‌دهند. پالس ساعت دوم حاصل ضرب  $R1$  در  $R2$  را به  $R3$  و  $C_i$  را به  $R4$  منتقل می‌کند. همان پالس ساعت  $A_2$  و  $B_2$  را به  $R1$  و  $R2$  انتقال می‌دهد. سومین



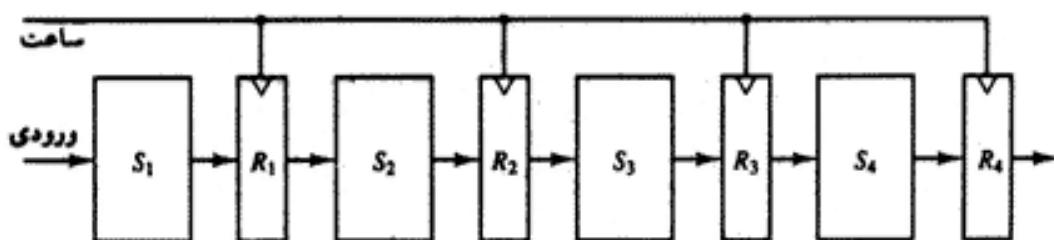
جدول ۹-۱ محتویات ثبات در مثال خط لوله

شاره پالس ساعت	قطعه ۱		قطعه ۲		قطعه ۳	
	R1	R2	R3	R4	R5	
1	$A_1$	$B_1$	—	—	—	
2	$A_2$	$B_2$	$A_1 \cdot B_1$	$C_1$	—	
3	$A_3$	$B_3$	$A_2 \cdot B_2$	$C_2$	$A_1 \cdot B_1 + C_1$	
4	$A_4$	$B_4$	$A_3 \cdot B_3$	$C_3$	$A_2 \cdot B_2 + C_2$	
5	$A_5$	$B_5$	$A_4 \cdot B_4$	$C_4$	$A_3 \cdot B_3 + C_3$	
6	$A_6$	$B_6$	$A_5 \cdot B_5$	$C_5$	$A_4 \cdot B_4 + C_4$	
7	$A_7$	$B_7$	$A_6 \cdot B_6$	$C_6$	$A_5 \cdot B_5 + C_5$	
8	—	—	$A_7 \cdot B_7$	$C_7$	$A_6 \cdot B_6 + C_6$	
9	—	—	—	—	$A_7 \cdot B_7 + C_7$	

پالس ساعت بطور همزمان بر روی هر سه قطعه عمل می‌کند. بدین ترتیب که A3 و B3 را در R1 و R2 می‌گذارد، حاصل ضرب R2 و R1 را به R3 منتقل و C2 را در R4 و بعلاوه حاصل جمع R3 و R4 را در R5 قرار می‌دهد. بنابراین برای پر کردن خط و دریافت اولین خروجی از R5، 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 مدامی است 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 جزء‌عمل‌ها 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 ۹-۳ نشان داده شده است. عملوندها از تمام قطعه‌ها با ترتیب ثابتی عبور می‌کنند. هر قطعه از یک مدار ترکیبی S تشکیل شده که جزء‌عمل خاصی را روی جریان داده درون لوله انجام می‌دهد.



شکل ۹-۳ 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 باشد. ما یک تکلیف<sup>۱</sup> 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 - زمان<sup>۲</sup> 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 ۹-۴ را دیده می شود. محور افقی نمودار زمان را بر حسب 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 ۱ انجام می شود. پس از پالس ساعت اول، قطعه ۲ مشغول اجرای  $T_1$  است، و در همان زمان قطعه ۱،  $T_2$  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 پرسود در هر پالس ساعت یک خروجی بدست خواهد آمد.

حال خط لوله ای مشکل 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.  $(n-1)t_p$  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،  $(n-1) + k$  پالس ساعت لازم است. بعنوان مثال دیاگرام ۹-۴ 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 غیرخط لوله ای با رابطه زیر تعریف می شود.

$$S = \frac{nt_n}{(K + n - 1) t_p}$$

قطعه	چرخه های ساعت								
	1	2	3	4	5	6	7	8	9
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 ۹-۴ دیاگرام فاصله - زمان برای خط لوله

1- Task

2- Space - Time diagram

هرچه تعداد تکلیف ها اضافه شوند،  $n$  از  $1 - k$  بیشتر می شود و  $1 - n + k$  بسته  $n$  میل می کند.  
تحت این شرط، رابطه فوق برابر است با

$$S = \frac{t_n}{t_p}$$

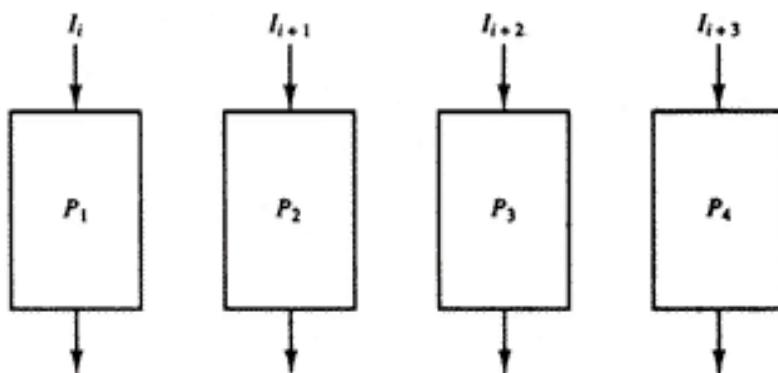
اگر فرض کنیم که زمان لازم برای غیرخط لوله همان زمان در خط لوله باشد داریم  $k t_p = t_n$ . لذا رابطه نسبت سرعت برابر خواهد بود با:

$$S = \frac{k t_p}{t_p} = k$$

رابطه فوق نشان می دهد که حداقل نسبت سرعت تثویری 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  $t_n = (k + n - 1) t_p = (4 + 99) \times 20 = 2060\text{ns}$  برای تکمیل پردازش نیاز دارد. با این فرض که  $n k t_p = 100 \times 80 = 8000\text{ns}$  برای تکمیل 100 تکلیف نیاز خواهد داشت. نسبت سرعت برابرست با  $\frac{8000}{2060} = 3.88$ . با افزایش تعداد تکلیف ها، افزایش سرعت بسته  $4$  میل می کند که با تعداد قطعات در خط لوله برابرست. اگر  $t_p = 60\text{ns}$  در نظر گرفته شود، نسبت افزایش سرعت  $\frac{3}{20} = 0.15$  خواهد شد.

برای تأکید بر مزیت سرعت تثویری 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 غیرخط 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 متواالی، 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 ۹-۵ 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 حداقل سرعت تشوری اش وجود دارد. قطعات مختلف ممکن است دارای زمان‌های اجرای مختلفی باشند. سیکل ساعت باید با تأخیر زمانی قطعه‌ای که حداقل 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 تکیک خط لوله امکان پردازش سریعتری را در مقایسه با یک پردازش سری فراهم می‌آورد، حتی اگر سرعت ماکریم تشوری هرگز بدست نیاید.

در طراحی کامپیوتر دو زمینه برای کاربرد خط لوله وجود دارد. خط لوله حسابی اعمال حسابی را بصورت جزء عمل‌ها<sup>۱</sup> برای اجرا در قطعات خط لوله تقسیم می‌کند. خط لوله دستورالعمل روی رشته‌ای از دستورات عمل می‌کند و ضمن آن فازهای بروزدشت، دیکد و اجرای سیکل دستورالعمل را توأم‌آجا می‌نمایند. دو نوع خط لوله فوق در زیر توضیح داده شده‌اند:

### ۹-۳ خط لوله حسابی

خط لوله حسابی معمولاً در کامپیوتراهای 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 ۱۰-۵ آمد تفکیک می‌گردد. در اینجا مثالی از یک خط لوله برای جمع و تفریق شناور را شاند می‌دهیم.

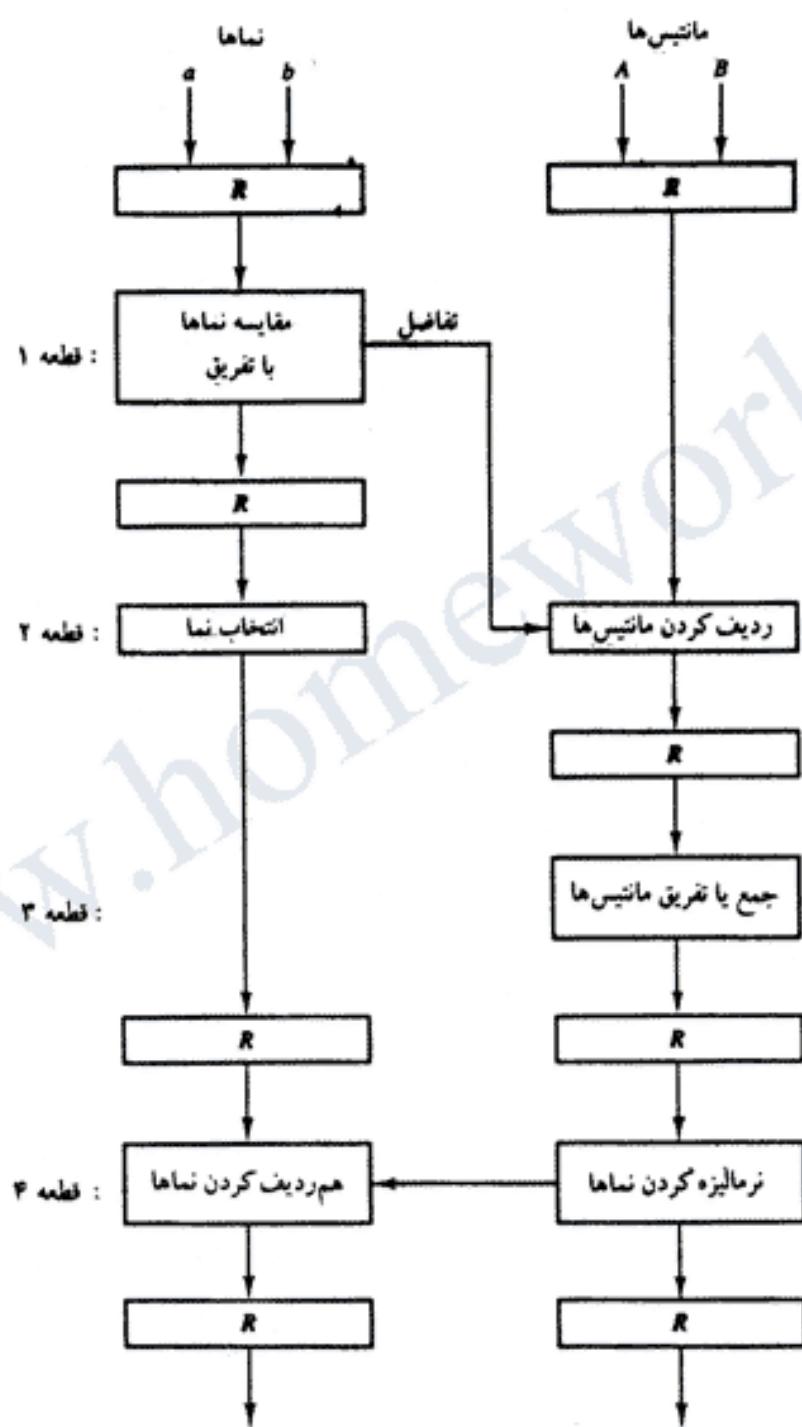
ورودی‌های خط لوله جمع کننده ممیز شناور دو عدد دو دو بی ممیز شناور نرم‌الیزه هستند

$$X = A \times 2^a$$

$$Y = B \times 2^b$$

A و B دو عدد کوچکتر از یک هستند که ماتیس‌ها را تشکیل می‌دهند و a و b نمایه‌ای آنها می‌باشند. جمع و تفریق ممیز شناور را می‌توان در چهار قطعه انجام داد، شکل ۹-۶. ثبات‌هایی که با R مشخص شده‌اند بین قطعات قرار گرفته‌اند تا نتایج میانی را ذخیره کنند. جزء عمل‌های اجرا شده در

1- Suboperation



شکل ۹-۶ خط لوله برای جمع و تغذیق ممیز شناور

قطعات عبارتند از:

- ۱- مقایسه نماها
- ۲- هم ردهی کردن<sup>۱</sup> مانتیس ها<sup>۲</sup>
- ۳- جمع یا تفریق مانتیس ها
- ۴- نرمالیزه کردن نتیجه

این اعمال در شکل ۱۰-۱۵ با کمی تغییر به منظور کاهش زمان اجرای زیر عمل ها 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<sup>۳</sup> رخ دهد مانتیس مجموع یا تفاضل یکبار به راست شیفت یافته و نما یک واحد افزایش می یابد. اگر یک فروریز<sup>۴</sup> 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 عمل های انجام شده در هر قطعه را مشخص نماید. به منظور سادگی، هرچند که در شکل ۶-۹ اعداد دودویی بکار رفته اند، از اعداد دهدی 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  $1 - 2 = 3$  بددست آید. نمای بزرگتر یعنی ۳، بعنوان نمای نتیجه انتخاب شده است. قطعه بعدی مانتیس Y را به راست شیفت می دهد تا روابط زیر بددست آیند:

$$X = 0.9504 \times 10^3$$

$$Y = 0.0820 \times 10^3$$

با این ترتیب دو مانتیس هم ردیف و دارای نمای یکسان می شوند. جمع دو مانتیس در قطعه ۳ 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 بددست می آید

$$Z = 0.10324 \times 10^4$$

مقایسه گر، شیفت دهنده، جمع گننده، تفریق گر، افزایش دهنده، و کاهش دهنده در خط لوله ممیز شناور

1- Align

2- Mantissas

3- Overflow

4- underflow

بروسیله مدارات ترکیبی ساخته می شوند. فرض کنید که تأخیر در چهار قطعه  $t_1 = 60\text{ns}$ ,  $t_2 = 70\text{ns}$ ,  $t_3 = 100\text{ns}$  و  $t_4 = 80\text{ns}$  باشد. پالس ساعت  $t_p = t_3 + t_4 = 110\text{ns}$  انتخاب می شود. یک مدار غیرخط لوله ممیز شناور جمع و تفریق معادل، زمان تأخیری برابر  $t_n = t_1 + t_2 + t_3 + t_4 + t_p = 320$  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 برروی رشته‌ای از داده‌ها بلکه برروی رشته‌ای از دستورات تیز می‌تواند اجرا شود. یک خط لوله دستورالعمل دستورات متوالی را از حافظه می‌خواند و در همان زمان دستور قبلی را در قطعه دیگری اجرا می‌نماید. این سبب می‌شود 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 برهمن 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 شده‌اند باید نادیده گرفته شود.

کامپیوتروی را با یک واحد برداشت و یک واحد اجرای دستور در نظر بگیرید که از خط لوله دو قطعه‌ای طراحی شده باشد. قطعه برداشت دستور می‌تواند به روشن بافر FIFO<sup>۱</sup> پیاده سازی شود. این واحد بیشتر تشکیل یک صف را می‌دهد 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 می‌برد. دستورات وارد بافر FIFO شده و بعداً بر اساس ترتیب اجرا می‌گردند. بنابراین رشته دستورات می‌توانند بصورت صف درآمده و منتظر دیگر شدن و سپس توسط واحد مربوطه اجرا گردند. مکانیزم صف‌بندی رشته دستورات روش بهینه‌ای را برای کاهش زمان متوسط دستیابی به حافظه فراهم می‌آورد. هر وقت که در بافر FIFO فضای وجود داشته باشد، واحد کنترل فاز برداشت دستور بعدی را آغاز می‌کند.

کامپیوتروهایی که دارای دستورات پیچیده‌ای هستند (پر دستور)<sup>۲</sup> علاوه بر فازهای برداشت و اجراء نیاز به فازهای دیگری برای تکمیل پردازش دستورات دارند. در حالات کلی‌تر، کامپیووتر هر دستور را با توجه به مراحل زیر پردازش می‌کند.

- ۱- برداشت دستور از حافظه
- ۲- دیگر دستور
- ۳- محاسبه آدرس مؤثر
- ۴- برداشت عملوند از حافظه
- ۵- اجرای دستور

## 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 وارد 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 پردازنه 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.

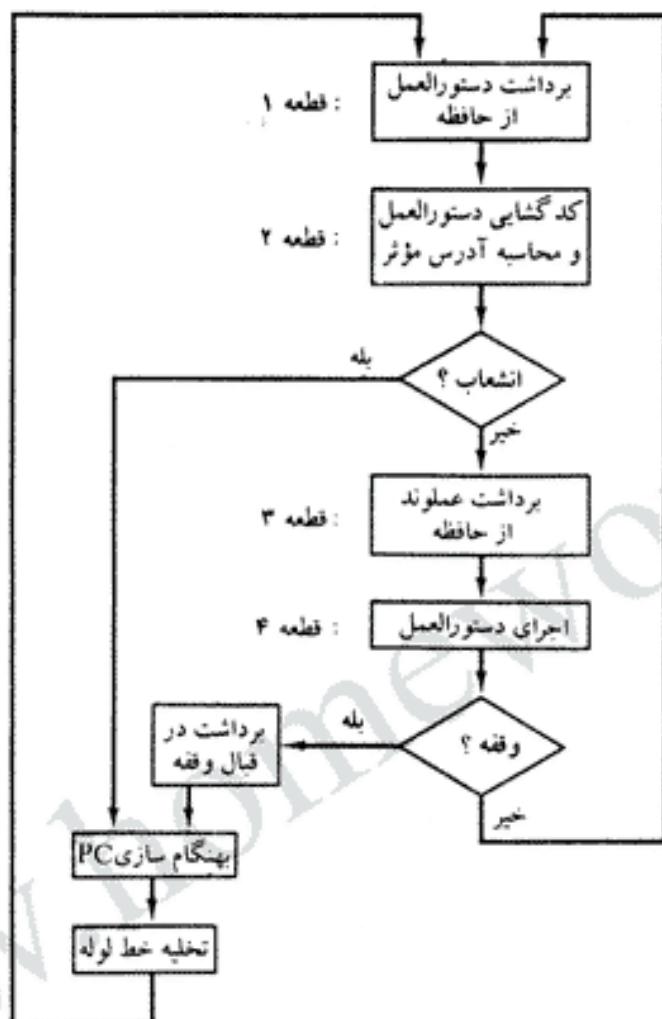
شکل ۹-۷ چگونگی پردازش دستور در CPU را با یک خط لوله چهار قطعه‌ای نشان می‌دهد. ضمن اینکه یک دستور در حال اجرا در قطعه ۴ است، دستور بعدی مشغول دریافت عملوند از حافظه در قطعه ۳ می‌باشد. آدرس موثر نیز ممکن است در یک مدار محاسباتی جداگانه برای سومین دستور محاسبه گردد، و هر وقت حافظه در دسترس باشد دستورات چهارم و بعد از آن قابل برداشت و جایگزینی در FIFO خواهند بود. بنابراین بطور همزمان تا چهار زیر عمل 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<sup>۱</sup> 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<sup>۲</sup> (تأیید) شد، موجب تخلیه خط لوله و آغاز از آدرس جدید می‌گردد.

شکل ۹-۸ 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.

F1 - ۱ قطعه‌ای است که دستور را برداشت می‌کند.

DA - ۲ قطعه‌ای است که دستور را دیکد و آدرس موثر را محاسبه می‌نماید.



شکل ۹-۷ خط لوله چهار قطعه‌ای

مرحله:	1	2	3	4	5	6	7	8	9	10	11	12	13
دستورالعمل:	FI	DA	FO	EX									
		FI	DA	FO	EX								
(انشاءابها)			FI	DA	FO	EX							
				FI	-	-	FI	DA	FO	EX			
					-	-	-	FI	DA	FO	EX		
									FI	DA	FO	EX	
										FI	DA	FO	EX
											FI	DA	FO

شکل ۹-۸ 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 انشعباب 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 دیگد می‌شود و دستور 4 از حافظه هم در قطعه FI برداشت می‌گردد. حال فرض کنید که دستور 3 یک دستور انشعباب باشد، به محض دیگد شدن این دستور در مرحله 4 و در قطعه DA، انتقال دستورات دیگر از FI به DA متوقف می‌شود تا دستور العمل انشعباب در مرحله 6 اجرا شود. اگر انشعباب صورت گیرد، دستور جدیدی در مرحله 7 برداشت می‌گردد. اگر انشعبابی صورت نگیرد، دستور قبلی برداشته شده در مرحله 4 می‌تواند بکار رود. سپس خط لوله تا رسیدن دستور انشعباب جدید بکار خود ادامه می‌دهد.

تا خبر 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 می‌گردد:

۱- 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.

۲- مشکل ناشی از واستگی داده‌ها<sup>۱</sup> نیز هنگامی رخ می‌دهد که دستوری به نتیجه دستور قبلی دیگری واستگی داشته باشد و آن نتیجه هنوز بدست نیامده باشد.

۳- مشکلات انشعباب از دستور انشعباب 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 پیش روی کند چراکه دستور قبلی کار خود را تمام نکرده است. واستگی 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 واستگی آدرس هنگامی رخ می‌دهد که آدرس عملوند، بعلت موجود نبودن اطلاعات لازم در روش آدرس دهی، قابل محاسبه نیست. مثلاً دستوری با روش

1- Data Dependency

آدرس دهن غیرمستقیم، اگر که دستور قبلی در حال بارگردان 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. کامپیوتروهای خط لوله با چنین مشکلات در وابستگی داده ها به روش های مختلفی برخورد می کنند. مستقیم ترین روش ایجاد همبندی های سخت افزاری<sup>۱</sup> می باشد. همبند مداری است که وظیفه اش شناسایی دستوراتی است که عملوند های مبدأ<sup>۲</sup> آنها مقصد عملوند های 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 بموضع 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 عملوند<sup>۳</sup> نام دارد سخت افزار خاص را برای شناسایی مشکل بکار برد و سپس با جهت دهن 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 قابل ثبات ها 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 کامپیوتروها بکار برد می شود سپردن مسئولیت رفع مشکل به کامپایلر می باشد که وظیفه تبدیل زبانهای سطح بالا را به زبان ماشین بعده دارد. کامپایلر برای چنین کامپیوتروهایی چنان طراحی شده تا وجود یک مشکل در داده را شناسایی کرده و با اجرای دستورات "هیچ کار"<sup>۴</sup> (NO-OP) و تغییر ترتیب اجرای دستور العمل ها، تأخیر لازم را در بارگردان داده ها بوجود آورد. این روش بنام بارگردان با تأخیر<sup>۵</sup> خوانده می شود. مثالی از بارگردان با تأخیر در بخش بعدی آورده شده است.

### دستکاری دستور العمل های انشعب

یکی از مسائل مهم در عملکرد خط لوله دستور العمل، وقوع یک دستور انشعب است. دستور انشعب می تواند شرطی و یا غیرشرطی باشد. یک انشعب بدون شرط همواره ترتیب اجرای برنامه را با بارگردان آدرس هدف در شمارنده برنامه عوض می کند. در یک انشعب شرطی، کنترل دستور مقصد را اگر شرط برقرار شده باشد انتخاب می نمایند در غیراینصورت دستور بعد از انشعب اجرا خواهد شد. همانطور که قبلاً اشاره شد، دستورات انشعب ترتیب اجرای رشته دستورات را شکسته و مشکلاتی را در کار خط لوله دستور ایجاد می نماید. کامپیوتروهای خط لوله ای تکنیک های سخت افزاری مختلفی را برای حداقل کردن افت سرعت عملکرد حاصل از دستورات انشعب بکار می بندند.

یکی از روش های دستکاری دستورات انشعب 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 انشعباب است. هر دو دستور تا ختم اجرای دستور انشعباب ذخیره می‌شوند. اگر شرط انشعباب برقرار باشد خط لوله از دستور مقصد انشعباب ادامه کار می‌دهد. روش کامل تراین است که از هر دو جهت عمل برداشت دستورات ادامه یابد تا تصمیم انشعباب اتخاذ گردد. در آن زمان کنترل درباره رشته صحیح دستورالعمل‌ها برای ادامه برنامه تصمیم می‌گیرد.

امکان دیگر بکارگیری بافر مقصد انشعباب<sup>۱</sup> یا BTB می‌باشد. BTB یک حافظه اشتراکی<sup>۲</sup> است (بخش ۴-۱۲ ملاحظه شود) که در قطعه برداشت خط لوله کار گذاشته می‌شود. هر ورودی به متشكل از آدرس انشعباب اجرا شده قبلی و دستور مقصد برای آن انشعباب است. این بافر چند دستور بعد از انشعباب را نیز ذخیره می‌نماید. وقتی که خط لوله دستور انشعباب را دیگد می‌کند BTB را برای آدرس دستورالعمل 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 انشعبابی که قبلاً انفاق افتاده‌اند بدون توقف در خط لوله موجودند.

روش دیگر با کمی تغییر با BTB، روش بافر حلقه<sup>۳</sup> 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 انشعبابات ذخیره می‌شود. بنابراین حلقه برنامه می‌تواند مستقیماً و بدون دستیابی به حافظه اجرا شود تا اینکه آخرین انشعباب موجب خروج از حلقه شود.

روش بعدی بکاررفته توسط برخی کامپیوترها، پیش‌بینی انشعباب<sup>۴</sup> است. یک خط لوله با پیش‌بینی انشعباب نتیجه حاصل از انشعباب را قبل از اجرای آن بكمک 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 انشعباب را از بین می‌برد.

روشی که توسط اغلب پردازنده‌های RISC مورد استفاده است انشعباب تأخیر داده شده<sup>۵</sup> می‌باشد. در این روش، کامپایلر دستورات انشعباب را شناسایی و کدهای زبان ماشین را جایجا 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 انشعباب است. این عمل سبب می‌شود تا کامپیوتر در طول اجرای دستورالعمل فوق دستور مقصد را برداشت کند و در نتیجه خط لوله پیوسته در حال کار نگهداشته شود. مثالی از این انشعباب 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
- 4- Branch Prediction

- 2- Associative memory
- 5- Delayed Branch

توانایی آن در بکارگیری یک خط لوله دستورالعمل کارآمد است. سادگی مجموعه دستورات می‌تواند با بکارگیری تعداد محدودی زیرعمل که هر کدام در یک سیکل ساعت اجرا می‌شود در پیاده سازی خط لوله دستورالعمل مورد استفاده قرار گیرد. به دلیل ثابت بودن طول قالب دستورالعمل‌ها عمل دیکد، همزمان با انتخاب ثبات انجام می‌شود. تمام دستورات دستکاری داده از نوع ثبات به ثبات هستند. چون تمام داده‌ها در ثبات‌ها قرار دارند نیازی به محاسبه آدرئن موثر یا برداشت عملوند از حافظه نیست. بنابراین خط لوله دستورالعمل با دو یا سه قطعه قابل پیاده‌شدن است. یک قطعه دستور را از حافظه برنامه برداشت می‌کند، و دیگری دستور را در ALU اجرا می‌کند. می‌توان قطعه سومی را هم برای ذخیره کردن نتیجه عمل ALU در یک ثبات مقصود بکار برد.

دستورات انتقال داده در RISC به دستورات بارکردن<sup>۱</sup> و ذخیره کردن<sup>۲</sup> محدودند. این دستورات از آدرس دهنی غیرمستقیم ثباتی استفاده می‌کنند 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 می‌برند که به دو حافظه جدا مرتبتند: یکی برای ذخیره کردن نتایج و دیگر برای ذخیره داده‌ها. بعضی اوقات دو حافظه قادرند با سرعت یکسان و برابر با سرعت پالس CPU عمل کنند و در این موقعیت به نام حافظه‌های نهان<sup>۳</sup> خوانده می‌شوند (بخش ۱۲-۶).

همانطور که در بخش ۸-۸ گفته شد یکی از مزایای مهم RISC توanایی 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 می‌افتد این است که آغاز هر دستور با هر پالس بوده و استفاده از پردازنده خط لوله در رسیدن به هدف، معنی نیل به اجرای دستورات در یک پالس است. مزیت RISC در مقابل CISC این است که RISC می‌تواند با استفاده از قطعات خط لوله که فقط یک سیکل ساعت نیاز دارد به هدف برسد، در حالیکه CISC چندین قطعه را در خط لوله بکار می‌برد و طولانی‌ترین 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 می‌کنند. پردازنده‌های RISC بعضی انکاء بر طراحی‌های سخت‌افزاری که با مشکلات ناشی از داده و تأخیرهای انشعاب مقابله می‌کنند، بر توanایی کامپایلر در شناخت و حداقل نمودن تأخیرات این برنامه‌ها متکی هستند. مثال‌های زیر نشان می‌دهد که چگونه یک کامپایلر می‌تواند برنامه‌های زبان ماشین را بهینه نماید تا مشکلات خط لوله جبران شود.

#### مثال: خط لوله دستورالعمل سه قطعه‌ای

نمونه‌ای از دستورات پردازنده RISC در جدول ۸-۱۲ لیست شدند. سه نوع دستورالعمل در این جدول دیده می‌شود. دستورات دستکاری داده روی داده‌های درون ثبات‌های پردازنده عمل می‌کنند.

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 بردند.

اکنون عملیات سخت افزاری را برای چنین کامپیوتري 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 مشکل 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 جزیی که در سه قطعه پیاده شده است تقسیم شود

I - برداشت دستور

ALU - عمل A

E - اجرای دستور

قطعه I دستور را از حافظه برنامه بردشت 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 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:  $R1 \leftarrow M[address1]$
- 2- LOAD:  $R2 \leftarrow M[address2]$
- 3- ADD:  $R3 \leftarrow R1 + R2$
- 4- STORE:  $M[address3] \leftarrow R3$

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 غایب 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 بارگردان 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 بارگردان قرار دهد یک دستور "هیچ کار" (no-op)<sup>۱</sup> در برنامه می‌گنجاند. این دستور العمل از حافظه برداشت می‌شود ولی هیچ کاری را انجام نداده و تنها یک سیکل ساعت وقت را تلف می‌کند. ایجاد تأخیر در استفاده از داده حافظه، بارگردان با تأخیر نام دارد.

شکل ۹-۹ (ب) برنامه قبل را همراه با یک دستور هیچ کار که پس از بارگردان 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 هیچ کار را به اندازه یک سیکل ساعت بتأخیر می‌اندازد تا جبران زمانی برای داده در خط لوله انجام شود. (توجه کنید که عمل "هیچ کار" هین سیکل 4 در قطعه A و هین سیکل 5 در قطعه E انجام می‌شود). مزیت بارگردان با تأخیر این است که موضوع واستگی داده توسط کامپایلر مورد توجه قرار می‌گیرد و نه توسط سخت افزار. این روش منجر به ساخت مدارات سخت افزاری 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 ۹-۸ دیدیم که یک دستور انشعاب عملکرد خط لوله را تا برداشت آدرس انشعاب با تأخیر مواجه می سازد. در مورد کاهش زمان تلف شده ناشی از انشعاب در بخش قبل بحث کردیم. روشنی که در اغلب پردازنده های RISC معمول است تکیه بر کامپایلر می باشد که توسط آن انشعاب ها در زمان مناسب در خط لوله قرار می گیرند. این روش انشعاب با تأخیر نامیده شده است.

کامپایلر مربوط به پردازنده ای که انشعاب با تأخیر را استفاده می کند طوری طراحی شده است که دستور العمل های 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 وایستگی های 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 می گردد. اثر کلی بمانند اجرای دستورات در فرم طبیعی آنهاست با این تفاوت که تأخیر انشعاب حذف گردیده است. این دیگر بعده کامپایلر است تا دستور العمل های مفیدی را یافته و پس از دستور العمل انشعاب قرار دهد. در صورت ناموفق بودن، کامپایلر دستورات هیچ کار را وارد خواهد کرد.

مثالی از انشعاب با تأخیر در شکل ۹-۱۰ دیده می شود. برنامه این مثال دارای پنج دستور العمل است.

- بار کردن حافظه در R1

- افزایش R2

- جمع R4 با R3

- تفرقی R6 از R5

- انشعاب به آدرس X

در شکل ۹-۱۰ (الف) کامپایلر دو دستور هیچ کار را بعد از انشعاب قرار می دهد. آدرس انشعاب، X، در سیکل ساعت 7 به PC منتقل می شود. برداشت دستور در X با تأخیری برابر با دو سیکل ساعت توسط دستور هیچ کار (no-op) صورت می گیرد. دستور واقع در آدرس X، فاز برداشت خود را در سیکل 8، یعنی پس از بهنگام شدن شمارنده PC، انجام می دهد.

آرایش برنامه شکل ۹-۱۰ (ب) با قرار گرفتن دستور العمل های جمع و تفرقی در بعد از انشعاب، بجای قبل از آن در برنامه اصلی، تغییر یافته است. بررسی زمانبندی خط لوله نشان می دهد که PC در سیکل ساعت 5 با مقدار X بهنگام می شود، ولی دستورات جمع و تفرقی به ترتیب صحیح از حافظه برداشت و اجرا می شوند. به بیان دیگر، اگر دستور بارکردن در آدرس 101 و نیز X برابر 350 باشد، دستور العمل انشعاب از مکان 103 برداشت می شود. دستور العمل جمع از مکان 104 برداشت و در سیکل 6 اجرا می شود. دستور العمل تفرقی از آدرس 105 برداشت و در سیکل ساعت 7 اجرا می گردد. چون مقدار X در سیکل ساعت 5 در قطعه PC منتقل می شود، دستور العمل 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) تغیر آرایش دستورالعمل ها

شکل ۹-۱۰ مثال انشعاب با تأخیر

## ۹-۶ پردازش برداری

رده‌های از مسائل محاسباتی وجود دارد که حل آن ورای توانایی‌های کامپیوترهای معمولی است. از مشخصه‌های این مسائل نیاز آنها به تعداد بسیار زیادی از محاسبات است که کامپیوترهای معمولی روزها حتی هفته‌ها وقت نیاز دارند. در بسیاری از کاربردهای علمی و مهندسی این مسائل قابل فرموله شدن بر حسب 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 نمونه‌هایی از زمینه‌های کاربردی با اهمیت در پردازش موازی می‌باشند.

- پیش‌بینی درازمدت وضع هوا

- اکتشافات نفت

- تحلیل داده‌های زلزله‌نگاری

- تشخیص پزشکی

- آنرودینامیک و شبیه سازی پروازهای فضایی
- هوش مصنوعی و سیستم های خبره
- نقشه برداری از زمین 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 رویدهای 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 سط्रی  $[V_1 \ V_2 \ V_3 \ ... \ V_n]$  نشان داده می شود. با عمودی چیدن اقلام داده ها می توان این بردار را بصورت عمودی هم نشان داد. یک کامپیوتر ترتیبی معمولی تنها قادر است پردازش عملوند ها را یک به یک پردازش نماید. در نتیجه، اعمال روی بردارها باید به محاسبات منفرد با متغیر های 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 فرترن را که در زیر آورده شده است در نظر بگیرید:

DO 20 I=1, 100

$$20 \quad C(I) = B(I) + A(I)$$

این برنامه مربوط به جمع دو بردار  $A$  و  $B$  بطول 100 برای تولید بردار  $C$  می باشد. برنامه فوق توسط رشته اعمال زیر بزیان 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<sup>۱</sup> شده و مراحل 100 بار تکرار می‌گرددند. کامپیوتروی که قادر به پردازش برداری است اثلاف وقت مربوط به برداشت و اجرای دستورالعمل‌ها را در حلقه برنامه حذف می‌کند. در این کامپیوتروها 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 جمع کننده ممیز شناور خط لوله‌ای مشابه با آنچه که در شکل ۹-۶ دیدیم انجام می‌شود. یک قالب ممکن برای دستورالعمل برداری در شکل ۱۱-۹ نشان داده شده است. این قالب أساساً یک دستور سه آدرسی با سه میدان مشخص کننده 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 می‌نماییم.

### ضرب ماتریسی

ضرب ماتریسی یکی از اعمال محاسباتی سنگین است که در کامپیوتروهای پردازش برداری انجام می‌شود. ضرب دو ماتریس  $n \times n$  از  $n^3$  عمل 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 سطروی یا مجموعه‌ای از  $m$  بردار ستونی تشکیل گردیده است. بعنوان مثال ضرب دو ماتریس A و B  $3 \times 3$  را در نظر بگیرید.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

کد عمل	آدرس شروع مبدأ ۱	آدرس شروع مقصود	آدرس شروع مقصد	طول بردار
--------	------------------	-----------------	----------------	-----------

شکل ۱۱-۹ قالب دستور برای پردازنده برداری

ماتریس حاصلضرب  $C$  یک ماتریس  $3 \times 3$  بوده و توسط ضرب داخلی زیر به عناصر ماتریس‌های  $A$  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  $i = 1$  و  $j = 1$  محاسبه می‌شود

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$$

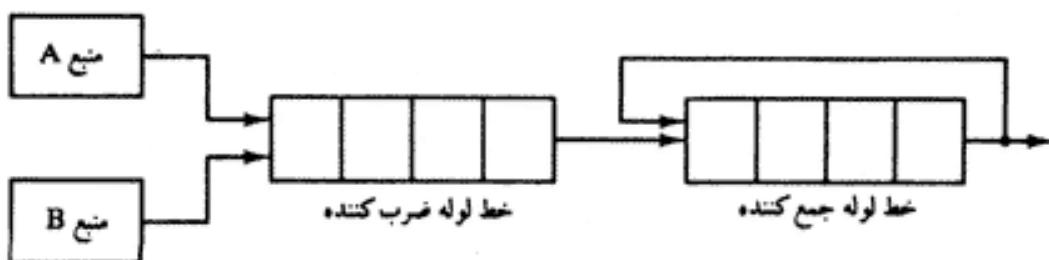
لازم به بودست آمدن این عنصر سه ضرب (پس از مقداردهی اولیه  $C_{11} = 0$ ) و سه جمع می‌باشد. تعداد کل ضرب‌ها و جمع‌های لازم برای محاسبه ماتریس حاصلضرب  $27 = 3 \times 9$  است.

اگر ما اعمال ضرب، جمع‌های مرتبط با هم  $ab + c$  را یک عمل تجمعی تصور کیم، 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

$$C = A_1B_1 + A_2B_2 + A_3B_3 + A_4B_4 + \dots + A_nB_n$$

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 ثبات‌های پردازنده تصور می‌شوند. همچنین فرض بر این است که خط لوله ضرب ممیز شناور و نیز جمع ممیز شناور هر یک دارای چهار قطعه باشند. تمام ثبات‌های قطعات در ضرب‌کننده و چمن‌کننده با 0 مقداردهی اولیه شده‌اند. بنابراین، خروجی جمع‌کننده در هشت سیکل اولیه 0 است تا اینکه هر دو لوله پر شوند. زوج‌های  $A_i$  و  $B_i$  با سرعت یک زوج در هر سیکل بداخل آورده شده و ضرب می‌شوند. پس از چهار سیکل اول، حاصلضرب شروع به جمع شدن با خروجی جمع‌کننده می‌نماید. در طول چهار



شکل ۹-۱۲ خط لوله برای محاسبه یک ضرب داخلی

سیکل بعدی به حاصلضربهایی که وارد خط لوله جمع‌کننده می‌شوند، ۰ اضافه می‌شود. در پایان سیکل هشتم، چهار حاصلضرب اول یعنی  $A_1B_1$  تا  $A_4B_4$  در چهار قطعه جمع‌کننده و چهار حاصلضرب بعدی یعنی  $A_5B_5$  تا  $A_8B_8$  در قطعه ضرب‌کننده می‌باشند. در آغاز سیکل نهم در خروجی جمع‌کننده  $A_1B_1$  و در خروجی ضرب‌کننده  $A_5B_5$  وجود دارد. به این ترتیب سیکل نهم جمع  $A_1B_1 + A_5B_5$  را در خط لوله جمع آغاز می‌نماید. سیکل دهم جمع  $A_2B_2 + A_6B_6$  را آغاز می‌کند و عمل بهمین ترتیب ادامه دارد. با این الگو جمع به چهار بخش زیر تقسیم می‌شود

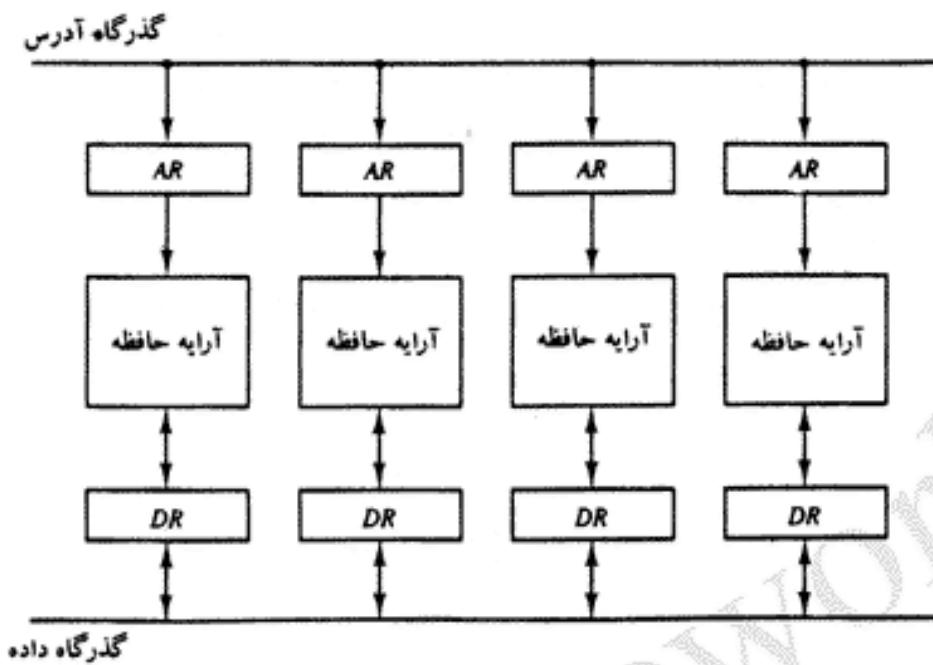
$$\begin{aligned} C = & A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} + \dots \\ & + A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} + \dots \\ & + A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} + \dots \\ & + A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} + \dots \end{aligned}$$

وقتی که دیگر جمله ضربی برای جمع شدن وجود نداشته باشد سیستم چهار صفر بداخل خط لوله ضرب ارسال می‌کند. سپس در هر یک از چهار قطعه تنها یک ضرب جزیی وجود خواهد داشت که مربوط به چهار جمع در چهار سطر معادله فوق است. چهار جمع جزیی در نهایت با هم جمع می‌شوند تا حاصل جمع نهایی حاصل گردد.

### برگی کردن حافظه<sup>۱</sup>

پردازنده‌های 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 می‌توان به تعدادی مازول<sup>۲</sup> که به گذرگاه‌های آدرس و داده مشترک وصلند تقسیم کرد. یک مازول حافظه را آرایه‌ای از حافظه به همراه ثبات‌های داده و آدرس مربوطه‌اش تشکیل می‌دهد. شکل ۹-۱۳ یک واحد حافظه چهار مازولی را نشان می‌دهد. همانطور که دیده می‌شود هر آرایه دارای ثبات آدرس AR و ثبات داده DR است. ثبات‌های آدرس، اطلاعات را از گذرگاه آدرس مشترک دریافت و ثبات‌های داده هم با گذرگاه داده دوطرفه تبادل اطلاعات می‌نمایند. دو بیت کم ارزش‌تر آدرس می‌تواند برای تفکیک چهار مازول بکار روند. سیستم مازولی اجازه می‌دهد تا یک مازول، دستیابی به حافظه را آغاز نماید در حالیکه سایر مازول‌ها در حال خواندن یا نوشتمن یک کلمه بوده و در واقع هر مازول مستقل از وضعیت مازول‌های دیگر به یک درخواست حافظه پاسخ می‌دهد.

مزیت حافظه مازولی این است که امکان استفاده از تکنیکی بنام برگی کردن حافظه را فراهم می‌آورد. در یک حافظه برگی شده به مازول‌های مختلف حافظه، آدرس‌های متفاوتی تخصیص یافته است. مثلاً،



شکل ۹-۱۳ سازمان حافظه چند مازولی

در یک سیستم با دو مازول برای حافظه، ممکن است آدرس‌های زوج در یک مازول و آدرس‌های فرد در دیگری باشد، وقتی تعداد مازول‌ها توانی از 2 است بیت‌های کم ارزش‌تر آدرس، مازول حافظه را انتخاب می‌نمایند و بقیه بیت‌ها مکان خاصی را برای دستیابی در مازول انتخاب شده مشخص می‌نمایند. حافظه مازولی در سیستم‌های خط لوله‌ای و پردازش برداری مفیدند. یک پردازنده برداری که از یک حافظه  $n$  برقی استفاده می‌کند می‌تواند  $n$  عملوند را از  $n$  مازول مختلف برداشت نماید. با توزیع دستیابی به حافظه، زمان موثر سیکل حافظه می‌تواند با فاکتوری در حدود تعداد مازول‌ها کاهش یابد. یک CPU با خط لوله دستورالعمل می‌تواند از تعدد مازول‌های حافظه بهره‌گیری نماید بطوری که هر قطعه در خط لوله می‌تواند مستقل از دستیابی قطعات دیگر، به حافظه دسترسی یابد.

### سوپر کامپیوتر

یک کامپیوتر تجاری با دستورالعمل‌های برداری به همراه عملیات حسابی معیز شناور خط لوله‌ای، سوپر کامپیوتر نامیده می‌شود. این کامپیوترها ماشین‌هایی قوی با قدرت اجرایی بالا بوده و اغلب برای محاسبات علمی بکار می‌روند. برای افزایش سرعت، قطعات بسیار نزدیک بهم نصب شده‌اند تا بدینوسیله فوایدی را که سیگنال‌ها طی می‌کنند حداقل نمایند. برای خارج ساختن گرما از مدارات و به منظور جلوگیری از سوختن آنها بعلت نزدیکی شان از تکنیک‌های خاصی استفاده می‌گردد. مجموعه دستورالعمل‌های سوپر کامپیوترها شامل دستورات استاندارد انتقال داده‌ها، دستکاری داده‌ها و دستورات کنترل برنامه در کامپیوترهای معمولی است. علاوه بر آنها، دستوراتی وجود دارند که

بردارها و ترکیبی از اسکالارها و بردارها را پردازش می نمایند. سوپر کامپیوتر، یک سیستم کامپیوتراست که بیشتر بلحاظ سرعت بالایش در محاسبات، سیستم های حافظه سریع و حجم و حجم بالای پردازش در آن شناخته شده است. این کامپیوتر با واحد های چند تابعی (چند کاره) تجهیز شده و هر واحد دارای پیکربندی لوله ای خاص خود می باشد. هر چند سوپر کامپیوتر قادر به اجرای برنامه های همه منظوره مربوط به کامپیوتراهای معمولی می باشد، ولی در اصل برای انواع محاسبات عددی مربوط به بردارها و ماتریس های اعداد ممیز شناور بهینه سازی شده است.

سوپر کامپیوتراها برای پردازش معمولی روزمره یک تشکیلات کامپیوترا مناسب نیستند. کاربرد این کامپیوتراها 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 کامپیوتراها، انجام تعداد اعمال روی اعداد ممیز شناور در هر ثانیه است که به آن فلاپس<sup>1</sup> می گویند. هر مگافلاپس برابر یک میلیون فلاپس و یک گیگافلاپس<sup>2</sup> نیز یک میلیارد فلاپس است. یک نمونه سوپر کامپیوترا از این رده دارای سیکل زمانی پایه ای برابر با ۴ تا ۲۰ نانو ثانیه می باشد. اگر پردازنده قادر به اجرای هر عمل ممیز شناور در یک سیکل باشد، بین ۵۰ تا ۲۵۰ 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 کامپیوترا بنام cray-1 در سال ۱۹۷۱ ساخته شد. این کامپیوترا از پردازش برداری همراه با ۱۲ 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 ۶۴ بیتی در طول یک سیکل ساعت ۱۲.۵ نانو ثانیه ای انجام شود. در نتیجه سرعت پردازش در خط لوله ۸۰ مگافلاپس در هین پردازش داده خواهد بود. ظرفیت حافظه این کامپیوترا ۴ میلیون کلمه ۶۴ بیتی است. این حافظه به ۱۶ بانک (بخش) تقسیم شده است و هر کدام دارای زمان دسترسی ۵۰ نانو ثانیه می باشدند. بنابراین هرگاه همه بانک ها با هم دستیابی شوند سرعت انتقال حافظه ۳۲۰ میلیون کلمه در ثانیه است. تحقیقات سازندگان cray موجب گشتش این کامپیوترا گردید و در نتیجه Cary X-MP و Cary Y-MP ساخته شدند. سوپر کامپیوترا جدید cary-2، دوازده برابر قوی تر از ۱ است.

یکی دیگر از سوپر کامپیوتراهای اولیه Fujitsu VP-200 است. این کامپیوترا دارای یک پردازنده اسکالر و یک پردازنده برداری است که می توانند بطور همزمان کار کنند. در این کامپیوترا هم، مانند سوپر کامپیوترا cary از ثبات ها و واحد های عملیاتی متعددی استفاده شده است تا عملیات ثبات - ثبات امکان پذیر باشد. در پردازنده برداری چهار خط لوله اجرای دستور العمل ها وجود دارند که هرگاه همزمان

کار کنند به سرعتی حدود 300 مگافلاپس دست خواهند یافت. حافظه اصلی 32 میلیون کلمه داشته که از طریق خط لوله‌های بارگذاری و ذخیره کردن به ثبات‌های برداری متصلند. VP200 دارای 83 دستورالعمل برداری و 195 دستور اسکالر است. مدل جدیدتر 2600 VP از سیکل ساعت 3.2 ns استفاده نموده و مدعی اوج سرعتی برابر با 5 گیگافلاپس است.

## ۹-۷ پردازنده آرایه

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<sup>۱</sup> نوعی پردازنده کمکی<sup>۲</sup> 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 پردازنده‌ی است که سازمان تک دستورالعمل - چندداده‌ای را دارد. این پردازنده دستورالعمل‌های برداری را با استفاده از واحدهای علمیاتی 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<sup>۳</sup> (عملیاتی) است. این واحدها عبارتند از واحد حساب با یک یا چند جمع و ضرب کننده ممیز شناور خط لوله‌ای. آرایه پرداز قادر است انواع مسائل ریاضی پیچیده را بکمک برنامه‌های کاربر حل نماید.

شکل ۹-۱۴ اتصالات درونی یک آرایه پرداز الحاقی را به یک کامپیوتر میزبان<sup>۵</sup> نشان می‌دهد. کامپیوتر میزبان یک کامپیوتر همه منظوره تجاری است و آرایه پرداز مانند یک ماشین در انتهای خط<sup>۶</sup> 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

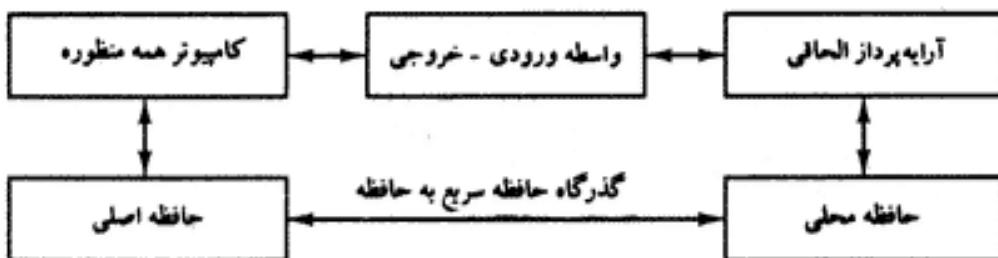
1- Array Processor

2- Attached Array Processor

3- Auxiliary Processor

4- Multiple Functional Unit    5- Host

6- Back-end machine



شکل ۹-۱۴ آرایه پرداز الحالی با کامپیوتر میزبان

از کامپیوترهای میزبان می‌باشد. مثلاً اگر FSP-164/MAX VAX11 را تا 100 مگافلاپس افزایش می‌دهد. هدف از آرایه پرداز الحالی ایجاد توانایی پردازش برداری کامپیوترهای معمولی با کسری از هزینه لازم برای سوپر کامپیوتر است.

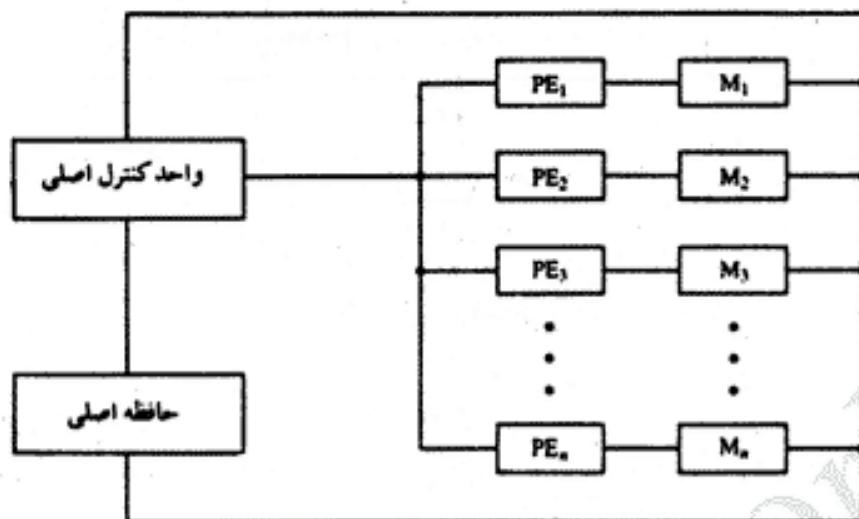
### آرایه پرداز SIMD

یک آرایه پرداز SIMD کامپیوتراست با واحدهای پردازشی متعدد که بطور موازی کار می‌کنند. واحدهای پردازش در این کامپیوترا برای انجام کارهای یکسان تحت واحد کنترل مشترک همگام شده‌اند، و به این ترتیب سازمانی با رشته دستورالعمل‌های واحد ولی داده‌های متعدد (SIMD) حاصل می‌گردد. بلاک دیاگرام شامل مجموعه‌ای از عناصر پردازشی یکسان (PE‌ها) است که هر کدام یک حافظه محلی M را دارا هستند (شکل ۹-۱۵). در هر عنصر پردازشی یک ALU، یک واحد حساب ممیز شناور و یک تعداد ثبات‌های عملیاتی وجود دارند و کنترل اصلی اعمال را در عناصر پردازشی کنترل می‌نماید. برنامه اصلی برای ذخیره برنامه بکار می‌رود. وظیفه کنترل اصلی دیکدکردن دستورالعمل‌ها و تعیین چگونگی اجرای دستورالعمل است. دستورات اسکالار و کنترل برنامه مستقیماً در داخل واحد کنترل اصلی اجرا می‌شوند. دستورالعمل‌های برداری بطور همزمان به تمام PE‌ها اعلام می‌شوند. هر PE از عملوندهای ذخیره شده در حافظه محلی اش استفاده می‌کند. عملوندهای برداری قبل از اجرای موازی دستورات در حافظه‌ها توزیع شده‌اند.

به عنوان مثال، جمع برداری  $C = A + B$  را در نظر بگیرید. کنترل اصلی ابتدا آمین مولفه‌های A و B یعنی  $a_i$  و  $b_i$  در حافظه محلی  $M_i$  در ازای  $i = 1, 2, 3, \dots, n$  را ذخیره می‌کند. سپس دستور جمع ممیز شناور  $c_i = a_i + b_i$  بین تمام PE‌ها منتشر می‌شود تا بدین ترتیب جمع بطور همزمان صورت گیرد. مولفه  $c_i$  در مکان‌های ثابتی در هر حافظه محلی ذخیره می‌گردد. در نتیجه مجموع برداری مورد نظر در یک سیکل جمع تولید می‌گردد.

برای کنترل وضعیت در هر PE در طول اجرای دستورالعمل‌های برداری از روش‌های ماسک کردن<sup>۱</sup>

1- Masking



شکل ۹-۱۵ سازمان آرایه‌پرداز SIMD

(پوشش دادن) استفاده می‌شود. هر PE دارای پرچمی است که هرگاه PE فعال باشد ۱ می‌شود و هر وقت PE غیرفعال باشد ۰ می‌گردد. بدین ترتیب تضمین می‌شود که فقط PE‌هایی که باید در حین اجرای دستور مشارکت داشته باشند فعال شوند. مثلاً فرض کنید که آرایه‌پردازی 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 قسمت‌های 64 کلمه‌ای تقسیم شوند.

معروف‌ترین پردازنده آرایه‌ای SIMD، کامپیوتر ILLIAC IV است که در دانشگاه ایلینویز<sup>۱</sup> طراحی و توسط شرکت باروز<sup>۲</sup> ساخته شد. از این کامپیوتر دیگر استفاده نمی‌شود. کامپیوترهای SIMD کامپیوترهای بسیار تخصصی هستند. این کامپیوترها عمدها برای مسائل عددی که قابل ارائه بشکل برداری یا ماتریسی باشند مناسبند. با این وجود در سایر انواع محاسبات یا اجرای برنامه‌های داده پردازی کارآیی چندانی ندارند.

### مسائل

۹-۱ در برخی از محاسبات علمی لازم است تا رابطه حسابی  $(C_i + D_i)(A_i + B_i)$  با رشتۀ ای از اعداد انجام شود. یک پیکربندی خط لوله را برای انجام آن مشخص کنید. محتوای تمام یثات‌های خط لوله را بازاء ۶ تا ۱ = ۱ لیست نمایید.

۹-۲ یک دیاگرام فاصله-زمان برای یک خط لوله شش قطعه‌ای رسم کنید که زمان پردازش هشت

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 ns 50 زمان نیاز دارد. همان تکلیف در یک خط لوله شش قطعه ای به یک سیکل ساعت 10 نانو ثانیه ای نیازمند است. نسبت افزایش سرعت خط لوله برای 100 تکلیف را معین کنید. حداکثر تسریع قابل دسترسی چقدر است؟
- ۹-۵ خط لوله شکل ۹-۲ دارای زمانهای انتشار زیر است: ns 40 برای خواندن عملوند از حافظه به ثبات  $R_1$  و  $R_2$ ، ns 5 برای انتشار سیگنال در ضرب کننده، ns 15 برای انتقال به  $R_3$  و ns 15 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 سایز جمع کننده لازم چیست؟
- (ب) چند قطعه در خط لوله است؟
- (ج) اگر تأخیر انتشار در هر قطعه ns 30 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:
- $$t_1 = 50 \text{ ns}, t_2 = 30 \text{ ns}, t_3 = 95 \text{ ns}, t_4 = 45 \text{ ns} \quad t_r = 5 \text{ ns}$$
- (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 برد؟
- 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 دستور انشعاب 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	$R1 \leftarrow M[312]$
ADD	$R2 \leftarrow R2 + M[313]$
INC	$R3 \leftarrow R3 + 1$
STORE	$M[314] \leftarrow R3$

- ۹-۱۲ برنامه‌ای را مثال بزنید که موجب اختشاش داده در خط لوله سه قطعه بخش ۹-۵ گردد.
- ۹-۱۳ مثالی ارائه نمایید که در آن در خط لوله سه قطعه‌ای بخش ۹-۵ با تأخیر عمل بارگردان را صورت دهد.
- ۹-۱۴ برنامه‌ای مثال بزنید که موجب اتلاف زمان ناشی از انشعاب در خط لوله سه قطعه‌ای بخش ۹-۵ گردد.
- ۹-۱۵ مثالی ارائه کنید که در آن از انشعاب با تأخیر در خط لوله سه قطعه‌ای بخش ۹-۵ استفاده شود.
- ۹-۱۶ ضرب دو ماتریس  $40 \times 40$  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  $60 \times 60$  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 معیز شناور نیاز دارد. مثلاً در یک سوپر کامپیوتر با سرعت اجرایی 100 مگافلاپس پردازش می‌شود. زمان اجرا چقدر است؟
- ۹-۲۰ کامپیوتر را با چهار پردازنده لوله‌ای معیز شناور در نظر بگیرید. فرض کنید که هر پردازنده از سیکل زمان ns 40 استفاده می‌کند. برای اجرای 400 عمل معیز شناور چقدر وقت لازم است؟ اگر همین عمل با استفاده از یک پردازنده لوله‌ای تکی با سیکل ساعت ns 10 انجام شود چه تفاوتی خواهد داشت؟

## ١٠

**معماری کامپیوتر**

۱۰-۵ عمل های حسابی ممیز شناور

۱۰-۶ واحد حساب دهدی

۱۰-۷ اعمال حسابی دهدی

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 عمدہ 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 روش های 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 دهدی 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 متفقی می توانند بصورت اندازه علامت دار یا متمم علامت دار نشان داده شوند. اگر فقط دستورالعمل جمع، برای عده های دودویی ممیز ثبات در نظر گرفته شود، پردازش حسابی بسیار ساده خواهد بود. اما اگر در پردازنده حسابی هر چهار عمل اصلی برای داده های دودویی و دهدی 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 مستله که بوسیله تعداد محدودی از رویه های متوالی بطور مناسب تعریف شده باشد **الگوریتم<sup>۱</sup>** خوانده می شود. برای جمع دو عدد با معیز ثابت و اعداد منفی در نمایش متمم ۲ علامت دار، **الگوریتمی** در بخش ۳-۳ 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 فزار دیجیتال نشان خواهیم داد. ما برای داده های زیر، جمع، تفریق، ضرب و تقسیم را بررسی خواهیم کرد:

- ۱- داده دودویی معیز ثابت با نمایش مقدار علامت دار.
- ۲- داده دودویی معیز ثابت با نمایش متمم ۲ علامت دار.
- ۳- داده معیز شناور دودویی
- ۴- داده دمدهای کد شده با دودویی

## ۱۰-۲ جمع و تفریق

همانطور که در بخش ۳-۳ گفته شد، سه راه برای نمایش اعداد دودویی معیز ثابت منفی وجود دارد: اندازه علامت دار، متمم ۱ علامت دار، یا متمم ۲ علامت دار. بسیاری از کامپیوترها وقتی اعمال حسابی با اعداد صحیح را انجام می دهند از نمایش متمم ۲ استفاده می کنند. برای اعمال معیز شناور، اکثر کامپیوترها نمایش اندازه علامت دار را برای ماتیس بکار می بروند. در این بخش ما **الگوریتم** های جمع و تفریق را برای داده های نمایش داده شده با مقدار علامت دار و مجددآ برای نمایش متمم ۲ علامت دار بدست می آوریم. توجه به این نکته مهم است که بدانیم نمایش انتخابی برای اعداد منفی به نمایش عدهها در ثبات ها، قبل و بعداز 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 اعداد های اندازه علامت دار

نمایش اعداد بصورت اندازه علامت دار آشنایت زیرا در محاسبات ریاضی روزمره بکار می رود. رویه جمع یا تفریق دو عدد دودویی علامت دار با قلم و کاغذ ساده و سرراست است. مروری براین رویه در بدست آوردن الگوریتم سخت افزاری می تواند مفید باشد.

مالندازه دو عدد را با  $A$  و  $B$  نشان می دهیم. 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 شده اند. سایر ستون ها در جدول، اعمالی را که باید برروی اندازه اعداد انجام شود نشان می دهد. 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 می آیند و می توان 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، اندازه ها را باهم مقایسه و عدد کوچکتر را از بزرگتر تفریق کنید. اگر  $A > B$  است، علامت  $A$  را برای جواب برگزینید ولی اگر  $B < A$  است علامت  $A$  را مکمل نمائید. اگر دو اندازه باهم مساوی نباشند،  $B$  را از  $A$  تفریق کرده و علامت حاصل را مثبت بگذارید.

این دو الگوریتم جز در مقایسه علامت ها مشابهند. رویه ای که باید برای علامت های یکسان در رویه جمع دنبال شوند مشابه رویه برای علامت های 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 ثبات ها ذخیره شوند.

جدول ۱۰-۱ جمع و تفریق اعداد اندازه علامت دار

عمل	جمع اندازه ها	تفریق اندازه ها		
		$A > B$	$A < B$	$A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

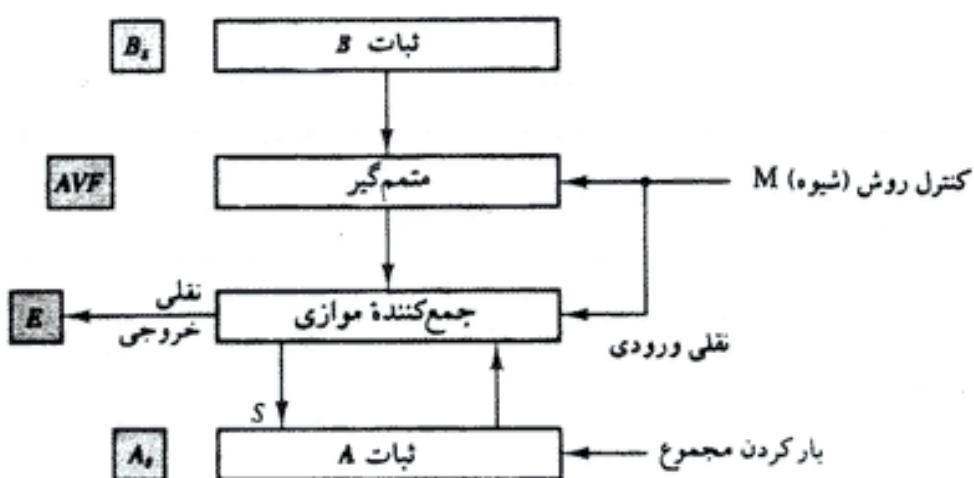
اجازه بدھید تا ثبات های A و B مقدار اعداد را ذخیره کنند و  $A_i$  و  $B_i$  فلیپ فلاب های نگهدارنده علامت آنها باشند. نتیجه عمل ممکن است به ثبات سومی انتقال یابد: با این وجود، اگر نتیجه به A و  $A_i$  منتقل شود صرفه جویی انجام شده است. بنابراین A و  $A_i$  بالاتفاق اثبات را می سازند.

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  $A=B$  یا  $A < B$  یا  $B-A$  یا  $A-B$  لازم است. رابطه علامت ها را مشخص نماید. سوم، دو مدار تفربیق گر موازی برای انجام  $A+B$  لازم است. رابطه علامت ها را می توان با استفاده از OR انحصاری با ورودی های  $A_i$  و  $B_i$  معین کرد.

رویه به یک مقایسه گر، یک جمع کننده و دو تفربیق گر نیاز دارد. با این وجود رویه متفاوتی می توان یافت که امکانات کمتری نیاز داشته باشد. اول، می دانیم که تفربیق را می توان با جمع و متمم انجام داد. دوم، نتیجه مقایسه می تواند از رقم نقلی انتها بیایی، پس از تفربیق مشخص گردد. بررسی های دقیقتر روش های دیگر نشان می دهد که استفاده از متمم 2 برای تفربیق و مقایسه رویه بهتری است زیرا فقط یک جمع کننده و یک متمم کننده نیاز دارد.

شکل ۱۰-۱ بلاک دیاگرام سخت افزار مربوط به پیاده سازی اعمال جمع و تفربیق می باشد. این دیاگرام از ثبات های A و B و فلیپ فلاب های  $A_i$  و  $B_i$  برای علامت تشکیل شده است. تفربیق بوسیله جمع A با متمم 2 عدد B انجام می شود. رقم نقلی خروجی به فلیپ فلاب E منتقل می گردد و در آن جا برای تعیین اندازه های نسبی اعداد چک می شود. فلیپ فلاب جمع - سرریز، AVF، بیت سرریز را وقتي که A و B جمع شوند نگه می دارد. ثبات A سایر ریز اعمال 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 ۱۰-۱ سخت افزار جمع و تفربیق اندازه علامت دار

کنترل روش (شیوه)<sup>۱</sup> M در اختیار می‌گذارد. متمم ساز از گیت‌های OR انحصاری و جمع کننده موازی مشکل از مدارات تمام جمع کننده مطابق شکل ۴-۷ در فصل ۴ ساخته شده است. سیگنال M به ورودی نقلی جمع کننده هم متصل شده است. وقتی  $M=0$  باشد، خروجی B به جمع کننده منتقل می‌شود، رقم نقلی ورودی ۰، و خروجی جمع کننده برابر مجموع  $A+B$  می‌گردد. هنگامی که  $M=1$  باشد، متمم ۱ عدد B به جمع کننده اعمال، رقم نقلی ورودی ۱ و خروجی ۱ خواهد شد. این عمل برابر با جمع A با متمم ۲ عدد B است، که معادل تفريق  $B-A$  است.

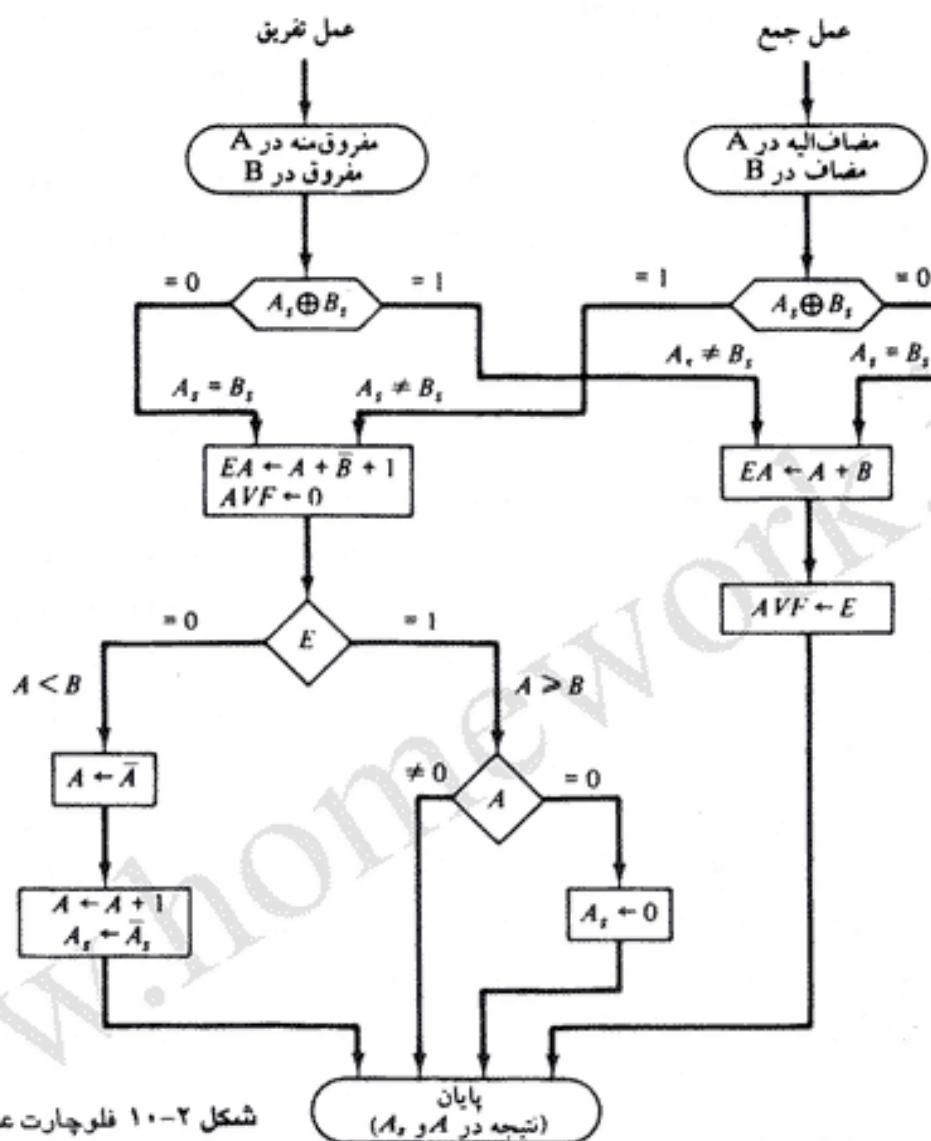
### الگوریتم سخت افزاری

فلوچارت الگوریتم سخت افزاری در شکل ۱۰-۲ نشان داده شده است. دو علامت  $A$  و  $B$  بوسیله گیت OR انحصاری مقایسه می‌شوند. اگر خروجی گیت ۰ باشد، علامت‌ها یکی هستند؛ اگر ۱ باشد علامت‌ها مختلفند. برای یک عمل جمع، علامت یکسان جمع اندازه‌ها را حکم می‌کند. برای عمل تفريق علامت متفاوت جمع اندازه‌ها را حکم می‌نماید. اندازه‌ها بوسیله ریز عمل  $EA \leftarrow A+B$  جمع می‌شوند، که  $EA$  ثباتی است که از ترکیب  $E$  و  $A$  بدست می‌آید. اگر رقم نقلی در  $E$ ، پس از جمع برابر ۱ باشد سریز رخ داده است. مقدار E به فلیپ فلاپ جمع سریز AVF منتقل می‌گردد.

اگر علامت‌ها در جمع مختلف باشند و یا در تفريق یکسان باشند عمل تفريق انجام می‌شود. اندازه‌ها با جمع A و متمم ۲ عدد B از یکدیگر تفريق می‌شوند. اگر اعداد از هم تفريق شوند هیچ سریزی رخ نمی‌دهد بنابراین  $AVF = B$  باشد. اگر  $A \geq B$  باشد  $E = B$  باشد و عدد درون A نتیجه صحیحی است. اگر A برابر ۰ باشد، علامت  $A$  باید مثبت گردد تا از داشتن صفر منفی جلوگیری شود. اگر E برابر ۰ باشد  $A > B$  است. برای این حالت لازم است متمم ۲ عدد A بدست آید. این عمل باریز عمل  $A \leftarrow A+1$  انجام می‌گیرد. با این وجود ما فرض خواهیم کرد که ثبات A مدارهای لازم برای متمم سازی و افزایش را داشته باشد، بنابراین متمم ۲ از این دو عمل حاصل خواهد شد. در مسیرهای دیگر فلوچارت، علامت نتیجه همان علامت A است و لذا هیچ تغییری در  $A$  لازم نیست. وقتی  $A < B$  باشد، علامت نتیجه متمم علامت اولیه A می‌باشد. در نتیجه لازم است  $A$  را برای بدست آوردن علامت صحیح متمم کنیم. نتیجه نهائی در A و علامت آن در  $A$  خواهد بود. مقدار در AVF بمعنى وقوع یک سریز است، مقدار نهایی در E اهمیتی ندارد.

### جمع و تفريق با داده‌های متمم ۲ علامت دار

نمایش متمم ۲ اعداد همراه با الگوریتم‌های حسابی برای جمع و تفريق در بخش ۳-۳ معرفی شدند. در اینجا آنها را برای سهولت مراجعة بطور خلاصه تکرار کنیم. سمت چپ‌ترین بیت عدد، علامت را نشان می‌دهد: ۰ برای مثبت و ۱ برای منفی. اگر بیت علامت ۱ باشد، تمام عدد به فرم متمم ۲ نمایش



شکل ۱۰-۲ فلوچارت عملهای جمع و تفریق

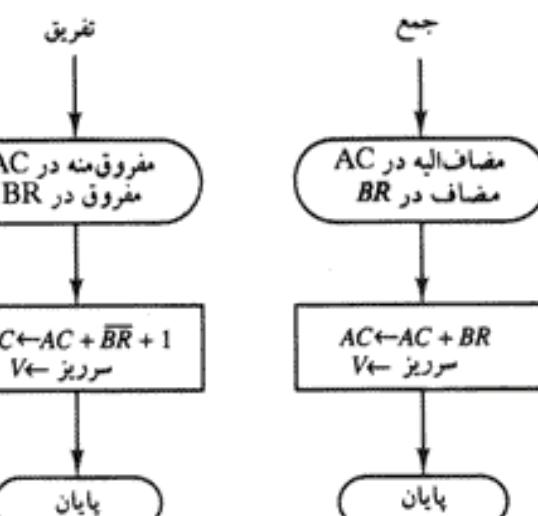
داده می شود. بنابراین  $+33$ - بصورت 00100001 و  $-33$ - بصورت 11011111 خواهد بود. توجه کنید که 11011111 متمم 2 عدد 00100001 می باشد و بالعکس.

در جمع دو عدد در فرم متمم 2 علامت دار، آن دو را بصورت عادی با هم جمع می کنیم و با بیتها علامت آنها نیز مانند سایر بیتها عدد رفتار می نمائیم. 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 منه 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 ۳-۳ 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 ۱۰-۳ نشان داده شده است. این شکل شبیه شکل ۱۰-۱ است جز اینکه بیت‌های علامت از بقیه ثبات‌ها جدا نشده‌اند. ما ثبات A را AC (انباره) و ثبات B را BR می‌نامیم. سمت چپ ترین 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. فلایپ فلاپ سرریز V پس از رخداد سرریز برابر ۱ می‌شود. در این حالت رقم نقلی خروجی در نظر گرفته نشده است.

الگوریتم جمع و تفریق دو عدد دودویی با نمایش متمم ۲ علامت‌دار در فلوچارت شکل ۱۰-۴ نشان داده شده است. حاصل جمع از جمع محتوای AC و BR (همراه با بیت‌های علامت) بدست می‌آید. بیت سرریز V هنگامی ۱ است که OR انحصاری دو رقم نقلی آخر ۱ باشند، در غیراینصورت ۰ است. عمل تفریق از جمع محتوای AC با متمم ۲ محتوای BR حاصل می‌گردد. گرفتن متمم ۲ عدد BR معنی تبدیل عدد مثبت به منفی است، و بالعکس. در حین این عمل سرریز باید وارسی گردد زیرا

شکل ۱۰-۳ سخت افزار جمع و تفریق متمم ۲ علامت‌دار



شکل ۱۰-۴ الگوریتم جمع و تفریق اعداد متمم ۲ 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 نگهداشیم عمل جمع و تفریق اعداد ساده‌تر خواهد بود. با این دلیل در اکثر کامپیوتراها این روش بجای روش آشنا‌تر مقدار علامت دار بکار می‌رود.

### ۱۰-۳ 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.

$$\begin{array}{r}
 & \text{مضروب} & 10111 \\
 & \times 10011 & \\
 \hline
 & 10111 & \\
 & 10111 & \\
 & 00000 + & \\
 & 00000 & \\
 \hline
 & 10111 & \\
 \hline
 437 & 110110101 &
 \end{array}$$

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 غیراینصورت صفرها در پائین کپی می‌گردند. اعدادی که در سطرهای متوالی کپی می‌شوند نسبت به سطر قبل 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 غیراینصورت منفی خواهد بود.

### پیاده سازی سخت افزاری برای داده‌های علامت دار

هنگام پیاده سازی ضرب در یک کامپیوتر دیجیتال، بهتر است فرآیند کمی تغییر بابد. 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 مرتبآً حاصلضرب‌های جزیی ۱ را در یک ثبات نگهداشی نمائیم. ثانیاً بجای شیفت مضروب به چپ، حاصلضرب جزیی به راست شیفت داده می‌شود که در نتیجه موقعیت نسبی حاصلضرب جزیی و مضروب همان موقعیت مطلوب خواهد بود. ثالثاً، وقتی که بیتی از مضروب فیه 0 باشد، لزومی ندارد که صفرها را با حاصلضرب جزیی

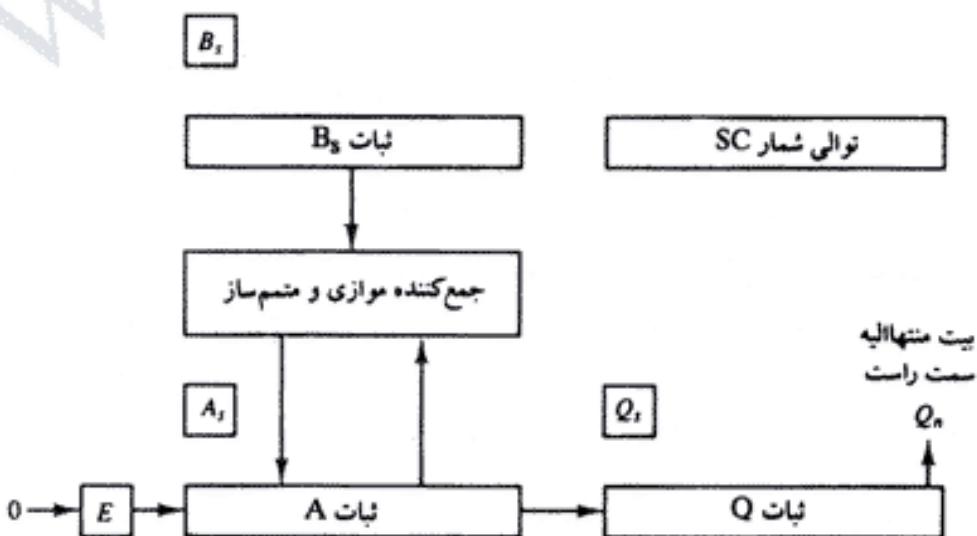
جمع کنیم زیرا مقدار آن را عوض نمی کند.

سخت افزار ضرب از امکانات شکل ۱۰-۱ بعلاوه دو ثبات اضافی دیگر تشکیل شده است. این ثبات به همراه ثبات های A و B در شکل ۱۰-۵ نشان داده شده اند. مضروب فیه در ثبات Q و علامت ش در Q<sub>8</sub> ذخیره می شود. توالی شمار SC ابتدأ با عددی برابر با تعداد بیت های مضروب فیه مقداردهی می شود. شمارنده پس از هر بار تشکیل حاصلضرب جزیی یک واحد کم می شود. وقتی که محتوای شمارنده به صفر برسد، حاصلضرب تکمیل و فرآیند متوقف می گردد.

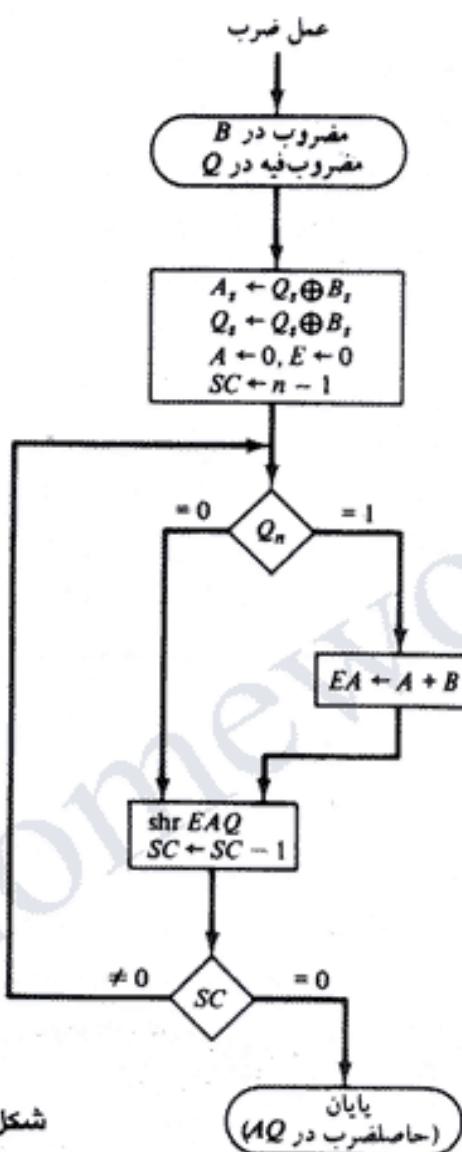
در آغاز مضروب در B و مضروب فیه در Q قرار دارد. حاصل جمع A و B تشکیل حاصلضرب جزیی را می دهد که به ثبات EA منتقل می گردد. حاصلضرب جزیی و مضروب فیه تواماً به راست شیفت داده می شوند. این شیفت توسط عبارت shr EAQ نشان داده شده است که شیفت به راست را در شکل ۱۰-۵ مشخص می نماید. بیت کم ارزشتر A به با ارزشترین بیت Q منتقل می شود، بیت E به با ارزشترین بیت A نقل مکان می یابد، و یک 0 وارد E می شود، یک بیت از حاصلضرب جزیی به داخل Q شیفت کرده و بیت های مضروب فیه را یک واحد به سمت راست می راند. با این ترتیب، سمت راست ترین فلیپ فلاپ در ثبات Q، که با Q<sub>8</sub> مشخص شده است، حاوی بیتی از مضروب فیه خواهد بود که در مرحله بعدی باید بررسی شود.

### الگوریتم سخت افزاری

شکل ۱۰-۶ فلوچارت الگوریتم سخت افزاری ضرب را نشان می دهد. ابتدأ مضروب در B و مضروب فیه در Q قرار دارد. علامت های مربوطه به ترتیب در B<sub>8</sub> و Q<sub>8</sub> 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 ۱۰-۵ سخت افزار عمل ضرب



شکل ۱۰-۶ فلوچارت 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 منتقل شده‌اند. چون هر عملوند باید همراه با علامتش ذخیره شود لذا یک بیت از کلمه توسط علامت اشغال شده و مقدار علمند 1-n 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 جزیی موجود در A جمع می‌شود. اگر 0 باشد، کاری انجام نمی‌شود. سپس ثبات EAQ یکبار بسمت راست شیفت داده می‌شود تا حاصلضرب جزیی را تشکیل دهد. توالی شمار 1 واحد کم شده و مقدار جدید آن چک می‌شود. اگر برابر صفر نباشد، فرآیند

تکرار شده و حاصلضرب جزیی جدید تشکیل می‌گردد. وقتی که  $SC=0$  شود فرآیند متوقف می‌گردد. توجه کنید که حاصلضرب جزیی حاصل در A هر بار یک بیت به Q منتقل می‌گردد تا نهایتاً جای مضروب فيه 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 رانگه می‌دارند.

مثال عددی قبلی در جدول ۱۰-۲ برای روشن شدن فرآیند ضرب سخت افزاری تکرار شده است. روند، مراحل مشخص شده در فلوچارت 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 فيه نیازی به جمع ندارند بلکه فقط جابجایی (شیفت) لازم دارند و رشته‌های ۱ در مضروب فيه از بیت مرتبه  $2^k$  تا بیت  $2^m$  را می‌توان معادل  $2^{k+1}-2^m$  تلقی کرد. مثلاً عدد دودویی ۰۰۱۱۱۰ (۱۴+) دارای رشته‌های ۱ از  $2^1$  تا  $2^3$  است، ( $m=1$ ،  $k=3$ ). این عدد را می‌توان بصورت  $14 = 2^{k+1}-2^m = 2^4-2^1 = 16-2 = 14$  نوشت. بنابراین ضرب  $M \times 14$  را، که در آن M مضروب و ۱۴ مضروب فيه 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 فيه و شیفت حاصلضرب جزیی نیاز دارد. قبل از شیفت، ممکن است مضروب طبق قواعد زیر با حاصلضرب جزیی

جدول ۱۰-۲ مثال عددی برای ضرب کننده دودویی

$B = 10111$	مضروب فيه	E	A	Q	SC
	مضروب فيه در $Q$	0	00000	10011	101
	$+ Q_n = 1$ + جمع		<u>10111</u>		
	اولین حاصلضرب جزیی	0	<u>10111</u>		
	شیفت EAQ به راست	0	01011	11001	100
	$+ Q_n = 1$ + جمع		<u>10111</u>		
	دومین حاصلضرب جزیی	1	<u>00010</u>		
	شیفت EAQ به راست	0	10001	01100	011
	$+ Q_n = 0$ + شیفت EAQ به راست	0	01000	10110	010
	$+ Q_n = 0$ + شیفت EAQ به راست	0	00100	01011	001
	$+ Q_n = 1$ + جمع		<u>10111</u>		
	پنجمین حاصلضرب جزیی	0	<u>11011</u>		
	شیفت EAQ به راست	0	01101	10101	000
حاصلضرب نهایی در AQ برابر ۰۱۱۰۱۱۰۱۰۱۰۱۰۱۱۰ است					

جمع می شود، از آن تفیریق شود و یا حاصلضرب جزیی بلا تغییر باقی بماند.

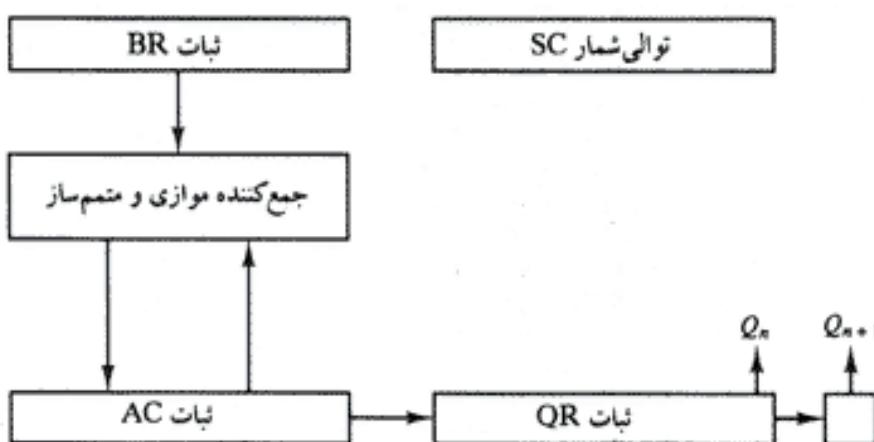
۱- به محض برخورد با اولین  $1 \text{ کم ارزش}$  در رشته ۱ ها در مضروب فیه، مضروب از حاصلضرب جزیی کم می شود.

۲- به محض برخورد با اولین  $0$  (بشرطی که قبل از آن  $1$  باشد) در رشته ای از  $0$  ها در مضروب فیه، مضروب با حاصلضرب جزیی جمع می شود.

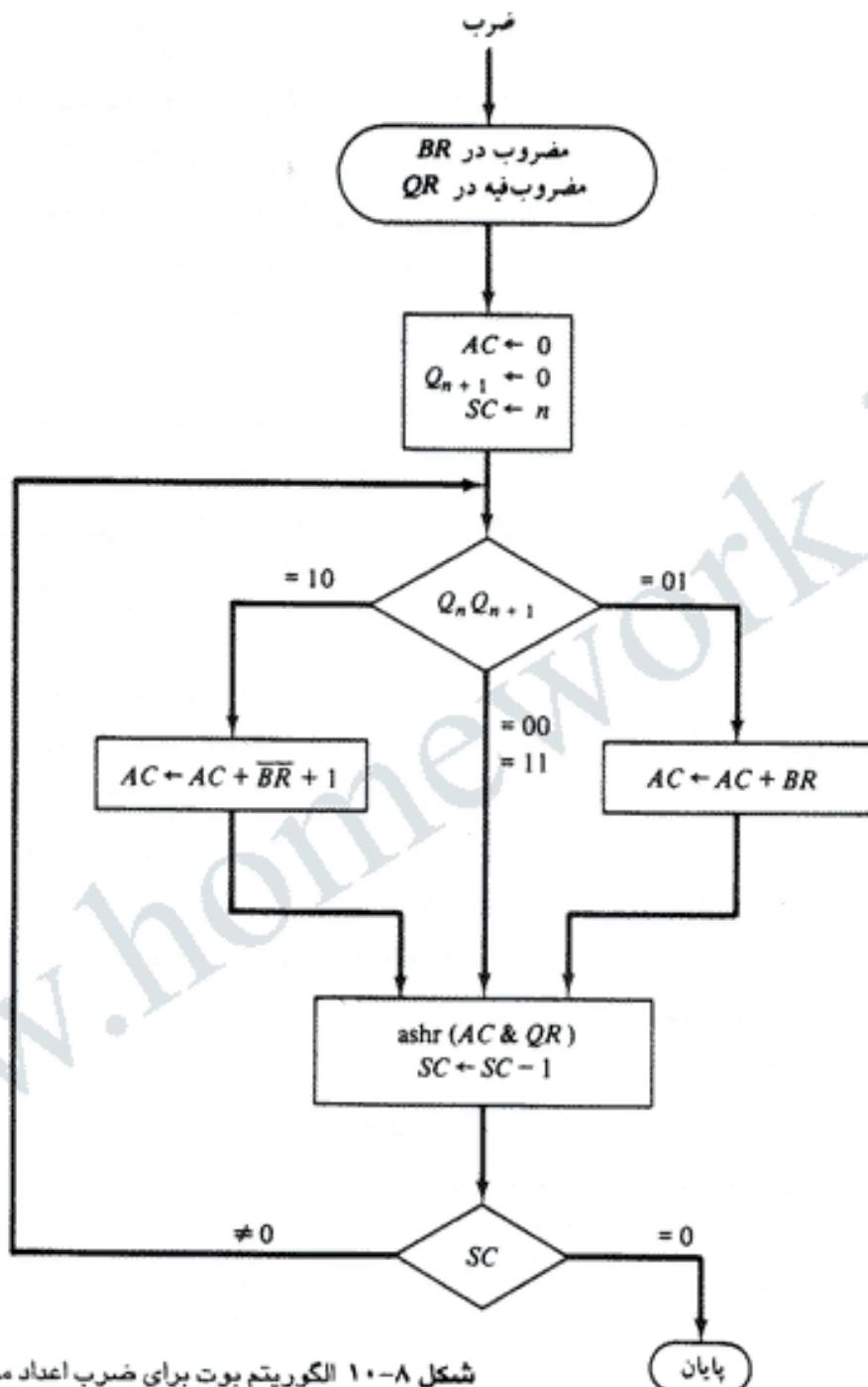
۳- وقتی که بیت جاری مضروب فیه همانند بیت قبلی باشد، حاصلضرب جزیی تغییری نمی کند.

الگوریتم فوق برای مضروب فیه های مثبت و یا منفی بفرم متمم ۲ قابل استفاده است. این بدان علت است که مضروب فیه منفی به رشته ای از  $1$  ها خاتمه می یابد و آخرین عمل تفیریق با وزن مناسب خواهد بود. مثلاً مضروب فیه  $14 -$  در نمایش متمم ۲ عبارتست از  $110010$  و بصورت  $-14 = 2^4 + 2^2 - 2^1$  با آن رفتار می شود.

پیاده سازی سخت افزار الگوریتم بوت آرایش ثبات شکل ۱۰-۷ را نیاز دارد. این شکل مشابه شکل ۱۰-۵ است جز اینکه بیت های علامت از بقیه ثبات ها تفکیک نشده اند. برای ملاحظه این اختلاف، ما ثبات های  $A$  و  $B$  و  $Q$  را بترتیب  $AC$ ،  $BR$  و  $QR$  می نامیم.  $Q_n$  کم ارزشترین بیت مفروق منه در ثبات  $QR$  است. یک فلیپ فلاپ اضافی  $Q_{n+1}$  برای بررسی همزمان دو بیت از مضروب فیه به  $QR$  ملحق شده است. فلوچارت الگوریتم بوت در شکل ۱۰-۸ نشان داده شده است.  $AC$  و بیت الحاقی ابتدا پاک شده اند و توالی شمار  $SC$  برابر تعداد بیت های مضروب فیه یعنی  $n$ ، می شود. دو بیت از مضروب فیه که در  $Q_n$  و  $Q_{n+1}$  قرار دارند مورد بررسی قرار می گیرند. اگر دو بیت برابر  $10$  باشند بدان معنی است که اولین  $1$  در رشته  $1$  ها فراسیده است. این حالت، تفیریق مضروب از حاصلضرب جزیی موجود در  $AC$  را لازم می دارد. 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 جزیی موجود در  $AC$  را لازم می دارد. هرگاه دو بیت برابر باشند



شکل ۱۰-۷ سخت افزار الگوریتم بوت



شکل ۱۰-۸ الگوریتم بوت برای ضرب اعداد متمم ۲ علامت دار

حاصل ضرب جزیی تغییری نمی‌کند. سرریز نمی‌تواند رخداد 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 حاصل ضرب جزیی 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 ۴-۶ ملاحظه شود). توالی شمار کاهش یافته

و حلقه محاسبه  $n$  بار تکرار می‌گردد.

مثال عددی الگوریتم بوت در جدول ۱۰-۳ برای  $n=5$  نشان داده شده است. این مثال، ضرب  $(-13) \times (-9)$  را نشان می‌دهد. توجه کنید که مضروب فيه در QR منفی و مضروب نیز در BR منفی است. حاصلضرب ۱۰ بیتی در AC و QR قرار داشته و مثبت است. مقدار نهایی  $Q_{n+1}$  علامت اولیه مضروب فيه است و نباید بخشی از حاصلضرب تلقی شود.

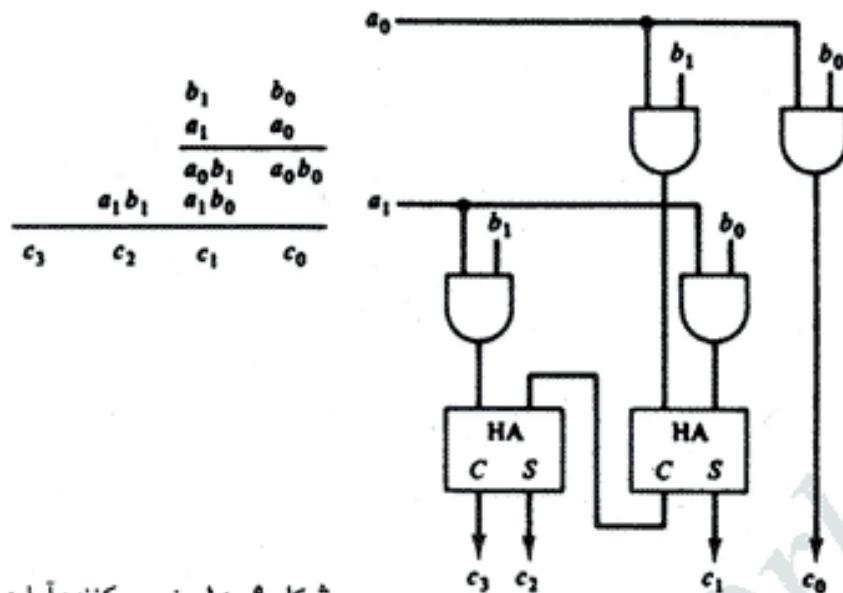
### ضرب کننده آرایه‌ای

تست یک به یک بیت‌های مضروب فيه و تشکیل حاصلضرب جزیی یک عمل ترتیبی است که دنباله‌ای از جزء اعمال جمع و شیفت را نیاز دارد. ضرب دو عدد دودویی بوسیله یک ریزعمل و یکمک مدار ترکیبی که بیت‌های ضرب را یکباره 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 ۲ بیت را مطابق شکل ۱۰-۹ در نظر بگیرید. بیت‌های مضروب  $b_1$  و  $b_0$  و بیت‌های مضروب فيه  $a_1$  و  $a_0$  و حاصلضرب  $c_3c_2c_1c_0$  می‌باشند. اولین حاصلضرب جزیی با ضرب  $a_0$  در  $b_1b_0$  بدست می‌آید. ضرب دو بیت مانند  $a_0$  و  $b_0$  برابر ۱ خواهد بود بشرطی که هر دو ۱ باشند، در غیراینصورت ۰ است. این با عمل AND یکسان است و می‌تواند با یک گیت AND پیاده سازی شود. همانطور که در شکل دیده می‌شود، اولین حاصلضرب جزیی با دو گیت AND حاصل می‌شود. دومین حاصلضرب جزیی از ضرب

جدول ۱۰-۳ مثال ضرب با الگوریتم بوت

$Q, Q_{n+1}$	$\frac{BR}{BR} + 1 = 01001$	AC	QR	$Q_{n+1}$	SC
1 0	مقدار اولیه تغیریق	00000 <u>01001</u> 01001	10011	0	101
1 1	ashr	00100	11001	1	100
0 1	ashr	00010	01100	1	011
	BR جمع	10111 11001			
0 0	ashr	11100	10110	0	010
1 0	ashr	11110	01011	0	001
	تغیریق	01001 00111			
	ashr	00011	10101	1	000



شکل ۱۰-۹ ضرب کننده آرایه ۲ بیت در ۲ بیت

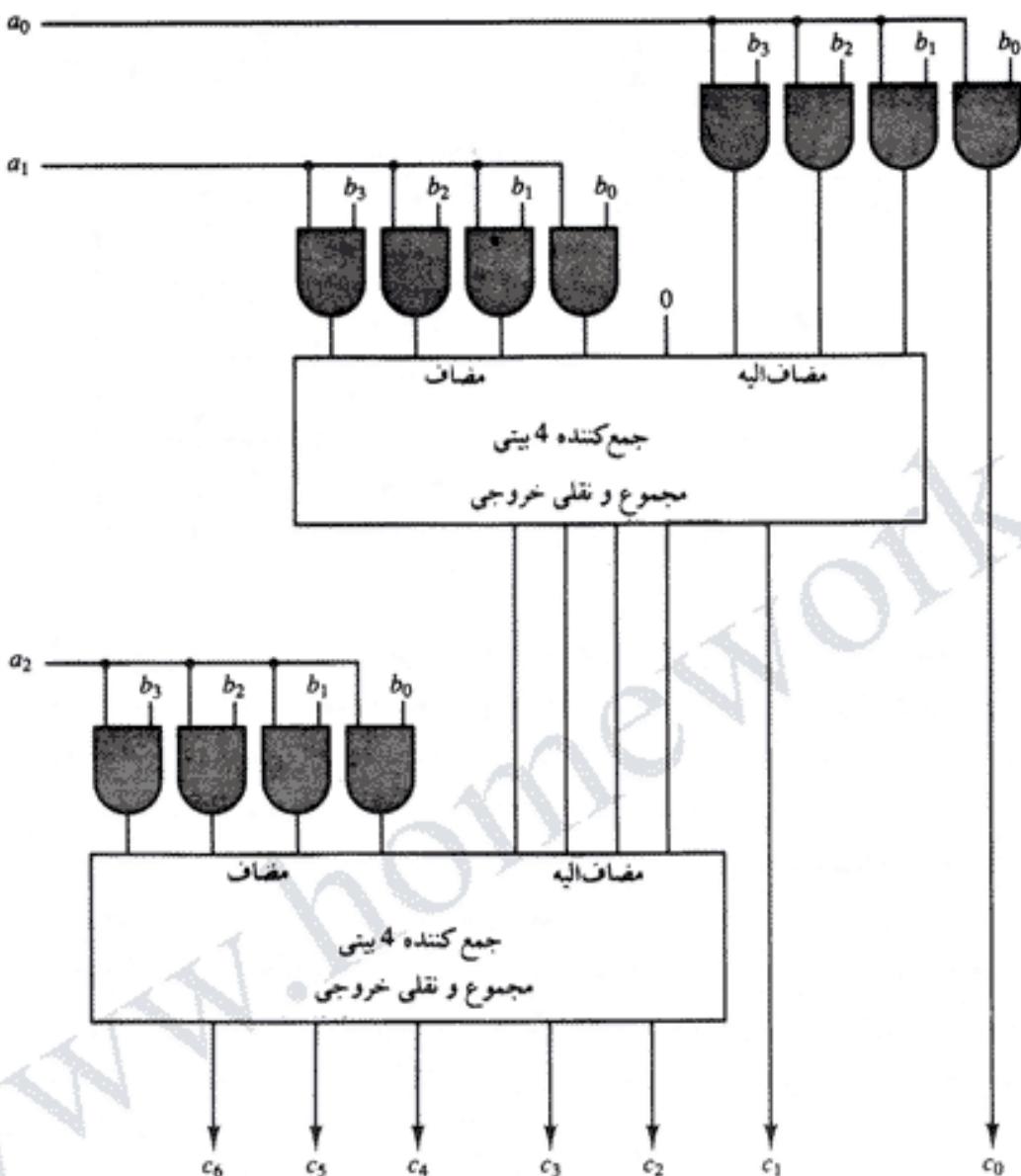
$a_1$  در  $b_1b_0$  و شیفت آن به چپ بدست می‌آید. دو حاصلضرب جزیی بوسیله دو مدار نیم جمع کننده (HA) با یکدیگر جمع می‌شوند. معمولاً، تعداد بیت‌های حاصلضرب جزیی بیشتر است و لازم خواهد بود تا از تمام جمع کننده استفاده شود. وقت داشته باشید که کم ارزشترین بیت حاصلضرب لازم نیست وارد جمع کننده شود زیرا توسط خروجی گیت AND تشکیل می‌گردد.

یک مدار ضرب کننده دودویی با بیت‌های بیشتر را می‌توان بروشی مشابه ساخت. هر بیت از مضروب فيه با هر بیت از مضروب، بتعادل بیت‌های مضروب فيه، AND می‌شود. خروجی دودویی در هر یک از سطوح گیت‌های AND بطور موازی با حاصلضرب جزیی قبلی جمع می‌شود و حاصلضرب جزیی جدیدی را تشکیل می‌دهد. آخرین سطح، حاصلضرب را ایجاد می‌نماید. برای مضروب فيه  $j$  بیتی و مضروب  $k$  بیتی نیاز به  $j \times k$  گیت AND و  $k - j$  گیت AND می‌باشد. 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 می‌کند. فرض کنید مضروب را با  $b_3b_2b_1b_0$  و مضروب فيه را با  $a_2a_1a_0$  نشان می‌دهیم. چون  $k=4$  و  $j=3$  است، ۱۲ گیت AND و ۲ گیت 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 الگوریتم‌های تقسیم

در تقسیم دو عدد ممیز ثابت در نمایش مقدار علامت دار با قلم و کاغذ، از اعمال مقایسه، شیفت و تفریق استفاده می‌شود. تقسیم دودویی از تقسیم دهدی ساده‌تر است زیرا ارقام خارج قسمت ۰ یا ۱ هستند و نیازی نیست تا بدانیم مقسوم یا باقیمانده چند بار در مقسوم علیه جای می‌گیرد. فرآیند تقسیم با



شکل ۱۰-۱۰ ضرب آرایه 4 بیت در 3 بیت

مثال عددی شکل ۱۰-۱۱ توضیح داده شده است. مفروم علیه B از پنج بیت و مفروم A از ده بیت تشکیل شده است. پنج بیت با ارزشتر مفروم با مفروم علیه مقابله می‌شوند. چون این پنج بیت از B کوچکتر است، مقابله B را با شش بیت بالرزشتر A آزمایش می‌کنیم. این عدد شش بیتی بزرگتر از B است، لذا رقم 1 را برای خارج قسمت در بالای بیت ششم مفروم علیه نویسیم. سپس مفروم علیه را یک بار به راست شبکت می‌دهیم و آن را از مفروم کم می‌کنیم. تفاصل بدست آمده باقیمانده جزیی<sup>۱</sup> خوانده می‌شود زیرا تقسیم ممکن است در همین مرحله پایان یابد و خارج قسمت 1 و باقیمانده‌ای برابر با این باقیمانده جزیی را داشته

---

#### 1- Partial Reminder

Divisor:	<u>11010</u>	خارج قسمت $Q = \frac{A}{B}$ مقسوم
$B = 10001$	<u>0111000000</u>	5 بیت از $A < B$ ، خارج قسمت 5 بیت دارد
	<u>01110</u>	6 بیت از $A \geq B$ هستند
	<u>011100</u>	جا به جایی B به راست و تفریق، قرار دادن 1 در Q
	<u>-10001</u>	7 بیت از باقیمانده بزرگتر یا مساوی با B هستند
	<u>-010110</u>	جا به جایی B به راست و تفریق، قرار دادن 1 در Q
	<u>--10001</u>	باقیمانده کوچکتر از B ، قرار دادن 0 در Q ، شیفت B به راست
	<u>--001010</u>	باقیمانده بزرگتر یا مساوی با B
	<u>---010100</u>	شیفت B به راست و تفریق، قرار دادن 1 در Q
	<u>----10001</u>	باقیمانده کوچکتر از B ، قرار دادن 0 در Q
	<u>-----000110</u>	باقیمانده نهایی
	<u>-----00110</u>	

شکل ۱۰-۱۱ مثال تقسیم دودویی

پاشیم. روند تقسیم با مقایسه باقیمانده جزیی با مقسوم علیه ادامه می‌یابد. اگر باقیمانده جزیی 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 جزیی کم می‌شود. 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 می‌شود. 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 جزیی را به چپ شیفت می‌دهیم و به این ترتیب موقعیت نسبی دو عدد همان موقعیت مورد نظر خواهد بود. تفریق را می‌توان با جمع A و متمم 2 عدد B انجام داد. نسبت اندازه ها از رقم نقلی انتهایی مشخص می‌گردد.

سخت افزار مربوط به پیاده سازی تقسیم همان سخت افزار ضرب است و از قطعات شکل ۱۰-۵ 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 می‌رود. مثال عددی مجددآ در شکل ۱۰-۱۲ 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 می‌شوند. مقسوم به سمت چپ شیفت داده می‌شود و متمم 2 مقسوم علیه با آن جمع شده و لذا عمل تفریق تحقق می‌یابد. اطلاعات مربوط به نسبت اندازه ها در E موجود است. اگر  $E=1$  باشد معنی  $A \geq B$  است. برای خارج قسمت، بیت 1 وارد  $Q_n$  می‌گردد و باقیمانده جزیی به سمت چپ شیفت پیدا می‌کند تا روند تکرار گردد. اگر  $E=0$  باشد، یعنی  $B > A$  است و بنابراین خارج قسمت در  $Q_n$  برابر 0 (که هنگام شیفت وارد آن شده است) می‌ماند. سپس مقدار B مجددآ جمع می‌شود تا باقیمانده جزیی در A به مقدار قبلی اش بازگردانده شود. باقیمانده جزیی به چپ شیفت داده شده و روند مجددآ تکرار می‌گردد تا اینکه

$B = 10001$ , مقسوم علیه:

$\bar{B} + 1 = 01111$

	E	A	Q	SC
shl EAQ	0	01110 11100 <u>01111</u>	00000 00000	5
جمع				
E = 1	1	01011		
$Q_n=1$ قرار دهد	1	01011	00001	*
shl EAQ	0	10110 <u>01111</u>	00010	
جمع				
E = 1	1	00101		
$Q_n=1$ قرار دهد	1	00101	00011	3
shl EAQ	0	01010 <u>01111</u>	00110	
جمع				
$Q_n=0$ , بگذارید E=0	0	11001 <u>10001</u>	00110	
B				2
جمع				
بازیابی باقیمانده	1	01010		
shl EAQ	0	10100 <u>01111</u>	01100	
جمع				
E = 1	1	00011		
$Q_n=1$ قرار دهد	1	00011	01101	1
shl EAQ	0	00110 <u>01111</u>	11010	
جمع				
$Q_n=0$ , بگذارید E=0	0	10101 <u>10001</u>	11010	
B				
جمع				
بازیابی باقیمانده	1	00110	11010	0
E را تأثیر نماید، بگیرید				
باقیمانده در A		00110		
خارج قسمت در Q			11010	

شکل ۱۰-۱۲ مثال تقسیم دودویی با سخت‌افزار دیجیتال

هر پنج بیت خارج قسمت ایجاد شود. توجه کنید که ضمن شیفت باقیمانده جزیی به چپ، بیت‌های خارج قسمت نیز شیفت داده می‌شود و پس از پنج شیفت، خارج قسمت در Q و باقیمانده در A خواهد بود. قبل از نمایش الگوریتم بصورت فلوچارت، باید علامت نتیجه و تحالت سرریز احتمالی را در نظر داشت. علامت خارج قسمت از علامت‌های مقسوم و مقسوم علیه تعیین می‌شود. اگر علامت‌های 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 سخت‌افزار مسئله ساز خواهد بود. دلیل این است که طول ثبات‌ها محدود است و نمی‌توانند عددی را که از طول استاندارد تجاوز کند نگه دارند. برای درک موضوع، سیستمی را که دارای ثبات‌های پنج بیتی است در نظر بگیرید. مالزیک ثبات برای نگهداری

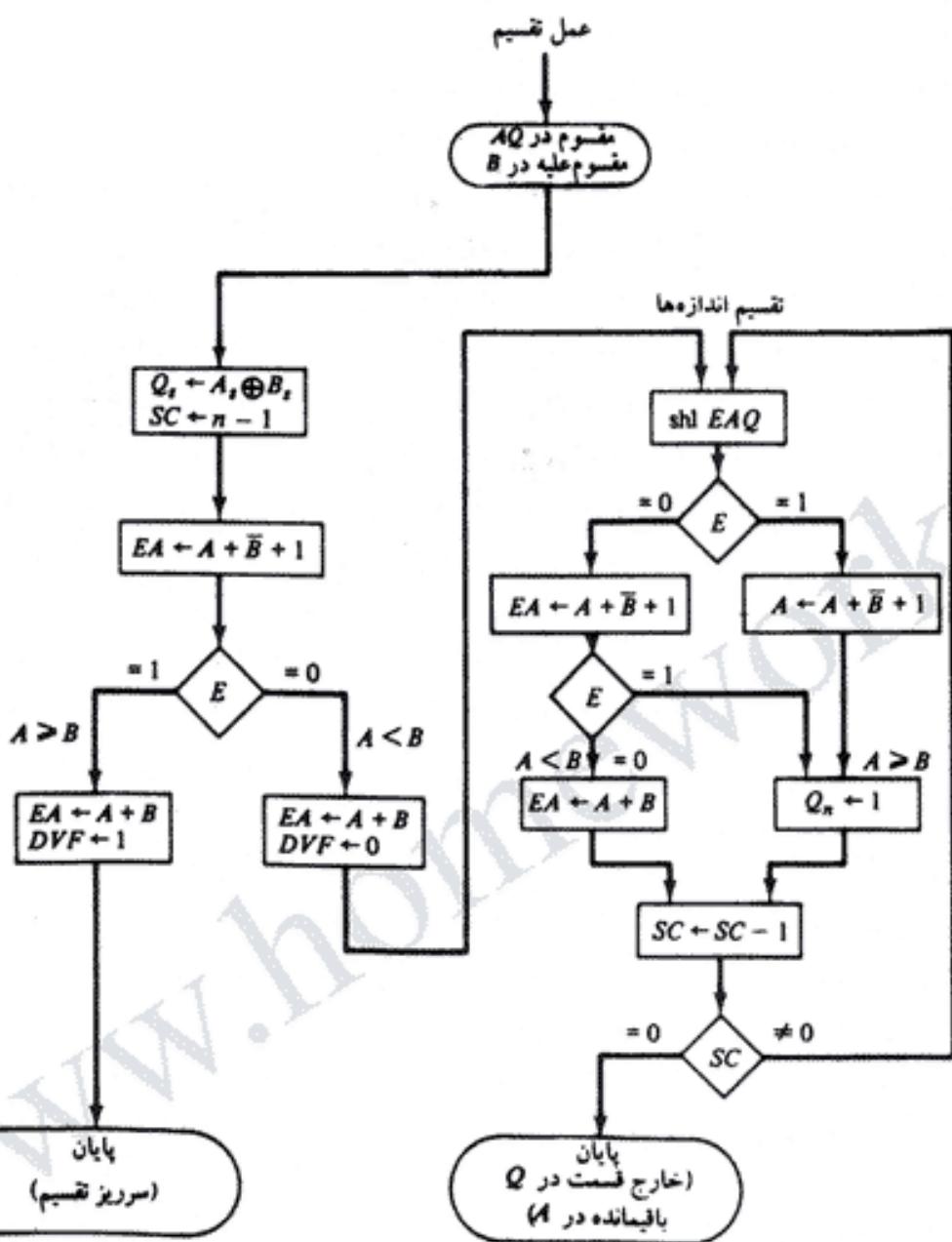
مقسم علیه و دو ثبات برای نگهداری مقسم استفاده می‌کنیم. با توجه به مثال شکل ۱۱-۱۰ می‌بینیم که خارج قسمت شش بیت است بشرطی که پنج بیت بالارزشتر مقسم، عددی بزرگتر از مقسم علیه باشد. خارج قسمت می‌باید دریک ثبات پنج بیت استاندارد ذخیره شود، بنابراین بیت سرریز مستلزم یک فلیپ‌فلاب اضافی برای ذخیره بیت ششم خواهد بود. در عملیات عادی کامپیوتر باید از بکارگیری این بیت سرریز خودداری شود زیرا در این صورت طول ۶ لی 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 يا نرم افزار کامپیوتر و يا ترکیبی از هر دو باید امکاناتی را جهت آشکارسازی این وضعیت فراهم کرد.

اگر طول مقسم دوبرابر مقسم علیه باشد، حالت سرریز را می‌توان بطريق زیر بیان کرد: حالت سرریز در تقسیم هنگامی اتفاق می‌افتد که بیت‌های نیمه پرازشتر مقسم، عددی بزرگتر یا مساوی مقسم علیه را تشکیل دهند. مسئله دیگری که باید در تقسیم درنظر داشت این است که از تقسیم بر صفر اختراز شود. حالت سرریز در تقسیم مراقبت از این وضعیت را بعده دارد. این بدان علت است که اگر مقسم علیه صفر باشد، مقسم قطعاً از مقسم علیه بزرگتر یا مساوی آن است. حالت سرریز معمولاً با اشدن فلیپ‌فلاب خاصی مشخص می‌گردد و آنرا فلیپ‌فلاب سرریز خوانده و با DVF نشان می‌دهیم. در قبال وقوع سرریز تقسیم بطريق مختلف می‌توان عکس العمل نشان داد. در بعضی کامپیوتراها مسئولیت با برنامه نویس است تا پس از هر دستورالعمل تقسیم DVF را چک کند. در این صورت می‌توان به زیروالی انشعباب کرده و اقدامی اصلاحی همچون تغییر مقیاس داده‌ها را برای جلوگیری از سرریز انجام داد. در برخی کامپیوتراهای قدیمی‌تر، وقوع سرریز تقسیم سبب توقف کامپیوتر می‌شود و این حالت، توقف تقسیم نامیده می‌شود. امروزه متوقف کردن کار کامپیوتر توصیه نمی‌شود زیرا اتلاف کننده وقت است. روشی که در اکثر کامپیوتراها انجام می‌شود تولید درخواست وقفه<sup>۱</sup> 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 انشعباب 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 ۱۰-۵ 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. مقسم در A و Q و مقسم 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 ۱۰-۱۳ فلوچارت عمل تقسیم

فرض می‌کنیم که عملوندات 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 سریز تقسیم با تقریق مقسوم علیه در  $B$  از نیمی از بیت‌های ذخیره شده در  $A$  تست می‌شود. اگر  $A \geq B$  باشد، فلیپ فلاپ سریز تقسیم،  $DVF$  برابر 1 شده و عمل بطور ناقص متوقف می‌شود. اگر  $A < B$  باشد، سریزی رخ نداده و مقدار مقسوم با جمع  $B$  با  $A$  بازیافته می‌شود. تقسیم اندازه‌های دور عدد با جابجا کردن مقسوم موجود در  $AQ$  به چپ شروع می‌شود که در این جابجا یابی پاراز شتر به  $E$

منتقل می شود اگر بیت انتقال یافته به  $E$  برابر ۱ باشد، در می یابیم که  $EA > B$  است زیرا  $EA$  از رقم ۱ 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 ۱ 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  $2^{n-1} EA - 2^{n-1}$  خواهد بود. از جمع متهم دو  $B$  با این مقدار داریم:

$$(EA - 2^{n-1}) + (2^{n-1} - B) = EA - B$$

اگر بخواهیم  $E$  برابر ۱ بماند رقم نقلی عبارت فوق به  $E$  منتقل نمی شود. اگر عمل شیفت به چپ مقدار ۰ را وارد  $E$  کند، مقسوم علیه با جمع متهم ۲ آن تفریق می شود و مقدار رقم نقلی به  $E$  انتقال می یابد. اگر  $E=1 \geq B$  است؛ بنابراین به  $Q_n$  ۱ داده خواهد شد. اگر  $E=0$  باشد،  $A > B$  و عدد اولیه با جمع  $B$  با  $A$  بازیابی می شود. در مورد اخیر مقدار ۰ را در  $Q_n$  باقی می گذاریم (مقدار ۰ 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  $Q$  و علامت باقیمانده نیز که همان علامت مقسوم است در  $A$  قرار دارد.

### سایر الگوریتم ها

روش سخت افزاری که در فوق شرح داده شد روش بازیافتنی<sup>۱</sup> خوانده می شود. دلیل این نامگذاری این است که باقیمانده جزیی از جمع مقسوم علیه با تفاضل منفی بازیافته می شود. دوروش دیگر برای تقسیم اعداد وجود دارد، روش مقایسه ای و روش غیربازیافتنی<sup>۲</sup>. 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 جزیی به چپ شیفت داده شده و مجدداً مقایسه تکرار می گردد. می توان مقایسه را قبل از تفریق با وارسی رقم نقلی انتهایی جمع کننده موازی، قبل از انتقال آن به ثبات  $E$  انجام داد.

در روش غیربازیافتنی، 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 بینیم چگونه این ممکن است، حالتی را در نظر بگیرید که در آن  $A > B$  است. باتوجه به فلوچارت ۱۱-۹ 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 ۲ ضرب می شود) و دوباره  $B$  از آن کم می شود. نتیجه این اعمال  $2(A-B) - B = 2A - B$  است. این نتیجه در روش غیربازیافتنی 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 غیربازیافتنی، اگر مقدار

قبلی  $Q_n$  برابر 1 باشد  $B$  تفریق می‌گردد، اما اگر مقدار قبلی  $Q_n$  برابر 0 باشد  $B$  جمع می‌شود و بازیابی باقیمانده جزیی 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 شیفت می‌یابد،  $B$  باید تفریق شود. همچنین، اگر آخرین بیت خارج قسمت 0 باشد، باقیمانده جزیی باید بازیابی شود تا باقیمانده نهایی صحیح باشد.

## ۱۰-۵ عمل‌های حسابی ممیز شناور

بسیاری از زبان‌های برنامه‌نویسی امکاناتی برای مشخص کردن عدددهای معیز شناور دارند. متداول‌ترین روش، مشخص کردن آنها با عبارت **real** (حقیقی) در برابر اعداد ممیز ثابت است که با عبارت **Integer** (صحیح) مشخص می‌گردد. 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 معیز شناور باشد. هرچند روش سخت افزاری گرانتر است، ولی کارآیی آن بقدری از روش نرم افزاری بیشتر است که در اکثر کامپیوترهای سخت افزار ممیز شناور کار گذاشته می‌شود و فقط در کامپیوترهای خیلی کوچک از آن صرف نظر نظر می‌شود.

### ملاحظات اساسی

نمایش ممیز شناور داده در بخش ۳-۴ معرفی شد. عدد ممیز در ثبات‌های کامپیوتراز دو بخش تشکیل می‌شود: مانتیس  $m$  و نمای  $e$ . این دو بخش نماینده عددی است که از ضرب  $m$  در  $2^e$  به نمای  $e$  بدست می‌آید. یعنی

$$m \times 2^e$$

مانتیس ممکن است که عددی کسری یا صحیح باشد. محل ممیز و مقدار پایه  $e$  فرض تصور شده‌اند و در ثبات‌ها وارد نشده‌اند. مثلاً نمایش کسری و پایه 10 را در نظر بگیرید. عدد دهدھی 537.25 در یک ثبات با  $e=3$  و  $m=53725$  نمایش داده می‌شود و چنین تفسیر می‌گردد که نماینده عدد ممیز شناور زیر است:

$$0.53725 \times 10^3$$

اگر بالرزشترین رقم مانتیس یک عدد ممیز شناور و غیرصفر باشد آن را نرمالیزه می‌کنیم. در نتیجه مانتیس دارای بیشترین تعداد ممکن رقم‌های معنی دار خواهد بود. صفر را نمی‌توان نرمالیزه کرد زیرا رقم غیرصفر ندارد. در نمایش ممیز شناور، صفر را با مقدار تمام 0 در مانتیس و نمایشان 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  $(1-10^{47}) \pm$  خواهد بود که تقریباً  $\pm 10^{14}$  می باشد. از این 48 بیت می توان برای نمایش یک عدد میز شناور با 36 بیت برای ماتیس و 12 بیت برای نماستفاده کرد. با فرض نمایش کسری برای ماتیس و با احتساب دو بیت 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. ماتیس می تواند 35 بیتی باشد ( جدا از علامت) و اگر عدد صحیح تلقی شود می تواند تا  $(2^{35}-1)$  را ذخیره نماید. این تقریباً برابر با  $10^{10}$  می باشد، که معادل یک عدد دهدۀ 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 میکرو کامپیوتر 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 ماتیس 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 ماتیس ها، باید بخش نمای دو عدد با هم مساوی شود. هم ردیف کردن با جابجایی یکی از ماتیس ها و تنظیم نمای آن تا جایی که با دیگری برابر شود انجام می شود. جمع اعداد ممیز شناور زیر را در نظر بگیرید.

$$\begin{aligned} & 0.5372400 \times 10^2 \\ & + 0.1580000 \times 10^{-1} \end{aligned}$$

لازم است که دونما قبل از جمع ماتیس ها باهم مساوی شوند. می توانیم عدد اول را سه مکان به چپ، یا عدد دوم را سه مکان به راست شیفت دهیم. وقتی که ماتیس ها در ثبات ها ذخیره شوند، شیفت به چپ سبب از دست دادن ارقام با ارزشتر می شود. روش دوم ترجیح دارد چون میزان دقت را کاهش می دهد در حالی که روش اول ممکن است موجب بروز خطأ شود. رویه هم ردیف کردن معمولاً با این ترتیب است که ماتیسی را که نمای کوچکتر دارد باندازه اختلاف بین نمایها به راست شیفت دهیم. پس از این عمل ماتیس ها می توانند با هم جمع شوند.

$$\begin{array}{r} 0.5372400 \times 10^2 \\ + 0.0001580 \times 10^2 \\ \hline 0.5373980 \times 10^2 \end{array}$$

هنگامی که دو ماتیس نرمالیزه شده باهم جمع ممکن است رقم سرریز داشته

باشد. سرریز را به سادگی می‌توان با شیفت حاصل جمع به راست و افزایش نما تصحیح کرد. در تفریق مثل مثال زیر نتیجه ممکن است دارای صفرهایی در مکان‌های بالارتیه باشد.

$$\frac{0.56780 \times 10^5}{- 0.56430 \times 10^5} \\ \frac{0.00350 \times 10^5}$$

وقتی عدد میز شناوری دارای ۰ در مکان‌های با ارزشتر باشد گوئیم فروریز<sup>۱</sup> دارد. برای نرمالیزه کردن اعداد فروریز دار، لازم است تا مانیس 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 مانیس را دوبار به چپ شیفت دهیم تا  $0.35000 \times 10^3$  حاصل شود. در بسیاری از کامپیوتراها رویه نرمالیزه کردن پس از هر عمل صورت می‌گیرد تا از نرمالیزه شدن نتایج اطمینان حاصل شود.

ضرب و تقسیم ممیز شناور نیازی به هم ردیف کردن مانیس‌ها ندارند. حاصلضرب با ضرب دو مانیس و جمع نهادها شکل می‌گیرد. عمل تقسیم از تقسیم دو مانیس و تفریق نهادها بدست می‌آید. اعمال اجرا شده با مانیس‌ها مشابه اعداد ممیز ثابت است، بنابراین هر دو نوع می‌توانند از ثبات‌ها و مدارهای مشترکی استفاده کنند. عملیات مربوط به نما مقایسه و افزایش (برای هم ردیفی مانیس‌ها)، جمع و تفریق (برای ضرب و تقسیم)، و کاهش (برای نرمالیزه کردن نتیجه) می‌باشد. نما را می‌توان به یکی از سه روش ممکن نمایش داد: مقدار علامت دار، متمم ۲ علامت دار و متمم ۱ علامت دار. چهارمین روشی که در بسیاری از کامپیوتراها بکار گرفته می‌شود نمای خورانده شده یا بایاس است. در این نمایش، بیت علامت بعنوان یک عضو جدا در نظر گرفته نمی‌شود. بایاس عدد مثبتی است که بهنگام تشکیل عدد ممیز شناور به هر نما اضافه می‌شود. در نتیجه تمام نهادها در درون مثبت خواهند بود. مثال زیر می‌تواند این نوع نمایش را روشن سازد. فرض کنید نهادها از ۵۰-تا ۴۹+ متغیر باشند. دروناً نما پس از جمع با بایاس با دو رقم نشان داده می‌شود. (بدون علامت). ثبات نما حاوی عدد  $e+50$  خواهد بود، که عنای واقعی است. لذا نهادها در ثبات بصورت اعداد مثبتی از ۹۹ تا ۰۰ نمایش داده خواهند شد. نهادهای مثبت در ثبات‌ها، محدوده اعداد ۹۹ تا ۵۰ را دارند. تفریق ۵۰ از آنها مقادیر مثبت ۰ تا ۴۹ را می‌دهد. نهادهای منفی دو ثبات‌ها محدوده ۴۹ تا ۰۰ را دارند. تفریق ۵۰ مقدار منفی ۱- تا ۵۰- را می‌دهد.

مزیت استفاده از نهادهای بایاس شده این است که فقط اعداد مثبت را شامل می‌شوند. در نتیجه مقایسه نسبت اندازه‌های آنها ساده‌تر بوده و لازم نیست به علامت آنها توجه کنیم. لذا هنگام هم ردیف کردن مانیس‌ها می‌توان از مقایسه گر مقدار (اندازه) برای مقایسه اندازه‌های آنها استفاده کرد. مزیت دیگر آنها این است که کوچکترین نمای پایین ممکن، تمام ۰ است. در این صورت نمایش ممیز شناور ۰

1- Underflow

مشکل از ماتیس ۰ و کوچکترین نمای ممکن خواهد بود.

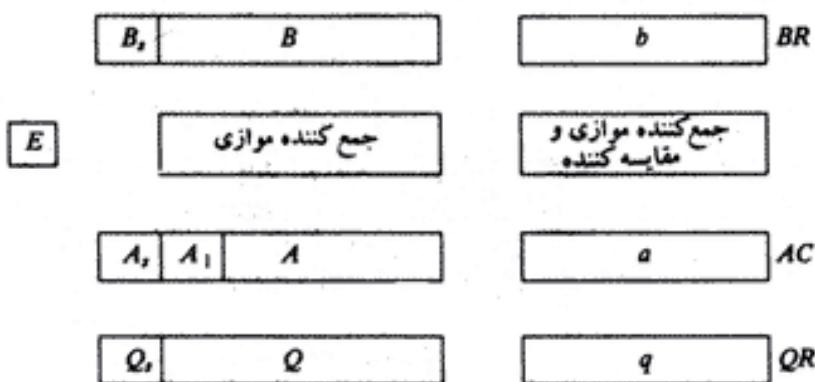
در مثال‌های بالا، از عده‌های دهدی برای نشان دادن برخی از مفاهیمی که هنگام کار با اعداد ممیز شناور باید آن‌ها را دانست استفاده کردیم. واضح است که مفاهیم مشابهی در مورد اعداد دودویی نیز صادق است. الگوریتم‌های حاصل در این بخش برای اعداد دودویی است. در مورد حساب دهدی در کامپیوتر در بخش بعدی صحبت خواهیم کرد.

### آرایش ثبات‌ها

آرایش ثبات‌ها برای اعمال ممیز شناور کاملاً مشابه اعمال ممیز ثابت است. قاعده کلی این است که همان ثبات‌ها و جمع کننده برای پردازش ماتیس بکار می‌روند. اختلاف موجود بر سر کار با نماهاست. سازمان ثبات‌ها برای اعمال ممیز شناور در شکل ۱۰-۱۴ نشان داده شده است. در این شکل سه ثبات QR، AC و BR وجود دارد. هر ثبات به دو زیر بخش تقسیم شده است. بخش ماتیس همان سمبول‌های حروف بزرگ نمایش ممیز ثابت را دارد. بخش نما از سمبول حروف کوچک متضطر استفاده می‌نماید.

فرض براین است که هر عدد ممیز شناور دارای یک ماتیس با نمایش مقدار علامت‌دار و یک نمای بایاس شده باشد. بنابراین AC دارای ماتیسی است که علامتش در  $a$  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 ترمالیزه شدن این عدد بیت مذکور باید ۱ باشد. وقت کنید که سمبول AC نماینده کل ثبات است، یعنی زنجیره  $a$  و  $A$  را

بطور مشابه، ثبات BR به  $b$ ،  $B$  و  $Q$  به  $Q$  و  $q$  تقسیم می‌شوند. یک جمع کننده موازی دوماتیس را جمع می‌کند و مجموع را به  $E$  و نقلی را به  $A$  انتقال می‌دهد. جمع کننده موازی جداگانه‌ای برای نمایها بکار رفته است. چون نمایها بایاس شده‌اند، علامت متمایزی ندارند ولی بصورت یک کمیت مثبت بایاس شده نشان داده شده‌اند. فرض براین است که اعداد ممیز شناور آنقدر بزرگ باشند که شانس



شکل ۱۰-۱۴ ثبات‌های مربوط به عملیات حسابی ممیز-شناور

وقوع سرریز بسیار کم باشد، و به این دلیل سرریز نما چشم پوشی می‌گردد. نمایا به یک مقایسه‌گر مقدار نیز متصل‌اند 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 ببایند 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.

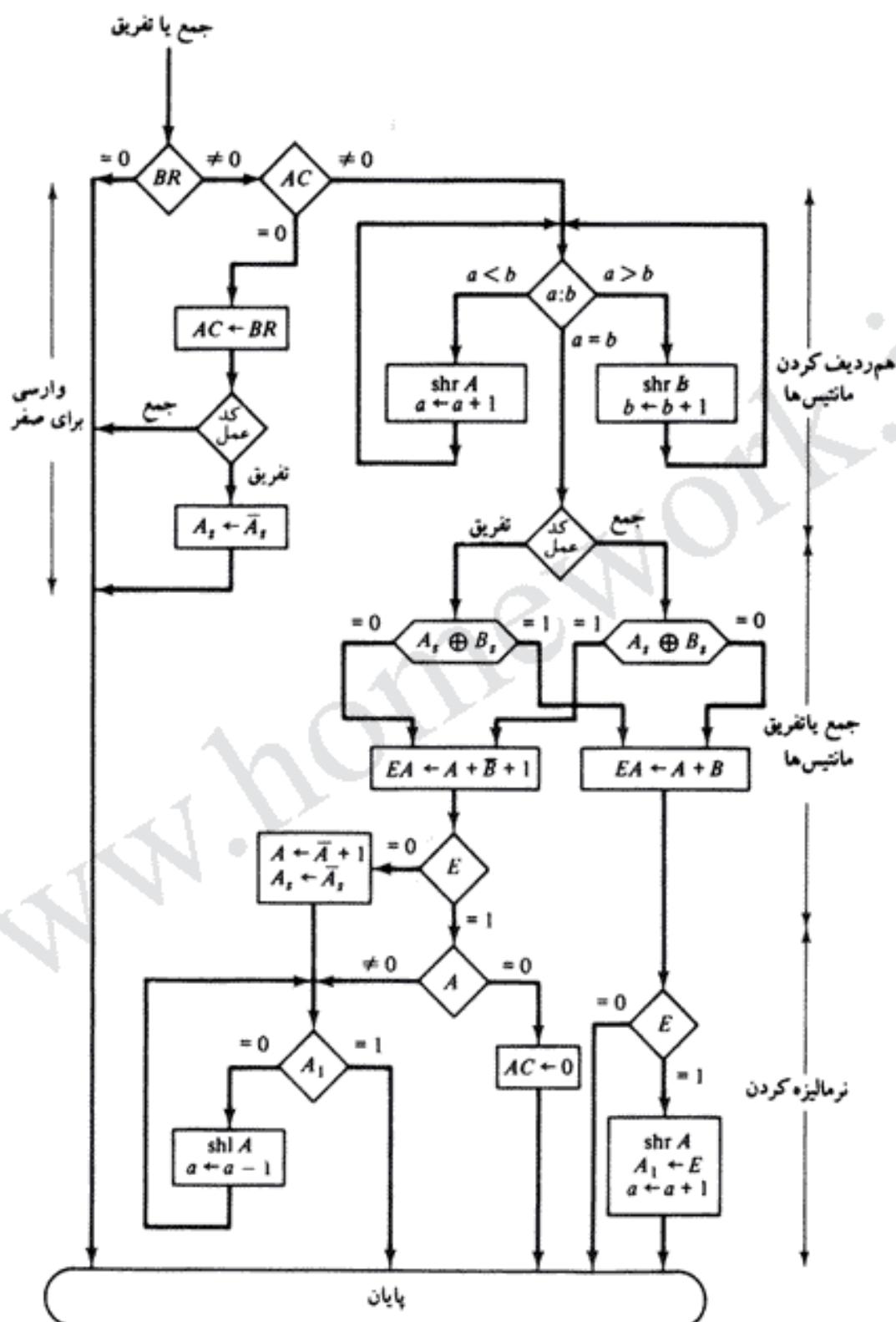
- ۱- تست صفر بودن عملوندها
- ۲- هم ردیف کردن مانتیس‌ها
- ۳- جمع یا تفریق مانتیس‌ها
- ۴- نرمالیزه کردن نتیجه

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 غیرنرمالیزه باشد. با بکارگیری رویه نرمالیزه کردن، نتیجه قبل از انتقال به حافظه نرمالیزه می‌شود.

فلوجارت جمع یا تفریق دو عدد دودویی ممیز شناور در شکل ۱۰-۱۵ 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 متنم می‌کنیم. اگر هیچیک از اعداد صفر نباشند اقدام به هم ردیف کردن مانتیس‌ها می‌نماییم.

مقایسه‌گر مقدار که به نمایهای  $a$  و  $b$  مربوط است سه خروجی تولید می‌کند تا نسبت اندازه‌ها را مشخص نماید. اگر دو نمای مساوی باشند، اجرای عملیات حسابی صورت می‌گیرد. اگر نمایها مساوی نباشند مانتیسی که نمای کوچکتری دارد به راست شیفت داده شده و نمای آن افزایش می‌یابد. این روند تا مساوی شدن نمایها ادامه می‌یابد.

جمع یا تفریق مانتیس‌ها شبیه الگوریتم جمع و تفریق اعداد ممیز ثابت ارائه شده در شکل ۱۰-۲ 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 معین شناور

مانتیس‌ها سرریز رخ دهد، به فلیپ فلاپ E منتقل می‌شود. اگر E برابر 1 باشد، بیت به A<sub>1</sub> 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 تغییر 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<sub>1</sub> برابر 0 باشد مانتیس دارای فروریز است. در این حال، مانتیس به چپ شیفت یافته و نما کاهش می‌یابد. بیت A<sub>1</sub> مجدداً چک شده و روند تا 1 شدن A<sub>1</sub> ادامه می‌یابد. وقتی که A<sub>1</sub>=1 باشد، مانتیس نرمالیزه شده و عملیات خاتمه یافته است.

### ضرب

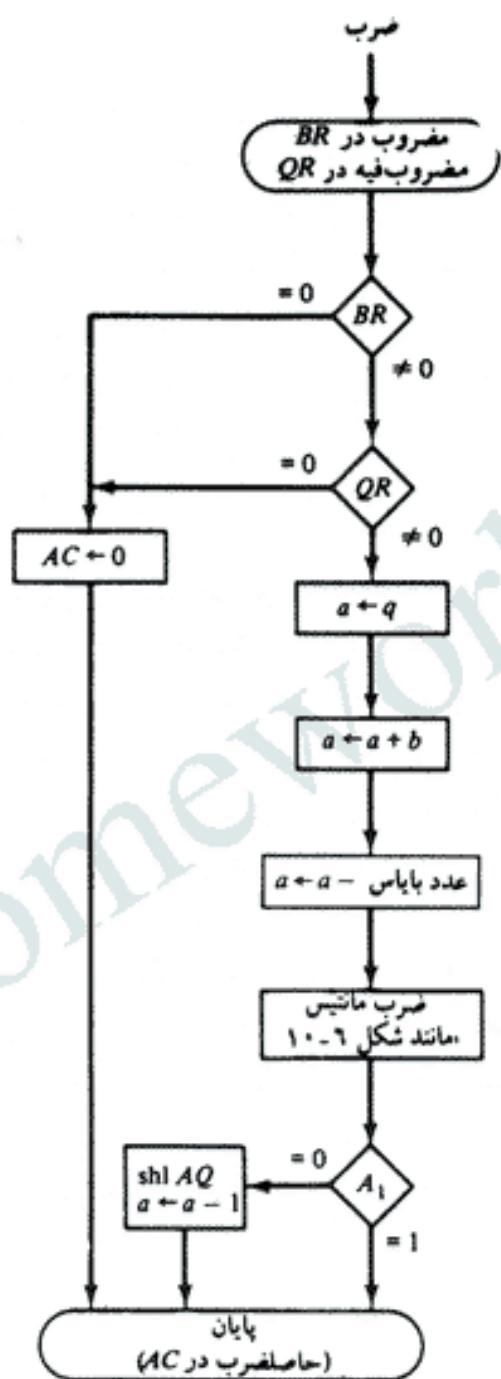
ضرب دو عدد ممیز شناور با ضرب مانتیس‌ها و جمع نمایها انجام می‌شود. در این عمل مقایسه نمایها و هم ردیف کردن مانتیس‌ها لزومی ندارد. ضرب مانتیس‌ها مشابه ضرب اعداد ممیز ثابت است و حاصلضرب با دقتی دو برابر حاصل می‌شود. پاسخ حاصل با دقت مضاعف<sup>۱</sup> در ممیز ثابت برای افزایش دقت حاصلضرب بکار می‌رود. در نمایش ممیز شناور، محدوده دقت معمولی<sup>۲</sup> همراه با نما به اندازه کافی دقت دارد و بنابراین اعداد بصورت غیرمضاعف یا معمولی نگهداری می‌شوند. لذا نیمه با ارزشتر حاصلضرب مانتیس‌ها و نمایها 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 نمایها
- ۳- ضرب مانتیس‌ها
- ۴- نرمالیزه کردن حاصلضرب

مراحل ۲ و ۳ می‌توانند بطور همزمان انجام شوند مشروط بر این که برای مانتیس‌ها و نمایها، جمع کننده‌های 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 نمایها ادامه می‌یابد.



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  $a$  منتقال 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 تفربیق عدد بایاس از مجموع بدست می‌آید.

ضرب ماتنیس‌ها 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 معکن است فروریز داشته باشد، بنابراین با ارزشترین بیت A چک می شود. اگر این بیت 1 باشد ضرب قبل نرمالیزه شده است. اگر 0 باشد ماتیس موجود در  $AQ$  به چپ شبیت پیدا کرده و نما کاهش داده می شود. توجه کنید که تنها یکبار شبیت برای نرمالیزه کردن کافی است. کوچکترین عملوند نرمالیزه کردن 0.1 است، لذا کوچکترین حاصلضرب معکن 0.01 می باشد. بنابراین، تنها یک صفر معکن است در منتها آیه سمت چپ وجود داشته باشد.

هرچند نیمه کم ارزشتر ماتیس در Q است، ولی ما آنرا برای ضرب ممیز شناور بکار نمی بریم و تنها مقدار داخل AC بعنوان حاصلضرب در نظر گرفته می شود.

### تقسیم

در تقسیم ممیز شناور نماها تفرق و ماتیس‌ها تقسیم می‌شوند. تقسیم ماتیس مشابه ممیز ثابت است جز اینکه مقسم ماتیسی با دقت معمولی دارد و در AC قرار داده می‌شود. بخاطر بسپارید که ماتیس مقسم کسری است و نه عدد صحیح. برای نمایش، مقسمی 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، مقسم 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 مقسم بزرگتر یا مساوی مقسم علیه باشد، کسر مقسم به راست شبیت 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 مقسم<sup>۱</sup> خوانده می‌شود.

تقسیم دو عدد ممیز شناور نرمالیزه شده همواره خارج قسمت نرمالیزه شده ای را بدست می‌دهد بشرطی که هم ردیف کردن مقسم 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 دهنی اولیه به ثباتها و تعیین علامت

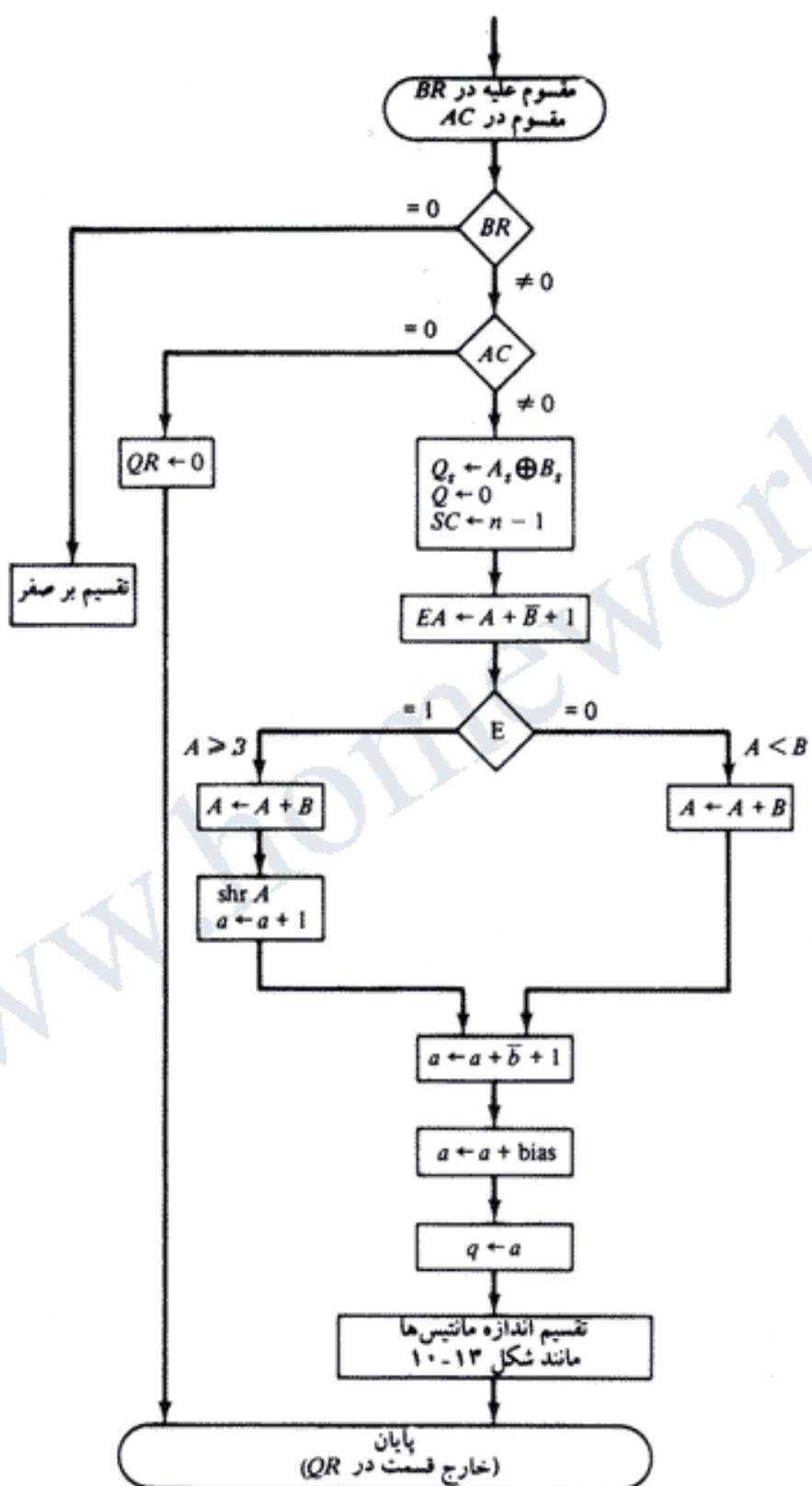
۳- هم ردیف کردن مقسم

۴- تفرق نمادها

۵- تقسیم ماتیس‌ها

فلوچارت برای تقسیم ممیز شناور در شکل ۱۷-۱۰ نشان داده شده است. دو عملوند برای وجود

1- Divident Alignment



شکل ۱۰-۱۷ تقسیم اعداد ممیز شناور

صفر چک شده‌اند. اگر مقسوم علیه صفر باشد نشانده‌نده اقدام به تقسیم بر صفر است، که عملی غیرمجاز است. عملیات با اعلام خطای 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  $A \geq C$  صفر باشد خارج قسمت موجود در QR صفر می‌شود و عمل خاتمه می‌باید. اگر عملونده‌ها صفر نباشند، در مرحله بعدی علامت خارج قسمت را تعیین و آن را در  $Q_0$  ذخیره می‌کنیم. علامت مقسوم در  $A_0$  بلا 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 سریز 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  $B < A$  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 مانیس‌ها همچون در حالت ممیز ثابت تقسیم می‌شوند. پس از عمل، خارج قسمت مانیس‌ها در  $Q$  و باقیمانده در  $A$  قرار خواهد گرفت. خارج قسمت ممیز شناور قبل نرمالیزه شده و در QR قرار دارد. نمای باقیمانده باید مانند نمای مقسوم باشد. ممیز دودویی برای مانیس 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 تعریف واگذار می‌شود.

## ۱۰-۶ واحد حساب دهدی

کاربر یک کامپیوتر داده‌ها را بصورت اعداد دهدی فراهم نموده و نتایج را هم بصورت دهدی دریافت می‌نماید. یک CPU با یک واحد حساب و منطق می‌تواند ریزاعمال را با اعداد دودویی انجام دهد. برای اجرای عملیات حسابی با داده‌های دهدی، لازم است داده‌های دهدی ورودی به اعداد دودویی تبدیل شوند، تا محاسبات با اعداد دودویی انجام شده، و نتایج به دهدی تبدیل شوند. این روش ممکن است در کاربردهایی که محاسبات زیادی دارند ولی مقدار داده‌های ورودی و خروجی آنها نسبتاً کم است، کارآ باشد. در مواردی که حجم داده‌های ورودی و خروجی زیاد است ولی محاسبات نسبتاً کم می‌باشد، بهتر است محاسبات درونی را مستقیماً با اعداد دهدی انجام دهیم. کامپیوترهایی که

قادرند محاسبات دهدی را اجرا نمایند باید داده های دهدی را به فرم کدهای دودویی ذخیره کنند. اعداد دهدی سپس به واحد محاسبات دهدی که قادر است ریز عمل های حساب دهدی را اجرا نماید اعمال می شود.

ماشین های حساب الکترونیکی بلاستثناء دارای واحد محاسبات دهدی داخلی می باشند زیرا حجم ورودی و خروجی آنها زیاد است. به نظر نمی رسد 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 داده های دودویی و دهدی دارای سخت افزار خاصی هستند. کاربرها بکمک دستورات برنامه ریزی شده می توانند خواسته خود، مبنی بر محاسبات دودویی یا دهدی توسط کامپیوتر را مشخص نمایند.

یک واحد محاسبات دهدی مدار دیجیتالی است که ریز عملیات دهدی را انجام می دهد. این مدار می تواند اعداد دهدی را جمع یا تفریق نماید، که این عمل اخیر معمولاً بکمک متمم 9 و یا 10 مفروض انجام می شود؛ مدار اعداد کد شده به دهدی را دریافت و نتایج را با همان کددودویی تولید می نماید. یک واحد محاسبات دهدی یک مرحله ای، 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. خروجی ها شامل چهار پایانه 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 نمی کند، 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 چهار بیتی اعمال کنیم. جمع کننده حاصل جمع را بصورت دودویی تشکیل می دهد و نتیجه حاصل بین 0 تا 19 خواهد بود. این اعداد دودویی در جدول ۴-۱۰ لیست شده و با سمبول های K و Z<sub>8</sub>، Z<sub>4</sub>، Z<sub>2</sub> و Z<sub>1</sub> نام گذاری شده اند. K، رقم نقلی و اندیس های زیر حروف Z نشان دهنده وزن های 8 و 4 و 2 و 1 بوده و به چهار بیت در کد BCD اختصاص دارند. اولین ستون در جدول حاصل جمع های دودویی را آنچنانکه در خروجی های یک جمع کننده دودویی چهار بیتی ظاهر می شوند نشان می دهد. خروجی حاصل جمع دو عدد دهدی باستی بصورت BCD نشان داده شود. مسئله در اینجا یافتن قاعده ساده ای است که بوسیله آن عدد دودویی در ستون اول به رقم BCD صحیحی که در ستون دوم نمایش داده شده تبدیل شود.

جدول ۱۰-۴ بدست آوردن جمع کننده BCD

مجموع دودویی					BCD					مجموع		
K	Z <sub>0</sub>	Z <sub>4</sub>	Z <sub>2</sub>	Z <sub>1</sub>	C	S <sub>0</sub>	S <sub>4</sub>	S <sub>2</sub>	S <sub>1</sub>	ددهی		
0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1	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		
<hr/>												
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	0	1	0	0	0	14		
0	1	1	1	1	1	0	1	0	0	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 می‌کند.

یکی از روش‌های جمع اعداد ددهی 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 می‌نماید. اگر نتیجه مساوی یا بزرگتر از 1010 باشد، با جمع 0110 با حاصل جمع اصلاح می‌شود. این عمل دوم بطور خودکار یک رقم نقلی را برای جفت رقم با ارزش بعدی تولید می‌کند. جفت ارقام بالارزشتر بعدی، همراه با نقلی ورودی باهم جمع شده و مجموع مربوطه را تولید می‌نمایند. اگر این نتیجه مساوی یا بزرگتر از 1010 می‌باشد، با جمع آن با 0110 تصحیح می‌گردد. رویه تا جایی که کلیه ارقام جمع شوند، ادامه می‌یابد.

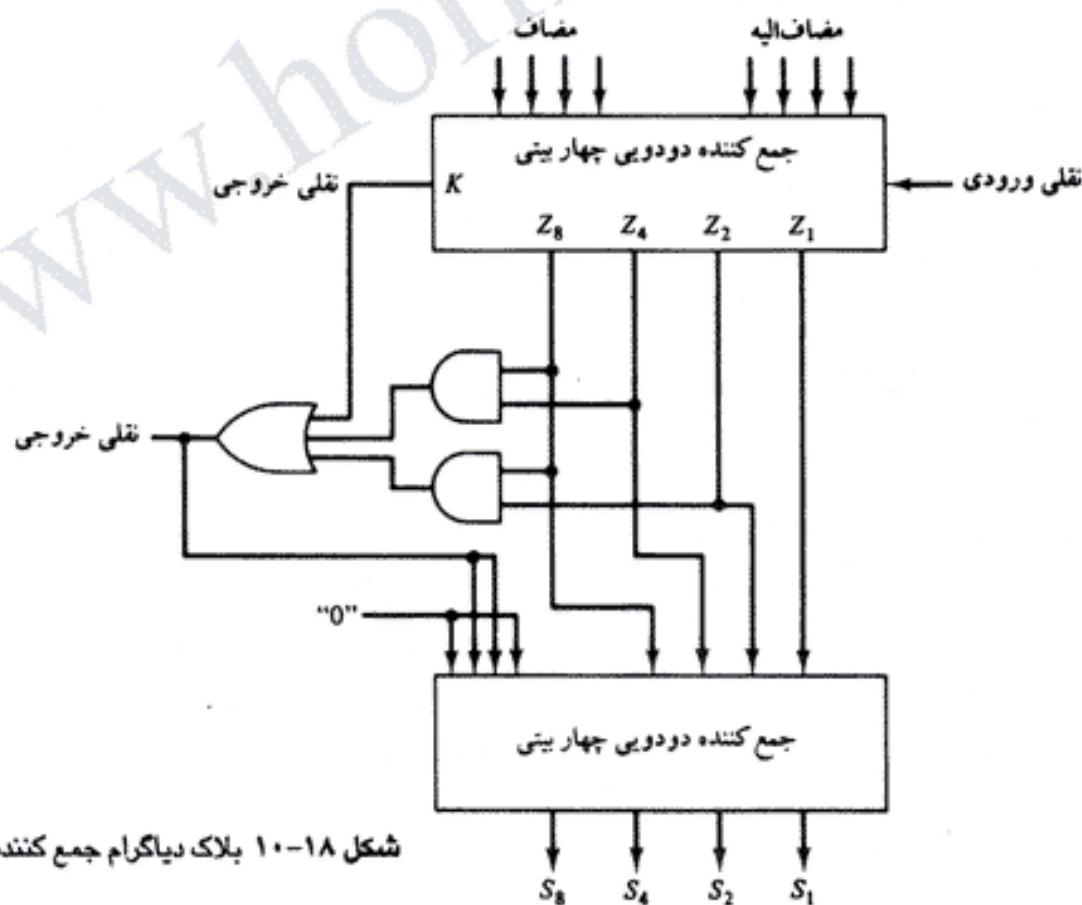
مداری که که ضرورت تصحیح را مشخص می‌کند از واردہای جدول بدست می‌آید. واضح است

که هرگاه مجموع دو دویی دارای نقلی خروجی مساوی ۱ باشد تصحیح لازم است. شش ترکیب دیگر از ۱۰۱۰ تا ۱۱۱۱ که اصلاح نیاز دارند دارای ۱ در مکان  $Z_8$  هستند،  $Z_4$  یا  $Z_2$  هم باید ۱ 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 دو دویی مقدار ۰۱۱۰ 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 ۰۱۱۰ با مجموع دو دویی، از یک جمع کننده دو دویی چهار بیتی دومی که در شکل ۱۰-۱۸ نشان داده شده است استفاده می کنیم. دو رقم دهدھی، همراه با رقم نقلی ورودی، ابتدا در جمع کننده فوقانی با هم جمع شده و مجموع دو دویی را تولید می کنند. مادامی که رقم نقلی خروجی برابر ۰ باشد، چیزی به مجموع دو دویی اضافه نمی شود. اگر رقم نقلی خروجی ۱ باشد، عدد دو دویی ۰۱۱۰ به مجموع دو دویی از طریق جمع کننده تحتانی اضافه می شود. رقم نقلی خروجی تولید شده بوسیله جمع کننده دو دویی تحتانی می تواند چشم پوشی شود، زیرا اطلاعاتی را که قبل از در خروجی رقم



شکل ۱۰-۱۸ بلک دیاگرام جمع کننده BCD

نقلی موجود است تولید نموده است.

یک جمع کننده موازی دهدی که  $n$  رقم دهدی را جمع می کند 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 کننده های BCD حاوی مدارات لازم برای تولید نقلی پیش بینی شده می باشند. بعلاوه، مدار جمع کننده برای اجرای عمل تصحیح نیازی به هر چهار تمام جمع کننده ندارد و لذا مدار می تواند بهینه شود.

### تفریق BCD

تفریق مستقیم دو عدد دهدی به یک مدار تفریق گر که از برخی لحاظ با جمع کننده BCD متفاوت است نیاز دارد. اگر تفریق را بكمک متمم 9 یا 10 مفروق و جمع آن با مفروق منه انجام دهیم اقتضادی تر خواهد بود. چون BCD یک کد خود متمم<sup>1</sup> نیست، متمم 9 آن را نمی توان با متمم کردن هر بیت در کد بدست آورد. این متمم را با تشکیل مداری که هر رقم BCD را از 9 تفریق می کند باید بدست آورد. متمم 9 یک رقم دهدی که بصورت BCD نشان داده شده باشد را می توان با متمم کردن بیتها در نمایش کد شده رقم بدست آورده بشرط اینکه تصحیحی صورت گیرد. دو روش تصحیح وجود دارد. در روش اول، عدد دودویی 1010 (10 دهدی) به هر رقم متمم اضافه شده و رقم نقلی بعداز هر جمع چشم پوشی می شود. در روش دوم، عدد دودویی 0110 (6 دهدی) قبل از متمم شدن رقم اضافه می شود. به عنوان یک مثال عددی، برای بدست آوردن متمم 9 عدد 0111 (7 دهدی) BCD بدین ترتیب محاسبه می شود که ابتدا هر بیت متمم می شود تا 1000 بدست آید. با جمع 1010 با آن و چشم پوشی از رقم نقلی عدد 0010 (2 دهدی) را خواهیم داشت. با استفاده از روش دوم، ما عدد 0110 را به 0111 اضافه می کنیم تا 1101 بدست آید. با متمم کردن هر بیت نتیجه مورد نظر 0010 بدست می آید. متمم کردن عدد چهار بیتی N مشابه تفریق آن از 1111 (15 دهدی) می باشد. با اضافه کردن معادل دودویی عدد 10 به آن داریم  $9-N+10=9-N+16-15$ . از طرفی 16 بمعنى رقم نقلی بوده و نادیده گرفته می شود، بنابراین نتیجه  $N-9$  خواهد شد. با افزودن معادل دودویی عدد دهدی 16 و متمم کردن آن نیز داریم  $9-N=(N+6)-15$ .

متمم 9 عدد BCD از طریق یک مدار ترکیبی نیز بدست می آید. وقتی که این مدار به یک جمع کننده BCD متصل گردد، یک جمع یا تفریق کننده BCD بدست می آید. اجازه بدید تا رقم مفروق (یا مضاف) با چهارمتغیر دودویی  $B_8, B_4, B_2$  و  $B_1$  مشخص شده باشد. همچنین M بیت مد (شیوه) باشد که جمع یا تفریق را کنترل می کند. هنگامی که  $M=0$  باشد، دو رقم جمع می شوند؛ اگر  $M=1$  باشد، ارقام از یکدیگر تفریق می گردند. همچنین، فرض کنید متغیرهای دودویی  $x_8, x_4, x_2$  و  $x_1$  خروجی های مدار متمم ساز (متمم 9) باشند. با بررسی جدول درستی این مدار مشاهده می شود (مسئله ۱۰-۳۰) که همواره باید متمم شود؛  $B_2$  همیشه در متمم 9 مثل رقم اصلی است؛  $x_4$  برابر 1 است اگر  $B_1$

1- Self Complement

انحصاری  $B_2$  و  $B_4$  برابر 1 باشند؛  $x_8 = B_8 B_4 B_2 = 000$  هم، اگر 0 باشد،  $x_1 = 1$  می‌باشد. توابع بولی تولیدکننده متمم 9 عبارتند از:

$$x_1 = B_1 M' + B'_1 M$$

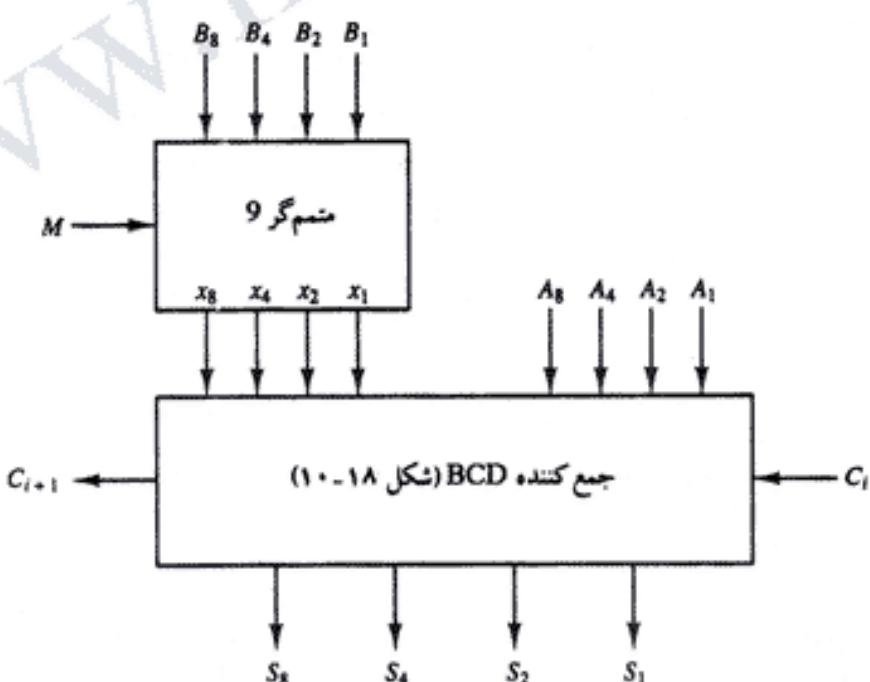
$$x_2 = B_2$$

$$x_4 = B_4 M' + (B'_4 B_2 + B_4 B'_2) M$$

$$x_8 = B_8 M' + B'_8 B'_4 M$$

باقطه به معادلات فوق در می‌باییم که وقتی  $M=0$  است،  $x=B$  می‌باشد. هنگامی که  $M=1$  باشد خروجی‌های  $X$  متمم  $B$  را تولید می‌کنند.

یک طبقه از مدار محاسبات دهدہی که قادر است دو رقم و  $BCD$  را جمع و یا تفریق کند در شکل ۱۰-۱۹ نشان داده شده است. این مدار مشکل از یک جمع کننده را  $BCD$  یک متمم سازاست. ورودی  $M$  کار مدار را کنترل می‌کند. اگر  $M=0$  باشد خروجی‌های  $S$  مجموع  $A$  و  $B$  را تولید می‌کنند. هنگامی که  $M=1$  باشد خروجی‌های  $S$  مجموع  $A$  بعلاوه متمم 9 عدد  $B$  را تولید می‌کنند. برای اعداد  $n$  رقمی به 1 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 دهدہی این است که  $M=1$  باشد و ورودی  $C_1$  در اولین طبقه را هم به 1 متصل کنیم. خروجی‌ها مجموع  $A$  بعلاوه متمم 10 عدد  $B$  را تولید می‌کنند، 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 دهدہی

## ۱۰-۷ اعمال حسابی دهدھی

الگوریتم های اعمال حسابی با داده های مشابه الگوریتم های همان اعمال با داده های دودویی است. در واقع بجز کمی تصحیح در الگوریتم ضرب و تقسیم، فلوچارت های یکسانی را برای دونوع داده می توان بکاربرد با این شرط که سمبول ریز عمل ها را بطور صحیح تفسیر کنیم. اعداد دهدھی با نمایش گروه های چهاربیتی در ثبات های کامپیووتر ذخیره می شوند. هر گروه چهاربیتی یک رقم دهدھی را نمایش می دهد و باید هنگام اجرای ریز عمل های دهدھی بصورت واحد در نظر گرفته شود.

برای سهولت، ما سمبول های یکسانی را برای ریز عمل حسابی دودویی و دهدھی بکار خواهیم برد ولی بصورت متفاوتی آنها را تعبیر خواهیم کرد. همان طور که در جدول ۱۰-۵ دیده می شود، یک خط روی سمبول حرف بمعنی متمم ۹ یک عدد دهدھی ذخیره شده در ثبات است. جمع ۱ با متمم ۹ متمم ۱۰ را تولید می کند. بنابراین برای اعداد دهدھی، سمبول  $A \leftarrow A + \bar{B} + 1$  بمعنی انتقال مجموع دهدھی تشکیل شده از جمع مقدار اولیه  $A$  با متمم ۱۰ عدد  $B$  است. استفاده از سمبول های یکسان برای متمم ۹ و متمم ۱، اگر هر دو نوع داده در یک سیستم بکار رود موجب سردرگمی می گردد. در این حالت بهتر خواهد بود سمبول های متفاوتی برای متمم ۹ بکار برد 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 دهدھی، بجز تعداد حالات مجاز برای ثبات یکسان است. افزایش یک شمارنده دودویی موجب طی کردن ۱۶ حالت از ۰۰۰۰ تا ۱۱۱۱ می گردد. یک شمارنده دهدھی ۱۰ حالت از ۰۰۰۰ تا ۱۰۰۱ را می شمارد و دوباره به ۰۰۰۰ باز می گردد، زیرا عدد ۹ آخرین عدد ممکن در شمارش دهدھی است. بطور مشابه، کاهش یک شمارنده دودویی رشته ۱۱۱۱ تا ۰۰۰۰ را طی می کند. شمارنده دهدھی در این فرم از ۱۰۰۱ تا ۰۰۰۰ را می شمارد.

در جلو شیفت به راست یا چپ دهدھی حرف  $A$  بکار رفته است تا مشخص کننده شیفت بر روی چهاربیتی باشد که ارقام دهدھی را دارا هستند. بعنوان مثال عددی، ثبات  $A$  که عدد دهدھی ۷۸۶۰ را بشكل BCD در خود دارد در نظر بگیرید. الگوی ۱۲ فلیپ فلاپ بقرار زیر است.

0111 1000 0110 0000

جدول ۱۰-۵ سمبول های ریز عمل های حسابی دهدھی

نمایش سبیلک	شرح
$A \leftarrow A + B$	جمع عدد های دهدھی و انتقال مجموع به $A$
$\bar{B}$	متمم ۹ از $B$
$A \leftarrow A + \bar{B} + 1$	محتواي $A$ بعلاوه متمم ۱۰ عدد $B$ در $A$ قرار گیرد
$Q_L \leftarrow Q_L + 1$	افزایش عدد BCD موجود در $Q_L$
$dshr A$	شیفت دهدھی ثبات $A$ به راست
$dshi A$	شیفت دهدھی ثبات $A$ به چپ

ریز عمل  $dshr$  عدد دهدھی را یک رقم به راست شیفت می دهد و لذا ۰۷۸۶ بدست می آید. این شیفت روی دسته های ۴ 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 دهدھی با نمایش مقدار علامت دار نیز صادق است با این شرط که سمبول ها را بطور صحیحی تعبیر کنیم. بطور مشابه الگوریتم اعداد دودویی با نمایش متمم ۲ علامت دار برای اعداد دهدھی متمم ۱۰ علامت دار نیز صادق است. برای داده های دودویی از یک جمع کننده دودویی و یک متمم ساز استفاده می شود. برای داده های دهدھی از واحد محاسباتی که قادر به جمع اعداد BCD بوده و بتواند متمم ۹ مفروق را طبق شکل ۱۰-۱۹ تشکیل دهد استفاده می شود.

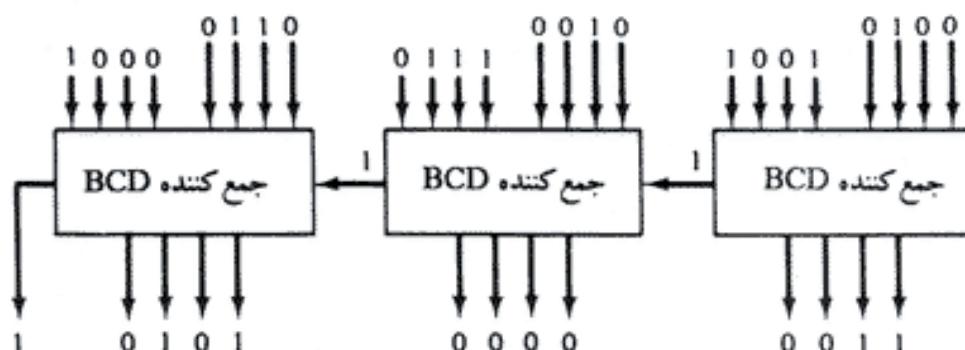
داده های دهدھی به سه طریق مختلف باهم جمع می شوند، شکل ۱۰-۲۰. روش موازی از یک واحد محاسبه دهدھی مت Shankl از جمع کننده های BCD بتعدد ارقام عدد استفاده می کند. مجموع بطور موازی تشکیل و فقط به یک ریز عمل نیاز دارد. در روش رقم- سری، بیت- موازی<sup>۱</sup>، ارقام فقط به یک جمع کننده BCD بصورت سری اعمال می شوند، در حالیکه بیت های هر کد بصورت موازی انتقال می یابند. حاصل جمع با شیفت اعداد دهدھی به جمع کننده BCD، یکی پس از دیگری تشکیل می گردد. برای  $k$  رقم دهدھی، این آرایش به  $k$  ریز عمل نیاز دارد، که هر یک متعلق به یک شیفت دهدھی است. در تمام جمع کننده های سری، بیت ها یکی پس از دیگری پداخ ت 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 ۱۰-۶ در مورد آن بحث شد، مجموع دودویی وارسی می شود. اگر این مجموع بزرگتر یا مساوی ۱۰۱۰ باشد، جمع آن با ۰۱۱۰ آن را تصحیح می کند و یک رقم نقلی را نیز برای جفت رقم بعدی تولید می نماید.

روش موازی سریع است ولی به تعادل 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.

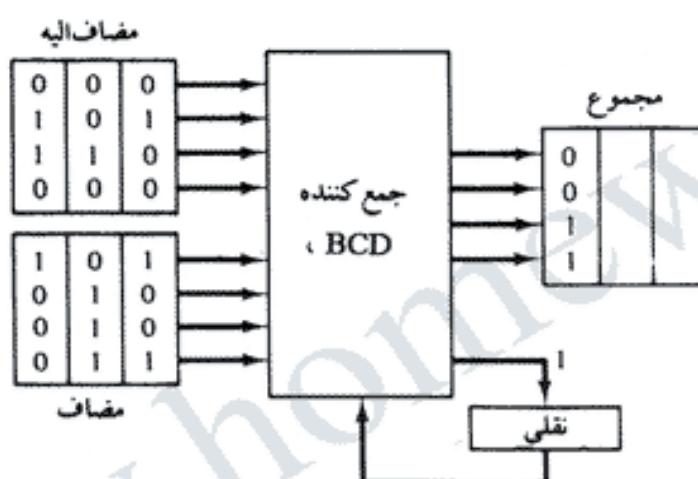
### ضرب

ضرب اعداد دهدھی ممیز ثابت، بجز در تولید حاصل ضرب جزیی، با روش دودویی مشابه است. در

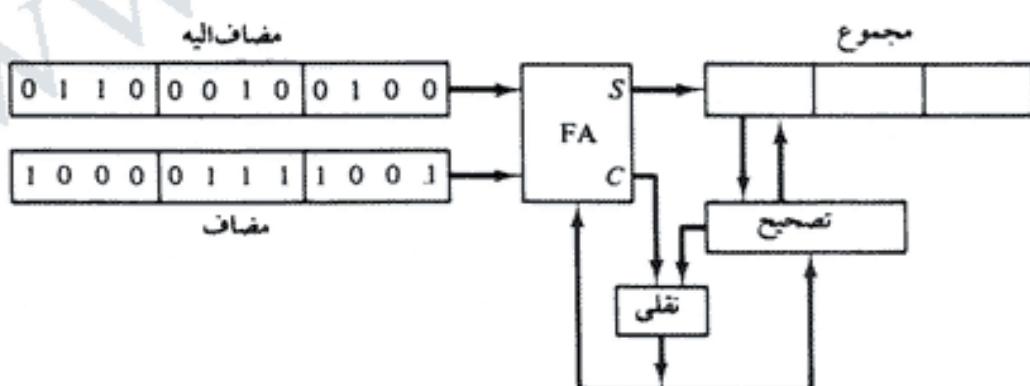
1- Digit-Serial Bit-Parallel



$$(الف) جمع دهدی موازی \quad 624+879=1503$$



$$(ب) جمع دهدی رقم سری، بیت - موازی$$



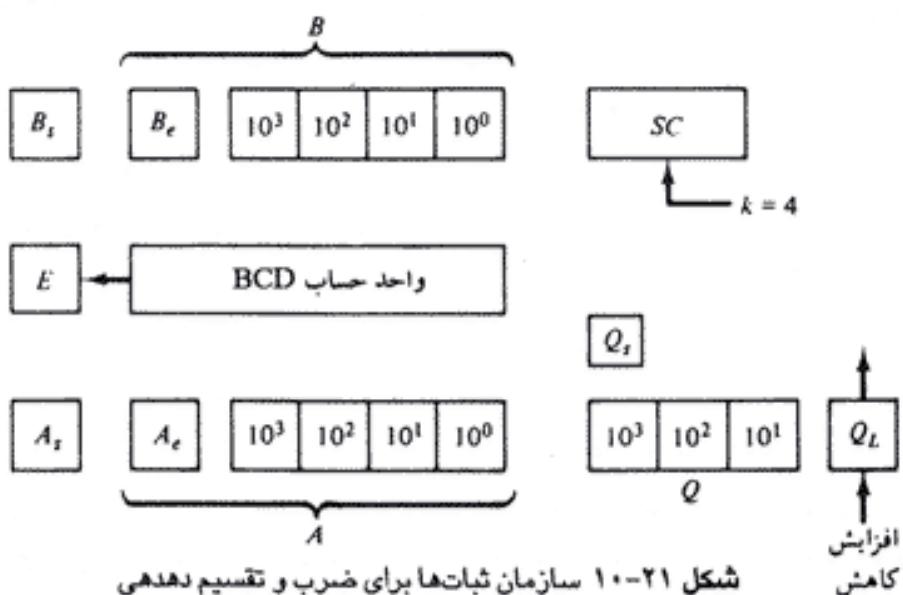
$$(ج) جمع دهدی تمام سری$$

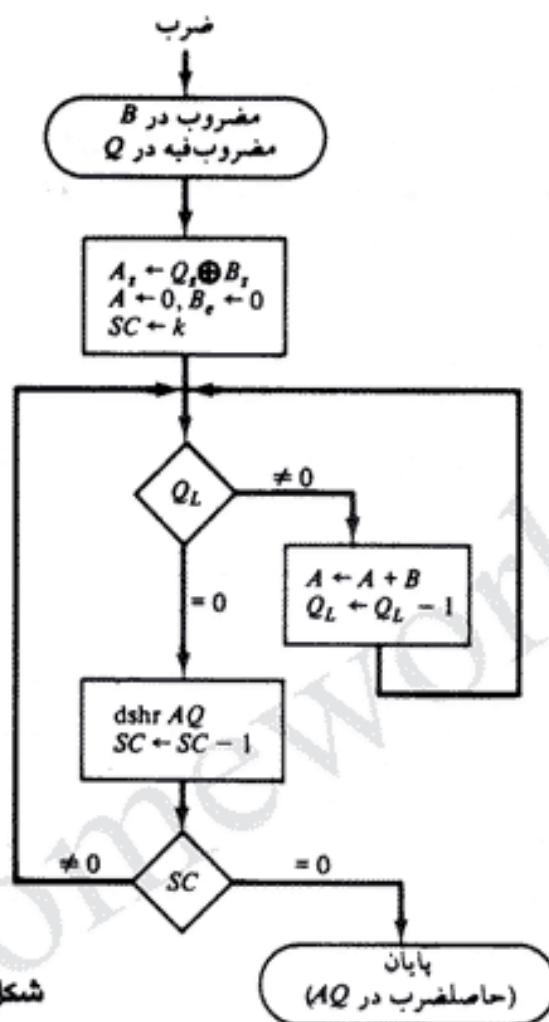
شکل ۱۰-۲۰ سه روش جمع اعداد ددهی

ضرب دهدہی ارقام در محدوده ۰ تا ۹ قرار دارند، در صورتی که در ضرب دودویی فقط دو رقم ۰ و ۱ وجود دارد، در حالت دودویی، مضروب، بشرطی که مضروب فیه ۱ باشد، با حاصلضرب جزیی جمع می شود. در حالت دهدہی مضروب باید در رقم مضروب فیه ضرب شده و سپس به حاصلضرب جزیی اضافه شود. این عمل را می توان با جمع مضروب با حاصلضرب جزیی به تعداد دفعاتی برابر با مقدار رقم مضروب فیه انجام داد.

سازمان ثبات ها برای ضرب دهدہی در شکل ۱۰-۲۱ 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 ۱۶ بیتی خواهد بود. سه ثبات A و B و Q وجود دارند که هر یک فلیپ فلاپ علامت مربوط به خود یعنی  $A_s, B_s, Q_s$  را دارا هستند. ثبات های A و B دارای چهار بیت اضافی دیگرند که با  $A_e, B_e$  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 سوریز احتمالی حاصل از جمع مضروب به حاصل ضرب جزیی در حین عمل ضرب است. هدف از  $B_e$  تشکیل متمم ۹ مقسوم علیه هنگام تغیر آن از باقیمانده جزیی 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 دهدہی برداشته شده از حافظه ۱۷ بیتی است. یک بیت (علامت) به  $B_s$  منتقل می شود و اندازه عملوند در ۱۶ بیت پائین تر B قرار می گیرد. در آغاز هر دو بیت  $A_e$  و  $B_e$  پاک می شوند. نتیجه عمل نیز ۱۷ بیتی است و در آن از بخش  $A_e$  ثبات A استفاده نمی شود.

الگوریتم ضرب دهدہی در شکل ۱۰-۲۲ 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  $Q_L$  وارسی می شود. اگر برابر ۰ نباشد، مضروب در B با حاصلضرب جزیی در A یکبار جمع شده و





شکل ۱۰-۲۲ فلوچارت ضرب دهدی

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 فيه با حاصلضرب جزیی جمع می شود. هر رقم سریز موقت در  $A$  قرار گرفته و می تواند بین ۰ تا ۹ باشد.

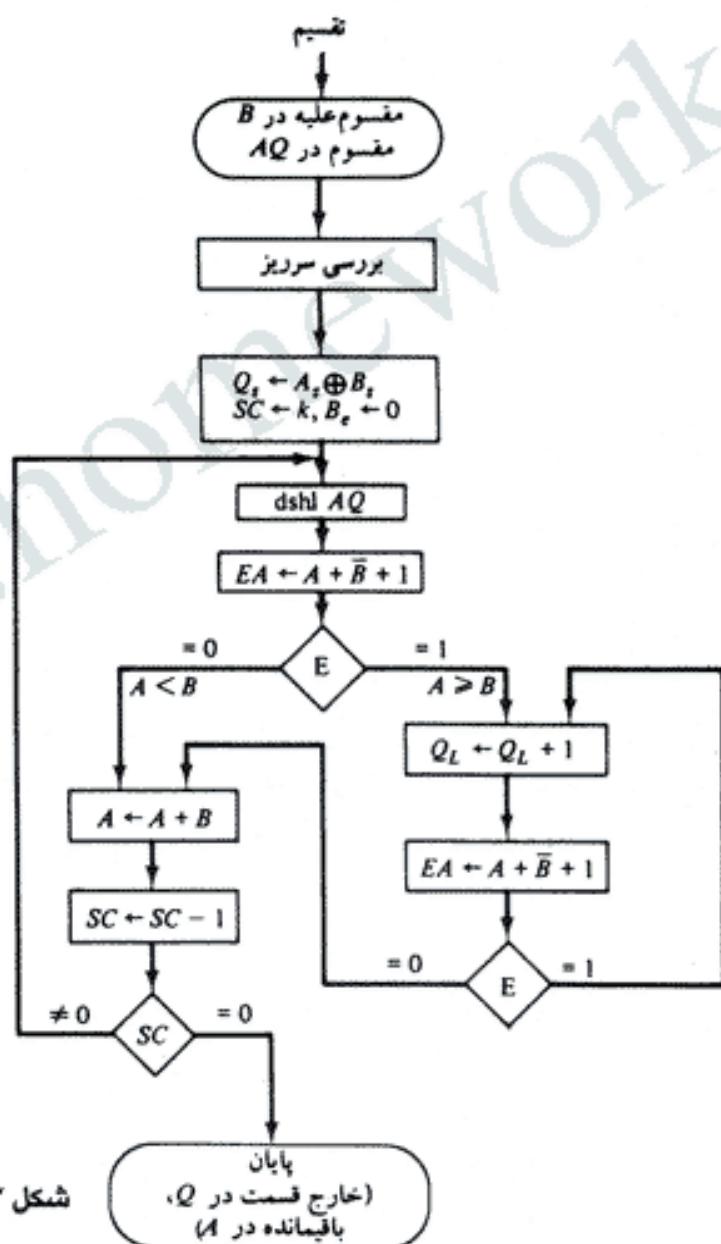
سپس حاصلضرب جزیی و مضروب فيه یکباره به راست شیفت داده می شود. این عمل یک ۰ را در قرار می دهد و رقم بعدی مضروب فيه به  $Q_L$  منتقل می شود. نهایتاً روند  $k$  مرتبه تکرار می شود تا حاصلضرب با طول دو برابر در  $AQ$  تشکیل شود.

### تقسیم

تقسیم دهدی مشابه تقسیم دودویی است با این تفاوت که ارقام خارج قسمت ممکن است هر یک مقادیر ۰ تا ۹ را داشته باشند. در روش تقسیم بازیافتنی، مقسوم علیه به تعداد دفعات لازم از مقسوم با باقیمانده جزیی 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 دهدۀ در شکل ۱۰-۲۳ نشان داده شده است. این الگوریتم بجزدر مورد نحوه تشکیل بیت‌های خارج قسمت مشابه الگوریتم تقسیم داده‌های دودویی است. مقسوم (یا باقیمانده جزیی) به چپ شیفت می‌یابد و با ارزشترین رقم در آن قرار می‌گیرد. سپس متمم ۱۰ مقسوم علیه با مقسوم جمع شده و بدیت ترتیب عمل تفریق مقسوم علیه از مقسوم انجام می‌گیرد. چون در ابتدا  $B_E = 0$  باشد، پاک شده است، متمم آن ۹ است. رقم نقلی در  $E$  نسبت اندازه‌های  $A$  و  $B$  را معین می‌کند. اگر  $E=0$  باشد،  $Q_L$  نشان دهنده  $B < A$  است. در این حالت مقسوم علیه جمع می‌شود تا باقیمانده جزیی بازیابی شود و  $Q_L$



در ۰ باقی می‌ماند (در حین شیفت صفر وارد شده است). اگر  $E=1$  باشد نشانه  $A \geq B$  است. رقم خارج قسمت موجود در  $Q_L$  یکبار افزایش یافته و مقسوم علیه مجدداً تفیریق می‌شود. این روند تا رسیدن به تفاصل که با  $E=0$  قابل تشخیص است ادامه می‌یابد. هرگاه این وضعیت رخ دهد. رقم خارج قسمت افزایش نیافته ولی مقسوم علیه جمع می‌شود تا باقیمانده مثبت بددست آید، در این وضعیت رقم خارج قسمت برابر با تعداد دفعاتی است که باقیمانده جزئی می‌تواند مقسوم علیه را در خود جای دهد. بیت‌های باقیمانده جزئی و خارج قسمت یکبار به چپ شیفت داده می‌شوند و روند  $k$  بار تکرار می‌شود تا  $k$  رقم خارج قسمت تشکیل گردد. سپس باقیمانده در ثبات  $A$  و خارج قسمت در ثبات  $Q$  خواهد بود. مقدار  $E$  صرفنظر می‌شود.

### اعمال ممیز شناور

اعمال حسابی دهدی ممیز شناور روند مشابهی با اعمال دودویی دارند. الگوریتم‌های بخش ۱۰-۵ می‌توانند برای داده‌های دهدی بکار رود بشرطی که سمبول‌های ریز اعمال بطور صحیحی تعبیر شوند. ضرب تقسیم مانعیس‌ها باید با استفاده از روش‌های فوق صورت گیرد.

### مسائل

۱۰-۱ اگر در شکل ۱۰-۱ بجای  $A + B$  عمل  $\bar{A} + \bar{B} + 1$  (متتم ۱ عدد  $A$  بعلاوه  $B$ ) را انجام دهیم متتم ساز لازم نیست. الگوریتمی به شکل فلوچارت برای جمع و تفیریق اعداد دودویی ممیز ثابت با نمایش اندازه علامت دار بددست آورید که در آن اندازه‌ها با دو ریز عمل  $\bar{A} \leftarrow A$  و  $\bar{B} \leftarrow B$  تفیریق شوند.

۱۰-۲ هر یک از مسیرهای فلوچارت شکل ۱۰-۲ را شماره گذاری کنید و سپس کل مسیری که الگوریتم هنگام محاسبه اعداد با اندازه علامت دار زیر طی می‌کند را مشخص نمایید. در هر مورد مقدار AVF را بنویسید. بیت منتها لیه سمت چپ در عده‌های زیر نماینده بیت علامت است.

- |                     |      |
|---------------------|------|
| 0 101101 + 0 011111 | الف) |
| 1 011111 + 1 101101 | ب)   |
| 0 101101 - 0 011111 | ج)   |
| 0 101101 - 0 101101 | د)   |
| 1 011111 - 0 101101 | ه)   |

۱۰-۳ اعمال حسابی زیر را با اعداد دودویی و اعداد منفی در نمایش متتم ۲ علامت دار انجام دهید. برای جای دادن هر عدد همراه با علامتش از ۷ بیت استفاده کنید. در هر حالت، با وارسی نقلی

ورودی و خروجی از بیت علامت وجود سرریز را وارسی کنید.

(الف)  $(+35) + (+40)$

(ب)  $(-35) + (-40)$

(ج)  $(-35) - (+40)$

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 (با مقدار). عددی منفی مانند  $x$ - بصورت  $2^k + (2^k - x)$  نمایش داده می شود که در آن اولین  $k$  نشان دهنده بیت علامت و  $(2^k - x)$  متمم 2 عدد  $X$  است. با عدد مثبت بصورت  $x+0$  نمایش داده می شود، که 0 بیت علامت و  $X$ ، مقدار  $k$  بیشتر آن است. با استفاده از این سمبول های کلی ، ثابت کنید که مجموع  $(Y) + (\pm X)$  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 اشکار 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 وارسی دونقلی آخر در این مورد به نتیجه منجر نمی شود.  
ب) رویه اصلاح شده ای را برای آشکار سازی یک سرریز وقته که اعداد متمم 1 استفاده می شوند پیشنهاد کنید.

۱۰-۷ الگوریتمی بفرم فلوچارت برای جمع و تفریق دو عدد دودویی ممیز ثابت وقته که اعداد منفی بصورت متمم 1 هستند بدست آورید.

۱۰-۸ ثابت کنید ضرب دو عدد  $n$  رقمی در پایه ۲ حاصلضربی بزرگتر از  $2^n$  رقم تولید نمی کند. نشان دهید که این عبارت بدأن 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) (ب) 10111 بر 101000111؛ (ب) 00001111 بر 10111 (مشابه ۱۰-۱) نشان دهید. (از مقسم هشت بیتی استفاده کنید).

۱۰-۱۱ نشان دهید که جمع  $B$  پس از عمل  $A + \bar{B}$  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؟

۱۰-۱۳ یک ضرب کننده آرایه‌ای طراحی کنید که دو عدد چهاربیتی را ضرب کند. از گیت AND برای جمع کننده دودویی استفاده کنید.

۱۰-۱۴ روند قدم به قدم ضرب با بکارگیری الگوریتم بوت را (مثل جدول ۱۰-۳) هنگامی که اعداد دودویی زیر در هم ضرب می‌شوند نشان دهید. فرض کنید ثبات‌های پنج بیتی اعداد علامت دار رانگه می‌دارند. مضروب در هر دو حالت ۱۵+ است.

$$\text{الف) } (+15) \times (+13)$$

$$\text{ب) } (+15) \times (-13)$$

۱۰-۱۵ الگوریتم را بفرم فلوچارت برای روش غیر بازیافتی تقسیم دودویی ممیز ثابت بدست آورید.

۱۰-۱۶ الگوریتم برای محاسبه ریشه دوم عدد دودویی ممیز ثابت بدست آورید.

۱۰-۱۷ یک عدد دودویی ممیز شناور هفت بیت برای نمای بایاس شده‌اش دارد. مقدار ثابت بایاس 64 است.

الف) لیست بایاس شده تمام نماها را از ۶۴- تا ۶۳+ بدست آورید.

ب) نشان دهید که یک مقایسه گر مقدار هفت بیت می‌تواند برای مقایسه اندازه نسبی دو نما بکار رود.

ج) نشان دهید که پس از جمع دو نمای بایاس شده لازم است ۶۴ را تفریق کنیم تا مجموع نماها به شکل بایاس شده بدست آید. چگونه می‌توان ۶۴ را با جمع کردن مقدار متمم ۲ آن تفریق کرد؟

د) نشان دهید که پس از تفریق دو نمای بایاس شده لازم است ۶۴ را اضافه کنیم تا تفاضل بایاس شده نماها را داشته باشیم

۱۰-۱۸ الگوریتم بشکل فلوچارت برای مقایسه دو عدد دودویی علامت دار وقتی که اعداد منفی به فرم متمم ۲ علامت دار می‌باشند بدست آورید

الف) با استفاده از عمل تفریق با عده‌های متمم ۲ علامت دار

ب) با مرور و مقایسه جفت بیت‌ها از چپ به راست

۱۰-۱۹ مسئله ۱۰-۱۸ وا برای اعداد دودویی مقدار علامت دار تکرار کنید.

۱۰-۲۰ فرض کنید  $\alpha$ ، تعداد بیت‌های مانتیس در یک عدد دودویی ممیز شناور است. هنگامی که

مانتیس‌ها در حین جمع یا تفریق هم ردیف می‌شوند، اختلاف نماها ممکن است بزرگتر از  $1 - n$  باشد.

اگر این حالت اتفاق بیفتد ماتریس با نمای کوچکتر کلاً از ثبات شیفت داده می‌شود. هم

ردیف کردن مانتیس رادر شکل ۱۰-۱۵ با اضافه کردن یک توالی شماری که تعداد شیفت‌ها را

می‌شمارد اصلاح کنید. اگر تعداد شیفت‌ها بزرگتر از  $1 - n$  باشد، بزرگترین عدد برای تعیین نتیجه

بکار خواهد رفت.

۱۰-۲۱ رویه هم ردیف کردن مانتیس‌ها در جمع یا تفریق اعداد ممیز شناور بشرح زیر می‌تواند بیان شود: نمای کوچکتر را از بزرگتر تفریق کنید و مانتیسی را که دارای نمای کوچکتر است به تعداد

تفاضل نماها به راست شیفت دهد. نمای مجموع (یا تفاضل) برابر با نمای بزرگتر است. بدون استفاده از مقایسه کننده مقدارها، با فرض نماهای بایاس شده، و در نظر گرفتن اینکه فقط AC را می‌توان جابجا کرد، الگوریتم را به شکل فلوچارت برای هم ردیف کردن ماتیس‌ها و قرار دادن نمای بزرگتر در AC بدست آورید.

۱۰-۲۲ نشان دهد که سرریز ماتیس بعداز عمل ضرب ممکن نیست.

۱۰-۲۳ نشان دهد که تقسیم دو عدد نرمالیزه ممیز شناور با ماتیس کسری همواره خارج قسمت نرمالیزه خواهد داد مشروط بر اینکه قبل از عمل تقسیم، هم ردیف کردن مقسم انجام شود.

۱۰-۲۴ فلوچارت شکل ۱۰-۱۷ را برای تهیه یک باقیمانده نرمالیزه ممیز شناور در AC تعمیم دهد. ماتیس باید یک کسر باشد.

۱۰-۲۵ در الگوریتم های اعمال حسابی ممیز شناور در بخش ۱۰-۵ امکان سرریز یا فروریز در نما نادیده گرفته می‌شود.

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 می‌دهد که نما از کوچکترین عددی که بتواند در ثبات جای گیرد کمتر باشد.

ج) نشان دهد که سرریز یا فروریز نما بوسیله سخت افزار می‌تواند آشکار شود.

۱۰-۲۶ اگر نمایش عدد صحیح را برای ماتیس اعداد ممیز شناور تصور کنیم، در حین ضرب و تقسیم با مشکلات خاصی رو برو می‌شویم. اجازه بدهید تعداد بیت‌های بخش مقدار ماتیس  $(n-1)$  باشد. برای نمایش صحیح:

الف) نشان دهد که اگر حاصل ضرب با دقت معمولی بکار رود، باید  $(n-1)$  به نمای حاصل ضرب در AC اضافه شود.

ب) نشان دهد که اگر از مقسم با ماتیس دقت معمولی استفاده شود، باید هنگامی که  $Q$  برابر صفر می‌شود  $(n-1)$  از نمای مقسم تغیریق شود.

۱۰-۲۷ سخت افزاری، که برای جمع و تغیریق دو عدد دهدی در نمایش اندازه علامت دار بکار می‌رود را نشان دهد. چگونگی آشکارسازی سرریز را نشان دهد.

۱۰-۲۸ نشان دهد که اگر ۳۵۶-۳۵۷ با جمع ۶۷۳ با متمم ۱۰ عدد ۳۵۶ و چشم پوشی از نقلی نهایی انجام می‌شود. بلاک دیاگرام سه مرحله از واحد حسابی دهدی را رسم کنید و نشان دهد چگونه این عمل پیاده سازی می‌شود. تمام بیت‌های ورودی و خروجی واحد حساب را لیست کنید.

۱۰-۲۹ نشان دهد که به جای جمع کننده دو دویی چهاربیتی پائین رتبه در شکل ۱۰-۱ می‌توان یک تمام جمع کننده و دو نیم جمع کننده قرار داد

۱۰-۳۰ با استفاده از تکینک های طراحی مدارهای ترکیبی، توابع بولی متمم ساز ۹ اعداد BCD شکل ۱۰-۱۹ را بدست آورید. دیاگرام منطقی را رسم کنید.

۱۰-۳۱ لازم است تا برای دو رقم دهدۀ در کد افزونی ۳ (جدول ۳-۶) یک جمع کننده طراحی شود.

نشان دهید که تصحیح پس از جمع دو رقم با جمع کننده چهاربیتی بصورت زیراست.

(الف) رقم نقلی خروجی برابر نقلی تصحیح نشده است

(ب) اگر نقلی خروجی برای ۱ باشد، ۰۰۱۱ اضافه می شود

(ج) اگر نقلی خروجی برابر ۰ باشد، ۱۱۰۱ اضافه می شود و نقلی حاصل از این جمع نادیده گرفته می شود. نشان دهید که جمع کننده افزونی ۳ می تواند با هفت تمام جمع کننده و دو معکوس کننده ساخته شود.

۱۰-۳۲ مداری را برای یک متمم ساز ۹ هنگامی که ارقام دهدۀ به فرم کد افزونی ۳ نمایش داده می شوند، بدست آورید یک ورودی کنترل مد<sup>۱</sup> 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 دهدۀ با نمایش متمم ۱۰ علامت دار اعداد منفی را نشان دهید. نشان دهید که چگونه سرریز آشکار می شود. الگوریتم بشکل فلوچارت تهیه کنید و خودرا از صحت تولید پاسخ صحیح بوسیله آن مطمئن سازید.

۱۰-۳۴ محتوای ثبات‌های A و B و Q و SC را در حین ضرب دهدۀ (شکل ۱۰-۲۲)، (الف) ۴70×152 و (ب) 999×199 بدست آورید. ثبات‌ها را به رقمی تصور کنید و دو میں عدد را مضروب فیه فرض نمائید.

۱۰-۳۵ محتوای ثبات‌های A و E و Q و SC را در حین تقسیم دهدۀ (شکل ۱۰-۲۳) ۱680/32 نشان دهید. ثبات‌ها را دو رقمی فرض کنید

۱۰-۳۶ نشان دهید که زیر ثبات A در شکل ۱۰-۲۱ در پایان (الف) ضرب دهدۀ چنانکه در شکل ۱۰-۲۲ مشخص شده است، و (ب) در تقسیم شکل ۱۰-۲۳، صفر است.

۱۰-۳۷ الگوریتم‌های حساب ممیز شناور در بخش‌های ۱۰-۵ را از داده‌های دودویی به داده‌های دهدۀ تغییر دهید. در یک جدول نحوه تفسیر هر یک از نمادهای ریز عمل‌ها را بنویسید.

## 11

**سازمان ورودی - خروجی**

- |   |  |
|---|--|
| ۱۱-۵ وقفه اولویت دار<br>۱۱-۶ دستیابی مستقیم به حافظه (DMA)<br>۱۱-۷ پردازنده ورودی و خروجی IOP<br>۱۱-۸ تبادل اطلاعات سری | ۱۱-۱ وسائل جانبی<br>۱۱-۲ واسطه ورودی - خروجی<br>۱۱-۳ انتقال غیرهمزمان داره<br>۱۱-۴ شیوه‌های انتقال |
|---|--|

**۱۱-۱ وسائل جانبی<sup>۱</sup>**

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 بی معنی 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 فوق العاده 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 مدامی 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 آنجامی تواند 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<sup>۱</sup> 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<sup>۲</sup> و چاپگرهای خروجی. 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 می‌نماییم.

مونیتورهای تصویری<sup>۳</sup> متدائل‌ترین وسایلهای 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<sup>۴</sup> (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 لتأثر 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 جارویی آن در امتداد عرض صفحه نمایش می‌شوند و لذا الگوی تصویری موردنظر بر روی صفحه ظاهر می‌گردد. یکی از ویژگیهای وسایل نمایش، مکان‌نمایی<sup>۵</sup> است که محل قرارگرفتن 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<sup>۶</sup> براساس محل مکان‌نما اطلاعات را اضافه یا حذف می‌نمایند. پایانه نمایش<sup>۷</sup> می‌تواند در روش تکنیسی<sup>۸</sup> 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<sup>۹</sup>، متن ویرایش شده ابتدا در یک حافظه محلی در داخل پایانه ذخیره می‌شود. این متن بعداً بصورت بلاکی از داده به کامپیوتر منتقل می‌گردد.

چاپگرهای کاراکتری وجود دارند: چرخ‌گردان<sup>۱۰</sup>، ماتریسی<sup>۱۱</sup> و لیزری. چاپگر چرخ گردان دارای چرخی

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 مناسب قرار گرفته و سپس یک آهن ریای الکتریکی حرف موردنظر را روی نوار فشار می‌دهد. چاپگر ماتریسی حاوی مجموعه‌ای از نقطه‌ها در امتداد مکانیزم چاپ است. مثلاً یک چاپگر ماتریسی  $5 \times 7$  قادر است 80 کاراکتر را در هر سطر چاپ کند که هر سطر نیز از هفت خط افقی تشکیل شده است. بنابراین هر خط  $400 = 5 \times 80$  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 یکار می‌رود. نحوه دستیابی به اطلاعات سری بوده و هر نوار از رکوردهایی تشکیل شده است و با عبور نوار از مقابله یک مکانیزم خواندن - نوشتن ثابت، می‌توان به رکوردها یکی پس از دیگری دست یافت. این روش یکی از ارزانترین و کنندترین 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 اکثرآ برای ذخیره انبوه برنامه‌ها و داده‌ها استفاده می‌شود. در رابطه با نقش نوارها و دیسک‌ها به عنوان حافظه کمکی در بخش ۱۲-۱ بیشتر بحث خواهیم کرد.

وسایل ورودی و خروجی دیگری که در سیستم‌های کامپیوتری با آنها مواجه می‌شویم پلاترهای دیجیتال (رسم)، کاراکتر خوان نوری و مغناطیسی، مبدل‌های آنالوگ به دیجیتال و انواع تجهیزات دریافت<sup>۱۲</sup> (اکتساب) داده می‌یاشند. از کامپیوترها برای کنترل فرآیندهای مختلف به صورت بلادرنگ<sup>۱۳</sup>، 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 عمده‌تاً 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 می‌نماییم.

## کاراکترهای الفبا عددی آسکی<sup>۱</sup>

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 ۱۱-۱ نشان داده شده است از هفت بیت برای کد کردن ۱۲۸ کاراکتر استفاده می‌کند. هفت بیت کد با  $b_7$  الی  $b_1$  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 ۹۴ کاراکتر چاپ شدنی و ۳۴ کاراکتر غیرچاپ شدنی است که برای مقاصد کنترلی بکار می‌رود. کاراکترهای چاپ شدنی عبارتند از ۲۶ حرف بزرگ الفبا از A تا Z، ۲۶ حرف کوچک الفبا، ۱۰ عدد از ۰ تا ۹ و ۳۲ کاراکتر قابل چاپ خاص مانند %، \* و \\$.

۳۴ کاراکتر کنترل در جدول ASCII با نام‌های خاص بکار رفته‌اند. این کاراکترها مجدداً در قسمت پائین جدول همراه با نام عملیاتی شان آورده شده‌اند. کاراکترهای کنترلی برای مسیردهی داده‌ها و آرایش مسیر 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 پسپرس<sup>۲</sup> (BS)، جدول بندی افقی<sup>۳</sup> (HT)، برگشت نورد<sup>۴</sup> (CR) هستند. جداکننده‌های اطلاعات برای جداسازی اطلاعات به بخش‌هایی مانند پاراگراف و صفحه بکار می‌روند. این دسته شامل کاراکترهایی مانند جداکننده رکوردها<sup>۵</sup> (RS) و جداکننده فایل‌ها<sup>۶</sup> (FS) است. کاراکترهای کنترل ارتباطات هنگام ارسال متن بین پایانه‌های دور از هم مفیدند. مثال‌هایی از این نوع عبارتند از آغاز متن<sup>۷</sup> STX و پایان متن<sup>۸</sup> ETX، که برای کادریندی یک متن پیام هنگام ارسال از طریق یک رسانه ارتباطی بکار می‌رond.

آسکی یک کد هفت بیتی است، ولی در اکثر کامپیوتراها داده‌ها بصورت کمیت هشت بیش بنام بایت دستکاری می‌شوند. بنابراین کاراکترهای ASCII اغلب بصورت یک کاراکتر در هر بایت ذخیره می‌شوند. بیت اضافی بسته به کاربرد گاهی برای مقاصد دیگر بکار می‌رود. مثلاً، بعضی چاپگرهای کاراکترهای آسکی را بصورت هشت بیتی با مقدار ۰ در بیت هشتم تشخیص می‌دهند. از ۱۲۸ کاراکتر هشت بیتی با مقدار ۱ در بیت هشتم برای دیگر سمبول‌ها 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)

$b_4 b_3 b_2 b_1$	000	001	010	011	100	101	110	111	$b_7 b_6 b_5$
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	

کاراکترهای کنترل	خروج از ارتباط داده
DLE	کنترل وسیله ۱
DC1	کنترل وسیله ۲
DC2	کنترل وسیله ۳
DC3	کنترل وسیله ۴
DC4	تصدیق متنی
NAK	مسگامی در حالت بیکاری
SYN	پایان بلاک ارسال
ETB	لغو
CAN	پایان محیط
EM	جاگیری
SUB	خروج
ESC	جداگذره فایل ها
FS	جداگذره گروه ها
GS	جداگذره رکوردها
RS	جداگذره واحدها
US	پاک کننده
DEL	

## ۱۱-۲ واسطه ورودی - خروجی

واسطه ورودی - خروجی روشهای 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

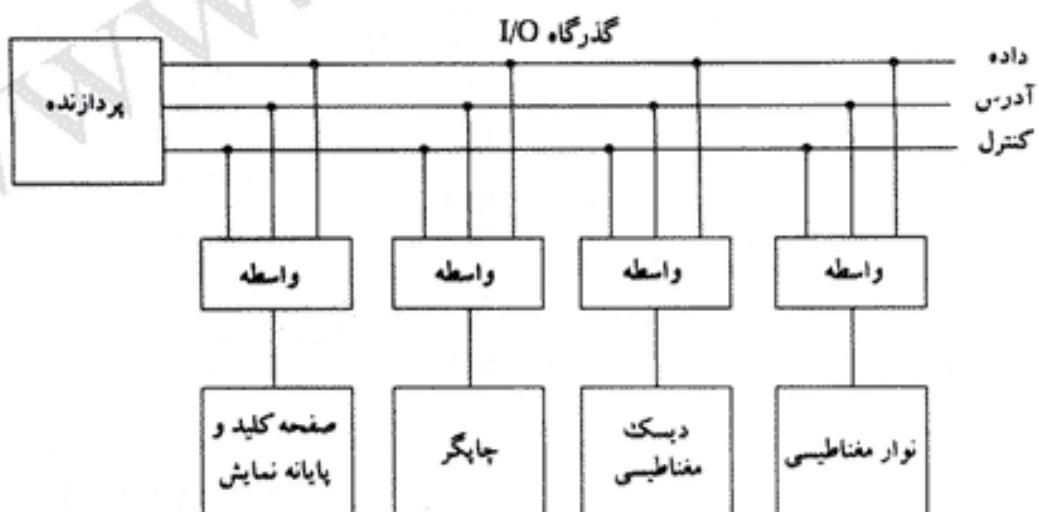
موجود بین کامپیوتر مرکزی و هر وسیله جانبی است. تفاوت های مهم موجود عبارتند از:

- ۱- وسایل جانبی و سایلی 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<sup>۱</sup> 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 مازول های واسطه

نمونه ای از اتصالات ارتباطی بین پردازنده و چند وسیله جانبی در شکل ۱۱-۱ نشان داده شده است. گذرگاه I/O از خطوط داده، خطوط آدرس و خطوط کنترل تشکیل شده است. دیسک مغناطیسی، چاپگر



شکل ۱۱-۱ اتصال گذرگاه I/O به وسیله های ورودی - خروجی

و پایانه عملاً در هر کامپیوتر چند منظوره بکار می‌رond. 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 وسائل جانبی تفسیر، و سیگنال‌های لازم را برای کنترل کننده‌های وسائل 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 دارد 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 وسائل 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 وسائل 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 متولی خاص خود را بسته به نوع عملکرد، تشخیص می‌دهد.

یک فرمان وضعیت برای آزمایش شرایط وضعیت 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 خطأ را تشخیص دهد. این خطأها با نشاندن<sup>۱</sup> (۱ شدن) بیت‌ها در ثبات وضعیت مشخص شده و پردازنده می‌تواند با فواصل زمانی معین 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

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 نوشتند 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 کامپیوتراهای انجام می شود که دارای یک پردازنده I/O (IOP) علاوه بر واحد پردازش مرکزی (CPU) می باشند. حافظه با هردو واحد CPU و IOP از طریق گذرگاه خود ارتباط برقرار می کند. IOP هم با وسائل 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 وسائل خارجی و حافظه درونی است. پردازنده I/O گاهی اوقات کانال داده<sup>۱</sup> هم خوانده می شود. در بخش ۷-۱۱ ماکار IOP را مفصل تر بحث خواهیم کرد.

### I/O ایزوله در مقابل نگاشت یافته در حافظه<sup>۲</sup>

بسیاری از کامپیوتراها 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 نوشتند جداگانه است. CPU با توانا سازی یکی از دو خط خواندن و نوشتند 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 نوشتند I/O بهنگام

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 نوشتan I/O ( برای خروجی ) را فعال می سازد. این عمل قطعات خارجی را که به گذرگاه مشترک متصلند مطلع می سازد که آدرس روی CPU خطوط آدرس متعلق به ثبات واسطه است و نه به یک کلمه حافظه. از طرف دیگر، وقتی که دستورالعمل یا عملوندی را از حافظه برداشت می کند، آدرس حافظه را روی خطوط آدرس قرار داده و خط کنترل خواندن یا نوشتan 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 بواسیله 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 نوشتan استفاده می کنند و تمایزی بین آدرس حافظه و I/O وجود ندارد. این آرایش، I/O نگاشت یافته در حافظه<sup>۱</sup> 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.

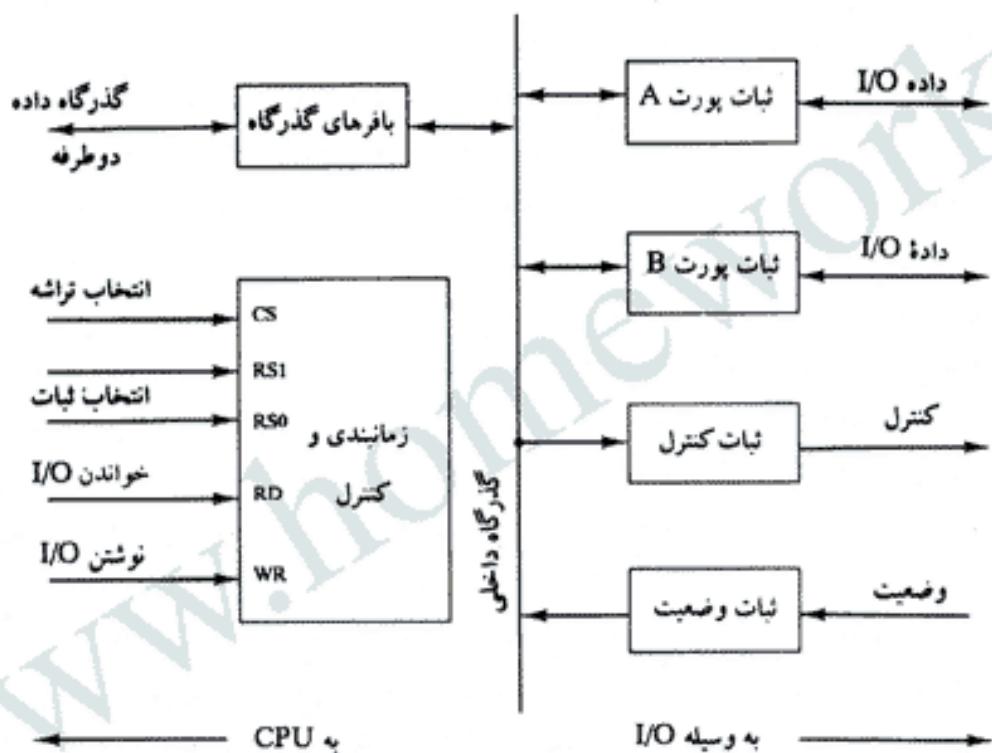
در سازمان I/O نگاشت یافته در حافظه، دستورالعمل خاصی برای ورودی و خروجی وجود ندارد. CPU قادر است داده های I/O واقع در ثبات واسطه را با همان دستوراتی که برای حافظه بکار می روند دستکاری کند. هر واسطه به صورت مجموعه ای از ثبات ها سازماندهی شده است و به درخواست های خواندن و نوشتan در فضای عادی آدرس پاسخ می 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 کند. مزیت این است که دستورات بارگردان<sup>۲</sup> و ذخیره کردن<sup>۳</sup> که برای خواندن و نوشتan 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

بیشتر از دستورات I/O است. با روش I/O نگاشت یافته در حافظه کلیه دستورالعمل‌های حافظه‌ای برای I/O هم قابل استفاده‌اند.

### مثالی از واسطه I/O

مثالی از واحد واسطه I/O در شکل ۱۱-۲ بصورت بلاک دیاگرام نشان داده شده است. این واسطه مشکل از دو ثبات داده بنام پورت<sup>۱</sup> (درگاه)، یک ثبات کنترل، یک ثبات 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
۰ × ×	همچو گذرگاه داده در ایندیکانس بالا
۱ ۰ ۰	ثبات پورت A
۱ ۰ ۱	ثبات پورت B
۱ ۱ ۰	ثبات کنترل
۱ ۱ ۱	ثبات وضعیت

شکل ۱۱-۲ مثالی از واحد واسطه I/O

۱- Port

وروودی های انتخاب تراشه و انتخاب ثبات، آدرس تشخیص یافته به مدار واسطه را معین می نمایند. خواندن و نوشتن O/I دو خط کنترل هستند که بترتیب ورودی یا خروجی را مشخص می کنند. چهار ثبات مستقیماً با وسایل O/I متصل به مدار واسطه ارتباط دارند.

داده I/O به واز وسایل می تواند به پورت A یا B منتقل گردد. مدار واسطه می تواند با یک وسیله ورودی، یا یک وسیله خروجی و یا وسیله ای که هم ورودی و هم خروجی است کار کند. اگر واسطه به یک چاپگر وصل باشد، فقط داده را خارج می کند، و اگر به یک کاراکتر خوان سرویس دهد، فقط داده را وارد می کند. یک واحد دیسک مغناطیسی داده را در هر دو جهت بطور غیرهمزن منتقل می نماید، بنابراین مدار واسطه می تواند از خطوط دو جهته استفاده کند. صدور فرمان به یک وسیله I/O با ارسال یک کلمه به یک ثبات مناسبی صورت می گیرد. در چنین سیستمی، کد تابع در گذرگاه I/O لازم نیست زیرا کنترل به ثبات کنترل ارسال می شود، اطلاعات وضعیت از ثبات وضعیت، و داده ها به و یا از پورت های A و B انتقال می یابند. بنابراین انتقال اطلاعات داده، کنترل و وضعیت همواره از طریق گذرگاه مشترک داده انجام می شود. تمایز بین اطلاعات داده، کنترل یا وضعیت با توجه به ثبات خاص در مدار واسطه ای که CPU با آن ارتباط برقرار می کند صورت می گیرد.

ثبات کنترل، اطلاعات کنترل را از CPU دریافت می نماید. با بارگردان بیت های معینی در ثبات کنترل، مدار واسطه و وسیله I/O متصل به آن می تواند در وضعیت های 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. بیت های ثبات وضعیت برای شرایط وضعیتی و ضبط خطاهایی که در حین انتقال داده ها ممکن است رخ دهد بکار می رود. 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.

ثبات های واسطه از طریق گذرگاه دو جهته داده با CPU ارتباط برقرار می کنند. گذرگاه آدرس واحد واسطه را از طریق انتخاب کننده تراشه و دو ورودی انتخاب ثبات انتخاب می کند. برای تشخیص آدرس اختصاص یافته به ثبات های واسطه، یک مدار خارجی (معمولأً یک دیکندر) بکار گرفته می شود. این مدار ورودی انتخاب تراشه<sup>۱</sup> (CS) را وقتی که واسطه بوسیله گذرگاه آدرس انتخاب شود فعال می سازد. دو ورودی انتخاب ثبات RS1 و RS0 معمولاً به دو خط کم ارزشتر گذرگاه آدرس وصل می شوند. این دو ورودی یکی از چهار ثبات را طبق جدول همراه دیاگرام انتخاب می نمایند. محتوای ثبات انتخاب شده از طریق گذرگاه داده، هنگامی که سیگنال خواندن O/I فعال شود، به CPU انتقال می یابد. اطلاعات دودویی را از طریق گذرگاه داده وقتی که ورودی نوشت O/I فعال شود، به ثبات انتخاب شده منتقل می سازند.

#### 1- Chip Select

### ۱۱-۳ 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 O/I 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<sup>۱</sup> (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<sup>۲</sup> یا ناهمگام با هم خوانیم. این روند بطور گسترده ای در بیشتر سیستم های کامپیوتري بکار گرفته شده است.

انتقال غیرهمزمان بین دو واحد مستقل مستلزم ارسال سیگنال های کنترل بین واحد های 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<sup>۳</sup> (استرووب) بوسیله یکی از واحد ها، 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 دهنی<sup>۴</sup> خوانده می شود.

روش پالس فعال ساز<sup>۵</sup> و دست دهنی در انتقال غیرهمزمان داده ها منحصر به انتقال O/I 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<sup>۶</sup> و واحد دریافت کننده را مقصد<sup>۷</sup> می خوانیم. مثلاً ضمن عمل خروج داده و یا نوشتمن، CPU یک منبع است ولی بهمگام ورود یا خواندن نقش یک مقصد را بعده 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 استرووب

روش کنترل انتقال داده غیرهمزمان بوسیله سیگنال استرووب از یک خط کنترل برای هر انتقال استفاده می کند. استرووب ممکن است بوسیله واحد منبع و یا مقصد فعال گردد. شکل ۱۱-۳ (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

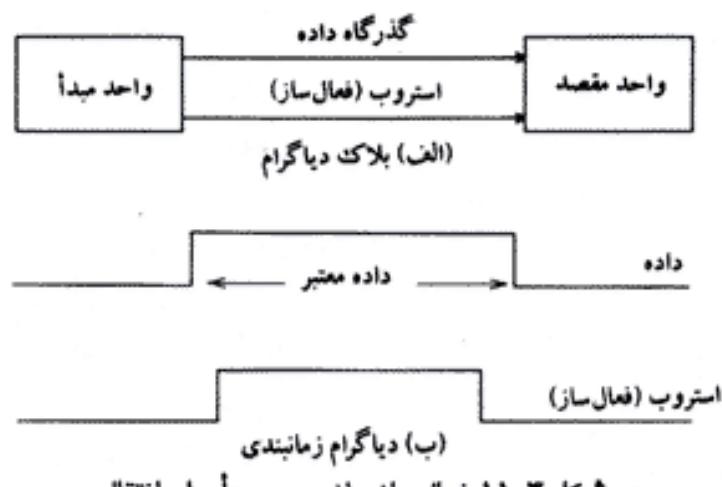
3- Strobe

4- Handshaking

5- Strobe

6- Source

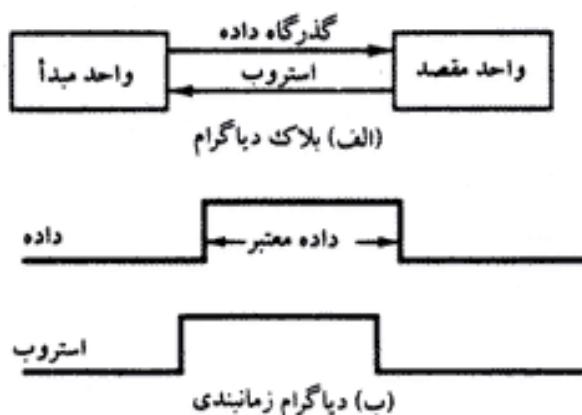
7- Destination



شکل ۱۱-۳ فعالسازی از سوی مبدأ برای انتقال

می‌کند. معمولاً گذرگاه برای انتقال تمام بایت یا کلمه چند خط دارد. استروپ یک خط تک واحدی است که واحد مقصد را بهنگام قرار گرفتن یک داده معتبر بر روی گذرگاه، مطلع می‌سازد. همانطور که در دیاگرام ۱۱-۳ (ب) دیده می‌شود، واحد منبع ابتدا داده‌ها را روی گذرگاه داده قرار می‌دهد. پس از یک تأخیر مختصر که برای اطمینان از تثبیت مقدار داده، در یک مقدار ثابت است منبع استروپ را فعال می‌سازد. اطلاعات روی گذرگاه داده و سیگنال استروپ برای زمانی کافی در حالت فعال باقی می‌مانند تا به واحد مقصد اجازه دریافت داده را بدهند. اغلب واحد مقصد لبه پائین رونده پالس استروپ را برای انتقال محتوای گذرگاه داده به یکی از ثبات‌های درونی بکار می‌برد. واحد منبع به فاصله کوتاهی پس از غیرفعال کردن پالس استروپ، داده را از گذرگاه برمی‌دارد. در واقع، منبع نیازی به تغییر داده روی گذرگاه ندارد. غیرفعال شدن سیگنال استروپ بیانگر این واقعیت است که گذرگاه داده دارای داده معتبر نیست. داده جدید معتبر پس از فعال شدن مجدد استروپ وجود خواهد داشت.

شکل ۴-۱۱ انتقال داده‌ای را که توسط واحد مقصد آغاز می‌شود نشان می‌دهد. در این حالت، واحد مقصد پالس استروب را فعال نموده و منبع را برای در اختیار گذاشتن داده مطلع می‌نماید. واحد منبع با قرار دادن اطلاعات دودویی موردنظر بر روی گذرگاه داده به آن پاسخ می‌دهد. داده باید به مدت کافی بر



شکل ۱۱-۴ استروب از سوی مقصد برای انتقال

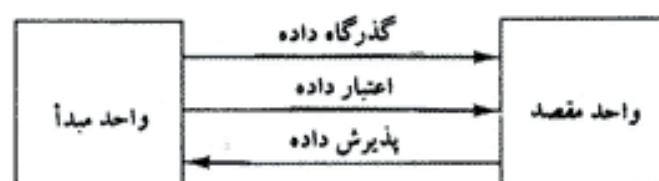
روی گذرگاه داده قرار گیرد تا واحد مقصد آن را بپذیرد. لب پایین رونده پالس استروب مجدداً برای تحریک ثبات مقصد بکار می‌رود. سپس واحد مقصد، منبع را غیرفعال می‌نماید. منبع نیز داده را پس از گذشت زمان از پیش تعیین شده‌ای از روی گذرگاه حذف می‌کند.

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 ۱۱-۳ می‌توانست یک سیگنال کنترل نوشتمن 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 نوشتمن است. بطور مشابه استروب شکل ۱۱-۴ 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 دست دهنده کنترل می‌گردد.

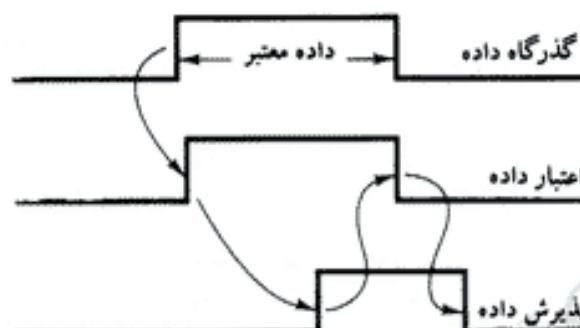
### دست دهنده

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 دست دهنی این مسئله را با معرفی یک سیگنال کنترل دیگر که پاسخی برای واحد آغازگر انتقال است، حل می‌کند. اساس روش دست دهنی دوسيمی 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 ۱۱-۵ رویه انتقال داده را وقتي که بوسیله منبع آغاز شود نشان می‌دهد. دو خط دست دهنی عبارتند از "داده معتبر"<sup>۱</sup> که بوسیله واحد منبع، و "پذیرش داده"<sup>۲</sup> که توسط واحد مقصد تولید شده‌اند. دیاگرام زمانی تبادل سیگنال‌ها را بین دو واحد نشان می‌دهد. رشته وقایع ارائه شده در بخش (ج) چهار حالت ممکن 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 غیر معتبر می‌نماید. پس از آن واحد مقصد سیگنال پذیرش داده را



(الف) بلاک دیاگرام



(ب) دیاگرام زمانبندی

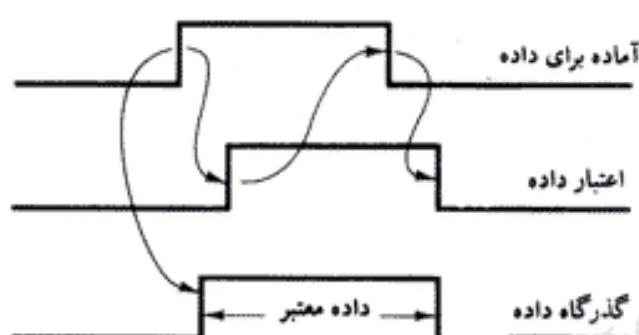
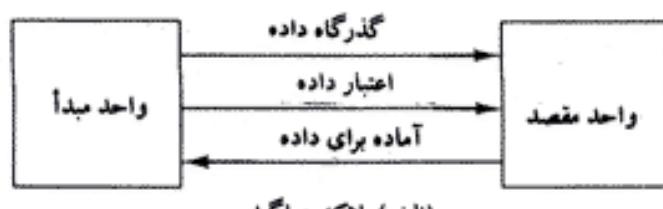


(ج) دنباله وقایع

شکل ۱۱-۵ 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 تأخیری از یک حالت به حالت بعدی مجاز است و اجازه می‌دهد تا هر واحد با سرعت انتقال داده خاص خودکار کند. سرعت انتقال را کندترین واحد معین می‌کند.

انتقال آزادشده بوسیله مقصد که بسمت روش دستدهی انجام می‌شود در شکل ۱۱-۶ نشان داده شده است. توجه کنید نام سیگنال تولید شده بوسیله واحد مقصد به "آماده برای داده" تغییر یافته است تا



(ب) دیاگرام زمانبندی



(ج) دنباله وقایع

شکل ۱۱-۶ 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 دست دهی الگوی یکسانی را مانند آنچه در حالت آغاز از سوی منبع دنبال می‌کند. توجه کنید که اگر سیگنال آماده برای داده را یعنوان 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 دست دهی 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 "اتمام وقت"<sup>۱</sup> 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 دهن را فعال سازد اجرا می‌شود. اگر سیگنال بازگشت دست دهن 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 روایت سرویس دهن مناسبی را برای رفع خطای اجرا نماید.

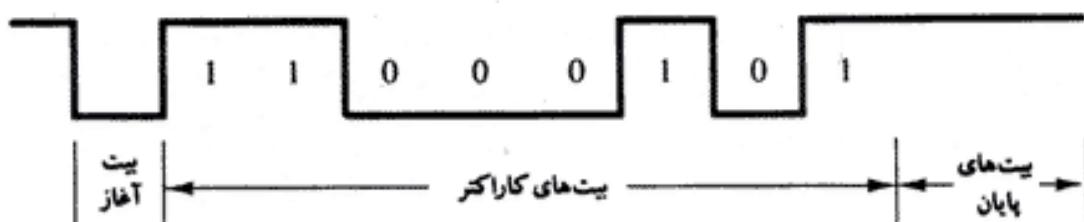
### انتقال سری غیرهمزمان

انتقال داده بین دو واحد ممکن است موازی و یا سری باشد. در انتقال موازی داده، هر بیت از پیام مسیر خاص خود را داشته و کل پیام بطور همزمان انتقال می‌باید. این بدان معنی است که یک پیام  $\pi$  بیتی باید از طریق  $\pi$  مسیرهای 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 موقتی که سرعت اهمیت دارد بکار می‌رود. انتقال سری کندر 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 مرتبًا با سرعتی که بوسیله پالس‌های ساعت دیکته می‌شود انتقال می‌بایند. در انتقال سری راه دور، هر واحد بوسیله یک پالس ساعت جداگانه با فرکانس یکسان راه اندازی می‌شود. سیگنال‌های همزمان سازی بطور متناوب بین دو واحد ارسال می‌شوند تا ساعتهاي 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 ۱۱-۸ 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 مشکل از سه بخش است: بیت شروع، بیت‌های کاراکتر، و بیت‌های توقف. قرار داد این است که وقتی کاراکتری انتقال نیابد، فرستند در وضعیت ۱ منطقی قرار می‌گیرد. اولین بیت، بیت شروع نام دارد و در وضعیت ۰ قرار داشته و برای مشخص کردن شروع کاراکتر بکار می‌رود. بیت آخر که بیت توقف نامیده می‌شود همواره ۱ است. مثالی از این قالب در شکل ۱۱-۷ نشان داده شده است.

1- Timeout



شکل ۱۱-۷ ارسال متواالی ناهمگام (غیرهمزنمان)

کاراکتر انتقال یافته طبق قانون انتقال مشخص می شود:

- ۱- وقتی کاراکتر در حال ارسال نیست خط در ۱ منطقی نگهداشته می شود.
- ۲- آغاز انتقال کاراکتر از بیت شروع که همواره ۰ است مشخص می گردد.
- ۳- بیت های کاراکتر همواره بدنیال 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 ۱ باقی بماند، مشخص می گردد.

با استفاده از این قوانین، گیرنده قادر به تشخیص بیت شروع پس از انتقال خط از ۱ به ۰ است. ساعتی در گیرنده، خط را در زمان های مناسب وارسی می کند. گیرنده سرعت انتقال بیت ها و تعداد بیت های کاراکترها مورد پذیرش را می دارد. پس از انتقال بیت های کاراکتر، یک یا دو بیت توقف ارسال می گردد بیت های توقف همواره در وضعیت ۱ منطقی هستند و انتهای کاراکتر را برای مشخص نمودن حالت انتظار یا بیکاری کاربردی می نمایند.

در انتهای کاراکتر خط بمدت زمان یک یا دو بیت در حالت ۱ نگهداشته می شود بطوری که فرستنده یا گیرنده می توانند مجددآ با یکدیگر همزمان شوند. طول زمان بقاء سیستم در این حالت به مقدار زمانی که برای دستگاهها لازم است بستگی دارد. برخی پایانه های الکترومکانیکی قدیمی دو بیت توقف بکار می بند و لی پایانه های جدیدتر از یک بیت توقف استفاده می کنند. خط، تا انتقال کاراکتر بعدی در حالت ۱ باقی می ماند. زمان توقف تضمین می کند که کاراکتر جدید به مدت یک یا دو برابر مدت زمان یک بیت روی خط ظاهر خواهد شد.

بعنوان مثال، انتقال سریال به یک پایانه کم سرعت را در نظر بگیرید که سرعت انتقال آن ۱۰ کاراکتر در ثانیه است. هر کاراکتر انتقال یافته متشکل از یک بیت شروع، هشت بیت اطلاعات، و دو بیت توقف است که جمیعاً ۱۱ بیت می شود. ۱۰ کاراکتر در ثانیه به معنی سرعت انتقال هر کاراکتر اطلاعات در ۰.۱ ثانیه است. چون ۱۱ بیت برای انتقال وجود دارد، زمان انتقال ۹.۰۹ میلی ثانیه خواهد بود. سرعت ارسال<sup>۱</sup> (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 باود 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 غیرهمزمان<sup>۱</sup> یا گیرنده فرستنده عام غیرهمزمان<sup>۲</sup> خوانده می‌شود.

### واسطه ارتباطی غیرهمزمان<sup>۳</sup>

بلاک دیاگرام مدار واسطه ارتباطی غیرهمزمان در شکل ۱۱-۸ 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 وارسی کرده و معلوم نماید که خطای رخ داده است یا خیر. ورودی انتخاب تراشه<sup>۴</sup> (CS) و کنترل‌های خواندن و نوشتن با CPU در ارتباطند. ورودی انتخاب ثبات (RS) به کنترل‌های خواندن (RD) و نوشتن (WR) 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 ارسال<sup>۵</sup> مورد استفاده، تعداد بیت‌های توقف هر کاراکتر، آیا بیت توازن باید ایجاد و چک شود یا خیر، و تعداد بیت‌های هر کاراکتر، واسطه را در وضعیت کاری خاص قرار می‌دهد. دو بیت در ثبات وضعیت به عنوان پرچم بکار می‌رودند. یکی از این بیت‌ها برای

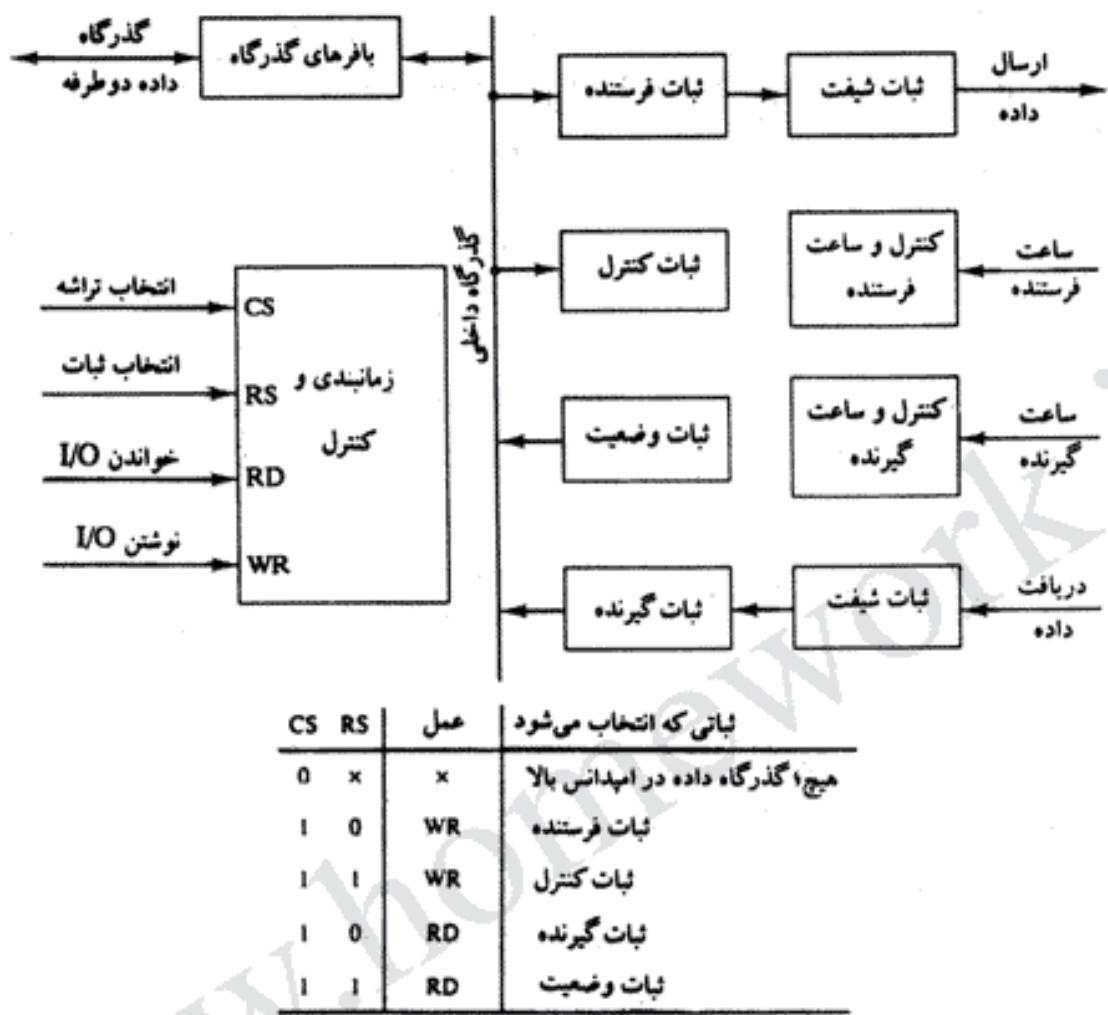
1- Asynchronous Communication Interface

2- Universal Asynchronous Receiver - Transmitter

3- Asynchronous

4- Chip Select

5- Baud rate



شکل ۱۱-۸ بلاک دیاگرام یک نوع واسطه ارتباطات غیرهمزن (غیرهمگام)

نشان دادن خالی بودن ثبات فرستنده ثبات بودن و دیگری برای نشان دادن پر بودن ثبات گیرنده بکار می‌رود. عملکرد بخش فرستنده واسطه بشرح زیر است. CPU ثبات وضعیت را می‌خواند و پرچم‌ها را برای خالی بودن ثبات فرستنده چک می‌کند. اگر این ثبات خالی باشد، CPU یک کاراکتر را به ثبات فرستنده ارسال می‌نماید و مدار واسطه پرچم را صفر می‌کند تا نشان دهد که ثبات پر است. اولین بیت ثبات فرستنده 0 می‌شود تا شروع را ایجاد کند. کاراکتر بطور موازی از ثبات فرستنده به ثبات شیفت منتقل می‌شود و تعداد مناسبی بیت‌های توقف که از پیش تعیین شده است به ثبات شیفت ملحق می‌گردد. سپس ثبات فرستنده بعنوان یک ثبات خالی تلقی می‌شود. حال کاراکتر می‌تواند بصورت هر بار یک بیت با جابجایی داده‌های موجود در ثبات شیفت با سرعت انتقال مشخص شده ارسال شود. CPU می‌تواند پس از وارسی پرچم در ثبات وضعیت، کاراکتر دیگری را به ثبات فرستنده انتقال دهد. در این

حالت گوئیم مدار واسطه بصورت دوبله بافروشده<sup>۱</sup> است. زیرا به محض اینکه کاراکتر قبلی شروع به ارسال شود کاراکتر جدید می‌تواند در ثبات فرستنده بار شود. عملکرد بخش گیرنده مدار واسطه مشابه فوق است. وقتی که خط بیکار است ورودی دریافت داده در وضعیت ۱ منطقی قرار می‌گیرد. واحد کنترل گیرنده بر خط دریافت داده نظارت می‌کند تا با مشاهده سیگنال ۰ وقوع بیت شروع را شناسایی کند. پس از مشاهده بیت شروع، بیت‌های کاراکتر که متعاقباً فرا می‌رسند با سرعت ارسال از پیش تعیین شده وارد ثبات شیفت گیرنده می‌شوند. پس از دریافت بیت‌های داده، مدار واسطه، توازن و بیت‌های توقف را چک می‌کند. سپس بیت‌های کاراکتر بدون بیت‌های شروع و توقف بطور موازی از ثبات شیفت به ثبات گیرنده انتقال می‌یابند. پرچم مربوطه در ثبات وضعیت ۱ می‌شود تا نشان دهنده پریومند ثبات گیرنده باشد. CPU ثبات وضعیت را خوانده و پرچم را وارسی می‌کند و در صورت ۱ بودن آن، داده را از ثبات گیرنده می‌خواند.

در حین انتقال، مدار واسطه هر خطای ممکنی را چک نموده و بیت مربوطه را در ثبات وضعیت ۱ می‌کند. در هر زمان که خطای رخ می‌دهد CPU می‌تواند ثبات وضعیت را وارسی کند. سه خطای ممکنی که مدار واسطه در طول ارسال وارسی می‌کند عبارتند از خطای توازن<sup>۲</sup>، خطای کادریندی<sup>۳</sup> و خطای تزاحم. خطای توازن هنگامی رخ می‌دهد که تعداد ۱ در داده‌های رسیده دارای توازن صحیحی نیست. خطای کادریندی هنگامی بوقوع می‌پیوندد که تعداد بیت‌های توقف در انتهای کاراکتر رسیده صحیح نباشد. خطای تزاحم در صورتی رخ می‌دهد که قبل از خواندن کاراکتر از ثبات گیرنده بوسیله CPU، کاراکتر بعدی در ثبات شیفت قرار گیرد. این خطای موجب از دست رفتن کاراکتر از رشته داده‌های دریافتی می‌شود.

### بافر اولین ورودی - اولین خروجی

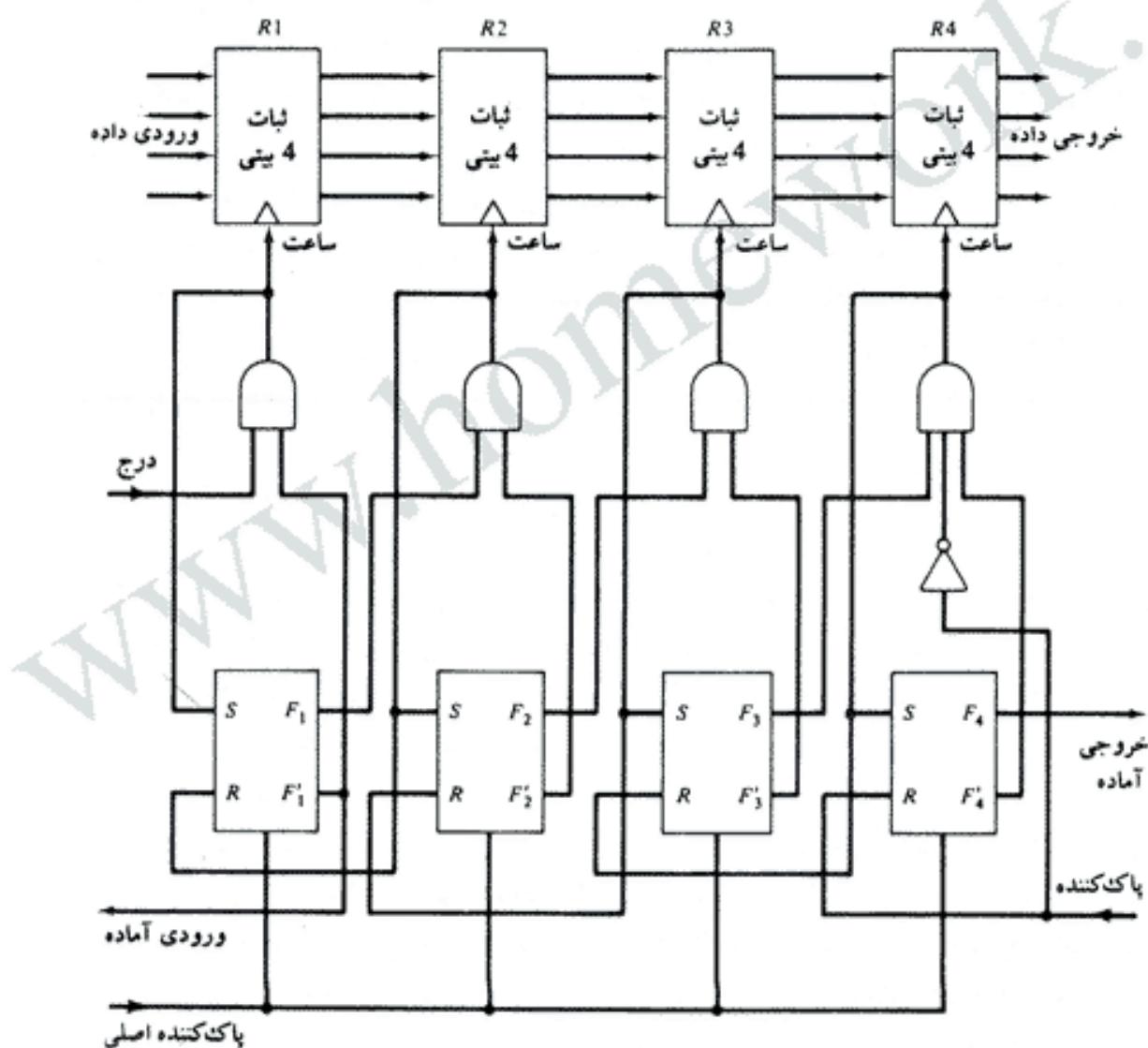
بافر اولین ورودی - اولین خروجی (FIFO) واحد حافظه‌ای است که اطلاعات را به گونه‌ای ذخیره می‌کند که اولین داده وارد شده اولین داده‌ای خواهد بود که خارج شود. بافر FIFO پایانه‌های ورودی - خروجی جداگانه‌ای دارد. ویژگی مهم این نوع بافر اینست که می‌تواند ورود و خروج داده‌ها را با دو سرعت متفاوت انجام دهد و داده‌های خروجی همیشه بهمان ترتیبی که وارد می‌شوند خواهند بود. هر گاه FIFO بین دو واحد قرار گیرد، قادر خواهد بود داده‌ها را با سرعتی از واحد مبدأ دریافت و با سرعتی دیگر به واحد مقصد تحویل دهد. اگر واحد منبع کندر از واحد مقصد باشد، می‌توان 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 موقعي مفید خواهد بود که داده‌های ورودی بیکاره فرا برستند و میانگین 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 می‌تواند در برخی از کاربردها مفید باشد و آن هنگامی است که داده‌ها بطور غیرهمزمان انتقال یابند. این بافر داده‌ها را که به تدریج فرا می‌رسند 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  $4 \times 4$  در شکل ۱۱-۹ نشان داده شده است. این بافر متشکل از چهار ثبات چهار بیت RI با  $I=1,2,3,4$  و یک ثبات کنترل با فلیپ فلاپ های  $F_i$ ،  $i=1,2,3,4$  می باشد. یعنی هر فلیپ فلاپ برای یک بیت، 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 ثبات افزایش یابد و تعداد کلمات نیز با افزایش تعداد ثبات ها اضافه می گردد.  
 فلیپ فلاپ  $F_i$  در ثبات کنترل که با ۱ بار شده است نشان می دهد که یک کلمه چهار بیتی در ثبات

شکل ۱۱-۹ دیاگرام مدار بافر FIFO با اندازه  $4 \times 4$

مربوطه اش RI ذخیره شده است. وجود ۰ 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 ۱ باشد ( $F_j=1$ ) 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 پذیرد. همان گذرپالس ساعت<sup>۱</sup>  $F_{i+1}$  را ۱ و  $F_i$  را ۰ 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 ثبات مانند RI برسد و فلیپ فلاپ بعدی آن یعنی  $F_{i+1}$  مقدار ۱ را داشته باشد، یا به آخرین ثبات  $R_4$  برسد، متوقف می گردد. یک پاک کننده اصلی برای پاک کردن کلیه فلاپ های ثبات کنترل بکار می رود.

داده ها بشرطی به داخل بافر وارد می شوند که سیگنال "وروودی آماده" فعال شده باشد. این هنگامی رخ می دهد که اولین 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<sup>۲</sup> ۱ بار می شوند. همان پالس ساعت  $F_1$  را ۱ می نماید که این خود کنترل ورودی آماده<sup>۳</sup> را غیرفعال نموده و این بنویه 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 ۱ باقی مانده ولی خط ورودی آماده در حالت ۰ 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 افتد در سمت خروجی جمع می شوند. خط کنترل "خروچی آماده"<sup>۴</sup> هنگامی که آخرین فلاپ فلاپ  $F_4$  با ۱ 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<sup>۵</sup> را فعال می سازد. این بنویه خود  $F_4$  را پاک کرده و سبب غیرفعال شدن "خروچی آماده" می گردد. اگر FIFO خالی باشد، داده ای از  $R_3$  وجود نداشته و  $F_4$  در وضعیت ۰ باقی می ماند. توجه کنید که دو خط کنترل "خروچی آماده" و "پاک کننده" تشکیل خطوط دست دهی برای منبع آغازگر را می دهند.

#### ۱۱-۴ شیوه های انتقال

اطلاعات دودویی دریافتی از وسائل 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 |                |

دستورالعمل‌های I/O را اجرا نموده و ممکن است داده را بطور موقت پذیرد، ولی مبدأ یا مقصد نهایی حافظه است. انتقال داده بین کامپیوتر مرکزی و وسایل I/O می‌تواند با شیوه‌های مختلفی انجام شود. برخی روش‌ها از CPU بعنوان مسیر میانی استفاده می‌کند، بعضی مستقیماً داده را به و یا از حافظه منتقل می‌سازند. 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 دستورالعمل‌های I/O نوشته شده در برنامه کامپیوتر است. انتقال هر قلم داده از سوی دستورالعمل در برنامه آغاز می‌شود. 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 می‌تواند دوباره انتقال را انجام دهد. در واقع این دیگر بعده دستورات اجرا شده برنامه در CPU است تا دقیقاً بر هر اتفاقی که در واحد واسطه و وسیله I/O رخ می‌دهد نظارت کند.

در روش I/O برنامه نویسی شده، CPU در یک حلقه از برنامه باقی می‌ماند تا واحد I/O مشخص نماید که برای انتقال داده آماده است. این فرآیند موجب اثلاف قابل توجه وقت می‌شود زیرا پردازنده را بی‌آنکه لازم باشد مشغول نگه می‌دارد. برای احتراز از اثلاف وقت می‌توان از امکانات وقفه<sup>۱</sup> 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<sup>۲</sup> را صادر کند. در خلال این مدت مدار واسطه وسیله جانبی را زیرنظر دارد. وقتی مدار واسطه مشخص کند که وسیله جانبی آماده انتقال داده است، یک سیگنال تقاضای وقفه را به کامپیوتر صادر می‌کند. به محض تشخیص سیگنال وقفه خارجی، CPU موقتاً کار خود را متوقف کرده، به برنامه سرویس دهن برای پردازش انتقال I/O انشعاب نموده، و سپس به کاری که در اصل در حال انجام بود باز می‌گردد.

انتقال داده بروش I/O برنامه نویسی شده بین CPU و دستگاه جانبی انجام می‌شود. در دستیابی مستقیم به حافظه<sup>۳</sup> (DMA)، مدار واسطه داده را از طریق گذرگاه حافظه تبادل می‌کند. CPU انتقال را با تهیه آدرس شروع برای واسطه و تعداد کلمات لازم برای انتقال آغاز نموده و سپس برای انجام سایر کارها پیش روی می‌نماید. وقتی که انتقال پایان یافته، DMA سیکل حافظه را از طریق گذرگاه حافظه تقاضا کند. وقتی که تقاضا بوسیله کنترل حافظه پذیرفته شد، DMA داده‌ها را مستقیماً به حافظه می‌فرستد. CPU فقط عمل مراجعة به حافظه خود را به تأخیر می‌اندازد تا انتقال مستقیم I/O به حافظه

۱- Interrupt

2- Interrupt request signal

3- Direct Memory Access

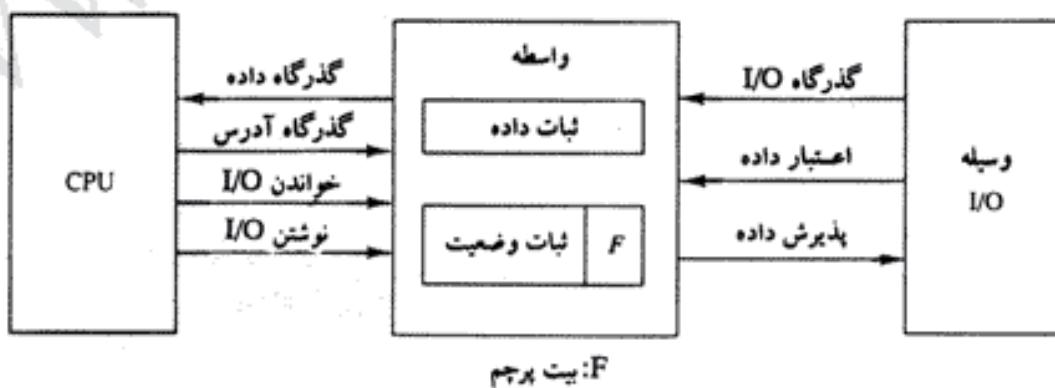
اجرا شود. چون معمولاً سرعت وسیله جانبی کمتر از سرعت پردازنده است، انتقال I/O - حافظه در مقایسه با مراجعات پردازنده به حافظه بندرت رخ می دهد. انتقال DMA در بخش ۱۱-۶ مفصل تر بحث شده است.

در بسیاری از کامپیوترها، مدار واسطه لازم برای ارجاع مستقیم به حافظه را در یک واحد که آن را پردازنده O/I (یا I/O)<sup>۱</sup> می‌نمایند قرار می‌دهد. IOP می‌تواند بسیاری از وسائل جانبی را از طریق یک DMA و امکان وقفه کنترل نماید. در یک چنین سیستمی، کامپیوتر به سه مأذول جداگانه تقسیم می‌شود: واحد حافظه، CPU، و IOP. پردازنگر O/I در بخش ۱۱-۷ 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 ۱۱-۱۰ نشان داده شده است. وسیله بایت‌های داده را یک به یک همانطور که در اختیار قرار می‌گیرند انتقال می‌دهد. وقتی که یک بایت در اختیار قرار گرفت، وسیله آن را روی گذرگاه I/O قرار می‌دهد و خط داده معتبر<sup>۲</sup> را فعال می‌سازد. مدار واسطه بایت را پذیرفته و در ثبات داده خود قرار می‌دهد و خط پذیرش داده<sup>۳</sup> را فعال می‌نماید. مدار واسطه بیتی را در ثبات وضعیت ۱ می‌نماید که ما آن را بیت F یا پرچم<sup>۴</sup> می‌خوانیم. در این لحظه وسیله جانبی می‌تواند خط داده معتبر را غیرفعال نماید ولی بایت دیگری را، تا غیرفعال شدن خط پذیرش داده



شکل ۱۱-۱۰ انتقال داده از وسیله I/O به CPU

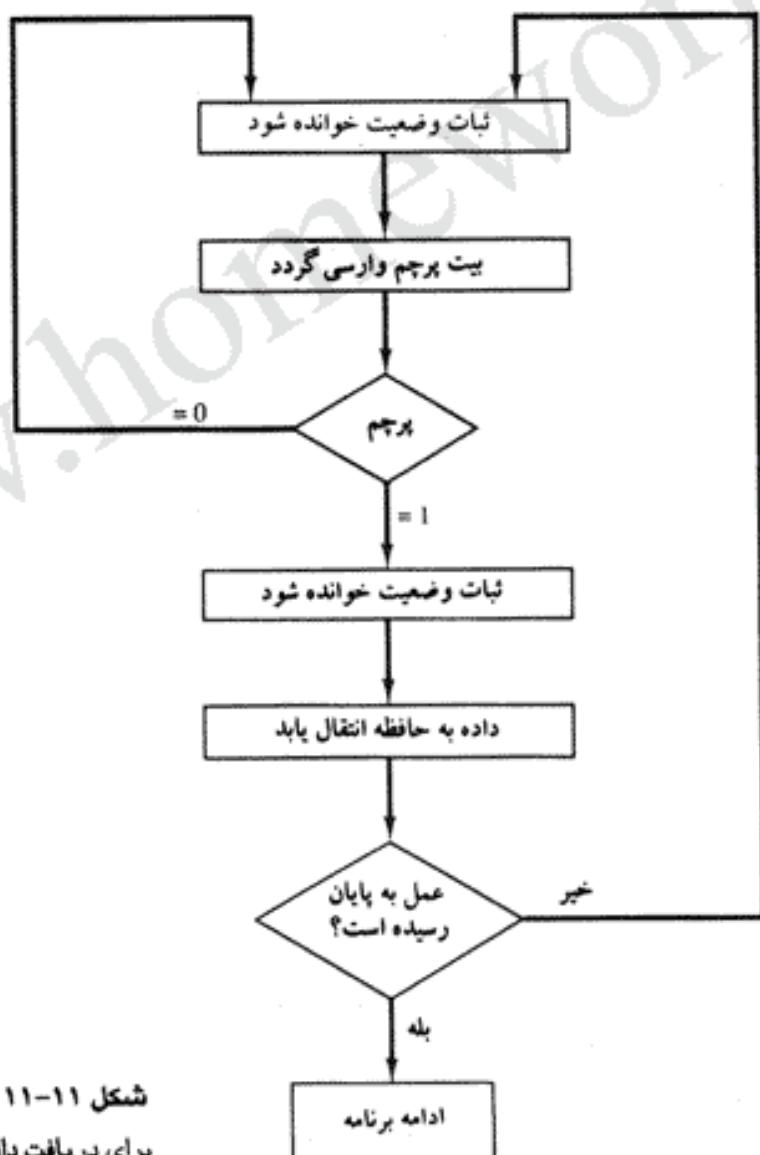
1- I/O Processor  
4- Flag

### 2- Data Valid

### 3- Data Accepted

بوسیله مدار واسطه، منتقل نخواهد کرد. این عملیات بر طبق روند دست دهنی مبتنی بر شکل ۱۱-۵ است. برنامهای برای کامپیوتر نوشته شده است تا پرچم را در ثبات وضعیت وارسی نموده و تعیین کند که آیا وسیله I/O بایتی در ثبات داده قرار داده است یا خیر. این عمل با خواندن ثبات وضعیت و قرار دادن آن در یک ثبات CPU و وارسی بیت پرچم صورت می‌گیرد. اگر پرچم برابر ۱ باشد، CPU داده را از ثبات داده می‌خواند. سپس بیت پرچم با ۰ 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 بتایراین 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 ۱ اگر این بیت ۱ نباشد، و یا به مرحله ۳ اگر بیت مذکور ۱ باشد.
- ۳- خواندن ثبات داده.

هر بایت بداخل یک ثبات پردازندۀ 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 O/I 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 وارسی می کند در یک میکروثانیه 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 میکروثانیه است. در نتیجه CPU بین هر دو بار انتقال 10 هزار بار پرچم را وارسی می کند. CPU هنگام وارسی پرچم بجای انجام یک کار پردازشی مفید دیگر، در واقع در حال اتلاف وقت است.

## I/O بکمک 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 مدامی که مشغول اجرای یک برنامه است پرچم را وارسی نمی کند. با این وجود، هنگامی که پرچم ۱ شده باشد، یک وقفه کامپیوتر را لحظه ای از اجرای برنامه جاری متوقف ساخته و ۱ 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 پشتۀ 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<sup>۱</sup> و دیگری وقفه

1- Vectored Interrupt

غیربرداری<sup>۱</sup> نام دارد. در وقفه غیربرداری، آدرس انشعب به مکان ثابتی از حافظه تخصیص می‌یابد. در وقفه برداری منبعی که وقفه را ایجاد می‌کند اطلاعات انشعب را نیز برای کامپیوتر فراهم می‌آورد. این اطلاعات بردار وقفه<sup>۲</sup> نامیده می‌شود. در بعضی از کامپیوتراها بردار وقفه آدرسی است که به مکانی از حافظه که روای سرویس I/O است اشاره می‌کند. سیستمی با وقفه برداری در بخش ۱۱-۵ آورده شده است.

### ملاحظات نرم افزاری

بحث قبلی به سخت افزار پایه مورد نیاز برای ارتباط وسایل I/O به کامپیوتر مربوط بود. کامپیوتر باید یک سری روای های نرم افزاری هم برای کنترل وسایل جانبی و انتقال داده‌ها بین پردازنده و وسایل جانبی داشته باشد. روای های I/O باید فرمان‌های کنترل را برای فعال کردن وسایل جانبی برای وارسی 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 خطاهای 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 روای های I/O برای دستگاه‌های جانبی استاندارد توسط سازنده بعنوان بخشی از سیستم کامپیوترا در اختیار گذاشته می‌شود. این روای ها معمولاً همراه با سیستم عرضه می‌شود. اکثر سیستم‌های عامل انواع متنوعی از برنامه‌های I/O برای پشتیبانی خط خاص دستگاه‌های جانبی برای کامپیوترا را به همراه دارند. روای های I/O معمولاً بصورت رویه‌های سیستم عامل در اختیار قرار دارند و کاربر بدون اینکه وارد برنامه‌های زبان ماشین شود با ارجاع به روای های مربوطه نوع انتقال مطلوب را مشخص می‌کند.

### ۱۱-۵ 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<sup>۳</sup> پاسخ داده و سپس برنامه به روای سرویس که انتقال مورد نظر را انجام می‌دهد انشعب می‌کند. همانطور که در بخش ۸-۷ بحث شد، بعضی از پردازنده‌ها PSW<sup>۴</sup> جاری را به پشته منتقل می‌کنند و یک PSW جدیدی را نیز برای روای سرویس بار می‌نمایند. در اینجا ما از PSW چشم پوشی 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 عاقب 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 پرسی<sup>۱</sup> استفاده می شود. در این روش، یک آدرس انشعباب مشترک برای تمام وقفه ها وجود دارد. برنامه ای که مراقبت از وقفه را بعده 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 انشعباب می کند. در غیراین صورت منبعی که اولویت پائیتر 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 انشعباب می نماید. روال سرویس دهی که اجرا می شود متعلق به وسیله ای است که بالاترین اولویت را در بین همه وسیله هایی که به کامپیوتر وقفه ارسال داشته اند دارد است. 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 تسريع عمل استفاده کرد.

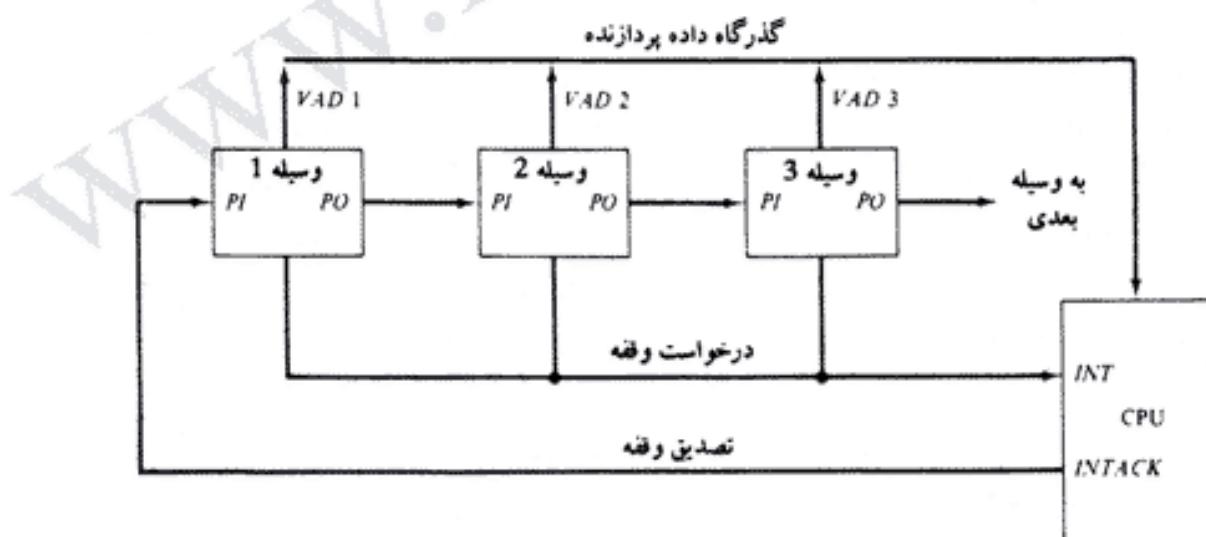
دستگاه وقفه اولویت دار سخت افزاری بصورت مدیر کل در سیستم وقفه عمل می کند. این دستگاه درخواست های وقفه را از چندین منبع دریافت و سپس تعیین می کند که کدام تقاضا دارای بالاترین اولویت است. آنگاه بر مبنای آن یک درخواست وقفه را به کامپیوتر صادر می نماید. برای تسريع عمل، هر منبع وقفه بردار وقفه خاص خود را برای دستیابی مستقیم به روال سرویس مخصوص آن دارد. به

## 1- Polling Procedure

این ترتیب همه پرسی لازم نیست زیرا همه تصمیم‌ها توسط دستگاه وقفه اولویت دار سخت افزاری اتخاذ می‌گردد. عمل اولویت دهنده سخت افزاری را می‌توان با اتصال سری یا موازی خطوط وقفه انجام داد. روش اتصال سری، اتصال زنجیره‌ای<sup>۱</sup> نیز خوانده می‌شود.

### اولویت زنجیره‌ای

روش زنجیره‌ای برای اولویت دهنده بر اتصال سری وسیله‌هایی که درخواست وقفه ارسال می‌دارند مبتنی است. وسیله‌ای که بالاترین اولویت را دارد در اولین مکان قرار داده می‌شود، و پس از 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 در خواست وقفه برای همه وسیله‌ها غیرمشترک بوده و یک اتصال منطقی سیمی<sup>۲</sup> را تشکیل می‌دهد. اگر هر وسیله سیگنال وقفه خود را در سطح پائین بپرسد، 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<sup>۳</sup> 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

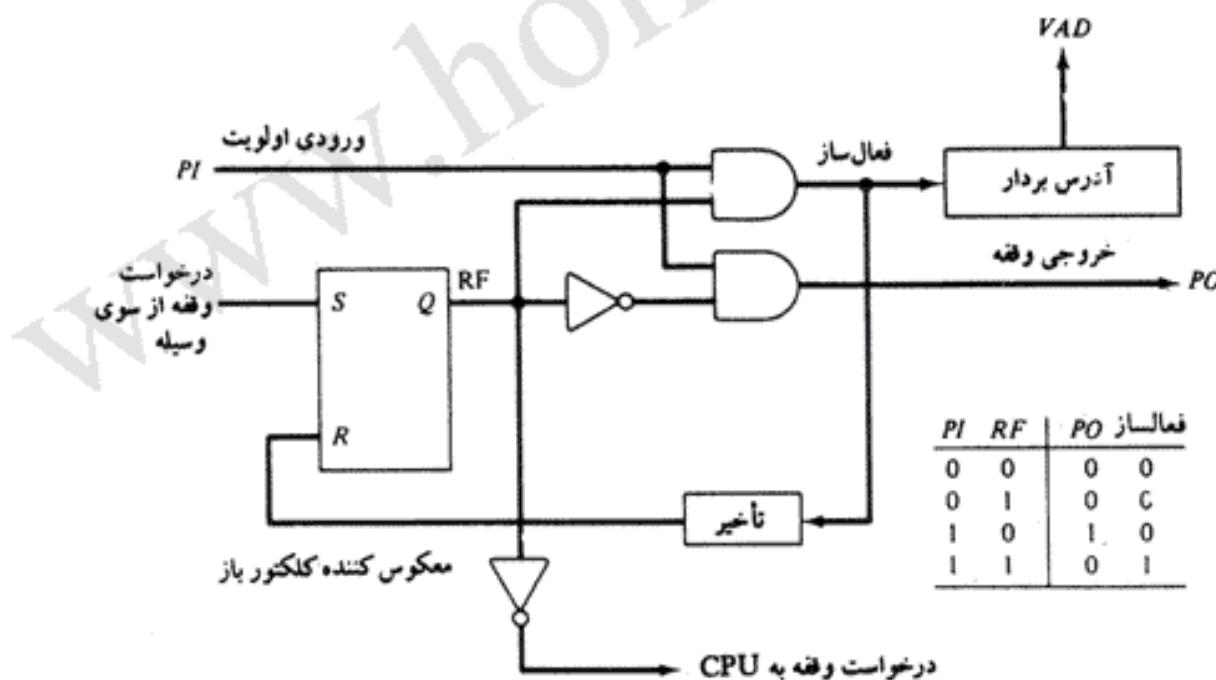
2- Wired logic

3- Acknowledge

سپس این وسیله آدرس بردار وقفه<sup>۱</sup> خود را (VAD) روی گذرگاه قرار می دهد تا CPU در سیکل وقفه از آن استفاده نماید.

وسیله ای که در ورودی PI دارای ۰ باشد در خروجی PO هم ۰ 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 ۱ در ورودی PI باشد سیگنال تصدیق را با قرار دادن ۰ در خروجی PO می بندد. اگر وسیله جانبی وقفه سرویس دهی شونده ای نداشته باشد، سیگنال تصدیق را با قرار دادن ۱ 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 را به 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 ۱ می کند. خروجی فلیپ فلاپ RF از طریق یک معکوس کننده کلکتور باز<sup>۲</sup>، 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  $PO=1$  و بردار آدرس غیرفعال



شکل ۱۱-۱۳ یک طبقه از آرایش اولویت دهنده زنجیره ای

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 براین 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

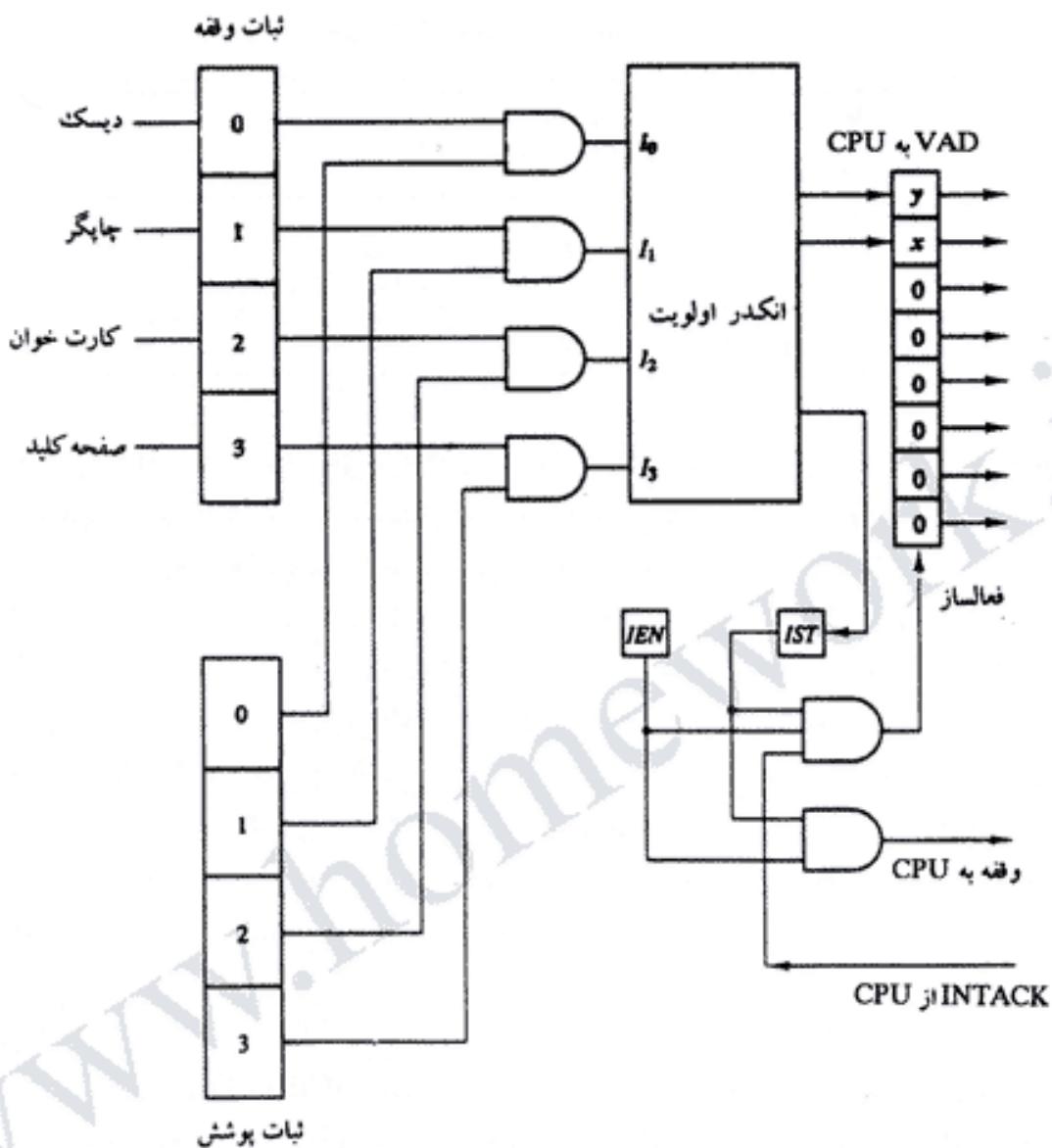
روش وقفه اولویت دار موازی از ثباتی استفاده می‌کند که بیت‌های آن بطور جداگانه توسط سیگنال وقفه هر وسیله 1 می‌شود. اولویت براساس محل بیت‌ها در ثبات تعیین می‌گردد. علاوه بر ثبات وقفه، مدار ممکن است دارای یک ثبات پوشش<sup>1</sup> باشد که هدف از آن کنترل وضعیت هر تقاضای وقفه است. ثبات پوشش (ماسک) را می‌توان برای غیرفعال شدن وقفه‌های پائین رتبه‌تر، در حین سرویس دهی به وقفه‌هایی با اولویت بالاتر، برنامه دهی 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 ۱۱-۱۴ 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 بیت 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 بواسطه 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 بواسطه 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، با فرایند 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 عکس العمل 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 ۱۱-۱۴ سخت‌افزار وقفه اولویت‌دار

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 بی‌همیت را تداعی می‌کنند. ورودی ۰ با بالاترین اولویت را دارد، لذا صرف نظر از مقادیر دیگر ورودی‌ها، وقتی که این ورودی ۱ باشد، خروجی تولید  $xy = 00$  می‌نماید.  $I_1$  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 ۰۱ خواهد بود.

## جدول ۱۱-۲ جدول ارزش کنگنار اولویت

ورودی ها				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 می شود که ورودی های با اولویت بالاتر 0 باشند و به همین منوال اولویت های پائین تر، فلیپ فلاپ وضعیت وقنه IST فقط وقتی که یک یا چند ورودی برابر 1 باشند، 1 می شود. اگر تمام ورودی ها 0 باشند، IST پاک می شود و سایر خروجی های انکدر بکار نمی روند، لذا با x علامت زده شده اند. این بدان علت است که آدرس بردار وقنه که  $0 = IST$  است به CPU مستقل نمی شود. توابع بول لیست شده در جدول منطق درونی انکدر را مشخص می کند. معمولاً یک کامپیووتر بیش از چهار منبع وقنه دارد. مثلاً یک انکدر با هشت ورودی، یک خروجی سه بیتی تولید می نماید. خروجی انکدر اولویت، برای تشکیل بخشی از آدرس بردار هر منبع وقنه بکار می رود. به سایر بیت های بردار آدرس هر مقداری را می توان اختصاص داد. مثلاً آدرس بردار را می توان با الحاق شش صفر به خروجی های x و لا انکدر تشکیل داد. با این انتخاب، شماره های دودویی 0، 1، 2، 3 به بردارهای وقنه مربوط به چهار وسیله O/I اختصاص می یابد.

## سیکل وقنه

فلیپ فلاپ تواناساز وقنه IEN که در شکل ۱۱-۱۴ نشان داده شده است می تواند بوسیله دستور العمل های برنامه 1 یا 0 شود. وقتی که IEN پاک شود، تقاضای وقنه ای که از IST می رسد بوسیله CPU نادیده گرفته می شود. بیت IEN کنترل شده بوسیله برنامه به برنامه نویس اجزا می دهد تا درمورد استفاده یا عدم استفاده از امکانات وقنه تصمیم بگیرد. اگر دستور العملی برای پاک کردن IEN در برنامه گنجانیده شود، بدان معنی است که برنامه نویس مایل نیست برنامه اش متوقف شود. دستور العملی برای 1 کردن IEN نشانه این است که در حین اجرای برنامه جاری از قابلیت وقنه استفاده خواهد شد. اکثر کامپیووترها دارای سخت افزار برای پاک کردن IEN، در مواقعي که وقنه بوسیله پردازنده تصدیق شوند، هستند.

در پایان هر سیکل دستور العمل، CPU بیت IEN و سیگنال وقنه را از IST وارسی می کند. اگر هر یک از آنها 0 باشد، کنترل با دستور بعدی به کار ادامه می دهد. اگر هر دو بیت IEN و IST برابر 1 باشند، CPU وارد سیکل وقنه می شود. در حین سیکل وقنه، CPU رشته ریزاعمال زیر را انجام می دهد.

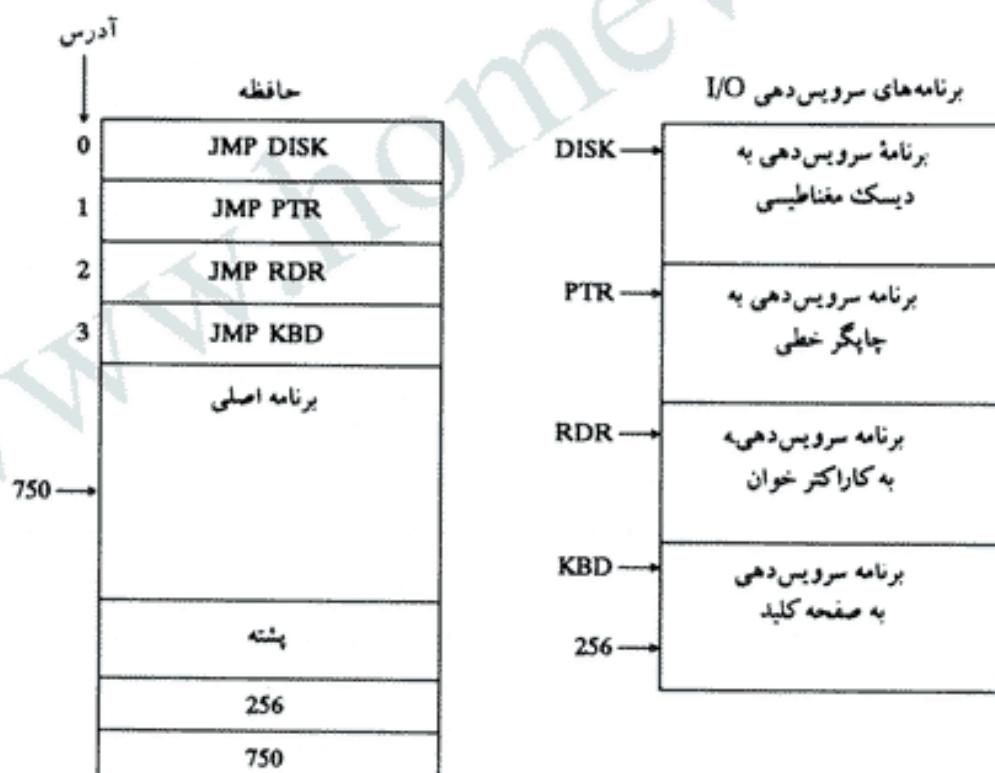
SP  $\leftarrow$  SP-1      اشاره گر پشته را کاهش بده

$M [SP] \leftarrow PC$	پوش کردن PC به داخل پشت
$INTACK \leftarrow 1$	تصدیق وقفه را فعال کن
$PC \leftarrow VAD$	بردار آدرس را به PC منتقل کن
$IEN \leftarrow 0$	وقفه بعدی را غیرفعال کن
	برو برای برداشت دستورالعمل بعدی

CPU آدرس برگشت را از PC به پشت 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<sup>۱</sup> بعدی IEN را پاک می‌کند. دستورالعملی که در حین فاز برداشت بعدی از حافظه خوانده می‌شود در بردار آدرس قرار خواهد داشت.

### روال‌های 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 ذخیره شده در حافظه برای سرویس دهی به وقفه‌ها

1- Fetch

و قله و کنترل ثبات های سخت افزار و قله روال های نرم افزاری نیز داشته باشد. شکل ۱۱-۱۵ برنامه هایی را که باید برای اداره سیستم و قله 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 سمبولیک هر روال نشان دهنده آدرس شروع برنامه سرویس است. پشته نشان داده شده برای ذخیره کردن آدرس بازگشت پس از هر وقfe است.

برای روشن کردن مطلب با یک مثال فرض کنید که صفحه کلید بیت و قله را مدامی که CPU در حال اجرای دستورالعملی در مکان 749 برنامه اصلی است، ۱ کند. در پایان سیکل دستورالعمل، کامپیوتر به یک سیکل و قله می رود. ابتدا آدرس بازگشت 750 را در پشته ذخیره کرده و سپس آدرس بردار 000000011 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 و قله اش را، هنگامی که CPU در حال اجرای دستورالعمل واقع در مکان 255 در برنامه KBD است، ۱ 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 وقfe 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 بر می گرداند. بنابراین، وسیله ای که دارای اولویت بالاتر باشد می تواند بوسیله ای که اولویت پائین تر را دارد وقfe دهد. فرض براین است که زمان صرف شده در سرویس به وسیله ای که وقfe 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 وقfe باید مجموعه ای از عملیات آغازین و پایانی را برای کنترل ثبات ها سیستم وقfe سخت افزاری داشته باشد. بخاطر بیاورید که فعال ساز وقfe IEN در پایان سیکل وقfe پاک می شود. این فلیپ فلاپ مجدداً باید برای فعال کردن تقاضاهای وقfe با اولویت بالاتر، ۱ شود، ولی نه قبل از غیرفعال شدن وقfe های با اولویت پائین تر. رشته اعمال آغازین هر روال سرویس وقfe باید دارای دستورالعمل هایی برای کنترل سخت افزار وقfe به شکل زیر باشد.

۱- صفر کردن بیت های سطح پائینتر ثبات پوشش

۲- پاک کردن بیت وضعیت وقfe IST

۳- ذخیره کردن محتوای ثبات های پردازنده

۴- ۱ کردن بیت فعال ساز وقfe IEN

۵- اجرای روال سرویس

بیت‌های سطح پائین تر 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 پائیتر دارند به بیت‌های بالا رتبه ثبات پوشش تخصیص یافته‌اند، می‌توان اولویت را در صورت تمایل عوض کرد زیرا برنامه نویس 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 ۱ شود. محتواهی ثبات پردازنده ذخیره می‌شوند زیرا ممکن است بوسیله برنامه‌ای که به آن وقفه داده شده است پس از بازگشت کنترل به آن موردنیاز واقع شود. سپس فعال ساز وقفه IEN برابر با ۱ 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 محتواهی ثبات‌های پردازنده
- ۳- پاک کردن بیت متعلق به منبعی که سرویس داده شده، در ثبات وقفه
- ۴- کردن بیت‌های با اولویت پائیتر در ثبات پوشش
- ۵- بازگرداندن آدرس بازگشت در PC و ۱ کردن IEN

بیتی در ثبات وقفه که به منبع وقفه متعلق است باید ۰ شود تا این منبع بتواند مجدداً وقفه را تقاضا کند. بیت‌هایی دارای اولویت پائین تر در ثبات پوشش (از جمله بیت مربوط به منبعی که به آن سرویس داده شد) ۱ 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 ابتداً پاک شده و پس از انتقال آدرس بازگشت در ثبات PC دوباره ۱ می‌شود.

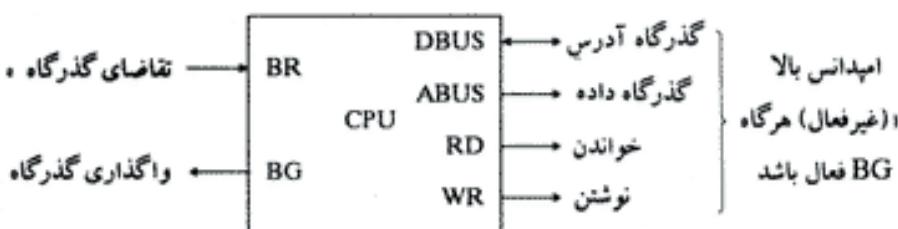
عملیات آغازین و پایانی لیست شده فوق عملیات سریار<sup>۱</sup> نامیده می‌شوند. این عملیات بخشی از برنامه اصلی سرویس نیستند ولی برای پردازش وقفه ضرورت دارند. همه عملیات سریار را می‌توان با نرم افزار پیاده‌سازی کرد. این عمل با گنجاندن دستورالعمل‌های مناسب در ابتدا و انتهای هر زیررول انجام می‌شود. برخی از اعمال سریار بوسیله سخت افزار بطور اتوماتیک انجام می‌گردد. محتواهی 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

## ۱۱-۶ دستیابی مستقیم به حافظه (DMA)<sup>۱</sup>

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 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 بطريق 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 ۱۱-۱۶ دو سیگنال کنترل در CPU که امکان انتقال DMA را فراهم می کنند نشان داده شده است. ورودی تقاضای گذرگاه<sup>۲</sup> (BR) بوسیله کنترل کننده DMA برای تقاضای وقفه از CPU بکار می رود تا کنترل گذرگاه ها را ره نماید. وقتی این ورودی فعال شود گذرگاه ها بوسیله سیگنال های کنترل خارجی را پایان داده و گذرگاه آدرس، گذرگاه داده، و خطوط خواندن و نوشتمن 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 (۳-۴ را ببینید). CPU خروجی واگذاری گذرگاه<sup>۳</sup> (BG) را فعال می کند تا به DMA اطلاع دهد که گذرگاه ها در حالت امپدانس - بالا هستند. اکنون DMA 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 بعده می گیرد، و آنرا به عملکرد عادی خود باز می گرداند.

وقتی که DMA کنترل سیستم گذرگاه را بعده می گیرد، مستقیماً با حافظه ارتباط برقرار می کند. انتقال می تواند به چندین طریق انجام شود. در انتقال یکباره<sup>۴</sup> DMA، یک رشته بلاک مشکل از یک تعداد کلمات حافظه در حالی که DMA بر گذرگاه حافظه حاکم است، بطور پیوسته بیکاره منتقل می گردد. این شیوه انتقال برای وسائل سریع مانند دیسک های مغناطیسی ضروری است. در این سیستم ها



شکل ۱۱-۱۶ سیگنال های گذرگاه CPU برای انتقال DMA

- 1- Direct Memory Access  
4- Burst Transfer

- 2- Bus Request

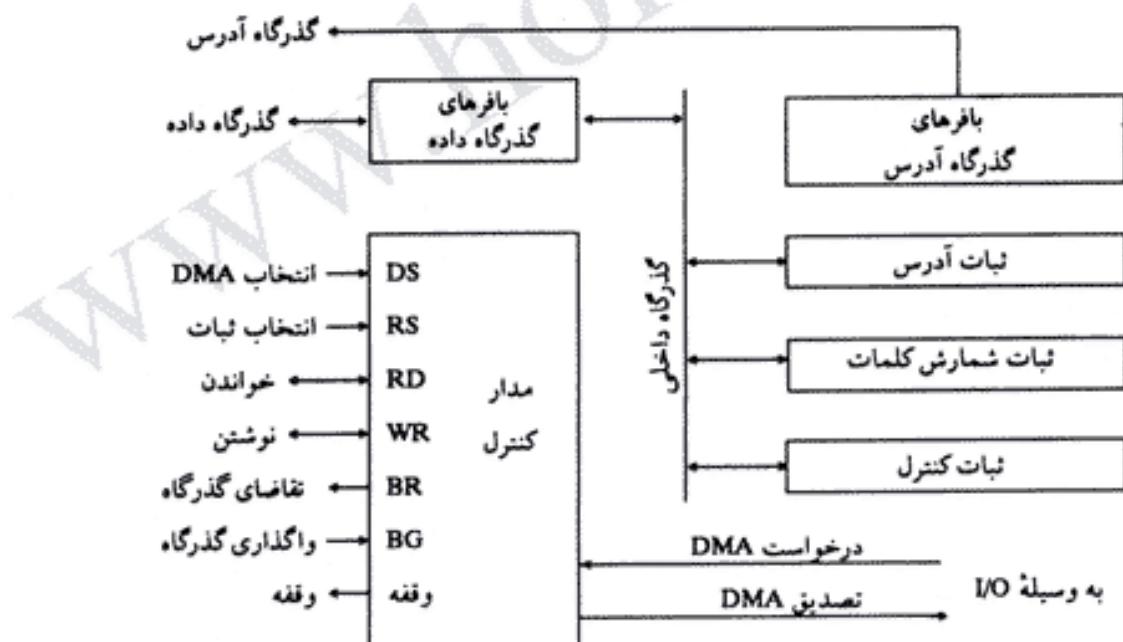
- 3- Bus Grant

نمی‌توان تالنتقال کامل بلاک داده‌ها، انتقال را متوقف یا کنند کرد. تکنیک دیگری که سرقت سیکل<sup>۱</sup> نامیده می‌شود به کنترل کننده DMA اجازه می‌دهد که هر بار یک کلمه داده را انتقال دهد و پس از آن کنترل گذرگاه‌ها را به CPU بازگرداند. CPU صرفاً عملکرد خود را به اندازه یک سیکل حافظه به تأخیر می‌اندازد تا عمل انتقال حافظه I/O بطور مستقیم صورت گرفته و یک سیکل حافظه را برباید.

DMA کنترل کنندہ

کنترل کننده DMA برای ارتباط با CPU و وسیله جانبی به مدارات معمول واسطه نیاز دارد. بعلاوه یک ثبات آدرس و یک ثبات شمارش کلمه<sup>۲</sup> و یک مجموعه خطوط آدرس هم مورد نیاز است. ثبات آدرس و خطوط آدرس برای ارتباط مستقیم با حافظه یکار می‌روند. ثبات شمارش کلمه تعداد کلماتی را که باید منتقل شود مشخص می‌کند. انتقال داده‌ها می‌تواند مستقیماً تحت کنترل DMA بین وسیله جانبی و حافظه صورت گیرد.

شکل ۱۱-۱۷ بلاک دیاگرام یک نمونه کنترل کننده DMA را نشان می‌دهد. کنترل کننده با CPU از طریق گذرگاه داده و خطوط کنترلی ارتباط برقرار می‌کند. ثبات‌های DMA بوسیله CPU از مسیر گذرگاه آدرس و با فعال کردن ورودی‌های DS (انتخاب DMA)<sup>۳</sup> و RS (انتخاب ثبات)<sup>۴</sup> انتخاب می‌شوند. ورودی‌های RD (خواندن) و WR (نوشتن) دووجهه هستند. وقتی که BG (واگذاری گذرگاه) صفر باشد،



شکل ۱۷-۱۷ پلاک دیاگرام کنترل کننده DMA

- 1- Cycle stealing
- 4- Register selection

## 2- Word Count Register

### 3- DMA 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 وسائل خارجی از طریق خطوط تقاضا و تصدیق و استفاده از رویه از قبل تعین شده دست دهن 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 وارسی 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 مت Shankل 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

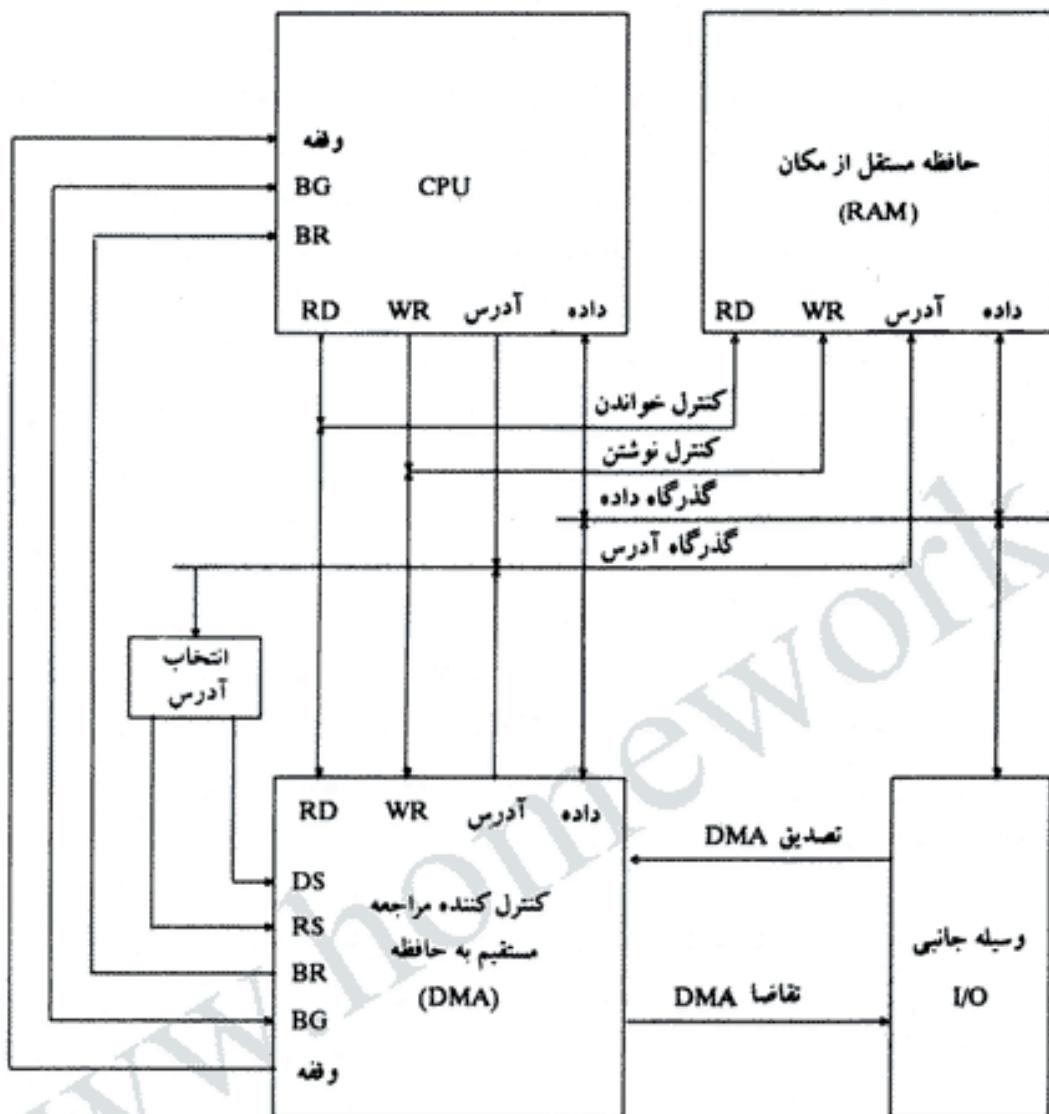
۳- کلمه کنترل برای مشخص کردن اطلاعات مربوط به شیوه انتقال مانند خواندن یا نوشتن

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 وارسی نماید.

## DMA انتقال

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 کامپیووتری

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 راه‌ها 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 جهته 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، RD و WR 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 رادر گذرگاه داده قرار می‌دهد (برای نوشتمن) یا کلمه را از گذرگاه داده دریافت می‌کند (برای خواندن). بنابراین DMA عمل خواندن یا نوشتمن 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<sup>۱</sup> و اصله 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 کنترلر 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 تقاضاً 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 معنی انتقال موفقیت آمیز کلیه کلمات است. CPU در هر لحظه قادر به خواندن این ثبات می‌باشد و بدینوسیله کلماتی را که قبلً منتقل شده‌اند وارسی می‌کند.

یک کنترل کننده DMA ممکن است بیش از یک کانال داشته باشد. در این حال، هر کانال دارای جفت سیگنال‌های کنترل تقاضاً و تصدیق می‌باشد که بطور جداگانه به وسائل جانبی متصل است. هر کانال، همچنین دارای آدرس و ثبات شمارش کلمه خود در کنترل کننده DMA است. میان کانال‌ها می‌توان اولویتی هم برپا کرد بطوری که کانال با اولویت بالاتر قبل از کانال‌های با اولویت پائین‌تر سرویس داده شود.

انتقال DMA در بسیاری از کاربردها بسیار مفید است. این روش در انتقال سریع اطلاعات بین دیسک‌های مغناطیسی و حافظه یکار می‌رود. همچنین برای بهنگام کردن صفحه نمایش در پایانه محاوره‌ای نیز مفید است. معمولاً، تصویری از نمایش صفحه پایانه<sup>۲</sup> 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.

## ۱۱-۷ پردازنده ورودی و خروجی<sup>۳</sup>

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 وسائل I/O را فراهم می‌سازد تخصیص داد. یک

1- Request Line

2- Terminal

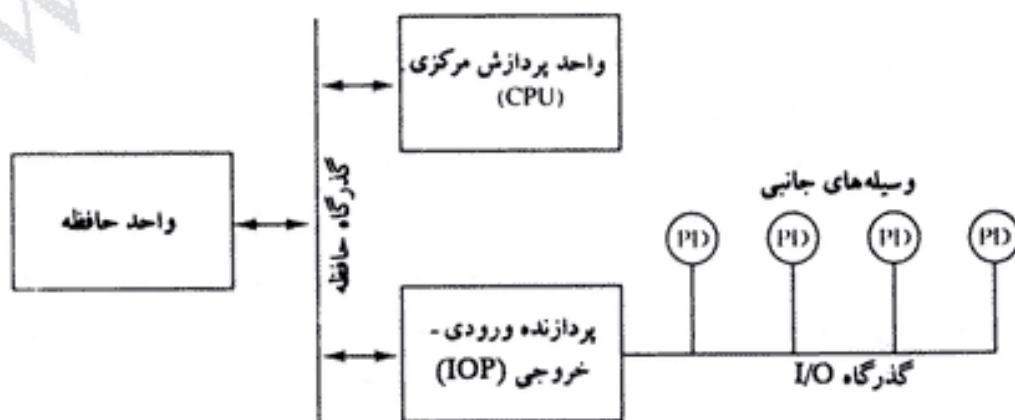
3- Input - Output Processor

پردازنده ورودی - خروجی (IOP) ممکن است بعنوان پردازنده‌ای با مراجعه مستقیم به حافظه طبقه‌بندی شود که با وسیله‌های I/O تبادل اطلاعات می‌کند. درین پیکربندی، سیستم کامپیوتر می‌تواند 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 می‌کند، و CPU را از علمیات سریار موجود در انتقال آزاد می‌سازد. یک پردازنده که با پایانه‌های دوردست از طریق تلفن و سایر وسایل مخابراتی بصورت سری تبادل اطلاعات می‌کند، پردازنده تبادل داده<sup>۱</sup> (DCP) نامیده می‌شود.

IOP مشابه CPU است جز اینکه برای انجام جزئیات پردازش I/O طراحی شده است. برخلاف کنترل‌کننده DMA که باید کلاً بوسیله CPU تنظیم شود، IOP می‌تواند دستورالعمل‌های خود را برداشت و اجرا کند. دستورالعمل‌های IOP بخصوص برای انتقال IOP طراحی شده‌اند. بعلاوه IOP می‌تواند پردازش دیگری مانند محاسبات، منطق، انشعاب و ترجمه کرد را نیز انجام دهد.

بلک دیاگرام یک کامپیوتر با دو پردازنده در شکل ۱۱-۱۹ دیده می‌شود. واحد حافظه یک موقعیت مرکزی را اشغال کرده است و بصورت دستیابی مستقیم به حافظه می‌تواند 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 بر عهده دارد. از این پس IOP بطور مستقل از CPU کار کرده و انتقال داده را از وسیله خارجی به حافظه ادامه می‌دهد.

قالب‌های داده وسایل جانبی با قالب‌های 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 ۱۱-۱۹ بلک دیاگرام کامپیوتر با پردازنده I/O

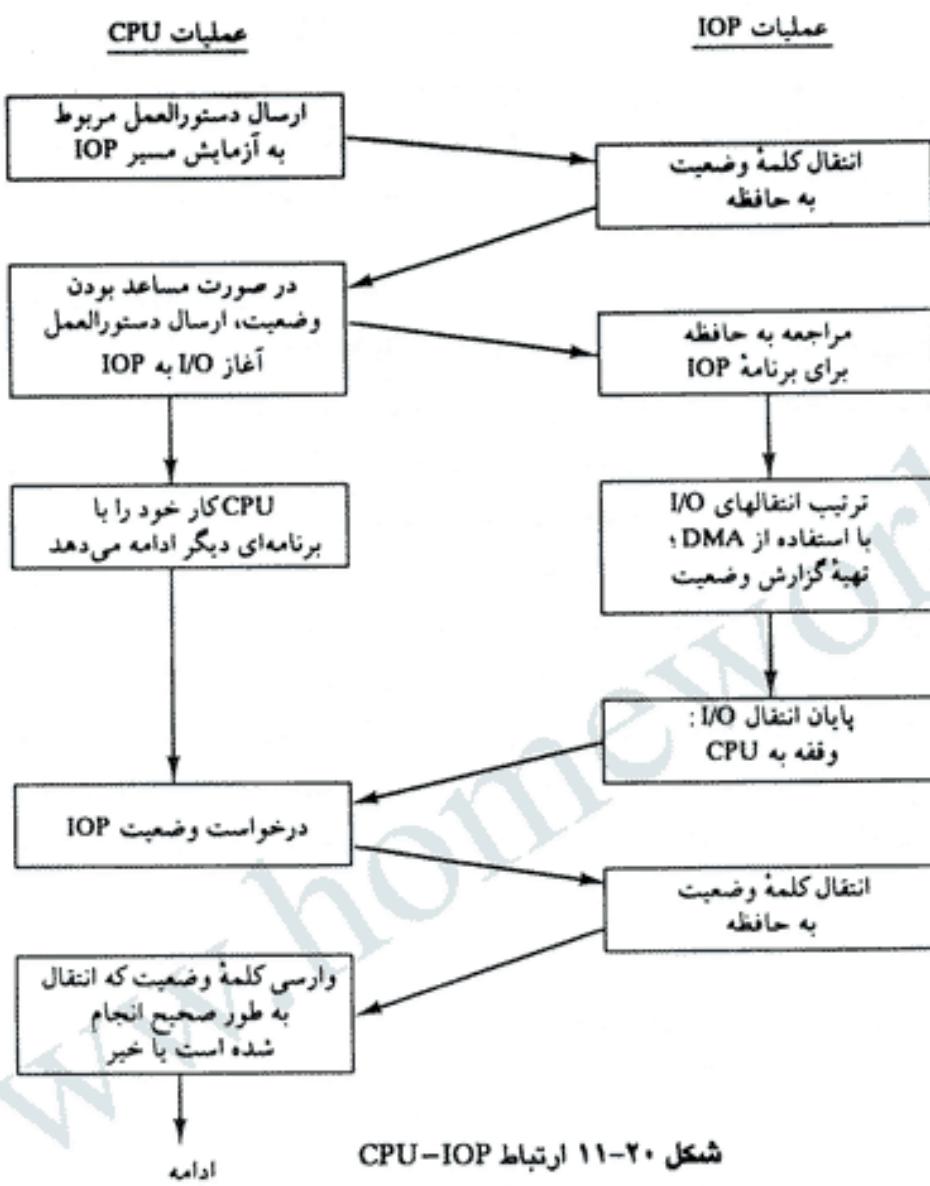
CPU برنامه خود را اجرا می کند داده ها در IOP با سرعت و ظرفیت بیت وسیله ورودی جمع می شوند. پس از اینکه داده ها در کلمه حافظه کنار هم قرار گرفتند<sup>۱</sup>، مستقیماً از IOP با سرقت یک سیکل حافظه از CPU به حافظه منتقل می شوند. بطور مشابه یک کلمه حافظه انتقال یافته از حافظه به IOP از IOP به وسیله خروجی با سرعت و ظرفیت وسیله منتقل می گردد.

ارتباط بین IOP و وسائل 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)<sup>۲</sup>، هر پردازنده از دیگری مستقل است و هر پردازشگر می تواند عملی را آغاز کند. در اکثر سیستم های کامپیوتی، 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 تعیین گیری بر انواع فعالیت های I/O را فراهم می کنند. IOP به نوعه خود با استفاده از وقه 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 وارسی 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 باید وسیس جزئیات انتقال را به CPU واگذار می کند.

### ارتباط CPU - IOP

تبادل اطلاعات بین CPU و IOP ممکن است بسته به کامپیوت 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 فلوچارت شکل ۱۱-۲۰ 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 وارسی 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 باید. اکنون CPU می تواند با برنامه دیگری کار خود را دنبال کند در حالیکه IOP با برنامه I/O مشغول است.



هر دو برنامه توسط انتقال DMA به حافظه مراجعه می‌کنند. وقتی که IOP اجرای برنامه خود را پایان می‌دهد، یک تقاضای وقفه به CPU ارسال می‌کند. CPU با صدور یک دستورالعمل جهت خواندن گزارش وضعیت واصله از IOP 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 موقتی آمیز و بدون خطا پایان یافته است.

IOP از کلیه انتقال داده‌ها بین چند واحد I/O و حافظه مراقبت می‌کند در حالیکه CPU برنامه دیگری را پردازش می‌نماید. IOP و CPU، 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 انتقال‌های حافظه توسط IOP بماند.

### کانال I/O در IBM 370

پردازنده I/O در کامپیوتر IBM370، کانال<sup>۱</sup> 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 دارد. 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 پلکسر 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 ۱۱-۲۱ قالب کلمات مربوط به عملکرد کانال را نشان می‌دهد.

آدرس وسیله	آدرس کانال	کد عمل
------------	------------	--------

(الف) قالب دستور العمل های I/O

شمارش	وضعیت	آدرس	کلید
-------	-------	------	------

(ب) قالب کلمه وضعیت کانال

شارش	برچم‌ها	آدرس داده	کد فرمان
------	---------	-----------	----------

(ج) قالب کلمه فرمان کانال

شکل ۱۱-۲۱ قالب‌های کلمات مربوط به I/O در IBM 370

<sup>۱</sup>- Channel

قالب دستورالعمل I/O سه میدان دارد: کد عملیات، آدرس کanal، و آدرس وسیله. سیستم کامپیووتر ممکن است دارای تعدادی کanal باشد، که به هر یک آدرس اختصاص یافته است. بطور مشابه، هر کanal ممکن است به چند وسیله وصل شده باشد که به هر وسیله آدرس اختصاص می‌یابد. کد عملیات یکی از هشت دستورالعمل I/O را مشخص می‌کند: شروع I/O، آزمایش I/O، پاک کردن I/O، توقف I/O، توقف وسیله، آزمایش کanal، و ذخیره شماره شناسایی کanal. کanalی که آدرس دهی شده به هر یک از دستورالعمل‌های I/O پاسخ می‌دهد و آن را اجرا می‌کند. همچنین یکی از چهار کد وضعیتی ممکن را در یک ثبات پردازنده بنام PSW (کلمه وضعیت پردازنده) قرار می‌دهد. CPU می‌تواند با وارسی 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 می‌کنند که کanal مشغول است یا خیر، عملیاتی است یا خیر.

قالب کلمه وضعیت کanal در شکل ۱۱-۲۱ (ب) نشان داده شده است. این کلمه همیشه در مکان 64 حافظه ذخیره می‌شود. میدان کلید<sup>۱</sup>، یک مکانیزم حفاظتی است که برای جلوگیری از مراجعه غیرمجاز از سوی یک کاربر به اطلاعات متعلق به کاربر دیگر یا سیستم عامل بکار می‌رود. میدان آدرس در کلمه وضعیت، آدرس آخرین کلمه استفاده شده بوسیله کanal را نشان می‌دهد. میدان شمارش، تعداد باقیمانده را پس از پایان انتقال نشان می‌دهد. اگر انتقال با موفقیت انجام شود میدان شمارش حاوی مقدار صفر خواهد بود. میدان وضعیت شرایط وسیله و کanal و هر گونه خطای را که در حین انتقال رخ داده باشد نشان می‌دهد.

تفاوت بین دستورالعمل‌های شروع I/O و شروع سریع I/O این است که اجرای دومی به صرف زمان کمتری از سوی CPU نیاز دارد. وقتی که کanal یکی از این دو دستورالعمل را دریافت کند، به خانه حافظه 72 برای برداشت اولین کلمه فرمان کanal (CCW) مراجعه می‌کند. قالب کلمه فرمان کanal در شکل ۱۱-۲۱ (ج) نشان داده شده است. میدان آدرس داده اولین آدرس بافر حافظه را مشخص می‌کند و میدان شمارش تعداد بایت‌هایی که در انتقال شرکت دارند را معین می‌نماید. میدان فرمان مشخص کننده یک عمل I/O و بیت‌های پرچم اطلاعات اضافی دیگر را برای کanal فراهم می‌کنند. میدان فرمان متعلق به کد عملی است که یکی از شش نوع کار اصلی I/O زیر را مشخص می‌نماید.

- ۱- نوشن - 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 می‌رود، مانند به عقب برگرداندن نوار، یا تغییر مکان مکانیزم دستیابی به دیسک<sup>۲</sup>
- ۵- درک - به کanal اطلاع می‌دهد تا کلمه وضعیت کanal 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 ۱۱-۳ 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.

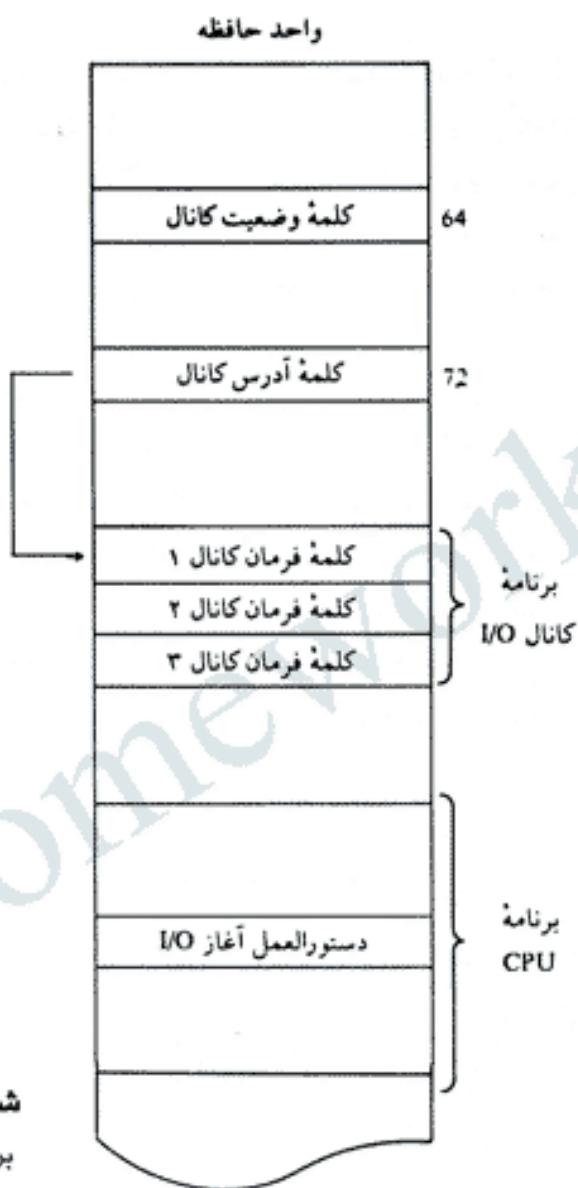
پرچم اول در اولین کلمه فرمان برابر 1 قرار داده می‌شود تا زنجیرهای شدن داده‌ها را مشخص کند. این امر منجر به ترکیب 60 بایت فرمان اول با 20 بایت فرمان بعدی آن و تشکیل یک رکورد 80 بایتی می‌شوند. این 80 بایت حتی اگر از دو بخش حافظه استفاده شده باشند بدون هرگونه جداسازی یا فاصله بر روی نوار نوشته می‌شوند. پرچم دوم در فرمان دوم 1 می‌گردد تا مشخص کننده زنجیرهای شدن فرمانها<sup>۱</sup> 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. هرگاه همه پرچم‌ها صفر باشند، نشان دهنده پایان عملیات I/O برای وسیله خاص مورد نظر است.

نقشه‌ای از حافظه که همه اطلاعات مربوط به پردازنده I/O را نشان می‌دهد در شکل ۱۱-۲۲ 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 ۱۱-۳ مثالی از برنامه کانال برای IBM 370

شارش	برچم‌ها	آدرس	وقد
نوشن نوار	4000	100000	60
نوشن نوار	6000	010000	20
نوشن نوار	3000	000000	40

#### 1- Command chaining



شکل ۱۱-۲۲ محل اطلاعات در حافظه  
برای عملیات I/O در IBM 370

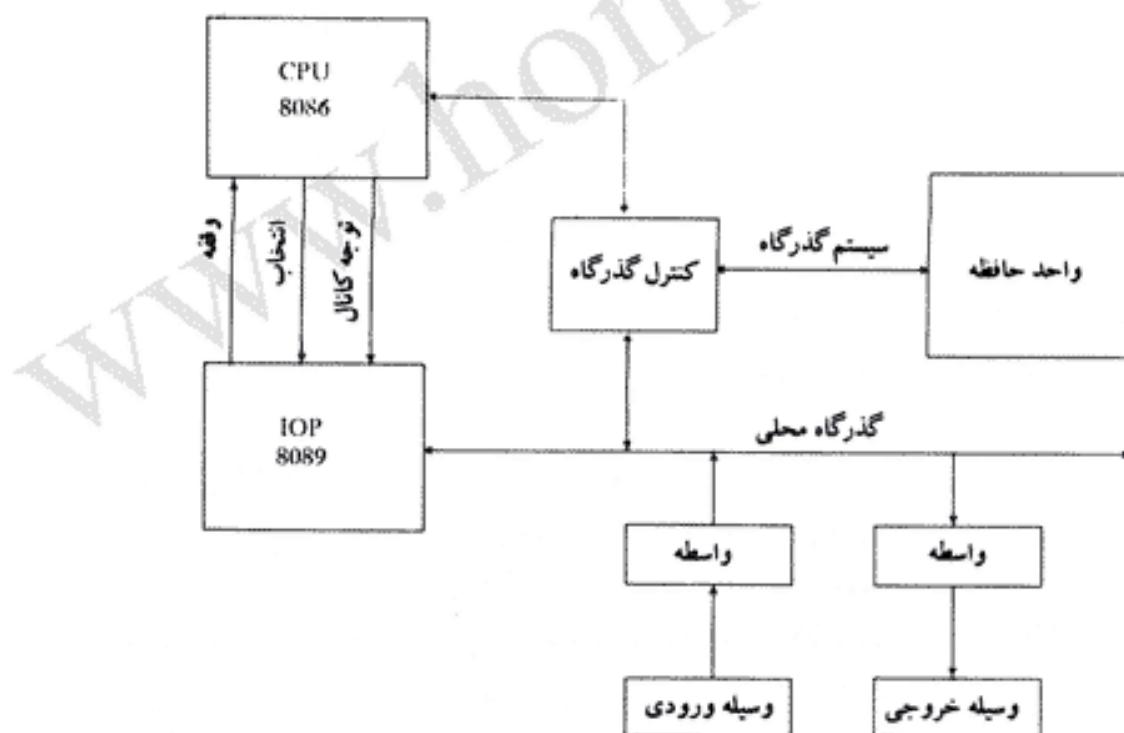
### Intel 8089 مدل IOP

پردازنده I/O مدل 8089 در یک بسته مدار مجتمع 40 پایه‌ای قرار دارد. در داخل 8089 دو واحد مستقل به نام کانال وجود دارد. هر کانال مشخصات و ویژگی‌های عمومی یک واحد پردازنده را همراه با امکان یک کنترل کننده با دستیابی مستقیم به حافظه<sup>۱</sup> (DMAC) دارد. 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 موردنظر را معین می‌نماید، اقدام به یک عمل I/O می‌کند. پردازنده 8089 این پیام را از حافظه می‌خواند، عمل را انجام می‌دهد، و

1- Direct Memory Access Controller

پس از اتمام کار به CPU اطلاع می دهد.

برخلاف کانال 370 IBM، که فقط دارای شش فرمان اصلی I/O است، پردازنده IOP 8089 پنجاه دستور العمل 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 دستور العمل ها خصوصاً برای انجام یک عمل ورود و خروج کارا انتخاب شده اند. این مجموعه دستور العمل 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 دستور العمل 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 کارآبی خوبی دارد. یک سیستم ریز کامپیوتری که از جفت مدار مجتمع 8086/8089 استفاده می کند در شکل ۱۱-۲۳ 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<sup>۱</sup> نامیده می شود، از حافظه مشترک 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<sup>۲</sup> ارتباط برقرار می کند خط انتخاب که بوسیله CPU بکار می رود.



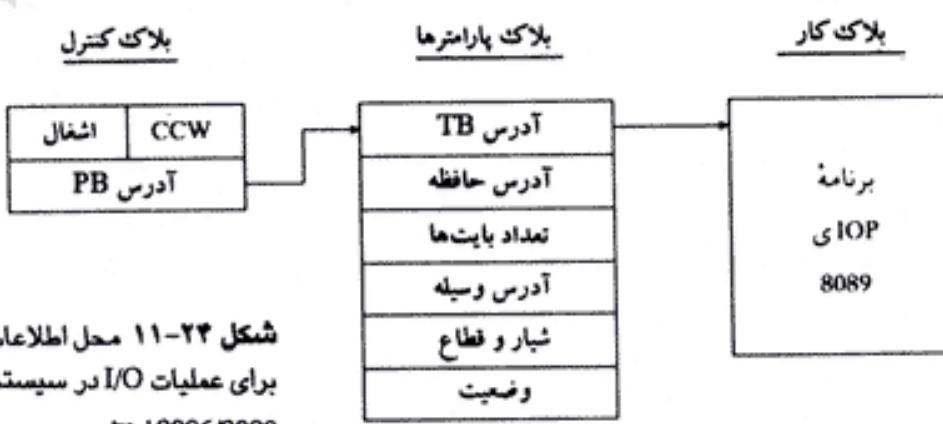
شکل ۱۱-۲۳ بلاک دیاگرام سیستم ریز کامپیوتر 8086/8089 اینتل

می‌رود برای انتخاب یکی از دو کانال در 8089 بکار می‌رود. IOP توجه CPU را با ارسال یک تقاضای وقفه جلب می‌کند.

CPU و IOP، با نوشتن پیام برای یکدیگر در سیستم حافظه، باهم ارتباط برقرار می‌کنند. CPU مکان پیام را آماده کرده و با فعال کردن خط توجه کانال IOP را آگاه می‌سازد. IOP پیام را خوانده، عملیات I/O درخواستی را انجام می‌دهد، و برنامه کانال مناسب را اجرا می‌نماید. وقتی که کانال برنامه خود را پایان رساند، یک تقاضا وقفه برای CPU ارسال می‌دارد.

این روش ارتباط، متشکل از برنامه‌های تقسیم شده‌ای است که بلاک<sup>۱</sup> نامیده شده و طبق شکل ۱۱-۲۴ در حافظه ذخیره می‌شوند. 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 می‌شود. پرچم اشغال بودن مشخص می‌کند که آیا IOP مشغول است یا آماده انجام یک عمل I/O جدید می‌باشد. CCW (کلمه فرمان کانال)<sup>۲</sup> توسط CPU برای مشخص کردن نوع عملی که IOP باید انجام دهد معین می‌شود. CCW در 8089 مفهوم مشابهی با کلمه فرمان در کانال IBM ندارد. در اینجا CCW بیشتر شبیه یک دستور العمل I/O است که یک عمل را برای IOP مشخص می‌کند، بلاک پارامتر حاوی داده‌های متغیری است که برنامه IOP برای اجرای کار خود از آن استفاده می‌کند و بلاک کار<sup>۳</sup> (تکلیف) حاوی برنامه واقعی است که باید در IOP اجرا شود.

CPU و IOP از طریق کنترل بلاک‌های پارامترها با یکدیگر کار می‌کنند. CPU پس از تست پرچم اشغال و اطمینان از در اختیار بودن IOP، استفاده از حافظه مشترک را بندست می‌گیرد. سپس اطلاعات لازم را در بلاک پارامترها پرمی‌کند و یک فرمان "شروع عمل" در CCW می‌نویسد. پس از برپایی بلاک تبادل اطلاعات<sup>۴</sup>، CPU سیگнал "توجه کانال" را فعال می‌کند تا IOP را برای آغاز عمل I/O مطلع سازد. سپس CPU با برنامه دیگری بکار اダメه می‌دهد. IOP با قراردادن آدرس بلاک کنترل در شمارنده



1- Block

2- Channel Command Word

3- Task Block

4- Communication Block

برنامه اش، به سیگنال توجه کانال پاسخ می دهد. IOP به بلاک کنترل مراجعه کرده و پرچم اشغال را ۱ می کند. پس از آن، عمل موجود در CCW را وارسی می نماید. سپس آدرس PB (بلاک پارامتر) و آدرس TB (بلاک تکلیف)<sup>۱</sup> را به ثبات داخلی IOP منتقل می سازد. IOP با استفاده از اطلاعات موجود در بلاک پارامتر شروع به اجرای برنامه در بلاک تکلیف می کند. واردہ های موجود در بلاک پارامترها به وسیله I/O بستگی دارد. پارامترهای لیست شده در شکل ۱۱-۲۴ برای انتقال داده به دیسک مغناطیسی و یا از آن، مناسبند. آدرس حافظه، آدرس شروع یک بافر حافظه است. شمارش بایت تعداد بایت های انتقال یافته را معین می نماید. آدرس وسیله، وسیله I/O خاصی که قرار است بکار گرفته شود را مشخص می کند. شماره های شیار<sup>۲</sup> و قطاع داده را روی دیسک قرار می دهد. وقتی که عمل I/O پایان یابد، IOP بیت های وضعیت خود را در مکان کلمه وضعیت بلاک پارامتر قرار می دهد و به CPU سیگنال وقفه می فرستند. CPU به کلمه وضعیت رجوع کرده و انتقال موققیت آمیز را وارسی می نماید.

## ۱۱-۸ تبادل اطلاعات سری

یک پردازنده تبادل داده ها یک پردازنده I/O است که داده را از پایانه های متعدد دور از هم که از طریق خطوط تلفن و دیگر خطوط ارتباطی به یکدیگر متصلند توزیع و یا جمع آوری می کند. این پردازنده نوعی پردازنده خاص I/O است که برای ارتباط مستقیم با شبکه های ارتباطی طراحی شده است. یک شبکه ارتباطی ممکن است از هر نوع وسیله متنوعی مانند چاپگر، وسایل نمایش محاوره ای، سنسورهای دیجیتال یا یک مرکز کامپیوتر دور دست مشکل باشد. با استفاده از یک پردازنده انتقال داده<sup>۳</sup>، کامپیوتر می تواند بطور متناوب به بخشی از هر تقاضای شبکه سرویس بدهد و بنابراین رفتار ظاهری آن بصورتی باشد که به کاربران متعدد بطور همزمان سرویس می دهد. با این ترتیب کامپیوتر قادر خواهد بود تا بطور موثری در محیط های تقسیم زمانی<sup>۴</sup> 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 می کند. تمام وسایل جانبی از گذرگاه بطور مشترک استفاده می کنند و از آن برای انتقال اطلاعات به و یا از پردازنده I/O استفاده می کنند. یک پردازنده ارتباطی (انتقال داده) از طریق یک جفت سیم با هر پایانه ارتباط برقرار می نماید. هر دو نوع اطلاعات داده و کنترل به فرم سری در آمده که این موجب سرعت انتقال بسیار کنترلی می شود. وظیفه پردازنده انتقال داده، ارسال و جمع آوری اطلاعات دیجیتال به و یا از هر پایانه، تعیین اینکه اطلاعات از نوع داده است یا کنترلی و پاسخ به تمام تقاضاهای طبق رویه های از پیش تعیین شده است. مسلمًا 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 پایانه های 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 کامپیووترها با سیگنال های دیجیتال با هم تبادل اطلاعات می کنند، نوعی تبدیل باید صورت گیرد. مبدل ها، دستگاه داده<sup>۱</sup>، کوپل کننده صوتی<sup>۲</sup> و یا مودم<sup>۳</sup> (مخفف مودولاتور و دمودولاتور) خوانده می شوند. مودم سیگنال دیجیتال را به علائم صوتی برای ارسال بر روی خط تلفن تبدیل می کند. روش های مدولاسیون مورد استفاده و نیز درجه کیفیت رسانه های ارتباطی و سرعتهای انتقال در بین آنها متفاوت است. بسته به روش ارسال پایانه دوردست، خط ارتباطی ممکن است به یک واسطه همزمان یا غیر همزمان متصل باشد. واسطه غیر همزمان، داده های سری را با بیت های شروع و توقف در هر کاراکتر مطابق شکل ۱۱-۷ دریافت می کند. این نوع واسطه مشابه واحد واسطه ارتباطی غیر همزمان در شکل ۱۱-۸ است.

انتقال همزمان (همگام) بیت شروع و توقف<sup>۴</sup> را برای کاراکترها استفاده نمی کند، و بنابراین از ارتباط جهت تبادل داده ها بصورت کاراکتری استفاده می شود. وسایلی که سرعت بالا دارند از انتقال همزمان برای رسیدن به این کارآیی استفاده می کنند. مودم هایی که در انتقال همزمان بکار می روند، دارای ساعت درونی بوده و این ساعت ها با فرکانس انتقال در خط ارتباط تنظیم می شوند. برای عملکرد صحیح مودم، لازم است ساعت در فرستنده و گیرنده در تمام مدت همگام باشند. با این وجود خط ارتباطی فقط حاوی بیت های داده است که از آنها اطلاعات ساعت باید استخراج شود. همگامی فرکانس توسط مودم گیرنده از گذارهای سیگنال که در داده های دریافتی رخ می دهد حاصل می گردد. هر شیفت فرکانسی که ممکن است بین ساعت های گیرنده و فرستنده رخ دهد بطور مداوم با نگهداشت ساعت فرستنده در فرکانس رشته بیت های رسیده تنظیم می شود. مودم داده دریافتی را همراه با ساعت فرستنده به واحد واسطه می فرستد. واسطه یا پایانه در سمت فرستنده هم از اطلاعات ساعت مودم خود استفاده می کند. به این ترتیب در فرستنده و گیرنده سرعت بیت یکسانی برقرار می شود.

برخلاف ارسال غیر همگام که در آن هر کاراکتر را می توان جداگانه همراه با بیت های شروع و توقف آنها ارسال کرد، انتقال همگام برای حفظ همگامی باید پیام پیوسته ای را ارسال نماید. پیام شامل گروهی از بیت های ارسال شده سری مانند یک بلاک از داده است. کل بلاک همراه با کاراکتر های کنترلی خاص در ابتداء و انتهای بلاک ارسال می شود. کاراکتر های کنترل در ابتدای بلاک اطلاعات لازم را برای تفکیک بیت های دریافتی به صورت کاراکتر های مستقل فراهم می کنند.

یکی از وظایف پردازنده انتقال داده وارسی خطاهای ارسال است. یک نوع خطرا را با وارسی بیت توازن در هر کاراکتر رسیده می توان تشخیص داد. روش دیگر که در پایانه های غیر همگام با اپراتوری انسانی بکار می رود انعکاس کاراکتر است. کاراکتری که از صفحه کلید به کامپیووتر ارسال می شود بوسیله پردازنده تشخیص داده شده و سپس به پایانه چاپگر منعکس می شود. اگر کاراکتر چاپ شده همان

1- Data set

2- Acoustion Coupler

3- Modem

4- Start-Stop

کاراکتری که از فشار کلید حاصل شده است نباید، اپراتور در می باید 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 وارسی تجمعی طولی<sup>۱</sup> (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 خطاء 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 وارسی خطاهای ارسال اطلاعات، وارسی تجمعی چرخشی<sup>۲</sup> (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 گیت های OR انحصاری به فرستنده پسخورد (برگشت<sup>۳</sup>) 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 طرفه<sup>۴</sup> 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 رادیویی 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 باید، نقش مودم ها برای معکوس کردن جهت انتقال تعویض می گردد. زمان لازم برای تعویض خط سیستم نیمه دو طرفه<sup>۵</sup> از یک جهت به جهت دیگر زمان تحویل خوانده 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 غیرهمپوش<sup>۶</sup> 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

3- Feedback

6- Nonoverlap

2- Cyclic Redundancy Check

5- Turnaround

ارتباط داده<sup>۱</sup> نامیده می شود. ارسال منظم اطلاعات در یک خط ارتباط داده توسط یک قرارداد<sup>۲</sup> 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 بر حسب روش کادریندی پیام ها<sup>۳</sup> 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. متدائل ترین کد مورد استفاده کد اسکی ASCII (کد استاندارد آمریکایی برای تبادل اطلاعات) است. این کد ۷ بیتی است و بیت هشتم برای توازن بکار می رود. کد دارای 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 سمبول های خاص است. لیستی از کاراکترهای ASCII در جدول ۱۱-۱ قابل ملاحظه است. کاراکترهای کنترل برای مسیردهی داده ها، آرایش متن با قالب مورد نظر، و صفحه آرایی<sup>۴</sup> متن چاپی بکار می روند. کاراکترهایی که ارسال داده ها را کنترل می کنند، کاراکترهای کنترل ارتباطات<sup>۵</sup> خوانده می شوند. این کاراکترها در جدول ۱۱-۴ لیست شده اند. هر کاراکتر ۷ بیتی است و با یک سمبول سه حرفی معروفی شده است. نقش هر کاراکتر در کنترل ارسال داده بطور خلاصه در ستون نقش در جدول آمده است.

جدول ۱۱-۴ کاراکترهای کنترل ارتباطات اسکی

معنی	عملکرد	سبل	کد
بکار و همگام	در حالت بیکار، همگامی تولید می کند	SYN	0010110
آغاز سرآیند	سرآیند بلاک پیام	SOH	0000001
آغاز متن	قبل از بلاک متن قرار گیرد	STX	0000010
پایان متن	در پایان بلاک متن قرار گیرد	ETX	0000011
پایان ارسال	عمل ارسال را پایان می دهد	EOT	0000100
تصدیق	تصدیق ثبت	ACK	0000110
تصدیق منفی	تصدیق منفی	NAK	0010101
استعلام	استعلام روش بودن پایانه	ENQ	0000101
پایان بلاک ارسال	پایان بلاک داده	ETB	0010111
کنترل رابط داده	کاراکتر کنترلی خاص	DLE	0010000

1- Data Link

2- Protocol

3- Message-Framing

4- Layout

5- Communication Control Character

کاراکتر SYN نقش همگام ساز را بین فرستنده و گیرنده بعهده 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 می کند. وقتی که فرستنده ارسال کاراکتر 8 بیتی را آغاز می کند، ابتدا چند کاراکتر را فرستاده و سپس، اقدام به فرستادن پیام واقعی می کند. رشته بیت های متوالی اولیه توسط گیرنده پذیرفته شده و به منظور یافتن کاراکتر SYN وارسی می شوند. به بیان دیگر، با هر پالس ساعت، گیرنده هشت بیت آخر دریافتی را وارسی می کند. اگر این هشت بیت با کاراکتر SYN مطابقت نکند، گیرنده بیت بعدی را می پذیرد، بیت بالا رتبه قبلی را حذف می نماید، و مجدداً هشت بیت آخر را برای یافتن کاراکتر SYN چک می کند. این عمل پس از هر پالس ساعت تکرار شده و بیت ها تا یافتن کاراکتر SYN دریافت می شوند. به محض تشخیص کاراکتر SYN، گیرنده کاراکتر را قادر نبودی می کند. از این پس گیرنده هر هشت بیت را شمارش کرده و آنرا بعنوان یک کاراکتر در نظر می گیرد. معمولاً گیرنده دو کاراکتر SYN متوالی را چک می کند 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 نمی کند. رشته پیوسته ای از کاراکتر SYN را ارسال می نماید. گیرنده این کاراکترها را بعنوان شرایطی برای همگام سازی خط می شناسد و به یک حالت بیکار و همگام می رود. در این حالت، هر دو واحد، همزمانی بیت و کاراکتر را حتی اگر اطلاعات معنی داری مخابره نشود حفظ می کنند.

پیام ها از طریق خط ارتباط داده همراه با قالب متشکل از میدان سرآیند<sup>۱</sup>، میدان متن، میدان وارسی خط ارسال می شوند. نمونه ای 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 می کنند. بدنبال کاراکترهای SYN، سرآیند قرار دارد، که با کاراکتر SOH<sup>۲</sup> شروع می شود (شروع سرآیند). سرآیند از اطلاعات آدرس و کنترلی تشکیل شده است. کاراکتر STX سرآیند را پایان داده و شروع متن را معین می نماید. پخش متن پیام از نظر طول متغیر بوده و می تواند حاوی هر کاراکتر اسکی یجز کاراکترهای کنترلی باشد. میدان متن با ETX پایان می پذیرد. آخرین میدان، یک کاراکتر وارسی بلاک<sup>۳</sup> (BCC) است که برای وارسی خط بکار می رود. این میدان معمولاً وارسی تجمعی طولی (LRC) یا وارسی تجمعی چرخشی (CRC) است. گیرنده پیام را دریافت کرده و BCC خود را محاسبه می کند. اگر BCC ارسالی با BCC محاسبه شود

SYN	SYN	SOH	سرآیند	STX	من	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<sup>۱</sup> (NAK) واکنش نشان می‌دهد. پیام، سپس مجدد ارسال و وارسی می‌شود. معمولاً قبل از اینکه خط اشکال دار فرض شود، ارسال مجدد پیام چند بار تکرار می‌شود. هنگامی که BCC ارسالی؛ محاسبه شده بوسیله گیرنده مطابقت نماید، پاسخ تصدیق مثبت بوده و کاراکتر ACK استفاده می‌شود.

### مثال ارسال داده

برای درک عملکرد یک پردازنده انتقال داده، اجازه بدھید روشی را که یک پایانه و پردازنده با هم ارتباط برقرار می‌کنند مثال بزنیم. تبادل اطلاعات بین واحد حافظه و CPU مشابه با هر پردازنده O/I است. نمونه پیامی که ممکن است از یک پایانه به پردازنده ارسال شود در جدول ۱۱-۵ دیده می‌شود. نگاهی به این پیام مشخص می‌نماید که تعدادی کاراکترهای کنترلی برای شکل دادن به پیام بکار رفته است. هر کاراکتر، از جمله کاراکترهای کنترلی بصورت کد دو دویی هشت بیتی شامل ۷ بیت کد اسکی بعلاوه یک بیت توازن فرد در با ارزشترین مکان، بطور سری ارسال می‌شوند. برای همگامی فرستنده و گیرنده دو کاراکتر SYN بکار رفته است. سرآیند با کاراکتر SOH شروع می‌شود و به دو کاراکتر که آدرس پایانه را معین می‌کنند ختم می‌گردد. در این مثال خاص، آدرس T<sub>4</sub> است، ولی عموماً می‌تواند مجموعه‌ای از دو یا چند کاراکتر گرافیکی باشد. کاراکتر STX سرآیند را ختم کرده و شروع ارسال متن را معین می‌نماید. داده متن در اینجا "request of balance number 1234" است. تک تک کاراکترهای این پیام

جدول ۱۱-۵ نمونه پیام ارسالی از یک پایانه به پردازنده

کد	نام	شرح
0001 0110	SYN	اولین کاراکتر Sync
0001 0110	SYN	دومین کاراکتر Sync
0000 0001	SOH	آغاز سرآیند
0101 0100	T	آدرس پایانه T <sub>4</sub> 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 نشده‌اند زیرا فضای بسیاری را اشغال می‌کنند. با این وجود، باید توجه داشت که هر کاراکتر در پیام دارای کد 8 بیتی است و هر بیت بطور سری ارسال می‌شود. کاراکتر کنترل EXT پایان کاراکترهای متن را مشخص می‌نماید. کاراکتر بعدی که بدنبال ETX آمده است یک کاراکتر وارسی طولی، LRC، است هر بیت در این کاراکتر بیت توازنی است که از محاسبه تمام بیت‌های واقع در یک ستون در بخش کد از جدول محاسبه می‌شود.

پردازنده انتقال داده، این پیام را دریافت کرده و به تحلیل آن می‌پردازد. پردازنده تشخیص می‌دهد که پایانه T4 است و متن پیام را ذخیره می‌کند. در حین دریافت کاراکترها، پردازنده توازن را در هر کاراکتر وارسی کرده و توازن طولی را نیز محاسبه می‌کند. LRC محاسبه شده با LRC کاراکتر دریافتی مقایسه می‌شود. اگر هر دو با هم مطابقت داشته باشند، یک تصدیق مثبت (ACK) به پایانه بازگردانده می‌شود. اگر عدم تطابق وجود داشته باشد، یک تصدیق منفی به پایانه برگردانده می‌شود، که یک ارسال مجدد از همان بلاک را آغاز خواهد کرد. اگر پردازنده پیام را بدون خطای بیابد، آنرا به حافظه انتقال داده و CPU را وقف می‌دهد. هنگامی که CPU وقفه را تصدیق نماید، پیام را تحلیل کرده و متن پیام را برای پاسخ به تقاضا آماده می‌کند. CPU یک دستور العمل را برای پردازنده انتقال داده می‌فرستد تا پیام را به پایانه ارسال دارد. نمونه پاسخ از پردازنده به پایانه در جدول ۱۱-۶ لیست شده است. پس از دو کاراکتر SYN، پردازنده پیام قبلی را با یک کاراکتر ACK تصدیق می‌کند. خط با کاراکتر SYN در حالت بیکاری قرار

جدول ۱۱-۶ نمونه پیام 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 می‌کند
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 می‌دهد
.	\$100.00	
.		
1011 0000		پایان ارسال متن
1000 0011	ETX	
1101 0101	LRC	کاراکتر توازن طولی

می‌گیرد و منتظر دریافت پاسخ می‌ماند. پیام دریافتی از CPU توسط پردازنده با گنجاندن کاراکترهای کنترلی لازم در ابتداء 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، جمع آوری 8 بیت از یک کاراکتر زمان معینی لازم دارد. در حین این جمع آوری پردازنده سایر خطوط ارتباطی را مولتی‌پلکس می‌نماید و هر یک را به نوبت سرویس می‌دهد. سرعت دورترین ترمینال‌ها در مقایسه با سرعت پردازنده فوق العاده کم است. این خصوصیت اجازه می‌دهد تا چند کاربر، مولتی‌پلکس شده و به کارآیی بهتری در سیستم اشتراک زمانی<sup>۱</sup> 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<sup>۲</sup>

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 بیتها بیش از یک داده باشد که باید ب بدون ارجاع به هر کد خاصی پذیرفته شوند گوئیم که حاوی داده‌های شفاف<sup>۳</sup> 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<sup>۴</sup> 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 00010000 DLE برابر در بخش متن پیام ظاهر شود، فرستنده یک DLE دیگر بدنیال آن می‌گنجاند. گیرنده تمام DLE ها را حذف و الگویی هشت بیتی بعدی را وارسی می‌کند. اگر این الگو یک DLE دیگر باشد، گیرنده آن را

1- Time – Sharing  
4- Data link escape

2- Data Transparency

3- Transparent data

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 شفاقت 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 شفاقت داده و سایر مشکلات مربوط به قرارداد مبتنی بر کاراکتر بکار رفته است، قرارداد مبتنی بر بیت است.

### قرارداد مبتنی بر بیت<sup>۱</sup>

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 وارسی خطاست. مرزهایی کادر از عدد هشت بیتی خاصی که پرچم نامیده می‌شود معین می‌گردد. مثالی‌هایی از قرارداد مبتنی بر بیت کنترل، ارتباط داده همگام<sup>۲</sup>، SDLC<sup>۳</sup>، است که بوسیله IBM، قرارداد کنترل ارتباط داده سطح بالا<sup>۴</sup> HDLC بکار رفته بوسیله سازمان بین‌المللی استاندارد و رویه کنترل پیشرفته مخابر داده، ADCCP<sup>۵</sup>، مورد استفاده انتیتو 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<sup>۶</sup> خوانده می‌شود. ایستگاه دیگر ایستگاه فرعی<sup>۷</sup> یا ثانویه نامیده می‌شود. قراردادهای مبتنی بر بیت وجود یک ایستگاه اصلی و یک یا چند ایستگاه فرعی را فرض می‌کنند. 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 ۱۱-۲۶ نشان داده شده است. کادر با پرچم هشت بیتی ۰۱۱۱۱۱۱۰ و بدنبال آن رشته آدرس و کنترل است. میدان اطلاعات محدودیتی در قالب ندارد و هر طولی می‌تواند داشته باشد. میدان وارسی کادر یک رشته CRC (وارسی چرخشی<sup>۸</sup>) مورد استفاده در کشف خطأ 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

برچم 01111110	آدرس 8 بیت	کنترل 8 بیت	اطلاعات هر تعداد بیت	وارسی کادر 16 بیتی	برچم 01111110
------------------	---------------	----------------	-------------------------	-----------------------	------------------

شکل ۱۱-۲۶ قالب کادر برای قرارداد مبتنی بر بیت

رشته‌های متواالی از ۱ ها دنبال شود. وقتی دو کادر بدنبال 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 ۱۵ بیت متواالی ۱، یا بیشتر، خط فعال تصور می‌شود. کادرهایی که دارای پیام‌های کنترل معین نیستند بدون میدان اطلاعات ارسال می‌گردند. یک کادر باید حداقل ۳۲ بیت بین دو پرچم برای میدان‌های آدرس، کنترل، و وارسی کادر داشته باشد. طول حداقل به شرایط کanal ارتباطی و قابلیت‌های آن در ارسال پیام‌های طولانی بدون خطأ وابسته است.

برای جلوگیری از وقوع یک پرچم در میانه یک کادر، در قرارداد مبتنی بر بیت از روشهای بنام درج صفر<sup>۱</sup> استفاده می‌شود. در این روش ایستگاه فرستنده پس از هر پنج ۱ متواالی، یک صفر را ارسال می‌کند. گیرنده همواره هر صفری را که پس از پنج ۱ آمده باشد حذف می‌کند. بنابراین الگوی بیتی ۰۱۱۱۱۱۰۱ ارسال گشته و بوسیله گیرنده پس از حذف ۰ درج شده، بصورت اولیه‌اش ذخیره می‌گردد. در نتیجه به هیچ وجه الگوی ۰۱۱۱۱۱۱۰ بین ابتداء 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 ۲۵۶ آدرس را مشخص کند. برخی قراردادهای مبتنی بر بیت امکان استفاده از میدان آدرس وسیعتری را در اختیار می‌گذارند. برای حصول به این امکان، کم ارزشترین بیت بایت آدرس فعلی، در صورت وجود بایت بعدی، صفر می‌گردد. تشخیص آخرین بایت آدرس با یافتن یک ۱ در کم ارزشترین بیت آن آدرس امکان پذیر می‌باشد.

پس از میدان آدرس، میدان کنترل قرارداد. میدان کنترل می‌تواند در سه قالب مختلف مطابق شکل ۱۱-۲۷ 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 متواالی را می‌فرستد هر کادر را شمارش و شماره‌گذاری می‌کند. این شمارش بوسیله ارسال شمار N شمرده می‌شود. ایستگاهی که کادرهای متواالی را دریافت می‌نماید، هر یک از کادرهای بدون خطای دریافتی را می‌شمارد. تعداد این کادرهای دریافتی را دریافت شمار N<sub>۲</sub> می‌شمارد. هر وقت کادری وارسی می‌شود و مشخص شود خطای رخ نداده است، N<sub>۳</sub> افزایش می‌یابد. گیرنده، کادرهای اطلاعاتی شماره دار

1- Zero Insertion

2- Send Count

3- Recieve Count

1	2	3	4	5	6	7	8
0		$N_S$		P/F		$N_r$	
1	0		کد		P/F		$N_r$
1	1		کد		P/F		کد

انتقال اطلاعات: ناظاری:

بدون شماره: شماره ارسالی:  $N_s$

شماره دریافتی:  $N_r$

P/F: نهایی / همه پرسی  
کد: کد دودویی

شکل ۱۱-۲۷ قالب میدان کنترل در قرارداد مبتنی بر بیت

پذیرفته شده را با برگرداندن شماره  $N_r$  به ایستگاه فرستنده تأیید می‌کند.

بیت P/F توسط ایستگاه اصلی برای همه پرسی 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 (معنی همه پرسی<sup>۱</sup>) و وقتی که ایستگاه فرعی در حال ارسال باشد، F (معنی نهایی<sup>۲</sup>) خوانده می‌شود. هر کادر ارسالی از ایستگاه اصلی به فرعی دارای بیت P برابر با ۰ است. وقتی که ایستگاه اصلی بکار خود پایان دهد و آماده پاسخ ایستگاه فرعی گردد، P برابر ۱ می‌گردد. سپس ایستگاه فرعی شروع به ارسال تعدادی کادر می‌کند که در آنها بیت F برابر ۰ است. وقتی که ایستگاه دوم آخرین کادر را ارسال نمود، بیت F را ۱ می‌کند. بنابراین، بیت P/F برای تعیین ختم ارسال داده بکار می‌رود.

قالب ناظاری میدان کنترل با ۰ یا ۱ بودن دو بیت اول مشخص می‌شود. دو بیت بعدی نوع فرمان را مشخص می‌نماید. بدنبال آن بیت P/F و کادر شماره رشته دریافتی<sup>۳</sup> آمده است. کادرهای قالب ناظار 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 ۱۱ بودن دو بیت اول مشخص می‌شود. پنج بیت موجود در این قالب می‌تواند تا ۳۲ 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. قادرهاي قالبی که شماره ندارند برای شروع عملیات ارتباط<sup>۱</sup>، گزارش خطاهای رویهای<sup>۲</sup>، 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 ۱۱-۲ برابر با معادل دودویی اعداد ۱۲، ۱۳، ۱۴ و ۱۵ است. مدار خارجی لازم که باید بین یک آدرس I/O هشت بیتی از CPU و در ورودی‌های RS1، CS و RSO از مدار واسطه وصل شوند را نشان دهید.

۱۱-۲ شش واحد واسطه از نوعی که در شکل ۱۱-۲ 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<sup>۳</sup> 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 کم ارزشتر 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 درگیری برای انسان داشته باشد نام ببرید.

۱۱-۴ نام کامل خود را به ASCII با استفاده از هشت بیت برای هر کاراکتر و یک ۰ در سمت چپ ترین 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 نگاشت 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 دستور العمل بعدی اگر پرچم ۱ باشد.

ب) جستجوی یک رکورد خاص بر روی یک دیسک مغناطیسی

ج) وارسی 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 دست دهنی مربوط به انتقال داده‌ها از وسیله I/O به واحد واسطه از نام‌های مختلفی استفاده می‌کند.. خط دست دهنی ورودی واسطه STB (مخلف

استروب یا فعال ساز<sup>۱</sup>) و خط دست دهنی خروجی واسطه IBF (مخفف پریبوند بافر ورودی<sup>۲</sup>) نامگذاری شده است. وجود یک سیگنال سطح بالا روی IBF نشان می دهد که داده توسط واسطه پذیرفته شده است. پس از سیگنال خواندن O/I از طرف CPU از طریق ثبات داده، IBF به سطح پایین می رود.

(الف) یک بلاک دیاگرام که CPU، واسطه و وسیله O/I همراه با اتصال های لازم بین آنها را نشان دهد رسم کنید.

(ب) یک دیاگرام زمانی برای انتقال دست دهنی رسم کنید.

(ج) برای انتقال از وسیله به واسطه و از واسطه به CPU یک چارت دنباله رخدادها<sup>۳</sup> را رسم کنید. یک CPU با ساعت ۲۰MHz ۱۱-۸ به یک واحد حافظه که زمان دسترسی آن ۴۰ns است متصل است. دیاگرام های زمانی خواندن و نوشت آنها را با استفاده از فعال کننده های READ و WRITE و تهیه کنید. در دیاگرام زمانی آدرس را هم نشان دهید.

۱۱-۹ مدار واسطه ارتباطی غیرهمگام شکل ۱۱-۸ بین یک CPU و یک چاپگر متصل شده است. فلوچارتی رسم کنید که دنباله عملیات ارسال کاراکتر از CPU به چاپگر را نشان دهد.

۱۱-۱۰ حداقل شش وضعیت مختلف را برای ۱ کردن بیت های ثبات وضعیت در یک واسطه ارتباطی غیرهمگام ارائه دهید.

۱۱-۱۱ اگر واسطه به پایانه ای 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 بیت های ثبات شیفت را وقتی که حرف W ارسال شود مشخص کنید.

۱۱-۱۲ بر روی یک خط Baud-1200 با هر یک از روش های زیر چند کاراکتر در هر ثانیه ارسال می شود (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 می شود. اطلاعات پاک شده در آن نیز n بایت در ثانیه است. حداقل ظرفیت بافر ۱۲ بایت است.

(الف) اگر m > n باشد چه مدت طول می کشد تا بافر خالی، پر شود.

(ب) اگر n < m باشد چقدر طول می کشد تا بافر پر، خالی شود.

(ج) اگر m = n باشد آیا بافر FIFO لازم است یا خیر؟

۱۱-۱۴ بیت های ثبات کنترل FIFO در شکل ۱۱-۹ عبارتند از F<sub>1</sub>F<sub>2</sub>F<sub>3</sub>F<sub>4</sub> = 0011. رشته عملیات داخلی را هر گاه یک قلم از داده از FIFO حذف و سپس داده جدیدی وارد شود معین کنید.

- ۱۱-۱۵ مقدایر "وروی آماده" و "خرنچی آماده" و بیت‌های کنترل F<sub>1</sub> تا F<sub>4</sub> در شکل ۱۱-۹ 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 ۱۱-۱۰ را برای انتقال داده از CPU به یک واسطه و سپس به یک وسیله I/O رسم کنید. رویه‌ای برای ۱ کردن و ۰ 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 اولویت دار چرخشی شکل ۱۱-۱۲، هرگاه پس از درخواست وقفه وسیله ۲ به CPU ولی قبل از تصدیق وقفه به وسیله CPU، وسیله ۱ درخواست وقفه کند چه اتفاقی می‌افتد.
- ۱۱-۲۱ کامپیوتری را بدون سخت‌افزار وقفه اولویت دار 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 ۱۱-۲ را برای کدگذار اولویت بدست آورید. نمودار منطقی مدار را رسم کنید.
- ۱۱-۲۳ سخت‌افزار وقفه اولویت دار موازی را برای سیستمی با هشت منبع وقفه طراحی کنید.
- ۱۱-۲۴ جدول درستی انکدر  $8 \times 3$  اولویت را بدست آورید. فرض کنید که سه خروجی xyz از انکدر اولویت برای تهیه یک آدرس بردار به شکل 101xyz 00 بکار روند. هشت آدرس بردار را با شروع از آن که بالاترین اولویت را دارد بنویسید.
- ۱۱-۲۵ در شکل ۱۱-۱۴ چه کاری باید انجام داد تا چهار مقدار VAD برابر با ۷۶، ۷۷، ۷۸، ۷۹ گردند؟
- ۱۱-۲۶ برای وارسی 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 نوشتمن 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 ۱۱-۱۸ صورت می گیرد.

۱۱-۲۹ یک کنترل کننده DMA کلمه های 16 بیتی را با استفاده از سرقت 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 دستور العمل ها را با سرعت یک میلیون دستور العمل در ثانیه برداشت و اجرا می کند. CPU در اثر انتقال به شیوه DMA تا چه حد کنند می شود؟

۱۱-۳۰ چرا DMA بهنگام تقاضای انتقال به حافظه نسبت به CPU بهنگام تقاضای انتقال به حافظه اولویت دارد.

۱۱-۳۱ فلوچارتی 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 بلاکی 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 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**

- |                             |                        |
|-----------------------------|------------------------|
| ۱۲-۵ حافظه کش               | ۱۲-۱ سلسله مراتب حافظه |
| ۱۲-۶ حافظه مجازی            | ۱۲-۲ حافظه اصلی        |
| ۱۲-۷ سخت افزار مدیریت حافظه | ۱۲-۳ حافظه کمکی        |
|                             | ۱۲-۴ حافظه تداعیگر     |

**۱۲-۱ سلسله مراتب حافظه**

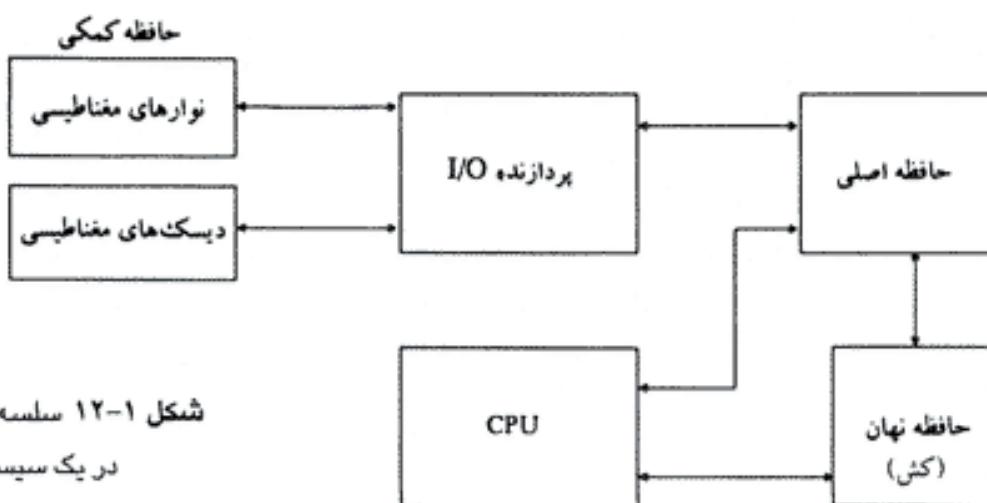
واحد حافظه در هر کامپیوتر دیجیتال یکی از بخش‌های اصلی است زیرا برای ذخیره برنامه‌ها و داده‌ها مورد نیاز می‌باشد. یک کامپیوتر بسیار کوچک با کاربرد محدود ممکن است بتواند کار معینی را بدون نیاز به افزایش ظرفیت ذخیره‌سازی انجام دهد. اما کامپیوترهای همه منظوره اگر علاوه بر ظرفیت حافظه اصلی، به امکانات ذخیره سازی اضافی مجهز باشند کارها را موثرتر اجرا خواهند نمود. در یک کامپیوتر معمولاً فضای واحد حافظه برای جای دادن همه برنامه‌های مورد استفاده کافی نیست. علاوه، اکثر کاربران کامپیوتر مقادیر قابل توجهی از نرم‌افزارهای پردازش داده را جمع‌آوری نموده و به این عمل همچنان ادامه می‌دهند. البته کلیه اطلاعات جمع‌آوری شده بطور همزمان مورد نیاز پردازنده نیست. بنابراین، مقرنون به صرفه‌تر است از وسایل ذخیره‌سازی ارزان قیمت بعنوان پشتیبانی، اطلاعاتی که در حال حاضر مورد نیاز CPU نیستند، استفاده شود. واحد حافظه‌ای که مستقیماً با CPU ارتباط دارد حافظه اصلی<sup>۱</sup> نامیده می‌شود. وسایلی که نقش پشتیبانی ذخیره‌سازی را بعهده دارند حافظه کمکی<sup>۲</sup> خوانده می‌شوند. معمول‌ترین وسیله حافظه کمکی بکار رفته در سیستم‌های کامپیوتری دیسک‌ها و نوارهای مغناطیسی هستند. این وسایل برای ذخیره برنامه‌های سیستم، فایل‌های بزرگ داده، و سایر اطلاعات پشتیبانی مورد استفاده‌اند. در حافظه اصلی فقط برنامه‌ها و داده‌های مورد استفاده جاری پردازنده جای می‌گیرند. کلیه اطلاعات دیگر در حافظه کمکی ذخیره می‌شود و هنگام لزوم به حافظه اصلی منتقل می‌گردند.

1- Main Memory

2- Auxiliary Memory

ظرفیت کل یک کامپیوتر را می‌توان بصورت یک سیستم سلسله مراتبی از اجزاء تصور کرد. این سیستم حافظه سلسله مراتبی متشكل است از کلیه وسایل ذخیره‌سازی در یک سیستم کامپیوتری، از حافظه کمکی گندولی با ظرفیت بالاگرفته تا حافظه اصلی نسبتاً سریع تر، و حتی حافظه کوچکتر ولی سریع تر گش (نهان) که بوسیله مدارهای منطقی پردازشگر سریع قابل دسترسی هستند. شکل ۱۲-۱ سریع تر گش (نهان) که بوسیله مدارهای منطقی پردازشگر سریع قابل دسترسی هستند. شکل ۱۲-۱ اجزاء سلسله مراتبی حافظه‌ای نوعی را نشان می‌دهد. در پایین سیستم سلسله مراتبی نوارهای مغناطیسی نسبتاً گند قرار دارد که برای ذخیره فایل‌های بکار می‌رود. پس از آن دیسک‌های مغناطیسی واقعند که بعنوان ذخیره‌سازی پشتیبانی بکار می‌رond. حافظه اصلی مکان میانی را دارد. و قادر است مستقیماً با CPU و وسایل کمکی از طریق پردازنده I/O ارتباط داشته باشد. وقتی برنامه‌هایی که در حافظه اصلی قرار ندارند بوسیله CPU مورد نیاز باشند، از حافظه کمکی بداخل آورده می‌شوند. برنامه‌هایی که در حال حاضر در حافظه اصلی قرار دارند ولی مورد نیاز نیستند به حافظه کمکی منتقل می‌گردند تا فضای کافی برای برنامه‌ها و داده‌های جاری بکار رفته ایجاد شود.

نوعی حافظه خاص بسیار سریع که گش نامیده می‌شود گاهی اوقات برای افزایش سرعت پردازش با در اختیار گذاشتن برنامه‌ها و داده‌های جاری برای CPU با سرعتی بالا، بکار می‌رود. حافظه گش در کامپیوترها برای جبران اختلاف سرعت بین دستیابی حافظه اصلی و مدار پردازنده مورد استفاده قرار می‌گیرد. مدار CPU معمولاً سریعتر از زمان دستیابی (سرعت) حافظه اصلی است، لذا سرعت پردازش را عمدتاً سرعت حافظه اصلی محدود می‌کند. تکنیکی که برای جبران عدم تطابق سرعت‌های عملیاتی بکار می‌رود استفاده از یک حافظه کش کوچک و فوق العاده سریع بین CPU و حافظه اصلی است که زمان دستیابی آن نزدیک به پریود ساعت پردازنده می‌باشد. حافظه کش برای ذخیره قطعه‌هایی از برنامه‌ها که در CPU در دست اجرا هستند و داده‌های موقتی که در محاسبات جاری مکرراً بکار می‌رond مورد استفاده قرار می‌گیرد. با فراهم نمودن سریع برنامه‌ها و داده‌ها، امکان افزایش سرعت عملکرد وجود خواهد داشت. مادامی که پردازنده I/O انتقال داده را بین حافظه کمکی و حافظه اصلی اداره می‌کند، سازمان کش به



شکل ۱۲-۱ سلسله مراتب حافظه  
در یک سیستم کامپیوتری

انتقال اطلاعات بین حافظه اصلی و CPU می‌پردازد. 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 می‌یابد و زمان دستیابی حافظه طولانی تر می‌گردد. حافظه کمکی ظرفیت ذخیره‌سازی بالایی را دارد، 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 می‌گردد. هدف نهایی از بکارگیری سلسله مراتب حافظه‌ای، فراهم آوردن بالاترین سرعت دستیابی متوسط ممکن با حداقل ارزش کل تمام حافظه سیستم است.

حافظه‌های کمکی و کش برای مقاصد مختلفی بکار می‌روند. کش آن بخش از برنامه و داده‌هایی را که بیشتر مورد استفاده‌اند ذخیره می‌کند، در حالی که حافظه کمکی بخش‌هایی را که در حال حاضر مورد استفاده CPU نیست نگه میدارد. 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 بلاک‌های داده است. نسبت زمان دستیابی بین کش و حافظه اصلی ۱ به ۷ است. مثلاً یک حافظه کش نوعی ممکن است زمان دستیابی ۱۰۰ns را داشته باشد، در حالیکه زمان دستیابی حافظه اصلی 700ns است. زمان دستیابی متوسط حافظه کمکی معمولاً ۱۰۰۰ برابر حافظه اصلی است. سایز بلاک در حافظه کمکی بین 256 تا 2048 کلمه است، در حالی که سایز بلاک در حافظه کش از ۱ تا 16 کلمه می‌باشد.

بسیاری از سیستم‌های عامل بصورتی طراحی می‌شوند تا CPU بتواند بطور همزمان تعدادی از برنامه‌های مستقل را اجرا نماید. این مفهوم که چندبرنامگی<sup>۱</sup> خوانده می‌شود، به وجود دو یا چند برنامه در بخش‌های متفاوت سلسله مراتب حافظه در یک زمان اشاره دارد. بدین ترتیب می‌توان تمام بخش‌های کامپیوتر را با کار بر روی چندین برنامه بطور سری، مشغول نگهداشت. مثلاً فرض کنید برنامه‌ای در CPU در دست اجراست و یک انتقال O/I لازم شود. CPU، پردازنده O/I را برای شروع اجرای انتقال مقداردهی اولیه می‌نماید. این عمل به CPU اجازه می‌دهد تا برنامه دیگری را اجرا کند. در یک سیستم چندبرنامگی، وقتی یک برنامه برای انتقال ورودی - خروجی منتظر می‌ماند، برنامه دیگری آماده استفاده از CPU است.

در چندبرنامگی، نیاز به اجرای برنامه‌ها بصورت قطعه به قطعه، تغییر اندازه حافظه اصلی در حال استفاده توسط یک برنامه مفروض، و جایجایی برنامه‌ها به بخش‌های مختلف سلسله مراتب حافظه وجود دارد. برنامه‌های کامپیوتر گاهی طولانی تر از آن هستند که در کل فضای موجود در حافظه اصلی جاگیرند. بعلاوه یک سیستم کامپیوتر چندین برنامه را بکار می‌برد و همه برنامه‌ها نمی‌توانند در یک حافظه اصلی در تمام مدت قرار بگیرند. معمولاً یک برنامه با داده‌های مربوطه‌اش در حافظه کمکی قرار می‌گیرند. وقتی که برنامه یا بخشی از برنامه قرار است اجرا شود، ابتدا به حافظه اصلی انتقال می‌یابد تا بوسیله CPU اجرا شود. بنابراین هر کس می‌تواند 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<sup>۱</sup> نامیده می‌شود. سخت‌افزار یک سیستم مدیریت حافظه در بخش ۱۲-۷ آرايه شده است.

## ۱۲-۲ حافظه اصلی

حافظه اصلی واحد ذخیره‌سازی مرکزی در یک سیستم کامپیوتراست. این واحد، حافظه نسبتاً بزرگ و سریع است که برای ذخیره برنامه‌ها و داده‌ها در جین عملکرد کامپیوترا 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<sup>۲</sup> و دینامیک<sup>۳</sup> در دسترس قرار دارند. RAM استاتیک اساساً از فلیپ‌فلایپها 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 میلی ثانیه 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 کامپیوترا 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<sup>۴</sup> بکار رفت، اما اکنون از آن برای مشخص کردن حافظه خواندنی و نوشتنی<sup>۵</sup> در مقابل حافظه فقط خواندنی<sup>۶</sup> 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 کامپیوترا جای داده می‌شوند و یا محتوای جداول ثابتی که پس از تکمیل ساخت کامپیوترا تغییر نمی‌کنند مورد استفاده قرار می‌گیرند.

در میان سایر امکانات در کامپیوترا، بخش ROM حافظه اصلی برای ذخیره برنامه آغازینی بنام بارکننده راهانداز<sup>۷</sup> بکار می‌رود. بارکننده راهانداز برنامه‌ای است که کار آن آغاز کار نرم‌افزار کامپیوترا 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

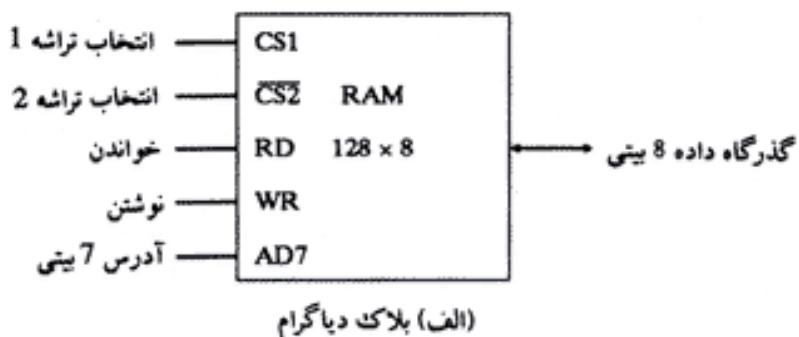
می‌رود. محتوای ROM پس از قطع و وصل توان الکتریکی تغییری نمی‌کند. آغاز بکار کامپیوتر مشکل است از روشن شدن دستگاه و شروع اجرای یک برنامه اولیه. بنابراین وقتی که دستگاه روشن شد، سخت‌افزار کامپیوتر شمارنده برنامه را با اولین آدرس بارکتnde راهانداز بار می‌کند. برنامه راهانداز بخشی از سیستم عامل را از دیسک به حافظه اصلی بار می‌کند و سپس کنترل به سیستم عامل انتقال می‌یابد، که کامپیوتر را برای استفاده عمومی آماده می‌نماید.

تراشه‌های RAM و ROM در اندازه‌های 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 تراشه‌ها، ساخت یک تراشه  $8 \times 1024$  را با تراشه‌های RAM با اندازه  $8 \times 128$  و تراشه‌های ROM با اندازه  $8 \times 512$  نشان می‌دهیم.

## تراشه‌های ROM و RAM

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. ویژه‌گی متداول دیگر، گذرگاه داده دو طرفه‌ای است که امکان انتقال داده از حافظه به CPU را در عمل خواندن، یا از CPU به حافظه را در حین عمل نوشتن فراهم می‌سازد. گذرگاه دو طرفه را می‌توان با بافرهای سه حالت ساخت. خروجی یک بافر سه حالت می‌تواند 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 سیگنال‌های معمولی منطقی هستند. حالت امپدانس بالا مانند یک مدار باز عمل می‌کند، و بدان معنی است که خروجی سیگنالی حمل نمی‌کند و معنای منطقی ندارد.

پلاک دیاگرام یک تراشه RAM در شکل ۱۲-۲ 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. ورودی‌های خواندن و نوشتن عمل مورد نظر را مشخص می‌کنند و دو ورودی کنترل انتخاب تراشه (CS) برای فعال کردن تراشه بهنگام انتخاب توسط ریزپردازنده، بکار می‌روند. وجود بیش از یک ورودی کنترل برای انتخاب تراشه، عمل دیکد شدن آدرس را هنگامی که چندین تراشه در ریزپردازنده وجود داشته باشد تسهیل می‌نماید. ورودی‌های خواندن و نوشتن گاهی بصورت یک خط W/R در می‌آیند. هر وقت تراشه انتخاب شود، دو حالت دودویی این خط دو عمل خواندن یا نوشتن را مشخص می‌کند. جدول کار لیست شده در شکل ۱۲-۲ (ب) طرز کار تراشه RAM را مشخص می‌نماید. واحد حافظه فقط هنگامی که  $CS1=1$  و  $CS2=0$  باشد فعال می‌شود. خط بالای متغیر انتخاب دوم نشان می‌دهد که این ورودی وقتی که برابر 0 باشد فعال است. اگر ورودی‌های انتخاب تراشه فعال نباشند، و یا اگر فعال باشند ولی ورودی‌های خواندن یا نوشتن فعال نباشند، حافظه مسدود و گذرگاه داده آن در حالت



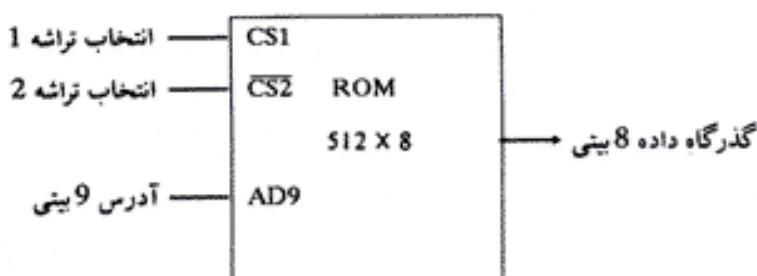
وضعیت گذرگاه داده	عمل حافظه	CS1	CS2	RD	WR
امپدانس بالا	سدود	0	0	x	x
امپدانس بالا	سدود	0	1	x	x
امپدانس بالا	سدود	1	0	0	0
ورودی داده به RAM	نوشتن	1	0	0	1
خروجی داده به RAM	خواندن	1	0	1	x
امپدانس بالا	سدود	1	1	x	x

(ب) جدول تابع

شکل ۱۲-۲ تراشه‌ای از نوع RAM

امپدانس بالا خواهد بود. اگر  $\overline{CS1}=1$  و  $\overline{CS2}=0$  باشد، حافظه می‌تواند 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 می‌کند ذخیره می‌نماید. اگر ورودی RD فعال شود، محتوای بایت انتخاب شده روی گذرگاه داده قرار می‌گیرد. سیگنالهای RD و WR عملکرد حافظه و بافرهای گذرگاه مربوط به گذرگاه داده دو جهته را کنترل می‌نماید.

یک تراشه ROM هم سازمان خارجی مشابهی دارد. با این وجود، چون ROM تنها خوانده می‌شود، گذرگاه داده تنها در حالت خروجی قرار می‌گیرد. بلاک دیاگرام یک تراشه ROM در شکل ۱۲-۳ دیده می‌شود. در مقایسه دو تراشه RAM و ROM با سایز برابر، ROM می‌تواند تعداد بیت‌های بیشتری نسبت به RAM داشته باشد، زیرا سلولهای دودویی ROM فضای کمتری را نسبت به



شکل ۱۲-۳ تراشه‌ای از نوع ROM

RAM اشغال می‌کنند. به همین دلیل، دیاگرام 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 کامپیوتی 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 کامپیوتی به یک RAM با ظرفیت 512 بایت و یک ROM با ظرفیت 128 بایت نیاز دارد. تراشه‌های RAM و ROM بکار رفته در شکل ۱۲-۲ و ۱۲-۳ نشان داده شده‌اند. نقشه آدرس‌های حافظه برای این پیکربندی در جدول ۱۲-۱ آمده است. ستون قطعات نشان می‌دهد که آیا ROM بکار رفته است یا RAM. ستون آدرس شانزده‌芻 شانزده‌芻 به هر تراشه محدوده‌ای از آدرس‌های معادل شانزده‌芻 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. X‌های 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. X‌ها همیشه به خطوط پائین رتبه گذرگاه اختصاص می‌یابند: خطوط 1

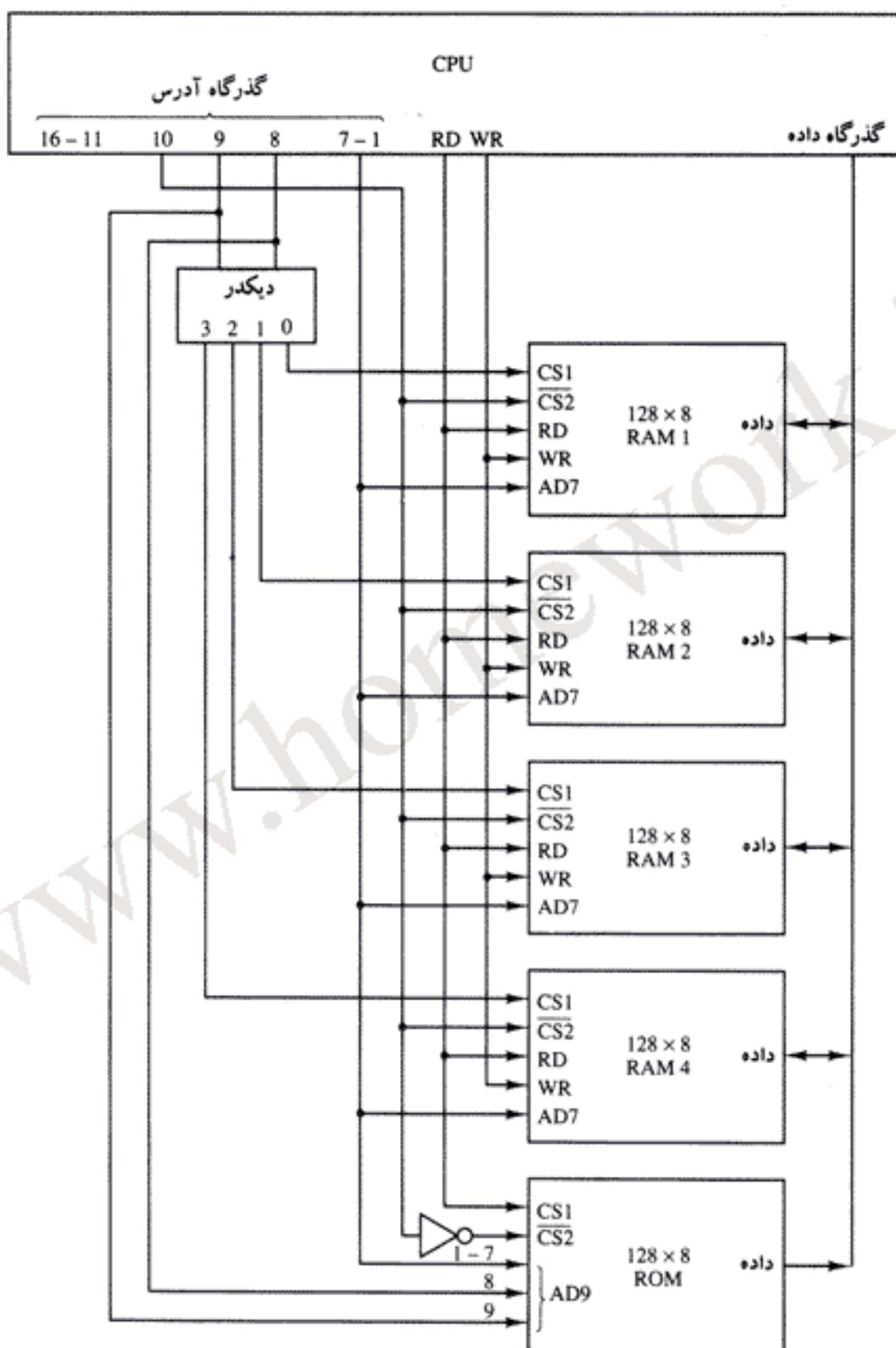
جدول ۱۲-۱ نقشه آدرس‌های حافظه برای ریزکامپیوت

قطعه	آدرس شانزده‌芻 شانزده‌芻	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 هر یک از هم متمایز کنیم. برای این مثال خاص، ما خطوط گذرگاه 8 و 9 را برابر نمایش چهار ترکیب دو دویی متمایز انتخاب می کنیم. وقت کنید هر چفت دیگری از خطوط گذرگاه که بکار نرفته باشند را می توان برای این منظور انتخاب کرد. جدول بخوبی نشان می دهد که 9 خط پائین رتبه آدرس فضای حافظه ای برابر با  $2^{q=512}$  بایت را ایجاد می کند. تمايز بین آدرس های RAM و ROM به یک خط دیگر گذرگاه انجام می شود. در اینجا فقط خط 10 را برای این منظور انتخاب می کنیم. هنگامی که خط 10 صفر باشد، CPU یک تراشه RAM را انتخاب می کند. و اگر این خط 1 باشد، ROM انتخاب می شود. آدرس معادل شانزده شانزدهی تراشه از اطلاعات زیر ستون گذرگاه آدرس فراهم می شود. خطوط گذرگاه آدرس به گروههای چهار بیتی تقسیم شده است بطوری هر گروه با یک رقم شانزده شانزدهی نمایش داده می شود. اولین رقم شانزده شانزدهی خطوط 13 تا 16 را نشان می دهد و همواره 0 است. رقم شانزده شانزدهی بعدی خطوط 9 الی 12 را نشان می دهد ولی خطوط 11 و 12 همیشه صفرند. محدوده آدرس های شانزده شانزدهی برای هر قطعه از X های مربوط به آن تعیین می شود. این X ها یک عدد دو دویی را از تماماً 0 تا تماماً 1 نمایش می دهند.

### اتصالات حافظه به CPU

تراشه های RAM و ROM از طریق گذرگاههای داده و آدرس به CPU متصل می شوند. خطوط پائین رتبه در گذرگاه آدرس بایت درون تراشه را انتخاب می کنند و سایر خطوط در گذرگاه آدرس تراشه را از طریق ورودی های انتخاب برمی گزینند. اتصال تراشه های حافظه به CPU در شکل ۱۲-۴ نشان داده شده است. این پیکریندی 512 بایت را برای RAM و 512 بایت را هم برای ROM فراهم می آورد. پیکریندی مذکور با توجه به جدول ۱۲-۱ پیاده سازی شده است. هر RAM هفت بیت پائین رتبه از گذرگاه آدرس را برای انتخاب یکی از 128 بایت ممکن دریافت می کند. ROM بخصوصی که انتخاب می شود با توجه به خطوط 8 و 9 گذرگاه است. این عمل از طریق یک دیکدر  $2 \times 4$  که خروجی های آن به ورودی های CS1 در هر تراشه RAM متصل است صورت می گیرد. بنا براین وقتی که خطوط آدرس 8 و 9 برابر 00 شوند، اولین تراشه RAM انتخاب می شود. وقتی که 01 باشد، دومین تراشه انتخاب می گردد، و الى آخر. خروجی های RD و WR از ریزپردازنده به ورودی های همه تراشه های RAM وصل می شوند. انتخاب بین RAM و ROM از طریق خط شماره 10 گذرگاه حاصل می گردد. RAM ها هنگامی انتخاب می شوند 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 ورودی های آدرس ROM اعمال شده اند. این اتصالات آدرس ها 0 تا 511 را به RAM و 512 تا 1023 را به ROM اختصاص می دهند. گذرگاه داده ROM تنها امکان خروجی شدن را دارد، در صورتیکه گذرگاه داده متصل



شکل ۱۲-۴ اتصال حافظه به CPU

به RAM می‌تواند اطلاعات را در هر دوچهت هدایت کند.

مثالی که ارائه شد پیچیدگی اتصالات بین تراشه‌های حافظه و CPU را نمایان می‌سازد. هر چه تعداد تراشه‌های متصله بیشتر شود، دیکدرهای بیشتری بین حافظه‌ها مورد نیاز است طراح باید نقشه حافظه‌ای را تهیه کرده و آدرس‌ها را به تراشه تخصیص دهد تا با استفاده از آن اتصالات لازم معین شد.

### ۱۲-۳ حافظه کمکی

متداول‌ترین دستگاه حافظه کمکی بکار رفته در سیستم‌های کامپیوتی دیسک‌ها و نوارهای مغناطیسی هستند. وسایل دیگری هم که نه کراراً ولی بکار می‌روند عبارتند از درام‌های مغناطیسی<sup>۱</sup>، حافظه جباب مغناطیسی<sup>۲</sup>، و دیسک‌های نوری می‌باشند. برای درک فیزیکی مکانیزم دستگاه‌های حافظه کمکی باید اطلاعاتی در مورد سیستم‌های مغناطیسی، الکترونیکی و الکترومکانیکی داشت. هر چند خواص فیزیکی این وسایل ذخیره‌سازی می‌تواند بسیار پیچیده باشد، ولی خصوصیات منطقی آنها قابل تشخیص و مقایسه است. مشخصه‌های 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<sup>۳</sup> (یا دسترسی) خوانده می‌شود. در وسایل الکترومکانیکی با قطعات متحرک مانند دیسک‌ها و نوارهای، زمان دستیابی مشکل است از زمان جستجوی<sup>۴</sup> لازم برای فراردادن هد خواندن - نوشتن بر روی یک مکان و زمان انتقال<sup>۵</sup> لازم برای انتقال داده به و یا از وسیله. چون زمان جستجو معمولاً خیلی طولانی‌تر از زمان انتقال است، حافظه کمکی بصورت مجموعه‌ای از رکوردها یا بلاک‌ها سازمان یافته است. یک بلاک تعداد مشخصی از کاراکترها یا کلمات است. معمولاً خواندن یا نوشتن بر روی تمام رکورد انجام می‌شود. سرعت انتقال تعداد کلمات یا کاراکترهایی است که وسیله می‌تواند در هر ثانیه، پس از قرارگرفتن در آغاز رکورد، انتقال دهد.

درام‌ها و دیسک‌های مغناطیسی از نظر عملکرد کاملاً مشابهند. هر دو از سطوح چرخان با سرعت بالا که از ماده مغناطیسی ضبط کننده پوشش یافته‌اند تشکیل شده‌اند. سطح چرخان درام، سیلندری شکل و دیسک سطح گرد مسطحی است. سطح ضبط کننده با سرعت یکنواختی چرخیده و در زمان دستیابی هرگز راهاندازی یا متوقف نمی‌شود. بیت‌ها بصورت نقطه‌های مغناطیسی با عبور سطح از زیر یک مکانیزم ساکن بنام هد نوشتن<sup>۶</sup>، بر روی آن ضبط می‌شوند. بیت‌های ذخیره شده با تغییر میدان حاصل از نقطه‌های ثبت شده در روی سطح که با عبور از کنار هد خواندن<sup>۷</sup> حاصل می‌شود آشکار می‌گردد. مقدار سطح موجود برای ضبط در یک دیسک بزرگتر از درامی است که از نظر ابعاد با آن یکسان

۱- Magnetic Drum

2- Mangetic Bubble Memory

3- Access time

4- Seek time

5- Transfer time

6- Write Head

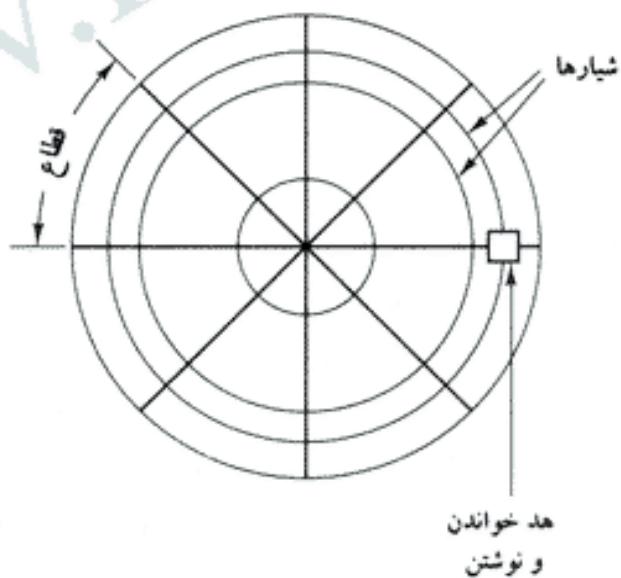
7- Read Head

باشد. بنابراین اطلاعات بیشتری را می‌توان بر روی یک دیسک هم اندازه با درام ذخیره کرده باین دلیل، دیسک‌ها در اکثر کامپیوتروها جایگزین درام شده‌اند.

### دیسک‌های 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 دیسک‌ها با هم و با سرعت زیادی می‌چرخند و به منظور دستیابی راه اندازی یا متوقف نمی‌شوند. بیت‌ها در سطح مغناطیس شده بصورت نقاطی بر روی دواير هم مرکزی بنام شیار<sup>۱</sup> ذخیره می‌شوند. شیارها معمولاً به بخش‌هایی به نام قطاع<sup>۲</sup> تقسیم می‌شوند. در بیشتر سیستم‌ها، حداقل اطلاعات که می‌تواند انتقال یابد یک قطاع است. تقسیم‌بندی یک دیسک به شیارها و قطاع‌ها در شکل ۱۲-۵ نشان داده شده است.

برخی از دستگاه‌ها از یک هد خواندن و نوشتن برای هر سطح دیسک استفاده می‌کنند. در این دستگاه‌ها قبل از خواندن و نوشتن، بیت‌های آدرس شیارها برای حرکت دادن هد خواندن یا نوشتن بر روی مکان موردنظر بکار می‌روند. در نوعی دیگر از سیستم‌های دیسک، هدهای خواندن - نوشتن جداگانه برای هر شیار در هر سطح تهیه شده است. سپس بیت‌های آدرس، شیار خاصی را بطور الکترونیکی از طریق یک دیکدر انتخاب می‌کنند. این نوع دستگاه‌ها گرانتر بوده و فقط بر روی سیستم‌های کامپیوتروی بسیار بزرگ یافت می‌شوند.



شکل ۱۲-۵ دیسک مغناطیسی

1- Track

2- Sector

در دیسک‌ها از شیارهای زمانبندی دائمی برای همگام کردن بیت‌ها و تشخیص قطاع‌ها استفاده می‌شود. یک سیستم دیسک با بیت‌های آدرسی که شماره دیسک، سطح دیسک، شماره قطاع، و شیار موردنظر در قطاع مربوطه را مشخص می‌کنند آدرس دهن می‌شود. پس از قرار گرفتن هد خواندن - نوشتن در شیار موردنظر، سیستم باید با چرخش دیسک تا رسیدن قطاع مشخص شده به زیر هد خواندن و نوشتن صبر کند. پس از رسیدن به ابتدای قطاع، انتقال اطلاعات با سرعت صورت می‌گیرد. دیسک‌ها ممکن است هدهای متعدد داشته باشند و بیت‌های چندین شیار بطور همزمان انتقال یابند.

در هر قطاع 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 قطاع، 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 قطاع، معین یکسان خواهد بود.

دیسک‌هایی که بطور دائمی به مجموعه دستگاه متصل‌اند و آنها را هر کاربری نمی‌توانند از مجموعه خارج کند، دیسک سخت<sup>۱</sup> می‌خوانند. دیسک خوانی که دیسک‌های قابل جابجاگی داشته باشد دیسک نرم<sup>۲</sup> خوانده می‌شود. دیسک‌هایی که در یک دیسک خوان نرم بکار می‌روند دیسک‌های قابل جابجاگی کوچکی هستند که از پلاستیک ساخته شده و با ماده مغناطیسی ضبط کننده پوشش داده شده‌اند. دو سایز برای این دیسک‌ها متداول است، قطر ۵.۲۵ و قطر ۳.۵ اینچ. دیسک‌های ۳.۵ اینچی کوچکترند و می‌توانند داده‌های بیشتری را نسبت به دیسک‌های ۵.۲۵ 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 باریکه ای از پلاستیک پوشیده از ماده مغناطیسی ضبط کننده 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 رشته ای 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

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 تازمانی که به رکورد بعدی پرسد 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 انتهای، واحد کنترل شروع یک فاصله را درک می کند. دستگاه نوار با مشخص کردن شماره رکورد و تعداد کاراکترهای رکورد آدرس دهنی می شود. رکوردها ممکن است دارای طول ثبات یا متغیر باشند.

#### ۱۲-۴ 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 سمبول را برای استخراج معادل دودویی جستجو می کند. ممکن است برای تعیین نام وضعيت 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 متواالی آدرس دهنی 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 کارآیی الگوریتم بستگی دارد. برای به حداقل رساندن تعداد مراجعات برای یافتن یک داده، الگوریتم های جستجوی متعددی ابداع شده است.

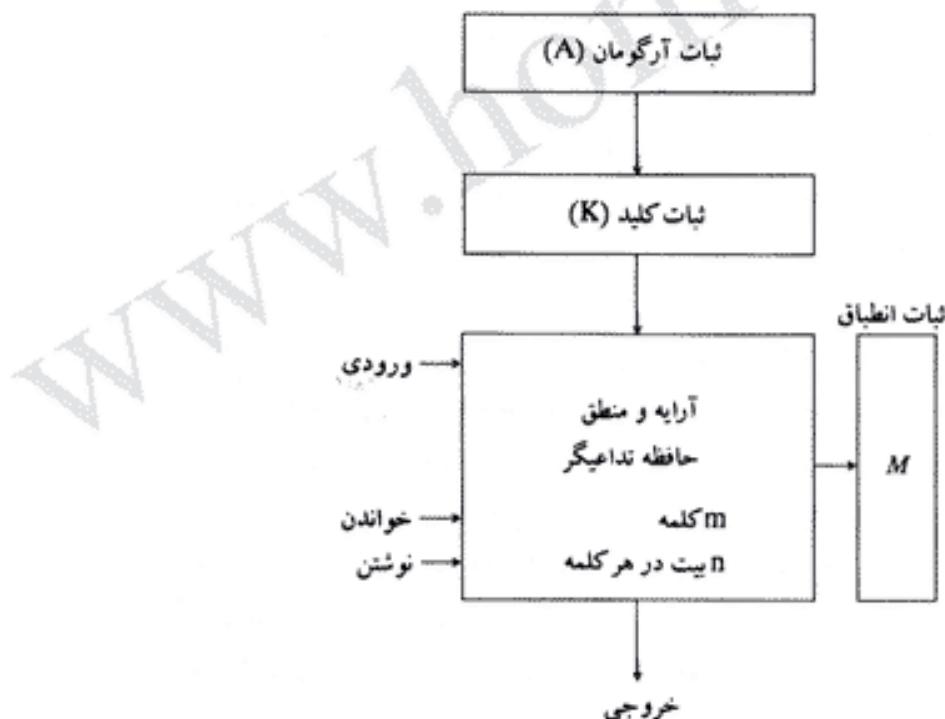
اگر بتوان داده های ذخیره شده در حافظه را به منظور دستیابی به آنها براساس محتوای خود داده و نه براساس یک آدرس شناسایی کرد، زمان لازم برای یافتن یک قلم داده تا حد قابل توجهی کاهش می یابد. واحد حافظه ای که دستیابی به آن براساس محتوا انجام می شود، حافظه تداعیگر<sup>۱</sup> یا حافظه قابل آدرس دهنی با محتوا (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 سلوی آن بعلاوه 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.

### سازمان سخت‌افزاری

بلاک دیاگرام یک حافظه تداعیگر در شکل ۱۲-۶ داده شده است. این حافظه متشکل از یک آرایه حافظه و منطق لازم برای  $n$  کلمه در هر کلمه است. ثبات آرگومان<sup>۱</sup> و ثبات کلید<sup>۲</sup> هر یک  $n$  بیت دارند، یعنی یک بیت به ازای هر یک از بیت‌های کلمات. ثبات انطباق  $M$  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 ۱ می‌کنند. پس از روند تطبیق، بیت‌هایی در ثبات انطباق که ۱ شده‌اند نشان می‌دهند که کلمات متناظر آنها با کلمه مورد جستجو مطابقت دارند. عمل خواندن با مراجعة متوالی به کلماتی از حافظه که بیت‌های متناظر آنها در ثبات انطباق ۱ شده‌اند انجام می‌شود.

ثبات کلید پوششی را برای انتخاب یک میدان خاص یا کلید آرگومان فراهم می‌آورد. اگر ثبات کلید تماماً ۱ 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 ۱ دارد مقایسه می‌شوند. بنابراین ثبات کلید پوششی 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 ۱۲-۶ بلاک دیاگرام حافظه تداعیگر

1- Argument Register

2- Switch Register

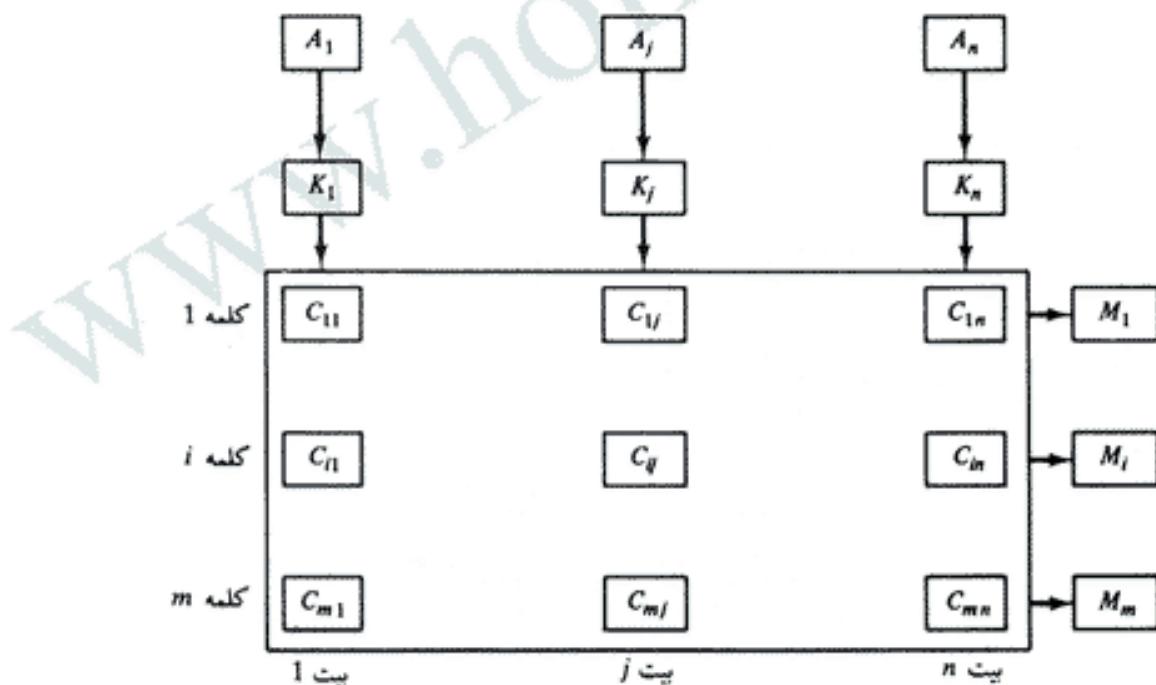
سمت چپ A با کلمات مقایسه زیرا K در این مکانها دارای 1 است.

A	101	111100
K	111	000000
1 کلمه	100	111100
2 کلمه	101	000001

انطباق ندارد      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 ۱۲-۷ نشان داده شده است.  
سلول‌های آرایه با حرف C و دو زیرنویس مشخص شده‌اند. اولین زیرنویس شماره کلمه و دومین  
زیرنویس مکان بیت را در کلمه مشخص می‌کند. باین ترتیب سلول  $C_{ij}$  مربوط به بیت  $j$  در کلمه  $i$  است.  
بیت  $j$  در ثبات آرگومان با همه بیت‌های ستون  $i$  از آرایه مقایسه می‌شود بشرطی که  $K_j = 1$  باشد. این  
عمل برای همه ستون‌های  $i = 1, 2, \dots, n$  انجام می‌شود.

اگر بین همه بیت‌های پوشش<sup>۱</sup> نیافته آرگومان و بیت‌های کلمه، تطابق وجود داشته باشد، بیت  
منتظر در ثبات انطباق یعنی  $M_i$  برابر با 1 می‌شود. اگر یک یا چند بیت پوشش نیافته و کلمه مطابقت  
نداشته باشند  $M_i$  برابر با 0 خواهد بود.



شکل ۱۲-۷ حافظه تداعیگر به  $m$  کلمه و  $n$  بیت بازه هر کلمه

#### 1- Mask

سازمان درونی یک سلول نوعی  $C_{ij}$  در شکل ۱۲-۸ نشان داده شده است. این سلول از یک المان ذخیره‌سازی فلیپ فلاب  $F_{ij}$  و مدارات لازم برای خواندن و نوشتمن و انطباق سلول‌ها تشکیل شده است. بیت ورودی در عمل نوشتمن به سلول ذخیره سازی انتقال می‌یابد. در حین عمل خواندن بیت ذخیره شده خوانده می‌شود. مدار منطقی محتوای سلول ذخیره سازی را با بیت متناظر پوشش نیافته آرگومان مقایسه کرده و یک خروجی را در اختیار مدار تضمین‌گیری قرار می‌دهد تا بیت  $M_i$  را ۱ کند.

### مدار انطباق

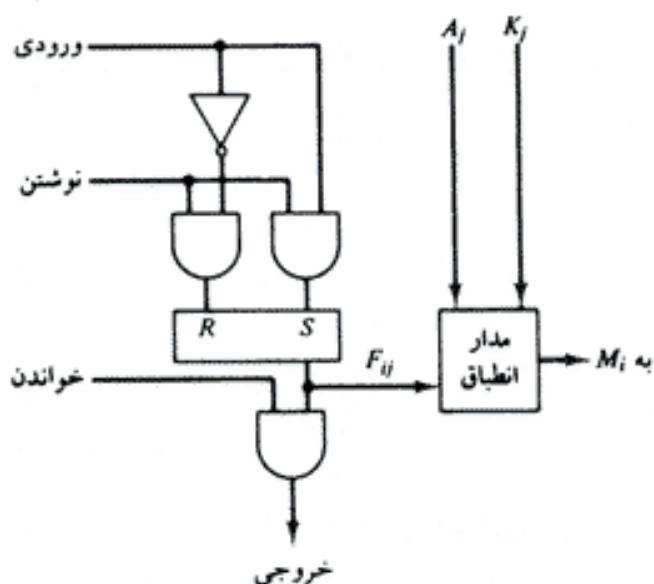
مدار منطقی برای هر کلمه را می‌توان از الگوریتم مقایسه دو عدد دودویی بدست آورد. ابتدا بیت‌های کلید را نادیده می‌گیریم و آرگومان  $A$  را با بیت‌های ذخیره شده در سلول‌های کلمه مقایسه می‌کنیم. کلمه  $A$  برابر با آرگومان در  $A$  است بشرطی که  $A_{ij} = F_{ij}$  بازاء  $i=1,2,\dots,n$  باشد. اگر هر دو بیت مساوی ۱، یا ۰ باشند، تساوی دو بیت بصورت تابع بول با عبارت زیر بیان می‌شود.

$$x_j = A_j F_{ij} + A'_j F'_{ij}$$

که در آن  $x_j = 1$  است اگر هر دو جفت بیت در مکان  $j$  مساوی باشد، در غیراینصورت  $x_j = 0$  خواهد بود. برای اینکه کلمه  $A$  برابر آرگومان  $M_i$  شود، تمام متغیرهای  $x_j$  باید برابر ۱ باشند. این وضعیت شرط ۱ 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 بیت‌های انطباق یافته در یک کلمه است.



شکل ۱۲-۸ یک سلول از حافظه تداعیگر

اکنون بیت کلید  $K_j$  را در مدار مقایسه می‌گنجانیم. 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  $x_j + 0 = x_j$  و  $x_j + 1 = 1$  هر وقت هم  $K_j = 0$  باشد، آنگاه  $K'_j = 1$  و  $x_j + 1 = 1$  هست. 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 خواهد داشت. فقط هنگامی که  $K_j = 1$  باشد مقایسه بیت ها تأثیر خواهد داشت. اکنون می‌توان مدار انطباق برای کلمه آ در یک حافظه تداعیگر را با تابع بولی زیر بیان کرد:

$$M_i = (x_1 + K'_1)(x_2 + K'_2)(x_3 + K'_3) \cdots (x_n + K'_n)$$

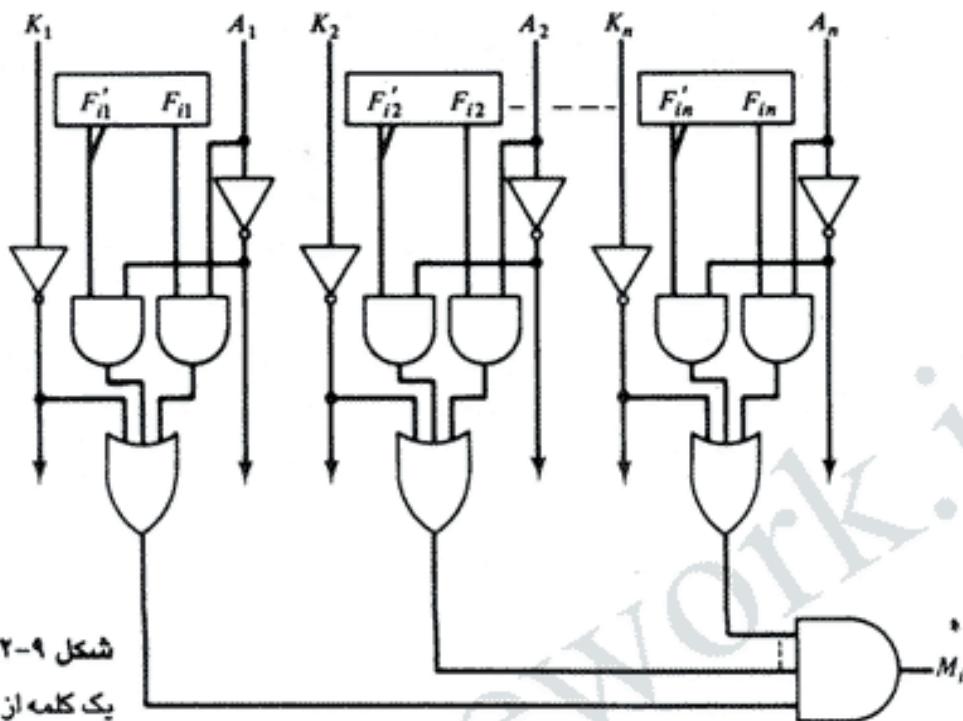
هر جمله در عبارت بشرطی 1 است که  $K_j$  متناظر آن 0 باشد. اگر  $K_j = 1$  باشد، جمله بسته به مقدار  $x_j$  برابر با 0 یا 1 خواهد بود. اگر همه جملات 1 باشد انطباق صورت گرفته و  $M_i$  برابر 1 خواهد بود. اگر تعریف اولیه  $x_j$  را بجای آن قرار دهیم، تابع بول فرق بصورت زیر بیان می‌شود:

$$M_i = \prod_{j=1}^n (A_j F_j + A'_j F'_j + K'_j)$$

که در آن  $\prod$  سمبول 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 ۱۲-۹ دیده می‌شود. هر سلول دو گیت AND و یک گیت OR نیاز دارد. معکوس کننده های  $A_j$  و  $K'_j$  یک بار برای هر ستون لازم است و برای همه بیت های ستون بکار می‌روند. خروجی تمام گیت های OR در سلول های 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 انطباق 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 ۱۲-۹ 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 ۱ است بطور متواالی 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 دهنی متواالی هر مکان میسر است. به این ترتیب مدار مورداداستفاده به یک حافظه با دستیابی اتفاقی (RAM) برای نوشتن و به یک حافظه قابل آدرس دهنی با محتوا برای خواندن تبدیل خواهد شد. در اینجا مزیت این است که آدرس برای ورودی مانند یک حافظه RAM دیگد می‌شود. در نتیجه بعضی داشتن  $m$  خط آدرس یعنی یک خط برای هر کلمه در حافظه.

تعداد خطوط آدرس بوسیله دیکدر به  $\lceil \log_2 m \rceil$  است.

اگر که قرار باشد کلمه ناخواسته‌ای پاک شود و کلمات جدیدی یک به یک وارد شوند، به ثبات خاصی برای تمايز بین کلمات فعل و غیر فعل نیاز است. تعداد بیت‌های این ثبات که گاهی اوقات ثبات نشانه نامیده می‌شود، به تعداد کلمات در حافظه خواهد بود. به ازاء هر کلمه فعل ذخیره شده در حافظه، بیت متناظری در ثبات نشانه 1 می‌شود. با 0 کردن بیت نشانه کلمه متناظر آن در حافظه پاک می‌شود. برای ذخیره کلمات در حافظه، ثبات نشانه مرور می‌شود تا اینکه اولین 0 دیده شود. باین ترتیب اولین کلمه غیر فعل و نیز محل برای نوشتن یک کلمه جدید مشخص می‌شود. پس از ذخیره کلمه جدید در حافظه، این کلمه با 1 کردن بیت نشانه فعل می‌شود. اگر قرار باشد یک مکان خالی با تمامًا 0 نشان داده شود می‌توان پس از حذف کلمه ناخواسته از حافظه، آن راپاک کرد. بعلاوه کلماتی که دارای بیت نشانه 0 هستند باید با آرگومان کلمه پوشش یابند (علاوه بر پوشش که با بیت‌های زیر انجام می‌شود) تا فقط کلمات فعل مقایسه شوند.

## ۱۲-۵ 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 محلیت ارجاع خوانده می‌شود. دلیل این خاصیت را می‌توان با بررسی این موضوع که یک برنامه کامپیوترو نواعاً بشکل مستقیم الخط پیش می‌رود و ضمن پیشروی با حلقه‌ها و زیر روال‌ها مکرراً برخورد می‌شود دریافت. وقتی که یک حلقه برنامه اجرا می‌شود، CPU متواتاً به مجموعه‌ای از دستورات که حلقه را در حافظه تشکیل می‌دهند رجوع می‌نماید. هر بار یک زیر روال فراخوانی شود، مجموعه دستورالعمل های آن از حافظه برداشت می‌شود. بنابراین حلقه‌ها و زیر روال‌ها تمایل به محلی کردن ارجاع به حافظه برای برداشت دستورالعمل‌ها دارند. ارجاع داده به حافظه نیز، تا حدودی میل به محلی بودن دارد. رویه‌های جستجوی جدول مکرراً به بخشی از حافظه که جدول در آن است مراجعه می‌کنند. رویه‌های تکراری به مکان‌های حافظه مشترک مراجعه می‌کنند و آرایه‌های اعداد به بخشی محلی از حافظه محدودند. نتیجه همه این ملاحظات خاصیت محلیت ارجاع است، که می‌گوید در فواصل زمانی کوتاه، آدرس‌های تولید شده توسط یک برنامه مکرراً به چند ناحیه محلی از حافظه اشاره می‌کنند، در حالی که بقیه برنامه بدفعات کمتری مورد دستیابی قرار می‌گیرند.

اگر بخش‌های فعل برنامه و داده‌ها در یک حافظه کوچک و سریع جای داده شوند، زمان دستیابی متوسط می‌تواند کاهش یافته و باین ترتیب زمان کل اجرای برنامه تقلیل می‌یابد. چنین حافظه سریع و کوچک، حافظه کش<sup>۱</sup> خوانده می‌شود. این حافظه همان طور که در شکل ۱۲-۱ دیده شد بین CPU و حافظه اصلی فوار می‌گیرد. 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 دستورالعمل‌ها و داده‌هایی که بیشتر دستیابی می‌شوند 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 محلیت ارجاع به حافظه، بخش عمدۀ اطلاعات یا در واقع نیازهای حافظه‌ای 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 می‌شود. اگر کلمه موردنظر در کش یافت شود، از این حافظه سریع خوانده می‌شود. اگر کلمه‌ای که بوسیله CPU آدرس دهنده شده در کش یافته نشود، حافظه اصلی برای خواندن کلمه آدرس دهنده می‌شود. سپس بلاکی از کلمات از جمله کلمه‌ای که در آن لحظه دستیابی شد از حافظه اصلی به حافظه کش انتقال می‌یابد. اندازه بلاک می‌تواند از یک کلمه (کلمه‌ای که در لحظه اول دستیابی می‌شود) و 16 کلمه مجاور آن باشد. بدین ترتیب، مقداری داده به کش انتقال می‌یابد و لذا ارجاعات آینده به حافظه، کلمات مورد نیاز را در حافظه کش سریع بیابد.

عملکرد حافظه کش غالباً براساس کمیتی که نسبت برد<sup>1</sup> نامیده می‌شود سنجیده می‌گردد. هنگامی که CPU به حافظه مراجعه کند و کلمه موردنظر را در حافظه کش بیابد گفته می‌شود برد صورت گرفته است. اگر کلمه موردنظر در حافظه کش یافته نشود، مسلماً در حافظه اصلی است و باخت<sup>2</sup> صورت گرفته است. نسبت تعداد برد‌ها تقسیم بر تعداد کل ارجاعات CPU به حافظه (برد بعلاوه باخت) نسبت برد است. بهترین نسبت برد اندازه‌گیری شده تجربی، اجرای برنامه‌های نمونه در کامپیوتر و سنجش تعداد برد‌ها و باخت‌ها در یک فاصله زمانی معین است. در این رابطه نسبت برد‌های 0.9 و بالاتر گزارش شده است. این نسبت برد بالا اعتبار خاصیت محلیت ارجاع را تأیید می‌کند.

زمان متوسط دستیابی به حافظه برای یک سیستم کامپیوتر با استفاده از حافظه کش می‌تواند بمقدار قابل ملاحظه‌ای 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 می‌باشد.

مشخصه اصلی حافظه کش زمان دستیابی سریع آنست. بنابراین، وقتی جستجویی برای کلمه در کش صورت گیرد انلاف زمانی کوتاه بوده یا اصلاً زمانی تلف نمی‌شود. انتقال داده از حافظه اصلی به حافظه کش فرآیند نگاشت<sup>3</sup> نام دارد. سه نوع رویه نگاشت هنگام بررسی سازمان حافظه کش عملاً مورد توجه است.

#### ۱- نگاشت تداعیگر<sup>4</sup>

1- Hit ratio

2- Miss

3- Mapping

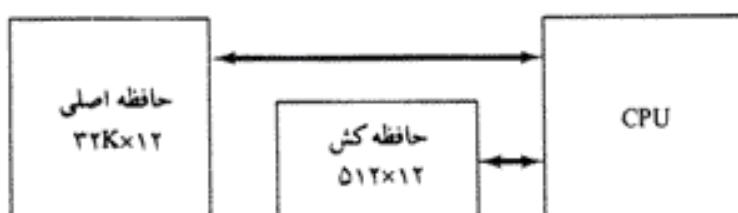
4- Associative Mapping

۲- نگاشت مستقیم<sup>۱</sup>۳- نگاشت تداعیگر مجموعه‌ای<sup>۲</sup>

برای کمک به درک بحث این سه رویه نگاشت ما از مثال خاصی برای سازمان حافظه که در شکل ۱۲-۱۰ نشان داده شده است استفاده می‌کنیم. حافظه اصلی می‌تواند ۳۲ کلمه ۱۲ بیتی را ذخیره کند. حافظه کش قادر است ۵۱۲ 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 ۱۵ بیتی را به حافظه کش ارسال می‌نماید. اگر بردی صورت گیرد، CPU داده ۱۲ 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

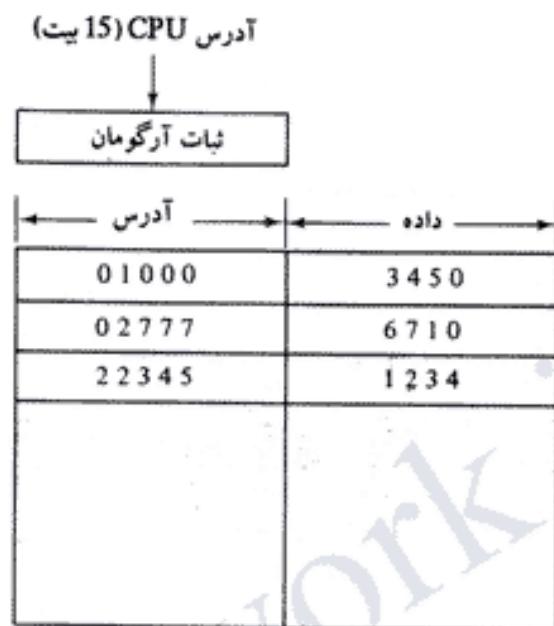
در سریع‌ترین و انعطاف‌پذیرترین حافظه کش از یک حافظه تداعیگر استفاده می‌شود. این سازمان در شکل ۱۲-۱۱ نشان داده شده است. حافظه تداعیگر هم آدرس و هم محتوای (داده) کلمه حافظه را ذخیره می‌کند. این توانایی اجازه می‌دهد تا هر مکان در حافظه کش بتواند هر یک از کلمات حافظه اصلی را ذخیره کند. دیگر، سه کلمه را که در حال حاضر در حافظه کش ذخیره شده نشان می‌دهد. مقدار آدرس ۱۵ بیتی بصورت یک عدد هشت هشتی پنج رقمی و کلمه ۱۲ بیتی متناظر آن بصورت یک عدد هشت هشتی چهار رقمی نشان داده شده است. یک آدرس ۱۵ 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 ۱۲ 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 موردنظر 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 ۱۲-۱۰ مثالی از حافظه کش

1- Direct Mapping

2- Set - Associative Mapping

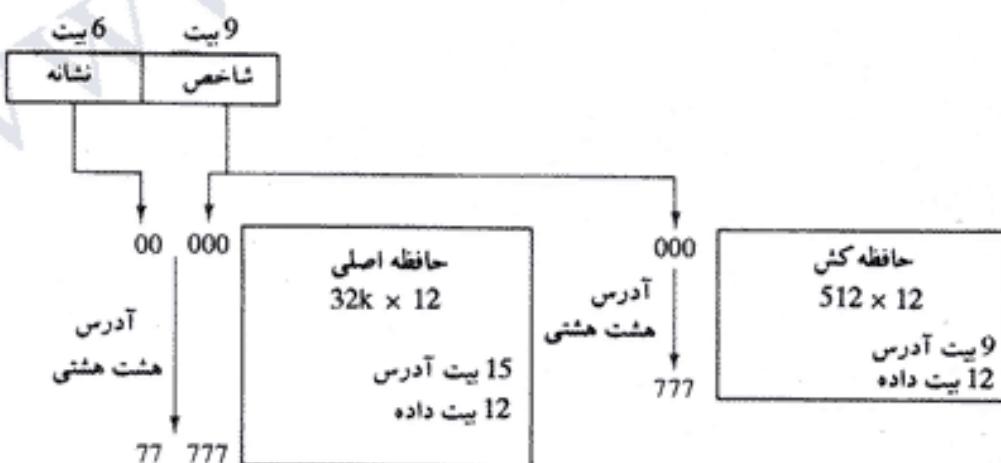


شکل ۱۲-۱۱ حافظه کش با نگاشت تداعیگر  
(همه اعداد در مبنای هشت هستند)

سلول‌های موجود در حافظه کش بصورت چرخشی<sup>۱</sup> جایگزین شوند. این رویه، سیاست اولین ورودی - اولین خروجی (FIFO) نام دارد.

#### نگاشت مستقیم

حافظه‌های تداعیگر در مقایسه با حافظه‌های 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 آدرس‌دهی بین حافظه‌های کش و اصلی

۱- Round - Robin order

قرار گرفته است. آدرس 15 بیتی CPU به دو میدان تقسیم شده است. نه بیت کم ارزشتر میدان شاخص<sup>۱</sup> و شش بیت باقیمانده میدان نشانه<sup>۲</sup> را تشکیل می‌دهند. شکل نشان می‌دهد که حافظه اصلی نیاز به آدرسی دارد که هم شامل بیت‌های شاخص و هم شامل بیت‌های نشانه می‌باشد. تعداد بیت‌های میدان شاخص برابر با تعداد بیت‌های 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 می‌شود:  $k$  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 می‌کند. سازمان داخلی کلمات در حافظه کش در شکل ۱۲-۱۳ (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، بیت‌های نشانه همراه با بیت‌های داده ذخیره می‌شوند. وقتی که CPU یک سیگнал تقاضای حافظه تولید می‌کند، میدان شاخص برای دستیابی به حافظه کش بکار می‌رود. میدان نشانه آدرس CPU، با نشانه موجود در کلمه خوانده شده از حافظه کش مقایسه می‌شود. اگر دو نشانه با هم تطبیق کنند یک برد صورت می‌گیرد 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 می‌شود. این کلمه سپس همراه با نشانه جدید در حافظه کش ذخیره شده و جایگزین مقدار قبلی می‌شود. عیب نگاشت مستقیم

آدرس شاخص

نشانه	داده
000	00 1220
777	02 6710

(ب) حافظه کش

آدرس حافظه

داده‌های حافظه
1220
2340
3450
4560
5670
6710

(الف) حافظه اصلی

شکل ۱۲-۱۳ سازمان حافظه  
کش مستقیم بر نگاشت مستقیم

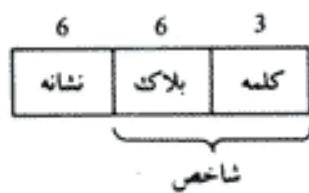
1- Index Field

2- Tag Field

این است که اگر دو یا چند کلمه که آدرسها بیشان دارای شاخص یکسانی ولی نشانه متفاوت‌اند مرتباً دستیابی شوند نسبت برد به عیزان قابل توجهی نداشته باشد. با این وجود این امکان باین ترتیب به حداقل رسانده شده است که این چنین کلمات در محدوده آدرسها نسبتاً دور از هم قرار دارند (در این مثال مضرب 512 مکان). برای درک نحوه عملکرد سازمان نگاشت مستقیم مثال عددی شکل ۱۲-۱۳ 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 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 ۱۲-۱۴ نشان داده شده است. میدان شاخص اکنون به دو بخش تقسیم شده است: میدان بلاک<sup>۱</sup> و میدان کلمه<sup>۲</sup>. در یک حافظه کش 512 کلمه‌ای، 64 بلاک 8 کلمه‌ای وجود دارد زیرا  $512 = 64 \times 8$  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	01	3450
	007	01	6578
	010		
	017		
بلاک 1			
بلاک 63	770	02	
	777	02	6710



شکل ۱۲-۱۴ حافظه کش با نگاشت مستقیم و اندازه بلاک 8 کلمه

1- Block Field

2- Word Field

### نگاشت تداعیگر مجموعه‌ای

قبل 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 اقلام 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  $36 = (6+12) \times 2$  بیت خواهد بود. هر آدرس شاخص 9 بیتی می‌تواند برای 512 کلمه جا مشخص کند. بنابراین اندازه حافظه کش  $512 \times 36$  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 ۱۲-۱۳ (الف) لیست شده‌اند. کلمات در آدرس‌های 01000 و 02000 از حافظه اصلی در حافظه کش با آدرس شاخص 000 شده‌اند. کلمات در آدرس‌های 02777 و 00777 در آدرس شاخص 777 در حافظه ذخیره شده‌اند. بطور مشابه، کلمات در آدرس‌های 00777 و 02777 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 موقع 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 هشت)

است. نسبت برد با افزایش اندازه مجموعه افزایش می‌یابد زیرا در اینصورت کلمات بیشتری با شاخص یکسان ولی نشانه‌های متفاوت می‌تواند در حافظه کش قرار داشته باشند. اما افزایش اندازه مجموعه تعداد بیت‌های کلمات کش را افزایش داده و مدارهای مقایسه پیچیده‌تری را نیاز خواهد داشت.

وقتی که باختنی در حافظه کش تداعیگر مجموعه‌ای 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<sup>۱</sup> (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 می‌کند. الگوریتم LRU، اطلاعات CPU را بر می‌گزیند. هر دو روشن FIFO و LRU می‌توانند با افزایش چند بیت در هر کلمه کش پیاده سازی شوند.

### نوشتن در حافظه کش

یکی از جنبه‌های مهم سازمان کش به درخواست‌های نوشتن مربوط است. وقتی که CPU در حین عمل خواندن کلمه‌ای را در کش می‌یابد، حافظه اصلی در عمل انتقال دخالتی نمی‌کند. با این وجود اگر عمل یک نوشتن باشد، دو راه برای عملکرد سیستم وجود دارد.

ساده‌ترین و متداول‌ترین رویه‌های مورد استفاده بهنگام کردن حافظه اصلی در مقابل هر نوشتن حافظه است، که حافظه کش هم به موازات آن، اگر دارای کلمه موجود در آدرس مشخص شده باشد، بهنگام می‌گردد. این روشن توییسی<sup>۲</sup> خوانده می‌شود. مزیت این روشن اینست که حافظه اصلی همواره داده یکسانی با حافظه کش دارد. این مشخصه در سیستم‌هایی که از روشن انتقال با دستیابی مستقیمه حافظه استفاده می‌کنند اهمیت دارد. روشن مذکور اعتبار داده‌هایی را که در تمام مدت در حافظه اصلی اند نصیین می‌کند، بنا براین وسیله DMA از طریق I/O که از طریق DMA ارتباط دارد جدیدترین اطلاعات را دریافت خواهد کرد.

روشن دوم، روشن پس توییسی<sup>۳</sup> پس توییسی خوانده می‌شود. در این روشن فقط مکان کش در حین عمل نوشتن بهنگام می‌شود. سپس مکان مزبور با یک پرچم مشخص می‌شود تا بعداً که کلمه موجود در آن از حافظه کش خارج می‌شود بداخل حافظه اصلی کپی شود. دلیل بکارگیری روشن پس توییسی 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 می‌گیرد، قدیمی بودن کپی واقع در حافظه اصلی مهم نیست، زیرا تقاضاهای کلمه از حافظه کش تأمین می‌شوند. فقط هنگامی که کلمه از حافظه کش خارج می‌شود لازم است نسخه صحیحی در حافظه اصلی نوشته شود. نتایج تحلیلی نشان می‌دهد که تعداد نوشتن‌ها در حافظه در یک برنامه، بین ۱۰ تا ۳۰ درصد کل ارجاعات به حافظه است.

### مقدار دهی اولیه در حافظه کش

جنبه دیگری از سازمان کش که باید مورد توجه قرار گیرد مقدار دهی 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 مقدار دهی می گردد. پس از مقدار دهی اولیه حافظه کش خالی تصویر خواهد شد، ولی در اصل حاوی مقداری داده های بی اعتبار است. مرسم است تا به هر کلمه در کش یک بیت اعتبار<sup>۱</sup> اختصاص یابد تا مشخص شود آیا کلمه حاوی داده معتبر است یا خیر.

حافظه کش با صفر کردن تمام بیت های معتبرش مقدار دهی اولیه می شود. اولین باری که یک کلمه خاص از حافظه اصلی بار شود بیت اعتبار آن به مقدار ۱ تغییر می یابد و تا مقدار دهی اولیه بعدی همچنان برابر ۱ باقی می ماند. معرفی بیت اعتبار بمعنی این است که آن کلمه با هیچ کلمه ای جایگزین نشود مگر اینکه بیت اعتبار ۱ باشد و عدم تطابق نشانه ها رخ دهد. اگر بیت اعتبار ۰ باشد، کلمه جدید بطور اتوماتیک جایگزین داده بی اعتبار می گردد. بنابراین شرط مقدار دهی اولین این اثر را دارد که تا زمانی که حافظه نهان از داده های معتبر پر شود موجب باخت می شود.

### ۱۲-۶ 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<sup>۲</sup>، مفهومی مورد استفاده در برخی از سیستم های کامپیوتری بزرگ است که این امکان را به کاربر می دهد تا برنامه را بگونه ای بنویسید 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 تصویر 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<sup>۳</sup> و مجموعه این آدرسها فضای آدرس<sup>۴</sup> خوانده می شود. یک آدرس در حافظه اصلی، مکان<sup>۵</sup> یا آدرس فیزیکی<sup>۶</sup> نامیده می شود.

1- Valid Bit

2- Virtual Memory

3- Virtual Address

4- Address Space

5- Location

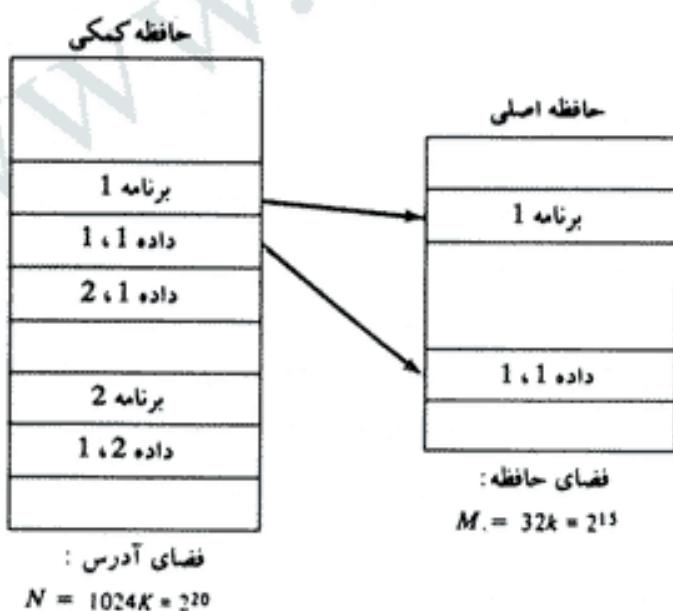
6- Physical Address

مجموعه چنین مکان‌هایی فضای حافظه<sup>۱</sup> خوانده می‌شود. بنا بر این فضای آدرس مجموعه‌ای از آدرس‌های تولید شده بوسیله برنامه‌ها می‌باشد، که به دستورالعمل‌ها و داده‌ها اشاره می‌کنند؛ و فضای حافظه هم متشکل از مکان‌های واقعی حافظه اصلی می‌باشد، که مستقیماً برای پردازش قابل آدرسی دهی 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  $2^{15} = 32K$  است. فرض کنید که کامپیوتر دارای حافظه‌ای مجازی برای ذخیره  $1024K = 2^{20}$  کلمه باشد. بنا بر این 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  $1024K = N$  و  $32K = M$  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 اجرای است. برنامه  $1$  و بخشی از داده مربوط به آن از حافظه کمکی به حافظه اصلی طبق شکل ۱۲-۱۶ انتقال یافته. 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



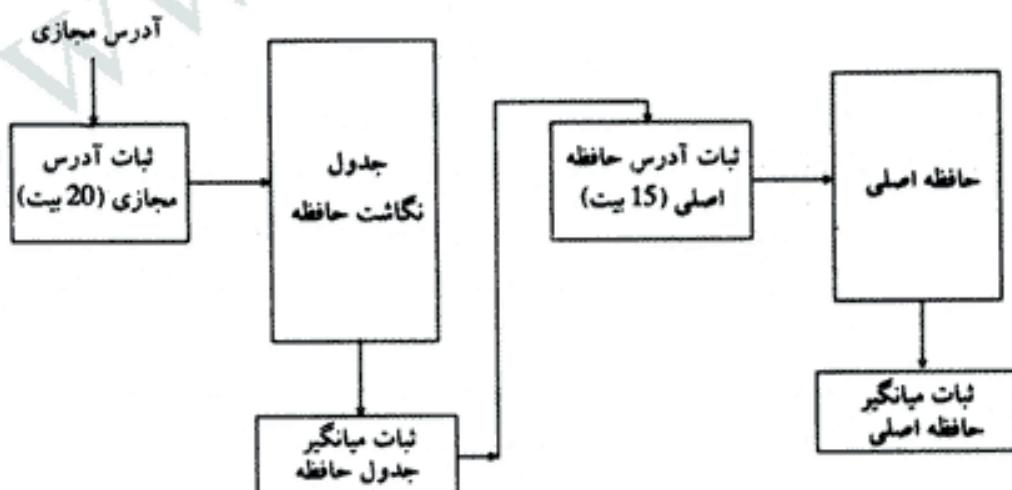
شکل ۱۲-۱۶ رابطه بین فضای آدرس و حافظه در یک سیستم حافظه مجازی

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 کارآیی عمل انتقال، حافظه کمکی یک رکورد کامل را به حافظه اصلی منتقل می کند). سپس همانطور که در شکل ۱۲-۱۷ 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<sup>۱</sup> می تواند در یک حافظه جداگانه طبق شکل ۱۲-۱۷ 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 تداعیگر 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<sup>۲</sup> اشاره به گروه هایی از فضای حافظه هم اندازه دارد. مثلاً اگر یک صفحه یا بلاک از 1K



شکل ۱۲-۱۷ 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"<sup>۱</sup> 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 ۱۲-۱۸ 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 p نشان داده شود: یک آدرس شماره صفحه و یک خط در صفحه. در کامپیوتري با  $2^p$  کلمه در هر صفحه، برای مشخص کردن آدرس خط و یقینه بیت های مرتبه بالاتر از حافظه مجازی برای مشخص کردن شماره صفحه مورد استفاده قرار می گیرد. در مثال شکل ۱۲-۱۸، 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 ۱۲-۱۹ نشان داده شده است. جدول صفحات حافظه از هشت کلمه تشکیل می شود، یعنی یک کلمه بازه هر صفحه آدرس جدول 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 ۱، ۲، ۵ و ۶ هم اکنون در حافظه اصلی بترتیب بلوک های ۰، ۳، ۱ و ۲ فرار دارند. یک بیت حضور در هر خانه مشخص می کند که صفحه از حافظه کمکی به حافظه اصلی منتقل شده است یا خیر. مقدار ۰ 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 پرازشتر 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 ۱ را داشته

صفحه ۰
صفحه ۱
صفحه ۲
صفحه ۳
صفحه ۴
صفحه ۵
صفحه ۶
صفحه ۷

فضای آدرس:

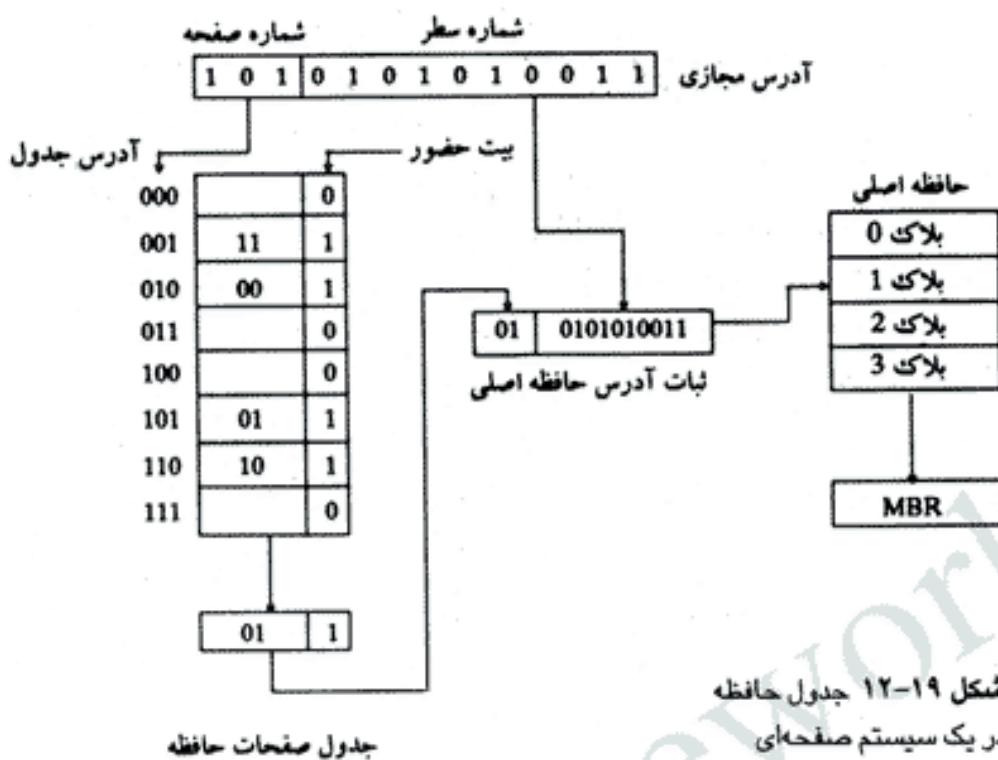
$$N = 8K = 2^{13}$$

بلاک ۰
بلاک ۱
بلاک ۲
بلاک ۳

فضای حافظه:

$$M = 4K = 2^{12}$$

شکل ۱۲-۱۸ تقسیم فضای آدرس و فضای حافظه به دسته های 1K کلمه ای



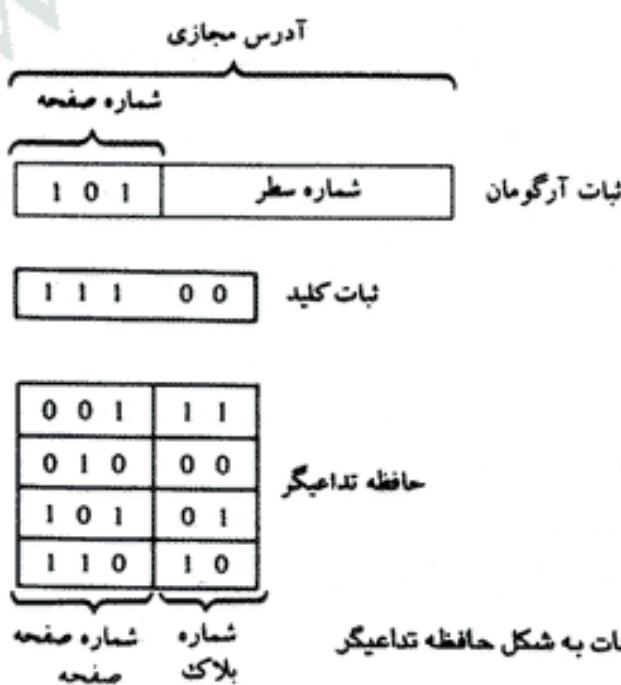
### جدول صفحات از حافظه تداعیگر

یک جدول صفحات حافظه با دستیابی تصادفی کارآیی لازم را از نظر بهره‌گیری دارد. 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 حداقل  $m$  بلاک حاوی شماره بلاک و بقیه خالی خواهد بود. یعنوان 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 دهنده 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 ۱۲-۱۹ ملاحظه شد به جای جدول حافظه صفحه از نوع با دستیابی تصادفی از آرایه حافظه تداعیگر استفاده می‌کنیم که هر یک از واردۀ های حافظه تداعیگر از دو میدان ساخته شده است (شکل ۱۲-۲۰). 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 نرم‌افزار مدیریت حافظه همراه با هم معماری حافظه مجازی را تشکیل می‌دهند.



شکل ۱۲-۲۰ 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 بهارجاع به صفحه‌ای کند که هنوز در حافظه کمکی قرار دارد. این وضعیت فقدان صفحه<sup>۱</sup> نامیده می‌شود. هنگامی که فقدان صفحه رخ دهد اجرای برنامه جاری متوقف می‌شود تا اینکه صفحه لازم به حافظه اصلی منتقل گردد. چون بارگردان 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 برنامه بعدی 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 سریگیرد.

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 متدائل‌تر 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 پشت 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 وارد حذف می‌شود. تعیین صفحه‌ای که قرار است حذف شود آسان است زیرا شماره شناسایی آن در بالای پشت 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 یعنوان کاندید حذف، نسبت به قدیمی‌ترین صفحه بار شده FIFO، جذاب‌تر می‌باشد. الگوریتم LRU را می‌توان با اختصار یک شمارنده برای هر صفحه‌ای که در حافظه اصلی قرار دارد پیاده‌سازی کرد. هر وقت صفحه‌ای مورد ارجاع قرار گیرد، شمارنده مربوط‌اش 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<sup>۲</sup> خوانده می‌شوند، زیرا شمارش، سن آنها را معین می‌کند، یعنی صفحه متناظر با آنها چه مدت قبل مورد ارجاع قرار گرفته است.

## 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 برنامگی 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 ۱۲-۶ 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<sup>۱</sup> 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، جداول سمبیل‌ها، یا برنامه یک کاربر.

استفاده مشترک از برنامه‌ها بخش لاینکی از یک سیستم چند برنامه‌ای است. مثلاً چندین کاربر که مایلند برنامه‌های فرتون خود را ترجمه<sup>۲</sup> 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 برنامه‌های است. 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 کامپیوتروی توسط کاربران عادی است. برنامه های معینی که جنبه سری دارند باید از دسترسی پرسنل غیرمجاز حفاظت شوند تا از سوء استفاده در فعالیت های محروم نمایند. آدرس تولید شده بوسیله یک برنامه قطعه ای آدرس منطقی خوانده می شود. این آدرس مشابه آدرس مجازی است جزاینکه فضای آدرس منطقی در عوض تقسیم به صفحات با طول ثابت، به قطعه هایی با طول متغیر تقسیم می گردد. 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 حفاظتش را نیز همراه خود دارد. برنامه های مشترک، در یک قطعه منحصر بفرد در فضای آدرس منطقی هر کاربر قرار داده می شود، تا بتوان از یک نسخه فیزیکی واحد بطور مشترک استفاده کرد. وظیفه واحد مدیریت حافظه تبدیل آدرس های منطقی به آدرس های فیزیکی مانند تبدیل در حافظه مجازی است.

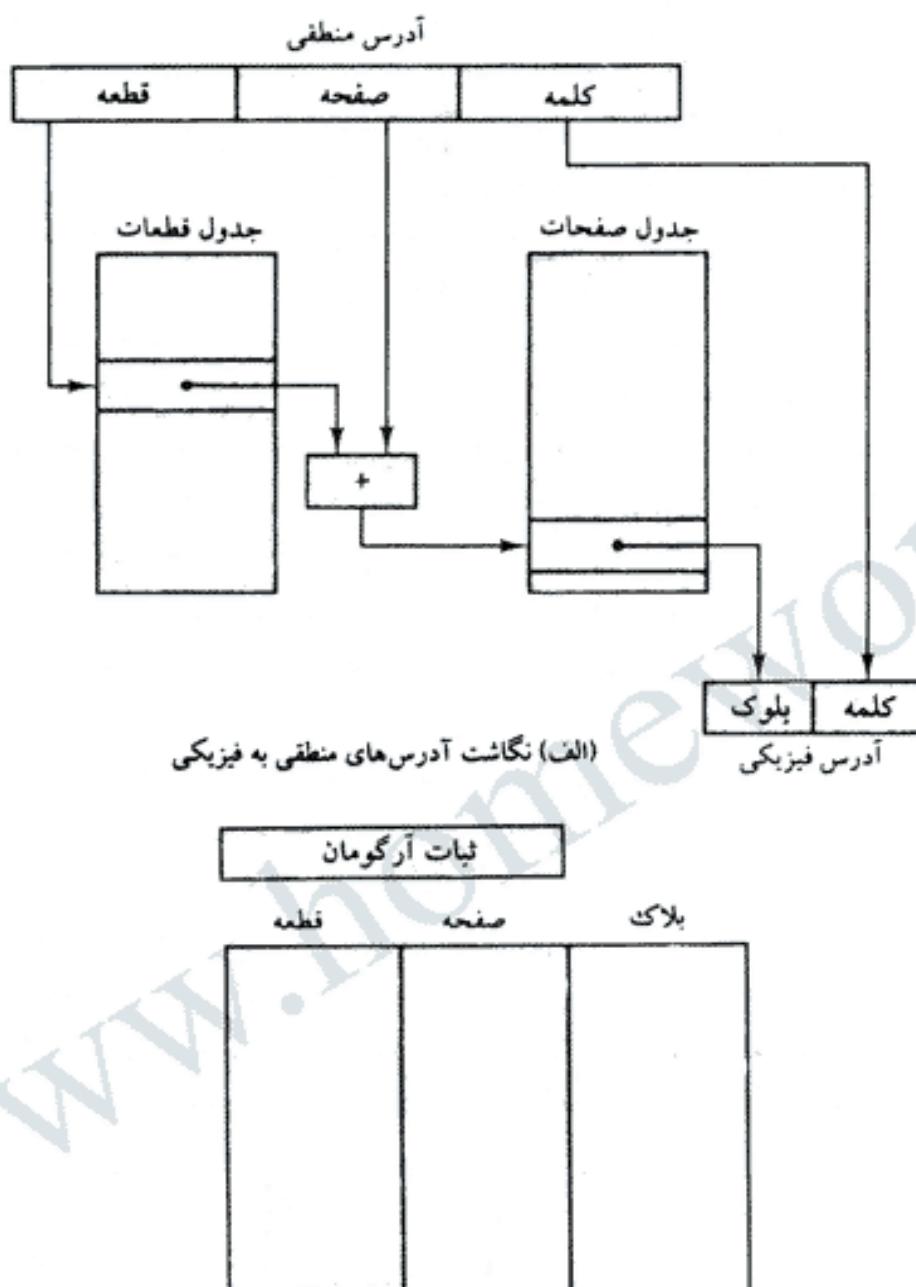
### نگاشت قطعه - صفحه‌ای

قبل اگفته شده که خاصیت فضای منطقی این است که از قطعات با طول متغیر استفاده می کند. طول هر قطعه می تواند 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 ۱۲-۲۱ را در نظر بگیرید. آدرس منطقی به سه میدان تقسیم شده است. میدان قطعه یک شماره قطعه را مشخص می کند. میدان صفحه، آن را در داخل قطعه و میدان کلمه نیز کلمه را در صفحه مشخص می کند. یک میدان صفحه با  $k^2$  بیت قادر است تا  $2^k$  صفحه را معین کند. یک شماره قطعه می تواند، فقط یک صفحه یا تا  $2^k$  صفحه متناظر باشد. بنابراین طول یک قطعه برطبق تعداد صفحاتی که به آن اختصاص داده می شود متغیر است.

تبدیل آدرس منطقی به آدرس فیزیکی بوسیله دو جدول انجام می شود، شکل ۱۲-۲۱ (الف). شماره قطعه آدرس منطقی، آدرس جدول قطعه را مشخص می کند. واردہا در جدول قطعه، اشاره گر<sup>1</sup> آدرس برای پایه<sup>2</sup> (شروع) جدول صفحه می باشد. پایه جدول صفحه به شماره صفحه که در آدرس منطقی داده می شود اضافه می شود. مجموع دو کمیت اشاره گر آدرسی را برای یک واردہ در جدول صفحه تولید می کند. مقداری که از جدول صفحات استخراج می شود شماره بلاک حافظه فیزیکی را مشخص می نماید. با کنار هم قراردادن میدان بلاک و میدان کلمه، آدرس تبدیل شده فیزیکی نهایی بدست می آید. 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 می نماید. برای اجتناب از این کاهش سرعت، یک حافظه تداعیگر سریع، برای نگهداری آخرین واردہ های ارجاع شده جدول بکار می رود. (این نوع حافظه گاهی با نام دستی ترجمه یا TLB خوانده می شود). اولین باری که یک بلاک ارجاع داده شود، مقدار آن همراه با

1- Pointer

2- Base



شکل ۱۲-۲۱ نگاشت در واحد مدیریت حافظه قطعه-صفحه‌ای

شماره‌های قطعه و صفحه طبق شکل ۱۲-۲۱ (ب) وارد یک حافظه تداعیگر می‌شود. بنابراین روند تبدیل ابتدا با جستجوی تداعیگر با شماره‌های قطعه و صفحه آزمایش می‌شود. اگر نتیجه موفقیت آمیز باشد، تأخیر تبدیل تنها متعلق به حافظه تداعیگر است. اگر انطباقی رخ ندهد، جدول تبدیل شکل ۱۲-۲۱ (الف) که آهسته‌تر است بکار رفته و نتیجه تبدیل بداخل حافظه تداعیگر برای کاربرهای آینده انتقال می‌یابند.

### 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  $4096 \times 256$  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

قطعه	صفحه	کلمه
4	8	8

(الف) قالب آدرس منطقی: 16 قطعه 256 صفحه ای که در هر صفحه 256 کلمه قرار دارد

بلک	کلمه
12	8

حافظه فیزیکی  
 $2^{20} \times 32$

(ب) قالب آدرس فیزیکی: 4096 بلاک 256 کلمه ای که هر کلمه 32 بیت دارد

شکل ۱۲-۲۲ مثالی از آدرس های منطقی و فیزیکی

آدرس شانزده شانزده	شماره صفحه
60000	صفحه 0
60100	صفحه 1
60200	صفحه 2
60300	صفحه 3
60400	صفحه 4
604FF	

(ب) بلاک های حافظه متناظر با قطعه - صفحه ها

(الف) تخصیص آدرس های حافظه منطقی

شکل ۱۲-۲۳ مثالی از تخصیص آدرس های حافظه منطقی و فیزیکی

بصورت یک عدد پنج رقمی شانزده شانزده 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 آدرس های 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 بیتی 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 ۱۲-۲۴ (ب) نشان می دهد که صفحات 2 و 4 از قطعه 6 قبل از مورد ارجاع قرار گرفته اند و بنابراین شماره بلاک متناظر آنها در حافظه تداعیگر ذخیره شده است.

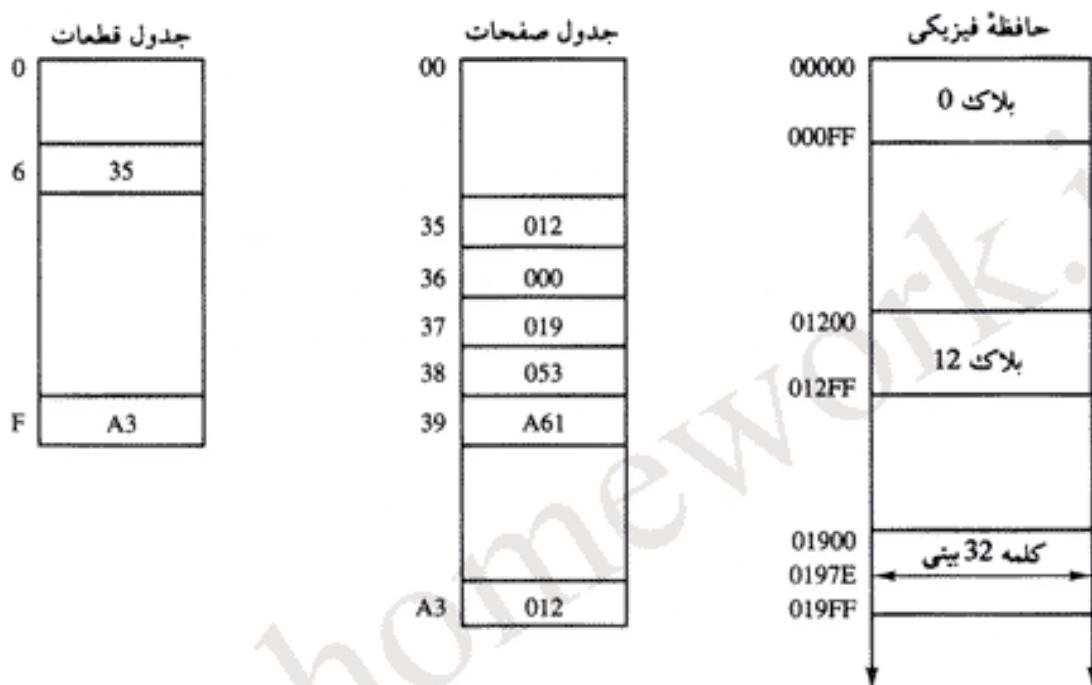
با توجه به مثال فوق آشکار است که سیستم مدیریت حافظه قادر است هر تعداد صفحات را به هر قطعه اختصاص دهد. هر صفحه منطقی را می توان به هر یک از بلاک های حافظه فیزیکی نگاشت کرد. صفحات را می توان بسته به نیاز فضای لازم برای حافظه، به بلاک های مختلفی در حافظه جابجا کرد. تنها بهنگام سازی موردنیاز تغییر شماره 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 دارد، و همانطور که در شکل ۱۲-۲۴ (الف) دیده می شود به بلاک 12 نگاشت می شود.

### حافظت 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 بست 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 بست 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 عدها در مبنای شانزده)

آدرس فیزیکی است. این کار را می‌توان با گنجاندن اطلاعات حفاظتی در جدول قطعات یا ثبات قطعه ساخت افزار مدیریت حافظه انجام داد.

محتوای هر واردہ در جدول قطعه یا یک ثبات قطعه توصیفگر<sup>۱</sup> خوانده می‌شود. 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 ۱۲-۲۵ 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 می‌سازد. این آدرس در تبدیل یک آدرس منطقی به فیزیکی بسیار موثر است.

میدان طول<sup>۲</sup>، اندازه قطعه را با مشخص کردن حداقل تعداد صفحات قابل اختصاص به قطعه معین می‌کند. اگر شماره صفحه خارج از مرز قطعه باشد. تخطی از اندازه صورت گرفته است. به این ترتیب یک برنامه معین و داده‌های آن نمی‌توانند به حافظه‌ای که سیستم عامل به آن اختصاص نداده است دست یابد. میدان حفاظت در توصیفگر قطعه، حقوق دستیابی مجاز به قطعه مورد نظر را مشخص می‌کند. در سازمان قطعه - صفحه، هر یک از واردہ‌ها 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 می‌گیرد. بعضی از حقوق دستیابی مورد نظر مقیم در حافظه برای حفاظت برنامه‌ها عبارتند از:

- ۱- حقوق کامل خواندن و نوشتن
- ۳- فقط اجرا (حفاظت از برنامه‌ها)
- ۲- فقط خواندن (حفاظت در برابر نوشتن)
- ۴- فقط سیستم (حفاظت از سیستم عامل)

حقوق کامل خواندن و نوشتن هنگامی که یک برنامه در حال اجرای دستورالعمل‌های خودش باشد به آن داده می‌شود. حفاظت در برابر نوشتن برای استفاده مشترک از برنامه‌های سیستم از قبیل برنامه‌های ابزاری<sup>۳</sup> و روال‌های کتابخانه‌ای<sup>۴</sup> دیگر مفید است. این برنامه‌های سیستم در ناحیه‌ای از حافظه ذخیره می‌شوند تا بوسیله کاربران متعدد مورد استفاده مشترک قرار گیرند. این برنامه‌ها را سایر برنامه‌ها می‌توانند بخوانند، اما نوشتن در آنها مجاز نیست. این خصوصیت آنها را از هرگونه تغییر توسط برنامه‌های دیگر حفاظت می‌کند.

شرط فقط - اجرا برنامه‌ها را از کپی شدن محافظت می‌نماید. این شرط قطعه را محدود می‌کند تا فقط در حین فاز برداشت مورد ارجاع قرار گیرد و نه در حین فاز اجرا. بنابراین به کاربران اجازه می‌دهد

حفاظت	طول	آدرس مبنای
-------	-----	------------

شکل ۱۲-۲۵ قالب نوعی توصیفگر قطعه

1- Descriptor  
4- Library Routine

2- Access Leights

3- Utilitys Programs

تادستورالعمل های برنامه در قطعه را اجرا کند اما از خواندن دستورالعمل ها بعنوان داده به قصد کشی 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 غیر مجاز 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،  $8 \times 128$  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 کامپیووتری از یک RAM با ظرفیت  $1024 \times 1$  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  $8 \times 1024$  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 کامپیووتری از تراشه های RAM با  $256 \times 8$  و ROM با  $1024 \times 8$  استفاده می کند. سیستم کامپیووتر 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 بالرزشتر گذرگاه آدرس 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 کامپیووتر 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 می‌کند بکار رفته است. توضیح دهد چگونه می‌توان از این پیکربندی استفاده کرده و ظرفیت حافظه را به هشت بانک 32K باقی افزایش داد تا یک بانک 256K باقی حاصل شود.

۱۲-۷ یک سیستم دیسک مغناطیسی دارای پارامترهای زیر است،

$$T_s = \text{زمان متوسط لازم برای قرار گرفتن هد مغناطیسی روی یک شیار}$$

$$R = \text{سرعت چرخش دیسک بر حسب دور بر ثانیه}$$

$$N_t = \text{تعداد بیت‌ها در شیار}$$

$$N_s = \text{تعداد بیت‌ها در قطاع}$$

زمان متوسط  $T_a$  که برای خواندن یک قطاع لازم است چقدر می‌باشد.

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 منضم منطبق یک کلمه را در حافظه تداعیگر بدست آورید. به بیان دیگر، نشان دهد که  $M'$  مجموع توابع OR انحصاری است. دیاگرام منطقی را برای  $M'$  رسم نموده و با یک معکوس‌کننده آنرا پایان دهید تا  $M_1$  بدست آید.

۱۲-۱۰ تابع بول را برای منطبق منطبق 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 بیت‌های کلید صفر است مشخص کند.

۱۲-۱۲ (الف) دیاگرام منطقی تمام سلول‌های یک کلمه در حافظه تداعیگر را رسم کنید. منطق خواندن و نوشتن شکل ۱۲-۸ و منطق منطبق شکل ۱۲-۹ را نیز اضافه کنید.

(ب) دیاگرام منطقی تمام سلول‌های عمودی یک ستون (Z) را در حافظه تداعیگر رسم کنید. یک خط خروجی مشترک را برای همه بیت‌هایی که در یک ستون قرار دارند اضافه کنید.

(ج) با استفاده از دیاگرام‌های (الف) و (ب) نشان دهید که اگر خروجی  $M_1$  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 دهد چگونه می‌توان 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 موقعی که آرگومان پوشش نیافته بزرگتر از (اما نه مساوی با) کلمه موجود در حافظه تداعیگر باشد، بدست آوردید.

۱۲-۱۵ یک حافظه کش تداعیگر دوتایی از بلاک‌های چهار کلمه‌ای استفاده می‌کند. حافظه کش می‌تواند مجموعاً 2048 کلمه از حافظه اصلی را در خود جای دهد. اندازه حافظه اصلی  $128 \times 32$  است.

(الف) همه اطلاعات لازم برای ساختن این حافظه کش را فرموله کنید.

ب) اندازه حافظه کش چقدر است.

۱۲-۱۶ 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 نوشتمن 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 درنظر گرفتن سیکل خواندن چقدر است.

ب) زمان دستیابی متوسط سیستم برای تقاضای خواندن و نوشتمن چقدر است.

ج) نسبت برد با در نظر گرفتن سیکل های نوشتمن چقدر است.

۱۲-۱۷ یک مجموعه حافظه کش تداعیگر چهارتایی در هر مجموعه چهار کلمه دارد. یک رویه جایگزینی مبتنی بر الگوریتم بر قدمی ترین مورد استفاده (LRU) باشمارنده های دویتی 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 برای 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 کند: A, B, C, D, E, C, A, D, E, B, D, C, A, D, E, C, B, A, C, D, B, A (با B عنوان مقادیر اولیه شروع کنید که در آنها A قدمی ترین مورد استفاده است).

۱۲-۱۸ یک کامپیوتر دیجیتال دارای واحد حافظه 16×64K 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.

۱۲-۱۹ 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.

۱۲-۲۰ 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 واردہ های زیر است:

صفحه	بلاک
0	3
1	1
4	2
6	0

لیستی از آدرس های مجازی (به دهدی) 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 ۱۲-۱۸). 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 شانزدهی 012AF می گردد معین کنید.

۱۲-۲۳ فضای آدرس منطقی در یک سیستم کامپیوتی متشكل 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 دو دویی 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<sup>۱</sup> 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 با چند IOP هم ممکن است موجود باشد. همان طور که در بخش ۹-۱ ذکر شد، چند پردازنده‌ها بعنوان سیستم‌هایی با رشته‌های چند دستوری<sup>۲</sup> و رشته‌های چند داده‌ای<sup>۳</sup> دسته‌بندی می‌شوند.

بین سیستم‌های چند پردازنده و چند کامپیوتری شباهت‌هایی وجود دارد زیرا هر دو آنها عملیات همزمان را پشتیبانی می‌کنند. معهذا اختلاف مهمی بین سیستمی با چند کامپیوتر و سیستمی با چند پردازنده وجود دارد. کامپیوتراها 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 پردازشگرهای برقرار شده و در نتیجه تمام اجزاء سیستم در حل یک مسئله همکاری نمایند.

گرچه بعضی کامپیوتراهای 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 ریز پردازنده 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 خطای 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<sup>۱</sup> مستقل می توانند بطور موازی انجام شوند.

۲- یک کار مستقل می تواند به چند کار کوچکتر<sup>۲</sup> موازی تقسیم شود.

یک کار<sup>۳</sup> کلی را می توان به تعدادی کار کوچکتر (یا تکلیف) تقسیم کرد بطوری که هر پردازنده بتواند مستقلانه<sup>۴</sup> یکی از آنها را انجام دهد. کارهای کوچک سیستم را می توان به پردازنده های کاربرد ویژه ای<sup>۵</sup> 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 کامپیوتراست که در آن یک پردازنده محاسبات ریاضی ممیز - شناور سرعت های بالا را انجام می دهد و کامپیوترا دیگری بر کارهای کوچک یک داده پردازی 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 زیانه ای برنامه نویسی که برای پردازش موازی مناسبند را در اختیار می گذارند. روش کارائی دیگر عبارتست از تهیه مترجمی<sup>۶</sup> 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 و استگی داده ها وارسی می کند. اگر بخشی از برنامه به داده های تولید شده در بخشی دیگر و استگی 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،<sup>1</sup> در طبقه چند پردازنده‌های دارای حافظه اشتراکی یا چند پردازنده‌های با کوپل محکم<sup>2</sup> 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 CPU‌های 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<sup>3</sup> یا کوپل سست<sup>4</sup> 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، محتواهای داده و برخی کدهای کشف خط‌آ تشکیل شده است. بسته‌ها بر اساس سیستم ارتباطی بکار رفته، به آدرس یک پردازنده خاص ارسال می‌شوند یا بوسیله اولین پردازنده آزاد دریافت می‌گردند. سیستم‌های با کوپل سست، وقتی عمل و عکس العمل متقابل بین کارهای کوچک حداقل است کارائتر می‌باشد، در صورتی که سیستم‌های با کوپل محکم می‌توانند درجه بالاتری از اینگونه اعمال و عکس العمل‌ها را بین تکالیف تحمل کنند.

## ۱۳-۲ ساختارهای اتصالات متقابل

اجزایی که یک سیستم چند پردازنده‌ای را تشکیل می‌دهند عبارتند از CPU‌ها که به وسائل ورودی-خروجی متصلند، و یک واحد حافظه که ممکن است به چند مأمور جداگانه تقسیم شده باشد. اتصالات فی‌مابین 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 فی‌مابین 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

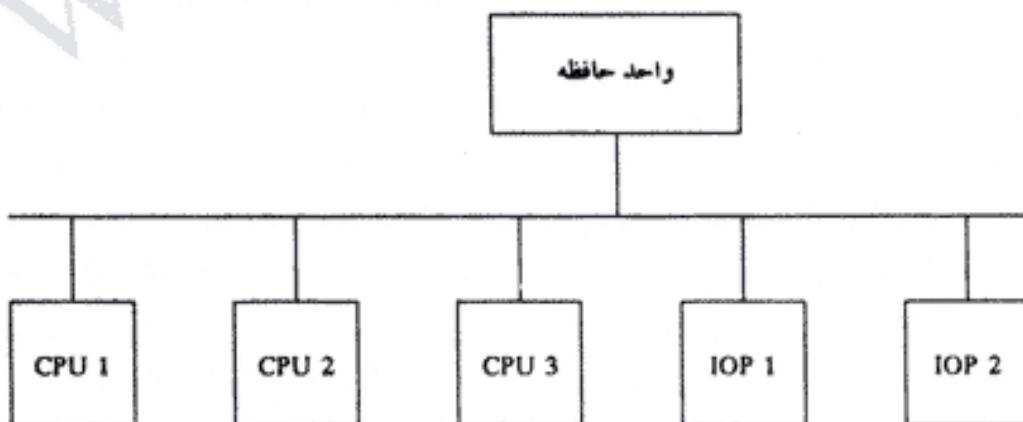
4- Loosely Coupled System

- ۱- حافظه چند پورتی
- ۲- سویچ های تقاطعی
- ۳- شبکه سوئیچینگ چند طبقه
- ۴- سیستم فوق مکعبی

### گذرگاه مشترک با تسهیم زمانی

یک چند پردازنده گذرگاه مشترک است از تعدادی پردازنده که از طریق مسیرهای مشترک به یک واحد حافظه متصل شده‌اند. یک گذرگاه مشترک با تسهیم زمانی<sup>۲</sup> برای پنج پردازنده در شکل ۱ نشان داده است. در هر لحظه از زمان تنها یک پردازنده با حافظه یا پردازنده دیگر ارتباط برقرار کند. عملیات انتقال توسط پردازنده‌ای انجام می‌شود 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 آزادبودن آزمایش کند، و پس از در اختیار بودن گذرگاه پردازنده می‌تواند واحد مقصد را برای آغاز انتقال آدرس دهی نماید. برای آگاه کردن واحد مقصد از نوع عمل، دستوری به آن صادر می‌شود. واحد گیرنده آدرس آن را در گذرگاه تشخیص داده و به سیگنال‌های کنترل که از فرستنده پس از آغاز انتقال می‌رسند پاسخ می‌دهد. سیستم بعلت اشتراک گذرگاه ممکن است دچار مشکلاتی در انتقال گردد زیرا یک گذرگاه بوسیله همه پردازنده‌ها به اشتراک گرفته شده است. این مشکلات باید با استفاده از یک کنترل کننده گذرگاه که واحدهای درخواست‌کننده را اولویت‌بندی می‌کند 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 می‌کند، کلیه پردازنده‌های دیگر یا مشغول کار داخلی می‌باشند یا باید بانتظار استفاده از گذرگاه بیکار بمانند. در نتیجه، سرعت انتقال کل در سیستم توسط سرعت یک گذرگاه



شکل ۱۳-۱ سازمان گذرگاه اشتراکی زمان مشترک

1- Multiport

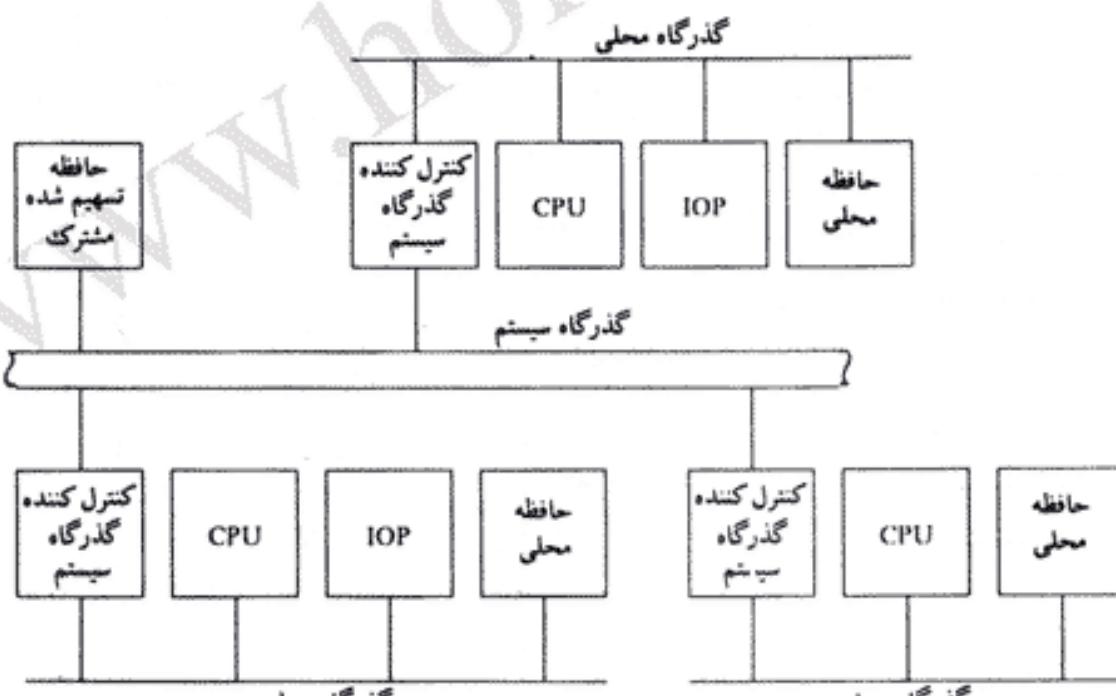
2- Time Shared Common Bus

محدود می شود. پردازنده ها در سیستم، اغلب از طریق بکارگیری دو یا چند گذرگاه برای انتقال های همزمان متعدد مشغول نگهدارنده می شوند. با این وجود این روش هزینه و پیچیدگی سیستم را افزایش می دهد. پیاده سازی مقرن به صرفه تری برای یک ساختار دو گذرگاهی در شکل ۱۳-۲ نشان داده است.

در این شکل تعداد گذرگاه های محلی وجود دارد که هر کدام به حافظه محلی خود و یک یا دو پردازنده متصل شده اند. هر گذرگاه محلی ممکن است به یک CPU، 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 خواهد 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 ۱۲-۶). 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 مازول حافظه<sup>۱</sup> و هر CPU استفاده



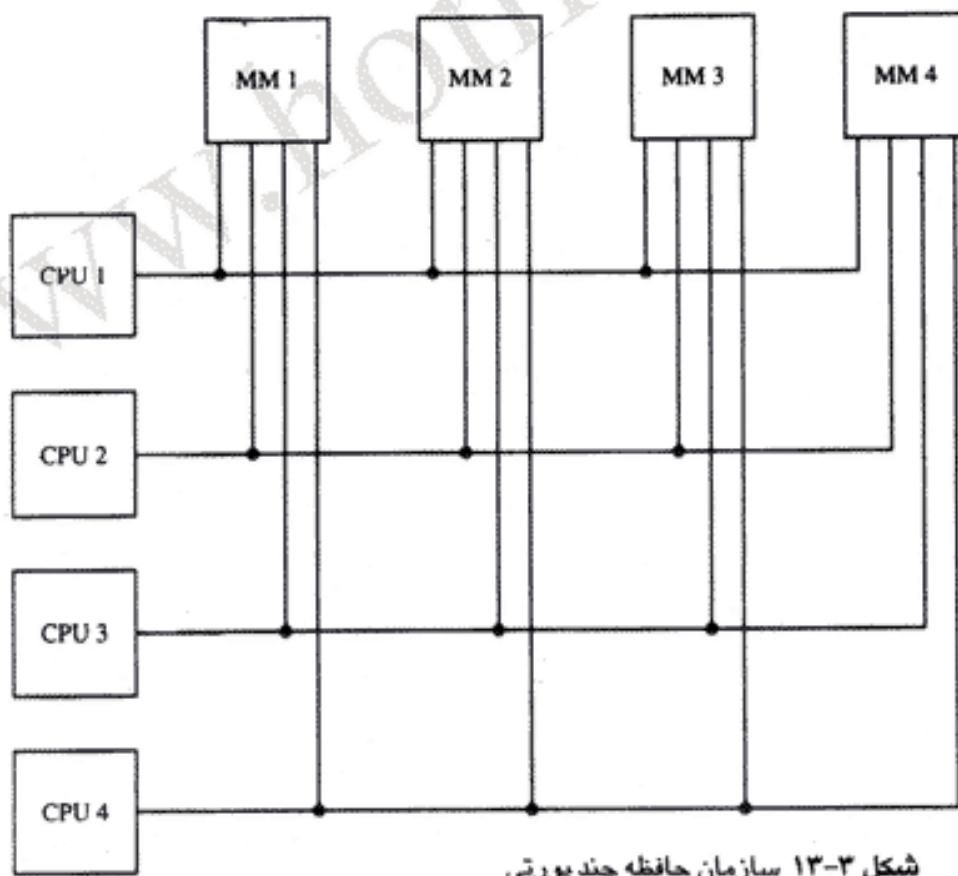
شکل ۱۳-۲ ساختار گذرگاه سیستم برای چند پردازنده ها

### 1- Memory Modules

می‌کند. این آرایش برای چهار CPU و چهار مازول حافظه (MMS) در شکل ۱۳-۳ نشان داده شده است. گذرگاه هر پردازنده به هر مازول 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 مازول حافظه چهار پورت دارد و هر پورت یکی از گذرگاه‌ها را سرویس می‌دهد. مازول 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 مازول 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 پورتی عبارتست از سرعت انتقال بالاکه بعلت 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 فی‌مایبن معمولاً برای سیستم‌هایی مناسب است که تعداد کمی پردازنده داشته باشد.

مازول‌های حافظه



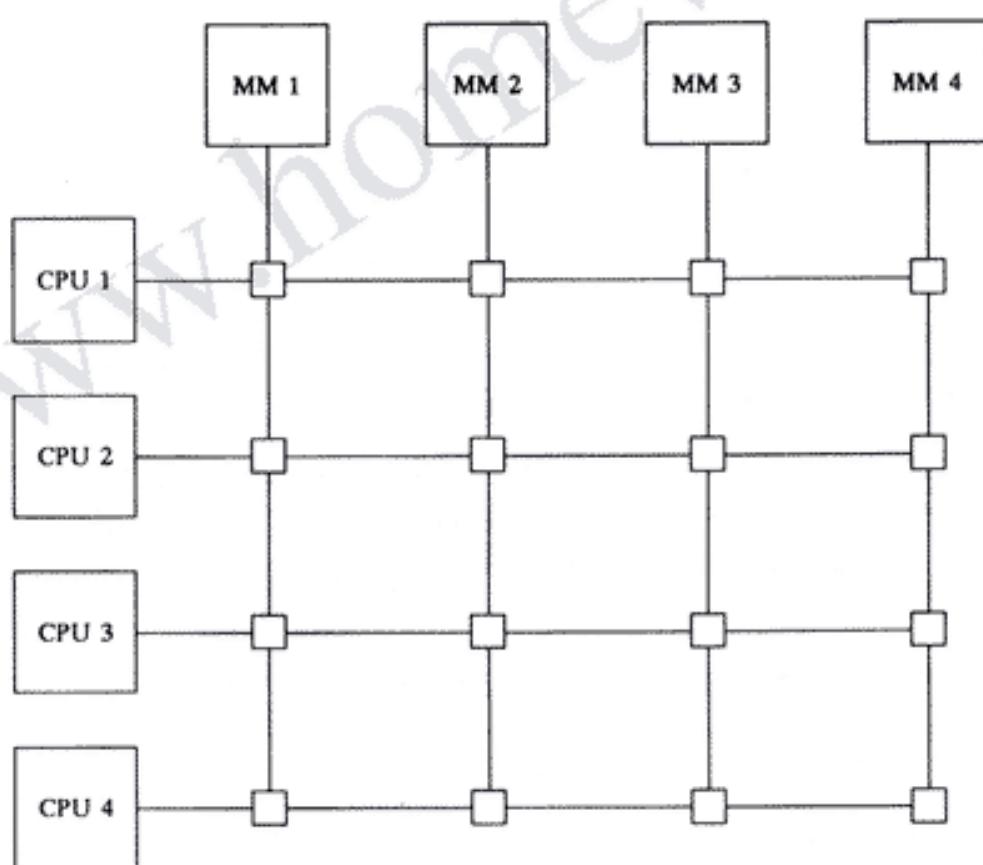
شکل ۱۳-۳ سازمان حافظه چندپورتی

### سوئیچ تقاطعی

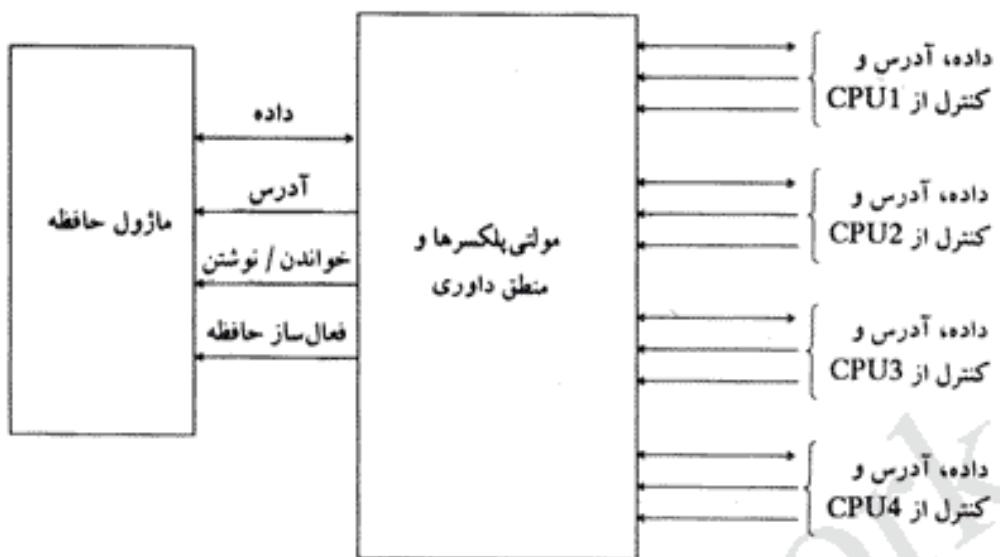
سازمان یک سوئیچ تقاطعی از تعدادی نقطه تقاطعی تشکیل می شود که در محل تقاطع گذرگاههای بین پردازنده‌ها و مسیرهای مازول حافظه قرار داده می شود. شکل ۱۳-۴ اتصالات بین چهار CPU و چهار مازول حافظه را با سوئیچ تقاطعی نشان می دهد. مربع کوچک در نقطه تقاطع یک کلید است که مسیر را از پردازنده به مازول حافظه معین می 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 مازول خاص بررسی می کند. همچنین مشکل درخواست‌های متعدد برای دستیابی به یک مازول آدرس دهنده شده را بر مبنای اولویت قبلی حل می کند.

شکل ۱۳-۵ طرح عملیاتی یک سوئیچ تقاطعی متصل به یک مازول حافظه را نشان می دهد. این مدار از مولتی پلکسرهایی تشکیل می شود که داده‌ها، آدرس، و کنترل را از یک CPU برای ارتباط با مازول حافظه انتخاب می کنند. سطوح اولیه توسط منطق داوری<sup>۱</sup> برقرار می شود تا هر وقت دو

مازول‌های حافظه



شکل ۱۳-۴ سوئیچ تقاطعی

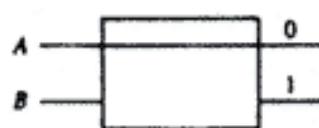


شکل ۱۳-۵ بلاک دیاگرام سوئیچ تقاطعی

یا چند CPU اقدام به مراجعته کنند یکی از آنها انتخاب شود. مولتی پلکسراها با کددودوبی که توسط یک گذرگاه اولویت در داخل منطق داوری ایجاد می‌شود کنترل می‌گردد. سازمان سوئیچ تقاطعی انتقال‌های هم‌زمان از همه مازول‌های حافظه را پشتیبانی می‌کند زیرا برای هر مازول یک مسیر جداگانه موجود است. با این وجود سخت افزار لازم برای پیاده‌سازی سوئیچ می‌تواند بسیار بزرگ و پیچیده باشد.

#### شبکه سوئیچینگ چند طبقه

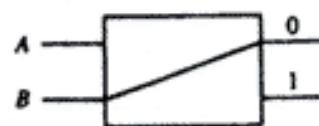
جزء اصلی یک شبکه چند طبقه، یک سوئیچ تبادلی دو ورودی - دو خروجی است. همانطور که در شکل ۱۳-۶ دیده می‌شود، سوئیچ  $2 \times 2$  دارای دو ورودی A و B، و دو خروجی 0 و 1 است. در رابطه با هر سوئیچ سیگنال‌های کنترلی (دراینجا نشان داده نشده‌اند) که این ارتباط بین پایانه‌های ورودی - و خروجی را برقرار کنند وجود دارند. سوئیچ قابلیت اتصال A را به هر یک از خروجی‌ها دارد. پایانه B در سوئیچ نقش مشابهی دارد. همچنین سوئیچ داری بین درخواست‌های مشکل‌ساز را دارد. اگر ورودی‌های A و B، هر دو یک پایانه خروجی را تقاضا کنند، تنها یکی از آنها متصل شده و دیگری بلوکه می‌شود. با استفاده از سوئیچ  $2 \times 2$  بعنوان بلاک ساختمانی، می‌توان یک شبکه چند طبقه برای کنترل ارتباط بین چند منبع و مقصد ساخت. برای ملاحظه نحوه انجام این کار، درخت دودوبی شکل ۱۳-۷ را در نظر بگیرید. دو پردازنده P<sub>1</sub> و P<sub>2</sub> از طریق سوئیچ‌ها به هشت مازول حافظه که بصورت 000 تا 111 شماره‌گذاری شده‌اند متصل شده‌اند. هر مسیر یک منبع به یک مقصد از بیت‌های شماره مقصد تعیین می‌شود. اولین بیت شماره مقصد خروجی سوئیچ را در اولین طبقه تعیین می‌کند. بیت دوم خروجی



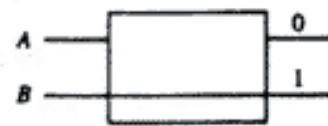
به ۰ وصل شده است



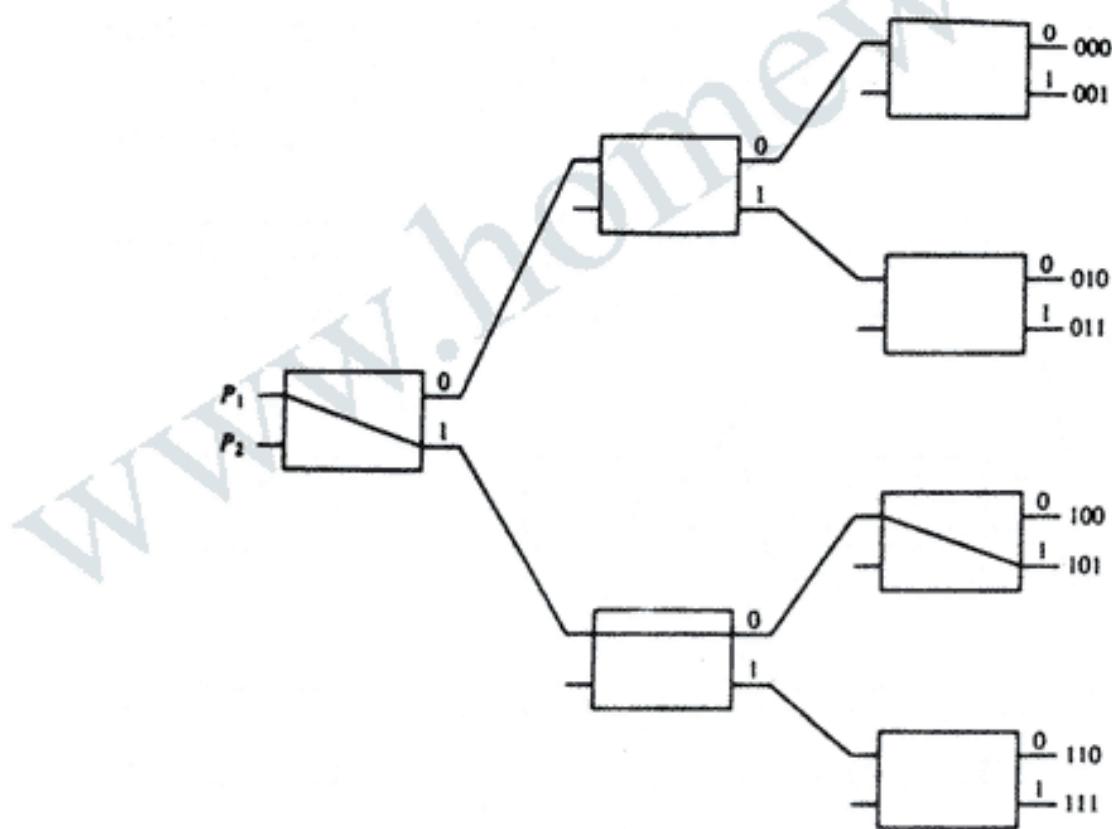
به ۱ وصل شده است



به ۰ وصل شده است



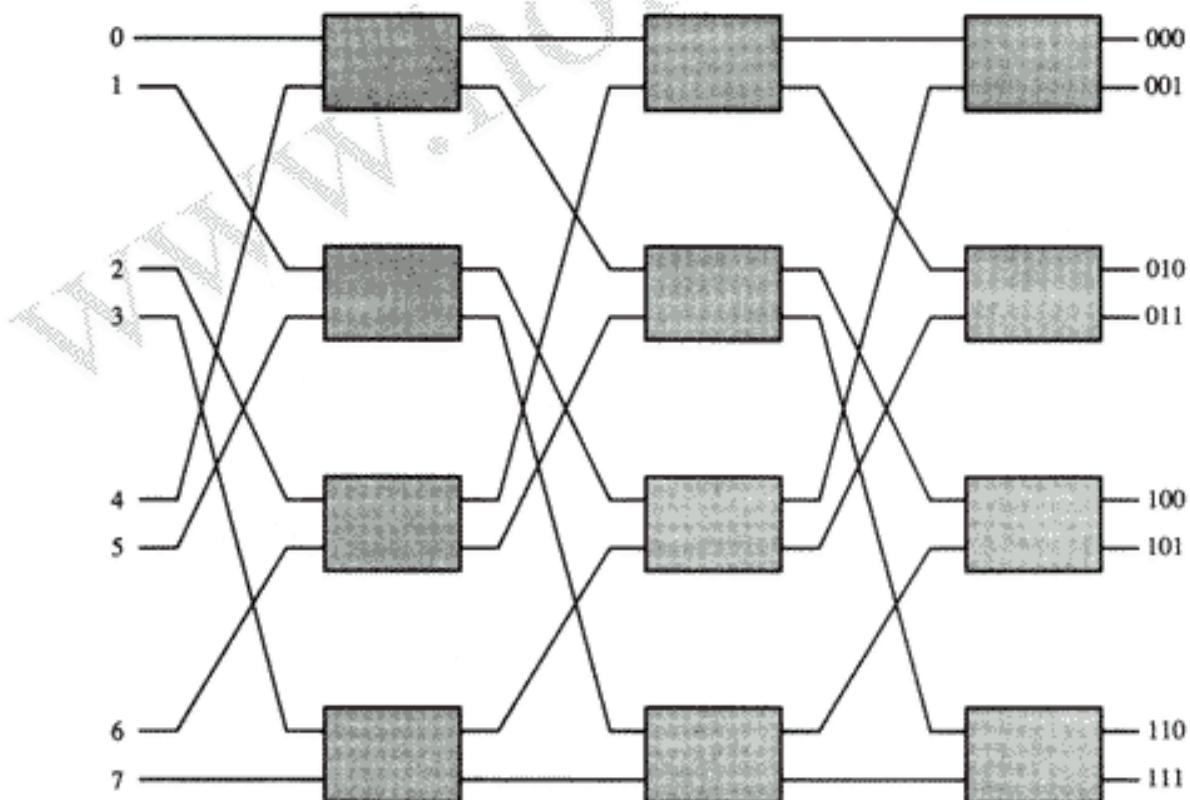
به ۱ وصل شده است

شکل ۱۳-۶ عملکرد یک سوئیچ متقابل  $2 \times 2$ شکل ۱۳-۷ درخت دودویی با سوئیچ های  $2 \times 2$ 

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 هر یک از هشت حافظه است. با این وجود الگوهای تقاضای معینی برای ارتباط نمی‌تواند بطور همزمان ایجاد شود. مثلاً، اگر  $P_1$  به یکی از مقصدات 000 تا 011 وصل شود،  $P_2$  فقط به یکی از مقصدات 100 تا 111 وصل می‌گردد.

توبولوژی‌های متعددی برای شبکه‌های سوئیچینگ چند طبقه جهت کنترل ارتباط پردازنده - حافظ در سیستم چند پردازنده با کوپل محکم یا کنترل ارتباط بین عناصر پردازنده در سیستم با کوپل سنت پیشنهاد شده است. نمونه‌ای از این توبولوژی‌ها شبکه سوئیچینگ شکل ۱۳-۸ می‌باشد. در این آرایش، دقیقاً یک مسیر از هر منبع به هر مقصد بخصوص وجود دارد. با این وجود برخی الگوهای مورد نظر را نمی‌توان بطور همزمان برقرار کرد. مثلاً هر دو منبع نمی‌توانند به مقصدات 000 و 001 وصل شوند. یک درخواست بخصوص در شبکه سوئیچینگ توسط مبداء آغاز می‌شود، و ضمن آن یک الگوی ۳ بیتی نماینده شماره مقصد را ارسال می‌کند. با حرکت الگوی دودویی در طول شبکه، هر یک از سطوح بیت متفاوتی را بررسی می‌کنند تا نحوه تنظیم سوئیچ  $2 \times 2$  تعیین شود. سطح 1 پارازش ترین بیت را بررسی می‌کند، سطح 2 بیت وسط، و سطح 3 کم ارزشترین بیت را چک می‌نماید. وقتی که تقاضا به هر یک از ورودی‌های سوئیچ  $2 \times 2$  می‌رسد، اگر بیت مشخص شده 0 باشد به خروجی بالایی و اگر 1 باشد به خروجی پائینی هدایت می‌شود.



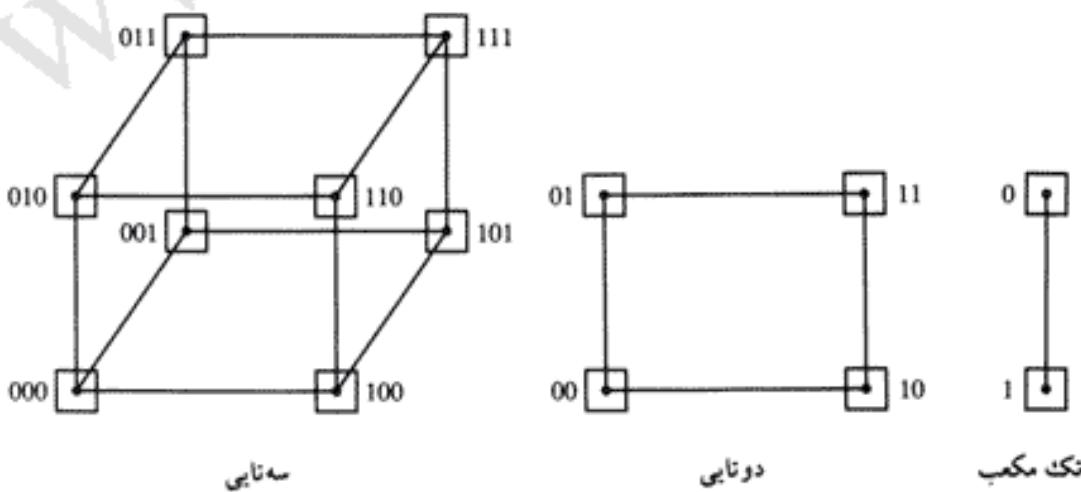
شکل ۱۳-۸ شبکه سوئیچینگ امکای  $8 \times 8$

در یک سیستم چند پردازنده با کوپل محکم، منبع یک پردازنده و مقصد مازول 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 نوشتمن باشد بکار می‌روند. در سیستم چند پردازنده با کوپل سمت، 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<sup>۱</sup>

ساختار چند پردازنده فوق مکعبی یا چند پردازنده  $n$  بعدی، یک سیستم با کوپل سمت مشکل 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 ۱۳-۹ ساختار یک فوق مکعب را برای  $n=1, 2, 3$  نشان می‌دهد. در ساختار مکعب یک تایی داریم  $n=1$  و  $n=2$  این ساختار حاوی دو پردازنده متصل بهم توسط یک مسیر است. ساختار مکعب دوناتی دارای  $n=2$  و  $n=3$  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 ۱۳-۹ ساختارهای فوق مکعبی برای  $n = 1, 2, 3$

1- Hypercube Interconnection

n تایی دارای 2<sup>n</sup> 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 دارد.

راهدهی 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 می‌تواند مستقیماً با گره 001 ارتباط برقرار کند. برای ارتباط با 011 باید دست کم از دو ضلع عبور کند (از 000 به 001، یا از 000 به 010، یا 011). برای ارتباط گره 000 با گره 111 باید دست کم از سه ضلع عبور کرد. یکی از روش‌های مسیریابی محاسبه OR انحصاری آدرس گره مبداء با آدرس گره مقصد است. در عدد دودویی حاصل، بیت‌های منتظر با محورهایی که روی آنها دو گره با هم متفاوتند مقدار 1 خواهند داشت. سپس پیام در طول این محورها ارسال می‌شود. مثلاً در ساختار مکعب سه تایی، برای ارسال پیامی از 010 به 001، OR انحصاری دو آدرس محاسبه می‌شود و نتیجه 011 را می‌دهد. پیام می‌تواند در طول محور دوم به 000 و سپس در طول محور سوم به 001 ارسال شود.

مثالی از معماری فوق مکعبی، مجموعه کامپیوتروی iPSC اینتل است. این سیستم از 128 (n=7) ریز‌کامپیووتر، که از طریق کانال‌های ارتباطی بهم متصل‌اند، تشکیل شده است. هر گره از یک CPU، یک پردازنده ممیز شناور، حافظه محلی، و واحدهای واسطه ارتباطات سری تشکیل می‌شود. هر یک از گره‌ها بر اساس برنامه‌های مقیم بطور مستقل بر روی داده‌های ذخیره شده در حافظه محلی عمل می‌کند. داده‌ها و برنامه‌های هر گره از طریق یک سیستم پیام رسانی از گره‌های دیگر یا از مدیر مکعب تأمین می‌شود. برنامه‌های کاربردی روی مدیر مکعب تهیه، ترجمه و سپس در گره‌های مختلف بار می‌شوند. محاسبات در سیستم توزیع و بطور موازی اجرا می‌گردد.

### ۱۳-۳ داوری بین پردازنده‌ها<sup>۱</sup>

سیستم‌های کامپیوتروی برای سهولت انتقال اطلاعات بین اجزاء مختلف دارای تعدادی گذرگاه در سطوح مختلف هستند. CPU حاوی گذرگاه‌های داخلی برای انتقال اطلاعات بین ثبات‌های پردازنده و ALU است. گذرگاه حافظه از خط انتقال داده، آدرس و اطلاعات خواندن و نوشتمن تشکیل شده است. یک گذرگاه I/O برای انتقال اطلاعات به وسائل ورودی و خروجی و بالعکس استفاده می‌شود. گذرگاهی که تعداد قابل توجهی از اجزاء مانند CPU‌ها و IOP را در یک سیستم چند پردازنده‌ای بهم وصل می‌کنند گذرگاه سیستم<sup>۲</sup> نامیده می‌شوند. مدارهای فیزیکی یک گذرگاه سیستم در تعدادی صفحات مدار چاپی<sup>۳</sup> مشابه قرار داده می‌شود. هر یک از این صفحات سیستم به یک مازول 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<sup>۱</sup> 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<sup>۲</sup> در صفحه نگهدارنده ای<sup>۳</sup> 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 بلا فاصله پاسخ داده شود. با این وجود، در صورت استفاده گذرگاه سیستم بوسیله پردازنده دیگر، پردازنده متقارضی باید صبر کند. بعلاوه، 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 ۱۳-۲، 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. بعلاوه خطوط توزیع نیروی الکتریکی نیز برای تأمین توان موردنیاز 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 کلأ 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 عددتاً 16 یا 32 می باشد. خطوط آدرس برای شناسایی آدرس یک گروه یا هر منبع یا مقصد دیگر مانند پورت های ورودی یا خروجی می باشد. تعداد خطوط آدرس حداقل ممکن ظرفیت حافظه را در سیستم معین می کند. مثلاً یک آدرس 24 خطی می تواند تا به<sup>۲۴</sup> (16 مگا) کلمه حافظه دستیابی نماید. خطوط داده و آدرس به بافرهای سه حالته متنه می شود (شکل ۴-۵). بافرهای آدرس یک جهته بوده و از پردازنده بسمت حافظه می باشند. خطوط داده دو جهته بوده (شکل ۱۲-۳) و انتقال داده را در هر دو جهت ممکن می سازند.

انتقال داده ها در گذرگاه سیستم می تواند همگام و یا غیر همگام 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 ساعتها جداولهای با فرکانس تقریباً یکسان در هر واحد داشته باشیم. سیگنال های همگام سازی بطور متناوب ارسال می شوند تا همه ساعتها سیستم همگام با یکدیگر باقی بمانند. در یک سیستم غیر همگام هر قلم داده ای که انتقال می یابد با سیگنال کنترل دست دهی (شکل ۱۱-۹) 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

زمینه‌بندی، 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 ۱۳-۱، 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 آدرس دهی 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 ۱۱-۲۱

جدول ۱۳-۱ سیگنال‌های چند گذرگاه ۷۹۶ استاندارد IEEE

نام سیگنال
داده آدرس
خطوط داده (۱۶ خط)
خطوط آدرس (۲۴ خط)
انتقال داده
خواندن حافظه
نوشتن حافظه
I/O خواندن
I/O نوشت
تصدیق انتقال
کنترل وقفه
درخواست وقفه (۸ خط)
تصدیق وقفه
سیگنال‌های کنترل متفرقه
ساعت اصلی
مقداردهی اولیه بیستم
فعال ساز بایت بالا
بازدارنده حافظه (۲ خط)
قفل گذرگاه
داوری گذرگاه
درخواست گذرگاه
درخواست گذرگاه مشترک
اسغال
ساعت گذرگاه
ورودی اولویت گذرگاه
خروجی اولویت گذرگاه
منبع تغذیه و زمین (۲۰ خط)
DATA0-DATA15
ADRS0-ADRS23
MRDC
MWTC
IORC
IOWC
TACK
INT0-INT7
INTA
CCLK
INIT
BHEN
INH1-INH2
LOCK
BREQ
CBRQ
BUSY
BCLK
BPRN
BPRO

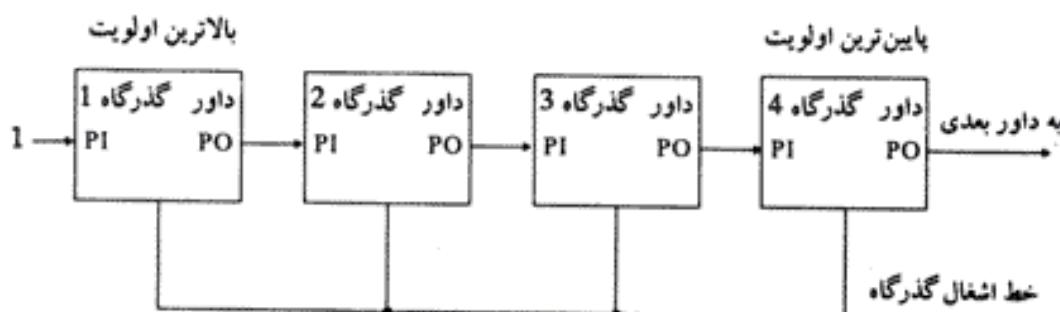
شرح داده شد اعمال می شوند. سیگنال های کنترل متفرقه قابلیت زمانبندی و مقداردهی اولیه را در اختیار می گذارند. خصوصاً سیگنال قفل کردن گذرگاه برای کاربردهای مربوط به چند پردازنده ها ضروری است. این سیگنال که توسط پردازنده فعال می شود برای جلوگیری از در اختیار قرار گرفتن گذرگاه توسط پردازنده های دیگر در حال اجرای دستورالعمل های تست و نشاندن به کار می رود. این دستورالعمل ها برای همگام سازی صحیح پردازنده لازم‌اند (بخش ۴-۱۳).

شش سیگنال داوری گذرگاه برای داوری بین پردازنده بکار می روند. این سیگنال ها را بعداً پس از بخش درباره رویه های داوری سری و موازی شرح می دهیم.

### رویه داوری سری

رویه های داوری 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 واحد های درخواست کننده کنترل گذرگاه سیستم می توان برقرار کرد. تکنیک اولویت دهنی سری از یک اتصال زنجیره دوار<sup>۱</sup> 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 اتصال 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 داراست. 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 ۱ نگهداشت 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 ۱ باشد و پردازنده مربوطه اش تقاضای کنترل گذرگاه را نکند در اینصورت خروجی PO آن برابر ۱ است. به این



شکل ۱۰-۱۳ داوری سری (زنجبیرهای)

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 بباید 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<sup>۱</sup> که در شکل ۱۳-۱۰ 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<sup>۲</sup> 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 بباید مفهوم این است که پردازنده دیگری درحال 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

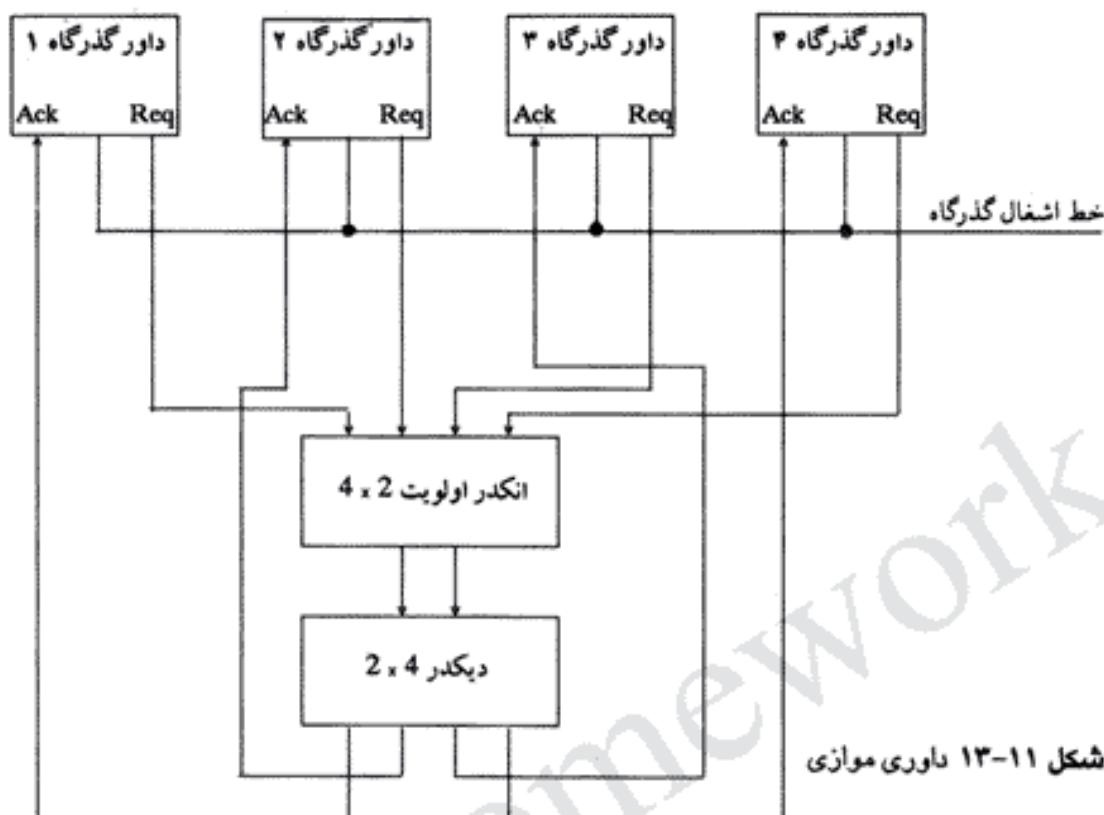
در تکنیک داوری موازی گذرگاه، از یک انکدر اولویت خارجی و یک دیکدر که طبق شکل ۱۳-۱۱ نشان داده شده است استفاده می شود. در روش موازی، هر داور گذرگاه یک خط خروجی تقاضای گذرگاه و یک خط ورودی تصدیق گذرگاه دارد. هر داور وقتی که پردازنده دستیابی به گذرگاه سیستم را درخواست کند، خط تقاضا را فعال می نماید. پردازنده پشرطی کنترل گذرگاه را بدست می گیرد که خط ورودی تصدیق فعال شده باشد، خط اشغال گذرگاه همچون روش زنجیره ای، کنترل بترتیب انتقال را انجام می دهد.

شکل ۱۳-۱۱ خطوط تقاضا را که از چهار داور به یک انکدر اولویت می روند نشان می دهد. خروجی انکدر یک کد دو بیتی را که نمایش دهنده واحدی است که بالاترین اولویت را در میان متقاضیان گذرگاه دارد تولید می کند. جدول درستی انکدر اولویت را می توان در جدول ۱۱-۲ (بخش ۱۱-۵) یافت. کد دو بیتی از خروجی دیکدر،  $4 \times 2$  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 ۱۳-۱ را توضیح می دهیم.

1- Bus Busy Line

2- Wired OR



شکل ۱۳-۱۱ داوری موازی

ورویدی الوبت گذرگاه<sup>۱</sup> BPRN و خروجی اولویت گذرگاه<sup>۲</sup> BPRO برای اتصال زنجیره‌ای مدارهای داوری 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 پائین‌تر در حال درخواست استفاده از گذرگاه سیستم هستند یا خیر. سیگنال‌های مورد استفاده برای پیاده‌سازی رویه داوری موازی، سیگنال درخواست گذرگاه BREQ و ورودی اولویت BPRN متناظر با سیگنال‌های درخواست و تصدیق شکل ۱۳-۱۱ هستند. ساعت گذرگاه BCLK برای همگام کردن همه همزمانی همه انتقال‌های گذرگاه بکار می‌رودند.

### الگوریتم‌های داوری دینامیک

دو روش داوری که در فوق توصیف شد از الگوریتم اولویت استاتیک استفاده می‌نماید زیرا اولویت هر وسیله توسط ارتباطش با گذرگاه ثبیت می‌گردد. در مقابل، یک الگوریتم داوری دینامیک به سیستم

1- Bus Priority - in

2- Bus Priority - out

توانایی تغییر اولویت وسائل را در حین عمل فراهم می‌کند. اکنون درباره چند رویه داوری که از الگوریتم‌های داوری دینامیک استفاده می‌کنند بحث خواهیم کرد.

الگوریتم برش زمانی<sup>۱</sup>، برشی با طول ثابت از زمان گذرگاه را بترتیب و بصورت چرخشی<sup>۲</sup> 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 هریک مقدار زمان مساوی برای ارتباط با گذرگاه داده می‌شود.

در یک سیستم گذرگاه که از همه پرسی<sup>۳</sup> استفاده می‌کند، بجای سیگنال واگذاری گذرگاه مجموعه‌ای از خطوط را بعنوان خطوط همه پرسی داریم که به همه واحدها متصلند. این خطوط بواسیله 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 پرسی با انتخاب یک پردازنده دیگر ادامه می‌یابد. دنباله همه پرسی معمولاً قابل برنامه ریزی است و لذا انتخاب اولویت را می‌توان تحت کنترل برنامه تغییر داد.

الگوریتم قدیمی ترین 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<sup>۴</sup>، تقاضاها بترتیبی که دریافت می‌شوند سرویس دهی می‌گردند. برای پیاده کردن این الگوریتم، کنترل کننده گذرگاه، صفحی را براساس زمان دریافت تقاضاهاي 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<sup>۵</sup> نوع تعمیم یافته دینامیکی از الگوریتم زنجیره‌ای است. در این روش کنترل کننده مرکزی گذرگاه موجود نیست، و خط اولویت از خروجی اولویت<sup>۶</sup> آخرین وسیله به ورودی اولویت<sup>۷</sup> اولین وسیله بازگشته و یک حلقه بسته را تشکیل می‌دهد. این نحوه اتصال مشابه به شکل ۱۳-۱۰ است جز اینکه خروجی PO داور ۴ به ورودی PI داور ۱ 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 Slit

2- Round - Robin Fashhion

3- Polling

4- First Come - First Serve

5- Rotating Daisy - Chain

6- Priority - Out

7- Priority - in

## ۱۳-۴ ارتباط و همگامی بین پردازنده ها

در یک سیستم چند پردازنده ای باید پردازنده های متفاوتی با امکان ارتباط با یکدیگر فراهم شوند. مسیر ارتباط می تواند از طریق کانال های مشترک ورودی - خروجی ایجاد شود. در یک سیستم چند پردازنده با حافظه اشتراکی، متداول ترین رویه، کنار گذاشتن بخشی از حافظه است که برای همه پردازنده ها قابل دسترسی است. هدف اصلی از بکارگیری این حافظه مشترک اینکه نقش یک مرکز پیام مشابه با یک صندوق پستی است که هر پردازنده می تواند پیامی برای سایر پردازنده ها باقی گذارد و پیام های موجود در آن را بردارد.

پردازنده فرستنده یک درخواست، پیام، یا یک رویه را ساخته و آن را در صندوق پستی حافظه قرار می دهد. 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 می شود. پردازنده گیرنده می تواند بصورت دوره ای صندوق پستی را برای یافتن پیام های معتبر وارسی کند. زمان پاسخ این رویه می تواند وقت گیر باشد، چون یک پردازنده فقط با همه پرسی پیام ها می تواند تقاضایی را شناسایی کند. رویه کارآکتر این است که پردازنده فرستنده با کمک سینکال وقفه پردازنده گیرنده را آگاه کند. این عمل را می توان توسط یک وقفه بین پردازنده ای که وسیله نرم افزار و با استفاده از دستورالعملی در برنامه پردازنده ها تولید می شود، انجام داد و وقتی اجرا شود وضعیتی مشابه با وقفه خارجی در پردازنده دوم بوجود می آورد. باین ترتیب پردازنده متوقف شده آگاه می شود 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 CPU ها قرار گیرد. با این ترتیب امکان استفاده مشترک از برنامه های سیستم ذخیره شده بر روی دیسک بوجود می آید. مسیر ارتباطی بین دو CPU ممکن است از طریق رابطی بین دو IOP متناظر با دو CPU ایجاد شده باشد. این نوع ارتباط اجازه می دهد تا هر CPU دیگری را عنوان یک وسیله I/O در نظر گرفته و لذا پیام ها 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 پردازنده ها وجود داشته باشد. این وظیفه به سیستم عامل محول شده است. در طراحی سیستم عامل برای چند پردازنده ها، از سه سازمان استفاده شده است. آرایش حاکم و تابع<sup>۱</sup>، سیستم عامل جداگانه و سیستم عامل توزیع شده.

در شیوه حاکم و تابع، همیشه یک پردازنده، که حاکم نامیده می شود، عملیات سیستم عامل را اجرا می کند. پردازنده های دیگر بمانند یک تابع عمل کرده و عملیات سیستم عامل را اجرا نمی کنند. اگر پردازنده تابع به سرویس سیستم عامل نیاز داشته باشد باید آن را با وقفه دادن به حاکم تقاضا کند و تا

و قله 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 سیستم‌های باکوپل 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، روالهای سیستم عامل بین پردازنده های موجود توزیع شده‌اند. با این وجود، هر تابع خاص از سیستم عامل در هر زمان فقط به یک پردازنده اختصاص داده می شود. این نوع سازمان دهن، سیستم عامل شناور<sup>۱</sup> 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 باکوپل 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. کارآایی 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 اطلاق می شود. همگامی برای ایجاد توالی صحیح در پردازش ها و نیز تضمین جلوگیری از مراجعة 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

می شوند ایجاد می کنند. یکی از متداولترین روش ها استفاده از سمافور دودویی<sup>۱</sup> 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 متقابل<sup>۲</sup> 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 باید 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 بتابه 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... برت یک عمل منفرد غیر قابل تقسیم انجام شود. در غیر اینصورت، دو یا چند پردازنده ممکن است بطور همزمان سمافور را تست کرده و هر یک جداگانه آن را 1 کنند و اجازه یابند تا باافق وارد بخش بحرانی گردند. این عمل سبب می شود تا اجرای همزمان بخش بحرانی بوجود آید که می تواند منجر به مقداردهی غلط پارامترهای کنترلی و از دست رفتن اطلاعات اساسی گردد.

سمافور را می توان با استفاده از دستور العمل تست و یک مکانیزم قفل سخت افزاری مقداردهی 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. دستور العمل تست و یک کننده سمافور را 1 می کند و مکانیزم قفل را در حین اجرای دستور العمل 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 سمبول TSL بمعنی عمل "تست و 1 شدن در حین قفل شدن" باشد.

## دستورالعمل

TSL SEM

در دو سیکل حافظه بدون تداخل اجرا می شود (اولین سیکل برای خواندن و دوم برای نوشتن)

$$\begin{array}{ll} R \leftarrow M[SEM] & \text{تست سمافور} \\ M[SEM] \leftarrow 1 & \text{کردن سمافور} \end{array}$$

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 باید، 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 نماینده 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 ۱۲-۶ تشریح شد. مزیت عمدی 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 متدائل 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.

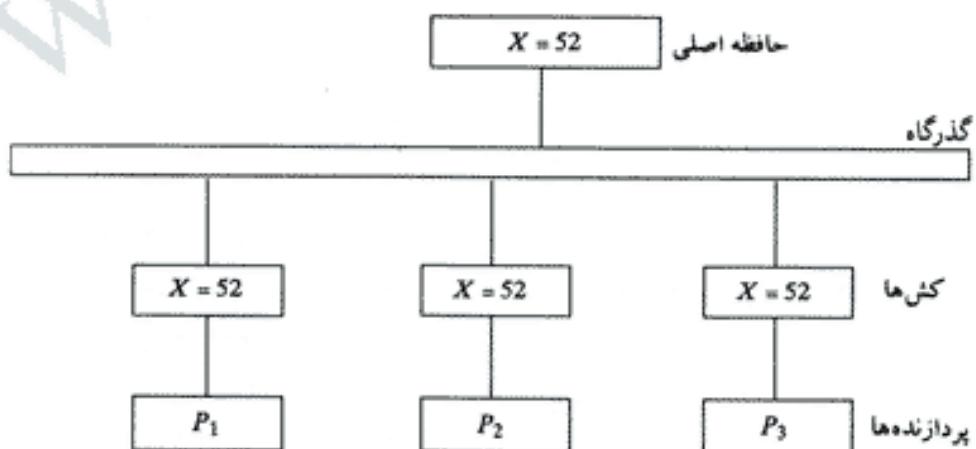
در یک سیستم چند پردازنده با حافظه مشترک، همه پردازنده ها از حافظه مشترک استفاده می کنند. بعلاوه هر پردازنده ممکن است یک حافظه محلی هم داشته باشد، که یک بخش یا تمام آن می تواند کش باشد. دلیل داشتن اجباری حافظه کش جداگانه برای هر پردازنده، کاهش میانگین زمان دستیابی در آن

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<sup>۱</sup> را بوجود می آورد. یک سیستم حافظه همبسته است اگر مقداری که یک دستور العمل باردهی بر می 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

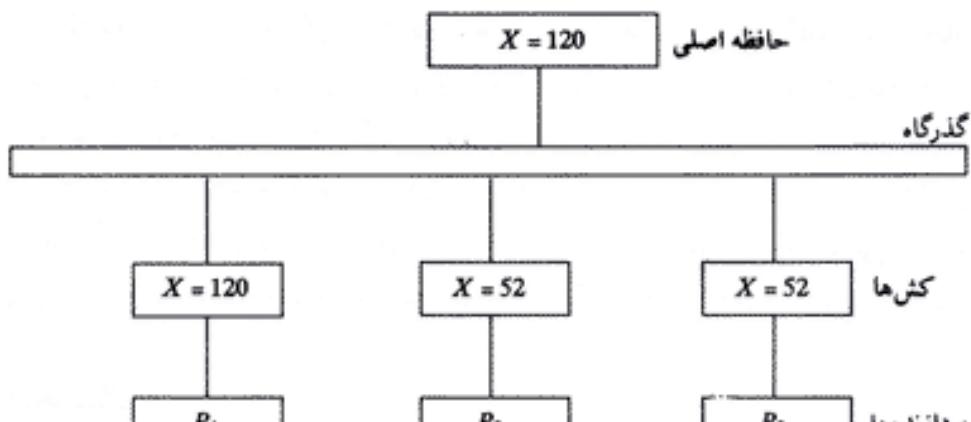
مشکلات همبستگی حافظه کش در چند پردازنده هایی که بدلیل نیاز به داده های مشترک قابل نوشتن دارای حافظه های اختصاص می باشند بوجود می آید. داده های فقط خواندنی را می توان با اطمینان و بدون مکانیزم های ایجاد کننده همبستگی حافظه کش تکثیر کرد. برای روشن شدن مسئله، آرایش سه پردازنده با حافظه هایی کش اختصاصی در شکل ۱۳-۱۲ را در نظر بگیرید. گاهی در حین کار سیستم، عنصری مثل X از حافظه اصلی در سه پردازنده P<sub>1</sub>, P<sub>2</sub>, P<sub>3</sub> 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. بار کردن بوسیله پردازنده های دیگر آخرين مقدار را برنامی 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<sub>1</sub> در

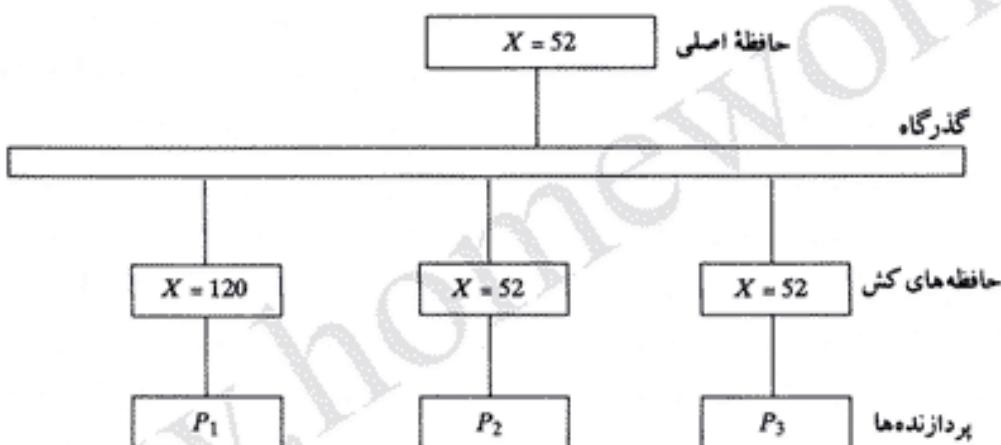


شکل ۱۳-۱۲ آرایش کش پس از بار کردن در X

1- Cache Coherence



(الف) با استفاده از سیاست کامل نویسی



(ب) با استفاده از سیاست پس نویسی

شکل ۱۳-۱۲ آرایش کش پشن از تغییره کردن در X بوسیله پردازندگه  $P_1$ 

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 ناهمانگ 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 ناهمانگی 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 مسروچ 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 روش های بکار رفته فقط اجازه ذخیره داده های غیر مشترک و فقط خواندنی را در حافظه های کش می دهد. چنان داده های قابل کش شدن<sup>1</sup> خواندن می شوند. داده های قابل نوشتن مشترک قابلیت ذخیره شدن در کش را ندارند.<sup>2</sup> 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، روشنی است که از یک جدول سراسری متتمرکز 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 های 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 روش های صرفاً سخت افزاری حل شود. دو روشهایی که قبل از رویه های مبتنی بر نرم افزار استفاده می نمایند که قابلیت نشانه گذاری برای جلوگیری از قرار گرفتن 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 گذرهای از سوی CPU ها و IOP ها

نظرارت داشته باشد. همه حافظه کش متصل به گذرگاه بطور مستمر بر شبکه برای یافتن یک عمل احتمالی نوشتمن نظرارت می‌کنند. این حافظه‌های کش باید در صورت کشف انتباق، بسته به روش مورد استفاده کپی‌های موجود خود را بهنگام و یا غیر معتبر نمایند. کنترل کننده گذرگاهی که بر این عمل نظرارت می‌کند کنترل کننده مداخله گر<sup>۱</sup> نامیده می‌شود. این کنترل کننده اساساً یک واحد سخت افزاری است که برای برقراری یک مکانیزم نظرارت بر گذرگاه در همه حافظه‌های نهان متصل به گذرگاه طراحی می‌شود. روش‌های مختلفی برای حل مسئله همبستگی حافظه نهان با استفاده از قرارداد کنترل کننده مداخله گر حافظه کش پیشنهاد شده است ساده‌ترین متد استفاده از سیاست کامل نویسی و استفاده از زیر است. تمام کنترل کننده‌های مداخله گر برای یافتن عملیات ذخیره حافظه بر گذرگاه نظرارت می‌کنند. هر گاه کلمه‌ای با نوشته شدن در یک حافظه کش بهنگام شود، مکان متناظر آن در حافظه اصلی نیز بهنگام می‌شود. کنترل کننده‌های مداخله گر محلی مربوط به کلیه حافظه‌های کش دیگر، حافظه خود را وارسی می‌کنند تا مشخص کنند کپی کلمه‌ای را که روی آن نوشته‌اند دارند پاچیر. اگر کپی در حافظه کش دیگر وجود داشته باشد، آن مکان بعنوان نامعتبر علامت گذاری می‌شود. چون همه حافظه کش همه عمل‌های نوشتن گذرگاه را زیر نظر دارند، هر گاه کلمه‌ای نوشته شود، اثر خالص این عمل بهنگام شدن آن در حافظه کش و حافظه اصلی و حذف آن از همه حافظه‌های کش دیگر خواهد بود. اگر در آینده، پردازنده‌ای به داده نامعتبر واقع در حافظه کش خود مراجعه نماید، پاسخی که دریافت می‌کند باخت در حافظه کش خواهد بود، و شکل بهنگام آن داده از حافظه اصلی آورده می‌شود. باین ترتیب از ایجاد کپی‌های متفاوت با کپی‌های جدید داده جلوگیری می‌گردد.

## مسائل

- ۱۳-۱ اختلاف بین چند پردازنده با کوپل محکم و چند پردازنده با کوپل سست را از نقطه نظر سازمان ساخت افزار و تکنیک‌های برنامه‌نویسی بحث کنید.
- ۱۳-۲ هدف از کنترل کننده گذرگاه سیستم در شکل ۱۳-۲ چیست؟ توضیح دهید که چگونه می‌توان سیستم را به صورتی طراحی کرد که ارجاع به حافظه محلی و ارجاع به حافظه مشترک را از هم تشخیص دهد.
- ۱۳-۳ در یک شبکه سوئیچ تقابلی که  $p$  پردازنده را به  $m$  مازول حافظه متصل می‌کند چند نقطه سوئیچ وجود دارد.
- ۱۳-۴ شبکه سوئیچینگ امگای  $8 \times 8$  در شکل ۱۳-۸ سه طبقه و چهار سوئیچ در هر طبقه و جمعاً ۱۲ سوئیچ دارد. برای یک شبکه سوئیچینگ  $n \times n$ ، چند طبقه و چند سوئیچ در هر طبقه لازم است.
- ۱۳-۵ فرض کنید که در شبکه سوئیچینگ امگای شکل ۸-۱۳ سیم بین سوئیچ واقع در ردیف اول و ستون دوم و سوئیچ واقع در ردیف دوم سوم قطع شود. چه مسیرهایی مسدود می‌شود؟

- ۱۳-۶ نموداری را برای یک شبکه سوئیچینگ امگای  $4 \times 4$  بسازید. تنظیم‌های سوئیچ‌ها را برای اتصال ورودی ۳ به خروجی ۱ نشان دهید.
- ۱۳-۷ برای طراحی یک شبکه ارتباطات چند طبقه از سه نوع سوئیچ استفاده شده است: سوئیچ تبادلی<sup>۱</sup> 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 سوئیچ‌های داوری و توزیع را نشان دهید.
- (ب) با استفاده از سوئیچ‌های داوری و تبادلی، یک شبکه  $8 \times 8$  با یک مسیر منحصر بفرد بین هر مبدأ و هر مقصد بسازید.
- (ج) با استفاده از سوئیچ‌های توزیع و تبادلی، یک شبکه  $8 \times 8$  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 ۷ به گره ۹ را که از حداقل تعداد گره‌های میانی استفاده می‌کنند 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 زنجیره‌ای شکل ۱۳-۱۰ نشان دهد.
- ۱۳-۱۰ مدار تحت کنترل مدار داوری موازی در شکل ۱۳-۱۱ در آبتداسکار است. سپس وسیله‌های ۲ و ۳ بطور همزمان گذرگاه را درخواست می‌کنند. مقادیر دودویی ورودی و خروجی را در انکدر و دیکدر مشخص نمایند و مشخص کنید به کدام داور گذرگاه سیگنال تصدیق داده می‌شود.
- ۱۳-۱۱ نشان دهید که چگونه مدار داوری شکل ۱۳-۱۰ را می‌توان برای فراهم نمودن یک رویه داوری زنجیره‌ای بسته اصلاح کرد. توضیح دهید که وقتی خط گذرگاه غیرفعال است چگونه اولویت تعیین می‌شود.
- ۱۳-۱۲ یک توپولوژی گذرگاه که در آن دو پردازنده از طریق یک بافر در حافظه مشترک با هم تبادل اطلاعات می‌کنند را در نظر بگیرید. اگر یکی از پردازنده‌ها بخواهد با پردازنده دیگری ارتباط برقرار کند، اطلاعات را در بافر حافظه قرار می‌دهد و پرچمی را ۱ می‌کند. پردازنده دیگر بطور متناوب پرچم‌ها را وارسی می‌کند 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 می‌توان 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 ۱ نمودن.
- ۱۳-۱۴ همبستگی حافظه کش چیست، و چرا در سیستم‌های چند پردازنده با حافظه مشترک مهم است؟ چگونه می‌توان این مسئله را با یک کنترل کننده مداخله گر حافظه نهان حل کرد؟

1- Interchange Switch