مدرسان شریف

# فصل اول

## «نمایش داده‌ها و اطلاعات»

### مقدمه

نخستین مبحث در دنیای دیجیتال، نمایش اعداد به صورت دودویی می‌باشد. در این فصل به الگوهای مختلف نمایش اعداد و همچنین نشان دادن آن‌ها در مبناهای مختلف به ویژه مبناهای پرکاربرد در سیستم‌های دیجیتال خواهیم پرداخت. نمایش‌های ممیز ثابت و ممیز شناور به عنوان دو روش کلی و پرکاربرد در محاسبات درون پردازنده‌ها برای نمایش اعداد حقیقی به صورت کامل تشریح شده‌اند. الگوهای دیگری چون BCD و نمایش افزونگی نیز به عنوان روش‌هایی با کاربردهای خاص نیز بیان گردیده‌اند.

### مبنای اعداد

یکی از مفاهیم پایه‌ای در محاسبات کامپیوتری، چگونگی نمایش اعداد است. در حالت کلی عدد N در مبنای r را می‌توان به شکل $N=(b_{n-1}\cdots b_1b_0)_r$ نشان داد که n تعداد رقم‌ها، r مبنای نمایش عدد و هر یک از $b_i$ ها $(i=0,1,\ldots,n-1)$ یک رقم از N محسوب می‌گردد که باید کوچکتر از r باشد $(0\le b_i<r)$. در محاسبات روزمره از مبنای 10 ( Dec ) استفاده می‌شود بنابراین رقم‌ها بین صفر تا 9 قرار دارند، چنانچه عدد موردنظر در مبنای 10 باشد، از نوشتن مبنا خودداری می‌کنیم.

منظور از مقدار یک عدد مانند N، معادل آن عدد در مبنای 10 می‌باشد که بصورت زیر محاسبه می‌گردد:

$$N=\underbrace{(b_{n-1}b_{n-2}----b_1b_0)_r}_{\text{نمایش N در مبنای r}}\rightarrow \underbrace{b_{r-1}r^{n-1}+b_{n-2}r^{n-2}+....+b_1r+b_0}_{\text{مقدار عددی N}}=\sum_{i=0}^{n-1}b_ir^i$$

این روش، روش چندجمله‌ای (polynomial) نامیده می‌شود. به عنوان مثال چنانچه عدد موردنظر دودویی (باینری) باشد رابطه فوق به صورت زیر خواهد بود:

$$N=(b_{n-1}b_{n-1}\cdots b_1b_0)_2=b_{n-1}\times 2^{n-1}+\cdots+b_1\times 2^1+b_0=\sum_{i=0}^{n-1}b_ir^i \qquad (b_i=0 \text{ یا } b_i=1)$$

**تذکر۱:** درمبناهای بزرگتر از 10 به منظور جلوگیری از ایجاد ابهام، اعداد دو رقمی 10، 11، 12 و ... (که معادل یک رقم در مبناهای بزرگتر از 10 می‌باشند) به ترتیب با حروف A، B، C و ... نشان داده می‌شوند.

مثال‌های زیر چگونگی محاسبه معادل عددی (معادل مبنای 10) چند عدد را در مبناهای دیگر با استفاده از رابطه بالا نمایش می‌دهد.

$$(11101)_2 \xrightarrow{\text{مقدار عددی}} 1\times 2^4+1\times 2^3+1\times 2^2+0\times 2^1+1\times 2^0=16+8+4+0+1=29=(2\times 10^1+9\times 10^0)$$

$$(120)_8 \xrightarrow{\text{مقدار عددی}} 1\times 8^2+2\times 8^1+0\times 8^0=64+16+0=80=(8\times 10^1+0\times 10^0)$$

$$(2AF5)_{16} \xrightarrow{\text{مقدار عددی}} 2\times 16^3+\underbrace{10}_{A}\times 16^2+\underbrace{15}_{F}\times 16^1+5=10997=(1\times 10^4+0\times 10^3+9\times 10^2+9\times 10^1+7\times 10^0)$$

از آنجا که سیستم‌های کامپیوتری از یک منطق دودویی استفاده می‌کنند بنابراین مبنای دو (باینری) مبنای پرکاربرد در مطالب مربوط به ذخیره‌سازی کامپیوتری خواهد بود و علاوه بر این مبناهایی مانند مبنای 8 و 16 که به سادگی به مبنای 2 قابل تبدیل می‌باشند، نیز استفاده گسترده‌ای دارند.

به عنوان مثال چند عدد ابتدایی مبنای 10، 8 و 16 در زیر آمده است.

چند عدد ابتدایی مبنای 10 (decimal): 0 و 1 و 2 و 3 و 4 و 5 و 6و 7 و8 و9 و 10 و 11 و 12 و 13 و14 و 15 و16 و 17 و ...

چند عدد ابتدایی مبنای 8 (octal): 0 و 1 و 2 و 3 و 4 و 5 و 6و 7 و 10 و 11 و 12 و 13 و14 و 15 و16و17 و 20 ...

چند عدد ابتدایی مبنای 16 (hexadecimal): 0 و 1 و 2 و 3 و 4 و 5 و 6و 7 و8 و9 و A و B و C و D و E و F و 10 و 11 و 13 و ...

دقت کنید که در برخی از متن‌ها در عوض اعداد 2، 8، 10 و 16 به عنوان مبنای عدد به ترتیب از واژه‌های bin، oct، hex و dec استفاده می‌شود.

روش نمایش استفاده شده، روش وزن‌دار نمایش اعداد (weighted number system) نیز نامیده می‌شود زیرا ارزش رقم‌ها با توجه به مکان قرار گرفتن آنها در عدد موردنظر متفاوت است.

**تذکر۲:** در روش نمایش وزن‌دار، رقمی که بیشترین وزن را دارد MSB (Most Significant Bit) یا با ارزش‌ترین بیت نامیده می‌شود و رقمی که کمترین وزن را دارد LSB (Least Significant Bit) یا کم ارزش‌ترین بیت نامیده می‌شود. به عبارت دیگر MSB سمت چپ‌ترین بیت یک عدد و LSB سمت راست‌ترین بیت آن عدد می‌باشد.

## تبدیل از مبنای 10 به یک مبنای دلخواه

در این قسمت عکس عمل انجام شده برای محاسبه مقدار یک عدد مانند N انجام می‌شود. همانگونه که مشاهده شد مقدار N به صورت زیر قابل محاسبه است.

$$\text{مقدار } (N) = b_{n-1} \times r^{n-1} + b_{n-2} \times r^{n-2} + \ldots + b_1 r + b_0$$

حال اگر این مقدار عددی را بر r تقسیم کنیم خواهیم داشت:

$$\frac{b_{n-1}r^{n-1} + b_{n-2}r^{n-2} + \ldots + b_1 r + b_0}{r} = b_{n-1}r^{n-2} + b_{n-2}r^{n-3} + \ldots b_1 + \frac{b_0}{r}$$

بنابراین $b_0$ به عنوان باقیمانده تقسیم بدست می‌آید و خارج قسمت برابر با $b_{n-1}r^{n-2} + b_{n-2}r^{n-3} + \ldots + b_1$ خواهد بود بدیهی است که باقیمانده تقسیم برابر با سمت راست‌ترین رقم (LSB) خواهد بود. حال چنانچه خارج‌قسمت بدست آمده از مرحله اول را بر r تقسیم کنیم، باقیمانده جدید برابر $b_1$ خواهد بود. با تکرار این عمل تا n مرحله رقم‌های بعدی عدد در مبنای r (یعنی $b_i$ ها) بدست می‌آید. به بیان دیگر برای محاسبه تمام رقم‌ها کافیست این عمل را تا جایی ادامه دهیم که خارج قسمت بر r قابل تقسیم نباشد ( $r >$ خارج‌قسمت) که در این حالت آخرین رقم یعنی $b_{n-1}$ همان آخرین خارج‌قسمت بدست آمده خواهد بود. این روش، روش باقیمانده (remainder method) نامیده می‌شود، واضح است که در این روش رقم‌ها از کم ارزش‌ترین به سمت با ارزش‌ترین رقم بدست می‌آیند.

**مثال۱: عدد 23 را به مبنای 2 تبدیل کنید.**

$23 \div 2 = 11$ ... 1 ← LSB

$11 \div 2 = 5$ ... 1

$5 \div 2 = 2$ ... 1 ↑ جهت چیدمان رقم‌ها

$2 \div 2 = 1$ ... 0

$1 \div 2 = 0$ ... 1 ← MSB

ارزش کم ↓ ارزش زیاد

$\Rightarrow (23)10 = (10111)_2$

MSB LSB

---

**مثال۲: نمایش عدد 3740 در مبنای شانزده کدام است؟**

۱) EC9 ۲) 9EC ۳) EC ۴) E9C

☑ **پاسخ:** گزینه «۴»

$(3740)_{10} \Rightarrow$

| 3740 | 16 |
|---|---|
| ⑫ | 233 |
| | ⑨ |

| 233 | 16 |
|---|---|
| ⑨ | 14 |
| ⑭ | |

| 14 | 16 |
|---|---|
| ⑭ | 0 |

$\Rightarrow 14912 \Rightarrow \begin{cases} 12 \rightarrow C \\ 9 \\ 14 \rightarrow E \end{cases}$

بنابراین نمایش عدد 3740 در مبنای 16 عبارتست از $(E9C)_{Hex}$

**مثال ۳: مقدار 37 در مبنای دو کدام یک از موارد زیر می‌باشد؟**

۱) 100100　　۲) 100101　　۳) 101100　　۴) 111001

☑ **پاسخ:** گزینه «۲» برای این کار مقدار 37 را به طور متوالی بر عدد 2 تقسیم می‌کنیم.

| مقدار | خارج قسمت | باقیمانده |
|---|---|---|
| 37/2 | 18 | 1(LSB) |
| 18/2 | 9 | 0 |
| 9/2 | 4 | 1 |
| 4/2 | 2 | 0 |
| 2/2 | 1(MSB) | 0 |

جهت چیدمان

بنابراین: $37 = (100101)_{bin}$

## اعداد ممیز ثابت (Fixed Point Numbers)

در بخش قبل مفاهیم پایه‌ای مبنا شرح داده شد. نکته مهم دیگر در چگونگی نمایش اعداد، فرمت آنهاست. به صورت کلی می‌توان گفت که هر عدد در هر مبنایی دارای سه بخش کلی است:

۱) علامت　　۲) بخش صحیح　　۳) بخش اعشار

تقسیم‌بندی‌های نمایش اعداد بر حسب نحوه نمایش این سه بخش می‌باشد. در ادامه این شکل‌های نمایش بررسی شده است.

در سیستم ممیز ثابت تعداد ارقام هر عدد ثابت می‌باشد و نقطه اعشار نیز در یک مکان ثابت قرار می‌گیرد. به عنوان مثال اعداد 5.12 و 0.23 و 9.14 در مبنای 10 دارای سه رقم می‌باشند که نقطه اعشار در دومین مکان از سمت راست قرار دارد. اعداد 11.01، 10.01 و 00.11 اعداد چهار رقمی در مبنای دو می‌باشند که نقطه اعشار در دومین رقم از سمت راست آنها قرار گرفته است. تفاوت اساسی در مقدار ذخیره شده از یک عدد ممیز ثابت در کامپیوتر با آنچه بر روی کاغذ نوشته می‌شود، در این است که نقطهٔ اعشار واقعاً ذخیره نمی‌گردد بلکه به صورت فرضی در یک مکان خاص در نظر گرفته می‌شود و در حقیقت در ذهن طراح سخت‌افزار است! و در حافظهٔ کامپیوتر قرار ندارد. به بیان بهتر، استنتاج کاربر یا پیاده‌ساز از اعداد به آنها معنی می‌دهد. به مثال پیش‌رو دقت کنید، فرض کنید در یک سیستم دیجیتال از سیستم ممیز ثابت برای انجام محاسبات استفاده می‌شود. در بخشی از این سیستم یک ضرب‌کننده داریم که ورودی‌ها و خروجی‌های آن در زیر نمایش داده شده است.


در این سیستم خروجی 11 بیتی است که بخش اعشار آن 10 بیت و صحیح آن یک بیت می‌باشد. ورودی‌ها 10 بیتی هستند که یک بیت بخش صحیح و 9 بیت بخش اعشار می‌باشد. بدیهی است که در یک ضرب‌کننده انتظار داریم تعداد بیت‌های خروجی در ضرب‌کننده برابر مجموع تعداد بیت‌های ورودی باشد. بر طبق این قاعده خروجی باید 20 بیتی باشد اما در این‌جا 11 بیتی است. احتمالاً در ذهن طراح محدودیت‌هایی اعمال گردیده است که سیستم را بدین صورت پیش‌بینی نموده است. شاید طراح مطمئن بوده است که حاصل ضرب دو عدد هرگز به عدد ۲ نمی‌رسد پس برای نمایش بخش صحیح یک بیت کافی است و بیت‌های پایین بخش اعشار نیز در تعیین دقت سیستم بی‌اثر هستند بنابراین آن‌ها را حذف نموده است. حال چنانچه ورودی‌ها و خروجی این سیستم را پیش روی یک فرد ناآشنا به سیستم قرار دهید، هیچ نتیجه‌ای از آنها نمی‌گیرد اما اگر فردی به مسایل فوق آشنا باشد، می‌تواند صحت عملکرد را بررسی کند زیرا به ورودی‌ها و خروجی به شکل زیر معنا می‌دهد.

| | ورودی/خروجی در حالت معمول | ورودی/خروجی از دیدگاه یک فرد آشنا | |
|---|---|---|---|
| inp1 | $a_9a_8\cdots a_1a_0$ | $a_9$ | $a_8a_7\cdots a_1a_0$ |
| inp2 | $b_9b_8\cdots b_1b_0$ | $b_9$ | $b_8b_7\cdots b_1b_0$ |
| outp | $c_{10}c_9\cdots c_1c_0$ | $c_{10}$ | $c_9c_8\cdots c_1c_0$ |
| | | بخش صحیح | بخش اعشار |

این مثال به خوبی نشان می‌دهد که استنتاج کاربر از اعداد نمایش داده شده به هر یک از ارقام معنا می‌دهد. حال به دو تعریف زیر دقت کنید.

❖ **تعریف:** بازه (range) : در یک سیستم نمایش ممیز ثابت فاصله بین کوچکترین و بزرگترین عدد قابل نمایش می‌باشد.

❖ **تعریف:** دقت (precision) : در یک سیستم نمایش ممیز ثابت برابر با فاصلهٔ بین دو عدد متوالی است.

با ذکر یک مثال مفهوم دو تعریف فوق را بیشتر بررسی می‌کنیم.

## تست‌های طبقه‌بندی شده فصل اول

۱ـ یک سیستم اعداد **balanced ternary** یک سیستم اعداد در پایه ۳ است که سه رقم **1 ,0** و **1−** (که با $\overline{1}$ نشان داده می‌شود) را دارد. عدد معادل دسیمال $35\frac{2}{9}$ در این سیستم برابر است با ... (علوم کامپیوتر ـ سراسری ۸۰)

۱) $11\overline{1}.01$ ۲) $11\overline{1}1.1\overline{1}$ ۳) $110\overline{1}.\overline{1}1$ ۴) $110\overline{1}.1\overline{1}$

۲ـ برای نمایش یک عدد دهدهی **30** رقمی به صورت باینری تقریباً چند بیت نیاز است؟ (علوم کامپیوتر ـ سراسری ۸۰)

۱) 30 ۲) 60 ۳) 90 ۴) 120

۳ـ کدام یک از اعداد دهدهی زیر نمایش دودویی دقیق دارند؟ (علوم کامپیوتر ـ سراسری ۸۱)

۱) 0.3 ۲) 0.2 ۳) 0.5 ۴) 0.1

۴ـ در یک کامپیوتر با قابلیت پردازش اعداد ممیز شناور، کلمه ۳۲ بیتی $b_1b_0 \ldots b_{31}\, b_{30}$ مشخص کننده مقدار عددی $2^S(\frac{1}{2}-b_{31})(1+\sum_{i=0}^{23}2^{i-24}b_i)$ می‌باشد که در آن $S=-64+\sum_{i=24}^{30}2^{i-24}b_i$ است. مقدار کوچکترین و بزرگترین عدد مثبت قابل نمایش در این ماشین برابرند با:

(مهندسی کامپیوتر ـ سراسری ۸۲)

۱) $2^{-65}\ ,\ 2^{64}-1$ ۲) $2^{-65}\ ,\ (1-2^{-25})2^{63}$ ۳) $2^{-63},(1-2^{-25})2^{64}$ ۴) $2^{-64},(1-2^{-24})2^{63}$

۵ـ در مورد اعداد $x=10000$ و $y=11110000$ که به صورت مکمل 2 نمایش داده شده‌اند کدام جمله صحیح است؟ (علوم کامپیوتر ـ سراسری ۸۲)

۱) حاصل جمع x و y برابر صفر خواهد بود.
۲) x نشان‌دهندهٔ عدد 16 و y نشان‌دهندهٔ عدد 240 است.
۳) هر دو نشان‌دهندهٔ عدد $-16$ هستند.
۴) هر دو نشان‌دهندهٔ عدد 16 هستند.

۶ـ عدد ممیز شناور معادل عدد **23.1−** در استاندارد **IEEE** به صورت هگزادسیمال کدام است؟ (علوم کامپیوتر ـ سراسری ۸۲)

۱) 41B8CCCC ۲) ClB8CCCC ۳) C1D66666 ۴) 8238CCCC

۷ـ اگر از روش مکمل 2 برای نمایش اعداد منفی استفاده شود، حاصل عبارت **1100 + 001100** چیست؟ (علوم کامپیوتر ـ سراسری ۸۳)

۱) $-16$ ۲) 0 ۳) 8 ۴) 24

۸ـ در نمایش اعداد **floating – point** وقتی که بخش توان **16** بیت است **bias** چه مقداری باید باشد؟ (علوم کامپیوتر ـ سراسری ۸۵)

۱) $-1$ ۲) صفر ۳) 32767 ۴) 65535

۹ـ در یک سیستم اعداد ممیز شناور **16** بیتی هستند. مقدار عددی یک عدد ممیز شناور با نمایش بیتی $b_1b_0 \ldots b_{15}$ برابر است با:

$$\left(b_{15}-\frac{1}{4}\right)\times M\times 2^S$$

$$S=\sum_{i=10}^{14}b_i(-2)^{i-10}\ ,M=\sum_{i=0}^{9}b_i\left(-\frac{1}{2}\right)^{10-i}$$

کوچکترین و بزرگترین عدد قابل نمایش در این سیستم به ترتیب برابرند با: (مهندسی کامپیوتر ـ سراسری ۸۶)

۱) $\left(2^{18}+2^{16}+2^{14}+2^{12}+2^{10}\right),-\left(2^{17}+2^{15}+2^{13}+2^{11}\right)$
۲) $\left(2^{18}+2^{16}+2^{14}+2^{12}+2^{10}\right),-3\left(2^{18}+2^{16}+2^{14}+2^{12}+2^{10}\right)$
۳) $3\left(2^{17}+2^{15}+2^{13}+2^{11}\right),-\left(2^{17}+2^{15}+2^{13}+2^{11}\right)$
۴) $3\left(2^{17}+2^{15}+2^{13}+2^{11}\right),-3\left(2^{18}+2^{16}+2^{14}+2^{12}+2^{10}\right)$

۱۰ـ عدد **16** بیتی **0111111111100000** را در نظر بگیرید. بزرگترین عددی که می‌توان با این عدد جمع کرد به‌طوری که **overflow** رخ ندهد، چیست؟ (علوم کامپیوتر ـ سراسری ۸۶)

۱) $2^5$ ۲) $2^5-1$ ۳) $2^{15}-1$ ۴) $2^{15}$

۱۱ـ در نمایش اعداد ممیز شناور اگر تعداد بیت‌های در نظر گرفته شده برای بخش توان، **5** بیت باشد، بایاس چه مقداری دارد؟

(علوم کامپیوتر ـ سراسری ۸۶)

۱) 15 ۲) 16 ۳) 31 ۴) 32

**۱۲ـ در یک سیستم نمایش اعداد ممیز شناور، 1 بیت برای علامت 8 بیت برای توان و 23 بیت برای مانتیس در نظر گرفته شده است. نمایش ممیز شناور عدد مقابل در این سیستم چیست؟** $-0.00000000000000000000001111111\times 2^{(-110)}10$ (علوم کامپیوتر ـ سراسری ۸۶)

۱) $-0.11111110000000000000000\times 2^{(-131)}10$
۲) $-0.00011111110000000000000\times 2^{(-128)}10$
۳) $1.111111\times 2^{(-132)}10$
۴) $-0.1111111\times 2^{(-131)}10$

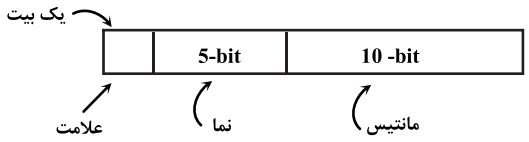
**۱۳ـ اگر سیستم نمایش اعداد ممیز شناور پایه 4 باشد آنگاه ..... نمی‌توانیم داشته باشیم.** (علوم کامپیوتر ـ سراسری ۸۶)

۱) اعداد نرمالیزه　۲) اعداد منفی　۳) بایاس　۴) hidden bit

**۱۴ـ در یک سیستم حسابی به جای استفاده از خود اعداد از log (لگاریتم) اعداد استفاده می‌شود. کدام‌یک از اعمال چهارگانه حسابی سخت‌تر انجام می‌شود؟** (مهندسی کامپیوتر ـ آزاد ۸۶)

۱) تقسیم　۲) ضرب　۳) ضرب و تقسیم　۴) جمع

**۱۵ـ محدوده‌ی دقت نمایش اعداد ممیز شناور با ساختار زیر را نشان دهید. فرض کنید که اعداد علامت‌دار نما را با مقدار ۱۶ بایاس می‌کنیم.**
(مهندسی کامپیوتر ـ سراسری ۸۷)




۱) $2^{-26} \le \text{دقت} \le 2^{5}$
۲) $2^{-15} \le \text{دقت} \le 2^{-10}$
۳) $2^{-5} \le \text{دقت} \le 2^{10}$
۴) $2^{-1} \le \text{دقت} \le 2^{-10}$

**۱۶ـ کدام یک از جملات زیر صحیح است؟** (علوم کامپیوتر ـ سراسری ۸۷)

۱) هر shift به راست حکم یک عمل تقسیم به 2 را دارد.
۲) هر shift به راست حکم یک عمل تقسیم به 2 را دارد فقط برای اعداد توان دو.
۳) هر rotate به راست حکم یک عمل تقسیم به 2 را دارد فقط برای اعداد توان دو.
۴) هر rotate به راست حکم یک عمل تقسیم به 2 را دارد.

**۱۷ـ اگر مجموعه ارقام مورد نظر** $\{0,1,2\}_{(2)}$ **باشد، یعنی این که عدد 2 را نیز در مبنای 2 بتوان پذیرفت. کدام‌یک از اعداد زیر عدد 8 در مبنای 10 را نشان می‌دهد؟** (مهندسی کامپیوتر ـ آزاد ۸۷)

۱) 202　۲) 122　۳) 120　۴) 210

**۱۸ـ در یک کامپیوتر اعداد ممیز شناور به صورت 32 بیتی ذخیره می‌شوند. مقدار عددی رشته 32 بیتی** $b_{31}b_{30}\cdots b_1b_0$ **برابر است با:**

$$2^{E-64}\times(2\times b_{31}-1)\times\sum_{i=0}^{23}(\bar{b}_i\times 2^{i-12}) \quad ; \quad E=\sum_{i=24}^{30}(2^{i-12}\times b_i)$$

**مقداری عددی کوچکترین عدد مثبت و کوچکترین عدد منفی قابل نمایش چقدر است؟** (مهندسی کامپیوتر ـ سراسری ۸۸)

۱) $-2^{51}, 2^{-64}$
۲) $-2^{75}, 2^{-76}$
۳) $2^{51}-2^{75}, 2^{-76}$
۴) $-2^{63}, 2^{-64}$

**۱۹ـ در یک پردازنده برای نمایش قسمت توان اعداد ممیز شناور از روش** $excess-64$ **استفاده می‌شود. در این روش عدد** $e_6e_5...e_0$ **به وسیله عدد** $e=-64+\sum_{i=0}^{6}2^i e_i$ **نمایش داده می‌شود. دو عدد این چنینی وارد یک جمع کننده موازی معمولی 7 بیتی می‌شود. کدام عمل باید انجام شود تا نتیجه نیز به صورت** $excess-64$ **باشد؟** (علوم کامپیوتر ـ سراسری ۸۹)

۱) خروجی باید مکمل شود.
۲) خروجی بدون تغییر بماند.
۳) بیت منتهی‌الیه سمت چپ (هفتمین بیت) خروجی جمع‌کننده باید مکمل شود.
۴) بیت carry نتیجه باید جزء نتیجه جمع تلقی گردد.

**۲۰ـ چند عدد بین دو عدد 255 و 256 وجود دارند که به صورت ممیز شناور قابل نمایش هستند؟** (علوم کامپیوتر ـ سراسری ۹۰)

۱) یک
۲) به تعداد $2^{23}$ عدد
۳) بین 255 و 256 عددی وجود ندارد.
۴) بی‌نهایت

**۲۱ـ عدد ممیز شناور معادل عدد هگزا دسیمال 3EA 80000 در استاندارد IEEE برای نمایش اعداد floating point کدام است؟**
(علوم کامپیوتر ـ سراسری ۹۰)

۱) 0.3125　۲) 0.328125　۳) 1.3125　۴) 1.328125

**۲۲ـ برای تعیین عدد بزرگ‌تر در میان دو عدد ممیز شناور هنجار شده دارای قسمت‌های «علامت»، «مانتیس» و «نما» ترتیب انجام مقایسه چه می‌باشد؟** (مهندسی کامپیوتر ـ آزاد ۹۰)

۱) علامت، مانتیس، نما ۲) مانتیس، نما، علامت ۳) علامت، نما، مانتیس ۴) نما، مانتیس، علامت

**۲۳ـ معادل ده دهی حاصل جمع دو عدد علامت‌دار متمم دو $10001.001,011.01$ چه می‌باشد؟** (مهندسی کامپیوتر ـ آزاد ۹۰)

۱) $-3.625$ ۲) $-4.375$ ۳) $-11.625$ ۴) $-20.375$

**۲۴ـ در کدام مبنای عددنویسی دو دو تا به‌جای 4 می‌شود 11؟** (مهندسی کامپیوتر ـ آزاد ۹۰)

۱) 2 ۲) 3 ۳) 4 ۴) 5

## پاسخنامه تست‌های طبقه‌بندی شده فصل اول

**۱ـ گزینه «۴»** دارا بودن ارقام $1,0,1-$ به معنای آن است که ما باید تمامی اعداد را در این سیستم با این سه رقم و توان‌هایی از 3 نمایش دهیم. در این جا بر خلاف حالت‌های معمولی امکان تفریق به دلیل داشتن $1-$ اضافه شده است. بنابراین برای حل مسأله باید تلاش کنیم تمامی توان‌های 3 را از درون عدد موردنظر استخراج کنیم تا آن که به عددی بزرگتر از عدد موردنظر برسیم. حال تعداد اختلاف را باید با سایر توان‌های 3 و علامت منفی جبران نماییم.

$$\left.\begin{array}{l} 35 = 3^3 + 3^2 + 0\times 3^1 - 1\times 3^0 = 110\,\bar{1} \\ \frac{2}{9} = \frac{1}{3} - \frac{1}{3^2} = 3^{-1} - 3^{-2} = 11 \end{array}\right\} \Rightarrow 35\frac{2}{9} = 110\,\bar{1}.1\,\bar{1}$$

**۲ـ گزینه «۴» روش اول:** $2^n - 1 \Rightarrow 2^n - 1 \geq 10^{30} - 1$ بزرگترین عدد قابل نمایش با $n$ بیت

$10^{30} - 1 =$ بزرگترین عدد 30 رقمی دهدهی

$$\Rightarrow 2^n \geq 10^{30} \Rightarrow \log 2^n \geq 30\log 10 \Rightarrow n\log 2 \geq 30 \Rightarrow n \geq \frac{30}{0/3} \simeq 100$$

پس به حداقل 100 بیت نیازمندیم.

**روش دوم:** $n = \log_2^A \Rightarrow n = \log_2^{10^{30}} = 30\log_2^{10} \simeq 30\times 3.3 \simeq 100$ = تعداد بیت‌های لازم برای نمایش عدد (A) به صورت کلی در مبنای ۲

**۳ـ گزینه «۳»** $0.5\times 2 = 1.0 \rightarrow (0.5)_{10} = (0.1)_2$ حجم حافظه نهان

شرط لازم برای مقدار دقیق دودویی داشتن، این است که رقم سمت راست 5 یا صفر باشد.

**۴ـ گزینه «۲»** در فرمت نشان داده شده $b_{31}$ بیت علامت است و بیتهای $b_0$ تا $b_{23}$ بیت‌های مانتیس و بیت‌های $b_{24}$ تا $b_{30}$ بیت نما هستند. اگر مانتیس و نما هر دو بیشترین مقدار باشند، مقدار عدد بزرگترین مقدار ممکن خواهد بود و کمترین مقدار عدد زمانی بدست می‌آید که مانتیس ونما هر دو کوچکترین مقدار را داشته باشند.

$$S_{min} = -64 + 0 = -64 \quad ; \quad S_{max} = -64 + 1 + 2 + 2^2 + \ldots + 2^5 + 2^6 = 63$$

$= 2^{63}(\frac{1}{2} - 0)(1 + 1 - 2^{-24}) = 2^{63}\times\frac{1}{2}(2 - 2^{-24}) = 2^{63}\times(1 - 2^{-25})$ بزرگترین عدد مثبت

$= 2^{-64}(\frac{1}{2} - 0)(1 + 0) = 2^{-65}$ کوچکترین عدد مثبت

**۵ـ گزینه «۳»** مقدار دهدهی یک عدد در سیستم مکمل ۲ بدین صورت است:

$$(a_{n-1}\ a_{n-2}\ldots a_1 a_\circ)_2 = a_{n-1}\times(-1)^{a_{n-1}}\times 2^{n-1} + a_{n-2}2^{n-2} + a_{n-1}2^{n-1} + \cdots a_1 2^1 + a_\circ 2^\circ$$

$$x = 10000 = 1\times 2^4 + (-1)^1 + 0\times 2^3 + 0\times 2^1 + 0\times 2^0 = -16$$

$$y = 11110000 = 1\times 2^7\times(-1)^1 + 1\times 2^6 + 1\times 2^5 + 1\times 2^4 = -128 + 64 + 32 + 16 = -16$$

**نکته:** چنانچه به تعداد دلخواه بیت علامت را در بیت‌های با ارزش یک عدد مکمل 2 تکرار کنیم، مقدار عدد تغییر نمی‌کند. به این عمل توسعه علامت (sign extension) می‌گویند.

مدرسان شریف

# فصل دوم

## «محاسبات کامپیوتری و الگوریتم‌های محاسباتی»

### مقدمه

در فصل پیش الگوهای نمایش متداول اعداد در سیستم‌های کامپیوتری تشریح گردید. در این فصل چگونگی انجام چهار عمل اصلی جمع، تفریق، ضرب و تقسیم در این الگوهای نمایش بیان می‌گردد. در بخش مدارات جمع یا تفریق کننده چند ساختار متداول استفاده شده، معرفی و مقایسه می‌گردند. در بخش ضرب‌کننده روش پرکاربرد بوث به صورت کامل تشریح می‌شود. در برخی موارد مدارات سخت‌افزاری که اعمال فوق را انجام می‌دهند نیز ارایه گردیده است. در بخش تشریح هر یک از چهار عملگر، دو بخش اعداد علامت دارد و بدون علامت در صورت لزوم به تفکیک تشریح شده‌اند.

در فصل قبل نحوه نمایش اطلاعات و داده‌ها در سیستم‌های کامپیوتری مورد بحث قرار گرفت در این فصل انجام محاسبات ریاضی مانند جمع، تفریق، ضرب و تقسیم بر روی اعداد ممیز ثابت و ممیز شناور مورد بررسی قرار خواهد گرفت و انجام محاسبات بر روی اعداد ذخیره شده به صورت مکمل 1 و مکمل 2 بحث خواهد شد اما تاکید بیشتر بر روی نحوهٔ ذخیره سازی مکمل 2 می‌باشد که در کامپیوترهای امروزی بیشتر استفاده می‌گردد.

### جمع و تفریق اعداد ممیز ثابت

**اعداد مکمل 2:**

در حالت کلی می‌توانیم تفریق دو عدد را به صورت روبرو در نظر بگیریم: $a - b = a + (-b)$

بنابراین برای انجام این تفریق کافیست مکمل 2 عدد b محاسبه گردد و حاصل با عدد a جمع شود (و در نتیجه می‌توان گفت برای عمل تفریق به سخت‌افزار اضافه نیاز نخواهیم داشت).

**مثال۱: عملیات جمع و تفریق مقابل را با استفاده از نمایش مکمل 2 در یک قالب یک بایتی انجام دهید.** **(الف 10+23** **(ب 2-5**

**پاسخ:** برای انجام 10+23 کافیست معادل مبنای دو هر یک از اعداد را نوشته، با یکدیگر جمع نماییم:

$$\begin{array}{l} (+10): \ 00001010 \\ +(+23): \ 00010111 \\ \hline 00100001 \Rightarrow +33 \end{array}$$

برای انجام 2-5 ابتدا مکمل 2 عدد 2 را محاسبه نموده و با 5 جمع می‌نماییم:

$$5 - 2 = 5 + (-2):$$

$$+2: 00000010 \xrightarrow{\text{مکمل 2}} -2: 11111110$$

$$\begin{array}{l} (-2): 11111110 \\ +(+5): \ 00000101 \\ \hline (1)\ 00000011 \Rightarrow +3 \end{array}$$

رقم نقلی صرف‌نظر

دقت کنید که در انجام عمل جمع در روش مکمل 2 از رقم نقلی ایجاد شده صرف‌نظر می‌گردد.

با توجه به محدودیت تعداد بیت‌های استفاده شده، نمی‌توان بازه $[-\infty, +\infty]$ را در محاسبات پوشش داد بنابراین در انجام برخی از محاسبات ممکن است سرریزی (over flow) رخ دهد. نکته زیر شرط بروز سرریزی را در عمل جمع مکمل 2 مشخص می‌کند.

**نکته۱:** در عمل جمع در حالتی که اعداد به صورت مکمل 2 ذخیره شده باشند تنها در صورتی سرریزی می‌تواند رخ دهد که علامت عملوندها یکسان باشد، در این حالت اگر بیت علامت نتیجه با بیت علامت عملوندها یکسان نباشد آنگاه می‌توان بروز سرریزی را نتیجه‌گیری نمود و در غیر این صورت نتیجه به دست آمده صحیح می‌باشد. به عبارت دیگر اگر جمع دو عدد مثبت، منفی یا جمع دو عدد منفی، مثبت شود، سرریزی اتفاق افتاده است.

**تذکر۱:** سرریزی به حالتی گفته می‌شود که در انجام محاسبات، عدد حاصل قابل ذخیره‌سازی نباشد.

**تذکر۲:** در حالت کلی اگر از n بیت برای نمایش داده‌های مکمل 2 استفاده شود آنگاه در صورتی که حاصل جمع در بازهٔ $[-2^{n-1}, 2^{n-1}-1]$ قرار نداشته باشد، سرریزی رخ خواهد داد.

**مثال۲: اگر از یک فرمت هشت بیتی برای نمایش اعداد مکمل 2 استفاده شده باشد آیا برای محاسبه (-4)+(-1) سرریزی رخ می‌دهد؟**

**پاسخ:**

$$(-1): 11111111$$
$$+(-4): 11111100$$
$$1111110111 \Rightarrow -5$$

1 ← رقم نقلی صرف‌نظر می‌شود

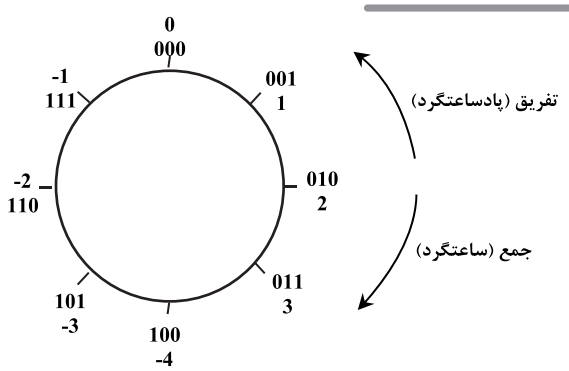
↑ بیت علامت

به دلیل یکسان بودن بیت علامت نتیجه و بیت علامت عملوندها سرریزی رخ نداده است.

---

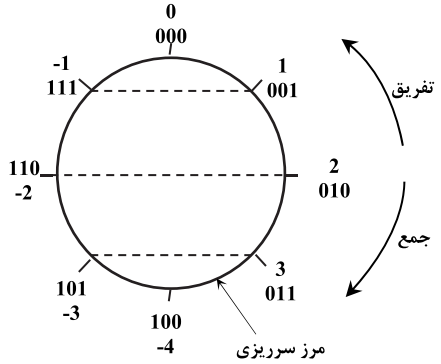
اگر از سه بیت برای نمایش اعداد مکمل 2 استفاده شده باشد آنگاه بازه قابل نمایش برابر $[-2^{3-1}, 2^{3-1}-1] = [-4, +3]$ خواهد بود که می‌توان نمودار ساعتی روبرو را برای آن در نظر گرفت.

این شکل دلیل صرف‌نظر نمودن از رقم نقلی را بهتر نشان می‌دهد (تنها حالت‌های محدودی قابل نمایش می‌باشند که با اضافه نمودن یک واحد به بزرگترین عدد کوچکترین عدد بدست می‌آید).

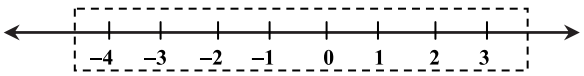



**نکته۲:** همان‌طور که از روی شکل پیداست همواره کوچک‌ترین عدد منفی و بزرگ‌ترین عدد مثبت بر روی دایره کنار هم قرار می‌گیرند، عبور از این همسایگی مرز وقوع خطای سرریز را نشان می‌دهد زیرا با انجام یک عمل جمع و یا تفریق از بزرگ‌ترین عدد مثبت به کوچک‌ترین عدد منفی رسیده‌ایم که نمی‌تواند صحیح باشد.

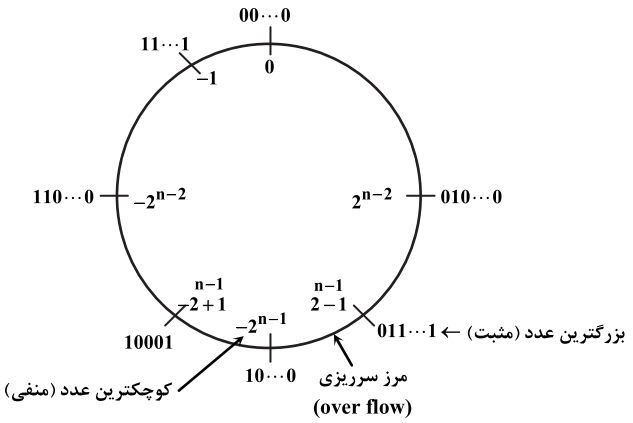
حال اگر نمودار ساعتی داده شده را به صورت روبه‌رو در نظر بگیریم، آنگاه مکمل 2 هر عدد دقیقاً عددی است که با خط‌چین افقی به آن متصل شده است و طبق مطلب بیان شده، برای اضافه نمودن m واحد به یک عدد کافیست با شروع از آن عدد به اندازه m واحد در جهت حرکت عقربه‌های ساعت حرکت نماییم و برای تفریق m واحد همین عمل در جهت عکس حرکت عقربه‌های ساعت انجام می‌گیرد. اما نتیجه به دست آمده در صورتی که از مرز بین 100 و 011 عبور نماییم، درست نخواهد بود (زیرا سرریزی رخ می‌دهد). به عنوان مثال اگر به عدد یک، چهار واحد اضافه شود و در نتیجه در جهت عقربه‌های ساعت چهار واحد حرکت کنیم آنگاه از مرز ذکر شده عبور می‌کنیم و در نهایت به عدد 3- خواهیم رسید که مشخصاً نتیجه درست نمی‌باشد.



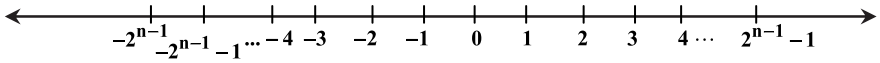

بنابراین می‌توان گفت که یک نمایش سه بیتی در نمایش مکمل 2 بازه زیر را نمایش می‌دهد:




در حالت کلی می‌توان شکل زیر را برای اعداد قابل نمایش در روش مکمل 2 بوسیله n بیت به صورت زیر در نظر گرفت:



که بازه قابل نمایش را می‌توان به صورت زیر در نظر گرفت:




**نکته۳:** اگر رقم نقلی وارد شده به MSB (با ارزش‌ترین بیت) با رقم نقلی خارج شده از آن مساوی نباشد، سرریز رخ داده است.

این موضوع به صورت بهتر در زیر بیان شده است.

$$\begin{array}{l} c_n\ c_{n-1}c_{n-2}...c_2\ c_1\ c_0 \rightarrow \text{رقم‌های نقلی} \\ \quad a_{n-1}a_{n-2}...a_2\ a_1\ a_0 \\ + \quad b_{n-1}b_{n-2}...b_2\ b_1\ b_0 \\ \hline s_{n-1}s_{n-2}...s_2\ s_1\ s_0 \rightarrow \text{رقم‌های حاصل‌جمع} \end{array}$$

آنگاه می‌توان بروز سرریزی را به صورت زیر کنترل نمود:

$\underset{\text{بیت سرریز}}{V} = C_n \oplus C_{n-1} \Rightarrow$ اگر $C_n \neq C_{n-1} : V = 1$ سرریزی رخ داده است

**مثال۳: اگر از یک فرمت یک بایتی برای نمایش مکمل 2 استفاده شده باشد آیا در عمل 50+80 در این سیستم سرریزی رخ می‌دهد؟**

☑ **پاسخ:** حاصل جمع دو عدد داده شده به صورت زیر انجام می‌شود.

$$\begin{array}{l} (+80): 01010000 \\ +(+50): 00110010 \\ \hline 10000010 \Rightarrow -126 \end{array}$$

بیت علامت

مطابق آنچه در بالا گفته شد $\left.\begin{array}{l} C_7 = 1 \\ C_8 = 0 \end{array}\right\} \rightarrow V = 1$

همان‌گونه که ملاحظه می‌گردد بیت علامت دو عملوند داده شده یکسان و مثبت است اما بیت علامت نتیجه متفاوت و منفی می‌باشد بنابراین سرریزی رخ داده است و نتیجه بدست آمده معتبر نیست.

---

**مثال۴: محاسبات ریاضی زیر را در سیستم مکمل 2 و با نمایش ۷ بیتی اعداد انجام دهید. مشخص کنید که آیا سر ریز رخ داده است یا خیر؟**

**الف)** $(+40)+(+35)$ **ب)** $(-40)+(-35)$

☑ **پاسخ:**

الف)

$$\begin{array}{ll} +35 & 0\ 100011 \\ +40 & 0\ 101000 \\ \hline 75 & 1\ 001011 \end{array}$$

$C_6 = 0 \quad C_5 = 1$

سرریزی رخ داده است $C_6 \oplus C_5 = 1$ رقم‌های نقلی

ب)

$$\begin{array}{ll} -35 & 1\ 011101 \\ -40 & 1\ 011000 \\ \hline -75 & 0\ 110101 \end{array}$$

$C_6 = 0 \quad C_5 = 1$

سرریزی رخ داده است $C_6 \oplus C_5 = 1$ رقم‌های نقلی

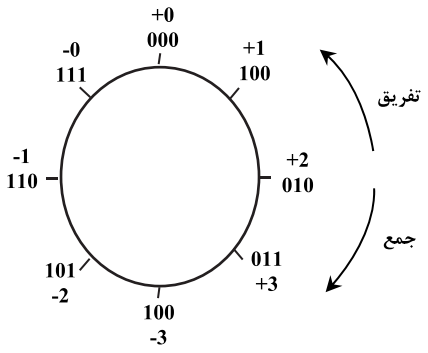
البته می‌توان به روش دیگر نیز رخ دادن سرریز را بدون محاسبه در مبنای دو پیش‌بینی کرد. مطابق با آنچه گفته شد در یک نمایش ۷ بیتی، اعداد در محدوده $[-2^{7-1}, 2^{7-1}-1]$ یعنی $[-64, 63]$ قابل نمایش می‌باشد. از آن‌جا که حاصل هر دو عبارت در خارج از این محدوده قرار دارد، پس در هر دو مورد سرریز رخ می‌دهد.

---

**اعداد مکمل 1:**

برای جمع و تفریق دو عدد مکمل 1 شبیه به حالت قبل می‌باشد با این تفاوت که برای اعداد منفی مکمل 1 آنها محاسبه می‌گردد. اگر از سه بیت برای نمایش اعداد مکمل یک استفاده شده باشد، نمودار ساعتی روبرو اعداد قابل نمایش را نشان می‌دهد.

همانگونه که در این شکل مشاهده می‌گردد اگر یک واحد به 0- اضافه شود آنگاه در صورتی که از رقم نقلی صرف نظر گردد حاصل 0+ می‌شود اما در صورتی که حاصل باید برابر 1+ باشد بنابراین در جمع دو عدد مکمل 1 در صورتی که رقم نقلی ایجاد شود آنگاه یک واحد به نتیجه خروجی اضافه می‌گردد.



**مثال۵: حاصل 5.5-1.0 را در روش مکمل 1 بدست آورید.**

همان‌طور که گفته شد، در روش مکمل یک باید رقم نقلی به نتیجه خروجی اضافه گردد.

$$(+5.5): 0101.1$$
$$(-1.0): 1110.0$$
$$0011.1$$
$$+ \quad 1.0$$
$$0100.1 \Rightarrow +4.5$$

(1: رقم نقلی)

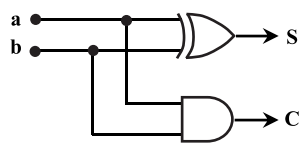
## سخت‌افزار مورد نیاز برای عمل جمع و تفریق

**نیم جمع‌کننده (Half Adder):** یک نیم جمع‌کننده برای جمع دو رقم باینری مانند a و b استفاده می‌گردد که دو بیت خروجی حاصل‌جمع (S) و رقم نقلی (carry) را ایجاد می‌نماید جدول درستی یک‌نیم جمع‌کننده به صورت مقابل می‌باشد.

| a | b | C (Carry) | S |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

بنابراین می‌توان نوشت:

$$\begin{cases} S = a \oplus b \\ C = a.b \end{cases}$$




در نتیجه نمودار منطقی نیم جمع‌کننده به صورت روبرو خواهد بود.

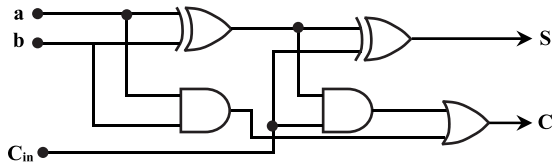
**تمام جمع‌کننده (Full Adder):** یک تمام جمع‌کننده سه رقم باینری مانند a و b و $c_{in}$ را به عنوان ورودی دریافت می‌کند و دو بیت خروجی S و $C_{out}$ را به عنوان حاصل‌جمع و رقم نقلی ایجاد می‌نماید. جدول درستی یک تمام جمع‌کننده به صورت زیر می‌باشد.

| a | b | $C_{in}$ | C | S |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

جدول درستی (FA)

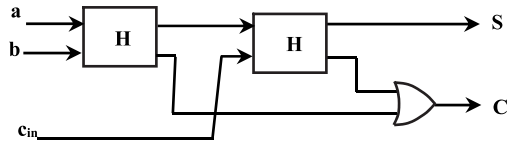
که با توجه به رابطه بین ورودیها و خروجی‌ها می‌توان نوشت.

$$\begin{cases} S = a \oplus b \oplus C_{in} \\ C = a.b + a.C_{in} + b.C_{in} \end{cases}$$

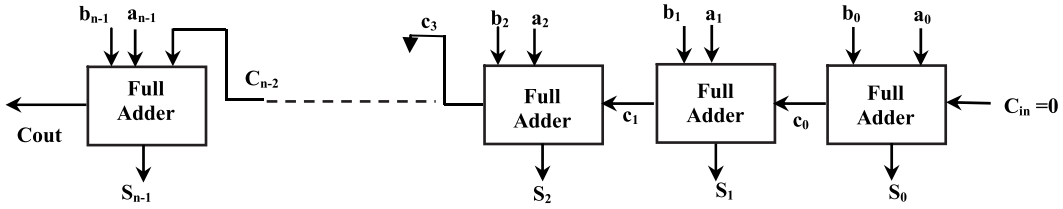



و بنابراین نمودار منطقی یک تمام جمع‌کننده به صورت روبه‌رو می‌باشد.

می‌توان با استفاده از دو نیم جمع‌کننده، یک تمام جمع‌کننده را به صورت زیر ایجاد نمود:




**جمع‌کننده با رقم نقلی موجی (ripple carry adder):** این جمع‌کننده برای جمع دو عدد n بیتی باینری استفاده می‌شود و از n تمام جمع‌کننده به صورت زیر تشکیل شده است.



که برای جمع دو عدد $A = a_{n-1}a_{n-2}\cdots a_0$ و $B = b_{n-1}b_{n-2}\cdots b_0$ در نظر گرفته شده است. در این جمع‌کننده بیت‌های A و B از سمت راست دو به دو با یکدیگر جمع می‌شوند و رقم نقلی به عنوان یکی از ورودیهای مربوط به تمام جمع‌کننده برای رقم بعدی استفاده می‌شود. در این جمع‌کننده ورودی $C_{in}$ مدار را می‌توان برابر صفر در نظر گرفت و یا می‌توان به جای اولین تمام جمع‌کننده از یک نیم جمع‌کننده استفاده نمود.

این جمع کننده به این دلیل رقم نقلی موجی نامیده می‌شود که در آن رقم نقلی جمع کننده iام به صورت موجی به جمع کننده $i+1$ام انتشار می‌یابد، بنابراین جمع کننده $i+1$ام تنها زمانی می‌تواند نتیجه بیت $i+1$ام حاصل‌جمع را محاسبه نماید که حاصل‌جمع کننده سمت راست آن (جمع‌کننده iام) نتیجه بیت iام را محاسبه نموده باشد و رقم نقلی خود را تولید کرده باشد.

**جمع‌کننده با پیش‌گویی رقم نقلی (CLA):**

هدف از طراحی جمع‌کننده با پیش‌گویی رقم نقلی (Carry Look-Ahead Adder) کاهش زمان تأخیر مربوط به انتشار رقم نقلی در جمع کننده موجی می‌باشد، ایده اصلی برای طراحی این جمع‌کننده بر اساس محاسبه مستقیم بیت $C_{in}$ (Carry -in) در جمع‌کننده $i+1$ام است (به جای اینکه بیت $C_{in}$ از جمع‌کننده iام گرفته شود).

در حالت کلی اگر از اعداد مبنای r استفاده شود آنگاه در جمع دو عدد $B = b_{n-1}....b_0$ و $A = a_{n-1}....a_0$ حاصل‌جمع $a_i$ و $b_i$ می‌توان دو حالت زیر را در نظر گرفت:

اگر $a_i + b_i \geq r \Rightarrow$ جمع‌کننده iام رقم نقلی تولید می‌کند

اگر $a_i + b_i = r-1 \Rightarrow$ رقم نقلی ورودی به جمع‌کننده iام به جمع‌کننده بعدی $(i+1)$ام منتشر می‌شود.

به عنوان مثال دو عدد در مبنای 10 را در نظر بگیرید.

$$A = \begin{matrix} a_2 & a_1 & a_0 \\ 1 & 4 & 5 \end{matrix} \quad , \quad B = \begin{matrix} b_2 & b_1 & b_0 \\ 1 & 5 & 5 \end{matrix}$$

در این حالت داریم:

$$\begin{array}{rr} & 145 \\ & +155 \\ \hline \text{رقم‌های نقلی} & 110 \\ \hline \text{نتیجه} & 300 \end{array}$$

حاصل‌جمع دو رقم اول به صورت $5+5 \geq 10$ می‌باشد بنابراین یک رقم نقلی ایجاد خواهد کرد، حاصل‌جمع دو رقم بعدی به صورت $4+5=10-1=9$ می‌باشد پس این دو رقم، رقم نقلی تولید نمی‌کنند اما رقم نقلی جمع قبلی را به رقم‌های سوم منتشر می‌کنند. حاصل‌جمع دو رقم نهایی ایجاد رقم نقلی نمی‌کند و هم‌چنین رقم نقلی ورودی را نیز منتشر نمی‌نماید.

در حالت باینری روابط ساده‌تر از مبناهای بزرگتر می‌باشد، برای تعیین این که آیا جمع رقم‌های iام از A و B ایجاد رقم نقلی می‌کند بیت $g_i$ را به عنوان تولید رقم نقلی مربوط به حاصل‌جمع $a_i$ و $b_i$ قرار می‌دهیم و $P_i$ را انتشار یا عدم انتشار رقم نقلی مربوط به جمع‌کننده iام در نظر می‌گیریم در این حالت خواهیم داشت:

$$\rightarrow g_i = a_i \text{ and } b_i = a_i \wedge b_i$$

$$\rightarrow p_i = a_i \text{ xOR } b_i = a_i \oplus b_i$$

بنابراین جمع‌کننده iام در صورتی یک رقم نقلی تولید می‌کند که هر دو بیت $a_i$ و $b_i$ برابر یک باشند و تنها در صورتی رقم نقلی ورودی را منتشر می‌کند که دقیقاً یکی از دو ورودی برابر یک باشند.

بنابراین با فرض این که $c_0$ رقم نقلی ورودی به کل مدار می‌باشد می‌توان رقم‌های نقلی دیگر را به صورت زیر پیش‌گویی نمود:

$$c_1 = g_0 \vee (p_0 \wedge c_0)$$

$$c_2 = g_1 \vee (p_1 \wedge c_1) = g_1 \vee (p_1 \wedge g_0) \vee (p_1 \wedge p_0 \wedge c_0)$$

$$c_3 = g_2 \vee (p_2 \wedge c_2) = g_2 \vee (p_2 \wedge g_1) \vee (p_2 \wedge p_1 \wedge g_0) \vee (p_2 \wedge p_1 \wedge p_0 \wedge c_0)$$

$$\vdots$$

می‌توان $p_i$ در مبنای دو را به صورت $p_i = a_i + b_i$ در نظر گرفت و رابط مربوط به رقم نقلی $C_i$ را به صورت زیر نوشت:

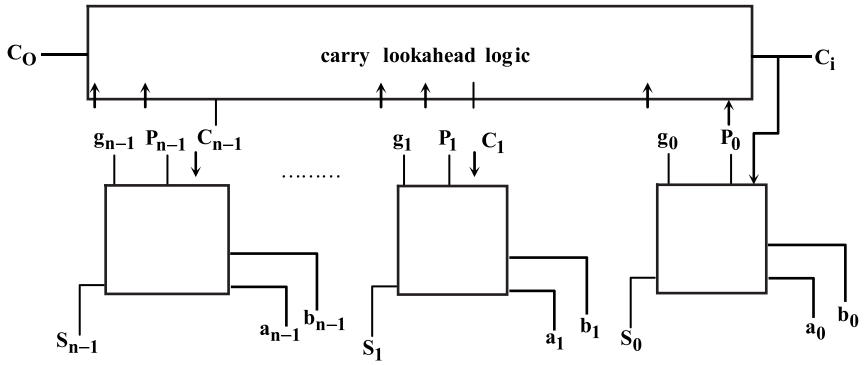
$$c_i = g_{i-1} + p_{i-1} \quad , \quad g_{i-1} = a_{i-1}b_{i-1} \quad , \quad p_{i-1} = a_{i-1} \oplus b_{i-1}$$

بنابراین داریم:

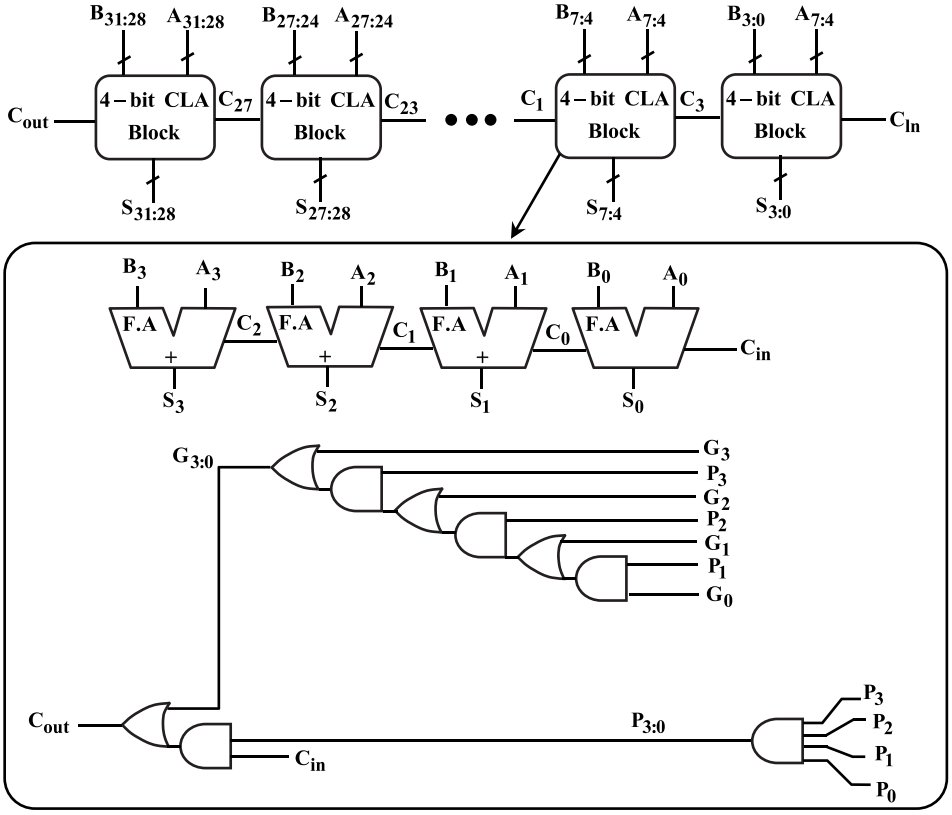
$$c_i = g_{i-1} + p_{i-2} + p_{i-1}p_{i-2}g_{i-3} + \cdots + p_{i-1}p_{i-2}\cdots p_1 g_\circ + g_{i-2}\cdots p_1 p_0 c_0$$

همان‌گونه که ملاحظه می‌گردد، محاسبه مستقیم رقم‌های نقلی باارزش‌تر نیاز به اعمال پیچیده‌تری دارد. با این حال استفاده از روش بیان شده به صورت قابل ملاحظه‌ای زمان تأخیر انتشار رقم نقلی را کاهش می‌دهد.

شکل زیر مدار مربوط به یک جمع‌کننده $n$ بیتی با رقم نقلی پیش‌بینی شده را نشان می‌دهد.




در حالت کلی می‌توان گفت عمق یک مدار جمع‌کننده موجی $n$ بیتی برابر $o(n)$ و عمق یک جمع کننده $n$ بیتی با رقم نقلی پیش‌گویی شده برابر $o(\log n)$ می‌باشد اما در عین حال مدار جمع‌کننده با رقم نقلی پیش‌گویی شده $o(n\log n)$ گیت استفاده می‌نماید در حالیکه جمع‌کننده با رقم نقلی با رقم نقلی موجی از $o(n)$ مدار استفاده می‌نماید. با توجه به این مطلب در برخی از طراحی‌ها یک جمع‌کننده بزرگ با رقم نقلی موجی در نظر گرفته می‌شود که در داخل آن از جمع‌کننده‌های کوچک با رقم نقلی پیش‌بینی شده استفاده می‌گردد.




توجه به این نکته لازم است که در هر بلوک، رقم نقلی بین ۴ تمام جمع کننده به صورت موجی منتشر می‌شود و تنها رقم خروجی بلوک پیش‌بینی می‌گردد.

**نکته۴:** عمق مدار جمع‌کننده $n$ بیتی با رقم نقلی پیش‌بینی شده برابر $o(\log n)$ و عمق مدار جمع‌کننده با رقم نقلی موجی $o(n)$ می‌باشد.

**نکته۵:** تعداد گیت استفاده شده در مدار جمع‌کننده $n$ بیتی با رقم نقلی پیش‌بینی شده از مرتبه $o(n\log n)$ و در مدار جمع‌کننده $n$ ـ بیتی با رقم نقلی موجی برابر $o(n)$ می‌باشد.

زمان تأخیر یک جمع‌کننده با رقم نقلی موجی به صورت زیر قابل محاسبه می‌باشد:

$$t_{ripple} = \underbrace{N}_{\text{تعداد بیت‌ها}} \times \underbrace{t_{FA}}_{\text{زمان تأخیر هر تمام جمع‌کننده}}$$

همان‌گونه که ملاحظه می‌شود زمان تأخیر این جمع‌کننده نسبت مستقیم با تعداد بیت‌ها دارد.

برای محاسبهٔ زمان تأخیر در جمع‌کننده با رقم نقلی پیش‌بینی شده باید مسیر بحرانی (مسیر بحرانی مسیری است که بیشترین تأخیر را دارد و بنابراین تأخیر کل مدار را تعیین می‌کند)، شناسایی می‌شود. اگر فرض کنیم یک جمع‌کننده $N$ بیتی با رقم نقلی پیش‌بینی شده به بلاک‌های $k$ بیتی تقسیم شده باشند آنگاه زمان تأخیر به صورت مقابل خواهد بود.

$$t_{CLA} = t_{pg} + t_{pg-block} + (\frac{N}{k} - 1)t_{AND-OR} + kt_{FA}$$

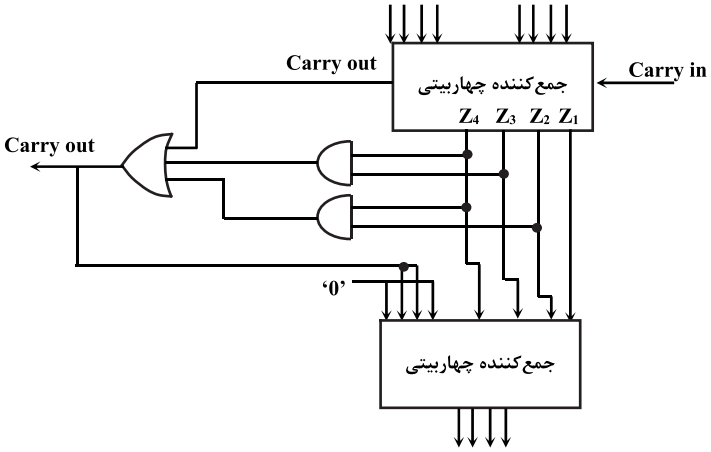
## تست‌های طبقه‌بندی شده فصل دوم

**۱ـ اگر بخواهیم ضرب $A=(110011)_2$ را در $B=(101110)_2$ با اعمال کد بوث (Booth) انجام دهیم به ترتیب چند عمل Shift و چند عمل Add/Sub باید انجام دهیم؟** (مهندسی کامپیوتر ـ سراسری ۷۷)

۱) 4 Add/sub , 5 shift　　۲) 3 Add/sub , 5 shift

۳) 6 Add/sub , 6 shift　　۴) 8 Add/sub , 6 shift

**۲ـ برای انجام عملیاتی 16 بیتی از 4 واحد عملیاتی 4 بیتی زیر استفاده شده است. این چهار واحد با روش Carry ripple (اتصال خروجی Carry یک واحد به ورودی Carry واحد بعدی) به یکدیگر متصل شده‌اند. چنانچه یکی از ورودی‌های 16 بیتی 375 و ورودی دیگر 9760 باشد نتیجه خروجی 16 بیتی چه خواهد بود؟ (تمام اعداد در پایه 16 هستند)** (مهندسی کامپیوتر ـ سراسری ۷۸)




۱) 9AD5

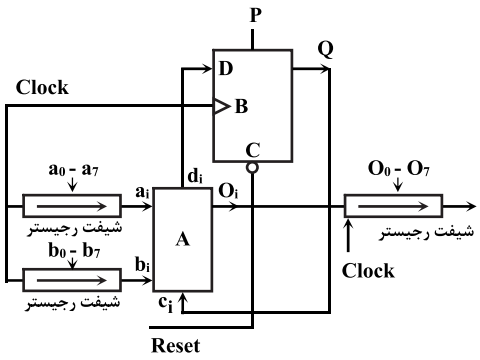
۲) 9135

۳) 0935

۴) 0135

**۳ـ در مدار زیر بلوک B یک فیلیپ‌فلاپ D است و عملیات زیر را انجام می‌دهد.** (مهندسی کامپیوتر ـ سراسری ۷۸)

**$d_i = a_ib_i + a_ic_i + b_ic_i$ و $o_i = a_i \oplus b_i \oplus c_i$ و $a_i$ و $b_i$ بیتهای داده‌های چند بیتی هستند که با هر کلاک توسط شیفت رجیستر به مدار A اعمال می‌شوند. چنانچه تأخیر محاسبه $O_i$ برابر با 20 نانو ثانیه و تأخیر محاسبه $d_i$ برابر 15 نانو ثانیه و تأخیر فیلیپ‌فلاپ 10 نانو ثانیه و پریود کلاک 40 نانو ثانیه باشد مدار فوق جمع دو عدد هشت بیتی a و b را در چه زمانی انجام می‌دهد.**




۱) 160 نانو ثانیه

۲) 280 نانو ثانیه

۳) 320 نانو ثانیه

۴) مدار فوق از نظر عملکرد و یا زمان‌بندی نمی‌تواند 2 عدد 8 بیتی را با هم جمع کند.

**۴ـ حداقل سخت‌افزار لازم برای ساخت یک ضرب کنندهٔ آرایه‌ای (Array Multiplier) که عدد ۵ بیتی $b_4b_3b_2b_1b_0$ را در عدد دوبیتی $a_1a_0$ ضرب می‌نماید ..... عدد AND دو ورودی و یک عدد جمع کننده ..... بیتی می‌باشد.** (مهندسی کامپیوتر ـ سراسری ۸۰)

۱) 6,12　　۲) 6,10　　۳) 5,12　　۴) 5,10

**۵ـ در عمل تقسیم در چه صورت divide-overflow رخ می‌دهد؟** (علوم کامپیوتر ـ سراسری ۸۰)

۱) فقط زمانی که تقسیم به صفر انجام شود.

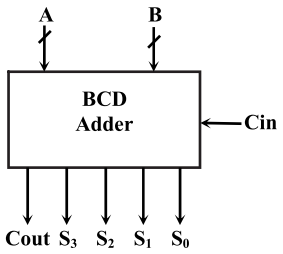
۲) عدد موجود در نیمه بالای خارج قسمت از مقسوم علیه بزرگ

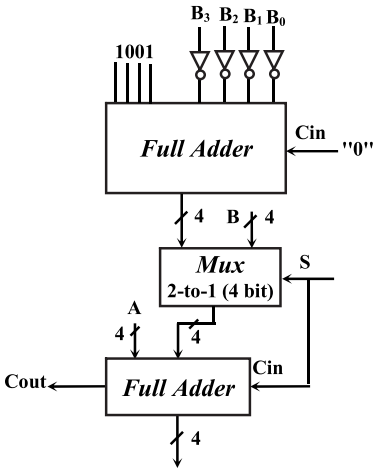
۳) باقیمانده یک عدد منفی باشد.

۴) هیچکدام

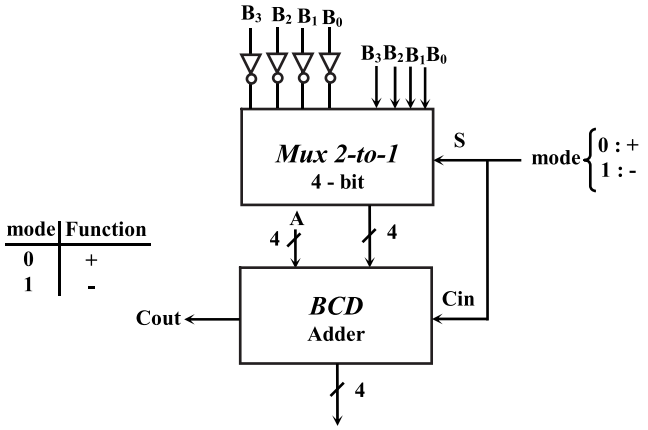
**۶ـ در صورتیکه شکل زیر یک جمع‌کننده BCD باشد، چگونه می‌توان آن را تبدیل به یک جمع کننده/تفریق کننده BCD نمود؟**

(مهندسی کامپیوتر ـ سراسری ۸۲)

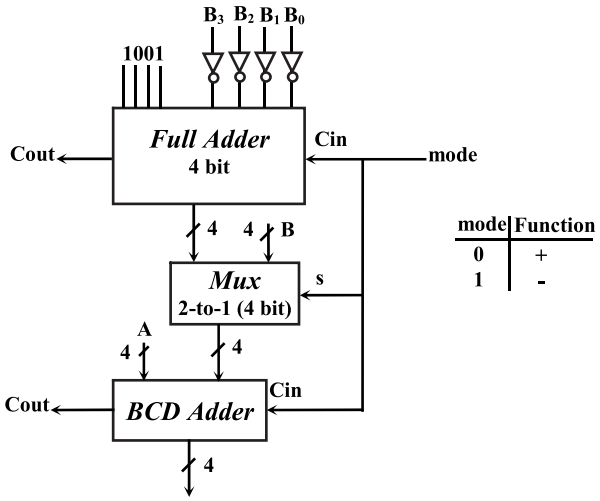


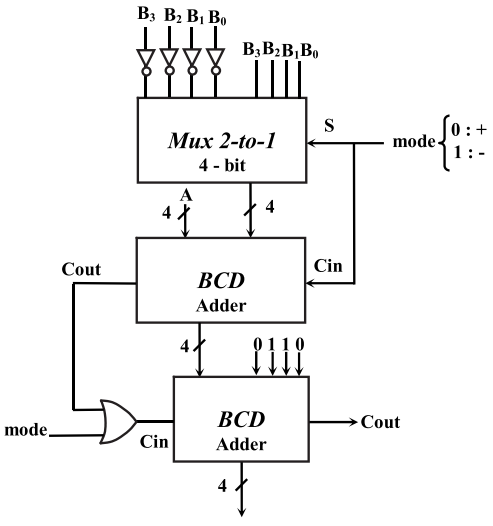

(۲)




(۱)




(۴)

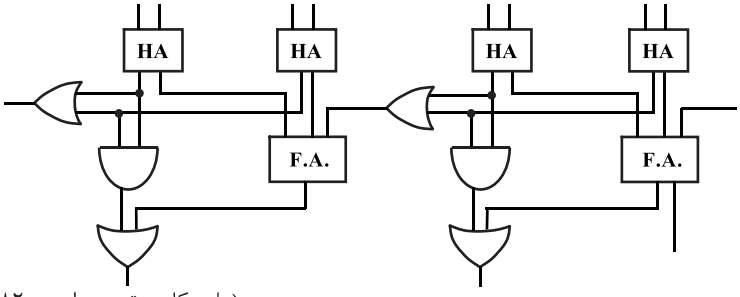



(۳)

**۷ـ اگر تعداد این سلولها دو برابر شود در تاخیر ماکزیمم چه اثری می‌گذارد؟**

(مهندسی کامپیوتر ـ سراسری ۸۱)

۱) 1.5 برابر می‌شود.
۲) دو برابر می‌شود.
۳) سه برابر می‌شود.
۴) تفاوتی نمی‌کند.



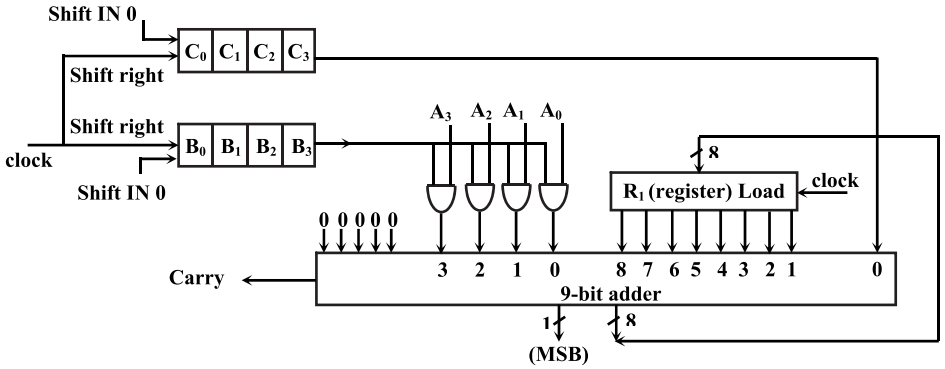

**۸ـ کدام گزاره صحیح است؟**

(علوم کامپیوتر ـ سراسری ۸۲)

۱) عمل جمع Floating – point همواره شرکت‌پذیر است.
۲) عمل shift left معادل ضرب در توانی از عدد 2 است.
۳) عمل shift right معادل تقسیم بر توانی از عدد 2 است.
۴) عمل جمع Floating – point جابجاپذیر نمی‌باشد.

**۹ـ در شکل مقابل سه عدد چهار بیتی $(A_3\,A_2\,A_1\,A_0)$، $(B_3\,B_2\,B_1\,B_0)$ و $(C_3\,C_2\,C_1\,C_0)$ را داریم. مقدار اولیه ثبات $R_1$ صفر است. محتویات اولیه شیفت‌دهنده را نیز داده‌ایم بعد از گذشت چهار پالس ساعت در خروجی جمع‌کننده چه مقداری قرار خواهد داشت؟**

(مهندسی کامپیوتر ـ سراسری ۸۳)

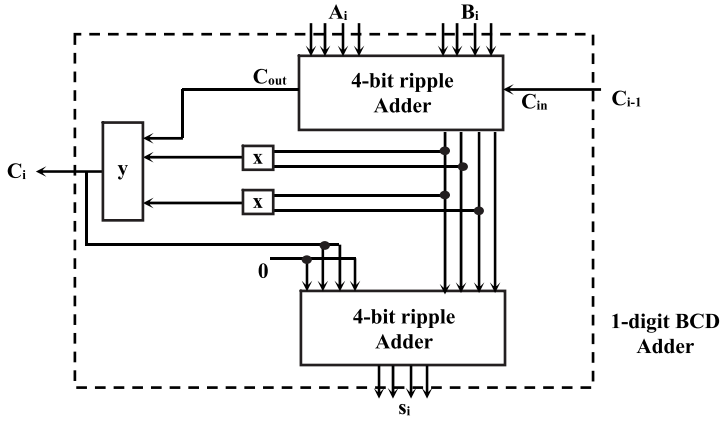



۱) $(A_3\,A_2\,A_1\,A_0)\times(B_3\,B_2\,B_1\,B_0)+C_3\,C_2\,C_1\,C_0$　　۲) $(C_3\,C_2\,C_1\,C_0)\times(B_3\,B_2\,B_1\,B_0)+A_3\,A_2\,A_1\,A_0$

۳) $(B_0\,B_1\,B_2\,B_3)\times(A_3\,A_2\,A_1\,A_0)+C_0\,C_1\,C_2\,C_3$　　۴) $(A_0\,A_1\,A_2\,A_3)\times(B_0\,B_1\,B_2\,B_3)+C_0\,C_1\,C_2\,C_3$

**۱۰ـ با استفاده از بلوکهای جمع کننده دهدهی یک رقمی (نشان داده شده در شکل زیر) مداری جهت انجام جمع موازی دهدهی 10 رقمی (با تکرار 10 واحد از این بلوک‌ها) ساخته‌ایم. گیت‌های x و y (در شکل) و تأخیر نهایی جمع کننده 10 رقمی برابرند با: (تأخیر گیت‌ها را 10ns فرض کنید)**

(مهندسی کامپیوتر ـ سراسری ۸۳)




۱) 1080ns, or, and

۲) 1060ns, or, and

۳) 1080ns, nor, and

۴) 1600ns, nand, nand

**۱۱ـ اگر بیت شماره 15 یک رجیستر 16بیتی cpu مقدار داشته باشد، کدام جمله در رابطه با مقدار موجود در این رجیستر صحیح‌تر است؟**

(علوم کامپیوتر ـ سراسری ۸۳)

۱) می‌تواند نشان دهنده یک عدد signed و یا unsigned باشد.

۲) می‌تواند نشاندهنده یک عدد مثبت یا منفی باشد.

۳) نشان دهنده یک عدد منفی است.

۴) می‌تواند نشان دهنده یک عدد مثبت یا منفی و یا یک عدد signed و یا unsingned باشد.

**۱۲ـ یکی از روش‌های ضرب دو عدد روش جمع و شیفت است. به این ترتیب:**

```
 101
×101
-----
 10 1   → ○ شیفت به راست
+101
-----
 11 1   → ○ شیفت به راست
+000
-----
 011  1 1
```

**یعنی حاصل‌ضرب دو عدد 1111 است. $(15 = 3\times5)$ دیاگرام قالبی چنین مداری برای ضرب دو عدد چهاربیتی در شکل دیده می‌شود. خط کنترل در حقیقت رقم باقیمانده B است که اگر '1' باشد، اجازه می‌دهد A با محتویات آکومولاتور (پس از شیفت) جمع شود. برای اجرای کامل عمل ضرب در مجموع چند پالس کنترل (اعم از بافره آکومولاتور و ثبات B) لازم است؟**

(مهندسی کامپیوتر ـ آزاد ۸۳)

۱) 12　　۲) 4　　۳) 16　　۴) 14

**۳۶ـ دو عدد A و B در نمایش ممیز شناور با طول میدان‌های زیر و با نمای اریب‌دار (Biased) چه حاصل‌جمعی دارد؟** (مهندسی کامپیوتر ـ سراسری ۹۵)

| S | E | F |
|---|---|---|

$N = (-1)^S \times 2^{E-biased} \times 1.F$

**S : Sign : 1 bit**
**F :Fraction :4 bits**
**E :Exponent : 3 bits**
**Biased = +4**

A | 0 | 111 | 0010 |

B | 0 | 111 | 0001 |

۱) | 0 | 111 | 0011 |
۲) | 0 | 111 | 0010 |
۳) | 0 | 111 | 0001 |
۴) غیرقابل نمایش

**۳۷ـ کدام مورد با توجه به الگوریتم ضرب Booth، برای ضرب دو عدد n بیتی علامت‌دار مکمل 2، صحیح نیست؟** (مهندسی کامپیوتر ـ سراسری ۹۶)

۱) اگر مضروب‌فیه عددی مثبت باشد: تعداد عمل جمع = تعداد عمل تفریق
۲) در هر صورت تعداد عمل جمع و تفریق برابر هستند.
۳) اگر مضروب‌فیه عددی منفی باشد: تعداد عمل تفریق = تعداد عمل جمع
۴) در این الگوریتم همیشه عمل تفریق قبل از عمل جمع انجام می‌شود.

**۳۸ـ چند مورد از گزاره‌های داده شده درست هستند؟** (مهندسی کامپیوتر ـ سراسری ۹۷)

**a) در ضرب دوعدد دودویی n بیتی به‌روش Booth، همیشه تعداد عمل تفریق بیشتر از تعداد عمل جمع است.**
**b) در تقسیم دوعدد دودویی به روش غیرجبرانی (non – restoring) نیازی به مقایسه‌گر (comparator) نیست.**
**c) در مدار ضرب‌کننده ترکیبی دو عدد دودویی 10 بیتی و 12 بیتی بدون علامت، از 120 گیت 2-input AND استفاده می‌شود.**
**d) در تقسیم جبرانی (restoring) دو عدد دودویی، تعداد عمل تفریق مورد نیاز با تعداد عمل جمع جبرانی برابر است.**

۱) ۱ ۲) ۲ ۳) ۳ ۴) ۴

**۳۹ـ برای جمع ترکیبی 9 عدد دودویی n بیتی به‌روش carry – save، حداقل تعداد جمع‌کننده carry – save کدام است؟**
(مهندسی کامپیوتر ـ سراسری ۹۷)

۱) 7 ۲) 9 ۳) 11 ۴) بستگی به n دارد.

**۴۰ـ دو عدد $A = 1010010$ : مضروب و $B = 1110011$ : مضروب فیه به روش Add & Shift در هم ضرب می‌شوند. تعداد عملیات جمع کدام است؟** (مهندسی کامپیوتر ـ سراسری ۹۸)

۱) 5 ۲) 4 ۳) 3 ۴) 2

## پاسخنامه تست‌های طبقه‌بندی شده فصل دوم

**۱ـ هیچکدام از گزینه‌ها صحیح نیست.** به طور کلی در روش بوث تعداد shiftها برابر تعداد بیت‌های ضرب کننده (مضروب فیه) می‌باشد و تعداد جمع و تفریق نیز برابر تعداد تغییر بیت در آن است. یعنی در این جا ۶ عمل shift نیاز داریم و دنباله‌های "10" تعداد تفریق‌ها و دنباله‌های "01" (در مضروب فیه) تعداد جمع‌ها را نشان می‌دهد بنابراین 3 عمل Add/sub نیاز داریم.

---

**۲ـ گزینه «۴»** مدار مورد نظر یک جمع کننده BCD است. 4 واحد از این مدار دو عدد دسیمال 4 رقمی را جمع می‌کند.

$$\begin{array}{r} 0375 \\ +\ 9760 \\ \hline 10135 \end{array}$$

خروجی ۴ رقمی برابر 0135 می‌باشد.

---

**۳ـ گزینه «۳»** سرعت مدار به پریود کلاک محدود می‌شود. ورودی‌های $a_i$ و $b_i$ در هر پریود کلاک (40ns) آماده می‌شود که در طول این مدت خروجی‌های $o_i$ و $d_i$ آماده می‌شوند. به راحتی می‌توان دید که پالس کلاک به اندازه‌ای طولانی می‌باشد که خروجی‌های لازم آماده باشند در غیر این صورت مدار رفتار درستی از خود نشان نمی‌داد. $8 \times 40 = 320ns$

---

**۴ـ گزینه «۴»** یک ضرب کننده آرایه‌ای j بیتی در k بیتی به $j \times k$ گیت AND و $(j-1)$ جمع کننده k بیتی نیاز دارد.

**۵ـ گزینه «۴»** هرگاه نیمه بالای مقسوم بیشتر یا مساوی مقسوم علیه باشد و تعداد بیت‌های خارج قسمت محدود باشد مانند آنچه در مثال فصل دیده شد، سرریز رخ می دهد. در حالت تقسیم بر صفر نیز سرریز رخ می‌دهد که حالتی از جمله اول است.

**۶ـ گزینه «۴»** برای تفریق A-B می‌توان A را با مکمل 10 عدد جمع کرد. یک روش مکمل 10 کردن این است که عدد B را مکمل 9 کنیم و با عدد ۱ جمع کنیم. برای محاسبه مکمل 9 می‌توان ابتدا عدد را مکمل ۱ کرد و سپس حاصل را با 1010 جمع نمود. لازم به ذکر است که با وارد کردن یک از طریق $C_{in}$ و دادن ورودی ثابت 1001، در حقیقت عدد با 1010 جمع می‌شود.

**۷ـ گزینه «۴»** ورودی تمام FAها پس از دو تاخیر در دسترس است (تاخیر در HA و OR) بنابراین افزایش سلولها تاثیری در تاخیر ندارد.

**۸ـ گزینه «۲ و ۳»**

الف ـ اگر x و y و z اعداد ممیز شناور باشند آنگاه $(x+y)+z \neq x+(y+z)$ زیرا به دلیل خطاهای گرد کردن ممکن است نتایج مختلف حاصل شود.

ب ـ عمل shift left به شرطی ضرب در توانی از ۲ می‌باشد که سرریز رخ ندهد.

ج ـ عمل شیفت به راست منطقی برای اعداد بی‌علامت و عمل شیفت به راست حسابی برای اعداد علامت‌دار معادل تقسیم صحیح بر توانی از عدد ۲ می‌باشد.

د ـ جمع ممیز شناور جابجاپذیر است.

گزینه‌های ۲ و ۳ می‌توانند صحیح باشند.

**۹ـ گزینه «۱»** برای سادگی ابتدا $C=0$ را در نظر می‌گیریم. در هر مرحله مقادیر بیتهای A به ترتیب در بیتهای $B_3$ ، $B_2$ ، $B_1$ و $B_0$ ضرب می شود و حاصل با نتیجه قبلی جمع می‌شود. در هر مرحله نتیجه قبلی یک بیت به چپ شیفت پیدا می‌کند چون حاصل جمع به بیتهای 1 الی 8 جمع‌کننده منتقل می‌شوند و بیت صفر به $(C=0)$ متصل است.

| | | | | | | |
|---|---|---|---|---|---|---|
| $A_3B_3$ | $A_2B_3$ | $A_1B_3$ | $A_0B_3$ | 0 | 0 | 0 |
| + | $A_3B_2$ | $A_2B_2$ | $A_1B_2$ | $A_0B_2$ | 0 | 0 |
| + | | $A_3B_1$ | $A_3B_1$ | $A_1B_1$ | $A_0B_1$ | 0 |
| + | | | $A_3B_0$ | $A_2B_0$ | $A_1B_0$ | $A_0B_0$ |
| | | | | | | حال ضرب $A\times B$ |

**۱۰ـ گزینه «۱»** تاخیر یک جمع کننده ۴ بیتی برابر است با:

تعداد بیت‌ها ← $(4 \times 2)$ ← تعداد گیت‌های سریال در جمع‌کننده

$8\times$ تأخیر یک گیت $=8\times 10=80ns$

$X_1$ تأخیر ناشی از ← Y تأخیر ناشی از

تأخیر نقلی در مرتبه 10دهم $=9\times 100=900ns$ ; انتقال نقلی به طبقه بعدی $=80+10+10=100ns$

تأخیر کل $=900+180ns=1080ns$ تأخیر جمع کننده BCD در طبقه آخر $=80ns+80ns+20=180ns$

تأخیر ماژول جمع‌کننده‌های بیتی پایینی ← تأخیر ماژول جمع‌کننده‌های بیتی بالایی ← تأخیر ناشی از دو گیت سریال X و Y

**۱۱ـ گزینه «۲»** علامت‌دار بودن یا نبودن یک عدد بستگی به نظر طراح دارد و به صورت خاصی مشخص نمی‌شود. با ارزش‌ترین بیت در اعداد علامت‌دار نشان‌دهنده مثبت یا منفی بودن عدد است.

**۱۲ـ گزینه «۱»** تعداد کلاک‌های لازم $B$ = بار کردن + تعداد بیت‌ها $\times 2$ + تعداد بیت‌ها $-1=1+4\times 2+3=12\,clock$

**۱۳ـ گزینه «۱»** با توجه به چگونگی تولیدی $(V=C_7\oplus C_8)$، در این ALU از روش مکمل ۲ استفاده می‌شود.

$A-B=A+(2,s\,compleme,t(B)=A+\overline{B}+1$

$=11110000+2,s\ complement(00010100)=111100000+11101100=11011100$

مدرسان شریف

# فصل سوم

## «ریز عملیات»

### مقدمه

این فصل مقدمه‌ای بر طراحی یک پردازنده کامپیوتری ساده می‌باشد. نخست زبان RTL و قواعد آن معرفی و تشریح می‌شود، در ادامه به عملیات منطقی و پرکاربرد شیفت و انواع آن پرداخته خواهد شد. پس از آن با هدف تشریح چگونگی تبادل داده‌ها میان بخش‌های مختلف یک پردازنده، گذرگاه و روش‌های پیاده‌سازی آن شرح داده می‌شود. در پایان به معرفی چارت ASM به عنوان یک راهکار پرکاربرد در طراحی مدارات سخت‌افزاری خواهیم پرداخت.

برای نمایش نقل و انتقال اطلاعات بین ثبات‌ها و تغییر آن‌ها از زبان نمادین RTL یا «زبان انتقال ثبات» (Register Transfer Language) استفاده می‌شود. در حقیقت می‌توان گفت کاربرد زبان RTL توصیف رفتار سخت‌افزار می‌باشد. این زبان یک زبان برنامه‌نویسی نمی‌باشد و صرفاً برای تشریح انتقال اطلاعات بین ثبات‌های یک سیستم دیجیتال استفاده می‌شود. دستورات یک زبان سطح بالا برای اجرا روی سخت‌افزار باید دستخوش چند پردازش واقع گردند. ابتدا دستورات یک زبان سطح بالا به زبان اسمبلی ترجمه می‌شود و پس از تولید کد عملیات در نهایت «ریز دستورات» یا میکروآپ‌ها (micro operation) ایجاد می‌گردند که بر روی سخت‌افزار اجرا خواهند شد.

تشکیلات سخت‌افزاری یک سیستم دیجیتال را می‌توان به صورت زیر دسته‌بندی نمود:

**۱) تعداد ثبات‌های سیستم**

**۲) ترتیب انجام ریز دستورات (ریز عملیات)**

**۳) وجود کنترلی که باعث شروع ریز عملیات می‌گردد**

قواعد مورد استفاده در زبان نمادین RTL در زیر بیان شده است:

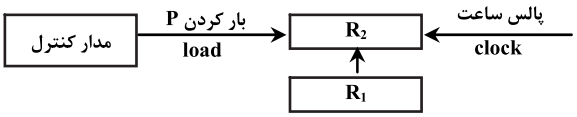
۱ـ اصولاً از حروف بزرگ برای نام‌گذاری ثبات‌های سیستم استفاده می‌شود.

۲ـ ثبات آدرس حافظه با MAR (Memory Address Register)، ثبات شمارنده‌ی برنامه با PC (Program counter) و ثبات دستورالعمل با IR (Instruction Resister) نمایش داده می‌شود.

۳ـ بیت‌های یک ثبات از سمت راست به چپ شماره‌گذاری می‌شوند.

۴ـ برای انتقال شرطی محتوای یک ثبات مانند $R_1$ به ثبات $R_2$ با شرط P، از نمادگذاری مقابل استفاده می‌شود: $P: R_2 \leftarrow R_1$

در این حالت در صورتی که $P=1$ باشد مقدار $R_1$ به $R_2$ منتقل خواهد شد.




مدار سخت‌افزاری این عملیات در شکل مقابل آمده است:

۵ـ برای انجام چند عمل در یک زمان از جداکننده «,» در بین آن‌ها استفاده می‌گردد. به عنوان مثال برای تعویض محتوای دو ثبات $R_1$ و $R_2$ در صورت برقرار بودن شرط t از ریز دستور زیر استفاده می‌گردد: $t: R_2 \leftarrow R_1\ ,\ R_1 \leftarrow R_2$

۶ـ در یک ریز عملیات برای مشخص نمودن قسمتی از یک ثبات از "( )" استفاده می‌گردد که شماره‌ی بیت‌های مورد نظر از ثبات داخل پرانتز نوشته می‌شود و برای آدرس‌دهی به حافظه از "[ ]" استفاده می‌شود. به عنوان مثال ریز دستور زیر محتویات ثبات $R_1$ را در آدرسی از حافظه که توسط AR مشخص شده قرار می‌دهد: $M[AR] \leftarrow R_1$

و ریز دستور روبرو بیت‌های 0 تا 7 (بایت کم ارزش) از ثبات $R_1$ را در ثبات AR قرار می‌دهد: $AR \leftarrow R_1(0-7)$

۷ـ برای این که دو شرط در یک ریز دستور AND و یا OR شوند در قسمت شرطی بین دو شرط به ترتیب "•" و "+" قرار داده می‌شود. به عنوان مثال در ریز دستور روبرو در صورتی که "$t_1$ OR $t_2 = 1$" باشد آن‌گاه محتویات $R_1$ به $R_2$ منتقل می‌گردد: $t_1 + t_2 : R_2 \leftarrow R_1$

و در دستور روبرو در صورتی که "$t_1$ AND $t_2 = 1$" باشد محتویات $R_1$ به $R_2$ منتقل می‌گردد: $t_1.t_2 : R_2 \leftarrow R_1$

اما دقت نمایید که عملگر "+" در قسمت دستورات به معنی جمع است، به عنوان مثال در ریز دستور زیر اگر "$P_1$ OR $P_2 = 1$" باشد آن‌گاه، حاصل جمع $R_2$ و $R_1$ به $R_3$ منتقل خواهد شد: $P_1 + P_2 : R_3 \leftarrow R_1 + R_2$

۸ـ برای انجام عملیات OR و AND بر روی بیت‌های متناظر دو ثبات به ترتیب از نمادهای $\vee$ و $\wedge$ استفاده می‌شود. همچنین برای مکمل کردن تمام بیت‌های یک ثبات مانند R از نماد $\overline{R}$ و برای برقرار نبودن یک شرط مانندP از نماد $P'$ استفاده می‌گردد.

به عنوان مثال در ریز عمل روبرو در صورتی که $P = 1$ و $q = 0$ باشد آن‌گاه مکمل دوی $R_1$ به $R_2$ منتقل می‌گردد. $p.q' : R_2 \leftarrow \overline{R}_1 + 1$

**مثال۱: $R_0, R_1, R_2$ و $R_3$ ثباتهای n بیتی هستند. فیلیپ‌فلاپهای E ، S ، $F_0$ ، $F_1$ و $F_2$ در سیستم وجود دارند، مجموعه ریز دستورهای زیر چه عملی را انجام می‌دهند؟ (در ابتدا $S = 1$ برقرار است)**

$$S : R_3 \leftarrow 0 \ , S \leftarrow 0 \ , F_0 \leftarrow 1, R_1 \leftarrow R_0 \ , R_3 \leftarrow \circ$$
$$F_0 : \ R_1 \leftarrow \overline{R}_1 \ , F_0 \leftarrow 0 \ , F_1 \leftarrow 1$$
$$F_1 : \ R_1 \leftarrow R_1 + 1 \ , F_1 \leftarrow 0 \ , F_2 \leftarrow 1$$
$$F_2 , R_2 \leftarrow R_2 + R_1 \ , F_2 \leftarrow 0 \ , F_3 \leftarrow 1$$
$$F_3 : \ F_3 \leftarrow 0 \ , IF(R_2 < 0) then [R_2 \leftarrow R_2 + R_0 \ , E \leftarrow 1] else [R_3 \leftarrow R_3 + 1, F_2 \leftarrow 1]$$
$$E : halt$$

۱) محتویات $R_2$ بر $R_0$ تقسیم می‌شود $R_2$ خارج قسمت و $R_3$ باقیمانده است.

۲) محتویات $R_2$ بر $R_0$ تقسیم می‌شود. $R_3$ خارج قسمت و $R_2$ باقیمانده است.

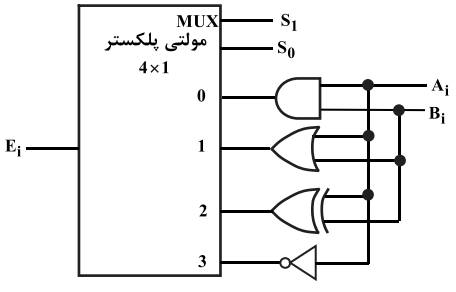
۳) محتویات $R_2$ با $R_2$ مقایسه می‌گردد اگر $R_2$ بزرگتر باشد $R_3$ صفر می‌شود و در غیر این صورت $R_3$ یک می‌شود.

۴) محتویات $R_0$ با $R_2$ مقایسه می‌گردد اگر $R_2$ کوچکتر باشد $R_3$ صفر می‌شود و در غیر این صورت $R_3$ یک می‌گردد.

☑ **پاسخ:** گزینه «۱» در دستور نخست یعنی S، برای آن که مقدار $R_0$ بدون تغییر حفظ شود، مقدار آن به رجیستر $R_1$ منتقل شده است. در شرط‌ها $F_0$ و $F_1$ مکمل دوم $R_1$ که در حقیقت همان $R_0$ می‌باشد، محاسبه می‌شود. (در $F_0$ هر بیت آن معکوس و در $F_1$ با یک جمع می‌گردد.) در شرط $F_2$، مقدار جدید $R_1$ که همان مکمل دوم $R_0$ می‌باشد، با $R_2$ جمع می‌گردد. بدین ترتیب مقدار $R_2 = R_2 - R_0$ خواهد بود. در $F_3$ کنترل می‌گردد که آیا $R_2$ منفی شده است یا خیر، اگر مثبت باشد به مقدار $R_3$ یک واحد افزوده می‌شود و به $F_2$ بر می‌گردیم. مجدداً مقدار $R_0$ از $R_1$ کم می‌شود زیرا در $R_1$ مکمل دوم $R_0$ قرار گرفته است. همان‌طور که دیده می‌شود، در هر بار اجرای دنبالهٔ $F_2$ و $F_3$، یک واحد به $R_3$ اضافه و به اندازهٔ $R_0$ از $R_2$ کاسته می‌شود. این دنباله اجرا می‌شود تا آنجا که مقدار $R_2$ کوچک‌تر از صفر شود، در این حالت مقدار $R_\circ$ به $R_2$ اضافه می‌شود و اجرای برنامه خاتمه می‌یابد، این شرط پایان یک تقسیم است. بنابراین مجموعه دستورات فوق عمل تقسیم $R_2$ بر $R_0$ را انجام می‌دهند. در پایان برنامه خارج قسمت در $R_3$ و باقی‌مانده در $R_2$ قرار دارد. (این عمل تقسیم به روش غیر ترمیم است زیرا هر بار از مقدار $R_2$ کاسته می‌شود و سرانجام تعداد باقی‌مانده را در خود ذخیره می‌کند.)

---

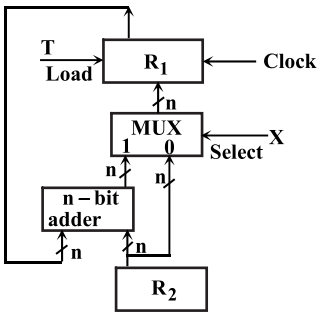
پر واضح است که می‌توان ریز عملیات بیان شده را بر روی تعداد خاصی از بیت‌های یک ثبات نیز انجام داد. شکل زیر مدار سخت‌افزاری لازم برای انجام عملیات منطقی شامل AND، OR، XOR و NOT را نشان می‌دهد:




| $S_1$ | $S_2$ | جدول خروجی | نوع عملیات |
|---|---|---|---|
| 0 | 0 | $E = A \wedge B$ | AND |
| 0 | 1 | $E = A \vee B$ | OR |
| 1 | 0 | $E = A \oplus B$ | XOR |
| 1 | 1 | $E = \overline{A}$ | مکمل |

به عنوان مثال برای این که چهار بیت کم ارزش یک ثبات را برابر 1010 قرار دهیم، ابتدا 0000 را با چهار بیت کم ارزش آن AND و سپس دنباله 1010 را با آن‌ها OR می‌نماییم. AND اولیه برای اطمینان از صفر شدن مقدار اولیه چهار بیت پایین ثبات موردنظر می‌باشد.

**مثال۲: در شکل زیر کدام یک از موارد انتقال رجیستری (RTL) قابل اجرا می‌باشد.**




۱) $x: R_1 \leftarrow R_2$

۲) $xT: R_1 \leftarrow R_2$

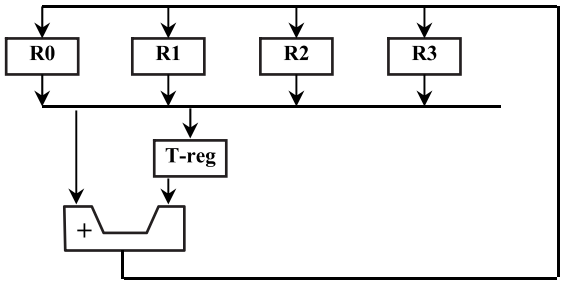
۳) $\overline{x}T: R_1 \leftarrow R_2$

۴) $T\overline{x}: R_1 \leftarrow R_1 + R_2$

☑ **پاسخ:** گزینه «۳» با توجه به شکل داده شده، اگر x برابر صفر و T برابر یک باشد آنگاه ورودی $R_2$ از مالتی پلکسر انتخاب و خط load از ثبات $R_1$ فعال می‌گردد. بنابراین ریز عمل $R_1 \leftarrow R_2$ انجام می‌گیرد.

---

**مثال۳: برای انجام هر یک از عملیات $R_1 \leftarrow R_1 + R_2$ و $R_1 \leftarrow R_2 + R_3$ در طرح مسیر داده شده (Data path) کاملاً سنکرون به شکل زیر، به ترتیب به چند پالس ساعت نیاز است؟ (پریود پالس ساعت به اندازه کافی زیاد است.)**




۱) 1 و 2

۲) 2 و 2

۳) 2 و 3

۴) 3 و 3

☑ **پاسخ:** گزینه «۲» برای $R_1 \leftarrow R_1 + R_2$ به کلاک‌های زیر احتیاج داریم:

کلاک اول : $T - Reg \leftarrow R_2$

کلاک دوم : $R_1 \leftarrow R_1 + T - Reg$

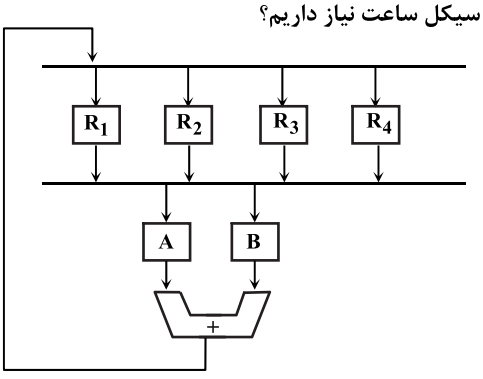
برای $R_1 \leftarrow R_2 + R_3$ به کلاک‌های زیر احتیاج داریم:

کلاک اول : $T - Reg \leftarrow R_2$

کلاک دوم : $R_1 \leftarrow T - Reg + R_3$

---

**مثال۴: فرض کنید در ساختار زیر بخواهیم عمل $R_1 \leftarrow 4 \times R_1$ را انجام دهیم، به چند سیکل ساعت نیاز داریم؟**




۱) 3

۲) 4

۳) 5

۴) 6

☑ **پاسخ:** گزینه «۲» پالس‌های لازم به صورت زیر است:

| | |
|---|---|
| پالس اول | $A \leftarrow R_1 \ , \ B \leftarrow R_1$ |
| پالس دوم $(A = B = R_1)$ | $R_1 \leftarrow A + B$ |
| پالس سوم (در این پالس A، B و $R_1$ دو برابر $R_1$ اولیه می‌باشند) | $A \leftarrow R_1 \ , \ B \leftarrow R_1$ |
| پالس چهارم (ثبات $R_1$ چهار برابر $R_1$ اولیه می‌باشد) | $R_1 \leftarrow A + B$ |

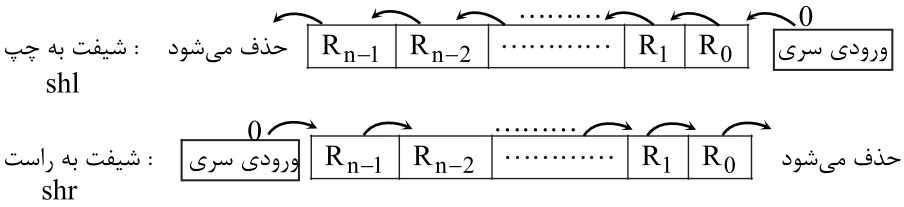
علاوه بر ریز عملیات منطقی و محاسباتی بیان شده، یکی از ریز عملیات مهم شیفت می‌باشد که در حالت کلی می‌توان این ریز عملیات را به سه دسته تقسیم نمود:

**۱ـ شیفت منطقی (logical shift)** **۲ـ شیفت چرخشی (rotate shift)** **۳ـ شیفت محاسباتی یا شیفت ریاضی (arithmetic shift)**

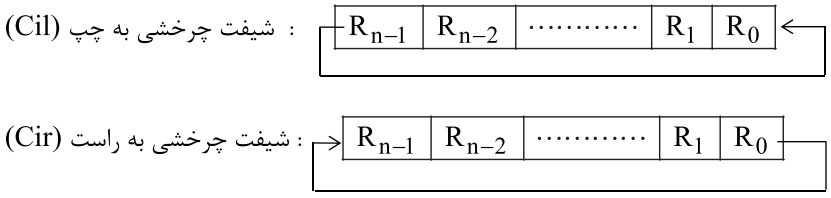
**شیفت منطقی:** دو نوع شیفت منطقی بر روی ثبات‌ها قابل انجام است.

۱) در شیفت به چپ از ورودی سری صفر وارد می‌شود و به بیت کم ارزش منتقل می‌گردد و بقیه بیت‌ها به سمت چپ شیفت داده می‌شوند.

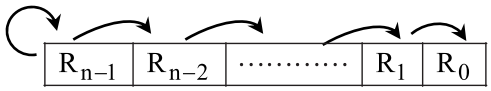
۲) در شیفت به راست ورودی سری مدار (که باز هم صفر می‌باشد) به با ارزش‌ترین بیت وارد می‌شود و بقیه بیت‌ها به سمت راست شیفت پیدا می‌کنند.




**شیفت چرخشی:** در شیفت چرخشی به چپ خروجی سری ثبات به ورودی سری ثبات متصل می‌باشد و بیت‌ها به سمت چپ شیفت می‌یابند و در شیفت به راست همین اتفاق با شیفت به راست بیت‌ها صورت می‌پذیرد:




**شیفت ریاضی:** در شیفت ریاضی ثبات‌ها حاوی اعداد علامت‌دار در نظر گرفته می‌شوند. در شیفت ریاضی به راست بیت‌ها شبیه به شیفت منطقی به راست شیفت پیدا می‌کنند اما در نهایت بیت علامت ثابت باقی می‌ماند شکل روبرو نحوه انجام شیفت ریاضی به راست را نشان می‌دهد:



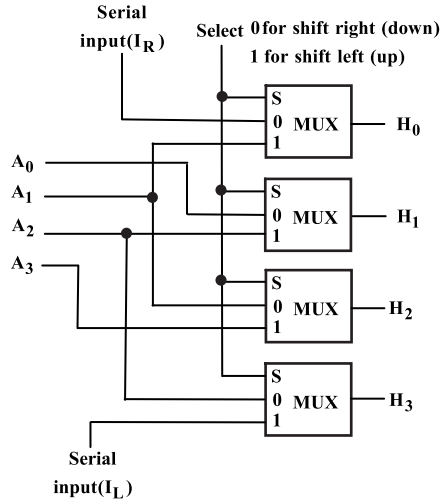

در شیفت ریاضی به چپ عملیات مانند شیفت منطقی به چپ صورت می‌پذیرد اما در صورتی که پس از انجام عمل شیفت بیت علامت تغییر یابد آن‌گاه نتیجه قابل قبول نمی‌باشد و سرریز رخ داده است.

**نکته۱:** هر شیفت ریاضی به راست معادل تقسیم عدد مورد نظر بر ۲ می‌باشد و هر شیفت ریاضی به چپ معادل ضرب عدد موردنظر در ۲ می‌باشد.

برای کنترل وجود یا عدم وجود سرریز در عملیات شیفت چرخشی به چپ کافیست قبل از عملیات شیفت کنترل شود که آیا دو بیت با ارزش با یکدیگر متفاوت می‌باشند یا نه. در صورت متفاوت بودن سرریز رخ خواهد داد. بنابراین می‌توان بیت سرریز $V$ را به صورت زیر تعریف نمود که در صورت صفر بودن $V$ سرریز رخ نمی‌دهد و در صورتی که $V=1$ باشد، سرریز رخ خواهد داد:

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

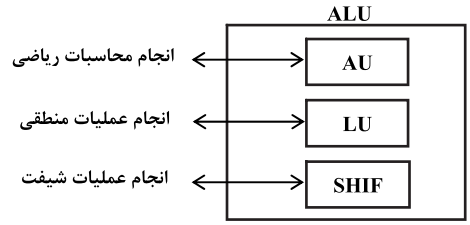
شکل زیر مدار سخت‌افزاری لازم برای عملیات شیفت را نشان می‌دهد:




جدول عملکرد

| Select | Output | | | |
|---|---|---|---|---|
| S | $H_0$ | $H_1$ | $H_2$ | $H_3$ |
| 0 | $I_R$ | $A_0$ | $A_1$ | $A_2$ |
| 1 | $A_1$ | $A_2$ | $A_3$ | $I_L$ |

با توجه به مطالب بیان شده می‌توان گفت هر بیت از مدار محاسبه و منطق ALU باید به صورت مقابل باشد:




**مثال۵: عملیات میکروی زیر در یک سیستم دیجیتال تعریف شده‌اند. S و F و D فلیپ‌فلاپ‌های کنترلی سیستم می‌باشد. ثبات‌های A، B و CNTR ثبات‌های پردازنده سیستم بوده و همگی n بیتی هستند. سیستم دارای یک مقایسه کننده n بیتی و یک تفریق کننده n بیتی است. سیستم در ابتدا با فعال شدن فلیپ‌فلاپ S شروع بکار می‌کند. سیستم زیر چه عمل ریاضی را انجام می‌دهد و نتیجه (نتایج) حاصل در کدام ثبات قرار می‌گیرد؟**

$$S: S \leftarrow 0 \quad , \quad F \leftarrow 1 \quad , \quad D \leftarrow 0 \quad , \quad CNTR \leftarrow 0$$
$$F: f(A \geq B)then(A \leftarrow A-B \ , \ CNTR \leftarrow CNTR+1)else(F \leftarrow 0 \ , \ D \leftarrow 1)$$
$$D: halt$$

۱) تقسیم A بر B و خارج قسمت در A
۲) تقسیم A بر B و خارج قسمت در B
۳) تقسیم B بر A و خارج قسمت در CNTR
۴) تقسیم A بر B و خارج قسمت در CNTR و باقیمانده در A

☑ **پاسخ:** گزینه «۴» در شرط F تا زمانی که A بزرگ‌تر یا مساوی B باشد، مقدار B را از A کم می‌کند و الگوریتم تا جایی ادامه می‌یابد که A کوچک‌تر از B شود. مشابه آن‌چه قبلاً بیان شد، مجموعه دستورات تقسیم A بر B را پیاده‌سازی می‌کند که در نهایت مقدار خارج قسمت در CNTR و باقی‌مانده در A قرار می‌گیرد زیرا هر بار مقدار B از A کاسته شده و نتیجه در ثبات A ذخیره می‌شود.

---

**مثال۶: یک مکانیزم سخت‌افزاری دارای دو ثبات A و B و چهار فلیپ‌فلاپ p،s،e و q توسط ریز دستورات زیر توصیف شده است. کدام گزینه در مورد عملکرد این مکانیزم صحیح است؟**

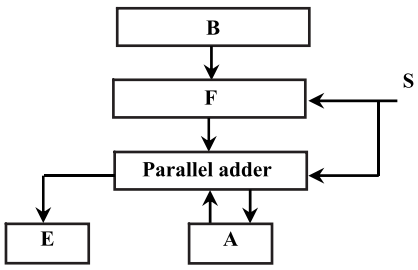
$$s: s \leftarrow 0 \ , \ e \leftarrow 0 \ , \ p \leftarrow 1 \ , \ q \leftarrow 0$$
$$p: p \leftarrow 0 \ , \ IF(A < B)then(e \leftarrow 1)else(q \leftarrow 1)$$
$$q: A \leftarrow A-B \ , \ p \leftarrow 1 \ , \ q \leftarrow 0$$
$$e: halt$$

۱) این مکانیزم حاصل تفریق $A-B$ را محاسبه و در A قرار می‌دهد.
۲) این مکانیزم باقیمانده تقسیم صحیح A بر B را محاسبه و در A قرار می‌دهد.
۳) از نظر سخت‌افزاری پیاده‌سازی مکانیزم توصیف شده به خاطر تناقض در عملیات غیر ممکن است.
۴) از نظر سخت‌افزاری پیاده‌سازی مکانیزم توصیف شده به خاطر عدم وجود ترتیب زمانی عملیات غیر ممکن است.

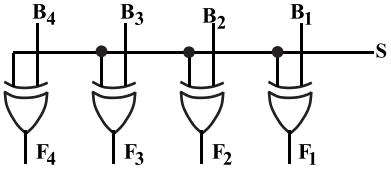
☑ **پاسخ:** گزینه «۲» این RTL با سیگنال s شروع و q خاتمه می‌یابد. مشابه مثال قبل عملیات تقسیم $A/B$ را با تفریق‌های متوالی انجام داده و در پایان باقیمانده در رجیستر A قرار می‌گیرد.

---

**مثال۷: مدار 4 بیتی ریاضی زیر با یک خط کنترلی S و دو ثبات 4 بیتی A و B مفروض است به جای بلوک F مداری با کدام یک از ترکیبهای زیر را قرار دهیم تا وقتی که $S=0$ باشد مدار عمل $A+B$ و وقتی $S=1$ باشد، مدار $A-B$ را با استفاده از مکمل دوی ثبات B انجام دهد.**




۱) ۴ دروازه NOT
۲) ۴ دروازه XNOR
۳) ۴ دروازه XOR
۴) هیچکدام



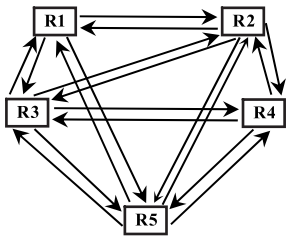

☑ **پاسخ:** گزینه «۳» می‌توان F را چهار دروازه XOR فرض کرد که خط کنترلی s را با تک‌تک بیت‌های B، XOR می‌کند. در این حالت اگر $S=0$ باشد، B به جمع‌کننده وارد می‌شود و اگر $S=1$ باشد، $\overline{B}+1$ یا همان مکمل 2 عدد B به جمع‌کننده وارد می‌شود به این ترتیب عمل تفریق را می‌توان انجام داد. شکل روبه‌رو F را نشان می‌دهد.

**مثال۸: در عملیات شیفت برای نمایش سرریز از رابطه $V = R_{n-1} \oplus R_{n-2}$ استفاده می‌شود. برای اینکه در یک شیفت به چپ سرریز رخ دهد مقدار V باید چه عددی را به خود اختصاص دهد؟**

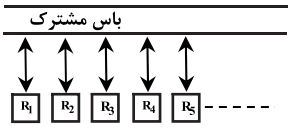
۱) $V = 1$　　۲) $V = 0$　　۳) $V = R_{n-1}$　　۴) $V = R_{n-2}$

☑ **پاسخ:** گزینه «۱» اگر $R_{n-1}$ و $R_{n-2}$ با هم متفاوت باشد، در شیفت به چپ سر ریز رخ می‌دهد. زیرا بیت علامت عوض می‌شود. در این حالت بیت سرریز (V) برابر یک می‌شود.

## گذرگاه(BUS)




برای ارتباط بین ثبات‌های مختلف یک سیستم و انجام ریز عملیات بیان شده، ثبات‌ها باید بتوانند اطلاعات خود را به یکدیگر منتقل نمایند. برای انجام این کار می‌توان تمام ثبات‌ها را مستقیماً به یکدیگر متصل کرد اما در این حالت با زیاد شدن تعداد ثبات‌ها، هزینه بسیار زیادی برای ایجاد اتصالات نیاز خواهد بود. به عنوان مثال اگر سیستم شامل پنج ثبات باشد آن‌گاه نحوه ارتباط آن‌ها به صورت روبه‌رو خواهد بود:



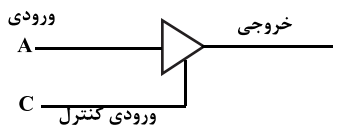

روش دیگر ایجاد ارتباط بین ثبات‌ها استفاده از یک گذرگاه مشترک بین آن‌هاست که تمام ثبات‌ها می‌توانند داده‌های خود را بر روی این گذرگاه مشترک قرار دهند و یا ورودی خود را از این گذرگاه دریافت نمایند. شکل روبرو یک حالت کلی از گذرگاه مشترک را نمایش می‌دهد.

در صورت استفاده از گذرگاه مشترک هزینه سخت‌افزاری ایجاد ارتباط بین ثبات‌ها کمتر خواهد بود اما مدار کنترلی پیچیده‌تر خواهد شد. در ادامه چگونگی ساخت سخت‌افزار لازم برای ایجاد یک گذرگاه مشترک به دو روش شرح داده شده است.

## پیاده‌سازی گذرگاه مشترک به وسیله گیت‌های بافر سه حالته

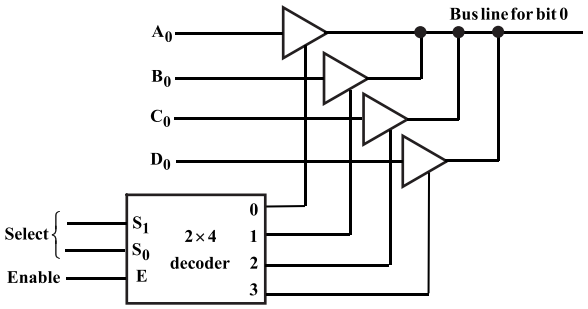
یک گیت بافر سه حالته (tri - state buffer) دارای یک ورودی معمولی، یک ورودی کنترلی و یک خروجی است که خروجی آن توسط مقدار ورودی و همچنین ورودی کنترلی تعیین می‌گردد.

در شکل زیر یک گیت بافر سه حالته نشان داده شده است.




| خروجی | A | ورودی کنترلی C |
|---|---|---|
| امپدانس بالا | 0 | 0 |
| امپدانس بالا | 1 | 0 |
| 0 | 0 | 1 |
| 1 | 1 | 1 |

در صورتی‌که ورودی کنترل برابر یک باشد آن‌گاه خروجی مدار همان ورودی A می‌باشد و اگر ورودی کنترلی برابر صفر باشد آن‌گاه خروجی مدار غیر فعال و در حالت امپدانس بالا (high impedance) می‌باشد.




با توجه به ویژگی مهم گیت‌های بافر سه حالته که می‌توانند دارای خروجی در حالت غیرفعال باشند، می‌توان خروجی چند گیت بافر سه حالته را به یکدیگر متصل نمود با این شرط که فقط ورودی کنترل یکی از آن‌ها برابر یک باشد.

در حالت کلی برای پیاده‌سازی یک گذرگاه مشترک که برای ایجاد ارتباط بین n عدد ثبات k بیتی قابل استفاده باشد، از k سطح از بافرهای سه حالته استفاده می‌گردد که در هر سطح n عدد گیت بافر سه حالته قرار دارد و همچنین به یک دیکدر با قابلیت انتخاب n حالت به عنوان خطوط کنترل گیت‌های سه حالته نیاز می‌باشد.

به عنوان مثال در شکل فوق یک سطح از گیت‌های بافر سه حالته برای ایجاد گذرگاه مشترک مشاهده می‌شود. این گذرگاه برای سیستمی که متشکل از چهار ثبات A، B، C و D می‌باشد، نشان داده شده است. در این شکل بیت اول از هر ثبات نشان داده شده است و به تعداد بیت‌های ثبات‌ها به سطوح این چنینی از گیت‌های بافر سه حالته نیاز خواهیم داشت که در هر لحظه با توجه به خطوط انتخاب دیکدر استفاده شده، بیت‌های یکی از ثبات‌ها بر روی گذرگاه قرار خواهد گرفت.

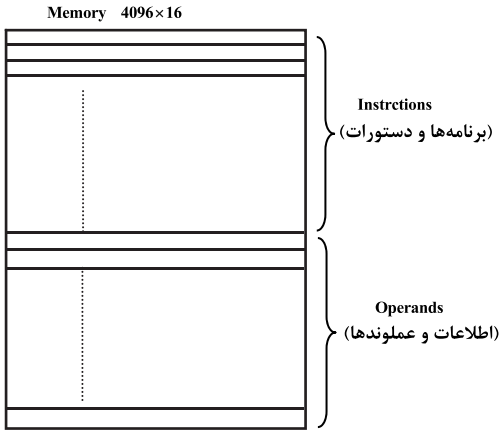
مدرسان شریف

# فصل چهارم

## «تشریح کامپیوتر پایه و پیاده‌سازی واحد کنترل»

### مقدمه

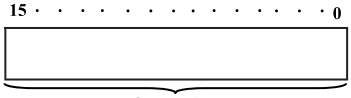
در این فصل یک کامپیوتر پایه به همراه اجزای اصلی آن معرفی می‌گردد. پس از آن به دو روش کلی طراحی واحد کنترل یعنی روش سخت‌افزاری و روش نرم‌افزاری اشاره خواهد شد و چگونگی پیاده‌سازی این دو روش به همراه مزایا و معایب آن‌ها تشریح می‌شود. برای درک بهتر بخش‌هایی از واحد کنترل ساختار کامپیوتر پایه نامبرده با استفاده از دو روش فوق پیاده‌سازی می‌گردد. مبحث وقفه نیز به صورت اجمالی در این فصل تشریح خواهد شد و در پایان فصل نیز حافظه نانو و کاربرد آن به همراه مثال‌هایی ذکر می‌شود.



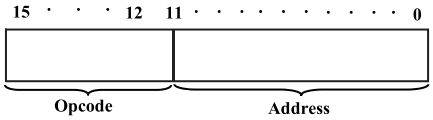

در ابتدا یک کامپیوتر پایه و تشکیلات مربوط به آن را تشریح می‌کنیم و در نهایت مطالب مربوط به کنترل ریز برنامه‌ریزی شده و مقایسه آن با کنترل سخت‌افزاری بحث می‌گردد. هر دستور کامپیوتر یک کد عملیاتی (Instruction code) نامیده می‌شود، این کد عملیاتی دنباله‌ای باینری است که نشان‌دهنده‌ی نوع عملیات موردنظر و آدرس عملوند مورد نیاز و همچنین آدرس مکانی از حافظه که نتیجه عملیات در آن ذخیره می‌گردد (در صورت لزوم)، می‌باشد. اصولاً فرض بر این است که دستورات کامپیوتر و اطلاعات مربوط به عملوندها در قسمت‌های جداگانه حافظه ذخیره شده‌اند. اگر در کامپیوتر پایه موردنظر از یک حافظه با 4096 کلمه 16 بیتی استفاده شده باشد آن‌گاه می‌توان شکل کلی حافظه را به صورت روبرو در نظر گرفت:

بنابراین هر دستورالعمل (Instruction) دارای 16 بیت می‌باشد و از آن‌جا که حافظه دارای $4096 = 2^{12}$ کلمه است بنابراین 12 بیت برای آدرس‌دهی حافظه مورد نیاز است و در نتیجه 4 بیت برای تعیین کد عملیات (op-code) در هر دستور قابل استفاده می‌باشد. شکل کلی یک عملوند و یک کد دستورالعمل را می‌توان به صورت زیر در نظر گرفت:

شکل کلی عملوند (operand)



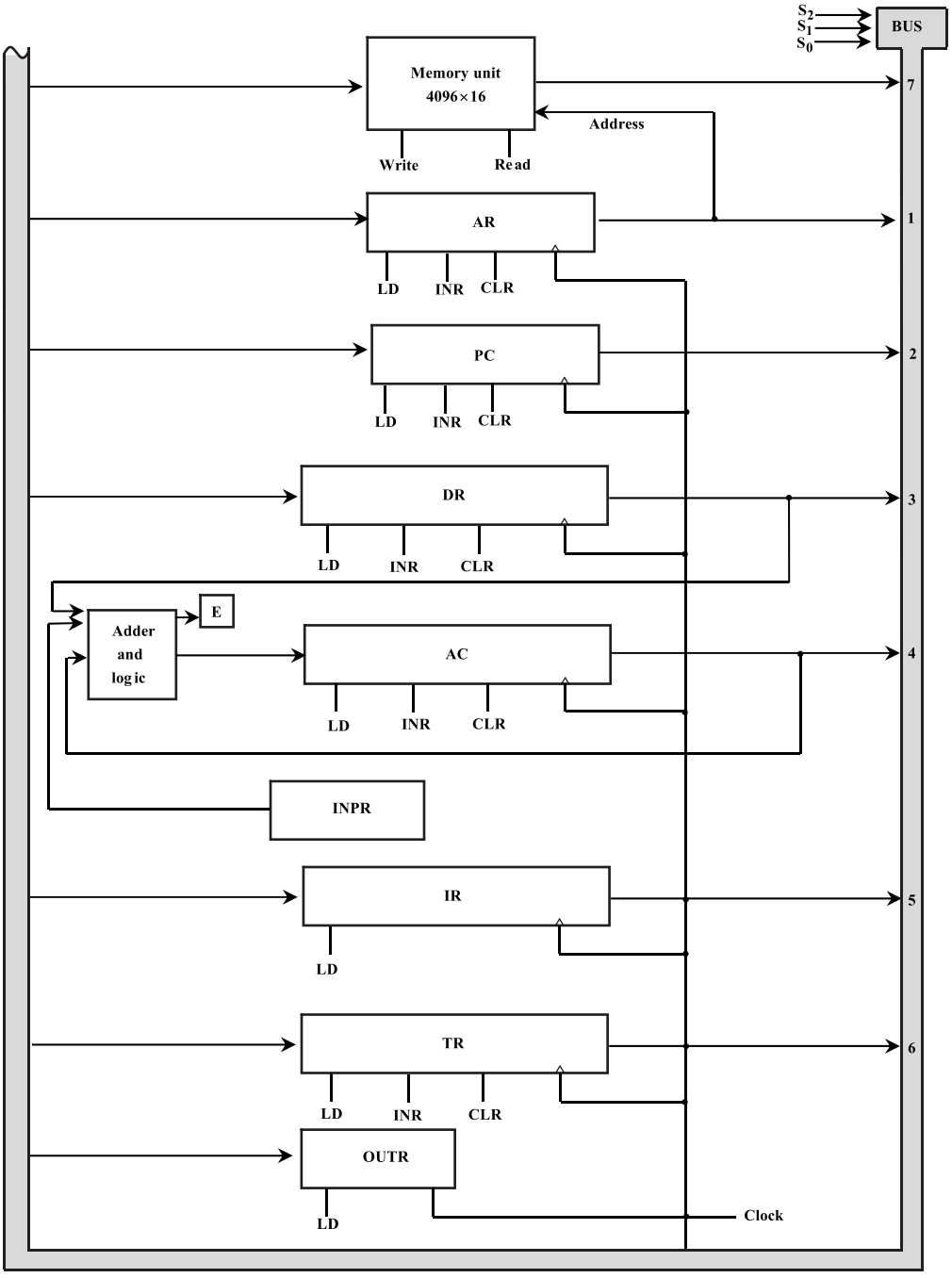

شکل کلی کد دستورالعمل
(Instruction format)




در دستوراتی که به آدرس‌دهی نیاز ندارند، قسمت 12 بیتی استفاده شده برای آدرس می‌تواند جهت اهداف دیگر مانند تعیین جزئی‌تر نوع دستورالعمل مورد بهره‌برداری قرار گیرد. انواع ثبات‌هایی که در کامپیوتر پایه مورد استفاده قرار می‌گیرد، در جدول زیر ارائه شده است.

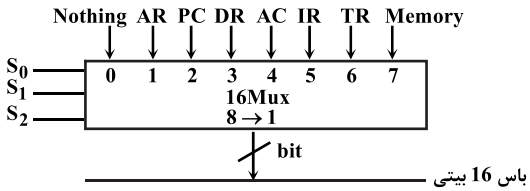
| عملکرد | نام ثبات | تعداد بیت‌ها | نماد ثبات |
|---|---|---|---|
| نگهداری عملوند حافظه | Data register | 16 | DR |
| نگهداری آدرس حافظه | Address register | 12 | AR |
| ثبات پردازنده | Accumulator | 16 | AC |
| نگهدای کد دستورالعمل | Instruction register | 16 | IR |
| نگهدای آدرس دستورالعمل | Program counter | 12 | PC |
| نگهداری داده‌های موقت | Temporary register | 16 | TR |
| نگهداری کاراکتر ورودی | Input register | 8 | INPR |
| نگهداری کاراکتر خروجی | Out put register | 8 | OUTR |

ثبات‌های PC و AR برای آدرس‌دهی به حافظه استفاده می‌گردند بنابراین هر کدام 12 بیتی می‌باشند، ثبات PC در هر لحظه آدرس دستوری را که باید پس از دستور در حال اجرا، اجرا شود مشخص می‌نماید و ثبات AR برای آدرس‌دهی به حافظه در عمل خواندن و یا نوشتن مورد استفاده قرار می‌گیرد. ثبات IR نگه‌دارنده دستور خوانده شده می‌باشد بنابراین 16 بیتی است و ثبات DR نیز حاوی اطلاعات خوانده شده از حافظه خواهد بود. ثبات AC یک ثبات همه منظوره است و ثبات TR برای ذخیره نتایج موقت در محاسبات استفاده می‌گردد همچنین فیلیپ‌فلاپ E برای نتیجه رقم نقلی ایجاد و در طول محاسبات استفاده شده است. نحوه ارتباط بین ثبات‌های مطرح شده توسط یک گذرگاه مشترک (که نحوه‌ی پیاده‌سازی آن در فصل قبل بررسی شد) در شکل زیر آمده است.




همان‌گونه که در شکل مشخص می‌باشد، یک گذرگاه مشترک 16 بیتی برای ایجاد ارتباط بین ثبات‌ها استفاده شده است که با استفاده از خطوط $S_2\,S_1\,S_0$ در هر لحظه یکی از ثبات‌ها می‌تواند انتخاب گردد. در صورت انتخاب یک ثبات و فعال بودن خط بار کردن آن (LD) اطلاعات موجود بر روی باس (گذرگاه مشترک) در آن ثبات قرار می‌گیرد.

با توجه به این که ثبات AC ورودی مستقیمی از باس دریافت نمی‌کند بنابراین اطلاعات خوانده شده از حافظه را می‌توان به تمام ثبات‌ها به جز AC وارد نمود. دقت کنید که ثبات‌های OUTR و INPR هشت بیتی‌اند و تنها با هشت بیت کم‌ارزش باس تبادل اطلاعات می‌نمایند و با استدلالی مشابه ثبات‌های AR و PC با 12 بیت کم‌ارزش باس تبادل اطلاعات می‌نمایند. در صورتی که ورودی INR یک ثبات فعال باشد آن‌گاه محتوای ثبات مربوطه یک واحد افزایش می‌یابد. ورودی CLR برای پاک کردن محتوای یک ثبات استفاده می‌گردد.




باس ذکر شده با استفاده از MUX و یا بافر سه حالته پیاده‌سازی می‌شود. خطوط خروجی باس با سه سیگنال $S_0$، $S_1$ و $S_2$ کنترل می‌شود. در صورت پیاده‌سازی باس با استفاده از MUX، ساختار آن به صورت شکل روبرو است. دقت کنید که خروجی باس به تمامی ورودی‌های ثبات‌ها (به جز AC) و ورودی‌های حافظه متصل است. البته همان طور که در شکل مشاهده می‌شود، ثبات AC می‌تواند داده‌های خود را بر روی گذرگاه مشترک قرار دهد.

با توجه به خطوط انتخاب شرح داده شده می‌توان شکل زیر را در مورد نحوه عملکرد خطوط انتخاب در نظر گرفت.

به طور خلاصه اجزای کامپیوتر پایه‌ای که در این فصل شرح داده شده است، به صورت زیر می‌باشد:

۱ـ یک حافظه با اندازه $4096\times16$ bit

۲ـ نه عدد ثبات: $SC, INPR, OUTR, TR, IR, AC, DR, PC, AR$

۳ـ هفت فیلیپ‌فلاپ: $FGO, FGI, IEN, R, E, S, I$

۴ـ دو عدد دیکدر: یک دیکدر $3\times8$ برای دیکد نمودن opcode دستورات و دیگری $4\times16$ برای ایجاد سیگنال‌های زمانی

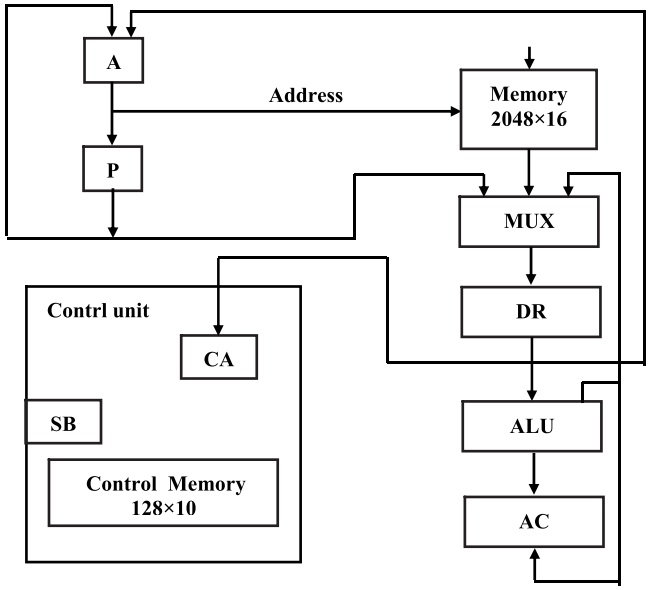
۵ـ یک باس مشترک 16 بیتی

۶ـ گیت‌های منطقی کنترل

برای این که هر یک از ثبات‌ها محتوای باس را دریافت نمایند (به جز ثبات‌های AC و INPR که مستقیماً از باس ورودی دریافت نمی‌کنند)، کافیست خط LD آن‌ها فعال گردد و برای این که محتوای باس در حافظه نوشته شود باید خط write حافظه فعال باشد. همچنین برای این که محتوای یک ثبات بر روی باس قرار گیرد کافیست که با استفاده از خطوط انتخاب $S_2S_1S_0$ ثبات موردنظر انتخاب گردد. قرار دادن محتویات حافظه بر روی باس توسط خطوط $S_2S_1S_0$ و فعال شدن خط Read انجام می‌شود. جدول زیر نحوه‌ی انتخاب ثبات‌های مختلف و حافظه را توسط خطوط $S_2S_1S_0$ نشان می‌دهد.

| $S_2$ | $S_1$ | $S_0$ | ثبات انتخاب شده برای قرار دادن محتویات بر روی باس |
|---|---|---|---|
| 0 | 0 | 0 | هیچ‌کدام |
| 0 | 0 | 1 | AR |
| 0 | 1 | 0 | PC |
| 0 | 1 | 1 | DR |
| 1 | 0 | 0 | AC |
| 1 | 0 | 1 | IR |
| 1 | 1 | 0 | TR |
| 1 | 1 | 1 | Memory |

**مثال۱: تعداد بیت‌های لازم برای رجیسترهای DR، AR، AC، CAR و SBR و همچنین بیت‌های PC را مشخص کنید.**




**AR : Address Register**

۱) $AC=16,\ CR=16$
$SBR=11,\ CAR=11$
$PC=12,\ AR=12$

**PC : Program Counter**

۲) $AC=16,\ CR=12$
$SBR=7,\ CAR=16$
$PC=12,\ AR=11$

**AC : Accumulator**

۳) $AC=12,\ CR=12$
$SBR=16,\ CAR=16$
$PC=11,\ AR=11$

**DR : Data Register**

۴) $AC=16,\ CR=16$
$SBR=7,\ CAR=7$
$PC=11,\ AR=11$
$AC=16,\ DC=16$

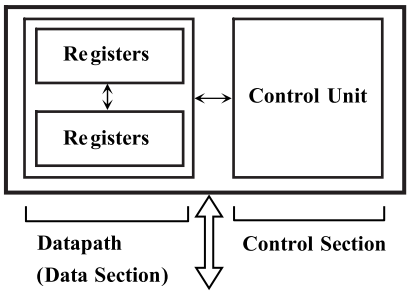
مدرسان شریف

# فصل پنجم

## «واحد پردازش مرکزی CPU»

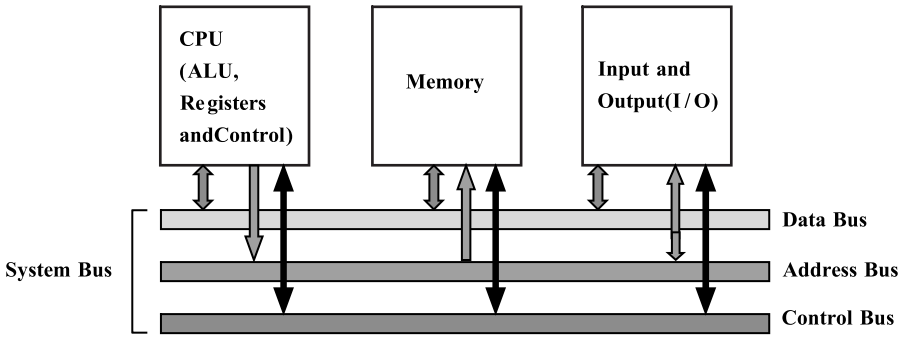
### مقدمه

این فصل به پردازنده و مسایل درون آن اختصاص دارد، نخست با معماری کلی داخلی پردازنده و چگونگی ارتباط آن با ماژول‌های بیرونی یعنی حافظه و ورودی ـ خروجی‌ها آشنا خواهیم شد. سپس به بررسی معماری‌های مختلف پردازنده از حیث تعداد عملوندها در دستورات پردازنده خواهیم پرداخت. پس از آن ساختار پشته معرفی می‌گردد و نحوه استفاده از آن برای محاسبات ریاضی شرح داده می‌شود. انواع آدرس‌دهی و مبحث وقفه موضوعات دیگری هستند که در ادامه فصل به آن‌ها اشاره می‌شود. سرانجام دو معماری متفاوت در پردازنده‌ها یعنی RISC و CISC معرفی، تشریح و مزایا و معایب هر کدام بیان می‌شود.



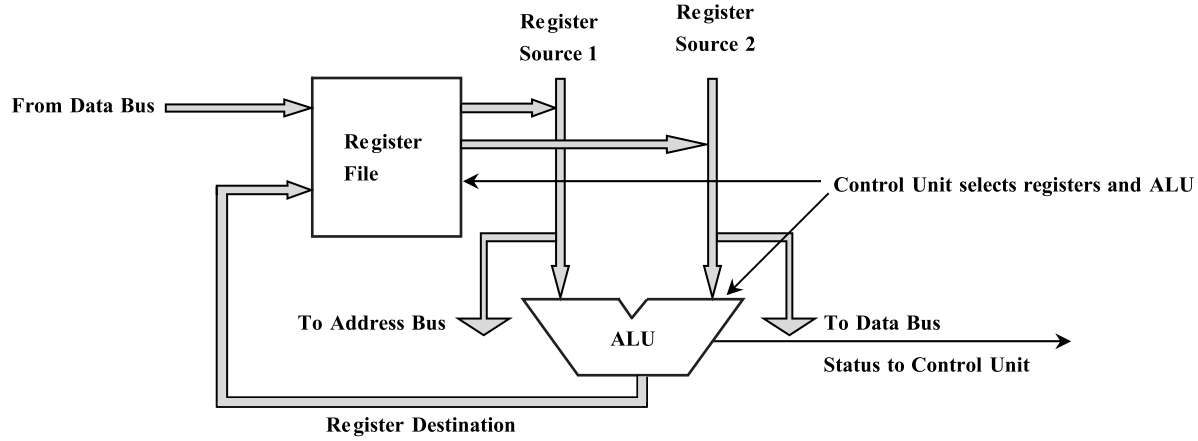

واحد پردازش مرکزی (Central Process Unit) ، عمل پردازش داده‌ها را در یک سیستم کامپیوتری انجام می‌دهد؛ واحد پردازش مرکزی را می‌توان متشکل از سه مؤلفه اصلی دانست که عبارتند از واحد کنترل، مجموعه ثبات‌های CPU و واحد محاسباتی ـ منطقی (Arithme Logic Unit) این سه مؤلفه را می‌توان در دو بخش کنترل (Control section) و بخش داده (data section) قرار داد. در برخی موارد به مجموعه ثبات‌های پردازنده، register file نیز می‌گویند.

بخش داده شامل رجیسترهای CPU و ALU می‌باشد. بخش کنترل وظیفه تفسیر دستورالعمل‌ها و تولید سیگنال‌های کنترلی برای انجام صحیح این دستورالعمل‌ها را بر عهده دارد. ثبات‌هایPC و IR و در برخی موارد DR ، ارتباط اصلی بین این بخش داده و کنترل را انجام می‌دهند. در حالت کلی دستورات از حافظه خوانده و به CPU منتقل می‌شوند. پس از تفسیر و اجرای ریز عملیات مورد نظر و انجام محاسبات در واحد ALU نتیجه به حافظه منتقل می‌گردد، ارتباطی مشابه آنچه بین پردازنده و حافظه وجود دارد، بین CPU و دستگاه‌های ورودی/ خروجی (I/O) نیز می‌تواند رخ دهد. این نحوه ارتباط در شکل زیر نشان داده شده است.




بدیهی است که حافظه مستقیماً آدرسی ایجاد نمی‌کند و آدرس دهی حافظه نیز از طریق CPU انجام می‌شود، بنابراین ارتباط بین گذرگاه آدرس (Address Bus) با CPU و حافظه به صورت یک‌طرفه از سمت پردازنده به گذرگاه آدرس و سپس حافظه خواهد بود. در حقیقت مجموعه رجیسترهای CPU را می‌توان به عنوان یک حافظه موقتی کوچک و سریع در نظر گرفت که مستقل از حافظه اصلی برای ذخیره نتایج موقتی استفاده می‌شوند. برای دسترسی به هر یک از رجیسترهای CPU نیز باید یک آدرس مناسب ایجاد گردد. به دلیل تعداد کم این رجیسترها، طول آدرس آنها بسیار کوتاه خواهد بود و به راحتی توسط یک مالتی پلکسر تولید می‌شوند.

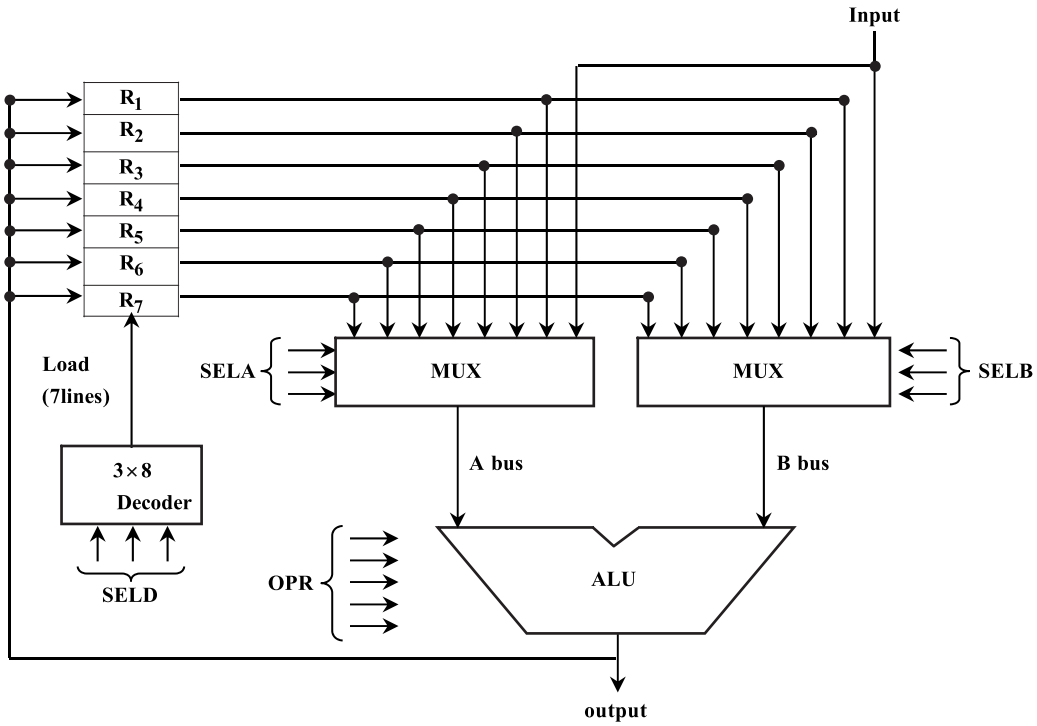
نحوه ارتباط بین قسمت‌های داخلی مسیر داده (data path) را می‌توان به صورت زیر در نظر گرفت.




در این شکل با توجه به سیگنال‌های کنترلی تعیین شده توسط واحد کنترلی، دو ورودی برای ALU به عنوان دو رجیستر منبع انتخاب می‌شوند و نتیجه نهایی از ALU به رجیستر مقصد منتقل می‌گردد. خطوط ضخیم نشان‌دهنده گذرگاه مشترک (Bus) می‌باشند که می‌توان آن را با یکی از روش‌های بررسی شده در فصل سوم پیاده سازی نمود.

برای انتخاب هر یک از رجیسترهای ورودی ALU می‌توان از یک مالتی پلکسر استفاده نمود و برای انتخاب رجیستر مقصد برای انتقال خروجی ALU کافیست از یک دیکدر برای آدرس‌دهی استفاده شود.

اگرهفت ثبات $R_1$ تا $R_7$ را در قسمت مسیر داده (data path) در نظر بگیریم آنگاه می‌توان نحوه اتصال قسمت‌های بخش داده را که شامل چگونگی ارتباط بین ALU و ثبات‌های پردازنده می‌باشد، به صورت زیر در نظر گرفت.




بیت‌های OPR در این شکل به منظور تعیین نوع عملیات محاسباتی و منطقی در ALU استفاده شده‌اند، هم چنین توسط دو مالتی پلکسر ثبات‌های ورودی ALU تعیین می‌شوند و با استفاده از دیکدر $3\times8$ موجود، خط بار کردن (load) یکی از ثبات‌ها به عنوان ثبات مقصد انتخاب می‌گردد.

به عنوان مثال برای انجام عمل $R_6 \leftarrow R_4 - R_5$ واحد کنترل باید با ایجاد سیگنال‌های کنترلی عملیات زیر را آغاز نماید:

- تعیین خطوط SELA به طوری که باعث انتخاب ثبات $R_4$ به عنوان خروجی MUX اول شود.
- تعیین خطوط SELB به طوری که باعث انتخاب ثبات $R_5$ به عنوان خروجی MUX دوم شود.
- تعیین بیت‌های OPR به طوری که باعث انجام تفریق $A-B$ گردد.
- تعیین خطوط SELD به طوری که خط بار کردن مربوط به ثبات $R_6$ فعال گردد.

بنابراین واحد کنترل باید ۹ بیت جهت تعیین خطوط انتخاب SELA، SELB و SELD و هم چنین ۵ بیت برای تعیین نوع عملیات ALU (که در شکل با OPR نشان داده شده است) را به عنوان یک کلمه کنترلی (control word) تولید نماید. بنابراین شکل کلی یک کلمه کنترلی را می‌توان به صورت زیر در نظر گرفت:

| 3-bit | 3-bit | 3-bit | 5-bit |
|---|---|---|---|
| SELA | SELB | SELD | ORR |

خطوط SELA، SELB و SELD طبق جدول زیر رجیستر مورد نظر را انتخاب می‌نمایند.

| Binary Code | SELA | Input | SELD |
|---|---|---|---|
| 000 | Input | Input | None |
| 001 | R1 | R1 | R1 |
| 010 | R2 | R2 | R2 |
| 011 | R3 | R3 | R3 |
| 100 | R4 | R4 | R4 |
| 101 | R5 | R5 | R5 |
| 110 | R6 | R6 | R6 |
| 111 | R7 | R7 | R7 |

در حالتی که خطوط انتخاب SELA، SELB برابر "000" باشند آنگاه مالتی پلکسرهای موجود ورودی خارجی را به ALU منتقل می‌کنند و اگر SELD = 000 باشد آنگاه هیچ ثباتی به عنوان ثبات مقصد استفاده نمی‌شود. اما در این حالت باز هم خروجی ALU می‌تواند به خارج منتقل گردد و در خروجی پردازنده قرار دارد.

**مثال۱: از دیدگاه برنامه‌نویسی که دستورهای ماشین را مورد استفاده قرار می‌دهد، معماری کامپیوتر شامل کدام یک از موارد زیرمی‌باشد؟**

۱) فقط از ثبات تشکیل شده است.

۲) از واحد محاسبه و منطق همراه ثباتها تشکیل شده است.

۳) شامل مجموعه دستورات، روش‌های آدرس‌دهی، تشکیلات CPU و ثبات‌های عمومی می‌باشد.

۴) هیچکدام

☑ **پاسخ:** گزینه «۲» برنامه‌نویس برای برنامه خود به ثبات‌ها، دستورالعمل‌ها، پشته و ... نیاز دارد. اما اجزای دیگر داخل CPU از دید برنامه‌نویس مخفی است و وظیفه تفسیر و تولید سیگنال‌های کنترلی مناسب با سخت‌افزار است و برنامه‌نویس ارتباطی با آن ندارد.

---

**تذکر۱:** واحدشیفت می‌تواند در قسمت ALU قرار گیرد یا می‌توان یک واحد شیفت‌دهنده را قبل از ورودی ALU قرار داد. (که در این حالت ALU دارای قابلیت preshifting می‌باشد) یا واحد شیفت‌دهنده می‌تواند پس از خروجی ALU قرار بگیرد (در این حالت ALU دارای قابلیت postshifing می‌باشد).

عملیاتی که توسط بیت‌های OPR می‌تواند تعیین گردد، در جدول زیر آمده است، دقت نمایید که نماد A نشان‌دهنده ثبات انباره (AC) نمی‌باشد بلکه منظور ثباتی است که توسط مالتی پلکسر اول انتخاب شده است و نماد B نیز نشان‌دهنده ورودی گذرگاه B در ALU می‌باشد.

به عنوان مثال اگر OPR = 00001 باشد آنگاه یک واحد به ثباتی که توسط گذرگاه به ALU وارد شده اضافه می‌گردد.

| OPR Select | عملیات | نماد |
|---|---|---|
| 00000 | Transfer A | TSFA |
| 00001 | Increment A | INCA |
| 00010 | Add A+B | ADD |
| 00101 | Subtract A-B | SUB |
| 00110 | Decrement A | DECA |
| 01000 | AND A and B | AND |
| 01010 | OR A and B | OR |
| 01100 | XOR A and B | XOR |
| 01110 | Complement A | COMA |
| 10000 | Shift right A | SHEA |
| 11000 | Shift left A | SHLA |

حال فرض کنید مطلوب است عمل $R_1 \leftarrow R_2 - R_3$ انجام شود. کلمه کنترلی متناظر با آن به ترتیب زیر ایجاد می‌شود.

- ورودی A از ALU باید ثبات $R_2$ را مشخص نماید: $SELA = 010$
- ورودی B از ALU باید ثبات $R_3$ را مشخص نماید: $SELB = 011$
- نتیجه باید در ثبات $R_1$ قرار داده شود: $SELD = 001$
- ریز عملیات تفریق باید در ALU انجام شود: $OPR = 00101$

از کنار هم قراردادن 4 فیلد فوق طبق شکل ارایه شده برای کلمه کنترلی نمونه، کلمه موردنظر در این مثال تولید می‌شود.

در جدول زیر چند کلمه کنترلی دیگر به عنوان نمونه ارائه شده است:

| Microoperation | SELA | SELB | SELD | OPR | کلمه کنترلی |
|---|---|---|---|---|---|
| R1←R2-R3 | R2 | R3 | R1 | SUB | 010 011 001 00101 |
| R4←R4∨R5 | R4 | R5 | R4 | OR | 100 101 100 01010 |
| R6←R6+1 | R6 | - | R6 | INCA | 110 000 110 00001 |
| R7←R1 | R1 | - | R7 | TSFA | 001 000 111 00000 |
| Output←R2 | R1 | - | None | TSFA | 010 000 000 00000 |
| Output←Input | Input | - | None | TSFA | 000 000 000 00000 |
| R4←sh1 R4 | R4 | - | R4 | SHLA | 100 000 100 11000 |
| R5←0 | R5 | R5 | R5 | XOR | 101 101 101 01100 |

در حالتی که فقط نیاز به یک ورودی در ALU دارند مانند افزایش یک واحدی یک رجیستر (INC)، انتقال اطلاعات یک رجیستر خاص به خروجی یا به یک رجیستر دیگر (TSFA)، تنها ورودی A از ALU استفاده می‌شود. در قسمت SELB از یک خط تیره استفاده شده است زیرا در این حالت مقدار SELB اهمیتی ندارد و در کلمه کنترلی متناظر مقدار "000" برای آن نوشته می‌شود، این مقدار اختیاری و انتخاب هر مقدار دیگری نیز امکان‌پذیر است.

**مثال۲: چرا در خیلی از کامپیوترها فقط شیفت منطقی به چپ وجود دارد و شیفت ریاضی به چپ به کار نمی‌رود؟**

۱) زیرا در شیفت ریاضی به چپ محتوای یک کلمه داخل یک ثبات قرار نمی‌گیرد.

۲) زیرا شیفت ریاضی به سمت چپ شبیه شیفت منطقی می‌باشد و از طرف کوچکترین بیت صفر وارد می‌شود.

۳) زیرا در شیفت ریاضی به چپ اعداد مکمل ۲ بکار نمی‌روند.

۴) هیچکدام

☑ **پاسخ:** گزینه «۲» همان‌طور که در بخش‌های قبلی گفته شد شیفت به چپ منطقی با شیفت به چپ ریاضی مشابه است و به همین دلیل می‌توان از شیفت منطقی به چپ به جای هر دو استفاده نمود. لازم به ذکر است که در شیفت ریاضی به چپ، رخ دادن سرریز نیز کنترل می‌گردد اما به هر حال در عمل تفاوتی با شیفت منطقی به چپ ندارد.

## تعداد آدرس‌ها در دستورالعمل‌های ماشین

به صورت سنتی یکی از عوامل مورد استفاده در دسته‌بندی معماری کامپیوترها، تعداد آدرس‌های استفاده شده در هر دستور می‌باشد در حالت کلی می‌توان گفت دستورات منطقی و محاسباتی بیشترین تعداد آدرس را نیاز دارند، در این نوع دستورات حداکثر دو عملوند مورد نیاز می‌باشد: دو آدرس برای عملوندها و یک آدرس نیز برای محاسبه نتیجه چنانچه آدرس دستور بعدی را نیز به عنوان جزئی از آدرس‌های یک دستورالعمل در نظر بگیریم، می‌توان گفت یک دستور ماشین حداکثر به چهار آدرس نیاز خواهد داشت. آدرس دستورالعمل بعدی معمولاً از طریق ثبات pc محاسبه می‌گردد. بنابراین در حالت کلی یک دستورالعمل می‌تواند حداکثر سه آدرس در خود داشته باشد.

تعداد آدرس‌ها در دستور به ساختار وابسته است، آنچه در بالا گفته شد می‌تواند در حالت کلی صادق باشد. هر چند می‌توان دستورالعمل‌هایی در ساختارهای خاص یافت که تعداد آدرس‌های آن‌ها حتی از 3 نیز بیشتر باشد اما اینگونه موارد بسیار معدود می‌باشند و بنابراین در بحث کلی قابل صرف‌نظر کردن هستند. آنچه به صورت متداول وجود دارد ساختارهایی با دستورهای 3 آدرسه، 2 آدرسه، 1 آدرسه و صفر آدرسه می‌باشد. در ادامه به این ساختارها خواهیم پرداخت. تعداد آدرس‌های دستورالعمل‌ها به تعداد ثبات‌های cpu بستگی دارد و می‌توان معماری‌ها را بر اساس این معیار در سه گروه زیر قرار داد:

- **Multi Register Organization** که cpu دارای چندین ثبات می‌باشد (در این حالت اصولاً دستورات دو آدرسی یا سه آدرسی استفاده می‌گردد).
- **Single Accumulator Organization** که cpu دارای یک ثبات انباره (Accumulator) همه منظوره می‌باشد (اصولاً در این حالت از دستورات تک آدرسی استفاده می‌شود).
- **Stack Organization** که cpu دارای حافظه پشته (stack) می باشد (اصولاً در این حالت از دستورات صفر آدرسی استفاده می‌گردد).

**دستورات سه آدرسی (three – addressed instructions)**

شکل کلی دستورات سه آدرسی به صورت مقابل می‌باشد. op , A , B C

که op یک عمل (operation) را مشخص می‌نماید و A، B و C می‌توانند نشان‌دهنده یک ثبات cpu و یا یک آدرس از حافظه باشند. تفسیر این دستور به صورت زیر می‌باشد.

$$\underbrace{op \qquad A,B,C}_{\text{شکل کلی دستور سه آدرسی}} \xrightarrow{\text{مفهوم}} A \leftarrow B \ op \ C$$

به عنوان مثال برنامه مربوط به محاسبه $X=(A+B)\times(C+D)$ در یک فرمت سه آدرسی به صورت زیر می‌باشد:

| Instruction | مفهوم |
|---|---|
| ADD R1 , A , B | R1←M[A]+M[B] |
| ADD R2 , C , D | R2←M[C]+M[D] |
| MUL X , R1 , R2 | M[X]←R1*R2 |

که در آن $R_1$ و $R_2$ ثباتهای cpu می‌باشند که در حقیقت برای ذخیره نتایج میانی استفاده می‌شوند. و منظور از M[A] عملوندی در حافظه است که با آدرس A مشخص می‌شود.

**مزیت:** طول برنامه‌ها کوچکتر می‌شود چرا که با تعداد دستورات کمتر نوشته می‌شود.

**عیب:** طول دستورات طولانی است.

**مثال۳: اشکال دستورهای سه آدرسی کدام یک از موارد زیر می‌باشد.**

۱) فرم باینری دستور دارای بیت‌های زیادی برای مشخص کردن سه آدرس است.

۲) کوتاه شدن برنامه محاسبات ریاضی

۳) استفاده از ثبات اکومولاتور بر روی داده‌ها

۴) هیچکدام

☑ **پاسخ:** گزینه «۱» با توجه به متن درس گزینه (۱) صحیح است.

---

**دستورات دو آدرسی (two – addressed instructions):**

شکل کلی دستورات دو آدرسی را می‌توان به صورت مقابل در نظر گرفت: op A,B

که مفهوم آن به صورت زیر می‌باشد.

$$op \quad A,B \xrightarrow{\text{مفهوم}} A \leftarrow A \ op \ B$$

بنابراین نتیجه اجرای عملگر موردنظر بر روی A و B در A قرار خواهد گرفت که می تواند یک ثبات cpu باشد و یا یک مکان از حافظه را مشخص نماید.

در زیر برنامه مربوطه به محاسبه $X=(A+B)\times(C+D)$ در یک فرمت دو آدرسی آمده است:

| Instruction | | مفهوم |
|---|---|---|
| MOV | R1 , A | R1←M[A] |
| ADD | R1 , B | R1←R1+M[A] |
| MOV | R2 , C | R2←M[C] |
| ADD | R2 , D | R2←R2+M[D] |
| MUL | R1 , R2 | R1←R1*R2 |
| MOV | X , R1 | M[X]←R1 |

**دستورات تک آدرسی (One – addressed instructions):**

صورت کلی دستورات تک آدرسی به شکل زیر می‌باشد:

$$op \quad A \xrightarrow{\text{مفهوم}} AC \leftarrow AC \ op \ A$$

در این حالت عملگر بر روی A و محتوای ثبات AC عمل می‌کند و نتیجه در ثبات AC ذخیره می‌شود.

مدرسان شریف

# فصل ششم

## «پردازش خط لوله‌ای (Pipeline processing)»

### مقدمه

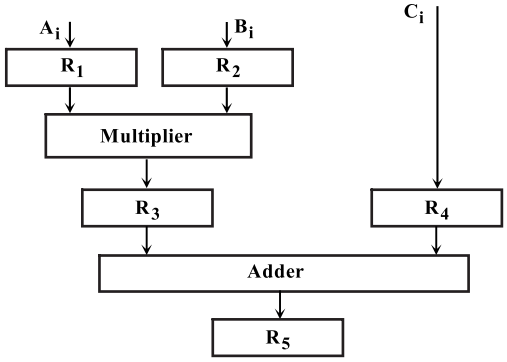
افزایش سرعت پردازش، کوتاه کردن زمان پاسخگویی و افزایش کارایی از اهداف همیشگی طراحان معماری کامپیوتر بوده است. یکی از تکنیک‌های پرکاربرد در این راستا، پردازش خط لوله‌ای می‌باشد که در این فصل بررسی می‌گردد. دو مبحث خط لوله برای محاسبات و دستورالعمل‌ها به صورت جداگانه تشریح می‌شود. مشکلات بالقوه موجود در خط لوله دوم یعنی دستورالعمل‌ها بیان و راهکار مقابله با آن‌ها نیز ارایه می‌گردد. در پایان به بحث پردازش برداری در راستای افزایش سرعت پردازش به ویژه در ابر کامپیوترها اشاره خواهد شد.

یکی از راه‌های بسیار مؤثر در افزایش کارایی یک سیستم پردازشی، استفاده از پردازش لوله‌ای می‌باشد. این الگو بسیار شبیه به رفتاری است که در خطوط تولید و ترکیب کردن قطعات برای رسیدن به یک محصول نهایی استفاده می‌شود. در کارخانجات تولیدی با شکستن محصول نهایی به اجزای متفاوت و تدارک یک خط تولید چند مرحله‌ای، تلاش می‌کنند تا در هر مرحله بخشی از محصول نهایی به سیستم اضافه گردد. به عنوان مثال خط تولید خودرو را در نظر بگیرید، در حالت معمولی می‌توان بخش‌های مختلف آن را بر روی میز قرار داد و تمامی کارگران همزمان بر روی آن آغاز به کار نمایند. بدیهی است در این حالت باید خودروی اول به مرحله پایانی برسد و سپس کارگران بر روی خودروی دوم مشغول به کار می‌شوند. پر واضح است که این روش بسیار غیر کارآمد است زیرا تمامی کارگران نمی‌توانند همزمان مشغول به کار باشند و در هر زمان چندین نفر بیکار خواهند بود و این به معنای کاهش کارایی و طولانی شدن تولید محصول خواهد بود. حالت دوم را در نظر بگیرید که ابتدا مراحل ترکیب قطعات خودرو به بخش‌های متوالی قابل جداسازی از هم تقسیم می‌شود. حال در هر بخش تعدادی کارگر مشغول به کارند. خودروی اول وارد مرحله نخست می‌شود. با پایان یافتن این مرحله، کارگران در بخش بعدی وظایف خود را انجام می‌دهند و در همین زمان خودروی دوم وارد مرحله اول می‌شود. به خوبی می‌توان دید که پس از n مرحله، یک خودرو آماده خواهد بود با این تفاوت که از این به بعد در هر سیکل کاری، یک خودرو در مرحله nام تولید می‌شود. با این روش با تقسیم‌بندی کل کار و موازی‌سازی بخش‌های قابل تفکیک، سرعت تولید محصول و کارایی خط تولید بهبود می‌یابد زیرا در هر لحظه کارگران در بخش مرتبط با تخصص خود مشغول خواهند بود. طراحان پردازنده‌ها با الهام از این ایده، تلاش کرده‌اند تا کارایی سیستم‌های دیجیتال را افزایش دهند، چنین ساختاری را پردازش خط لوله‌ای می‌گویند.

همان‌طور که در بالا دیدیم، در پردازش خط لوله‌ای پردازش‌های سریال به چند بخش تقسیم می‌شود. هر کدام از این بخش‌ها توسط یک قطعه (segment) که قابلیت انجام پردازش را دارد، اجرا می‌گردد. قطعات پردازشی موردنظر به صورت پشت سر هم قرار می‌گیرند و نتیجه پردازش هر قطعه به قطعه بعدی داده می‌شود. در پایان نتیجه نهایی به عنوان خروجی آخرین قطعه ایجاد خواهد شد. در این حالت همپوشانی در انجام پردازش‌های مختلف رخ خواهد داد و قطعات مختلف پردازش‌های متفاوتی را به صورت همزمان انجام می‌دهند. بنابراین انتظار می‌رود استفاده از این تکنیک باعث افزایش سرعت به صورت چشمگیری گردد. این همپوشانی در پردازش با استفاده از ثباتهایی که بین قطعه‌های مختلف قرار داده می‌شود، امکان پذیر خواهد بود. هر ثبات، خروجی قطعه قبلی را به عنوان ورودی بخش بعد در خود نگه می‌دارد.

یک پالس ساعت به قطعه‌های مختلف متصل می‌شود و بیشترین تأخیر بین دو ثبات که بیانگر زمان لازم برای انجام پردازش‌ها در بخش مربوطه می‌باشد، تعیین‌کننده مقدار پالس ساعت یا همان کلاک کاری سیستم می‌باشد. به بیان دیگر در هر بخش از یک ساختار لوله‌ای، تعدادی مدار ترکیبی خواهیم داشت. بدیهی است که با توجه به انجام پردازش‌های متفاوت در بخش‌های مختلف، میزان تأخیر آن‌ها نیز متفاوت خواهد بود. چنان‌چه بیشترین مقدار این تأخیرها برابر $t_\Delta$ باشد، $\frac{1}{t_\Delta}$ کلاک کاری سیستم خواهد بود. در هر کلاک، پردازش‌ها در طول خط لوله یک واحد پیش می‌روند و هر قطعه خروجی خود را به قطعه بعدی خواهد داد.

**مثال۱: فرض کنید نیاز به انجام عمل $A \times B + C$ برای هفت دسته داده عددی مختلف به عنوان A، B و C می‌باشد در این حالت خط لوله زیر برای انجام این عمل در نظر گرفته می‌شود:**




☑ **پاسخ:** ثبات‌های $R_1$ و $R_2$ قطعه اول، ثبات‌های $R_3$ و $R_4$ قطعه دوم و ثبات $R_5$ قطعه سوم را تشکیل می‌دهند، هر ثبات در هر پالس ساعت، داده جدیدی دریافت می‌کند. نقش هریک از سه قطعه به صورت زیر می‌باشد.

Segment 1 : $R_1 \leftarrow A_i$ و $R_2 \leftarrow B_i$ (قطعه اول) ($A_i$ و $B_i$ دریافت داده‌های)

Segment 2 : $R_3 \leftarrow R_1 \times R_2$ و $R_4 \leftarrow C_i$ (قطعه دوم) ($A_i \times B_i$ انجام ضرب و دریافت ورودی $C_i$)

Segment 3 : $R_5 \leftarrow R_3 + R_4$ (قطعه سوم) (انجام عمل جمع نهایی)

---

پر واضح است که ضرب و جمع همان مدارت ترکیبی هستند که بین دو ثبات در بخش‌های متفاوت قرار گرفته‌اند. با توجه به تأخیر بیشتر عمل ضرب، این بخش تعیین‌کننده کلاک کاری سیستم فوق خواهد بود.

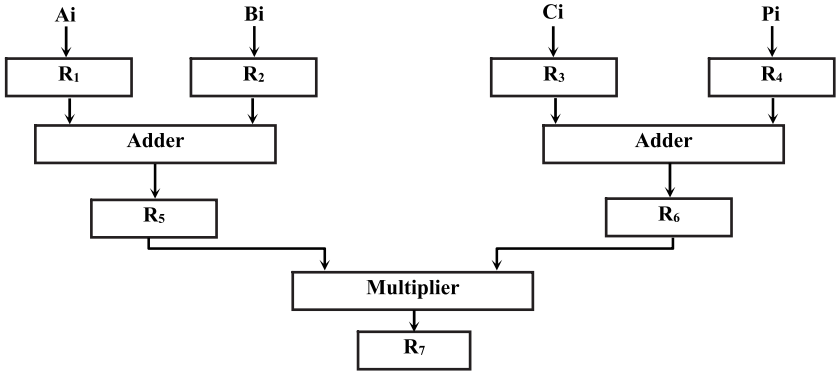
نحوه عملکرد این خط لوله به ازای هفت دسته ورودی به صورت زیر می‌باشد.

| Clock Pulse Number | Segment 1 | | Segment 2 | | Segment 3 |
|---|---|---|---|---|---|
| | $R_1$ | $R_2$ | $R_3$ | $R_4$ | $R_5$ |
| 1 | $A_1$ | $B_1$ | — | — | — |
| 2 | $A_2$ | $B_2$ | $A_1 * B_1$ | $C_1$ | — |
| 3 | $A_3$ | $B_3$ | $A_2 * B_2$ | $C_2$ | $A_1 * B_1 + C_1$ |
| 4 | $A_4$ | $B_4$ | $A_3 * B_3$ | $C_3$ | $A_2 * B_2 + C_2$ |
| 5 | $A_5$ | $B_5$ | $A_4 * B_4$ | $C_4$ | $A_3 * B_3 + C_3$ |
| 6 | $A_6$ | $B_6$ | $A_5 * B_5$ | $C_5$ | $A_4 * B_4 + C_4$ |
| 7 | $A_7$ | $B_7$ | $A_6 * B_6$ | $C_6$ | $A_5 * B_5 + C_5$ |
| 8 | — | — | $A_7 * B_7$ | $C_7$ | $A_6 * B_6 + C_6$ |
| 9 | — | — | — | — | $A_7 * B_7 + C_7$ |

همانگونه که ملاحظه می‌شود در کلاک اول $A_1$ و $B_1$ به ترتیب به $R_1$ و $R_2$ بار می‌شوند. در پالس دوم $A_2$ و $B_2$ توسط قطعه اول به ترتیب به $R_1$ و $R_2$ وارد می‌شوند. به طور همزمان با بارگذاری $R_2$ و $R_1$، قطعه دوم حاصلضرب $A_1 \times B_1$ را در $R_3$ قرار می‌دهد. ورودی $C_1$ نیز به ثبات $R_4$ بار می‌گردد. در کلاک سوم $A_3$ و $B_3$ توسط قطعه اول در ثبات‌های $R_1$ و $R_2$ قرار می‌گیرند و حاصلضرب $A_2 \times B_2$ توسط قطعه دوم محاسبه می‌شود، ورودی $C_2$ دریافت و در $R_4$ قرار می‌گیرد. عمل جمع $R_3 + R_4$ که دارای حاصل $A_1 \times B_1 + C_1$ می‌باشد، انجام و نتیجه در $R_5$ قرار می‌گیرد. همان گونه که مشاهده می‌شود در کلاک سوم خط لوله پر می‌شود. از این زمان به بعد در طی هر پالس ساعت یک خروجی تولید می‌گردد. همان‌طور که قبلاً نیز اشاره شد، این عملیات تا زمانی که آخرین خروجی محاسبه گردد، ادامه می‌یابد. در حالت کلی پس از پر شدن خط لوله، به ازای هر پالس ساعت، یک خروجی تولید می‌گردد.

**مثال۲: در یک سیستم خاص منظوره باید عملیات $(A_i + B_i)(C_i + D_i)$ روی جریانی از عددها اجرا شود. یک خط لوله طراحی کنید که این عمل را انجام دهد.**

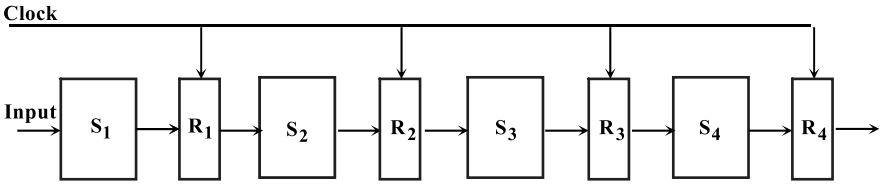
**پاسخ:**




بدیهی است که می‌توان عملیات را به دو بخش جداگانه که درون پرانتزها نشان داده شده‌اند، تقسیم‌بندی کرد. این دو بخش می‌توانند به صورت موازی اجرا گردند و از حاصل این دو عبارت، نتیجه نهایی را به دست آورد.

اصولاً عملیاتی که قابلیت تجزیه به چند زیر عملیات با پیچیدگی تقریباً یکسان را داشته باشد، می‌توان با تکنیک خط لوله پیاده سازی نمود. کارایی خط لوله در مواقعی که عملیات یکسانی به تعداد دفعات زیادی انجام می‌شود، مشهود خواهد بود.

شکل کلی یک خط لوله چهار طبقه را می‌توان به صورت زیر در نظر گرفت. در این شکل $R_i$ ثبات‌های واسط بین قطعات و $S_i$ در هر قطعه یک مدار ترکیبی است. $(i=1,2,3,4)$ ۳




به کل عملیاتی که در طول یک خط لوله انجام می‌شود، یک تکلیف (task) گفته می‌شود. برای بررسی نحوهٔ عملکرد یک خط لوله از نمودار فضا ــ زمان (space- time) استفاده می‌شود که در این نمودار، تکلیف موجود در هر قطعه از خط لوله با توجه به پالس ساعت‌های مختلف نشان داده می‌شود. به عنوان مثال برای خط لوله با چهار قطعه نمایش داده در شکل یک، نمودار فضا ـ زمان زیر نحوه انجام شش تکلیف $T_1$ تا $T_6$ را نشان می‌دهد.

| شماره پالس : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Segment 1 | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | | | |
| Segment 2 | | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | | |
| Segment 3 | | | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | |
| Segment 4 | | | | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |

این نمودار نشان می‌دهد که در کلاک اول تکلیف $T_1$ در قطعه اول قرار دارد، در کلاک دوم تکلیف $T_1$ در قطعه دوم و تکلیف $T_2$ در قطعه اول اجرا می‌شود. سرانجام در کلاک چهارم تکلیف $T_1$ در قطعه چهارم قرار می‌گیرد و خط لوله پر می‌شود. پس از پایان کلاک چهارم، در هر سیکل عملیات یک تکلیف تکمیل می‌گردد. محور عمودی قطعات مختلف خط لوله و محور افقی سیکل‌های متوالی را نشان می‌دهد.

### میزان تسریع (speed up) یک خط لوله

اگر یک خط لوله با $k$ سطح را در نظر بگیریم، همان‌طور که دیدیم $k$ پالس ساعت طول می‌کشد تا خط لوله پر شود. حال اگر $n$ دستور وجود داشته باشد آنگاه دستور اول برای تکمیل شدن به $k$ پالس ساعت نیاز خواهد داشت. مابقی دستورات هر کدام در یک پالس ساعت تکمیل می‌گردند بنابراین اگر مدت زمان یک پالس ساعت در خط لوله را $t_p$ در نظر بگیریم، زمان مورد نیاز برای انجام $n$ تکلیف به صورت زیر است:

دستور اول ↘ ، بقیه دستورات ⟵

$$t_{pipe} = [k + (n-1)] \times t_p$$

پیش‌تر گفته شد که $t_p$ برابر ماکزیمم تأخیر موجود در مدارهای ترکیبی قطعات مختلف خط لوله می‌باشد. حال اگر زمان پالس ساعت در حالت بدون خط لوله را $t_{unp}$ در نظر بگیریم، زمان موردنیاز بدون خط لوله به صورت مقابل خواهد بود: $t_{unpipe} = n \times t_{unp}$

مدرسان شریف

# فصل نهم

## «اندازه‌گیری کارایی (performance)»

کارایی یک سیستم کامپیوتری می‌تواند بر اساس معیارهای مختلفی مانند سرعت حافظه، سرعت پردازنده، حجم حافظه نهان و ... تعیین شود. در این کتاب کارایی یک سیستم کامپیوتری در حالت کلی بر اساس سرعت اجرای دستورات معنا پیدا می‌کند.

تمام عملیاتی که توسط یک پردازنده اجرا می‌شوند (از قبیل واکشی، رمزگشایی، انجام محاسبات ریاضی و ...) هماهنگ با یک پالس ساعت یا کلاک انجام می‌شوند. هر چقدر فرکانس این کلاک بیشتر باشد، سرعت اجرا نیز بیشتر خواهد بود. معیار اندازه‌گیری فرکانس کلاک، هرتز (Hertz) می‌باشد که با "Hz" نشان داده می‌شود و نشان‌دهنده «تعداد سیکل‌های ساعت در ثانیه» می‌باشد و اغلب با واحدهای مگاهرتز (MHz) و گیگا هرتز (GHz) بیان می‌شود که به ترتیب یک میلیون پالس در ثانیه و یک میلیارد پالس در ثانیه می‌باشند. فاصله بین دو پالس ساعت زمان سیکل (cycle time) نامیده می‌شود.

در سیستم‌های مختلف دستورات متفاوت به تعداد کلاک‌های متفاوتی برای اجرا نیاز دارند. چنان‌چه از خط لوله استفاده شود، بسته به طراحی مراحل و چگونگی موازی‌سازی دستورات، ممکن است تعداد کلاک‌های لازم برای اجرای یک دستورالعمل مشخص در ساختارهای متفاوت، یکسان نباشد. با توجه به مطالب گفته شده، نرخ کلاک به تنهایی نمی‌تواند کارایی سیستم را نشان دهد. در ادامه معیارهای اصلی که برای اندازه‌گیری کارایی استفاده می‌شوند، بررسی شده است.

### معیار CPI

اگر در یک پردازنده فرکانس ساعت دارای مقدار f باشد آنگاه زمان کلاک عبارت است از: $\text{clock time} = \frac{1}{f}$

زمان کلاک را با T نمایش می‌دهیم.

برای دستیابی به یک معیار مناسب برای کارایی یک سیستم کامپیوتری دو پارامتر زمان پاسخ (response time) و مقدار گذردهی (throughput) به صورت زیر تعریف می‌شوند:

- زمان پاسخ عبارت است از زمان بین شروع و پایان یک کار که زمان اجرا نیز نامیده می‌شود.
- میزان گذردهی به مقدار کار انجام شده در واحد زمان گفته می‌شود.

هرچه پارامتر نخست کوچک‌تر باشد و دومی (یعنی میزان گذردهی) افزایش یابد، کارایی سیستم بهبود می‌یابد. با ذکر یک مثال این دو پارامتر را بیشتر تشریح می‌کنیم.

**مثال۱: فرض کنید دو تغییر زیر بر روی یک سیستم کامپیوتری ایجاد شده باشد:**

**۱) جایگزینی پردازنده با یک پردازنده سریع‌تر**

**۲)اضافه نمودن چند پردازنده (از نوع پردازنده اولیه) به سیستم**

**در حالت اول به دلیل افزایش سرعت پردازنده می‌توان هر کاری را در زمان کمتری نسبت به سیستم اصلی انجام داد و هم‌چنین میزان کار انجام شده در واحد زمان افزایش خواهد یافت، بنابراین زمان پاسخ کاهش و گذردهی افزایش می‌یابد.**

**در حالت دوم زمان انجام یک کار تغییر نمی‌کند بنابراین زمان پاسخ تفاوتی نخواهد داشت اما مقدار کار انجام شده در یک بازه زمانی و در نتیجه گذردهی افزایش می‌یابد.**

فرض کنید در یک برنامه $n$ نوع دستور متفاوت وجود دارد که هر کدام به تعداد کلاک‌های متفاوتی برای اجرا نیاز دارند. اگر تعداد دستورهای نوع $i$ام را با $I_i$، تعداد کلاک‌های لازم به ازای هر دستورالعمل (Clock per Instruction)از نوع $i$ام را با $CPI_i$ و تعداد کل دستورات را با $I_C$ نمایش‌دهیم، میانگین تعداد کلاک به ازای هر دستور که با CPI نشان داده می‌شود، به صورت زیر محاسبه می‌شود:

$$CPI = \frac{\sum_{i=1}^{n}(CPI_i \times I_i)}{I_C}$$

میزان زمان لازم برای اجرای برنامه موردنظر به صورت مقابل محاسبه می‌شود: $T_{total} = I_C \times CPI \times T$

که در آن:

$T_{total}$ : زمان مورد نیاز برای اجرای یک برنامه = تعداد دستورات ($I_C$) × تعداد کلاک به ازای هر دستورالعمل (CPI) × زمان هر کلاک (T)

**مثال۲: اگر در یک برنامه 60 درصد دستورات از کلاس A، 18 درصد از کلاس B، 12 درصد از کلاس C و 10 درصد از کلاس D باشند و مقدار CPI برای این چهار کلاس به ترتیب 1، 2، 4 و 8 باشد، میزان CPI کلی کدام است؟**

۱) 2 ۲) 2.24 ۳) 1.78 ۴) 3.4

☑ **پاسخ:** گزینه «۲» با توجه به اطلاعات داده شده مقدار کلی CPI به صورت زیر قابل محاسبه است:

$$CPI = \underbrace{0.6\times1}_{\text{کلاس A}} + \underbrace{0.18\times2}_{\text{کلاس B}} + \underbrace{0.12\times4}_{\text{کلاس C}} + \underbrace{0.1\times8}_{\text{کلاس D}} = 2.24$$

---

**مثال۳: فرض کنید دو پیاده‌سازی از یک مجموعه دستورالعمل وجود دارد که می‌توان آن را بر روی دو کامپیوتر A و B اجرا کرد. کامپیوتر A و B دارای ویژگی‌های زیر می‌باشند (برای یک برنامه یکسان):**

**کامپیوتر A:** $\textbf{clock cycle time} = 250ps\ ,\ \textbf{CPI} = 2$

**کامپیوتر B:** $\textbf{clock cycle time} = 500ps\ ,\ \textbf{CPI} = 1.2$

**در این صورت اگر برنامه موردنظر دارای 20 دستور باشد آنگاه زمان اجرای برنامه بر روی کامپیوترهای A و B به ترتیب کدام است (از راست به چپ)؟**

۱) 1ns و 2ns ۲) 10ns و 12ns ۳) 1.2ns و 2ns ۴) 10ns و 20ns

☑ **پاسخ:** گزینه «۲» تعداد کلاک‌های لازم برای اجرای برنامه موردنظر بر روی هر یک از کامپیوترها به صورت زیر است:

$CPI_A = 20\times2 = 40\times$ تعداد دستورات = تعداد کلاک‌ها در کامپیوتر A

$CPI_B = 20\times1.2 = 24\times$ تعداد دستورات = تعداد کلاک‌ها در کامپیوتر B

بنابراین زمان اجرای برنامه بر روی این دو کامپیوتر عبارت است از:

$$A:\ \text{Total cpu time} = 250ps\times40 = 10000ps = 10ns$$

$$B:\ \text{Total cpu time} = 500ps\times24 = 12000ps = 12ns$$

---

**مثال۴: در یک برنامه کامپیوتری 25 درصد از دستورات اعمال جمع و تفریق ممیز شناور می‌باشند (FP) و 2 درصد از آن‌ها محاسبه ریشه اعداد ممیز شناور می‌باشند (FPSQR). در این صورت اگر هر دستور جمع و تفریق ممیز شناور نیاز به 4 کلاک و سایر دستورات به صورت میانگین نیاز به 1.33 کلاک به ازای هر دستور داشته باشند آنگاه CPI کل چقدر می‌باشد؟**

۱) 2.2 ۲) 1.2 ۳) 3 ۴) 2

☑ **پاسخ:** گزینه «۴» با توجه به روابط مطرح شده مقدار CPI به ازای دو دسته دستور شرح داده شده که دستورات FPSQR نیز در دسته سایر دستورات قرار دارند، به صورت زیر قابل محاسبه است:

$$CPI = \frac{25\times4 + 75\times1.33}{100} = 25\%\times4 + 75\%\times1.33 = 2$$

چنان‌چه اطلاعات دقیق‌تری از سیستم مورد بررسی در دست باشد، می‌توان آنالیز زمانی دقیق‌تری را انجام داد. فرض کنید تعداد سیکل‌های لازم برای اجـرا و رمزگشایی دستورات برابر $p$ و تعداد دستورات مراجعه به حافظه برابر $m$ باشد. چنان‌چه نسبت زمان سیکل حافظه به زمان سیکل پردازنده را با $k$ نشـان دهیم، زمان مورد نیاز برای اجرای تمام دستورات یک برنامه که دارای $I_C$ دستور می‌باشد، به صورت زیر قابل محاسبه است:

$$T_{total} = I_C \times [p + (m \times k)] \times T$$

مطابق با رابطه فوق پنج فاکتور اصلی را می‌توان در تعیین کارایی در نظر گرفت که به صورت زیر توسط چهار ویژگی سیستم تحت تأثیر قرار می‌گیرند:

۱) **نحوه طراحی مجموعه دستورالعمل‌ها (instruction set architecture)**: که بر روی پارامترهای $I_C$ و $p$ تأثیر گذار است. به بیان دیگر تعداد کل دستورالعمل‌ها و تعداد سیکل‌های لازم برای دستورات رمزگشایی و اجرا را تحت تأثیر قرار می‌دهد.

۲) **فن‌آوری کامپایلر (compiler technology) استفاده شده**: با توجه به نحوه ایجاد دستورات زبان ماشین از دستورات یک زبان سـطح بـالا بـر روی پارامترهای $p$ ،$I_C$ و $m$ تأثیرگذار است.

۳) **نحوه پیاده‌سازی پردازنده (processor implimentation)**: نحوه طراحی پردازنده بر روی پارامترهای $p$ و $T$ تأثیرگذار می‌باشد.

۴) **نحوه پیاده‌سازی سلسله مراتب حافظه و حافظه نهان (cache and memory hierarchy)**: که بر روی پارامترهای $k$ و $T$ تأثیرگذار خواهد بود.

جدول زیر چگونگی ارتباط میان پارامترهای تأثیرگذار در کارایی و ویژگی‌های مرتبط در سیستم کامپیوتری را به خوبی نشان می‌دهد.

| | $I_c$ | p | m | k | T |
|---|---|---|---|---|---|
| **Instruction set architecture** | **X** | **X** | | | |
| **Compiler technology** | **X** | **X** | **X** | | |
| **Processor implementation** | | **X** | | | **X** |
| **Cache and memory hierarchy** | | | | **X** | **X** |

بنابراین در کل می‌توان گفت:

- زمان کلاک به طراحی و پیاده‌سازی سخت‌افزار بستگی خواهد داشت.
- مقدار CPI به طراحی مجموعه دستورات بستگی دارد.
- تعداد دستورات به طراحی مجموعه دستورات و فن‌آوری کامپایلر بستگی خواهد داشت.

## کارایی نسبی و تسریع

در حالت کلی زمانی که هدف مقایسه کارایی دو سیستم $A$ و $B$ برای اجرای یک برنامه خاص می‌باشد، می‌توان کارایی هر یک را به صورت زیر در نظر گرفت:

$$\text{کارایی سیستم A} = \frac{۱}{\text{زمان اجرای برنامه در سیستم A}}$$

$$\text{کارایی سیستم B} = \frac{۱}{\text{زمان اجرای برنامه در سیستم B}}$$

و بنابراین در صورتی که کارایی سیستم $A$ برای اجرای برنامه موردنظر بیشتر از کارایی سیستم $B$ باشد باید داشته باشیم:

$$\text{کارایی سیستم A} > \text{کارایی سیستم B} \Rightarrow \frac{۱}{\text{زمان اجرا در سیستم A}} > \frac{۱}{\text{زمان اجرا در سیستم B}}$$

$$\Rightarrow \text{زمان اجرا در A} < \text{زمان اجرا در B}$$

و در حالت کلی اگر کارایی سیستم $A$ به اندازه $n$ برابر بهتر از کارایی سیستم $B$ باشد باید داشته باشیم:

$$\frac{\text{کارایی A}}{\text{کارایی B}} = \frac{\text{زمان اجرای مورد نیاز در سیستم B}}{\text{زمان اجرای مورد نیاز در سیستم A}} = n$$

حال در صورتی که در یک کاربرد خاص از سیستم $A$ به جای سیستم $B$ استفاده شود، میزان تسریع برابر $n$ خواهد بود.

**مثال۵: فرض کنید یک سیستم کامپیوتری برای اجرای یک برنامه با شرح زیر موجود است:**

**- 25 درصد از دستورات از نوع ممیز شناوری (FP) می‌باشند که میانگین CPI برای آنها 4 است.**

**- میانگین CPI برای سایر دستورات غیر ممیز شناور برابر 1.33 می‌باشد.**

**- از بین دستورات ممیز شناور 8 درصد دستورات محاسبه جذر می‌باشد (FPSQR) که CPI برای آنها برابر 20 می‌باشد.**

**حال اگر دو سیستم جایگزین برای سیستم اولیه با ویژگی‌هایی به صورت زیر وجود داشته باشد:**

**A) کاهش CPI به 2 برای دستورات FPSQR B) کاهش CPI به 2.5 برای تمام دستورات FP**

**در این صورت اگر سیستم سریع‌تر را استفاده کنیم مقدار تسریع کدام است؟**

۱) 1.5 ۲) 1.7 ۳) 1.23 ۴) 1.62

☑ **پاسخ:** گزینه «۳» میزان CPI کل در حالت اولیه برابر است با:

$$CPI = 4 \times 25\% + 1.33 \times 75\% = 2$$

حال اگر از حالت A استفاده شود میزان CPI عبارت است از:

$$CPI_A = CPI - 2\% \times (CPI_{old\ FPSQR} - CPI_{new\ FPSQR}) = 2 - 2\% \times (20 - 2) = 1.64$$

(دقت کنید که 8 درصد از دستورات FP به معنی 2 درصد از کل دستورات است. بنابراین دستورات FPSQR به اندازه 2 درصد از کل دستورات می‌باشند.)

و در حالت دوم میزان CPI جدید عبارت است از:

$$CPI_B = 75\% \times 1.33 + 25\% \times 2.5 = 1.625$$

همان‌گونه که ملاحظه می‌شود سیستم B دارای CPI کلی کمتری است بنابراین سریع‌تر است. در نتیجه میزان تسریع عبارت است از:

$$Speedup = \frac{\text{زمان در حالت اولیه}}{\text{زمان در صورت استفاده از سیستم B}}$$

با توجه به این که تعداد دستورات و زمان کلاک‌ها ثابت است، تسریع به صورت زیر خواهد بود:

$$Speedup = \frac{\text{CPI}\times\text{زمان هر کلاک} \times \text{تعداد دستورات در حالت اولیه}}{\text{CPI}\times\text{زمان هر کلاک} \times \text{تعداد دستورات در حالت اولیه}} = \frac{2}{1.625} = 1.23$$

---

**مثال۶: فرض کنید دو کامپایلر برای ترجمه یک برنامه سطح بالا وجود دارد که هر یک از آنها دستورات زبان سطح بالا را به یکی از سه نوع دستور A، B و C ترجمه می‌کنند که تعداد دستورات ایجاد شده در هر کامپایلر به صورت زیر می‌باشد:**

| نوع دستور / کامپایلر | A | B | C |
|---|---|---|---|
| کامپایلر 1 | 2 | 1 | 2 |
| کامپایلر 2 | 4 | 1 | 1 |

**اگر میزان CPI لازم برای اجرای سه دستور کلاس A، B و C به صورت زیر باشد:**

| | A | B | C |
|---|---|---|---|
| CPI | 1 | 2 | 3 |

**در این صورت میزان CPI به ازای دستورات ایجاد شده توسط کامپایلرهای مطرح شده به ترتیب از راست به چپ کدام است؟**

۱) 1.5، 2 ۲) 2 ،1.5 ۳) 2.5، 1.5 ۴) 1.5، 2.5

☑ **پاسخ:** گزینه «۱» تعداد دستورات موردنیاز در صورت استفاده از هر یک از کامپایلرها به صورت زیر خواهد بود:

کامپایلر ۱ : $2+1+2=5$ کامپایلر ۲ : $4+1+1=6$

بنابراین تعداد کلاک‌های لازم در هر دو حالت عبارت است از:

تعداد کلاک‌ها در صورت استفاده از کامپایلر ۱ $= (2\times1)+(1\times2)+(2\times3)=10$

تعداد کلاک‌ها در صورت استفاده از کامپایلر ۲ $= (4\times1)+(1\times2)+(1\times3)=9$

همان‌گونه که مشاهده می‌شود علیرغم تعداد بیشتر دستورات در کامپایلر دوم، این کامپایلر سریع‌تر نیز می‌باشد بنابراین مقدار CPI در دو حالت به صورت زیر است:

$$CPI_1 = \frac{\text{تعداد کلاک‌ها در حالت اول}}{\text{تعداد دستورالعمل‌ها در حالت اول}} = \frac{10}{5} = 2$$

$$CPI_2 = \frac{\text{تعداد کلاک‌ها در حالت دوم}}{\text{تعداد دستورالعمل‌ها در حالت دوم}} = \frac{9}{6} = 1.5$$