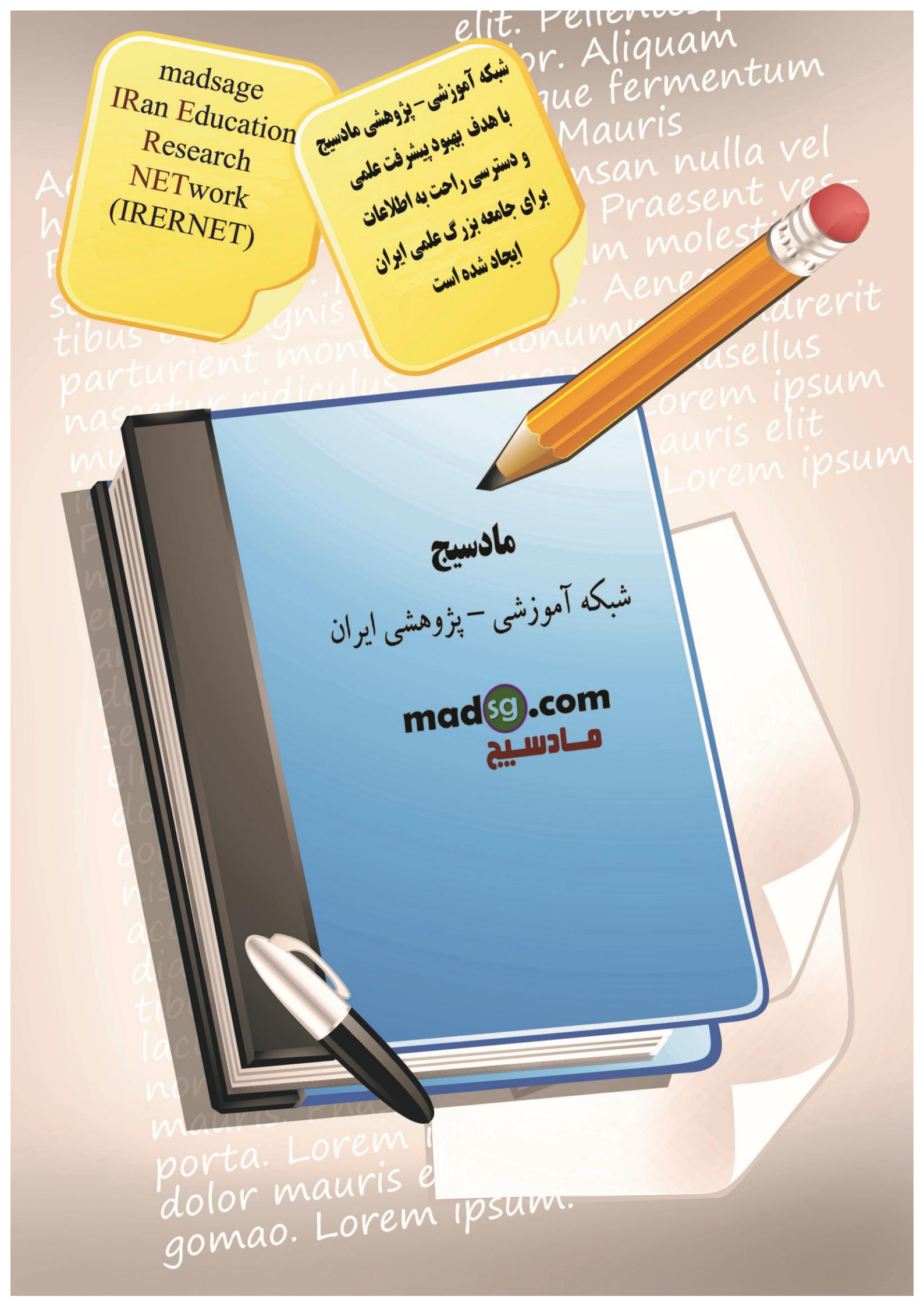

madsage
IRan Education
Research
NETwork
(IRERNET)
شبکه آموزشی-پژوهشی مادسیج
با هدف بهبود پیشرفت علمی
و دسترسی راحت به اطلاعات
برای جامعه بزرگ علمی ایران
ایجاد شده است
مادسیج
شبکه آموزشی - پژوهشی ایران
madsg.com
مادسیج

دانشکده فناوری اطلاعات و کامپیوتر

معماری کامپیوتر

دکتر حمید حسن پور

هو اللطیف

فهرست

فصل دوم- طراحی واحد محاسبه و منطق

۱- نحوه نمایش داده در کامپیوتر

۱-۱ : اعداد بی سی دی (BCD)

۱-۲ : اعداد افزایش سه تایی (EX - 3)

۱-۳ : کد گری (Gray)

۱-٤ : کد گری - افزایش سه تایی (EX-3 Gray)

۱-۵ : اعداد ممیز شناور (Floating Point)

۲- مدارات جمع کننده

۲-۱ : طراحی جمع کننده کامل (Full Adder)

۲-۲ : طراحی جمع کننده سریال (Serial Adder)

۲-۳ : طراحی جمع کننده با رقم نقلی پله‌ای (Ripple Carry Adder)

۲-٤ : طراحی جمع کننده با پیش بینی رقم نقلی (Carry Lookahead Adder)

۳- مدارات ضرب کننده

۳-۱ : ضرب کننده برای دو عدد قدر مطلق علامت

۱-۳ : طراحی واحدکنترل به روش ریزبرنامه سازی (Micro Programmed)

۲- سازمان حافظه (Memory Organization)

۲-۱ : جنس حافظه

۲-۱-۱ : حافظه نیمه رسانا (Semiconductor)

۲-۱-۲ : حافظه حلقه های مغناطیسی(Magnetic Core)

۲-۲ : سلسله مراتب ذخیره سازی اطلاعات

۲- ۳ : انواع Cache

۲- ۳ -۱ Direct Mapping

۲- ۳ -۲ Set – Associative

۲-۳-۳ – Associative mapping

۲-٤ :حافظه انجمنی

۳-روشهای متفاوت انتقال اطلاعات بین کامپیوتر ودستگاههای جانبی

۳-۱ :برنامه ریزی ورودی / خروجی (programed I/O)

۳-۲ : وقفه(Intrupt)

۳-۳ : دسترسی مستقیم به حافظه DMA ((Direct Memory Access)

۲- آشنایی با سطوح طراحی و ادوات مورد استفاده در رجیسترها

۲-۱ : سطوح طراحی در کامپیوتر

همانطور که می دانید طراحی در کامپیوتر در سه سطح انجام می شود.

۱) سطح گیت (Gate Level)

۲) سطح ثبات (Register Level)

۳) سطح پردازنده (Processor Level)

به عنوان مثال طراحی یک ساعت دیجیتال رادر نظر می گیریم .

۲-۱-۱- درسطح گیت

یک مدار مجتمع (IC) را در نظر بگیرید که زمان رامی شمارد وبه صورت Seven Segment نمایش می دهد .

IC timer

(شکل ۲-۱)

۲-۱-۲- درسطح ثبات

در اینجا به کمک گیتهای موجود طراحی را انجام می دهیم وثباتهایی که کارهای بخصوصی را انجام می دهند با یکدیگر ارتباط دارند .

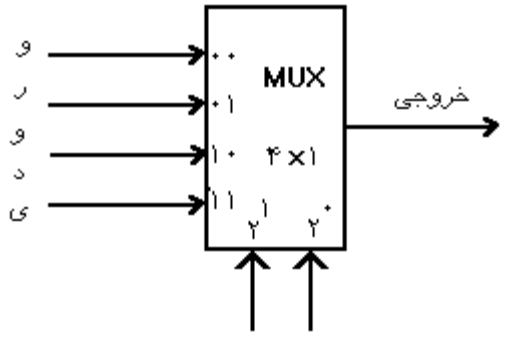
۲-۱-۳- در سطح پردازنده

در این سطح واحد پردازنده سیستم طراحی می شود، که این بحث مربوط به درس ریزپردازنده ها می باشد . در درس معماری طراحی رجیسترها مورد بحث است وبیشتر مطالب مربوط بـه طراحی پیاده سازی این قسمت از کامپیوتراست.

۲-۲ : مالتی پلکسر(Multi plexer)

مالتی پلکسر یک مدار ترکیبی است که به کمک آن می توان اطلاعات واقع در چند مسیر را به یک مسیر هدایت نمود . اگر در مالتی پلکسرها 2^n مسیر ورودی وجود داشته باشد ؛ n خط انتخاب برای این مسیرها وجود دارد که به کمک آن می توان اطلاعات موجود در یک مسیر ورودی را به خروجی منتقل نمود.

در مالتی پلکسر MUX(m ×n)، m تعداد مسیرهای ورودی وn تعداد بیتهای یک مسیر می باشد . در شکل زیر یک مالتی پلکسر 1 × 4 راداریم .



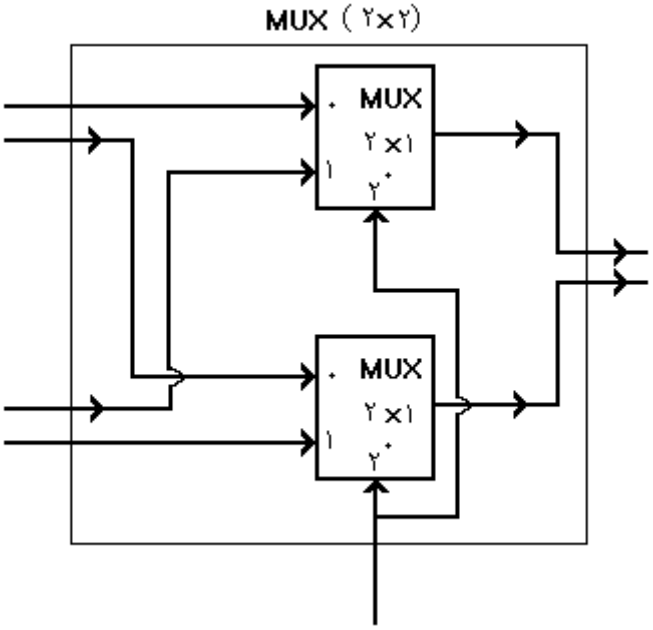
(شکل ۲-۲)

نکته: مسیر هرچند بیتی باشد خط انتخاب همیشه یک بیتی است وباید همواره ارزش آن معلوم شود. خروجی نیز یک مسیر می باشد که ممکن است هر چند بیتی باشد.

روشهایی وجود دارد که بتوان مالتی پلکسرها را به گونه‌ای به هم مرتبط نمود تا تعداد مسیرهای ورودی ویا تعداد بیتهای خروجی را افزایش داد.

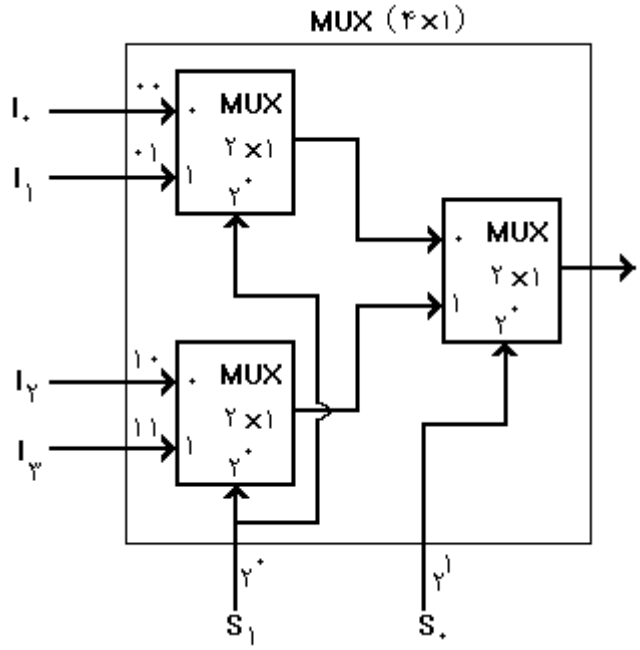
افزایش تعداد ورودی را می توان براساس اتصال درختی مالتی پلکسرها وعمل افزایش تعداد بیتهای خروجی را می توان با اتصال موازی مالتی پلکسرها انجام داد.

مثال : با استفاده از Mux (2 × 1) ، Mux (2 × 2) را طراحی کنید .



(شکل ۲-۳)

مثال : می خواهیم با استفاده از(2 × 1) Mux، یک (4 × 1) Mux بسازیم .



(شکل ۲-٤)

۲-۲-۱- کاربردهای مالتی پلکسر

۱- برای هدایت اطلاعات از بین چندین منبع ورودی، به یک مقصد خروجی مشترک .

۲- برای پیاده سازی توابع بولی وترکیبی.

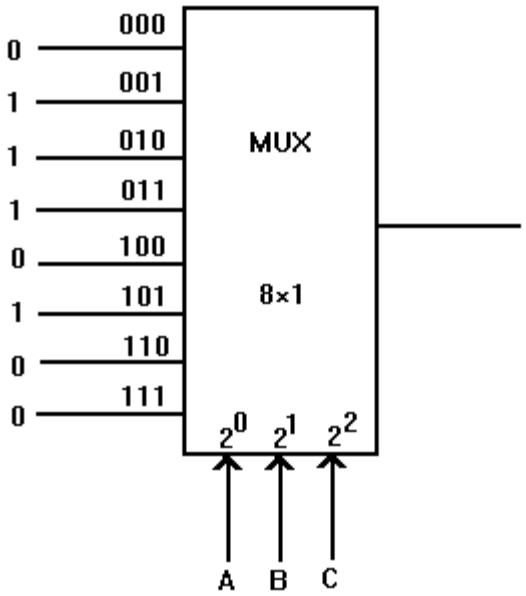
۳- برای طراحی سیستم باس یا گذرگاه (BUS).

۲-۲-۱-۲- کاربرد Mux در طراحی توابع

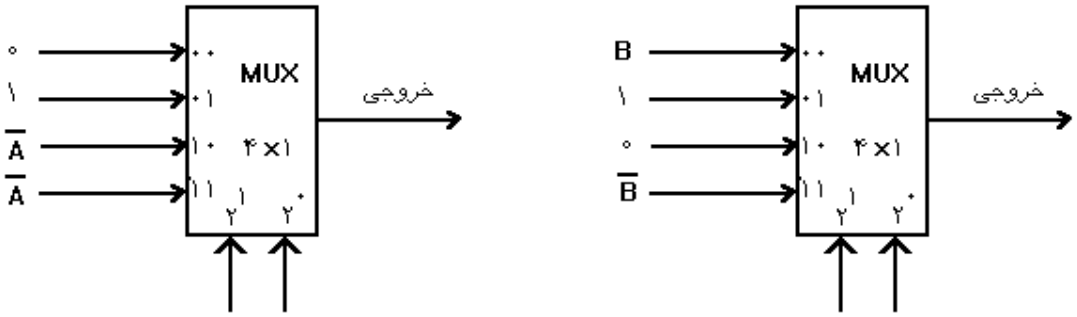
هر تابع n متغیره را می توان به کمک $MUX(2^n \times 1)$ پیاده سازی نمود .در این پیاده سازی متغیرهای تابع به خطوط انتخاب متصل شده وبسته به آنکه این تابع حاوی چه مینترمهایی باشد در ورودی برای شماره های مربوطه یک قرار می دهیم.

به عنوان مثال : تابع $F(A,B,C) = \Sigma (1,2,3,5)$ را به کمک Mux (8×1) پیاده سازی نمایید.

سپس همین تابع را با استفاده از Mux (4×1) طراحی کنید.



(شکل ۲-۵)



A \ BC	$\overline{B}\overline{C}$	$\overline{B}C$	$B\overline{C}$	BC
A	۴	⑤	۶	۷
$\overline{A}$	۰	①	②	③

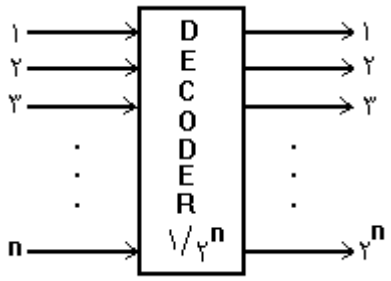
B \ AC	$\overline{A}\overline{C}$	$\overline{A}C$	$A\overline{C}$	AC
B	②	③	۶	۷
$\overline{B}$	۰	①	۴	⑤

(شکل ۲-۶)

۲-۳ : کدگشا (Decoder)

مداری ترکیبی است با n ورودی و 2^n خروجی که با اعمال یک کد بـه ورودی آن تنهـا یک

خط از خروجی فعال می شود.



(شکل ۲-۷)

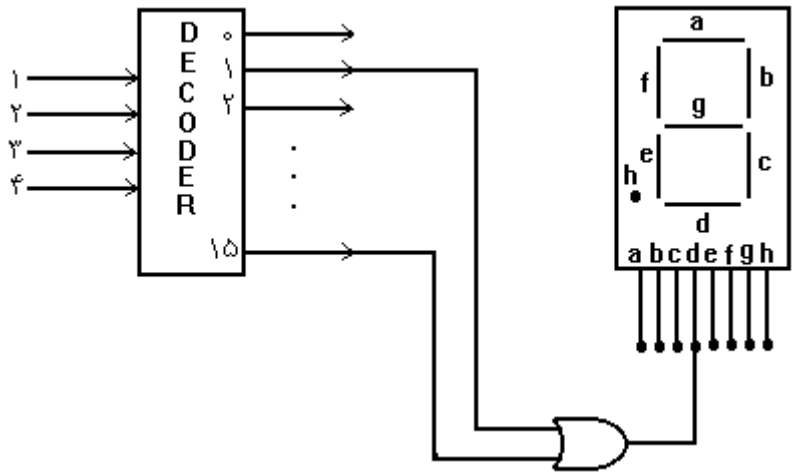
۲-۳-۱- کاربردهای کد گشا

۱- نمایش اعداد بی سی دی بر روی Seven Segment

۲- آدرس دهی خطوط حافظه

۳- پیاده سازی توابع ترکیبی

کاربرد اول – نمایش اعداد بی سی دی بر روی Seven Segment "7seg"

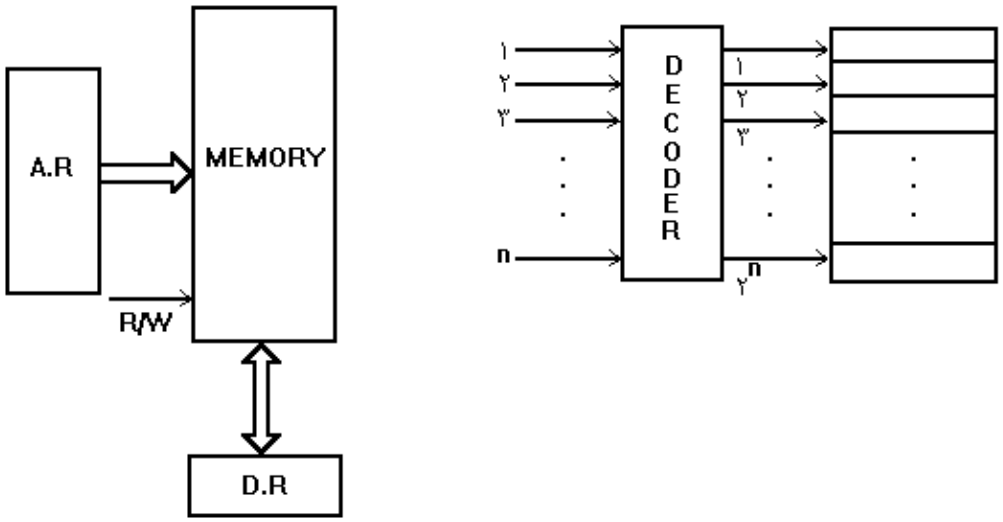


(شکل ۲-۸)

برای مثال Segment d ، هم در0 هم در 9 وجود دارد. پس این دو سیم رابوسیله گره به هم وصلی نمی کنیم ، چون IC می سوزد. بنابراین باید از یک گیت OR برای این منظور استفاده شود.

کاربرد دوم - آدرس دهی خطوط حافظه

مثالی از یک آدرس دهی که توسط کدگشا انجام می گیرد، ثبات آدرس می باشد که آدرس محلی ازحافظه را به صورت کد نگاه می دارد که این کد بوسیله کد گشا رمز گشایی می شود.



(شکل ۲-۹)

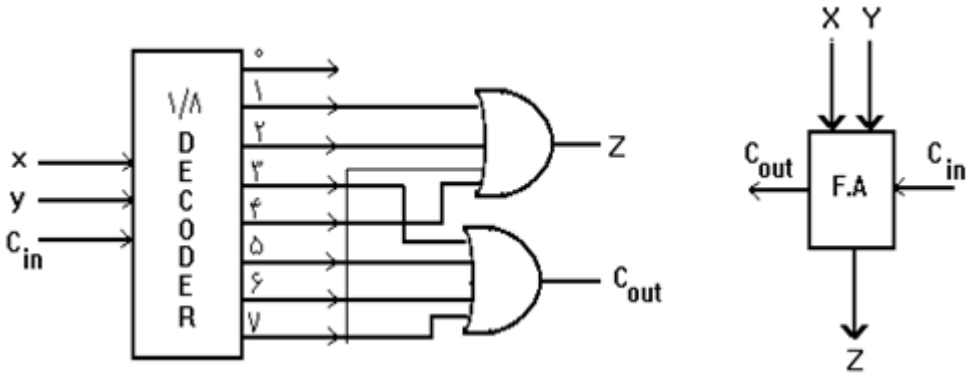
کاربرد سوم - پیاده سازی توابع ترکیبی

هر تابع با n ورودی و m خروجی را می توان به کمک یک کدگشا $\frac{1}{2^n}$ و m عدد گیت OR پیاده

سازی نمود.

مثال : با استفاده از کد گشا وگیتهای مورد نیاز یک جمع کننده کامل (Full adder) بسازید .

x	y	C_{in}	Z	C_{out}
۰	۰	۰	۰	۰
۰	۰	۱	۱	۰
۰	۱	۰	۱	۰
۰	۱	۱	۰	۱
۱	۰	۰	۱	۰
۱	۰	۱	۰	۱
۱	۱	۰	۰	۱
۱	۱	۱	۱	۱

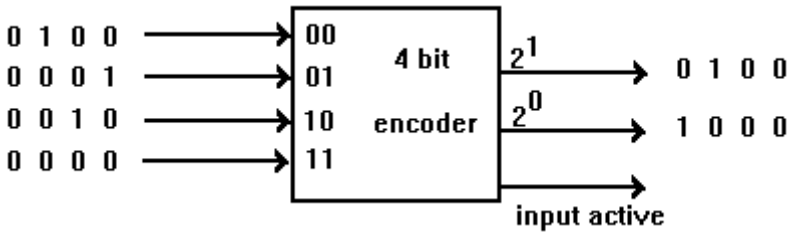


(شکل ۲-۱۰)

۲-٤ : کدگذار (encoder)

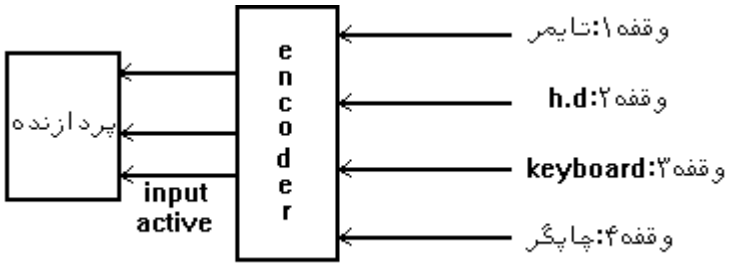
یک مدار ترکیبی با 2^n ورودی و n خروجی می باشد. در کد گذار در صورتیکه پایه‌ای از ورودی را فعال کنیم در خروجی شماره آن پایه در اختیار قرار می گیرد ود رواقع عکس عمل کد گشا را انجام می دهند.

شکلی از یک کد گذار با ٤ ورودی و۲ خروجی



(شکل۲-۱۱)

از کاربردهای آن می توان دریافتن آدرس وقفه های مختلفی که به پردازنده ای فرستاده می شود اشاره کرد.



(شکل ۲-۱۲)

دیگر نیاز نیست که برای هر وسیله که وقفه نیاز دارد ، خطوط جداگانه‌ای برای آدرس ایجاد کرد وبا یک کد گذار می توان 2^n وسیله را با n خط ، آدرس دهی نمود.

در اینجا ما به دو مشکل برمی خوریم :

اشکال ۱) : اگـر بطـور همزمـان در ورودی بـیش از یـک خـط فعـال باشـد ، کدنامشخصـی رادر خروجی می دهد که این عیب کدگذار است لذا کدگذار ها را با رعایت حق تقدم می سازند. بـدین ترتیب که در صورت فعال شدن بیش از یـک ورودی بـه طـور همزمـان ، ورودی کـه حـق تقـدم بیشترداشته باشد فعال می شود که به این کدگذار ها Periority encoder یعنی کدگـذار بـا رعایـت حق تقدم می گویند.

اشکال (۲): عیب دیگر کدگذار این است که اگر ورودی۰۰۰۱ ویا ۰۰۰۰باشد ، خروجـی ۰۰ مـی شود . بنابر این بـایدخطی بـه کـــد گـــذاراضـــافه شـود کـه آن را input Avtive مـی نامنـد. در صورتیکه هر یک از ورودیها فعال باشد این خط فعال می شـود واگـر هیئکـدام از ورودیهـا فعـال نباشد خط input Active نیز فعال نمی شود.بدینوسیله این مشکل نیز حل می شود.

۲-۵ : آرایه منطقی (Logic Array)

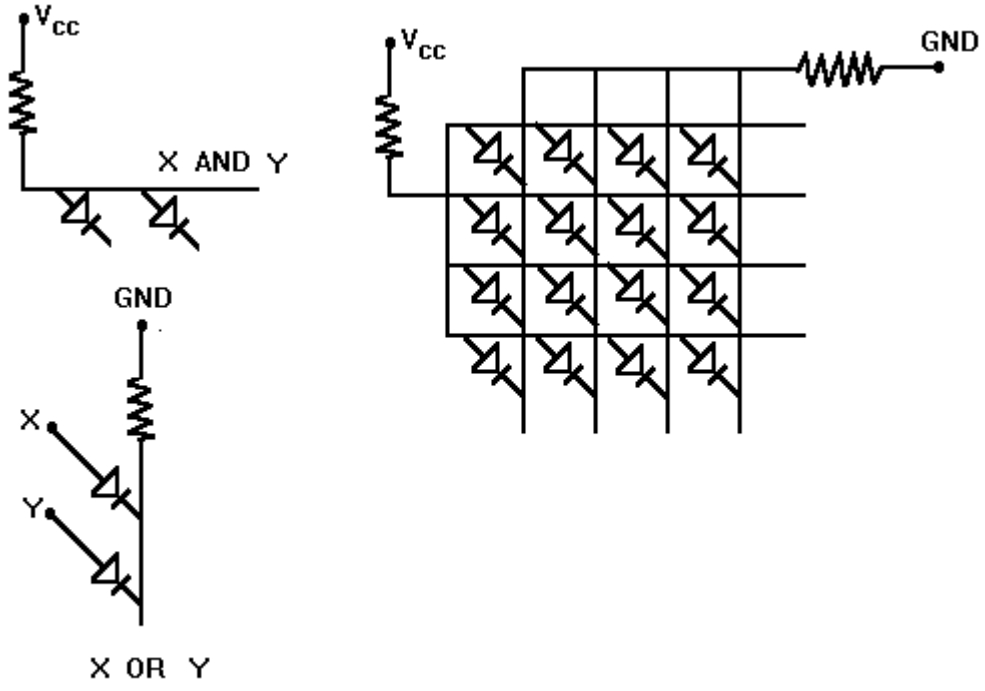
یک مدار ترکیبی است که به صورت آرایه های از دیودها مـی باشـد . کـه در آن آرایـه بـه کمـک دیودها می توان صفرها ویک ها را داشته باشیم . به کمک دیودها ودر نتیجه آرایه منطقی می تـوان گیتهای AND ویا OR را ساخت . از آنجاییکه هر تابع ترکیبی را مـی تـوان بـه کمـک ایـن گیتهـا ساخت،بنابر این از آرایه های منطقی می توان برای ثیاده سازی هر نوع تابعی استفاده نمـود. اغلـب در مواقعی که تابع شلوغ باشد استفاده از آرایه منطقی بسیار مناسب خواهد بود. بیشـتر در طراحـی واحد کنترل کامپیوتر از آرایه های منطقی استفاده می شود . ازانواع آرایه منطقی ، مـی تـوان بـه PLA وROM اشاره نمود.

PLA مخفف Pragramaing Lagic array می باشد،که به آن آرایه منطقی قابل برنامه ریزی می گویند.

۲-۵-۱- آرایه منطقی برنامه ریزی (PLA)

به کمک PLA می توان مجموعه توابع ترکیبی را به صورت یک مدار دو سطحی متناظر با مجموع حاصلضربها تولید نمود. از کاربردهای اصلی PLA ، می توان به طراحی توابع ترکیبی موجود درواحد کنترل اشاره نمود . هر سطر از PLA یک گیت AND وهر ستون یک گیت OR می باشد.

در زیر ساختمان داخلی یک PLA نشان داده شده است:



(شکل ۲-۱۳)

نکته : خطهای موجود بین سطرها نشانگر فیوز می باشد که همان دیوداست.

نکته : در(p × q) PLA ، p تعداد سطر و q تعداد ستون می باشد.

مثال : به کمک PLAتوابع زیر را ثیاده سازی کنید .

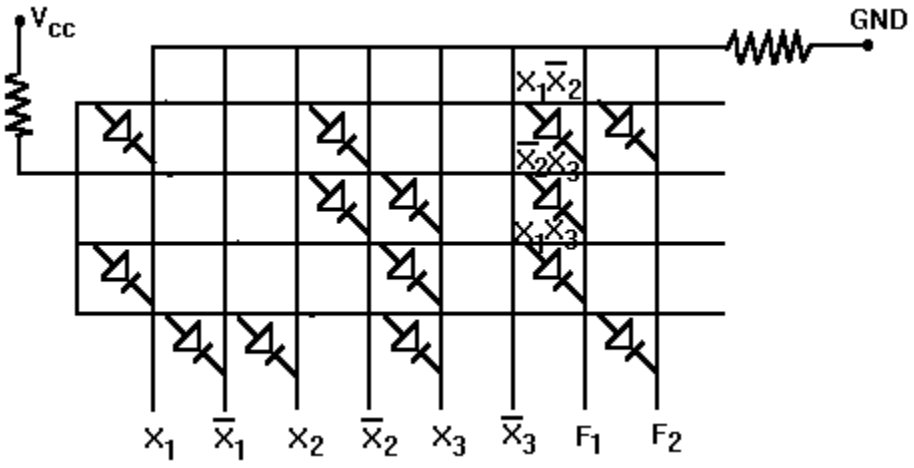
$$F_1 = x_1\overline{x_2} + \overline{x_2}x_3 + x_1x_3$$
$$F_2 = x_1\overline{x_2} + \overline{x_1}x_2x_3$$

حل : ابتدا باید تعداد سطرها وستونهای مورد نیاز برای پیاده سازی را مشخص کنیم .چون PLA راباسطر وستونش می شناسند .برای پیاده سازی توابع باید دید که چه مقدار گیت AND مستقل، مورد نیاز است که تعداد این گیتهای AND مستقل برابر با تعداد سطر مورد نیاز در PLA می باشد در این مثال، ٤گیت AND مستقل وجود دارد . برای بدست آوردن ستونها باید مقدار متغیر را در دو ضرب کرده وآنرا با تعداد توابع جمع کنیم.

q = تعداد ستونها = تعداد توابع + متغیرها + آنها not

یا $q = m + ۲ \times n$

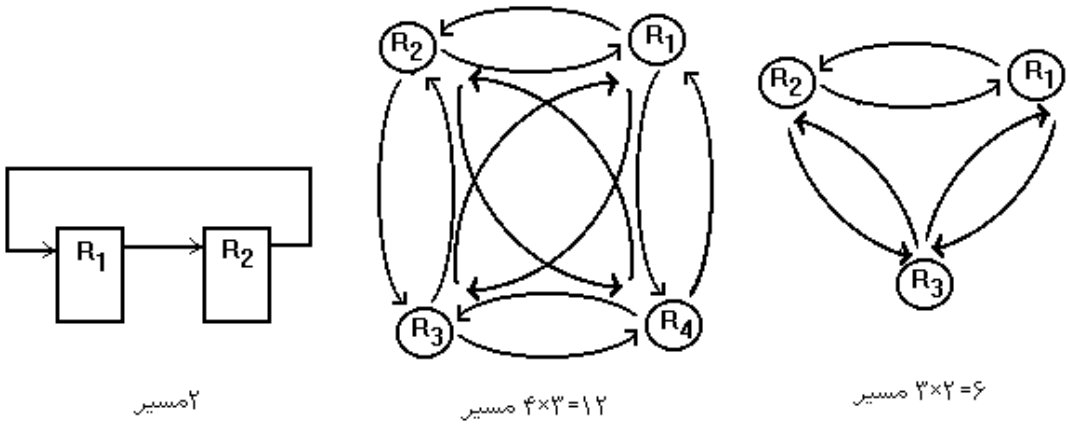
متغیرها تعداد توابع



(شکل ۲-١٤)

۲-٦ : گذرگاه (BUS)

مجموعه‌ای از سیمهای رابط برای انتقال یک کلمه N بیتی از یک مبداء به یک مقصد که معمولاًمبداء ها ومقصدها ثباتهای کامپیوتر می باشند . در واقع مسیری که برای انتقال اطلاعات وجود دارد را BUS می گوییم .



(شکل ۲-۱۵)

باس (BUS) بردو نوع است :

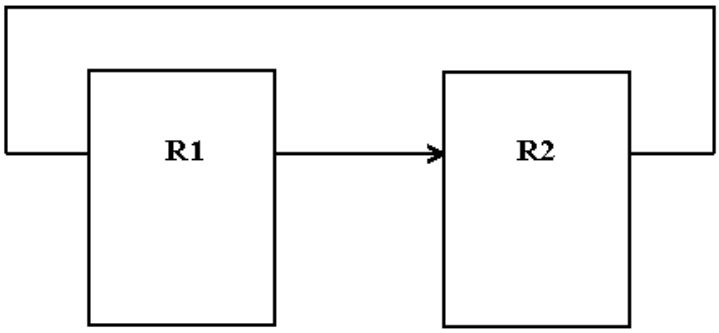
۱- باس اختصاصی

۲- باس مشترک

۲-٦-۱- BUS اختصاصی (dedicated Bus)

بدین معنی که بین هر دو ثبات دلخواه BUS اختصاصی طراحی می گردد. البته می دانیم در این طراحی تعداد BUS ها ودر نتیجه تعداد سیمها بالا بوده وامکان Noise بیشتر می گردد. از این جهت در سیستمهای کامپیوتری این شیوه مطلوب نیست.

مثالی از BUS اختصاصی



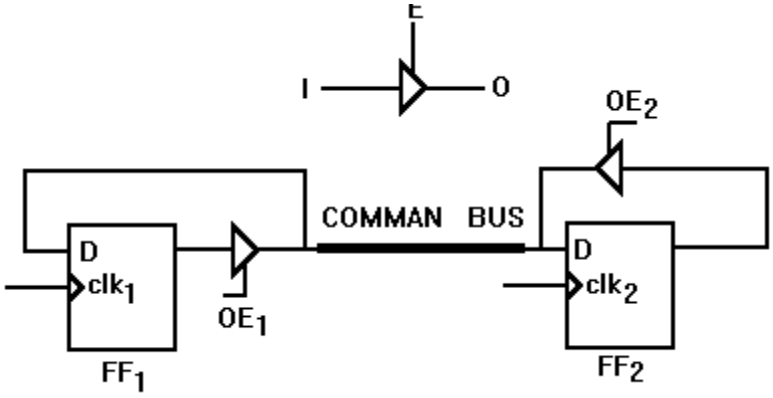
(شکل ۲-۱۶)

نکته : به طور کلی برای n ثبات ، n(n-1) مسیر مورد نیاز است .

۲-۶-۲- BUS مشترک (Common BUS)

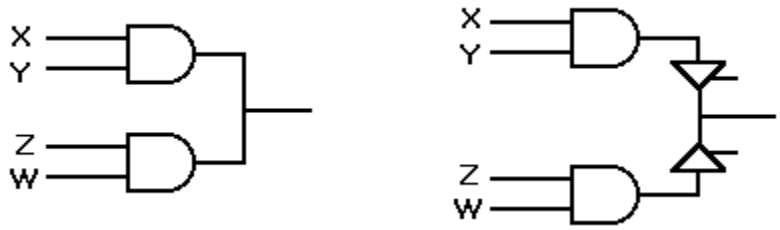
در این شیوه طراحی هر تعداد ثبات که در یک سیستم موجود باشد، تنها از یک BUS استفاده می کند. یک کامپیوتر رقمی نوعاً ثباتهای زیادی دارد وباید مسیرهایی برای انتقال اطلاعات از یک ثبات به ثبات دیگر در آن فراهم شود . اگر خطوط جداگانه‌ای بین هر ثبات وثباتهای دیگر سیستم بکار رود، تعداد سیمها بیش از حد خواهد شد. روش کاراتر برای انتقال اطلاعات بین ثباتهادر پیکر بندی با ثباتهای متعدد، همین روش سیستم گذرگاه مشترک است.

مثالی از BUS مشترک



(شکل ۲-۱۷)

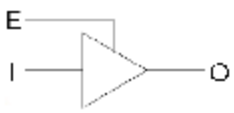
نکته : نمی توان خروجی دو گیت یا فیلیپ فلاپ رابه هم وصل کرد چون ممکن است یکی High (بالا 5V) و دیگری low (پایین 0V) باشد در نتیجه IC ما می سوزند.



(شکل ۲-۱۸)

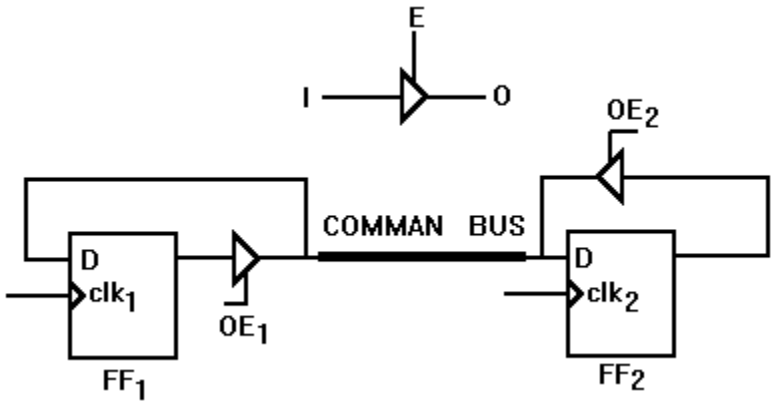
۲-٦-۳- بافر یا درایو یا بافر درایو

بافر درایو را به صورت زیر نمایش می دهند :



این‌درایو دارای یک سوییچ (E) است که اگر صفـــر باشد، مدار قطع واگر یک باشد مدارمتصل است و ولتاژ ورودی و خروجی با هم برابرند و جریان را تقویت می کند که به آن 3-State Buffer یا بافر سه حالته می گویند.

مثال : نحوه انتقال داده بین ثبات A و ثبات B



(شکل ۲-۱۹)

FF1------> FF2

۱- باید ثبات A فعال باشد تا اطلاعات از آن به ثبات B برود.

1) OE1=1 , OE2=0

۲- ضمن اینکه ثبات A فعال است به ثبات B دستور ذخیره کردن را می دهیم .

2) OE1=1 , OE2=0 , Clk2=1

حال اگر بخواهیم اطلاعات موجود در ثبات B را به A منتقل کنیم بدین صورت عمل می نماییم.

FF2 -------> FF1

۱- باید ثبات B فعال باشد تا اطلاعات از آن به ثبات A برود.

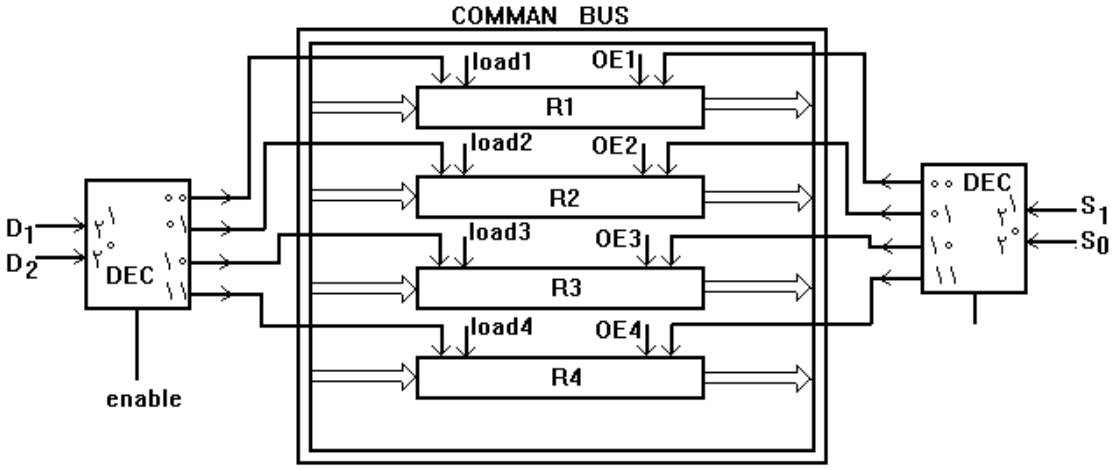
1) OE2=1 , OE1=0

۲- ضمن اینکه ثبات B فعال است به ثبات A دستور ذخیره کردن را می دهیم.

2) OE2=1 , OE1=0 , Clk1=1

۲-٦-٤- طراحی دو نوع BUS مشترک

۲-٦-٤-۱- طراحی یک BUS مشترک بین چهار ثبات که خروجی آنها از نوع 3-State می باشد.



(شکل ۲-۲۰)

مثال: اگر بخواهیم اطلاعات R_1 رابه R_3 منتقل کنیم.

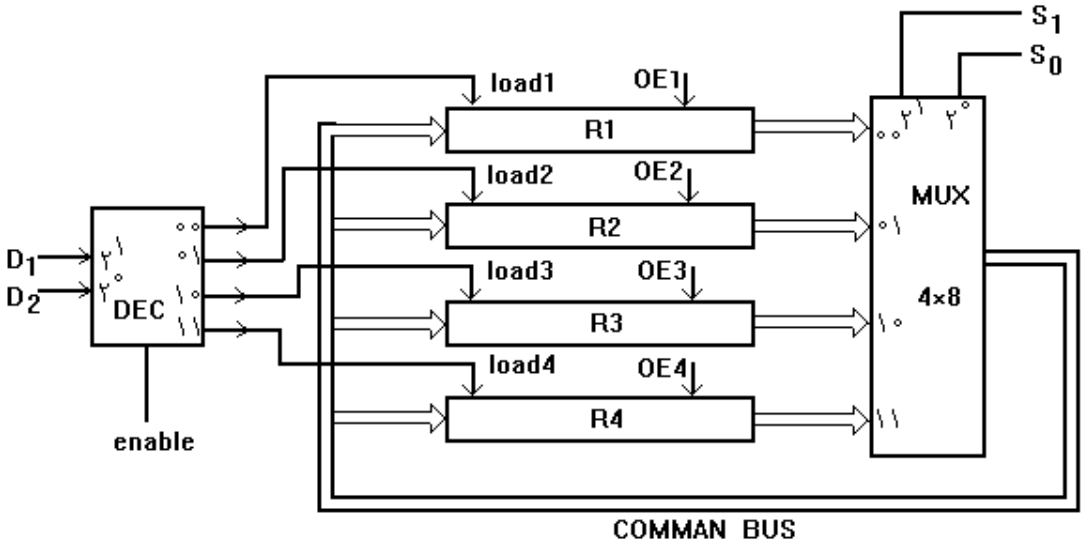
R_1 -------> R_3

1) $S_1S_0 = 00$, enable = 0 , $D_1D_0 = 10$

2) $S_1S_0 = 00$, enable = 1 , $D_1D_0 = 10$

برای اینکه بخواهیم هربار یک ثبات انتخاب شود از کدگشا استفاده کرده ایم که در حقیقت کدگشاها در اینجا انتخابگر مبدأ و مقصد می باشند.

۲-٦-٤-۲- طراحی یک باس مشترک بین چهار ثبات که خروجی آنها از نوع 3-State نیست.



(شکل ۲-۲۱)

مثال: اگر بخواهیم محتوای R_2 را در R_4 بریزیم داریم .

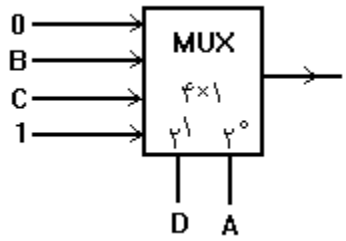
$$R_2 \text{ -------> } R_4$$

1) $S_1S_0 = 01$, $D_1D_0 = 11$, enable = 0

2) $S_1S_0 = 01$, $D_1D_0 = 11$, enable = 1

تمرینات

۱- عبارت بولی (4 × 1) Mux زیر را بنویسید؟



۲- با استفاده از هر تعداد (1 × 2) Mux یک (2 × 4) Mux طراحی کنید؟

۳- تابع $f(A,B,C) = \Sigma(0,2,6,7)$ رادرسه‌حالت‌مختلف‌با(1 × 4) Muxطراحی‌کنید؟

٤- با استفاده از دومالتی پلکسر در توابع موجود یک مدار تفریق کننده کامل (Full-Subtractor) را پیاده سازی نمایید؟

۵-با استفاده از کدگشا (Decoder) و گیتهای OR لازم یک ضرب کننده دو بیتی طراحی کنید.

٦- به کمک کدگشا (Decoder) و گیتهای OR لازم یک مدار Full adder- Full Subtractor طراحی نمایید.

۷- مداری طراحی کنید که یک عد د دو بیتی را در یافت وآن را به تابعی که درزیر آمده است اعمال نموده ونتیجه رابر روی 7Seg نمایش دهد . (به کمک کدگشا)

$$x \leq 2$$
$$x > 2$$
$$f(x) = \begin{cases} x^2 + 1 \\ F \end{cases}$$

۸- به کمک کدگشا (Decoder) و گیتهای مورد نیاز یک مقایسه گر دو بیتی طراحی نمایید.

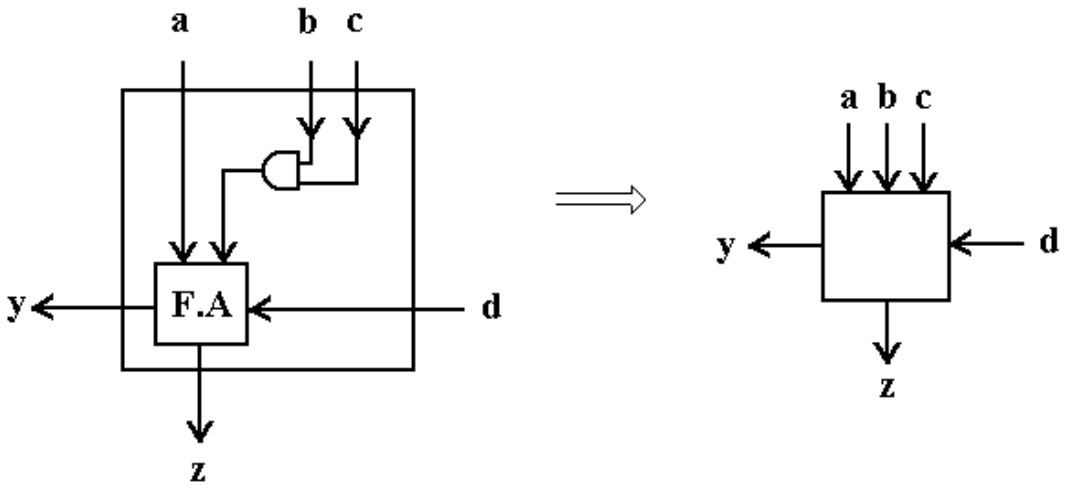
۹- یک کدگذار (Encoder) با رعایت حق تقدم 4 بیتی با استفاده از گیتهای AND و OR طراحی نمایید.

۱۰- مداری طراحی کنید که بتواند 4 ثبات 8 بیتی را به نحو خواسته شده با هم جمع نماید و حاصل را در ثبات دیگر مورد نظر قرار دهد در این طراحی ثباتها را از نوع 3-State بگیرید و جمع کننده را به صورت یک واحد 4 بیتی.

۱۱- یک مدار ترتیبی طراحی کنید که عدد باینری بدون علامت و با طول n را در ۳ ضرب نماید. عدد n بصورت سریال از بیت کم ارزشتر به مدار وارد و حاصل یعنی ۳×n به صورت سریال در طرف دیگر خارج می گردد.

۱۲- یکی از بخشهای CPU مدار شیفت دهنده به نام Buffer Shift می باشد که توسط آن می تواند شیفتهای سریع به تعداد بیتهای خواسته شده به سمت چپ یا راست اعمال نمود. یک Buuffer Shift چهار بیتی طراحی کنید که توسط آن بتوان تا ٤ بیت انتقال به چپ یا راست انجام داد .(ثبات را در این مسأله چهاربیتی بگیرید.)

۱۳- چگونه می توان به کمک سلول زیر یک ضرب کننده ۳ بیتی طراحی کرد .



۳- طراحی پردازنده

۳ -۱ : وظیفه اصلی CPU

پردازنده قسمتی از یک کامپیوتر می باشد که وظیفه اصلی آن اجرای دنباله‌ای از دستورالعملهای موجود در حافظه اصلی می باشد. برای این کار هر دستورالعمل قبل از اجراشدن بایستی از حافظه اصلی خوانده شود (Fetch) و سپس دستورالعمل رمزگشایی شود (Decoding) آنگاه به داخل پردازنده منتقل شده وسپس دستورالعمل اجرامی شود(Execute).

مجموعه زیر دستوراتی که ، برای اجرای کامل یک دستور لازم است چرخه دستورالعمل یا Inst.Cycle گفته می شود.

چرخه دستورالعمل به دو گروه تقسیم می شود:

۱- Fetchh.Cycle :واکشی دستور از حافظه .

۲- **Execute Cycle**: decode کردن یا رمز گشایی دستور و اجرای زیر دستورات لازم .

مراحل اجرای یک دستورالعمل:

واکشی دستورالعمل: برداشت دستورالعمل بعدی از حافظه

کدبرداری از دستورالعمل: بررسی دستورالعمل برای مشخص شدن اینکه:

چه عملی باید توسط دستورالعمل انجام گیرد (به عنوان مثال جمع)

چه عملوندهایی مورد نیازند، و نتایج باید کجا قرار گیرند.

واکشی عملوندها: عملوندها برداشت می شوند.

اجرا: اجرای عملیات بر روی عملوندها

بازنویسی نتیجه: نوشتن نتیجه در محل مخصوص

دستورالعمل بعدی: تعیین اینکه دستورالعمل بعدی از کجا گرفته شود.

چه چیزی در یک ISA (معماری مجموعه دستور العمل) مشخص می شود؟

ISA: Instruction Set Architecture

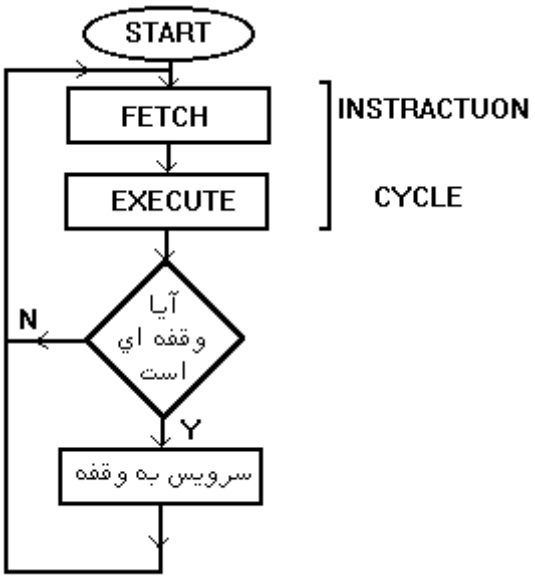
کدبرداری از دستورالعمل: اعمال و عملوندها چگونه تعیین می گردند؟

واکشی عملوندها: عملوندها ممکن است کجا باشند؟ چه تعداد؟

اجرا: چه اعمالی می تواند انجام گیرد؟ چه نوع داده و چه اندازه هایی؟

بازنویسی نتایج: نتایج کجا نوشته می شوند؟ چه تعداد؟

دستورالعمل بعدی: دستورالعمل بعدی را چگونه می توان انتخاب نمود؟



(شکل ۳-۱)

ریزعمل:

انجام کارهای جزیی برای رسیدن به هدف اصلی را ریز عمل گویند.

Cpu Clock time :

مدت زمان لازم برای اجرای یک ریزدستورالعمل (Micro Inst) را Cpu Clock time (t_{cpu}) می نامند.

Cpu Clock Rate :

تعداد ریزعملهایی که در واحد زمان انجام می شود را به عنوان Cpu Clock Rate گویند. که معمولاًبا واحد هرتز (مگا هرتز) بیان می شود وآن را بصورت $\frac{1}{t_{cpu}}$ نمایش می دهند ، که ملاک اندازه گیری سرعت پردازنده هاست .

برای مثال اگر سرعت Cpu کامپیوتری 100 MHZ باشد یعنی 100×2^{20} ریز عمل رادر یک ثانیه انجام می دهد.

۲-۳ : وظیفه فرعی Cpu

Cpu علاوه بر اجرای دستورالعمل ، بر واحدهای دیگر کامپیوتر نیز نظارت دارد مانند نظارت بر نقل و انتقال اطلاعات از حافظه به دستگاههای ورودی و خروجی (I/o device) و یا نظارت بر کارهای I/o . اما اینگونه کارها به ندرت انجام می گیرد. از این جهت تا لحظه دریافت تقاضا از دستگاههای جانبی CPU توجهی به آنها نمی کند. در صورتی که دستگاه I/o تقاضای سرویس داشته باشد تقاضا را با فرستادن سیگنال Intrupt یا وقفه به Cpu اعلام می کند.

سوٴال : چرا پردازنده در پایان اجرای دستورالعمل جاری به وقفه توجه می کند؟

برای جواب دادن به این سوٴال به مثال زیر توجه کنید.

مثال : اگر کامپیوتر در حال اجرای برنامه‌ای که حاوی دستورالعملهای زیر، به زبان ماشین باشد .

```
ADD    Ax, Num
I NC   Bx
```

اگر دستورالعمل اول را دریافت کرده و سپس وقفه فرارسد ابتدا دستورالعمل اول را اجرا کرده و بعد وقفه را اجرا می نماید و بعد از پاسخ به وقفه به سراغ اجرای دستورالعمل دوم می رود . زیرا اگر در حین اجرای دستورالعمل اول به وقفه توجه کند، نتیجه و یا محتوای ثباتی‌که برروی آن کارانجام گرفته است دست نخورده باقی می ماند یعنی مانند این است که دستورالعمل اول وجود نداشته است و چون PC (Program Counter) ما یک واحد اضافه شده است بنابراین امکان دسترسی مجدد به دستورالعمل اول را نخواهیم داشت بنابراین برنامه ما به اشتباه اجرا می شود.

همیشه در طراحی CPU سعی بر آن است که حداقل هزینه و حداکثر سرعت را داشته باشیم. به این خاطر در طراحی مدارات کامپیوتر اگر چه همه کارهای آن بکمک عملیاتی چون جمع ، ضرب و تفریق و تقسیم انجام می شود ، اما از لحاظ سخت افزاری تنها یک مدار جمع کننده وجود دارد که همه این کارها را انجام می دهد.

اگر قرار باشد یک پردازنده با سرعت بالا عمل کند ، بایستی ادواتی که با آن درگیری مستقیم دارند نیز، هم سرعت با آن باشند. (زیرا در غیر اینصورت سرعت نهایی پایین می آید) از جمله ادواتی که با Cpu رابطه مستقیم دارد، حافظه اصلی کامپیوتر است.

امروزه می دانیم برای اجرای برنامه‌ها ظرفیت زیادی از حافظه مورد نیاز است. اگر قرار باشد که این حافظه با حجم زیاد خود هم سرعت پردازنده باشد، به عبارتی از تکنولوژی ساخت پردازنده استفاده کند، قیمت حافظه و در نتیجه قیمت کامپیوتر به مراتب بالا می رود. از این جهت برای کاهش قیمت معمولاً حافظه ها را با سرعتی حتی تا یک دهم سرعت پردازنده در اختیار آن قرار می دهند. (البته در PC ها اینطور نیست بلکه ضریب سرعت بین حافظه و پردازنده بین ٤ تا ٦می باشد) بدین ترتیب قیمت حافظه و همچنین قیمت کامپیوتر در کل مطلوب خواهد بود. اما مشکل اختلاف سرعت بین حافظه با ظرفیت بالا و پردازنده را به کمک حافظه دیگری که از نظر سرعت و تکنولوژی همسان با پردازنده است (که این حافظه را اصطلاحاً Cache می نامند) برطرف می سازند.

سرعت حافظه

مدت زمان لازم برای انجام دو read یا دو Write متوالی را سرعت حافظه گویند. که معمولاً با سرعت Cpu به صورت $\frac{t_{mem}}{t_{cpu}}$ مقایسه می شود.(مقدار آن بین 1تا 10 می باشد)اگر سرعت حافظه برابر با Cpu باشد ، $\frac{t_{mem}}{t_{cpu}}$ برابر یک می شود. در این صورت فضای حافظه اصلی به صورت یک ثبات(Register) تلقی خواهد شد (کارایی یک کامپیوتر نیز با این نسبت سنجیده می شود.)

۳-۳ : ارتباط CPU با I/O وحافظه اصلی

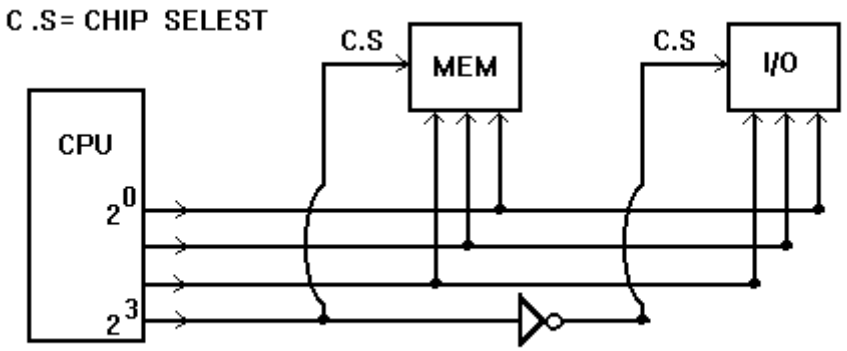
می دانیم که پردازنده علاوه بر حافظه اصلی ، دستگاههای مختلف I/O رادارا می باشد که بایستی بتواند آنها را آدرس دهی نماید . نحوی آدرس دهی حافظه اصلی ودستگاه I/O در سیستم های کامپیوتری به طور عمده به دو دسته تقسیم می شود :

دسته اول : Memory Mapped I/O

دسته دوم : I/O Mapped I/O

۱-۳-۳ : Memory Mapped I/O

در این روش از کل فضای آدرس دهی حافظه اصلی ، بخشی از آن به I/O های مختلف تعلق دارد وبقیه فضای باقیمانده به حافظه اصلی اختصاص می یابد. بنابراین فضای آدرس دهی I/O باید در فضای آدرس دهی حافظه ، Mapped شود.



(شکل ۳-۲)

در اینجا فرض شده که CPU هیچ وسیله جانبی ندارد وفقط چهار خط آدرس دارد.

نکته : تا زمانی که با ارزش ترین بیت دارای ارزش صفر است ، I/O آدرس دهی می شود و وقتی دارای ارزش یک باشد حافظه اصلی آدرس دهی می شود .

با این آدرسها حافظه اصلی فعال می شود. با این آدرسها I/O فعال می شود.

۱۰۰۰	۰۰۰۰
۱۰۰۱	۰۰۰۱
|	|
|	|
|	|
۱۱۱۱	۰۱۱۱

زمانی از این روش استفاده می شود که سرعت I/O وسرعت حافظه با هم برابر باشد . وقتی سرعتها برابر نیست باید وقفه نرم افزاری ایجاد نماییم.

مثال : LDA 2

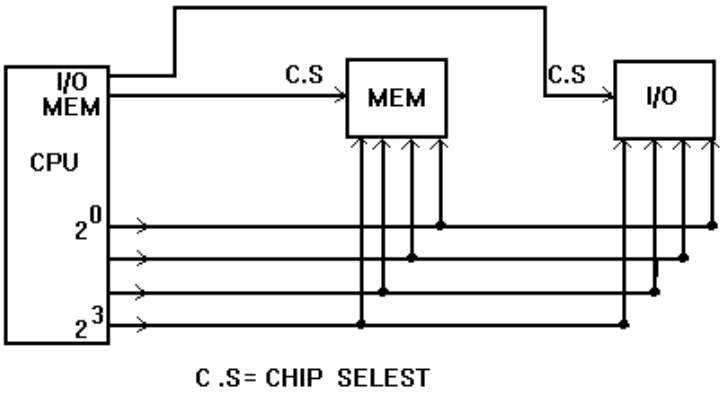
توقف زمان Delay

ADA 5

در مثال فوق در دستور LDA 2 از یکی از وسایل جانبی I/O باآدرس 2 ، مقداری را Load می کند . یعنی بدین صورت از ابزارهای جانبی استفاده می کنیم وچون سرعت I/O با سرعت حافظه اصلی برابر نیست بصورت نرم افزاری یک توقف ایجاد می نما ییم . مثلاً بادستور Delay این عمل را انجام می دهیم ، تاکمی توقف زمانی ایجاد شود تا عمل I/O قبلی پایان یابد وسپس عمل بعدی که استفاده از وسایل جانبی می باشد ، صورت گیرد .

I/O Mapped I/O : ۲-۳-۳

هنگامی که حافظه وI/O هم سرعت نباشد دستورالعملهای مختلفی برای I/O وحافظه وجود دارد ونیز فضای آدرس دهی حافظه ، از فضای آدرس دهی I/O مجزا است وحتی دو دسته دستورالعمل مجزا برای کارکردن با حافظه و I/O وجود دارد.



(شکل ۳-۳)

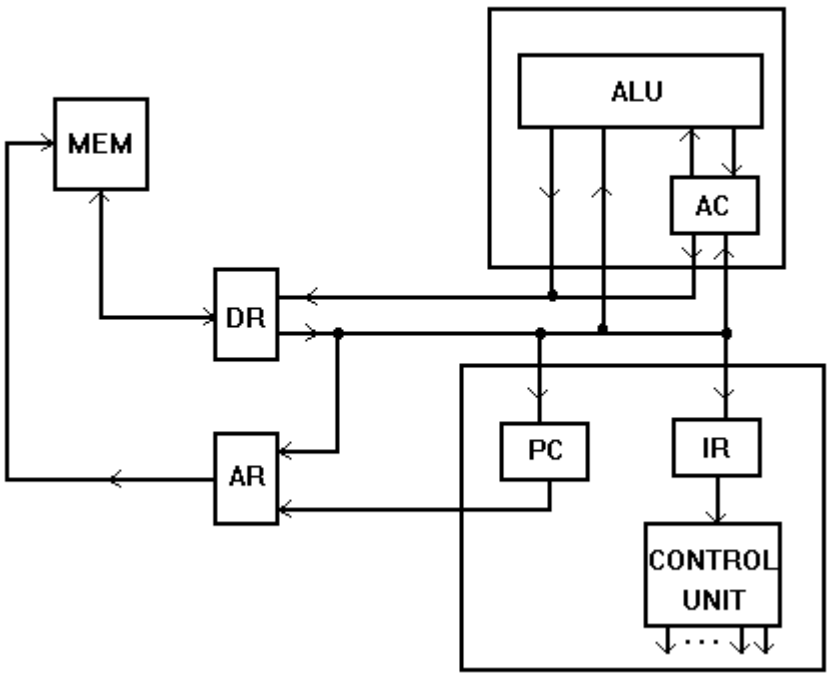
برای مثال دستورات I/O با in وout همراه هستند ولی دستورات حافظه با MOV همراه هستند

.

نکته : معمولاً بطورواقعی در سیستمهای کامپیوتر مشاهده می شود که خطوط آدرس دهی به حافظه بطور کامل می باشد ولی همی این خطوط به I/O برای آدرس دهی متصل نمی باشد زیرا تعداد I/O به اندازی تعداد آدرسهای حافظه نیست.

۳-٤ : ماشین وان نیومن (Von Neuman)

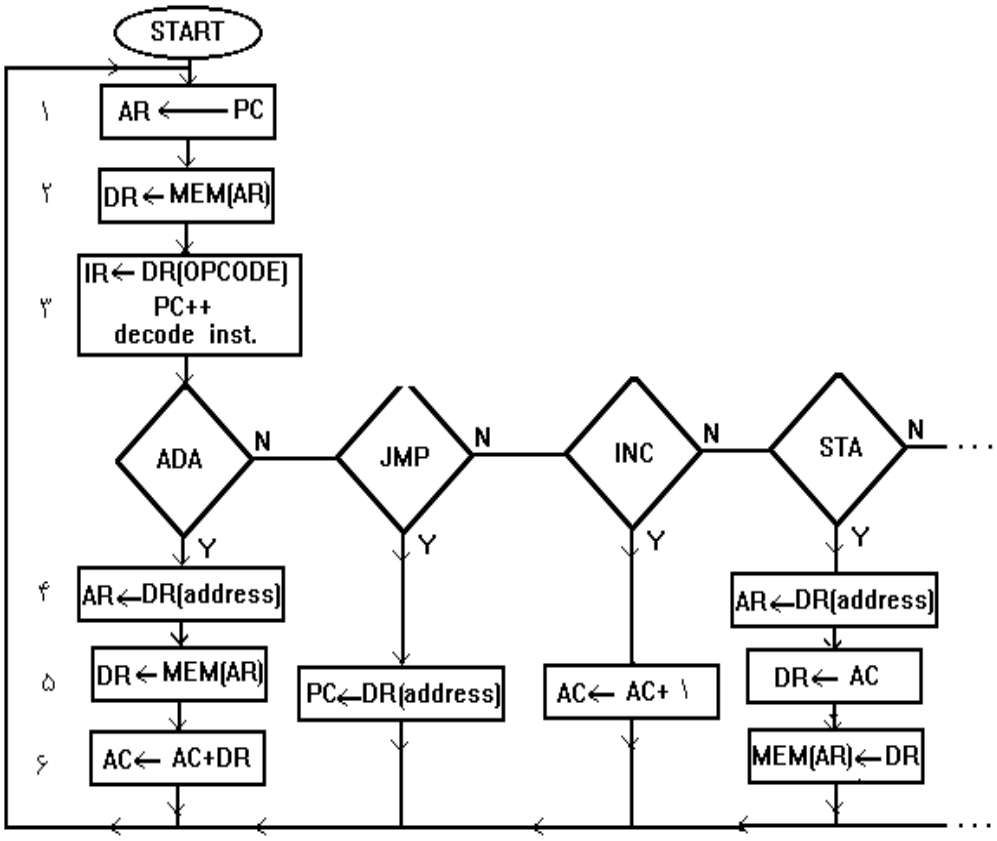
وان نیومن به همراه همکارانش یک کامپیوتر به نام IAS ساخت که پایه واساس کامپیوترهای امروزی می باشد وکامپیوتر ایشان یک کامپیوتر تک آدرسه بود .



(شکل ۳-٤)

۳-٤-۱ : دستورالعملهای ماشین وان نیومن :

وقتی بخواهیم یک پردازنده طراحی کنیم باید اول بدانیم چه سخت افزارهایی را نیاز داریم واین انتخاب سخت افزارها بسته به دستوراتی است که ریز پردازنده نیاز دارد.



(شکل ۳-۵)

با توجه به دستورات بالا مثلا اگر بخواهیم محتوای AC را با سطر ۱۰۰ حافظه جمع کنیم (ADA 100) ، ابتدا پردازشگر به سطری از حافظه که این دستور درون آن نوشته شده است رفته و این دستور را درون DR قرار می دهد و Decode شده و قسمت های آن مجزا می شوند و بخش آدرس آن که ۱۰۰ است ، درون AR قرار می گیرد . سپس به خانه ۱۰۰ حافظه مراجعه شده و اطلاعات آن درون DR قرار می گیرد . حال به راحتی می توان محتوای ثبات های AC و DR را باهم جمع نمود .

مراحل دستور های بالا باید گفت که این دستورات را باید طوری ادغام کنیم که همزمان اجرا شدن آن تاثیری روی هم نداشته باشد و باعث بالاتر رفتن سرعت برنامه می شود . البته دستور PC++ در مرحله دوم هم می توانست قرار داشته باشد . زیرا خللی در اجرای برنامه نخواهد داشت . انجام

این مراحل به ترتیب است یعنی اگر مرحله اول رخ ندهد ، مرحله دوم هم نمیتواند اتفاق بیفتد که این مسئله مربوط به پردازش موازی است که بعدا بحث خواهد شد.

* آیا می توانید بوسیله ریز پردازنده‌ای که دراختیار دارید یک دستورالعمل شرطی بکار برید ؟ خیر ، چون در این ماشین ما دارای فلاگهای خاصی نیستیم ، که بر اساس آن دستورات شرطی را اعمال کنیم .

* آیا این ماشین می تواند دستور LDA M 100 را اجرا کند ؟ بله

با قراردادن 100 بطور مستقیم در AC این دستور امکان پذیر است . AC<------ AR (DR)

* آیا این ماشین می تواند دستور JnG 100 را انجام دهد ؟

خیر ، زیرا فلاگ n را برای تشخیص منفی بودن حاصل عملیات دارا نمی باشد.

* آیا این ماشین می تواند دستور LDA I 100 راانجام دهد ؟ I یک اشاره گر است بنابراین۲ بار به حافظه باید مراجعه کنیم ، یک بار به آن مراجعه کنیم که آدرس را بدست آوریم و باردیگر از آدرسی در داخل حافظه می خوانیم تا Data را بدست آوریم .

DR ← MEM (AR)

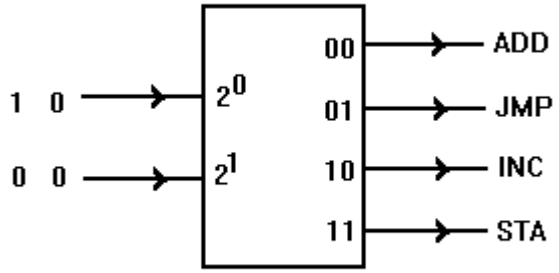
AR ← DR (Address)

DR ← MEM (AR)

* آیا این ماشین می تواندستور LDA X 100 را انجام دهد؟

خیر ، باید محتوای ثبات Indexing Reg با ACجمع شود که چنین دستوری را در ماشین نداریم. پس باید یک ثبات Index Reg به آن اضافه کنیم تا آن دستور قابل اجرا باشد.

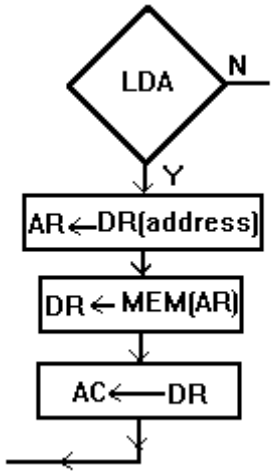
نکته : به علت استفاده از کدگشا در قسمتهای مقایسه‌ای ، این قسمتها شماره گذاری نشده است. یعنی واحد زمانی برای آنها در نظر گرفته نمی شود . همانطور که می دانید در مدار های ترکیبی اتلاف وقت نداریم . بدین صورت که کد گشا یکی ازدستورات را تشخیص داده وفوراً بدون اتلاف وقت ریز دستورات مربوط به آن دستورالعمل را اجرامی نماید . این تشخیص دستور، توسط کدگشا از روی Opcode آن دستورالعمل می باشد .



(شکل ۳-٦)

مثال : دستورالعمل LDA 5 چگونه اجرا می شود ؟

LDA 5 که محتوای خانه ۵ از حافظه را روی AC بار می کند.در ابتدا PC اشاره می کند به سطری که دستور LDA 5 در آن قرار دارد و آن را در DR قرار می دهد . دراین مرحله بایستی آدرس DR را که همان ۵ است را به AR انتقال دهد وسپس محتوای این خانه جدید که آدرسش در همان AR می باشد را به DR منتقل می نماید وسپس DR را به AC می ریزد. چون بین حافظه وALU هیچ ارتباطی وجود ندارد مرحله آخر الزامی است.



(شکل ۳-۷)

نکته : وقتی دستگاه راه اندازی می شود، PC طوری طراحی شده که به جای مشخصی اشاره کند وباید سیستم عامل را در آن آدرس مشخص قرار دهیم تا دستورات از آن آدرس شروع ، وبه اجرا درآیند تا سیر منطقی اجرای دستورات صورت گیرد. یعنی به ابتدای برنامه سیستم عامل که روی ROM قرار دارد رفته و آن را روی RAM می آورد و سپس به ابتدای برنامه ای که خودش Load کرده است ، رفته و سیستم عامل را اجرا می کند .

مثال : مدت زمانی که طول می کشد تا دستورالعمل جمع اجرا شود چند ثانیه می باشد؟

(فرض کنید سرعت CPU برابر با ۶۰ MHZ باشد)

مدت زمانی که طول می کشد تا دستورالعمل جمع اجرا شود .

$6 \times \frac{1}{60 \times 2^{20}} = \frac{1}{2^{21}}$ تعداد مراحل یا

تعداد ریز عملها

فرکانس (قطر پالس‌ها)

تمرین : ده دستورالعمل از کامپیوتری که در اختیار دارید را در نظر بگیرید .

الف) ابتدا حدس بزنید که این دستورالعمل ها چه تعداد ریز دستور نیاز دارند .

ب) با استفاده از کتاب ها و منابعی که در اختیار دارید این تعداد را بدست آورید .

ج) با استفاده از برنامه محاسبه کنید . (استفاده از حلقه)

د) این ده دستورالعمل را در یک برنامه به کار ببرید .

نکته : برنامه ی خود را با استفاده ار زبان اسمبلی بنویسید .

۳-٤-۲: روشهای گسترش ماشین وان نیومن

می توان به راحتی قابلیتهای بیشتری به ماشین پیشنهادی Von - neuman داد.

۱) می دانیم که اگر تعداد ثباتهای یک پردازنده بیشتر باشد، می توان برنامه را با سرعت بیشتری اجرا نمود . به راحتی می توان ثباتهای بیشتری به این ماشین اضافه نماییم .

۲) این ماشین فقط دستورالعملهای جمع وتفریق را انجام می دهد و می توان به راحتی با افزودن مدارات جزیی به آن قابلیت ضرب و تقسیم را نیز به آن بخشید .

۳) ماشین Von neuman نمی تواند دستورات پرشی از نوع شرطی را اجرا کند .اما به راحتی می توان با اضافه کردن Flag Register به آن ، این قابلیت را به ماشین Von neuman داد.

٤) اگر در پردازنده ای سرعت حافظه اصلی نسبت به CPU کم باشد ، می توان با دراختیار گذاشتن حافظه نوع Cache ، در عمل خواندن ونوشتن از حافظه سرعت را بهبود بخشید .

۳-۵ : پردازش موازی (Parallel Processing) :

پردازش موازی اصطلاحی است برای مشخص کردن دسته وسیعی از روشهایی که برای پردازش همزمان اطلاعات با هدف افزایش سرعت محاسباتی یک سیستم کامپیوتری به کار می رود .

یک سیستم پردازش موازی می تواند به جای پردازش ترتیبی دستورالعمل ها به سبک کامپیوترهای نسبی ، اطلاعات را به طور همزمان پردازش کند وبه این ترتیب به سرعت بالاتری دست یابد . مثلاً در حالی که یک دستورالعمل در Alu در حال اجرا است دستور بعدی می تواند از حافظه خوانده شود .

سیستم می تواند دو Alu و یا بیشتر داشته باشد ودر یک زمان دو دستورالعمل بابیشتر را اجرا کند . علاوه براین می تواند دارای دو یا چند پردازنده باشد که همزمان کار کنند.

هدف از پردازش موازی تسریع قابلیت پردازش کامپیوتر وافزایش توان عملیاتی آن ، یعنی میزان پردازش قابلیت انجام در طول یک فاصلی زمانی معین است . در پردازش موازی حجم سخت افزار افزایش می یابد وبه همراه آن هزینی سیستم نیز بالا می رود . اما پیشرفت‌های تکنولوژی هزینه های سخت افزاری را قدری کاهش داده است که به همین علت تکنیکهای پردازش موازی از نظر اقتصادی به صرفه شده است .

نمونه‌ای از پردازش موازی در زیر آمده است

مثال ۱ : Parallel adder در مقابل Serial adder

Parallel adder نوعی جمع کننده می باشد که مجموعه‌ای از بیت ها را با ثیش بینی رقم نقدی جمع می کند وSerial adder در هر لحظه یک بیت را با بیت متناظر جمع می بندد .

مثال ۲ : مراحل Execute , Fetch یک دستورالعمل

اعمال Execute , Fetch در دو مرحله مستقل صورت می گیرند . بنابراین هنگامی که دستورالعملی در حال اجرا است می توانیم دستور جدید را Fetch کنیم . در آن موقع هردو کار صورت می گیرد، یکی واکشی دستورالعمل جدید ودیگری اجرای دستورالعمل قبلی .

در مثال اول از پردازش موازی ، عناصر پردازش‌را افزایش می دهیم . (یعنی تعداد F.A ها را افزایش می دهیم) در مثال دوم انجام یک عمل به چند قسمت مستقل تقسیم می شود .

بطور کلی پردازش موازی به دو طریق امکان پذیر است

۱- Multiunit Parallel Process

۲- Pipelining Parallel Process

مثال اول از نوع Multiunit می باشد ، چون به سخت افزار وهزینه بیشتری نیازمند است ومثال دوم براساس Pipeline می باشد که یک واحد به چند تکه تقسیم می شود بطورکلی برای پیاده سازی پردازش موازی به سخت افزار بیشتری نیاز است که این نیازمندی در روش Multiunit مشهودتر است.

کامپیوترها را از لحاظ نحوی پردازش به چهار گروه تقسیم می کنند.

۱- SISD (Single Instruction Stream Single Data Stream)

این کامپیوترها دارای یک CPU و یک ALU می باشند ودر آن از پردازش موازی استفاده نمی شود . یعنی فقط یک دستورالعمل در هر لحظه انجام می شود . نظیر پردازنده های 8085 , 80280

۲- SIMD (Single Instruction Stream Multiple Data Stream)

پردازش موازی از نوع Multiunit دارای یک CPU وچند ALU است ونیز دارای دستوراتی برای پردازش بر روی بردارها می باشد .

۳- MISD (Multiple Instruction Stream Single Data Stream)

این کامپیوترها بصورت Pipeline عمل می کنند . نمونه‌ای از کامپیوتر که براین اساس است کامپیوتر ۱- Carry می باشد که یک سوپرکامپیوتر است .

٤- MIMD (Multiple Instruction Stream Multiple Data Stream)

این کامپیوترها دارای چندCPU وچند ALU هستند که پردازش موازی را چه از لحاظ Pipeline وچه از لحاظ Processor انجام می دهد . نمونه‌ای از کامپیوتر که براین اساس عمل می کند IBM 308 می باشد.

کامپیوترهای شخصی (PC) تا حدی براساس شیوه سوم عمل می کنند (یعنی MISD) در کامپیوترهای شخصی 8080 به بالا ، این به دوقسمت تقسیم می شود . قسمت اول وظیفه دریافت دستورالعمل (Fetch) را برعهده دارد وقسمت دوم وظیفه اجرای دستورالعمل را بعهده دارد .

قسمت اول دارای یک صف است که دستورالعمل از حافظی Fetch شود وارد پشته (Stack) می شود سپس این دستورالعمل به قسمت دوم می رود واجرا می گردد . درزمانی که قسمت اول بیکار است ، دستورالعمل دوم و سوم را می گیرد تا کار اجرای دستورالعمل اول تمام شود.پس ازپایان اجرای دستورالعمل‌اول، دستورالعملهای دوم و سوم به ترتیب‌اجرامی‌گردد.

در کامپیوتر ۸۰۸۸ این صف ظرفیت شش دستورالعمل را دارد و در کامپیوتر ۸۰۸٦ این صف ظرفیت نه دستورالعمل را دارد.

مثالی از Pipline در ALU

در نظر بگیرید که آرایه a,b را از هم کم کنیم و حاصل را در c بریزیم . برای آنکه بشود کاری را بصورت Pipline انجام داد ، باید آنرا به قسمتهای مستقل تقسیم کنیم . اگر یک عمل به n قسمت **مستقل** تقسیم شود می توان گفت که سرعت اجرا تقریباً n برابر می شود. هر چه تعداد تقسیمات زیاد شود ، سرعت اجرا هم بیشتر می گردد. برای آنکه a,b را از هم کم کنیم در ابتدا از b مکمل یک می گیریم و سپس با یک جمع می کنیم و حاصل را با a جمع بسته و در c قرارمی دهیم.

۱) $\overline{b}$

۲) $\overline{b}$ +۱

۳) a + $\overline{b}$ +۱

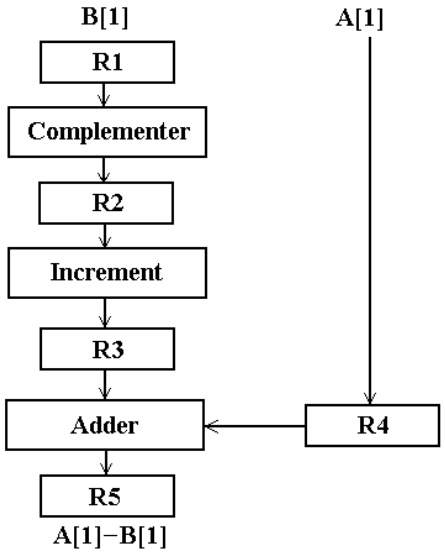
٤) $a + \overline{b} + 1 \rightarrow c$

دیاگرام و نتیجه حاصل از دیاگرام برای محاسبه c[I]=A[I]-B[I]

NO	R1	R2	R3	R4	R5
1	B1	---	---	---	---
2	B2	$\overline{B}$ 1	---	---	---
3	B3	$\overline{B}$ 2	$\overline{B}$ 1+1	A1	---
4	B4	$\overline{B}$ 3	$\overline{B}$ 2+1	A2	C1
5	B5	$\overline{B}$ 4	$\overline{B}$ 3+1	A3	C2
6	B6	$\overline{B}$ 5	$\overline{B}$ 4+1	A4	C3

بین هر مرحله از این برنامه باید یک ثبات قرار داشته باشد . زیرا لزومی ندارد که زمان اجرای هر مرحله باهم برابر باشد و بعد بر اساس آنها مراحل بعدی انجام شود . در روش Pipelining هیچگاه خروجی هر مرحله نباید با ورودی اتباط داشته باشد . در غیر این صورت نمی توان آن را به این حالت نوشت.

اولین محصول را در لحظه ٤ داریم و پس از آن در هر لحظه یک محصول داریم و این ٤ لحظه اول همان تاخیری است که باعث می شود بگوئیم تقریبا سرعت n برابر می شود و باید دقت کنیم که اولین A در لحظه ۳ وارد می شود .

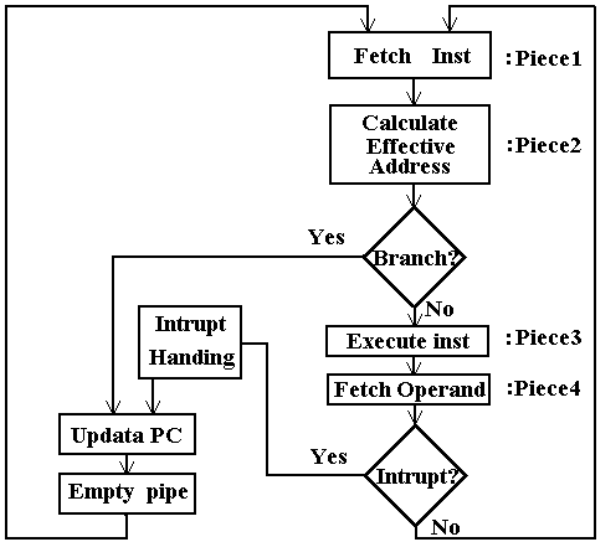


(شکل ۳-۸)

مثالی از پردازش موازی در اجرای دستورالعمل در CPU

در صورتیکه بتوان مراحل اجرای یک دستورالعمل را به n قسمت مستقل تقسیم کرد،می توان سرعت اجرای دستور را تقریباً n برابر نمود. در اجرای یک دستورالعمل حداقل می توان چرخه دستور را به دو قسمت مستقل و مساوی Execute, Fetch تقسیم نمود. در دیاگرام زیر چرخه

اجرای دستورالعمل به چهار قسمت مستقل تقسیم شده است .



(شکل ۳-۹)

در اینجا باید آدرس موثر حساب شود پس از آدرس باید آرگومان مربوط را Fetch شود وسپس دستورالعمل اجرا گردد. یعنی آرگومان در AC ذخیره شود. می دانیم که در تقسیم بندی رمزی اجرای دستورالعمل در دو قسمت Execute , Fetch است اما تقسیم بندی جدید همی قسمت ها را شامل نمی شود ، یعنی با دستورالعملهای زیر نیازی به بخش محاسبه آدرس موثر نداریم زیرا Operand ها در داخل ثبات هستند اما بهرحال باید از مراحل بگذرند که هر مرحله برای خود یک تاخیر دارد. سرعت مراحل به اجبار باید با هم برابر باشد زیرا دستورالعملها بصورت موازی اجرا می شود . در اجرای دستورالعمل پرش Update PC می رسیم . برای این دستور باید صفت خاصی در نظر گرفت که برای همین از Empty Pipe استفاده می شود که اگر دستورالعمل از نوع وقفه باشد به برنامه سرویس دهی وقفه می رود و وقفه را سرویس دهی می کند . وقتی که از دستورالعمل پرش ویا وقفه استفاده می شود کارایی روش Pipeline کاسته می شود . مثلاً در نظر بگیرید دارای دو عملیات B,A باشیم .

A : LDA 100

B : JMP 50

A را در ابتدا وارد می کنیم ، قطعا باید A زودتر اجرا شود وسپس B اجرا گردد . حال اگر A وارد سیر غیر پرشی وB وارد سیر پرشی شود باید سرعت هر دو با هم برابر باشد ، چرا که در غیر

اینصورت باعث ایجاد اختلال در الگوریتم می شود . به همین دلیل در دو دستور ابتدا سطر ۱۰۰ ، Load می شود سپس عمل پرش صورت می گیرد. یعنی دو سطر باید هماهنگ وهم سرعت باشند . همه ی دستورات و مراحل باید خودشان را با کند ترین دستوالعمل تطبیق دهند.

تمرینات

۱- وظایفی که CPU بعهده دارد را بصورت فلوچارتی نمایش دهید؟

۲- توضیح دهید که با چندنوع کدگذار (Encoder) می توان چهار وقفه سخت افزاری رابه یک پردازنده اعمال نمود؟

۳- اگر سرعت CPU کامپیوتری 133 MHZ باشد ، تعداد ریزعملهای انجام شده در مدت دو ثانیه در این کامپیوتر چه مقدار می تواند باشد؟

٤- مدت زمانی که طول می کشد تا دستورالعمل ذخیره سازی اجرا شود چند ثانیه می باشد؟ (فرض کنید سرعت CPU برابر با 100 MHZ است .)

۵- مدت زمان اجرای قطعه برنامه زیر را در کامپیوترهایی با سرعت 100 MHZ و133 MHZ بدست آورده وبا یکدیگر مقایسه نمایید؟

```
1        LDA
2        ADA
DEC
3        SBA
INC
4        STA
```

٦- یک ماشین دارای دستورات ١٦ بیتی است وهر آدرس حافظه ٦ بیتی می باشد . بعضی از دستورات تک آدرسی وبعضی دوآدرسی می باشند . اگر n دستورالعمل دوآدرسی باشند ، حداکثر تعداد دستورات ممکن تک آدرسی چندتا است؟

۱) $2^6 \times (16 \times n)$ ۲) $2^{16} - 2^{12} \times n$ ۳) $2^{10} - 2^6 \times n$ ٤) $2^{10} - n$

۷- به ماشین وان نیومن امکاناتی را اضافه کنید که توسط آن دستورات با آدرس دهی نوع Indexing رابتوان اجرا نمود؟

۸- فلوچارت اجرایی دستورالعملهای زیر را ترسیم نمایید؟

1) LDA m 10
2) LDA I 10
3) LDA x 10

۹- به ماشین وان نیومن امکاناتی را اضافه نمایید که توسط آن بتوان دستورات پرش شرطی را اجرا نمود؟

۱۰- کدام دستورالعمل این عملیات را به ترتیب انجام می دهد؟

1) MAR <----- R2
2) DR <----- M[MAR]
3) MAR <----- DR
4) M[MAR] <---- R1 , R2 <---- R2 + 1

۱) Move مستقیم R1 با اضافه شدن خودکار ثبات R2 (Post - Increment)

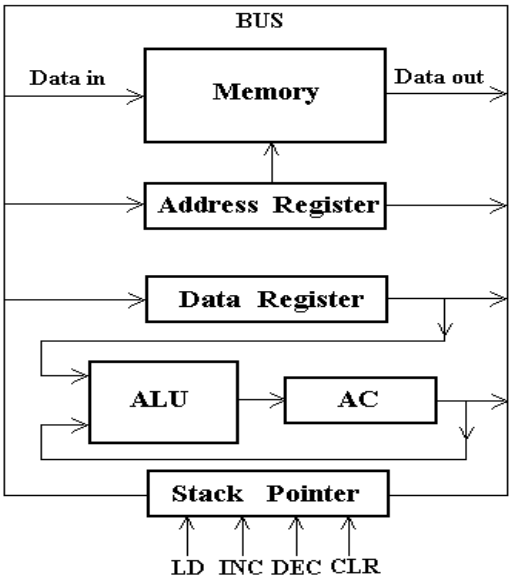
۲) Move غیرمستقیم R1 وMove غیرمستقیم R2 واضافه شدن خودکار R2

۳) Move مستقیم R1 با اضافه شدن خودکار ثبات R2

٤) Push مستقیم R1 براساس ثبات R2

۱۱- از یک کامپیوتر با سازمان اکومولاتور ، باس وتعدادی از ثباتهاوحافظه ومسیرهای مبادله اطلاعات نشان داده است . SP به محل ثربالای ثشته اشاره می کند . اگر دستورات POP AC , PUSH AC در این کامپیوتر تعبیه شده باشند، تعداد Clock لازم برای فقط سیکل های اجرایی (Executecycle) دو دستور PUSH , POP بترتیب چیست ؟

۱) ۲و٤ ۲) ٤و۲ ۳)۲و۲ ٤) ۳و۳

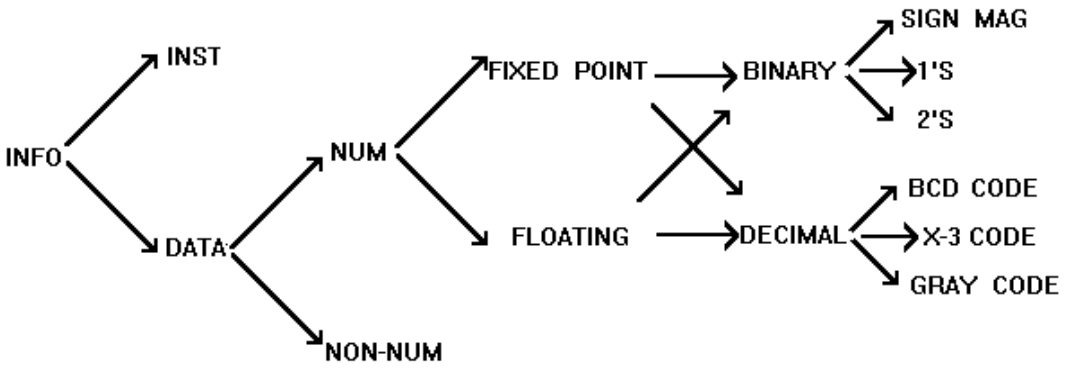


۱۲- در پیاده سازی .....انجام یک عمل را به چند قسمت مجزا تقسیم می کنند . زمان لازم برای اجرای یک دستورالعمل در حالت کلی برابر با مجموع مدت تأخیر در هر قسمت نیست ، بلکه برابر با ..... می باشد . با پیاده سازی این روش سرعت اجرای دستور افزایش می یابد اما اگر .....استفاده شود کارایی آن از بین می رود.

فصل دوم- طراحی واحد محاسبه و منطق

۱-نحوه نمایش داده در کامپیوتر

برای طراحی یک Alu لازم است بدانیم محاسبات مورد نظر می خواهد برروی داده ای با چه قرار داد نمایش اعمال شود.



(شکل ۱-۱)

معرفی کدهای دهدهی (Decimal Code) وآشنایی با ضوابط حاکم بر آنها در انجام محاسبات .

رقم در مبنای 10	BCD	EX - 3	Gray	EX - 3 Gray
۰	۰۰۰۰	۰۰۱۱	۰۰۰۰	۰۰۱۰
۱	۰۰۰۱	۰۱۰۰	۰۰۰۱	۰۱۱۰
۲	۰۰۱۰	۰۱۰۱	۰۰۱۱	۰۱۱۱

۰۱۰۱	۰۰۱۰	۰۱۱۰	۰۰۱۱	۳
۰۱۰۰	۰۱۱۰	۰۱۱۱	۰۱۰۰	٤
۱۱۰۰	۰۱۱۱	۱۰۰۰	۰۱۰۱	٥
۱۱۰۱	۰۱۰۱	۱۰۰۱	۰۱۱۰	٦
۱۱۱۱	۰۱۰۰	۱۰۱۰	۰۱۱۱	۷
۱۱۱۰	۱۱۰۰	۱۰۱۱	۱۰۰۰	۸
۱۰۱۰	۱۱۰۱	۱۱۰۰	۱۰۰۱	۹

(جدول ۱ – ۱)

۱-۱- اعداد بی سی دی (BCD)

چون در تبدیل اعداد به باینری با تقسیمات متوالی وقت زیادی گرفته می شود، لذا در بعضی از کامپیوترها از کد BCD استفاده می کنند.

$(37)_{10} = (100101)_2$

BCD = 0011 011
3 7

در مبنای ۱۰ در مبنای ۲

$$A - B = A + \overline{B} + 1,\ A + 9's(B) + 1$$

عیب اول BCD

وقتی به مبنای ۱۰ می رویم، بدست آوردن 9's اعداد BCD مشکل می باشد ونیازمند طراحی مدارات جدیدی است . مثلاً در تبدیل عدد ۳ به ٦ که درواقع عدد ٦ متمم ۳ در روش 9's می باشد مشکل وجود دارد. یعنی نمی توان به سادگی با تبدیل صفرها به یک ها ویک ها به

صفرها متمم یک عدد را از روش 9's بدست آورد . ولی درروش EX-3 می توان به راحتی 9's یک عدد را با تبدیل یکها به صفر وصفرها به یک بدست آورد .

مثال : $(1)_{10} = (0100)_{X-3} \xrightarrow{9's} (1011)_{X-3} = (8)_{10}$

عیب دوم BCD

عیب دوم این است که حاصل جمع دو عدد BCD ، الزاماً BCD نیست . در مثال۱، دو عدد BCD با هم جمع شدند وحاصل نیز BCD است ولی در مثال ۲ دوعدد BCD با هم جمع شده اند وحاصل BCD نمی باشد .

* برای این که عدد را به BCD تبدیل کنیم باید مقادیر را که در رنج BCD نیستند با ٦ جمع کنیم .

مثال ۱

۰۰۱۱ +
3
۰۱۱۰
+ 6
۱۰۰۱
9

مثال ۲

7 + ۰۱۱۱ +
+ 5 ۰۱۰۱
12 + ۱۱۰۰ حاصل BCD نیست
۱۱۰ دوعددباهم جمع شوند
$(12)_{10}$ ۱/۰۰۱۰ وحاصل BCD است.

حاصل BCD است .

۱-۲- اعداد افزایش سه تایی (EX - 3)

با اضافه کردن عدد۳ به کد BCD اعداد X - 3 بوجود می آید . اعداد ۰ به ۹ و ۱ به ۸ و .... با ایجاد یک 9's به هم تبدیل می شوند بنابراین مشکل اول در کد BCD ، در کد EX-3 حل می شود ولی مشکل دوم در کد EX-3 نیز وجود دارد.

مثال ۱

۰۱۱۰ 7 + ۱۰۱۰

3 +

۱۰۰۱ 5 ۱۰۰۰

6

۱۱۱۱ - 12 ۱۰۰۱۰ +

9

۰۰۱۱ ۰۰۱۱

۱۱۰۰ ۱۰۱۰۱

مثال ۲

در مثال ۱ انتظار جواب ۹ برای کد EX-3 یعنی ۱۱۰۰ رادایم که باید ۳ واحد از آن کم کنیم تا جواب درست حاصل شود ودر مثال ۲ انتظار جواب ۱۲ را داریم که باید با ۳ جمع شود تا ۱۰۱۰۱ که معادل ۱۲ است بدست آید . پس در بعضی از مواقع ۳ واحد کم می شود ودر بعضی از مواقع ۳ واحد اضافه می شود .

نکته : هر جا که رقم نقلی رخ دهد ۳ واحد اضافه می کنیم واگر رقم نقلی رخ ندهد ۳ واحد کم می کنیم .

نکته : کد EX-3 بهتر از کد BCD می باشد .

نکته : در سرعت ها و فرکانس های بالا ، به دلیل میدان مغناطیسی بالا ، Noise ایجاد می شود .

۱-۳- کد گری (Gray)

در این کد هر عدد، با عدد قبلی و بعدی آن در یک بیت اختلاف دارد . درا ین روش احتمال خطا و Noise بسیار کاهش می یابد چون عملاً از یک شمارنده (Counter) استفاده می شود . البته در کد BCD و EX-3 چون خطوط تغییر زیاد می باشند، ممکن است فیلیپ فلاپهای موجود در شمارنده همزمان عمل نکند وبدین صورت عمل کنند.

۰۰۱۱	۰۰۱۰	۰۱۱۰	۰۱۰۰

۳ <----------> ٤

بنابراین احتمال اینکه هنوز فیلیپ فلاپی باشد که اثر نکرده باشد وسیستم، آن عدد را به عنوان عدد بعدی قبول کند وجود دارد، لذا دچار اشتباه می شود . (مانند مراقبین جلسه که با گفتن اجرای دستور شماره N عمل شماری N را انجام می دهند). ولی در Gray Code چون فقط یک فیلیپ فلاپ تغییر می کند این خطا به حداقل می رسد.

۳ <-----> ٤

کد گری محاسبه پذیر نیست و برای انتقال داده هاست و با استفاده از آن اطلاعات واقعی از یک محیط نمونه گیری شده و در جای دیگر انتقال داده می شود . کاربرد بعدی آن در طراحی شمارنده واحد کنترل می باشد .

۱-۳-۱- کاربردهای کد خاکستری

کاربرد اول : طراحی شمارنده واحد کنترل

در داخل واحد کنترل عملیات براساس شماره ونظم خاصی انجام می گیرند و برای اینکه این مراحل درست و منظم انجام شوند نیاز به روشی است که اعدادی که نشان دهنده ی هر مرحله است ، به درستی بعد از هم شمارش شوند که این کار با استفاده از کد گری امنیت بیشتری خوااهد داشت .

کاربرد دوم : در انتقال اطلاعات

۱) مثلاً اگر بخواهیم متوسط دمای هوای محیط را در یک روز بدست آوریم ، کامپیوتری داریم که تشخیص می دهد ، درجه هوا طبق یک مسیر پیوسته افزایش یا کاهش می یابد.

مثلاً درجه هوا از ۳ به ٤ تغییر می کند ، لذا باید ۳ بیت عوض گردد که زمان زیادی می برد واحتمال خطا وجود دارد.

۳ ----> ۰۰۱۱

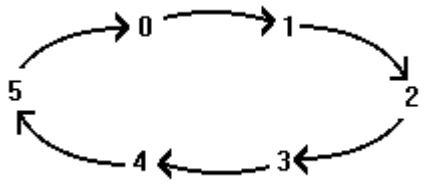
٤ ----> ۰۱۰۰

۲) در تبدیل اطلاعات آنالوگ به دیجیتال

انتقال اطلاعات از آنالوگ به دیجیتال بدین صورت انجام می گیرد که اگر برای مثال خطوط تغییرات زیاد باشد، ایجاد Noise می کند. لذا باید با سرعت کمتری انتقال دهیم تا Noise آن کمتر گردد. در مخابرات از سیستم Gray استفاده می شود تا Noise بوجود نیاید .

نکته : کد Gray در محاسبات به کار نمی رود ، چون اگر عمل محاسبه از رنج خارج شود امکان هیچگونه اصلاحی وجود ندارد . ولی در انتقال اطلاعات وشمارنده های واحد کنترل کاربرد بسیار وسیعی دارد . در مخابرات می توان از آنها برای کم کردن حجم کابلها استفاده نمود .

نکته : در ماشین Von Neuman حداکثر با ٦ عمل دستورات انجام می شد .پس شمارندی آن بصورت زیر عمل می کند وباید Gray باشد. یعنی هر عدد با عدد قبلی فقط در یک بیت اختلاف داشته باشد .



(شکل ۱ – ۲)

۱-٤ - کد گری - افزایش سه تایی (EX-3 Gray)

در تبدیل عدد ۹ به عدد ۰ که در واقع عدد ۹ عدد قبل از ۰ است ، بیش از یک بیت اختلاف وجود دارد (۰۰۰۰ < ------ ۱۱۰۱) لذا کد جدیدی بنام کد EX-3 Gray را معرفی می کنیم ، که در آن از EX-3 معادل Gray می گیریم .

۱-۵ - اعداد ممیز شناور (Floating Point)

اعداد ممیز شناور که با نمای علمی نمایش داده می شوند وبصورت زیر می باشند.

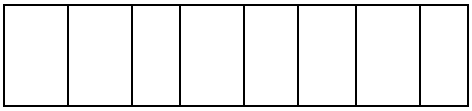
توان

$$N = MB^E$$

پایه مانتیس

مثال : $125 = 125 \times 10^2$

نقطه اعشار وجود ندارد و علامتی فرضی است . فقط مانتیس و توان درج می شود . اگر پایه B یک عدد ثابت فرض شود . (B=Const) در ارائه مطلب هیچ کمبودی احساس نخواهد شد . اگر B را ثابت د رنظر بگیریم در آنصورت هر عدد را می توان بصورت N = (M , E) نشان داد.



نقطه اعشار

در کامپیوترهای مختلف نیز یک قرار داد ثابت وجود دارد . بخشی از مکان حافظه را به مانتیس وبخشی را به توان اختصاصی می دهند . در کامپیوترها نقطه اعشار در سمت چپ قسمت M بصورت ثابت فرض می شود و مانند مثال زیر نقطه اعشار بصورت شناور نمی باشد . اعداد باید در کامپیوتر یصورت نرمال ذخیره شوند .

مثال :

$$125 = 1.25 \times 10^{2} = 12.5 \times 10^{1} = 125.0 \times 10^{0}$$

سؤال : چرا در داخل کامپیوتر از اعداد با ممیز شناور استفاده می شود ؟

چون در این روش دامنه نمایش اعداد وسیعتر می شود .

فرض کنید که ما چهار بیت فضا داشته باشیم که با این چها ربیت محدوده 0 تا 15 را در روش Fixed Point می توانیم نشان دهیم ولی در Floating Point به صورت زیر عمل می شود .

	Fixed Point	B = 2	B = 16
0000	0	0	0
0001	1	0.25	0.25
0010	2	0.5	0.5
0011	3	0.75	0.75

0100	4	0	0
0101	5	0.5	4
0110	6	1	8
0111	7	1.5	12
1000	8	0	0
1001	9	1	64
1010	10	2	128
1011	11	3	198
1100	12	0	0
1101	13	2	1024
1110	14	4	2048
1111	15	6	3072

(جدول ۱- ۲)

در اعداد ممیز شناور (Floating Point) دامنه نمایش افزایش می یابد .اما دقت نمایش ثایین می آید . (یعنی اعداد مابین آن را نمی توان نمایش داد) واز بالا به ثایین یعنی از سمت اعداد کوچک به سمت اعداد بزرگ که می آییم ، دقت اعدادکمتر می شود. ولی اعداد رنج ثایین دقت خوبی دارند.

* در اعداد ممیز شناور (Floating Point) هر چه پایه (Base) بیشتر باشد ، دامنه نمایش بیشتر اما دقت نمایش کمتر خواهد بود.

۱-۵-۱- معایب اعداد ممیز شناور

عیب اول: هر چه مقدار پایه (Base) بیشتر باشد ، دامنه وسیعتر ودقت اعداد کمتر است .

عیب دوم : یک عدد چندین فرم نمایش را داراست ،که کامپیوتر دچار مشکل می شود ومقایسه توسط کامپیوتر دشوار می گردد. برای رفع عیب دوم اعداد ممیز شناور عدد را بصورت نرمال در می آوریم و آنها را در داخل کامپیوتر ذخیره می کنیم .

۱-۵-۲- نرمال سازی

اولاً) وقتی عددی رامی خواهیم در کامپیوتر ذخیره کنیم باید بگونه‌ای باشد که مانتیس آن عددی بین صفر ویک باشد (یعنی $0 < M < 1$)

ثانیاً) اولین رقم بعد از ممیز صفر نباشد .

برای مثال اگرعدد ۲/۵ رابخواهیم درداخل کامپیوتر نشان دهیم می توانیم صورتهای مختلف زیر را برای آن داشته باشیم :

$$2.5 = 25 \times 10^{-1} = 0.25 \times 10^{1} = 0.025 \times 10^{2}$$

طبق قاعده نرمال سازی دو مورد 25×10^{-1} æ0.025×10^{2} مورد قبول نیستند . زیرا در عدد 25×10^{-1} مانتیس باید عددی بین صفر ویک باشد ، که اینطور نیست ودر عدد 0.025×10^{2} اولین رقم بعد از ممیز باید غیرصفر باشد که در اینجا هم رعایت نشده است .

نحوه نمایش اعداد Floating Point در IMB 360/370 به صورت زیر است .

اعداد ممیز شناور فضای ٤ بایتی یعنی ۳۲ بیتی را اشغال می کنند ومبنا (Base) عدد ١٦ می باشد . از این ۳۲ بیت سمت چث ترین بیت ، بیت علامت می باشد و ۷ بیت متعلق به توان و ٢٤ بیت باقی مانده متعلق به مانتیس عدد می باشد.

	7 Bit	24 Bit

مانتیس ۰ توان بیت

علامت

Floating Point Fixed Point

حداکثر	$16^{63} \times 1$	$+2^{31}$
حداقل	$16^{-64} \times 1$	$-2^{32} - 1$

مثال : آیا می توان عددی را به فرم زیر در حافظه سیستم کامپیوتر داشته باشیم ؟

Base = 2

۱	۰۰۰۱۰۱۰	۰۱۰۱۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰

E ۰ M

بیت علامت

جواب:

خیر، زیرا اولین رقم بعد از ممیز صفر می باشد. پس می بایست عدد را به فرم نرمال در آورد . برای عمل نرمال سازی با انتقال مانتیس به سمت چث (Shift Left) درواقع مانتیس را دو برابر می کنیم و برای جبران این عمل به ازای هر بار عمل شیفت، از توان یکی کم می کنیم که در واقع با این کار عدد را تقسیم بر دو می نماییم و بدین صورت از بروز خطا در عمل نرمال سازی جلوگیری می کنیم . پس داریم :

عدد نرمال شده = $2M \times 2^{E-1}$

۱	۰۰۰۱۰۰۱	۱۰۱۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰

بیت E ۰ M

علامت

مثال : آیا می توان عددی را به فرم زیر در حافظه کامپیوتر داشته باشیم ؟

Base = 16

۰	۰۰۱۰۱۰۰	۰۰۰۰۱۰۰۰۰۰۰۰۰۰۰۰۰۰۰

بیت E ۰ M

علامت

جواب

خیر، زیرا اولین رقم بعد از ممیز صفر می باشد. پس می بایست عد رابه فرم نرمال در آورد . در ماشینی که با پایه ۱٦ کار می کند برای شیفت دادن به چث باید ٤ بیت به ٤ بیت به سمت چث شیفت دهیم وبرای جبران آن از توان ۱٦ یک واحد کم می کنیم . پس در ماشینهای با Base = 16 باید حتماً ٤ بیت صفر بعد از نقطه ممیز اعشار وجودداشته باشد ودر غیر اینصورت نمی توان عملیات نرمال سازی راانجام داد. بنابراین اعداد با کمتر از ٤ صفر بعد از ممیز را ، به صورت نرمال فرض می کنیم .

$$(0.00001)_2 \times 16^{20} = (0.10000)_2 \times 16^{20} \times 16^{-1} = (0.1000)_2 \times 16^{19}$$

$$2^{-4} = 16^{-1}$$

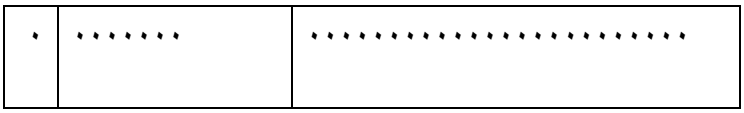
۰	۰۰۱۰۰۱۱	۱۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰

بیت E ۰ M

علامت

نکته : اگر دو عدد نرمال راباهم جمع کنیم یا درهم ضرب کنیم حاصل ممکن است نرمال نباشد وباید دوباره به نرمال تبدیل گردد. به همین خاطر درکامپیوتر، اعمال روی اعداد ممیز شناور زمان زیادی می گیرد.

سؤال : اگر مانتیس ما صفر باشد ماشین چگونه با آن برخورد می نماید؟



بیت E ۰ M

علامت

برای نرمال سازی حداکثر تا ٦ بار عمل Shift Left را انجام می دهد وسپس متوقف می شود. چون در عمل نرمال سازی به ازای هر شیفت به سمت چپ ، یک واحد از توان کم می‌گردد ودراقداماتی که برای نرمال سازی عدد صفر به عمل می آید، توان به سمت منفی تر شدن میل می کند. باید کاری کنیم که منفی ترین عدد شکل صفر باشد . برای رفع این مشکل باید توان بصورت

اریب دار، ذخیره گردد . اگر n بیت برای توان داشته باشیم نیاز است که اریب برابر 2^{n-1} باشد .

۱۰۰۰۰۰۰ <------ > ۰۱۱۱۱۱۱

$-(2)^{7-1} = -64$ $+(2)^{7-1} - 1 = 63$

با اضافه کردن ٦٤ واحد به توان بصورت قدرمطلق نه بصورت علامت دار (چون در آن صورت از رنج خارج می گردد) می توان این مشکل را حل نمود . با این روش منفی ترین عدد ، عددی خواهد بود که همه بیتهایش صفر است واز دیدگاه کامپیوتر عدد به فرم 0×16^{-64} ، صفر واقعی می باشد .

اعداد ممیز شناور به صورت اریب دار در کامپیوتر ذخیره می شوند. چون ممکن است که حاصل جمع دو عدد اریب دار الزاماً اریب دار نباشد ،که در آن صورت مجبور به اریب دار کردن حاصل عملیات می باشیم. پس به همین خاطر در کامپیوتر اعمال روی اعداد ممیز شناور زمان زیادی می برد . علاوه بر این زمان ممکن است اعداد ورودی و محصولات نرمال نباشند و زمانی هم برای نرمال سازی لازم است .

مثال : عدد ۸۰ برای کامپیوتر IBM 360/380 به چه صورتی نمایش داده می شود (بصورت Floating Point) .

حل :

$$Number = MB^E$$

$$80 = M \times 16^E = M \times 16^2 \Rightarrow M = \frac{80}{16^2} = 0.3125$$

شاید این سوال پیش آید که چرا E را برابر با ۱ نگرفتیم ؟ علت آن این است که درآن صورت ، $M = \frac{80}{16} = 3$ می شود حال آنکه M باید عددی بین ۱ > M > ۰ باشد. به همین علت E را برابر با ۲ می گیریم.

$$M = 0.3125 = (0.0101)_2 , E = 2$$

$$E + 64 = 66 = (100010)_2$$ اریب دار شد

۰	۱۰۰۰۰۱۰	۰۱۰۱۰۰۰۰۰۰۰۰۰۰۰۰۰۰۰
	E	M

بیت

علامت

چون Base = 16 می باشد پس نیازی به نرمال سازی نیست ، بلکه عدد حاصل نرمال است. البته می توانستیم که E = 3 در نظر بگیریم که در اینصورت هم مقدار ۱ > M > ۰ می باشد ولی هر چه مقدار E بزرگتر گرفته شود نیاز به عملیات نرمال سازی بیشتری خواهد بود.

$M = 0.01953125 = (0.000001010)_2 , E = 3$

$E + 64 = 3 + 64 = 67 = (1000011)_2$ (اریب دار)

۰	۱۰۰۰۰۱۱	۰۰۰۰۰۱۰۱۰۰۰۰۰۰۰۰۰۰۰

بیت E ۰ M

علامت

۱

۰۱۰۱۰۰۰۰۰۰۰۰۰۰۰۰۰۰

۰	۱۰۰۰۰۱۰	۰۱۰۱۰۰۰۰۰۰۰۰۰۰۰۰۰۰

بیت E ۰ M

علامت

در کامپیوتر های شخصی (pc) نحوه ی ذخیره سازی به سه روش double و float و integer بوده و مانتیس در آن بین ۱ و ۲ است . عدد (۰.۱۰۱۰۱) را در پایه ۲ در نظر بگیرید . اگر بخواهیم این عدد را در یک pc ذخیره کنیم ، بصورت زیر خواهد شد :

1.0101×2^{-1}

که در pc فقط اعداد (0.0101) به عنوان مانتیس و (1-) به عنوان توان درج خواهد شد .

در مورد محاسبات بین اعداد اعشاری به مثال زیر توجه کنید . البته این محاسبات با توجه به کامپیوتر های IBM است :

فرض کنید دو عدد A و B را می خواهیم باهم جمع کنیم . چون بیت های توان ۳ تا است پس میزان اریب ۲ ۲ یعنی ٤ خواهد بود . ابتدا باید توان ها را یکی کنیم . مسلما نمی توان عدد B را به چپ شیفت دهیم زیرا بیت پر ارزش آن از بین خواهد رفت به همین خاطر عدد A را به راست شیفت می دهیم . سپس حاصل جمع دو مانتیس را در C قرار می دهیم . در انتهای جمع چون رقم نقلی داریم ، کلیه ی اعداد مانتیس حاصل را به راست شیفت داده و یکی به توان اضافه می کنیم و سپس عدد C را نرمال سازی می کنیم .

A :

۰	۰۱۰	۱۰۱۱۰۰۰۰

B :

۰	۰۱۱	۱۱۰۱۰۰۰۰

C :

A :

۰	۰۱۱	۰۱۰۱۱۰۰۰

B:

۰	۰۱۱	۱۱۰۱۰۰۰۰

۰	۱۰۰	۱۰۰۱۰۱۰۰

تمرینات

۱- مزایا ومعایب کد BCD را بیان نمایید؟

۲- علت افزایش ٦ واحدبه کد BCD در برخی از عملیات ریاضی انجام گرفته بر روی این کد چیست ؟ (با مثال توضیح دهید)

۳- مزایا ومعایب کد افزایش سه تایی (EX-3) را بیان نمایید؟

٤- استفاده از ..... برای ..... مفید است . زیرا کدهای مجاور متوالی تنها در یک بیت با هم اختلاف دارند.

۵- مزایا ومعایب کد Gray وهمچنین موارد استفاده این کد را بیان نمایید؟

٦- لزوم استفاده از کد Gray EX-3 را بیان نمایید ؟

۷- در n بیت اگر عددی به فرم علمی ذخیره شود ، ..... نسبت به اعداد با ممیز ثابت در اختیار خواهد بود که البته ..... این روش کمتر است .

۸- نمایش اعداد به فرم ممیز شناور چه مزایایی دارد وچرا بر روی اعداد با ممیز شناور عمل نرمال سازی انجام می گیرد ؟

۹- عملیات بر روی اعداد ممیز شناور د رمقایسه با ممیز ثابت ، توسط کامپیوتر سریعتر انجام می گیرد . چرا ؟

۱۰- عدد $(80A00000)_{16}$ موجود در حافظه کامپیوتر IBM 360 / 370 نشان دهنده چه عدد Floating Point در مبنای دسیمال می باشد؟

۲- مدارات جمع کننده

درطراحی مدار جمع کننده کامپیوتر معمولاً از روش متمم دو (2's) استفاده می شود و باید مدار طراحی شده دارای سرعت کافی وبالایی باشد . زیرا تمامی عملیات جمع ، تفریق وضرب وتقسیم توسط این مدار انجام خواهد گرفت . جمع کننده یک مدار ترکیبی است وهر مدار ترکیبی را می توان در دو سطح پیاده سازی نمود. هر چه تعداد سطوح کمتر گردد عمل جمع سریعتر صورت می گیرد.

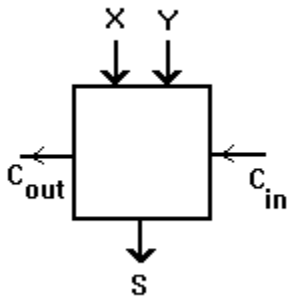
اگر چه می توان از لحاظ تئوری برای طراحی جمع کننده n بیتی آنرا در دو سطح (سریعترین حالت ممکن) پیاده سازی نمود. اما از لحاظ عملی امکان پذیر نیست . زیرا برای طراحی چنین مداری ، قطعاً به گیتی با n ورودی نیاز خواهد بود که اگر n عدد بزرگی باشد ، عملاً گیتهایی با ورودی بالا کمتر وجود دارند . به این خاطر نمی توان مدار جمع کننده را برای حالتی که n عدد بزرگی باشد در دو سطح پیاده سازی نمود. بدین جهت مجبوریم که دو عدد n بیتی را به گروههای m بیتی تقسیم کنیم وسپس با هم جمع ببندیم ورقم نقلی (Carry) هر گروه را به گروه بعدی منتقل کنیم .

انواع جمع کننده ها

۲-۱ : طراحی جمع کننده کامل (Full Adder)

این مدار دو عدد تک بیتی را با هم جمع می کند و اگر رقم نقلی هم از قبل باشد در عملیات اثر می دهد .

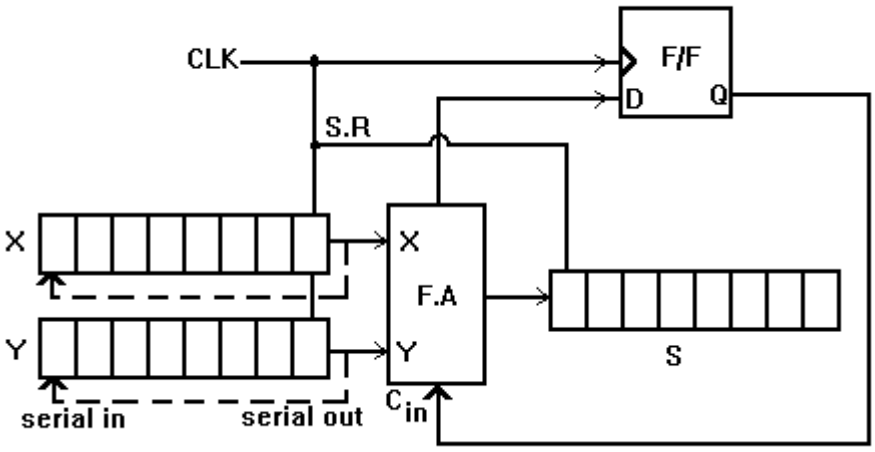
$$S = x \oplus y \oplus c_{in}$$
$$c_{out} = xy \oplus c_{in}(x \oplus y)$$



(شکل ۲-۱)

۲-۲ : طراحی جمع کننده سریال (Serial Adder)

این مدار برای جمع دو عدد n بیتی با حداقل قیمت بکار می رود ولی زمان زیادی مصرف می کند.



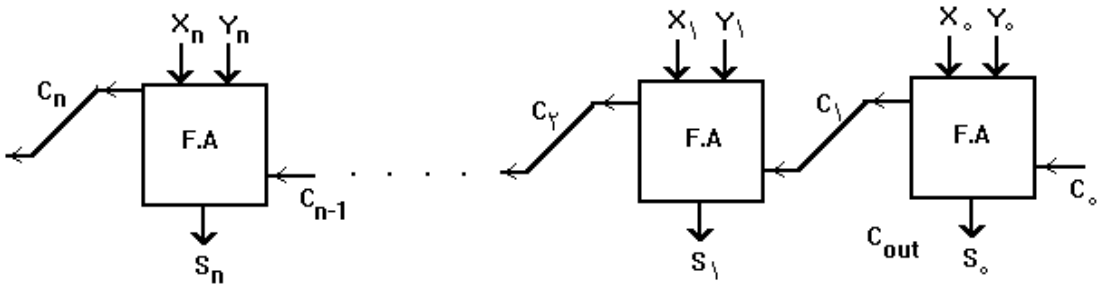
(شکل ۲-۲)

* هر بار که clk زده می شود ، یک شیفت به راست در ثباتهای s,y,x صورت می گیرد.

* اگر میزان تاخیر F.A برابر d باشد ومیزان تاخیر فیلیپ فلاپ(F/F) برابر D باشد آنگاه برای n بیت ، جمع زمانی معادل n(d+D) صرف خواهد شد . پس کمترین هزینه را داریم ولی زمان زیادی مصرف می شود.

* برای اینکه محتوای ثباتهای y,x از بین نرود ، باید وقتی شیفت به راست می دهیم دوباره Serial out - را به Serial - in وصل کنیم تا دوباره در انتهای عملیات عدد x,y را بطور کامل داشته باشیم.

۲-۳ : طراحی جمع کننده با رقم نقلی پله‌ای (Ripple Carry Adder)



(شکل ۲-۳)

d_A = زمان تأخیر در فیلیپ فلاپ اول

هر F.A پس از اعمال ورودیها بعد از یک d_A خروجی را به ما می دهد . برای n بیت، زمانی معادل nd_A نیاز می باشد . زیرا هر F.A باید منتظر رقم نقلی F.A قبلی باشد وسرعت در این روش نسبت به جمع کننده سریال بیشتر است وتعداد F.A ها برای یک عدد n بیتی برابر n تا می باشد ، پس هزینه آن بیشتر از جمع کننده سریال می شود .

٤-٢ : طراحی جمع کننده با پیش بینی رقم نقلی (Carry Lookahead Adder)

اگر بتوان رقم نقلی را پیش بینی کرد ، دیگر نیازی نیست که منتظر بمانیم تا رقم نقلی F.A قبلی به F.A بعدی برسد. بنابراین ، این روش که همان جمع کننده با پیش بینی رقم نقلی می باشد را پیشنهاد می کنیم تا عملیات جمع سریعتر انجام گیرد .

	c_{out}	y	x
١	٠	٠	٠
٢	c_{in}	١	٠
٣	c_{in}	٠	١
٤	١	١	١

حالت اول

این حالت نیاز به رقم نقلی ندارد واصطلاحاً در این حالت رقم نقلی دفن می شود.

حالت دوم وسوم

این حالت وابسته به رقم نقلی می باشد و اصطلاحاً رقم نقلی انتشار پیدا می کند .

(P_i = Propagated Carry)

حالت چهارم

این حالت نیاز به رقم نقلی ندارد چون خودش رقم نقلی را بطور خودکار تولید می نماید واصطلاحاً در این حالت رقم نقلی تولید می شود .

(G_i = Generated Carry)

وضعیت انتشار ۱ $P_i = x_i \oplus y_i$

وضعیت تولید ۲ $G_i = x_i y_i$

داشتیم : $c_{out} = xy + (x \oplus y) c_{in}$ ۳

$\xrightarrow{1æ2æ3}$ در دوسطح طراحی می گردد. $C_i = G_i + P_i C_{i-1} \rightarrow$

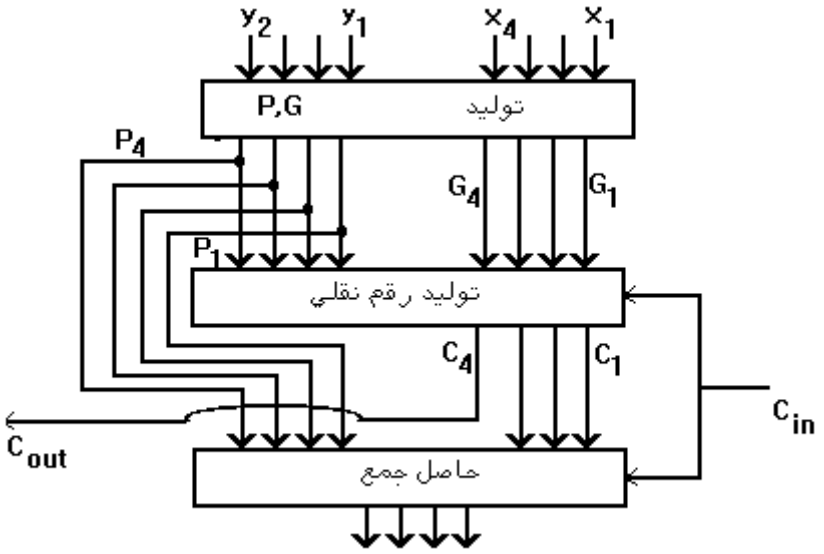
$C_0 = c_{in}$

$C_1 = G_1 + P_1 c_{in}$

$C_2 = G_2 + P_2 C_1 = G_2 + P_2 (G_1 + P_1 c_{in}) = G_2 + P_2 G_1 + P_2 P_1 c_{in}$

$C_3 = G_3 + P_3 C_2 = G_3 + P_3 (G_2 + P_2 G_1 + P_2 P_1 c_{in}) = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 c_{in}$

$C_4 = G_4 + P_4 C_3 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 c_{in}$



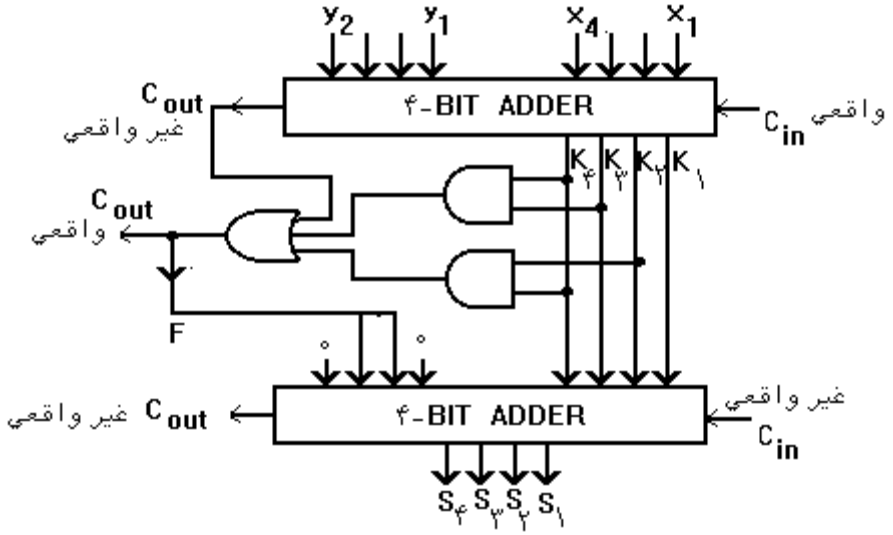
(شکل ۲-٤)

$S_i = x_i \oplus y_i \oplus c_{in} = P_i \oplus c_{in}$

درمجموع مدار ما دارای چهار سطح است . برای یک مدار ۱۰ بیتی هم نیاز به چهار سطح می باشد. می دانیم که هر F.A دو سطحی است ، پس کل این مدار به اندازه دو F.A تأخیر دارد

بنابراین ازنظر زمانی این روش سریعتر است. اما اگر این مدار را برای ۱۰۰ بیت طراحی کنیم ممکن است در مکان تولید رقم نقلی مشکل پیش آید ، زیرا که ما نیاز به گیتهایی با ۱۰۰ ورودی داریم که این کارمشکلی است چون گیتها نهایتاً دارای ۸ ورودی می باشند. پس تعداد سطوح در مکان تولید رقم نقلی زیاد می شود که زمان تأخیر یا delay این مدار جمع کننده افزایش می یابد.

مثال : جمع کننده‌ای برای جمع دو عدد بی سی دی (BCD) طراحی کنید . (٤ بیتی)



(شکل ۲-۵)

در صورتی حاصل عملیات جمع، BCD نیست که یکی از اعداد 15,14,13,12,11,10 حاصل شود یا درحاصل عملیات c_{out} رخ دهد که در این دو صورت تابع F فعال می شود .

نکته: هر دو Carry out ، غیر واقعی هستند وتابع F رقم نقلی ماست .

$K_{4K_3} \diagup K_2 K_1$

	۰۰	۰۱	۱۱	۱۰
۰۰			۱	
۰۱			۱	
۱۱			۱	۱
۱۰			۱	۱

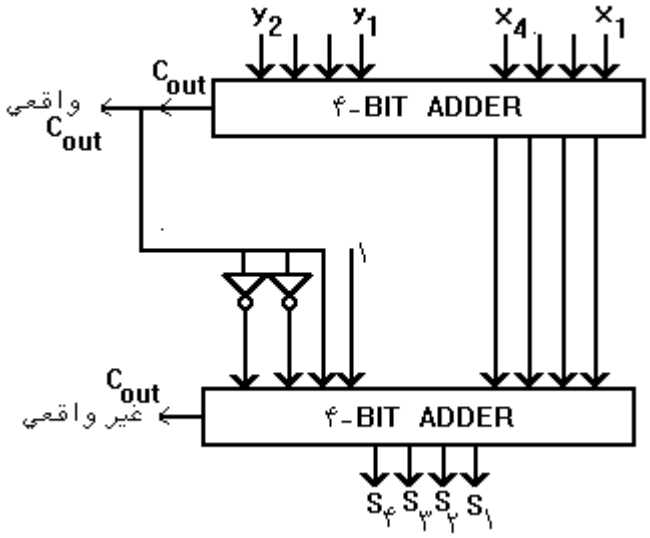
$$F = K_4\overline{K}_3K_2\overline{K}_1 + K_4\overline{K}_3K_2K_1 + K_4K_3\overline{K}_2\overline{K}_1 + K_4K_3\overline{K}_2K_1 + K_4K_3K_2\overline{K}_1 + K_4K_3K_2K_1$$

$$F = K_4K_3 + K_4K_2 + C_{OUT}$$

مثال : جمع کننده‌ای برای جمع دو عدد افزایش سه تایی (Execss-3) طراحی کنید ؟

حل : در این حالت اگر پس از عملیات جمع c_{out} داشته باشیم ۳ واحد به حاصل عملیات اضافه می کنیم واگر c_{out} نداشته باشیم ۳ واحد از حاصل عملیات کم می کنیم .

عدد	نمایش در مبنی دو			
+3	۰	۰	۱	۱
-3	۱	۱	۰	۱
	$\overline{c}_{out}$	$\overline{c}_{out}$	c_{out}	۱



(شکل ۲-٦)

تمرینات

۱- مداری برای جمع دو عدد BCD به روش Pipelining طراحی نمایید؟

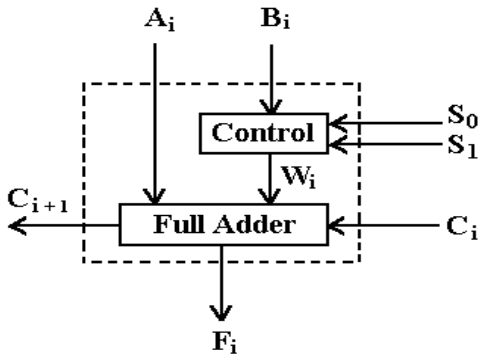
۲- مداری برای جمع دو عدد EX-3 به روش Pipelining طراحی نمایید؟

۳- چگونه می توان جمع کننده‌ای برای جمع دو عدد Gray طراحی نمایید؟

٤- برای جمع دو عدد n بیتی می توان با ترکیب دو سطحی سریعترین مدار را پیاده سازی نمود. اما این ایده در عمل مورد استفاده قرار نمی گیرد . چرا؟

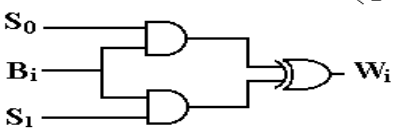
۵- برای ساخت یک ALU از بلوکهایی مشابه بلوک زیر استفاده می شود .

c1 بیت Gray متصل به اولین بلوک است و S0 , S1 برای تمام بلوکها یکسان است . مدار کنترل باید چگونه باشد تا توابع زیر را بتوان تولید کرد .

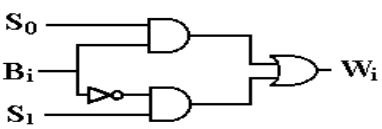


S_1	S_0	C_1	F
0	0	0	A
0	0	1	A+1
0	1	0	A+B
0	1	1	A+B+1
1	0	0	$A+\overline{B}$
1	0	1	$A+\overline{B}+1$
1	1	0	A−1
1	1	1	A

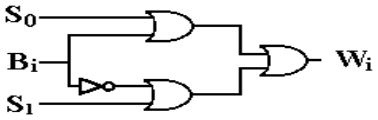
(1



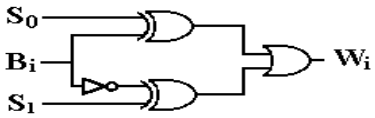
(2



(3



(4



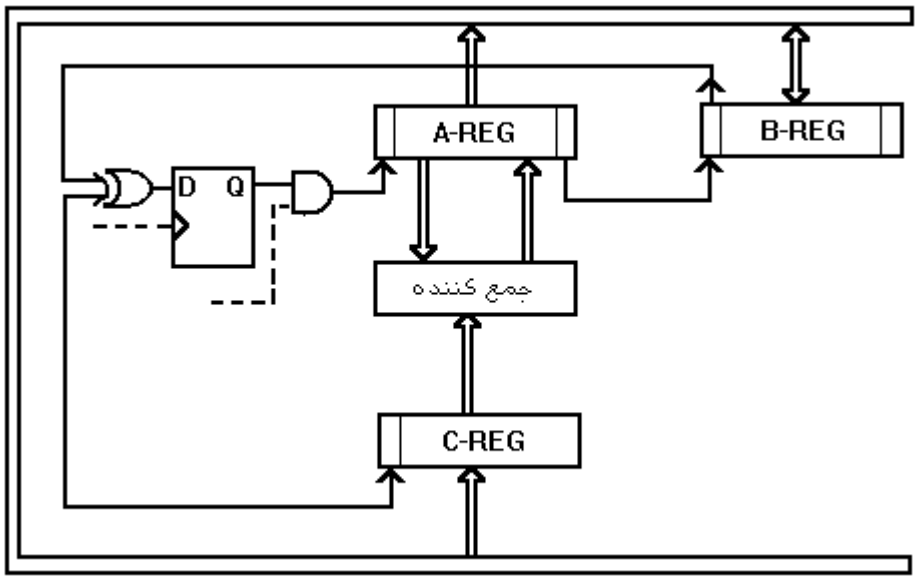
۳- مدارات ضرب کننده

در داخل کامپیوتر عمل ضرب توسط جمع های متوالی انجام می شود . برای عمل ضـرب از 3 ثبات استفاده می شود . وقتی دو عدد n بیتی در هم ضرب می شوند به یک فضـای 2n بیـت نیـاز دارند.

۳-۱ : ضرب کننده برای دو عدد قدر مطلق علامت

چون ضرب دو عدد قدر مطلق علامت برخلاف جمع آنها دارای مدار وعمل ساده تری است . لـذا ابتدا آنرا توضیح می دهیم تا با الگوریتم عمل آشنا تر شـوید . در ایـن عمـل کـه بـا الهـام از روش ضرب دستی به وقوع می پیوندد ، باید برای عمل ضرب برای دو عدد n بیتـی یـک ثبـات 2nبیتـی رادرنظربگیریم وبرای‌کل‌عمل ضرب بایدn بار شیفت به راست صورت گیرد .

مدار ضرب کننده برای ضرب دو عدد قدرمطلق علامت

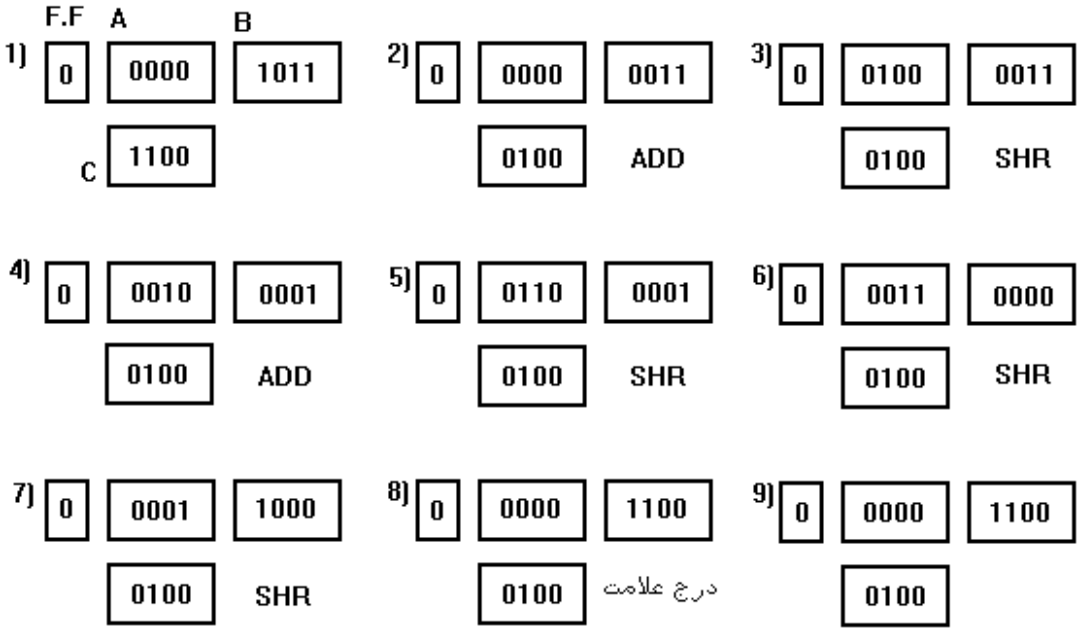


(شکل ۳-۱)

ابتدا به بیت کم ارزش عدد دوم (ثبات B) توجه می شود، اگر یک بود ثبات C و ثبات A را با هم جمع می کند وحاصل (ثبات B,A) به سمت راست شیفت داده می شود . حاصل عملیات که 2n بیتی می باشد ، در دوثبات B,A قرار می گیرد.

در فیلیپ فلاپ ، بیت علامت حاصلضرب که از پای انحصاری کردن بیت علامت ثبات B با C بدست می آید را ذخیره می کنیم وسپس بیت علامت ثبات B وثبات C را Clear می کنیم یعنی قدر مطلق می گیریم . سپس طبق الگوریتم قبل در هم ضرب می نماییم ودر پایان هم بیت علامت ضرب را در مکان مربوطه قرار می دهیم .

مثال : دو عدد قدرمطلق علامت 3- , 4- را به کمک مدار ضرب کنندی چهار بیتی به روش قدرمطلق علامت در هم ضرب نمایید وکلیه مراحل را نشان دهید ؟



به تعداد ShR که در مدار انجام می شود بیت داریم . چون اعداد چهار بیتی هستند باید چهار مرتبه شیفت به راست داشته باشیم ، قدم آخر نیز تأثیر بیت علامت است که در فیلیپ فلاپ D نگهداری شده است .

۳-۲ : ضرب دو عددمتمم 2 یا 2's

یک روش این است که مدار همان مدار سخت افزاری قبلی باشد وبا یک کارنرم افزاری دو عدد رااز متمم 2 به قدر مطلق علامت تبدیل کرده وپس از عمل ضرب عدد قدرمطلق رابه متمم 2 تبدیل نمود که چون این عملیات وقت گیر می باشد پس به صرفه نیست.

روش دیگر اینکه مداری طراحی کنیم که عیناً دو عدد متمم دو را درهم ضرب کند . دو عدد با توجه به علامتشان دارای چهار حالت نسبت به هم می باشند:

$$x \times y = ?$$

$$X = x_m \ldots\ldots\ldots x_1 x_0$$

$Y = y_n \ldots\ldots\ldots\ldots y_1 y_0$

1) $x_n = 0, y_n = 0$

2) $x_n = 1, y_n = 0$

3) $x_n = 0, y_n = 1$

4) $x_n = 1, y_n = 1$

حالت اول : هر دو عدد مثبت باشند ($y_n = 0, x_n = 0$)

که این روش عیناً براساس ضرب دو عدد قدر مطلق علامت حل می شود .

حالت دوم : اگر x_n منفی و y_n مثبت باشد ($y_n = 0, x_n = 1$)

این حالت را با یک مثال دنبال می کنیم.

مثال: حاصلضرب دو عدد 3-,5 را به روش دستی نمایش دهید؟

حل : چون دو عدد چهار بیتی هستند به فضای ۸ بیتی برای نمایش حاصلضرب نیاز داریم.

۱۱۰۱	-3
۰۱۰۱ ×	5 ×
۰۰۰۰۱۱۰۱	-15
۰۰۰۰۰۰۰۰	
۰۰۱۱۰۱۰۰	
۰۰۰۰۰۰۰۰ +	
۰۱۰۰۰۰۰۱	

مشاهده می شود که حاصل عملیات نادرست می باشد . علت اینست که چون عدد منفی از یک مکان 4 بیتی به مکان8 بیتی منتقل می شود، می بایست آن عدد منفی رابسط علامت بدهیم . پس داریم :

۱۱۰۱

۰۱۰۱ ×

۱۱۱۱۱۱۰۱

۰۰۰۰۰۰۰۰

۱۱۱۱۰۱۰۰

۰۰۰۰۰۰۰۰ +

۱۱۱۱۱۰۰۰۱

$(11110001)_2 = -(00001111)_2 = -15$

حال می خواهیم این عملیات را درکامپیوتر انجام دهیم .

در واقع ما نیازی به فیلیپ فلاپ قبلی نداریم . زیرا این فیلیپ فلاپ برای ضبط بیت علامت بود که به آن نیازی نیست ، چون هر بار که حاصل ثبات A را در ثبات C می ریزیم ، اگر نیاز به شیفت به راست باشد ، ثبات C رابسط علامت می دهیم واین عمل توسط سیمی که بیت علامت را در خودش کپی می کند صورت می گیرد .

نکته :در عمل ضرب هنگامی که می خواهیم شیفت به راست انجام دهیم این کار حتماً باید با حفظ علامت صورت گیرد .

حالت سوم : اگر x_n مثبت و y_n منفی باشد ($y_n = 0, x_n = 1$)

این حالت را با یک مثال دنبال می کنیم.

مثال : حاصلضرب دو عدد 5 , 3 – را به روش دستی نمایش دهید؟

حل : چون دو عدد چهار بیتی هستند ، به فضای ۸ بیتی برای نمایش حاصلضرب نیاز داریم .

۰۱۰۱ 5

۱۱۰۱ × -3 ×

-15

۰۰۰۰۰۱۰۱

۰۰۰۰۰۰۰۰

۰۰۰۱۰۱۰۰

۰۰۱۰۱۰۰۰ +

۰۱۰۰۰۰۰۱

مشاهده می شود که حاصل عملیات نادرست می باشد . برای رفع این مشکل باید به ازای آخرین بیت از عدد دوم ، در صورت یک بودن به جای عمل جمع ، عمل تفریق را انجام دهیم .

۱۱۰۱

۱۱۰۱ ×

۰۰۰۰۰۱۰۱

۰۰۰۰۰۰۰۰

۰۰۰۱۰۱۰۰ +

۰۰۰۱۱۰۰۱

۰۰۱۰۱۰۰۰ −

۱۱۱۱۰۰۰۱

$(11110001)_2 = -(00001111)_2 = -15$

حال می خواهیم این عملیات را درکامپیوتر انجام دهیم.

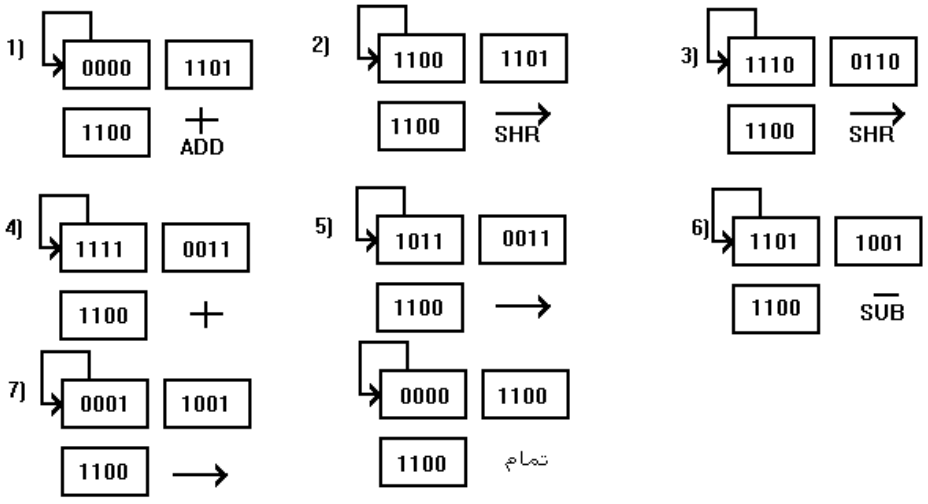
باید به ازای آخرین بیت از ثبات B در صورت یک بودن به جای عمل جمع ، عمل تفریق را انجام دهیم ونیازی به تغییرات اساسی نیست .

حالت چهارم : حالتی که x_n و y_n هر دو منفی باشند. ($x_n = 1, y_n = 1$)

حالت چهارم ترکیبی از موارد دوم وسوم است . با توجه به بحثهای انجام شده می توان اینطور نتیجه گرفت که برای ضرب دو عدد متمم ۲، بایستی بدین صورت عمل کنیم .

همواره به سمت راست ترین بیت ثبات B نگاه می کنیم . اگر این بیت یک باشد محتوای ثبات A را به ثبات C اضافه می کنیم وسپس زوج ثبات B و C را به صورت حسابی (بصورت سخت افزاری) به راست انتقال می دهیم وبه ازاء بیت صفر ، فقط زوج ثبات B و C را به صورت حسابی به سمت راست شیفت می دهیم . به ازای تمامی بیتهای ثبات B این کار را انجام می دهیم . چنانچه آخرین بیت از ثبات B یک باشد به جای عمل جمع ، عمل تفریق را به کار می بریم . (در هر لحظه اگر در هنگام انتقال به راست سرریز رخ دهد یک از سمت چپ ثبات C وارد می شود.)

مثال : می خواهیم دو عدد 3- , 4- را به صورت 2's توسط مدار ضرب کننده ٤ بیتی با هم ضرب نماییم.



نکته : در مرحله آخر ، چون LSB برابر ۱ است پس عمل تفریق را انجام می دهیم وبرای این کار از عدد واقع درثبات زیرین، 2's می گیریم وبه جای تفریق،عمل جمع را انجام می دهیم .

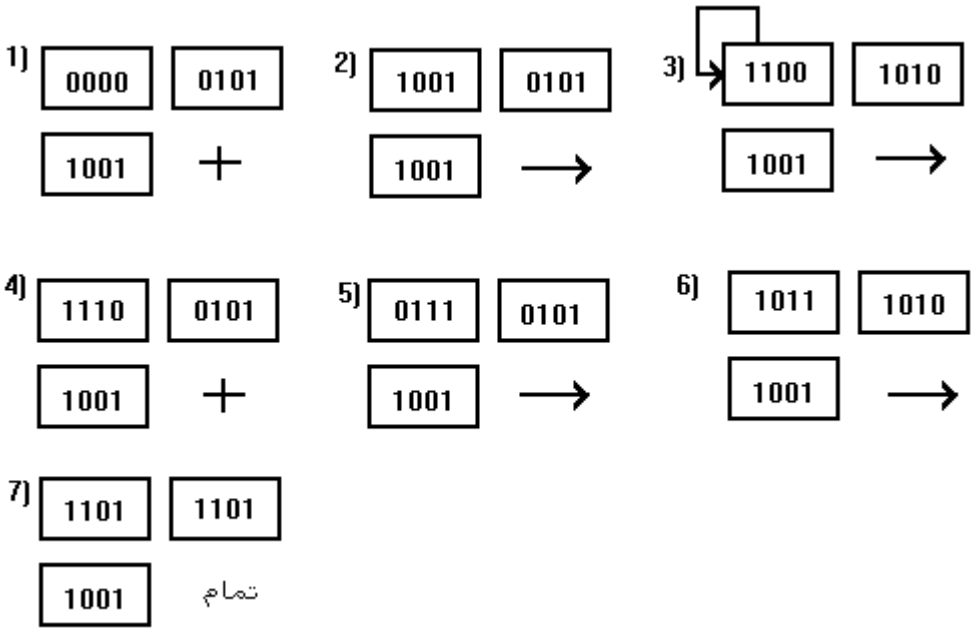
نکته : در متمم 2 بروز رقم نقلی هیچ گونه اهمیتی ندارد .

مثال : دو عدد A , B را در 4 بیت ضرب نمایید؟

-7 A ×

+5 B

-35



در این مثال در مرحله ی ۵ سرریز رخ داده است بنابراین هنگام انتقال به سمت راست از سمت چپ ثبات C عدد یک وارد می شود .

نکته : اگر در جمع دو عدد مثبت ، حاصل عملیات منفی شود ، در این حالت نیز سرریز رخ داده است که در اینصورت از سمت چپ ثبات C مقدار صفر وارد می شود .

۳-۳ : روشهای سریع ضرب

هر چه تعداد بیت های صفر زیاد باشد محاسبه در روشهای قبل آسانتر است . درپیاده سازی ضرب به روش سریع ، به ازای اولین بیت یک ، عمل تفریق وبه ازای اولین بیت صفر بعد از مجموعه‌ایی از یکها ، عمل جمع را انجام می دهیم وبه راحتی از روی بقیه یک ها عبور می کنیم .

01111 --------> 10000 - 00001

علت این کار، این است که در روشهای قبلی به ازای هر بیت ، باید هم عمل جمع وهم عمل شیفت را انجام دهیم . ولی در این روش ما فقط یک بار عمل جمع و یک بار عمل تفریق راانجام می‌دهیم وبه ازای بیتهای یک فقط عمل شیفت به راست را انجام می دهیم. بنابراین عمل ضرب سریعتر از روشهای قبل انجام می شود .

۳-۳-۱: روش ضرب Booth

برای تشخیص دنباله‌ای از یکها باید ابتدا همه بیتها را چک کنیم تا دنباله‌ای از یکها را مشخص نماییم این عمل ، عمل پیچیده‌ای است . لذا شخصی بنام Booth الگوریتمی سخت افزاری را ارائه نمود که به راحتی می توانست این مشکل را حل نماید.

در این روش در هر لحظه به بیت جاری (بیت با ارزش کمتر) و بیت ماقبل آن توجه می شود .این دو بیت ٤ وضعیت نسبت به یکدیگر دارند که در برخورد با هر یک مطابق الگوی زیر عمل می شود :

	بیت ماقبل	بیت جاری
۱- در حالت اول انتقال به راست	۰	۰
۲- در حالت دوم عمل تفریق		۰
۱		

۳- در حالت سوم عمل جمع ۱

۰

٤- در حالت چهارم انتقال به راست ۱

۱

چون در هر لحظه به بیت ماقبل نیاز مـی باشـد ، پـس یـک F/F داریـم کـه بیـت ماقبـل را در آن نگهداری می کنیم. بیتی که شیفت به راست داده می شود به F/F منتقل می شود (از ثبات B) وهـر بار به بیت کم ارزش ثبات B (بیت جاری) ومحتوای F/F که بیانگر بیـت قبلـی اسـت نگـاه مـی کنیم.

نکته : در شروع کار محتوای F/F برابر با صفر می باشد .

مثال : فرض کنید که عدد $(00111100)_2$داخل ثباتهای A و B قرار داشته باشد . با استفاده از الگوریتم Booth آن را تحلیل کنید ؟

بیت جاری

F/F

در ابتدادردنباله‌ای‌ازصفرهاهستیم پس‌انتقال‌به‌راست‌انجام‌می‌شود. ۰ ۰

درآغازدنباله‌ایی از یکهاهستیم پس عمل تفریق انجام می شود. ۰ ۱

دردنباله‌ایی ازیکهاهستیم پس انتقال به راست انجام می شود. ۱ ۱

دردنباله‌ایی ازیکهاهستیم پس انتقال به راست انجام می شود. ۱ ۱

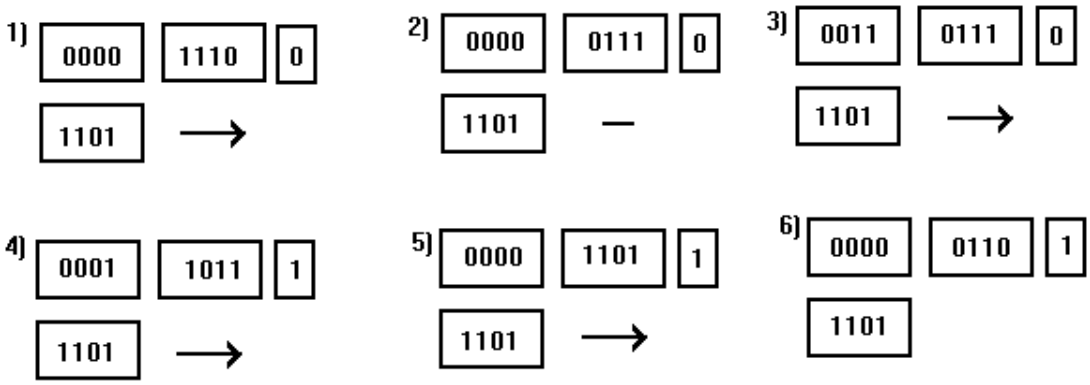
دردنباله‌ایی ازیکهاهستیم پس انتقال به راست انجام می شود. ۱ ۱

دنباله‌ایی از یکها پایان یافت پس عمل جمع انجام می شود. ۱ ۰

دردنباله‌ایی از صفرهاهستیم پس انتقال به راست انجام می شود. ۰ ۰

مثال : × 3- را به روش الگوریتم Booth انجام دهید:

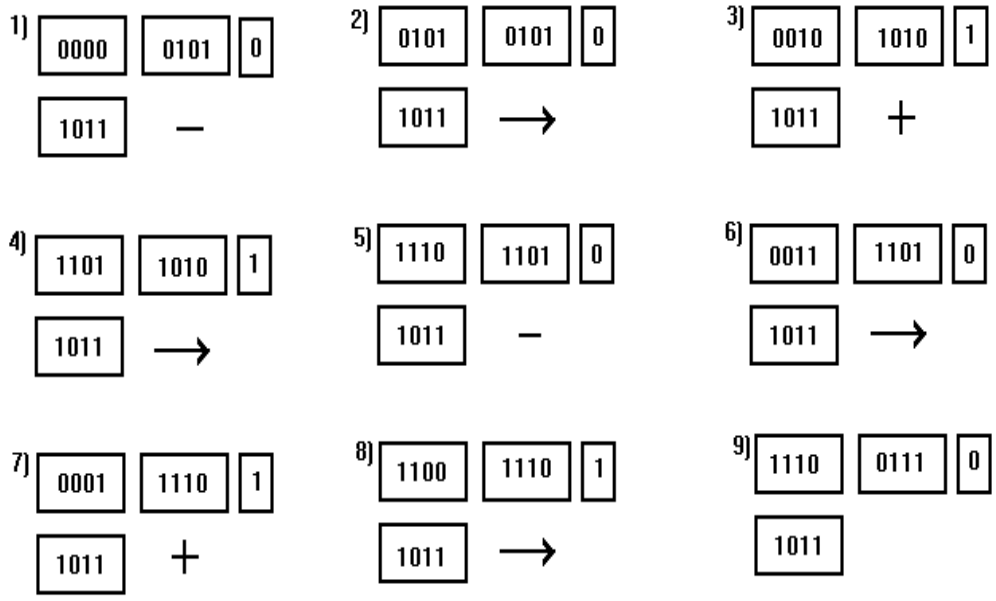
-2



در حل این مثال به روش الگوریتم Booth چون دو عدد ٤ بیتی است ، پس ٤ عمل Shift Right وفقط یک عمل تفریق صورت می گیرد ولی با روش قبلی ۲ بار عمل جمع، یکبار عمل تفریق وچهار مرتبه عمل ShR صورت می گیرد .

مثال : × 5- به روش Booth عمل کنید.

+ 5

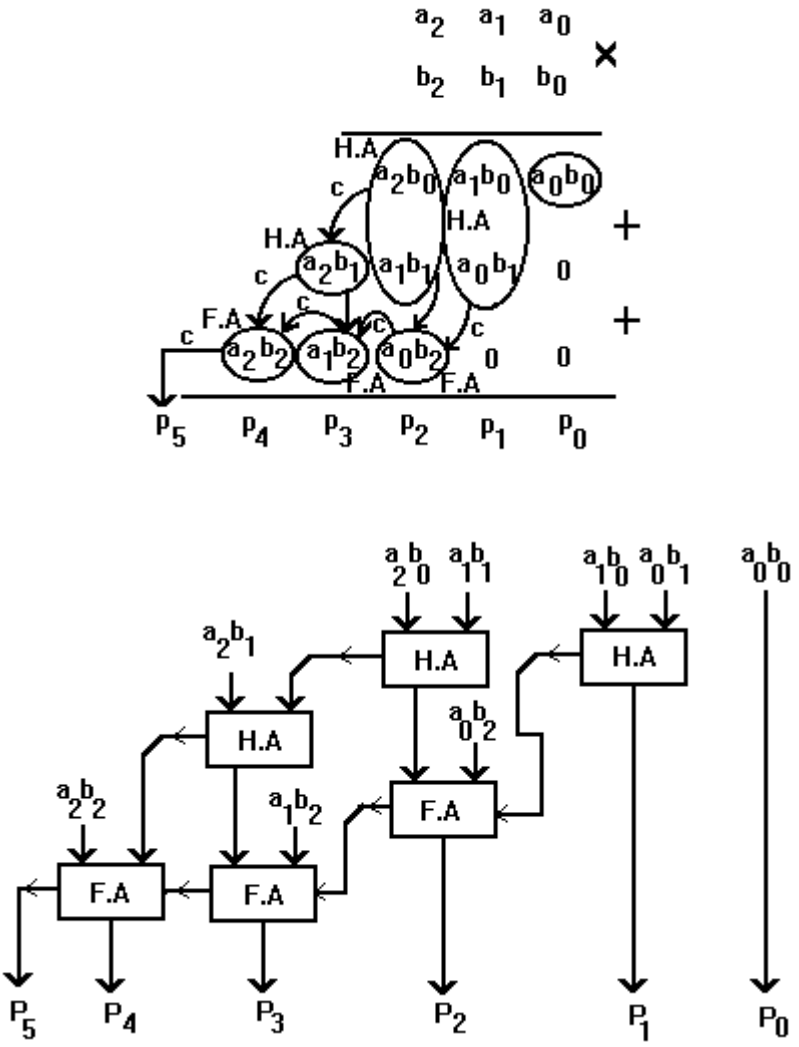


* عیب این روش این است که اگر یک ها وصفرها بصورت یک در میان در ثبات B قرار گیرند مانند مثال فوق بدترین حالت را داریم . یعنی می بایست به ازای یک ها عمل تفریق وبه ازاء صفرها عمل جمع را انجام دهیم .

* روشهای ضرب سریع دیگری نیز وجود دارد که سرعت انجام کار آنها بیشتر است وبرای مثال اگر n صفر متوالی مشاهد ه شود باید آناً n شیفت صورت گیرد ، پس این روش سرعت بیشتری دارد .برای پیاده‌سازی این مدارات ضرب کننده نیاز به واحد کنترل می باشد که این از نظر هزینه به صرفه نیست.

۳-۳-۲ : ضرب به کمک مدارات ترکیبی (ضرب آرایه‌ای)

فرض کنید که می خواهیم دو عد ۳ بیتی را در هم ضرب کنیم بنابراین داریم :



(شکل ۳-۲)

نکته : بطور کلی اگر بخواهیم در ضرب آرایه‌ایی دو عد n بیتی را در هم ضرب کنیم ، به n عدد H.A و $n^2 - 2n$ عدد F.A نیاز خواهیم داشت .

تمرینات

۱- دو عدد ۱۳- و۱۵ را به روش قدر مطلق علامت در هم ضرب نمایید وکلیه مراحل رانشان دهید؟

۲- دو عدد ۱۳- و ۵- را به روش متمم ۲ را در هم ضرب نمایید وکلیه مراحل را نشان دهید؟

۳- دو عدد ۷- و ۳ را در ٤ بیت به فرم متمم ۲ نمایش دهید وسپس براساس روش Booth با هم ضرب نمایید . کلیه مراحل را طی عمل ضرب نشان دهید؟

٤- در نظر بگیرید که دو عدد A = +٦ و B = -۷ را بخواهیم به روش Booth را در هم ضرب نماییم ، در هر یک از ۲ مرحله زیر فقط بنویسید که چه تعداد عمل جمع وچه تعداد عمل تفریق لازم است ؟ (ثباتها را ٤ بیتی در نظر بگیرید)

الف) $A \times B$ (A ضرب شونده و B ضرب کننده می باشد.)

ب) $B \times A$ (B ضرب شونده و A ضرب کننده می باشد.)

۵- یک ضرب کنندی آرایه‌ای طراحی کنید که دو عدد ٤ بیتی را ضرب کند . از گیتهای AND و جمع کننده های دو دویی استفاده کنید ؟

٦- الگوریتمی را به شکل فلوچارت برای عمل ضرب به روش Booth رسم نمایید؟

۷- درستی یا نادرستی عبارات زیر را مشخص نمایید؟

- پیچیدگی ضرب دو عدد ، به انتخاب کد جهت نمایش آنها بستگی دارد.

- در یک کامپیوتر که کلیه ی ثباتهای آن n بیتی هستند ، در محاسبه ی دو عبارت زیر نمی توان گفت که سرعت محاسبه ی عبارت دوم بیشتر از سرعت محاسبه ی عبارت اول است.

۱) $M \times K$ ۲) $N \times K$ $(N < M)$

- در کلیه روشهای ضرب ، عمل ضرب n ، Shift وحاصل ۲n بیت خواهد شد.

- در ضرب دو عدد متمم ۲ ، علامت دو عدد نیز مانند سایر بیتها پردازش می شود.

٤- مدارات تقسیم کننده

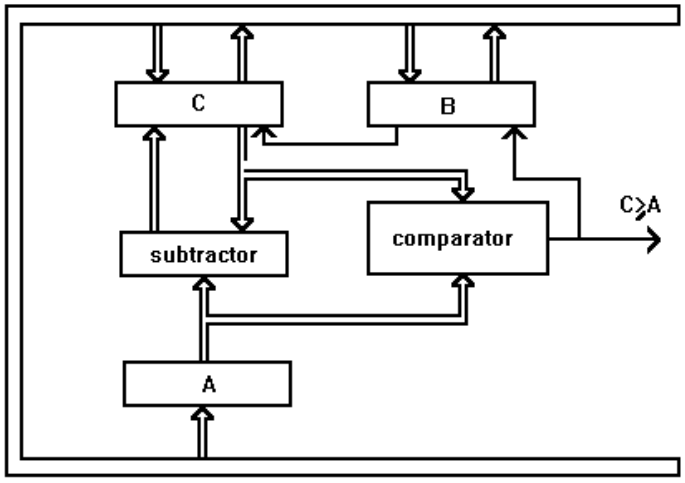
عمل تقسیم هم مانند عمل ضرب الهام گرفته از روش تقسیم دستی است .در عمل تقسیم مقسوم ازلحاظ تعدادارقام معمولاً ازمقسوم علیه بزرگتراست.برای‌کم کردن هزینه ها بهتر است مدار بصورت یکثارچه باشد . بطوریکه اگرمقسوم علیه ما n بیتی باشد ، باید مقسوم ما در ثبات 2n بیتی ذخیره گردد . پس تا اینجا به ۳ ثبات n بیتی نیازمندیم .همچنین ثبات ما باید قابلیت شیفت به چپ (ShL) را داشته باشد.چون ما خودمان در عمل تقسیم دستی هرلحظه یک بیت به جلو می رویم ولی در ثبات باید یک شیفت به سمت چپ داشته باشیم ونیز به یک تفریق کننده در مدار نیاز داریم تا عمل تفریق را انجام دهد. همچنین قبل از عمل تقسیم باید یک مقایسه کننده وجود داشته‌باشدتا مشخص نماید که مقسوم‌کوچکترازمقسوم‌علیه‌است یا خیر.

٤-١ : تقسیم به روش مقایسه‌ای

برای عمل تقسیم به روش مقایسه‌ای به موارد زیر نیازمندیم :

١- به ٣ ثبات n بیتی ٢- ثباتها قابلیت شیفت به چپ داشته باشند.

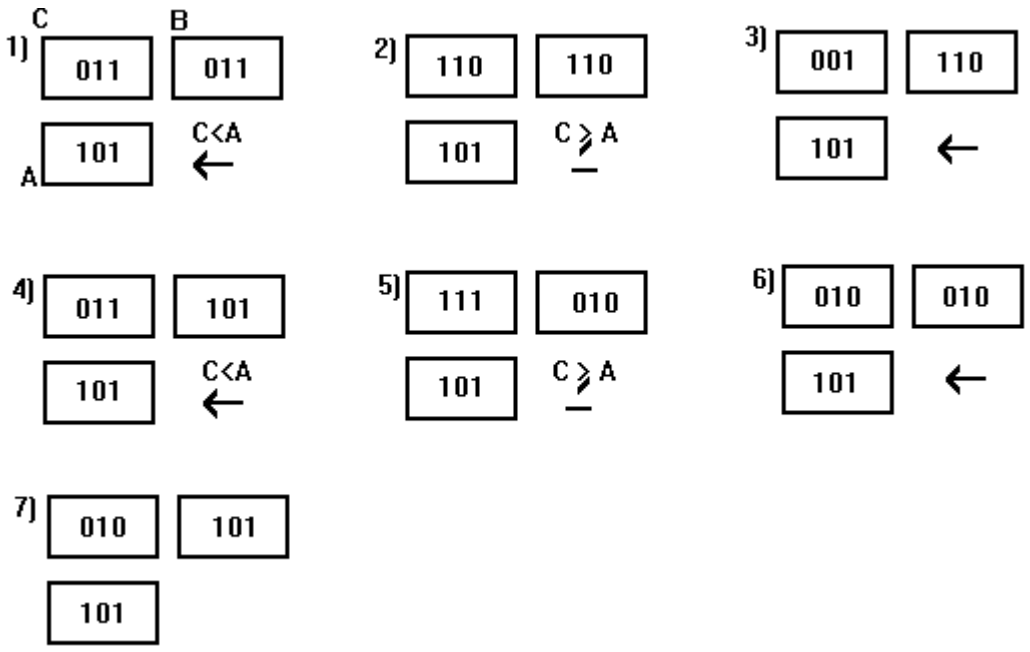
٣- مدار تفریق کننده ٤- مدار مقایسه کننده



(شکل ٤-١)

در این روش مقسوم در زوج ثبات Cو B قرار می گیرد ومقسوم علیه در ثبات A قرار دارد و عمل تفریق از سمت ثبات C صورت می گیرد وبعد از هر بار عمل تفریق نیز یک شیفت به چپ صورت می گیرد . البته قبل از عمل تفریق نیز واحد کنترل از طریق مقایسه گرمی فهمد که عدد کوچکتروقابل تقسیم است ، یا نه ویک پیغام به واحد کنترل می دهد و واحد کنترل دستور عمل تفریق را صادر می کند وخارج قسمت تقسیم در ثبات B و باقی مانده در ثبات C قرارمی گیرد.

مثال : عمل تقسیم ٢٧÷٥ را انجام دهید ؟ (ثباتها همگی ٣ بیتی هستند واعداد نیز بدون علامت هستند.)



نکته : پس در تقسیم تعداد ShL برابر $n+1$ می باشد که n تعداد بیتها است وبه ازاء آخرین ShL فقط ثبات B ، ShL می گردد وثبات C را ShL نمی دهیم .

نکته :اگر در عمل تقسیم اولین مقایسه ما موفقیت آمیز باشد (یعنی $C \geq A$ باشد) نتیجه ی حاصل حتماً سرریز خواهد بود . چون همیشه از تعداد بیتهای ثبات که n می باشد ، عمل ShL ما یکی بیشتر است ، یعنی n+1 ، ShL صورت می گیرد . پس سرریز رخ می دهد. در چنین مواقعی راه حلی نیست زیرا جواب ما از ثبات ما یک بیت بیشتر است و ثبات نمیتواند آنرا نشان دهد.

نکته : به علت وجودمقایسه گر هزینه مدار زیاد می شود ومعمولاً در مدارات کامپیوتر از این شیوه استفاده نمی شود . اگر n زیاد باشد هزینه مقایسه گر از هزینه کل مدار بیشتر است.

نکته : برای این که مقایسه کننده را از مدار خارج کنیم ، از روش دیگری استفاده می کنیم که دو عدد را از هم کم کرده وبا وجود رقم نقلی ،بزرگتر ، کوچکتر ومساوی را تشخیص می دهیم ودر واقع از همان تفریق کننده برای عمل مقایسه استفاده می کنیم.

نکته : در کامپیوترها برای اینکه هزینه ی کمتری صرف شود ، باید ثباتها قابلیت ShR و ShL راداشته باشند وعمل جمع وتفریق توسط یک مدار که هم جمع کننده وهم تفریق کننده است

صورت گیرد . در نتیجه عمل ضرب و تقسیم توسط یک مدار صورت گیرد ودر هنگام ضرب ،باید جمع کننده وShR ودر عمل تقسیم باید ShL وتفریق کننده داشته باشیم.

۲-٤ : تقسیم به روش Restoring

دراین روش مقایسه گر را از مدار قبلی برداشته وبرای مقایسه از تفریق کننده استفاده می نماییم وبدین صورت عمل می کنیم که هر بار جهت مقایسه A - C می گردد و حاصل به داخل ثبات C انتقال داده می شود ودو وضعیت پیش می آید.

الف – حاصل مقداری غیر منفی یعنی بزرگتر یا مساوی صفر باشد .

ب– حاصل مقداری منفی باشد یعنی کوچکتر از صفر باشد .

الف : حاصل مقداری غیرمنفی (یعنی $C \geq A$) باشد .

لذا می بایست طبق روش گذشته عمل A - C در ثبات C انجام گیرد که در اینجا انجام شد . پس از این مرحله کافی است که فقط یک عمل انتقال به چپ (ShL) داشته باشیم ویک را در حین انتقال از سمت راست ، به ثبات B وارد کنیم.

ب: حاصل مقداری منفی (C < A) باشد.

اگر عملیات منفی باشد ، نشان دهندی آن است که C < A بود ه لذا نمی بایست عمل تفریق را انجام دهیم . در نتیجه لازم است مقدار قبلی ثبات C بازیابی (Restore) شود . برای انجام اینکار لازم است محتوای ثبات A به ثبات C اضافه گردد . سپس یک واحد ، زوج ثبات B و C را به سمت چپ انتقال می دهیم وصفر را از سمت راست ثبات B وارد می کنیم.

نکته: در این روش هزینه کمتر ومدار ساده تربوده اما سرعت آن از روش قبلی کمتر است . یعنی بار محاسباتی بیشتری دارد وآن هم به خاطر عمل Restore می باشد.

نکته : در این روش در همه ی مراحل یک تفریق داشته ودر بعضی مراحل یک جمع داریم پس سرعت تقریباً $\frac{1}{2}$ برابر سرعت روش قبلی می شود .

۳-٤ : تقسیم به روش Non Restoring

وقتی عمل C - A را انجام دهیم حاصل به ثبات C می رود، در این حالت اگر حاصل مثبت باشد نیاز به عمل Restore نداریم ولی وقتی حاصل در ثبات C منفی شود ،نیاز به عمل Restoring داریم .

.حال این روش را بررسی می کنیم.

عمل Restoring وقتی منفی شود C-A +A

سپس یک انتقال‌به‌چپ بایدبدهیم‌که معادل‌دوبرابرکردن‌حاصل‌عملیات‌می‌باشد.

(C-A +A)*2

ودر مرحله ی بعد نیزباید مقدارA از این عبارت کم کنیم .

(C-A+A)*2-A

تا برای مرحله بعد عمل $C \geq A$ را بررسی نماییم.

----> (C-A+A)*2-A = 2C-A

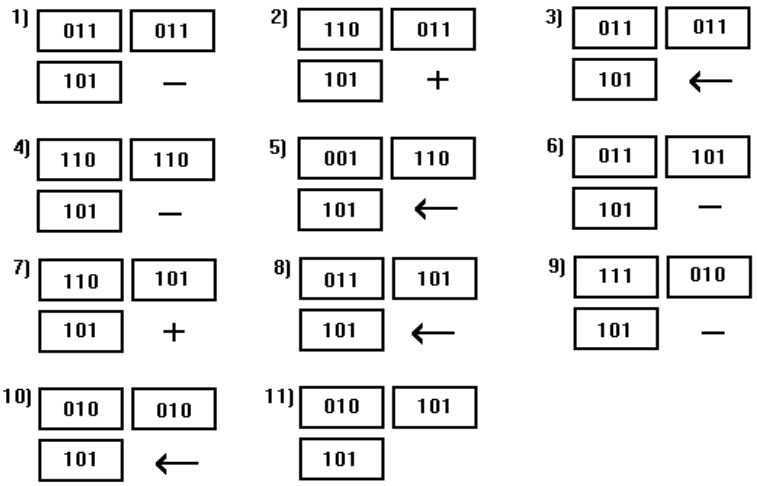
بجای اعمال قبلی که در این روش گفته شده ، روش معادل زیر را انجام دهیم .

(C-A)*2+A=2C-A

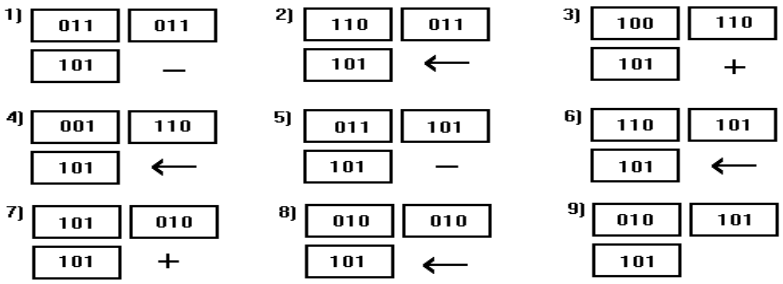
یعنی اگر حاصل منفی بود ShL داده و در مرحلی بعد عمل جمع A صورت می گیرد .پس دو رابطه ۱ و ۲ معادل هم می باشند .

مثال :تقسیم ۵÷ ۲۷ را به روش Restoring و Non - Restoring انجام دهید .

روش Restoring :



روش Non - Restoring:



تمرینات

۱- دو عدد ۱۷ و ۵ را در چهار بیت به فرم بدون علامت نمایش دهیدو سپس به کمک روش مقایسه‌ای بر هم تقسیم وکلیه مراحل را نشان دهید؟

۲- دو عدد ۱۳ و ۵ را به روش Restoring بر هم تقسیم نمایید ؟

۳- دو عدد ۱۹ و ۵ را به روش Non Restoring بر هم تقسیم نمایید؟

٤- الگوریتمی را به شکل فلوچارت برای عمل تقسیم به روش Restoring دو دویی ممیز - ثابت به دست آورید؟

۵- الگوریتمی را به شکل فلوچارت برای عمل تقسیم به روش Non Restoring دودویی ممیز- ثابت به دست آورید؟

٦- آیا عبارت زیر درست است ؟

- در عمل تقسیم قبل از آنکه محاسبه خاتمه یابد، سر ریز قابل تشخیص بوده ونیاز به ادامی عملیات نیست.

۵- اعمال اصلی دراعداد ممیز شناور

۵-۱ : عمل جمع دراعداد ممیز شناور

دوعدد صحیح را به راحتی می توان با هم جمع کرد .ولی دو عدد با ممیز شناور را به سادگی نمیتوان یا هم جمع نمود ومکان قرار گرفتن اعداد مهم است .

مثال:

102
+ 7 =109

$102 \times 10^{2} + 7 \times 10^{-3}$ ---->10200/000

00000/007 +

10200/007

بصورت کامپیوتری :

در کامپیوتر Base = 2 در نظر گرفته می شود ومی بایست توان دو عدد ممیز شناور با هم برابر شوند برای این منظور دوراه موجود است .

یک راه این است که توان عدد بزرگتر را به عدد کوچکتر تبدیل نمود .

مثال : $N_2 = 0.11 \times 10^1, N_1 = 0.101 \times 2^3$ M E

	E	M
N_1	۰۱۱	۱۰۱
N_2	۰۰۱	۱۱

حال برای اینکه از توان عدداول دو واحد کم کنیم تا برابر باتوان عدد دوم یعنی یک شود ، کافیست مانتیس عدد اول را دو بار به سمت چپ شیفت دهیم ، مشاهده می شود که انجام این عملیات باعث تغییر در مقدار عدد اول می شود پس این روش نادرست است .راه حل منطقی این است که توان عدد کوچکتررا به توان عدد بزرگتر تبدیل کنیم .

N_1	۰۱۱	۱۰۱۰
N_2	۰۱۱	۰۰۱۱
		+
$N_1 + N_2$	۰۱۱	۱۱۰۱

در این روش به ازای هر بار افزایش توان یک بار محتوای مانتیس را به سمت راست شیفت می دهیم . در واقع به اندازه اختلاف دو توان اعدادی که می خواهند با هم جمع شوند ، مانتیس عدد کوچکتر ShR می دهیم وبه همین اندازه به توان اضافه می کنیم .

۱۱۱۱۱۱۱۱۱	۰۰۱

نکته : اگر عدد ما به صورت زیر باشد وما بخواهیم برای رسیدن به توان عدد دیگر مانتیس آن را به سمت راست شیفت بدهیم ، دراینصورت دومقدار یک از سمت راست حذف می گردد در واقع دقت عمل کم می شود وهیچ راهی برای جلوگیری از این کارنیست.

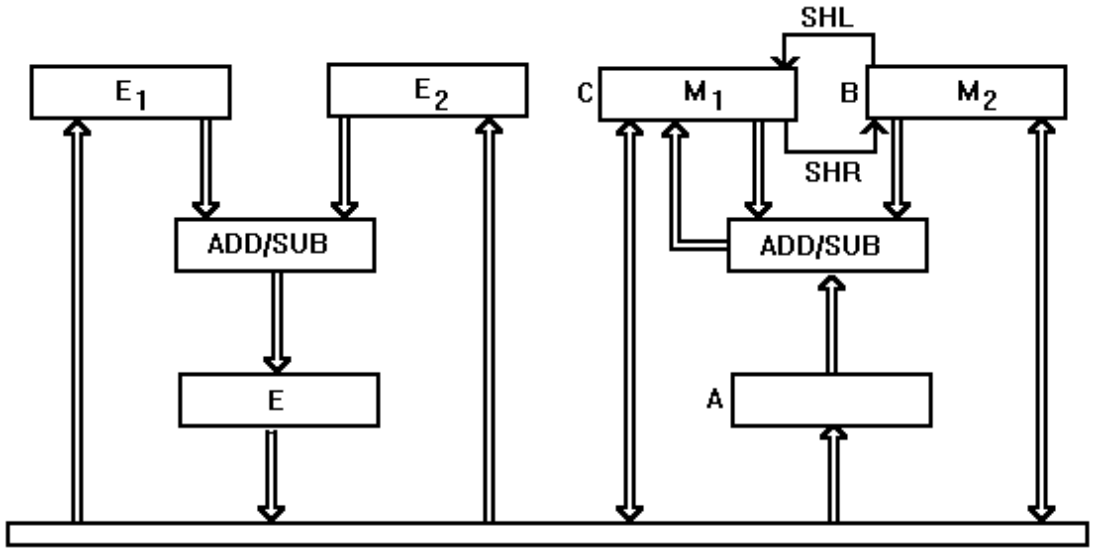
۵-۲ : عمل ضرب در عدد ممیز شناور

از لحاظ الگوریتمی ، الگوریتم ضرب آسانتر از جمع دو عدد ممیز شناور است .

مثال : $102 * 10^{2} * 7 * 10^{-3} = 714 * 10^{-1}$

در ضرب دوعدد ممیز شناور کافیست مانتیسها را در هم ضرب نموده وتوانها را باهم جمع کنیم.

۵-۳ : مدار جمع وتفریق دو عدد ممیز شناور



(شکل ۵-۱)

در ابتدا باید ببینیم که کدامیک از مفسرها کوچکتر است تا آنرا به مفسر بزرگتر تبدیل کنیم . این کار را باتفریق دو مفسر انجام می دهیم ($E_1 - E_2$) واگر حاصل تفریق منفی بود، یعنی E_1 کوچکتر است .اگر حاصل مثبت بود E_2 کوچکتر است . حال حاصل $E_1 - E_2$ در E قرار می گیرد .

به اندازه قدر مطلق E (یعنی | E |) مانتیس عدد کوچکتر را به سمت راست شیفت می دهد . سپس دو عدد M_2, M_1 نقاط اعشارشان زیرهم قرارمی گیرد و برای عمل جمع یا تفریق آماده می باشد . البته توان عدد بزرگتر به منظور توان نهایی حاصل جمع یا تفریق در قسمت توان حاصل عملیات قرار می گیرد .

نکته : اگر در قسمت توانها عمل جمع یا تفریق را انجام دهیم وبرای قسمت مانتیسها ، بجای عمل جمع وتفریق عمل ضرب وتقسیم صورت گیرد ، این مدار می تواند اعمال جمع وتفریق وضرب

وتقسیم دو عدد ممیز شناور را انجام دهد. البته باید ثباتهای C و B را با قابلیت ShL , ShR داشته باشیم .

تمرینات

۱- توضیح دهید که برای جمع بستن دوعدد Floating Point چه اقداماتی لازم است ؟

۲- توضیح دهید که برای تفریق دوعدد Floating Point چه اقداماتی لازم است ؟

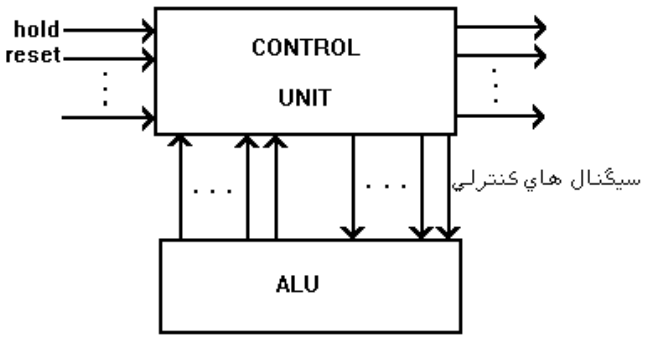
۳- توضیح دهید که برای ضرب دوعدد Floating Point چه اقداماتی لازم است ؟

٤- توضیح دهید که برای تقسیم دوعدد Floating Point چه اقداماتی لازم است ؟

فصل سوم - طراحی واحد کنترل وحافظه

۱- طراحی واحد کنترل

وظیفه واحد کنترل دریافت دستورات ازحافظه وتفسیر آنها وانجام دادن کارهای لازم (فرمانهای لازم) برای اجرای آن دستورات می باشد .



(شکل ۱-۱)

سیگنالهای کنترلی انواع مختلفی دارند :

۱- سیگنالهای کنترلی که از واحد کنترل به Alu می رود .

برای مثال ممکن است دستور ShR در یک ثبات و یا دستور خواندن اطلاعات از Bus وهزاران دستور دیگر باشد.

۲- سیگنالهای کنترلی که از طرف Alu به واحد کنترل می آید.

برای مثال ممکن است در جمع دو عدد رقم نقلی رخ دهد که دراین صورت واحد Alu به واحد CU (Control Unit) سیگنال می فرستد.

۳- سیگنالهای کنترلی که از یک ریزپردازنده در کامپیوتر دیگر به واحد کنترل می آیند .

برای مثال عمل Resert کردن یا عمل Hold کردن .

٤- سیگنالهای کنترلی که ممکن از واحد کنترل به یک جای دیگر غیر از Alu فرستاده شود .

برای مثال به یک ریز پردازنده دیگر بروند.

البته سیگنالهای کنترلی دیگری نیز وجود دارند که هر کدام وظیفه مخصوص به خود دارند.

نکته : در این درس هدف مطالعه سیگنالهای کنترلی است که بین Aluو CU وجود دارد ودر مورد بقیه سیگنالها بحث نمی شود .

بطور کلی طراحی واحد کنترل به دو طریق انجام می گیرد.

۱- روش سیم بندی شده (Hard Wired)

۲- روش ریزبرنامه سازی (Micro Programmed)

۱-۱ : روش سیم بندی شده

در این حالت واحدکنترل همانند یک مدار ترتیبی است که سیگنالهای کنترلی را یکی پس از دیگری تولید می کند . این روش ، سریعترین روش و کم هزینه ترین روش است اما هیچگونه انعطاف پذیری ندارد . یعنی اگر بخواهیم تغییرات جزیی در مدار واحد کنترل دهیم ، لازم است که طراحی مورد نظر از نو انجام شود .

امنیت روش H.W در مقابل Noise کم است چراکه تعداد زیاد سیم در فرکانس های بالا ایجاد Noise می کند.

به سه طریق می توان واحد کنترل رابراساس روش سیم بندی شده طراحی کرد .

۱- State Table Methed

۲- Delay Element Method

۳- Sequence Counter Method

State Table Method : ۱-۱-۱

در این روش از طراحی واحد کنترل بحث بر سر آن است که طراحی مدار بگونه‌ایی باشد که حداقل فیلیپ فلاپها راداشته باشد . به عبارت دیگر این روش از طراحی واحد کنترل نسبت به دو روش دیگر دارای هزینه کمتر وامنیت بیشتری در مقابل Noise می باشد .

البته طراحی واحد کنترل براساس این روش مشکل است و نیاز به بحث مدارات منطقی پیشرفته می باشد .از این روش هنگامی استفاده می شود که واحد کنترل کوچک باشد .

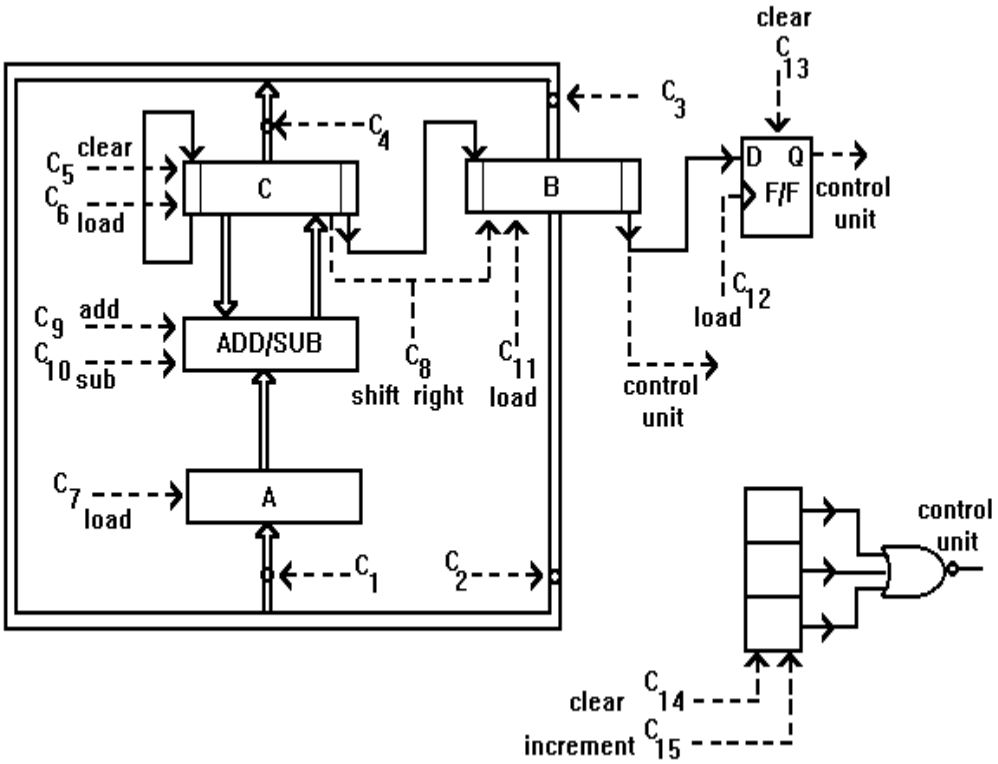
Delay Element Method : ۲-۱-۱

برای آشنایی بیشتر با این روش از طراحی مثال زیر را بررسی می نماییم .

مثال : طراحی واحد کنترل برای ضرب دو عدد به روش Booth

برای طراحی واحد کنترل ، ابتدا مدار را طراحی می کنیم . سپس سیگنالهای کنترلی (خطوط نقطه چین) را مشخص نموده و براساس الگوریتمی پیاده سازی می نماییم . در این

الگوریتم A با B ضرب شده و حاصل در زوج ثبات C : B قرار می گیرد . چون ثبات های A و B هشت بیتی هستند ، برای واحد کنترل به شمارنده ای با ۳ بیت نیاز داریم .



(شکل ۱-۲)

الگوریتم ضرب به روش Booth

۱- اطلاعات موجود در گذرگاه را در ورودی ثبات A آماده می نماییم.

۲- مضروب را وارد ثبات A ، فیلیپ فلاپ و ثبات C را Clear می نماییم.

۳- اطلاعات موجود در گذرگاه را در ورودی ثبات B آماده می کنیم.

٤- مضروب ما فیه را وارد ثبات B وشمارنده واحد کنترل را Clear می نماییم.

۵- به B(0) : FF توجه شود

اگر برابر 10 بودند آنگاه C - A -----> C

اگر برابر 01 بودند آنگاه C + A -----> C

٦- مجموعه C : B : FF را یک بیت به راست ، بطور حسابی تغییر مکان می دهیم و به شمارنده واحد کنترل یک واحد اضافه می کنیم.

٧- اگر $Count \leq 7$ بود ، ادامه کار از مرحله 5 انجام شود .

٨- پایان

* چون اطلاعات در حافظه است ، در یک مرحله زمانی باید اطلاعات را به داخل BUS آورده ودر مرحله دیگر اطلاعات را به داخل ثبات A بیاوریم و انجام این دو عمل بطور همزمان ممکن نیست. چون در این صورت اطلاعات نادرستی در A ذخیره خواهد شد .

* هر چه شماره ها (مراحل الگوریتم) بیشتر باشد زمان اجرایی بیشتر است ، چون هر شماره زمان خاص خود را دارد و هر چه مراحل کمتر باشد زمان کمتری مصرف می شود . پس مطلوبتر است .

* ثبات C وF/F می بایست Clear گردند ،که این اعمال می توانند هم زمان اجرا گردند و اگر آنها را در یک سطر دیگر جدای از مرحله دوم بنویسیم زمان بیشتری مصرف می شود . چون این اعمال به هم ربطی ندارند پس می توان آنها را در مرحله دوم قرار دهیم .

* در مرحله ٣ و ٤ نیز عدد مضروب فیه را به گذرگاه آورده و در مرحله ٤ وارد ثبات B می کنیم. چون ثباتهای ما٨ بیتی است، پس عملیات باید ٨ بار تکرار شود . پس شمارنده برای شمارش آن نیاز داریم . این شمارندی ٣ بیتی، که برای شمارش ٨ عدد است را در ابتدا باید Clear کنیم .

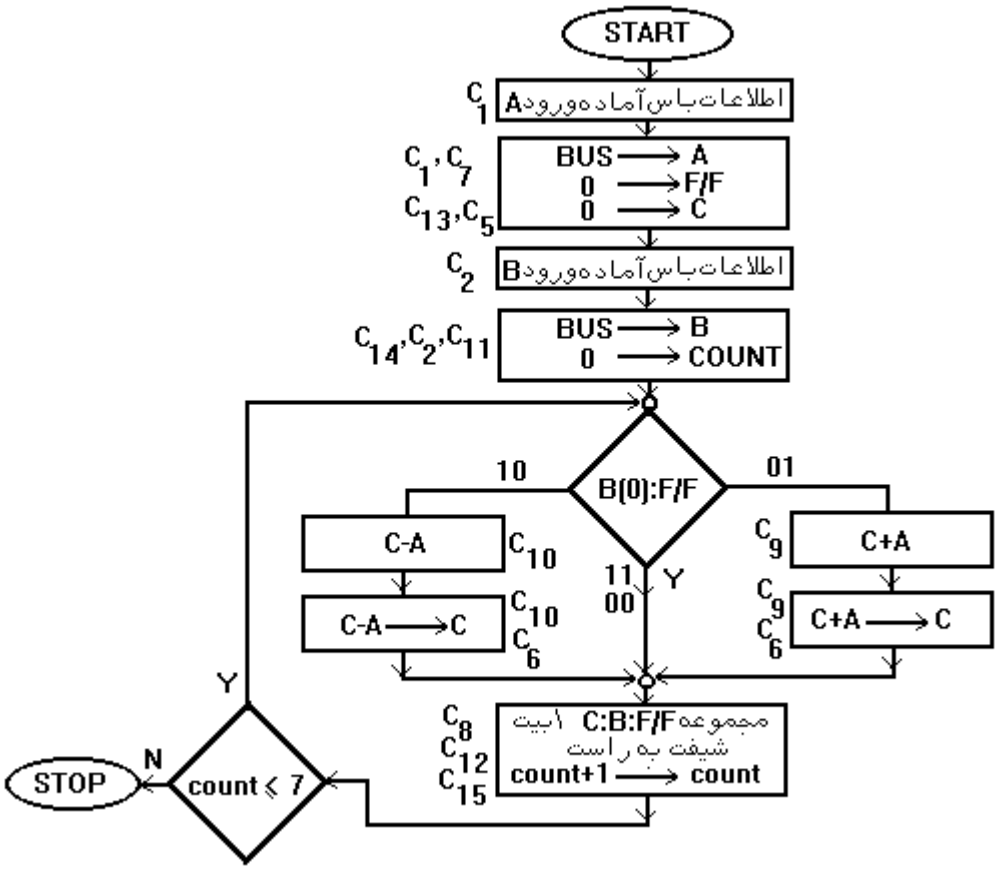
*در مرحله ٥ چون در واقع عمل مقایسه صورت می گیرد پس شماره به خود نمی گیرد .

* در مرحله ٦ عمل ShR بصورت حسابی وبه اندازه یک بیت صورت می گیرد وچون عمل اضافه کردن یک واحد به شمارنده ، به ShR ربطی ندارد . پس می تواند بصورتی اشتراکی با ShR در یک مرحله صورت گیرد.

* درمرحله ۷ اگر شمارنده از ۷ گذشته باشد ، به مرحله پایانی می رود و در غیر اینصورت از مرحله ۵ ، مراحل را ادامه می هد.

از این لحظه به بعد تصمیم می گیریم که به روش سیم بندی شده الگوریتم را طراحی کنیم یا روش ریز برنامه سازی، واگر از روش سیم بندی شده استفاده می نماییم کدامیک از سه روش موجود در این روش را انتخاب می کنیم.

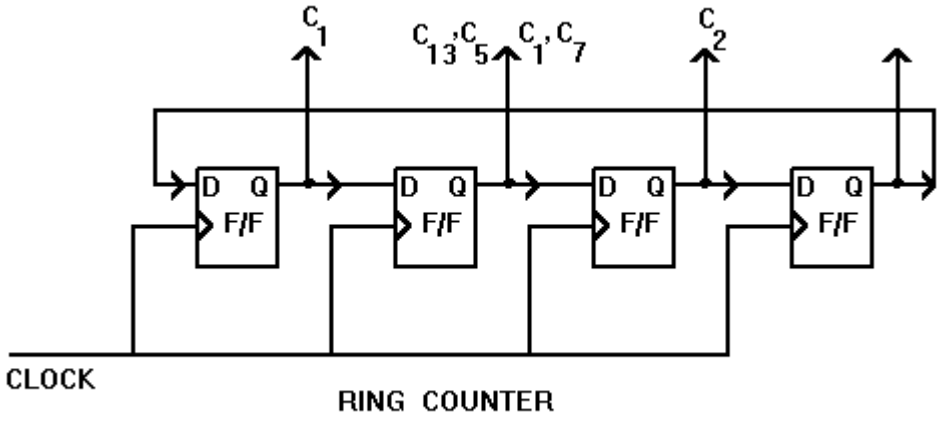
طراحی واحد کنترل الگوریتم Booth به روش Delay Element Method :



(شکل ۱-۳)

برای طی کردن یک بار مدار ، اگر از مسیر مستقیم برویم به ۵ پالس و در غییر این صورت به ۷ پالس زمانی احتیاج داریم . در کنار هر BOX نوشته شده که با کدامیک از سیگنالهای کنترلی فعال می شود . پس از این به بعد نوبت به عمل طراحی رسیده است . ما باید وسیله‌ایی داشته

باشیم که این سیگنالها ($C_1 \ldots\ldots C_{15}$) را به ترتیب و در هر لحظه یکی از آنها را فعال کند واگر این کار را انجام دهیم درواقع واحد کنترل را طراحی کرده ایم .برای اینکار ، تعدادی فیلیپ فلاپ رادرنظر می گیریم ، که همه آنها از نوع فیلیپ فلاپ D می باشند وClock آنها به هم متصل می باشد .



(شکل ۱-٤)

در لحظه اول همه فیلیپ فلاپها صفر هستند و در لحظه بعدی یک رابه فیلیپ فلاپ اول می دهیم . مشاهده می شود که یک ، بصورت چرخشی بین فیلیپ فلاپها به گردش در می آید درواقع این همان Ring Counter ، در درس مدارات منطقی می باشد .در لحظه اول باید سیگنال C_1 را فعال کند ودر لحظه دوم سیگنال C_1, C_7, C_{13}, C_5 که مربوط به BOX دوم درالگوریتم است فعال شود و به همین صورت الی آخر . علت اینکه این روش را Delay Element می نامند به خاطر وجود فیلیپ فلاپ می باشد که ایجاد تأخیر می نماید.

برای تبدیل فلورچات به مدار الکتریکی به این نکات توجه کنید :

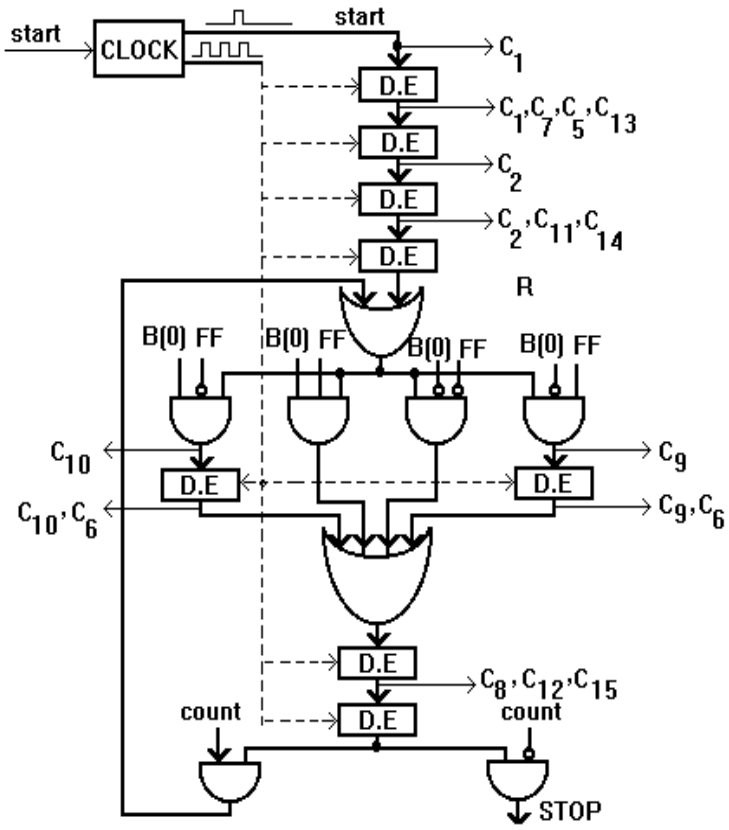
۱-بین هردو BOX یک Delay Element (D.E) قرارداده وبجای هر BOX یک سیم می گذاریم .

۲- هنگامیکه اطلاعات از چند مسیر وارد یک مسیر می شود باید گیت OR بکار ببریم.

۳- در نقاط تصمیم گیری باید گیت AND بگذاریم .

با رعایت این سه نکته می توان چارت را به صورت یک واحد کنترل پیاده سازی کرد .

* عمل مقایسه نیز نیاز به زمان (D. E) ندارد.

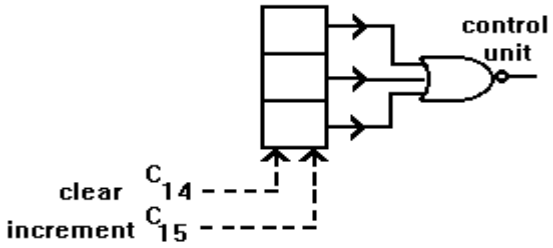


(شکل ۱-۵)

توضیح راجع به نحوه عملکرد شمارنده

در شروع کار پس از رسیدن به مرحله ۷ مقدار Count ، برابر یک می باشد. بنابراین خروجی OR گیت ما یک است، پس حلقه اجرا می شود و باعث قطع حلقه مانمی شود . وقتی که Count برابر ۸ شد ، خروجی گیت OR ما برابر با صفر شده بنابراین از حلقه خارج می شود .

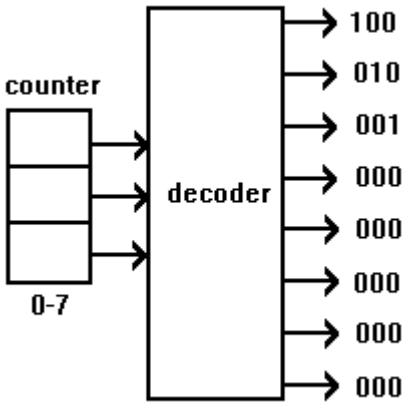
در اینجا نیازی نیست که خروجی های Count به یک NOR گیت برود ، زیرا در مطالب بعدی نیاز به گیت NOR می باشد . پس برای اصلاح عملیات کافی است وقتی که گیت OR را باگیت NOR عوض می کنیم ورودیهای Count در AND گیتهای پایین شکل (۱ -۵) را Not می کنیم تا جواب مطلوب حاصل شود ولی در این روش تعداد FF ها زیاد است .



(شکل ۱-٦)

Seqence Counter Method : ۳-۱-۱

در این روش سعی می شود که تعداد فیلیپ فلاپها (F/F) به نحوی کاسته شود، زیرا در روش Delay Element تعداد فیلیپ فلاپها (F/F) زیاد می باشد . در این روش از یک شمارنده معمولی و یک کدگشا (Decoder) بجای شمارنده حلقوی که در روش Delay Element بود ، استفاده می کنیم و در واقع شمارنده عمل شمارش وکد گشا نیز عمل کدگشایی را انجام می دهد. بنابراین هزینه خیلی پایین می آید . با استفاده از Decoder که دائما از ابتدا تا انتها تکرار می شود ، حلقه های مختلفی را طی می کنیم .



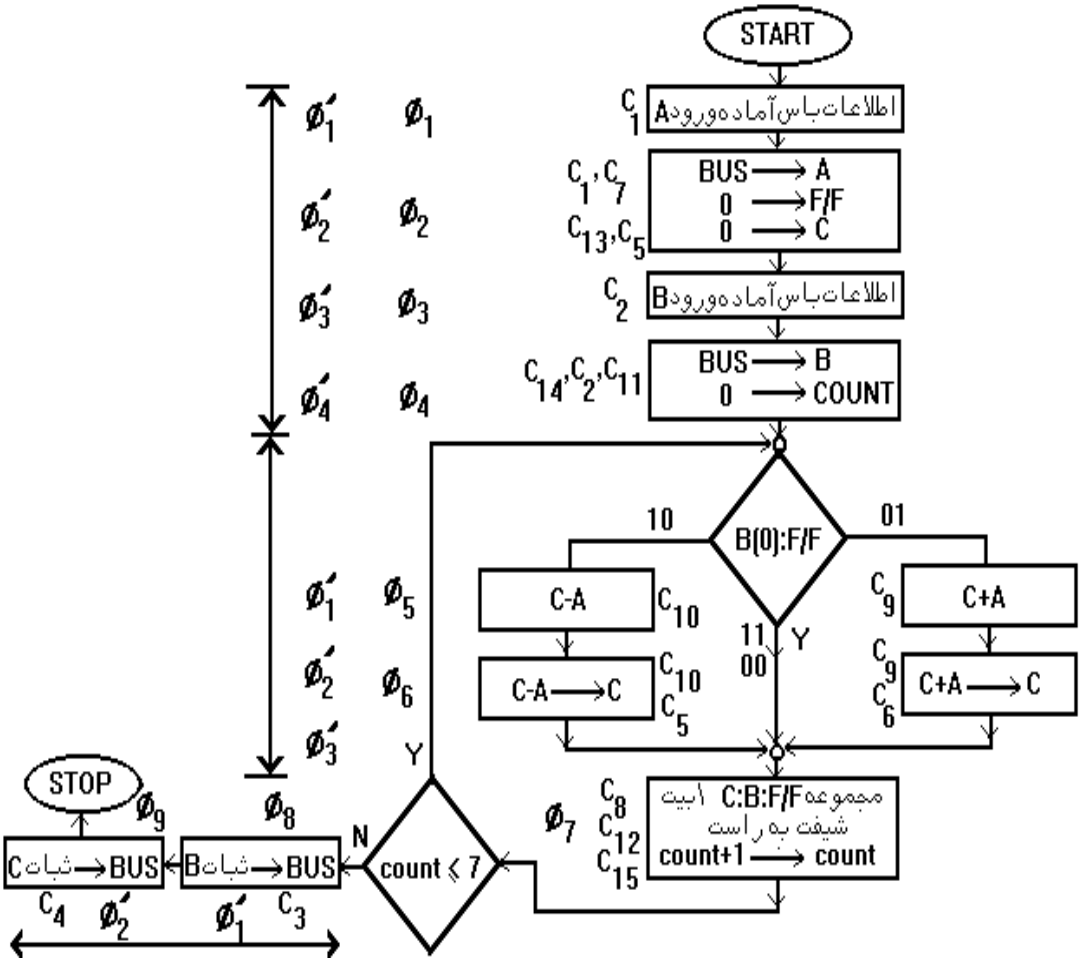
(شکل ۱-۷)

در الگوریتم قبلی یک عمل چندین بار تکرار می شد ، (در یک حلقه قرار می گرفت) واین عمل باعث پیچیدگی مدار در روش S.C (Sequence Counter) می شود وهزینه این مدار

نیز از روش D.E بیشتر می گردد . لذا از الگوریتمی استفاده می کنیم که فقط در دو مرحله با الگوریتم قبلی تفاوت دارد .

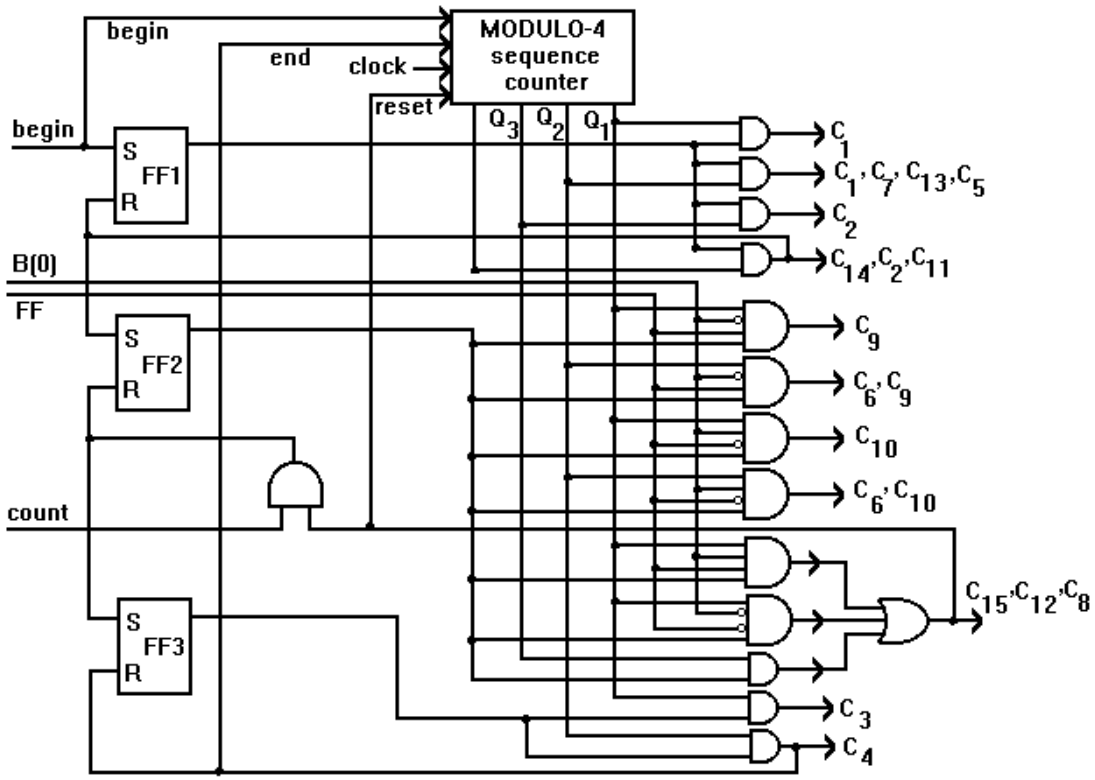
برای درک بهتر روش Sequence Counter مثال زیر را بررسی می نماییم.

مثال :طراحی واحد کنترل الگوریتم Booth به روش Sequence Counter Method



(شکل ۱-۸)

اگر بخواهیم از روش D.E استفاده کنیم باید یک شمارنده حلقوی ۹ بیتی داشته باشیم ولی در روش S.C باید الگوریتم به سه قسمت تقسیم شود . قسمت اول عملیات φ'_1 تا φ'_4 وقسمت وسط، از φ'_1 تا φ'_3 و قسمت آخر نیز از φ'_1 تا φ'_2 را می شمارد . در اینجا حداکثر باید تا ٤ شمارش شود زیرا مرحله دوم که بیشترین زیر مرحله را دارد ، شامل ٤ قسمت می باشد .



(شکل ۱-۹)

حال از یک شمارنده استفاده می کنیم که همه قسمت ها را بشمارد وبرای این شمارش ، ازشمارنده‌ایی به پیمانه چهار وسه فیلیپ فلاپ استفاده می کنیم.

برای اینکه وقتی شمارنده می شمارد متوجه شویم ، عمل شمارش برای کدام قسمت از فلورچات در حال انجام می باشد ، از سه فیلیپ فلاپ که هر کدام برای یک قسمت می باشد استفاده می کنیم و عمل شمارش برای هر قسمتی باشد فیلیپ فلاپ مربوط به آن Set (یعنی یک) شده وبقیه فیلیپ فلاپها صفر می شوند. هنگامی که C_4 فعال شد ، شمارش تمام است و این یعنی عمل ضرب به انتهای خودش رسیده است .

* وقتی Clock به شمارنده داده می شود قسمت Begin نیز که حالت شروع است ، فعال شده ویک Start از آن نیز به شمارنده می رود.

۱-۲ : روش ریز برنامه سازی

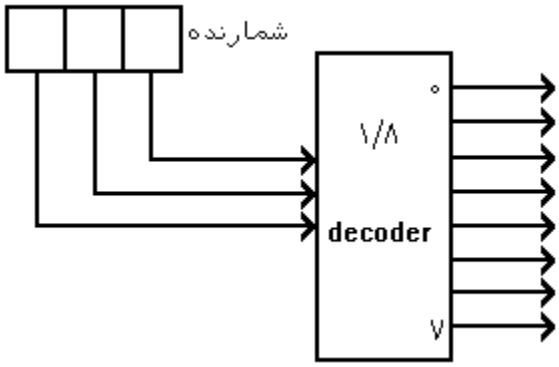
این روش که اولین بار در سال ۱۹۵۰ توسط ویلکس (Wilkes) پیاده سازی شد ، روشی است که دارای هزینه بالا ، سرعت کم ،اما از انعطاف پذیری بالایی برخوردار است . این روش بدین صورت انجام می شود که سیگنالهای کنترلی خوانده می شوند وبه مقاصد مربوطه هدایت می گردند.

برای درک بهتر این دو روش از طراحی واحد کنترل به مثال زیر توجه کنید .

مثال : فرض کنید هر ثبات از سه قسمت تشکیل شده باشد . مثلاً یک پایه ShL ویک پایه ShR ویک پایه Load (بارکردن) داشته باشد ، که اگر محتوای آنها یک باشد ، عمل مورد نظر را انجام دهد واگر محتوای آنها صفر بود این اعمال را انجام ندهد. حال این پایه ها را شماره گذاری می کنیم و واحد کنترل شماره آنها را معین می کند . برای مثال اگر بخواهد عمل ShR را انجام دهد، شماره ۲ را فعال می کند وبه همین صورت برای هر پایه شماره آن را می فرستد.

فرض کنید که بخواهیم اعمالی را طبق شماره های 1,2,6,5,3 انجام دهیم. برای این منظور می توانیم شمارنده‌ایی را طراحی کنیم که به ترتیب اعداد 1,2,6,5,3 را بشمارد .

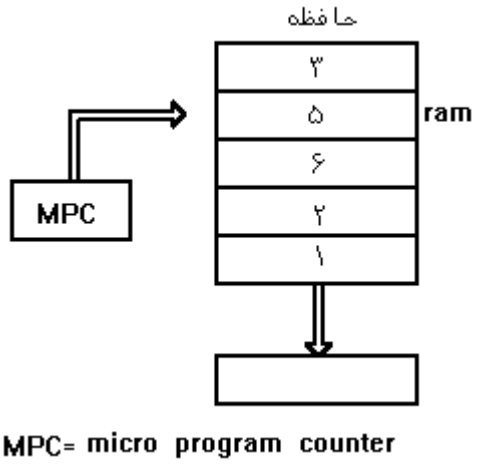
چون بزرگترین عدد که همان عدد 6 می باشد ، در 3 بیت جای می شود . بنابراین سه خروجی از واحد کنترل خارج می شود . این سه خروجی را به کدگشا وصل می کنیم تا وقتی شمارندی ما می شمارد ، کدگشا شماره متناظر با ثبات مورد نظر را فعال نماید .



(شکل ۱-۱۰)

حال اگر اشتباهی در ترتیب شمارنده صورت گرفته باشد ، برای مثال بجای شماره های 1,2,6,5,3 می بایست شماره های 1,3,6,5,2 را وارد می کردیم ، در اینصورت باید یک واحد شمارنده مجزا برای آن طراحی نماییم. (همانطور که در مدار منطقی داشتیم) پس نتیجه می گیریم که روش سیم بندی شده انعطاف پذیر نمی باشد .

ولی در روش ریز برنامه سازی یک حافظه داریم ،که اعداد مورد نظر به ترتیبی که می خواهیم شمارش شوند را در سطرهای حافظه قرار می دهیم . در این حالت اگر بخواهیم اعداد موجود در سطرهای حافظه را عوض کنیم به راحتی می توان جای اعداد را عوض نمود. پس این روش بسیار انعطاف پذیر است.

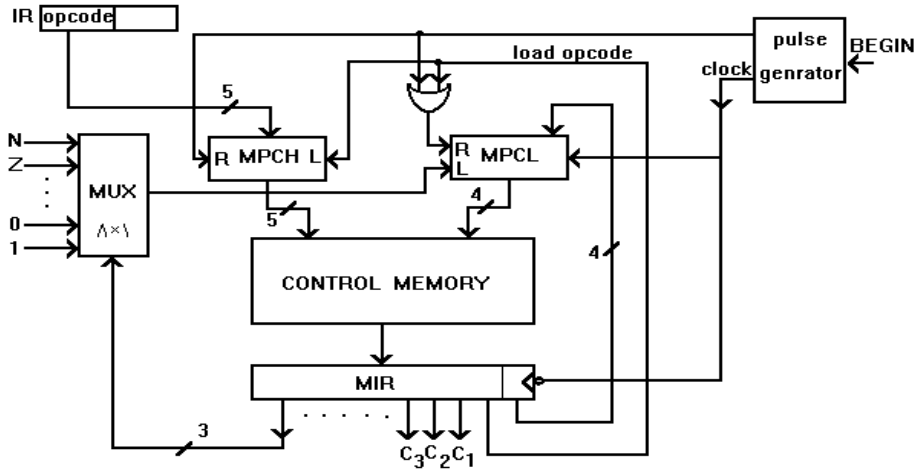


(شکل ۱-۱۱)

نکته : عملاً در کامپیوترها نیز از روش ریز برنامه سازی استفاده می کنند هر چند که پرهزینه وکم سرعت است .

۱-۳ : طراحی واحدکنترل به روش ریزبرنامه سازی (Micro Programmed)

حال ببینیم واحد کنترل براساس ریز برنامه سازی چه ساختاری دارد.

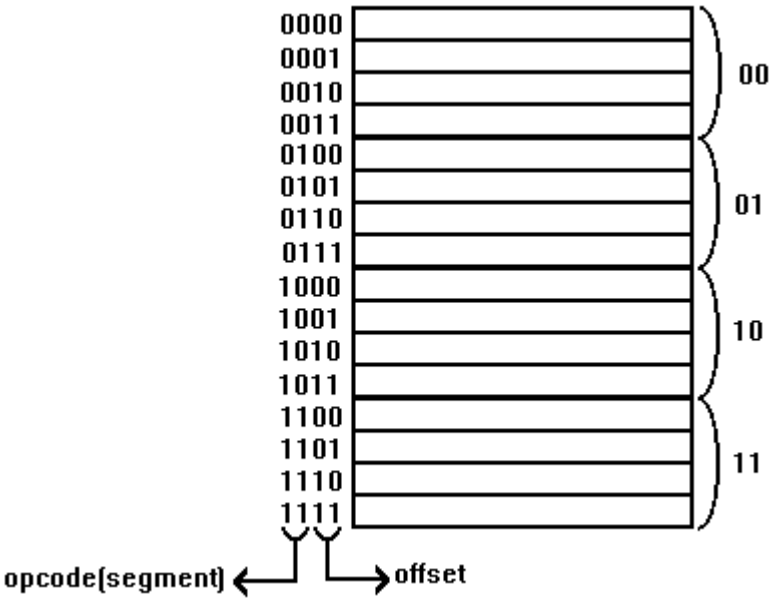


(شکل ۱-۱۲)

در لحظه اول که کامپیوتر را روشن می کنیم از طرف Pulse Genrator یک پالس فرستاده می شود تا $\mu PCL, \mu PCH$ را Clear یا Reset کند و به اجبار در سطر اول حافظه قرار خواهد گرفت، و اولین دستوری که در آن ذخیره شده ، اجرا می گردد . بنابراین باید در اولین سطر حافظه ، برنامه‌ای را که می خواهیم درابتدا اجرا شود ذخیره می کنیم . سپس در سطر اول حافظه دستورالعمل پرش را قرارمی دهیم که برنامه خاصی در آدرس مشخص را در لحظه اول اجرا کند، مثلاً برنامه سیستم عامل را اجرا کند.

در طراحی واحد کنترل براساس ریز برنامه سازی حافظه‌ای وجود دارد که به آن حافظه کنترلی می گوییم وسیگنالهای کنترلی درداخل این حافظه قراردارد و حافظه کنترلی به K قسمت

مساوی تقسیم می شود (K برابر است با تعداد دستورالعمل های ماشین) وهر قسمت از این K قسمت ریز دستورات یک دستور ماشین را شامل می شود . با این عمل آدرس های مربوط به هر سطر از حافظه کنترلی را می توان به دو دسته تقسیم نمود. یک آدرس بیانگر قسمت مربوطه از حافظه کنترلی است (آدرس با ارزشتر) و آدرس دیگر مربوط به شماره سطر از آن قسمت می باشد (آدرس کم ارزشتر) که البته در شکل(۱-۱۳) مشخص است .



(شکل ۱-۱۳)

آدرس بخش با ارزشتر در واقع بیانگر Opcode دستورالعمل می باشد . برای اجرای یک دستورالعمل از زبان ماشین Opcode آن دستور العمل موجود در IR به داخل μPCH منتقل‌می‌شودودر همان لحظه μPCL، Clear می شود این عمل توسط خطLoad Opcode انجام می گیرد .با این کار به ابتدای قسمت مربوط به سیگنالهای کنترلی این دستور از ماشین واقع در حافظه کنترلی مرتبط می شویم ، و از آن جا سیگنالها یکی پس ازدیگری فعال می گردند . درنوشتن ریز برنامه قطعاً نیاز به پرش (از نوع شرطی یا غیر شرطی) به یک محل خاص از حافظه کنترلی خواهیم داشت . این عمل با انتخاب شرط مربوطه از Mux ومهیا کردن آدرس محل پرش در ورودی μPCL تغییر می یابد . در این حالت در صورتی که شرط مربوطه برقرار باشد خروجی Mux یک وعمل Load آدرس جدید به داخل μPCL انجام می گیرد در غیر این صورت عمل پرش انجام نشده وبه μPCL یک واحد اضافه می گردد.

ورودیهای صفرویک در Mux برای نیل به اهداف زیراست .

اگر بخواهیم بدون هیچ شرطی پرش انجام شود یک را انتخاب می کنیم واگر بخواهیم اصلاً پرش نداشته باشیم صفر را انتخاب می کنیم ودر صورت شرطی بودن پرش، یکی از شرطهای قرار داده در ورودی Mux را انتخاب می کنیم .

علت اینکه از گیت OR استفاده می کنیم این است که μPCL در دو مرحله نیاز به Reset شدن دارد.

۱- در مرحله شروع ۲- بعد از Load شدن هر دستور

حافظه کنترلی به هر سطری که اشاره می کند ریز دستورات مربوط به حافظه کنترلی به μIR منتقل می شود. $\mu PCL, \mu PCH$ دارای دو Clk هستند .هر پالس که می آید به μPCL یک واحد اضافه می کند و μIR مقداری که حافظه کنترلی به بیرون می فرستد را در خود ذخیره می نماید .

سؤال : دلیل اینکه μPCL با لبه بالا رونده (مثبت) فعال می شود ولی μIR با لبه پایین رونده (منفی) فعال می شود چیست؟

به این دلیل که لحظه‌ای که μPC مقداری به خود می گیرد ، سطر مربوطه از حافظه کنترلی در همان لحظه اطلاعات آماده شده را در خروجی فراهم نمی سازد .

چون مقدار μPC با یک تأخیر Increment می شود ، پس باید مقداری تأخیر در فرستادن Clk به μPC باشد که این تأخیر را با اعمال Clk بالبه منفی تصحیح می کنیم تا μPCL ما به حالت پایدار از نظر وضعیتی برسد.

ظرفیت μPCH برابر است با تعداد بیتهای Opcode وظرفیت μPCL برابر است با طولانی ترین ریز دستورات برای انجام یک دستور . (یعنی یک ریز برنامه حداکثر به چه تعداد دستور العمل نیاز دارد)

با توجه به ویژگیهای مدار ، حافظه کنترلی بصورت ٤٠*٥١٢ می باشد . بافرض اینکه سخت افزاری که در اختیار داریم به ۳۲ سیگنال کنترلی نیاز داشته باشد و سه بیت هم جهت انتخاب وضعیت (Select Condition) ویک بیت هم جهت Load وچهار بیت هم برای خطی که از μIR می رود که در این صورت داریم :

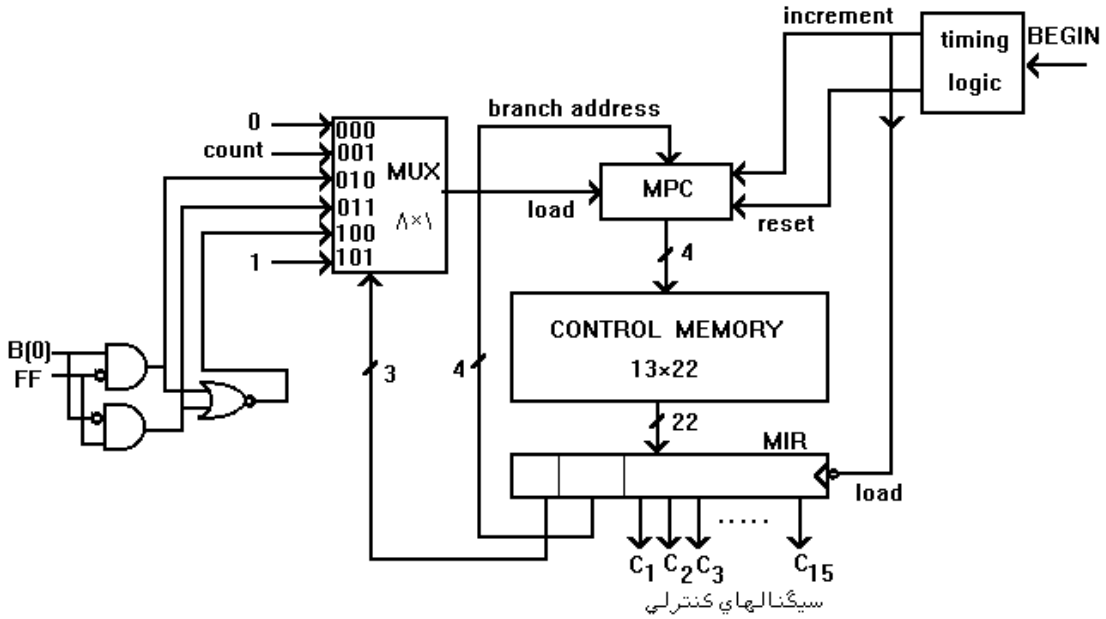
تعداد ستونها = $32 + 3 + 1 + 4 = 40$

تعداد سطرهای حافظه کنترلی برابر با تعداد خطوط ریز برنامه نوشته شده برای آن سخت افزار مورد نظر می باشد .(که در اینجا برابر است با ٥١٢)

مثال : نحوه ساختن واحد کنترل برای الگوریتم ضرب Booth :

چون در اینجا واحد کنترل را فقط برای یک دستورالعمل طراحی می کنیم ، بنابراین از یک ثبات استفاده می نماییم ، پس PC ما به یک قسمت μPC تبدیل می شود وبه عنوان نمونه دارای بخش μPCH نمی باشد. ولی اگر بخواهیم برای کامپیوتر پیاده سازی کنیم باید PC دارای دو قسمت باشد .

مدار لازم بصورت بلوک دیاگرام در ز یر آمده است :



(شکل ۱-۱٤)

در ابتدا برای تعیین تعداد سطرهای حافظه کنترلی می بایست تعداد سطرهای ریز برنامه نوشته شده رابدست آوریم . که در الگوریتم Booth از سطر ۰ تا سطر ۱۲ می باشد که جمعاً ۱۳ سطر می شود، وبرای محاسبه تعداد ستونها ی حافظه کنترلی داریم :

تعداد خطوطی‌که از μIR به μPC می رود + انتخاب حالت + تعدادسیگنالهای‌کنترلی=تعدادستونها

تعداد = $15 + 3 + 4 = 22$

ستونها

پس ظرفیت حافظه کنترلی در روش Booth برابر است با (۲۲*۱۳).

برای B(0) و FF سه حالت مختلف داریم و هنگامی که count به ۸ برسد شرط توقف فراخوانده می شود .

مجموعه سیگنالهای کنترلی :

{c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15}

Condition Branch

C.M.	Select	Address	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15
۰۰۰	۰۰۰۰	۱	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰
۰۰۰۱	۰۰۰	۰۰۰۰	۱	۰	۰	۰	۱	۰	۱	۰	۰	۰	۰	۰	۱	۰	۰
۰۰۱۰	۰۰۰	۰۰۰	۰	۱	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰
۰۰۱۱	۰۰۰	۰۰۰	۰	۱	۰	۰	۰	۰	۰	۰	۰	۱	۰	۰	۰	۰	۰
۰۱۰۰	۰۱۰	۱۰۰	۰	۱	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰
۰۱۰۱	۱۰۰	۱۰۱	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۰
۰۱۱۰	۰۰۰	۰۰۰	۰	۰	۰	۰	۰	۰	۰	۰	۱	۰	۰	۰	۰	۰	۰
۰۱۱۱	۱۰۱	۱۰۱	۰	۰	۰	۰	۰	۱	۰	۰	۱	۰	۰	۰	۰	۰	۰
۱۰۰۰	۰۰۰	۰۰۰	۰	۰	۰	۰	۰	۰	۰	۰	۰	۱	۰	۰	۰	۰	۰
۱۰۰۱	۰۰۰	۰۰۰	۰	۰	۰	۰	۰	۱	۰	۰	۰	۱	۰	۰	۰	۰	۰

.

۱۰۱۰ ۰۰۱ ۰۱۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۱ ۰ ۰ ۰ ۱ ۰ ۰ ۱

.

۱۰۱۱ ۰۰۰ ۰۰۰ ۰ ۰ ۱ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰

.

۱۱۰۰ ۰۰۰ ۰۰۰ ۰ ۰ ۰ ۱ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰ ۰

.

تمرینات

۱- مدار لازم برای ضرب دو عدد علامتدار به روش Booth را رسم نمایید ودر مورد این روش توضیح دهید. مراحل اجرای زیر عمل برای ضرب دو عدد را بصورت الگوریتمی بیان نمایید؟

۲- می خواهیم در مدار سوٴال یک این امکان را بوجود آوریم که در ضرب دو عدد A × K وA×K پس از وارد کردن A,K وبه دست آوردن نتیجه ، جهت ضرب دو عدد B , K نیاز به وارد کردن مجدد عدد K نباشد ؟

۳- آیا عبارت زیر درست است ؟

- در طراحی واحد کنترل براساس روش Delay element و Sequence Counter هر دو اساس طراحی یکسانی دارند اما روش دوم به گونه‌ای است که هزنی طراحی کمتری دارد .

٤- به دو طریق می توان واحد کنترل را طراحی نمود :

۱- ..... ۲- .....

روش ..... سریعتر وروش ..... انعطاف پذیر تر است .

۵- الف) ALU طراحی کنید که بتواند دستورات جمع ، تفریق ، ضرب و فاکتوریل را برای اعداد ۸ بیتی انجام دهید؟ (طراحی براساس روش جمع های متوالی)

ب) برای ALU طراحی شده ، واحد کنترل براساس روش ریز برنامه سازی طراحی نمایید وریزبرنامی لازم برای عمل فاکتوریل را بنویسید؟

٦- مطلوب است طراحی یک کامپیوتر با ۵ دستورالعمل زیر ، براساس روش ریزبرنامه سازی . این کامپیوتر دارای حافظی RAM به ظرفیت ۳۲ کلمه ٤بیتی و AC و Insruction Register ۸ بینی است .

CLA		AC <----- 0
LDA	y	AC <----- c(y)
ADA	y	AC <----- AC + c(y)

JMP	y	PC <----- y
JC	y	IF Carry Flag = 1 Then PC<----- y

۷- شکل کلی طراحی واحد کنترل براساس ریز برنامه سازی برای یک کامپیوتر را رسم نموده ونحوی عملکرد آن را توضیح دهید ؟

۸- الف) چهار ثبات ۸ بیتی با خروجی 3- State در اختیار داریم . به کمک یک جمع کننده ٤ بیتی مداری طراحی کنید که بتوان هر یک از دو ثبات خواسته شده را با هم جمع نموده وحاصل را در ثبات دیگری قرار دهد؟

ب) برای مدار قسمت (الف) واحد کنترل نوع ریز برنامه سازی را طراحی کنید . نوشتن ریز برنامه لازم نیست؟

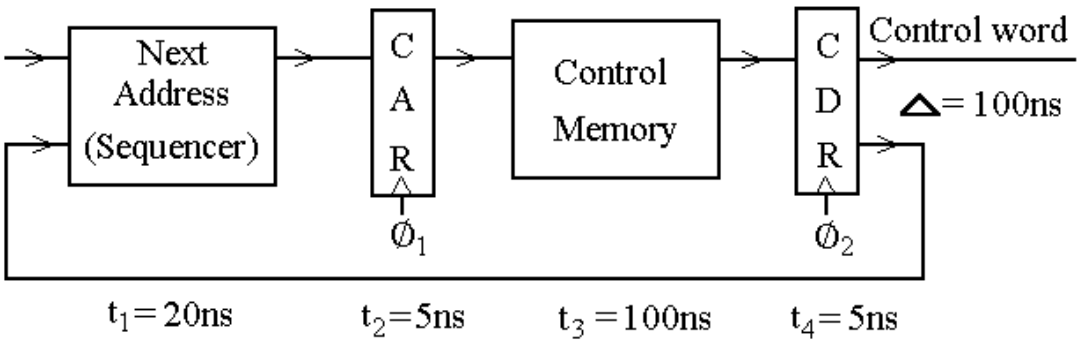
۹- نمودار یک واحد کنترل به روش ریز برنامه سازی نشان داده شده است . t ها تأخیر در اجزاء و Δ تأخیر در اجرای زیرعمل و ($\mu - op$) می باشد .

الف) حداقل پریود کلاک ۲۳۰ ns می باشد .

ب) حداقل پریود کلاک ۱۳۰ ns می باشد.

ج) با حذف CDR حداقل کلاک ۲۰۵ ns می شود .

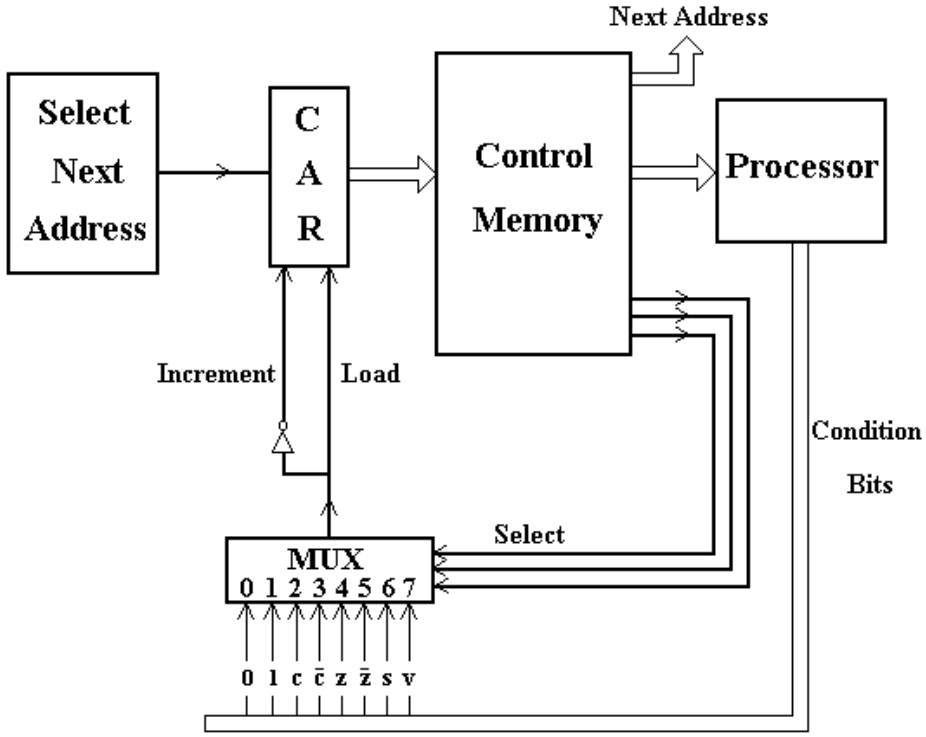
د) با حذف CDR حداقل کلاک ۲۲۵ ns می شود.



کدامیک از گزینه های زیر درست است ؟

۱) فقط ب ۲) فقط ج ۳) الف ود ٤) ب وج

۱۰- شکل زیر بخشی از واحد کنترل زیربرنامه پذیر می باشد . مشخص کنید اگر A=36 و Select (0101)2 و CAR=20 و پردازندی دستور R1-R2 را اجرا کرده باشد ، کدام گزاره صحیح است (R1 , R2 بدون علامت فرض شده اند وتفریق به روش مکمل دو انجام می شود .)



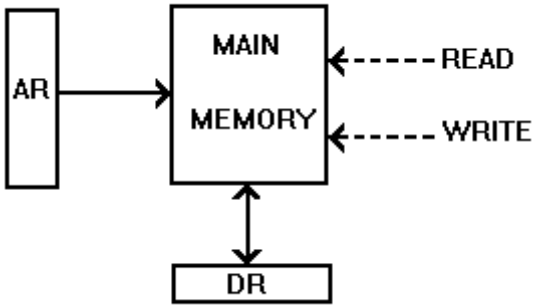
۱) IF $(R_1 \geq R_2)$ Then CAR <----- 36 else CAR <----- 21

۲) IF (R1 > R2) Then CAR < ------ 21 else CAR <----- 36

۳) IF $(R_1 \leq R_2)$ Then CAR <----- 21 else CAR <------36

٤) IF (R1 < R2) Then CAR <------ 36 else CAR <----- 21

۲- سازمان حافظه (Memory Organization)



(شکل ۲-۱)

۲-۱ : جنس حافظه

بطور کلی تکنولوژی ساخت حافظه به دو دسته زیرتقسیم می شود :

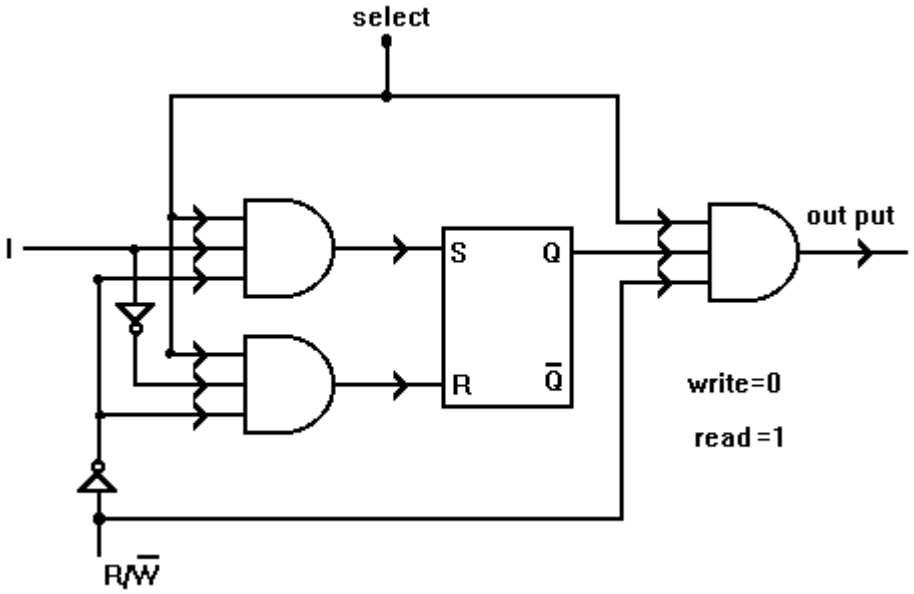
۱) نیمه رسانا (Semiconduetor)

۲) حلقه های مغناطیسی (Magnetic Core)

۲-۱-۱ : حافظه نیمه رسانا (Semiconductor)

این نوع از حافظه با قطع برق محتوای خود را از دست می دهد. اما مخرب در برابر عمل خواندن نیست .

هر سلول از حافظه نوع نیمه رسانا که در کامپیوتر های امروزی همان بیت هاست . هسته ی آن یک فیلیپ فلاپ بوده و بقیه برای کنترل آن می باشد .



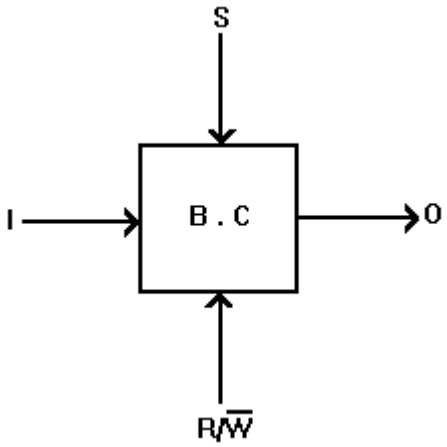
(شکل ۲-۲)

Read =۱ خروجی خود را آزاد می کند .

Write = ۰ ورودی خود را آزاد می کند .

وقتی عمل نوشتن صورت می گیرد که اولاً Select = 1 شود ، ثانیاً برای نوشتن W=0 باشد . وقتی عمل خواندن انجام می شود که Select = 1 شود ، ثانیاً برای خواندن R=1 باشد .

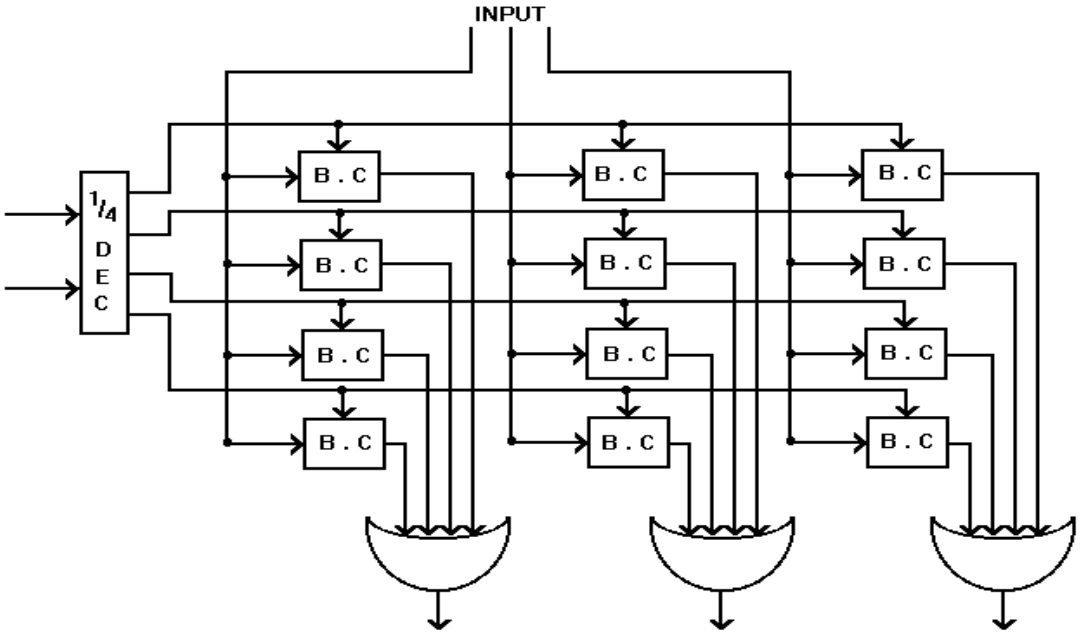
بطور کلی یک سلول از حافظه (Binery cell) را می توان بدین صورت نشان داد.



(شکل ۲-۳)

ساختار داخلی حافظه نوع نیمه رسانا ۳*٤

* منظور از ۳*٤ یعنی اینکه ٤ تا کلمه دارد وتعداد بیتهای هر کلمه ۳ تا می باشد .



(شکل ۲-٤)

* خروجی سلولها را با هم OR کردیم ، چون نمی توانیم آنها را به هم وصل کنیم زیرا باعث سوختن IC می شود.

* در این ساختار اطلاعات به همه سلولها داده می شود واز هرکدام که خواستیم اطلاعات مورد نیاز را می خوانیم .

* سیگنال R/W به همه سلولها وصل است وما آنرا نشان نداده ایم .

هنگامی که می خواهیم مثلا سطر یک را آدرس دهی کنیم ، آدرس آن توسط Decoder اعمال و سطر یک فعال می شود . به این ترتیب که بیت های سطر یک با توجه به اینکه پایه Select آن (که به Decoder متصل است) فعال است ، فعال می شوند و سطر یک آدرس دهی می گردد.

قیمت RAM به این دلیل بالاست که به ازای هر بیت یک FF می خواهد و اگر تعداد سطرها فقط یک مگا بایت هم باشد ، 20_2 گیت OR نیاز داریم و مسلما این حافظه به دلیل اینکه باید از FF و گیت OR عبور کند ، تاخیر (Delay) خواهد داشت .

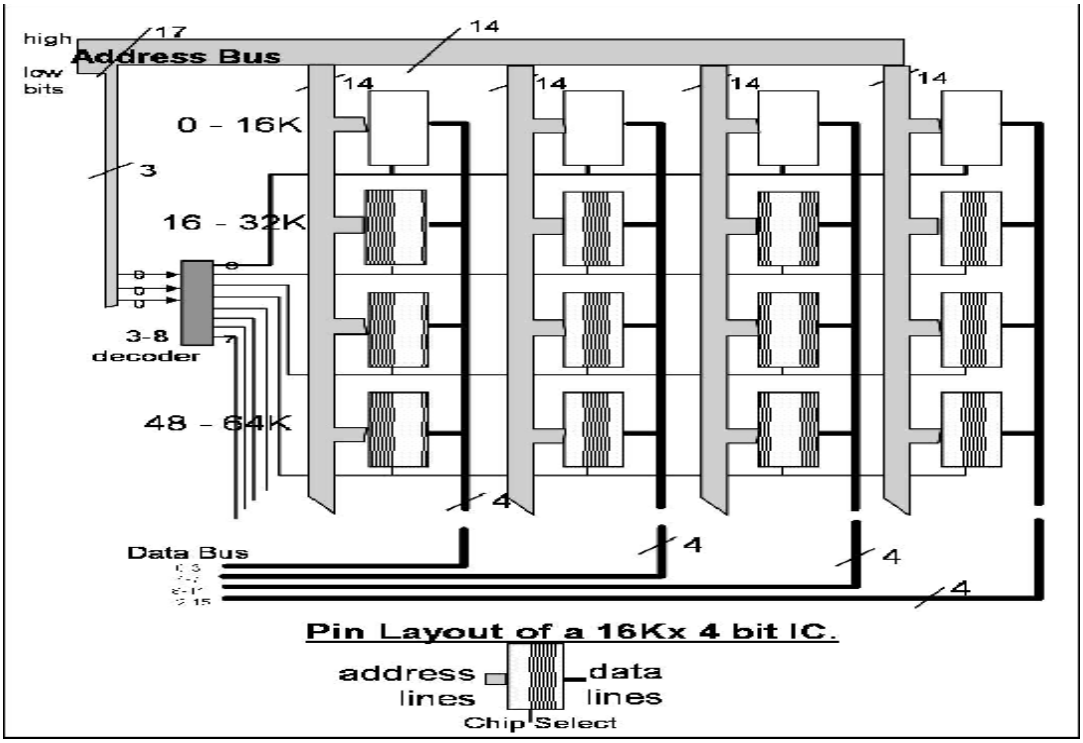
گاهی اوقات که می خواهیم RAM ها را تبدیل کنیم ، می توانیم RAM های ک.چکتر را کنار هم قرار دهیم .

ساختار حافظه زیر را که یک حافظه ی 128K *16 bit است را در نظر بگیرید . این RAM از ۲۰ تا RAM 16K * 4 bit ساخته شده است که خود این RAM ها هر کدام از RAM های کوچکتری ساخته شده اند .

برای طراحی باس های داده و آدرس به صورت زیر عمل می کنیم :

باس داده را ٤ بیت ، ٤ بیت جدا کرده و خطوط کم ارزش تر را به RAM اولی می دهیم . (کل بیت های باس داده ١٦ بیت است) و باس آدرس را که ١٧ بیت است ، ١٤ بیت آن را به هر کدام از RAM ها متصل کرده و ٣ بیت کم ارزش تر آن را به Decoder متصل می کنیم و مشخص می شود که کدام سطر فعال می شود .

برای جبران کندی حافظه از برگ برگ کردن حافظه استفاده می شود .(Memory Inter Leaving)



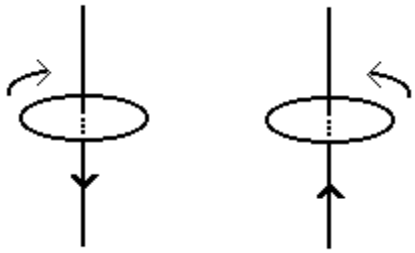
(شکل ۲ – ۵)

۲-۱-۲ : حافظه حلقه های مغناطیسی(Magnetic Core)

این نوع از حافظه با قطع برق محتوای خود را حفظ می کند . با خواندن یک سطر محتوای آن سطر از بین می رود ، از این جهت در این نوع حافظه به دنبال هر عمل خواندن یک عمل نوشتن انجام می شود وقبل از هر عمل نوشتن در حافظه باید یک عمل خواندن صورت گیرد . زیرا برای نوشتن در یک محل از حافظه باید ابتدا آن محل را Clear کنیم بنابراین یک بار آن محل را می خوانیم تا Clear شود . بدین ترتیب سرعت خواندن ونوشتن اطلاعات کمترمی شود .

سلول حافظه از نوع حلقه مغناطیسی

هر سلول از حافظه نوع حلقه مغناطیسی بنابر قاعده ونسبت به جهت جریان وجهت میدان مغناطیسی حالت صفر یا یک را به خود می گیرد . می دانیم که اگر از یک مسیر بسته میدان مغناطیسی عبور کند ، طبق قانون دست راست جریان در سیم ایجاد می شود .

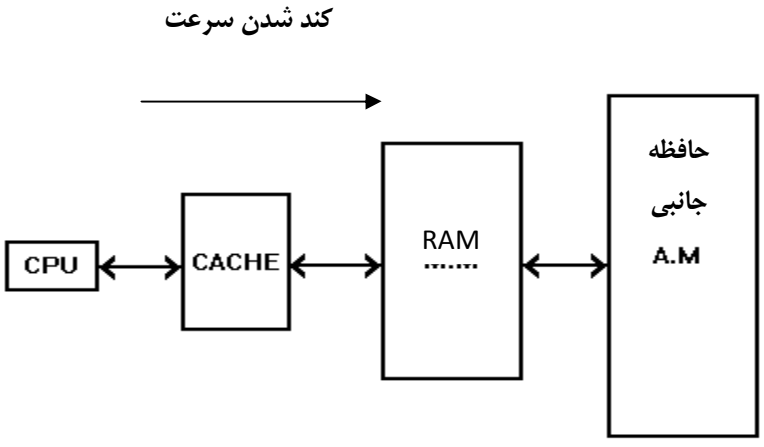


(شکل ۲-٦)

برای خواندن سیم هایی از این حلقه ها عبور می کند که در جهت مشخصی از آن جریانی عبور می دهیم که خود یک میدان مغناطیسی ایجاد می کند . میدان قبلی را با بعدی مقایسه می کنیم . اگر هم جهت باشند تغییری حاصل نمی شود ولی اگر هم جهت نباشند ، جهت میدان قبلی تغییر می کند و بعد از آن چون اطلاعات پاک می شود آن ها را دوباره می نویسیم .

۲-۲ : سلسله مراتب ذخیره سازی اطلاعات

شکل زیر اجزای یک سیستم سلسله مراتبی حافظه را نشان می دهد .



(شکل ۲-۷)

چون حجم اطلاعاتی که باید پردازش شود زیاد می باشد . پس ظرفیت حافظه ها هم بالا می رود واز آنجا که CPU ، روز به روز در حال پیشرفت است وسرعتش بیشتر می گردد،اگر بخواهیم با پیشرفت CPU سرعت حافظه را زیاد کنیم ، به علت گرانی حافظه، هزینه زیاد می شود . لذا از Cache استفاده می کنیم که از حافظه اصلی حجمش کمتر بوده ولی سرعتش بیشترمی باشد . در واقع سرعت آن حداکثر، برابر سرعت CPU است . Cache در کامپیوتر های جدید معمولا دو سطحی است: داخلی و خارجی که Cache داخلی درون CPU قرار دارد . به ازای ظرفیت Cache اطلاعات از حافظه اصلی به داخل Cache می آید. برای مثال اگر ظرفیت Cache ٤ کیلو بایت باشد به اندازه 4KB از حافظی اصلی برروی Cache می نشیند و Cache نیز با CPU در ارتباط می باشد . به همین خاطر پیشنهاد می شود که در داخل برنامه ازحلقه هایی با بدنه طولانی استفاده نشود . زیرا حلقه های کوتاه کاملاً در Cache جای می گیرند ، ولی اگر حلقه طولانی باشد ، باید نیمی از حلقه به Cache بیاید ونیمه دیگر ، دوباره به Cache فراخوانی شود ودر نتیجه کارایی Cache پایین می آید . به همین دلیل است که از دستور Goto در برنامه ها کمتر استفاده می شود ویا اصلاً استفاده نمی شود .

فرض کنید که ظرفیت Cache ۱۰۰ سطر و حلقه ی ما به طول ۱۵۰ سطر باشد . ابتدا ۱۰۰ سطر اول درون Cache قرار می گیرد و برای آوردن ۵۰ خط بعدی باید سطرهای قبلی پاک شود . در نتیجه برنامه ناقص خواهد شد و سرعت خیلی پایین می آید .

تمرین : برآورد اندازه Cache در کامپیوتر

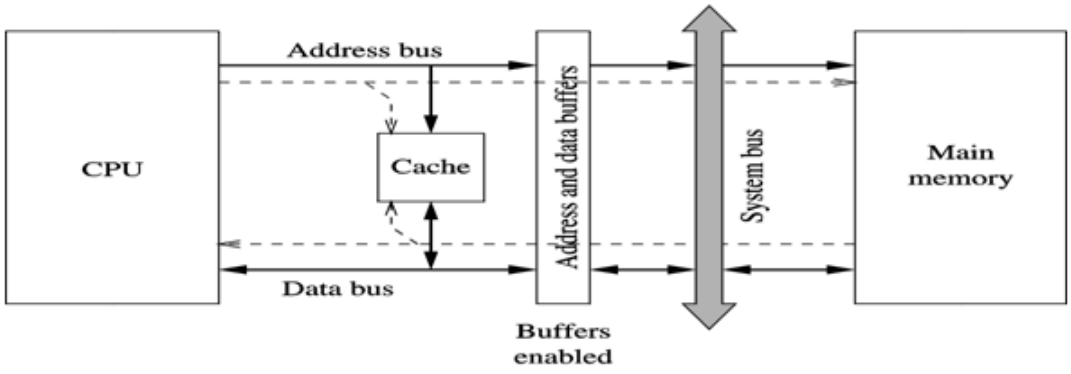
فرض کنید ظرفیت Cache ، 4Kb باشد . ابتدا برنامه ای بنویسید که آرایه ی 1Kb را پردازش کرده و در هر مرحله 1Kb به آن اضافه کرده تا بتوانید میزان تقریبی Cache را بدست آورید و همچنین زمانی که هر پردازش در هر مرحله طول کشیده است را ذخیره و باهم مقایسه کرده و نمودار آن را رسم کنید.

نرخ برخورد (hit ratio) در Cache یعنی احتمال اینکه یک سطر در حافظه باشد ، چقدر است و از طریق تقسیم تعداد دفعاتی که محلی که می خواهیم و در Cache هست به کل دفعاتی که عمل خواندن صورت می گیرد ، بدست می آید.

هر چه از حلقه های بزرگ تر از سایز Cache استفاده کنیم ، hit ratio پایین می آید و در نتیجه باعث پایین آمدن کارایی Cache خواهد شد .

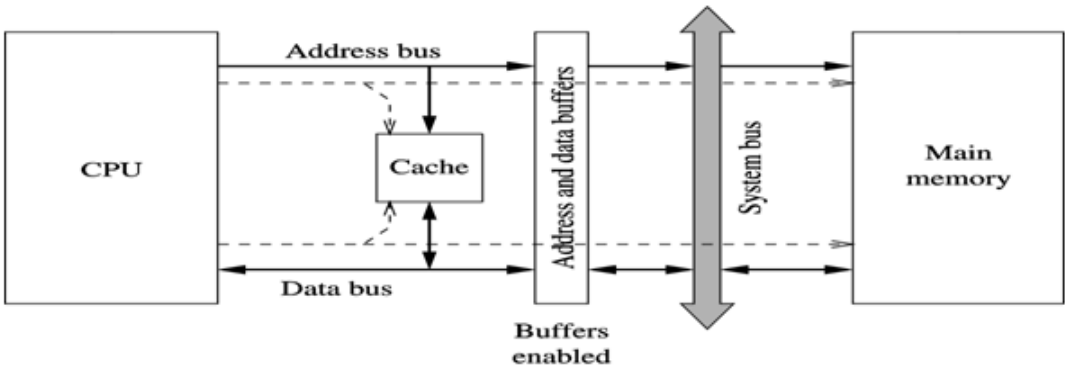
Miss ratio = 1 – hit ratio

هنگامی که CPU می خواهد اطاعاتی را از حافظه بخواند و hit نیست ، اطلاعات همزمان هم به داخل Cache و هم به داخل CPU می رود .



(شکل ۲–۸)

و هنگامی که CPU می خواهد اطلاعاتی را بنویسد و hit باشد ، اطلاعات همزمان هم در Cache و هم در RAM نوشته می شود .



(شکل ۲ – ۹)

تمرین : برای ماتریسی که در اختیار دارید ، نحوه ی پردازش مناسب چگونه است ؟ سطری یا ستونی

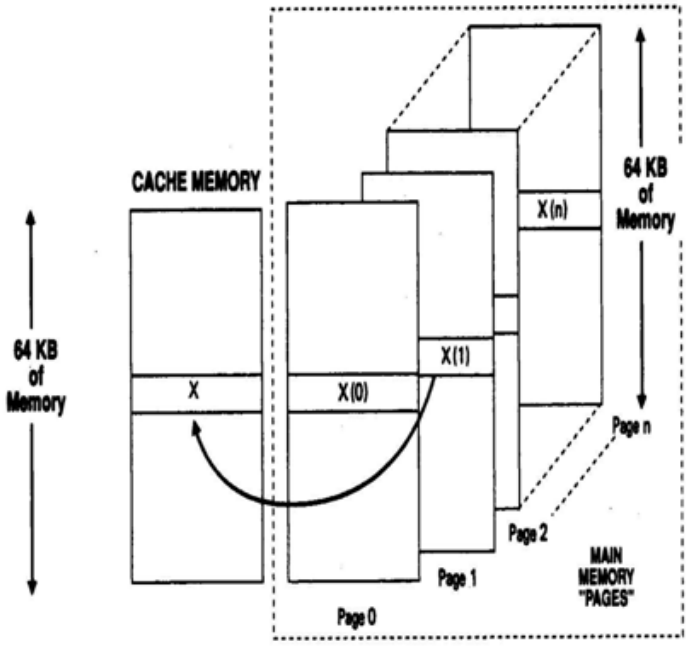
برای این کار برنامه ای بنویسید و برای یک ماتریس یک با پردازش سطری را انجام داده و یک بار پردازش ستونی سپس اندازه ی ماتریس را تغییر داده و مدت زمان انجام هر کدام از مراحل را ذخیره کرده و نمودار آنرا رسم نمایید .

انتقال اطلاعات از RAM به Cache بلاک بلاک است و آدرس آن دو قسمت دارد : شماره بلاک و offset بلاک (سطر مربوطه در بلاک مشخص شده) ولی انتقال اطلاعات از CPU به Cache بصورت word است.

۲- ۳ : انواع Cache

۲- ۳- ۱- Direct Mapping

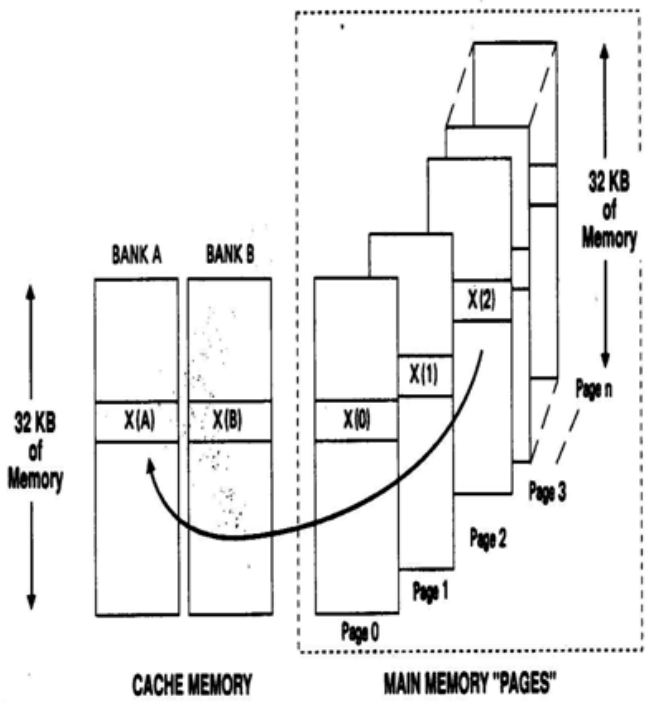
در این نوع از Cache هر سطر از RAM جای مشخصی در Cache دارد. فضای RAM به طولی برابر Cache تقسیم شده و هر سطر در جایی مشخص درج می شود که این روش ساده ترین و کم هزینه ترین است .



(شکل ۲ – ۱۰)

Set – Associative – ۲ -۳ -۲

در این نوع Cache ، هر سطر از RAM در تعدادی محل از Cache قابل نشستن است . برای Cache دو تا پشتیبان داریم . یعنی دو آزادی عمل (یا در نیمه چپ یا در نیمه راست)



(شکل ۲- ۱۱)

Associative mapping – ۳-۳-۲

در این نوع از Cache ، هر سطر از RAM در هر جا از Cache می تواند بنشیند و این روش پر هزینه ترین خواهد بود .

هر سطر از Cache شامل قسمت های Date و tag و valid می باشد که Data همان اطلاعات درون Cache و tag قسمت آدرس آن است و valid مشخص می کند که آیا این سطر مطابق RAM است یا خیر

مثال : فرض کنید دستور زیر صادر شود.

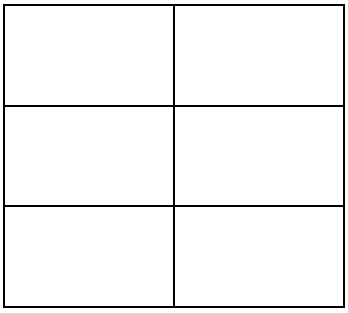
LDA 10

CPU ابتدا به سراغ Cache می رود واگردستور مورد نظر در آن وجود داشته باشد، از آن استفاده می کند ولی اگر این دستور در Cache نباشد، از حافظه اصلی آنرا به Cache انتقال می دهد وسپس از آن استفاده می کند (یعنی آنرا اجرا می کند.)

مشکل اول

از کجا بدانیم محتوای خانه 10 که آن را در درون Cache مورد جستجو قرار می دهیم، چیست ؟ برای رفع این مشکل،Cache را به شکل زیر در نظر می گیریم بنابراین در مثالLDA 10 بدنبال‌شماره‌ایی بنام‌سطر10درداخل‌ستون شماره‌سطرهایCache می گردد .

محتوی	شماره سطر



(شکل ۲-۱۲)

مشکل دوم

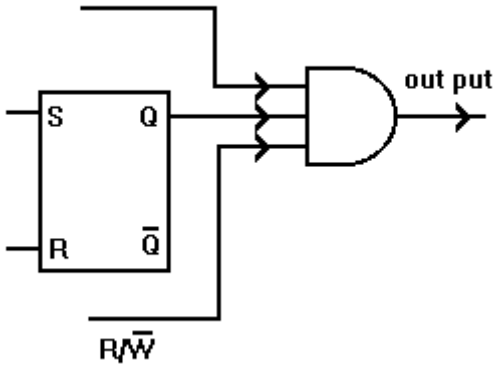
وقتی که گفته می شود سطر 10 را بخوان ، در داخل Cache جستجو می نمایدکه آیا سطر 10 در Cache وجود دارد یا خیر ؟ برای این عمل جستجو ، باید تمامی حافظه Cache را پیمایش نماید که این کارایی Cache را پایین می آورد و عملیات مورد نظر طولانی می باشد . پس برای رفع این مشکل ساختار حافظه انجمنی را بیان می کنیم . در واقع حافظه Cache کمی امکاناتش از حافظه اصلی بیشتر است.

جستجو به صورت موازی صورت می گیرد یعنی یک پالس کافی است تا به نقطه ی مورد نظر برسیم و این یکی از دلایلی است که قیمت Cache بالاست .

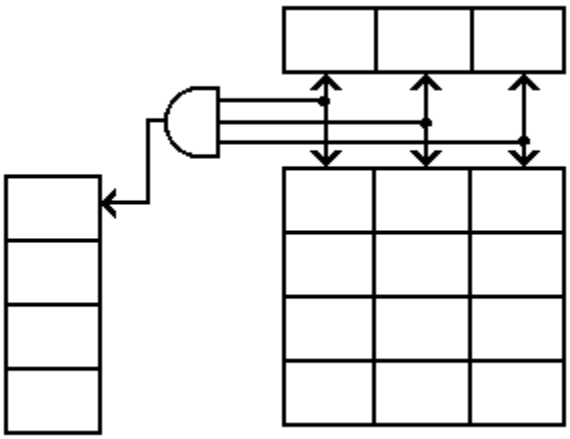
۲-٤ :حافظه انجمنی

در حافظه انجمنی مقایسه بصورت موازی انجام می شود وما در این حافظه ها داده (Data)را می دهیم وآدرس آنرا می گیریم ، برخلاف حافظه های معمولی که آدرس را می دهیم وداده موردنظر را دریافت می داریم .

ساختار داخلی حافظه انجمنی



(شکل ۲-۱۳)



(شکل ۲-۹)

* حافظه انجمنی مانند حافظه های معمولی قابلیت دستیابی بصورت تصادفی (Random) را دارد.

قسمتهای متناظر که در شکل (۲-۹) نشان داده شده است با هم یای انحصاری (EX - Clusive NOR) می شوند وحاصل این یاهای انحصاری با هم AND می شود وبه ازای هر سطر از حافظه انجمنی یک بیت را در نظر می گیریم ، که حاصل AND شده برای هر سطر در بیت همان سطر قرارمی گیرد . واضح است وقتی حاصل بیت انجمنی یک می شود که ، محتوای ثبات باسطر حافظه انجمنی برابر باشد ومی تواند چندین بیت از بیتهای تشخیص داده موردنظر در حافظه انجمنی Set (یعنی یک) باشند . در این حالت این داده در چند سطر از حافظه انجمنی تکرار شده است .

حال این سوٴال مطرح می شود که وقتی اطلاعات موردنیاز در داخل Cache نباشد، باید CPU اطلاعات را از داخل حافظه اصلی به داخل CPU بیاورد وسپس آنرا به داخل Cache ببرد، در اینصورت لزوم وجود Cache چیست ؟

در پاسخ باید گفت که اطلاعات به داخل CPU آورده نمی شود . بلکه CPU یک خط کنترلی را فعال کرده وآن وسیله دیگر ، این کار را انجام می دهد . پس این امر موضوع جدیدی را که در فصل آینده بحث خواهد شد ، فراهم می آورد .

تمرینات

۱- حافظی RAM از جنس ..... برخلاف ..... با قطع شدن برق محتوای خود را از دست می دهد. سرعت حافظه از جنس ..... کمتر از دیگری است.

۲- در ..... برخلاف حافظه های RAM با دادن محتوا ، آدرس محل یا محلهای قرار گرفتن آن در حافظه بدست می آید.

۳- ساختار داخلی حافظی RAM نوع Semiconductor برای یک حافظی 4 × 5 را نمایش دهید؟

٤- مدار داخلی یک سلول از حافظی RAM را رسم نمایید؟

۵- جایگاه حافظی Cache ونحوی دسترسی به اطلاعات واقع در حافظی اصلی برای CPU ای که مجهز به حافظی Cache باشد را توضیح دهید؟

۳-روشهای متفاوت انتقال اطلاعات بین کامپیوتر ودستگاههای جانبی

برنامه ها و توابع محاسباتی توسط دو مولفه یکی CPU ودیگری حافظه اصلی انجام می پذیرد . اما می دانیم که در هر کامپیوتر ، ما نیازمند به برقراری ارتباط بادنیای کامپیوتر هستیم ، تا بتوانیم اطلاعات را جهت پردازش از خارج بگیریم ویاآنکه اطلاعات پردازش شده را به‌کاربر نشان بدهیم .از دستگاههای ورودی وخروجی می توان به دیسک خوانها ، چاپگرها و ... اشاره نمود، که آنها را با نام ابزارهای ورودی / خروجی (I/O Device) می شناسیم .

شیوه های مختلفی برای رد و بدل کردن اطلاعات بین I/O Device وکامپیوتر وجود دارد که این شیوه ها عبارتند از :

۱) Programmed I/O

۲) Intrupt

۳) DMA (Direct Memory Access)

۳-۱ : برنامه ریزی ورودی / خروجی (programed I/O)

در این روش ردو بدل شدن اطلاعات بین دستگاههای جانبی و حافظی اصلی با دخالت مستقیم CPU انجام می شودو دقیقاً از کانال پردازنده عبور می کند به عبارت دیگر دستگاههای جانبی با حافظی اصلی ارتباط ندارند . برای آنکه اطلاعات از دستگاههای جانبی وارد حافظه شود ، بایستی توسط دستورالعملی به داخل پردازنده انتقال یافته وسپس توسط دستورالعمل دیگری از پردازنده به حافظی اصلی انتقال یابد وچون از کانال پردازنده عبور می کند و اینکه دائما باید CPU ، I/O ها را چک کند ، دارای سرعت کمی است .

البته دراین روش بایستی دقت داشت به اینکه ، در هنگام خواندن اطلاعات توسط CPU از دستگاه جانبی ، آن دستگاه برای چنین کاری مهیا باشد . برای عمل دست دادن (Hand Shaking) لازم است که CPU مدت زیادی در انتظار به سر برد (تلفن بدون زنگ) . برای رفع این مشکل روش دوم را پیشنهاد می کنند.

۳-۲ : وقفه(Intrupt)

در این روش CPU دائماً مشغول اجرای دستورات خود است وچنانچه دستگاه خارجی نیاز به سرویس داشته باشد ، وقفه‌ای به پردازنده ارسال می شود. پردازنده نیز در پایان دستورالعمل جاری به آن وقفه رسیدگی خواهد کرد . دستگاه ها در این روش به یک Encoder متصل هستند و هنگامی که درخواستی ارسال می شود ، مشخص می شود که چه دستگاهی و یا چه آدرسی درخواست سرویس داده است . (مثل هنگامی که زنگ تلفن به صدا در می آید .) در پایان هر دستورالعمل وقفه ها چک می شوند . اگر وقفه داشتیم چون مدار ترکیبی است بدون اینکه زمانی تلف شود ، برنامه تغییر مسیر داده و با توجه اولویت بندی Encoder به سرویس لازم رسیدگی می کند .

با استفاده از Intrupt Acknowledge اطلاعات را در CPU ذخیره کرده و بعد از رسیدگی به سرویس وقفه بازمی گردد و چون اطلاعات قبل خود را ذخیره دارد ، به برنامه خودش ادامه می دهد .

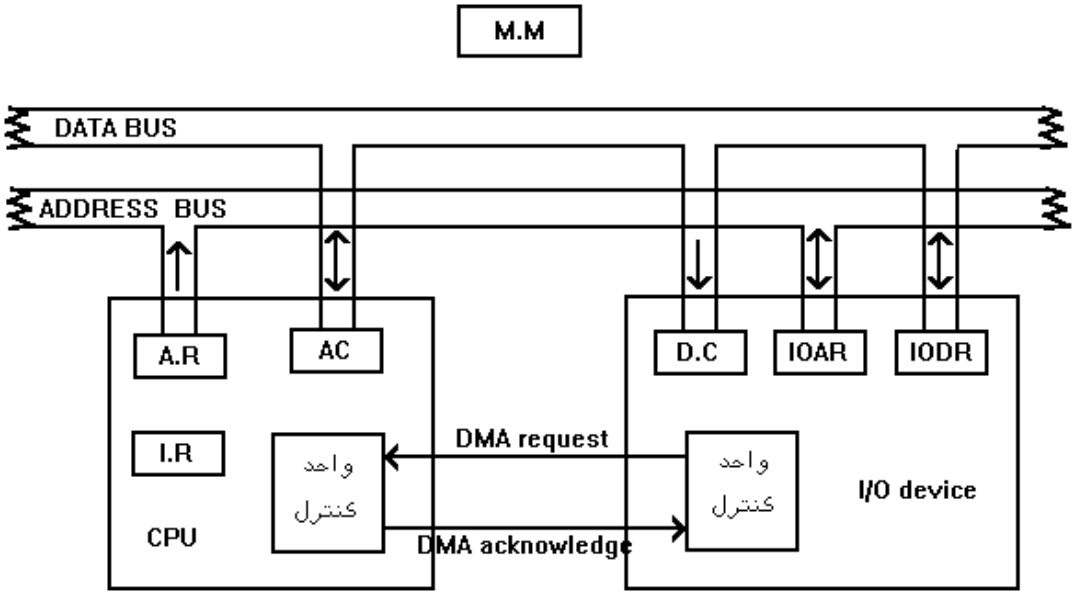
دو نوع وقفه داریم : نرم افزاری و سخت افزاری که وقفه های نرم افزاری غیرقابل چشم پوشی هستن ولی با استفاده از گیت AND می توان از وقفه های سخت افزاری چشم پوشی کرد .

۳-۳ : دسترسی مستقیم به حافظه (Direct Memory Access) DMA))

یکی دیگر از معایب روش اول آن است که برای رد وبدل شدن اطلاعات از دستگاه جانبی به داخل حافظه ، حتماً بایستی داده از کانال CPU عبور کند ،که در حجمهای بالا باعث کندی سرعت می شود . برای رفع چنین مشکلی تکنیک DMA را مطرح می سازند که دراین تکنیک دستگاه جانبی بدون دخالت مستقیم CPU بتواند اطلاعات رادر داخل حافظه اصلی ذخیره ویااز آن باز یابی کند .

انتقال داده ها بین یک وسیلی ذخیره سازی سریع از قبیل دیسک مغناطیسی وحافظه را اغلب سرعت CPU محدود می کند حذف CPU از مسیر انتقال وایجاد این امکان که دستگاه جانبی مستقیماً گذرگاههای حافظه را کنترل کند ، سرعت انتقال را بالا می برد . در حین انتقال به روش DMA ، CPU بیکار است وکنترلی بر گذرگاههای حافظه ندارد یک کنترل کننده DMA به منظور اداری امر انتقال به طور مستقیم بین وسیلی I/O وحافظه ، کنترل گذرگاهها را برعهده می گیرد . در این روش داده ها (Data) بدون آنکه از کانال پردازنده بگذرد به مقصد می رسد ، شیوی انجام کار به صورت زیر می باشد .

هنگامی که قرار است دادهای بین دستگاه I/O وحافظی اصلی رد وبدل شود ، پردازنده آدرسی از حافظی اصلیکه دستگاه I/O می تواند با آن آدرس ارتباط برقرار کند را به آن می دهد وسپس پردازنده خود را BUS جدا نموده وارتباط دستگاه جانبی وحافظی اصلی با توجه به آدرسی که در اختیار دارد به طور مستقیم برقرار می شود. به طور کلی در مدت زمانهایی که پردازنده به حافظی اصلی نیازی نداشته باشد اجازی ارتباط مستقیم (DMA) بین دستگاه جانبی وحافظی اصلی را می دهد . در شکل (۳-۱) نحوی انتقال اطلاعات بین دستگاه جانبی وحافظی اصلی را مشاهده می کنید.

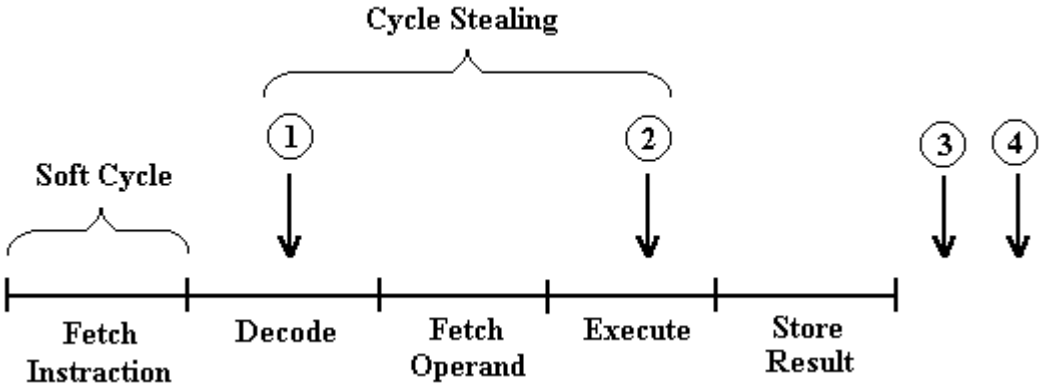


(شکل ۳-۱)

در واقع به DMA در پایان Inst Cycle جاری وزمانی که CPU به حافظه نیازی نداشته باشد اجازه داده می شود تا ارتباط مستقیم خود را برقرار نماید.

آدرس ابتدای اطلاعات مورد نیاز را در IOAR وتا جایی را که می خواهد در DC (یعنی تعداد سطرها را در آن قرار می دهد)که مخفف Data Count است ، قرار می دهد . حالا که CPU دیگر اطلاعات را به I/O داد ، خود را از I/O جدا می کند وکارهای دیگر خودش را انجام میدهد ، حال در وسایل I/O هم AR ، هم DR وهم تعداد داده ها را داریم (در DC) پس می تواند با Bus ارتباط برقرار کند و CPU با جداکردن خود ازوسایل IO این اجازه ارتباط رامی دهد پس درزمانهای مناسب DMA Aknowledge را فعال می کند وخط DMA Aknowledge را فعال نگه می دارد ودستگاه I/O هنگامی که این خط را فعال دید با حافظی اصلی براساس آدرس مشخص شده ارتباط برقرار می کند . هر لحظه که دستگاه I/O یک داده رامنتقل می کند یک واحد از DC ، کم می شود تا به صفر برسد . هر زمان که DC به صفررسیدسیگنالهابه واحد کنترل آن فرستاده نمی شود از این جهت سیگنال DMA Request را غیر فعال می کند وCPU هم سیگنال DMA Aknowledge را غیرفعال می سازد وبه این ترتیب ارتباط از بین می رود .

سوال : در چه زمانی CPU به حافظه اصلی نیاز ندارد؟



(شکل ۳ - ۲)

در لحظات ۱ و ۲ CPU نیاز به حافظه ندارد، مثلاً در سیستم Unix می توانیم ماکار خود را انجام دهیم وعمل چاپ نیز همزمان انجام شود و یادر هنگام رد وبدل شدن اطلاعات بین بافر نرم افزاری وسخت افزاری نیز همین شرط راداریم . یعنی در لحظاتی که ما به CPU کاری نداریم ، این عمل صورت می گیرد ودر واقع از تکنیک DMA استفاده می کند.به وقفه در لحظه ۳ جواب داده می شود یعنی در پایان اجرای دستورالعمل . ولی به DMA در شماره های ۱ و۲ و٤ جواب داده می شود یعنی در حین اجرای دستورالعمل وپایان اجرای دستورالعمل CPU به DMA جواب می دهد .

* بعضی از حلقه های دستورالعمل نیاز به ذخیره (Store) هم دارند ، مثل دستور مثال ۱ ولی در دستور مثال ۲ نیاز به قسمت Store cycle نیست .

مثال ۱ : ADD NUM , AX

مثال ۲: MOV AX , SUM

* در لحظاتی که CPU نیاز به حافظه اصلی (M.M) ندارد ، DMA Acknowledge می فرستد. ولی در Dos پس از بیکاری CPU به کارهای دیگر رسیدگی می شود .

در درس ذخیره وبازیابی اشاره شد که در بافرینگ مضاعف وقتی یک بافر در حال پردازش است ، بافر بعدی در حال پر شدن است که با تکنیک DMA صورت می گیرد ، CPU از موقعیتهای ۱ و ۲ استفاده می کند.همچنین اگرعمل پردازش بافر ۱ تمام شود ، ولی عمل پرشدن بافر ۲ تمام نشود باید منتظر بمانیم تا بافر ۲ پر شود وسپس عمل پردازش بافر ۲ صورت گیرد که در اینصورت CPU از موقعیت ٤ یعنی CPU Boned استفاده می کند.

CPU Boned یعنی برنامه با CPU درگیر است و مشغول کارهای پردازشی است .

Cycle Stealing (دزدیدن چرخه)

در هنگام اجرای یک دستور العمل ، دستورالعمل به زیر حلقه هایی (Sub Cycle) تقسیم می شود ودر زیرحلقه هایی که دستورالعمل به CPU نیاز ندارد ودستگاههای خارجی از CPU استفاده می کنند را Cycle Stealing می گویند .

تمرینات

۱- در روش ..... قبل از ارسال Data بین CPU وI/O آمادگی I/O برای برقراری این ارتباط توسط CPU تست می شود که به آن اصطلاحاً ..... گفته می شود.

۲- DMA چیست ؟ در مورد آن توضیح دهید؟

۳- آیا عبارت زیر درست می باشد ؟

- در روش DMA ارتباط دستگاه I/O با حافظی اصلی به طور مستقیم انجام می گیرد اما این ارتباط با مدیریت CPU می باشد . CPU درزمانی که به حافظه نیاز ندارد و تقاضای DMA وجود ندارد خود را از BUS جدا کرده واجزای ارتباط با حافظه را به I/O می دهد.

٤- آیا عبارت زیر درست می باشد؟

- به تقاضای وقفه در پایان Inst.Cycle توجه می شود ، اما به تقاضای DMA علاوه بر آن در طی یک Inst.Cycle نیز توجه می شود .

۵- توضیح دهید که CPU درچه مواقعی به تقاضای Interupt وDMA پاسخ می دهد؟

شبکه آموزشی- پژوهشی مادسیج
با هدف بهبود پیشرفت علمی
و دسترسی راحت به اطلاعات
برای جامعه بزرگ علمی ایران
ایجاد شده است

madsg.com

مادسیج

IRan Education & Research NETwork
(IRERNET)