زبان ماشین و اسمبلی

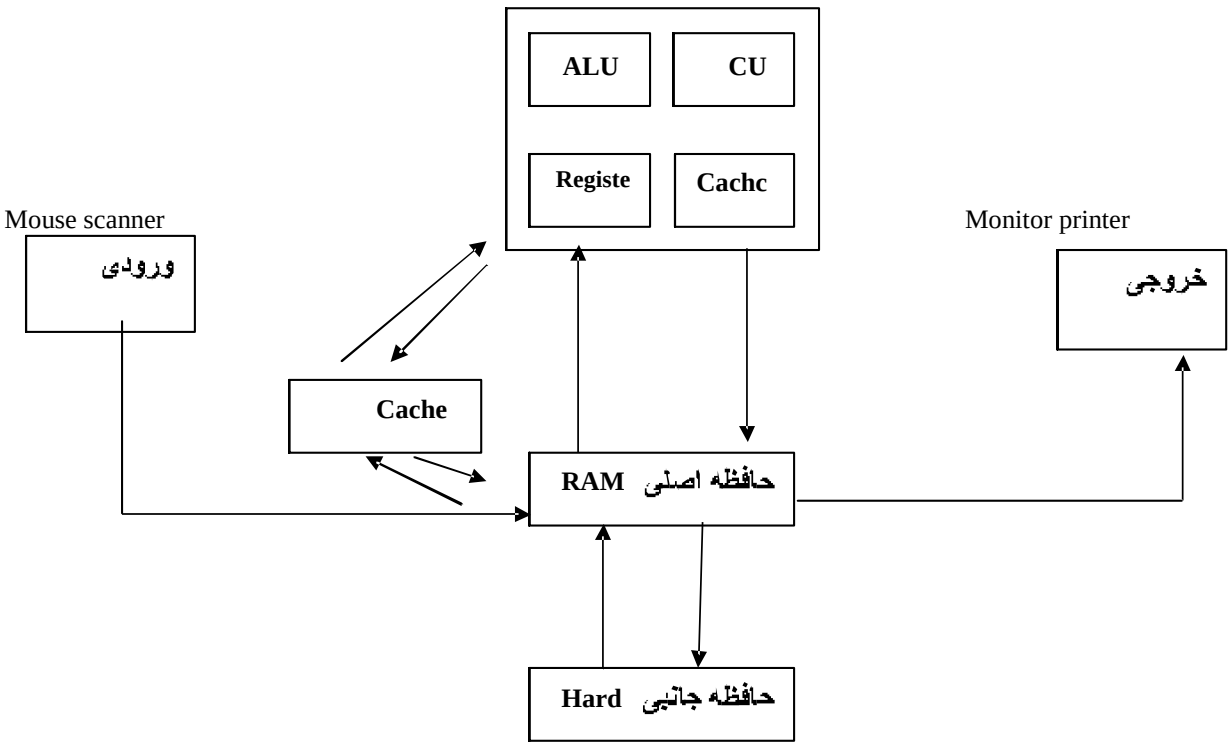
مدرس : امیر کوچکی

ساختمان کامپیوتر

وظیفه Cpu :

CU
- واکشی اطلاعات
- رمز گشایی اطلاعات

ALU : اجرای دستورات



نحوه تبادل اطلاعات بین RAM و CPU :

وقتی داده ها از رم خوانده می شود واحد کنترل آدرس را محاسبه کـرده و آن آدرس را در گذرگاه آدرس قرار می دهد .

واحد حافظه گذرگاه آدرس را می خواند و داده های درخواستی را از حافظه به گذرگاه داده قرار می دهد و سیگنالی به CPU می فرستد و اعلام می کند که داده ها آماده است .

ثبات ها : ثبات ها حافظه های 32,16,8 بیتی در کامپیوتر هست .

انواع مختلف ثبات ها :

ثبات های عمومی : برای انجام کارهای محاسباتی و منطقی مثل AX , BX , CX

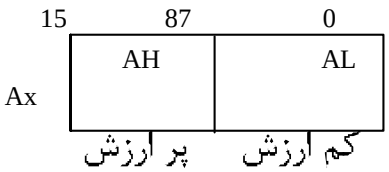
ثبات سگمنت : برای قسمت بندی برنامه مورد استفاده قرار می گیرد .

ثبات فلگ : برای تعیین وضعیت CPU پس از انجام هر عملیات (محاسباتی یا منطقی)

ثبات اندیس : برای آدرس دهی به هر قسمت از سگمنت استفاده می شود .

ثبات های عمومی عبارتند از Ax , Bx که بیت های این ثبات ها از سمت راست به چپ از صفر شماره گذاری می شوند . مثلاً ثبات Ax از 0 تا 15 شماره گذاری می شود .

ثبات Ax : این ثبات در اعمالی که نیاز به ورودی – خروجی و محاسبات زیاد است مورد استفاده قرار می گیرد . این ثبات به دو بخش تقسیم می شود ، بخش بالایی ککم ارزش AL و بخش پر ارزش AH نامیده می شود .



ثبات Bx : این ثبات نیز در محاسبات بکار می رود و به دو بخش BL و BH تقسیم می شود .

ثبات Cx : این ثبات برای کنترل تعاد دفعات حلقه تکرار مورد استفاده قرار می گیرد . (ثبات شمارنده).

ثبات Dx : برای عملیات ضرب و تقسیم از این ثبات استفاده می شود .

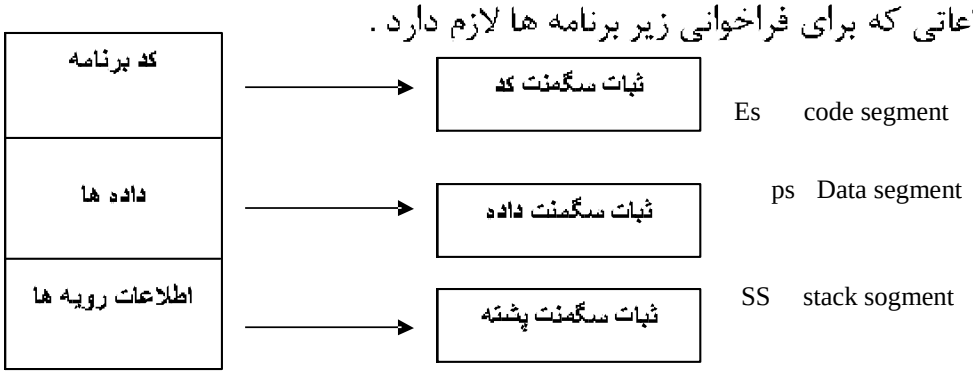
ثبات سگمنت : سگمنت ناحیه ای از حافظه است که آدرس شروع آن بر ۱۶ قابل تقسیم است .

انواع سگمنت : شامل سگمنت که ، داده ، پشته می باشد .

سگمنت کد : دستورالعملهای زبان ماشین که باید اجراء شوند در این ماشین قرار می گیرند .

سگمنت داده : داده های برنامه در این سگمنت قرار می گیرد .

سگمنت پشته : این سگمنت حاوی آدرس های برگشت از رویه است و به طور کلی هر اطلاعاتی که برای فراخوانی زیر برنامه ها لازم دارد .



ثبات دهی سگمنت عبارتند از : Ds , Cs و SS و هر کدام 16 بیتی اند .

هر ثبات سگمنت آدرس شروع یک سگمنت را در خودش ذخیره می کند یعنی ثبات Cs یا ثبات سگمنت که حاوی آدرس شروع سگمنت کد برنامه است که در آدرس دهی برنامه مورد استفاده قرار می گیرد .

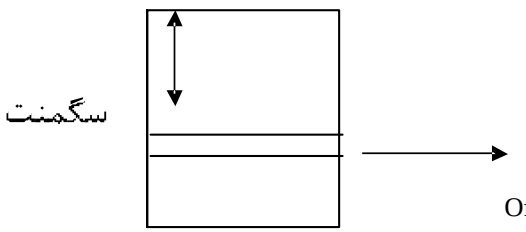
ثبات Ds : آدرس شروع سگمنت داده ها را در خودش ذخیره می کند .

ثبات SS : آدرس شروع سگمنت پشته را نگهداری می کند .

ثبات اندیس :

ثبات های اندیس حاوی Offest داده ها و دستورالعمل ها در داخل سگمنت ها هستند .

منظور از Offest فاصله متغیر یا دستورالعمل از ابتدای سگمنت تا المان است .



ثبات های اندیس شامل DI , SP , BP و ..... هستند .

ثبات وضعیت و کنترل :

ثبات IP : برای کنترل اجراء دستور است .

ثبات Flag (پرچم) : برای تعیین وضعیت CPU بعد از عملیات

نکته : ثبات IP همواره حاوی Offest دستور اجرائی بعدی در سگمنت کد است .

ثبات های IP و CS برای تعیین آدرس بعدی بکار می روند .

شکل ثبات Flag

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
				O	D	I	T	S	Z		A		P		C

ثبات Flag ثبات مخصوصی است که بیت های آن وضعیت پردازنده یا نتیجه عملیات محاسباتی را نشان می دهد . هر بیت دارای نامی است و تعدادی از بیت ها بلا استفاده است و حالت فعلی کامپیوتر و نتایج پردازش را مشخص می کند .

بیت C : مخفف Cary است به معنای رقم نقلی از آخرین بیت در انجام محاسبات است . این بیت را CF می نامند .

بیت P : بیت P مخفف Parity است و برای کنترل صحت اطلاعات بکار می رود . اگر این بیت یک باشد بیانگر این است که تعداد بیت های شیفت داده شده زوج است و اگر صفر باشد تعداد بیت های انتقال داده شده فرد است . این بیت را PF می نامند .

بیت A : مخفف Auxilary Carry است و به معنی رقم نقلی است . چنانچه در محاسبات هشت بیتی رقم سوم ایجاد شود این بیت برابر یک خواهد بود .

مثال :

1 1 1
*Ax*000 01110
*Bx*00001010

00011000

بیت Z : مخفف zero به معنای صفر است و چنانچه نتیجه عملیات محاسباتی برابر با صفر باشد این بیت برابر 1 خواهد شد . این بیت را ZF می نامند .

*Ax*00001110

*Bx*00001010

00011000 Zf=0

بیت S : مخفف Sign و به معنی علامت است و برای بررسی نتیجه عملیات محاسباتی به کار می رود . اگر نتیجه عملیات منفی باشد بیت برابر 1 می شود وگرنه برابر با صفر خواهد شد . این بیت را با SF نشان می دهند .

بیت T : مخفف Trap به معنی قدم به قدم است ، چنانچه این بیت برابر یک باشد اجرای برنامه به صورت دستور به دستور انجام می شود . این بیت را با TF نشان می دهند .

$x = 10$
$y = 20$

if $x = y$ *then*
  $x = x + y$
endif

$z = x * y$

بیت I : مخفف Interrupt به معنی وقفه است . اگر این بیت یک باشد سیستم به وقفه ها پاسخ می دهد وگرنه وقفه نادیده گرفته می شود . نام دیگر آن هم IF است .

بیت D : مخفف Direction به معنی جهت است . اگر شیفت از چپ به راست باش مقدار بین یک در غیر این صورت صفر است . با DF نشان می دهیم .

بیت O : مخفف Orerflov به معنی سرریز است . چنان چه در انجام محاسبات آخرین بیت به دلیل سرریز شدن از بین برود بیت 5 برابر با صفر خواهد شد . نام دیگر آن Of می باشد .

۱-۱- مقادیر دودوئی (Binary)

بشر با توجه به تعداد انگشت‌هایش از ده رقم 0,1,2,3,4,5,6,7,8,9 برای ایجاد مقادیر و اعداد و انجام محاسبات روی آنها استفاده می‌نماید. به بیانی دیگر بشر در یک سیستم دهدهی یا Decimal کار می‌کند. از طرف دیگر کامپیوتر در یک سیستم دودویی یا Binary کار می‌کند و فقط دو رقم 1 و 0 را می‌شناسد. در نتیجه هر مقداری که به کامپیوتر داده شود بایستی تبدیل به یک سری 0 و 1 گردد تا بتواند در کامپیوتر ذخیره و مورد استفاده در محاسبات قرار گیرد. برای تبدیل مقادیر از سیستم دهدهی به سیستم دودویی بایستی آن مقدار بطور متوالی بر 2 تقسیم نمائیم. بعنوان مثال عدد 50 را در نظر بگیرید.

مثال ۱-۱

مقدار	تقسیم بر	نتیجه	باقیمانده
50	2	25	0
25	2	12	1
12	2	6	0
6	2	3	0
3	2	1	1
1	2	0	1

عدد 50 معادل 110010 در سیستم دودویی می‌باشد.

به منظور تبدیل مقداری از سیستم باینری به سیستم دهـدهی، ارقـام عـدد را می‌بایستی بترتیب از راست به چپ در 1، 2، 8، 16 ... ضرب نموده بـا هـم جمـع نمائیم. به عنوان مثال عدد 11010 در سیستم دودویی را در نظر بگیرید.

مثال ۲-۱

1	1	0	1	0
16	8	4	2	1
16*1+			16+	
8*1			8	
4*0			0	
2*1			2	
1*0			0	
			26	

بعبارت دیگر ارقام را بایستی بترتیب در 2^0 ، 2^1، 2^2، 2^3، 2^4 ، ... ضرب نمود.

1	1	0	1	0
2^4	2^3	2^2	2^1	2^0

مثال ۳-۱

عدد 37 را به سیستم دودویی تبدیل نمائید.

مقدار	تقسیم بر	نتیجه	باقیمانده
37	2	18	1
18	2	9	0
9	2	4	1
4	2	2	0
2	2	1	0
1	2	0	1

بنابراین مقدار 37 برابر با 100101 در سیستم دودویی می‌باشد.

مثال ٤-١

عدد 1101101 را به سیستم دهدهی تبدیل نمائید.

1	1	0	1	1	0	1
64	32	16	8	4	2	1

64+
32
8
4
1

109

نتیجه میشود که عدد 1101101 در سیستم دودویی معادل 109 در سیستم دهدهی می‌باشد.

۲-۱- جمع و تفریق در سیستم دودویی

جمع و تفریق در سیستم دودویی شبیه جمع و تفریق در سیستم دهدهی می‌باشد با این تفاوت که به جای ده بر یک، دو بر یک (Carry) ایجاد می‌شود. فرض کنید دو مقدار 3 و 10 در سیستم دودویی با هم جمع نمائیم. ابتدا بایستی هر کدام از این مقادیر را به سیستم دودویی تبدیل نموده سپس آنها را با هم جمع نمائیم.

10	2	5	0
5	2	2	1
2	2	1	0
1	2	0	1

ملاحظه می‌شود که 10 در سیستم دودویی برابر است با 1010.

از طرف دیگر مقدار 3 در سیستم دودویی را بدست می‌آوریم.

3	2	1	1
1	2	0	1

حال دو مقدار 11 و 1010 با هم جمع می‌نمائیم.

1 Carry

1010+

11

1101

در مورد 1+1 بایستی در نظر داشت که نتیجه میشود 10. که یک carry یک به ستون بعدی منتقل می‌گردد.

مثال ۵-۱

مجموع دو مقدار 20 و 17 را بدست آورید.

ابتدا مقادیر 17 و 20 را به سیستم دودویی تبدیل می‌نمائیم.

20	2	10	0
10	2	5	0
5	2	2	1
2	2	1	0
1	2	0	1

مقدار 20 میشود 10100 در سیستم دودویی.

17	2	8	1
8	2	4	0
4	2	2	0
2	2	1	0
1	2	0	1

این نشان می‌دهد که 17 معادل 10001 در سیستم دودویی می‌باشد.

حال

1 Carry

10001+

10100

100101

که این مقدار یعنی 100101 اگر به سیستم دهدهی تبدیل شود برابر است با

1	0	0	1	0	1
32	16	8	4	2	1

32+

4

1

37

در مورد تفریق در سیستم دهدهی همانطوریکه ملاحظه می‌گردد در صورت لزوم یک 1 در سیستم دهدهی قرض گرفته می‌شود.

مثال ٦-١

534 -

281

253

ولی در سیستم دودویی در صورت لزوم یک 1 در سیستم دودویی قرض گرفته که borrow نامیده می‌شود. مثال

$$\begin{array}{r} 1011- \\ 0110 \\ \hline 0101 \end{array}$$

۳-۱- بایت (Byte)

در حافظه کامپیوتر فقط مقادیر 0 و 1 ذخیره میشود. به ارقام 0 و 1 بیت گفته میشود. بیت مخفف کلمات binary digit می‌باشد. به هر هشت بیت کنار هم در حافظه کامپیوتر بایت گفته میشود. بیت‌های یک بایت از 0 تا 7 شماره گذاری شده و بیت شماره 0 بیت کم ارزش‌ترین یا LSB و بیت شماره 7 بیت با بیشترین ارزش یا MSB می‌باشد.

7	6	5	4	3	2	1	0
1	0	0	1	1	0	1	1

هر بایت 256 وضعیت مختلفه از 0 و 1 را ایجاد می‌نماید. بنابراین اعداد صحیح بین 0 تا 255 را می‌توان در یک بایت قرار داد. از طرف دیگر در کامپیوتر از 256 کارکتر مختلف می‌توان استفاده نمود. با استفاده از جدول کد ASCII می‌توان به هر کاراکتر یک کد منحصر بفرد بین 0 تا 255 تخصیص داد. بنابراین هر کاراکتر عملاً یک بایت اشغال می نماید.

٤-۱- مقادیر منفی

اعداد و مقادیر منفی در کامپیوتر با استفاده از روش مکمل 2 نمایش داده می‌شوند. برای نمایش یک مقدار منفی در کامپیوتر بایستی مراحل زیر را طی نمود.

۱-ابتدا عدد را بدون علامت تصور نموده آنرا به سیستم دودویی تبدیل نمائید.
۲-سپس آنقدر رقم 0 در سمت چپ نتیجه مرحله 1 قرار می‌دهیم تا تعداد ارقام آن مضربی از هشت گردد. چنانچه نتیجه مرحله 1 از هشت رقم بیشتر باشد بایستی آنقدر 0 در سمت چپ قرار دهیم تا شانزده رقمی گردد.
۳-سپس ارقام نتیجه مرحله 2 را مکمل می‌نمائیم یعنی 0 به 1 و 1 به 0 تبدیل می‌کنیم.
٤-نتیجه بدست آمده را در سیستم دودویی با 1 جمع می‌نمائیم.

مثال ۷-۱

عدد 26- را در نظر بگیرید. ابتدا عدد 26 را به سیستم دودویی تبدیل می‌نمائیم.

26	2	13	0
13	2	6	1
6	2	3	0
3	2	1	1
1	2	0	1

که میشود 11010.

حال نتیجه بدست آمده را هشت رقمی می‌نمائیم.

00011010

سپس 0 ها را به 1 و 1 ها را به 0 تبدیل می‌کنیم.

11100101

حال نتیجه بدست آمده را با 1 جمع می‌نمائیم.

$$\begin{array}{r} 11100101+ \\ \quad\quad\quad 1 \\ \hline 11100110 \end{array}$$

عدد 11100110 در سیستم دودویی نمایش 26– می‌باشد که یک بایت اشغال می‌نماید. نکته مهمی که بایستی در نظر داشت این است که MSB اعداد منفی در روش مکمل 2 همیشه 1 می‌باشد.

مثال ۸–۱

عدد 35– را به سیستم دودویی تبدیل نمائید.

مقدار	تقسیم بر	نتیجه	باقیمانده
35	2	17	1
17	2	8	1
8	2	4	0
4	2	2	0
2	2	1	0
1	2	0	1

که نتیجه می‌شود 35 معادل 100011 در سیستم دودویی می‌باشد. حال نتیجه بدست آمده را هشت رقمی می‌نمائیم.

00100011

سپس 0ها را به ۱ و 1 ها را به 0 تبدیل می‌کنیم.

11011100

حال نتیجه بدست آمده را با 1 جمع می‌کنیم

$$\begin{array}{r} 1101100 + \\ \quad\quad 1 \\ \hline 11011101 \end{array}$$

مقدار 11011101 در سیستم دودویی معادل 35– می‌باشد. که همانطوریکه ملاحظه میگردد بیت MSB آن برابر با یک می‌باشد.

مثال ۹-۱

عمل زیر را با استفاده از روش مکمل 2 انجام دهید.

27-
20

این عمل تفریق در حقیقت بمنزله جمع دو مقدار زیر می‌باشد.

27+(-20)

حال مقادیر 20– و 27 را به سیستم دودویی تبدیل نموده.

27	2	13	1
13	2	6	1
6	2	3	0
3	2	1	1
1	2	0	1

مقدار 27 معادل 11011 در سیستم دودویی می‌باشد. حال ابتدا مقـدار 20 را به سیستم دودویی تبدیل نمود.

20	2	10	0
10	2	5	0
5	2	2	1
2	2	1	0
1	2	0	1

مقدار 20 برابر است با 10100 در سیستم دودویی. حال 20- را در سیـستم دودویی بدست می‌آوریم. برای این کار ابتدا عدد را هشت رقمی نموده

00010100

سپس صفرها را به 1 و یکها را به صفر تبدیل می‌نمائیم.

11101011

آنگاه مقدار 1 به آن اضافه می‌نمائیم.

```
 11101011+
        1
----------
 11101100
```

نتیجه می‌شود که مقدار 20– برابر است با 11101100 در سیستم دودویی.

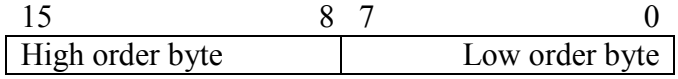
حال دو مقدار 20– و 27 را در سیستم دودویی با هم جمع می‌نمائیم.

```
 11101100 +
    11011
----------
100000111
```

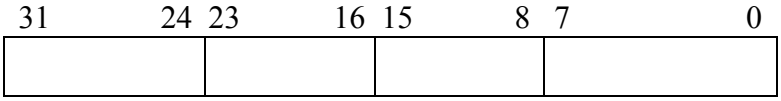
با توجه به آنکه نتیجهٔ جمع دو بایت بصورت یک بایت می‌باشد بیت 1 سمت چپ بایستی حذف گردد، نتیجه می‌شود 111 که برابر با 7 می‌باشد.

۵-۱- گروه‌بندی بیت‌ها

به هر هشت بیت کنار هم بایت گفته می‌شود. دو بایت کنار هم یعنی شانزده بیت متوالی را word می‌نامند(البته در بعضی از کامپیوتر ها هر کلمه می تواند شامل ٤ بایت باشد). بیت‌های یک word از 0 تا 15 شماره‌گذاری می گردد. در یک word بایت سمت راست را بایت مرتبه پائین (Low order byte) و بایت سمت چپ را بایت مرتبه بالا (High order byte)گفته می‌شود.



در یک Word بیت شماره 0 را LSB و بیت شماره 15 را MSB می‌نامند. از طرف دیگر چهار بایت متوالی تشکیل یک Double word میدهند.



هر هشت بایت متوالی تشکیل یک Quadword میدهد و نهایتاً هر هشتاد بیت متوالی یا ده بایت متوالی تشکیل یک Tenbyte می‌دهد. جدول ذیل مقادیری که در یک byte ، word، double word قرار می‌گیرند را نشان می‌دهد.

جدول ۱-۱

نوع	مقادیر بدون علامت	مقادیر علامت دار
Byte	0 تا 255	–128 تا 127
Word	0 تا 65535	–32768 تا 32767
Double word	0 تا 2^{32}-1	-2^{31} تا 2^{31}-1

بایستی توجه داشت که عملیات باینری روی بیت‌ها انجام می‌شود. جدول عملگر جمع بصورت زیر می‌باشد.

جدول ۲-۱

بیت 1	بیت 2	نتیجه	دوبریک (carry)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

جدول عملگر تفریق نیز بصورت زیر می‌باشد.

جدول ۳-۱

بیت 1	بیت 2	نتیجه	یک قرضی (borrow)
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0

٦-۱- عملیات در سیستم مبنای شانزده

ارقام در سیستم مبنای شانزده یا Hexadecimal عبارتند از 0 تا 15. بمنظور جلوگیری از ابهام، ارقام 10 تا 15 را بترتیب با حروف A تا F نشان داده میشوند.

A	10
B	11
C	12
D	13
E	14
F	15

برای تبدیل مقداری از سیستم دهدهی به سیستم مبنای شانزده آن عدد را بطور متوالی بر 16 تقسیم می‌نمائیم. بعنوان مثال عدد 174 را در نظر بگیرید.

مثال ۱۰-۱

مقدار	تقسیم بر	نتیجه	باقیمانده	
174	16	10	14	E
10	16	0	10	A

که نتیجه می‌شود AE.

مثال ۱۱-۱

عدد 3740 را از سیستم دهدهی به سیستم مبنای شانزده تبدیل نمائید.

مقدار	تقسیم بر	نتیجه	باقیمانده	
3740	16	233	12	C
233	16	14	9	
14	16	0	14	E

چون 14 معادل E می‌باشد و C معادل 12 می‌باشد بنابراین جواب مـی‌شـود E9C در سیستم مبنای شانزده.

مثال ۱۲-۱

مقدار 27845 را به سیستم مبنای شانزده تبدیل نمائید.

مقدار	تقسیم بر	نتیجه	باقیمانده	
27845	16	1740	5	
1740	16	108	12	C
108	16	6	11	B
6	16	0	6	

مقدار 27845 برابر با 6CC5 در سیستم شانزدهی می‌باشد.

برای تبدیل مقداری از سیستم شانزدهی به سیستم دهدهی ارقام عدد را از سمت راست بترتیب در 16^0،16^1،16^2،16^3، ... ضرب نموده با هم جمع می‌نمائیم.

مثال ۱۳-۱

عدد 2AF5 را در نظر بگیرید.

2	A	F	5
16^3	16^2	16	1

$5*1+$
$F*16$
$A*16^2$
$2*16^3$

$5 +$
$15*16$
$10*256$
$2*4096$
10997

که نتیجه منجر میشود به 2AF5 که برابر با 10997 می‌باشد.

مثال ١٤-١

مقدار 4F2 در سیستم مبنای شانزده چه مقدار در سیستم دهدهی می‌باشد؟ برای اینکار ابتدا رقم 2 را در 1، رقم F را در 16 و رقم 4 را در 16^2 ضرب می‌نمائیم. سپس مقادیر بدست آمده را با هم جمع می‌کنیم.

4	F	2
16^2	16	1

$$4*16^2+$$
$$F*16$$
$$2*1$$

که منجر میشود به

$$4*256+$$
$$15*16$$
$$2*1$$

که نتیجه می‌شود

$$1024+$$
$$240$$
$$2$$
$$1266$$

مقدار 4F2 در سیستم شانزدهدهی برابر با 1266 در سیستم دهدهی می‌باشد.

از طرف دیگر هر رقم در سیستم مبنای شانزده را می‌توان بوسیله چهار رقم در سیستم باینری نمایش داد.

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

حال برای تبدیل یک مقدار در سیستم مبنای شانزده به سیستم دودویی می‌توان از جدول مذکور استفاده نموده و ارقام را با مقدار معادل آن جایگزین نمود. به عنوان مثال عدد 2FA5B را در نظر بگیرید. با جایگزینی هر رقم با چهار رقم معادل آن در سیستم دودویی نتیجه زیر حاصل می‌گردد.

00101111101001011011

به منظور تبدیل یک مقدار از سیستم دودویی به سیستم مبنای شانزده ابتدا ارقام را از سمت راست چهار تا چهار تا جدا نموده سپس با استفاده از جدول فوق مقادیر معادل را قرار می‌دهیم.

مثال ۱۵-۱

111011001011101

که ابتدا بصورت زیر در می‌آوریم.

0111 0110 0101 1101

که معادل 765D می‌باشد.

۷-۱- عملیات در سیستم مبنای هشت (Octal)

ارقام در سیستم مبنای هشت عبارتند از 0 تا 7 . برای تبدیل مقداری از سیستم دهدهی به سیستم مبنای هشت بایستی آن مقدار را بطور متوالی بر هشت تقسیم نمود. بعنوان مثال عدد 125 را در نظر بگیرید.

مثال ۱۶-۱

مقدار	تقسیم بر	نتیجه	باقیمانده
125	8	15	5
15	8	1	7
1	8	0	1

که نتیجه می‌شود 175 در سیستم مبنای هشت.

مثال ۱۷-۱

بمنظور تبدیل مقداری از سیستم مبنای هشت به سیستم دهدهی، ارقام عدد را از سمت راست بترتیب در 1، 8، 8^2، 8^3، ... ضرب نموده نتایج حاصله را با هم جمع می‌نمائیم. بعنوان مثال عدد 237 در سیستم مبنای هشت را در نظر بگیرید.

2 3 7

8^2 8 1

7*1+	7
3*8	24
$2*8^2$	128
	159

که نتیجه می‌شود 159 در سیستم دهدهی.

مثال ۱۸-۱

عدد 4260 را از سیستم دهدهی به سیستم مبنای هشت تبدیل نمائید.

مقدار	تقسیم بر	نتیجه	باقیمانده
4260	8	532	4
532	8	66	4
66	8	8	2
8	8	1	0
1	8	0	1

نتیجه می‌شود که 4260 در سیستم دهدهی معادل 10244 در سیستم مبنای هشت می‌باشد.

مثال ۱۹-۱

عدد 382 را به سیستم مبنای هشت تبدیل نمائید.

مقدار	تقسیم بر	نتیجه	باقیمانده
382	8	47	6
47	8	5	7
5	8	0	5

که نتیجه می‌شود 576 در سیستم مبنای هشت.

مثال ۲۰-۱

مقدار 4327 را از سیستم مبنای هشت به سیستم دهدهی تبدیل نمائید. برای اینکار ابتدا رقم 7 را در 1، رقم 2 را در 8، رقم 3 را در 8^2 و نهایتاً رقم 4 را در 8^3 ضرب می‌نمائیم سپس مجموع مقادیر بدست آمده را محاسبه می‌نمائیم.

4 3 2 7

8^3 8^2 8 1

که نتیجه می‌شود

$4*8^3$ +

$3*8^2$

2*8

7*1

که معادل است با

4*512+

3*64

2*8

7*1

که نهایتاً برابر است با

2048+

192

16

7

2263

بنابراین مقدار 4327 در سیستم مبنای هشت برابر است با 2263 در سیستم دهدهی.

بایستی توجه نمود که هر رقم در سیستم مبنای هشت را می‌توان بوسیله سه رقم در سیستم دودویی نمایش داد.

0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

برای تبدیل مقداری از سیستم هشت تائی به سیستم دودویی کافی است که به جای هر رقم در سیستم مبنای هشت سه رقم معادل آنرا قرار داد. بعنوان مثال عدد 417 در سیستم مبنای هشت معادل 100001111 در سیستم دودویی می‌باشد.

بمنظور تبدیل مقداری از سیستم دودویی به سیستم مبنای هشت کافی است که ارقام عدد از طرف راست سه تا سه تا جدا نموده و به جای آنها مقدار معادل در سیستم مبنای هشت قرار دهیم.

مثال ۲۱-۱

عدد 10110111010111 در سیستم دودویی در نظر بگیرید که می‌توان بصورت زیر جدا نمود.

010 110 111 010 111

که جواب نهائی میشود 26727 در سیستم مبنای هشت.

از طرف دیگر برای تبدیل مقداری از سیستم مبنای هشت به سیستم مبنای شانزده و برعکس می‌بایستی ابتدا مقدار را به سیستم دودویی تبدیل نموده سپس به سیستم مبنای هشت یا مبنای شانزده تبدیل نمود.

مثال ۲۲-۱

عدد 2AFB5 را در نظر بگیرید.

2AFB5

2	A	F	B	5
0010	1010	1111	1011	0101

حال سه رقم سه رقم از سمت راست جدا نموده.

000 101 010 111 110 110 101

که نهایتاً برابر با 527665 در سیستم مبنای هشت می‌باشد.

۸-۱- مقادیر اعشاری

به منظور تبدیل یک مقدار اعشاری به سیستم دودویی ابتدا قسمت صحیح آنرا به طریق گفته شده به سیستم دودویی تبدیل نموده، سپس قسمت اعشاری آنرا جدا نموده بطور مکرر در 2 ضرب می‌نمائیم. بعنوان مثال عدد 14.725 را در نظر بگیرید. عدد 14 بصورت 1110 در سیستم دودویی می‌باشد. برای تبدیل قسمت اعشاری یعنی 0.725 به سیستم دودویی آنرا در 2 ضرب می‌نمائیم.

مثال ۲۳-۱

$$\begin{array}{r} 0.725* \\ \quad 2 \\ \hline 1.450 \end{array}$$

قسمت صحیح یعنی 1 را جدا نموده، قسمت اعشار را در 2 ضرب می‌نمائیم.

$$\begin{array}{r} 0.45* \\ \quad 2 \\ \hline 0.90 \end{array}$$

قسمت صحیح یعنی 0 را جدا نموده، قسمت اعشار را در 2 ضرب می‌کنیم.

$$\begin{array}{r} 0.8* \\ 2 \\ \hline 1.6 \end{array}$$

و به همین روال کار را ادامه می‌دهیم.

$$\begin{array}{r} 0.6* \\ 2 \\ \hline 1.2 \end{array}$$

جواب می‌شود 1110.10111

برای تبدیل یک مقدار اعشاری از سیستم دودویی به سیستم دهدهی قسمت صحیح آنرا از سمت راست بترتیب در 2^0 ، 2^1 ، 2^2 ، 2^3 ، .... ضرب نموده با هم جمع می‌کنیم سپس قسمت اعشار آنرا بترتیب از سمت چپ در 2^{-1}، 2^{-2} ، 2^{-3} ، 2^{-4}،... ضرب نموده با هم جمع می‌نمائیم. بعنوان مثال عدد 1101.01011 در سیستم دودویی را در نظر بگیرید.

مثال ۲٤-۱

1	1	0	1	.	0	1	0	1	1
8	4	2	1		$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$

$$1*8+4*1+2*0+1*1+0*\frac{1}{2}+1*\frac{1}{4}+0*\frac{1}{8}+1*\frac{1}{16}+1*\frac{1}{32}$$

که خلاصه می‌شود

$$8+4+1+\frac{1}{4}+\frac{1}{16}+\frac{1}{32}=13.34375$$

۱-۳- برنامه و دستورالعملها

در زبان اسمبلی برنامه تشکیل شده است از تعدادی دستورالعملهای اجرائی که بیانگر عملیاتی است که بایستی انجام شود. این سری دستورالعملها همانطوریکه میدانیم SOURCE CODE یا کد منبع نامیده می‌شود. مانند هر زبان برنامه‌نویسی دیگر زبان اسمبلی شکل و قالب از پیش تعریف شده‌ای برای کد منبع دارد. هر دستورالمل اسمبلی شامل چهار فیلد می‌باشد.

فیلد ملاحظات فیلد عملوند فیلدعملیات فیلداسم

البته بایستی توجه داشت که در بعضی از دستورالعملها از تمام فیلدها استفاده نمی‌گردد.

۲-۳- قانون نامگذاری

نام در زبان اسمبلی حداکثر می‌تواند شامل 31 کاراکتر باشد. کاراکترها شامل حروف Z تا A و ارقام 9 تا 0 و سیمبلهای مخصوص @ ؟ . $ ـ می‌باشد. موارد ذیل بایستی در نامگذاری رعایت گردد.

۱-اسم نمی‌تواند با یک رقم شروع گردد.

۲-اسم نبایستی یکی از کلمات ذخیره شده در اسمبلی باشد.

۳-در صورتیکه از ۰ در نام استفاده گردد، بایستی اولین کاراکتر نام باشد. کلمات زیر اسامی مجاز در اسمبلی می‌باشند.

LOOP1	B@A2
X	. XY2
Y2A	SUM2
A_5B	ADDX
COUNT	

کلمات زیر مجاز نمی‌باشند.

LOOP	NEAR
LABEL	ADD
2AB	(5AX
FAR	A2.B

۳-۳- متغیرها (Variables)

نام متغیر مشخص کننده محلی از حافظه می‌باشد که بوسیله برنامه قابل دسترسی می‌باشد و محتوی آنرا در حین اجرای برنامه می‌توان تغییر داد. تعریف متغیر شامل آدرس، نوع داده و اندازه آن می‌باشد. از متغیرها می‌توان بعنوان عملوند در دستورالعملها استفاده نمود. برای متغیرها از نوع بایت از DB، متغیرهای از نوع word از DW و متغیرهای از نوع double word از DD استفاده می‌گردد.

٤-۳- برچسب‌ها (Labels)

از برچسب‌ها بعنوان آدرس دستورالعمل در برنامه‌های کاربردی استفاده می‌شود. از برچسب‌ها به دو صورت استفاده می‌گردد. اگر برچسب در همان سگمنت کد باشد نوع آن NEAR در غیر اینصورت از نوع FAR می‌باشد. در صورتیکه نوع آن NEAR باشد می‌توان بعد از نام برچسب از : استفاده نمود و دیگر نیازی به کلمه NEAR نمی‌باشد.

مثال ۱-۳ LOOP1:

یا

LOOP1 LABEL NEAR

در صورتیکه برچسب از نوع FAR باشد استفاده از کلمه FAR الزامی می‌باشد.

MYCODE LABEL FAR

۵-۳- ثابت‌ها (Constants)

ثابت‌ها مقادیری هستند که در دستورالعملهای برنامه‌ها مورد استفاده قرار می‌گیرند. ثابت‌ها از انواع ذیل می‌باشند.

۱- **Binary** : شامل یک سری 0 و 1 می‌باشد که در انتهای آنها حرف B قرار داده می‌شود.

مثال ۲-۳

110111 B
1000 B

۲- **Decimal** : شامل ارقام 0 تا 9 می‌باشد و بطور اختیاری می‌توان حرف D را به آخر آن اضافه نمود.

40
یا
40D

۳- **Hexadecimal** : شامل ارقام 0 تا 9 و حروف A تا F می‌باشد که در انتهای آنها حرف H اضافه می‌گردد.

مثال ٤-۳

32H
0FFH

اگر مقداری در سیستم مبنای شانزده با یکی از حروف A تا F شروع گردد بایستی 0 به ابتدای آن اضافه گردد. در این صورت کامپیوتر آنرا با نام یک برچسب یا متغیر اشتباه نمی‌گیرد.

٤- Octal: شامل ارقام 0 تا 7 می‌باشد که در انتهای آنها حرف O قرار می‌گیرد. می‌توان به جای O از حرف Q نیز استفاده نمود.

مثال ٥-٣

6O
24O
12Q

٥- Character : ثابت‌های کاراکتری شامل هر کاراکتر از کدهای ASCII می‌باشد که بین علامت نقل قول ′ یا ″ قرار می‌گیرند.

مثال ٥-٣

′B′
″JOHN″
′BOB′

٦- Floating point : این نوع data نمایش مقادیر اعشاری بصورت نمائی می‌باشد.

مثال ٦-٣

SINE DD 0.332E-1

که بوسیله اکثر کامپیوترها حمایت نمی‌گردد.

۳-۶- فیلد عملیات

در فیلد عملیات نام دستورالعمل واقعی ریزپردازنده یا عملی که بایستی انجام شود ذکر می‌گردد. نام دستورالعمل بین 2 تا 6 کاراکتر می‌باشد.

مثال ۳-۸

MOV	REP
CMP	LOOP
REPNE	
LEA	

۳-۷- فیلد عملوند

این فیلد شامل آدرس dataهائی که بایستی بوسیله فیلد عملیات پردازش گردد می‌باشد. فیلد عملوند با حداقل یک فاصله از فیلد عملیات جدا میشود. بعضی از دستورالعملها فاقد عملوند می‌باشند. سایر دستورالعملها یک یا دو عملوند دارند که با کاما از هم جدا می‌شوند. مانند

CBW	عملوند ندارد	
NOP	عملوند ندارد	
CLC	عملوند ندارد	
NOT AL		یک عملوند دارد
MOV AX , Y		دو عملوند دارد

در مواردی که فیلد عملوند دارای دو عملوند می‌باشد عملوند اول را عملوند مقصد و عملوند دوم را عملوند مبداء می‌نامند.

مثال ۱۰-۳

```
AND   AX , X
```

که AX را عملوند مقصد و X را عملوند مبداء می‌نامند.

۸-۳- فیلد ملاحظات (Comment)

این فیلد آخرین فیلد دستورالعمل می‌باشد که شامل توضیحات در مورد دستورالعمل یا برنامه می‌باشد. این فیلد از سایر فیلدها توسط ; جدا می‌گردد.

مثال ۱۱-۳

```
MOV   AH, 45H; Parameter for reading a character
```

دستورالعملها می‌توانند فقط شامل فیلد Comment باشند. در اینصورت دستورالعمل با ; شروع می‌شود.

مثال ۱۲-۳

```
; This is an assembly Program
; For calculating the n factorial.
```

۹-۳- تکنیکهای آدرس‌دهی

ریزپردازنده 80286 از هفت روش آدرس‌دهی استفاده می‌نماید که عبارتند از

۱- آدرس‌دهی بدون واسطه

۲- آدرس‌دهی ثبات

۳- آدرس‌دهی مستقیم

٤- آدرس‌دهی غیرمستقیم ثبات

۵- آدرس‌دهی مبنا

٦- آدرس‌دهی اندیس مستقیم

۷- آدرس‌دهی اندیس مبنا

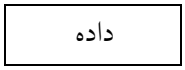
از این امکانات متنوع آدرس‌دهی برای عملوندها استفاده می‌شود.

۱-۹-۳- آدرس دهی بدون واسطه

در این مد آدرس‌دهی داده می‌تواند 8 بیت یا 16 بیت طول داشته باشد و بعنوان عملوند در دستورالعمل استفاده می‌گردد.

مثال ۱۳-۳

MOV BL , 10

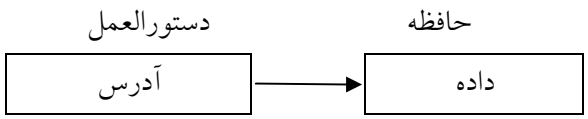


۲-۹-۳- آدرس‌دهی مستقیم

در این روش آدرس داده که شانزده‌بیت می‌باشد جزء دستورالعمل می‌باشد.

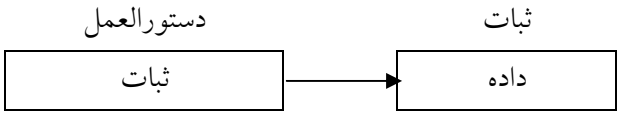
مثال

MOV AX,TABLE



۳-۹-۳- آدرس‌دهی ثبات

در این مد آدرس‌دهی داده در ثباتی قرار دارد که بوسیله دستورالعمل مشخص می‌شود.



برای عملوند شانزده‌بیتی از ثباتهای AX، BX، CX، DX، BP، SI، DI استفاده می‌گردد.

مثال ۱۵-۳

```
MOV   AX , CX
```

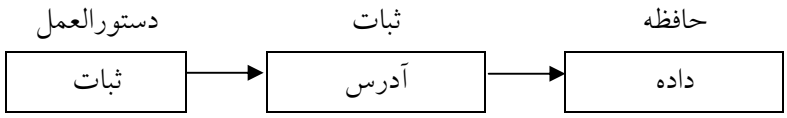
و در مورد عملوند هشت بیتی از ثباتهای AH ، AL، BL، BH، CL، CH، DL ، DH استفاده می‌گردد.

مثال ۱۶-۳

```
MOV   DL , AL
```

٤-۹-۳- آدرس‌دهی غیرمستقیم ثبات

در این روش آدرس داده در یکی از ثباتهایBX ، DI، SI قرار داده می‌شود.




```
MOV   AX , [BX]
```

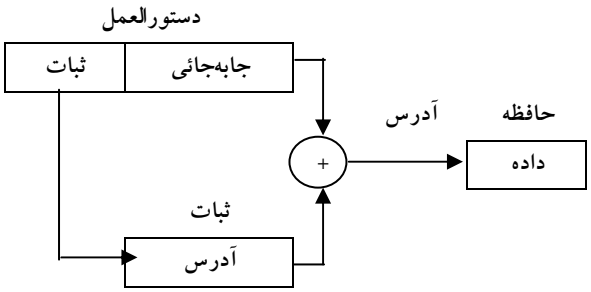
۵-۹-۳- آدرس‌دهی مبنا

در این روش آدرس داده در یکی از ثباتهای BX، BP، SI ، DI قرار داده می‌شود. در این روش یک جابه‌جائی باندازه 8 بیت یا 16 بیت دارد.

MOV AX,[BX]+4

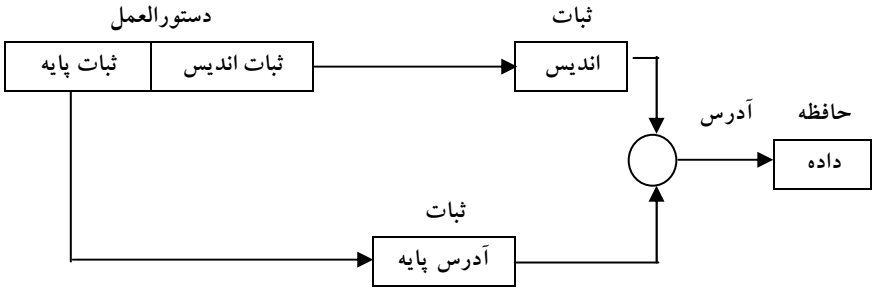
که 4، مقدار جابه جائی و آدرس داده در BX قرار داده شده است. دو دستور ذیل معادل دستور فوق می باشد.

MOV AX, 4[BX]
MOV AX, [BX+4]



٦-٩-٣- آدرس دهی اندیس مستقیم

دراین روش آدرس داده در یکی از ثباتBX قرار داده می‌شود. و از ثباتSI بعنوان اندیس استفاده می گردد. در حقیقت آدرس عبارتست از مجموع BX با SI .



مثال ۱۷-۳

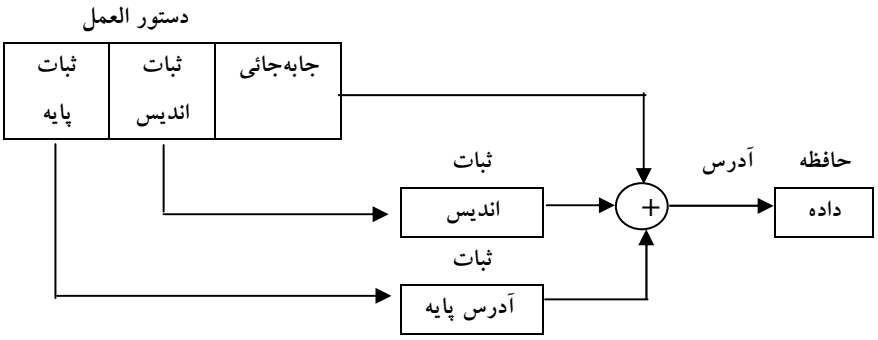
```
MOV   AX,[BX][DI]
```

۷-۹-۳- آدرس‌دهی اندیس مبنا

در این روش آدرس داده شبیه قبل بوده با این تفاوت که یک جا به جائی هشت بیتی یا شانزده‌بیتی نیز وجود دارد.

مثال ۱۸-۳

```
MOV   AX, VALUE [BX][DI]
MOV   AX, [BX+2][DI]
MOV   AX,[BX] [DI+2]
```



۱-٤- انتقال داده‌ها در حافظه

انتقال داده‌ها بین مکانهای مختلف حافظه اصلی و ثباتها بوسیله دستورالعمل MOV انجام می‌شود. شکل کلی این دستورالعمل بصورت زیر می‌باشد.

MOV dst , src

این دستورالعمل محتوی src را در dst قرار داده و محتوی src بدون تغییر باقی می‌ماند.

MOV CL, -30

-30 را در ثبات CL قرار می‌دهد.

MOV X, 25H

مقدار 37 را در مکان X در حافظه قرار می‌دهد.

MOV AX, BX

محتوی ثبات BX را در AX قرار می‌دهد و محتوی ثبات BX بدون تغییر باقی می‌ماند.

MOV DS, AX

محتوی ثبات AX را در DS قرار می‌دهد.

MOV AX, TABLE

محتوی حافظه TABLE را در ثبات AX قرار می‌دهد.

MOV TABLE, AX

محتوی ثبات AX را در مکان TABLE از حافظه اصلی قرار می‌دهد.

درمورد دستورالعمل MOV بایستی در نظر داشت که :

۱-هر دو عملوند یعنی sdt و src بایستی از نوع بایت یا هر دو از نوع word باشند.

۲-هر دو عملوند نمی‌توانند متغیر باشند. یعنی دستورالعمل زیر غلط می‌باشد.

```
MOV     X, Y
```

۳-هیچکدام از عملوندها نمی‌توانند ثبات IP باشند.

٤-هیچکدام از عملوندها نمی‌توانند ثبات فلگ باشند.

۵-محتوی یک ثبات سگمنت را نمی‌توان مستقیماً بیک ثبات سگمنت دیگر منتقل نمود. این کار را بایستی بصورت غیر مستقیم انجام داد.

```
MOV    AX ,  ES
MOV    DS , AX
```

٦-عمولند dst نمی‌تواند ثبات CS باشد.

۷-در دستورالعمل MOV بجزء در مواردیکه src ثابت باشد حتماًیکی از عملوندها بایستی ثبات باشد.

۸-یک ثابت را نمی‌توان مستقیماً به یک ثبات سگمنت منتقل نمود.

این کار را بایستی بصورت زیر انجام داد.

```
MOV    AX, DATA_SEG
MOV    DS , AX
```

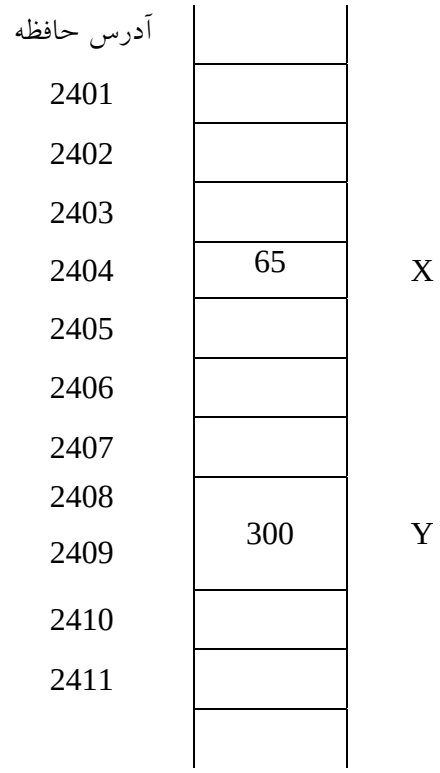
۹-دستورالعمل MOV روی هیچ فلگی اثر ندارد.

همانطور که میدانید هر مکان از حافظه یا متغیر دارای مشخصات زیر می‌باشد.

۱-نام، که از قانون نامگذاری تبعیت می‌نماید.

۲-آدرس، که نشان دهنده مکان متغیر در حافظه می‌باشد.

۳-مقدار، که محتوی آن مکان را نشان می‌دهد.



در بالا حافظه اصلی را نشان می‌دهد که X متغیری از نوع بایت بـا محتـوی 65 و آدرس آن 2404 می‌باشد. همچنین Y متغیری است از نوع word با محتـوی 300 که آدرس آن 2408 می‌باشد. بایستی توجه داشت که آدرس هـر data آدرس بایت شروع آن data می‌باشد. متغیر y چون بایتهـای 2408 و 2409 را در حافظـه اشغال می‌کند آدرس آن 2408 می‌باشد. حال چنانچه بخواهیم آدرس متغیـری را در ثبات BX قرار دهیم از دستورالعمل زیر استفاده می‌گردد.

```
MOV     BX , OFFSET     Y
```

با اجرای این دستورالعمل آدرس 2408 در ثبات BX قرار می‌گیرد.

BX	Y
2408	300

با اجرای دستورالعمل

```
MOV     BX, Y
```

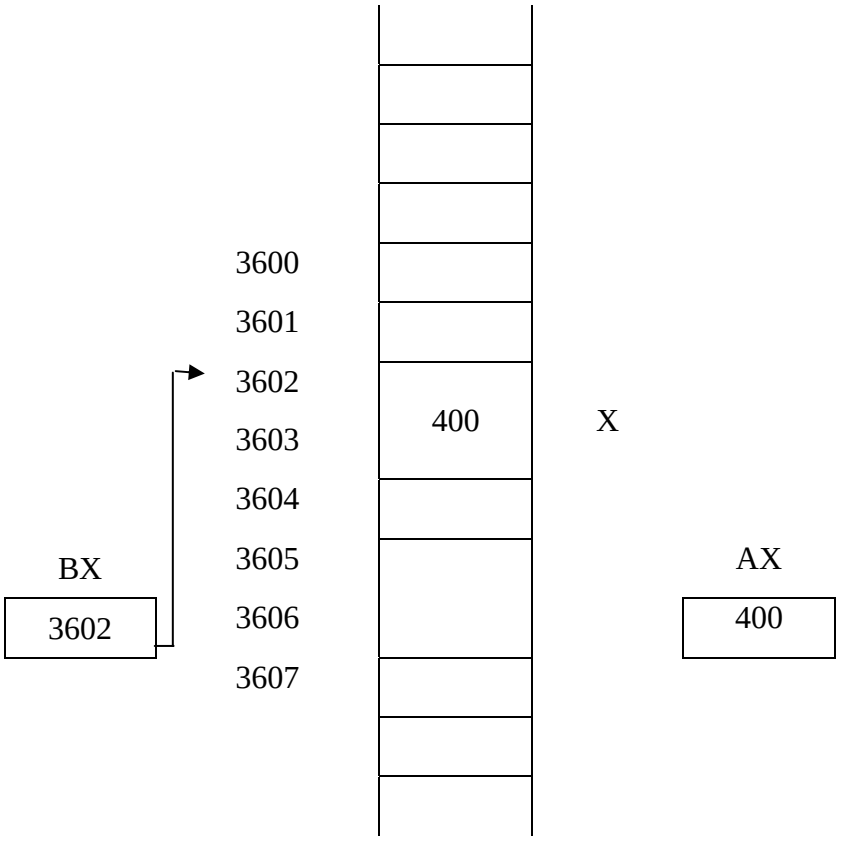
محتوی Y یعنی 300 در ثبات BX قرار داده می‌شود.

BX	Y
300	300

معمولاً آدرس متغیرهـا را در یکـی از ثباتهـای SI, DI, BP, BX قـرار داده می‌شود. حال سه دستورالعمل زیر را در نظر بگیرید.

```
X       DW  400
MOV     BX, OFFSET      X
MOV     AX, [BX]
```

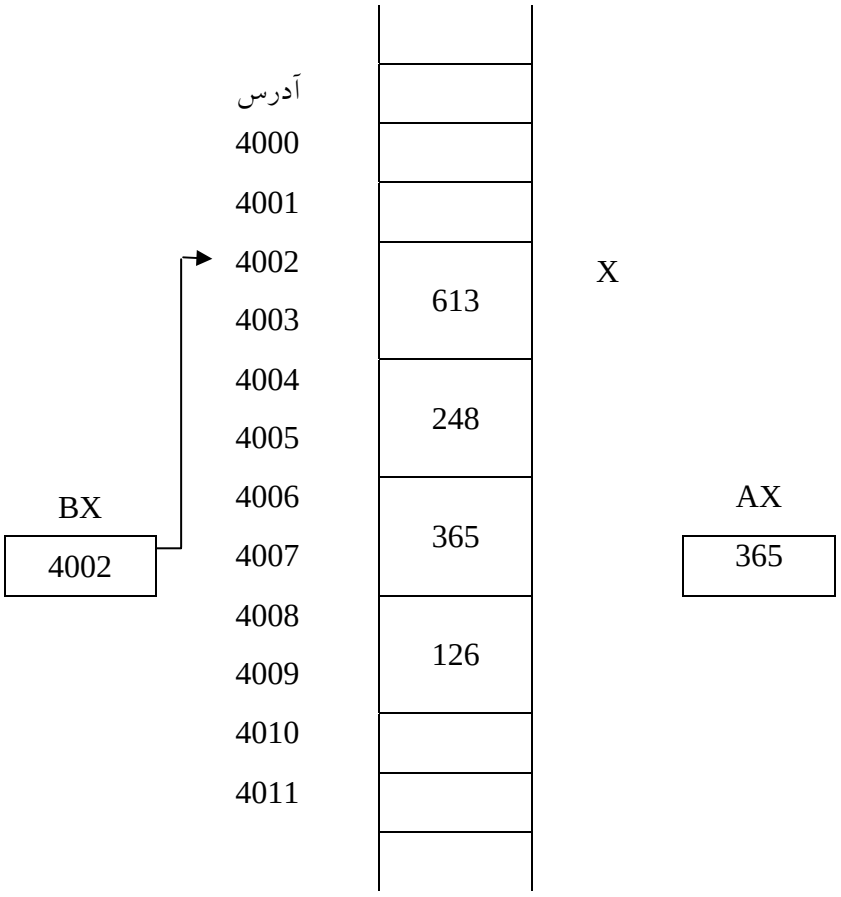
همانطور که در شکل زیر نشان داده شده است. X مکانی از حافظه است که یک word را اشغال نموده است. دستورالعمل دوم آدرس متغیر X را در ثبات BX قرار می‌دهد. دستورالعمل آخر محتـوی مکـانی از حافظـه کـه بوسـیله BX اشـاره می‌شود را به ثبات AX منتقل می‌نماید.



مثال ۱-٤

دستورالعملهای ذیل را در نظر بگیرید.

```
X   DW 613,248,365,126
MOV  BX, OFFSET  X
MOV  AX, [BX] +4
```



اولین دستورالعمل ایجاد یک آرایه چهار عنصری از نوع word می‌نماید با مقادیر 613 ، 248 ، 365 ، 126 بترتیب در آدرسهای4002، 4004، 4006، 4008. دستورالعمل دوم آدرس متغیر X را در ثبات BX قرار می‌دهد. آخرین دستورالعمل 4 واحد به آدرس درون BX اضافه نموده به آدرس 4006 می‌رسد، سپس مقداری که در آدرس 406 حافظه قرار دارد یعنی 365 به ثبات AX منتقل می‌گردد. بایستی توجه داشت که محتوی ثبات BX پس از اجرای دستورالعمل‌های فوق 4002 می‌باشد.

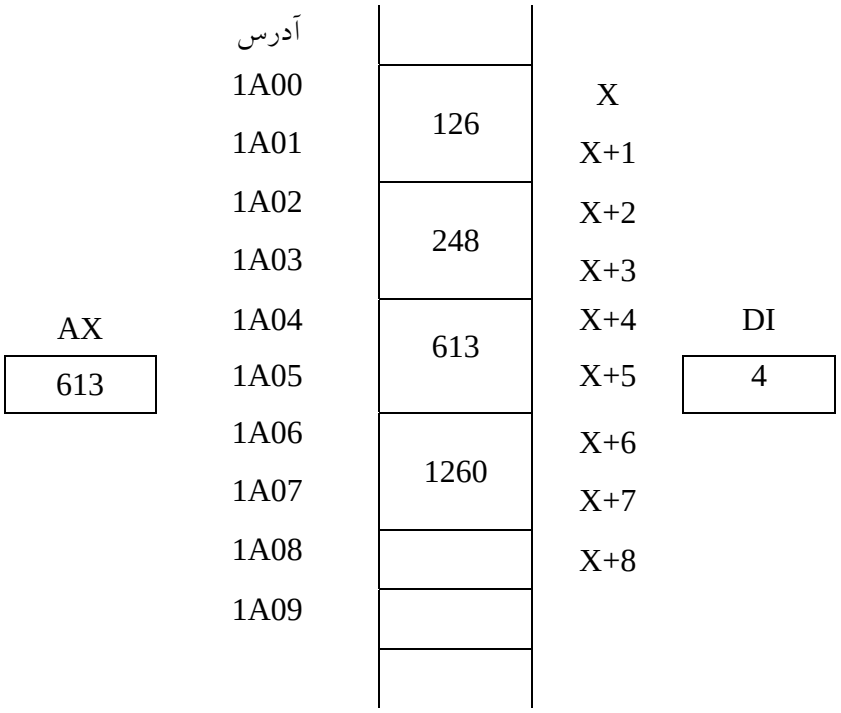
بایستی توجه داشت که سه دستورالعمل ذیل معادلند

```
MOV   AX, [BX]+4
MOV   AX, 4[BX]
MOV   AX, [BX+4]
```

مثال ۲-٤

دستورالعمل‌های ذیل را در نظر بگیرید.

```
 X     DW   126,248,613,1260
MOV    DI, 4
MOV    AX, X[DI]
```



دستورالعمل اول ایجاد یک آرایه می‌نماید بنام X از نوع word با مقادیر 126، 248،613،1260بترتیب در آدرس‌های1A00، 1A02، 1A04، 1A06 از

حافظه. دستورالعمل دوم مقدار 4 را در ثبات DI قرار می‌دهد. آخرین دستورالعمل محتوی X+4 یعنی مقدار 613 را در ثبات AX قرار می‌دهد.

مثال ۳-٤

دستورالعمل ذیل را در نظر بگیرید.

```
MOV    AX, X[BX][DI]
```

این دستورالعمل محتوی آدرسی که از مجموع مقادیرX ، BX ، DI بدست می‌آید را در ثبات AX قرار می‌دهد. در این دستور می‌توان بجای BX از BP و به جای DI از SI استفاده نمود.

مثال ٤-٤

```
MOV    AX, X[BP][DI]
MOV    AX, X[BX][SI]
MOV    AX, X[BP][SI]
```

نهایتاً دستورالعمل MOV نیز می‌تواند به یکی از شکل ذیل باشد.

```
MOV    AX, [BX][DI+2]
MOV    AX, [BX+2][DI]
MOV    AX,[BX][DI+2]
MOV    AX,[BX+DI+2]
```

از این فرم‌های MOV برای استخراج داده‌ها از یک آرایه دو بعدی استفاده می‌گردد.

۲-٤- دستورالعمل LEA

دستورالعمل LEA نیز آدرس متغیر را در یکی از ثباتهای SI، DI، BX، BP می‌دهد. شکل کلی این دستورالعمل بصورت زیر می‌باشد.

LEA dst , src

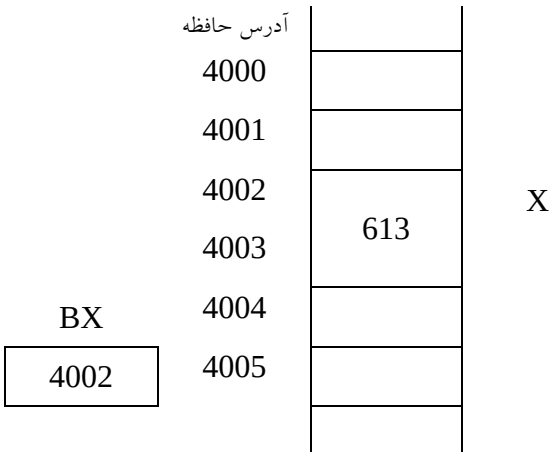
1- src یک متغیر از نوع بایت یا word می‌باشد.

2- dst یکی از چهار ثباتBX ، BP ، SI ، DI می‌باشد.

3- دستورالعمل LEA بر هیچ فلگی اثر ندارد.

مثال ٥-٤

```
X    DW  613
LEA  BX , X
```



در حقیقت عملکرد این دستور معادل دستور ذیل می‌باشد.

MOV BX,OFFSET X

مثال ٦-٤

```
LEA    SI, COL[BX]
```

این دستورالعمل آدرس متغیر COL را با محتـوی BX جمـع نمـوده آدرس بدست آمده را در SI قرار می‌دهد.

مثال ٧-٤

```
X      DB     ?
LEA    BX,    X
```

متغیر X از نوع بایت تعریف گردیـده و آدرس آن در ثبـات BX قـرار داده شده است.

بایستی توجه داشت که دستورالعمل LEA بر هیچ فلگی اثر ندارد.

٣-٤- مبادله داده‌ها

بـرای مبادلـه محتـوی دو آدرس داده از دسـتورالعمل XCHG اسـتفاده می‌گردد. شکل کلی دستورالعمل بصورت زیر می‌باشد.

```
XCHG    dst, src
```

این دستورالعمل محتوی src با dst را مبادله می‌نماید یعنی محتـوی src در dst قرار می‌گیرد و محتوی dstدر src.

در مورد دستورالعمل XCHG بایستی در نظر داشت که:

١- dst , src نمی‌تواند ثابت باشند.

٢- dst , src بایستی هر دو از نوع بایت یا هر دو از نوع word باشند.

٣- dst , src هر دو متغیر نمی‌توانند باشند.

٤- این دستورالعمل بر روی هیچ فلگی اثر ندارد.

مثال ۸-٤

```
MOV    AX, 1000
MOV    X, 3000
XCHG   X, AX
```

قبل از اجرای دستورالعمل XCHG

AX	X
1000	3000

بعد از اجرای دستورالعمل XCHG

AX	X
3000	1000

مثال ۹-٤

```
XCHG    AX , BX
```

محتوی BX را در AX و محتوی AX را در X قرار می‌دهد.

```
X  DB    65
Y  DB    48
MOV      AL , X
XCHG     AL, Y
MOV      X , AL
```

دستورالعملهای فوق باعث مبادله مقـادیر X و Y کـه هـر دو از نـوع بایـت می‌باشند می‌شود. قبل از اجرای دستورالعمل‌های فوق

X	Y
65	48

پس از اجرای دستورالعملهای فوق

AL	X	Y
48	48	65

بایستی توجه داشت که دستورالعمل XCHG روی هیچ فلگی اثر ندارد.

٤-٤- جمع و تفریق

جمع دو مقدار بوسیله دستورالعمل ADD انجام می‌شود. شکل کلی این دستورالعمل بصورت زیر می‌باشد.

```
ADD    dst , src
```

این دستورالعمل محتوی src را با محتوی dst جمع نموده نتیجه را در dst قرار می‌دهد و مقدار src بدون تغییر می‌ماند.

dst ⟵ dst + src

مثال ١٠-٤

```
MOV   AX , 613
MOV   BX, 248
ADD   AX , BX
```

دستورالعمل اول مقدار 613 را در AX قرار می‌دهد. دستورالعمل دوم مقدار 248 را در BX قرار می‌دهد. دستورالعمل سوم مجموع AX و BX را محاسبه و نتیجه را در AX قرار می‌دهد. که همانطور که ملاحظه می‌شود مقدار BX تغییر نمی‌کند.

قبل از اجرای دستورالعمل فوق

AX	BX
613	248

بعد از اجرای دستورالعملهای فوق مقادیر عبارتند از:

AX	BX
861	248

در مورد دستورالعمل ADD بایستی موارد زیر را در نظر داشت.

١- هر دو عملوند بایستی از نوع بایت یا هر دو از نوع word باشند.

٢- هر دو عملوند نمی توانند از نوع متغیر باشند.

۳- به جزء در مواردیکه یکی از عملوندها ثابت باشد حتماً یکی از عملوندها بایستی از نوع ثبات باشد.

٤- دستورالعمل ADD بر روی فلگ‌های AF, CF, OF, PF, SF, ZF اثر دارد.

مثال ۱۱-٤

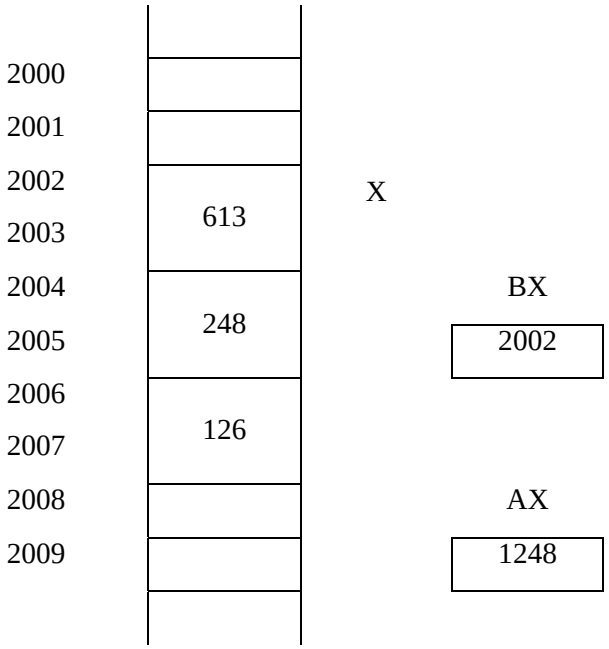
```
X       DB   13
MOV     AL , 10
ADD     X , AL
```

دستورالعمل اول X را از نوع بایت تعریف نموده و مقدار آنرا 13 قرار می‌دهد. دستورالعمل دوم مقدار 10 را در ثبات AL قرار می‌دهد. دستورالعمل سوم مقدار AL را با محتوی X جمع نموده نتیجه که می‌شود 23 را در X قرار می‌دهد. و محتوی AL نیز 10 می‌باشد.

مثال ۱۲-٤

```
X       DW  613,248,126
MOV     BX, OFFSET  X
MOV     AX, 1000
ADD     AX, [BX + 2]
```

اولین دستورالعمل یک آرایه سه عنصری از نوع word ایجاد می‌نماید. مقادیر عناصر آرایه بترتیب عبارتند از613، 248، 128دستورالعمل دوم آدرس X را در BX قرار می‌دهد. دستورالعمل سوم مقدار 1000 را در AX قرار می‌دهد. آخرین دستورالعمل محتوی دومین عنصر آرایه را یعنی 248 با محتوی AX جمع نموده و نتیجه را در AX قرار می‌دهد.



مثال ۱۳-٤

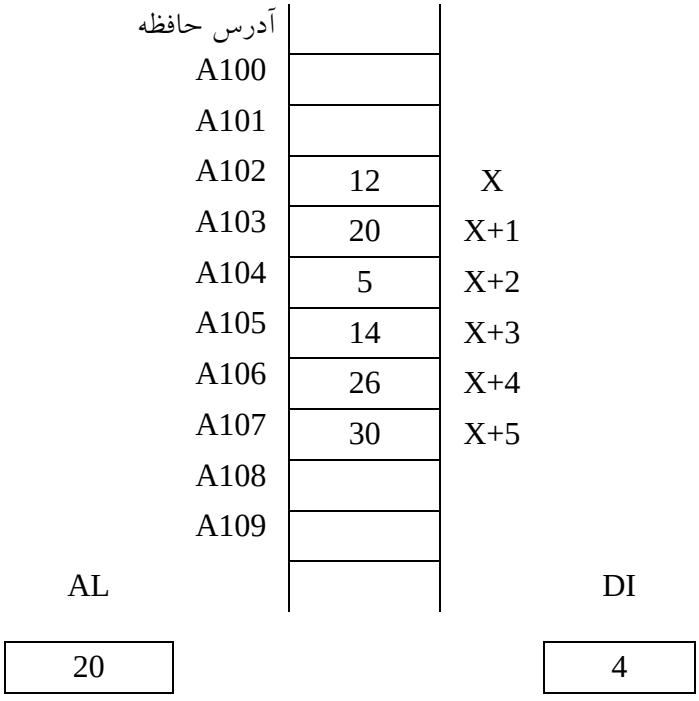
```
X       DB    18
MOV     AL , -18
ADD     AL , X
```

دستورالعمل اول مقدار 18 را در X قرار می‌دهد. دستورالعمل دوم مقدار 18- را در AL قرار می‌دهد. آخرین دستورالعمل محتوی X را با محتوی AL جمع نموده، نتیجه را که 0 می‌شود در AL قرار می‌دهد. بایستی توجه داشت که پس از انجام عمل جمع مقدار فلگ ZF برابر با یک می‌شود.

مثال ١٤-٤

```
X       DB  12,20,5,14,26,30
MOV     DI ,  5
MOV     AL ,  20
ADD     X [DI] , AL
```

دستورالعمل اول یک آرایه 6 عنصری از نوع بایت با مقادیر بترتیب 12، 20، 5، 14، 26، 30 تعریف می‌نماید. دستورالعمل دوم مقدار 5 را در DI قرار می‌دهد. دستورالعمل سوم مقدار 20 را در AL قرار می‌دهد. آخرین دستورالعمل محتوی مکانی از حافظه که بوسیله X+5 مشخص می‌شود را با AL جمع می‌نماید. یعنی مقدار 30 را با 20 جمع نموده و نتیجه را در محل X+5 از حافظه قرار می‌دهد. و مقادیر ثباتهای AL و DI بدون تغییر باقی‌می‌ماند.



پس از اجرای دستورالعملهای فوق شکل حافظه بصورت زیر در می‌آید.

A100		
A101		
A102	12	X
A103	20	X+1
A104	5	X+2
A105	14	X+3
A106	26	X+4
A107	50	X+5
A108		
A109		

از دستورالعمل ADC نیز برای جمع مقادیر می‌توان استفاده نمود. فرم کلی این دستورالعمل بصورت زیر است.

ADC dst , src

که محتوی src را با محتوی dst جمع نموده نتیجه را با مقدار فلگ CF جمع نموده و نهایتاً نتیجه بدست آمده را در dst قرار می‌دهد.

dst ⟵ dst + src + CF

در موقع استفاده از این دستورالعمل بایستی توجه داشت که :

۱- هر دو عملوند src , dst بایستی از نوع بایت یا هر دو از نوع word باشند.

۲- src , dst هر دو متغیر نمی‌توانند باشند.

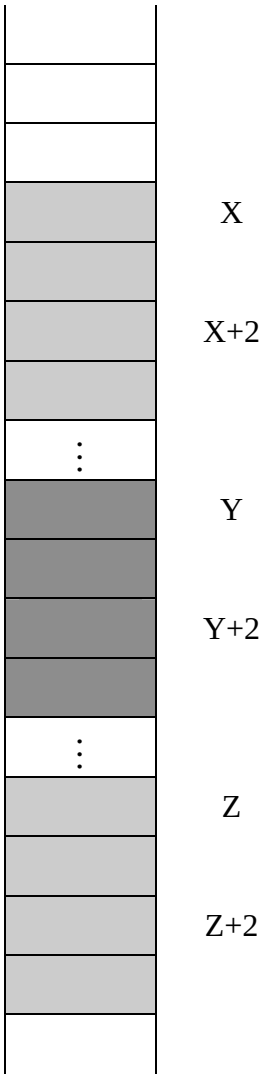
۳- به جزء در مواردیکه یکی از عملوندهای dst , src ثابت باشد یکی از عملوندها بایستی حتماً از نوع ثبات باشد.
٤- دستورالعمل ADC روی فلگ‌هایZF ، SF، AF، PF، OF، CF اثر دارد.

مثال ۱۵-٤

```
X  DW   ?
MOV     AX , 1000
MOV     X, 3000
ADC     AX , X
```

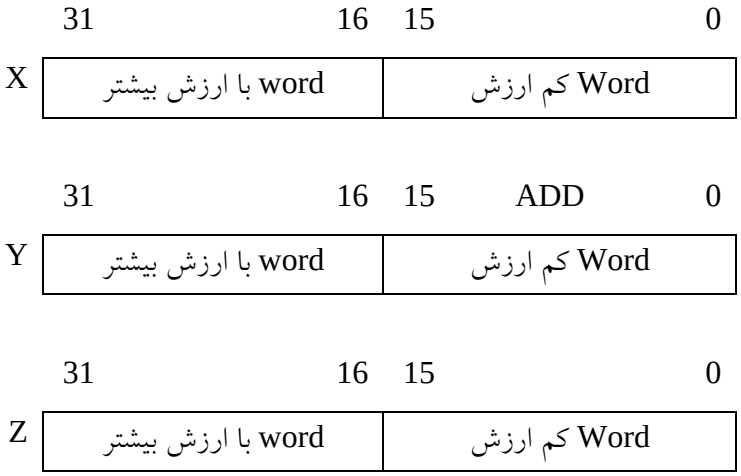
دستورالعمل اول X را از نوع word تعریف می‌نماید. دستورالعمل دوم مقدار 1000 را در ثبات AX قرار می‌دهد. دستورالعمل سوم مقدار 3000 را در متغیر X قرار می‌دهد. آخرین دستورالعمل محتوی X را با محتوی ثبات AX جمع نموده و چنانچه CF=1 باشد یک واحد به جمع بدست آمده اضافه نموده 4001 را در ثبات AX قرار می‌دهد. و چنانچه مقدار CF=0 باشد مقدار 4000 را در ثبات AX قرار می‌دهد.

یکی از کاربردهای مهم دستورالعمل ADC در محاسبه مجموع دو مقدار از نوع double word می‌باشد. فرض کنید می‌خواهیم مجموع دو متغیرY و X از نوع double word را محاسبه نموده و نتیجه را در متغیر Z از نوع double word قرار دهیم.



همانطور که در شکل ملاحظه می‌گردد، متغیر X از نوع double word را می‌توان بصورت دو تا word بنامهای X و X+2 در نظر گرفت. بطور مشابه همین عمل را در مورد Y و Z می‌توان انجام داد. برای جمع دو متغیر از نوع double word کافی است که ابتدا دو تا word با ارزش کم‌تر را جمع نموده با

استفاده از دستورالعمل ADD سپس دو تا word با ارزش بیشتر را با هم جمع نموده با استفاده از دستورالعمل ADC.



قطعه برنامه‌ای که دو متغیرX و Y از نوع double word را با هم جمع نموده و نتیجه را در متغیر Z از نوع double word قرار می‌دهد بصورت ذیل می‌باشد:

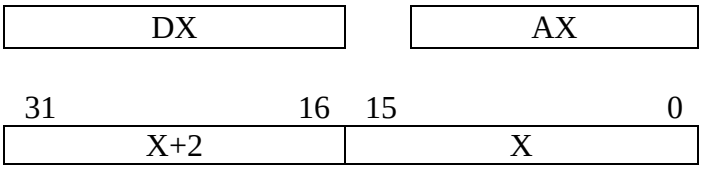
```
X       DD      60000
Y       DD      40000
Z       DD      ?
MOV     AX , X
ADD     AX , Y
MOV     Z , AX
MOV     AX, X+2
ADC     AX , Y+2
MOV     Z +2, AX
```

سه دستورالعمل اول متغیرهای Z , Y, X را از نوع double word تعریف نموده و مقدار X را برابر با 60000 و مقدار Y را برابر با 40000 قرار می‌دهد. سه دستورالعمل بعدی دو تا word با ارزش کم X و Y را با هم جمع نموده نتیجه را

در word با ارزش کم Z قرار می‌دهد. سه دستورالعمل بعدی دو تا word با ارزش بیشترX و Y را با هم جمع نموده با استفاده از ADC و نتیجه را در word با ارزش بیشتر Z قرار می‌دهد. بایستی توجه داشت که در دستورالعملهای MOV ، ADD، ADC هر دو عملوند نمی‌توانند متغیر باشند.

مثال ١٦-٤

قطعه برنامه زیر مقدار X از نوع double word را با مقدار ثباتهای DX و AX جمع نموده نتیجه را در X قرار می‌دهد.



ابتدا محتوی X را با AX جمع نموده با استفاده از دستورالعمل ADD سپس محتوی X+2 را با DX با استفاده از دستورالعمل ADC جمع می‌نمائیم.

```
X       DD   40000
MOV     AX , 300
MOV     DX, 400
ADD     X , AX
ADC     X+2 , DX
```

قطعه برنامه ذیل مجموع متغیرهای X و Y از نوع double word و عدد 24 را محاسبه و نتیجه را در متغیر W قرار می‌دهد. یعنی :

$$W \longleftarrow X + Y + 24$$

برای اینکـار ابتـدا X ، Y ، W را بـصورت double word تعریـف نمـوده سپس مجموع مقادیر X و Y را بدست می‌آوریم. آنگاه عـدد 24 را بـصورت یـک مقدار از نوع double word در نظر گرفته با مجموع قبلی بدست آمده جمع نموده و نهایتاًنتیجه نهائی را در W قرار می‌دهیم.

```
X       DD      ?
Y       DD      ?
Z       DD      ?
MOV     AX , X
MOV     DX,X+2
ADD     AX,Y
ADC     DX,Y+2
ADD     AX,24
ADC     DX ,0
MOV     W, AX
MOV     W+2 , DX
```

برای انجام عمل تفریق از دستورالعمل SUB اسـتفاده مـی‌گـردد. فـرم کلـی دستورالعمل SUB عبارتست از

```
SUB     dst , src
dst  ←  dst – src
```

مقدار src از dst کم می‌شود نتیجه در dst قرار می‌گیرد. در بکـارگیری ایـن دستورالعمل بایستی موارد زیر را در نظر داشت.

۱- هر دو عملوند بایستی از یک نوع باشند هر دو از نوع بایـت یـا هـر دو از نـوع word باشند.

۲- هر دو عملوند نبایستی از نوع متغییر باشند.

۳- به جـزء در مواردیکـه یکـی از عملونـدها ثابـت باشـد حتمـاً بایـستی یکـی از عملوندها از نوع ثبات باشد.

٤- دستورالعمل SUB بر روی فلگ‌هایZF ، CF، OF، PF، AF، SF اثر دارد.
٥- محتوی عملوند dst در عمل SUB تغییر نمی‌کند.

مثال ١٧-٤

```
MOV    AL , 10
MOV    BL , 6
SUB    AL , BL
```

دستورالعمل اول مقدار 10 را در AL قرار می‌دهد. دستورالعمل دوم مقدار 6 را در BL قرار می‌دهد و دستورالعمل سوم محتوی BL را از محتوی AL کم نموده نتیجه را در AL قرار می‌دهد و مقدار BL تغییر نمی‌کند.

قبل از اجرای دستورالعمل SUB

AL	BL
10	6

بعد از اجرای دستورالعمل SUB

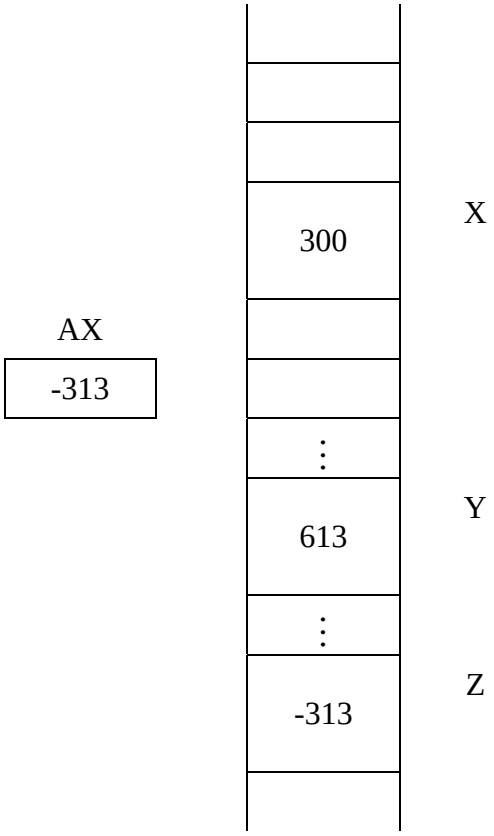
AL	BL
4	6

مثال ١٨-٤

```
X      DW     300
Y      DW     613
Z      DW     ?
MOV    AX, X
SUB    AX , Y
MOV    Z , AX
```

سه دستورالعمل اول سه متغیر X با مقدار 300 و Y با مقدار 613 و Z از نوع word تعریف می‌نماید. دستورالعمل چهارم محتوی متغیر X را به AX منتقل می‌نماید. دستورالعمل بعدی محتوی متغیر Y را از محتوی ثبات AX کم نموده نتیجه را در AX قرار می‌دهد. آخرین دستورالعمل محتوی ثبات AX را به متغیر Z

منتقل می‌نماید. در نتیجه اجرای دستورالعمل SUB مقـدار فلـگ SF=1 مـی‌شـود، زیرا نتیجه تفریق مقداری منفی است.

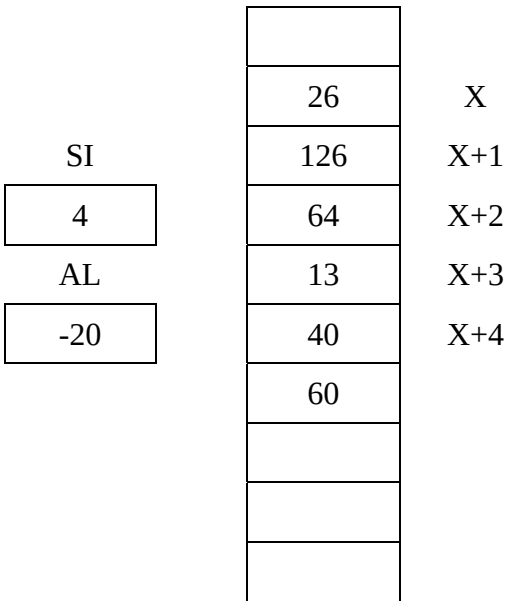


مثال ۱۹-٤

```
ARR     DB  26,126,64,13,40,60
MOV     SI, 4
MOV     AL, 20
SUB     AL, ARR [SI]
```

اولـین دسـتورالعمل یـک آرایـه شـش عنـصری از نـوع بایـت بـا مقـادیر 6,126,64,13,40,60 تعریف می‌نماید. دستورالعمل دوم مقـدار 4 را در ثبـات SI

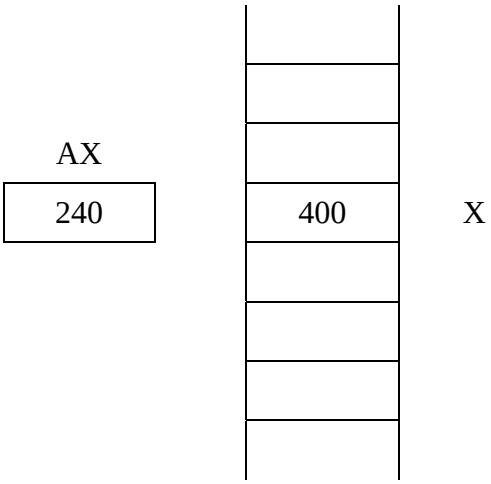
قرار می‌دهد. دستورالعمل سوم مقدار 20 را در ثبات AL قرار می‌دهد. آخرین دستورالعمل، محتوی ARR+4 را از AL کم نموده نتیجه را که می‌شود 20– در AL قرار می‌دهد. و مقدار فلگ SF برابر با یک می‌شود.



مثال ۲۰-٤

```
X       DW      600
MOV     AX , 248
SUB     AX, 48
SUB     X , AX
```

اولین دستورالعمل متغیر X را از نوع word با مقدار 613 تعریف نموده، دستورالعمل دوم مقدار 248 را در AX قرار می‌دهد. دستورالعمل سوم مقدار 48 را از محتوی ثبات AX کم نموده و نتیجه را که می‌شود 200 در AX قرار می‌دهد. آخرین دستورالعمل محتوی AX از محتوی متغیر X کم نموده نتیجه که می‌شود 400 را در متغیر X قرار می‌دهد.



از دستورالعمل SBB نیز برای عمل تفریق استفاده میگردد. شکل کلی این دستورالعمل بصورت زیر می‌باشد.

SBB dst , src

dst ← dst – src – CF

محتوی src از محتوی dst کم می‌شود سپس مقدار CF را از نتیجه کم نموده آنگاه نتیجه را در dst قرار می‌دهد.

مثال ۲۱-٤

```
MOV   AX , 1000
SBB   AX , 800
```

دستورالعمل اول مقدار 1000 را در AX قرار می‌دهد. دستورالعمل دوم 800 را از محتوی AX کم نموده چنانچه مقدار فلگ CF برابر با یک باشد نتیجه می‌شود 199 در غیر اینصورت نتیجه می‌شود 200.

مواردیکه بایستی در استفاده از دستورالعمل SBB رعایت گردند عبارتند از:

۱- هر دو عملوند بایستی از نوع بایت یا هر دو عملوند از نوع word باشند.

۲- هر دو عملوند نمی‌توانند از نوع متغیر باشند.

۳- بجز در مواردیکه یکی از عملوندها ثابت باشد حتماًبایستی یکی از عملوندها از نوع ثبات باشد.
٤- دستورالعمل SBB روی فلگ‌هایAF، PF، ZF، SF، OF، CF اثر دارد.

مثال ۲۲-٤

با در نظر گرفتن اینکه مقدار فلگ CF برابر با صفر می‌باشد قطعه برنامه زیر را اجرا نمائید.

```
MOV    AL , 100
SBB    AL , 60
```

دستورالعمل اول مقدار 100 را در ثبات AL قرار می‌دهد و چون دستورالعمل MOV روی هیچ فلگی اثر ندارد، مقدار فلگ CF بدون تغییر مقدار صفر می‌ماند. دستورالعمل دوم در اینجا چون مقدار CF برابر با صفر است دقیقاً مانند دستورالعمل SUB عمل نموده مقدار 60 را از محتوی AL کم نموده و نتیجه را که برابر با 40 می‌باشد در AL قرار می‌دهد.

قبل از اجرای دستور SBB

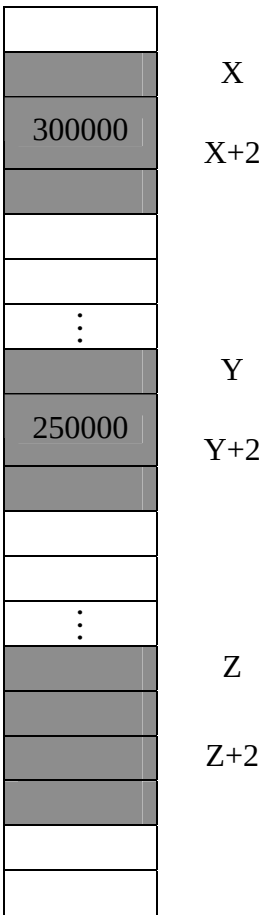
CF	AL
0	100

بعد از اجرای دستور SBB

CF	AL
0	40

از کاربردهای مهم دستورالعمل SBB در تفریق دو مقدار از نوع double word می‌باشد. بعنوان مثال دو متغیرY و X را از نوع double word در نظر بگیرید قطعه برنامه‌زیر تفاضل آنها را محاسبه نموده نتیجه را در متغییر Z که از نوع double word می‌باشد قرار می‌دهد.

```
X       DD      300000
Y       DD      250000
Z       DD      ?
MOV     AX, X
SUB     AX, Y
MOV     Z , AX
MOV     AX , X+2
SBB     AX, Y+2
MOV     Z+2, AX
```

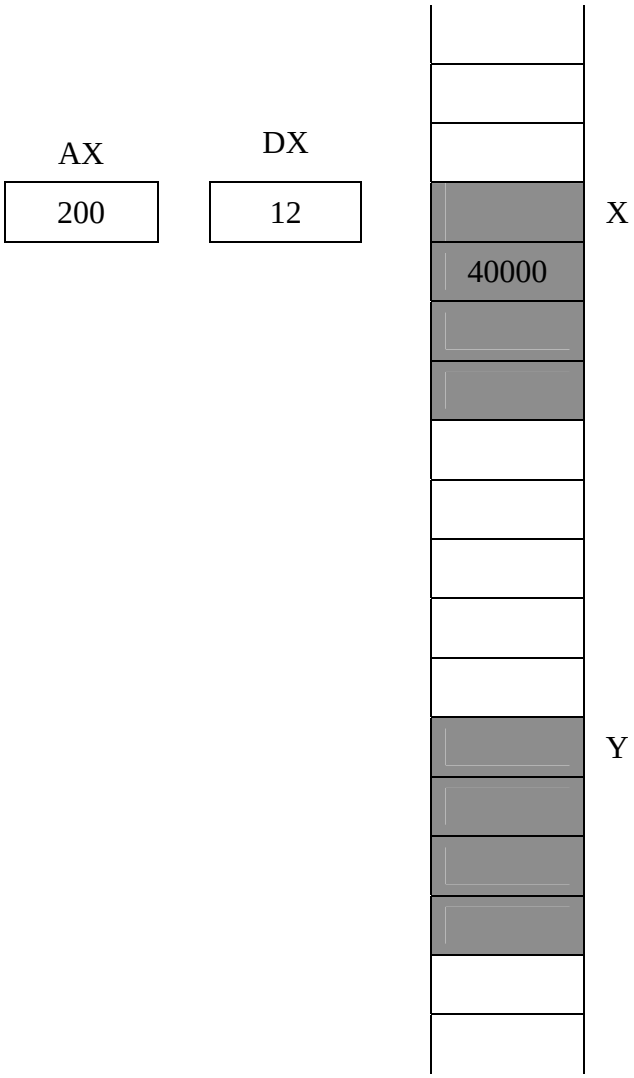


سه دستورالعمل اول متغیرهای X , Y, Z را از نوع double word تعریف می‌نماید سه دستورالعمل بعدی تفاضل دو Word بنامهای Y و X را محاسبه نموده و نتیجه را در دو بایت اول Z قرار می‌دهد. سه دستورالعمل آخر تفاضل دو بایت آخر X و دو بایت آخر Y را با استفاده از دستورالعمل SBB محاسبه نموده و نتیجه را در دو بایت آخر Z قرار می‌دهد.

مثال ۲۳-٤

قطعه برنامه‌ریز تفاضل مقدار متغیر X از نوع double word با محتوی ثباتهای AX:DX محاسبه می‌نماید و نتیجه را در متغیر Y قرار می‌دهد.

```
X       DD      40000
Y       DD      ?
MOV     AX,200
MOV     DX, 12
SUB     DX, X
SBB     AX, X+2
MOV     Y, DX
MOV     Y+2, AX
```



۵-٤- ضرب دو مقدار

دستورالعمل MUL و IMUL برای ضرب دو مقدار استفاده می‌گردد.

دستورالعمل MUL وقتی استفاده می‌گردد که عملوندها بصورت بدون علامت (unsigned) در نظر گرفته شوند. از دستور IMUL وقتی استفاده می‌گردد

که عملوندها بصورت علامت دار(signed) در نظر گرفته شوند. شکل کلی دستور MUL بصورت زیر می‌باشد.

MUL Opr

در مورد عملوند opr نکات زیر را بایستی رعایت نمود.

الف) عملوند opr بایستی از نوع بایت یا word باشد.

ب) عملوند opr می‌تواند متغییر یا ثبات باشد.

ج) عملوند نمی‌تواند ثابت باشد.

د) دستورالعمل MUL روی فلگ‌هایCF و OF اثر دارد.

هـ) در این دستورالعمل مقادیر فلگ‌هایSF، AF، ZF، PF تعریف نشده‌اند.

و) چنانچه عملوند opr از نوع بایت باشد. محتوی ثبات AL در محتوی opr ضرب شده نتیجه در AX قرار می‌گیرد.

ز) چنانچه عملوند opr از نوع word باشد محتوی ثبات AX در محتوی opr ضرب گردیده نتیجه در DX:AX قرار می‌گیرد.

مثال ٢٤-٤

```
MOV   BL, 11H
MOV   AL, 0B4H
MUL   BL
```

دستور العمل اول مقدار 11H یعنی 17 را در ثبات BL قرار می‌دهد.

BL

00010001

دستورالعمل دوم مقدار 0B4H را در ثبات AL قرار می‌دهد. همانطوریکه میدانیم B4 در مبنای شانزده معادل 180 در مبنای ده می‌باشد.

AL

10110100

در اینجا گرچه MSB ثبات AL یک می‌باشد ولی چون از دستورالعمل MUL استفاده می‌شود محتوی AL را بصورت منفی در نظر نمی‌گیریم.

17 * 180 = 3060

مقدار 3060 در ثبات AX قرار می‌گیرد.

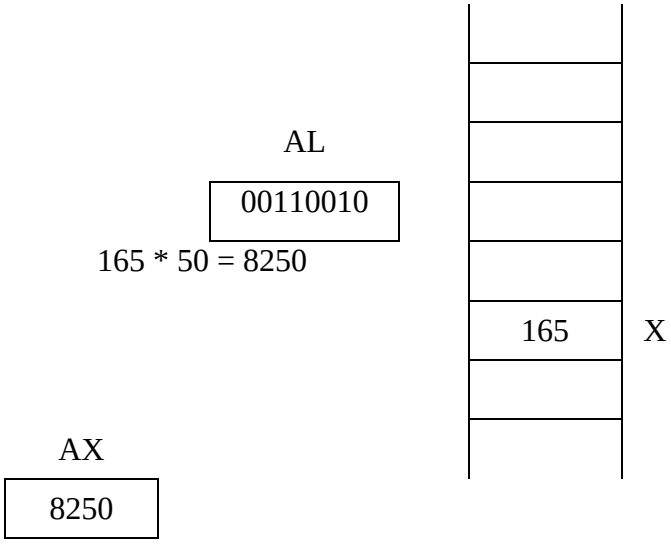
AX

3060

مثال ۲۵-٤

```
X       DB      ?
MOV     X, 0A5H
MOV     AL, 62O
MUL     X
```

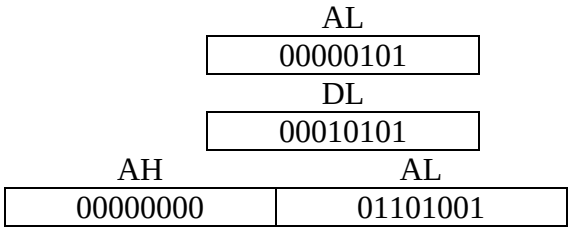
اولین دستورالعمل متغیر X را از نوع بایت تعریف نموده،دومین دستورالعمل 0A5H یعنی مقدار 165 را در متغیر X قرار داده و دستورالعمل سوم 62 در مبنای هشت یعنی مقدار 50 را در ثبات AL قرار می‌دهد.



مثال ٢٦-٤

```
MOV    AL , 5
MOV    DL , 21
MUL    DL
```

اولین دستورالعمل مقدار 5 را در ثبات AL قرار داده. دومین دستورالعمل مقدار 21 را در ثبات DL قرار داده. سومین دستورالعمل محتوی ثبات AL را در محتوی ثبات DL ضرب نموده نتیجه را در AX قرار می‌دهد.



در اینجا مقدار OF و CF هر دو صفر می‌شود که نشان دهنده اینست که نتیجه حاصلضرب دو بایت در یک بایت جای می‌شود و نتیجه در AL قرار می‌گیرد و مقدار ثبات AH صفر می‌باشد.

دستورالعمل MUL نیز برای محاسبه حاصلضرب دو مقدار از نوع word نیز می‌توان استفاده نمود. برای این کار یکی از عملوندها بایستی حتماًدر ثبات AX قرار گیرد. بایستی توجه داشت که نتیجه حاصلضرب در ثباتهای DX:AX قرار می‌گیرد.

مثال ٢٧-٤

```
MOV    AX, 2000
MOV    BX, 15
MUL    BX
```

اولین دستورالعمل مقدار 2000 را در ثبات AX قرار می‌دهد. دومین دستورالعمل مقدار 15 را در ثبات BX قرار می‌دهد. سومین دستورالعمل حاصلضرب محتوی ثباتهای AX و BX را محاسبه نموده نتیجه حاصلضرب را در ثباتهای DX:AX قرار می‌دهد.

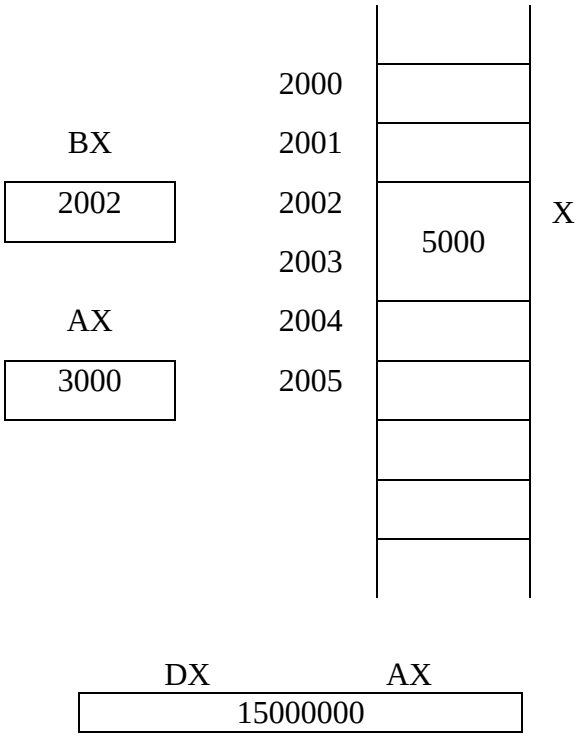
15 * 2000 = 30000

یعنی مقدار 30000 را در ثباتهای DX:AX قرار می‌دهد. در اینجا چون نتیجه در یک Word جا می‌شود مقدار فلگ‌های OF, CF برابر با صفر می‌شود.

مثال ۲۸-٤

```
X       DW      5000
MOV     AX, 3000
LEA     BX , X
MUL     [BX]
```

اولین دستورالعمل متغیر X را از نوع Word با مقدار 5000 تعریف نموده دومین دستورالعمل مقدار 3000 را در ثبات AX قرار می‌دهد. سومین دستورالعمل آدرس X را در ثبات BX قرار می‌دهد. آخرین دستورالعمل مقداری که توسط ثبات BX اشاره می‌شود یعنی 5000 را در محتوی AX یعنی 3000 ضرب نموده نتیجه در ثباتهای DX:AX قرار می‌دهد.



دستورالعمل IMUL نیز برای حاصلضرب دو مقدار استفاده می‌گردد. با این تفاوت که عملوندها را بصورت علامتدار (Signed) در نظر می‌گیرد. مثال

```
MOV     AL, 11H
MOV     BL, 0B4H
IMUL        BL
```

دستورالعمل اول مقدار 11 H یعنی 17 را در ثبات AL قرار می‌دهد.

AL
00010001

دستورالعمل دوم مقدار B4H یعنی 10110100 را در ثبات BL قرار می‌دهد.

BL
10110100

چون MSB محتوی BL برابر با یک می‌باشد محتوی BL را بصورت منفی در نظر می‌گیریم.

1	0	1	1	0	1	0	0
128	64	32	16	8	4	2	1

$$128 + 32 + 16 + 4 = 180$$
$$180 - 2^8 = 180\text{-}256 = \text{-}76$$

آخرین دستورالعمل مقدار 76– را در 17 ضرب نموده نتیجه یعنی 1292– در ثبات AX قرار می‌گیرد.

شکل کلی این دستورالعمل بصورت زیر می‌باشد.

IMUL opr

در مورد عملوند opr نکات زیر را بایستی رعایت نمود.

الف) عملوند بایستی از نوع بایت یا word باشد.

ب) عملوند بایستی از نوع متغییر یا ثبات باشد.

ج) عملوند نبایستی ثابت باشد.

د) دستورالعمل IMUL روی فلگ‌های OF, CF اثر دارد.

هـ) در مورداین دستورالعمل مقادیر فلگ‌هـای SF، AF،ZF و PF تعریف نشده می‌باشند.

و) چنانچه عملوند opr از نوع بایت باشد محتوی ثبات AL در محتوی opr ضرب شده نتیجه حاصلضرب در AX قرار داده می‌شود.

ز)چنانچه عملوند opr از نوع word باشد محتوی ثبات AX در محتوی opr ضرب شده نتیجه در DX:AX قرار داده می‌شود.

مثال ۲۹-٤

```
X       DW    ?
MOV     X, -300
MOV     AX, 20
IMUL    X
```

اولین دستورالعمل X را از نوع word تعریف نمـوده، دومـین دسـتورالعمل مقدار 300– را در متغیر X قرار داده، سـومین دسـتورالعمل مقـدار 20 را در ثبـات AX قرار داده حاصلضرب یعنی 6000- را در ثباتهای DX:AX قرار می‌دهـد. در اینجا مقدار فلگ‌های CF و OF برابر با صفر می‌شود که نتیجه می‌شود مقـدار DX برابر با صفر می‌باشد و مقدار AX برابر با 6000– می‌باشد.

```
X       DB    10110110B
MOV     AL, 10010001B
IMUL    X
```

اولین دستورالعمل مقدار X را از نوع بایت بصورت زیر تعریف می‌نماید.

متغیر X

10110110

دومین دستورالعمل مقدار AL را بصورت زیر تعریف می‌نماید.

AL

10010001

آخرین دستورالعمل محتوی ثبات AL را در مقدار متغیـر X ضـرب نمـوده نتیجه را در AX قرار می‌دهـد. چـون از دسـتورالعمل IMUL اسـتفاده گردیـده و MSB متغیر X و ثبات AL برابر با یـک مـی‌باشـد مقـادیر متغیـر X و ثبـات AL بصورت منفی در نظر گرفته می‌شود.

1	0	1	1	0	1	1	0
128	64	32	16	8	4	2	1

128 + 32 + 16 + 4 + 2 = 192
192 – 256 = -64

حال

1	0	0	1	0	0	0	1
128	64	32	16	8	4	2	1

128 + 16 + 1 = 145
145 – 256 = -91

٦-٤- ضرب دو مقدار 32 بیتی بدون علامت

از دستور MUL وقتی استفاده می‌شود که عملوندها هشت یا شانزده بیتی باشند. اما برای ضرب دو مقدار بدون علامت 32 بیتی بایستی از الگوریتم زیر استفاده نمود. همانطوریکه وقتی دو مقدار را روی کاغذ می‌خواهیم ضرب نمائیم برای جمع حاصلضرب‌های جزئی، آنها را یک ستون بطرف چپ شیفت می‌دهیم از این روش بایستی استفاده نمود برای ضرب مقادیر بزرگ. بعنوان مثال دو مقدار 124 و 103 را در نظر بگیرید.

مثال ٣٠-٤

		1	2	4	*	
		1	0	3		
		3	7	2		حاصلضرب جزئی ١
	0	0	0			حاصلضرب جزئی ٢
1	2	4				حاصلضرب جزئی ٣
1	2	7	7	2		حاصلضرب نهائی

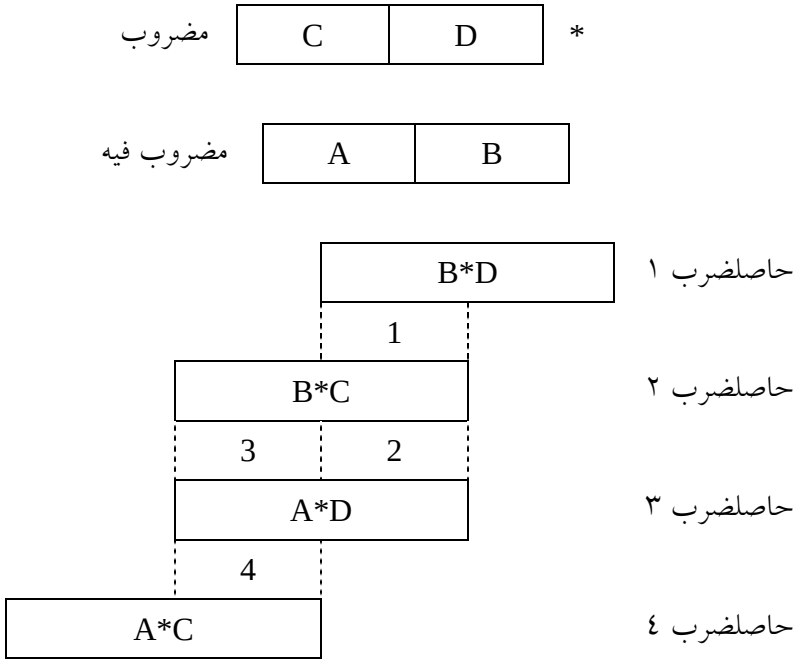
همانطور که دقت می‌کنید

$$103*124 = (3*124) + (0*124) + (100*124)$$

یا

$$103*124 = (3*1*124) + (0*10*124) + (1*100*124)$$

با این طریق می‌توان با استفاده از دستور MUL دو مقدار 32 بیتی را بدون علامت را در هم ضرب و به یک نتیجه 64 بیتی رسید. در شکل زیر D ، C ، B و A، هر کدام بصورت 2 بایت در نظر گرفته شده است.



حاصلضرب نهائی (64 بیتی) = مجموع

برنامه این الگوریتم در فصل نهم کتاب داده شده است.

۷-٤- تقسیم دو مقدار

بااستفاده از دستورالعمل DIV می‌توان دو مقدار را بر هم تقسیم نمود. شکل کلی دستورالعمل DIV بصورت زیر می‌باشد:

```
DIV    Opr
```

در مورد عملوند opr بایستی نکات زیر را رعایت نمود:

الف) opr بایستی از نوع بایت یا word باشد.

ب) opr نمی‌تواند ثابت باشد.

ج) opr بایستی از نوع ثبات یا متغیر باشد.

د) چنانچه opr از نوع بایت باشد محتوی محتوی ثبات AX بر opr تقسیم شده، خارج قسمت را در ثبات AL و باقیمانده را در ثبات AH قرار می‌دهد.

هـ) چنانچه opr از نوع word باشد محتوی ثباتهای DX:AX را بر opr تقسیم نموده، نتیجه تقسیم را در AX و باقی مانده را در DX قرار می‌دهد.

ز) دستورالعمل DIV هر دو عملوند را بصورت باقی مانده unsigned (بدون علامت) در نظر می‌گیرد.

مثال ۳۱-٤

```
MOV    AX , 130
MOV    BL , 5
DIV    BL
```

در اولین دستورالعمل محتوی AX می‌شود 130، دومین دستورالعمل مقدار 5 را در ثبات BL قرار می‌دهد. آخرین دستورالعمل محتوی AX را بر محتوی BL تقسیم نموده نتیجه تقسیم را در AL و باقیمانده را در AH قرار می‌دهد.

AX

0000000010000010

BL
00000101

پس از اجرای دستورالعمل تقسیم داریم که

خارج قسمت AL
00011010

باقیمانده AH
00000000

BL
00000101

بایستی توجه داشت که دستورالعمل DIV بر روی هیچ فلگی اثر ندارد و مقدار فلگ‌هایAF، OF، PF، SF، ZF، CF تعریف نشده می‌باشند. در ضمن بایستی توجه داشت که مقدار عملوند opr بدون تغییر باقی می‌ماند.

مثال ۳۲-٤

```
X       DB   10110100B
MOV          AX, 0400H
DIV          X
```

اولین دستورالعمل مقدار 10110100B را در متغیر X قرار می‌دهد.

X
10110100

1	0	1	1	0	1	0	0
128	64	32	16	8	4	2	1

128 + 32 + 16 + 4 = 180

دومین دستورالعمل مقدار 0400H را در ثبات AX قرار می‌دهد.

AX

0000010000000000

محتوی ثبات AX برابر با 1024 می‌باشد. آخرین دستورالعمل مقدار 1024 را بر 180 تقسیم نموده مقدار خارج قسمت یعنی 5 را در ثبات AL و باقیمانده یعنی 124 را در ثبات AH قرار می‌دهد. پس از انجام عمل تقسیم داریم که

AL

00000101

AH

01111100

متغیر X

10110100

مثال ۳۳-٤

```
X     DW   2600
MOV   AX, 00A2H
MOV   DX, 0B1CH
DIV   X
```

DX	AX
0000101100001100	0000000010100010

اولین دستورالعمل متغیر X را از نوع Word با مقدار 2600 تعریف نموده، دومین دستورالعمل مقدار 00A2H را در ثبات AX قرار داده و دستورالعمل‌سوم مقدار BlC در سیستم 16 تائی را در ثبات AX قرار داده و نهایتاً آخرین دستورالعمل محتوی DX: AX یعنی 0B1C00A2 در سیستم 16 تائی را بر

2600 تقسیم نموده خارج قسمت را در AX و باقیمانده در ثبات DX قرار می‌دهد و مقدار X بدون تغییر یعنی مقدار 2600 باقی می‌ماند.

دستورالعمل IDIV مشابه دستورالعمل DIV می‌باشد با این تفاوت که عملوندها را بصورت علامتدار در نظر می‌گیرد. شکل کلی این دستورالعمل بصورت زیر می‌باشد.

IDIV Opr

در مورد استفاده از دستورالعمل IDIV بایستی نکات زیر را در نظر داشت.

الف) عملوند opr بایستی از نوع بایت یا word باشد.

ب) عملوند opr نمی‌تواند ثابت باشد.

ج) عملوند opr بایستی از نوع ثبات یا متغیر باشد.

د) چنانچه opr از نوع بایت باشد محتوی ثبات AX بر مقدار opr تقسیم نموده خارج قسمت را در ثبات AL و باقیمانده را در ثبات AH قرار می‌دهد.

هـ) چنانچه opr از نوع word باشد محتوی ثبات DX:AX را بر opr تقسیم نموده و نتیجه تقسیم را در AX و باقیمانده را در ثبات DX قرار می‌دهد.

ز) دستورالعمل IDIV، هر دو عملوند را بصورت Signed (علامتدار) در نظر می‌گیرد.

مثال ٣٤-٤

```
MOV   BL, 0B4H
MOV   AX, 0400H
IDIV  BL
```

BL

10110100

AX

0000010000000000

اولین دستورالعمل مقدار B4 در سیستم مبنای 16 را در ثبات BL قرار می‌دهد چون از دستورالعمل IDIV استفاده شده است عملوندها را بصورت علامت دار در نظر می‌گیرد. یعنی اگر MSB عملوند برابر با یک باشد آن را منفی تلقی می‌نماید بنابراین مقدار ثبات BL را بصورت زیر در نظر می‌گیرد.

1	0	1	1	0	1	0	0
128	64	32	16	8	4	2	1

128 + 32 + 16 + 4 = 180
180 – 256 = -76

دومین دستورالعمل مقدار 400H را در ثبات AX قرار می‌دهد، که مقدار آن برابر با 1024 می‌باشد. آخرین دستورالعمل مقدار 1024 را بر 76– تقسیم نموده، نتیجه تقسیم برابر با 13– می‌باشد که در ثبات AL قرار داده می‌شود و باقیمانده راکه برابر با 36 می‌باشد در ثبات AH قرار می‌دهد و مقدار BL بدون تغییر باقی می‌ماند. مقادیر ثباتها پس از اجرای دستورالعملها عبارتند از :

AL
11110011
AH
00100100
BL
10110100

بایستی توجه داشت که دستورالعمل IDIV روی هیچ فلگی اثر ندارد و مقادیر فلگ‌هایAF، CF، OF، PF، ZF، SF تعریف نشده می‌باشند.

مثال ٣٥-٤

```
MOV     AX, 2ACH
MOV     DX, 0B2H
MOV     BX, 004AH
IDIV    BX
```

اولین دستورالعمل مقدار 2AC در سیستم مبنای 16 را در ثبات AX و مقدار B2 در سیستم مبنای 16 را در ثبات DX قرار داده محتوی ثباتهای DX:AX یعنی 00B202AC در سیستم 16 مبنای را بر 004A در سیستم مبنای 16 تقسیم نموده نتیجه تقسیم را در AX و خارج قسمت را در DX قرار می‌دهد.

مثال ٣٦-٤

```
X       DB    0A2H
MOV     AX, 0502H
IDIV    X
```

اولین دستورالعمل مقدار A2 در سیستم 16 تائی یعنی 10100010 در سیستم دودویی را در متغیر X قرار می‌دهد.

متغیر X

10100010

1	0	1	0	0	0	1	0
128	64	32	16	8	4	2	1

128 + 32 + 2 = 162
162 – 256 = -94

دستورالعمل دوّم مقدار 0502H را در ثبات AX قرار می‌دهد.

AX

0000010100000010

1	0	1	0	0	0	0	0	0	1	0
1024	512	256	128	64	32	16	8	4	2	1

$$1024 + 256 + 2 = 1282$$

آخرین دستورالعمل مقدار 1282 را بر 94– تقسیم نموده خارج قسمت که برابر با 13– می‌باشد را در ثبات AL و باقیمانده را که معادل 60 می‌باشد در ثبات AH قرار می‌دهد و مقدار X همان مقدار قبلی یعنی A2H را دارد.

متغیر X

10100010

AL

11110011

AH

00111100

۸-٤- دستورالعملهای کاهش و افزایش

با استفاده از دستورالعمل DEC می‌توان یک واحد از مقدار عملوند کم نمود. شکل کلی دستورالعمل بصورت زیر می‌باشد.

DEC opr

نکات ذیل را بایستی در موقع استفاده از این دستورالعمل در نظر داشت.

الف) opr بایستی از نوع word یا بایت باشد.

ب)opr نمی‌تواند ثابت باشد.

ج)این دستور العمل فقط روی فلگ‌های SF ، OF، ZF ، AF ، PF اثر دارد.

مثال۳۷-٤

```
MOV     AX, 3000
DEC     AX
```

دستورالعمل اول مقدار 3000 را در ثبات AX قرار مـی‌دهـد. دسـتورالعمل دوم یک واحد از محتوی AX کم نموده نتیجه را در AX قرار می‌دهد.

قبل از اجرای DEC

AX
3000

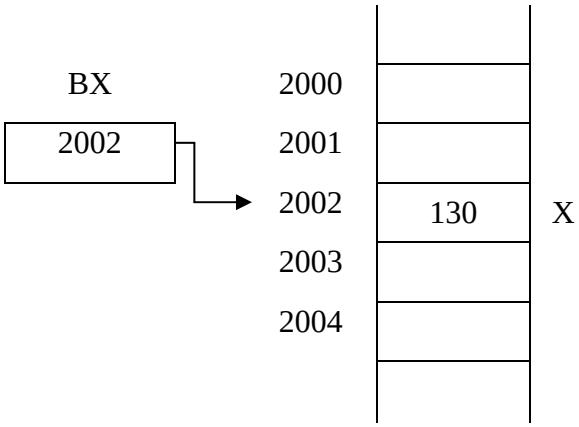
بعد از اجرای DEC

AX
2999

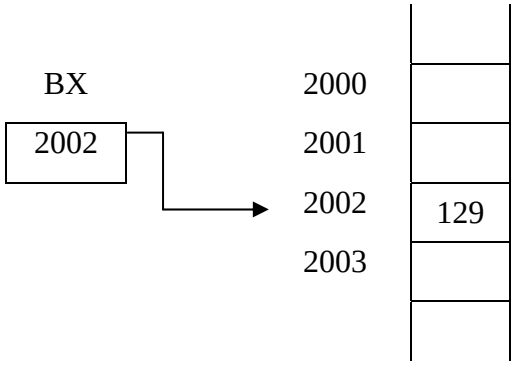
مثال ۳۸-٤

```
X     DB    130
LEA   BX , X
DEC     [BX]
```

اولین دستورالعمل مقدار 130 را در متغیر X قرار داده، دومین دسـتورالعمل آدرس متغیر X را در ثبات BX قرار می‌دهد. آنگاه محتوی محلی کـه بوسـیله BX اشاره می‌شود را یکی کاهش می‌دهد.



بعد از اجرای دستورالعمل DEC



دستورالعمل INC باعث می‌شود که یک واحد به عملوند اضافه گردد. شکل کل این دستورالعمل عبارتست از

INC Opr

در مورد استفاده از این دستورالعمل بایستی نکات ذیل را رعایت نمود.

الف) opr نمی‌تواند ثابت باشد.

ب) opr بایستی از نوع ثبات یا متغیر باشد.

ج) opr بایستی از نوع بایت یا word باشد.

د) این دستورالعمل روی فلگ‌های SF ، OF، ZF ، AF و PF اثر دارد.

مثال ۳۹-٤

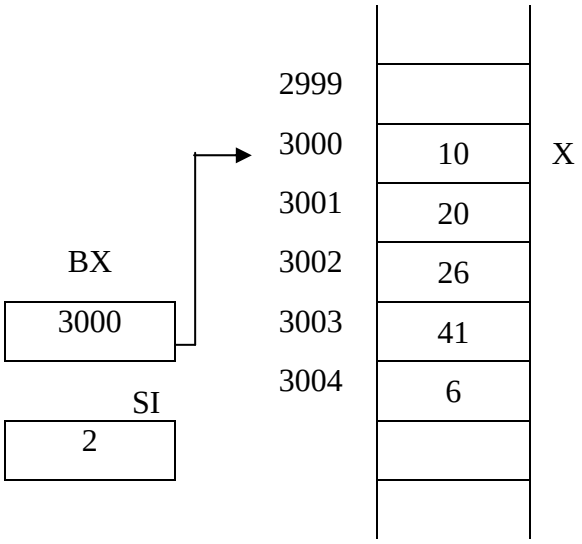
```
MOV     AL, 100
INC     AL
```

مقدار AL را به 101 افزایش می‌دهد.

مثال ٤٠-٤

```
X       DB  10,20,26,44,6
MOV         SI , 2
MOV     BX, OFFSET  X
INC         [BX] [SI]
```

اولین دستورالعمل یک آرایه 5 عنصری از نوع بایت بنام X ایجاد می‌نماید. مقادیر عناصر آرایه عبارتند از بترتیب6، 40، 26، 20، 10. دستورالعمل دوم مقدار 2 را در رجیسر SI قرار می‌دهد. دستورالعمل سوم آدرس متغیر X را در ثبات BX قرار می‌دهد. آخرین دستورالعمل یک واحد به محتوی خانه‌ای از حافظه که بوسیله محتوی BX+2 اشاره می‌کند اضافه می‌گرداند.



در حقیقت مقدار خانه حافظه با آدرس 3002 از 26 به 27 افزایش می‌یابد.

۹-٤- دستورالعمل محاسبه مکمل ۲

برای پیدا نمودن مکمل 2 یک مقدار، از دستورالعمل NEG استفاده می‌گردد. شکل کلی آن بصورت زیر می‌باشد.

NEG Opr

الف) مقدار مکمل 2 عملوند opr را محاسبه نموده در opr قرار می‌دهد.

ب) opr می‌تواند از نوع بایت یا word باشد.

ج) opr می‌تواند ثبات یا متغیر باشد.

د) opr ثابت نمی‌تواند باشد.

هـ) روی فلگ‌های SF ، OF، CF، ZF ، AF و PF اثر دارد.

مثال ٤١-٤

```
MOV     AX, -100
NEG       AX
```

مقدار AX را به 100 تغییر می‌دهد.

```
X       DB      ?
MOV       X, 26
NEG         X
```

مقدار X که از نوع بایت می‌باشد نهایتاً برابر با 26- می‌باشد.

در فصل نهم نحوه نوشتن برنامه و اجزای آن بیان گردیده است.

دستورالعمل پرش غیر شرطی

دستورالعمل پرش غیر شرطی در زبان اسمبلی JMP می‌باشد. این دستورالعمل معادل دستورالعمل GOTO در سایر زبانهای برنامه‌نویسی می‌باشد.

شکل کلی این دستورالعمل بصورت زیر می‌باشد. این دستور روی هیچ فلگی اثر ندارد.

```
JMP        آدرس
```

مثال ۱-۵

```
JMP        LAB2
```

با اجرای این دستورالعمل کنترل به LAB2 منتقل می‌گردد. بایستی توجه داشت که کنترل بدون هیچ گونه قید و شرطی به LAB2 منتقل می‌گردد.

مثال ۲-۵

```
      MOV   AL, 5
      ADD   AL, BL
      JMP   LAB1
      MUL   BL
      INC   BL
LAB1: SUB   CX , 2
      ⋮
```

در قطعه برنامه فوق ابتدا مقدار 5 در ثبات AL قرار می‌گیرد، سپس مقدار BL به آن اضافه گردید. آنگاه کنترل به LAB1 منتقل می‌گردد و دستورالعمل SUB به بعد اجرا می‌گردد. بایستی توجه داشت که دو دستور MUL و INC اجرا نمی‌شوند.

۲-۵- دستورالعملهای پرش شرطی

دستورالعملهای پرش شرطی وقتی اجرا می‌گردد که در برنامه شرطی برقرار گردد. شکل کلی این دستورالعملها بصورت زیر می‌باشد.

JX آدرس

که X یک رشته یک تا سه کارکتری می‌باشد.

مثال ۳-۵

JZ LAB2

در صورتیکه مقدار ZF برابر با یک باشد کنترل به LAB2 در برنامه منتقل می‌گردد.

مثال ٤-۵

JS LAB5

در صورتیکه مقدار SF برابر با یک باشد کنترل به LAB5 در برنامه منتقل می‌گردد.

مثال ۵-۵

JNO LAB20

چنانچه مقدار OF برابر با صفر باشد کنترل به LAB20 در برنامه منتقل می‌گردد. از این دستورالعمل معمولاً پس از اجرای عملیات ریاضی استفاده می‌شود.

مثال ٦-۵

```
        MOV   AX, -100
        ADD   AX, BX
        SUB   AX, CX
        JNZ   LABNEXT
              ⋮
LABNEXT:  MOV CX,10
              ⋮
```

در قطعه برنامه بالا ابتدا مقـدار 100– در ثبـات AX قـرار داده مـی‌شـود و سپس مقدار BX به آن افزوده می‌گردد و سپس مقـدار CX از آن کـسر مـی‌گـردد. حال چنانچه نتیجه محاسبه یعنی مقدار AX مخالف صـفر باشـد کنتـرل بـه آدرس LABNEXT در برنامه منتقل می‌گردد.

جدول ذیل انواع دستورالعمل‌های پرش شرطی را نشان می‌دهد.

جدول ۱–۵

شرط تست	نام دیگر دستورالعمل	نام دستورالعمل	عمل
ZF=1	JE	JZ	انشعاب روی صفر
ZF=0	JNE	JNZ	انشعاب روی مخالف صفر
SF=1		JS	انشعاب روی علامت منفی
SF=0		JNS	انشعاب روی علامت غیر منفی
OF=1		JO	انشعاب روی سرریزی
OF=0		JNO	انشعاب روی عدم سرریزی
PF=1	JPE	JP	انشعاب روی ایجاد بیت توازن
PF=0	JPO	JNP	انشعاب روی عدم ایجاد بیت توازن
CF=1		JC	انشعاب روی ایجاد بیت نقلی
CF=0		JNC	انشعاب روی عدم ایجاد بیت نقلی

در جدول بالا حروف مخفف کلمات زیر می‌باشند.

Z	ZERO	
S	SIGN	
N	NOT	
P	PARITY	
O	OVERFLOW	
O	ODD	در JPO
E	EQUAL	
J	JUMP	
E	EVEN	در JPE
C	CARRY	

بایستی توجه داشت که دستورالعملهای پرش در حقیقت نقش دستورالعمل IF در سایر زبانهای برنامه‌نویسی را دارد.

مثال ۷-۵

```
TOT   DW  ?
MOV   TOT, 0
MOV   CX, 10
BEGIN: ADD   TOT, CX
DEC   CX
JNZ   BEGIN
```

در قطعه برنامه بالا متغیر TOT از نوع word تعریف گردیده و مقدار آن برابر با صفر قرار داده شده است. مقدار اولیه CX نیز برابر با 10 می‌باشد. قطعه برنامه نقش یک حلقه تکرار دارد که مقادیر 1 تا 10 را با هم جمع می‌نماید یعنی تا مادامیکه مقدار CX مخالف صفر می‌باشد، مقدار CX با TOT جمع می‌گردد و یک واحد از CX کم می‌گردد.

مثال ۸-۵

```
X   DW   ?
MOV   AX ,X
SUB   AX, 100
NEG   AX
JNS   ACT2
      ⋮
ACT2 :   ADD BX , AX
      ⋮
```

مقدار X در ثبات AX قرار داده شده آنگاه 100 واحد کاهش داده شده سپس مقدار AX در منفی یک ضرب شده حال چنانچه مقدار AX منفی نباشد کنترل به ACT2 منتقل می‌گردد. در غیر اینصورت اجرای دستورالعملهای بعدی ادامه می‌یابد.

۳-۵- دستورالعمل مقایسه

دستورالعمل مقایسه در زبان اسمبلی CMP می‌باشد. شکل کلی دستورالعمل بصورت زیر می‌باشد.

```
CMP   opr1 , opr2
```

الف) opr1 و opr2 هر دو از نوع بایت یا word می‌باشند. این دستورالعمل مقادیر عملوندها را تغییر نمی‌دهد.

ب) opr1 و opr2 می‌توانند هر دو ثبات باشند.

ج) opr1 و opr2 هر دو نمی‌توانند متغیر باشند.

د) opr1 و opr2 هر دو نمی‌توانند ثابت باشند.

هـ)دستورالعمل CMP مانند دستورالعمل SUB عمل می‌کند، با این تفاوت که نتیجه درجائی ذخیره نمی‌گردد بلکه مقادیر فلگ‌ها را تغییر می‌دهد.

ز) این دستورالعمل روی فلگ‌هایAF ، OF، SF، PF، ZF، SF اثر دارد.

مثال ۹-۵

```
CMP     AX , BX
```

این دستورالعمل دو مقدار AX , BX را با هم مقایسه می‌نماید. در حقیقت مقدار BX را از AX کم نموده و برحسب نتیجه بدست آمده مقادیر فلگها را تعیین می‌نماید.

مثال ۱۰-۵

```
CMP     AL , 10
JZ      LAB2
```

مقدار AL را با 10 مقایسه نموده در صورتیکه برابر باشند کنترل به LAB2 منتقل می‌گردد.

تعدادی دستورالعمل‌های پرش شرطی وجود دارند که با دستورالعمل CMP استفاده می‌گردند. دستورالعملهای پرش زیر وقتی استفاده می‌گردند که عملوندها بصورت بدون علامت (Unsigned) در نظر گرفته شوند.

جدول ۲-۵

نام	نامهای دیگر	شرط	فلگ‌ها
JB	JNAE , JC	Opr 1 < Opr2	CF = 1
JNB	JAE, JNC	Opr 1 > = Opr2	CF = 0
JBE	JNA	Opr 1 < = Opr2	CF ∨ ZF = 1
JNBE	JA	Opr 1 > Opr2	CF ∨ ZF = 0

دستورالعملهای پرش زیر وقتی استفاده می‌شوند که عملوندها بصورت علامتدار (Signed) در نظر گرفته شوند.

جدول ۳-۵

نام	نامهای دیگر	شرط	فلگ‌ها
JL	JNGE	Opr1 < Opr 2	SF ⊕ OF =1
JNL	JGE	Opr 1 > = Opr2	SF ⊕ OF =0
JLE	JNG	Opr1 < = Opr2	(SF ⊕ OF) ∨ ZF=1
JNLE	JG	Opr1 > Opr2	(SF⊕OF) ∨ ZF=0

حروف مخفف کلمات ذیل می‌باشند.

B	Below
A	Above
G	Greater than
E	Equal to
L	Less than
C	Carry
N	Not

مقصود از علامت V عملگر OR و مقصود از علامت ⊕ عملگر Exclusive OR می‌باشد.

```
        MOV   AX, [BX]

        CMP   AX, [DI]  ; مقدار اول در AX با مقدار دوم مقایسه می‌شود
        JBE   DONE      ; آیا مقدار اول کمتر یا مساوی مقدار دوم می‌باشد؟
        XCHG  AX , [DI] ; در غیر اینصورت مبادلهٔ مقادیر
        MOV   [BX] , AX
DONE:
        ⋮
```

در قطعه برنامه بالا دو مقدار از حافظه که بوسیله ثباتهای DI و BX مشخص می‌شوند را بترتیب صعودی مرتب می‌نماید.

مثال ۱۱-۵

```
        CMP     AL , 10 ;
        JAE     LABI
        ⋮
LAB1: JA    LAB 2
        ⋮
LAB2:
        ⋮
```

اگر محتوی AL بزرگتر از 10 باشد کنتـرل بـه LAB2 ، اگـر محتـوی AL مساوی 10 باشد کنترل به LAB1 در غیر اینصورت کنترل به دسـتورالعمل بعـد از دستورالعمل JAE منتقل می‌گردد.

```
CMP   AL , BL
JE    ZERO
```

کنترل به آدرس ZERO منتقل مـی‌گـردد اگـر مقـادیر BL و AL مـساوی می‌باشند.

مثال ۱۲-۵

```
MOV   AX , -100
CMP   BX , AX
JG    LAB2
```

عملوندهای CMP علامت دار در نظر گرفته می‌شوند.

مثال ۱۳-۵

```
MOV     AX , 100
CMP     BX, AX
JA      LAB2
```

عملوندهای CMP بدون علامت در نظر گرفته می‌شوند.

٤-۵- دستورالعملهای تکرار

هر وقت بخواهیم تعدادی دستورالعمل بصورت مکرر اجرا گردد از دستورالعملهای تکرار بایستی استفاده نمائیم. دستورالعمل تکرار در زبان اسمبلی LOOP می‌باشد که شکل کلی آن بصورت زیر می‌باشد.

LOOP آدرس

هر وقت کنترل بدستور LOOP میرسد ابتدا مقدار ثبات CX یک واحد کاهش یافته سپس محتوی ثبات CX با صفر مقایسه می‌گردد و در صورتیکه محتوی ثبات CX مخالف صفر باشد کنترل به آدرس داده شده منتقل می‌گردد. تعداد دفعات تکرار عملاً بایستی در ثبات CX قرار داد. دستورالعمل LOOP روی هیچ فلگی اثر ندارد.

مثال ١٤-۵

```
        MOV     CX , 10
LABI:
        ⋮
        LOOP    LABI
```

این قطعه برنامه معادل برنامه پاسکال زیر می‌باشد یعنی دامنه تکرار ده بار اجرا می‌گردد.

```
FOR   I: =1   TO  10 DO
      BEGIN

        ⋮
      END ;
```

قطعه برنامه زیر مجموع عناصر آرایـه X کـه از نـوع Word و N عنـصری می‌باشد را محاسبه نموده نتیجه را در متغیر TOTAL قرار می‌دهد.

```
N          DW       ?
TOTAL      DW       ?
X          DW            مقادیر عناصر آرایه
MOV        CX , N
MOV        AX, 0  ;      مجموع برابر با صفر
MOV        SI , AX   ;   SI برابر با صفر
START_LOOP:   ADD  AX,X [SI];    جمع عناصر
              ADD   SI, 2
              LOOP   START _ LOOP
              MOV    TOTAL, AX
```

قطعه برنامه زیر آرایـه N عنـصری A از نـوع word را بـصورت صـعودی بروش حبابی مرتب می‌نماید.

```
                Bubble  ; Sort
                N       DW      ?
                MOV     CX , N
                DEC     CX
LOOP 1:         MOV     DI , CX
                MOV     BX , 0
LOOP2:          MOV     AX , A[BX]
                CMP     AX, A[BX+2]
                JGE     CONTINUE
                XCHG    AX, A[BX+2]
                MOV     A [BX] , AX
CONTINUE:       ADD     BX,2
                LOOP    LOOP2
                MOV     CX , DI
                LOOP    LOOP1
```

شکل دیگر دستور تکرار بصورت زیر می‌باشد.

LOOPNE آدرس

یا

LOOPNZ آدرس

کار دستورالعمل LOOPNE یا LOOPNZ مانند دستورالعمل LOOP می‌باشد با این تفاوت که شرط تکرار آن است که مقدار CX مخالف صفر و مقدار ZF برابر با صفر باشد. این دستورالعمل روی هیچ فلگی اثر ندارد.

مثال ۱۵-۵

```
        ARR     DB
        N       DW
        MOV     CX , N
        MOV     SI , -1
        MOV     AL, 20H; ASCII code for blank
NEXT:   INC     SI
        CMP     AL, ARR[SI]; test for blank
        LOOPNE  NEXT
        JNZ     NOT_FOUND
```

قطعه برنامه بالا رشتهٔ داده شده N عنصری ARR از نوع بایت را جستجو می‌نماید که آیا کارکتر blank یا فاصله در رشته وجود دارد یا خیر؟

توجه داشته باشید که دستور تکرار بالا وقتی متوقف می‌شود که عناصر رشته همه مورد بررسی قرار گرفته باشند یا به کارکتر فاصله رسیده باشیم. شکل دیگر دستورالعمل تکرار بصورت زیر می‌باشد.

LOOPE آدرس

یا

LOOPZ آدرس

دستورالعمل LOOPE یا LOOPZ مانند دستورالعمل LOOP عمل می‌نماید با این تفاوت که شرط تکرار آن است که مقدار CX مخالف صفر و مقدار ZF برابر با یک باشد. این دستورالعمل روی هیچ فلگی اثر ندارد.

```
; BX = offset  of   the   starting address
; DX = offset  of   the ending address
; BX = offset  of   nonzero (if found)
; BX = DI (if not found)
```

```
                SUB     DI , BX
                INC     DI ; تعداد بایت = (DI)-(BX)+1
                MOV     CX , DI
                DEC     BX
NEXT:           INC     BX ; point to next location
                CMP     BYTE  PTR [BX], 0 ; Compare it to zero
                LOOPE   NEXT ; go compare next byte
                JNZ     NZ _ FOUND; Nonzero byte found?
                 ⋮      ; NO.
NZ_FOUND:
                 ⋮      ; Yes.
```

برنامه فوق یک بلوک از حافظـه کـه آدرس شـروع ان توسـط ثبـات BX و آدرس انتهای آن توسط ثبات DI مشخص شده را بایت بـه بایـت جـستجو نمـوده برای یافتن عنصری که مخالف صفر باشد.

شکل دیگر دستور تکرار بصورت زیر می‌باشد.

JCXZ آدرس

در حقیقت این دستورالعمل یک نوع دستورالعمل پرش می‌باشد که براساس فلگها عمل نمی‌کند بلکه براساس محتوی ثبات CX عمل می‌کند. چنانچـه محتـوی CX مساوی صفر باشد کنترل به آدرس داده شده منتقل می‌شود. ایـن دسـتورالعمل روی هیچ فلگی اثر ندارد. نهایتاً جدول ذیل را داریم.

جدول ٤-٥

عمل	نام	نام دیگر	شرط تکرار
LOOP	**LOOP**		**CX < > 0**
LOOP While equal or zero	LOOPE	LOOPZ	CX < > 0 and ZF=1
LOOP while not equal or nonzero	LOOPNE	LOOPNZ	CX < > 0 and ZF=0
Branch on CX	JCXZ		CX=0

۱-٦- عملیات منطقی

از دستورالعملهای منطقی برای انجام عملیات منطقی استفاده می‌شود. این دستورالعملها بصورت بیتی روی عملوندها عمل می‌نماید. عملیات منطقی عبارتند از NOT، AND، OR، XOR و TEST .

۱-۱-٦- دستورالعمل NOT

شکل کلی دستورالعمل NOT بصورت زیر می‌باشد.

NOT opr

الف) opr می‌تواند از نوع wordیا بایت باشد.

ب) opr می‌تواند متغیر یا ثبات باشد.

ج) این دستورالعمل بیت‌های opr را از 0 به 1 و از 1 به 0 تبدیل می‌نماید. بعبارت دیگر مکمل 1 عملوند را می‌دهد.

د) دستورالعمل NOT روی هیچ فلگی اثر ندارد.

مثال ۱-٦

```
MOV     DL , 8AH
NOT       DL
```

قبل از اجرای دستور NOT

DL
10001010

بعد از اجرای دستور NOT

DL
01110101

بنابراین محتوی ثبات DL به 75H تغییر می‌نماید.

۲-۱-٦- دستورالعمل AND

شکل کلی دستورالعمل AND بصورت زیر می‌باشد.

```
AND   dst , src
```

الف) عملوندهای src و dst هر دو از نوع بایت یا word می‌باشند.
ب) عملوندهای src و dst هر دو متغیر یاهر دو ثابت نمی‌توانند باشند.
ج) وقتی عملوند src ثابت باشد عملوند dst بایستی از نوع ثبات یا متغیر باشد.
د) بیت‌های dst و src نظیر به نظیر مطابق جدول ذیل and می‌شوند و نتیجه در dst قرار می‌گیرد.

جدول ۱-٦

بیت اول	بیت دوم	بیت دوم **and** بیت اول
0	0	0
0	1	0
1	0	0
1	1	1

مثال ۲-٦

```
MOV   AL , 5BH
MOV   DH, 4DH
AND   AL, DH
```

مقادیر ثباتها قبل از اجرای دستورالعمل AND عبارتست از:

AL	01011011
DH	01001101

مقادیر ثباتها پس از اجرای دستورالعمل AND عبارتند از:

DH	01001101	بدون تغییر
AL	01001001	نتیجه

۳-۱-٦- دستورالعمل OR

شکل کلی دستورالعمل OR بصورت زیر می‌باشد.

```
OR    dst , src
```

الف) عملوندهای dst و src از نوع بایت یا word می‌باشند.

ب) عملوندهای dst و src هر دو متغیر یا هر دو ثابت نمی‌توانند باشند.

ج) وقتی عملوند src ثابت باشد عملوند dst بایستی از نوع متغیر یا ثبات باشد.

د) بیت‌های dst و src نظیر به نظیر مطابق جدول ذیل or می‌شود ونتیجه در dst قرار می‌گیرد.

جدول ۲-٦

بیت اول	بیت دوم	بیت دوم **or** بیت اول
0	0	0
0	1	1
1	0	1
1	1	1

مثال۳-٦

```
MOV   BL , 0A5H
MOV   AL, 2AH
OR    AL , BL
```

مقادیر ثبات ها قبل از اجرای دستورالعمل OR

BL	10100101
AL	00101010

مقادیر ثباتها بعد از اجرای دستورالعمل OR

BL	10100101
AL	10101111

٤-١-٦- دستورالعمل XOR

شکل کلی دستورالعمل XOR بصورت زیر می‌باشد.

XOR dst , src

الف)src و dst هر دو از نوع بایت یا word می‌باشند.

ب) src و dst هر دو متغیر یا ثابت نمی‌تواند باشند.

ج) بیت‌های src و dst نظیر به نظیر با استفاده از جدول ذیل xor گردیده نتیجه در dst قرار می‌گیرد و مقدار src بدون تغییر باقی می‌ماند.

جدول ٣-٦

بیت اول	بیت دوم	بیت دوم **xor** بیت اول
0	0	0
0	1	1
1	0	1
1	1	0

مثال ٤-٦

```
MOV   CL, 2DH
MOV   AL , 0C2H
XOR   AL , CL
```

مقادیر ثباتها قبل از اجرای دستورالعمل XOR

AL	11000010
CL	00111101

مقادیر ثباتها بعد از اجرای دستورالعمل XOR

CL	0011 1101
AL	1111 1111

٥-١-٦- دستورالعمل TEST

شکل کلی دستورالعمل TEST بصورت زیر می‌باشد:

```
TEST   opr1, opr2
```

الف) opr1 و opr2 هر دو از نوع بایت یا word می‌باشد.

ب) opr1 و opr2 هر دو ثابت یا هر دو متغیر نمی‌توانند باشند.

ج) این دستورالعمل مانند دستورالعمل AND عمل می‌نماید ولی نتیجه را در جائی ذخیره نمی‌کند یعنی دو عملوند بدون تغییر باقی می‌مانند و فقط مقادیر فلـگ‌هـا را تغییر می‌دهد.

مثال ٥-٦

```
MOV   AL , 25
MOV   DH , 0E4H
TEST  AL , DH
```

مقادیر ثباتها قبل از اجرای دستورالعمل TEST

AL	00011001
DH	11100100

مقادیر ثباتها پس از اجرای دستورالعمل TEST

AL	00011001
DH	11100100

بایستی توجه کرد که دستورالعمل TEST باعث می‌شود که مقدار ZF برابر با یک گردد.

مثال ٦-٦

```
      MOV   AL , 0ABH
      NOT   AL
      TEST  AL, 10100101B
      JZ    YES
      ⋮
YES:
      ⋮
```

قطعه برنامه فوق مشخص می‌نماید که آیا مقادیر بیت‌های7، 5، 2 و 0 ثبات AL برابر با یک می‌باشد یا خیر؟ مقدار 10100101B عملاً MASK می‌باشد که

در بیت‌هائی که می‌خواهیم برای یک بودن تست شود مقدار یک و در سایر بیت‌ها مقدار صفر را قرار می‌دهیم.

AL	10101011

پس از اجرای دستور NOT

AL	01010100
MASK	10100101

در این مثال پس از اجرای دستورالعمل TEST مقدار ثبات AL بدون تغییر باقی ماند و فقط مقدار فلگ ZF برابر با یک می‌شود.

مثال ۷-٦

```
        OR      DL, 00000101B
        XOR     DL, 01000010B
        AND     DL, 11100111B
        MOV     AL, DL
        NOT     AL
        TEST    AL, 10000010B
        JZ      EXIT
        ⋮
EXIT:
        ⋮
```

قطعه برنامه فوق ابتدا بیت‌های شماره 0 و2 ثبات DL را یک می‌کند و بیت های شماره 4 و 3 را به صفر تبدیل می‌کند و بیت‌های شماره 1 و6 را مکمل می-نماید در ضمن چنانچه بیت‌های شماره 7و1 برابر با یک باشند کنترل به EXIT منتقل می‌نماید.

مثال ۸-٦

قطعه برنامه زیر بیت‌های شماره فرد ثبات AL را مکمل می‌نماید. یعنی 1 به 0 و 0 به 1 تبدیل می‌نماید.

```
MOV    AL, 0C7H
MOV    MASK, 10101010B
XOR    AL, MASK
```

AL	11000111
MASK	10101010

پس از اجرای دستورالعمل XOR مقادیر AL و MASK عبارتند از:

AL	01101101
MASK	10101010

مثال ۹-٦

قطعه برنامه زیر بیت‌های شماره زوج ثبات AL را به یک تبدیل می‌نماید.

```
MOV    AL, 0A6H
MOV    MASK, 55H
OR     AL, MASK
```

AL	10100110
MASK	01010101

پس از اجرای دستورالعمل OR

AL	11110111
MASK	01010101

مثال ۱۰-٦

قطعه برنامه زیر بیت‌های شماره فرد AL را به صفر تبدیل می‌نماید.

```
MOV    AL, 0C7H
MOV    MASK, 55H
AND    AL, MASK
```

AL	11000111
MASK	01010101

پس از اجرای دستورالعمل AND

AL	01000101
MASK	01010101

در قطعه برنامه زیر اگر بیت‌های شماره 1 و 14 یا بیت‌های شماره 9 و 7 ثبات AX برابر با یک باشند کنترل به TASK 1 و اگر بیت 3 یا 4 برابر با یک باشند کنترل به TASK2 در غیر اینصورت کنترل به TASK3 منتقل می‌گردد.

```
        NOT     AX
        TEST    AX, 4002H
        JZ      TASK1
        TEST    AX, 280H
        JZ      TASK1
        NOT     AX
        TEST    AX, 18H
        JNZ     TASK2
TASK3:
        ⋮
TASK1:
        ⋮
TASK2:
        ⋮
```

۲-٦- عملیات شیفت

عملیات شیفت باعث تغییر مکان بیت‌های یک بایت یا یک word بطرف چپ یا راست می‌شود. دستورالعملهای متعددی برای این کار مورد استفاده قرار می‌گیرند که عبارتند از:

Shift	Logical Left	SHL
Shift	Logical Right	SHR
Shift	Arithmetic Left	SAL
Shift	Arithmetic Right	SAR

۱-۲-٦- دستورالعمل SHL

شکل کلی دستورالعمل SHL بصورت زیر می‌باشد.

```
SHL     opr , cnt
```

الف) cnt تعداد بیت‌هائی می‌باشد که بطرف چپ شیفت داده می‌شود. در صورتیکه cnt مخالف یک باشد از ثبات CL استفاده می‌نمائیم.

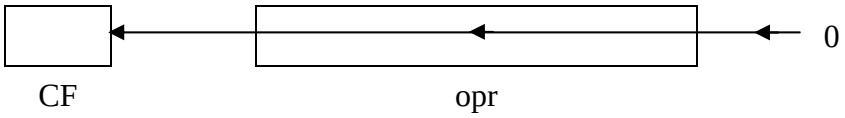
ب) opr می‌تواند از نوع بایت یا word باشد.

ج) oprمی‌تواند متغیر یا ثبات باشد.

د) opr ثابت نمی‌تواند باشد.

هـ) روی فلگهایOF ، SF ،ZF، PF و CF اثر دارد.

ز) بیت‌های oprرا با اندازه cnt بیت بطرف چپ شیفت می‌دهد و از طـرف راسـت با صفر پر می‌شود.



مثال ۱۱-٦

```
SHL   AX, CL
SHL   BL, CL
SHL   AL, 1

STC
MOV   CL, 3
MOV   DL, 8DH
SHL   DL , CL
```

دستورالعمل STC مقدار فلگ CF را به یک تبدیل می‌نماید.

قبل از اجرای دستورالعمل SHL

CL	00000011	CF	1
DL	10001101		

بعد از اجرای دستورالعمل SHL

CL	00000011
DL	01101000
CF	0

۲-۲-٦- دستورالعمل SHR

شکل کلی دستورالعمل SHR بصورت زیر می‌باشد.

SHR opr, cnt

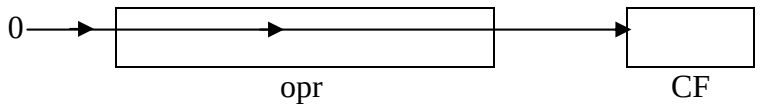
الف) opr می‌تواند از نوع بایت یا word باشد.

ب) opr می‌تواند متغیر یا ثابت باشد. ولی ثابت نمی‌تواند باشد.

ج) روی فلگهای CF ،ZF، PF اثر دارد.

د) اگر مقدار cnt برابر با یک باشد خودش را می‌نویسم در غیر اینصورت از ثبات CL استفاده می‌نمائیم.

هـ) بیت‌های opr را با اندازه cnt بیت بطرف راست شیفت داده و از طرف چپ با صفر پر می‌نمائیم.



مثال ۱۲-٦

```
SHR   DL, 1
SHR   AL, CL
```

مثال ۱۳-٦

```
STC
MOV   CL , 3
MOV   DL, 8DH
SHR   DL, CL
```

محتوی ثبات DL را سه بیت بطرف راست شیفت می‌دهد.

CL	00000011
DL	10001101
CF	1

مقادیر پس از اجرای دستورالعمل SHR عبارتند از

CL	00000011
DL	00010001
CF	1

۳-۲-٦- دستورالعمل SAL

شکل کلی دستورالعمل SAL بصورت زیر می‌باشد.

```
SAL   opr , cnt
```

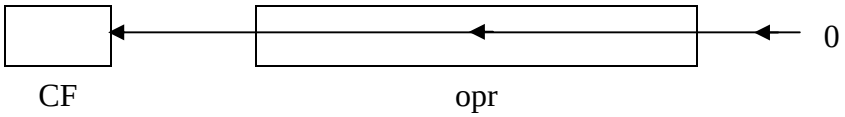
الف) opr می‌تواند از نوع بایت یا word باشد.

ب) cnt اگر یک باشد مقدار یک را می‌نویسم در غیر اینصورت از ثبات CL استفاده می‌نمائیم.

ج) opr ثابت نمی‌تواند باشد.

د) روی فلگهایOF ، SF ،ZF ، PF و CF اثر دارد.

هـ) بیت‌های opr را باندازه cnt بیت بطرف چپ شیفت می‌دهد و از طرف راست با صفر پر می‌شود.



مثال ١٤-٦

```
SAL   DL, 1
SAL   AL, CL
SAL   [BX] , CL
SAL   ARR [SI], CL
```

مثال ١٥-٦

```
 STC
MOV   CL , 3
MOV   DL, 8DH
 SAL   DL, CL
```

قبل از اجرای دستورالعمل SAL

CL	00000011
DL	10001101
CF	1

بعد از اجرای دستورالعمل SAL

CL	00000011
DL	01101000
CF	0

٤-٢-٦- دستورالعمل SAR

شکل کلی دستورالعمل SAR بصورت زیر می‌باشد.

SAR opr , cnt

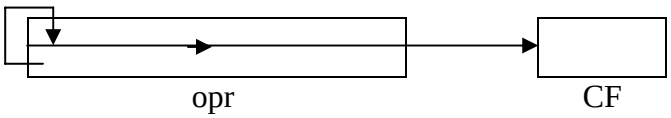
الف) opr می‌تواند از نوع ثبات یا متغیر باشد.

ب) opr نمی تواندثابت باشد.

ج) opr می‌تواند از نوع بایت یا word باشد.

د) cnt اگر معادل یک باشد مقدار 1 در غیر اینصورت از ثبات CL استفاده می‌نمائیم.

هـ) بیت‌های opr را باندازهٔ cnt بیت بطرف راست شیفت داده و از سمت چپ با MSB پر می‌نماید.



مثال ٦-١٦

```
SAR   DL, 1
SAR   AX , CL
SAR   [BX], CL
```

مثال ٦-١٧

```
MOV   CL,3
STC
MOV   DL, 8DH
SAR   DL, CL
```

مقادیر قبل از اجرای دستورالعمل SAR

CL	00000011
DL	10001101
CF	1

مقادیر بعد از اجرای دستورالعمل SAR

CL	00000011
DL	11110001
CF	1

۳-٦- عملیات چرخش (Rotate)

عملیات چرخش باعث دور زدن بیت‌های یک بایت یا word می‌شوند. بعبارت دیگر مانند دستورالعمل‌های شیفت باعث خارج شدن بیت‌ها از بایت یا word نمی‌شود. دستورالعملهای چرخش عبارتند از:

Rotate	Left	ROL
Rotate	Right	ROR
Rotate	Left through Carry	RCL
Rotate	Right through Carry	RCR

۱-۳-٦- دستورالعمل ROL

شکل کلی دستورالعمل ROL بصورت زیر می‌باشد.

ROL opr , cnt

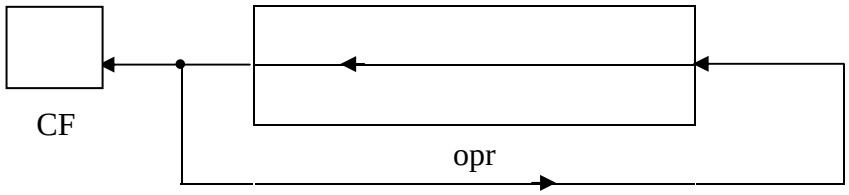
الف) opr می‌بایستی از نوع بایت یا word باشد.

ب) opr می‌بایستی متغیر یا ثبات باشد.

ج) اگر مقدار cnt برابر با یک باشد عدد 1 را می‌نویسم در غیر اینصورت از ثبات CL استفاده می‌کنیم.

د) روی فلگ CF اثر دارد.

هـ) این دستورالعمل باندازه cnt بیت از سمت چپ چرخش می‌دهد.



مثال ۱۸-٦

```
ROL   DL, CL
ROL   BX , 1
ROL   [BX] , CL
```

مثال ۱۹-٦

```
MOV   CL,3
MOV   DL, 8DH
 STC
ROL   DL, CL
```

مقادیر قبل از اجرای دستورالعمل ROL

CL	00000011
DL	10001101
CF	1

مقادیر بعد از اجرای دستورالعمل ROL

CL	00000011
DL	01101100
CF	0

۲-۳-٦- دستورالعمل ROR

شکل کلی دستورالعمل ROR بصورت زیر می‌باشد:

ROR opr , cnt

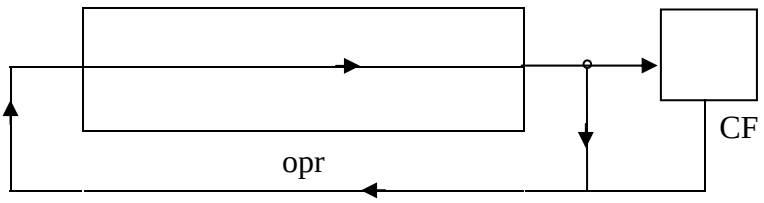
الف) opr از نوع بایت یا word می‌باشد.

ب) opr می‌بایستی از نوع متغیر یا ثبات باشد. opr ثابت نمی‌تواند باشد.

ج) cnt اگر معادل یک باشد عدد 1 را می‌نویسم در غیر اینصورت از ثبات CL استفاده می‌نمائیم.

د) روی فلگ CF اثر دارد.

هـ) باندازهٔ cnt بیت بطرف راست چرخش می‌دهد.



مثال ۲۰-٦

```
ROR   DL, 1
ROR   BX, CL
ROR   AL, CL
ROR   X[DI],CL
```

مثال ۲۱-٦

```
MOV   CL, 3
STC
MOV   DL, 8DH
ROR   DL, CL
```

مقادیر قبل از اجرای دستورالعمل ROR

CL	00000011
DL	10001101
CF	1

مقادیر بعد از اجرای دستورالعمل ROR

CL	00000011
DL	10110001
CF	1

۳-۳-٦- دستورالعمل RCL

شکل کلی دستورالعمل RCL بصورت زیر می‌باشد.

```
RCL   opr , cnt
```

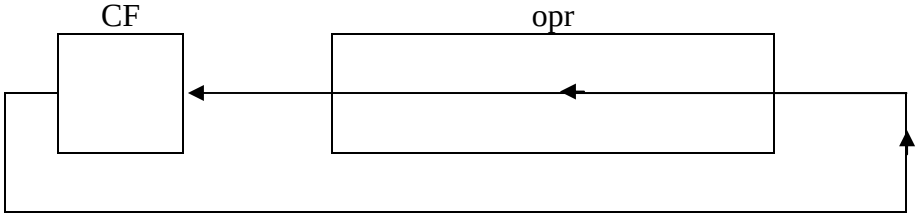
الف) چنانچه مقدار cnt یک باشد مقدار 1 نوشته می‌شود در غیر اینصورت از ثبات CL استفاده می‌گردد.

ب) opr بایستی از نوع بایت یا word باشد.

ج) opr بایستی متغیر یا ثبات باشد. opr مقدار ثابت نمی‌تواند باشد.

د) روی فلگ CF اثر دارد.

هـ) دستورالعمل RCL باندازه cnt بیت از بیت‌های opr را از طرف چپ و از طریق بیت CF چرخش می‌دهد.



مثال ۲۲-٦

```
RCL   DL, 1
RCL   BX, CL
RCL   AL, CL
RCL   BL, 1
RCL   [BX],CL
```

مثال ۲۳-٦

```
MOV   CL, 3
MOV   DL, 8DH
 STC
RCL   DL, CL
```

مقادیر قبل از اجرای دستورالعمل RCL

DL	10001101
CL	00000011
CF	1

مقادیر بعد از اجرای دستورالعمل RCL

CL	00000011
DL	01101110
CF	0

٤-٣-٦- دستورالعمل RCR

شکل کلی دستورالعمل RCR بصورت ذیل می‌باشد:

RCR opr , cnt

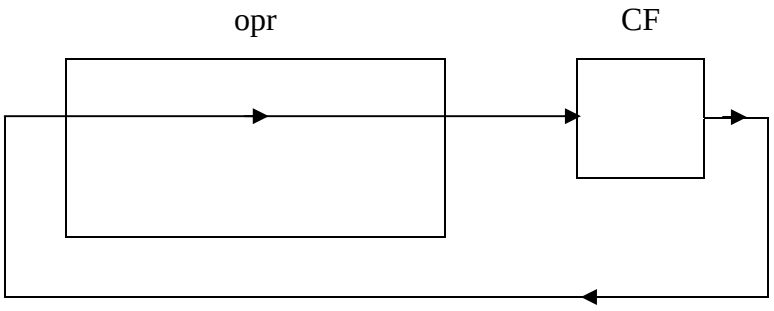
الف) opr بایستی از نوع word یا بایت باشد.

ب) opr بایستی ثبات یا متغیر باشد و ثابت نمی‌تواند باشد.

ج) اگر مقدار cnt برابر با یک باشد بایستی مقدار 1 را نوشت در غیر اینصورت از ثبات CL استفاده نمود.

د) روی فلگ CF اثر دارد.

هـ) دستورالعمل RCR باندازه cnt بیت بطرف راست از طریق فلگ CF چرخش می‌دهد.



مثال ٦-٢٤

```
RCR   DL, 1
RCR   [BX], CL
RCR   AL, CL
RCR   X[BX][DI],CL
RCR   [BX], 1
```

مثال ٦-٢٥

```
MOV   CL, 3
MOV   DL, 8DH
 STC
 RCR   DL, CL
```

مقادیر قبل از اجرای دستورالعمل RCR

CL	00000011
DL	10001101
CF	1

مقادیر بعد از اجرای دستورالعمل RCR

CL	00000011
DL	01110001
CF	1

٤-٦- عملیات فلگ‌ها

دستورالعملهای مربوط به عملیات روی فلگها در ذیـل داده شـده‌انـد. ایـن دستورالعملها عبارتند از:

جدول ٤-٦

Clear Carry	CLC	CF صفر می‌شود
Complement Carry	CMC	CF مکمل می‌شود
Set Carry	STC	CF یک می‌شود
Clear Direction	CLD	DF صفر می‌شود
Set Direction	STD	DF یک می‌شود
Clear Interrupt	CLI	IF صفر می‌شود
Set Interrupt	STI	IF یک می‌شود

همانطوریکه ملاحظه می‌شود این دستورالعملها فاقد عملوند می‌باشـند. ایـن دستورالعمل‌ها فقط روی فلگ مربوطه اثر دارنـد. بعنـوان مثـال CLC فقـط روی فلگ CF اثر می‌کند و مقدار قبلی آنرا تغییر می‌دهد.