دانشگاه آزاد اسلامی

جزوه درس معماری کامپیوتر

مدرس : استاد یوسفی

معماری کامپیوتر

مدرس: یوسفی

فصل اول

قطعات دیجیتال

مدارهای مجتمع

مدارهای دیجیتال با مدارهای مجتمع ساخته می شوند. یک مدار مجتمع (یا آی سی)، یک کریستال کوچک نیمه هادی بنام تراشه است که قطعات الکترونیکی را برای گیت های دیجیتال در خود دارد. اتصالات داخل تراشه، مدار مورد نیاز را به وجود می آورد. تراشه در داخل یک محفظه پلاستیک و یا سرامیک جاسازی می شود و اتصالات آن با سیم های طلایی نازک به پایه های خارجی جوش داده می شود تا مدار مجتمع بوجود آید. تعداد پایهه ا ممکن است از ١٤ پایه در بسته های کوچک تا ١٠٠ پایه یا بیشتر در بسته های بزرگتر تغییر کند. هر مدار مجتمع یا آی سی دارای یک مشخصه عددی است که روی سطح بسته بندی آن برای شناسایی چاپ می شود. هر سازنده یک کتابچه راهنما یا کاتالوگ با شرح دقیق و تمام اطلاعات لازم درباره آی سی های ساخت خود را چاپ می کند.

با پیشرفت تکنولوژی مدارهای مجتمع، تعداد گیت هایی که می توانست در یک تراشه جای گیرد به میزان قابل ملاحظه ای افزایش یافت. تراشه هایی که دارای چند گیت داخلی بودند و آن دسته که چند صد گیت را دارا بودند در بسته هایی با ظرفیت یا مقیاس کوچک، متوسط و یا بزرگ جای داده شدند.

مدارهای مجتمع با مقیاس کوچک (SSI) دارای چند گیت مستقل در یک بسته واحد هستند. ورودی ها و خروجی های گیت ها مستقیما به پایه های بسته متصل اند. تعداد گیت ها معمولا کمتر از ١٠ و محدود به تعداد پایه ها درIC می باشد.

قطعات مجتمع با مقیاس متوسط (MSI) دارای تقریبا ١٠ الی ٢٠٠ گیت در هر بسته می باشند. این وسیله ها معمولا توابع دیجیتال ساده همچون دیکدرها، جمع کننده ها و ثبات ها را اجرا می نمایند.

مدارها یا وسایل مجتمع با مقیاس بزرگ (LSI) بین ۲۰۰ تا چند هزار گیت در هر بسته دارند. این بسته ها سیستم های دیجیتالی همچون پردازنده ها، تراشه های حافظه و ماژول های قابل برنامه ریزی را شامل می شوند. قطعات مجتمع با مقیاس بسیار بزرگ (VLSI) حاوی هزاران گیت در یک بسته اند. مثال هایی از این گروه عبارتند از: آرایه های بزرگ حافظه و تراشه های پیچیده ریز کامپیوترهاVLSI ها بدلیل کوچکی و ارزانی انقلابی در تکنولوژی ساخت سیستم های کامپیوتری بوجود آورده و به طراحان امکان ساخت و ایجاد ساختارهایی را دادند که قبلا اقتصادی نبودند.

مدارهای مجتمع دیجیتال نه تنها براساس عملکرد منطقی شان طبقه بندی می شوند بلکه از نظر تنکولوژی خاص مدارهایی که به آن تعلق دارند نیز دسته بندی می گردند. تکنولوژی بکار رفته در مدار را خانواده منطقی دیجیتال می خوانند. هر خانواده منطق، مدار الکترونیکی پایه خاصی را داراست که مدارها و توابع دیجیتال پیچیده تر براساس آن تهیه می شوند. مدار پایه در هر تکنولوژی یک گیتNAND، NOR و یا معکوس کننده است. در نام گذاری تکنولوژی، از قطعات الکترونیک بکار رفته در ساخت مدار پایه معمولا استفاده می شود. بسیاری از خانواده های مختلف منطقی بصورت دارهای مجتمع در سطح تجاری عرضه شده اند. متداول ترین خانواده ها در زیر معرفی شده اند.

TTL – منطق ترانزیستور – ترانزیستور

ECL – منطق کوپل امیتر

MOS – منطق فلز – اکسید – نیمه هادی

CMOS – منطق فلز – اکسید – نیمه هادی مکمل

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

خانواده منطقی ترانزیستور- ترانزیستور گونه تکامل یافته تکنولوژی قدیمی تری است که در آن از دیود و تزانزیستور برای ساخت گیت پایه NAND استفاده می شدهاست. این تکنولوژی منطق دیود- ترانزیستور (DTL) خوانده می شده است. بعدها برای بهبود عملکرد مدار بجای دیود از ترانزیستور استفاده شد و نام خانواده جدید، منطق ترانزیستور – ترانزیستور گذاشته شد. 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، TTL توان پائین (یا کم مصرف)، TTL شوتکی، TTL شوتکی توان پایین و TTL شوتکی پیشرفته. منبع تغذیی مدارهای TTL، ۵ ولت است و دو سطح منطقی تقریبا ۰ و ۳/۵ ولت می باشند.

خانواده کوپل امیتر سریع ترین مدارهای دیجیتال را بفرم مجتمع در اختیار می گذارد. ECL در مدارهایی مانند سوپر کامپیوترها و پردازنده های سیگنال که در انها سرعت بالا ضرورت دارد، بکار می رود. ترانزیستورها در گیت های ECL در حالت غیراشباع کار می کنند و رسیدن به تاخیرهای انتشاری در حد ۱ تا ۲ نانو ثانیه در آنها میسر است.

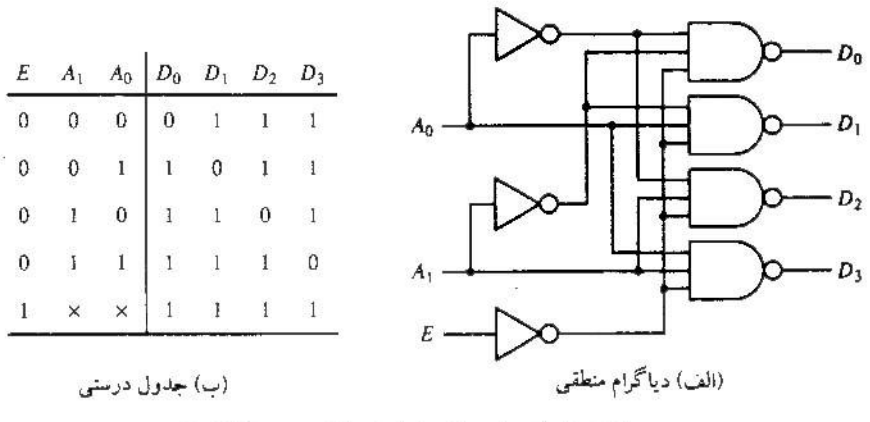
منطق فلز – اکسید – نیمه هادی (MOS) یک ترانزیستور تک قطبی است که به جریان یک نوع حامل الکتریکی وابسته است. این حامل ها ممکن است الکترون (در نوع کانال n) یا حفره (در نوع کانال P) باشند. این، برخلاف ترانزیستور دوقطبی بکار رفته در گیت های TTL و ECL است، که در حین عملکرد هر دو نوع حامل در آن وجود دارند. یک MOS کانال p را PMOS و یک MOS کانال n را

NMOS می نامند. معمولا در مدارهایی که فقط یک نوع ترانزیستور PMOS و NMOS که بشکل مکمل در تام مدارها بسته شده اند بکار رفته است. بزرگترین مزیتCMOSنسبت به دوقطبی تراکم بالای مدارها در بسته بندی، ساده بودن تکنیک ساخت و عملکرد مقرون به صرف آن بدلیل مصرف توان کم است.

دیکدرها

دیکدر مداری است ترکیبی که اطلاعات دودویی را از طریق n خط ورودی دریافت نموده و آنها را به حداکثر 2^n خط خروجی مستقل منحصر بفرد تبدیل می کند. اگر اطلاعات دیکد شدهn بیتی ترکیبات بلااستفاده یا بی اهمیت داشته باشد در اینصورت دیکدر دارای خروجی های کمتر از2^n خواهد بود.

دیکدرهایی که در این بخش مورد بحث قرار گرفته اند دیکدرهای n به m خطی نام دارند که در آن $m \leq 2^n$ است. هدف از بکارگیری آنها تولید 2^n ترکیب دودویی از n متغیر ورودی است. دیکدر دارای n ورودی و m خروجی بوده و دیکدر $n \times m$ نیز نامیده می شود.



شکل ۱-۱: دیکدر ۲ به ۴ خطی با گیت هایNAND

مولتی پلکسرها

مولتی پلکسر یک مدار ترکیبی است که اطلاعات دودویی را از یکی از2^n ورودی دریافت و آن را به یک مسیر خروجی هدایت مینماید. مجموعهای از خطوط ورودی انتخاب خط داده خاصی را برای خروجی انتخاب میکنند. یک مولتی پلکسر 2^n به ۱ 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.

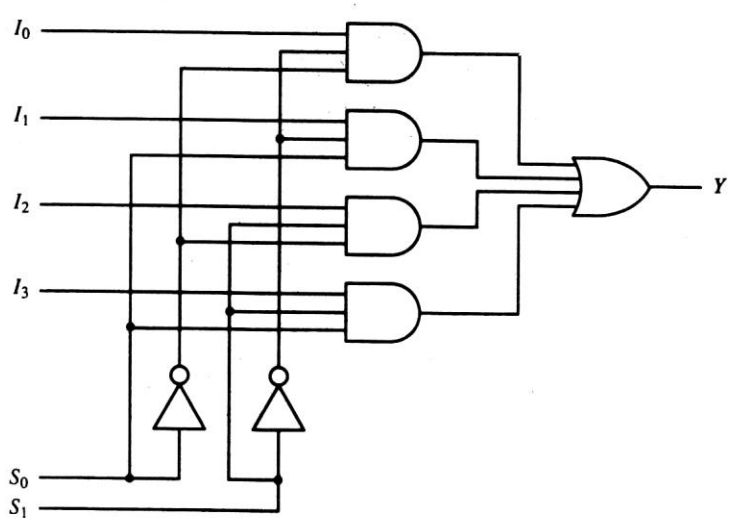
یک مولتی پلکسر 4 به 1 در شکل ۲-۱ نشان داده شده است. هریک از چهار ورودی دادهI_0 تا I_3 به ورودی یک گیت AND اعمال شدهاست. دو خط انتخاب S_1 و S_0 برای انتخاب یک گیت AND بخصوص دیکد شدهاند. خروجی گیت های AND به ورودی های یک گیت OR اعمال شده تا تنها خروجی مدار را ایجاد نماید.

جدول تابع برای مولتی پلکسر در جدول ۱-۱ نشان داده شده است. این جدول رابطه بین چهار ورودی داده و تنها خروجی مدار را بصورت تابعی از ورودهای انتخاب S_1 و S_0 نشان می دهد. هرگاه ورودی های انتخاب برابر 00 باشند، خروجی Y برابر I_0 است. اگر ورودی های انتخاب 01 شوند، ورودی I_1 به خروجی Y متصل است، و برای دو ورودی دیگر نیز استدلال مشابهی وجود دارد. مولتی پلکسر را انتخاب کننده داده هم مینامند، چون این مدار یکی از چند ورودی داده را بر میگزیند و اطلاعات دودویی را به خروجی هدایت مینماید.

جدول۱-۱: جدول تابع برای مولتی پلکسر۴ به ۱ خطی

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

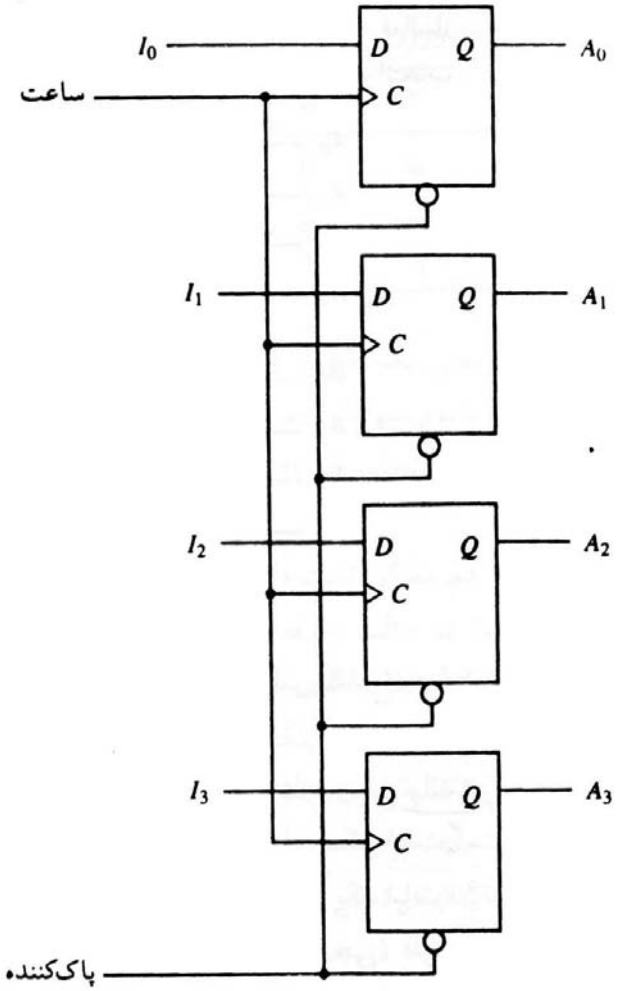
همانند دیکدرها، مولتی پلکسرها هم ممکن است دارای ورودی تواناساز باشند تا عملکرد آن را کنترل نمایند. هروقت ورودی تواناساز در حالت غیرفعال باشد، خروجی ها از کار میافتند، و اگر در حالت فعال قرار گیرد، مدار به صورت یک مولتی پلکسر بطور عادی عمل میکند. ورودی تواناساز برای اتصال دو یا چند مولتی پلکسر و تشکیل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



شکل۲-۱: مولتی پلکسر ۴ به ۱

ثبات ها

ثبات مجموعهای از فیلپ فلاپ هاست و هر فلیپ فلاپ قادر است یک بیت از اطلاعات را در خود 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 فلاپ ها، ممکن است گیتهای ترکیبی، که کارهای پردازش داده معینی را انجام میدهند، را هم در خود داشته باشد. یک ثبات در تعریف جامعتر خود متشکل از گروهی از فلیپ فلاپها و گیتهاست که بر تغییر حالت (خروجی) فلیپ فلاپها تاثیر دارند. فلیپ فلاپها اطلاعات دودویی را ذخیره میکنند و گیت ها زمان و چگونگی انتقال اطلاعات به ثبا ت را کنترل مینمایند.



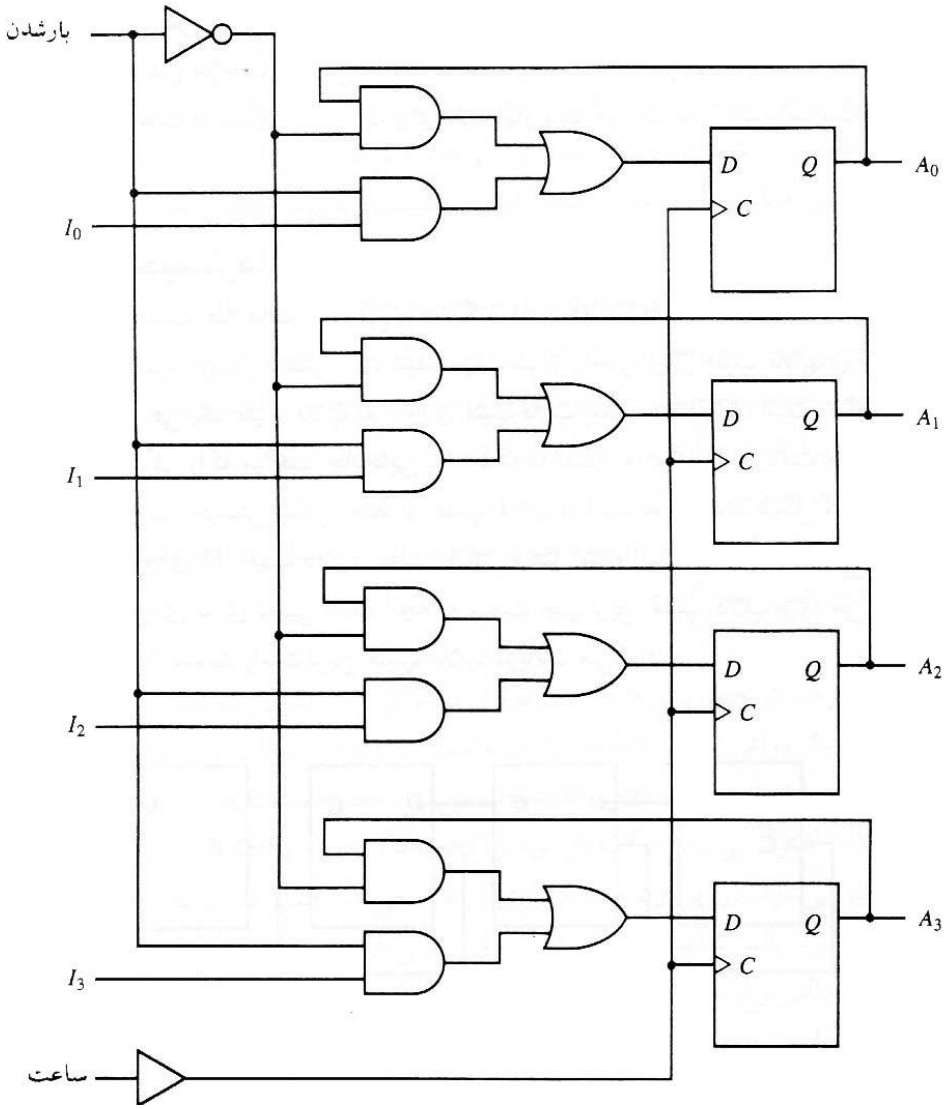
شکل۳-۱: ثبات ۴ بیتی

انتقال اطلاعات جدید بداخل ثبات «بار کردن» ثبات نامیده میشود. اگر تمام بیتهای ثبات بطور همزمان با یک انتقال پالس ساعت بار شوند، گوئیم بار شدن بصورت موازی صورت گرفته است. یک گذار یا انتقال پالس ساعت اعمال شده به ورودیهای C در ثبات شکل ۳-۱ تمام چهار ورودی I_0 تا I_3 را به صورت موازی بار خواهد کرد. در این آرایش، اگر بخواهیم محتوای ثبات بدون تغییر باقی بماند باید پالس ساعت به مدار اعمال نگردد.

ثبات با بار شدن موازی

بسیاری از سیستمهای دیجیتال دارای یک مدار ساعت اصلی میباشند که رشتهای از پالس ساعت را فراهم مینماید. این پالسهای ساعت به تمام فلیپ فلاپها و ثباتها در سیستم اعمال میشوند. ساعت اصلی مانند پمپی که ضربان ثابتی رابه تمام بخشها ارسال کند عمل مینماید. سیگنال کنترل جداگانه را باید بکار برد تا در مورد تاثیر پالس ساعت خاص بر روی ثبات خاص تصمیم گیری نماید.

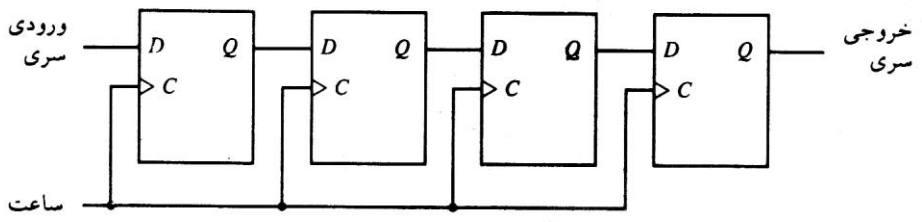
یک ثبات چهاربیتی با ورودی کنترل بار کردن که از طریق گیت ها به ورودیهای D میرسند در شکل٤-١ نشان داده شدهاند. ورودی هایC پالسهای ساعت را در همه زمانها دریافت میکنند. گیت بافر در ورودی ساعت توان دریافتی لازم را از مولد ساعت کاهش می دهد. هرگاه پالس ساعت فقط به ورودی یک گیت در عوض چند گیت وصل شود توان کمتری لازم است ولی چنانچه این گیت بافر مورد استفاده قرار نگرفته و پالس ساعت به هر چهار ورودی مستقیما وصل شود توان بیشتری از مولد پالس اخذ خواهد شد.



شکل ۴-۱: ثبات ۴ بیتی با بارشدن موازی

شیفت رجیسترها

ثباتی که قادر است اطلاعات دودویی را در یک یا دو جهت انتقال دهد شیفت رجیستر (یا ثبات انتقال) نامیده میشود. آرایش منطقی یک شیفت رجیستر از زنجیرهای از فلیپ فلاپهای متوالی تشکیل شده، که در آن خروجی یک فلیپ فلاپ به ورودی فلیپ فلاپ بعدی متصل شده است. تمام فلیپ فلاپها پالس ساعت مشترکی را که موجب جابجایی یک طبقه به طبقه بعدی است دریافت میکنند.



شکل ۵-۱: شیفت رجیستر ۴ بیتی

شیفت رجیستر دو جهته با بارشدن موازی

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

۱ -یک ورودی پالس های ساعت برای همزمانی تمام عملیات

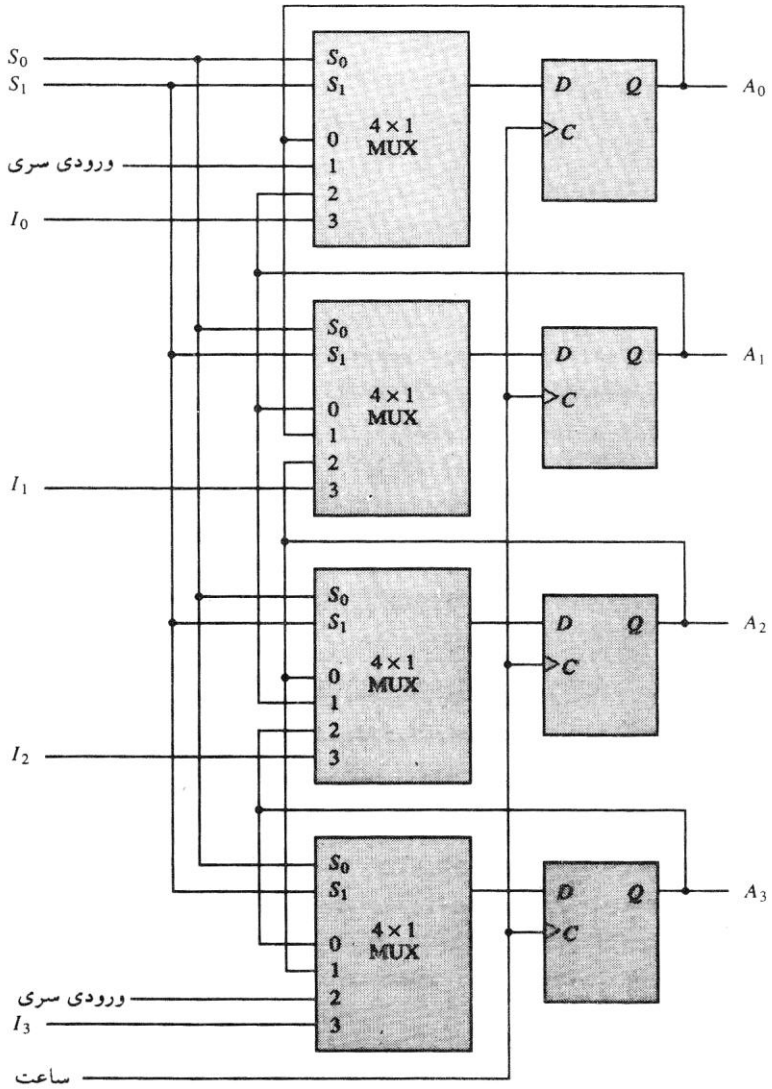
۲ -عمل شیفت به راست و یک خط ورودی سری برای شیفت به راست

۳ -عمل شیفت به چپ و یک خط ورودی سری مربوط به شیفت به چپ

٤ -عمل بارشدن موازی و n خط ورودی برای این انتقال موازی

٥ -n خط خروجی موازی

٦ -یک حالت کنترلی که با وجود اعمال مداوم پالس های ساعت، اطلاعات موجود در ثبات را بدون تغییر باقی گذارد.



شکل ۶-۱: شیفت رجیستر دوطرفه با بارشدن موازی

یک شیفت رجیستر چهاربیتی با بارشدن موازی در شکل ٦-١ نشان داده شده است. هرطبقه از یک فلیپ فلاپ D و یک مولتی پلکسر4×1 تشکیل شده است.

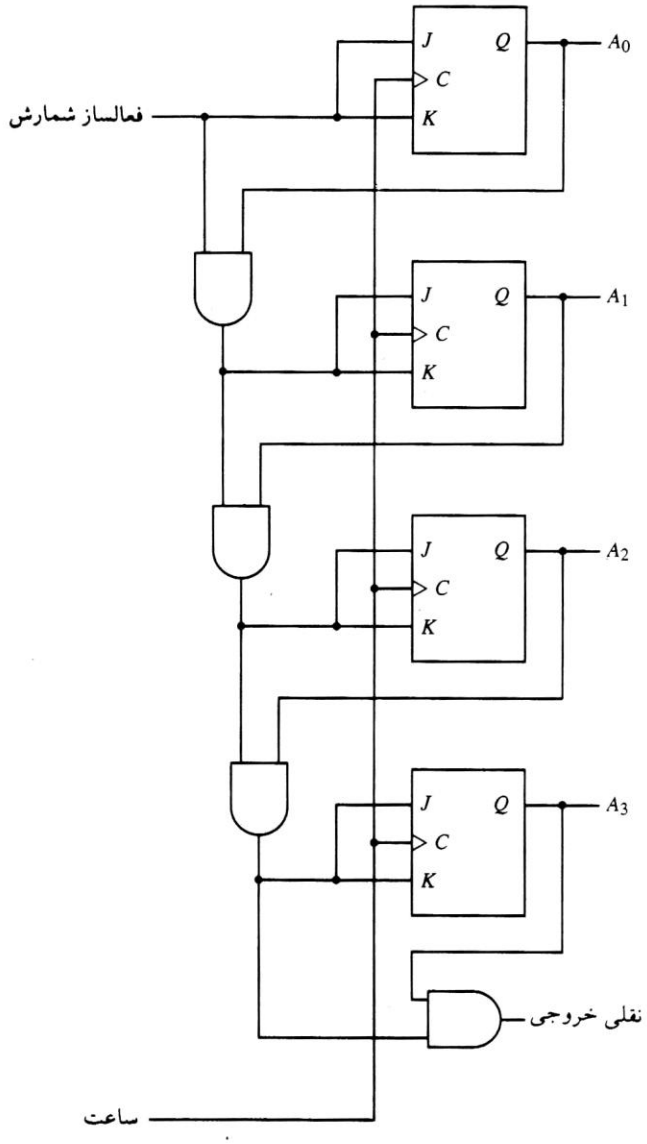
جدول۲-۱: جدول تابع برای ثبات شکل۶-۱

عملکرد ثبات	شیوه کنترل	
	S_0	S_1
بلاتغییر	0	0
شیفت به راست (پایین)	1	0
شیفت به چپ (بالا)	0	1
بارشدن موازی	1	1

شیفت رجیسترها اغلب برای ارتباط سیستم های دیجیتال، با فواصل دور از یکدیگر، بکار میروند. مثلا 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 پرهزینه است. ممکن است استفاده از یک خط مقرون به صرفه تر 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 می توان آن را از طریق خروجیهای موازی در ثبات برداشت کرد. بنابراین فرستنده یک عمل تبدیل موازی به سری داده ها را انجام میدهد و بالعکس گیرنده دادههای سری را به موازی باز میگرداند.

شمارنده های دودویی

ثباتی که با اعمال پالسهای ورودی رشته ای از حالت های از پیش تعیین شده را طی میکند، شمارنده خوانده میشود. پالسهای ورودی ممکن است پالس های ساعت و یا از یک منبع خارجی حاصل شده باشند. همچنین ممکن است فواصل زمانی پالس یکنواخت و یا تصادفی (غیریکنواخت) باشند. شمارنده ها را تقریبا در تمام تجهیزاتی که در آن ها مدارهای منطقی دیجیتال وجود دارد می توان یافت. از شمارندهها می توان برای شمارش تعداد دفعات وقوع یک واقعه استفاده کرد و نیز برای تولید سیگنال های زمانی بندی در کنترل رشتهای از اعمال در کامپیوترها بکار برد.

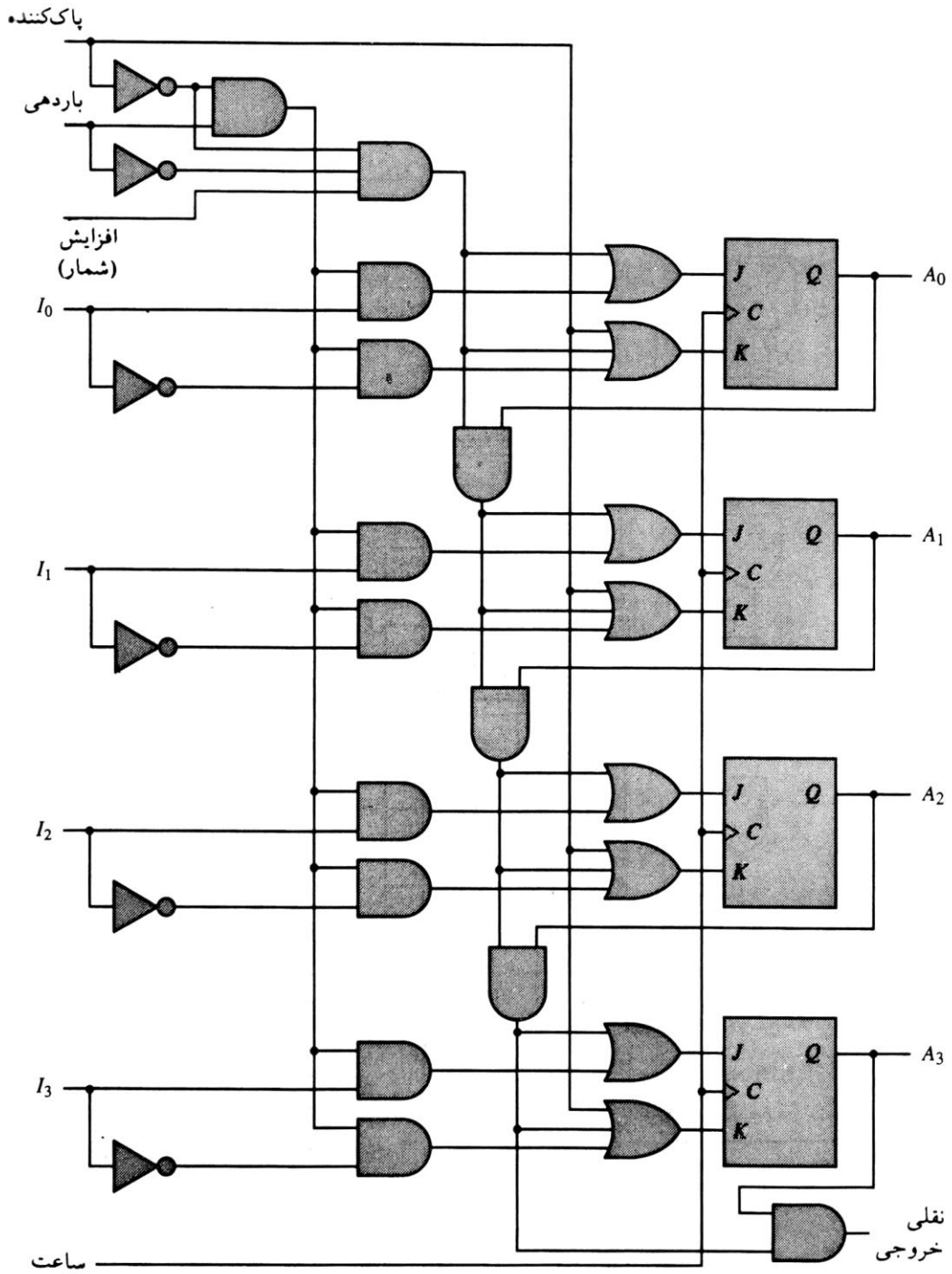


شکل ۷-۱: شمارنده همزمان ۴ بیتی همزمان

شمارنده دودویی با بار شدن موازی

شمارندههایی که در سیستمهای دیجیتال مورد استفاده قرار میگیرند اغلب نیاز به امکان بار شدن موازی برای دریافت یک مقدار اولیه قبل از شمارش دارند. شکل۷-۱ دیاگرام منطقی یک شمارنده دودویی که دارای امکان بار شدن موازی و نیز پاک شدن همزمان به0 میباشد را نشان میدهد.

عملکرد مدار در جدول۳-۱ خلاصه شده است



شکل ۸-۱: شمارنده دودویی ۴ بیتی با بارشدن موازی و پاک شدن همزمان

جدول ۳-۱: جدول تابع برای ثبات شکل ۸-۱

عمل	افزایش	بارشدن	پاک شدن	ساعت
بلاتغییر	0	0	0	↑
۱ واحد افزایش شمارش	1	0	0	↑
بارکردن ورودی های I_0 تا I_3	×	1	0	↑
پاک کردن خروجی ها به 0	×	×	1	↑

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

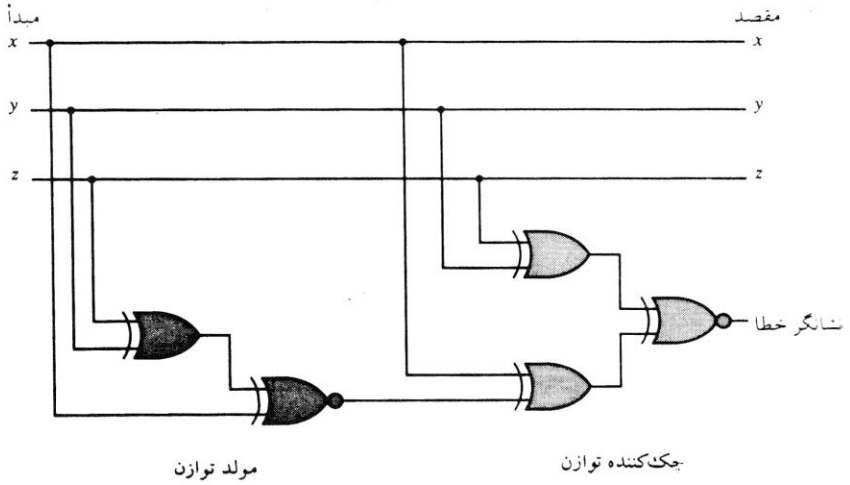
کدهای آشکارسازی خطا

اطلاعات دودویی انتقالی از طریق برخی از وسایل ارتباطی در معرض پارازیت های خارجی قرار دارند که ممکن است بیت ها را از 0 به 1 و بالعکس تبدیل کند. کد کشف خطا یک کد دودویی است که خطاهای رقمی رادر طول ارسال کشف می کند. خطاهای درک شده را نمی توان اصلاح کرد و فقط وجود آنها معلوم می شود. اگر خطاها غیرمتوالی و تصادفی رخ دهند، اطلاعات خاصی که خطا دارد مجددا ارسال میگردد. اگر خطا بیش از حد معقول رخ دهد سیستم از نظر وجود اشکال چک میشود.

متداولترین کد کشف خطا، بیت توازن است. بیت توازن بیتی است اضافی که به یک پیام دودویی اضافه می شود تا تعداد 1 ها را در آن زوج یا فرد نماید. یک پیام سه بیتی و دو نوع بیت توازن ممکن در جدول ٤-١ نشان داده شده است. بیت فرد P چنان اختیار می شود تا مجموع تمام 1 ها در پیام ارسالی و بیت P در نظر گرفته می شود. در هر کاربرد خاص یکی از انواع توازن بکار می رود. روش توازن زوج این اشکال را دارد که یکی از ترکیبات بیت ها تماما0 خواهد داشت، در حالی که در توازن فرد همیشه یکی از بیت ها (از چهار بیتی که پیام و P را تشکیل می دهند) 1 خواهد بود، دقت کنید که P فرد متمم P زوج است.

جدول ۴-۱: جدول تولید بیت توازن

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



شکل۹-۱: کشف خطا بوسیله بیت توازن فرد

تمرین:مدارهای یک مولد توازن سه بیتی و چک کننده توازن چهار بیتی را با استفاده از بیت توازن زوج بدست آورید.

فصل دوم

انتقال ثبات ها و ریز عمل ها

زبان انتقال ثبات

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

ماژولهای دیجیتال در بهترین فرم براساس ثباتها و عملیاتی که روی داده ای ذخیره شده در آنها انجام میشود تعریف میگردند. عملیاتی که روی دادههای ذخیره شده در ثبات ها صورت میگیرد ریزعمل نام دارد. به بیان دیگر ریزعمل 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 ریزعمل عبارتند از: شیفت، شمارش، پاک کردن و بارکردن است. برخی از مؤلفههای دیجیتال که در فصل ۱ معرفی شدند ثباتهایی هستند که ریزعمل ها را پیاده سازی میکنند. مثلا، یک شمارنده با قابلیت بارشدن موازی میتواند ریزعمل های افزایش و بارشدن را اجرا کند. یک ثبات شیفت (جابجایی) دوطرفه قادر است ریزعمل های شیفت به راست و چپ را انجام دهد.

سازمان داخلی یک کامپیوتر دیجیتال به بهترین نحو توسط موارد زیر مشخص میشود:

۱ -مجموعه ثبات های آن و وظایف آن ها

۲ -رشته ریزعملهای انجام شده روی اطلاعات دودویی ذخیره شده در ثبات ها

۳ -واحد کنترلی که موجب آغاز رشته ریزعملها میشود.

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

نحوه بیان سمبلیک مورد استفاده برای بیان انتقالهای ریزعملی در بین ثباتها، زبان انتقال ثباتها خوانده میشود. اصطلاح «انتقال ثباتها» بیانگر وجود مدارات منطقی سخت افزاری است که میتواند یک ریزعمل بیان شده را اجرا نماید و نتیجه عمل را به همان ثبات یا ثبات دیگر انتقال دهد. کلمه «زبان» از برنامه نویسان اقتباس شده است، که این کلمه را برای زبان های برنامه نویسی بکار می برند. زبان برنامه نویسی رویه ای برای نوشتن سمبلهایی است که فرایند محاسباتی خاصی را مشخص میکند. بطور مشابه، یک زبان طبیعی مانند انگلیسی، سیستمی است برای نوشتن سمبل ها و ترکیب آنها بصورت کلمات و جملات برای ارتباط بین انسانها. زبان انتقال ثبات هم سیستمی است برای بیان رشته ریزعمل ها بین ثبات های یک ماژول دیجیتال بصورت سمبلیک. چنین زبانی ابزار مناسبی برای توصیف فشرده و دقیق سازمان داخلی کامپیوترهای دیجیتال است. همچنین میتوان از آن برای تسهیل روند طراحی سیستم های دیجیتال استفاده کرد.

اعتقاد بر این است که زبان انتقال ثبات بکار رفته تا حد امکان ساده باشد، بنابراین بخاطر سپردن آن طولی نخواهد کشید. ما همچنان به تعریف سمبل برای انواع ریزعمل ها ادامه خواهیم داد، و همزمان با آن سخت افزاری که ریزعمل بیان شده را پیاده سازی کند معرفی میکنیم. علائم سمبلیک معرفی شده در این فصل، در فصلهای بعدی برای مشخص کردن انتقالات ثبات ها، ریزعملها، و توابع کنترلی که سازمان

سخت افزاری داخلی کامپیوترهای دیجیتال را بیان میکنند، بکار میروند. به محض آشنایی با این زبان، سایر سمبلهای بکار رفته بسادگی قابل آموختن است زیرا بیشتر تفاوتهای بین زبانهای انتقال ثبات در جزئیات آنهاست، نه در کلیات.

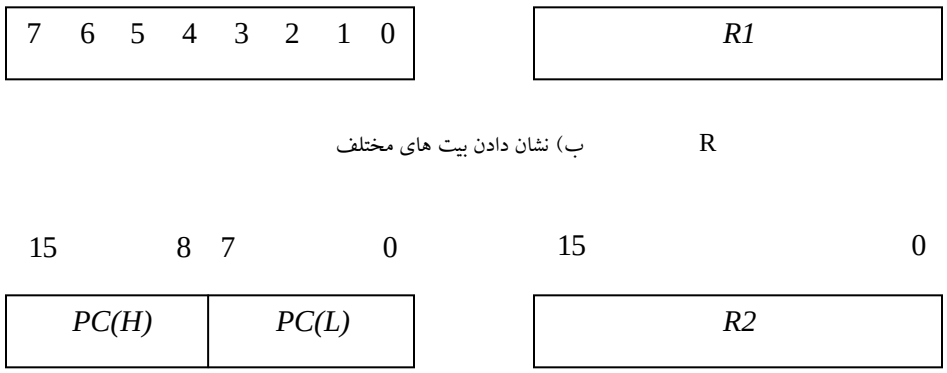
انتقال ثبات

ثباتهای کامپیوتر با حروف بزرگ الفبای انگلیسی (و گاهی عددهایی به دنبال آن ها) برای نشان دادن کار ثبات مشخص می شوند. مثلا، ثباتی که آدرس را برای یک واحد حافظه نگه میدارد ثبات آدرس حافظه نام داشته و با MAR مشخص میشود. سایر نام ها برای ثبات ها عبارتند از PC (برای شمارنده برنامه)، IR (برای ثبات دستورالعمل)، و R1 (برای ثبات پردازنده). فلیپ فلاپ های تشکیل دهنده یک ثبات n بیتی به طور متوالی از 0 الی n-1 از راست به چپ شماره گذاری میشوند. شکل ۱-۲ نمایش ثباتها را بصورت بلاک دیاگرام نشان میدهد. متداول ترین راه نمایش یک ثبات، استفاده از کادر مستطیل شکل همراه با نام ثبات در داخل آن است، شکل۱-۲ (الف). بیت های مختلف ثبات مانند قسمت (ب) در شکل متمایز میشوند. شماره گذاری بیت ها در یک ثبات ۱٦ بیتی، مانند قسمت (ج) در بالای آن مشخص میشود. یک ثبات ۱٦ بیتی هم در قسمت (د) به دو قسمت تقسیم شده است. به بیت های 0 تا 7 علامت KL (برای بایت کم ارزشتر) و به بیت های 8 تا15 سمبل H (برای بایت باارزشتر) بکار میرود. نام ثبات 16 بیتی PC است. سمبل (PC(0-7) به بایت کم ارزشتر اتلاق می شود و PC(8-15) یا PC(H) به بایت باارزشتر اشاره می کند.

انتقال اطلاعات از یک ثبات به ثبات دیگر بصورت سمبلیک با استفاده از عملگر جایگزین مشخص می شود، عبارت:

$$R2 \leftarrow R1$$

بیانگر یک انتقال از ثبات R1 به ثبات R2 است. این عبارت جایگزین شدن محتوای R2 را با R1 مشخص می نماید. بنابه تعریف، محتوای ثبات R1 پس از انتقال تغییر نمی کند.



الف) ثبات R ب) نشان دادن بیت های مختلف

ج) شماره گذاری بیت ها د) ثباتی که به دو بخش تقسیم شده

شکل ۲-۱: بلوک دیاگرام ثبات

عبارتی که مشخص کننده انتقال ثبات ها باشد دلالت بر این دارد که مدارهایی از خروجیهای ثبات مبدأ به ورودیهای ثبات مقصد وجود دارند و ثبات مقصد دارای قابلیت بارشدن موازی است. معمولا می خواهیم که انتقال تحت شرایط کنترل از پیش تعیین شدهای انجام شود. این موضوع را میتوان 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 میشود. گاهی بهتر است که متغیرهای کنترل را با مشخص کردن یک تابع کنترل از عمل انتقال ثبات جدا کنیم. تابع کنترل یک متغیر بولی است که برابر 1 یا 0 است. تابع کنترل در عبارت

بصورت زیر مشخص میشود: $P : P2 \leftarrow R1$

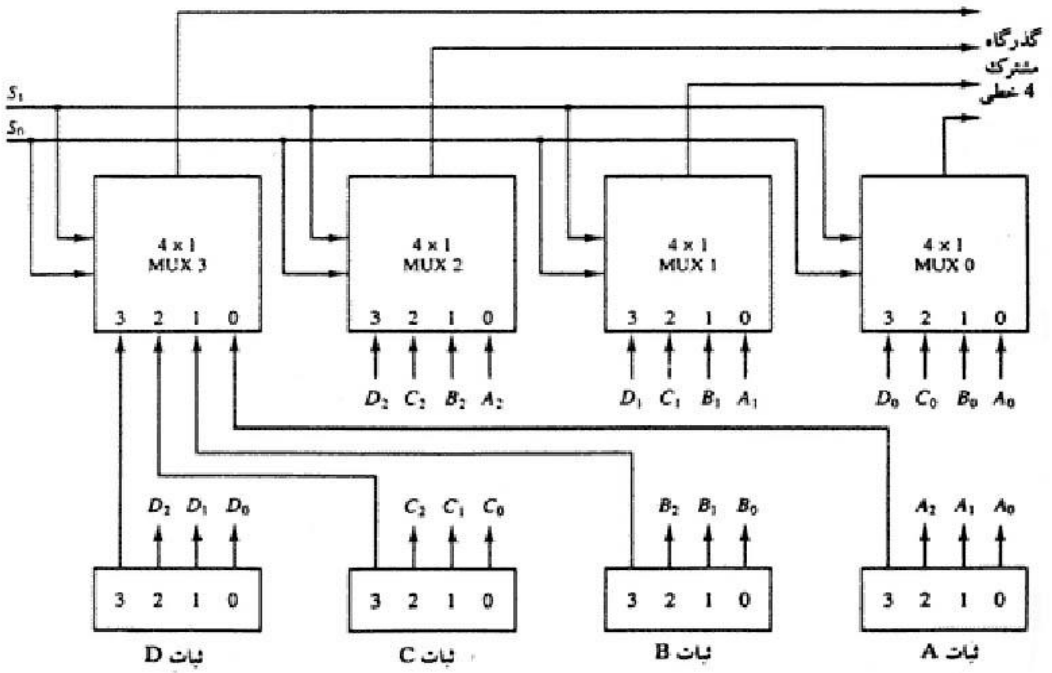
شرط کنترل با علامت نقل، : ، خاتمه می یابد. این سمبل بیان میدارد که لازمه عمل انتقال توسط سخت - افزار این است که $P=1$ باشد.

سمبلهای اصلی زبان انتقال ثبات ها در جدول ۱-۲ لیست شدهاند. ثباتها با حروف بزرگ نشان داده می - شوند و ممکن است اعدادی به دنبال این حروف آورده شود. برای مشخص کردن بخشی از یک ثبات از پرانتز استفاده میشود و محدوده بیتها یا یک نام سمبلیک برای آن بخش در داخل پرانتز ذکر میشود. علامت فلش بیانگر انتقال اطلاعات و جهت آن است. از کاما برای جدا کردن دو یا چند عمل که همزمان انجام میشوند استفاده میشود. عبارت $T : R2 \leftarrow R1, \quad R1 \leftarrow R2$

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

انتقالهای گذرگاهی و حافظهای

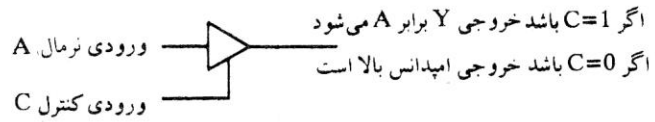
یک کامپیوتر دیجیتال نوعاً ثباتهای زیادی دارد و باید در آن مسیرهایی برای انتقال اطلاعات از یک ثبات به ثبات دیگر فراهم شود. اگر خطوط جداگانهای بین هر ثبات و دیگر ثباتهای سیستم به کار رود، تعداد سیمها بیش از حد خواهد شد. روش کاراتر برای انتقال اطلاعات بین ثباتها در یک آرایش چند ثباتی، سیستم گذرگاه مشترک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



شکل ۲-۲: سیستم گذرگاه برای چهار ثبات

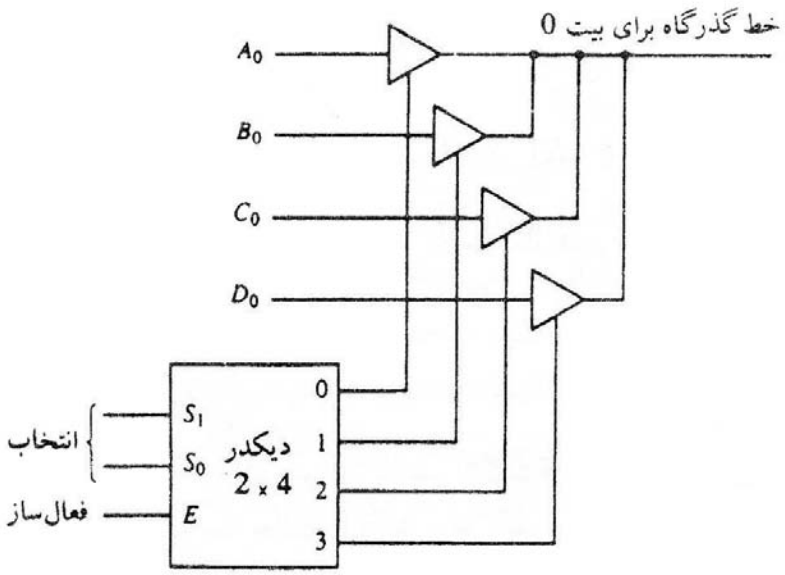
بافرهای سه حالتی گذرگاه

گذرگاه سیستم میتواند بجای مولتی پلکسر از 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 1 و 0 منطقی همانند گیتهای معمولی هستند. حالت سوم، حالت امپدانس بالاست. این حالت مانند مدار باز عمل می کند، یعنی خروجی قطع است و مفهوم منطقی ندارد.



شکل۳-۲: سمبل گرافیکی بافر سه حالته

ساختن یک سیستم گذرگاه با میانگیرهای سه حالتی در شکل٤-۲ نشان داده شده است.



شکل ۴-۲: خط گذرگاه با بافرهای سه حالت

ریزعمل ها

ریزعمل یک عمل جزیی است که روی داده های ذخیره شده در ثبات ها انجام می شود. ریزعمل هایی که بیش از همه در کامپیوترهای دیجیتال با آن مواجه می شویم به چهار دسته زیر طبقه بندی می شوند:

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

ریزعملهای حسابی عملیات محاسباتی را روی دادههای عددی ذخیره شده در ثبات ها انجام می-دهند.

ریزعملهای منطقی عملیات دستکاری بیتها را روی دادههای غیرعددی ذخیره شده در ثبات ها انجام میدهند.

ریزعملهای شیفت اعمال شیفت (جابجایی) را روی داده های ذخیره شده در ثبات ها اجرا می - کنند.

ریزعمل های انتقال ثبات

این نوع ریزعمل محتوای اطلاعاتی را، هنگامی که از ثباتهای مبدأ به ثبات مقصد منتقل میشود، تغییر نمیدهد. سه نوع دیگر ریزاعمال محتوای اطلاعاتی را حین انتقال تغییر میدهند.

ریزعملهای حسابی پایه

ریزعملهای حسابی پایه عبارتند از جمع، تفریق، افزایش، کاهش و شیفت. ریزعمل حسابی زیر

$$R3 \leftarrow R1 + R2$$

بیانگر ریزعمل «جمع» است. این عبارت بیان میکند که محتوای ثبات R1 با محتوای ثبات R2 جمع شده و حاصل جمع به ثبات R3 انتقال مییابد. برای پیادهسازی این عبارت با سخت افزار، ما به سه ثبات و قطعات منطقی که عمل جمع را انجام میدهند نیاز داریم. سایر ریزعملهای اصلی حسابی در جدول ۱-۲ لیست شده اند. تفریق غالبا با استفاده از متمم سازی و جمع پیاده سازی میشود. در عوض بکارگیری عملگر «منها»، ما تفریق را به وسیله عبارت زیر مشخص میکنیم:

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

$\overline{R2}$ سمبل متمم 1 ثبات R2 است. با جمع 1 با متمم 1 متمم 2 تولید میشود. جمع محتوای R1 با متمم 2 محتوای ثبات R2 برابر است باR1-R2.

ریزاعمال افزایش و کاهش را بترتیب بوسیله عمل های جمع با 1 و تفریق 1 انجام میدهیم. این ریزاعمال با مدارهای ترکیبی یا با شمارنده های بالا-پایین پیاده سازی میشوند.

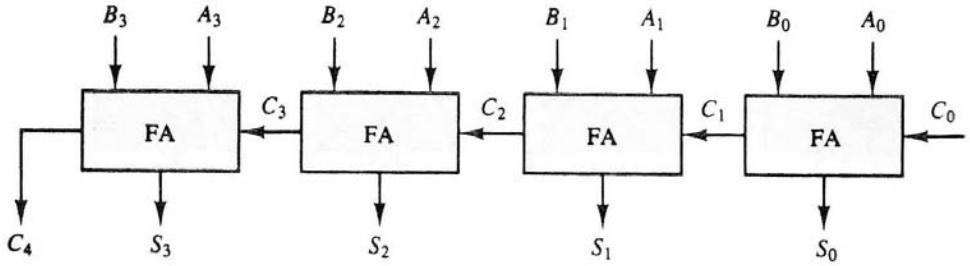
اعمال حسابی ضرب و تقسیم در جدول ۱-۲ لیست نشدهاند. این دو عمل، اعمال حسابی معتبری هستند ولی در مجوعه دستورات پایه منظور نشدهاند. تنها جایی که این اعمال بعنوان ریزعمل در نظر گرفته میشوند، در یک سیستم دیجیتال است که در آن این اعمال بصورت مدارهای ترکیبی پیاده سازی می شوند. در چنین حالتی، سیگنالهایی این اعمال را در طول گیت ها منتشر می نمایند و نتیجه عمل میتواند به محض اینکه سیگال 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 ریزعمل های جمع و شیفت اجرا می شود. تقسیم بارشته ای از ریزعمل های تفریق و شیفت صورت می گیرد. مشخص کردن سخت افزار برای این موارد مستلزم استفاده از لیست عباراتی است که در آنها ریزعمل های جمع و تفریق و شیفت به کار رفته اند.

جدول۱-۲: ریزعمل های حسابی

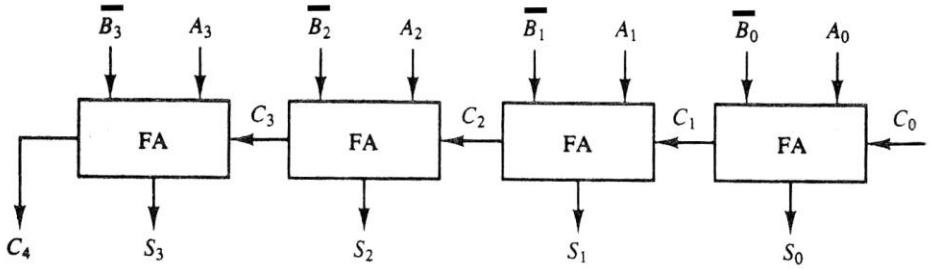
نمایش سمبلیک	شرح
$R3 \leftarrow R1 + R2$	محتوای R1 بعلاوه R2 به R3 منتقل می شود
$R3 \leftarrow R1 - R2$	محتوای R1 بعلاوه R2 به R3 منتقل می شود
$R2 \leftarrow \overline{R2}$	محتوای R1 بعلاوه R2 به R3 منتقل می شود
$R2 \leftarrow \overline{R2} + 1$	محتوای R2 متمم می شود (منفی می شود)
$R3 \leftarrow R1 + \overline{R2} + 1$	R1 بعلاوه متمم2 از R2 (تفریق)
$R1 \leftarrow R1 + 1$	یک واحد افزایش در محتوای R1
$R1 \leftarrow R1 - 1$	یک واحد کاهش در محتوای R1

جمع کننده دودویی

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



شکل ۵-۲: جمع کننده دودویی ۴ بیتی

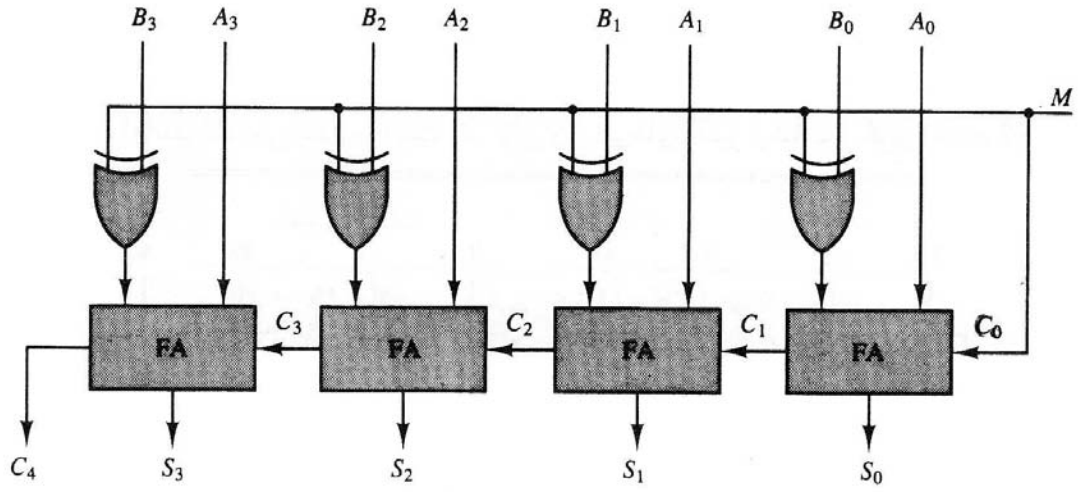


شکل ۶-۲: تفریق کننده دودویی ۴ بیتی

جمع و تفریق کننده دودویی

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 1 حاصل می شود. متمم ۱ را هم با گیت های معکوس کننده میتوان ایجاد کرد و جمع کردن یک واحد را نیز می توان از طریق ورودی نقلی انجام داد.

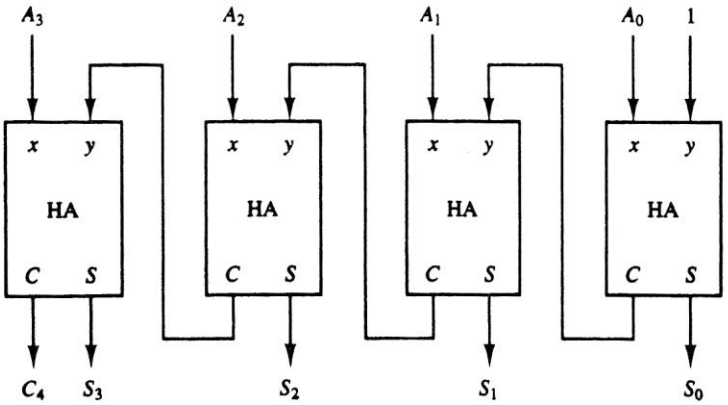
اعمال جمع و تفریق را می توان با افزودن یک گیتOR انحصاری با هم ترکیب کرده و بصورت یک مدار مشترک درآورد. یک مدار جمع و تفریق کننده چهاربیتی در شکل ۷-۲ نشان داده شده است. ورودی حالتM عمل را کنترل می کند.



شکل ۷-۲: جمع و تفریق کننده ۴ بیتی

افزایشگر دودویی

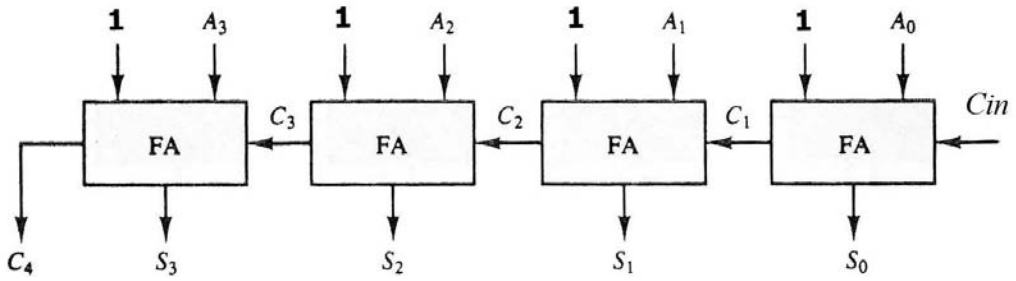
دیاگرام مدار ترکیبی افزایشگر چهاربیتی در شکل ۸-۲ نشان داده شده است. یکی از ورودی های نیمجمع کننده با ارزش کمتر به ۱ متصل شده است و ورودی دیگر به بیت کم ارزشتر عددی که قرار است افزایش یابد وصل می شود. رقم نقلی خروجی از یک نیم جمع کننده به یکی از ورودی های نیم جمع کننده مرتبه بالاتر بعدی اتصال یافته است. مدار، چهار بیت A_0 تا A_3 را دریافت کرده، یک واحد به آن اضافه می کند، و خروجی افزایش یافته را در S_0 تا S_3 تولید مینماید. رقم نقلی خروجی C_4 فقط پس از حالت دودویی 1111 برابر 1 خواهد شد. در این حالت S_0 تا s_3 هم 0 می شوند.



شکل۸-۲: افزایشگر دودویی ۴ بیتی

کاهشگر دودویی

در این مدار می خواهیم عدد A را یک واحد کاهش دهیم یعنی S=A-1



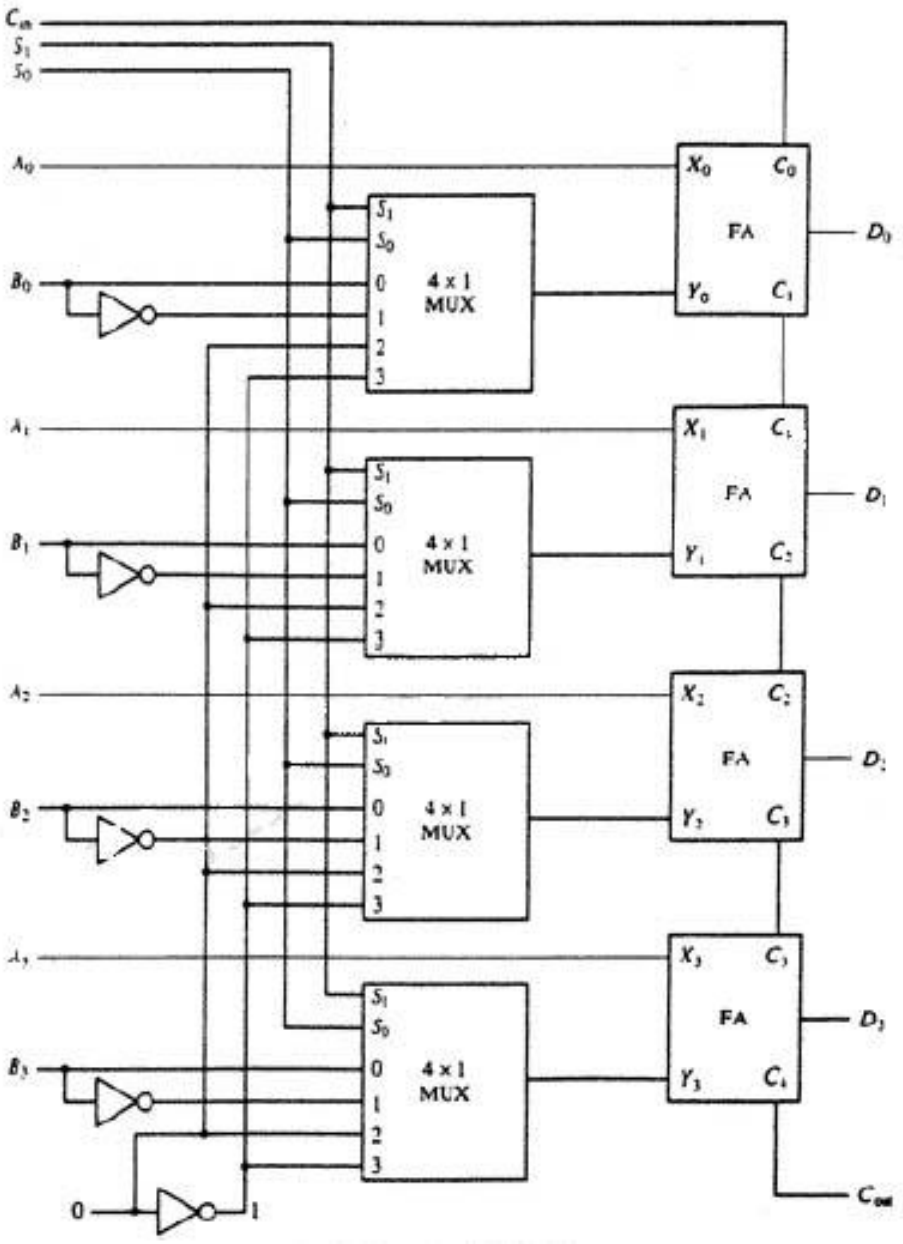
شکل۹-۲: کاهشگر دودویی ۴ بیتی

مدار حسابی ٤ بیتی

دیاگرام یک مدار حسابی ٤ بیتی در شکل ۱۰-۲ نشان داده شده است. در این دیاگرام چهار مدار تمام کننده وجود دارد که جمع چهار بیت را به وجود میآورد و چهار مولتی پلکسر نیز اعمال مختلف را انتخاب میکنند. دو ورودی چهار بیتی A و B و یک خروجی4 بیت نیز وجود دارند.

جدول۲-۲: جدول تابع مدار حساب

ریزعمل	خروجی $D=A+Y+C_{in}$	ورودی Y	انتخاب		
			C_{in}	S_0	S_1
جمع	$D = A + B$	B	0	0	0
جمع با نقلی	$D = A + B + 1$	B	1	0	0
تفریق با فرض	$D = A + \bar{B}$	$\bar{B}$	0	1	0
تفریق	$D = A + \bar{B} + 1$	$\bar{B}$	1	1	0
انتقال A	$D = A$	0	0	0	1
افزایش A	$D = A + 1$	0	1	0	1
کاهش A	$D = A - 1$	1	0	1	1
انتقال A	$D = A$	1	1	1	1



شکل ۱۰-۲: مدار حساب ۴ بیتی

ریزعمل های منطقی

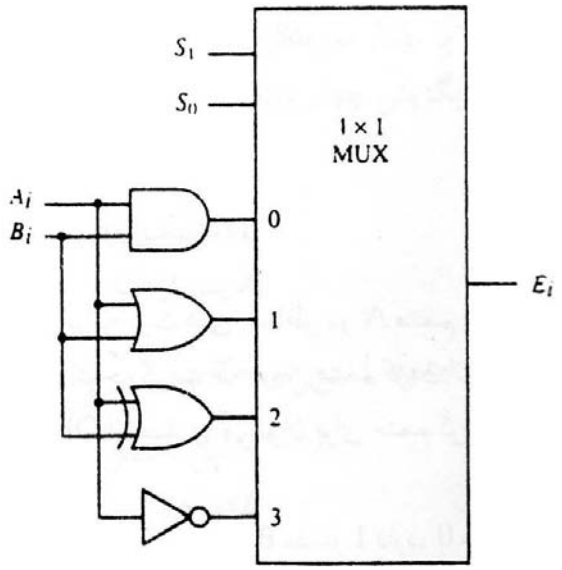
ریزعملهای منطقی اعمال دودویی را برای رشته ای از بیتهای ذخیره شده در ثباتها مشخص مینمایند. اعمال هر بیت ثبات را بطور جداگانه در نظر گرفته و با آنها بعنوان متغیر دودویی رفتار میکند.

شانزده عمل منطقی وجود دارد که میتوان با دو متغیر دودویی انجام داد. این عمل ها به صورت جبری در جدول۳-۲ آورده شده است.

شکل ۱۱-۲ یک طبقه از مدار را که چهار ریزعمل اصلی منطقی را اجرا مینماید، نشان میدهد.

جدول ۳-۲: شانزده ریزعمل منطقی

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



(الف) نمودار منطقی

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	$E = \bar{A}$	متمم

(ب) جدول تابع

شکل ۱۱-۲: یک طبقه از مدار منطق

ریزعمل های شیفت

سه نوع شیفت وجود دارد: منطقی، چرخشی و حسابی.

شیفت منطقی مقدار 0 را از طریق ورودی سری انتقال میدهد. ما سمبل shl و shr را بترتیب برای ریزعملهای شیفت به چپ و شیفت به راست برمیگزینیم. مثلاً

$$R1 \leftarrow shl\ R1$$

$$R2 \leftarrow shr\ R2$$

دو نمونه ریزعمل میباشند که محتوای ثبات R1 را یک بیت به چپ و محتوای ثبات R2 را یک بیت به راست شیفت میدهند. سمبل ثبات در هر دو طرف فلش باید یکسان باشند. فرض بر این است که حین شیفت، بیت انتقالی به مکان انتهایی از طریق ورودی سری0 باشد.

شیفت چرخشی (که عمل چرخش نیز خوانده می شود) بیتهای ثبات را از طریق دو انتها بدون از دست دادن هرگونه اطلاعات میچرخاند. این عمل با اتصال خروجی سری به ورودی سری ثبات تحقق می یابد. اما سمبل های cil و cir را بترتیب برای چرخش به چپ و چرخش به راست بکار خواهیم برد. سمبلهای بکار رفته و ریزعمل های شیفت در جدول ٤-٢ نشان داده شدهاند.

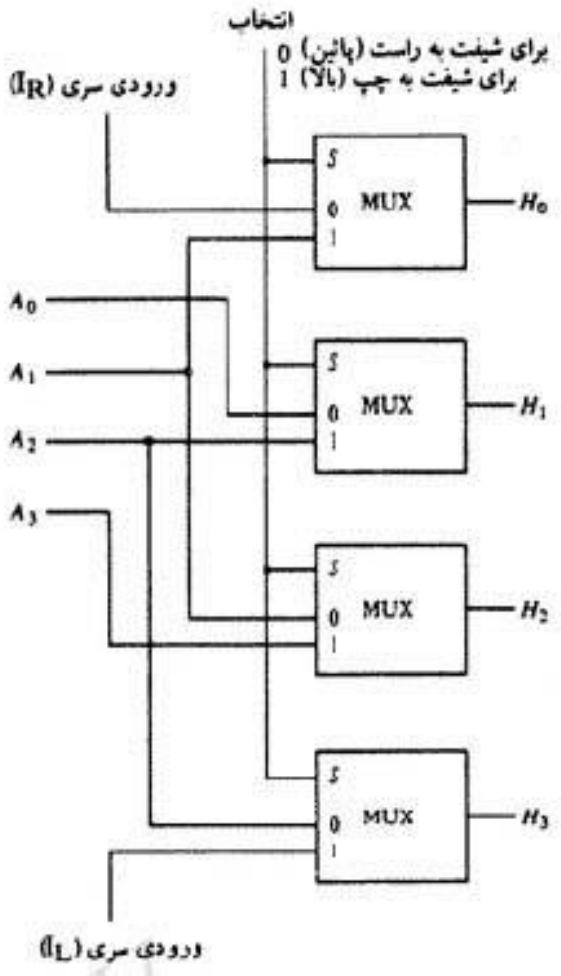
شیفت حسابی ریزعملی است که یک عدد دودویی علامتدار را به چپ یا راست شیفت میدهد. شیفت حسابی به چپ، یک عدد دودویی علامت دار را در ۲ ضرب میکند. یک شیفت حسابی به راست نیز عدد را بر ۲ تقسیم مینماید. شیفتهای حسابی نباید بیت علامت را تغییر دهند زیرا وقتی عدد را در ۲ ضرب یا تقسیم می کنیم علامت همچنان باقی میماند. سمت چپترین بیت در ثبات بیت علامت را نگه میدارد و بقیه بیت ها عدد را حفظ میکنند. بیت علامت برای اعداد مثبت، 0 و برای منفی، 1 است. اعداد

منفی در فرم متمم ۲ هستند. شکل ۱۲-۲ 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 و R_0 کم ارزشترین بیت است.

جدول ۴-۲ زیرعمل های شیفت

سمبل نشاندهنده	توضیح
$R \leftarrow shlR$	شیفت ثبات R به چپ
$R \leftarrow shrR$	شیفت ثبات R به راست
$R \leftarrow cilR$	شیفت چرخش ثبات R به چپ
$R \leftarrow cirR$	شیفت چرخش ثبات R به راست
$R \leftarrow ashlR$	شیفت حسابی ثبات R به چپ
$R \leftarrow ashrR$	شیفت حسابی ثبات R به راست

یک شیفت دهنده ترکیبی را می توان مانند شکل ۱۳-۲ با مولتی پلکسر ساخت. شیفت دهنده ٤ بیتی دارای چهار ورودی داده A_0 تا A_3 و چهار خروجی داده H_0 الی H_3 است. دو ورودی سری نیز وجود دارد، یک 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 می شود (در دیاگرام به سمت پایین). اگر $S=1$ باشد، داده ورودی به چپ شیفت می یابد (به سمت بالای دیاگرام). با n ورودی و خروجی داده به n مولتی پلکسر نیاز دارد. دو ورودی سری بوسیله مولی پلکسر دیگری قابل کنترل اند تا سه نوع شیفت ممکن را فراهم آورند.



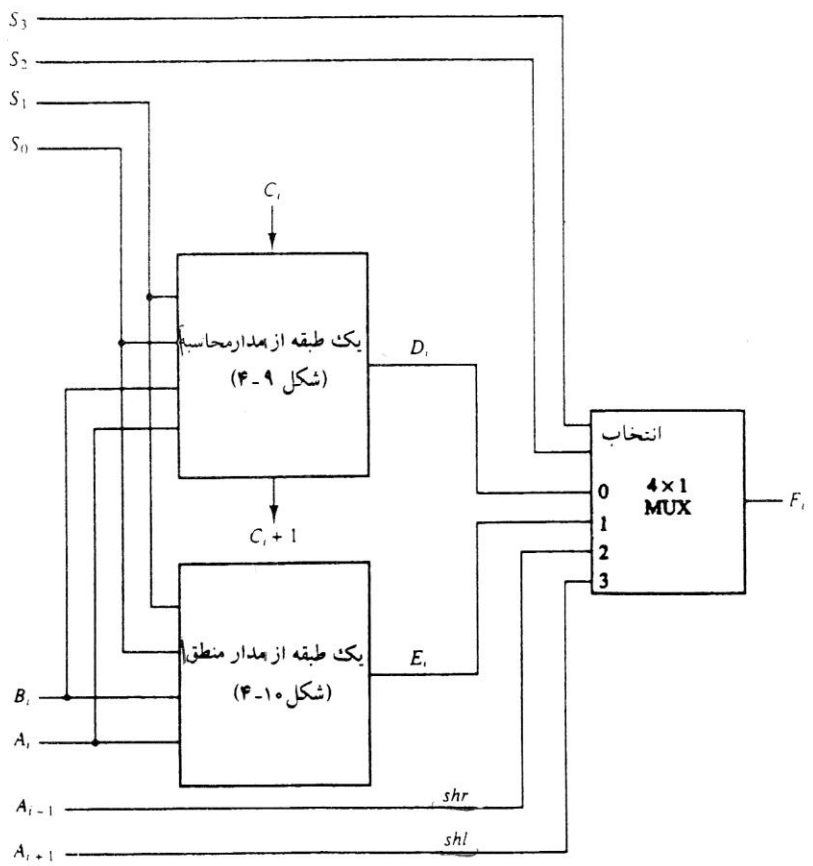
جدول تابع

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

شکل۱۳-۲: شیفت ترکیبی ۴ بیتی

واحد حساب، منطق و شیفت

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



شکل ۱۴-۲: یک طبقه از واحد حساب، منطق و شیفت

جدول ۵-۲: تابع برای واحد حساب، منطق و شیفت

انتخاب کننده عمل						
S_3	S_2	S_1	S_0	C_{in}	عمل	تابع
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	$F = A+\overline{B}$	تفریق با فرض
0	0	1	0	1	$F = A+\overline{B}+1$	تفریق
0	0	1	1	0	$F = A-1$	کاهش A
0	0	1	1	1	$F = A$	انتقال A
0	1	0	0	×	$F = A \wedge B$	AND
0	1	0	1	×	$F = A \vee B$	OR
0	1	1	0	×	$F = A \oplus B$	XOR
0	1	1	1	×	$F = \overline{A}$	متمم کردن A
1	0	×	×	×	$F = shrA$	شیفت A به راست و به داخل F
1	1	×	×	×	$F = shlA$	شیفت A به چپ و به داخل F

فصل سوم

سازمان و طراحی

یک کامپیوتر پایه

کدهای دستورالعمل ها

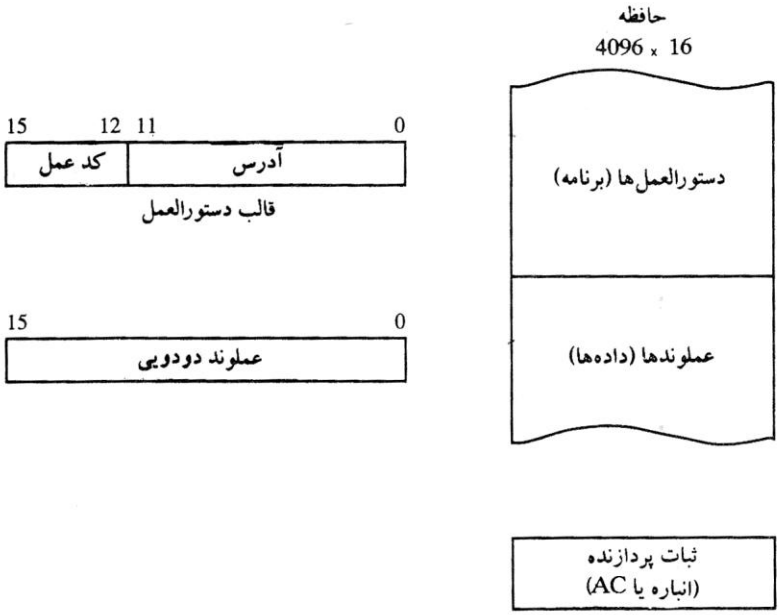
در این فصل یک کامپیوتر پایه را معرفی میکنیم و نشان خواهیم داد که چگونه میتوان عملکرد آن را توسط عبارات انتقال ثبات مشخص نمود. سازمان کامپیوتر بوسیله ثباتهای داخلیاش، زمانبندی و ساختار کنترل، و مجموعه دستوراتی که به کار میبرد تعریف میگردد. پس از آن طراحی کامپیوتر مشروحاً انجام شده است. هرچند که کامپیوتر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

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

یک کد دستورالعمل مجموعه ای از بیت هاست که انجام یک عمل خاص را به کامپیوتر فرمان میدهد. این کد معمولا به دو قسمت تقسیم میشود و هریک تفسیر خاص خود را داراست. اصلیترین بخش کد دستور بخش عمل آنست. کد عمل در یک دستور گروهی از بیت هاست که اعمالی مانند جمع، تفریق، ضرب، شیفت و متممسازی را تعریف مینمایند.

این عمل باید بر روی دادههایی که در ثباتهای پردازشگر و یا حافظه ذخیره شدهاند صورت گیرد. بنابراین در کد دستور نه تنها عمل بلکه ثبات و یا حافظههایی که دادهها در آنها میتوانند یافت شوند و یا در آنها ذخیره شوند نیز مشخص میشود. کلمات حافظه توسط آدرسشان در دستورالعمل معین میگردند.

سادهترین راه سازماندهی یک کامپیوتر یک ثبات پردازنده و قالب کد دستورالعملی متشکل 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 را در حافظه نشان میدهد. این عملوند از حافظه خوانده شده و بر روی آن بعنوان داده ذخیره شده در ثبات پردازشگر عمل میشود.



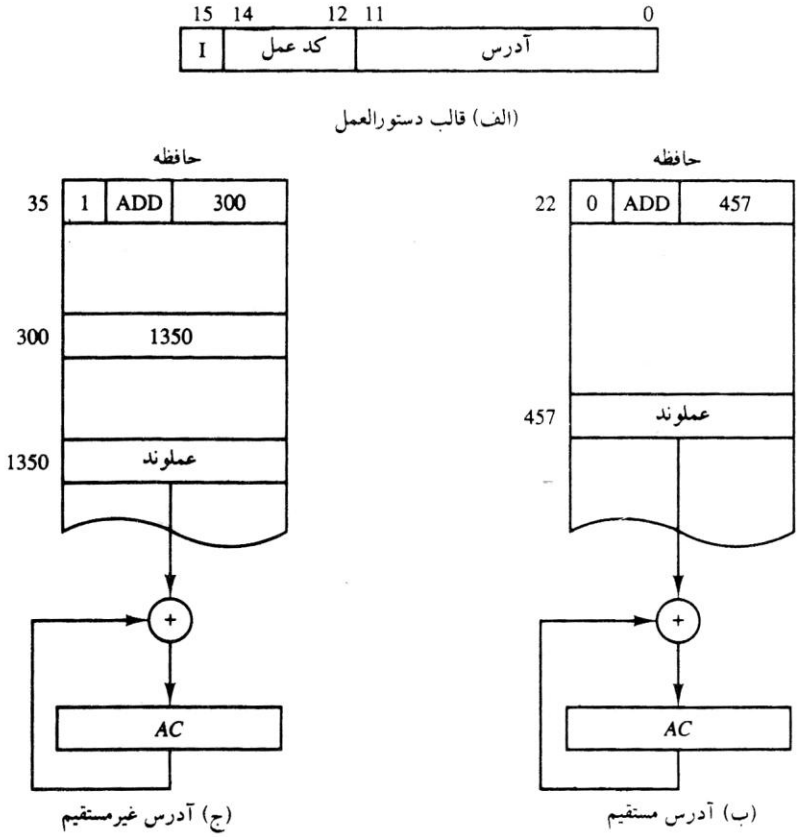
شکل۱-۳: سازمان مبتنی بر برنامه ذخیره شده

آدرس غیرمستقیم

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

بیت I برای آدرس دهی مستقیم برابر0 و برای آدرس دهی غیرمستقیم1 میباشد.

لذا دستورالعملها با آدرس غیرمستقیم برای برداشتن عملوند نیاز به دو ارجاع به حافظه دارند. اولین ارجاع برای خواندن آدرس عملوند؛ و دومی برای خواندن خود عملوند است.



شکل ۳-۳: نمایش آدرس مستقیم و غیرمستقیم

ثبات های کامپیوتر

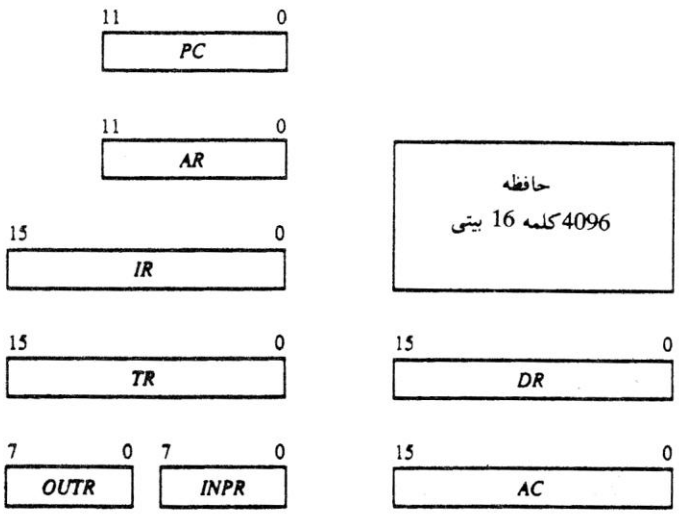
دستورالعملهای کامپیوتر معمولا در مکانهایی از حافظه بطور متوالی ذخیره شده و هریک بنوبت اجرا میگردند. واحد کنترل یک دستور را از آدرس خاصی در حافظه خوانده و آن را اجرا میکند. سپس بههمین ترتیب با خواندن دستور بعدی و اجرای آن، روند را ادامه میدهد. اینگونه توالی در خواندن و اجرای دستورات، به شمارنده نیاز دارد 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 کامپیوتر به ثبات هایی در پردازشگر جهت دستکاری داده ها و نیز یک ثبات برای ذخیره سازی آدرس حافظه نیاز دارد.

این ثبات ها در جدول۱-۳ هم همراه با شرحی مختصر از کارشان و تعداد بیت های آن ها آورده شده اند.

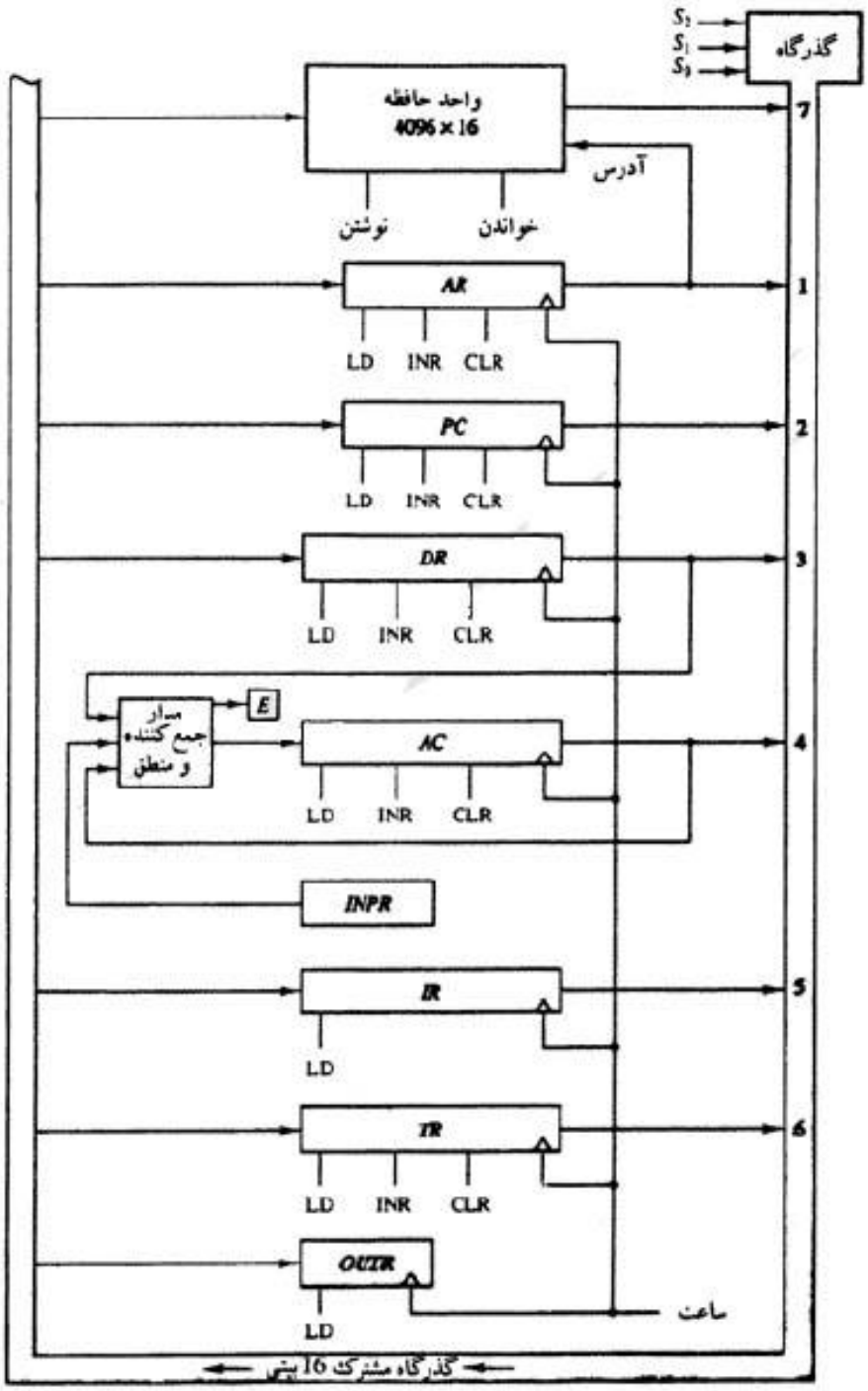
واحد حافظه 4096 کلمه 16 بیتی ظرفیت دارد. دوازده بیت از کلمه برای مشخص کردن آدرس عملوند لازم است. لذا سه بیت برای بخش عمل دستور و یک بیت برای تعیین مستقیم یا غیرمستقیم بودن آدرس باقی میماند.

جدول ۱-۳: لیست ثبات های کامپیوتر ساده

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



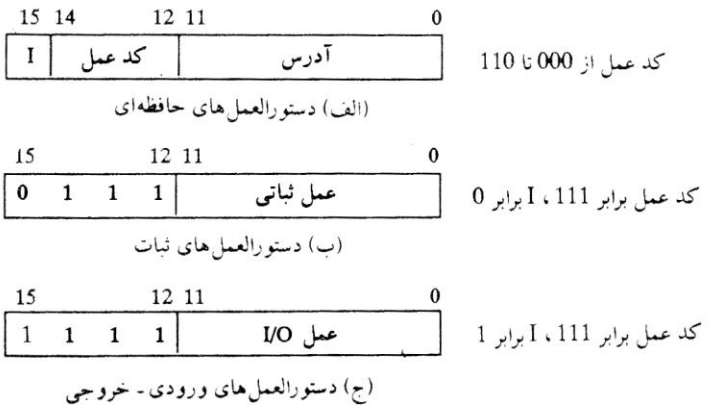
شکل ۴-۳: ثبات ها و حافظه کامپیوتر



شکل۵-۳: ثبات های کامپیوتر پایه متصل به یک گذرگاه مشترک

دستورالعمل های کامپیوتر

بخش عمل کد دستور سهبیتی است و مفهوم بقیه بیتها به نوع کد وابسته است. دستورالعملهای ارجاع به حافظه از ۱۲ بیت برای تعیین آدرس و ۱ بیت برای مشخصکردن روش آدرس دهی I استفاده میکنند. برای آدرسدهی مستقیم، I برابر 0 و برای آدرسدهی غیرمستقیم I برابر 1 است، شکل ٦-۳. دستورالعملهای ارجاع به ثبات با کد عملی 111 و یک بیت 0 در منتهاالیه سمت چپ آن (بیت ۱۵) قابل تشخیصاند. دستورالعملهای ثباتی عمل را بر روی AC و یا هر تستی بر روی آن را مشخص مینمایند. در این دستورات عملوندی از حافظه مورد نیاز نیست؛ بنابراین از ۱۲ بیت باقیمانده برای مشخص کردن عمل و یا تست مورد نظر استفاده میشود. بطور مشابه، دستورات ورودی – خروجی نیز نیاز به ارجاع به حافظه ندارند و با کد عملیاتی 111 و 1 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 میسازد.



شکل ۶-۳: قالب دستورالعمل ها در کامپیوتر پایه

واحد کنترل کامپیوتر نوع دستور العمل را با توجه به بیتهای مکانهای ۱۲ تا ۱۵ دستورالعمل نتشخیص میدهد. اگر سه بیت کد عمل در مکانهای ۱۲ تا ١٤ برابر 111 نباشد دستور از نوع ارجاع به حافظه است و بیت ۱۵ بعنوان بیت روش آدرسدهی I در نظر گرفته میشود. اگر سه بیت کد عمل 111

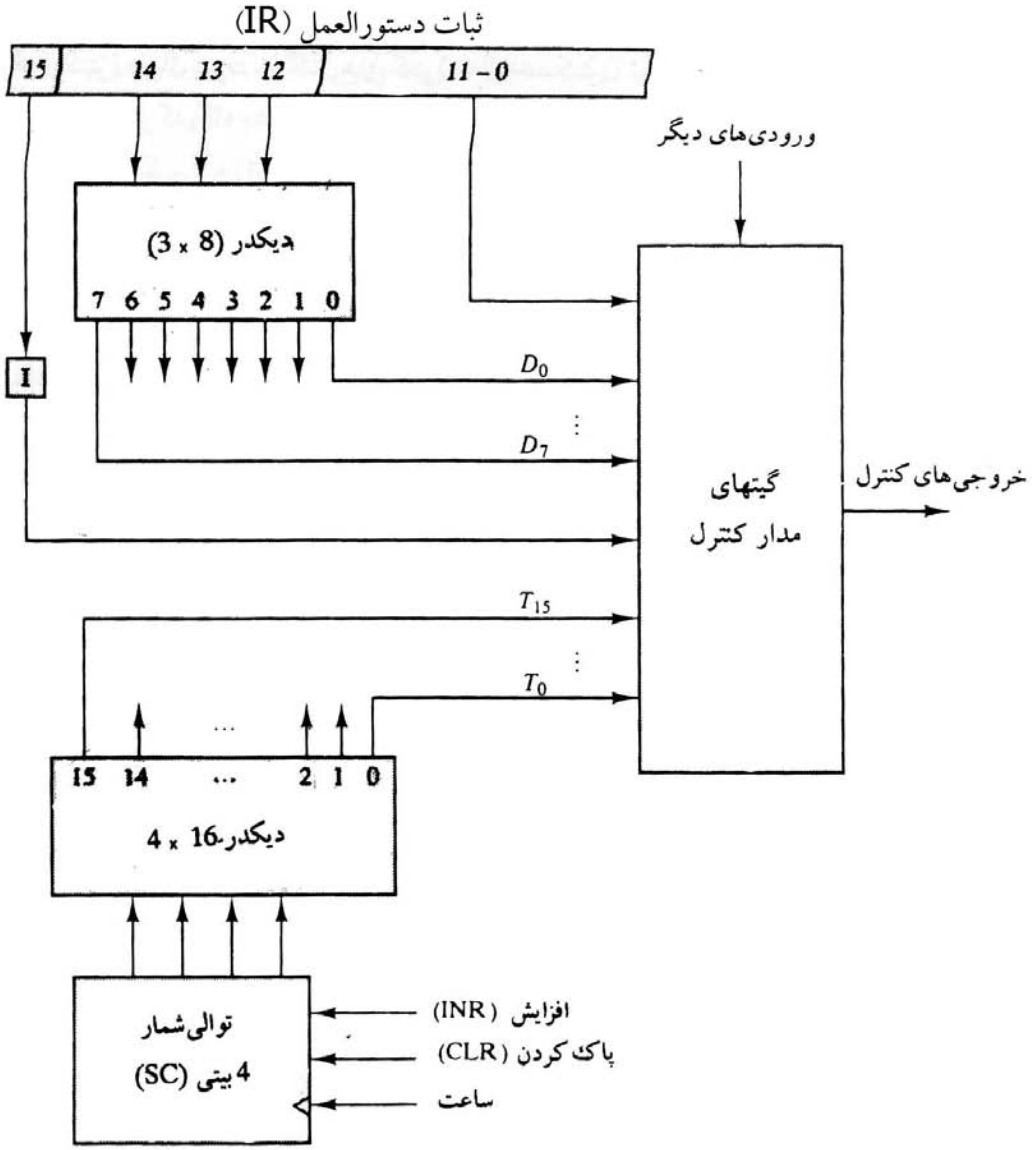
باشد واحد کنترل بیت مکان ۱۵ را بررسی مینماید. اگر این بیت 0 باشد، دستور از نوع ارجاع به ثبات است. اگر بیت مذکور 1 باشد، دستور از نوع ورودی – خروجی است. هنگامی که کد عمل 111 باشد بیت مکان ۱۵ با i مشخص می شود ولی بعنوان بیت روش آدرس دهی بکار نمیرود.

دستورات لیست شده در جدول۲-۳ از حداقل مجموعه ای تشکیل یافته که قادر است قابلیت های ذکر شده در بالا را فراهم آورد.

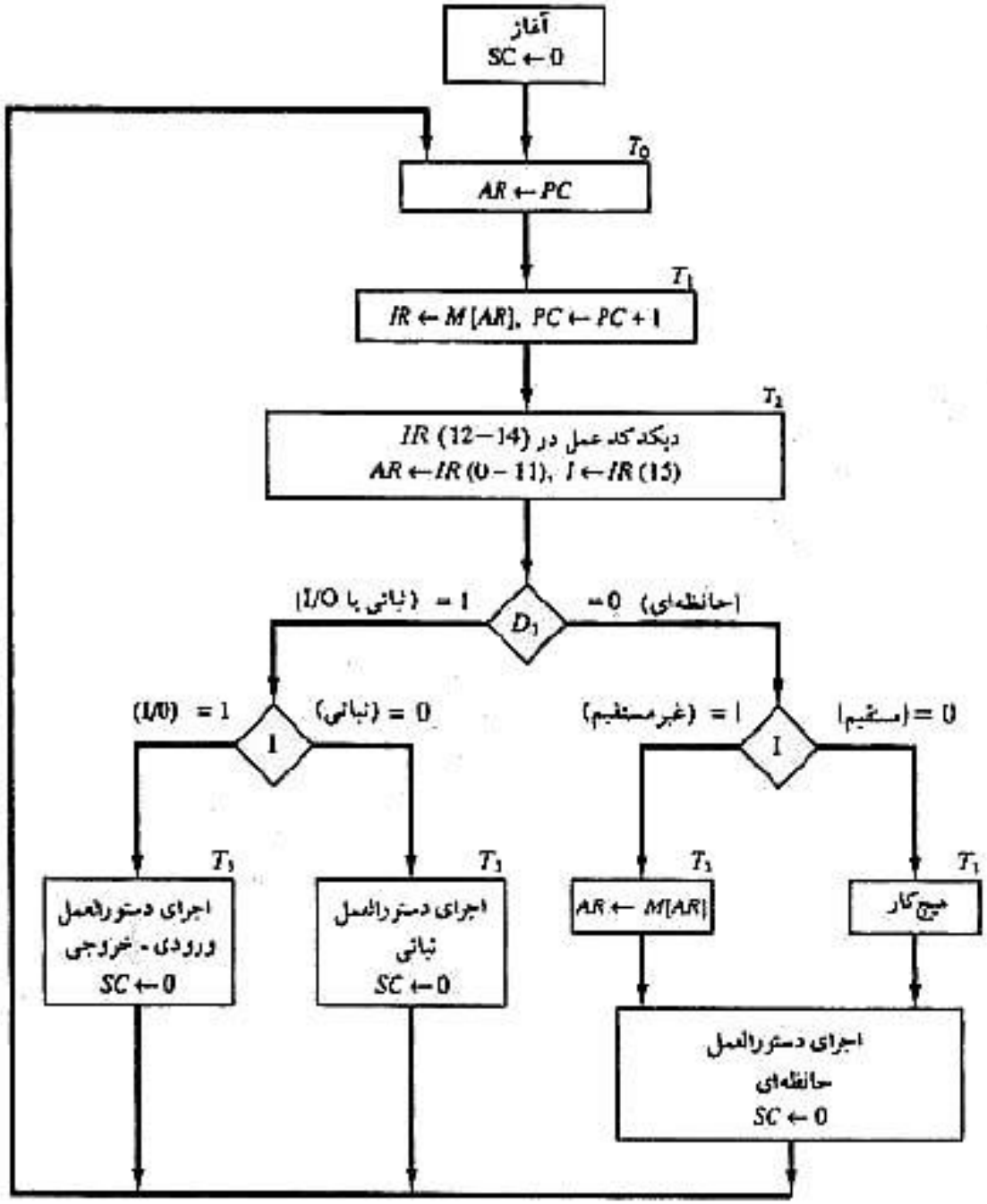
انتقال اطلاعات از حافظه به AC با دستور بار کردن AC (LDA) انجام میشود. ذخیره اطلاعات از AC در حافظه هم توسط دستور ذخیره کردن در AC (STA) صورت میگیرد. دستورالعملهای انشعاب BUN و BSA و ISZ همراه با چهار دستور گذر امکان کنترل برنامه و بررسی شرایط وضعیتی بوجود میآورند. دستورالعمل های ورودی INP و خروجی OUT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

جدول ۲-۳: دستورالعملهای کامپیوتر پایه

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



شکل۷-۳: واحد کنترل کامپیوتر



شکل۸-۳: فلوچارت سیکل دستورالعمل (آرایش اولیه)

AND با AC

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

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

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

تابع کنترل این دستور خروجی دیکدر D_0 را بکار میبرد زیرا این خروجی وقتی که عمل AND با کد 000 اجرا میشود فعال میگردد. دو سیگنال زمانبندی برای اجرای دستور فوق لازم است. لبه پالس مربوط به سیگنال T_4، عملوند را از حافظه به DR انتقال میدهد. لبه پالس بعدی یعنی T_5، نتیجه عمل AND بر روی AC و DR را به AC منتقل و SC را هم صفر میکند که این خود سبب واگذاری کنترل به سیگنال T_0 میگردد تا سیکل دستورالعمل جدیدی آغاز شود.

ADD با AC

این دستور محتوای حافظه مشخص شده توسط آدرس موثر را با مقدار AC جمع میکند. حاصل جمع به AC و رقم نقلی خروجی C_{out} به فلیپ فلاپ E (بیت گسترش انباره) منتقل میشود. ریزاعمال لازم برای این دستور عبارتست از:

$$D_1T_4 : DR \leftarrow M[AR]$$
$$D_1T_5 : AC \leftarrow AC + DR, E \leftarrow C_{out}, SC \leftarrow 0$$

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

خواهد بود و این خروجی رشته ریزعملهایی که واحد کنترل هنگام اجرای یک دستورالعمل حافظه ای دنبال میکند را تعیین مینماید.

LDA: بارکردن AC

این دستور محتوای حافظه ای که توسط آدرس موثرش مشخص شده را بهAC منتقل میکند.

$$D_2T_4 : DR \leftarrow M[AR]$$
$$D_2T_5 : AC \leftarrow DR, SC \leftarrow 0$$

با بازنگری مجدد شکل۸-۳ ملاحظه می شود که مسیر مستقیمی از گذرگاه به AC وجود ندارد. مدار جمع کننده و منطقی، اطلاعات را از DR دریافت میکند 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 ریزعمل حفظ کنیم.

STA: ذخیره کردن AC در حافظه

این دستور محتوای AC را در حافظه ای 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 ریزعمل انجام دهیم.

$$D_3T_4 : M[AR] \leftarrow AC, SC \leftarrow 0$$

BUN: انشعاب بدون شرط

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

$$D_4T_4 : PC \leftarrow AR, SC \leftarrow 0$$

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

BSA: انشعاب با ذخیره آدرس برگشت

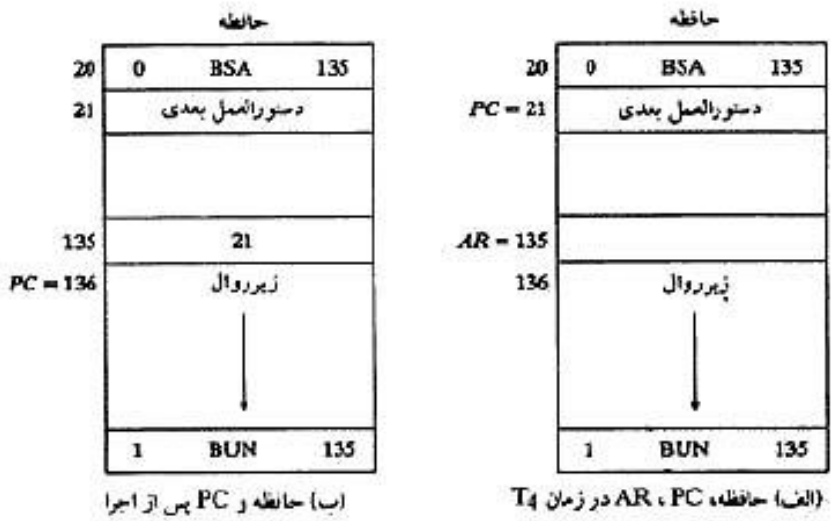
این دستور برای انشعاب به بخشی از برنامه که زیرروال یا رویه خوانده میشود مفید است. هروقت این دستور اجرا شود BSA آدرس دستور بعدی را که در PC موجود است در حافظهای که توسط آدرس موثر معین میشود ذخیره مینماید. سپس آدرس موثر بعلاوه ۱ به PC منتقل میشود تا بعنوان آدرس اولین دستور در زیرروال مورد استفاده قرار گیرد.

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

یک مثال عددی که نشان دهنده چگونگی بکار رفتن این دستور است در شکل ۹-۳ دیده میشود. فرض بر این است که دستور BSA در حافظه ۲۰ قرار گرفته باشد. بیت I برابر 0 و بخش آدرس دستور ۱۳۵ تصور میشود. پس از فازهای برداشت و دیکد، PC حاوی 21 خواهد بود که آدرس دستور بعدی در

برنامه است (این همان آدرس بازگشت است). AR حاوی آدرس موثر 135 است. این مطلب در قسمت (الف) شکل ملاحظه میگردد. دستورالعمل BSA عمل عددی زیر را اجرا می نماید.

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



شکل۹-۳

نتیجه این عمل در بخش (ب) از شکل مزبور 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 موثر را که در مکان 135 قراردارد 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 معمولا به آن فراخوانی زیرروال میگویید. دستور BUN در انتهای زیرروال عملی را که به آن بازگشت از زیرروال گویند انجام میدهد. در بسیاری از کامپیوترهای تجاری آدرس برگشت از زیرروال در یکی از ثبات های پردازشگر یا در بخشی از حافظه بنام پشته ذخیره میشود .

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

$$D_5T_4 : M[AR] \leftarrow PC, AR \leftarrow AR + 1$$
$$D_5T_5 : PC \leftarrow AR, SC \leftarrow 0$$

سیگنال زمانبندی T_4 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 تا لبه پالس بعدی تکمیل میگردد. در T_5 گذرگاه برای انتقال AR به PC مورد استفاده قرار میگیرد.

ISZ: افزایش و گذر اگر نتیجه صفر

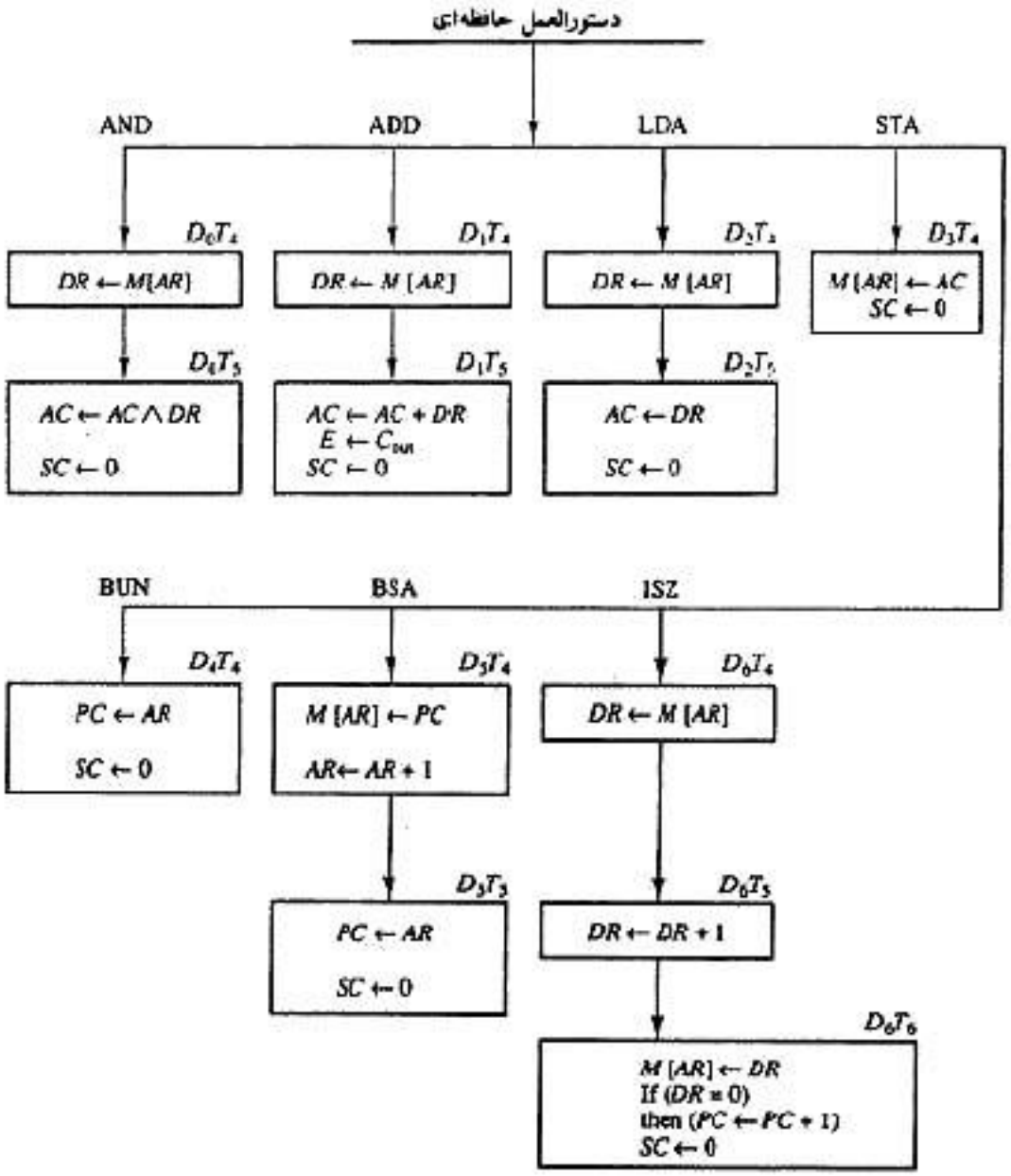
این دستور کلمهای که توسط آدرس موثر مشخص میشود را یک واحد افزایش میدهد و اگر مقدار افزایش یافته 0 شود، PC را یک واحد افزایش میدهد. برنامه نویس معمولا یک عدد منفی را (بفرم متمم ۲) در حافظه قرار میدهد. این عدد منفی مرتبا افزایش مییابد و نهایتا به صفر میرسد. در این لحظه PC یک واحد اضافه می شود تا از دستور بعدی در برنامه گذر 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 ریزعمل های زیر انجام می شود:

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

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

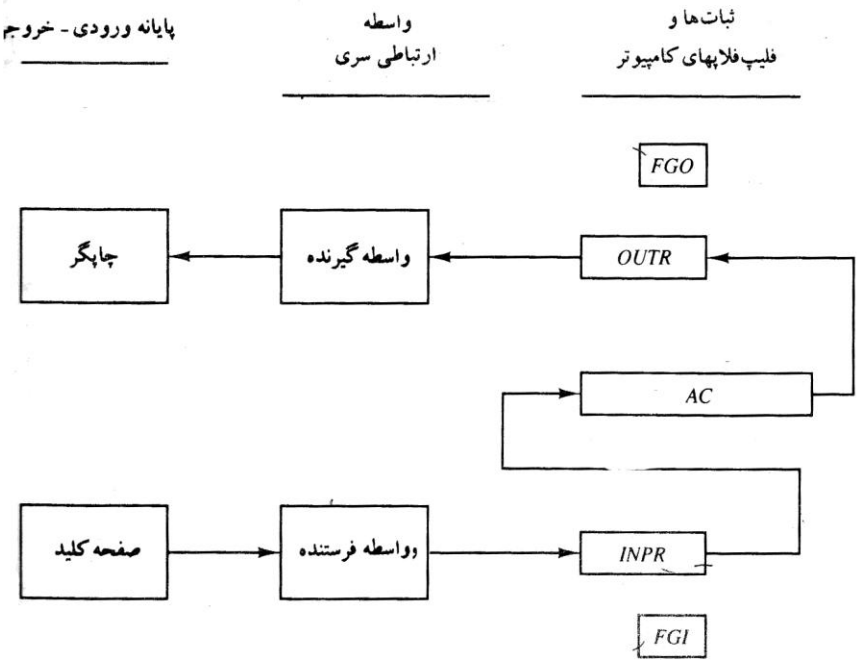
$D_6\ T_{46}: M[AR] \leftarrow DR, if(DR = 0) then\ (PC \leftarrow PC + 1), SC \leftarrow 0$



شکل۱۰-۳: فلوچارت دستورالعمل های حافظه ای

آرایش ورودی و خروجی

فرایند انتقال اطلاعات بقرار زیر است



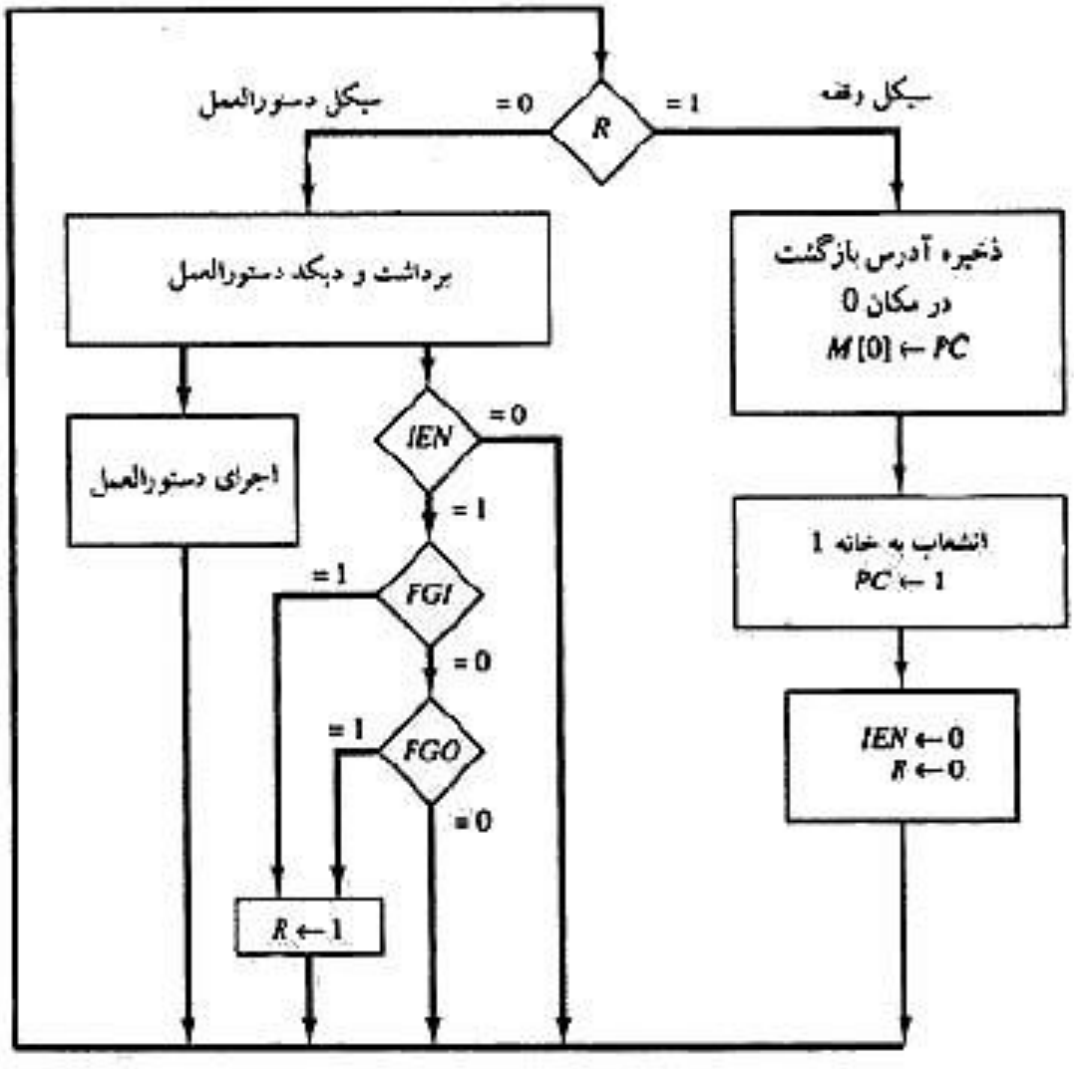
شکل۱۱-۳: آرایش ورودی –خروجی

جدول۳-۳: دستورالعملهای ورودی- خروجی

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

وقفه برنامه

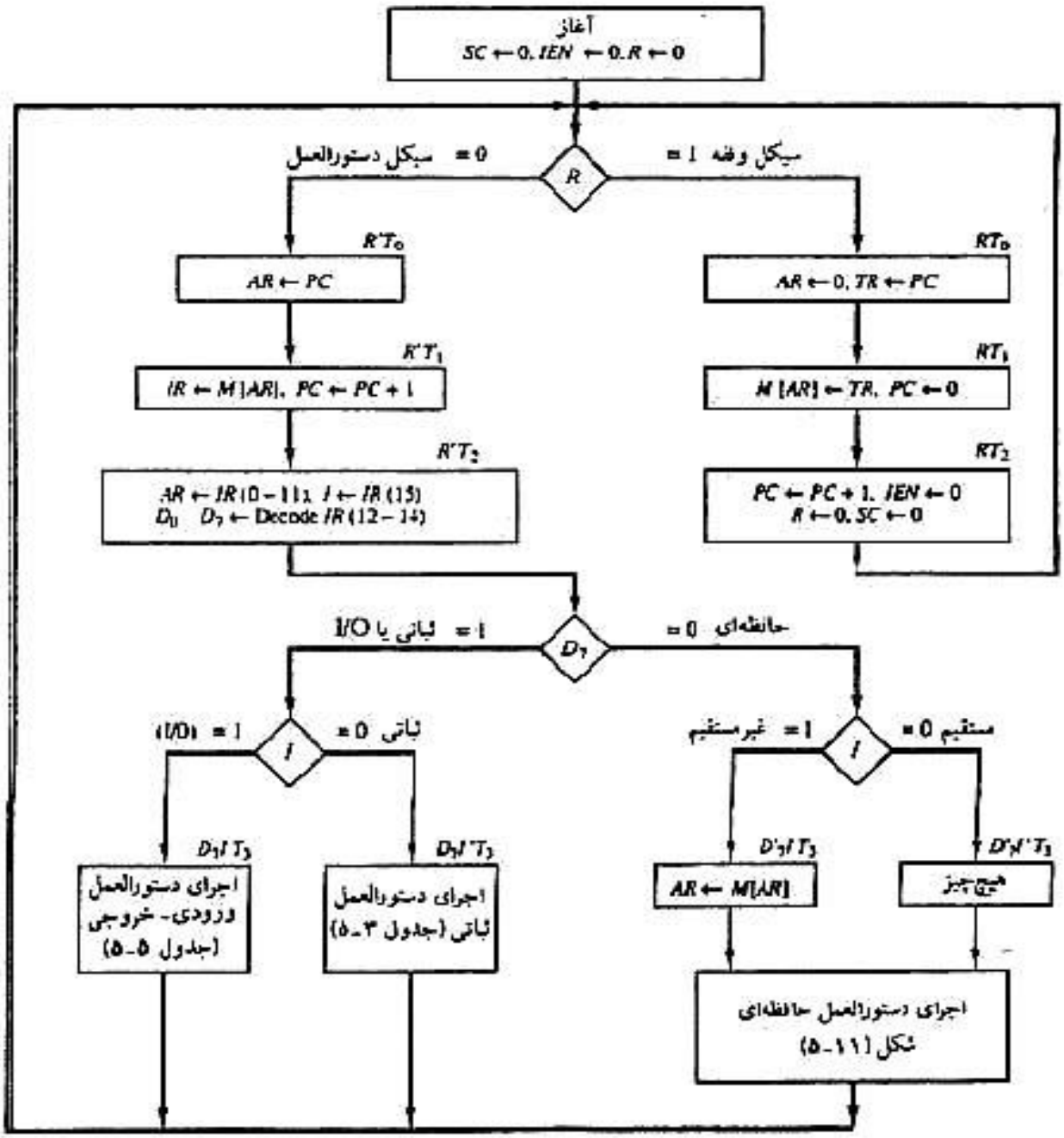
فرض کنید که وسیله ورودی –خروجی میتواند اطلاعات را حداکثر با میزان ۱۰ کاراکتر در ثانیه منتقل نماید. این سرعت معادل با 100,000 میکروثانیه برای هر کاراکتر میباشد. وقتی که کامپیوتر بیت پرچم را چک میکند و تصمیم میگیرد اطلاعات را انتقال ندهد دو دستورالعمل اجرا میشود. این بدان معنی است که در سرعت ماکزیمم، کامپیوتر بین هر دو انتقال 500000 بار پرچم را چک خواهد کرد. کامپیوتر مادامی که در حال چک کردن پرچم هاست بعوض انجام کار مفیدی وقت را تلف میکند



شکل فلوچارت سیکل وقفه

تشریح کامل کامپیوتر

فلوچارت نهایی سیکل دستورالعمل به همراه سیکل وقفه برای کامپیوتر پایه در شکل۱۲-۳ نشان داده شده است.



شکل ۱۲-۳: فلوچارت برای اعمال کامپیوتر

جدول ۴-۳: توابع کنترل و اعمال جزیی کامپیوتر پایه

برداشت	$R'T_0$: $R'T_1$:	$AR \leftarrow PC$ $IR \leftarrow M[AR], PC \leftarrow PC+1$
دیکد	$R'T_2$:	$D_0,\ldots,D_7 \leftarrow DecodeIR(12-14),$ $AR \leftarrow IR(0-11), I \leftarrow IR(15)$
غیرمستقیم	$D_7'T_3$:	$AR \leftarrow M[AR]$
وقفه:	$T_0T_1T_2(IEN)(FGI+FGO)$: RT_0 : RT_1 : RT_2 :	$R \leftarrow 1$ $AR \leftarrow 0, TR \leftarrow PC$ $M[AR] \leftarrow TR, PC \leftarrow 0$ $PC \leftarrow PC+1, IEN \leftarrow 0, R \leftarrow 0, SC \leftarrow 0$
حافظه ای: AND	D_0T_4 : D_0T_5 :	$DR \leftarrow M[AR]$ $AC \leftarrow AC \wedge DR, SC \leftarrow 0$
ADD	D_1T_4 : D_1T_5 :	$DR \leftarrow M[AR]$ $AC \leftarrow AC+DR, E \leftarrow C_{out}, SC \leftarrow 0$
LDA	D_2T_4 : D_2T_5 :	$DR \leftarrow M[AR]$ $AC \leftarrow DR, SC \leftarrow 0$
STA	D_3T_4 :	$M[AR] \leftarrow AC, SC \leftarrow 0$
BUN	D_4T_4 :	$PC \leftarrow AR, SC \leftarrow 0$
BSA	D_5T_4 : D_5T_5 :	$M[AR] \leftarrow PC, AR \leftarrow AR+1$ $PC \leftarrow AR, SC \leftarrow 0$
ISZ	D_6T_4 : D_6T_5 : D_6T_6 :	$DR \leftarrow M[AR]$ $DR \leftarrow DR+1$ $M[AR] \leftarrow DR, if\ (DR=0)then$ $(PC \leftarrow PC+1), SC \leftarrow 0$
ثباتی :	$D_7I'T_{3=r}$ (مشترک در همه دستورالعمل های ثباتی) $IR(i)=B_1(I=0.1.2\ldots11)$ r:	$SC \leftarrow 0$
CLA	rB_{11}:	$AC \leftarrow 0$
CLE	rB_{10} :	$E \leftarrow 0$
CMA	rB_9 :	$AC \leftarrow \overline{AC}$
CME	rB_8 :	$E \leftarrow \overline{E}$
CIR	rB_7 :	$AC \leftarrow shrAC, AC(15) \leftarrow E, E \leftarrow AC(0)$
CIL	rB_6 :	$AC \leftarrow shlAC, AC(0) \leftarrow E, E \leftarrow AC(15)$
INC	rB_5 :	$AC \leftarrow AC+1$
SPA	rB_4 :	$If\ (AC(15)=0)then(PC \leftarrow PC+1)$
SNA	rB_3:	$If\ (AC(15)=1)then(PC \leftarrow PC+1)$
SZA	rB_2 :	$If\ (AC=0)then(PC \leftarrow PC+1)$
SZE	rB_1 :	$If\ (E=0)then(PC \leftarrow PC+1)$
HLT	rB_0 :	$S \leftarrow 0$
ورودی – خروجی :	$D_7IT_3=P$ (مشترک در همه دستورالعمل های ورودی‌خروجی) $IR(i)=B_1(i=6,7,8,9,10,11)$ p :	$SC \leftarrow 0$
INP	pB_{11}:	$AC(0-7) \leftarrow INPR, FGI \leftarrow 0$
OUT	pB_{10}:	$OUTR \leftarrow AC(0-7), FGO \leftarrow 0$
SKI	pB_9 :	$If\ (FGI=1)then(PC \leftarrow PC+1)$
SKO	pB_8 :	$iF(FGO=1)then(PC \leftarrow PC+1)$
ION	pB_7 :	$IEN \leftarrow 1$
IOF	pB_6 :	$IEN \leftarrow 0$

فصل چهارم

برنامه نویسی کامپیوتر پایه

زبان ماشین

برنامه مجموعهای از دستورالعملها برای هدایت کامپیوتر در اجرای یک کار دادهپردازی مورد نظر است. زبان های مختلفی برای نوشتن یک برنامه کامپیوتری وجود دارد. ولی کامپیوتر میتواند تنها برنامه - هایی را اجرا کند که در داخل آن به شکل دودویی ارائه شده باشند. برنامههای نوشته شده به هر زبان دیگر، قبل از اجرا بوسیله کامپیوتر باید به دستورالعمل دودویی ترجمه شوند. برنامههای نوشته شده برای یک کامپیوتر میتوانند یکی از دسته های زیر باشند:

کد دودویی: این کد رشتهای از دستورالعملها و عملوندها به شکل دودویی است که شکل واقعی آنها را آنطور که در حافظه کامپیوتر ظاهر میشوند، نشان میدهد.

جدول ۱-۴: دستورالعمل های کامپیوتر

سمبل	کد شانزده شانزدهی	شرح
AND	8 یا0	AND کردن M با AC
ADD	9 یا1	جمع M با AC، نقلی به E
LDA	A یا2	بارکردن AC در M
STA	B یا3	ذخیرهAC در M
BUN	C یا4	انشعاب غیرشرطی به m
BSA	D یا5	ذخیره آدرس برگشت در m و انشعاب به m+1
ISZ	E یا6	افزایش M و گذر در صورت صفر بودن نتیجه
CLA	7800	صفر کردن AC
CLE	7400	صفر کردن E
CMA	7200	متمم کردنAC
CME	7100	متمم کردنE
CIR	7080	چرخش به راست E وAC
CIL	7040	چرخش به چپ E وac
INC	7020	افزایشAC
SPA	7010	گذر در صورت مثبت بودن AC
SNA	7008	گذر در صورت منفی بودن AC
SXA	7004	گذر در صورت صفر بودن AC
SZE	7002	گذر در صورت صفر بودن E
HLT	7001	توقف کامپیوتر
INP	F800	دریافت اطلاعات ورودیو 0 کردن پرچم
OUT	F400	ارسال اطلاعات خروجی و 0 کردن پرچم
SKI	F200	گذر در صورت 1 بودن پرچم ورودی
SKO	F100	گذر در صورت 1 بودن پرچم خروجی
ION	F080	فعال کردن وقفه
IOF	F040	غیرفعال کردن وقفه

کد هشت هشتی یا شانزده شانزدهی: این کد معادل ترجمه شده کد دودویی به هشت هشتی یا شانزده شانزدهی است.

کد سمبلیک: در این کد، کاربر از سمبلها (حروف، اعداد یا کاراکترهای خاص) برای بخش عملیاتی، بخش آدرس، و سایر قسمت های کد دستورالعمل استفاده میکند. هر دستورالعمل سمبلیک را میتوان به یک دستورالعمل کد شده با دودویی ترجمه کرد. این ترجمه توسط برنامه خاصی به نام اسمبلر انجام میشود. چون اسمبلر سمبل ها را ترجمه میکند، این نوع برنامه سمبلیک، برنامه بزبان اسمبلی خوانده میشود.

جدول ۲-۴: برنامه دودویی برای جمع کردن دو عدد

خانه حافظه	کد دستورالعمل			
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

برنامه جدول ۳-٤ یک برنامه به زبان اسمبلی برای جمع دو عدد است. سمبلORG 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.

جدول ۳-۴:برنامه به زبان اسمبلی برای جمع دو عدد

	ORG 0	مبداء برنامه خانه 0 است
	LDA A	بارکردن عملوند از A
	ADD B	جمع کردن عملوند در B
	STA C	ذخیره مجموع در C
	HLT	توقف کامپیوتر
A,	DEC 83	عملوند دهدهی
B,	DEC -23	عملوند دهدهی
C,	DEC 0	مجموع در خانه C ذخیره می شود
	END	پایان برنامه نمادین

زبان های برنامه نویسی سطح بالا: این برنامه ها که بزبان های خاصی نوشته میشوند بخاطر دریافت تاثیر رویه هایی است که به منظور حل مسئله خاصی بکار میروند و نه صرفا بخاطر تاثیر بر رفتار سختافزار کامپیوتر. مثالی از یک برنامه بزبان سطح بالا فرترن است. برنامه بصورت رشتهای از عبارات براساس نحوه تفکر فرد بهنگام حل یک مسئله نوشته میشود. با این وجود هر عبارت باید قبل از اجرا در کامپیوتر به رشتهای از دستورات دودویی تبدیل شود. برنامه ای که یک برنامه دیگر بزبان سطح بالا را به دودویی ترجمه میکند کامپایلر نامیده میشود.

جدول ۴-۴: برنامه زبان اسمبلی برای تفریق دو عدد

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

جدول۵-۴ : لیست برنامه ترجمه شده جدول ۴-۴

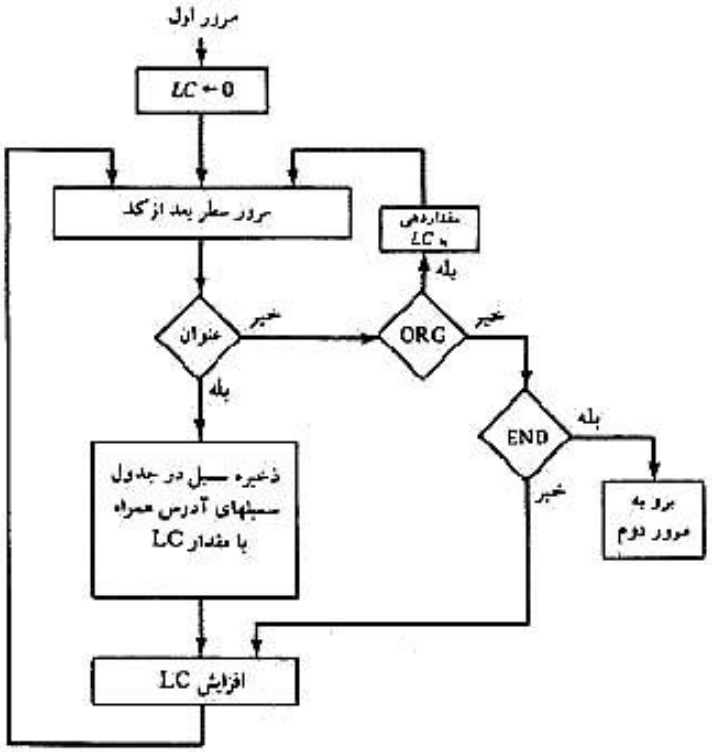
کد شانزده شانزدهی		
عمل	محتوا	برنامه سمبلیک
		ORG 100
100	2107	LDA SUB
101	7200	CMA
102	7020	INC
103	1106	ADD MIN
104	3108	STA DIF
105	7001	HLT
106	0053 MIN,	DEC 83
107	FFE9 SUB,	DEC -23
108	0000 DIF,	HEX 0
		END

مرور اول

یک اسمبلر دو مروری سراسر برنامه سمبلیک را دوبار مرور میکند. در مرور اول: جدولی را تولید میکند که مقدار معادل دودویی همه سمبل های آدرس تعریف شده توسط کاربر را نشان میدهد. ترجمه به دودویی در مرور دوم صورت میگیرد.

جدول ۶-۴: نمایش سطر کد PL3, LDA SUB I در کامپیوتر

کلمه حافظه	سمبل	کد شانزده شانزدهی	نمایش دودویی
1	P L	504C	0101 0000 0100 1100
2	3 ,	332C	0011 0011 0010 1100
3	L D	4C44	0100 1100 0100 0100
4	A	4120	0100 0001 0010 0000
5	S U	5355	0101 0011 0101 0101
6	B	4220	0100 0010 0010 0000
7	I CR	490D	0100 1001 0000 1101



شکل ۱-۴: فلوچارت مرور اول اسمبلر

مرور دوم

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

۱ -جدول شبه دستورالعملها

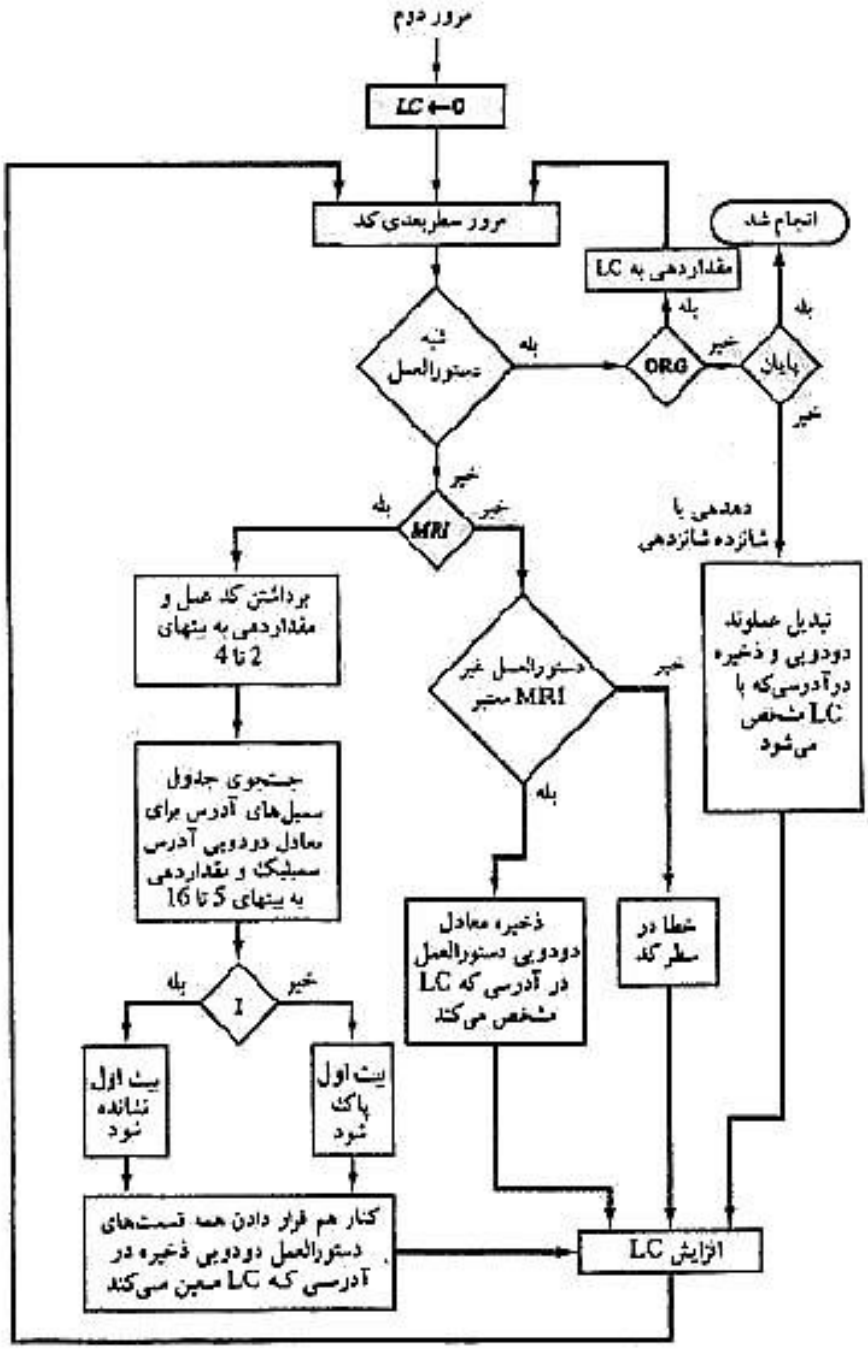
۲ -جدول MRI

۳ -جدول non-MRI

٤ -جدول سمبل آدرس

واردههای جدول شبه دستورالعملها چهار سمبل دارد: HEX, DEC, END, ORG. هر وارده هنگام برخورد با آن در برنامه، اسمبلر را به زیرروال شبه دستور مربوطه، هدایت میکند. جدول MRI حاوی هفت سمبل 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 می باشد.

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



شکل ۲-۴: فلوچارت مرور دوم اسمبلر

حلقه در برنامهنویسی

برنامه جمع ۱۰۰ عدد به صورت زیر نوشته میشود که در آن مفاهیم اشارهگر، شمارنده همراه با عمل آدرس دهی غیرمستقیم بکار برده شده است تا یک حلقه در برنامه بوجود آید. اشارهگر به آدرس عملوند جاری اشاره مینماید و شمارنده تعداد دفعات اجرای حلقه برنامه را میشمارد.

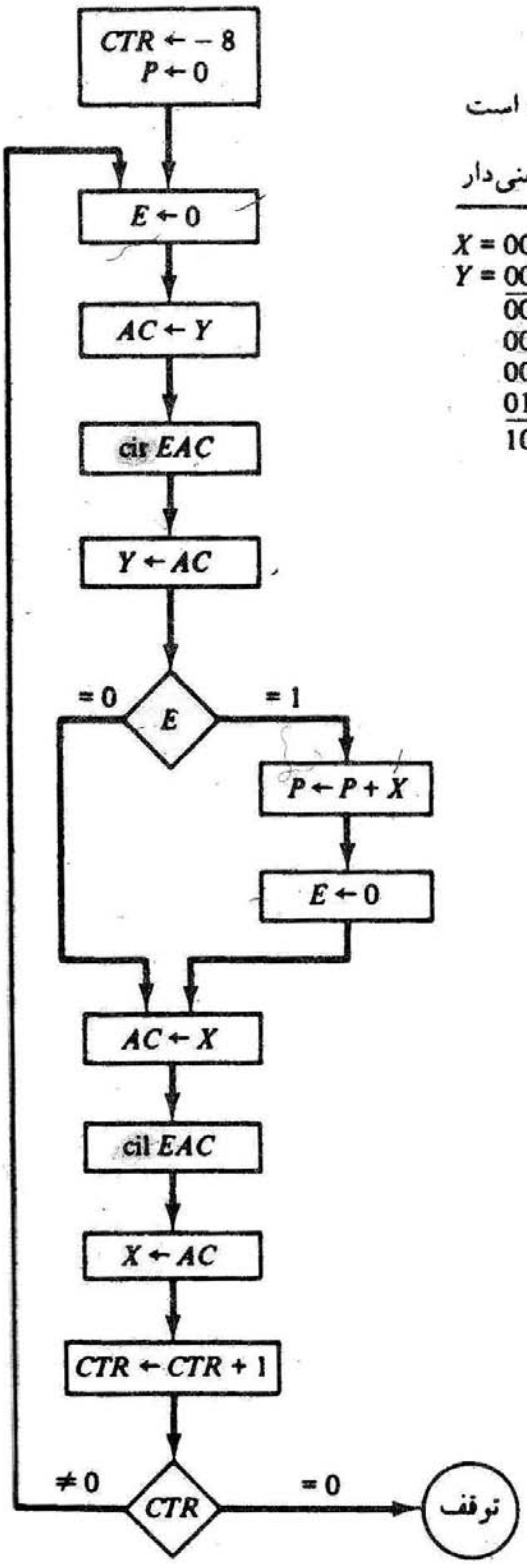
جدول ۷-۴: برنامه سمبلیک برای جمع ۱۰۰ عدد

سطر		
1	ORG 100	مبداء برنامه100 شانزده شانزدهی
2	LDA ADS	بار کردن آدرس ابتدای عملوندها
3	STA PTR	ذخیره در اشاره گر
4	LDA NBR	بارکردن100 منفی
5	STA CTR	ذخیره در شمارنده
6	CLA	پاک کردن انباشتگر
7	ADD PTR I	جمع یک عملوند باAC
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	HEX 150	آدرس ابتدای عملوندها
14	HEX 0	محل نگهداری شده برای اشاره گر
15	DEC -100	ثابتی که بعنوان مقدار اولیه به اشاره گر داده می شود
16	HEX 0	محل ذخیره شده برای شمارنده
17	HEX 0	محل ذخیره مجموع
18	ORG 150	مبداء عملوندها۱۵۰ در مبنای شانزده شانزدهی است
19	DEC 75	نخستین عملوند
.		
.		
.		
118	DEC 23	آخرین عملوند
119	END	انتهای برنامه نمادین

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 آن ها از ١٦ بیت تجاوز نمیکند. میتوان برنامه را اصلاح کرد تا علامت اعداد نیز در نظر گرفته شود و یا از اعداد ١٦ بیتی استفاده گردد. در این صورت حاصلضرب ممکن است تا31 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 **٣-٤** نشان داده شده است، فرایند ضرب از وارسی بیت های مضروب فیه Y و جمع مضروب X بتعداد 1 های موجود در Y تشکیل میشود، بشرطی که مقدار X از هر 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 حافظه را که با P نشان داده شده برای حفظ مجموعهای میانی در نظر میگیریم.



X حاوی مضروب است
Y حاوی مضروب‌فیه است
P محل تشکیل حاصلضرب است

مثال با چهار رقم معنی‌دار

		P
X =	0000 1111	
Y =	0000 1011	0000 0000
	0000 1111	0000 1111
	0001 1110	0010 1101
	0000 0000	0010 1101
	0111 1000	1010 0101
	1010 0101	

شکل ۳-۴: فلوچارت برنامه ضرب

جدول ۸-۴: برنامه ضرب دو عدد مثبت

	ORG 100	
LOP,	CLE	پاک کردن E
	LDA Y	بار کردن مضروب فیه
	CIR	انتقال بیت مضروب فیه به E
	STA Y	ذخیره مضروب فیه جابجا شده
	SZE	چک بیت، صفر است یا نه
	BUN ONE	بیت ۱ است، به ONE برو
ONE,	BUN ZRO	بیت 0 است، به ZRO برو
	LDA X	بارکردن مضروب
	ADD P	جمع یا حاصلضرب جزیی
	STA P	ذخیره حاصلضرب جزیی
ZRO,	CLE	پاک کردن E
	LDA X	بارکردن مضروب
	CIL	چرخش به چپ
	STA X	ذخیره مضروب جابجا شده
	ISZ CTR	افزایش شمارنده
	BUN LOP	شمارنده صفر نیست، تکرار حلقه
	HLY	شمارنده صفر است، توقف
CTR,	DEC -8	محل شمارنده
X,	HEX 000F	محل ذخیره مضروب
Y,	HEX 000B	محل ذخیره مضروب فیه
P,	HEX 0	محل تشکیل حاصلضرب
	END	

جمع با دقت مضاعف

هنگامی که دو عدد ۱۶ بیتی بی علامت در هم ضرب شوند، نتیجه حاصلضرب۳۲ بیتی خواهد بود که باید در دو کله حافظه ذخیره شود. هرگاه عددی در دو کلمه حافظه جای گیرد گوییم دقت مضاعف دارد. وقتی که یک حاصلضرب جزیی محاسبه میشود، لازم است یک عدد با دقت مضاعف، با مضروب شیفت یافته که خود عددی با دقت مضاعف است، جمع شود.

جدول ۹-۴: برنامه جمع دو عدد با دقت مضاعف

	LDA A L	بارکردن بخش کم ارزشتر A
	ADD BL	جمع بخش کم ارزشتر B، نقلی در E
	STA CL	ذخیره در بخش کم ارزشتر C
	CLA	پاک کردن AC
	CIL	چرخش برای آوردن نقلی به درون AC(0)
	ADD A H	جمع کردن بخش باارزشتر A و نقلی
	ADD BH	جمع کردن بخش پرارزشتر B
	STA H HLT	ذخیره در بخش پرارزشتر C
AL,	--	محل عملوندها
AH,	--	
BL,	--	
BH	--	
CL,	--	
CH,	--	

اعمال منطقی

هر شانزده عمل منطقی را می توان بصورت نرم افزاری پیاده سازی کرد زیرا هر تابع منطقی را می توان با AND و متممسازی پیاده سازی نمود. مثلا عملOR در دستورات کامپیوتر پایه وجود ندارد. اما با استفاده از تئوری دمورگان رابطه x+y=(x'+y')' را میتوان نتیجه گرفت. عبارت دوم فقط از اعمال AND و متمم سازی تشکیل شده است. برنامهای که عمل OR را روی دو عملوند منطقی A و B انجام میدهد بصورت زیر است:

LDA A	Load first operand A	اولین عملوند درA را بار کن
CMA	Complement to get $\overline{A}$	با متمم سازی$\overline{A}$ را بدست بیاور
ST A TMP	Store in a temporary location	حاصل را در یک مکان موقت ذخیره کن
LDA B	Load Second operand B	عملوند دوم را درB بارکن
CMA	Complement to get $\overline{B}$	با متمم سازی را بدست بیاور
AND TMP	AND with $\overline{A}$ to get $\overline{A} \wedge \overline{B}$	حاصل را با برای بدست آوردنAND کن
CMA	Complement again to get $A \wedge B$	حاصل را مجددا متمم کن تاAVB بدست آید

زیرروال ها

مکررا اتفاق افتاده است که یک قطعه از کد در بخشهای مختلفی از برنامه به طور تکراری نوشته شود. مسلماً به جای 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 آن ها بدفعات زیاد استفاده کرد زیرروال نام دارد. هربار که زیرروال در بخش اصلی برنامه به کار رود، انشعابی به ابتدای زیرروال صورت میگیرد. پس از اجرای زیرروال، انشعابی دیگر موجب بازگشت به برنامه اصلی میگردد.

جدول ۱۰-۴ برنامه ای برای نمایش طریقه استفاده از زیرروال ها

	ORG 100	برنامه اصلی
100	LDA X	باردهیX
101	BSA SH4	انشعاب به زیرروال
102	STA X	ذخیره عدد جابجا شده
103	LDA Y	بارکردنY
104	BSA SH4	انشعاب دوباره به زیرروال
105	STA Y	ذخیره عدد جابجاشده
106	HLT	
107	HEX 1234	
108	HEX 4321	
		زیرروال چهار بار جابجایی
109	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	صفر کردن(16-13)AC
10F	BUN SH4	بازگشت به برنامه اصلی
110	HEX FFF0	عملوند پوشش

برنامه زیر نمونه ای از اشتراک پارامترها را نمایش می دهد.

جدول ۱۱-۴: برنامه ای برای نمایش اشتراک پارامترها

	ORG 200	
200	LDA X	بارکردن عملوند اول در AC
201	BSA OR	انشعاب به زیرروال OR
202	HEX 3AF6	محل ذخیره عملوند دوم
203	STA Y	محل برگشت از زیرروال
204	HLT	
205	HEX 7B95	محل ذخیره عملوند اول
206	HEX 0	محل ذخیره نتیجه
207	HEX 0	زیرروال OR
208	CMA	متمم کردن عملوند اول
209	STA TMP	ذخیره در مکان موقت
20A	LDA OR I	بارکردن عملوند دوم
20B	CMA	متمم کردن عملوند دوم
20C	AND TMP	AND کردن متمم عملوند اول
20D	CMA	متمم مجدد برای یافتن OR
20E	ISZ OR	افزایش آدرس بازگشت
20F	BUN OR I	بازگشت به برنامه اصلی
210	HEX 0	محل ذخیره موقت
	END	

برنامه زیر زیرروالی برای انتقال بلاک داده ها را بیان می کند.

جدول ۱۲-۴: زیرروال انتقال بلاک داده ها

		برنامه اصلی
	BSA MVE	انشعاب به زیرروال
	HEX 100	آدرس شروع مبدأ داده ها
	HEX 200	آدرس شروع مقصد داده ها
	DEC -16	تعداد اقلامی که باید انتقال یابد
	HLT	
MVE,	HEX 0	زیرروال MVE
	LDA MVE I	بارکردن آدرس مبدأ
	STA PT1	ذخیره در اشاره گر اول
	ISZ MVE	افزایش آدرس بازگشت
	LDA MVE I	بارکردن آدرس مقصد
	STA PT2	ذخیره در اشاره گر دوم
	ISZ MVE	افزایش آدرس بازگشت
	LDA MVE I	بارکردن تعداد اقلام
	STA CTR	ذخیره در شمارنده
	ISX MVE	افزایش آدرس برگشت
LOP,	LDA PT1 I	بارکردن یک قلم از داده های 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	۱۶ بار تکرار
	BUN MVE I	بازگشت به برنامه اصلی
PT1,	-	
PT2,	-	
CTR,	-	

برنامه نویسی ورودی- خروجی

یک کاراکتر کد شده به دودویی هنگامی وارد کامپیوتر میشود که یک دستورالعمل INP (ورودی) اجرا شود. یک کاراکتر کد شده به دودویی وقتی به دستگاه خروجی منتقل میشود که یک دستورالعمل OUT اجرا گردد.

جدول ۱۳-۴: برنامه ورود و خروج یک کاراکتر

	: (الف) ورودی یک کاراکتر	
چک کردن پرچم ورود و خروج یک کاراکتر	SKI	CIF,
پرچم برابر 0 است، انشعاب برای وارسی مجدد	BUN CIF	
پرچم 1 است، دریافت کاراکتر	INP	
چاپ کاراکتر	OU	
ذخیره کاراکتر	STA CHR	
	HLT	
محل ذخیره کاراکتر	-	CHR,
	: (ب) خروجی یک کاراکتر	
بارکردن کاراکتر در AC	LDA CHR	
چک کردن پرچم خروجی	SKO	COF,
پرچم 0 است، انشعاب برای وارسی مجدد	BUN COF	
پرچم برابر 1 است، ارسال کاراکتر	OUT	
	HLT	
کاراکتر مورد نظر "W" است	HEX 0057	CHR,

دستکاری کاراکتر

یک کامپیوتر صرفاً یک ماشین حساب نیست بلکه دستکاری کننده سمبل ها نیز هست. کاراکترهای کد شده به دودویی که سمبل ها را می سازند می توانند به وسیله دستورالعمل های کامپیوتر برای انجام انواع کارهای داده پردازی دستکاری شوند. یکی از این کار ها فشرده کردن دو کاراکتر در یک کلمه است.

جدول ۱۴-۴: 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 ۱۵-۴: برنامه ذخیره کاراکترهای ورودی در یک بافر

	LDA ADS	بارکردن آدرس ابتدای بافر
	STA PTR	آغاز اشاره گر
LOP,	BSA IN2	پرش به زیرروال IN2 (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

عمل مقایسه بدین ترتیب صورت می گیرد که متمم ۲ یک کلمه بدست آمده و با کلمه دوم جمع حسابی میشود. 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 زیرروالی در یک برنامه جستجوی جدول مورد استفاده قرار داد.

جدول ۱۶-۴: برنامه مقایسه دو کلمه

	LDA WD1	بارکردن کلمه اول
	CMA	
	INC	تشکیل متمم 2
	ADD WD2	اضافه کردن کلمه دوم
	SZA	گذر اگر AC صفر باشد
	BUN UEQ	انشعاب به روال «اگر نامساوی»
	BUN EQL	انشعاب به روال «اگر مساوی»
WD1,	_	
WD2,	_	

برنامه وقفه

برنامه سرویس دهی به یک وقفه در برنامه ای به شکل زیر توضیح داده شده است.

جدول ۱۷-۴: برنامه سرویس دهی به یک وقفه

مکان			
0	ZRO,	_	محل ذخیره آدرس برگشت
1		BUN SRV	انشعاب به روال سرویس
100		CLA	قسمتی از برنامه در حال اجرا
101		ION	فعال کردن قابلیت وقفه
102		LDA X	
103		ADD Y	شروع وقفه
104		STA Z	برنامه بعد از وقفه به این مکان باز می گردد
.		.	
.		.	
.			روال سرویس دهی به وقفه
200	SRV,	STA SAC	ذخیره محتوای AC
		CIR	انتقال E به AC(1)
		STA SE	ذخیره محتوای E
		SKI	چک کردن پرچم ورودی
		BUN NXT	پرچم صفر است، پرچم بعدی چک شود
		INP	پرچ 1 است، دریافت کاراکتر
		OUT	چاپ کاراکتر
		STA PT1 I	ذخیره کاراکتر دربافر ورودی
		ISZ PT1	افزایش اشاره گر ورودی
	NXT,	SKO	چک کردن پرچ خروجی
		BUN EXT	پرچم صفر است، خروج
		LDA PT2 I	بارکردن کاراکتر از بافر خروجی
		OUT	ارسال کاراکتر
		ISZ PT2	افزایش اشاره گر خروجی
	EXT,	LDA SE	بازیابی مقدار AC(1)
		CIL	انتقال آن به E
		LDA SAC	بازیابی محتوای AC
		ION	فعال کردن وقفه
		BUN ZRO I	بازگشت به برنامه در حال اجرا
	SAC,	_	محل ذخیره AC
	SE,	_	محل ذخیرهE
	PT1,	_	اشاره گر میانگیر ورودی
	PT2,	_	اشاره گر میانگیر خروجی