۸-۲۹ دو عدد 8 بیتی A=01000001 و B=10000100 مفروضند.

الف- معادل دهدهی هر عدد را با فرض اینکه (۱) آنها بدون علامت هستند و (۲) آنها علامت دارند بدست آورید.

ب - دوعدد دودویی را جمع کنید و نتیجه را تفسیر کنید با این فرض که (۱) آنها بدون علامتند (۲) آنها علامت دارند.

ج- مقادیر بیت‌های 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 ۱۱-۸ لیست کنید.

۸-۳۰ برنامه‌ای در یک کامپیوتر دو عدد بدون علامت A و B را با انجام عمل A-B و تصحیح بیت‌های وضعیت مقایسه می‌کند. اگر A=01000001 و B=10000100 باشد،

الف- اختلاف را محاسبه و نتیجه دودویی را تفسیر کنید.

ب - مقادیر بیت‌های وضعیت C (قرض) و Z را معین کنید.

ج- دستورات انشعاب شرطی را از جدول ۱۱-۸ که اتفاق می‌افتند[1] لیست کنید

۸-۳۱ برنامه‌ای در یک کامپیوتر دوعدد علامت دار A و B را با انجام عمل A-B و تصحیح بیت‌های وضعیت مقایسه می‌کند. اگر A=01000001 و B = 10000100 باشند،

الف- اختلاف را محاسبه و نتیجه دودویی را تفسیر کنید

ب - مقادیر بیت‌های S و Z و V را معین کنید.

ج- دستورات انشعاب شرطی را از جدول ۱۱-۸ که اتفاق می‌افتند لیست کنید

۸-۳۲ محتویات بالاترین مکان حافظه یک پشته 5320 است. محتوای اشاره‌گر پشته SP نیز 3560 می‌باشد. یک دستور فراخوانی دو کلمه‌ای در آدرس 1120 حافظه قرار دارد و در آدرس 1121 هم میدان آدرس آن یعنی 6720 ذخیره شده است. محتوای PC، SP و بالاترین مکان پشته چیست؟

الف- قبل از برداشت دستور فراخوانی از حافظه

ب- پس از اجرای دستور فراخوانی

ج- پس از بازگشت از زیرروال

۸-۳۳ اختلافات عمده بین دستور انشعاب، فراخوانی و وقفه برنامه چیست؟

۸-۳۴ پنج مثال از وقفه خارجی و پنج مثال از وقفه داخلی را ارائه نمائید. فرق بین وقفه نرم افزاری و فراخوانی چیست؟

۸-۳۵ کامپیوتری با انتقال PC و PSW به یک وقفه پاسخ می‌دهد. سپس PSW جدیدی را از مکان IAD که توسط آدرس وقفه مشخص می‌شود می‌خواند. اولین آدرس برنامه سرویس وقفه از IAD + 1 خوانده می‌شود.

الف- رشته عملیات جزیی را برای سیکل وقفه لیست می‌کند.

1- True Condition

ب - رشته عملیات را برای بازگشت از زیرروال وقفه لیست کنید.

۸-۳۶ مثالهایی از کامپیوترهای دارای قالب دستورات متغیر عبارتند از IBM 370 و VAX11 و Intel 386. قالب دستورات یکی از آنها را با دستورات با طول ثابت RISCI مقایسه کنید.

۸-۳۷ سه کامپیوتر از دریچه‌های ثبات زیر استفاده می‌کنند. سایز دریچه‌ها و تعداد کل ثبات در هر ثبات چقدر است؟

	کامپیوتر ۱	کامپیوتر ۲	کامپیوتر ۳
ثبات‌های عام	10	8	16
ثبات‌های محلی	10	8	16
ثبات‌های مشترک	6	8	16
تعداد دریچه‌ها	8	4	16

۸-۳۸ مثالی از یک دستور کامپیوتر RISCI که اعمال زیر را انجام دهید ارائه کنید.

الف - یک ثبات را کاهش دهد.

ب - یک ثبات را مکمل کند.

ج- علامت محتوای ثبات را معکوس می‌کند

د- یک ثبات را 0 کند.

هـ- یک عدد علامت‌دار را به 4 تقسیم کند.

و- هیچ کاری انجام ندهد.

۸-۳۹ دستوری از RISCI را به زبان اسمبلی بنویسید که سبب پرش به آدرس 3200 گردد بشرطی که بیت وضعیت Z=1 باشد.

الف- با استفاده از روش بلافصل

ب - با استفاده از روش نسبی (PC = 3400 فرض شود)

۹ پردازش خط لوله‌ای و برداری

۹-۱ 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

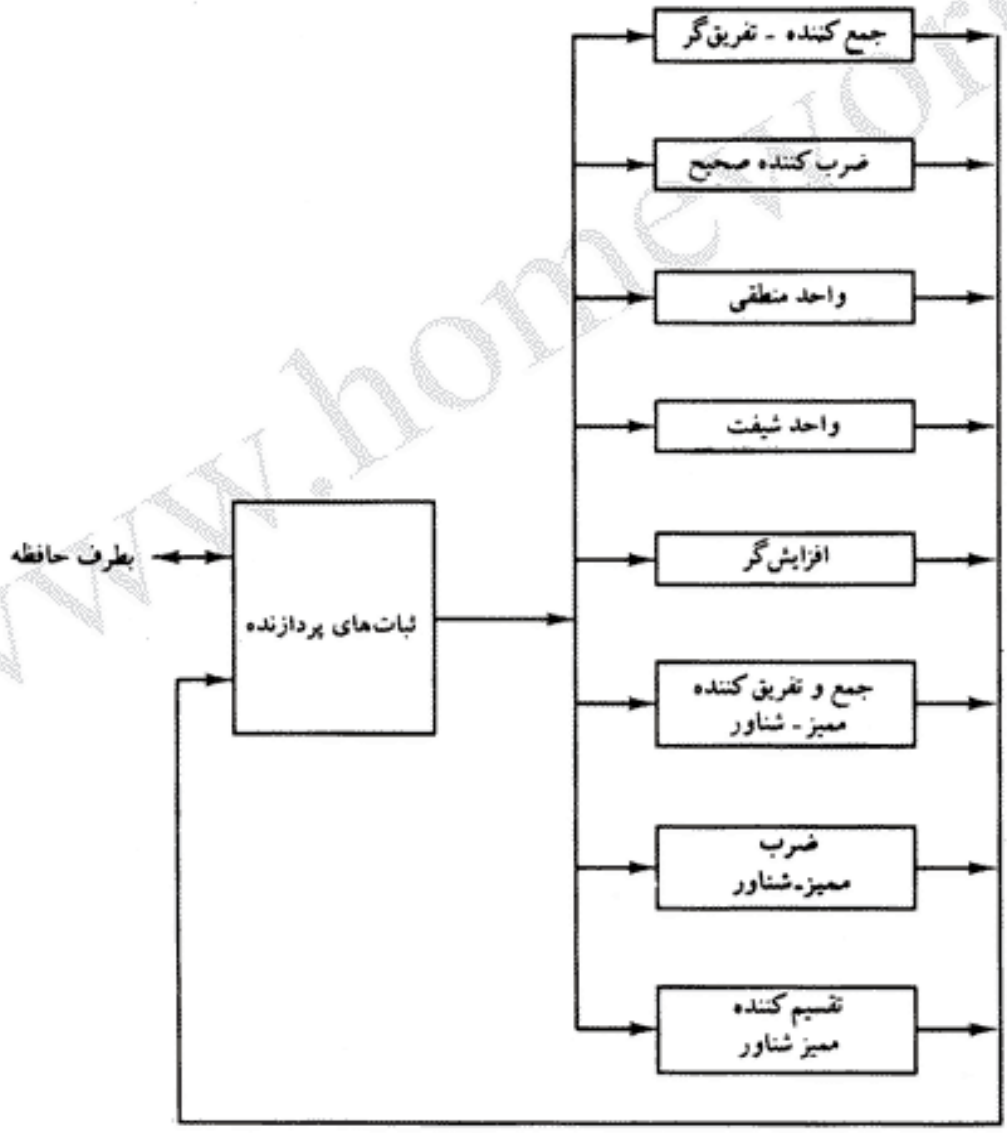
پردازش موازی به معنی بکارگیری تکنیک‌های متنوعی در پردازش همزمان داده‌ها است که به منظور افزایش سرعت محاسبات سیستم‌های کامپیوتری مورد استفاده قرار گرفته‌اند. یک سیستم پردازش موازی به جای پردازش متوالی دستورات، قادر است پردازش همزمان داده‌ها را برای رسیدن به سرعت پردازش بیشتر انجام دهد. مثلاً، ضمن اجرای دستوری در ALU ، دستور بعدی قابل خواندن از حافظه است. سیستم ممکن است دارای دو یا چند ALU باشد و لذا قادر خواهد بود دو یا چند دستور را بطور همزمان اجرا نماید. بعلاوه سیستم ممکن است دو یا چند پردازنده داشته باشد که همزمان با هم عمل نمایند. در هر صورت همانطور که اشاره شد هدف از پردازش موازی بالا بردن سرعت پردازش کامپیوتر و افزایش دفعات پردازش در طول بازه معینی از زمان است. با بکارگیری تکنیک پردازش موازی حجم سخت افزاری افزایش می‌یابد و همراه با آن قیمت سیستم نیز بالا می‌رود. با این وجود توسعه تکنولوژی، بهای سخت‌افزاری را تا حدی پائین آورده است که در پردازش موازی کاملاً محسوس است.

پردازش موازی را می‌توان از زوایای مختلفی بررسی کرد. ما در پائین ترین 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[1]

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 وشیفت قابل تفکیک به سه واحد بوده و عملوندها تحت نظر واحد کنترل به هر واحد رانده می‌شوند.

شکل ۱-۹ 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 چهارگوشه‌های دیاگرام مشخص شده‌اند. جمع کننده و ضرب کننده عدد صحیح روی اعداد صحیح عمل می‌کنند. اعمال ممیز شناور بین سه مدار که موازی با هم عمل می‌کنند تقسیم شده‌اند. اعمال منطقی، شیفت و افزایش در روی داده‌های مختلف بطور



شکل ۱-۹ پردازنده با واحدهای عملیاتی چندگانه

همزمان عمل می‌نمایند. کلیه واحدها از یکدیگر مستقل بوده و لذا یک عدد، شیفت و دیگری افزایش داده می‌شود. سازمان چند تابعی معمولاً به یک واحد کنترل پیچیده نیاز دارد تا کلیه فعالیت‌ها را در میان قسمت‌های مختلف هماهنگ نماید.

روش‌های مختلف برای طبقه‌بندی پردازش موازی وجود دارد. می‌توان از دیدگاه سازمان داخلی یا از نظر اتصالات درونی بین پردازنده‌ها و یا از نظر جریان اطلاعات درون سیستم آنها را طبقه‌بندی کرد. یکی از دسته‌بندی‌ها توسط M. y. Flynn پیشنهاد گردید که طی آن سازمان یک کامپیوتر را با توجه به تعداد دستورات و داده‌های دستکاری شده در یک زمان مورد بررسی قرار می‌دهد. روال معمول یک کامپیوتر برداشت[1] دستورات از حافظه و اجرای آن‌ها در پردازنده است. دستورات خوانده شده متوالی از حافظه رشته دستورات[2] را تشکیل می‌دهند. داده‌هایی نیز که رشته اعمال روی آنها انجام می‌شود رشته داده‌ها را تشکیل می‌دهند. پردازش موازی ممکن است روی رشته دستورات یا روی رشته داده‌ها یا هر دو باشد. دسته‌بندی Flynn کامپیوتر را به چهار گروه مهم مطابق زیر تقسیم می‌کند:

رشته تک دستوری، رشته تک داده‌ای[3] (SISD)
رشته تک دستوری، رشته چند داده‌ای[4] (SIMD)
رشته چند دستوری، رشته تک داده‌ای[5] (MISD)
رشته چند دستوری، رشته چند داده‌ای[6] (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 می‌شوند و سیستم ممکن است توانایی پردازش موازی داخلی داشته و یا نداشته باشد. در این حالت پردازش موازی می‌تواند توسط واحدهای چندتابعی یا پردازش خط لوله صورت گیرد.

SIMD سازمانی را نشان می‌دهد که دارای چند واحد پردازش تحت یک واحد کنترل مشترک است. همه پردازنده‌ها دستورالعمل واحدی را توسط واحد کنترل دریافت می‌کنند ولی روی داده‌های مختلفی عمل می‌نمایند. واحد مشترک حافظه باید از چند ماژول[7] (بخش) تشکیل شده باشد تا در یک زمان بتواند با تمام پردازنده‌ها ارتباط برقرار کند. MISD فقط از نظر تئوری حائز اهمیت است چون عملاً هیچ سیستمی با چنین سازمانی ساخته نشده است. سازمان MIMD به کامپیوترهایی اختصاص دارد که توان پردازش چندین برنامه را در یک زمان دارند. اغلب سیستم‌های چند پردازنده‌ای و سیستم‌های چند

1- Fetch
2- Instruction Stream
3- Single Instruction Stream Single Data Stream
4- Single Instruction Stream Multiple Data Stream
5-Multiple Instruction Stream Single Data Stream
6-Multiple Instruction Stream Multiple Data Stream
7- Module

کامپیوتری در این دسته طبقه‌بندی می‌شوند.

دسته‌بندی Flynn بستگی به اختلافات بین عمل واحد کنترل و واحد پردازش داده دارد. این دسته‌بندی بیشتر به مشخصات رفتاری سیستم کامپیوتر می‌پردازد تا به اتصالات درونی آن. یک نوع پردازش موازی که منطبق بر تقسیم‌بندی Flynn نیست روش خط لوله[1] است. دو نمونه موجود از این طبقه‌بندی یکی پردازنده‌های آرایه‌ای[2] SIMD بحث شده در بخش ۷-۹ و دیگری چند پردازنده MIMD ارائه شده در فصل ۱۳ است.

در این فصل ما پردازش موازی را تحت 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[3] محاسبه‌ها یا فازهای سیکل دستورالعمل بطور هم پوش[4] (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 اتلاق 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 ساده ترین راه در شناخت یک ساختار خط لوله این باشد که تصور کنیم هر قطعه از یک ثبات ورودی و بدنبال آن یک مدار ترکیبی ساخته شده است. ثبات داده را نگهمیدارد و مدار ترکیبی جزءعملی را در ارتباط با قطعه انجام می‌دهد. خروجی مدار ترکیبی در یک قطعه مفروض به ثبات ورودی قطعه دیگر تحویل 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.

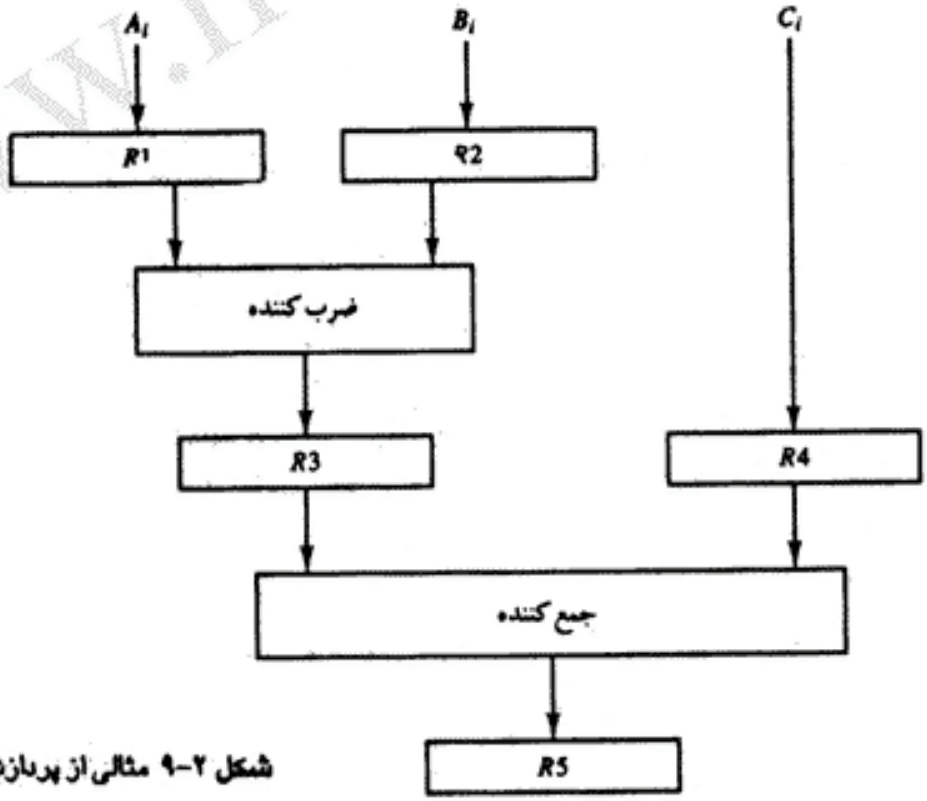
سازمان خط لوله توسط مثال ساده‌ای بیان می‌گردد. فرض کنید که می‌خواهیم یک عمل ترکیبی ضرب و جمع را با رشته‌ای از اعداد انجام دهیم.

$$A_i * B_i + C_i \qquad \text{for } i = 1, 2, 3, \ldots, 7$$

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 ۲-۹ 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:

دریافت A_i و B_i	$R1 \leftarrow A_i \quad$ و $\quad R2 \leftarrow B_i$
ضرب و دریافت C_i	$R3 \leftarrow R1 * R2 \quad$ و $\quad R4 \leftarrow C_i$
جمع C_i با حاصلضرب	$R5 \leftarrow R3 + R4$

پنج ثبات بازاء هر پالس با داده‌های جدیدی بار می‌شوند. اثر هر پالس در جدول ۱-۹ دیده می‌شود. اولین پالس ساعت A1 و B1 را به داخل R1 و R2 انتقال می‌دهند. پالس ساعت دوم حاصلضرب R1 در R2 را به R3 و C_i را به R4 منتقل می‌کند. همان پالس ساعت A2 و B2 را به R1 و R2 انتقال می‌دهد. سومین



شکل ۲-۹ مثالی از پردازش لوله‌ای

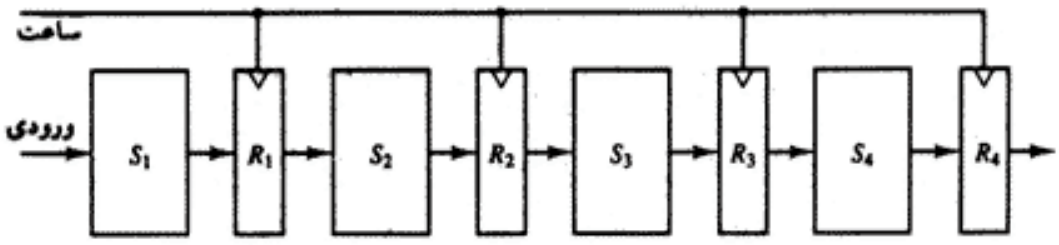
جدول ۱-۹ محتویات ثبات در مثال خط لوله

شماره پالس ساعت	قطعه 1		قطعه 2		قطعه 3
	R1	R2	R3	R4	R5
1	A_1	B_1	—	—	—
2	A_2	B_2	$A_1 * B_1$	C_1	—
3	A_3	B_3	$A_2 * B_2$	C_2	$A_1 * B_1 + C_1$
4	A_4	B_4	$A_3 * B_3$	C_3	$A_2 * B_2 + C_2$
5	A_5	B_5	$A_4 * B_4$	C_4	$A_3 * B_3 + C_3$
6	A_6	B_6	$A_5 * B_5$	C_5	$A_4 * B_4 + C_4$
7	A_7	B_7	$A_6 * B_6$	C_6	$A_5 * B_5 + C_5$
8	—	—	$A_7 * B_7$	C_7	$A_6 * B_6 + C_6$
9	—	—	—	—	$A_7 * B_7 + C_7$

پالس ساعت بطور همزمان بر روی هر سه قطعه عمل می‌کند. بدین ترتیب که A3 و B3 را در R1 و R2 می‌گذارد، حاصلضرب R2 و R1 را به R3 منتقل و C2 را در R4 و بعلاوه حاصل‌جمع R3 و R4 را در R5 قرار می‌دهد. بنابراین برای پرکردن 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 مادامی که داده‌های ورودی جدید بداخل سیستم جریان داشته باشند ادامه دارد. وقتی دیگر داده‌ای وجود نداشت، اعمال پالس باید تا خروج آخرین خروجی از خط لوله ادامه یابد.

بررسی‌های کلی

هر عملی که بتواند به رشته‌ای از 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 ۳-۹ نشان داده شده است. عملوندها از تمام قطعه‌ها با ترتیب ثابتی عبور می‌کنند. هر قطعه از یک مدار ترکیبی S_i تشکیل شده که جزءعمل خاصی را روی جریان داده درون لوله انجام می‌دهد.



شکل ۳-۹ مثالی از پردازش لوله‌ای

قطعات توسط ثبات‌های R_i که نتایج میانی قطعات را نگه‌میدارند از هم جدا شده‌اند. اطلاعات بین قطعات توسط پالس ساعت مشترکی که به تمام ثبات‌ها بطور همزمان اعمال می‌شود جریان می‌یابد. ما یک تکلیف[1] را به عنوان کل یک عمل که با طی شدن تمامی قطعات خط لوله انجام می‌شود تعریف می‌کنیم.

رفتار خط لوله را می‌توان توسط یک نمودار فاصله- زمان[2] تشریح کرد. این نمودار استفاده از قطعه را بصورت تابعی از زمان نشان می‌دهد. نمودار فاصله- زمان یک خط لوله چهارقطعه‌ای در شکل ۴-۹ دیده می‌شود. محور افقی نمودار زمان را برحسب پالس‌های ساعت و محور عمودی شماره قطعه را در آن نشان می‌دهد. نمودار، شش تکلیف T_1 الی T_6 را نشان می‌دهد که در چهار قطعه اجرا شده است.

در ابتدا تکلیف T_1 توسط قطعه 1 انجام می‌شود. پس از پالس ساعت اول، قطعه 2 مشغول اجرای T_1 است، و در همان زمان قطعه 1، T_2 را انجام می‌دهد. 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 می‌کند. تعداد قطعات در سیستم اهمیتی ندارد، به محض اینکه خط لوله پرشود در هر پالس ساعت یک خروجی بدست خواهد آمد.

حال خط لوله‌ای متشکل از k قطعه را در نظر بگیرید که در آن پالس ساعت t_p برای انجام n تکلیف بکار رفته است. اولین تکلیف T_1 زمانی برابر با kt_p را نیاز دارد تا پایان یابد زیرا در لوله تعداد k قطعه وجود دارد. $(n-1)$ تکلیف باقیمانده در لوله با سرعت یک تکلیف در هر پالس ساعت خارج می‌شوند و پس از زمانی برابر با $(n-1)t_p$ کلیه تکلیف ها پایان می‌یابند. بنابراین برای تکمیل n تکلیف با k قطعه در لوله، $k+(n-1)$ پالس ساعت لازم است. بعنوان مثال دیاگرام ۴-۹ چهار قطعه و شش تکلیف را نشان می‌دهد. می‌بینید 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 لوله‌ای با رابطه زیر تعریف می‌شود.

$$S = \frac{nt_n}{(K + n - 1)\, t_p}$$

قطعه:	1	2	3	4	5	6	7	8	9	چرخه‌های ساعت →
1	T_1	T_2	T_3	T_4	T_5	T_6				
2		T_1	T_2	T_3	T_4	T_5	T_6			
3			T_1	T_2	T_3	T_4	T_5	T_6		
4				T_1	T_2	T_3	T_4	T_5	T_6	

شکل ۴-۹ دیاگرام فاصله - زمان برای خط لوله

1- Task

2- Space – Time diagram

هرچه تعداد تکلیف ها اضافه شوند، n از 1 − k بیشتر می‌شود و 1− n + k بسمت n میل می‌کند. تحت این شرط، رابطه فوق برابر است با

$$S = \frac{t_n}{t_p}$$

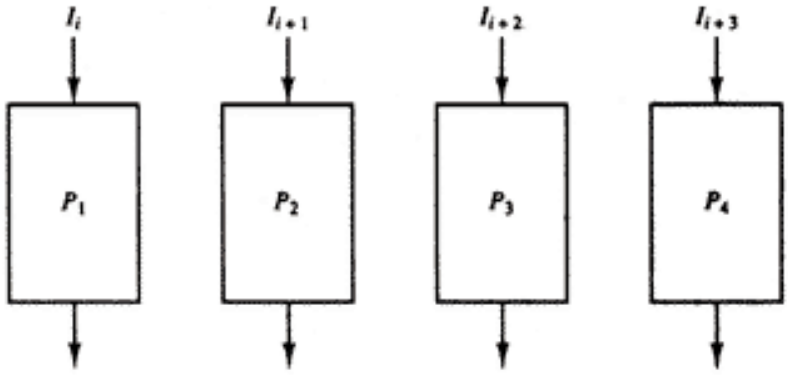
اگر فرض کنیم که زمان لازم برای غیرخط لوله همان زمان در خط لوله باشد داریم $t_n = kt_p$. لذا رابطه نسبت سرعت برابر خواهد بود با:

$$S = \frac{kt_p}{t_p} = k$$

رابطه فوق نشان می‌دهد که حداکثر نسبت سرعت تئوری در خط لوله می‌تواند 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=20ns باشد. اگر خط لوله دارای k=4 قطعه باشد و کلاً 100 تکلیف پردازش گردد، سیستم خط لوله زمانی برابر $t_p = (4 + 99)\times20=2060ns$ $(k + n - 1)$ برای تکمیل پردازش نیاز دارد. با این فرض‌که $t_n=kt_p=4\times20=80ns$ باشد یک سیستم غیرخط لوله‌ای زمانی برابر با $nkt_p=100\times80=8000ns$ برای تکمیل 100 تکلیف نیاز خواهد داشت.نسبت سرعت برابرست با 8000 / 2060=3.88 . با افزایش تعداد تکلیف ها، افزایش سرعت بسمت 4 میل می‌کند که با تعداد قطعات در خط لوله برابرست. اگر $t_p = 60ns$ در نظر گرفته شود، نسبت افزایش سرعت 3 = 20 / 60 خواهد شد.

برای تأکید بر مزیت سرعت تئوری یک پردازش خط لوله توسط یک واحد چند تابعی، لازم است که k واحد مشابه را ساخت تا به موازات عمل کنند. بدین ترتیب که می توان انتظار داشت تا k قطعه خط لوله، همانند k مدار همسان غیر خط لوله‌ای تحت شرایط عملکرد یکسان کار کنند. این مطلب در شکل ۵-۹ نشان داده شده است که در آن چهار مدار یکسان به موازات بسته شده‌اند. هر مدار P معادل با یک مدار خط لوله کار می‌کند. در عوض دریافت داده‌های متوالی، همچون مدار خط لوله، مدارات



شکل ۵-۹ واحدهای چند تابعی بصورت موازی

موازی چهار داده ورودی همزمان را دریافت می‌کنند و چهار کار را در یک زمان انجام می‌دهند. از نظر سرعت این عملکرد معادل با عملکرد یک خط لوله چهار قطعه‌ای است. توجه داشته باشید چهار مدار شکل ۵-۹ یک سازمان SIMD را تشکیل می‌دهند چون دستورات یکسانی برای عمل روی داده‌های مختلف بطور موازی عمل می‌کنند.

دلایل متعددی برای علت عمل نکردن خط لوله در وضعیت حداکثر سرعت تئوری‌اش وجود دارد. قطعات مختلف ممکن است دارای زمان‌های اجرای مختلفی باشند. سیکل ساعت باید با تأخیر زمانی قطعه‌ای که حداکثر تأخیر در انتشار را دارد برابر باشد. در نتیجه سایر قطعات مجبورند وقت خود را تا پالس ساعت بعدی بیهوده تلف کنند. بعلاوه صحیح نیست تصور کنیم که خط لوله دارای زمان تأخیر مشابه با مدار غیر خط لوله است. بسیاری از ثبات‌های واسطه در مدار تک واحدی که مداری ترکیبی است لازم نیستند. با این وجود تکنیک خط لوله امکان پردازش سریعتری را در مقایسه با یک پردازش سری فراهم می‌آورد، حتی اگر سرعت ماکزیمم تئوری هرگز بدست نیاید.

در طراحی کامپیوتر دو زمینه برای کاربرد خط لوله وجود دارد. خط لوله حسابی اعمال حسابی را بصورت جزء عمل‌ها[1] برای اجرا در قطعات خط لوله تقسیم می‌کند. خط لوله دستورالعمل روی رشته‌ای از دستورات عمل می‌کند و ضمن آن فازهای برداشت، دیکد و اجرای سیکل دستورالعمل را تواماً اجرا می‌نمایند. دو نوع خط لوله فوق در زیر توضیح داده شده‌اند:

۹-۳ خط لوله حسابی

خط لوله حسابی معمولاً در کامپیوترهای بسیار سریع یافت می‌شوند و برای پیاده‌سازی اعمال ممیز شناور، ضرب اعداد با ممیز ثابت و سایر محاسباتی از این قبیل که در مسائل علمی یافت می‌شوند بکار می‌روند. یک ضرب کننده خط لوله‌ای در اصل یک ضرب‌کننده 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 جزءعمل‌ها طبق آنچه در بخش ۵-۱۰ آمد تفکیک می‌گردد. در این جا مثالی از یک خط لوله برای جمع و تفریق شناور را نشان می‌دهیم.

ورودی‌های خط لوله جمع‌کننده ممیز شناور دو عدد دودویی ممیز شناور نرمالیزه هستند

$$X = A \times 2^a$$
$$Y = B \times 2^b$$

A و B دو عدد کوچکتر از یک هستند که مانتیس‌ها را تشکیل می‌دهند و a و b نماهای آنها می‌باشند. جمع و تفریق ممیز شناور را می‌توان در چهار قطعه انجام داد، شکل ۶-۹. ثبات‌هایی که با R مشخص شده‌اند بین قطعات قرار گرفته‌اند تا نتایج میانی را ذخیره کنند. جزء عمل‌های اجرا شده در

1- Suboperation



شکل ۶-۹ خط لوله برای جمع و تفریق ممیز شناور

قطعات عبارتند از:

۱- مقایسه نماها

۲- همردیف کردن[1] مانتیس‌ها[2]

۳- جمع یا تفریق مانتیس‌ها

۴- نرمالیزه کردن نتیجه

این اعمال در شکل ۱۵-۱۰ با کمی تغییر به منظور کاهش زمان اجرای زیرعمل‌ها آورده شده است. نماها توسط تفریق مقایسه شده تا اختلاف آنها پیدا شود. نمای بزرگتر بعنوان نمای نتیجه انتخاب می‌شود. تفاضل نماها نشان می‌دهد که چندبار مانتیس مربوط به نمای کوچکتر باید به راست شیفت داده شود. به این ترتیب دو مانتیس هم ردیف می‌شوند. باید توجه داشت که برای کاهش زمان شیفت، مدار شیفت دهنده بصورت مدار ترکیبی تهیه می‌شود. دو مانتیس در قطعه سوم جمع یا تفریق می‌شوند. نتیجه در قطعه چهار نرمالیزه می‌شود. وقتی که یک سرریز[3] رخ دهد مانتیس مجموع یا تفاضل یکبار به راست شیفت یافته و نما یک واحد افزایش می‌یابد. اگر یک فروریز[4] رخ دهد تعداد صفرهای جلو مانتیس بیانگر تعداد شیفت به چپ در مانتیس بوده و عددی که باید از نما کسر شود را معین می‌کند.

مثال عددی زیر می‌تواند ریز عمل‌های انجام شده در هر قطعه را مشخص نماید. به منظور سادگی، هرچند که در شکل ۶-۹ اعداد دودویی بکار رفته‌اند، از اعداد دهدهی استفاده می‌کنیم. دو عدد ممیز شناور نرمالیزه زیر را در نظر بگیرید.

$$X = 0.9504 \times 10^3 \qquad Y = 0.8200 \times 10^2$$

دو نما در اولین قطعه از یکدیگر کسر می‌شوند تا 1 =2 − 3 بدست آید. نمای بزرگتر یعنی 3، بعنوان نمای نتیجه انتخاب شده است. قطعه بعدی مانتیس Y را به راست شیفت می‌دهد تا روابط زیر بدست آیند:

$$X = 0.9504 \times 10^3 \qquad Y = 0.0820 \times 10^3$$

باین ترتیب دو مانتیس هم ردیف و دارای نمای یکسان می‌شوند. جمع دو مانتیس در قطعه 3 حاصل جمع زیر را تولید می‌نماید

$$Z = 1.0324 \times 10^3$$

حاصل جمع با نرمالیزه کردن آن تنظیم می‌گردد بطوری که دارای اولین رقم غیرصفر در قسمت اعشاری باشد. این عمل توسط شیفت به راست و افزایش نما بدست می‌آید

$$Z = 0.10324 \times 10^4$$

مقایسه‌گر، شیفت‌دهنده، جمع‌کننده، تفریق‌گر، افزایش‌دهنده، و کاهش دهنده در خط لوله ممیز شناور

1- Align 2- Mantissas 3- Overflow
4- underflow

بوسیله مدارات ترکیبی ساخته می‌شوند. فرض کنید که تأخیر در چهار قطعه $t_1 = 60ns$ ، $t_2 = 70ns$ ، $t_3 = 100ns$ و $t_4 = 80ns$ و ثبات مدار واسطه $t_r = 10ns$ باشد. پالس ساعت $t_p = t_3 + t_r = 110ns$ انتخاب می‌شود. یک مدار غیرخط لوله ممیز شناور جمع و تفریق معادل، زمان تأخیری برابر $t_n = t_1 + t_2 + t_3 + t_4 + t_r = 320$ دارد. در این حال جمع‌کننده 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 نیز 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 برهم خوردن توالی گردد. در این صورت، خط لوله باید تخلیه شده و تمام دستوراتی که بعد از دستور انشعاب قرار دارند و از حافظه خوانده شده‌اند باید نادیده گرفته شود.

کامپیوتری 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 روش بافر FIFO[1] 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 دیکد 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 فضائی وجود داشته باشد، واحد کنترل فاز برداشت دستور بعدی را آغاز می‌کند. بافر همانند یک صف عمل نموده و کنترل، دستورات را از آن برای واحد اجرا استخراج می‌کند.

کامپیوترهایی که دارای دستورات پیچیده‌ای هستند (پر دستور)[2] علاوه بر فازهای برداشت و اجرا نیاز به فازهای دیگری برای تکمیل پردازش دستورات دارند. در حالات کلی‌تر، کامپیوتر هر دستور را با توجه به مراحل زیر پردازش می‌کند.

۱- برداشت دستور از حافظه
۲- دیکد دستور
۳- محاسبه آدرس موثر
۴- برداشت عملوند از حافظه
۵- اجرای دستور

1- First IN First OUT 2- RISC

۶- ذخیره نتیجه در مکان مناسب

مشکلات عدیده‌ای وجود دارند که مانع اجرای دستورات خط لوله در حداکثر سرعتشان می‌گردند. قطعات مختلف ممکن است زمان‌های متفاوتی را برای اطلاعات وارده صرف نمایند. از برخی قطعات در عملیات خاصی ممکن است پرش شوند. مثلاً یک دستور با روش آدرس ثبات نیازی به محاسبه آدرس موثر ندارد. دو یا سه قطعه ممکن است بطور همزمان نیاز به حافظه داشته باشند که در نتیجه یک قطعه مجبور است تا پایان کار دیگری با حافظه صبر کند. مشکلات و پیچیدگی‌های دستیابی به حافظه گاهی اوقات با بکارگیری دو گذرگاه حافظه برای دستیابی به دستورات و داده‌ها در دو بخش مجزا حل می‌شود. در این روش، یک کلمه دستور و یک کلمه داده بطور جداگانه از دو واحد مجزا خوانده می‌شوند.

طراحی یک خط لوله دستورالعمل هنگامی بهینه خواهد بود که سیکل دستور به قطعات زمانی برابر تقسیم گردد. زمانی که هر مرحله برای تکمیل کار خود نیاز دارد بستگی به دستور و روش اجرای آن دارد.

مثال: خط لوله دستور چهار قطعه‌ای

فرض کنید که دیکد یک دستور با محاسبه آدرس موثر بتواند در یک قطعه انجام شود. بعلاوه فرض کنید که اغلب دستورات نتایج خود را در داخل یک ثبات پردازنده قرار دهند. بنابراین اجرای دستور و ذخیره سازی نتایج بتواند در یک قطعه صورت گیرد. این فرضیات خط لوله دستور را به چهار قطعه کاهش می‌دهد.

شکل ۷-۹ چگونگی پردازش دستور در CPU را با یک خط لوله چهار قطعه‌ای نشان می‌دهد. ضمن اینکه یک دستور در حال اجرا در قطعه 4 است، دستور بعدی مشغول دریافت عملوند از حافظه در قطعه 3 می‌باشد. آدرس موثر نیز ممکن است در یک مدار محاسباتی جداگانه برای سومین دستور محاسبه گردد، و هر وقت حافظه در دسترس باشد دستورات چهارم و بعد از آن قابل برداشت و جایگزینی در FIFO خواهند بود. بنابراین بطور همزمان تا چهار زیرعمل در سیکل دستور می‌توانند با هم انجام شوند و تا چهار دستور مختلف نیز در حال پیشروی برای اجرای همزمان می‌باشند.

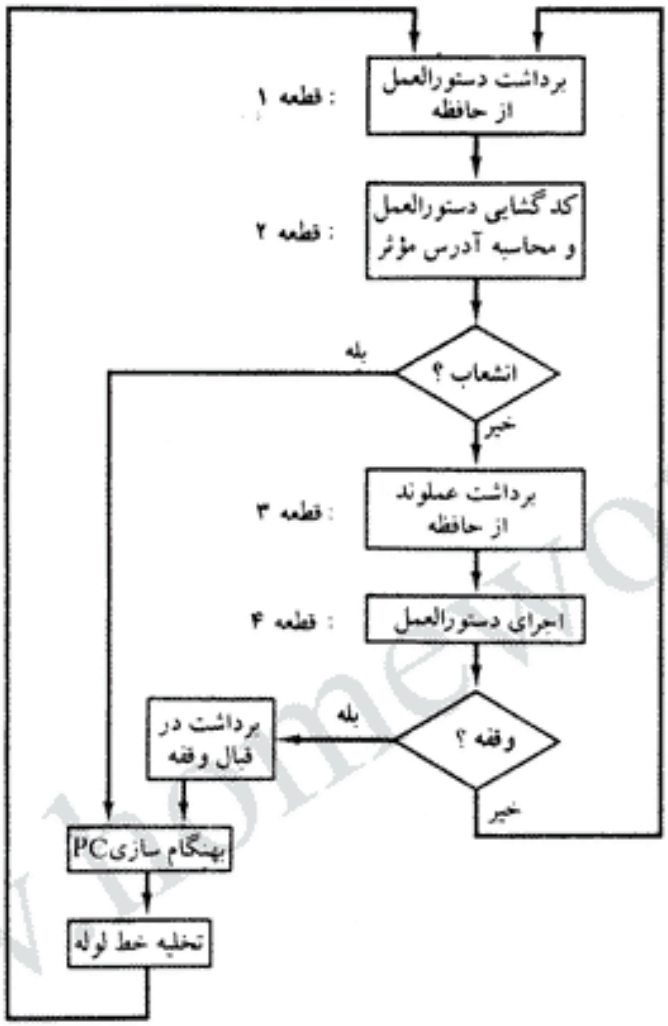
گاهی اوقات یکی از دستورات ممکن است از نوع کنترل برنامه باشد که منجر به انشعاب و خروج عادی از ترتیب معمولی برنامه گردد. در این حال عمل نیمه‌تمام[1] در دو قطعه آخر تکمیل و کلیه اطلاعات ذخیره شده در بافر دستورالعمل پاک می‌شوند. خط لوله سپس، از آدرس جدید ذخیره شده در شمارنده برنامه آغاز بکار می‌کند. بطور مشابه یک تقاضای وقفه هم وقتی تصدیق[2] (تأیید) شد، موجب تخلیه خط لوله و آغاز از آدرس جدید می‌گردد.

شکل ۸-۹ طرز کار یک خط لوله دستورالعمل را نشان می‌دهد. زمان در روی محور افقی به قسمت‌های مساوی تقسیم شده است. چهار قطعه در دیاگرام هم با سمبل‌های اختصاری نشان داده شده‌اند.

۱- FI قطعه‌ای است که دستور را برداشت می‌کند.

۲- DA قطعه‌ای است که دستور را دیکد و آدرس موثر را محاسبه می‌نماید.

1- Pending 2- Acknowledge



شکل ۷-۹ خط لوله چهار قطعه‌ای

مرحله:		1	2	3	4	5	6	7	8	9	10	11	12	13
دستورالعمل:	1	FI	DA	FO	EX									
	2		FI	DA	FO	EX								
(انشعابها)	3			FI	DA	FO	EX							
	4				FI	-	-	FI	DA	FO	EX			
	5					-	-	-	FI	DA	FO	EX		
	6									FI	DA	FO	EX	
	7										FI	DA	FO	EX

شکل ۸-۹ زمانبندی خط لوله دستورالعمل

۳- FO قطعه‌ای است که عملوند را برداشت می‌کند.
۴- EX قطعه‌ای است که دستور را اجرا می‌کند.

فرض بر این است که پردازنده دارای حافظه‌های جداگانه برای دستورات و داده‌ها می‌باشد بطوری که عملیات در FI و FO بطور همزمان اجرای گردند. در صورت غیاب یک دستور انشعاب هر قطعه بر روی دستور جداگانه‌ای عمل می‌کند. بنابراین در مرحله چهارم، دستور 1 در قطعه EX اجرا می‌گردد؛ عملوند دستور 2 در قطعه FO برداشت می‌شود، دستور 3 در قطعه DA دیکد می‌شود و دستور 4 از حافظه هم در قطعه FI برداشت می‌گردد.

حال فرض کنید که دستور 3 یک دستور انشعاب باشد، به محض دیکد شدن این دستور در مرحله 4 و در قطعه DA ، انتقال دستورات دیگر از FI به DA متوقف می‌شود تا دستورالعمل انشعاب در مرحله 6 اجرا شود. اگر انشعاب صورت گیرد، دستور جدیدی در مرحله 7 برداشت می‌گردد. اگر انشعابی صورت نگیرد، دستور قبلی برداشته شده در مرحله ۴ می‌تواند بکار رود. سپس خط لوله تا رسیدن دستور انشعاب جدید بکار خود ادامه می‌دهد.

تأخیر دیگری که ممکن است در خط لوله رخ دهد این است که ضمن نیاز قطعه FO به حافظه برای برداشت داده، EX نیز نیاز به حافظه برای ذخیره کردن نتیجه خود داشته باشد. در این حال FO تا پایان کار EX منتظر می‌ماند.

بطور کلی سه مشکل عمده باعث انحراف دستور خط لوله از روند معمولی می‌گردند:

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 وابستگی داده‌ها[1] نیز هنگامی رخ می‌دهد که دستوری به نتیجه دستور قبلی دیگری وابستگی داشته باشد و آن نتیجه هنوز بدست نیامده باشد.
۳- مشکلات انشعاب از دستور انشعاب و سایر دستوراتی که PC را عوض می‌کنند ناشی می‌شود.

وابستگی به داده‌ها

مشکلی‌که ممکن است سبب افت سرعت در یک خط لوله دستورگردد بعلت احتمال برخورد داده‌ها یا آدرس‌ها می‌باشد. برخورد هنگامی رخ می‌دهد که دستور نمی‌تواند بعلت دستور قبلی پیشروی کند چراکه دستور قبلی کار خود را تمام نکرده است. وابستگی داده هنگامی رخ می‌دهد که دستوری، داده‌ای راکه هنوز آماده نیست لازم دارد. مثلاً اگر یک دستور در قطعه FO بخواهد عملوندی را برداشت کند که در همان لحظه توسط دستور قبلی در قطعه EX در حال تولید باشد، دستور دوم باید تا در دسترس قرار گرفتن عملوند تأمل نماید. بطور مشابه نوعی وابستگی آدرس هنگامی رخ می‌دهد که آدرس عملوند، بعلت موجود نبودن اطلاعات لازم در روش آدرس‌دهی، قابل محاسبه نیست. مثلاً دستوری با روش

1- Data Dependency

آدرس‌دهی غیرمستقیم، اگر که دستور قبلی در حال بارکردن آدرس مزبور در ثبات باشد، نمی‌تواند برای دریافت عملوند جلو برود. بنابراین دستیابی به عملوند باید تا در دسترس قرار گرفتن آدرس مورد نیاز بتعویق بیفتد. کامپیوترهای خط لوله با چنین مشکلات در وابستگی داده‌ها به روشهای مختلفی برخورد می‌کنند

مستقیم‌ترین روش ایجاد همبندهای سخت افزاری[1] می‌باشد. همبند مداری است که وظیفه‌اش شناسایی دستوراتی است که عملوندهای مبدأ[2] آنها مقصد عملوندهای دستورات بعدی در خط لوله است. شناسایی این دستورات سبب می‌شود تا دستوراتی که منابع آنها در دسترس نیستند با تأخیری کافی از پالس‌های ساعت بموقع اجرا شوند و بدین ترتیب مشکل مرتفع گردد. این روش ترتیب برنامه را با اضافه کردن سخت‌افزار و نگهداری تأخیر لازم در آن حفظ می‌کند.

روش دیگری که ارسال عملوند[3] نام دارد سخت‌افزار خاصی را برای شناسایی مشکل بکار برده و سپس با جهت‌دهی به داده از طریق مسیرهای خاص بین قطعات از بروز مشکل جلوگیری می‌کند. مثلاً در عوض انتقال نتیجه حاصل از ALU به ثبات مقصد، استفاده از آن را در دستور بعدی بعنوان منبع چک می‌کند و در صورت لزوم نتیجه را مستقیماً به ورودی ALU فرستاده و بدین ترتیب فایل ثبات‌ها را کنار می‌گذارد. این روش به مسیرهای سخت‌افزاری اضافی از طریق مولتی‌پلکسرها، علاوه بر مدار شناسایی اشکال، نیاز دارد.

روشی که در برخی کامپیوترها بکار برده می‌شود سپردن مسئولیت رفع مشکل به کامپایلر می‌باشد که وظیفه تبدیل زبانهای سطح بالا را به زبان ماشین بعهده دارد. کامپایلر برای چنین کامپیوترهایی چنان طراحی شده تا وجود یک مشکل در داده را شناسایی کرده و با اجرای دستورات "هیچ‌کار"[4] (NO–OP) و تغییر ترتیب اجرای دستورالعمل‌ها، تأخیر لازم را در بارکردن داده‌ها بوجود آورد. این روش بنام بارکردن با تأخیر[5] خوانده می‌شود. مثالی از بارکردن با تأخیر در بخش بعدی آورده شده است.

دستکاری دستورالعمل‌های انشعاب

یکی از مسائل مهم در عملکرد خط لوله دستورالعمل، وقوع یک دستور انشعاب است. دستور انشعاب می‌تواند شرطی و یا غیرشرطی باشد. یک انشعاب بدون شرط همواره ترتیب اجرای برنامه را با بارکردن آدرس هدف در شمارنده برنامه عوض می‌کند. در یک انشعاب شرطی، کنترل دستور مقصد را اگر شرط برقرار شده باشد انتخاب می‌نمایند در غیراینصورت دستور بعد از انشعاب اجرا خواهد شد. همانطور که قبلاً اشاره شد، دستورات انشعاب ترتیب اجرای رشته دستورات را شکسته و مشکلاتی را در کار خط لوله دستور ایجاد می‌نماید. کامپیوترهای خط لوله‌ای تکنیک‌های سخت‌افزاری مختلفی را برای حداقل کردن افت سرعت عملکرد حاصل از دستورات انشعاب بکار می‌برند.

یکی از روشهای دستکاری دستورات انشعاب دریافت زود هنگام دستور مقصد علاوه بر دستور بعد

1- Hardware interlock
2- Source operand
3- Operand forwarding
4- NO Operation, NO–OP
5- Delayed Load

از انشعاب است. هر دو دستور تا ختم اجرای دستور انشعاب ذخیره می‌شوند. اگر شرط انشعاب برقرار باشد خط لوله از دستور مقصد انشعاب ادامه کار می‌دهد. روش کامل‌تر این است که از هر دو جهت عمل برداشت دستورات ادامه یابد تا تصمیم انشعاب اتخاذ گردد. در آن زمان کنترل درباره رشته صحیح دستورالعمل‌ها برای ادامه برنامه تصمیم می‌گیرد.

امکان دیگر بکارگیری بافر مقصد انشعاب[1] یا BTB می‌باشد. BTB یک حافظه اشتراکی[2] است (بخش ۴-۱۲ ملاحظه شود) که در قطعه برداشت خط لوله کار گذاشته می‌شود. هر ورودی به BTB متشکل از آدرس انشعاب اجرا شده قبلی و دستور مقصدبرای آن انشعاب است. این بافر چند دستور بعد از انشعاب را نیز ذخیره می‌نماید. وقتی که خط لوله دستور انشعاب را دیکد می‌کند BTB را برای آدرس دستورالعمل جستجو می‌نماید. اگر آدرس را در BTB بیابد، دستور مستقیماً در دسترس است و برداشت پیش هنگام دستور برای مسیر جدید ادامه می‌یابد. اگر دستور در BTB نباشد، خط لوله به رشته دستور جدیدی شیفت نموده و دستور مقصد را در BTB ذخیره می‌نماید. مزیت این روش این است که دستورات انشعابی که قبلاً اتفاق افتاده‌اند بدون توقف در خط لوله موجودند.

روش دیگر با کمی تغییر با BTB، روش بافر حلقه[3] است. این بافر یک فایل ثبات کوچک و سریع است که تحت نظر قطعه برداشت دستور خط لوله می‌باشد. وقتی که یک حلقه در برنامه شناسایی شود در یک بافر حلقه به طور کامل همراه با انشعابات ذخیره می‌شود. بنابراین حلقه برنامه می‌تواند مستقیماً و بدون دستیابی به حافظه اجرا شود تا اینکه آخرین انشعاب موجب خروج از حلقه شود.

روش بعدی بکاررفته توسط برخی کامپیوترها، پیش‌بینی انشعاب[4] است. یک خط لوله با پیش‌بینی انشعاب نتیجه حاصل از انشعاب را قبل از اجرای آن بکمک مدارهای منطقی اضافی پیش‌بینی می‌نماید. خط لوله سپس شروع به دریافت پیش‌هنگام رشته و دستورات مسیر پیش‌بینی شده می‌نماید. یک پیش‌بینی صحیح زمان تلف شده بوسیله انشعاب را از بین می‌برد.

روشی که توسط اغلب پردازنده‌های RISC مورد استفاده است انشعاب تأخیر داده شده[5] می‌باشد. در این روش، کامپایلر دستورات انشعاب را شناسایی و کدهای زبان ماشین را جابجا نموده و دستورات مفیدی که خط لوله را در حال کار نگهدارد داخل آن می‌نماید. مثالی از این نوع وارد کردن دستورات "هیچ‌کار" پس از دستور انشعاب است. این عمل سبب می‌شود تاکامپیوتر در طول اجرای دستورالعمل فوق دستور مقصد را برداشت کند و در نتیجه خط لوله پیوسته در حال کار نگهداشته شود. مثالی از این انشعاب تأخیر یافته در بخش بعدی ارائه شده است.

۹-۵ خط لوله RISC

کامپیوتر کم دستور (RISC) در بخش ۸-۸ معرفی شد. از جمله مشخصات مربوط به RISC

1- Branch Targat Buffer
2- Associtive memory
3- Loop Buffer
4- Branch Prediction
5- Delaged 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 آدرس 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 مقصد بکار برد.

دستورات انتقال داده در RISC به دستورات بارکردن[1] و ذخیره کردن[2] محدودند. این دستورات از آدرس‌دهی 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 مواقع به نام حافظه‌های نهان[3] خوانده می‌شوند (بخش ۶-۱۲).

همانطور که در بخش ۸-۸ گفته شد یکی از مزایای مهم RISC توانایی اجرای دستورات با سرعت یک دستور در هر پالس ساعت است. البته نباید تصور کرد که هر دستور در یک پالس از حافظه برداشت و اجرا شود. در واقع آنچه عملاً اتفاق می‌افتد این است که آغاز هر دستور با هر پالس بوده و استفاده از پردازنده خط لوله در رسیدن به هدف، بمعنی نیل به اجرای دستورات در یک پالس است. مزیت RISC در مقابل CISC این است که RISC می‌تواند با استفاده از قطعات خط لوله که فقط یک سیکل ساعت نیاز دارند به هدف برسد، درحالیکه 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 بعوض اتکاء بر طراحی‌های سخت‌افزاری که با مشکلات ناشی از داده و تأخیرهای انشعاب مقابله می‌کنند، بر توانایی 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 ۱۲-۸ 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 می‌باشند که آدرس موثر بکار رفته را از افزایش محتویات دو ثبات یا ثبات و ثابت جابجایی (تغییر مکان) حاصل از دستور بدست می‌آورند. دستورات کنترل برنامه مقادیر ثبات و یک ثابت را برای ارزیابی آدرس انشعاب جهت انتقال به شمارنده برنامه PC یا یک ثبات بکار می‌برند.

اکنون عملیات سخت‌افزاری را برای چنین کامپیوتری ملاحظه کنید. بخش کنترل، دستور را از حافظه برنامه به یک ثبات دستورالعمل منتقل می‌کند. دستورالعمل، همزمان با انتخاب ثبات‌های لازم بهنگام اجرا دیکد می‌شود. واحد پردازنده متشکل از تعدادی ثبات و یک واحد محاسباتی منطقی ALU است که اعمال محاسباتی، منطقی و شیفت را انجام می‌دهد. یک حافظه داده برای ذخیره یا بارکردن داده در ثبات واقع در فایل ثبات‌ها لازم است. سیکل دستور می‌تواند به سه عمل جزیی که در سه قطعه پیاده شده است تقسیم شود

I - برداشت دستور
A - عمل ALU
E - اجرای دستور

قطعه I دستور را از حافظه برنامه برداشت می‌کند. دستور، دیکد شده و یک عمل ALU در قطعه A انجام می‌شود. ALU، بسته به دستور دیکد شده برای انجام سه وظیفه بکار می‌رود؛ یا عملی را برای دستورالعمل‌های دستکاری داده‌ها انجام می‌دهد، یا آدرس موثر یک دستور بارکردن و یا ذخیره را محاسبه می‌نماید و یا آدرس انشعاب را برای دستورالعمل‌های کنترل برنامه محاسبه می‌کند. قطعه E خروجی ALU، را به یکی از سه مقصد، بسته به نوع دستور، هدایت می‌نماید. یا نتیجه عمل ALU را به ثبات مقصدی واقع در فایل ثبات‌ها ارسال می‌دارد، یا آدرس موثر را جهت بارکردن یا ذخیره کردن به حافظه داده‌ها می‌فرستد و یا آدرس انشعاب را به شمارنده برنامه منتقل می‌کند.

بارکردن با تأخیر

اکنون عملکرد چهار دستور زیر را در نظر بگیرید:

1- LOAD: R1 ← M[address1]
2- LOAD: R2 ← M[address2]
3- ADD: R3 ← R1 + R2
4- STORE: M[address3] ← R3

اگر خط لوله سه قطعه‌ای 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

سیکل‌های ساعت :	1	2	3	4	5	6
1. Load *R1*	I	A	E			
2. Load *R2*		I	A	E		
3. Add *R1* + *R2*			I	A	E	
4. Store *R3*				I	A	E

(الف) زمانبندی خط لوله با مشکل غیاب داده

سیکل‌های ساعت :	1	2	3	4	5	6	7
1. Load *R1*	I	A	E				
2. Load *R2*		I	A	E			
3. No-operation			I	A	E		
4. Add *R1* + *R2*				I	A	E	
5. Store *R3*					I	A	E

(ب) زمانبندی خط لوله با بارکردن تأخیری

شکل ۹-۹ زمانبندی خط لوله سه قطعه‌ای

است. قطعه A در سیکل ساعت 4 داده‌ای را از R2 بکار می‌برد، ولی کمیت موجود در R2 معتبر نیست‌زیرا مقدار صحیح، هنوز از حافظه به آن منتقل نشده است. در اینجا کامپایلر باید مطمئن شود که دستور بعدی قادر به استفاده داده از حافظه هست. اگر موفق به این کار نشود و دستوری را نتواند پس از بارکردن قرار دهد یک دستور " هیچ‌کار"[1] (no-op) در برنامه می‌گنجاند. این دستورالعمل از حافظه برداشت می‌شود ولی هیچ‌کاری را انجام نداده و تنها یک سیکل ساعت وقت را تلف می‌کند. ایجاد تأخیر در استفاده از دادهٔ حافظه، بارکردن با تأخیر نام دارد.

شکل ۹-۹ (ب) برنامه قبل را همراه با یک دستور هیچ‌کار که پس از بارکردن R2 آمده است نشان می‌دهد. داده در سیکل ساعت 4 در R_2 بار می‌شود. دستور ADD مقدار R2 را در مرحله 5 استفاده می‌کند. بنابراین دستور هیچ‌کار اجرا را به اندازه یک سیکل ساعت بتأخیر می‌اندازد تا جبران زمانی برای داده در خط لوله انجام شود. (توجه کنید که عمل " هیچ‌کار" حین سیکل 4 در قطعه A و حین سیکل 5 در قطعه E انجام می‌شود). مزیت بارکردن با تأخیر این است که موضوع وابستگی داده توسط کامپایلر مورد توجه قرار می‌گیرد و نه توسط سخت‌افزار. این روش منجر به ساخت مدارات سخت‌افزاری ساده‌تر می‌گردد زیرا قطعه نیازی به کنترل ثبات جهت معتبر یافتن و یا نیافتن محتوای آن ندارد.

1- No - Operation

انشعاب با تأخیر

در شکل ۸-۹ دیدیم که یک دستور انشعاب عملکرد خط لوله را تا برداشت آدرس انشعاب با تأخیر مواجه می‌سازد. در مورد کاهش زمان تلف شده ناشی از انشعاب در بخش قبل بحث کردیم. روشی که در اغلب پردازنده‌های RISC معمول است تکیه بر کامپایلر می‌باشد که توسط آن انشعاب‌ها در زمان مناسب در خط لوله قرار می‌گیرند. این روش انشعاب با تأخیر نامیده شده است.

کامپایلر مربوط به پردازنده‌ای که انشعاب با تأخیر را استفاده می‌کند طوری طراحی شده است که دستورالعمل‌های قبل و بعد از انشعاب را تحلیل نموده و توالی برنامه را با وارد کردن دستورات مفید در آن تغییر دهد. مثلاً کامپایلر با توجه به وابستگی‌های برنامه قادر است یک یا دو دستور قبل از انشعاب را به مرحله تأخیر بعد از انشعاب منتقل کند. سپس این دستورات از حافظه برداشت شده و در خط لوله، همزمان با اجرای انشعاب در سایر قطعات، اجرا می‌گردد. اثر کلی بمانند اجرای دستورات در فرم طبیعی آنهاست با این تفاوت که تأخیر انشعاب حذف گردیده است. این دیگر بعهده کامپایلر است تا دستورالعمل‌های مفیدی را یافته و پس از دستورالعمل انشعاب قرار دهد. در صورت ناموفق بودن، کامپایلر دستورات هیچ‌کار را وارد خواهد کرد.

مثالی از انشعاب با تأخیر در شکل ۱۰-۹ دیده می‌شود. برنامه این مثال دارای پنج دستورالعمل است.

- بار کردن حافظه در R1
- افزایش R2
- جمع R3 با R4
- تفریق R5 از R6
- انشعاب به آدرس X

در شکل ۱۰-۹ (الف) کامپایلر دو دستور هیچ‌کار را بعد از انشعاب قرار می‌دهد. آدرس انشعاب، X، در سیکل ساعت 7 به PC منتقل می‌شود. برداشت دستور در X با تأخیری برابر با دو سیکل ساعت توسط دستور هیچ‌کار (no-op) صورت می‌گیرد. دستور واقع در آدرس X، فاز برداشت خود را در سیکل 8، یعنی پس از بهنگام شدن شمارنده PC، انجام می‌دهد.

آرایش برنامه شکل ۱۰-۹ (ب) با قرار گرفتن دستورالعمل‌های جمع و تفریق در بعد از انشعاب، بجای قبل از آن در برنامه اصلی، تغییر یافته است. بررسی زمانبندی خط لوله نشان می‌دهد که PC در سیکل ساعت 5 با مقدار X بهنگام می‌شود، ولی دستورات جمع و تفریق به ترتیب صحیح از حافظه برداشت و اجرا می‌شوند. به بیان دیگر، اگر دستور بارکردن در آدرس 101 و نیز X برابر 350 باشد، دستورالعمل انشعاب از مکان 103 برداشت می‌شود. دستورالعمل جمع از مکان 104 برداشت و در سیکل 6 اجرا می‌شود. دستورالعمل تفریق از آدرس 105 برداشت و در سیکل ساعت 7 اجرا می‌گردد. چون مقدار X در سیکل ساعت 5 در قطعه E به PC منتقل می‌شود، دستورالعمل برداشت شده از حافظه در سیکل ساعت 6 از آدرس 350، دستور موجود در آدرس انشعاب است.

سیکل‌های 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

سیکل‌های ساعت :	1	2	3	4	5	6	7	8
1. Load	I	A	E					
2. Increment		I	A	E				
3. Branch to *X*			I	A	E			
4. Add				I	A	E		
5. Subtract					I	A	E	
6. Instruction in *X*						I	A	E

(ب) تغییر آرایش دستورالعمل‌ها

شکل ۱۰-۹ مثال انشعاب با تأخیر

۹-۶ پردازش برداری

رده‌ای 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 برحسب جملات برداری و یا ماتریسی هستند که برای پردازش موازی مناسبند.

کامپیوترهایی که قابلیت پردازش برداری را دارند در کارهای تخصصی کاربرد دارند. موارد زیر نمونه‌هایی از زمینه‌های کاربردی با اهمیت در پردازش موازی می‌باشند.

- پیش‌بینی درازمدت وضع هوا
- اکتشافات نفت
- تحلیل داده‌های زلزله‌نگاری
- تشخیص پزشکی

- آئرودینامیک و شبیه سازی پروازهای فضایی
- هوش مصنوعی و سیستم‌های خبره
- نقشه‌برداری از ژن انسان
- پردازش تصویر

بسیاری از اعمال محاسباتی بدون استفاده از کامپیوترهای پیشرفته و پیچیده در زمان معقول تکمیل نمی‌شوند. برای دستیابی به سطح بالا و مطلوبی از عملکرد، لازم است تا از سریع‌ترین و مطمئن‌ترین سخت‌افزار استفاده گردد و رویه‌های ابتکاری از پردازش برداری و موازی به کار گرفته شوند.

عملیات برداری

بسیاری از مسائل علمی نیاز به عملیات محاسباتی بر روی آرایه‌های بزرگی از اعداد دارند. این اعداد معمولاً بصورت بردارها و ماتریس‌هایی از اعداد ممیز شناور فرموله می‌شوند. یک بردار، مجموعه مرتبی از یک آرایه یک بعدی از داده‌هاست. برداری مانند V بطول n بصورت یک بردار سطری $V = [V_1\ V_2\ V_3\ \ldots\ V_n]$ نشان داده می‌شود. با عمودی چیدن اقلام داده‌ها می‌توان این بردار را بصورت عمودی هم نشان داد. یک کامپیوتر ترتیبی معمولی تنها قادر است پردازش عملوندها را یک به یک پردازش نماید. در نتیجه، اعمال روی بردارها باید به محاسبات منفرد با متغیرهای شاخص‌دار تقسیم شود. عنصر V_i از بردار V بصورت V(I) نوشته می‌شود که در آن شاخص (یا اندیس) I به آدرس حافظه یا ثباتی که در آن عدد ذخیره شده است اشاره می‌نماید. 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    C(I) = B(I) + A(I)
```

این برنامه مربوط به جمع دو بردار A و B بطول 100 برای تولید بردار C می‌باشد. برنامه فوق توسط رشته اعمال زیر بزبان ماشین نوشته شده است:

```
      Initialize I = 0
20    Read A(I)
      Read B(I)
      Store C(I) = A(I) + B(I)
      Increment I = I + 1
      If I ≤ 100 go to 20
      Continue
```

رشته عملیات فوق حلقه‌ای از برنامه را ایجادمی‌کند که جفت عملوندها را از آرایه‌های A و B خوانده و جمع ممیز شناور را انجام می‌دهد. سپس متغیر کنترل حلقه بهنگام[1] شده و مراحل 100 بار تکرار می‌گردند.

کامپیوتری که قادر به پردازش برداری است اتلاف وقت مربوط به برداشت و اجرای دستورالعمل‌ها را در حلقه برنامه حذف می‌کند. در این کامپیوترها اعمال توسط دستورالعمل‌های برداری واحدی مانند آنچه در زیر آمده انجام می‌شود.

$$C\ (1{:}100) = A(1{:}100) + B(1{:}100)$$

در این دستور برداری، آدرس اول عملوندها، طول بردارها و عملیات مورد نظر، همگی منظور شده‌اند. جمع توسط یک جمع‌کننده ممیز شناور خط لوله‌ای مشابه با آنچه که در شکل ۶-۹ دیدیم انجام می‌شود.

یک قالب ممکن برای دستورالعمل برداری در شکل ۱۱-۹ نشان داده شده است. این قالب اساساً یک دستور سه آدرسه 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 می‌نمایند.

ضرب ماتریسی

ضرب ماتریسی یکی از اعمال محاسباتی سنگین است که در کامپیوترهای پردازش برداری انجام می‌شود. ضرب دو ماتریس $n \times n$ از n^2 ضرب داخلی یا n^3 عمل ضرب - جمع تشکیل شده است. یک ماتریس $n \times m$ از اعداد، دارای n سطر و m ستون بوده و می‌توان فرض کرد که از مجموعه n بردار سطری یا مجموعه‌ای از m بردار ستونی تشکیل گردیده است. بعنوان مثال ضرب دو ماتریس A و B، 3×3 را در نظر بگیرید.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix}$$

طول بردار	آدرس شروع مقصد	آدرس شروع مبدأ ۲	آدرس شروع مبدأ ۱	کد عمل

شکل ۱۱-۹ قالب دستور برای پردازنده برداری

1- Update

ماتریس حاصلضرب C یک ماتریس 3×3 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 $J = 1$ و $i = 1$ محاسبه می‌شود

$$c_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31}$$

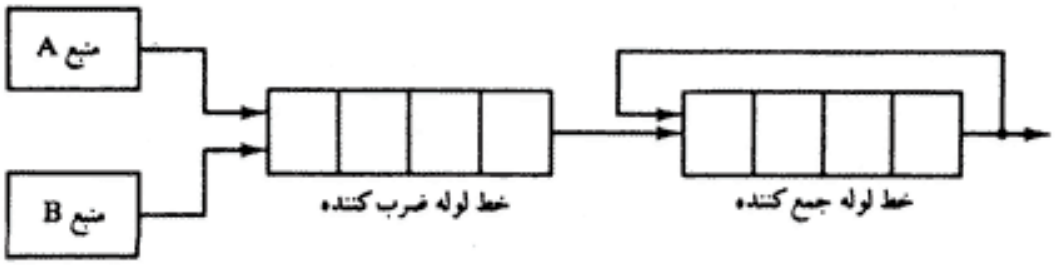
لازمه بدست آمدن این عنصر سه ضرب (پس از مقداردهی اولیه $C_{11} = 0$) و سه جمع می‌باشد. تعداد کل ضرب‌ها و جمع‌های لازم برای محاسبه ماتریس حاصلضرب $27 = 3 \times 9$ است.

اگر ما اعمال ضرب، جمع‌های مرتبط با هم $c + ab$ را یک عمل تجمعی تصور کنیم، ضرب دو ماتریس $n \times n$ نیاز به n^3 عمل ضرب - جمع دارد. محاسبات شامل n^2 ضرب داخلی است که هر ضرب داخلی هم n عمل ضرب و جمع نیاز دارد، البته فرض بر این است که قبل از محاسبه هر عنصر در ماتریس به C مقدار اولیه 0 داده شود.

بطور کلی، ضرب داخلی از مجموع K جمله ضرب مطابق زیر تشکیل یافته است

$$C = A_1B_1 + A_2B_2 + A_3B_3 + A_4B_4 + \cdots + A_kB_k$$

در یک کاربرد نوعی، k ممکن است برابر 100 و حتی 1000 باشد. محاسبه ضرب داخلی در یک خط لوله با پردازش برداری در شکل ۱۲-۹ 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 جمع‌کننده 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 ۱۲-۹ خط لوله برای محاسبه یک ضرب داخلی

سیکل بعدی به حاصلضرب‌هایی که وارد خط لوله جمع‌کننده می‌شوند، 0 اضافه می‌شود. در پایان‌سیکل هشتم، چهار حاصلضرب اول یعنی A_1B_1 تا A_4B_4 در چهار قطعه جمع‌کننده و چهارحاصلضرب 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 $A_6B_6 + A_2B_2$ را آغاز می‌کند و عمل بهمین ترتیب ادامه دارد. با این الگو جمع به چهار بخش زیر تقسیم می‌شود

$$
\begin{aligned}
C = {} & A_1B_1 + A_5B_5 + A_9B_9 + A_{13}B_{13} + \cdots \\
& + A_2B_2 + A_6B_6 + A_{10}B_{10} + A_{14}B_{14} + \cdots \\
& + A_3B_3 + A_7B_7 + A_{11}B_{11} + A_{15}B_{15} + \cdots \\
& + A_4B_4 + A_8B_8 + A_{12}B_{12} + A_{16}B_{16} + \cdots
\end{aligned}
$$

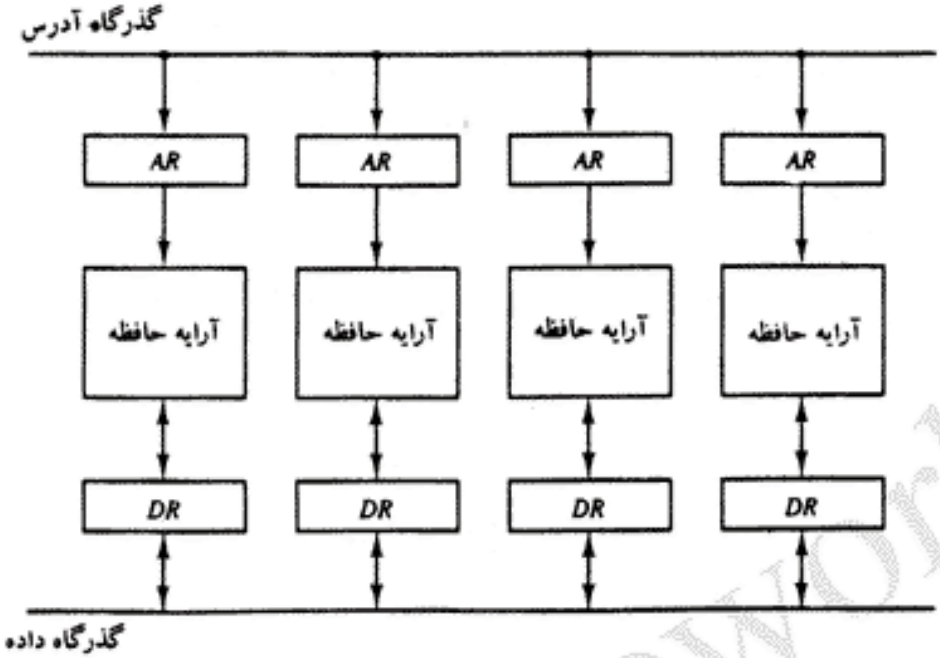
وقتی که دیگر جمله ضربی برای جمع شدن وجود نداشته باشد سیستم چهار صفر بداخل خط لوله ضرب ارسال می‌کند. سپس در هر یک از چهار قطعه تنها یک ضرب جزیی وجود خواهد داشت که مربوط به چهار جمع در چهار سطر معادله فوق است. چهار جمع جزیی در نهایت با هم جمع می‌شوند تا حاصل جمع نهایی حاصل گردد.

برگی‌کردن حافظه[1]

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 ماژول[2] که به گذرگاه‌های آدرس و داده مشترکی وصلند تقسیم کرد. یک ماژول حافظه را آرایه‌ای از حافظه به همراه ثبات‌های داده و آدرس مربوطه‌اش تشکیل می‌دهد. شکل ۱۳-۹ یک واحد حافظه چهار ماژولی 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 کم‌ارزش‌تر آدرس می‌توانند برای تفکیک چهار ماژول بکار روند. سیستم ماژولی اجازه می‌دهد تا یک ماژول، دستیابی به حافظه را آغازنماید در حالیکه سایر ماژول‌ها در حال خواندن یا نوشتن یک کلمه بوده و در واقع هر ماژول مستقل از وضعیت ماژول‌های دیگر به یک درخواست حافظه پاسخ می‌دهد.

مزیت حافظه ماژولی این است که امکان استفاده از تکنیکی بنام برگی کردن حافظه را فراهم می‌آورد. در یک حافظه برگی شده به ماژول‌های مختلف حافظه، آدرس‌های متفاوتی تخصیص یافته است. مثلاً،

1- Interteaving 2- Module



شکل ۱۳-۹ سازمان حافظه چند ماژولی

در یک سیستم با دو ماژول برای حافظه، ممکن است آدرس‌های زوج در یک ماژول و آدرس‌های فرد در دیگری باشد، وقتی تعداد ماژول‌ها توانی از 2 است بیت‌های کم‌ارزش‌تر آدرس، ماژول حافظه را انتخاب می‌نمایند و بقیه بیت‌ها مکان خاصی را برای دستیابی در ماژول انتخاب شده مشخص می‌نمایند.

حافظه ماژولی در سیستم‌های خط لوله‌ای و پردازش برداری مفیدند. یک پردازنده برداری که از یک حافظه n برگی استفاده می‌کند می‌تواند n عملوند را از n ماژول مختلف برداشت نماید. با توزیع دستیابی به حافظه، زمان موثر سیکل حافظه می‌تواند با فاکتوری در حدود تعداد ماژول‌ها کاهش یابد. یک CPU با خط لوله دستورالعمل می‌تواند از تعدد ماژول‌های 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 ممیز 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 فواصلی را که سیگنال‌ها طی می‌کنند حداقل نمایند. برای خارج ساختن گرما از مدارات و به منظور جلوگیری از سوختن آنها بعلت نزدیکی‌شان 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 اسکالرها و بردارها را پردازش می‌نمایند. سوپر کامپیوتر، یک سیستم کامپیوتری است که بیشتر بلحاظ سرعت بالایش در محاسبات، سیستم‌های حافظه سریع و حجیم و حجم بالای پردازش در آن شناخته شده است. این کامپیوتر با واحدهای چندتابعی (چندکاره) تجهیز شده و هر واحد دارای پیکربندی لوله‌ای خاص خود می‌باشد. هرچند سوپر کامپیوتر قادر به اجرای برنامه‌های همه منظوره مربوط به کامپیوترهای معمولی می‌باشد، ولی در اصل برای انواع محاسبات عددی مربوط به بردارها و ماتریس‌های اعداد ممیز شناور بهینه‌سازی شده است.

سوپر کامپیوترها برای پردازش معمولی روزمره یک تشکیلات کامپیوتر مناسب نیستند. کاربرد این کامپیوترها به کاربردهای علمی همچون پیش‌بینی عدد وضع هوا، تحلیل موج‌های زلزله‌نگاری، و تحقیقات فضایی محدود است. ضمناً بعلت قیمت گران کاربرد و بازار محدودی نیز دارد.

یکی از معیارهای بکار رفته برای ارزیابی کامپیوترها، انجام تعداد اعمال روی اعداد ممیز شناور در هر ثانیه است که به آن فلاپس [1] می‌گویند. هر مگلفلاپس برابر یک میلیون فلاپس و یک گیگافلاپس [2] نیز یک میلیارد فلاپس است. یک نمونه سوپر کامپیوتر از این رده دارای سیکل زمانی پایه‌ای برابر با 4 تا 20 نانوثانیه می‌باشد. اگر پردازنده قادر به اجرای هر عمل ممیز شناور در یک سیکل باشد، بین 50 تا 250 مگافلاپس قابلیت اجرایی دارد. این سرعت از زمانی که اولین پاسخ تولید می‌شود برقرار خواهد بود و شامل زمان راه‌اندازی اولیه خط لوله نیست.

اولین سوپر کامپیوتر بنام cray−1 در سال ۱۹۷۱ ساخته شد. این کامپیوتر از پردازش برداری همراه با 12 واحد علمیاتی موازی جداگانه استفاده می‌کند. هر واحد عملیاتی به قطعاتی تقسیم شده است تا داده‌های ورودی را از طریق خط لوله پردازش نماید. همه واحدها می‌توانند بطور موازی با عملوندهایی که در ثبات‌های متعدد CPU ذخیره شده‌اند (بیش از ۱۵۰ ثبات) بطور همزمان کارکنند. یک عمل ممیز شناور می‌تواند روی دو مجموعه عملوند 64 بیتی در طول یک سیکل ساعت 12.5 نانوثانیه‌ای انجام شود. در نتیجه سرعت پردازش در خط لوله 80 مگافلاپس در حین پردازش داده خواهد بود. ظرفیت حافظه این کامپیوتر 4 میلیون کلمه 64 بیتی است. این حافظه به 16 بانک (بخش) تقسیم شده است و هر کدام دارای زمان دسترسی 50 نانوثانیه می‌باشند. بنابراین هرگاه همه بانک‌ها با هم دستیابی شوند سرعت انتقال حافظه 320 میلیون کلمه در ثانیه است. تحقیقات سازندگان cray موجب گسترش این کامپیوتر گردید و در نتیجه Cary X−MP و Cary Y−MP ساخته شدند. سوپر کامپیوتر جدید cary−2، دوازده برابر قوی‌تر از cary−1 است.

یکی دیگر از سوپر کامپیوترهای اولیه Fujitsu VP−200 است. این کامپیوتر دارای یک پردازنده اسکالر و یک پردازنده برداری است که می‌توانند بطور همزمان کارکنند. در این کامپیوتر هم، مانند سوپر کامپیوتر cary از ثبات‌ها و واحدهای عملیاتی متعددی استفاده شده است تا عملیات ثبات- ثبات امکان‌پذیر باشد. در پردازنده برداری چهار خط لوله اجرای دستورالعمل‌ها وجود دارند که هرگاه همزمان

1- Flops 2- gigaflops

کار کنند به سرعتی حدود 300 مگافلاپس دست خواهند یافت. حافظه اصلی 32 میلیون کلمه داشته که از طریق خط لوله‌های بارکردن و ذخیره کردن به ثبات‌های برداری متصلند. VP200 دارای 83 دستورالعمل برداری و 195 دستور اسکالر است. مدل جدیدتر VP 2600 از سیکل ساعت 3.2 ns استفاده نموده و مدعی اوج سرعتی برابر با 5 گیگافلاپس است.

۹-۷ پردازنده آرایه[1]

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[2] نوعی پردازنده کمکی[3] 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 - چنددداده‌ای را داراست. 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.

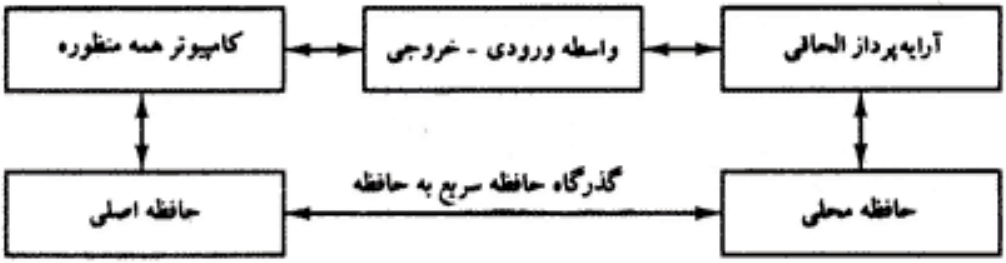
آرایه پرداز الحاقی

یک آرایه‌پرداز 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[4] (عملیاتی) است. این واحدها عبارتند از واحد حساب با یک یا چند جمع و ضرب‌کننده ممیز شناور خط لوله‌ای. آرایه‌پرداز قادر است انواع مسائل ریاضی پیچیده را بکمک برنامه‌های کاربر حل نماید.

شکل ۹-۱۴ اتصالات درونی یک آرایه‌پرداز الحاقی را به یک کامپیوتر میزبان[5] نشان می‌دهد. کامپیوتر میزبان یک کامپیوتر همه منظوره تجاری است و آرایه‌پرداز مانند یک ماشین در انتهای خط[6] است که توسط کامپیوتر میزبان هدایت می‌شود. آرایه‌پرداز 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 آرایه‌پرداز، کاربرانی را که نیاز به پردازش معمولی داده‌ها دارند سرویس می‌دهد. سیستم به همراه آرایه‌پرداز نیازهای کاربردهای حسابی پیچیده را برآورده می‌سازد.

برخی از سازندگان آرایه‌پردازهای 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 ۱۴-۹ آرایه‌پرداز الحاقی با کامپیوتر میزبان

از کامپیوترهای میزبان می‌باشد. مثلاً اگر FSP-164/MAx که یک سیستم ممیز شناور است به VAX11 متصل شود قدرت محاسبه VAX را تا 100 مگافلاپس افزایش می‌دهد. هدف از آرایه‌پرداز الحاقی ایجاد توانایی پردازش برداری کامپیوترهای معمولی با کسری از هزینه لازم برای سوپر کامپیوتر است.

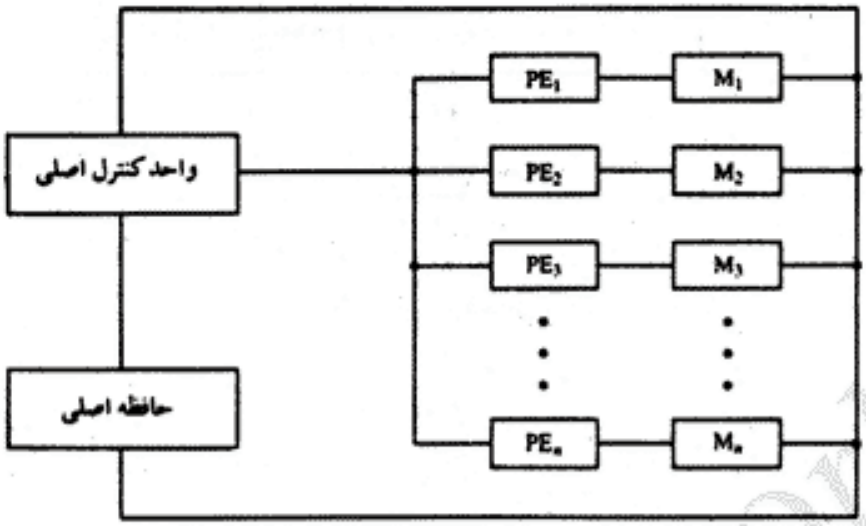
آرایه‌پرداز SIMD

یک آرایه‌پرداز SIMD کامپیوتری است با واحدهای پردازشی متعدد که بطور موازی کار می‌کنند. واحدهای پردازش در این کامپیوتر 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 (PE ها) است که هر کدام یک حافظه محلی M را دارا هستند (شکل ۱۵-۹). 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 اسکالر و کنترل برنامه مستقیماً در داخل واحد کنترل اصلی اجرا می‌شوند. دستورالعمل‌های برداری بطور همزمان به تمام PE ها اعلام می‌شوند. هر PE از عملوندهای ذخیره شده در حافظه محلی‌اش 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 i اُمین مولفه‌های A و B یعنی a_i و b_i در حافظه محلی M_i در ازای $i = 1, 2, 3, ..., n$ را ذخیره می‌کند. سپس دستور جمع ممیز شناور $c_i = a_i + b_i$ بین تمام PE ها 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 ماسک‌کردن[1]

1- Masking



شکل ۹-۱۵ سازمان آرایه‌پرداز SIMD

(پوشش دادن) استفاده می‌شود. هر PE دارای پرچمی است که هرگاه PE فعال باشد 1 می‌شود و هر وقت PE غیرفعال باشد 0 می‌گردد. بدین ترتیب تضمین می‌شود که فقط PE هایی 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 64کلمه‌ای تقسیم شوند.

معروف‌ترین پردازنده آرایه‌ای SIMD، کامپیوتر ILLIAC IV است که در دانشگاه ایلینویز[1] طراحی و توسط شرکت باروز[2] ساخته شد. از این کامپیوتر دیگر استفاده نمی‌شود. کامپیوترهای SIMD کامپیوترهای بسیار تخصصی هستند. این کامپیوترها عمدتاً برای مسائل عددی که قابل ارائه بشکل برداری یا ماتریسی باشند مناسبند. با این وجود در سایر انواع محاسبات یا اجرای برنامه‌های داده پردازی کارآیی چندانی ندارند.

مسائل

۹-۱ در برخی از محاسبات علمی لازم است تا رابطه حسابی $(A_i + B_i)\ (C_i + D_i)$ با رشته‌ای از اعداد انجام شود. یک پیکربندی خط لوله را برای انجام آن مشخص کنید. محتوای تمام ثبات‌های خط لوله را بازاء 6 تا $i = 1$ لیست نمایید.

۹-۲ یک دیاگرام فاصله- زمان برای یک خط لوله شش قطعه‌ای رسم کنید که زمان پردازش هشت

1- Illinois 2- Burroughs

تکلیف را نشان دهد.

۹-۳ تعداد سیکل‌هایی ساعتی که برای 200 تکلیف در یک خط لوله شش قطعه‌ای لازم است را معین کنید.

۹-۴ یک سیستم غیرخط لوله‌ای برای پردازش یک تکلیف 50 ns زمان نیاز دارد. همان تکلیف در یک خط لوله شش قطعه‌ای به یک سیکل ساعت 10 نانوثانیه‌ای 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: 40 ns برای خواندن عملوند از حافظه به ثبات R_1 و R_2، 45 ns برای انتشار سیگنال در ضرب‌کننده، 5 ns برای انتقال به R_3 و 15 ns برای جمع دو عدد و قرار دادن آن در R_5.

(الف) حداقل سیکل ساعتی که می‌توان بکار برد چقدر است؟

(ب) یک سیستم غیرخط لوله‌ای می‌تواند همین عمل را با حذف R_3 و R_4 انجام دهد. ضرب و جمع عملوندها بدون استفاده از خط لوله چقدر طول می‌کشد؟

(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 می‌خواهیم یک خط لوله ضرب ممیز شناور که دو عدد صحیح دودویی هشت بیتی را ضرب می‌کند طراحی کنیم. هر قطعه از تعدادی گیت AND و یک جمع کننده دودویی شبیه به ضرب‌کننده 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 30 ns باشد، زمان متوسط لازم برای ضرب دو عدد ممیز ثابت در خط لوله چقدر است؟

۹-۷ زمان تأخیر چهار قطعه در خط لوله شکل ۹-۶ بقرار زیر است:

$t_1 = 50$ ns، $t_2 = 30$ ns، $t_3 = 95$ ns و $t_4 = 45$ ns. تأخیر ثبات‌های واسط $t_r = 5$ ns.

(الف) مجموع 100 جفت عدد در خط لوله چقدر طول می‌کشد؟

(ب) چگونه می‌توان زمان کل را به نصف زمان محاسبه شده در قسمت (الف) رساند؟

۹-۸ چگونه می‌توانید جمع‌کننده ممیز شناور خط لوله‌ای شکل ۹-۶ را برای جمع 100 عدد ممیز شناور $X_1 + X_2 + X_3 + \ldots + X_{100}$ بکار برید؟

۹-۹ یک خط لوله دستورالعمل شش قطعه‌ای را برای یک کامپیوتر فرموله کنید. عملیات هر قطعه را نیز مشخص نمایید.

۹-۱۰ چهار روش سخت‌افزاری ممکن را که می‌توان 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 ۸-۹ آغاز شود. عملیات انجام شده در چهار قطعه در چهار مرحله را مشخص کنید

```
Load    R1←M[312]
ADD     R2←R2 + M[313]
INC     R3←R3 + 1
STORE   M[314]←R3
```

۹-۱۲ برنامه‌ای را مثال بزنید که موجب اغتشاش داده در خط لوله سه قطعه بخش ۵-۹ گردد.

۹-۱۳ مثالی ارائه نمایید که در آن در خط لوله سه قطعه‌ای بخش ۵-۹ با تأخیر عمل بارکردن را صورت دهد.

۹-۱۴ برنامه‌ای مثال بزنید که موجب اتلاف زمان ناشی از انشعاب در خط لوله سه قطعه‌ای بخش ۵-۹ گردد.

۹-۱۵ مثالی ارائه کنید که در آن از انشعاب با تأخیر در خط لوله سه قطعه‌ای بخش ۵-۹ استفاده شود.

۹-۱۶ ضرب دو ماتریس 40 × 40 را در یک پردازنده برداری در نظر بگیرید.

(الف) چند جمله ضرب در هر حاصلضرب داخلی وجود دارد و چند حاصلضرب داخلی باید محاسبه شود.

(ب) چند عمل ضرب- جمع برای محاسبه ماتریس لازم است.

۹-۱۷ در شکل ۹-۱۲ وقتی که ماتریس‌های 60 × 60 درهم ضرب شوند چند سیکل ساعت برای پردازش حاصلضرب داخلی در خط لوله لازم است؟ چند حاصلضرب داخلی وجود دارد و چند سیکل ساعت برای محاسبه ماتریس حاصلضرب لازم است؟

۹-۱۸ آدرس‌های مربوط به یک آرایه 1024 کلمه، که قرار است در حافظه شکل ۹-۱۳ ذخیره شوند را مشخص نمایید.

۹-۱۹ محاسبه‌ای برای پیش‌بینی وضع هوا به 250 میلیارد عمل ممیز شناور نیاز دارد. مسئله در یک سوپر کامپیوتر با سرعت اجرایی 100 مگافلاپس پردازش می‌شود. زمان اجرا چقدر است؟

۹-۲۰ کامپیوتر را با چهار پردازنده لوله‌ای ممیز شناور در نظر بگیرید. فرض کنید که هر پردازنده از سیکل زمان 40 ns استفاده می‌کند. برای اجرای 400 عمل ممیز شناور چقدر وقت لازم است؟ اگر همین عمل با استفاده از یک پردازنده لوله‌ای تکی با سیکل ساعت 10 ns انجام شود چه تفاوتی خواهد داشت؟

www.homework.ir مشاوره و انجام پروژه هاي دانشجويي دانلود فصلهايي از كتب دانشگاهي

۱۰ معماری کامپیوتر

۱۰-۱ مقدمه
۱۰-۲ جمع و تفریق
۱۰-۳ الگوریتم ضرب
۱۰-۴ الگوریتم‌های تقسیم
۱۰-۵ عمل های حسابی ممیز شناور
۱۰-۶ واحد حساب دهدهی
۱۰-۷ اعمال حسابی دهدهی

۱۰-۱ مقدمه

دستورالعمل‌های حسابی در کامپیوترهای دیجیتال با دستکاری داده‌ها، نتایج لازم برای حل مسائل محاسباتی را تولید می‌کنند. این دستورالعمل‌ها محاسبات ریاضی را انجام می‌دهند و مسئول مجموعه فعالیت‌های عمده‌ای در پردازش داده‌ها در کامپیوترند. چهار عمل اصلی حساب عبارتند از جمع، تفریق، ضرب و تقسیم. با استفاده از این چهارعمل، می‌توان توابع حسابی دیگر را تنظیم کرده و مسائل علمی را با استفاده از روشهای محاسبات عددی حل کرد.

پردازنده حسابی بخشی از یک واحد پردازنده است که اعمال حسابی را اجرا می‌کند. نوع داده‌ای که در طول اجرای دستور در داخل ثبات‌های پردازنده قرار می‌گیرند در تعریف دستورالعمل مشخص می‌گردد. یک دستورالعمل حسابی ممکن است داده‌های دودویی یا دهدهی داشته باشد، و در هر حال داده‌ها ممکن است بفرم ممیز ثابت یا ممیز شناور باشند. اعداد ممیز ثابت ممکن است صحیح یا کسری باشند. اعداد منفی می‌تواند بصورت اندازه علامت‌دار یا متمم علامت دار نشان داده شوند. اگر فقط دستورالعمل جمع برای عددهای دودویی ممیز ثبات در نظر گرفته شود، پردازش حسابی بسیار ساده خواهد بود. اما اگر در پردازنده حسابی هر چهار عمل اصلی برای داده‌های دودویی و دهدهی به شکل ممیز ثابت و ممیز شناور در نظرگرفته شوند، موضوع پیچیده‌تر می‌شود.

در سنین کودکی آموختیم که چگونه چهار عمل اصلی را در نمایش اندازه علامت دار انجام دهیم. این دانش هنگام پیاده‌سازی عمل‌ها با سخت‌افزار ارزشمند است. با این وجود طراح باید واقعاً با سلسله مراحل لازم برای انجام عمل و حصول نتیجه صحیح آشنا باشد. راه حل هر مسئله که بوسیله تعداد محدودی از رویه‌های متوالی بطور مناسب تعریف شده باشد الگوریتم [1] خوانده می‌شود. برای جمع دوعدد با ممیز ثابت و اعداد منفی در نمایش متمم 2 علامت‌دار، 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 علامتدار باشند، الگوریتم کمی پیچیده‌تر است و پیاده‌سازی آن به مدارهای جمع و تفریق، و مقایسه علامت‌ها و اندازه‌های 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 الگوریتم‌ها فلوچارت است. مراحل محاسباتی در فلوچارت در داخل کادرهای مستطیلی مشخص می‌شوند. مراحل تصمیم‌گیری در داخل کادرهای لوزی شکل نشان داده می‌شوند که از آن‌ها دو یاچند مسیر مختلف جدا می‌شود.

در این فصل الگوریتم‌های حسابی متعددی را بدست می‌آوریم و رویه‌ای را برای پیاده‌سازی آنها با سخت فزار دیجیتال نشان خواهیم داد. ما برای داده‌های زیر، جمع، تفریق، ضرب و تقسیم را بررسی خواهیم کرد:

۱- داده دودویی ممیز ثابت با نمایش مقدار علامت دار.

۲- داده دودویی ممیز ثابت با نمایش متمم 2 علامت‌دار.

۳- داده ممیز شناور دودویی

۴- داده دهدهی کد شده با دودویی

۲-۱۰ جمع و تفریق

همانطورکه در بخش ۳-۳گفته شد، سه راه برای نمایش اعداد دودویی ممیز ثابت منفی وجود دارد: اندازه علامت‌دار، متمم 1 علامت‌دار، یا متمم 2 علامت‌دار. بسیاری ازکامپیوترها وقتی اعمال حسابی با اعداد صحیح را انجام می‌دهند از نمایش متمم 2 استفاده می‌کنند. برای اعمال ممیز شناور، اکثر کامپیوترها نمایش اندازه علامت دار رابرای مانتیس بکار می‌برند. در این بخش ما الگوریتم‌های جمع و تفریق را برای داده‌های نمایش داده شده با مقدار علامت دار و مجدداً برای نمایش متمم 2 علامت دار بدست می‌آوریم.

توجه به این نکته مهم است که بدانیم نمایش انتخابی برای اعداد منفی به نمایش عددها در ثبات‌ها، قبل و بعداز اجرای عمل حسابی مربوط است. این بدان معنی نیست که ازحساب متمم نمی‌توان در مرحله میانی استفاده کرد. مثلاً در تفریق با اعداد درنمایش اندازه علامت دار، استفاده از حساب متمم ساده‌تر 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 نمی‌دهد.

1- Algorithm

جمع و تفریق با داده‌های اندازه علامت دار

نمایش اعداد بصورت اندازه علامت دار آشناست زیرا در محاسبات ریاضی روزمره بکار می‌رود. رویه جمع یا تفریق دو عدد دودویی علامت دار با قلم و کاغذ ساده و سرراست است. مروری براین رویه در بدست آوردن الگوریتم سخت افزاری می‌تواند مفید باشد.

مااندازه دو عدد را با A و B نشان می‌دهیم. وقتی که اعداد علامت دار، جمع یا تفریق شوند، بسته به علامت و عمل انجام شده، هشت حالت برای بررسی وجود دارد. این حالات در اولین ستون جدول ۱-۱۰ لیست شده‌اند. سایر ستون‌ها در جدول، اعمالی را که باید برروی اندازه اعداد انجام شود نشان می‌دهد. آخرین ستون برای جلوگیری از صفر منفی لازم است. به بیان دیگر وقتی دو عدد مساوی از یکدیگر تفریق شوند، نتیجه باید 0+ باشد نه 0−.

الگوریتم جمع و تفریق از جدول بدست می‌آیند و می‌توان آنها را به صورت زیر بیان کرد (کلمات داخل پرانتز باید برای الگوریتم تفریق بکار روند):

الگوریتم جمع (تفریق): وقتی که علامت A و B یکسان (مختلف) باشند، دو اندازه را باهم جمع کرده و علامت A را به حاصل ملحق کنید. اگر علامت A و B مختلف (یکسان) باشند،اندازه‌ها را باهم مقایسه و عدد کوچکتر را از بزرگتر تفریق کنید. اگر A>B است، علامت A را برای جواب برگزینید ولی اگر A<B است علامت A را مکمل نمائید. اگر دو اندازه باهم مساویند، B را از A تفریق کرده و علامت حاصل را مثبت بگذارید.

این دو الگوریتم جز در مقایسه علامت‌ها مشابهند. رویه‌ای که باید برای علامت‌های یکسان در رویه جمع دنبال شوند مشابه رویه برای علامت‌های مختلف در تفریق است و بالعکس.

پیاده سازی سخت افزاری

برای پیاده سازی دو عمل حسابی با سخت افزار ابتدا لازم است هر دو عدد در ثبات‌ها ذخیره شوند.

جدول ۱-۱۰ جمع و تفریق اعداد اندازه علامت‌دار

عمل	جمع اندازه‌ها	تفریق اندازه‌ها		
		وقتی $A > B$	وقتی $A < B$	وقتی $A = B$
$(+A) + (+B)$	$+(A + B)$			
$(+A) + (-B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(-A) + (+B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$
$(-A) + (-B)$	$-(A + B)$			
$(+A) - (+B)$		$+(A - B)$	$-(B - A)$	$+(A - B)$
$(+A) - (-B)$	$+(A + B)$			
$(-A) - (+B)$	$-(A + B)$			
$(-A) - (-B)$		$-(A - B)$	$+(B - A)$	$+(A - B)$

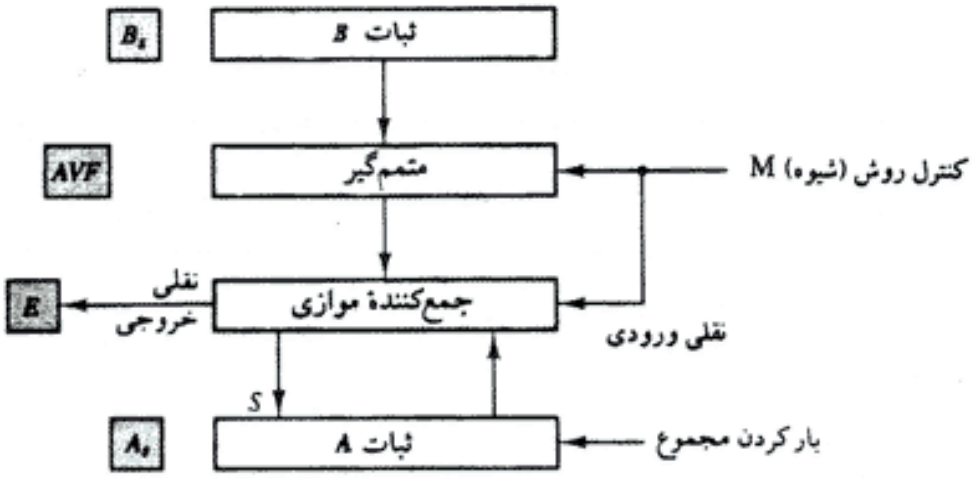
اجازه بدهید تا ثبات‌های A و B مقدار اعداد را ذخیره کنند و A_s و B_s فلیپ فلاپ‌های نگهدارنده علامت‌آنها باشند. نتیجه عمل ممکن است به ثبات سومی انتقال یابد: با این وجود، اگر نتیجه به A و A_s منتقل شود صرفه‌جویی انجام شده است. بنابراین A و A_s بالاتفاق انباره را می‌سازند.

اکنون پیاده سازی سخت افزار الگوریتم‌های فوق را ملاحظه کنید. اول، یک جمع کننده موازی لازم است تا ریزعمل A+B انجام شود. دوم، یک مدار مقایسه گر لازم است تا A>B، A=B یا A<B را مشخص نماید. سوم، دو مدار تفریق گر موازی برای انجام A−B یا B−A لازم است. رابطه علامت‌ها را می‌توان با استفاده از OR انحصاری با ورودی‌های A_s و B_s معین کرد.

رویه به یک مقایسه‌گر، یک جمع کننده و دو تفریق گر نیاز دارد. با این وجود رویه متفاوتی می‌توان یافت که امکانات کمتری نیاز داشته باشد. اول، می‌دانیم که تفریق را می‌توان با جمع و متمم انجام داد. دوم، نتیجه مقایسه می‌تواند از رقم نقلی انتهایی، پس از تفریق مشخص گردد. بررسی‌های دقیقتر روش‌های دیگر نشان می‌دهد که استفاده از متمم 2 برای تفریق و مقایسه رویه بهتری است زیرا فقط یک جمع کننده و یک متمم کننده نیاز دارد.

شکل ۱-۱۰ بلاک دیاگرام سخت افزار مربوط به پیاده سازی اعمال جمع و تفریق می‌باشد. این دیاگرام از ثبات‌های A و B و فلیپ فلاپ‌های A_s و B_s برای علامت تشکیل شده است. تفریق بوسیله جمع A با متمم 2 عدد B انجام می‌شود. رقم نقلی خروجی به فلیپ فلاپ E منتقل می‌گردد و درآن جا برای تعیین اندازه‌های نسبی اعداد چک می‌شود. فلیپ فلاپ جمع - سرریز، AVF، بیت سرریز را وقتی که A و B جمع شوند نگه می‌دارد. ثبات A سایر ریزاعمال راکه ممکن است هنگام مشخص کردن دنباله مراحل در الگوریتم نیاز داشته باشیم فراهم می‌سازد.

جمع A بعلاوه B بوسیله جمع کننده موازی انجام می‌شود. خروجی S (جمع) جمع کننده به ورودی ثبات A اعمال می‌شود. متمم ساز، خروجی B یا متمم B را، بسته به وضعیت



شکل ۱-۱۰ سخت افزار جمع و تفریق اندازه علامت‌دار

کنترل روش (شیوه)[1] M در اختیار می‌گذارد. متمم ساز از گیت‌های OR انحصاری و جمع کننده موازی متشکل از مدارات تمام جمع کننده مطابق شکل ۷-۴ 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 0، و خروجی جمع کننده برابر مجموع A+B می‌گردد. هنگامی که M=1 باشد، متمم 1 عدد B به جمع‌کننده اعمال، رقم نقلی ورودی 1 و خروجی $S=A+\overline{B}-1$ خواهد شد. این عمل برابر با جمع A با متمم 2 عدد B است، که معادل تفریق A−B است.

الگوریتم سخت افزاری

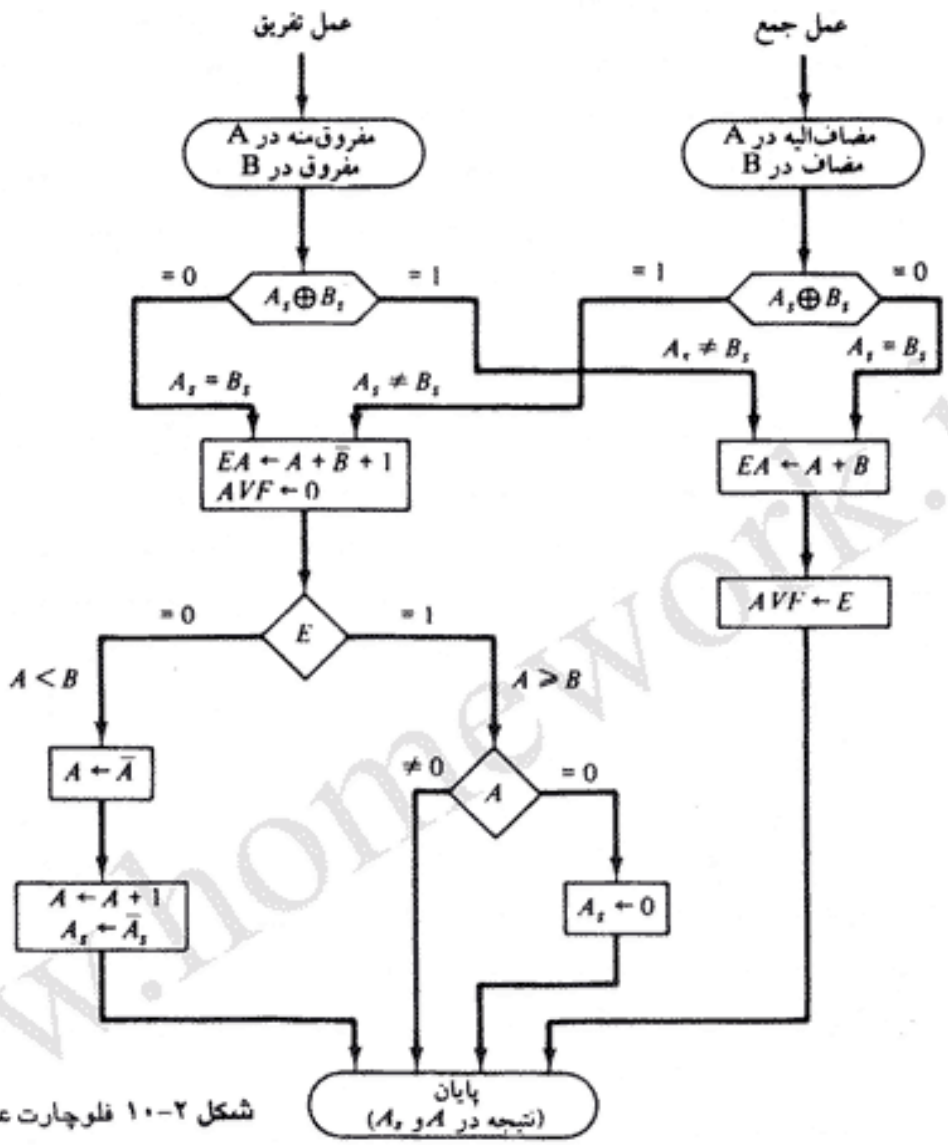
فلوچارت الگوریتم سخت افزاری در شکل ۲-۱۰ نشان داده شده است. دو علامت A_s و B_s بوسیله گیت OR انحصاری مقایسه می‌شوند. اگر خروجی گیت 0باشد، علامت‌ها یکی هستند؛ اگر 1 باشد علامت‌ها مختلفند. برای یک عمل جمع، علائم یکسان جمع اندازه‌ها را حکم می‌کند. برای عمل تفریق علائم متفاوت جمع اندازه‌ها را حکم می‌نماید. اندازه‌ها بوسیله ریزعمل A+B ← EA جمع می‌شوند، که EA ثباتی است که از ترکیب E و A بدست می‌آید. اگر رقم نقلی در E، پس از جمع برابر 1 باشد سرریز رخ داده است. مقدار E به فلیپ فلاپ جمع سرریز AVF منتقل می‌گردد.

اگر علامت‌ها در جمع مختلف باشند و یا در تفریق یکسان باشند عمل تفریق انجام می‌شود. اندازه‌ها با جمع A و متمم 2 عدد B ازیکدیگر تفریق می‌شوند. اگر اعداد از هم تفریق شوند هیچ سرریزی رخ نمی‌دهد بنابراین AVF برابر 0 خواهد شد. اگر A≥B باشد E برابر 1 است و عدد درون A نتیجه صحیحی است. اگر A برابر 0 باشد، علامت A_s باید مثبت گردد تا از داشتن صفر منفی جلوگیری شود. اگر E برابر 0 باشد A<B است. برای این حالت لازم است متمم 2 عدد A بدست آید. این عمل باریزعمل $A\leftarrow\overline{A}+1$ انجام می‌گیرد. با این وجود ما فرض خواهیم کرد که ثبات A مدارهای لازم برای متمم سازی و افزایش را داشته باشد، بنابراین متمم 2 ازاین دو عمل حاصل خواهد شد. در مسیرهای دیگر فلوچارت، علامت نتیجه همان علامت A است و لذا هیچ تغییری در A_s لازم نیست. معهذا، وقتی A<B باشد، علامت نتیجه متمم علامت اولیه A می‌باشد. درنتیجه لازم است A_s را برای بدست آوردن علامت صحیح متمم کنیم. نتیجه نهائی در A و علامت آن در A_s خواهد بود. مقدار در AVF بمعنی وقوع یک سرریز است، مقدار نهایی در E اهمیتی ندارد.

جمع و تفریق با داده‌های متمم 2 علامت دار

نمایش متمم 2اعداد همراه با الگوریتم‌های حسابی برای جمع و تفریق در بخش ۳-۳ معرفی شدند. دراینجا آنها را برای سهولت مراجعه بطور خلاصه تکرار کنیم. سمت چپ‌ترین بیت عدد، علامت را نشان می‌دهد: 0 برای مثبت و 1 برای منفی. اگر بیت علامت 1 باشد، تمام عدد به فرم متمم 2 نمایش

1- Mode Control



شکل ۲-۱۰ فلوچارت عمل‌های جمع و تفریق

داده می‌شود. بنابراین 33+ بصورت 00100001 و 33- بصورت 11011111 خواهند بود. توجه کنید که 11011111 متمم 2 عدد 00100001 می‌باشد و بالعکس.

در جمع دو عدد در فرم متمم 2 علامت‌دار، آن دو را بصورت عادی با هم جمع می‌کنیم و با بیتهای علامت آنها نیز مانند سایر بیتهای عدد رفتار می‌نمائیم. هر رقم نقلی حاصل از بیت علامت چشم پوشی می‌شود. برای تفریق ابتدا متمم 2 مفروق را بدست آورده و سپس آن را با مفروق منه جمع می‌کنیم.

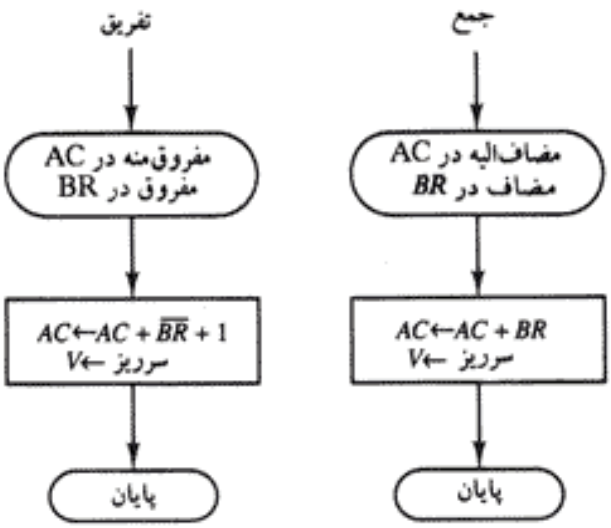
وقتی دوعدد n رقمی جمع شوند حاصل جمع n+1 رقم خواهد داشت و گوئیم که سرریز رخ داده است. اثر سرریز روی حاصل جمع اعداد متمم 2 علامت دار در بخش ۳-۳ توضیح داده شد. وقوع سرریز را می‌توان با بررسی دو نقلی آخر بوجود آمده در جمع تحقیق کرد. هرگاه این دو نقلی به یک گیت OR انحصاری اعمال شوند، در صورتی که خروجی این گیت 1 باشد سرریز رخ داده است.

آرایش ثبات‌ها برای پیاده‌سازی سخت افزار در شکل ۳-۱۰ نشان داده شده است. این شکل شبیه شکل ۱-۱۰ است جز اینکه بیت‌های علامت از بقیه ثبات‌ها جدا نشده‌اند. ما ثبات A را AC (انباره) و ثبات B را BR می‌نامیم. سمت چپ‌ترین بیت در AC و BR بیت علامت اعداد را نشان می‌دهند. دو بیت علامت همراه با سایر بیت‌ها در متمم ساز و جمع کننده موازی، جمع یا تفریق می‌شوند. فلیپ فلاپ سرریز V پس از رخداد سرریز برابر 1 می‌شود. در این حالت رقم نقلی خروجی در نظر گرفته نشده است.

الگوریتم جمع و تفریق دو عدد دودویی با نمایش متمم 2 علامت‌دار در فلوچارت شکل ۴-۱۰ نشان داده شده است. حاصل جمع از جمع محتوای AC و BR (همراه با بیت‌های علامت) بدست می‌آید. بیت سرریز V هنگامی 1 است که OR انحصاری دو رقم نقلی آخر 1 باشند، در غیراینصورت 0 است. عمل تفریق از جمع محتوای AC با متمم 2 محتوای BR حاصل می‌گردد. گرفتن متمم 2 عدد BR بمعنی تبدیل عدد مثبت به منفی است، و بالعکس. در حین این عمل سرریز باید وارسی گردد زیرا



شکل ۳-۱۰ سخت‌افزار جمع و تفریق متمم 2 علامت‌دار



شکل ۴-۱۰ الگوریتم جمع و تفریق اعداد متمم 2 علامت‌دار

دو عددی که با هم جمع شده‌اند ممکن است هم علامت باشند. برنامه نویس باید بداند که اگر سرریز رخ دهد، نتیجه غلطی در AC بدست خواهد آمد.

با مقایسه این الگوریتم با معادل آن برای اعدادی که به شکل مقدار (اندازه) علامت دار نشان داده شده‌اند متوجه می‌شویم که اگر اعداد منفی را بفرم متمم 2 علامت دار نگهداریم عمل جمع و تفریق اعداد ساده‌تر خواهد بود. با این دلیل در اکثرکامپیوترها این روش بجای روش آشناتر مقدار علامت‌دار بکار می‌رود.

۳-۱۰ الگوریتم ضرب

ضرب دو عدد ممیز ثابت دودویی با نمایش مقدار علامت دار با قلم وکاغذ، توسط فرآیند شیفت‌های متوالی و جمع انجام می‌شود. این روش را با مثالی می‌توان بهتر تشریح نمود.

```
23      10111    مضروب
19    × 10011    مضروب فیه
        10111
       10111
      00000   +
     00000
    10111
437 110110101    حاصلضرب
```

فرآیند بدین صورت است که بیت‌های مضروب فیه متوالیاً، با شروع ازکم ارزشترین بیت نظاره شوند. اگر بیت مضروب فیه 1 باشد، مضروب در پائین کپی می‌شود، درغیراینصورت 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 یابد. اولاً، در عوض تهیه ثبات‌هایی برای ذخیره و جمع همزمان چند عدد دودویی به تعداد بیت‌های مضروب فیه، بهتر است جمع کننده‌ای برای جمع فقط دو عدد دودویی در نظر گرفته شود و مرتباً حاصلضرب‌های جزیی[1] را در یک ثبات نگهداری 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

1- Partial Product

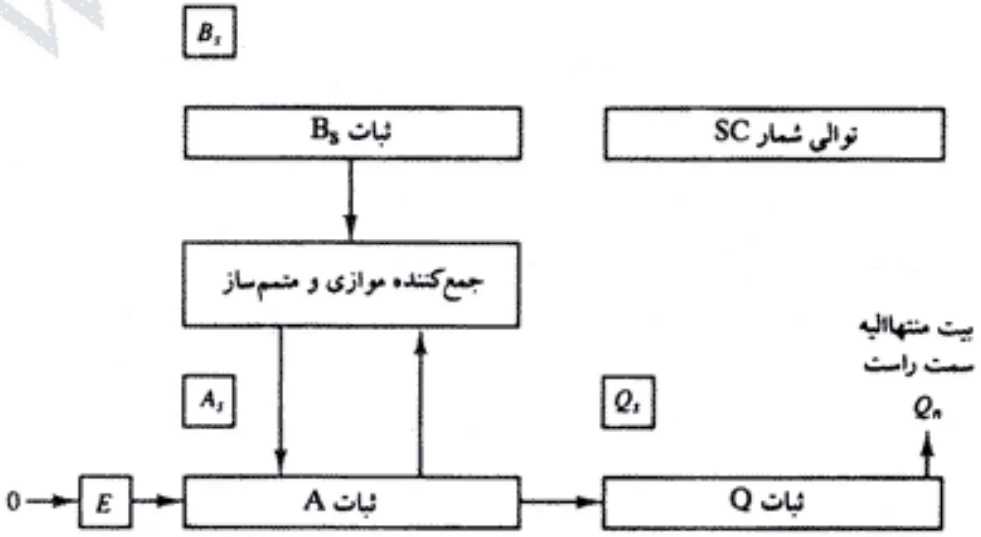
جمع کنیم زیرا مقدار آن را عوض نمی کند.

سخت افزار ضرب از امکانات شکل ۱-۱۰ بعلاوه دو ثبات اضافی دیگر تشکیل شده است. این ثبات به همراه ثبات‌های A و B در شکل ۵-۱۰ نشان داده شده‌اند. مضروب فیه در ثبات Q و علامتش در Q_s ذخیره می‌شود. توالی شمار SC ابتداً با عددی برابر با تعداد بیت‌های مضروب فیه مقداردهی می‌شود. شمارنده پس ازهربار تشکیل حاصلضرب جزیی یک واحد کم می‌شود. وقتی که محتوای شمارنده به صفر برسد، حاصلضرب تکمیل و فرآیند متوقف می‌گردد.

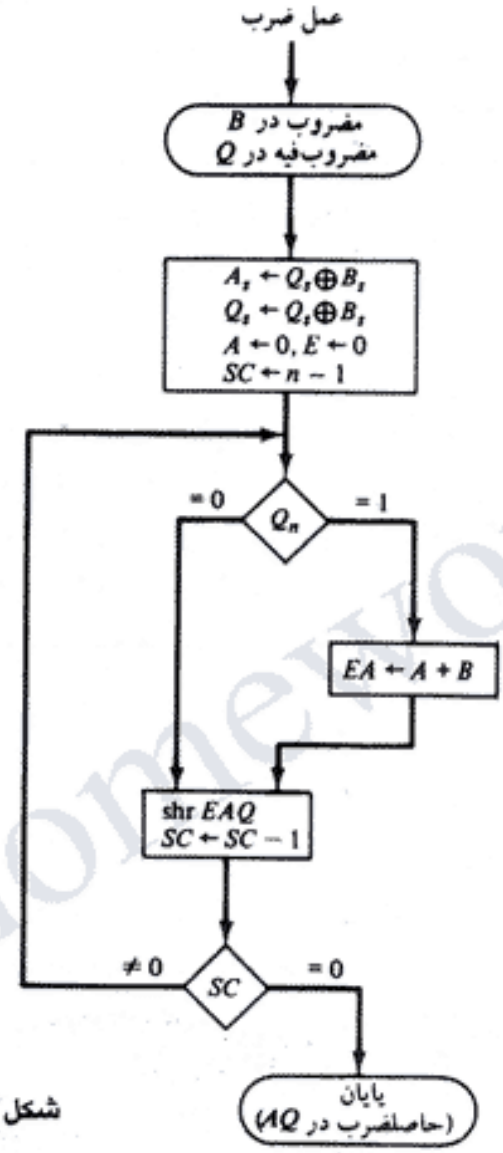
در آغاز مضروب در B و مضروب فیه در Q قرار دارد. حاصل جمع A و B تشکیل حاصلضرب جزیی را می‌دهند که به ثبات EA منتقل می‌گردد. حاصلضرب جزیی و مضروب فیه تواماً به راست شیفت داده می‌شوند. این شیفت توسط عبارت shr EAQ نشان داده شده است که شیفت به راست را در شکل ۵-۱۰ مشخص می‌نماید. بیت کم ارزشتر A به با ارزشترین بیت Q منتقل می‌شود، بیت E به با ارزشترین بیت A نقل مکان می‌یابد، و یک 0 ولرد E می‌شود، یک بیت از حاصلضرب جزیی به داخل Q شیفت کرده و بیت‌های مضروب فیه را یک واحد به سمت راست می‌رانند. باین ترتیب، سمت راست‌ترین فلیپ فلاپ در ثبات Q، که با Q_n مشخص شده است، حاوی بیتی از مضروب فیه خواهد بود که در مرحله بعدی باید بررسی شود.

الگوریتم سخت افزاری

شکل ۶-۱۰ فلوچارت الگوریتم سخت افزاری ضرب را نشان می‌دهد. ابتداً مضروب در B و مضروب‌فیه در Q قرار دارد. علامت‌های مربوطه به ترتیب در B_s و Q_s می‌باشند. علامت‌ها با هم مقایسه شده و علامت هر دو ثبات A و Q برابر علامت حاصلضرب می‌شود، زیرا حاصلضرب نهایی با طول دو برابر در



شکل ۵-۱۰ سخت‌افزار عمل ضرب



شکل ۶-۱۰ 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 علموند $n-1$ بیتی خواهد بود.

پس از دادن مقدار اولیه، بیت کم ارزشتر مضروب فیه که در Q_n قرار دارد تست می‌شود. اگر این بیت برابر 1 باشد، مضروب فیه موجود در B با حاصلضرب 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

تکرارشده 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 فیه را می‌گیرد. حاصلضرب نهایی در هر دو ثبات A و Q واقع است بدین ترتیب که A بیت‌های با ارزشتر و Q بیت‌های کم ارزشتر را نگه می‌دارند.

مثال عددی قبلی در جدول ۲-۱۰ برای روشن شدن فرآیند ضرب سخت افزاری تکرار شده است. روند، مراحل مشخص شده در فلوچارت را دنبال می‌کند.

الگوریتم ضرب بوت

الگوریتم بوت رویه‌ای را برای ضرب اعداد دودویی در نمایش متمم 2 علامتدار ارائه می‌نماید. مبنای کار الگوریتم بر این اساس استوار است که رشته‌های 0 در مضروب فیه نیازی به جمع ندارند بلکه فقط جابجایی (شیفت) لازم دارند و رشته‌های 1 در مضروب فیه از بیت مرتبه 2^k تا بیت 2^m را می‌توان معادل $2^{k+1}-2^m$ تلقی کرد. مثلاً عدد دودویی 001110 (+14) دارای رشته‌های 1 از 2^1 تا 2^3 است، (m=1 , k=3). این عدد را می‌توان بصورت $2^{k+1}-2^m=2^4-2^1=16-2=14$ نوشت. بنابراین ضرب M×14 را، که در آن M مضروب و 14 مضروب‌فیه است، می‌توان به صورت $M\times2^4-M\times2^1$ انجام داد. لذا حاصلضرب با چهار بار شیفت مضروب به چپ و تفریق Mکه یکبار به چپ شیفت داده شده است بدست می‌آید.

همانند همه روشهای ضرب، الگوریتم بوت نیز به بررسی بیت‌های مضروب فیه و شیفت حاصلضرب جزیی نیاز دارد. قبل از شیفت، ممکن است مضروب طبق قواعد زیر با حاصلضرب جزیی

جدول ۲-۱۰ مثال عددی برای ضرب کننده دودویی

B = 10111 مضروب‌فیه	E	A	Q	SC
مضروب‌فیه در Q	0	00000	10011	101
Q_n=1، جمع B		10111		
اولین حاصلضرب جزیی	0	10111		
شیفت EAQ به راست	0	01011	11001	100
Q_n=1، جمع B		10111		
دومین حاصلضرب جزیی	1	00010		
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 B		10111		
پنجمین حاصلضرب جزیی	0	11011		
شیفت EAQ به راست	0	01101	10101	000
حاصلضرب نهایی در AQ برابر 0110110101است				

جمع می‌شود، از آن تفریق شود و یا حاصلضرب جزیی بلاتغییر باقی بماند.

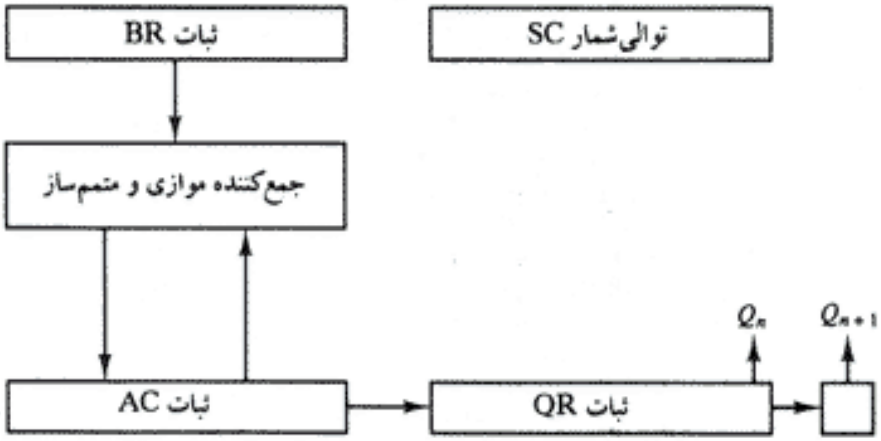
۱- به محض برخورد با اولین 1 کم ارزش در رشته 1 ها در مضروب فیه، مضروب از حاصلضرب جزیی کم می‌شود.

۲- به محض برخورد با اولین 0 (بشرطی که قبل از آن 1 باشد) در رشته‌ای از 0 ها در مضروب فیه، مضروب با حاصلضرب جزیی جمع می‌شود.

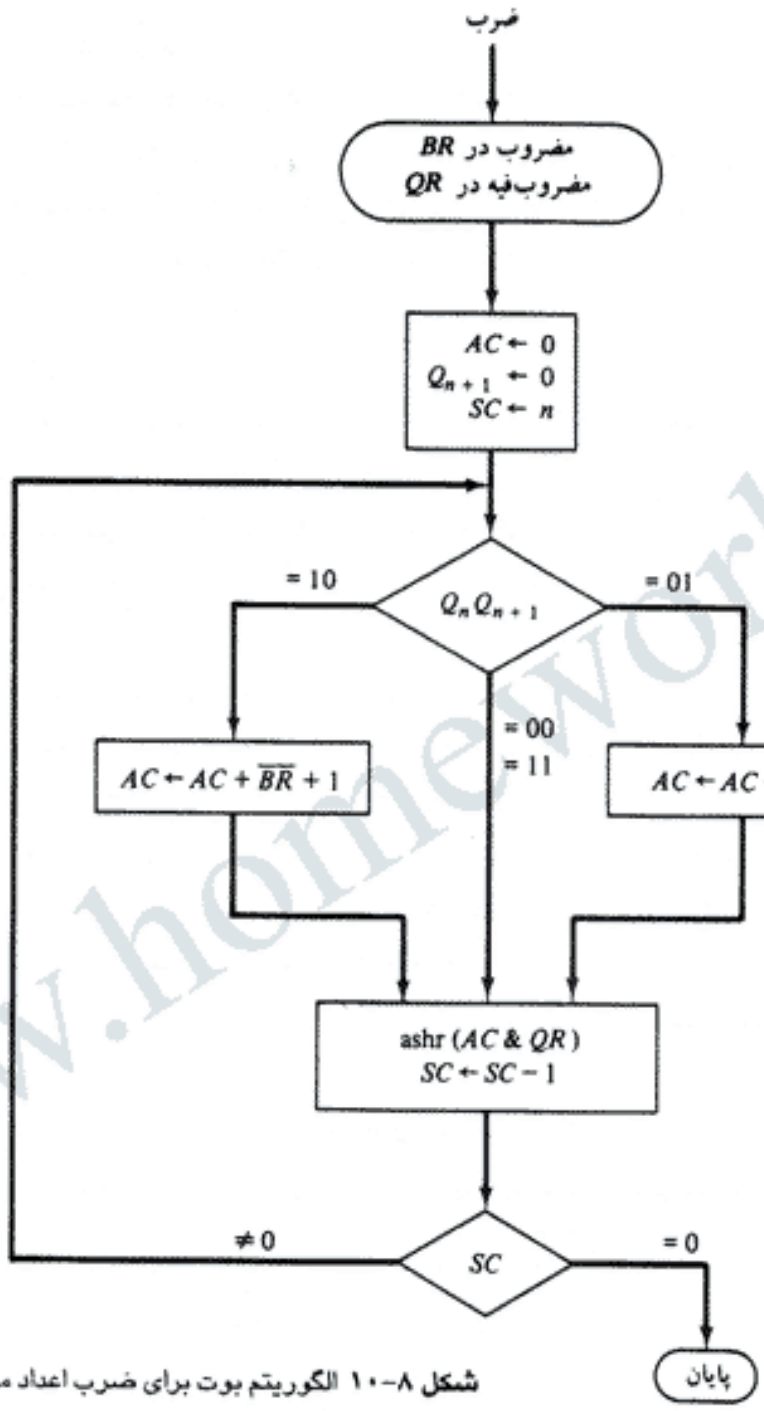
۳- وقتی که بیت جاری مضروب فیه همانند بیت قبلی باشد، حاصلضرب جزیی تغییری نمی‌کند.

الگوریتم فوق برای مضروب فیه‌های مثبت و یا منفی بفرم متمم 2 قابل استفاده است. این بدان علت است که مضروب فیه منفی به رشته‌ای از 1 ها خاتمه می‌یابد و آخرین عمل تفریق باوزن مناسب خواهد بود. مثلاً مضروب فیه 14- در نمایش متمم 2 عبارتست از 10010 و بصورت $-14 = -2^4+2^2-2^1$ با آن رفتار می‌شود.

پیاده‌سازی سخت افزار الگوریتم بوت آرایش ثبات شکل ۷-۱۰ را نیاز دارد. این شکل مشابه شکل ۵-۱۰ است جز اینکه بیت‌های علامت از بقیه ثبات‌ها تفکیک نشده‌اند. برای ملاحظه این اختلاف، ما ثبات‌های A و B و Q را بترتیب AC ، BR و QR می‌نامیم. Q_n کم ارزشترین بیت مفروق منه در ثبات QR است. یک فلیپ فلاپ اضافی Q_{n+1} برای بررسی همزمان دوبیت از مضروب‌فیه به QR ملحق شده است. فلوچارت الگوریتم بوت در شکل ۸-۱۰ نشان داده شده است. AC و بیت الحاقی ابتداً پاک شده‌اند و توالی شمار SC برابر تعداد بیت‌های مضروب فیه یعنی n ، می‌شود. دو بیت از مضروب فیه که در Q_n و Q_{n+1} قرار دارند مورد بررسی قرار می‌گیرند. اگر دو بیت برابر 10 باشند بدان معنی است که اولین 1 در رشته 1 ها فرا رسیده است. این حالت، تفریق مضروب از حاصلضرب جزیی موجود در AC را لازم می‌دارد. اگر دو بیت 01 باشد، مفهوم این است که بااولین 0 در رشته 0 ها برخورد شده است. این حالت جمع مضروب با حاصلضرب جزیی موجود در AC را لازم می‌دارد. هرگاه دو بیت برابر باشند



شکل ۷-۱۰ سخت‌افزار الگوریتم بوت



شکل ۸-۱۰ الگوریتم بوت برای ضرب اعداد متمم 2 علامت‌دار

حاصلضرب جزیی تغییری نمی‌کند. سرریز نمی‌تواند رخ دهد زیرا جمع و تفریق مضروب یک در میان انجام می‌شود. در نتیجه دو عددی که جمع می‌شوند، همواره علامت‌های مخالف دارند، و باین ترتیب امکان وقوع سرریز وجود نخواهد داشت. قدم بعدی شیفت حاصلضرب 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 بلاتغییر می‌ماند (بخش ۶-۴ ملاحظه شود). توالی شمار کاهش یافته

و حلقه محاسبه n بار تکرار می‌گردد.

مثال عددی الگوریتم بوت در جدول ۳-۱۰ برای n=5 نشان داده شده است. این مثال، ضرب $(-9)\times(-13)=+117$ را نشان می‌دهد. توجه کنید که مضروب فیه در QR منفی و مضروب نیز در BR منفی است. حاصلضرب 10 بیتی در AC و QR قرار داشته و مثبت است. مقدار نهایی Q_{n+1} علامت اولیه مضروب فیه است و نباید بخشی ازحاصلضرب تلقی شود.

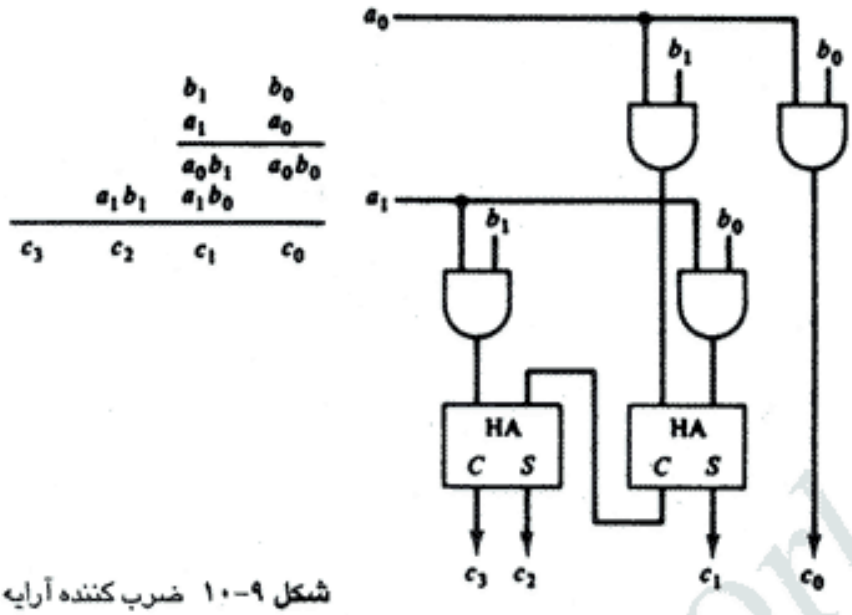
ضرب‌کننده آرایه‌ای

تست یک به یک بیت‌های مضروب فیه و تشکیل حاصلضرب جزیی یک عمل ترتیبی است که دنباله‌ای از جزء اعمال جمع و شیفت را نیاز دارد. ضرب دوعدد دودویی بوسیله یک ریزعمل و بکمک مدار ترکیبی که بیت‌های ضرب را یکباره ایجاد کند تشکیل می‌گردد. این روش راه سریعی برای ضرب دو عدد است زیراکل 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 بااین علت تا پیدایش مدارهای مجتمع اقتصادی تشخیص داده نشد.

برای اینکه بینیم چگونه یک ضرب کننده آرایه‌ای با یک مدار ترکیبی پیاده سازی می‌شود، ضرب دو عدد 2 بیت را مطابق شکل ۹-۱۰ در نظر بگیرید. بیت‌های مضروب b_1 و b_0 و بیت‌های مضروب فیه a_1 و a_0 و حاصلضرب $c_3c_2c_1c_0$ می‌باشند. اولین حاصلضرب جزیی با ضرب a_0 در b_1b_0 بدست می‌آید. ضرب دو بیت مانند a_0 و b_0 برابر 1 خواهد بود بشرطی که هر دو 1 باشند، در غیراینصورت 0 است. این با عمل AND یکسان است و می‌تواند با یک گیت AND پیاده سازی شود. همانطور که در شکل دیده می‌شود، اولین حاصلضرب جزیی با دو گیت AND حاصل می‌شود. دومین حاصلضرب جزیی از ضرب

جدول ۳-۱۰ مثال ضرب با الگوریتم بوت

Q_nQ_{n+1}	$BR = 10111$ $\overline{BR}+1 = 01001$	AC	QR	Q_{n+1}	SC
	مقدار اولیه	00000	10011	0	101
1 0	تفریق BR	01001			
		01001			
	ashr	00100	11001	1	100
1 1	ashr	00010	01100	1	011
0 1	جمع BR	10111			
		11001			
	ashr	11100	10110	0	010
0 0	ashr	11110	01011	0	001
1 0	تفریق BR	01001			
		00111			
	ashr	00011	10101	1	000



شکل ۹-۱۰ ضرب کننده آرایه 2 بیت در 2 بیت

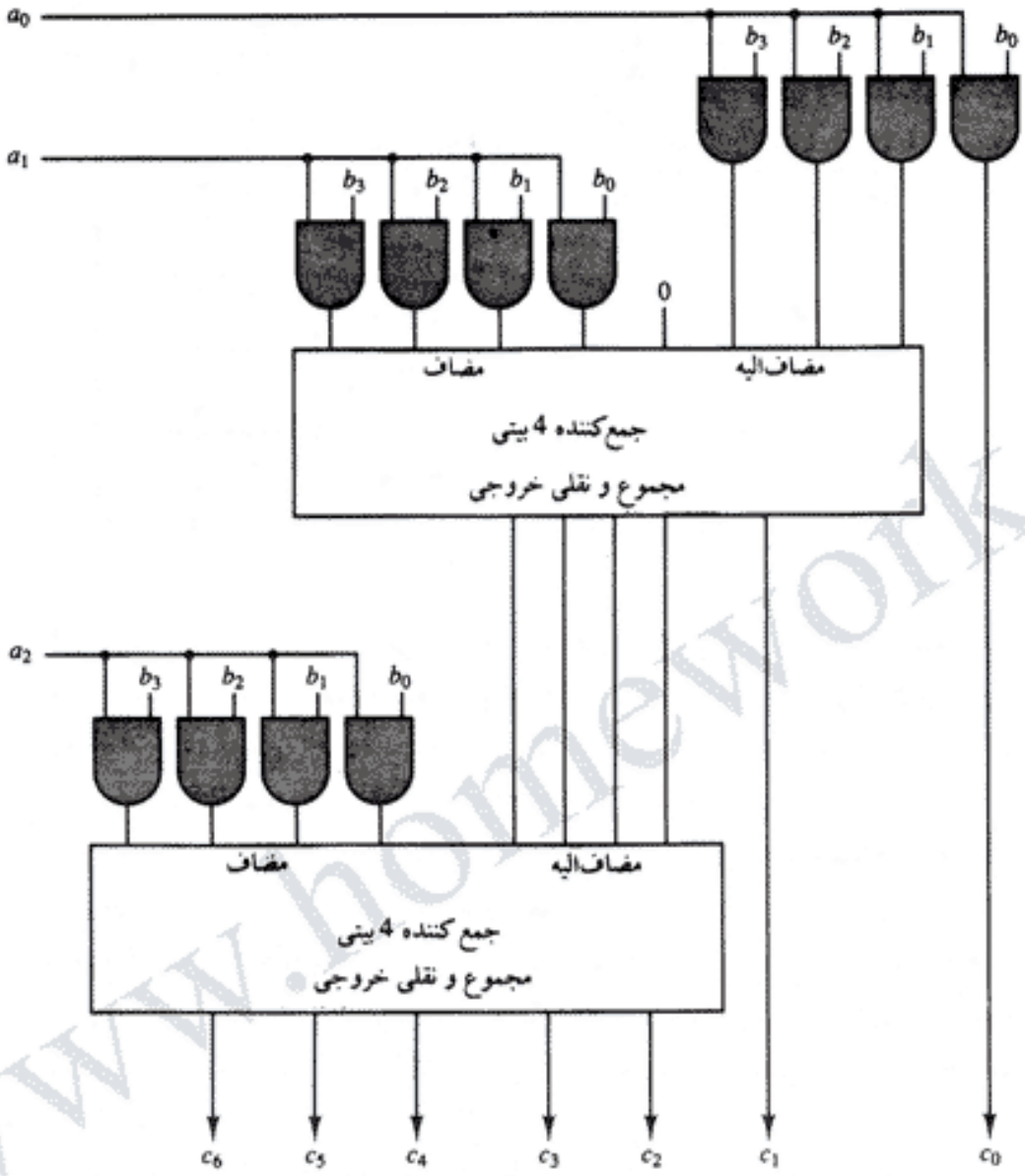
a_1 در b_1b_0 و شیفت آن به چپ بدست می‌آید. دو حاصلضرب جزیی بوسیله دو مدار نیم جمع‌کننده (HA) با یکدیگر جمع می‌شوند. معمولاً، تعداد بیت‌های حاصلضرب جزیی بیشتر است و لازم خواهد بود تا از تمام جمع کننده استفاده شود. دقت داشته باشید که کم ارزشترین بیت حاصلضرب لازم نیست وارد جمع کننده شود زیرا توسط خروجی گیت AND تشکیل می‌گردد.

یک مدار ضرب کننده دودویی با بیت‌های بیشتر را می‌توان بروشی مشابه ساخت. هر بیت از مضروب فیه با هربیت از مضروب، بتعداد بیت‌های مضروب‌فیه، 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 فیه j بیتی و مضروب k بیتی نیاز به jxk گیت AND و (j−1)k جمع کننده k بیتی برای تولید حاصلضرب j+k بیتی لازم است.

به عنوان مثال دوم، یک مدار ضرب کننده را در نظر بگیرید که یک عدد دودویی چهاربیتی را در یک عدد سه بیتی ضرب می‌کند. فرض کنید مضروب را با $b_3b_2b_1b_0$ و مضروب فیه را با $a_2a_1a_0$ نشان می‌دهیم. چون k=4 و J=3 است، 12 گیت AND و 2 جمع کننده 4 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

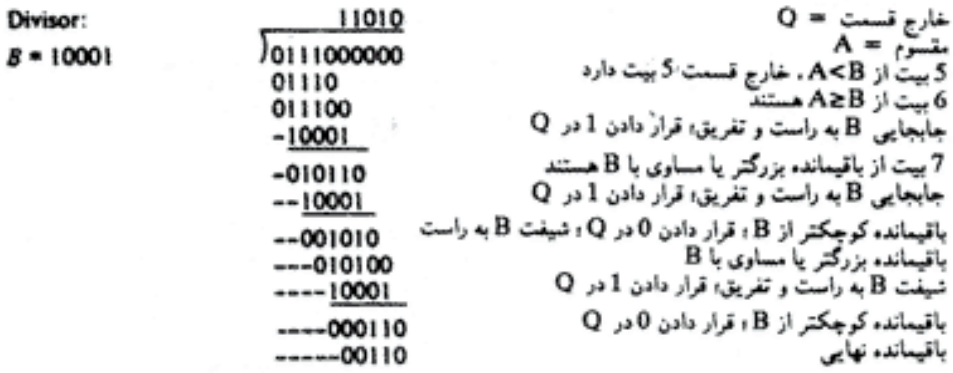
در تقسیم دو عدد ممیز ثابت در نمایش مقدار علامت‌دار با قلم و کاغذ، از اعمال مقایسه، شیفت و تفریق استفاده می‌شود. تقسیم دودویی از تقسیم دهدهی ساده تر است زیرا ارقام خارج قسمت 0 یا 1 هستند و نیازی نیست تا بدانیم مقسوم یا باقیمانده چند بار در مقسوم علیه جای می‌گیرد. فرآیند تقسیم با



شکل ۱۰-۱۰ ضرب آرایه 4 بیت در 3 بیت

مثال عددی شکل ۱۱-۱۰ توضیح داده شده است. مقسوم علیه B از پنج بیت و مقسوم A از ده بیت تشکیل شده است. پنج بیت با ارزشتر مقسوم با مقسوم علیه مقایسه می‌شوند. چون این پنج بیت از B کوچکتر است، مقایسه B را با شش بیت باارزشتر A آزمایش می‌کنیم. این عدد شش بیتی بزرگتر از B است، لذا رقم 1 را برای خارج قسمت در بالای بیت ششم مقسوم می‌نویسیم. سپس مقسوم علیه را یک بار به راست شیفت می‌دهیم و آن را از مقسوم کم می‌کنیم. تفاضل بدست آمده باقیمانده جزیی [1] 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



شکل ۱۱-۱۰ مثال تقسیم دودویی

باشیم. روند تقسیم با مقایسه باقیمانده جزیی با مقسوم‌علیه ادامه می‌یابد. اگرباقیمانده 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 داده‌های بانمایش مقدار علامت دار

هنگام پیاده‌سازی 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 ومتمم 2 عدد B انجام داد. نسبت اندازه‌ها از رقم نقلی انتهایی مشخص می‌گردد.

سخت افزار مربوط به پیاده‌سازی تقسیم همان سخت افزار ضرب است و از قطعات شکل ۵-۱۰ تشکیل شده است. در اینجا ثبات EAQ به چپ شیفت داده شده و 0 در Q_n وارد شده و مقدار قبلی E هم از دست می‌رود. مثال عددی مجدداً در شکل ۱۲-۱۰ برای روشن کردن روند تقسیم نشان داده شده است. مقسوم علیه در ثبات B و مقسوم که طولی دو برابر دارد در ثبات‌های 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 اندازه‌ها در E موجود است. اگر E=1 باشد بمعنی A≥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 A<B است و بنابراین خارج قسمت در Q_n برابر 0 (که هنگام شیفت وارد آن شده است) می‌ماند. سپس مقدار B مجدداً جمع می‌شود تا باقیمانده جزیی در A به مقدار قبلی‌اش بازگردانده شود. باقیمانده جزیی به چپ شیفت داده شده و روند مجدداً تکرار می‌گردد تا اینکه

مقسوم علیه: $B = 10001$, $\overline{B} + 1 = 01111$

	E	A	Q	SC
مقسوم:		01110	00000	5
shl EAQ	0	11100	00000	
جمع B+1		01111		
E = 1	1	01011		
Q_n=1 قرار دهید	1	01011	00001	4
shl EAQ	0	10110	00010	
جمع B+1		01111		
E = 1	1	00101		
Q_n=1 قرار دهید	1	00101	00011	3
shl EAQ	0	01010	00110	
جمع B+1		01111		
Q_n=0 ، E=0 بگذارید	0	11001	00110	
جمع B		10001		
				2
بازیابی باقیمانده	1	01010		
shl EAQ	0	10100	01100	
جمع B+1		01111		
E = 1	1	00011		
Q_n=1 قرار دهید	1	00011	01101	1
shl EAQ	0	00110	11010	
جمع B+1		01111		
Q_n=0 ، E=0 بگذارید	0	10101	11010	
جمع B		10001		
بازیابی باقیمانده	1	00110	11010	0
E را نادیده بگیرید				
باقیمانده در A		00110		
خارج قسمت در Q			11010	

شکل ۱۲-۱۰ 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 حالت 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 واگر مخالف باشند، علامت منفی است. علامت باقیمانده همانند علامت مقسوم است.

سرریز در تقسیم

عمل تقسیم ممکن است منجر به سرریز در خارج قسمت شود. هنگام استفاده از کاغذ و قلم، این امر مشکلی نیست ولی هنگام پیاده سازی عمل با سخت افزار مسئله ساز خواهد بود. دلیل این است که طول 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. ماازیک ثبات برای نگهداری

مقسوم علیه و دو ثبات برای نگهداری مقسوم استفاده می‌کنیم. با توجه به مثال شکل ۱۱-۱۰ می‌بینیم که خارج قسمت شش بیتی است بشرطی که پنج بیت باارزشتر مقسوم، عددی بزرگتر از مقسوم علیه باشد. خارج قسمت می‌باید دریک ثبات پنج بیتی استاندارد ذخیره شود، بنابراین بیت سرریز مستلزم یک فلیپ‌فلاپ اضافی برای ذخیره بیت ششم خواهد بود. در عملیات عادی کامپیوتر باید از بکارگیری این بیت سرریز خودداری شود زیرا در این صورت طول گل خارج قسمت بیش از مقداری خواهد بود که بتوان آن را به واحدهای حافظه‌ای که طول استاندارد را دارند، یعنی ثبات‌ها، منتقل کرد. در سخت افزار و یا نرم‌افزار کامپیوتر و یا ترکیبی از هر دو باید امکاناتی را جهت آشکارسازی این وضعیت فراهم کرد.

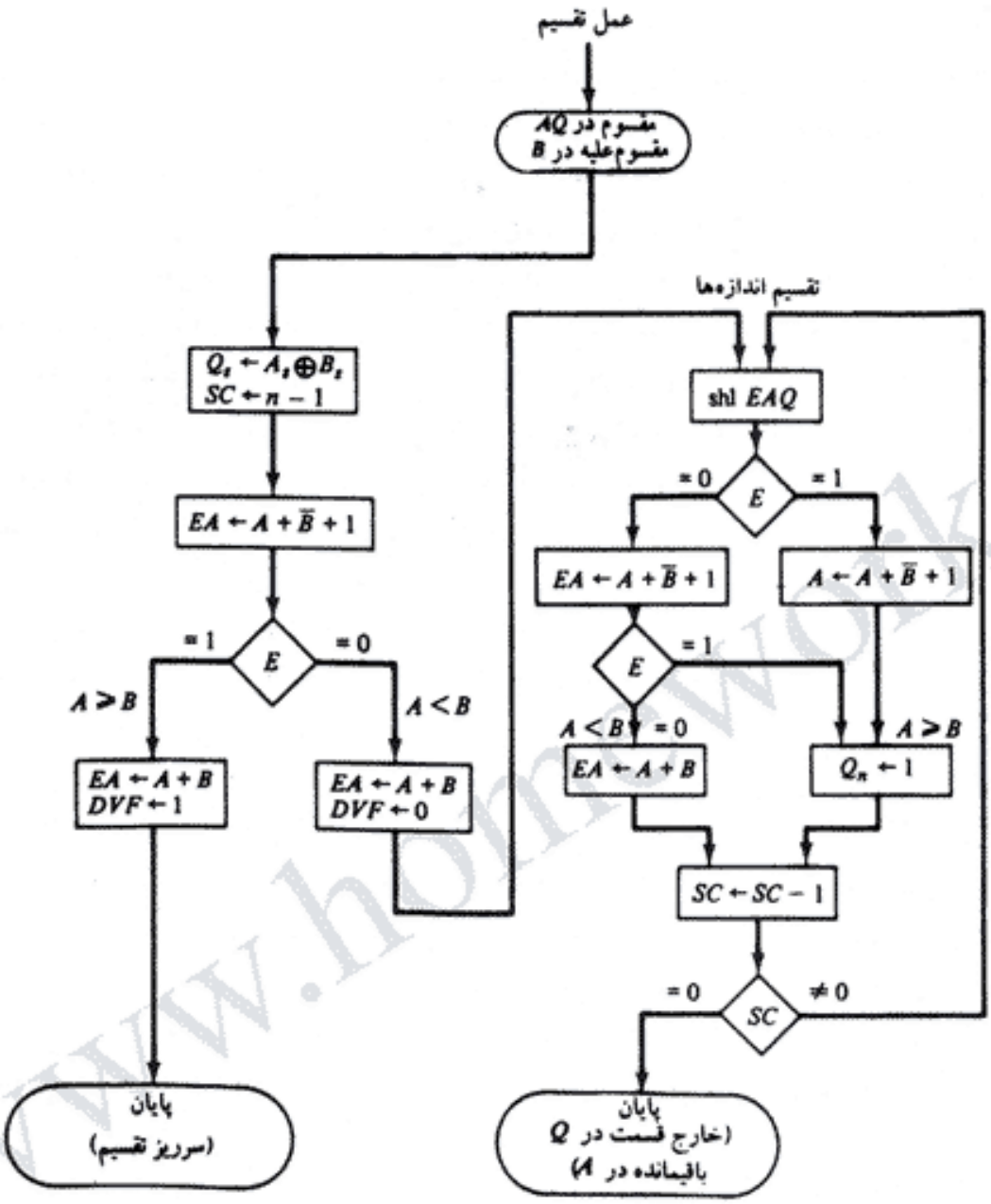
اگر طول مقسوم دوبرابر مقسوم علیه باشد، حالت سرریز را می‌توان بطریق زیر بیان کرد: حالت سرریز در تقسیم هنگامی اتفاق می‌افتد که بیت‌های نیمه پرارزشتر مقسوم، عددی بزرگتر یا مساوی مقسوم علیه را تشکیل دهند. مسئله دیگری که باید درتقسیم درنظر داشت این است که از تقسیم برصفر احتراز شود. حالت سرریز در تقسیم مراقبت از این وضعیت را بعهده دارد. این بدان علت است که اگر مقسوم علیه صفر باشد، مقسوم قطعاً از مقسوم علیه بزرگتر یا مساوی آن است. حالت سرریز معمولاً با 1شدن فلیپ فلاپ خاصی مشخص می‌گردد و آنرا فلیپ فلاپ سرریز خوانده و با DVF نشان می‌دهیم.

در قبال وقوع سرریز تقسیم بطرق مختلف می‌توان عکس العمل نشان داد. در بعضی کامپیوترها مسئولیت با برنامه نویس است تا پس از هر دستورالعمل تقسیم DVF را چک کند. در این صورت می‌توان به زیرروالی انشعاب کرده و اقدامی اصلاحی همچون تغییر مقیاس داده‌ها را برای جلوگیری از سرریز انجام داد. در برخی کامپیوترهای قدیمی‌تر، وقوع سرریز تقسیم سبب توقف کامپیوتر می‌شد و این حالت، توقف تقسیم نامیده می‌شد. امروزه متوقف کردن کار کامپیوتر توصیه نمی‌شود زیرا اتلاف کننده وقت است. روشی که در اکثر کامپیوترها انجام می‌شود تولید درخواست وقفه[1] بهنگام 1 شدن DVF است. این وقفه موجب می‌شود تا کامپیوتر اجرای برنامه جاری را بتعویق انداخته و به روال سرویس دهی برای اقدام اصلاحی مورد تقاضا انشعاب نماید. معمول‌ترین اقدام اصلاحی خارج شدن از برنامه و چاپ پیغام خطایی است که دلیل عدم امکان اتمام برنامه را شرح می‌دهد. از این پس، وظیفه کاربری که برنامه را نوشته این است که تا مقیاس داده‌ها را تغییر دهد و یا هر اقدام اصلاحی دیگری را بعمل آورد. بهترین راه جلوگیری از سرریز تقسیم استفاده از داده‌های ممیز شناور است. ما در بخش ۵-۱۰ خواهیم دید که سرریز تقسیم بسادگی بهنگام استفاده از اعداد ممیز شناور قابل پیش گیری است.

الگوریتم سخت افزاری

الگوریتم سخت افزاری سرریز در فلوچارت شکل ۱۳-۱۰ نشان داده شده است. مقسوم در A و Q و مقسوم علیه در B قرار دارد. علامت نتیجه به عنوان قسمتی از خارج قسمت به Q_s منتقل می‌شود. برای مشخص کردن تعداد بیت‌های خارج قسمت، عدد ثابتی در توالی شمار SC قرار داده می‌شود. همانند

1- Interrupt Request



شکل ۱۳-۱۰ فلوچارت عمل تقسیم

فرض می‌کنیم که عملوندها از واحد حافظه n بیتی به ثبات‌ها منتقل شوند. چون عملوند همراه با علامتش ذخیره می‌شود، یک بیت از کلمه حافظه بوسیله علامت و $n-1$ بیت بوسیله اندازه آن اشغال می‌گردد.

حالت سرریز تقسیم با تفریق مقسوم علیه در B از نیمی از بیت‌های ذخیره شده در A تست می‌شود. اگر $A \geq B$ باشد، فلیپ فلاپ سرریز تقسیم، DVF برابر 1 شده و عمل بطور ناقص متوقف می‌شود. اگر $A < B$ باشد، سرریزی رخ نداده و مقدار مقسوم با جمع B با A بازیافته می‌شود. تقسیم اندازه‌های دو عدد با جابجاکردن مقسوم موجود در AQ به چپ شروع می‌شود که در این جابجایی بیت پرارزشتر به E

منتقل می‌شود اگر بیت انتقال یافته به E برابر 1 باشد، درمی‌یابیم که EA>B است زیرا EA از رقم 1 و بدنبال آن $n-1$ بیت تشکیل شده است، در حالیکه B فقط $n-1$ بیت است. در این حالت، باید B از EA تفریق شود و مقدار 1 بعنوان یک بیت خارج قسمت در Q_n قرار می‌گیرد. چون ثبات A فاقد بیت بالا رتبه مقسوم است (که در E قراردارد) مقدار آن $EA-2^{n-1}$ خواهد بود. از جمع متمم دو B با این مقدار داریم:

$$(EA - 2^{n-1}) + (2^{n-1} - B) = EA - B$$

اگر بخواهیم E برابر 1 بماند رقم نقلی عبارت فوق به E منتقل نمی‌شود.

اگر عمل شیفت به‌چپ مقدار 0 را وارد E کند، مقسوم علیه با جمع متمم 2 آن تفریق می‌شود و مقدار رقم نقلی به E انتقال می‌یابد. اگر E=1 شود، A≥B است؛ بنابراین به Q_n، 1 داده خواهد شد. اگر E=0 باشد، A<B و عدد اولیه با جمع B با A بازیابی می‌شود. در مورد اخیر مقدار 0 را در Q_n باقی می‌گذاریم (مقدار 0 هنگام شیفت وارد آن شده است).

این روند مجدداً در حالی که ثبات A حاوی باقیمانده است تکرار می‌گردد. پس از $n-1$ تکرار، اندازه خارج قسمت در ثبات Q و باقیمانده در ثبات A خواهد بود. علامت خارج قسمت در Q_s و علامت باقیمانده نیز که همان علامت مقسوم است در A_s قرار دارد.

سایر الگوریتم‌ها

روش سخت افزاری که در فوق شرح داده شد روش بازیافتی[1] خوانده می‌شود. دلیل این نامگذاری این است که باقیمانده جزیی از جمع مقسوم علیه با تفاضل منفی بازیافته می‌شود. دوروش دیگر برای تقسیم اعداد وجود دارد، روش مقایسه‌ای و روش غیربازیافتی[2]. در روش مقایسه‌ای، A و B قبل از تفریق مقایسه می‌شوند. سپس اگر A≥B باشد B از A تفریق می‌شود. اگر A<B باشد کاری انجام نمی‌شود. باقیمانده جزیی به چپ شیفت داده شده و مجدداً مقایسه تکرار می‌گردد. می‌توان مقایسه را قبل از تفریق با وارسی رقم نقلی انتهایی جمع کننده موازی، قبل از انتقال آن به ثبات E انجام داد.

در روش غیربازیافتی، اگر اختلاف منفی باشد B جمع نمی‌شود، و در عوض تفاضل منفی به چپ شیفت داده شده و سپس با B جمع می‌گردد. برای اینکه ببینیم 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 می‌شود تا A بازیافته شود. در دور بعدی اجرای حلقه، این عدد به چپ شیفت داده شده (یا در 2 ضرب می‌شود) و دوباره B از آن کم می‌شود. نتیجه این اعمال $2(A-B+B) - B = 2A-B$ است. این نتیجه در روش غیربازیافتی بارها کردن A−B بهمان گونه که هست ، یعنی A−B، بدست می‌آید. در مرتبه بعدی اجرای حلقه، عدد به چپ شیفت داده شده و B به آن اضافه می‌شود تا 2(A−B)+B=2A−B بدست آید که همان نتیجه قبلی است. بنابراین، در روش غیربازیافتی، اگر مقدار

1- Restoring Method 2- Nonrestoring Method

قبلی Q_n برابر 1 باشد B تفریق می‌گردد، اما اگر مقدار قبلی Q_n برابر 0 باشد B جمع می‌شود و 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.

۵-۱۰ عمل های حسابی ممیز شناور

بسیاری از زبان‌های برنامه‌نویسی امکاناتی برای مشخص کردن عددهای ممیز شناور دارند. متداول‌ترین روش، مشخص کردن آنها با عبارت real (حقیقی) در برابر اعداد ممیز ثابت است که با عبارت Integer (صحیح) مشخص می‌گردند. هر کامپیوتری که دارای کامپایلر برای اینگونه زبانهای سطح بالا باشد باید امکاناتی برای انجام عمل‌های حسابی ممیز شناور داشته باشد. این عملیات غالباً در سخت افزار کامپیوتر پیاده سازی می‌شود. اگر سخت افزاری برای این اعمال وجود نداشته باشد، کامپایلری باید طراحی شود که دارای مجموعه‌ای از زیرروال‌های نرم افزاری ممیز شناور باشد. هرچند روش سخت افزاری گرانتر است، ولی کارآیی آن بقدری از روش نرم افزاری بیشتر است که در اکثر کامپیوترها، سخت افزار ممیز شناور کار گذاشته می‌شود و فقط در کامپیوترهای خیلی کوچک از آن صرف نظر می‌شود.

ملاحظات اساسی

نمایش ممیز شناور داده در بخش ۴-۳ معرفی شد. عدد ممیز درشباتهای کامپیوتر از دو بخش تشکیل می‌شود: مانتیس m و نمای e. این دو بخش نماینده عددی است که از ضرب m در r به نمای e بدست می‌آید. یعنی

$$m \times r^e$$

مانتیس ممکن است که عددی کسری یا صحیح باشد. محل ممیز و مقدار پایه r فرض تصور شده‌اند و در ثبات‌ها وارد نشده‌اند. مثلاً نمایش کسری و پایه 10 را در نظر بگیرید. عدد دهدهی 537.25 در یک ثبات با m=53725 و e=3 نمایش داده می‌شود و چنین تفسیر می‌گردد که نماینده عدد ممیز شناور زیر است:

$$0.53725 \times 10^3$$

اگر باارزشترین رقم مانتیس یک عدد ممیز شناور و غیرصفر باشد آن را نرمالیزه می‌کنیم. در نتیجه مانتیس دارای بیشترین تعداد ممکن رقم‌های معنی‌دار خواهد بود. صفر را نمی‌توان نرمالیزه کرد زیرا رقم غیرصفر ندارد. در نمایش ممیز شناور، صفر را با مقدار تمام 0 در مانتیس و نما نشان می‌دهیم.

نمایش ممیز شناور محدوده اعدادی راکه می‌توان در یک ثبات جای داد مشخص می‌کند. کامپیوتری با کلمه‌های 48 بیتی را درنظر بگیرید. چون یک بیت برای علامت رزرو شده است محدوده اعداد صحیح

ممیز ثابت $\pm(2^{47}-1)$ خواهد بود که تقریباً $\pm 10^{14}$ می‌باشد. از این 48 بیت می‌توان برای نمایش یک عدد میز شناور با 36 بیت برای مانتیس و 12 بیت برای نمااستفاده کرد. با فرض نمایش کسری برای مانتیس و با احتساب دوبیت برای علامت‌ها، محدوده اعدادی که می‌توان 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 $2^{11} - 1 = 2047$ می‌باشد، بدست می‌آید. بزرگترین عددی که می‌توان آن را جای داد 10^{615} است، که عدد بسیار بزرگی است. مانتیس می‌تواند 35 بیتی باشد (جدا از علامت) و اگر عدد صحیح تلقی شود می‌تواند تا $(2^{35}-1)$ را ذخیره نماید. این تقریباً برابر با 10^{10} می‌باشد، که معادل یک عدد دهدهی 10 رقمی است.

کامپیوترهایی که دارای طول کلمه‌های کوتاه‌تری هستند برای نمایش اعداد ممیز شناور از دو یا چند کلمه استفاده می‌نمایند. یک میکروکامپیوتر 8 بیتی ممکن است از چهار کلمه برای نمایش یک عدد ممیز شناور استفاده کند. یکی از کلمات 8 بیتی برای نما و 24 بیت سه کلمه دیگر برای مانتیس در نظر گرفته شده‌اند.

اعمال حسابی با اعداد ممیز شناور بسیار پیچیده‌تر از اعداد ممیز ثابت است و اجرای آنها زمان بیشتری می‌برد و سخت‌افزار پیچیده‌تری نیاز دارد. جمع یا تفریق دو عدد مستلزم معین کردن محل ممیز است زیرا قبل از جمع یا تفریق مانتیس‌ها، باید بخش نمای دو عدد با هم مساوی شود. هم ردیف کردن با جابجایی یکی ازمانتیس‌ها و تنظیم نمای آن تا جایی که با دیگری برابر شود انجام می‌شود. جمع اعداد ممیز شناور زیر را در نظر بگیرید.

$$\begin{array}{r} 0.5372400 \times 10^{2} \\ + \ 0.1580000 \times 10^{-1} \end{array}$$

لازم است که دونما قبل از جمع مانتیس‌ها باهم مساوی شوند. می‌توانیم عدد اول را سه مکان به چپ، یا عدد دوم را سه مکان به راست شیفت دهیم. وقتی که مانتیس‌ها در ثبات‌ها ذخیره شوند، شیفت به چپ سبب از دست دادن ارقام با ارزشتر می‌شود. روش دوم ترجیح دارد چون میزان دقت راکاهش می‌دهد در حالی که روش اول ممکن است موجب بروز خطا شود. رویه هم ردیف کردن معمولاً با این ترتیب است که مانتیسی راکه نمای کوچکتر دارد باندازه اختلاف بین نماها به راست شیفت دهیم. پس از این عمل مانتیس‌ها می‌توانند 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 دومانتیس نرمالیزه شده باهم جمع شوند حاصل 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 بالارتبه باشد.

$$\begin{array}{r} 0.56780 \times 10^5 \\ -\ 0.56430 \times 10^5 \\ \hline 0.00350 \times 10^5 \end{array}$$

وقتی عدد میز شناوری دارای 0 در مکان‌های با ارزشتر باشد گوئیم فروریز[1] دارد. برای نرمالیزه کردن اعداد فروریز دار، لازم است تا مانتیس به چپ شیفت داده شود و از نما یک واحد کسر گردد تا رقم غیرصفر در اولین مکان ظاهر شود. در مثال فوق، لازم است مانتیس را دوبار به چپ شیفت دهیم تا 0.35000×10^3 حاصل شود. در بسیاری از کامپیوترها رویه نرمالیزه کردن پس از هر عمل صورت می‌گیرد تا از نرمالیزه شدن نتایج اطمینان حاصل شود.

ضرب و تقسیم ممیز شناور نیازی به هم‌ردیف کردن مانتیس‌ها ندارند. حاصلضرب با ضرب دو مانتیس و جمع نماها شکل می‌گیرد. عمل تقسیم از تقسیم دو مانتیس و تفریق نماها بدست می‌آید.

اعمال اجرا شده با مانتیس‌ها 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 هم‌ردیفی مانتیس‌ها)، 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 علامت‌دار، متمم 2 علامت‌دار و متمم 1 علامت‌دار.

چهارمین روشی که در بسیاری از کامپیوترها 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 نماها در درون مثبت خواهند بود. مثال زیر می‌تواند این نوع نمایش را روشن سازد. فرض کنید نماها از 50−تا 49+ متغیر باشند. درونا نما پس از جمع با بایاس با دو رقم نشان داده می‌شود. (بدون علامت). ثبات نما حاوی عدد e+50 خواهند بود، که e نمای واقعی است. لذا نماها در ثبات بصورت اعداد مثبتی از 00 تا 99 نمایش داده خواهند شد. نماهای مثبت در ثبات‌ها، محدوده اعداد 99 تا 50 را دارند. تفریق 50 از آنها مقادیر مثبت 0 تا 49 را می‌دهد. نماهای منفی دو ثباتها محدوده 49 تا 00 را دارند. تفریق 50 مقدار منفی 1− تا 50− را می‌دهد.

مزیت استفاده از نماهای 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 مانتیس‌ها می‌توان از مقایسه‌گر مقدار (اندازه) برای مقایسه اندازه‌های آنها استفاده کرد. مزیت دیگر آنها این است که کوچکترین نمای پایین ممکن، تمام 0 است. در این صورت نمایش ممیز شناور 0

1- Underflow

متشکل از مانتیس 0 و کوچکترین نمای ممکن خواهد بود.

در مثال‌های بالا، از عددهای دهدهی برای نشان دادن برخی از مفاهیمی که هنگام کار بار 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 دهدهی در کامپیوتر در بخش بعدی صحبت خواهیم کرد.

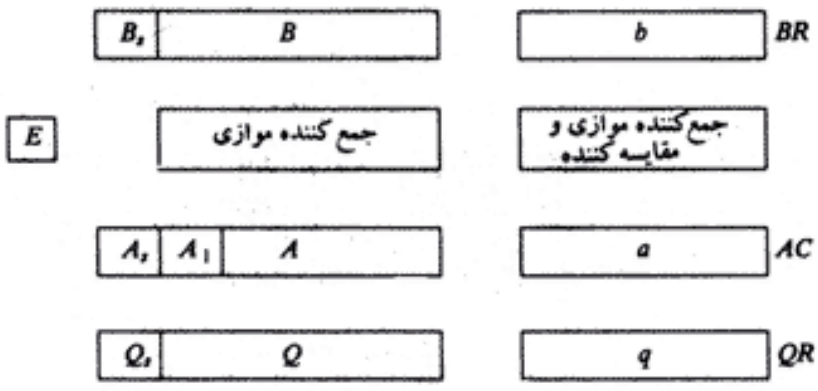
آرایش ثبات‌ها

آرایش ثبات‌ها برای اعمال ممیز شناور کاملاً مشابه اعمال ممیز ثابت است. قاعده کلی این است که همان ثبات‌ها و جمع کننده برای پردازش مانتیس بکار می‌روند. اختلاف موجود بر سر کار با نماهاست.

سازمان ثبات‌ها برای اعمال ممیز شناور در شکل ۱۴-۱۰ نشان داده شده است. در این شکل سه ثبات BR، AC و QR وجود دارد.هر ثبات به دو زیر بخش تقسیم شده است. بخش مانتیس همان سمبل‌های حروف بزرگ نمایش ممیز ثابت را دارد. بخش نما از سمبل حروف کوچک متناظر استفاده می‌نماید.

فرض براین است که هر عدد ممیز شناور دارای یک مانتیس با نمایش مقدار علامت‌دار و یک نمای بایاس شده باشد. بنابراین AC دارای مانتیسی است که علامتش در A_s 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، که A_1 می‌باشد، را بطور جداگانه نشان می‌دهد. برای نرمالیزه شدن این عدد بیت مذکور باید 1 باشد. دقت کنید که سمبل AC نماینده کل ثبات است، یعنی زنجیره a، A و A_s

بطور مشابه، ثبات BR به B_s، B و b و QR به Q_s، Q و q تقسیم می‌شوند. یک جمع‌کننده موازی دومانتیس را جمع می‌کند و مجموع را به A ونقلی را به E انتقال می‌دهد. جمع کننده موازی جداگانه‌ای برای نماها بکار رفته است. چون نماها بایاس شده‌اند، علامت متمایزی ندارند ولی بصورت یک کمیت مثبت بایاس شده نشان داده شده‌اند. فرض براین است که اعداد ممیز شناور آنقدر بزرگ باشند که شانس



شکل ۱۴-۱۰ ثبات‌های مربوط به عملیات حسابی ممیز- شناور

وقوع سرریز بسیار کم باشد، و به این دلیل سرریز نما چشم پوشی می‌گردد. نماها به یک 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 ممیزشناور که از حافظه بیایند و یا به حافظه بروند همواره نرمالیزه هستند.

جمع و تفریق

درحین 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.

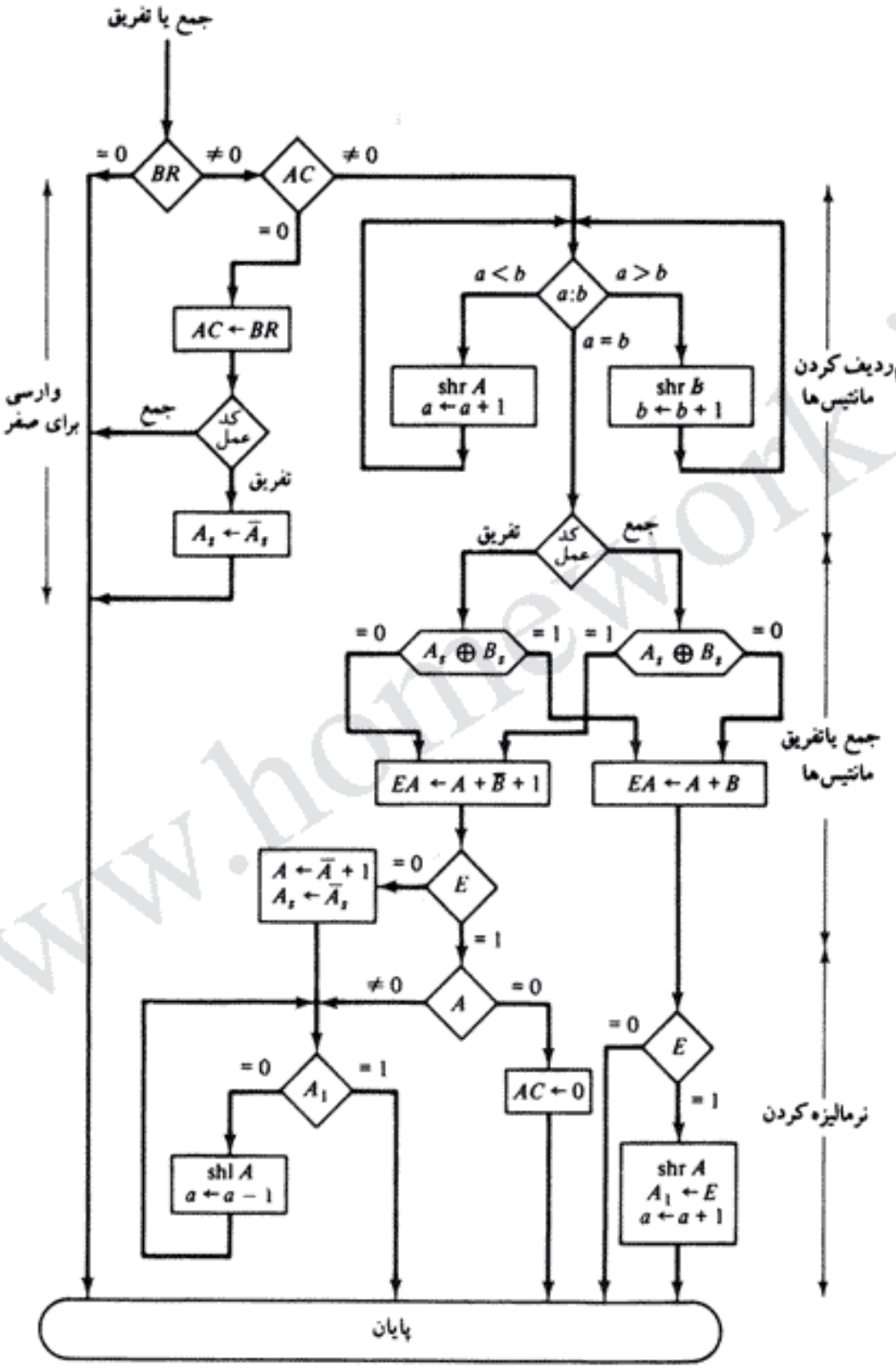
۱- تست صفربودن عملوندها
۲- هم ردیف کردن مانتیس‌ها
۳- جمع یا تفریق مانتیس‌ها
۴- نرمالیزه کردن نتیجه

عدد ممیز شناوری که صفرباشد قابل نرمالیزه شدن نیست. اگر این عدد طی محاسبات مورد استفاده قرار گیرد، نتیجه نیز ممکن است صفرشود. 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، نیتجه ممکن است غیرنرمالیزه باشد. با بکارگیری رویه نرمالیزه کردن، نتیجه قبل از انتقال به حافظه نرمالیزه می‌شود.

فلوچارت جمع یا تفریق دوعدد دودویی ممیز شناور در شکل ۱۵-۱۰ نشان داده شده است. اگر BR برابر صفر باشد، عمل متوقف می‌شود، و مقدار موجود در AC نتیجه عمل خواهد بود. اگر AC صفرباشد، محتوای BR را به AC منتقل می‌کنیم و اگر قرار باشد اعداد از هم تفریق شوند علامت آنرا متمم می‌کنیم. اگر هیچیک از اعداد صفر نباشند اقدام به هم ردیف کردن مانتیس‌ها می‌نمائیم.

مقایسه‌گر مقدار که به نماهای a و b مربوط است سه خروجی تولید می‌کند تا نسبت اندازه‌ها را مشخص نماید. اگر دو نما مساوی باشند، اجرای عملیات حسابی صورت می‌گیرد. اگر نماها مساوی نباشند مانتیسی که نمای کوچکتری دارد به راست شیفت داده شده و نمای آن افزایش می‌یابد. این روند تا مساوی شدن نماها ادامه می‌یابد.

جمع یا تفریق مانتیس‌ها شبیه الگوریتم جمع و تفریق اعداد ممیز ثابت ارائه شده در شکل ۲-۱۰ است. بخش اندازه بسته به عمل موردنظر و علامت‌های دو مانتیس جمع یا تفریق می‌شوند. اگر در جمع



شکل ۱۵-۱۰ جمع و تفریق اعداد ممیز شناور

مانتیس‌ها سرریز رخ دهد، به فلیپ فلاپ E منتقل می‌شود. اگر E برابر 1 باشد، بیت به A_1 منتقل می‌گردد و تمام بیت‌های دیگر A به راست شیفت می‌یابند. برای حفظ عدد صحیح، نما باید افزایش یابد. در این حالت هیچ فروریزی صورت نمی‌گیرد زیرا مانتیس اولیه که در حین نرمالیزه کردن شیفت داده نشده است از قبل نرمالیزه بوده است.

اگر اندازه‌ها از هم تفریق شوند، نتیجه ممکن است صفر یا فروریز داشته باشد. اگر مانتیس صفرباشد. کل عدد ممیز شناور در AC صفر می‌شود. در غیراینصورت مانتیس باید حداقل دارای یک بیت 1 باشد. اگر با ارزشترین بیت در A_1 برابر 0 باشد مانتیس دارای فروریز است. در این حال، مانتیس به چپ شیفت یافته و نما کاهش می‌یابد. بیت A_1 مجدداً چک شده و روند تا 1 شدن A_1 ادامه می‌یابد. وقتی که $A_1=1$ باشد، مانتیس نرمالیزه شده و عملیات خاتمه یافته است.

ضرب

ضرب دو عدد ممیز شناور با ضرب مانتیس‌ها و جمع نماها انجام می‌شود. در این عمل مقایسه نماها و هم ردیف کردن مانتیس‌ها لزومی ندارد. ضرب مانتیس‌ها مشابه ضرب اعداد ممیز ثابت است و حاصلضرب با دقتی دو برابر حاصل می‌شود. پاسخ حاصل با دقت مضاعف[1] در ممیزثابت برای افزایش دقت حاصلضرب بکار می‌رود. در نمایش ممیز شناور، محدوده دقت معمولی[2] همراه با نما به اندازه کافی دقت دارد و بنابراین اعداد بصورت غیرمضاعف یا معمولی نگهداری می‌شوند. لذا نیمه با ارزشتر حاصلضرب مانتیس‌ها و نماها همراه با هم برداشته شده و حاصلضربی با دقت معمولی را برای ممیز شناور بوجود می‌آورد.

الگوریتم ضرب به چهار بخش زیر تقسیم می‌شود

۱- چک کردن صفر

۲- جمع کردن نماها

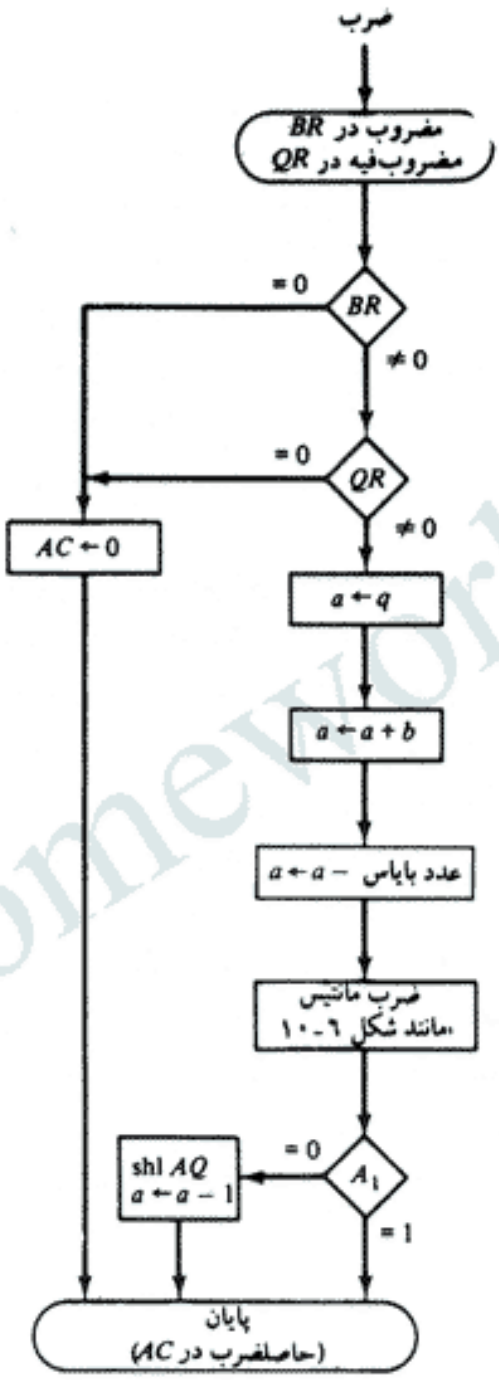
۳- ضرب مانتیس‌ها

۴- نرمالیزه کردن حاصلضرب

مراحل ۲ و ۳ می‌توانند بطور همزمان انجام شوند مشروط بر این که برای مانتیس‌ها و نماها، جمع‌کننده‌های جداگانه‌ای وجود داشته باشند.

فلوچارت برای ضرب ممیز شناور در شکل ۱۶-۱۰ نشان داده است. دو عملوند برای تعیین اینکه آیا حاوی صفرند یا خیر چک می‌شوند. اگر هر یک از عملوندها برابر صفر باشند، حاصلضرب در AC برابر صفر می‌شود و عمل خاتمه می‌یابد. اما اگر هیچیک از عملوندها صفر نباشد، روند با جمع نماها ادامه می‌یابد.

1- Dauble - Precision 2- Single - Precision



شکل ۱۶-۱۰ ضرب اعداد ممیز شناور

نمای مضروب فیه در q است و جمع کننده a و b را جمع می‌کند. لازم است نما را از q به a انتقال دهیم و سپس دو نما را با هم جمع کنیم، و مجموع را به a منتقل نمائیم. چون هر دو نما بعلت جمع با یک مقدار ثابت بایاس شده‌اند، نمای مجموع دو برابر بایاس خواهد بود. نمای بایاس شده صحیح پس از تفریق عدد بایاس از مجموع بدست می‌آید.

ضرب مانتیس‌ها مشابه حالت ممیز ثابت انجام می‌شود و حاصلضرب در A و Q قرار می‌گیرد. در اینجا سرریز حین ضرب نمی‌تواند رخ دهد، لذا نیازی به چک کردن آن نیست.

حاصلضرب ممکن است فروریز داشته باشد، بنابراین با ارزشترین بیت A چک می‌شود. اگر این بیت 1 باشد ضرب قبلا نرمالیزه شده است. اگر 0 باشد مانتیس موجود در AQ به چپ شیفت پیدا کرده و نما کاهش داده می‌شود. توجه کنید که تنها یکبار شیفت برای نرمالیزه کردن کافی است. کوچکترین عملوند نرمالیزه کردن 0.1 است، لذا کوچکترین حاصلضرب ممکن 0.01 می‌باشد. بنابراین، تنها یک صفر ممکن است در منتها الیه سمت چپ وجود داشته باشد.

هرچند نیمه کم ارزشتر مانتیس در Q است، ولی ما آنرا برای ضرب ممیز شناور بکار نمی‌بریم و تنها مقدار داخل AC بعنوان حاصلضرب در نظر گرفته می‌شود.

تقسیم

در تقسیم ممیز شناور نماها تفریق و مانتیس‌ها تقسیم می‌شوند. تقسیم مانتیس مشابه ممیز ثابت است جز اینکه مقسوم مانتیسی با دقت معمولی دارد و در AC قرار داده می‌شود. بخاطر بسپارید که مانتیس‌مقسوم کسری است و نه عدد صحیح. برای نمایش، مقسومی با دقت معمولی باید در ثبات Q قرار گیرد و ثبات A پاک شود. صفرهای داخل A در سمت چپ ممیز دودویی قرار داشته و اهمیتی ندارند. در نمایش کسری، مقسوم با دقت معمولی در ثبات A قرار می‌گیرد و ثبات Q پاک می‌شود. صفرهای موجود در Q در سمت راست ممیز دودویی بوده و اهمیتی ندارند.

تست سرریز تقسیم همانند نمایش ممیز ثابت است. با این وجود، سرریز تقسیم در ممیز شناور مشکلی ایجاد نمی‌کند. اگر مقسوم بزرگتر یا مساوی مقسوم علیه باشد، کسر مقسوم به راست شیفت داده می‌شود و نمای آن یک واحد افزایش می‌یابد. برای عملوندهای نرمالیزه شده، این عمل برای تضمین عدم وقوع سرریز تقسیم کافی است. عمل فوق هم ردیف کردن مقسوم[1] خوانده می‌شود.

تقسیم دو عدد ممیز شناور نرمالیزه شده همواره خارج قسمت نرمالیزه شده ای را بدست می‌دهد بشرطی که هم ردیف کردن مقسوم قبل از تقسیم انجام شده باشد. بنابراین برخلاف سایر اعمال، خارج قسمت حاصل از تقسیم نیازی به نرمالیزه شدن ندارد.

الگوریتم تقسیم‌می‌تواند به پنج بخش زیر تقسیم شود.

۱- چک کردن برای وجود صفر

۲- مقدار دهی اولیه به ثباتها و تعیین علامت

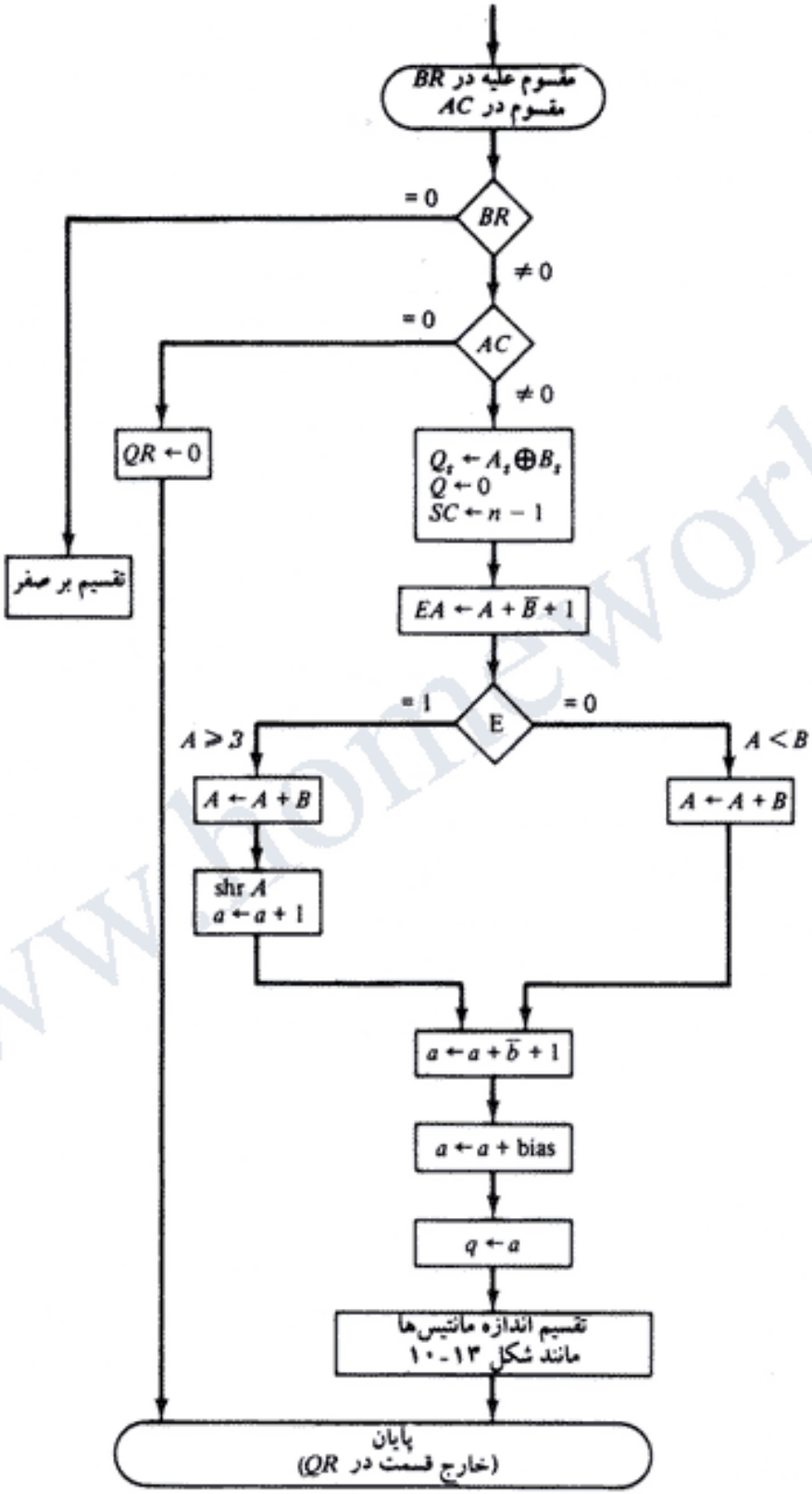
۳- هم ردیف کردن مقسوم

۴- تفریق نمادها

۵- تقسیم مانتیس‌ها

فلوچارت برای تقسیم ممیز شناور در شکل ۱۷-۱۰ نشان داده شده است. دو عملوند برای وجود

1- Divident Alignment



شکل ۱۷-۱۰ تقسیم اعداد ممیز شناور

صفر چک شده‌اند. اگر مقسوم علیه صفرباشد نشاندهنده اقدام به تقسیم بر صفر است، که عملی غیرمجاز است. عملیات با اعلام خطایی متوقف می‌شود. راه دیگر این است که خارج قسمت واقع در QR را با مثبت‌ترین عدد ممکن (اگر مقسوم مثبت باشد) یا منفی‌ترین عدد ممکن (اگر مقسوم منفی) برابر کنیم. اگر مقسوم موجود در AC صفر باشد خارج قسمت موجود در QR صفر می‌شود و عمل خاتمه می‌یابد.

اگر عملوندها صفر نباشند، در مرحله بعدی علامت خارج قسمت را تعیین و آن را در Q_s ذخیره می‌کنیم. علامت مقسوم در A_s بلاتغییر رها می‌شود تا علامت باقیمانده نیز باشد. ثبات Q پاک شده و توالی شمار SC با تعداد بیت‌های خاوج قسمت بار می‌شود.

هم ردیف کردن مقسوم مشابه چک کردن سرریز تقسیم در تقسیم ممیز ثابت است. هم ردیف کردن صحیح مقسوم هنگامی است که مقسوم کسر کوچکی از مقسوم علیه باشد. این دو کسر توسط تفریق مقایسه می‌شوند. رقم نقلی E نسبت اندازه‌های آنها را معین می‌کند. کسر مقسوم با جمع کردن با مقسوم علیه به مقدار اولیه‌اش بازگردانده می‌شود. اگر $A \geq B$ باشد لازم است A یکبار به راست شیفت داده شود و نمای مقسوم افزایش یابد. چون هر دو عملوند نرمالیزه هستند، با هم ردیف کردن $A<B$ تضمین خواهد شد.

در عمل بعدی مقسوم علیه ازنمای مقسوم تفریق می‌شود. چون هر دو نما در ابتدا بایاس شده بودند، با عمل تفریق تفاضل بایاس نشده آنها بدست می‌آید. سپس عدد بایاس جمع شده و نتیجه به q منتقل می‌گردد زیرا خارج قسمت در QR تشکیل می‌شود.

اندازه مانتیس‌ها همچون در حالت ممیز ثابت تقسیم می‌شوند. پس از عمل، خارج‌قسمت مانتیس‌ها در Q و باقیمانده در A قرار خواهد گرفت. خارج قسمت ممیز شناور قبلاً نرمالیزه شده و در QR قرار دارد. نمای باقیمانده باید مانند نمای مقسوم باشد. ممیز دودویی برای مانتیس باقیمانده $(n-1)$ مکان در سمت چپ A_1 قرار دارد. می‌توان باقیمانده را به کسر نرمالیزه شده با تفریق n-1 از نمای مقسوم و شیفت و کاهش تا رسیدن به 1 در A_1 تبدیل کرد. این عمل در فلوچارت نشان داده نشده است و بعنوان تمرین واگذار می‌شود.

۶-۱۰ واحد حساب دهدهی

کاربر یک کامپیوتر داده‌ها را بصورت اعداد دهدهی فراهم نموده و نتایج را هم بصورت دهدهی دریافت می‌نماید. یک CPU با یک واحد حساب و منطق می‌تواند ریزاعمال را با اعداد دودویی انجام دهد. برای اجرای عملیات حسابی با داده‌های دهدهی، لازم است داده‌های دهدهی ورودی به اعداد دودویی تبدیل شوند، تا محاسبات با اعداد دودویی انجام شده، و نتایج به دهدهی 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 دهدهی انجام دهیم. کامپیوترهایی که

قادرند محاسبات دهدهی را اجرا نمایند باید داده‌های دهدهی را به فرم کدهای دودویی ذخیره کنند. اعداد دهدهی سپس به واحد محاسبات دهدهی که قادر است ریزعمل‌های حساب دهدهی را اجرا نماید اعمال می‌شود.

ماشین‌های حساب الکترونیکی بلااستثناء دارای واحد محاسبات دهدهی داخلی می‌باشند زیرا حجم ورودی و خروجی آنها زیاد است. به نظر نمی‌رسد که برای تبدیل اعداد ورودی از طریق صفحه کلید به دودویی و بالعکس و سپس تبدیل مجدد نتایجی که باید نمایش داده شود به دهدهی، دلیلی وجود داشته باشد. چون این روند نیاز به مدار خاصی داشته و زمان اجرای بیشتری هم نیاز دارد. اغلب کامپیوترها برای محاسبات حسابی با هر دو نوع داده‌های دودویی و دهدهی دارای سخت افزار خاصی هستند. کاربرها بکمک دستورات برنامه ریزی شده می‌توانند خواسته خود، مبنی بر محاسبات دودویی یا دهدهی توسط کامپیوتر را مشخص نمایند.

یک واحد محاسبات دهدهی مدار دیجیتالی است که ریزعملیات دهدهی را انجام می‌دهد. این مدار می‌تواند اعداد دهدهی را جمع یا تفریق نماید، که این عمل اخیر معمولاً بکمک متمم 9 و یا 10 مفروق انجام می‌شود. مدار اعداد کد شده به دهدهی را دریافت و نتایج را با همان کددودویی تولید می‌نماید. یک واحد محاسبات دهدهی یک مرحله‌ای، از نُه متغیر ورودی دودویی تشکیل شده و پنج متغیر خروجی دودویی نیز دارد، چون حداقل چهار بیت برای نمایش هر رقم کد شده به دهدهی لازم است. هر مرحله باید چهار ورودی برای رقم مضاعف، چهار رقم برای مضاف الیه، و یک رقم نقلی ورودی داشته باشد. خروجی‌ها شامل چهارپایانه برای رقم حاصل جمع و یک رقم نقلی خروجی است. البته، انواع متفاوتی از مدارات امکان پذیر است که به کد مورد استفاده برای نمایش دهدهی بستگی دارد.

جمع کننده BCD

جمع حسابی دو رقم دهدهی به شکل BCD، همراه با رقم نقلی احتمالی از طبقه قبلی را در نظر بگیرید. چون هر رقم ورودی از 9 تجاوز نمی‌کند، حاصل جمع خروجی 19=1+9+9 تجاوز نخواهد کرد، که عدد 1 در جمع، رقم نقلی ورودی است. فرض کنید که دو رقم BCD را به یک جمع کننده دودویی چهاربیتی اعمال کنیم. جمع کننده حاصل جمع را بصورت دودویی تشکیل می‌دهد و نتیجه حاصل بین 0 تا 19 خواهد بود. این اعداد دودویی در جدول ۴-۱۰ لیست شده و با سمبل‌های K و Z_8،Z_4 ، Z_2 و Z_1 نام گذاری شده‌اند. K، رقم نقلی و اندیس‌های زیر حروف Z نشان دهنده وزن‌های 8 و 4 و 2 و 1 بوده و به چهاربیت در کد BCD اختصاص دارند. اولین ستون در جدول حاصل جمع‌های دودویی را آنچنانکه در خروجی‌های یک جمع کننده دودویی چهاربیتی ظاهر می‌شوند نشان می‌دهد. خروجی حاصل جمع دو عدد دهدهی بایستی بصورت BCD نشان داده شود. مسئله در اینجا یافتن قاعده ساده‌ای است که بوسیله آن عدد دودویی در ستون اول به رقم BCD صحیحی که در ستون دوم نمایش داده شده تبدیل شود.

جدول ۴-۱۰ بدست آوردن جمع کننده BCD

مجموع دودویی					مجموع BCD					
K	Z_8	Z_4	Z_2	Z_1	C	S_8	S_4	S_2	S_1	دهدهی
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	2
0	0	0	1	1	0	0	0	1	1	3
0	0	1	0	0	0	0	1	0	0	4
0	0	1	0	1	0	0	1	0	1	5
0	0	1	1	0	0	0	1	1	0	6
0	0	1	1	1	0	0	1	1	1	7
0	1	0	0	0	0	1	0	0	0	8
0	1	0	0	1	0	1	0	0	1	9
0	1	0	1	0	1	0	0	0	0	10
0	1	0	1	1	1	0	0	0	1	11
0	1	1	0	0	1	0	0	1	0	12
0	1	1	0	1	1	0	0	1	1	13
0	1	1	1	0	1	0	1	0	0	14
0	1	1	1	1	1	0	1	0	1	15
1	0	0	0	0	1	0	1	1	0	16
1	0	0	0	1	1	0	1	1	1	17
1	0	0	1	0	1	1	0	0	0	18
1	0	0	1	1	1	1	0	0	1	19

با بررسی محتوای جدول، واضح است که وقتی حاصل جمع دودویی برابر یا کمتر از 1001 باشد، عدد BCD متناظر، مانند خود آن عدد است و لذا تبدیلی لازم نیست. وقتی که حاصل جمع بزرگتر از 1001 باشد نمایش نامعتبری از BCD را خواهیم داشت. جمع عدد دودویی 6 (0110) با حاصلجمع دودویی، نمایش BCD را تصحیح نموده و یک رقم نقلی خروجی لازم را نیز تولید می‌کند.

یکی از روش‌های جمع اعداد دهدهی 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 باارزشتر 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.

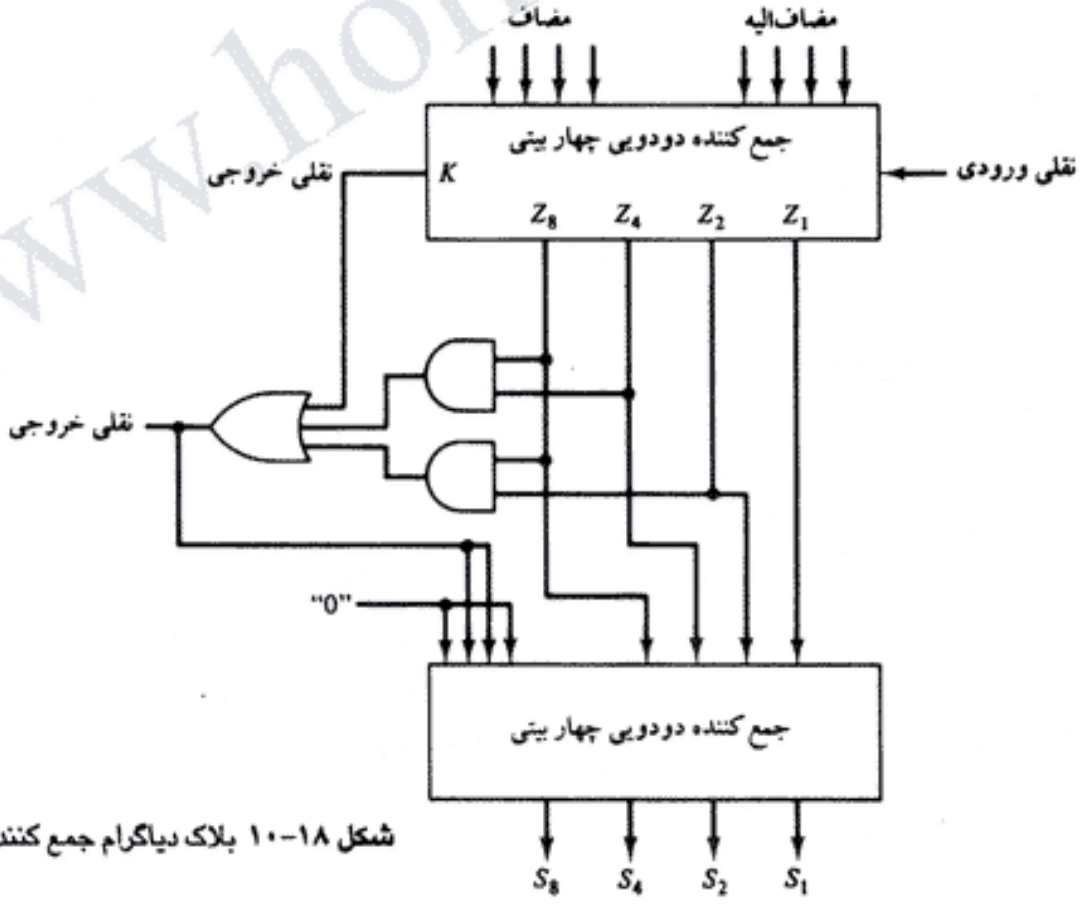
مداری که که ضرورت تصحیح را مشخص می‌کند از وارده‌های جدول بدست می‌آید. واضح است

که هرگاه مجموع دودویی دارای نقلی خروجی مساوی 1 باشد تصحیح لازم است. شش ترکیب دیگر از 1010 تا 1111 که اصلاح نیاز دارند دارای 1 در مکان Z_8 هستند، Z_4 یا Z_2 هم باید 1 داشته باشند، شرایط تصحیح و رقم نقلی خروجی با تابع بولی زیر بیان می‌شود.

$$C = K + Z_8 Z_4 + Z_8 Z_2$$

هنگامی که C=1 باشد، لازم است به مجموع دودویی مقدار 0110 را اضافه کنیم و یک رقم نقلی برای طبقه‌بندی تولید نمائیم.

یک جمع‌کننده BCD مداری است که دو رقم BCD را بصورت موازی جمع کرده و ارقام مجموع را نیز بفرم BCD تولید می‌کند. یک جمع کننده BCD باید دارای مدار تصحیح کننده در ساختمان داخلی خود باشد. برای جمع 0110 بامجموع دودویی، ازیک جمع کننده دودویی چهار بیتی دومی که در شکل ۱۸-۱۰ نشان داده شده است استفاده می‌کنیم. دو رقم دهدهی، همراه با رقم نقلی ورودی، ابتدا در جمع‌کننده فوقانی با هم جمع شده و مجموع دودویی را تولید می‌کنند. مادامی که رقم نقلی خروجی برابر 0 باشد، چیزی به مجموع دودویی اضافه نمی‌شود. اگر رقم نقلی خروجی 1 باشد، عدد دودویی 0110 به مجموع دودویی از طریق جمع کننده تحتانی اضافه می‌شود. رقم نقلی خروجی تولید شده بوسیله جمع کننده دودویی تحتانی می‌تواند چشم پوشی شود، زیرا اطلاعاتی راکه قبلاً در خروجی رقم



شکل ۱۸-۱۰ بلاک دیاگرام جمع کننده BCD

نقلی موجود است تولید نموده است.

یک جمع کننده موازی دهدهی که n رقم دهدهی را جمع می‌کند به n طبقه جمع کننده BCD نیاز داردکه در آن هر خروجی نقلی از یک طبقه به ورودی نقلی طبقه با ارزشتر بعدی متصل است. برای دستیابی به تاخیرهای انتشار کوتاهتر، جمع کننده‌های BCD حاوی مدارات لازم برای تولید نقلی پیش بینی شده می‌باشند. بعلاوه، مدار جمع کننده برای اجرای عمل تصحیح نیازی به هر چهار تمام جمع کننده ندارد و لذا مدار می‌تواند بهینه شود.

تفریق BCD

تفریق مستقیم دوعدد دهدهی به یک مدار تفریق‌گر که از برخی لحاظ با جمع کننده BCD متفاوت است نیاز دارد. اگر تفریق را بکمک متمم 9 یا 10 مفروق و جمع آن با مفروق منه انجام دهیم اقتصادی تر خواهد بود. چون BCD یک کد خود متمم[1] نیست، متمم 9 آن را نمی‌توان با متمم کردن هر بیت در کد بدست آورد. این متمم را با تشکیل مداری که هر رقم BCD را از 9 تفریق می‌کند باید بدست آورد.

متمم 9 یک رقم دهدهی که بصورت BCD نشان داده شده باشد را می‌توان با متمم کردن بیت‌ها در نمایش کد شده رقم بدست آورد بشرط اینکه تصحیحی صورت گیرد. دو روش تصحیح وجود دارد. در روش اول، عدد دودویی 1010 (10 دهدهی) به هر رقم متمم اضافه شده و رقم نقلی بعداز هرجمع چشم پوشی می‌شود. در روش دوم، عدد دودویی 0110 (6 دهدهی) قبل از متمم شدن رقم اضافه می‌شود. به عنوان یک مثال عددی، برای بدست آوردن متمم 9 عدد 0111 (7 دهدهی) BCD بدین ترتیب محاسبه می‌شود که ابتدا هر بیت متمم می‌شود تا 1000 بدست آید. با جمع 1010 با آن و چشم پوشی از رقم نقلی عدد 0010 (2 دهدهی) را خواهیم داشت. با استفاده از روش دوم، ما عدد 0110 را به 0111 اضافه می‌کنیم تا 1101 بدست آید. با متمم کردن هر بیت نتیجه مورد نظر 0010 بدست می‌آید. متمم کردن عدد چهاربیتی N مشابه تفریق آن از 1111 (15 دهدهی) می‌باشد. با اضافه کردن معادل دودویی عدد 10 به آن داریم $15-N+10=9-N+16$. از طرفی 16 بمعنی رقم نقلی بوده و نادیده گرفته می‌شود، بنابراین نتیجه $9-N$ خواهد شد. با افزودن معادل دودویی عدد دهدهی 16 و متمم کردن آن نیز داریم $15-(N+6)=9-N$.

متمم 9 عدد BCD از طریق یک مدار ترکیبی نیز بدست می‌آید. وقتی که این مدار به یک جمع‌کننده BCD متصل گردد، یک جمع یا تفریق کننده BCD بدست می‌آید. اجازه بدهید 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 می‌کند. هنگامی که M=0 باشد، دو رقم جمع می‌شوند؛ اگر M=1 باشد، ارقام از یکدیگر تفریق می‌گردند. همچنین، فرض کنید متغیرهای دودویی x_8، x_4، x_2 و x_1 خروجی‌های مدار متمم ساز (متمم 9) باشند. با بررسی جدول درستی این مدار مشاهده می‌شود (مسئله ۳۰-۱۰) که B_1 همواره باید متمم شود؛ B_2 همیشه در متمم 9 مثل رقم اصلی است؛ x_4 برابر 1 است اگر OR

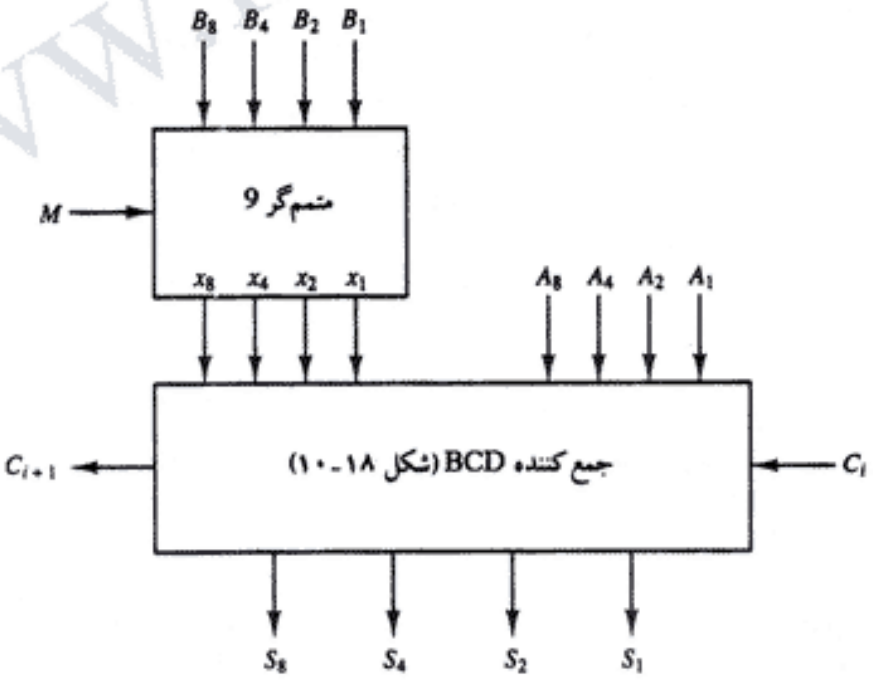
1- Self Complement

انحصاری B_2 و B_4 برابر 1 باشند؛ x_8 هم، اگر $B_8B_4B_2=000$ باشد، برابر 1 می‌باشد. توابع بولی تولیدکننده متمم 9 عبارتند از:

$$x_1 = B_1M' + B'_1M$$
$$x_2 = B_2$$
$$x_4 = B_4M' + (B'_4 B_2 + B_4B'_2) M$$
$$x_8 = B_8M' + B'_8 B'_4 M$$

باتوجه به معادلات فوق در می‌یابیم 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 دهدهی که قادر است دو رقم و BCD را جمع و یا تفریق کند در شکل ۱۹-۱۰ نشان داده شده است. این مدار متشکل از یک جمع کننده را BCD یک متمم سازاست. ورودی M کار مدار راکنترل 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 n طبقه از این نوع احتیاج است. خروجی نقلی C_{i+1} از یک طبقه باید به ورودی C_i طبقه با ارزشتر متصل شود. بهترین راه تفریق دو عدد دهدهی 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 ۱۹-۱۰ یک طبقه از یک واحد حساب دهدهی

۱۰-۷ اعمال حسابی دهدهی

الگوریتم‌های اعمال حسابی با داده‌های مشابه الگوریتم‌های همان اعمال با داده‌های دودویی است. در واقع بجز کمی تصحیح در الگوریتم ضرب و تقسیم، فلوچارت‌های یکسانی را برای دونوع داده می‌توان بکاربرد با این شرط که سمبل ریز عمل‌ها را بطور صحیح تفسیر کنیم. اعداد دهدهی با نمایش BCD گروه‌های چهاربیتی در ثبات‌های کامپیوتر ذخیره می‌شوند. هر گروه چهاربیتی یک رقم دهدهی را نمایش می‌دهد و باید هنگام اجرای ریزعمل‌های دهدهی بصورت واحد در نظر گرفته شود.

برای سهولت، ما سمبل‌های یکسانی را برای ریزعمل حسابی دودویی و دهدهی بکار خواهیم برد ولی بصورت متفاوتی آنها را تعبیر خواهیم کرد. همان طور که در جدول ۵-۱۰ دیده می‌شود، یک خط روی سمبل حرف بمعنی متمم 9 یک عدد دهدهی ذخیره شده در ثبات است. جمع 1 با متمم 9 متمم 10 را تولید می‌کند. بنابراین برای اعداد دهدهی، سمبل $A \leftarrow A+\bar{B}+1$ بمعنی انتقال مجموع دهدهی تشکیل شده از جمع مقدار اولیه A با متمم 10 عدد B است. استفاده از سمبل‌های یکسان برای متمم 9 و متمم 1 ، اگر هر دو نوع داده در یک سیستم بکار رود موجب سردرگمی می‌گردد. در این حالت بهتر خواهد بود سمبل‌های متفاوتی برای متمم 9 بکار برده شود. اگر یک نوع داده مورد استفاده قرار گیرد نماد مورد استفاده متعلق به همان نوع داده است.

افزایش یا کاهش ثبات برای اعداد دودویی و دهدهی، بجز تعداد حالات مجاز برای ثبات یکسان است. افزایش یک شمارنده دودویی موجب طی کردن 16 حالت از 0000 تا 1111 می‌گردد. یک شمارنده دهدهی 10 حالت از 0000 تا 1001 را می‌شمارد و دوباره به 0000 باز می‌گردد، زیرا عدد 9 آخرین عدد ممکن در شمارش دهدهی است. بطور مشابه ، کاهش یک شمارنده دودویی رشته 1111 تا 0000 را طی می‌کند. شمارنده دهدهی در این فرم از 1001 تا 0000 را می‌شمارد.

در جلو شیفت به راست یا چپ دهدهی حرف d بکار رفته است تا مشخص کننده شیفت بر روی چهاربیتی باشد که ارقام دهدهی را دارا هستند. بعنوان مثال عددی، ثبات A که عدد دهدهی 7860 را بشکل BCD در خود دارد در نظر بگیرید. الگوی 12 فلیپ فلاپ بقرار زیر است.

0111 1000 0110 0000

جدول ۵-۱۰ سمبل‌های ریزعمل‌های حسابی دهدهی

شرح	نمایش سمبلیک
جمع عددهای دهدهی و انتقال مجموع به A	$A \leftarrow A + B$
متمم 9 از B	$\bar{B}$
محتوای A بعلاوه متمم 10 عدد B در A قرار گیرد	$A \leftarrow A + \bar{B} + 1$
افزایش عدد BCD موجود در Q_L	$Q_L \leftarrow Q_L + 1$
شیفت دهدهی ثبات A به راست	dshr A
شیفت دهدهی ثبات A به چپ	dshl A

ریزعمل dshr عدد دهدهی را یک رقم به راست شیفت می‌دهد و لذا 0786 بدست می‌آید. این شیفت روی دسته‌های 4 بیتی انجام شده و محتوای ثبات را بصورت زیر تغییر می‌دهد.

0000 0111 1000 0110

جمع و تفریق

الگوریتم جمع و تفریق اعداد دودویی با نمایش مقدار علامت دار برای اعداد دهدهی با نمایش مقدار علامت دار نیز صادق است با این شرط که سمبل‌ها را بطور صحیحی تعبیر کنیم. بطور مشابه الگوریتم اعداد دودویی با نمایش متمم 2 علامت دار برای اعداد دهدهی متمم 10 علامت دار نیز صادق است. برای داده‌های دودویی از یک جمع کننده دودویی و یک متمم ساز استفاده می‌شود. برای داده‌های دهدهی از واحد محاسباتی که قادر به جمع اعداد BCD بوده و بتواند متمم 9 مفروق را طبق شکل ۱۹-۱۰ تشکیل دهد استفاده می‌شود.

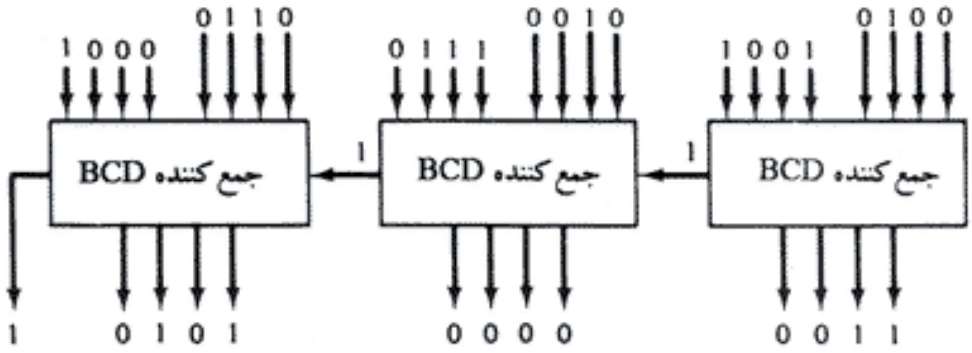
داده‌های دهدهی به سه طریق مختلف باهم جمع می‌شوند، شکل ۲۰-۱۰. روش موازی از یک واحد محاسبه دهدهی متشکل از جمع کننده‌های BCD بتعداد ارقام عدد استفاده می‌کند. مجموع بطور موازی تشکیل و فقط به یک ریز عمل نیاز دارد. در روش رقم- سری، بیت- موازی[1]، ارقام فقط به یک جمع‌کننده BCD بصورت سری اعمال می‌شوند، در حالیکه بیت‌های هر کد بصورت موازی انتقال می‌یابند. حاصل جمع با شیفت اعداد دهدهی به جمع کننده BCD، یکی پس از دیگری تشکیل می‌گردد. برای k رقم دهدهی، این آرایش به k ریزعمل نیاز دارد، که هر یک متعلق به یک شیفت دهدهی است. در تمام جمع‌کننده‌های سری، بیت ها یکی پس ازدیگری بداخل تمام جمع کننده شیفت پیدا می‌کنند. مجموع دودویی حاصل از چهار بار شیفت باید تصحیح شود تا به صورت یک رقم BCD معتبر درآید. برای این تصحیح که در بخش ۶-۱۰ در مورد آن بحث شد، مجموع دودویی وارسی می‌شود. اگر این مجموع بزرگتر یا مساوی 1010 باشد، جمع آن با 0110 آن را تصحیح می‌کند و یک رقم نقلی را نیز برای جفت رقم بعدی تولید می‌نماید.

روش موازی سریع است ولی به تعداد زیادی جمع کننده نیاز دارد. روش رقم- سری، بیت- موازی تنها به یک جمع کننده BCD نیاز دارد که مشترکاً بوسیله کلیه ارقام مورد استفاده قرار می‌گیرد. این روش کندتر از روش موازی است زیرا شیفت ارقام به زمان احتیاج دارند. تمام روش‌های سری به حداقل امکانات نیاز دارند ولی خیلی کند عمل می‌کنند.

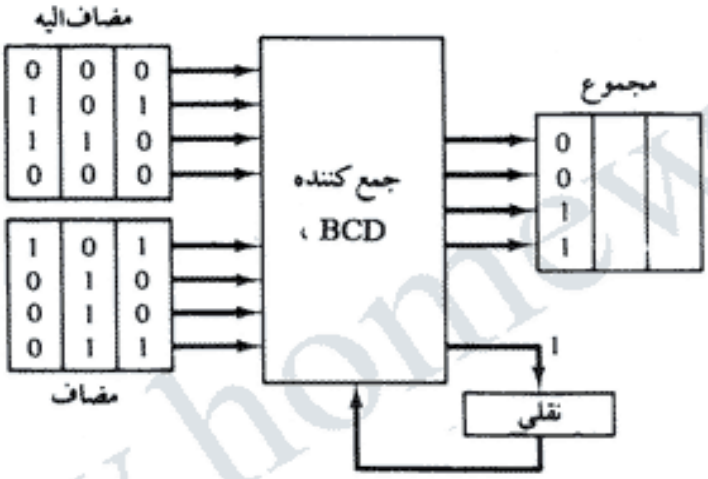
ضرب

ضرب اعداد دهدهی ممیز ثابت، بجز در تولید حاصلضرب جزیی، با روش دودویی مشابه است. در

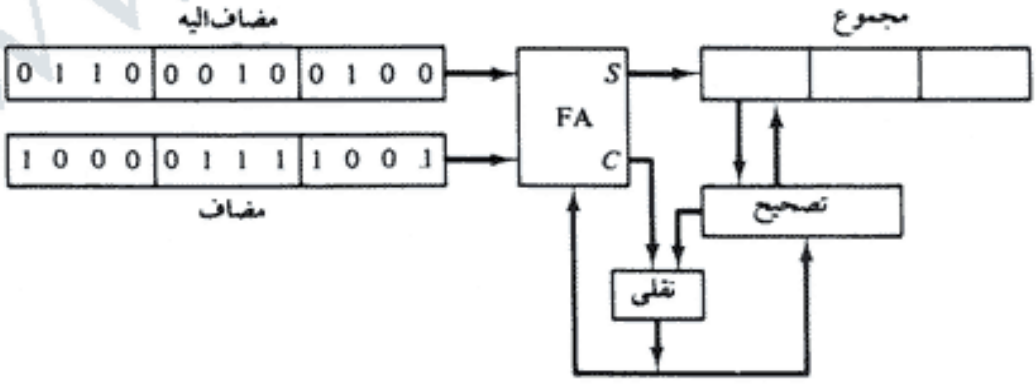
1- Digit – Serial Bit – Parallel



(الف) جمع دهدهی موازی 624+879=1503



(ب) جمع دهدهی رقم سری، بیت- موازی



(ج) جمع دهدهی تمام سری

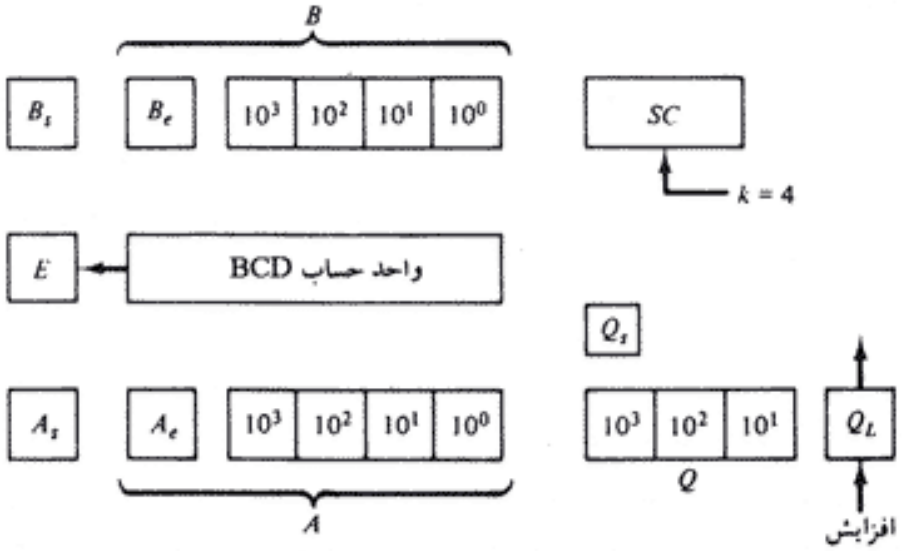
شکل ۲۰-۱۰ سه روش جمع اعداد دهدهی

ضرب دهدهی ارقام در محدوده 0 تا 9 قرار دارند، در صورتی که در ضرب دودویی فقط دو رقم 0 و 1 وجود دارد. در حالت دودویی، مضروب ، بشرطی که مضروب فیه 1 باشد، با حاصلضرب جزیی جمع می‌شود. در حالت دهدهی مضروب باید در رقم مضروب فیه ضرب شده و سپس به حاصلضرب جزیی اضافه شود. این عمل را می‌توان با جمع مضروب با حاصلضرب جزیی به تعداد دفعاتی برابر با مقدار رقم مضروب فیه انجام داد.

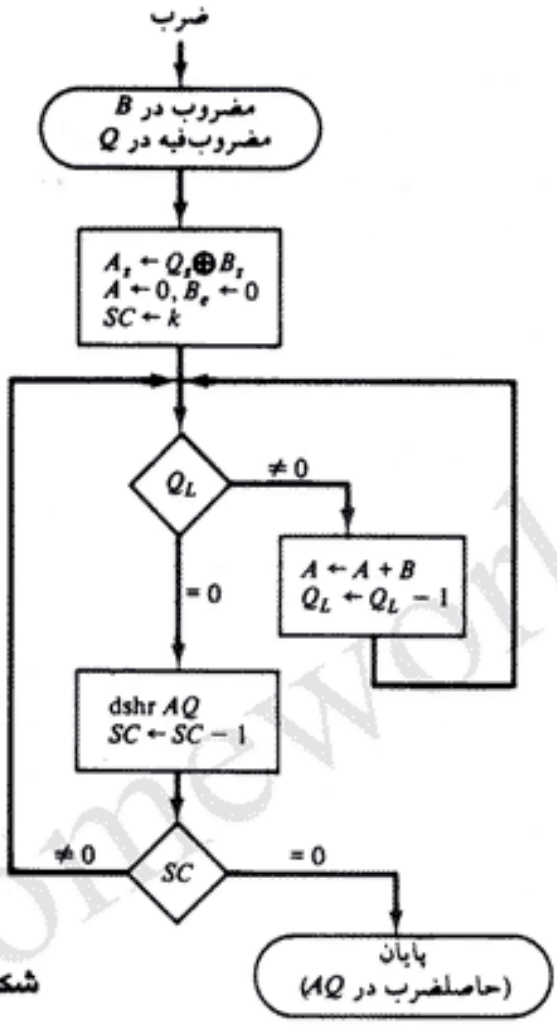
سازمان ثبات‌ها برای ضرب دهدهی در شکل ۲۱-۱۰ نشان داده شده است. در اینجا فرض می‌کنیم که اعداد چهار رقمی باشند و هر رقم چهار بیت را اشغال نماید، در نتیجه هر عدد 16 بیتی خواهد بود. سه ثبات A و B و Q وجود دارند که هر یک فلیپ فلاپ علامت مربوط به خود یعنی A_s ، B_s, Q_s را دارا هستند . ثبات‌های A و B دارای چهار بیت اضافی دیگرند که با A_e و B_e نشان داده شده و به منظور گسترش ثبات‌ها 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 می‌دهد. رقم نقلی نهایی به فلیپ فلاپ E می‌رود. هدف از رقم A_e جای دادن سرریز احتمالی حاصل ازجمع مضروب به حاصل ضرب جزیی در حین عمل ضرب است. هدف از B_e تشکیل متمم 9 مقسوم علیه هنگام تفریق آن از باقیمانده جزیی در حین عمل تقسیم است. کم ارزشترین رقم ثبات Q با Q_L مشخص می‌شود. این رقم را می‌توان افزایش یا کاهش داد.

یک عملوند Q دهدهی برداشته شده از حافظه،17 بیتی است. یک بیت (علامت) به B_s منتقل می‌شود و اندازه عملوند در 16 بیت پائین تر B قرار می‌گیرد. درآغاز هر دو بیت A_e و B_e پاک می‌شوند. نتیجه عمل نیز 17 بیتی است و در آن از بخش A_e ثبات A استفاده نمی‌شود.

الگوریتم ضرب دهدهی در شکل ۲۲-۱۰ دیده می‌شود . در آغاز،کل ثبات A و B_e پاک شده وتوالی شمار SC با عددی مانند k که برابر تعداد ارقام مضروب فیه است پر می‌شود. کم‌ارزشترین رقم مضروب‌فیه در Q_L وارسی می‌شود. اگر برابر 0 نباشد، مضروب در B با حاصلضرب جزیی در A یکبار جمع شده و



شکل ۲۱-۱۰ سازمان ثبات‌ها برای ضرب و تقسیم دهدهی



شکل ۲۲-۱۰ فلوچارت ضرب دهدهی

Q_L یک واحد کم می‌شود. مجدداً Q_L چک می‌شود و روند تا صفر شدن آن ادامه می‌یابد. بدین ترتیب مضروب موجود در B به تعداد دفعاتی برابر با رقم مضروب فیه با حاصلضرب جزیی جمع می‌شود. هر رقم سرریز موقت در A_e قرار گرفته و می‌تواند بین 0 تا 9 باشد.

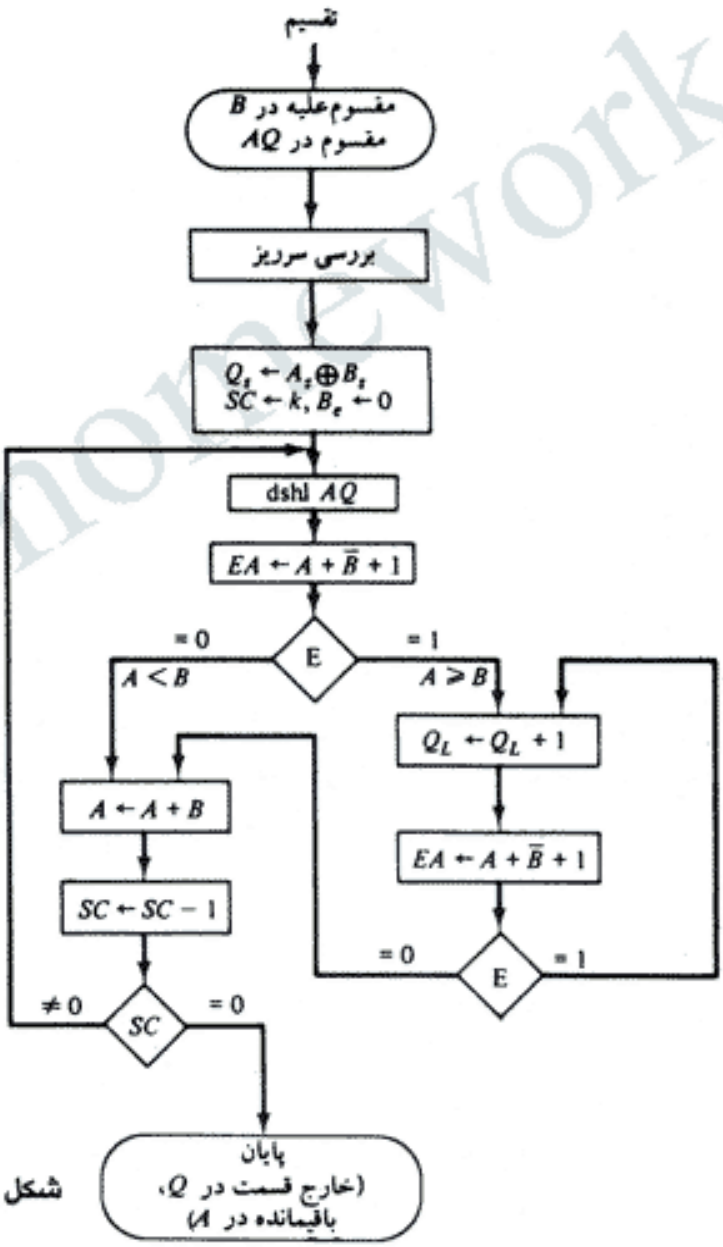
سپس حاصلضرب جزیی و مضروب‌فیه یکبار به راست شیفت داده می‌شود. این عمل یک 0 را در A_e قرار می‌دهد و رقم بعدی مضروب‌فیه به Q_L منتقل می‌شود. نهایتاً روند k مرتبه تکرار می‌شود تا حاصلضرب با طول دو برابر در AQ تشکیل شود.

تقسیم

تقسیم دهدهی مشابه تقسیم دودویی است با این تفاوت که ارقام خارج قسمت ممکن است هر یک مقادیر 0 تا 9 را داشته باشند. در روش تقسیم بازیافتی، مقسوم علیه به تعداد دفعات لازم از مقسوم یا باقیمانده جزیی تفریق می‌شود تا اینکه باقیمانده منفی بدست آید. سپس با جمع کردن مقسوم علیه، باقیمانده صحیح بازیابی می‌شود. رقم موجود در خارج قسمت منعکس کننده تعداد تفریق‌های انجام

شده منهای تفریقی است که موجب تفاضل منفی شده است.

الگوریتم تقسیم دهدهی در شکل ۲۳-۱۰ 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 A_e قرار می‌گیرد. سپس متمم 10 مقسوم علیه با مقسوم جمع شده و بدیت ترتیب عمل تفریق مقسوم علیه از مقسوم انجام می‌گیرد. چون B_e در ابتدا پاک شده است، متمم آن 9 است. رقم نقلی در E نسبت اندازه‌های A و B را معین می‌کند. اگر E=0 باشد، نشان دهنده A<B است. در این حالت مقسوم علیه جمع می‌شود تا باقیمانده جزیی بازیابی شود و Q_L



شکل ۲۳-۱۰ فلوچارت تقسیم دهدهی

در 0 باقی می‌ماند (درحین شیفت صفر وارد شده است). اگر E=1 باشد نشاهنده A≥B است. رقم خارج قسمت موجود در Q_L یکبار افزایش یافته و مقسوم علیه مجدداً تفریق می‌شود. این روند تا رسیدن به تفاضل 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 بدست 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 خواهند بود. مقدار E صرفنظر می‌شود.

اعمال ممیز شناور

اعمال حسابی دهدهی ممیز شناور روند مشابهی با اعمال دودویی دارند. الگوریتم‌های بخش ۵-۱۰ می‌تواند برای داده‌های دهدهی بکار رود بشرطی که سمبل‌های ریز اعمال بطور صحیحی تعبیر شوند. ضرب تقسیم مانتیس‌ها باید با استفاده از روش‌های فوق صورت گیرد.

مسائل

۱-۱۰ اگر در شکل ۱-۱۰ بجای $A + \bar{B} + 1$ عمل $B + \bar{A}$ (متمم 1 عدد A بعلاوه B) را انجام دهیم متمم ساز لازم نیست . الگوریتمی به شکل فلوچارت برای جمع و تفریق اعداد دودویی ممیز ثابت با نمایش اندازه علامت دار بدست آورید که در آن اندازه‌ها با دو ریزعمل $\bar{A} \leftarrow A$ و $\bar{E} \leftarrow$ A+B تفریق شوند.

۲-۱۰ هر یک از مسیرهای فلوچارت شکل ۲-۱۰ را شماره گذاری کنید و سپس کل مسیری که الگوریتم هنگام محاسبه اعداد با اندازه علامت دار زیر طی می‌کند را مشخص نمائید. در هر مورد مقدار AVF را بنویسید. بیت منتهاالیه سمت چپ در عددهای زیر نماینده بیت علامت است.

الف) 0 101101 + 0 011111

ب) 1 011111 + 1 101101

ج) 0 101101 − 0 011111

د) 0 101101 − 0 101101

هـ) 1 011111 − 0 101101

۳-۱۰ اعمال حسابی زیر را با اعداد دودویی و اعداد منفی در نمایش متمم 2 علامت دار انجام دهید. برای جای دادن هر عدد همراه با علامتش از 7 بیت استفاده کنید. در هر حالت، با 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 (یا مقدار). عددی منفی مانند x− بصورت $2^k+(2^k-x)$ نمایش داده می‌شود که در آن اولین 2^k نشان دهنده بیت علامت و (2^k-x) متمم 2 عدد X است. عدد مثبت بصورت x+0 نمایش داده می‌شود، که 0 بیت علامت و X ، مقدار k بیتی آن است. با استفاده از این سمبل‌های کلی ، ثابت کنید که مجموع (±Y) + (±X) را می‌توان با جمع عددهایی شامل بیت‌های 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 نمی‌شود.

ب) رویه اصلاح شده‌ای را برای آشکار سازی یک سرریز وقتی که اعداد متمم 1 استفاده می‌شوند پیشنهاد کنید.

۱۰-۷ الگوریتمی بفرم فلوچارت برای جمع و تفریق دو عدد دودویی ممیز ثابت وقتی که اعداد منفی بصورت متمم 1 هستند بدست آورید.

۱۰-۸ ثابت کنید ضرب دو عدد n رقمی در پایه r حاصلضربی بزرگتر از 2n رقم تولید نمی‌کند. نشان دهید که این عبارت بدان معنی است که سرریزی در عمل ضرب رخ نمی‌دهید.

۱۰-۹ محتوای ثبات‌های E، A، Q و SC (مطابق جدول ۲-۱۰) را در حین روند ضرب دو عدد دودویی 11111 (مضروب) و 10101 (مضروب فیه) نشان دهید. علامت‌ها در نظر گرفته نشده‌اند.

۱۰-۱۰ محتوای ثبات‌های E، A، Q و SC (مطابق شکل ۱۲-۱۰) را در حین روند تقسیم (الف) 10100011 بر 1011؛ (ب) 00001111 بر 0011 نشان دهید. (از مقسوم هشت بیتی استفاده کنید.)

۱۰-۱۱ نشان دهید که جمع B پس از عمل $A + \bar{B} + 1$ مقدار اولیه A را باز می‌گرداند. 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 ۳-۱۰) هنگامی که اعداد دودویی زیر در هم ضرب می‌شوند نشان دهید. فرض کنید ثبات‌های پنج بیتی اعداد علامت دار را نگه می‌دارند. مضروب در هر دو حالت 15+ است.

الف) (+13)× (+15)

ب) (−13) × (+15)

۱۰-۱۵ الگوریتمی را بفرم فلوچارت برای روش غیر بازیافتی تقسیم دودویی ممیز ثابت بدست آورید.

۱۰-۱۶ الگوریتمی 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 64− تا 63+ بدست آورید.

ب) نشان دهید که 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 64 را تفریق کنیم تا مجموع نماها به شکل بایاس شده بدست آید. چگونه می‌توان 64 را با جمع کردن مقدار متمم 2 آن تفریق کرد؟

د) نشان دهید که پس از تفریق دو نمای بایاس شده لازم است 64 را اضافه کنیم تا تفاضل بایاس شده نماها را داشته باشیم

۱۰-۱۸ الگوریتمی بشکل فلوچارت برای مقایسه دوعدد دودویی علامت دار وقتی که اعداد منفی به فرم متمم 2 علامت دار می‌باشند بدست آوردید

الف) با استفاده از عمل تفریق با عددهای متمم 2 علامت دار

ب) با مرور و مقایسه جفت بیت‌ها ازچپ به راست

۱۰-۱۹ مسئله ۱۰-۱۸ را برای اعداد دودویی مقدار علامت دار تکرار کنید.

۱۰-۲۰ فرض کنید n، 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 $n-1$ 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 $n-1$ 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 راکه دارای نمای کوچکتر است به تعداد

تفاضل نماها به راست شیفت دهید. نمای مجموع (یا تفاضل) برابر با نمای بزرگتر است. بدون استفاده از مقایسه کننده مقدارها، با فرض نماهای بایاس شده، و در نظرگرفتن اینکه فقط AC را می‌توان جابجا کرد، الگوریتمی را به شکل فلوچارت برای هم ردیف کردن مانتیس‌ها و قرار دادن نمای بزرگتر در AC بدست آورید.

۱۰-۲۲ نشان دهید که سرریز مانتیس بعداز عمل ضرب ممکن نیست.

۱۰-۲۳ نشان دهید که تقسیم دو عدد نرمالیزه ممیز شناور با مانتیس کسری همواره خارج قسمت نرمالیزه خواهد داد مشروط بر اینکه قبل از عمل تقسیم، هم ردیف کردن مقسوم انجام شود.

۱۰-۲۴ فلوچارت شکل ۱۰-۱۷ را برای تهیه یک باقیمانده نرمالیزه ممیز شناور در AC تعمیم دهید. مانتیس باید یک کسر باشد.

۱۰-۲۵ درالگوریتم 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 محل‌هایی راکه 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 دهید که سرریز یا فروریز نما بوسیله سخت افزار می‌تواند آشکار شود.

۱۰-۲۶ اگر نمایش عدد صحیح را برای مانتیس اعداد ممیز شناور تصور کنیم، درحین ضرب و تقسیم با مشکلات خاصی روبرو می‌شویم. اجازه بدهید تعداد بیت‌های بخش مقدار مانتیس $(n-1)$ باشد. برای نمایش صحیح:

الف) نشان دهید که اگر حاصلضرب با دقت معمولی بکار رود، باید $(n-1)$ به نمای حاصلضرب در AC اضافه شود.

ب) نشان دهید که اگر از مقسوم با مانتیس دقت معمولی استفاده شود، باید هنگامی که Q برابر صفر می‌شود $(n-1)$ از نمای مقسوم تفریق شود.

۱۰-۲۷ سخت افزاری، که برای جمع و تفریق دو عدد دهدهی در نمایش اندازه علامت دار بکار می‌رود را نشان دهید. چگونگی آشکارسازی سرریز را نشان دهید.

۱۰-۲۸ نشان دهید که 356−673 با جمع 673 با متمم 10 عدد 356 و چشم پوشی از نقلی نهایی انجام می‌شود. بلاک دیاگرام سه مرحله از واحد حسابی دهدهی را رسم کنید و نشان دهید چگونه این عمل پیاده سازی می‌شود. تمام بیت‌های ورودی و خروجی واحد حساب را لیست کنید.

۱۰-۲۹ نشان دهید که به جای جمع کننده دودویی چهاربیتی پائین رتبه در شکل ۱۰-۱ می‌توان یک تمام جمع کننده و دو نیم جمع کننده قرار داد

۱۰-۳۰ با استفاده از تکنیک های طراحی مدارهای ترکیبی، توابع بولی متمم ساز 9 اعداد BCD شکل ۱۰-۱۹ را بدست آورید. دیاگرام منطقی را رسم کنید.

۱۰-۳۱ لازم است تا برای دو رقم دهدهی در کد افزونی 3 (جدول ۶-۳) یک جمع کننده طراحی شود. نشان دهید که تصحیح پس از جمع دو رقم با جمع کننده چهاربیتی بصورت زیراست.
الف) رقم نقلی خروجی برابر نقلی تصحیح نشده است
ب) اگر نقلی خروجی برای 1 باشد، 0011 اضافه می‌شود
ج) اگر نقلی خروجی برابر 0 باشد، 1101 اضافه می‌شود و نقلی حاصل از این جمع نادیده گرفته می‌شود. نشان دهید که جمع کننده افزونی 3 می‌تواند با هفت تمام جمع کننده و دو معکوس‌کننده ساخته شود.

۱۰-۳۲ مداری را برای یک متمم ساز 9، هنگامی که ارقام دهدهی به فرم کد افزونی 3 نمایش داده می‌شوند، بدست آورید یک ورودی کنترل مد[1] تعیین می‌کند آیا رقم متمم شده است یا خیر. مزیت استفاده از این کد نسبت به کد BCD چیست؟

۱۰-۳۳ سخت افزار بکار رفته برای جمع و تفریق دو عدد دهدهی با نمایش متمم 10 علامت دار اعداد منفی را نشان دهید. نشان دهید که چگونه سرریز آشکار می‌شود. الگوریتمی 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 دهدهی (شکل ۱۰-۲۲) ، (الف 470×152 و (ب) 999×199 بدست آورید. ثبات‌ها را سه رقمی تصور کنید و دومین عدد را مضروب‌فیه فرض نمائید.

۱۰-۳۵ محتوای ثبات‌های A، و E و Q و SC را در حین تقسیم دهدهی (شکل ۱۰-۲۳) 1680/32 نشان دهید. ثبات‌ها را دو رقمی فرض کنید

۱۰-۳۶ نشان دهید که زیر ثبات A_e در شکل ۱۰-۲۱ در پایان (الف) ضرب دهدهی 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 دهدهی تغییر دهید. در یک جدول نحوه تفسیر هر یک از نمادهای ریزعمل‌ها را بنویسید.

1- Mode Control

www.homework.ir مشاوره و انجام پروژه هاي دانشجويي
دانلود فصلهايي از كتب دانشگاهي

۱۱ سازمان ورودی - خروجی

۱۱-۱ وسایل جانبی
۱۱-۲ واسطه ورودی - خروجی
۱۱-۳ انتقال غیرهمزمان داده
۱۱-۴ شیوه‌های انتقال
۱۱-۵ وقفه اولویت دار
۱۱-۶ دستیابی مستقیم به حافظه (DMA)
۱۱-۷ پردازنده ورودی و خروجی IOP
۱۱-۸ تبادل اطلاعات سری

۱۱-۱ وسایل جانبی[1]

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 ازیک منبع خارجی و یا انتقال نتایج بشکل مفهوم بی‌معنی است.

آشناترین راه ورود اطلاعات بداخل یک کامپیوتر استفاده از صفحه کلیدهایی شبیه ماشین‌های تحریر است که به شخص اجازه می‌دهد تا اطلاعات الفبا عددی را مستقیماً وارد نماید. هر بار که کلیدی فشرده شود، پایانه کاراکتر کد شده بفرم دودویی را به کامپیوتر می‌فرستد. بالاترین سرعت ممکن برای ورود اطلاعات با این روش به سرعت ماشین نویس بستگی دارد. از طرف دیگر، واحد پردازش مرکزی وسیله فوق‌العاده سریعی است که می‌تواند عملیات را با سرعت بسیار بالا انجام دهد. وقتی که اطلاعات ورودی از طریق یک صفحه کلید کند به پردازنده ارسال شود، پردازنده باید مادامی که منتظر رسیدن اطلاعات است بیکار بماند. برای استفاده کارا از یک کامپیوتر مقدار زیادی از برنامه‌ها و داده‌ها از قبل آماده و روی یک محیط ذخیره سازی مانند نوار یا دیسک مغناطیسی قرار داده می‌شود. سپس اطلاعات موجود در دیسک با سرعت بالایی به حافظه کامپیوتر منتقل می‌شود. نتایج محاسبات نیز به یک محیط

1- Peripheral Devices

ذخیره سازی سریع، مانند دیسک منتقل می‌گردد و بعداً از آنجا می‌تواند به یک چاپگر ارسال و لذا نتایج بصورت خروجی چاپ شده در اختیار قرار می‌گیرد.

وسایلی که تحت کنترل مستقیم کامپیوتر می‌باشند، گفته می‌شود روی خط[1] 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[2] و چاپگرها. وسایلی که ذخیره‌سازی کمکی را برای سیستم فراهم می‌نمایند عبارتند از دیسک‌ها و نوارهای مغناطیسی. وسایل جانبی وسیله‌های الکترومکانیکی و الکترومغناطیسی با پیچیدگی‌های خاص خود هستند. ما در اینجا بحث مختصری را از کار آنها را بدون جزئیات ساختمان داخلی آنها ارائه می‌نمائیم.

مونیتورهای تصویری[3] متداولترین وسیله‌های جانبی هستند. این وسایل از یک صفحه کلید بعنوان وسیله ورودی و یک واحد نمایش تصویری بعنوان وسیله خروجی تشکیل شده‌اند. انواع متنوعی از نمایشگرهای تصویر موجودند، ولی معمول‌ترین آنها از لامپ اشعه کاتودی[4] (CRT) استفاده می‌نمایند. CRT حاوی یک تفنگ الکترونی است که یک پرتوالکترونی 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 ولتاژ متغیری را دریافت کرده و موجب می‌شود تا پرتو در نقاط موردنظر بر صفحه بتابد و آن مکان را روشن نماید. سیگنالهای افقی و عمودی پرتو را هدایت می‌کنند و موجب حرکت جاروبی آن در امتداد عرض صفحه نمایش می‌شوند و لذا الگوی تصویری موردنظر بر روی صفحه ظاهر می‌گردد. یکی از ویژگیهای وسایل نمایش، مکان‌نمایی[5] است که محل قرار گرفتن کاراکتر بر روی صفحه نمایش را مشخص می‌کند. مکان نما را می‌توان به هر جایی در صفحه، برای هر کاراکتر، آغاز هر کلمه، یا هر خط انتقال داد. کلیدهای ویرایشی[6] براساس محل مکان‌نما اطلاعات را اضافه یا حذف می‌نمایند. پایانه نمایش[7] می‌تواند در روش تک‌نویسی[8] 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[9]، متن ویرایش شده ابتدا در یک حافظه محلی در داخل پایانه ذخیره می‌شود. این متن بعداً بصورت بلاکی از داده به کامپیوتر منتقل می‌گردد.

چاپگرها، داده‌ها یا متن خروجی کامپیوتر را بطور دائمی روی کاغذ ثبت می‌کنند. سه نوع عمده از چاپگرهای کاراکتری وجود دارند: چرخ‌گردان[10]، ماتریسی[11] 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 مناسبی قرار گرفته و سپس یک آهن ربای الکتریکی حرف موردنظر را روی نوار فشار می‌دهد. چاپگر ماتریسی حاوی مجموعه‌ای از نقطه‌ها در امتداد مکانیزم چاپ است. مثلاً یک چاپگر ماتریسی 5×7 قادر است 80 کاراکتر را در هر سطر چاپ کند که هر سطر نیز از هفت خط افقی تشکیل شده است. بنابراین هر خط 400=80×5 نقطه دارد. هر نقطه می‌تواند بسته به کاراکتر خاصی که روی خط چاپ می‌شوند، چاپ بشود یا نشود. چاپگر لیزری از یک استوانه عکاسی گردان استفاده می‌کند که با استفاده از آن تصاویر کاراکتر نقش می‌شوند. سپس الگو مشابه ماشین فتوکپی روی کاغذ منتقل می‌گردد.

غالباً نوارهای مغناطیسی برای ذخیره فایل‌های داده، مثل وضعیت حقوقی کارکنان یک شرکت بکار می‌رود. نحوه دستیابی به اطلاعات سری بوده و هر نوار از رکوردهایی تشکیل شده است و با عبور نوار از مقابل یک مکانیزم خواندن - نوشتن ثابت، می‌توان به رکوردها یکی پس از دیگری دست یافت. این روش یکی از ارزانترین و کندترین روشهای ذخیره‌سازی است ولی این مزیت را داراست که هنگام بکار گرفتن می‌توان نوار را خارج کرد. دیسک‌های مغناطیسی سطوح دوار سریعی دارند که با مواد مغناطیسی پوشانده شده‌اند. دستیابی اطلاعات با حرکت دادن یک مکانیزم متحرک خواندن و نوشتن و قرار دادن آن بر روی یکی از شیارها سطح مغناطیسی صورت می‌گیرد. از دیسک اکثراً برای ذخیره انبوه برنامه‌ها و داده‌ها استفاده می‌شود. در رابطه با نقش نوارها و دیسک‌ها به عنوان حافظه کمکی در بخش ۱-۱۲ بیشتر بحث خواهیم کرد.

وسایل ورودی و خروجی دیگری که در سیستم‌های کامپیوتری با آنها مواجه می‌شویم پلاترهای دیجیتال (رسام)، کاراکتر خوان نوری و مغناطیسی، مبدل‌های آنالوگ به دیجیتال و انواع تجهیزات دریافت[۱۲] (اکتساب) داده می‌باشند. از کامپیوترها برای کنترل فرآیندهای مختلف به صورت بلادرنگ[۱۳]، مانند ماشینهای ابزار، عملیات خط مونتاژ و فرآیندهای شیمیایی و صنعتی استفاده می‌شود. برای چنین کاربردهایی، باید روشی برای درک شرایط وضعیتی در فرآیند و ارسال سیگنال‌های کنترلی به فرآیند تحت کنترل فراهم شود.

سازمان ورودی - خروجی یک کامپیوتر تابعی از اندازه کامپیوتر و وسیله‌های متصل به آن است. تفاوت یک سیستم کوچک یا بزرگ عمدتاً به مقدار سخت افزاری که کامپیوتر برای ارتباط با وسایل جانبی در اختیار دارد و تعداد این وسایل بستگی دارد. چون هر وسیله جانبی رفتار متفاوتی با وسیله دیگر دارد، بحث درباره جزئیات اتصال بین کامپیوتر و هر دستگاه جانبی مشکل خواهد بود. در این فصل تکنیک‌های معینی را که در اکثر دستگاه‌های جانبی مشترک است ارائه می‌نمائیم.

11- Dot Matrix
12- Data Acquisition Equipment
13- Real - Time Syste

کاراکترهای الفبا عددی آسکی[1]

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 اسکی است. این کد، همانطور که در جدول ۱-۱۱ نشان داده شده است از هفت بیت برای کد کردن 128 کاراکتر استفاده می‌کند. هفت بیت کد با b_1 الی b_7 نشان داده شده‌اند، که در آن b_7 با ارزشترین بیت است. حرف A، مثلاً در ASCII بصورت 1000001 (ستون 100 و سطر 0001) نمایش داده می‌شود. کد اسکی شامل 94 کاراکتر چاپ شدنی و 34 کاراکتر غیرچاپ شدنی است که برای مقاصد کنترلی بکار می‌رود. کاراکترهای چاپ شدنی عبارتند از 26 حرف بزرگ الفبا از A تا Z، 26 حرف کوچک الفبا، 10 عدد از 0 تا 9 و 32 کاراکتر قابل چاپ خاص مانند %، * و $.

34 کاراکتر کنترل در جدول ASCII با نام‌های خاص بکار رفته‌اند. این کاراکترها مجدداً در قسمت پائین جدول همراه با نام عملیاتی‌شان آورده شده‌اند. کاراکترهای کنترلی برای مسیردهی داده‌ها و آرایش مسیر متن چاپ به شکلی از پیش تعیین شده به کار می‌روند. سه نوع کاراکتر کنترل موجود است: شکل‌دهنده‌ها، جداکننده‌های اطلاعات و کاراکترهای کنترل ارتباطات. شکل‌دهنده‌ها کاراکترهایی هستند که آرایش و نظم چاپ را کنترل می‌نمایند. این کاراکترها شامل کنترل‌های شناخته شده ماشین‌های تحریر، مانند کلید پسبَر[2] (BS)، جدول بندی افقی[3] (HT)، برگشت نورد[4] (CR) هستند. جداکننده‌های اطلاعات برای جداسازی اطلاعات به بخش‌هایی مانند پاراگراف و صفحه بکار می‌روند. این دسته شامل کاراکترهایی مانند جداکننده رکوردها[5] (RS) و جداکننده فایل‌ها[6] (FS) است. کاراکترهای کنترل ارتباطات هنگام ارسال متن بین پایانه‌های دور از هم مفیدند. مثال‌هایی از این نوع عبارتند از آغاز متن[7] STX و پایان متن[8] ETX، که برای کادربندی یک متن پیام هنگام ارسال از طریق یک رسانه ارتباطی بکار می‌روند.

اسکی یک کد هفت بیتی است، ولی در اکثر کامپیوترها داده‌ها بصورت کمیت هشت بیتی بنام بایت دستکاری می‌شوند. بنابراین کاراکترهای ASCII اغلب بصورت یک کاراکتر در هر بایت ذخیره می‌شوند. بیت اضافی بسته به کار برد گاهی برای مقاصد دیگر بکار می‌رود. مثلاً، بعضی چاپگرها کاراکترهای اسکی را بصورت هشت بیتی با مقدار 0 در بیت هشتم تشخیص می‌دهند. از 128 کاراکتر هشت بیتی با مقدار 1 در بیت هشتم برای دیگر سمبل‌ها مانند الفبای یونانی یا فونت‌های ایتالیک استفاده می‌شود. هنگام استفاده در انتقال داده‌ها، بیت هشتم ممکن است برای نشان دادن توازن کاراکتر کد شده با دودویی هم بکار رود.

1- American Standard Code for Information Interchange (ASCII)
2- Backspace
3- Horizontal Tabulation
4- Carriage Return
5- Record Separator
6- File Separator
7- Start of Text
8- End of Text

جدول ۱-۱۱ کد استاندارد آمریکایی برای تبادل اطلاعات (اسکی)

$b_4 b_3 b_2 b_1$	$b_7 b_6 b_5$ = 000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	'	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	—	o	DEL

کاراکترهای کنترل

NUL	هیچ	DLE	خروج از ارتباط داده
SOH	شروع تیتر	DC1	کنترل وسیله 1
STX	آغاز متن	DC2	کنترل وسیله 2
ETX	پایان متن	DC3	کنترل وسیله 3
EOT	استعلام	DC4	کنترل وسیله 4
ENQ	تصدیق	NAK	تصدیق منفی
ACK	بوق	SYN	همگامی در حالت بیکاری
BEL	پسبَر	ETB	پایان بلاک ارسال
BS	توقف افقی	CAN	لغو
HT	تعویض سطر	EM	پایان محیط
LF	توقف عمودی	SUB	جایگزین
VT	تعویض صفحه	ESC	خروج
FF	برگشت نورد	FS	جداکننده فایل‌ها
CR	انتقال به خارج	GS	جداکننده گروه‌ها
SO	انتقال به داخل	RS	جداکننده رکوردها
SI	انتقال به داخل	US	جداکننده واحدها
SP	فضای خالی	DEL	پاک‌کننده

۱۱-۲ واسطه ورودی - خروجی

واسطه ورودی - خروجی روشی را برای انتقال اطلاعات بین محیط‌های ذخیره سازی داخلی و وسیله‌های I/O خارجی در اختیار می‌گذارند. وسایل جانبی متصل به یک کامپیوتر اتصالات ارتباطی خاصی برای وصل به واحد پردازش نیاز دارند. منظور از اتصالات ارتباطی از میان برداشتن تفاوت‌های

موجود بین کامپیوتر مرکزی و هر وسیله جانبی است. تفاوت‌های مهم موجود عبارتند از:

۱- وسایل جانبی وسایلی الکترومکانیکی یا الکترومغناطیسی هستند و نحوه عملکرد آنها با CPU و حافظه که وسایلی الکترونیکی می‌باشند متفاوت است. بنابراین ممکن است تبدیل سیگنال‌ها 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.

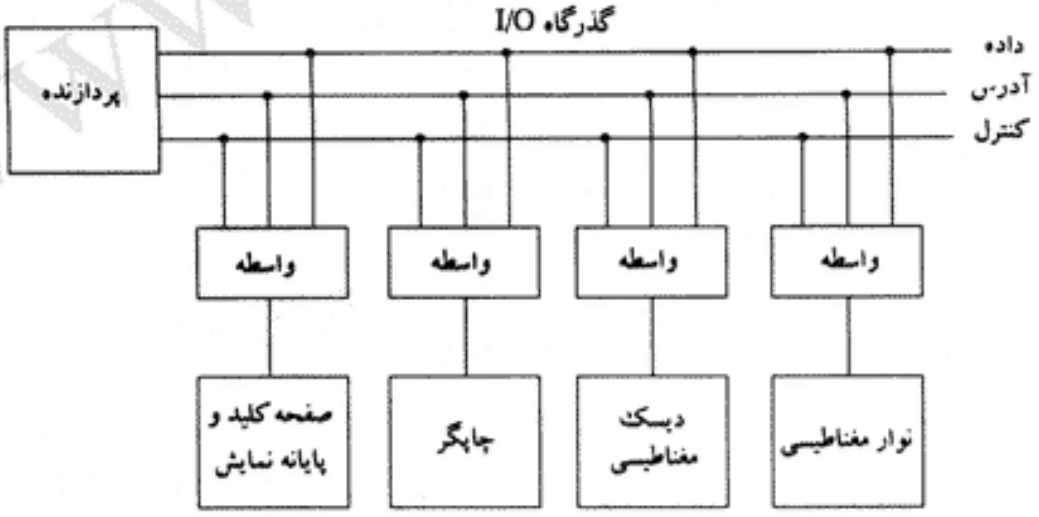
۳- کدها و قالب‌های داده‌ها در وسایل جانبی با قالب کلمات در CPU و حافظه تفاوت دارند.

۴- روش‌های عملکرد دستگاه‌های جانبی با یکدیگر متفاوت بوده و هر یک باید به نوعی کنترل شوند به نحوی که عملکرد دیگر دستگاه‌های متصل به CPU مختل نگردد.

برای برطرف کردن این تفاوت، سیستم‌های کامپیوتری قطعات و مدارهای خاصی بین CPU و دستگاه‌های جانبی برای مدیریت بر همه انتقال‌های ورودی و خروجی و همزمان سازی بین آنها را دارا هستند. این مدارات واحدهای واسطه[1] نامیده می‌شوند زیرا واسطه بین گذرگاه پردازنده و دستگاه جانبی می‌باشند. بعلاوه، هر وسیله جانبی ممکن است کنترل کننده خاصی برای خود داشته باشد که بر عملیات مکانیزم خاص در آن وسیله نظارت می‌کند.

گذرگاه I/O و ماژول‌های واسطه

نمونه‌ای از اتصالات ارتباطی بین پردازنده و چند وسیله جانبی در شکل ۱-۱۱ نشان داده شده است. گذرگاه I/O از خطوط داده، خطوط آدرس و خطوط کنترل تشکیل شده است. دیسک مغناطیسی، چاپگر



شکل ۱-۱۱ اتصال گذرگاه I/O به وسیله‌های ورودی- خروجی

1- Interface Unit

و پایانه عملاً در هر کامپیوتر چند منظوره بکار می‌روند. نوار مغناطیسی در بعضی از کامپیوترها بعنوان پشتیبان ذخیره سازی بکار می‌رود. برای هر وسیله جانبی، یک واحد واسطه وجود دارد. هر واحد واسطه، آدرس و کنترل دریافتی از گذرگاه I/O را دیکد کرده، سپس آنها را برای وسایل جانبی تفسیر، و سیگنال‌های لازم را برای کنترل کننده‌های وسایل جانبی فراهم می‌آورد. واحد واسطه انتقال داده را هماهنگ نموده و انتقال بین وسیله جانبی و پردازنده را مدیریت می‌کند. هر وسیله جانبی کنترل کننده خاص خود را داراست و توسط آن وسیله الکترومکانیکی خاصی را بکار می‌اندازد. مثلاً، یک کنترل‌کننده خاص حرکت کاغذ، زمانبندی چاپگر، و انتخاب کاراکتر چاپ شونده راکنترل می‌کند. یک کنترل‌کننده را می‌توان بطور جداگانه یا به همراه وسیله جانبی در مجموعه جاسازی کرد.

گذرگاه I/O پردازنده به تمام واسطه‌های وسایل جانبی متصل است. برای ارتباط با یک وسیله خاص، پردازنده آدرس یک وسیله را روی خطوط آدرس قرار می‌دهد. وقتی که واسطه آدرس خود را در می‌یابد، مسیر بین خطوط گذرگاه و وسیله‌ای راکه کنترل می‌نماید فعال می‌سازد. کلیه وسایل جانبی که آدرسشان با آدرس روی گذرگاه مربوط نمی‌شود بوسیله مدار واسطه غیرفعال می‌گردند.

همزمان با وجود آدرس روی خطوط آدرس، پردازنده یک کد عملیات را روی خطوط کنترل قرار می‌دهد. واسطه انتخاب شده به کد عملیات پاسخ داده و آنرا اجرا می‌نماید. کد عملیات برای I/O یک فرمان تلقی شده و در واقع دستورالعملی است که در مدار واسطه و واحد جانبی اجرا می‌گردد. تفسیر فرمان، به وسیله جانبی که پردازنده آنرا احضار می‌کند بستگی دارد. یک واحد واسطه ممکن است چهار نوع فرمان دریافت کند. این فرمان‌ها عبارتند از کنترل، وضعیت، داده خروجی، داده ورودی.

یک فرمان کنترل برای فعال کردن وسیله جانبی و اینکه چه کاری را انجام دهد صادر می‌گردد. مثلاً، به یک واحد نوار مغناطیسی ممکن است فرمان داده شود تا باندازه یک رکورد بعقب برگردد، نوار به ابتدای آن برگردانده شود، و یا از ابتدا به سمت جلو حرکت داده شود. فرمان کنترل خاص صادره به وسیله جانبی بستگی داشته و هر وسیله جانبی رشته فرمان‌های متوالی خاص خود را بسته به نوع عملکرد، تشخیص می‌دهد.

یک فرمان وضعیت برای آزمایش شرایط وضعیتی مختلف در مدار واسطه و وسیله جانبی بکار می‌رود. مثلاً، کامپیوتر ممکن است مایل به چک کردن وضعیت یک وسیله جانبی قبل از آغاز انتقال داده باشد. در هنگام انتقال ممکن است مدار واسطه وقوع یک یا چند خطا را تشخیص دهد. این خطاها با نشاندن[1] (1 شدن) بیت‌ها در ثبات وضعیت مشخص شده و پردازنده می‌تواند با فواصل زمانی معینی آنها را بخواند.

فرمان خروجی داده سبب می‌شود تا مدار واسطه داده‌ها را از گذرگاه به یکی از ثبات‌های خود انتقال دهد. بعنوان مثال واحد نوار را در نظر بگیرید. کامپیوتر نوار را با صدور یک فرمان کنترل به حرکت در می‌آورد. آنگاه پردازنده، وضعیت نوار را با یک فرمان وضعیت ملاحظه می‌کند و وقتی که نوار در محل

1- Set

صحیحی قرار گرفته باشد، پردازنده یک فرمان خروجی داده را صادر می‌نماید. واحد واسطه به آدرس و فرمان پاسخ داده و اطلاعات را از خطوط داده در گذرگاه به ثبات بافر منتقل می‌کند. سپس واحد واسطه با کنترل کننده نوار ارتباط برقرار کرده و داده‌ای را که قرار است روی نوار ذخیره شود ارسال می‌نماید.

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 می‌دهد. پردازنده وجود داده را توسط فرمان وضعیت چک کرده و سپس یک فرمان ورودی داده را صادر می‌کند. واحد واسطه داده راروی خطوط داده گذارده و پردازنده آن را می‌پذیرد.

گذرگاه I/O در برابر گذرگاه حافظه

علاوه برارتباط با I/O، پردازنده باید با واحد حافظه هم ارتباط برقرار کند. همانند گذرگاه I/O ، گذرگاه حافظه هم حاوی داده، آدرس و خطوط کنترل نوشتن و خواندن است. برای ارتباط گذرگاه‌های کامپیوتر با حافظه و I/O سه روش وجود دارد.

۱- از دو گذرگاه جداگانه، یکی برای حافظه و دیگری برای I/O استفاده شود.

۲- برای هر دو بخش حافظه و I/O از یک گذرگاه مشترک استفاده شود ولی هر کدام خطوط کنترل جداگانه داشته باشند.

۳- هر دو بخش حافظه و I/O از گذرگاه و خطوط کنترل مشترک استفاده نمایند.

در اولین روش، کامپیوتر دارای مجموعه گذرگاه جداگانه داده، آدرس و کنترل است، یک مجموعه برای دستیابی به حافظه و دیگری برای دستیابی به I/O. عمل دستیابی در کامپیوترهایی انجام می‌شود که دارای یک پردازنده I/O (IOP) علاوه بر واحد پردازش مرکزی (CPU) می‌باشند. حافظه با هردو واحد CPU و IOP از طریق گذرگاه خود ارتباط برقرار می‌کند. IOP هم با وسایل ورودی و خروجی از طریق گذرگاه I/O جداگانه‌ای با خطوط خاص آدرس، داده و کنترل ارتباط برقرار می‌کند. هدف از IOP فراهم کردن مسیر مستقلی برای انتقال اطلاعات بین وسایل خارجی و حافظه درونی است. پردازنده I/O گاهی اوقات کانال داده[1] هم خوانده می‌شود. در بخش ۷-۱۱ ما کار IOP را مفصل‌تر بحث خواهیم کرد.

I/O ایزوله در مقابل نگاشت یافته در حافظه[2]

بسیاری ازکامپیوترها برای انتقال اطلاعات بین حافظه و I/O از یک گذرگاه مشترک استفاده می‌کنند. تمایز بین انتقال حافظه و I/O بلحاظ استفاده از خطوط خواندن و نوشتن جداگانه است. CPU با تواناسازی یکی از دو خط خواندن و نوشتن مشخص می‌کند که آدرس روی خطوط آدرس مربوط به کلمه حافظه است یا به ثبات واسطه مربوط می‌باشد. خطوط کنترل خواندن I/O یا نوشتن I/O بهنگام

1- Data Channel 2- Isolated I/O

انتقال حافظه فعال می‌شوند. این پیکربندی آدرس تمام واسطه‌های I/O را از آدرس‌های تخصیص یافته به حافظه، جدا کرده و روش I/O ایزوله (جدا شده) برای تخصیص آدرس در گذرگاه مشترک نامیده می‌شود.

در آرایش I/O ایزوله، CPU دارای دستورالعمل‌های 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 می‌کند، آدرس مربوط به دستورالعمل را روی خطوط آدرس مشترک می‌گذارد. در همان زمان، خط کنترل خواندن I/O (برای ورودی) یا نوشتن I/O (برای خروجی) را فعال می‌سازد. این عمل قطعات خارجی را که به گذرگاه مشترک متصلند مطلع می‌سازد که آدرس روی خطوط آدرس متعلق به ثبات واسطه است و نه به یک کلمه حافظه. از طرف دیگر، وقتی که CPU دستورالعمل یا عملوندی را از حافظه برداشت می‌کند، آدرس حافظه را روی خطوط آدرس قرار داده و خط کنترل خواندن یا نوشتن حافظه را فعال می‌نماید. این عمل قطعات خارجی را مطلع می‌سازد که آدرس مربوط به کلمه حافظه است و نه متعلق به یک واسطه I/O.

روش I/O ایزوله آدرس‌های حافظه و I/O را طوری از هم جدا می‌کند که حافظه‌ها بوسیله آدرس‌های تخصیص یافته به مدار واسط تحت تأثیر قرار نمی‌گیرند 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 می‌رود که فقط از یک مجموعه از سیگنالهای خواندن و نوشتن استفاده می‌کنند و تمایزی بین آدرس حافظه و I/O وجود ندارد. این آرایش، I/O نگاشت یافته در حافظه[1] نامیده می‌شود. کامپیوتر با ثبات‌های مدارات واسطه مانند بخشی از حافظه سیستم رفتار می‌کند. در این روش نمی‌توان از آدرس‌های تخصیص یافته به ثبات‌های واسطه، برای کلمات حافظه استفاده کرد. بدین ترتیب محدوده آدرس‌های ممکن برای حافظه کاهش می‌یابد.

در سازمان I/O نگاشت یافته در حافظه، دستورالعمل خاصی برای ورودی و خروجی وجود ندارد. CPU قادر است داده‌های I/O واقع در ثبات واسطه را با همان دستوراتی که برای حافظه بکار می‌روند دستکاری کند. هر واسطه به صورت مجموعه‌ای از ثبات‌ها سازماندهی شده است و به درخواست‌های خواندن و نوشتن در فضای عادی آدرس پاسخ می‌دهد. نوعاً، قطعه‌ای از کل فضای آدرس برای ثبات‌های واسطه رزرو می‌شود، اما در حالت کلی، می‌توانند در هر آدرسی واقع شوند، بشرطی که کلمه‌ای از حافظه به همان آدرس پاسخ ندهد.

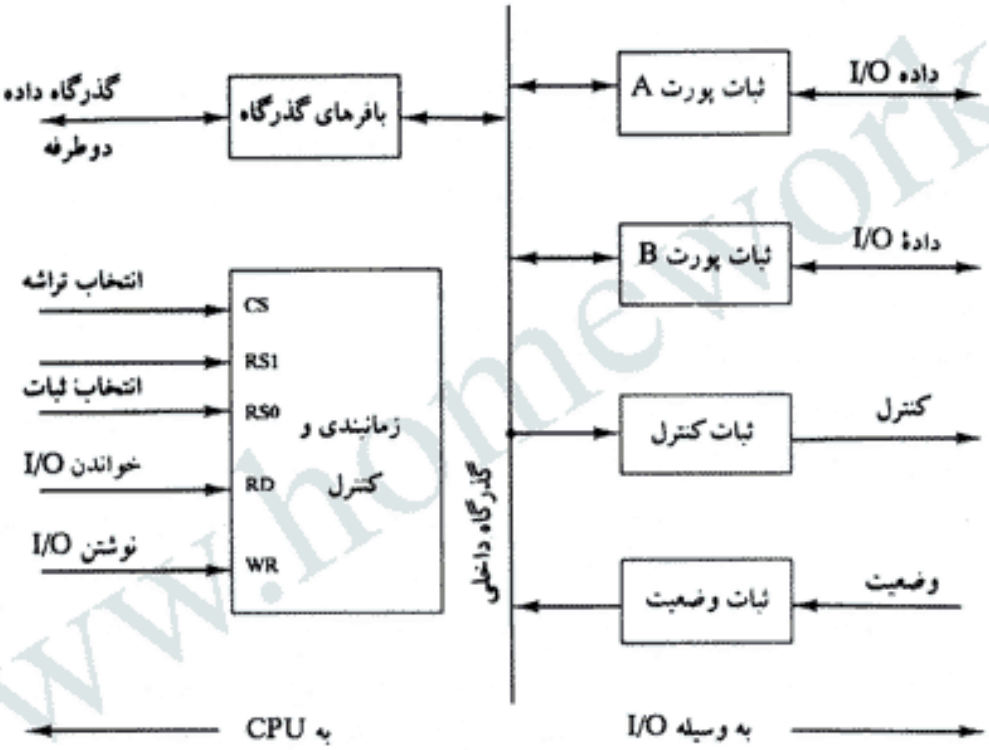
کامپیوترهایی که دارای سازمان I/O نگاشت یافته در حافظه هستند می‌توانند با استفاده از دستورالعمل‌های حافظه‌ای به داده‌های I/O دست یابند. این سازمان به کامپیوتر اجازه می‌دهد تا از دستورات یکسانی برای انتقال ورودی - خروجی و حافظه‌ای استفاده کنند. مزیت این است که دستورات بارکردن[2] و ذخیره کردن[3] که برای خواندن و نوشتن از حافظه بکار می‌روند برای ورود و یا خروج داده از ثبات‌های I/O نیز مورد استفاده قرار گیرند. معمولاً در کامپیوترها دستورات حافظه‌ای

1- Memory Mapped I/O 2- Load 3- Store

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 ۲-۱۱ بصورت بلاک دیاگرام نشان داده شده است. این واسطه متشکل از دو ثبات داده بنام پورت[1] (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
0	×	×	هیچ؛ گذرگاه داده در امپدانس بالا
1	0	0	ثبات پورت A
1	0	1	ثبات پورت B
1	1	0	ثبات کنترل
1	1	1	ثبات وضعیت

شکل ۲-۱۱ مثالی از واحد واسطه I/O

1- Port

ورودی‌هـای انتخاب تـراشـه و انـتخاب ثـبات، آدرس تـخصیص یـافته بـه مـدار واسـطه را مـعین می‌نمایند.خواندن و نوشتن I/O دو خط کنترل هستند کـه بـترتیب ورودی یـا خـروجی را مشـخص می‌کنند. چهار ثبات مستقیماً با وسایل I/O متصل به مدار واسطه ارتباط دارند.

داده I/O به واز وسایل می‌تواند به پورت A یا B منتقل گردد. مدار واسطه می‌تواند با یک وسیله ورودی ، یا یک وسیله خروجی و یا وسیله‌ای که هم ورودی و هم خروجی است کار کند. اگر واسطه به یک چاپگر وصل باشد، فقط داده را خارج می‌کند، و اگر به یک کاراکتر خوان سرویس دهد، فقط داده را وارد می‌کند. یک واحد دیسک مغناطیسی داده را در هر دو جهت بطور غیرهمزمان منتقل مـی‌نماید، بنابراین مدار واسطه می‌تواند از خطوط دو جهته استفاده کند. صدور فرمان به یک وسیله I/O با ارسال یک کلمه به یک ثبات مناسبی صورت می‌گیرد. در چنین سیستمی ، کد تابع در گذرگاه I/O لازم نیست زیرا کنترل به ثبات کنترل ارسال می‌شود، اطلاعات وضعیت از ثبات وضعیت، و داده‌هـا بـه و یـا از پورت‌های A و B انتقال می‌یابند. بنابراین انتقال اطلاعات داده، کنترل و وضـعیت هـمواره از طـریق گذرگاه مشترک داده انجام می‌شود. تمایز بین اطلاعات داده، کنترل یا وضعیت با توجه به ثبات خاص در مدار واسطه‌ای که CPU با آن ارتباط برقرار می‌کند صورت می‌گیرد.

ثبات کنترل، اطلاعات کنترل را از CPU دریافت می‌نماید. با بارکردن بیت‌های مـعینی در ثـبات کنترل، مدار واسطه و وسیله I/O متصل به آن می‌تواند در وضعیت‌های عملیاتی متنوعی قرار داده شود. مثلاً ، پورت A ممکن است بعنوان ورودی و پورت B بعنوان خروجی تعریف شوند. به یک واحد نوار مغناطیسی ممکن است فرمان برگشت به عقب داده شود یا آن را بسمت جلو به حرکت درآورد. بیت‌های ثبات وضعیت برای شرایط وضعیتی و ضبط خطاهایی که در حین انتقال داده‌ها ممکن است رخ دهد بکار می‌رود. مثلاً، یک بیت وضعیت ممکن است نشان دهد که یک قلم داده جدید از وسیله I/O وارد شده است. بیت دیگری در ثبات وضعیت ممکن است وقوع یک خطای توازن را در حین انتقال معین نماید.

ثبات‌های واسطه از طریق گذرگاه دو جهته داده با CPU ارتباط برقرار می‌کنند. گذرگاه آدرس واحد واسطه را از طریق انتخاب کننده تراشه و دو ورودی انتخاب ثبات انتخاب می‌کند. برای تشخیص آدرس اختصاص-یافته به ثبات‌های واسطه، یک مدار خارجی (معمولاً یک دیکدر) بکار گرفته می‌شود. این مدار ورودی انتخاب تراشه [1] (CS) را وقتی که واسطه بوسیله گذرگاه آدرس انتخاب شود فعال می‌سازد. دو ورودی انتخاب ثبات RS1 و RS0 معمولاً به دو خط کم ارزشتر گذرگاه آدرس وصل می‌شوند. این دو ورودی یکی از چهار ثبات را طبق جدول همراه دیاگرام انتخاب می‌نمایند. محتوای ثبات انتخاب شده از طریق گذرگاه داده، هنگامی که سیگنال خواندن I/O فعال شود، به CPU انتقال می‌یابد. CPU اطلاعات دودویی را از طریق گذرگاه داده وقتی که ورودی نوشتن I/O فعال شود، به ثبات انتخاب شده منتقل می‌سازند.

1- Chip Select

۳-۱۱ انتقال غیرهمزمان داده

عملیات درونی در یک سیستم دیجیتال بوسیله پالس‌های ساعت که بوسیله یک مولد پالس تهیه می‌شود همزمان می‌شوند. پالس‌های ساعت به تمام ثبات‌های درون یک واحد اعمال شده و کلیه انتقال داده‌ها بین ثبات‌های درونی همزمان با وقوع پالس ساعت رخ می‌دهند. معمولاً هر دو واحد CPU و واسطه I/O مستقل از یکدیگر طراحی می‌شوند. اگر ثبات‌های داخلی مدار واسطه از ساعت مشترکی با ثبات‌های CPU استفاده کنند، انتقال بین دو واحد همزمان[1] (یا همگام) خوانده می‌شود. در اکثر موارد، زمانبندی درونی در هرواحد از دیگری مستقل است زیرا هر یک از ساعت اختصاصی خود برای ثباتهای خودشان استفاده می‌کنند. در این صورت دو واحد را غیرهمزمان[2] یا ناهمگام با هم خوانیم. این روند بطور گسترده‌ای در بیشتر سیستم‌های کامپیوتری بکار گرفته شده‌اند.

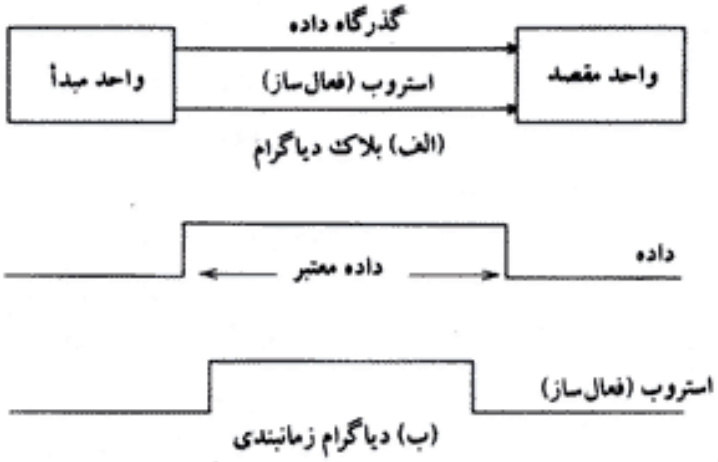
انتقال غیرهمزمان بین دو واحد مستقل مستلزم ارسال سیگنال‌های کنترل بین واحدهای ارتباط یافته است تا زمانی را که داده باید انتقال یابد مشخص کنند. یکی از راه‌های دستیابی به این هدف تهیه پالس فعال‌ساز[3] (استروب) بوسیله یکی از واحدها، برای مشخص کردن زمان وقوع انتقال به واحد دیگر است. روش دیگر که عموماً بکار می‌رود همراهی اقلام داده با یک سیگنال کنترل است که حضور داده را در گذرگاه مشخص می‌نماید. واحدی که داده را دریافت می‌کند با یک سیگنال کنترل دیگر دریافت آنرا تأیید می‌نماید. این نوع توافق بین دو واحد مستقل ، دست دهی[4] خوانده می‌شود.

روش پالس فعال‌ساز[5] و دست دهی در انتقال غیرهمزمان داده‌ها منحصر به انتقال I/O نیست. در واقع این روش‌ها بطور گسترده‌ای و در فرصت‌های متعددی که انتقال داده بین دو واحد صورت می‌گیرد بکار می‌روند. در حالت کلی ما واحد انتقال دهنده را بعنوان منبع[6] و واحد دریافت کننده را مقصد[7] می‌خوانیم. مثلاً ضمن عمل خروج داده و یا نوشتن، CPU یک منبع است ولی بهنگام ورود یا خواندن نقش یک مقصد را بعهده دارد. مرسوم است که انتقال غیرهمزمان بین دو واحد مستقل را بوسیله دیاگرام زمانبندی مشخص نمایند تا بدینوسیله رابطه زمانی موجود بین سیگنالهای کنترل و داده‌ها در گذرگاه مشخص شود. رشته سیگنال‌های کنترل در یک انتقال غیرهمزمان به این بستگی دارد که آغازگر انتقال، واحد منبع یا مقصد باشد.

کنترل بوسیله استروب

روش کنترل انتقال داده غیرهمزمان بوسیله سیگنال استروب از یک خط کنترل برای هر انتقال استفاده می‌کند. استروب ممکن است بوسیله واحد منبع و یا مقصد فعال گردد. شکل ۳-۱۱ (الف) یک انتقال با آغازگر منبع را نشان می‌دهد. 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

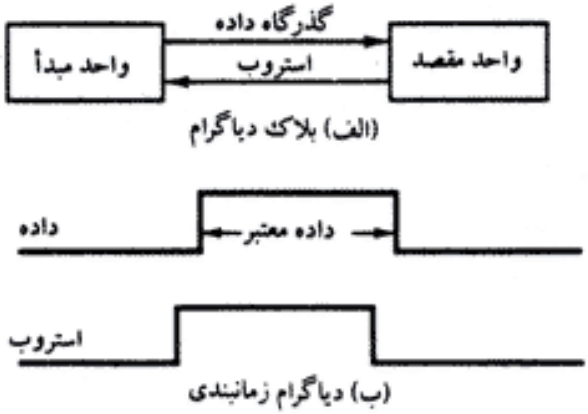


شکل ۳-۱۱ فعال‌سازی از سوی مبدأ برای انتقال

می‌کند. معمولاً گذرگاه برای انتقال تمام بایت یا کلمه چند خط دارد. استروب یک خط تک واحدی است که واحد مقصد را بهنگام قرار گرفتن یک داده معتبر بر روی گذرگاه ، مطلع می‌سازد.

همانطور که در دیاگرام ۳-۱۱ (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 استروب را فعال می‌سازد. اطلاعات روی گذرگاه داده وسیگنال استروب برای زمانی کافی در حالت فعال باقی می‌مانند تا به واحد مقصد اجازه دریافت داده را بدهند. اغلب واحد مقصد لبه پائین رونده پالس استروب را برای انتقال محتوای گذرگاه داده به یکی از ثبات‌های درونی بکار می‌برد. واحد منبع به فاصله کوتاهی پس از غیرفعال کردن پالس استروب، داده را از گذرگاه برمی‌دارد. در واقع، منبع نیازی به تغییر داده روی گذرگاه ندارد. غیرفعال شدن سیگنال استروب بیانگر این واقعیت است که گذرگاه داده دارای داده معتبر نیست. داده جدید معتبر پس از فعال شدن مجدد استروب وجود خواهد داشت.

شکل ۴-۱۱ انتقال داده‌ای راکه 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 ۴-۱۱ استروب از سوی مقصد برای انتقال

روی گذرگاه داده قرار گیرد تا واحد مقصد آن را بپذیرد. لبه پایین رونده پالس استروب مجدداً برای تحریک ثبات مقصد بکار می‌رود. سپس واحد مقصد، منبع را غیرفعال می‌نماید. منبع نیز داده را پس از گذشت زمان از پیش تعیین شده‌ای از روی گذرگاه حذف می‌کند.

در اکثر کامپیوترها پالس استروب در واقع بوسیله پالس‌های ساعت در CPU کنترل می‌شود. CPU همواره کنترل گذرگاه‌ها را در دست دارد و به واحدهای خارجی اطلاع می‌دهد که چگونه داده‌های خود را منتقل سازند. مثلاً، استروب شکل ۳-۱۱ می‌توانست یک سیگنال کنترل نوشتن درحافظه از CPU به واحد حافظه باشد. منبع، که CPU است، کلمه‌ای را روی گذرگاه داده قرار داده و واحد حافظه را، که مقصد است مطلع می‌نماید، که این خود یک عمل نوشتن است. بطورمشابه استروب شکل ۴-۱۱ می‌توانست یک سیگنال کنترل خواندن حافظه از CPU به یک حافظه باشد. مقصد، یعنی CPU، عمل خواندن را آغاز می‌کند تا حافظه را که منبع است برای قرار دادن کلمه انتخابی روی گذرگاه داده مطلع سازد.

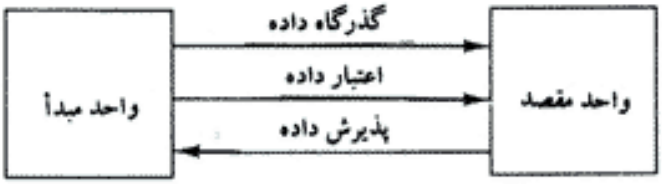
انتقال داده بین CPU و یک واحد واسطه مشابه فوق می‌باشد. انتقال داده بین یک واسطه ویک وسیله I/O معمولاً بوسیله مجموعه‌ای از خطوط دست دهی کنترل می‌گردد.

دست دهی

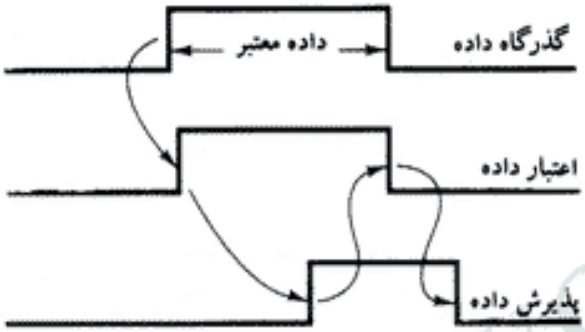
عیب روش استروب این است که واحد منبع که انتقال را شروع می‌کند قادر نیست بداند که آیا واقعاً مقصد داده را که بر روی گذرگاه داده قرار گرفته دریافت کرده است یا خیر. بطورمشابه، یک واحد مقصد که انتقال را آغاز می‌کند نمی‌داند که آیا منبع داده را روی گذرگاه قرار داده است یا نه. روش دست‌دهی این مسئله را با معرفی یک سیگنال کنترل دیگر که پاسخی برای واحد آغازگر انتقال است، حل می‌کند. اساس روش دست‌دهی دوسیمی انتقال داده بشرح زیر است. یک خط کنترل در جهت ارسال داده در گذرگاه ،یعنی از منبع به مقصد است. این خط بوسیله منبع برای مطلع ساختن مقصد از وجود داده معتبر بر روی گذرگاه بکار می‌رود. خط دیگر در جهت دیگر، یعنی از مقصد به منبع می‌باشد. این خط بوسیله واحد مقصد بکار رفته و منبع را مطلع می‌سازد که آیا می‌تواند داده بپذیرد یا خیر. دنباله سیگنال‌های کنترل در طول انتقال به واحدی بستگی دارد که انتقال را آغاز می‌نماید.

شکل ۵-۱۱ رویه انتقال داده را وقتی که بوسیله منبع آغاز شود نشان می‌دهد. دو خط دست دهی عبارتند از "داده معتبر"[1] که بوسیله واحد منبع، و "پذیرش داده"[2] 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 ممکنی 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 واحدمنبع سیگنال داده معتبر را غیرفعال ساخته و داده را بر روی گذرگاه غیرمعتبر می‌نماید. پس از آن واحد مقصد سیگنال پذیرش داده را

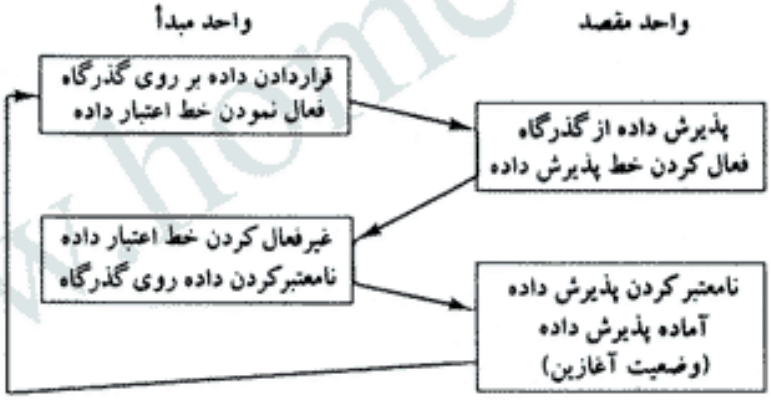
1- Data valid 2- Data accepted



(الف) بلاک دیاگرام



(ب) دیاگرام زمانبندی

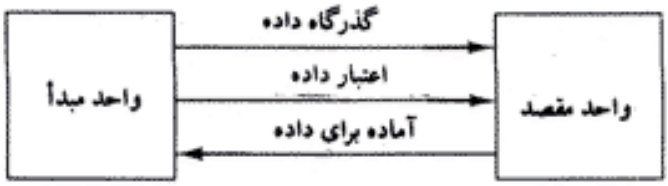


(ج) دنباله وقایع

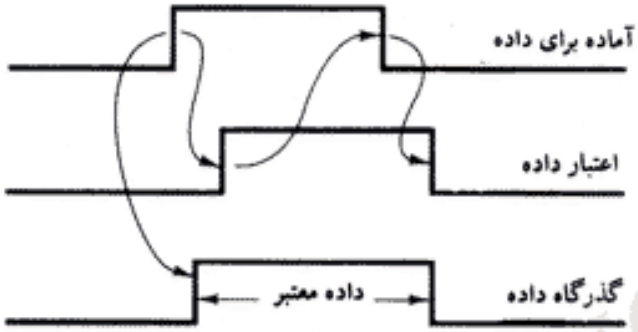
شکل ۵-۱۱ انتقال از سوی مبدأ با استفاده از دست‌دهی

غیرفعال نموده، و سیستم به حالت اولیه خود باز می‌گردد. منبع، تا هنگامی که مقصد آمادگی پذیرش داده جدید را با غیرفعال کردن سیگنال پذیرش داده انجام ندهد، داده بعدی را ارسال نمی‌کند. در این روش هر زمان تاخیری از یک حالت به حالت بعدی مجاز است و اجازه می‌دهد تا هر واحد با سرعت انتقال داده خاص خودکار کند. سرعت انتقال را کندترین واحد معین می‌کند.

انتقال آزادشده بوسیله مقصد که بسمت روش دست‌دهی انجام می‌شود در شکل ۶-۱۱ نشان داده شده است. توجه کنید نام سیگنال تولید شده بوسیله واحد مقصد به "آماده برای داده" تغییر یافته است تا



(الف) بلاک دیاگرام



(ب) دیاگرام زمانبندی



(ج) دنباله وقایع

شکل ۶-۱۱ انتقال از سوی مقصد با استفاده از دست‌دهی

منعکس کننده مفهوم جدید باشد. واحد منبع در این حالت تا دریافت سیگنال "آماده برای داده" از واحد مقصد، داده را روی گذرگاه قرار نمی‌دهد. از این به بعد روند دست دهی الگوی یکسانی را مانند آنچه در حالت آغاز از سوی منبع دنبال می‌کند. توجه کنید که اگر سیگنال آماده برای داده را بعنوان پذیرش داده در نظر بگیریم، رشته وقایع در هر دو مورد یکسان خواهد بود. در واقع تنها اختلاف بین انتقال آغاز با منبع و آغاز با مقصد در انتخاب حالت اولیه آنهاست.

روش 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"[1] قابل تشخیص است، وطی آن اگر انتقال داده در مدت زمان از پیش تعیین شده‌ای پایان نپذیرد آلارم صادر خواهد کرد. مکانیزم اتمام وقت با استفاده از یک ساعت داخلی هنگامی که واحد، یکی از سیگنال‌های کنترل دست دهی را فعال سازد اجرا می‌شود. اگر سیگنال بازگشت دست دهی در مدت زمان معینی بازنگردد، واحد رخداد خطایی را فرض خواهد کرد. سیگنال اتمام وقت برای وقفه پردازنده نیز می‌تواند مورد استفاده قرار می‌گیرد و لذا روال سرویس‌دهی مناسبی را برای رفع خطا اجرا نماید.

انتقال سری غیرهمزمان

انتقال داده بین دو واحد ممکن است موازی و یا سری باشد. در انتقال موازی داده، هر بیت از پیام مسیرخاص خود را داشته و کل پیام بطور همزمان انتقال می‌یابد. این بدان معنی است که یک پیام n بیتی باید از طریق n مسیرهادی جداگانه منتقل شود. در انتقال سری، هر بیت در پیام یکی پس از دیگری ارسال می‌شود. این روش مستلزم استفاده از یک جفت رسانا یا یک مسیر رسانا و یک زمین مشترک است. انتقال موازی سریعتر است ولی نیاز به چندین سیم دارد. این روش برای مسیرهای کوتاه و مواقعی که سرعت اهمیت دارد بکار می‌رود. انتقال سری کندتر است ولی ارزانتر می‌باشد زیرا فقط یک جفت رسانا نیاز دارد.

انتقال سری می‌تواند همزمان یا غیر همزمان باشد. در انتقال همزمان هر دو واحد یک فرکانس ساعت مشترک داشته و بیت‌ها مرتباً با سرعتی که بوسیله پالس‌های ساعت دیکته می‌شود انتقال می‌یابند. در انتقال سری راه دور، هر واحد بوسیله یک پالس ساعت جداگانه با فرکانس یکسان راه‌اندازی می‌شود. سیگنال‌های همزمان سازی بطور متناوب بین دو واحد ارسال می‌شوند تا ساعتهای آنها را همگام با یکدیگر نگهدارد. در ارسال غیرهمزمان، اطلاعات دودویی فقط زمانی ارسال می‌شود که موجود باشد و هرگاه اطلاعاتی برای ارسال وجود نداشته باشد خط بی‌کار می‌ماند این برخلاف انتقال همزمان است که در آن بیتها باید بطور پیوسته ارسال شوند تا فرکانس ساعت در هر دو واحد بایکدیگر همزمان باقی بمانند. انتقال سری همزمان در بخش ۸-۱۱ بیشتر بحث شده است.

تکنیک انتقال سری غیرهمزمان داده که در اکثر پایانه‌های محاوره‌ای بکار می‌رود از بیت‌های خاصی که در دو طرف کد کاراکتر قرار داده می‌شود استفاده می‌کند. با این تفکیک ، هر کاراکتر متشکل از سه بخش است: بیت شروع، بیت‌های کاراکتر، و بیت‌های توقف. قرار داد این است که وقتی کاراکتری انتقال نیابد، فرستند در وضعیت 1 منطقی قرارمی‌گیرد. اولین بیت، بیت شروع نام دارد و در وضعیت 0 قرار داشته و برای مشخص کردن شروع کاراکتر بکار می‌رود. بیت آخر که بیت توقف نامیده می‌شود همواره 1 است. مثالی از این قالب در شکل ۷-۱۱ نشان داده شده است.

1- Timeout