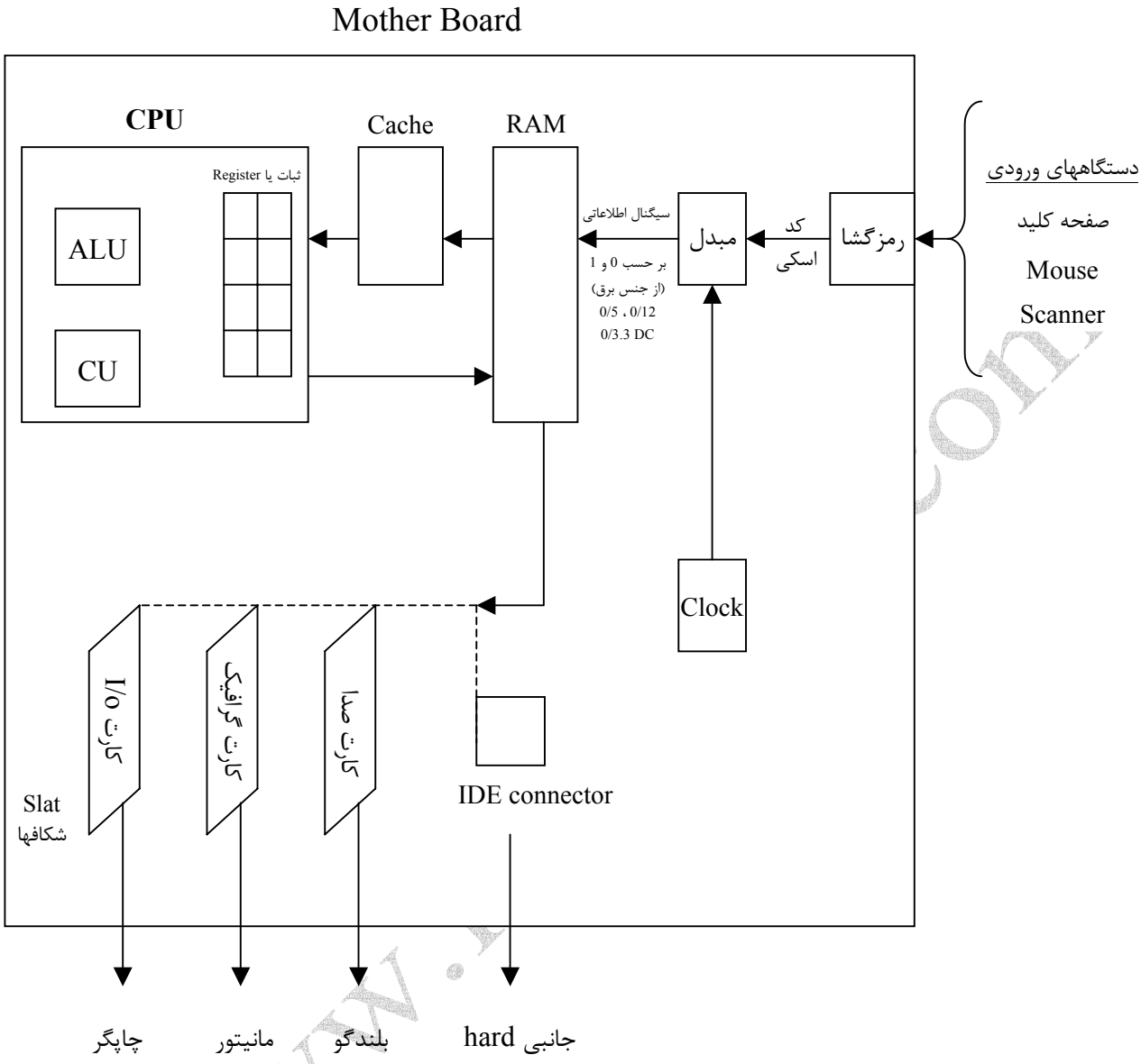
آشنایی با سیستم درونی و عملکرد داخلی کامپیوتر



تذکر ۱ : کلیه اطلاعات ورودی از طریق بردهای کامپیوتر به درون کامپیوتر (مادربرد) منتقل می‌شوند. ایـن اطلاعـات ورودی درون کامپیوتر تبدیل به سیگنالهای اطلاعاتی می‌شوند که بر حسب صفر و یک است و از جنس ولتاژ و برق می‌باشد.

تذکـر ۲ : تولید سیگنالهای اطلاعاتی به کمک قطعه‌ای به نام Clock یا ساعت زمانبندی و تایمینگ انجام می‌شوند.

تذکـر ۳ : حافظـه RAM یـک حافظه الکترونیکی است که از تعدادی سلول برای نگهداری اطلاعات استفاده می‌کند. هر سلول یک مدار فلیپ فلاپ می‌باشد.

تذکر ۴ : هر فلیپ فلاپ تنها می‌تواند مقدار صفر و یک را در خود نگهدارد.

تذکر ۵ : واحد نگهداری اطلاعات در هر سلول یا فلیپ فلاپ یک بیت است.

تذکر ۶ : اطلاعات ورودی از RAM به درون حافظه Cache می‌آید و سپس از طریق آن برای پردازش به CPU منتقل می‌شود.

تذکر ۷ : CPU پس از پردازش ، اطلاعات را برای انتقال به دستگاههای خروجی ، به حافظه RAM بازمی‌گرداند.

تذکر ۸ : حافظه RAM اطلاعات خروجی را به کارتهای واسط که درون Slatها قرار دارند می‌فرستد.

سؤال : Bus چیست؟

جواب : به کلیه مسیرهای ارتباطی بین قطعات درون کامپیوتر Bus می‌گویند.

تذکر ۱ : در کامپیوتر ۳ نوع Bus وجود دارد :

۱- **Address Bus :** نشانی و آدرس محلهای مختلف منتقل می‌شوند.

۲- **Control Bus :** از طریق این مسیر ، دستورات و فرامین منتقل می‌شوند.

۳- **Data Bus :** کلیه داده‌ها و اطلاعات از طریق آن منتقل می‌شوند.

تذکر ۲ : مسیرهای Bus مسیرهای دوطرفه غیر همزمان می‌باشند.

تذکر ۳ : واحد سنجش قدرت Bus ، بیت می‌باشد. بدین معنی که چه تعداد بیت هر Bus می‌تواند به طور همزمان منتقل نماید.

تذکر ۴ : واحد کنترل وظیفه دارد که بگوید کدام قطعه چه موقع چه کاری را انجام دهد.

تذکر ۵ : در سیستم بدن انسان ، نخاع و مخچه نقش و رُل CU را اجرا می‌کنند.

تذکر ۶ : ثبّاتها ، حافظه‌های موقت و سریع می‌باشند که در محاسبات از آنها استفاده می‌کنیم.

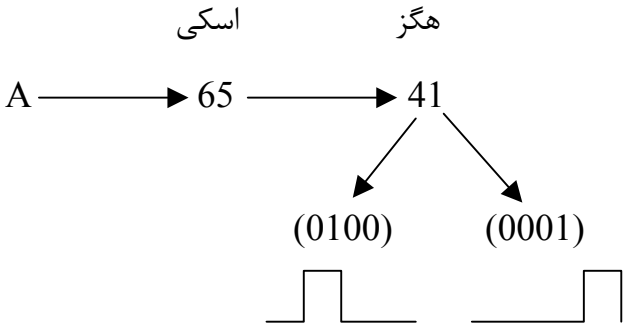
سؤال : Cache چیست؟

جواب : حافظه‌ای است سریع و گران قیمت که در کامپیوتر کاربرد فراوانی جهت سرعت بخشیدن دارد. این حافظه باعث می‌گردد حالت انتظار در CPU پائین بیاید و CPU با سرعت بیشتری دستورات و فرامین را اجرا نماید.

تذکر ۱ : کلیه اطلاعات که در کامپیوتر منتقل می‌شوند بصورت سیگنالهایی هستند که از مبنای 2 تولید شده‌اند. وزن این اعداد در مبنای 2 ، NBCD می‌باشد.

تذکر ۲ : کلیه اطلاعات درون کامپیوتر در مبنای 16 ذخیره‌سازی می‌شوند.

تذکر ۳ : شکل موج سیگنالهای درون کامپیوتر مربعی است.



آشنایی با تبدیل مبناها در کامپیوتر

$(13.25)_{10}$ ===➔ $(1101.01)_2$

$(1100.01)_2$ ===➔ $(12.25)_{10}$

$(25.25)_{10}$ ===➔ $(19.4)_{16}$

$(1E.5)_{16}$ ===➔ $(00011110.0101)_2$

مکمل‌ها

مکمـل‌هـا ابزاری می‌باشند برای نشان دادن مفهوم قرینه در کامپیوتر. در هر مبنا دو نوع مکمل وجود دارد. برای مثال در مبنای 2 دو نوع مکمل 1 و مکمل 2 وجود دارد.

- ✓ **مکمـل 1** : مکمـل واسـط می‌باشد که از تبدیل کردن صفرها به یک و یک‌ها به صفر بدست می‌آید.
- ✓ **مکمل 2** : این مکمل به طور واقعی نشاندهنده مفهوم در کامپیوتر می‌باشد که با اضافه کردن یک واحد به مکمل 1 بدست می‌آید.

آشنایی جزئی‌تر با داده‌های عددی و پردازش آنها

نکته ۱ : کلیه اعداد منفی در کامپیوتر را می‌توانیم بصورت مکمل 2 ذخیره‌سازی نمائیم.

تمرین : مقدار 23 - را در قالب 8 بیتی محاسبه نمائید.

0 0 0 1 0 1 1 1 ===➔ + 23

1 1 1 0 1 0 0 0 ===➔ مکمل یک

+ 1

1 1 1 0 1 0 0 1 ===➔ مکمل دو = - 23

تمرین : مقدار 65 - را در قالب 8 بیتی محاسبه نمائید.

0 1 0 0 0 0 0 1 ===➔ + 65
1 0 1 1 1 1 1 0 ===➔ مکمل یک

+ 1
1 0 1 1 1 1 1 1 ===➔ مکمل دو = - 65

نکـــته ۲ : در هـنگام تبدیل داده‌ها و اطلاعات ، هر بیت دارای ارزشی می‌باشد که شما باید این حالت ارزش را در نظر بگیرید.

نکته ۳ : در هنگام انجام محاسبات ، انتقال بیت از بیت سوم به بیت چهارم را Auxiliary می‌گویند در صورتی که انتقال بیت هشتم به بیت نهم را Cary می‌گویند.

Ax

Cary Auxiliary Cary = 0 1 1(1)

0 0 1 0 0 1 1 0 + 0 0 1 0 1 0 0 1 +
0 0 0 1 0 0 1 0 0 0 0 1 1 0 0 1
0 0 1 1 1 0 0 0 0 1 0 0 0 0 1 0

نیبل
یک بایت

Ax

Cary (1) 1(1) 1 1
1 0 0 1 0 0 1 1 +
1 0 0 1 1 1 1 0
1 0 0 1 1 0 0 0 1

تذکر : مقدار Auxiliary و مقدار Cary پس از انجام محاسبات در ثبّات Flag تنظیم و سِت (set) می‌گردد.

Flag Register ثبّات می‌باشد که به شما وضعیت پردازش را نشان می‌دهد.

نکـته ۴ : در برخـی از محاسبات ، حالت سرریز یا Over flow رخ می‌دهد که باعث می‌گردد ثبّات Flag بیت OF آن به یک یا صفر تغییر یابد.

Cary (1)

1 0 0 1 0 0 1 1 +	OF = 0	1 0 0 1 0 0 1 1 +
0 1 0 1 0 0 1 1	Cary = 0	1 0 1 1 1 0 1 0
1 1 1 0 0 1 1 0		(1) 0 1 0 0 1 1 0 1
		OF = 1
		Cary = 1

سرریز
(Over flow)

تذکـر : حالـت Over flow و Cary در برخی از موارد بطور همزمان رخ می‌دهد که شما می‌توانید برخـی از حالـتهای Cary را در نظـر نگـیرید و تـنها بصورت یک Warning (هشدار) باشد. همانند مثالهای فوق.

سؤال : بیت علامت چیست؟

جواب : به سمت چپ‌ترین بیت ، بیت علامت می‌گویند که نشان دهنده نوع عدد است.

(0) 0 0 1 0 1 1 1 ===➔ + 23

(1) 1 1 0 1 0 0 1 ===➔ - 23

سمت چپ‌ترین بیت را بیت علامت می‌گویند

- اگر بیت علامت صفر باشد عدد مثبت است
- اگر بیت علامت یک باشد عدد منفی است

نکـته ۵ : بـا توجـه بـه نوع محاسبه و داده‌های ورودی شما می‌توانید دستورات مورد نیاز را مشخص نمائـید. بـر خـلاف خیلـی از زبانهـا ، در زبان ماشین فرمان یا دستور در ابتدای سطر قرار می‌گیرد که می‌تواند به اندازه یک یا دو بایت باشد که در اصطلاح به آن OP - Code می‌گویند.

نکته ۶ : داده‌ها و محاسبات می‌توانند به صورت با علامت و یا بدون علامت باشند. اگر با علامت باشند سـمت چـپ‌تریـن بیـت به عنوان داده لحاظ نمی‌شود و چنانچه بدون علامت باشند ، سمت چپ‌ترین بیت را بیت داده‌ای فرض کرده و به آن ارزش می‌دهند.

نکــته ۷ : بـا توجه به داده‌هایی که در نظر می‌گیرد از لحاظ با علامت بودن یا بدون علامت بودن رِنج داده‌ای نیز تغییر می‌کند. همانند دو حالت زیر

128	64	32	16	8	4	2	1	
0	0	0	0	0	0	0	0	
1	1	1	1	1	1	1	1	
						- 127	+ 127	داده با علامت 8 بیت
						0	255	داده بدون علامت 8 بیت

نکــته ۸ : بـا توجـه بـه نـوع دسـتورات ، داده‌هـا تعییـن می‌شوند و بر عکس با توجه به نوع داده‌ها ، دستورات مربوطه را باید استفاده نمائیم.

ADD ◯ , ◯
AAD ◯ , ◯

نکــته ۹ : بـرای شناسایی این نکته که آیا داده عددی زوج است یا فرد باید به بیت سمت راست داده توجه نمائید. اگر این بیت صفر باشد داده زوج است و چنانچه یک باشد این داده فرد است.

نکته ۱۰ : در مبنای 10 دو نوع مکمل وجود دارد :

الف) مکمل 9 : برای بدست آوردن مکمل 9 کافی است تمام ارقام را از عدد 9 کم کنید.

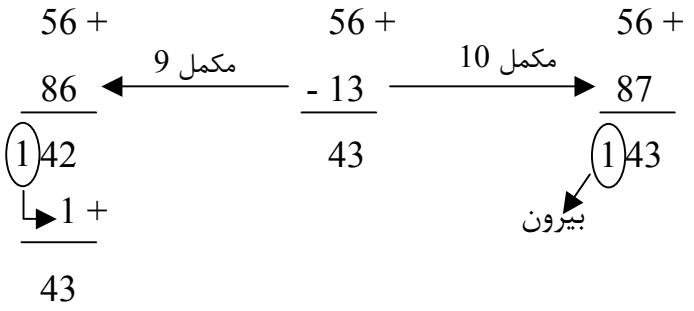
$(46)_{10} \xrightarrow{\text{مکمل 9}} 53$

ب) مکمل 10 : برای بدست آوردن مکمل 10 شما باید رقم اول را از عدد 10 کم کنید و الباقی ارقام را از عدد 9 کم کنید.

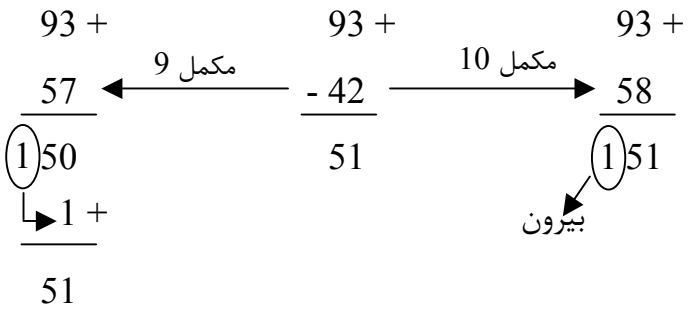
$(46)_{10} \xrightarrow{\text{مکمل 10}} 54$

نکته ۱۱ : در هنگام انجام محاسبات ، در نظر گرفتن مکمل 10 یا مکمل 9 کاملاً اختیاری است و شما می‌توانید بر حسب نیاز روش دلخواه خود را انتخاب نموده و به جواب درست برسید.

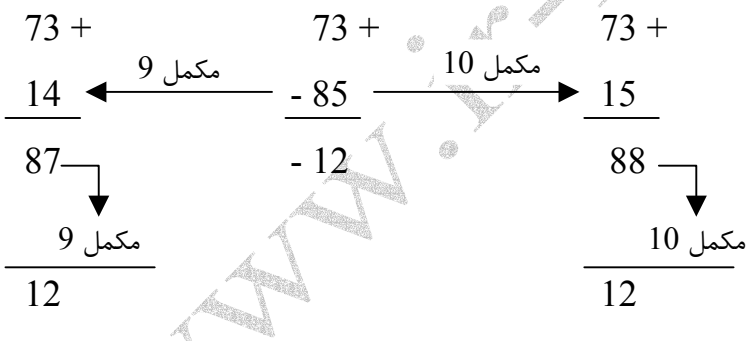
مثال :



تمرین :



تذکـــر : در دو مـثال بالا مؤلفه دوم از مؤلفه‌های اول کوچکتر می‌باشد. اگر مؤلفه دوم بزرگتر از مؤلفه اول باشد باید طبق الگوی زیر عمل نمائید.



نکته ۱۲ : در برخی از محاسبات ، رویداد Over flow و Cary دارای اهمیت می‌باشد و در برخی از محاسبات ، دارای اهمیت نمی‌باشد. مثال زیر نشان می‌دهد که حالت Cary flag دارای ارزش نیست و شما می‌توانید آنرا در نظر نگیرید زیرا دو داده جزء داده‌های با علامت می‌باشد.

65 → 0 1 0 0 0 0 0 1
- 65 → 1 0 1 1 1 1 1 1 با علامت
0 (1)0 0 0 0 0 0 0 0

CF = 1 OF = 1

تمرین : محاسبات زیر را انجام دهید و بگوئید کدام یک از flagها سِت می‌شود.

35 + ===➔ 0 0 1 0 0 0 1 1
24 ===➔ 0 0 0 1 1 0 0 0
0 0 1 1 1 0 1 1

CF = 0
OF = 0
Ax = 0

---

97 + ===➔ 0 1 1 0 0 0 0 1
59 ===➔ 0 0 1 1 1 0 1 1
1 0 0 1 1 1 0 0

CF = 0
OF = 0
Ax = 0

---

135 + ===➔ 1 0 0 0 0 1 1 1
- 24 ===➔ 1 1 1 0 1 0 0 0
1 0 1 1 0 1 1 1 1

CF = 1
OF = 1
Ax = 0

سؤال : بیت توازن چیست؟

جـواب : هـرگاه بخواهـیم در هـنگام انـتقال اطلاعات کنترل کنیم که داده‌ها و بیت‌ها درست منتقل شـده‌انـد یـا نـه از بیـت تـوازن استفاده می‌کنیم. شما می‌توانید بیت توازن را همانند یک روش فرض نمائـید. ایـن بیـت در بیـت flag قـرار دارد. در بیـت تـوازن بدینصورت مطرح می‌کنیم که اگر تعداد یـک‌هـای ارسالی فرد باشد بیت توازن یک است و چنانچه تعداد یک‌های ارسالی زوج باشد بیت توازن صفر است.

نکـته ۱۳ : در کامپیوترهای مدل 8086 و 8088 پهنای Bus (Data Bus) دیتا 16 بیت می‌باشد در صورتیکه پهنای Bus (Address Bus) آدرس 20 بیت می‌باشد.

$m = 2^n \longleftrightarrow \text{Log}_2^m = n$

تعداد خانه‌های حافظه (حالتها)

تعداد بیت‌های مورد نیاز بر حسب صفر و یک

$65536 = 2^{16}$

اندازه Segment ⟵ 64 KB

$1024 \times 1024 = 2^{20} = 1\ MB$

تذکر : ساختار Data Bus و Address Bus به صورت زیر می‌باشد :

	8088	8086
Data	8	16
Add	16	16
Data	16 (به همراه قرض)	16
Add	20 (۴ بیت قرضی)	20 (۴ بیت قرضی)

- **کامپیوترهای مدل 80286** : در کامپیوترهای 80286 می‌توان 16 میلیون بایت از حافظه را آدرس‌دهی نمود. بدین معنی که دارای 16 خط دیتا و 20 خط آدرس می‌باشد. بر خلاف کامپیوترهای مدل قبلی ، این کامپیوتر دارای 2 مُد آدرس‌دهی می‌باشد.

Real mode مُد واقعی

Protect mode مُد حفاظت شده

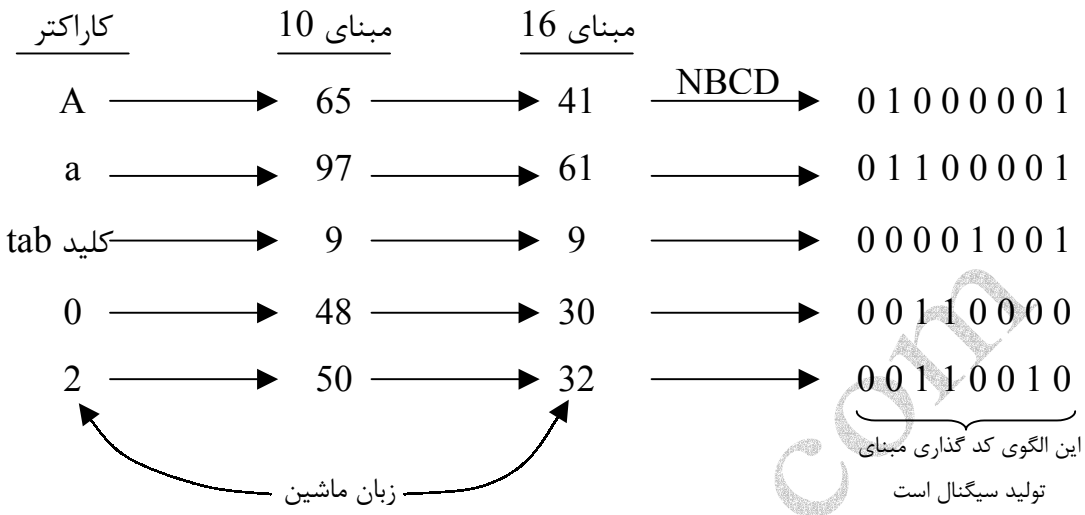
- **کامپیوترهای مدل 80386** : در کامپیوترهای مدل 80386 اندازه ثبات‌ها ، 32 بیتی می‌باشد و همچنین پهنای Bus آنها 32 بیتی می‌باشد. در این مدل از CPUها علاوه بر 2 مُد آدرس‌دهی قبلی ، حالت یا مُد مجازی نیز ارائه شده است.

Virtual mode مُد مجازی

- **کامپیوترهای مدل 80486** : این مدل از کامپیوترها دارای ثبات‌های 32 بیتی و پهنای Bus نیز 32 بیتی می‌باشد. در این کامپیوترها به اندازه 8 KB حافظه پنهان یا Cache نیز وجود دارد. این حافظه Cache بیرون CPU قرار گرفته است.
- **کامپیوترهای پنتیوم** : این مدل از CPUها دارای ثبات‌های 32 بیتی می‌باشد که پهنای Bus آنها 64 بیتی است و تمام حافظه پنهان درون CPUها قرار گرفته است.

تذکر : علت تنوع مدل پنتیوم در Pentium II و Pentium III مستقل بودن گذرگاهها می‌باشد و همچنین قطعه‌ای است که محاسبات اعداد اعشاری را انجام می‌دهد.

آشنایی بیشتر و دقیق‌تر با کدهای اسکی



در هنگام انجام فعالیت به روی داده‌های عددی ، رعایت کردن نکته‌ای که در مثال زیر بیان شده است مهم می‌باشد زیرا برای بدست آوردن نتیجه درست باید عدد 30 را از کد خارج نمائیم.

2 + 3

B8 02 03

Add 2 , 3

3	2

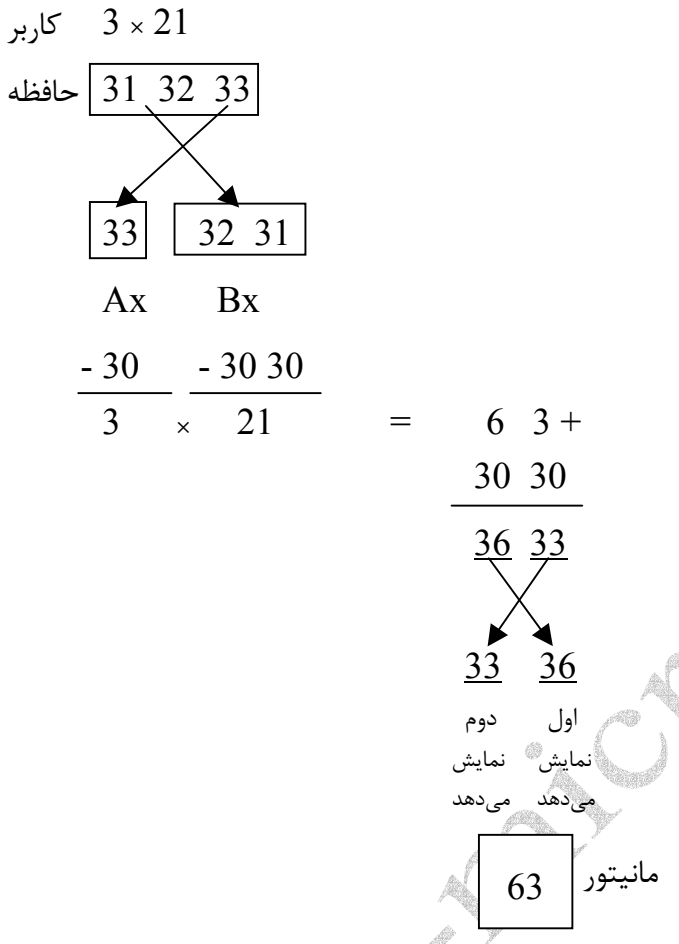
حافظه

33	32

32	33
Ax	Bx
- 30	- 30
2	3

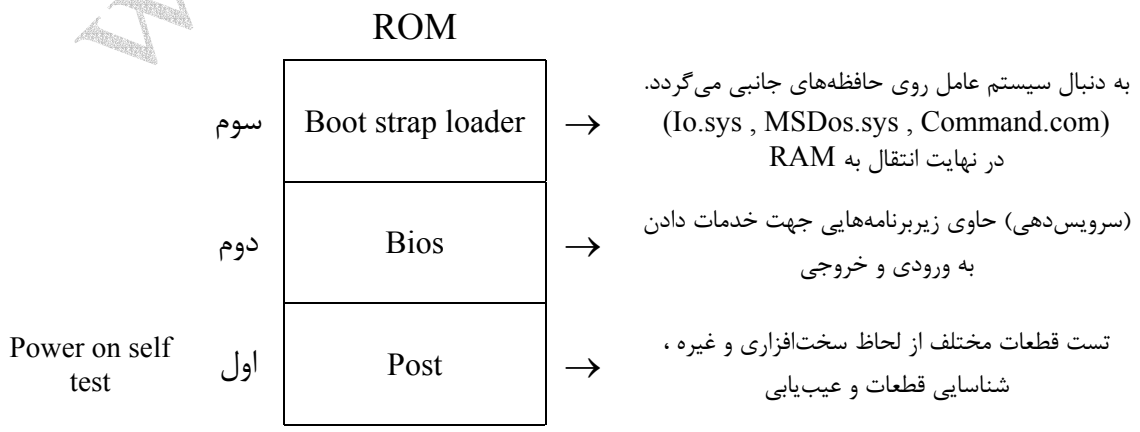
===➔ 5 + 30 = 35 ⟶ نشاندهنده کاراکتر 5 می‌باشد

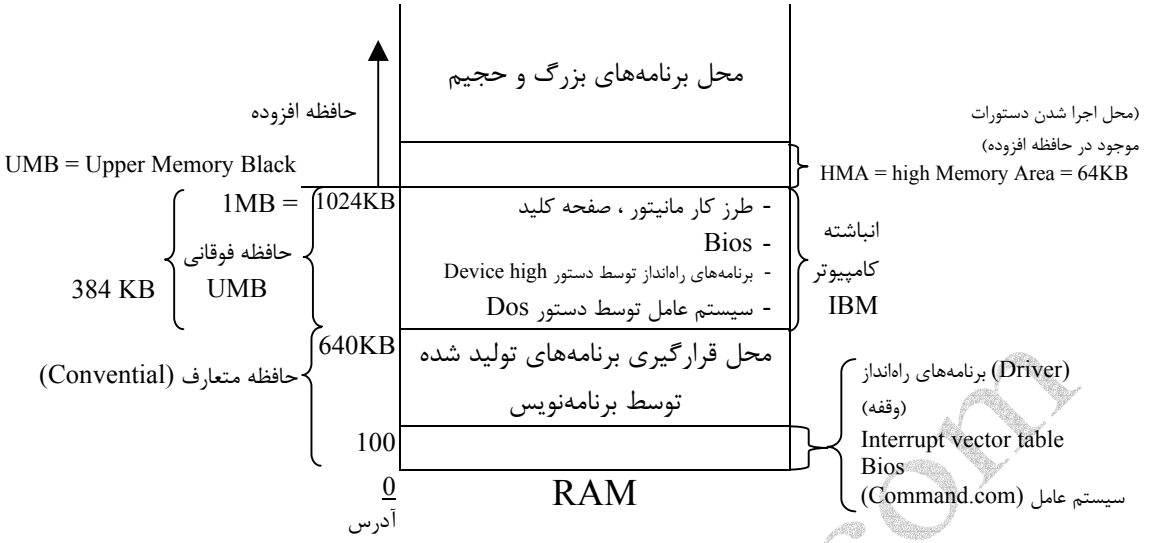
تمرین : فرآیند انجام محاسبه 21×3 را نشان دهید.



اطلاعـات ورودی و خروجـی کاراکتر است اما درون کامپیوتر بصورت کد اسکی در مبنای 16 می‌باشد که به شکل سیگنالهای اطلاعاتی در مبنای 2 و با وزن NBCD تولید می‌شوند.

آشنایی با ساختار حافظه RAM





تذکـــر ۱: سیســـتم عـــامل پـــس از بـــارگذاری در حافظـــه RAM ، دو فـــایل Config.sys و Autoexec.Bat را اجرا می‌کند و تمام دستوراتی که در این فایل وجود دارد را اجرا می‌کند. دو تا از مهمترین دستورات در فایل Config.sys عبارتند از :

- ✓ himem.sys
- ✓ EMM386.EXE

تذکر ۲: کلیه برنامه‌هایی که در محیط IBM قرار است اجرا شوند باید در حافظه متعارف قرار گیرند و یا اینکه بصورت حافظه متعارف شبیه‌سازی گردند.

نکات مهم پیرامون حافظه RAM

۱- حافظـه RAM یک برد الکترونیکی می‌باشد که با اجرا شدن برنامه سیستم عامل بخشهای گفته شده بالا بوجود می‌آید.

۲- بخشـهای مـتعارف ، فوقانـی و افـزوده ، بخشـهای منطقـی و نرم‌افزاری می‌باشند و با وجود سیستم عامل ایجاد می‌گردند.

۳- این تقسیم‌بندی منطقی تنها در حافظه RAM کامپیوترهای IBM وجود دارد.

۴- در سیسـتم‌هـای IBM سـه مُد آدرس‌دهی وجود دارد که عبارتند از : Real ، Protect و Virtual

۵- اگـر بـرنامه شـما در حافظه متعارف قرار داشته باشد و محدوده آدرس‌دهی شما بین 0 و تا

640 کیلو بایت باشد آنرا آدرس‌دهی واقعی یا Real می‌گویند.

۶- اگر بخشی از برنامه شما در حافظه افزوده قرار داشته باشد و شما مجبور باشید برای آدرس‌دهی از فضای بالای 1 MB استفاده نمائید این حالت را آدرس‌دهی Protect می‌گویند.

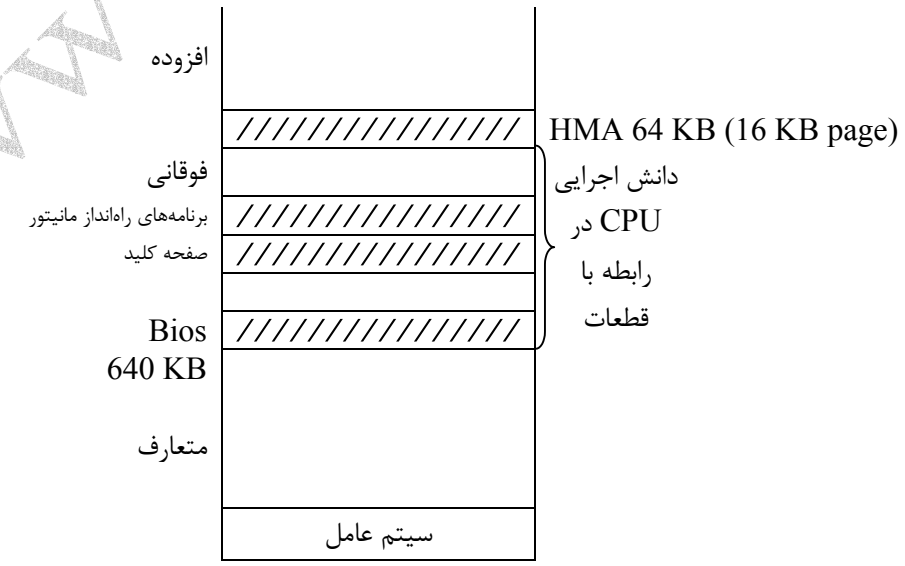
۷- اگر حالت آدرس‌دهی شما جدا از دو مورد گفته شده و در فضاهای دیگری باشد آنرا مجازی یا Virtual می‌نامند.

۸- پس از بارگذاری سیستم عامل در حافظه RAM کنترل و ناظم سیستم ، سیستم عامل می‌باشد. به همین علت سیستم عامل را پل ارتباطی بین سخت‌افزار و نرم‌افزار می‌دانند.

۹- فایل Config.sys توسط کاربر ایجاد می‌شود و کاربر با استفاده از فرامین سیستم عامل پیکربندی‌های کامپیوتر را بر حسب نیاز خود انجام می‌دهد. در این فایل دستوراتی نظیر معرفی برنامه‌های راه‌انداز ، تعداد Buffer ، تعداد فایل‌های باز و همچنین فعال‌سازی حافظه‌های افزوده و فوقانی قرار دارد.

۱۰- دو تا از مهمترین دستوراتی که در فایل Config.sys وجود دارد دو دستور himem.sys و EMM386.exe می‌باشد. این دو دستور برای مدیریت فایل‌ها و حافظه RAM استفاده می‌گردد.

۱۱- هرگاه دو فرمان himem.sys و EMM386.exe را اجرا می‌کنیم شما توانسته‌اید از فضاهای خالی حافظه فوقانی و حافظه افزوده بهره ببرید. با این عمل فضاهای گفته شده مدیریتش در اختیار شما قرار می‌گیرد و CPU می‌تواند به دستورات درون این فضاها دسترسی داشته باشد.



۱۲- پس از Config.sys ، فایل Autoexec.bat اجرا می‌شود. این فایل نیز توسط کاربر ایجاد می‌شود و حاوی دستوراتی است که قرار است سیستم عامل در ابتدای روشن شدن کامپیوتر اجرا نماید.

۱۳- درون فایل Autoexec.bat فرامین اجرایی سیستم عامل قرار دارد. دستوراتی نظیر تعیین مسیر دستورات ، تعیین علامت اعلان Dos و غیره

تذکر : شما می‌توانید از بخشهای مختلف حافظه RAM استفاده نمائید. برای انجام این منظور شما باید بتوانید به آدرس فیزیکی و واقعی آن محل دسترسی داشته باشید.

۱۴- کلیه برنامه‌هایی که در حافظه افزوده قرار دارند برای اجرا شدن به محیط شبیه‌سازی شده HMA می‌آیند در صورتیکه هسته اصلی برنامه‌ها در قسمت متعارف قرار می‌گیرند.

سؤال : برنامه راه‌انداز چیست؟

جواب : کلیه برنامه‌هایی هستند که وظیفه دارند تبادل اطلاعات بین کامپیوتر و دستگاه مربوطه را فراهم نمایند. به عبارتی ساده‌تر برنامه‌های راه‌انداز از طرز کار و روش تبادل اطلاعات بین کامپیوتر و دستگاه مربوطه را درون خود دارند.

۱۵- هرگاه برنامه‌های راه‌انداز توسط دستور Device اجرا می‌گردند این برنامه‌ها در حافظه متعارف قرار می‌گیرند و چنانچه دستور Device high اجرا شود در حافظه فوقانی قرار می‌گیرد.

۱۶- در کامپیوترهای قدیمی‌تر ، حافظه‌های توسعه یافته نیز وجود داشته است. این حافظه‌ها بصورت اختیاری بوده‌اند که برنامه‌ها جهت اجرا شدن می‌توانستند از این فضا استفاده نمایند.

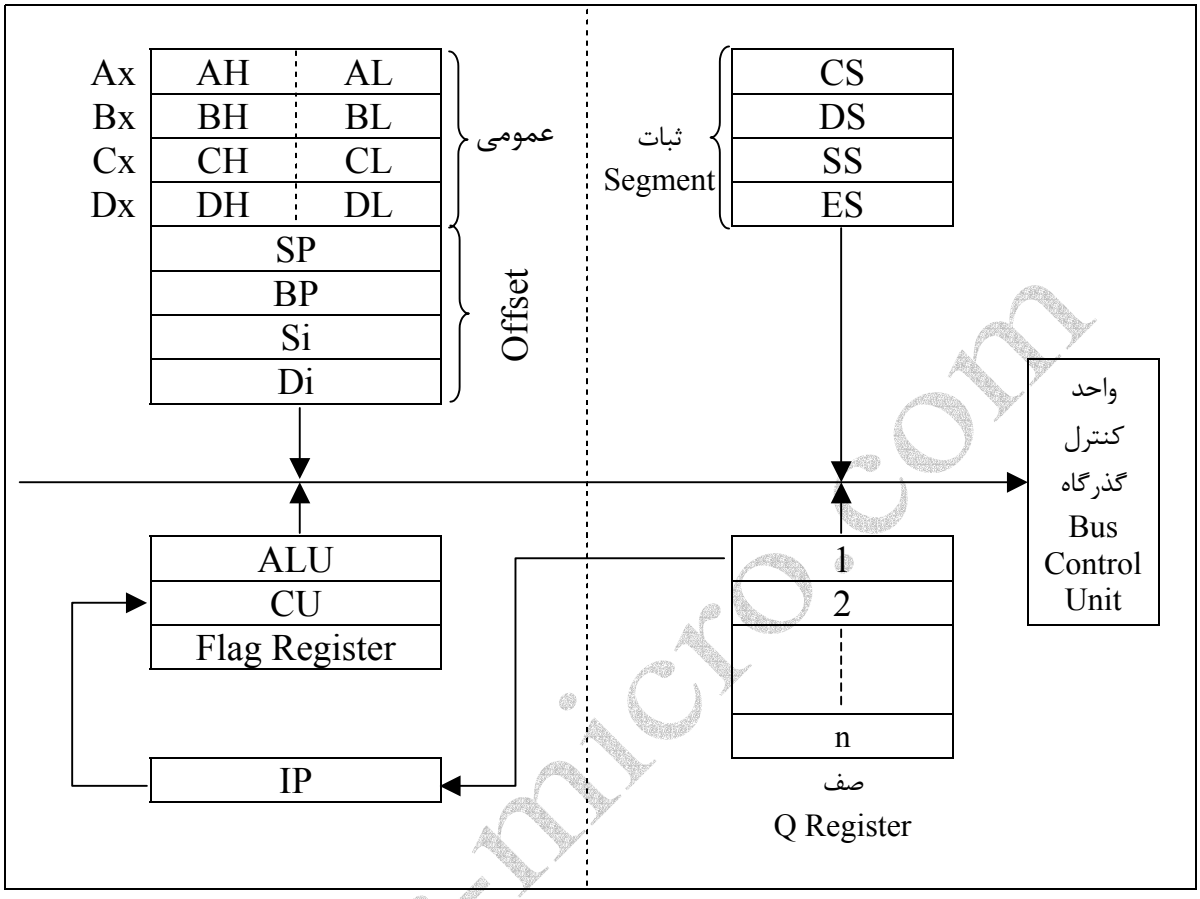
مثال مهم در مورد شناسایی سه بیت Ax و OF و CF

+ 13 + 0 0 0 0 1 1 0 1 + + 9 0 0 0 0 1 0 0 1 + 22 0 0 0 1 0 1 1 0 → + 22 بیت علامت نشاندهنده عدد مثبت Ax = 1 CF = 0 OF = 0	- 13 + 1 1 1 1 0 0 1 1 + + 9 0 0 0 0 1 0 0 1 - 4 1 1 1 1 1 1 1 0 → کاربرفهم - 4 (مکمل ۲) جواب بیت علامت نشاندهنده عدد منفی Ax = 0 CF = 0 OF = 0
+ 13 + 0 0 0 0 1 1 0 1 + - 9 1 1 1 1 0 1 1 1 + 4 1 0 0 0 0 0 1 0 0 → + 4 بیت علامت نشاندهنده عدد مثبت CF = Invalid حالت CF را در نظر نگیرید Ax = 1 CF = 1 OF = 1	- 13 + 1 1 1 1 0 0 1 1 + - 9 1 1 1 1 0 1 1 1 - 22 1 1 1 0 1 0 1 0 → کاربرفهم - 22 (مکمل ۲) جواب بیت علامت نشاندهنده عدد منفی CF = Valid حالت CF را در نظر بگیرید Ax = 0 CF = 1 OF = 1

تمرین : محاسبات فوق را بر روی عدد 23 و 13 اجرا کنید.

+ 23 + 0 0 0 1 0 1 1 1 + + 13 0 0 0 0 1 1 0 1 + 36 0 0 1 0 0 1 0 0 → + 36 بیت علامت نشاندهنده عدد مثبت Ax = 1 CF = 0 OF = 0	- 23 + 1 1 1 0 1 0 0 1 + + 13 0 0 0 0 1 1 0 1 - 10 1 1 1 0 1 1 0 → کاربرفهم - 10 (مکمل ۲) جواب بیت علامت نشاندهنده عدد منفی Ax = 1 CF = 0 OF = 0
+ 23 + 0 0 0 1 0 1 1 1 + - 13 1 1 1 1 0 0 1 1 + 10 1 0 0 0 1 0 1 0 → + 10 بیت علامت نشاندهنده عدد مثبت CF = Invalid حالت CF را در نظر نگیرید Ax = 1 CF = 1 OF = 1	- 23 + 1 1 1 0 1 0 0 1 + - 13 1 1 1 1 0 0 1 1 - 36 1 1 0 1 1 1 0 0 → کاربرفهم - 36 (مکمل ۲) جواب بیت علامت نشاندهنده عدد منفی CF = Valid حالت CF را در نظر بگیرید Ax = 0 CF = 1 OF = 1

آشنایی جزیی و دقیق‌تر با CPU



CPU از لحـاظ منطقی و از دید برنامه‌نویسی به دو منطقه تقسیم می‌شود. این تقسیم‌بندی بر اساس شـرح وظایف قطعات داخل CPU می‌باشد. بدین منظور که ثبات‌های اجرایی و قطعات اجرایی که در محاسـبات و اجـرای دستورات دخالت دارند را بخش اجرایی می‌نامند و قطعاتی که در فعالیت اجرایی شرکت نمی‌کنند را قسمت ارتباطی می‌نامند.

تذکر : در شمای بالا قطعات و ثبات‌هایی را مشاهده می‌کنید که در زبان ماشین و زبان اسمبلی با آنها سر و کار داریم.

سؤال : از دید زبان ماشین ، CPU دارای چه قسمتهایی است؟

الـف) واحـد اجرایـی Execute Unit (EU) : این واحد حاوی قطعات و ثبات‌هایی می‌باشد که وظـیفه دارد دسـتورات و فرامیـن را اجرا نماید. این بخش حاوی ثبات‌های عمومی ، ثبات‌های افست ، ALU ، CU ، IP و ثبات Flag می‌باشد.

ب) واحد گذرگاه (مسیر) ارتباطی Bus Interface Unit (BIU) : این بخش حاوی ثبات‌ها و قطعاتی است که تنها ارتباط دهنده می‌باشند و همچنین تنها اطلاعات را درون خود نگهداری می‌کنند.

تذکر ۱ : ثبات‌های دیگری نیز در مدلهای جدیدتر CPU وجود دارد که هر کدام فعالیت خاصی را بر عهده دارند.

تذکر ۲ : ALU تنها می‌تواند محاسبات عددی ، منطقی و شیفت دادن‌ها را اجرا نماید.

سؤال : ثبات چیست؟

جواب : حافظه‌ای است موقتی ، سریع و گران قیمت که اندازه آن می‌تواند ۲ بایت یا ۴ بایت باشد و در مدلهای قدیمی‌تر ۱ بایت بوده است.

تذکر ۱ : ثبات‌ها هر کدام دارای اسمی منحصر به فرد می‌باشند و توسط این اسم شناسایی می‌شوند.

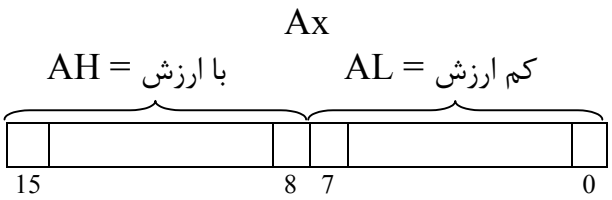
تذکر ۲ : با قطع برق ، اطلاعات درون ثبات‌ها از بین می‌رود.

تذکر ۳ : داده درون ثبات می‌تواند کاراکتر و یا عدد باشد. برداشت از آن به دیدگاه شما بستگی دارد.

تذکر ۴ : تمام ثبات‌های عمومی از دو بخش تشکیل شده‌اند.

۱- بخش کم ارزش

۲- بخش با ارزش



تذکر ۱ : ثبات‌ها دارای دو کاربرد می‌باشند : کاربردهای عمومی و کاربردهای اختصاصی. برای مثال شما می‌توانید از Ax جهت قرار داد مقادیر و انجام محاسبات استفاده نمائید. در صورتیکه سیستم نیز از Ax استفاده‌هایی را می‌کند.

تذکر ۲ : ثبات‌ها چهار نوع می‌باشند.

۱- ثابت‌های عمومی : این ثبات‌ها شامل Ax , Bx , Cx و Dx می‌باشند.

۲- ثابت‌های افست : ثبات‌هایی نظیر Sp , Bp , Si , Di و IP می‌باشند.

۳- ثبات Flag : این ثبات وضعیت عملکرد CPU را به شما نشان می‌دهد.

۴- ثبات Segment : این ثبات‌ها عبارتند از CS , DS , SS و ES. این ثبات‌ها حاوی آدرس شروع Segmentها می‌باشند.

Segment چیست؟ بخشی از حافظه متعارف می‌باشد که یک فضای محدود و مرزبندی شده در دل یک پاراگراف می‌باشد. اندازه Segmentها 64 کیلو بایت می‌باشد که آدرس شروع آنها عددی

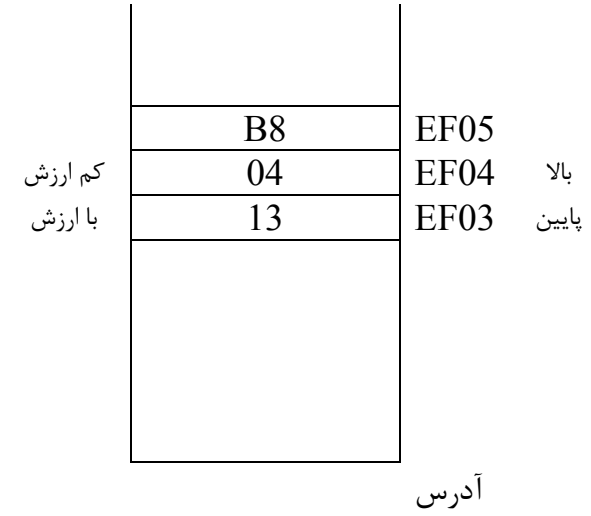
است که به عدد 16 (10 Hex) بخش‌پذیر است.

تذکر ۱: شروع هر Segment با مرز یک پاراگراف یکی می‌باشد.

تذکر ۲: درون هر ثبات Segment عددی وجود دارد که حاصل جمع این عدد با مقدار افست (برای مثال IP) به CPU محل واقعی و فیزیکی را نشان می‌دهد.

تذکـــر ۳: در حافظـه ، داده‌هـای بـا ارزش در آدرس‌هـای پاییـن قـرار دارند و داده‌های کم ارزش در آدرس‌های بالایی قرار دارند.

				کم ارزش	با ارزش
اسمبلی	mov	Ax	,	04	13
ماشین	B8	04		13	



تذکر ۱: سگمنت‌ها فضاهای کاری می‌باشند. بدین معنی که درون این فضاها کلیه دستورات ، فرامین و داده‌ها قرار داده می‌شوند.

تذکـــر ۲: برای برنامه‌نویسی به زبان ماشین باید ابتدا سگمنت‌هایی را از سیستم عامل تقاضا نمائید و سپس سیستم عامل از درون حافظه RAM (در بخش متعارف) فضایی را به ما اختصاص می‌دهد (که کجا بودن آن زیاد مهم نیست) و به اندازه 64 کیلو بایت و در مبنای 16 می‌باشد.

تذکر ۳: هر سگمنت دارای یک آدرس شروع می‌باشد.

انواع سگمنت

۱. **Code Segment:** ناحـیه‌ای اسـت کـه درون آن کلیه دستورات اجرایی برنامه و همچنین

کلیه فرامین در آن نوشته می‌شود (همان نقش Begin و End را در پاسکال بازی می‌کند).

۲. **Data Segment** : سگمنتی است که در آن کلیه ثابت‌ها ، متغیرها و داده‌ها تعریف می‌گردد.

تذکر : امکان دارد در برخی از برنامه‌ها Data Segment نداشته باشیم اما حداقل یک Code Segment داریم.

۳. **Stack Segment** : سگمنتی است که در آن کلیه آدرس‌های رفت و بازگشت زیربرنامه‌ها ذخیره می‌گردد و همچنین از آن برای انتقال اطلاعات به درون زیربرنامه نیز استفاده می‌گردد.

تذکر : از Stack Segment می‌توانیم جهت جابجایی داده‌ها و اطلاعات استفاده نمائیم.

۴. **Extra Segment** : این سگمنت اضافی برای انجام محاسبات بر روی رشته‌ها و آرایه‌ها استفاده می‌شود.

تذکر ۱ : همانطور که می‌دانید سگمنت‌ها دارای آدرس شروع می‌باشند و بطور تصادفی در یکی از قسمتهای حافظه متعارف قرار دارند.

تذکر ۲ : شما می‌توانید در یک برنامه ، یک یا چند سگمنت تقاضا کنید اما همیشه حداقل یک Code Segment دارید.

تذکر ۳ : شما می‌توانید در یک برنامه یک یا چند Code Segment داشته باشید و همینطور سگمنت‌های دیگر اما Stack Segment یکی است (فرض کنید این سگمنت به Stack Segment سیستم متصل باشد).

تذکر ۴ : برای اینکه اطلاعات را درون Stack Segment قرار دهیم از دستور Push استفاده می‌کنیم و همچنین برای اینکه اطلاعات را از درون Stack به بیرون انتقال دهیم از دستور pop استفاده می‌کنیم.

تذکر ۵ : آدرس شروع سگمنت‌ها 4 رقمی است و در مبنای 16 می‌باشد.

سؤال : افست (offset) چیست؟

جواب : به معنی تفاوت مکانی و یا انطباق می‌باشد. ثبات‌های افست تعیین مکان می‌کنند و همانند یک شمارنده حاوی آدرسی می‌باشند.

تذکر ۱ : آدرس سگمنت‌ها مطلق (و ثابت) می‌باشند. در صورتیکه هر سطر دستور درون سگمنت‌ها دارای یک آدرس می‌باشد (نسبی نسبت به سگمنت). این آدرس عددی است 4 رقمی در مبنای 16 که بین 0 0 0 0 و f f f f می‌باشد و همین رِنج در مبنای 10 بین 0 تا 65535 می‌باشد.

تذکـر ۲ : پـس از تقاضای سگمنت ما باید با دستور Assume آدرس شروع هر سگمنت را به درون ثبات سگمنت مربوطه‌اش بریزیم.

تذکر ۳ : باید آدرس شروع Code Segment را درون ثبات سگمنت CS بریزیم.

تذکر ۴ : باید آدرس شروع Data Segment را درون ثبات سگمنت DS بریزیم.

تذکر ۵ : باید آدرس شروع Stack Segment را درون ثبات سگمنت SS بریزیم.

تذکر ۶ : باید آدرس شروع Extra Segment را درون ثبات سگمنت ES بریزیم.

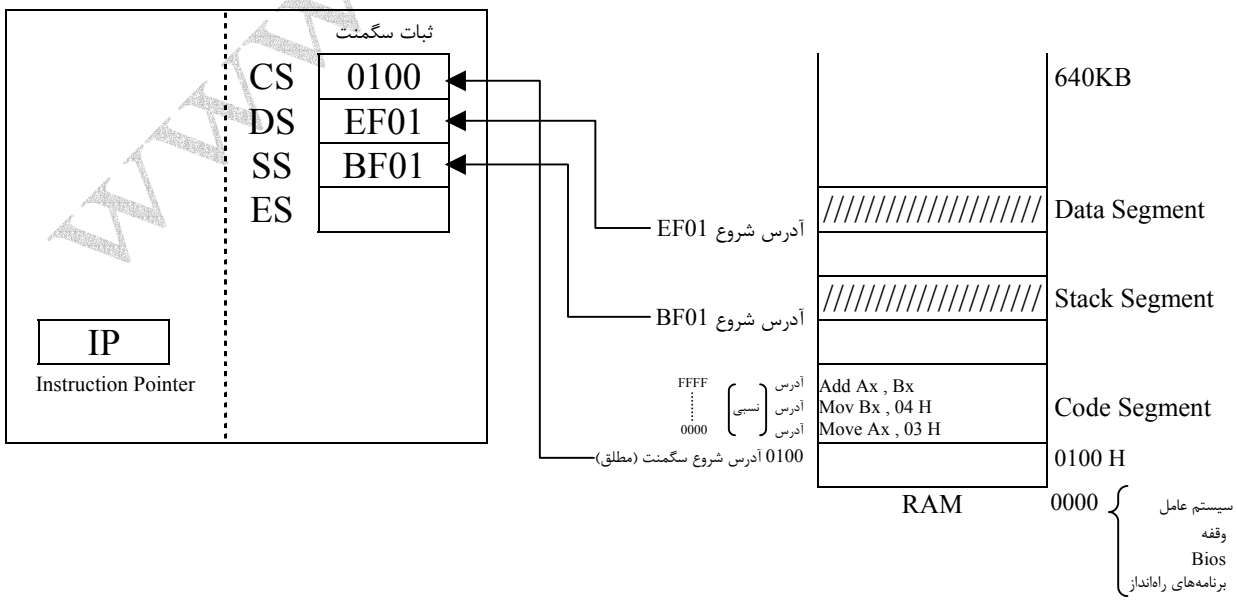
تذکـر ۷ : IP ثباتـی است که جزء ثبات‌های افست می‌باشد و همانند یک شمارنده ، آدرس نسبی هر دستور را نشان می‌دهد.

تذکر ۸ : برای بدست آوردن آدرس فیزیکی و واقعی هر دستور در حافظه RAM شما باید الگوی زیر را اجرا نمائید.

0 1 0 0 0 +	n n n n 0 +	0 آدرس شروع هر سگمنت +
0 0 0 1	m m m m	مقدار ثبات IP (نسبی)
آدرس فیزیکی اولین دستور	آدرس فیزیکی	آدرس فیزیکی هر دستو

تذکـر ۱ : هـر سگمنت دارای یک افست مخصوص بخود است و افست مربوط به Code Segment نیز افست IP می‌باشد.

تذکـر ۲ : علـت اضـافه کـردن یـک صـفر به جلوی آدرس سگمنت به خاطر بوجود آمدن فضای یک مگابایتـی حافظـه RAM مـی‌باشـد. چون با قرار دادن یک صفر ، تعداد ارقام در مبنای 2 بیست رقم خواهد شد.



تذکر ۱ : هـر بایت حافظه (درون سگمنت) دارای یک آدرس چهار رقمی می‌باشد. به عبارتی بایت‌های حافظه قابل آدرس‌دهی می‌باشد.

تذکـر ۲ : کلـیه دسـتورات زبان اسمبلی به کد ماشین ترجمه می‌شود و کدهای ماشین درون حافظه قرار می‌گیرند.

تذکر ۳ : هر کد ماشین از دو رقم در مبنای 16 تشکیل شده است.

تذکر ۴ : مقدار IP از صفر شروع می‌شود اما دو تا دو تا افزایش می‌یابد زیرا خواندن اطلاعات از درون حافظـه بصـورت دو بایتـی انجـام می‌شود (در سیستم‌های قدیمی‌تر دو تا یک بایتی و در سیستم‌های جدیدتر سه بایتی نیز می‌باشد).

0005	
0004	
0003	8B
0002	04
0001	B8
0000	03
0100 H	یک بایت داده

Segment
سیستم عامل

اسمبلی mov Ax , 03 H

کد ماشین $(\underbrace{B8}_{OP-Code}\ \underbrace{03}_{داده})_{16}$

اسمبلی mov Bx , 04 H

کد ماشین 8B 04

تذکر ۱ : مقدار IP دو تا دو تا افزایش می‌یابد.

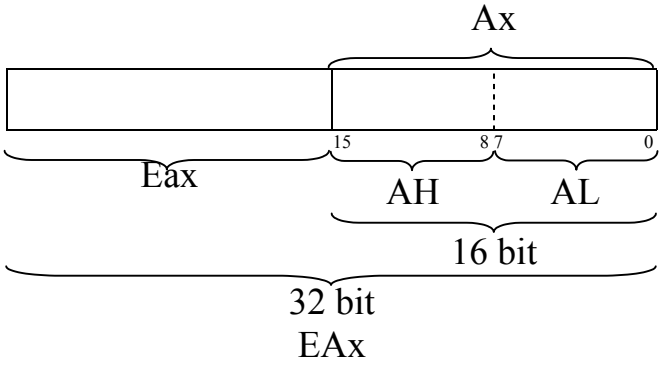
تذکر ۲ : اجرای دستورات در سیستم‌های قدیمی‌تر در دو Fetch انجام می‌گرفت. Fetch اول گرفتن دستور یا Op - Code و Fetch دوم دریافت داده‌ها و یا اطلاعات.

تذکـر ۳ : Fetch بـه معنی گرفتن اطلاعات از حافظه RAM و انتقال آن به درون ثبات‌های CPU است.

آشنایی با ثبات‌های درون CPU

۱- **ثـبات Ax :** ایـن ثـبات بـه مفهوم آکومولاتور (Accumulator) می‌باشد که در محاسبات شـرکت داده مـی‌شود و همچنین در آن محتوای نتایج محاسبات نیز ذخیره می‌گردد و از آن برای انتقال اطلاعات و داده‌ها به درون وقفه‌ها (Interrupt) استفاده می‌گردد.

تذکر : این ثبات 16 بیتی می‌باشد و در سیستم‌های جدید 32 بیتی است.



۲- **ثبات Base Register (Bx)** : این ثبات به عنوان ثبات پایه در آدرس‌دهی‌ها استفاده می‌گردد و همچنین برای انجام محاسبات از آن استفاده می‌گردد.

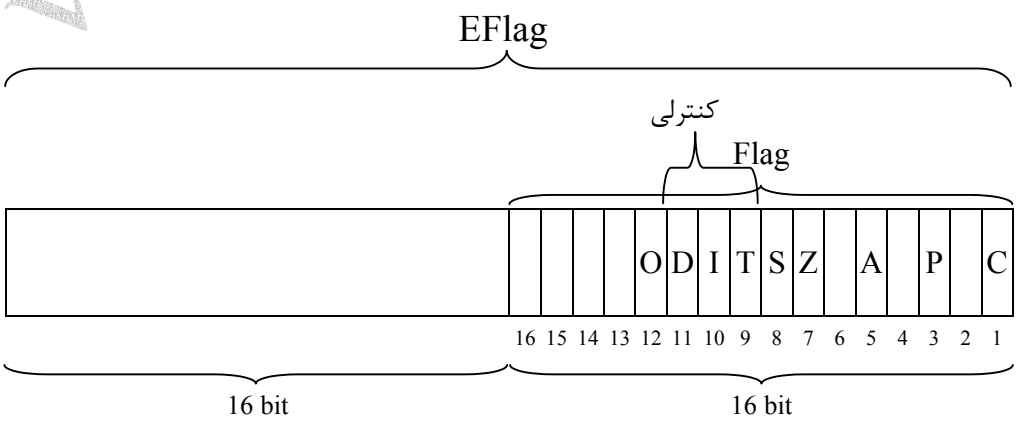
۳- **ثبات Counter Register (Cx) (ثبات شمارنده)** : این ثبات جهت انجام محاسبات استفاده می‌گردد و همچنین در ایجاد حلقه‌ها بعنوان متغیر شمارنده از آن استفاده می‌گردد.

۴- **ثبات Destination Register (Dx)** : این ثبات کاربردهای متنوع و گوناگونی دارد :

- برای انجام محاسبات
- بیت‌های اضافی که در Ax ذخیره نمی‌شوند در این ثبات قرار می‌گیرند.
- برای انجام محاسبات در رشته‌ها و آرایه‌ها

تذکر : تمام ثبات‌های عمومی گفته شده دارای ساختاری همانند Ax در کامپیوترهای جدید دارند.

۵. **ثبات Flag** : وضعیت پردازش را به شما نشان می‌دهد بدین معنی که CPU پس از پردازش وضعیت اجرای دستورات را در این ثبات به شما نشان می‌دهد. این ثبات در سیستم‌های جدید 32 بیتی می‌باشد. در صورتیکه در سیستم‌های قدیمی‌تر 16 بیت بوده است. 9 بیت آن فعال که 6 بیت آن مربوط به وضعیت پردازش دستورات است و 3 بیت آن جزء بیت‌های کنترلی است.



✓ **بیـت اول ← Cary Flag (CF)** : این بیت نشان دهنده رقم انتقالی در انجام محاسبات مـی‌باشـد. رقم انتقالی بیتی است که از بیت هفتم به هشتم منتقل شده است یا اینکه از بیت پانـزدهم بـه شـانزدهم منـتقل شـده اسـت (آخریـن بیـت نقلی می‌باشد). این بیت همچنین نشـاندهنده مقـدار شـیفت می‌باشد که دو وضعیت دارد. اگر رقم نقلی نباشد صفر است و اگر انتقال انجام شده باشد یک است که کد آنها بصورت زیر است

0 → NC

1 → PC

✓ **بیت سوم ← Parity Flag (PF)** : این بیت نشاندهنده بیت توازن می‌باشد. یعنی اینکه اطلاعـات درسـت منتقل شده‌اند یا خیر. این بیت همچنین نشاندهنده تعداد بیت‌های شیفت داده شـده مـی‌باشـد. اگـر تعـداد بیـت‌های شیفت داده شده زوج باشد مقدار آن یک است و چنانچه تعداد شیفت‌ها فرد باشد مقدار آن صفر است.

0 → PO (فرد)

1 → PE (زوج)

✓ **بیـت پنجم ← Auxiliary Flag (AF)** : این بیت نشاندهنده رقم نقلی از بیت سوم به بیت چهارم می‌باشد. اگر رقم نقلی نداشته باشید و یا اینکه تعداد شیفت‌ها زوج باشد مقدار آن صفر است در غیر اینصورت یک است.

0 → NA (زوج)

1 → PA (فرد)

✓ **بیـت هفتم ← Zero Flag (ZF)** : ایـن بیت به شما نشان می‌دهد که نتیجه محاسبات یـا نتـیجه مقایسـه برابـر صفر شده است یا خیر. اگر نتیجه محاسبات صفر باشد و یا اینکه دو داده با هم مساوی باشند مقدار این بیت یک است و در غیر اینصورت صفر است. مثال :

CMP Ax , Bx

ZF = 1 اگر محتوای Bx و Ax برابر باشد.

محاسبه { AL = 3 → 0 0 0 0 0 0 1 1 ; BL = 4 → 0 0 0 0 0 1 0 0 ; 0 0 0 0 0 1 1 1 } ➔ = مقدار نتیجه برابر صفر نیست ZF = 0

مقایسه { AL = 5 → 0 0 0 0 0 1 0 1 ; BL = 6 → 0 0 0 0 0 1 1 0 } ➔ = دو داده برابر نیستند ZF = 0

- ✓ **بیت هشتم ← Sign Flag (SF)** : این بیت نشان دهنده همان بیت علامت در نتیجه محاسبات می‌باشد. این بیت دو وضعیت دارد. اگر صفر باشد نشان دهنده مثبت بودن نتیجه محاسبات می‌باشد و چنانچه یک باشد نشان دهنده منفی بودن محاسبات می‌باشد.
- ✓ **بیت نهم ← Trace / Trap Flag (TF)** : با استفاده از این بیت ما می‌توانیم مشخص نمائیم اجرای دستورات بصورت تک مرحله‌ای انجام شود و یا اینکه بصورت یکی یکی و خط به خط اجرا شود. اگر مقدار این بیت یک باشد اجرای دستورات یکی یکی می‌باشد و چنانچه صفر باشد دستورات یکجا اجرا می‌شوند.
- ✓ **بیت دهم ← Interrupt Flag (IP)** : این بیت برای شما مشخص می‌کند که سیستم می‌تواند به وقفه‌ها پاسخ دهد یا خیر. اگر مقدار این بیت صفر باشد سیستم نمی‌تواند به وقفه‌ها پاسخ دهد و چنانچه یک باشد سیستم می‌تواند به وقفه‌ها پاسخ دهد.
- ✓ **بیت یازدهم ← Direction Flag (DF)** : با استفاده از این بیت ، جهت انجام عملیات‌ها بر روی رشته‌ها و آرایه‌ها را مشخص می‌کنیم. بدین معنی که با استفاده از این بیت اولین خانه آرایه را ما می‌توانیم مشخص نمائیم. اگر این بیت مقدار یک باشد جهت انجام عملیات‌ها (مقایسه یا شیفت) از سمت راست به چپ و چنانچه صفر باشد انجام عملیات‌ها از سمت چپ به راست می‌باشد.
- ✓ **بیت دوازدهم ← Over Flow Flag (OF)** : این بیت نشاندهنده مقدار سرریزی بیت‌ها می‌باشد که همان حالت Over Flow است. اگر مقدار سرریزی داشته باشید مقدار این بیت یک است و اگر نداشته باشید مقدار این بیت صفر است.

0 → NO

1 → PO

تمرین : با انجام محاسبه زیر اثر بیت‌ها در ثبات Flag را اعلام نمائید.

66 -	0 1 0 0 0 0 1 0 +	CF = 1
22	1 1 1 0 1 0 1 0	Ax = 0
44	1 0 0 1 0 1 1 0 0	OF = 1
		SF = 0
		ZF = 0

تمرین : نتیجه محاسبه زیر را در ثبات Flag نشان دهید و نتیجه را به 32 مقایسه کنید.

42 - 0 0 1 0 1 0 1 0 +

28 1 1 1 0 0 1 0 0

14 1 0 0 0 0 1 1 1 0

				O	D	I	T	S	Z		A		P		C
				1	C	C	C	0	0		0		0		1
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	غیر معتبر

مقایسه با 32

نتیجه 1 0 0 0 0 1 1 1 0

32 0 0 1 0 0 0 0 0 } برابر نیستند

				O				S	Z		A		P		C
				0	D	I	T	0	0		0		0		0
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

تمرین : بیت‌های Flag را پس از انجام محاسبه زیر نشان دهید.

Mov AL , 4 4 = 0 0 0 0 0 1 0 0

				O				S	Z		A		P		C
				0	D	I	T	0	0		0		0		0
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

۶. **ثبات Start Index (SI) :** این ثبات جزء ثبات‌های اندیس می‌باشد که دارای 16 بیت است و مدل جدیدتر آن ESI و دارای 32 بیت می‌باشد. از این ثبات برای آدرس‌دهی‌ها و همچنین اندیس آرایه‌ها استفاده می‌گردد. ***کاربرد خاص آن مبدأی است برای انجام عملیات بر روی رشته‌ها***

تذکر خیلی مهم : ثبات SI با ثبات سگمنت DS ترکیب می‌شود.

۷. **ثبات Destination Index (DI)** : این ثبات به عنوان ثبات اندیس می‌باشد که دارای 16بیت است و در کامپیوترهای جدید بصورت EDI و 32 بیتی می‌باشد. از این ثبات جهت انجام عملیات بر روی آرایه‌ها و رشته‌ها استفاده می‌گردد. کاربرد خاص آن بعنوان مقصد جهت انجام عملیات بر روی رشته‌ها می‌باشد.

تذکر : این ثبات نیز با ثبات سگمنت DS ترکیب می‌گردد.

سؤال : اجرای دستورات در زبان ماشین در چند مرحله اجرا می‌گردد؟ بدین مفهوم که اجرای دستورات در هر Fetch چگونه است؟

جواب : اجرای دستورات در سه مرحله اجرا می‌گردد :

- **مرحله اول** : دریافت Op - Code دستورات و یا داده‌ها و قرار دادن آن در صف اجرای دستورات
- **مرحله دوم** : رمزگشایی دستور و شناسایی دستور می‌باشد. این عمل توسط CU صورت می‌پذیرد.
- **مرحله سوم** : اجرای دستورات که توسط ALU انجام می‌گیرد.

سؤال : حافظه‌ها دارای چند خاصیت مشترک می‌باشند؟

جواب : تمام حافظه‌ها باید خاصیت‌های زیر را دارا باشند :

۱- قابل خواندن و نوشتن اطلاعات باشند.

۲- تمام حافظه‌ها باید دارای ظرفیت یا گنجایش باشند.

۳- تمام بایت‌های حافظه باید قابل آدرس‌پذیری و دست‌یابی باشند که در برخی از حافظه‌ها واحدهای بزرگتری قابل آدرس‌دهی می‌باشند.

۴- زمان دسترسی باید در حافظه‌ها وجود داشته باشد.

۵- در تمام حافظه‌ها باید سرعت انتقال وجود داشته باشد.

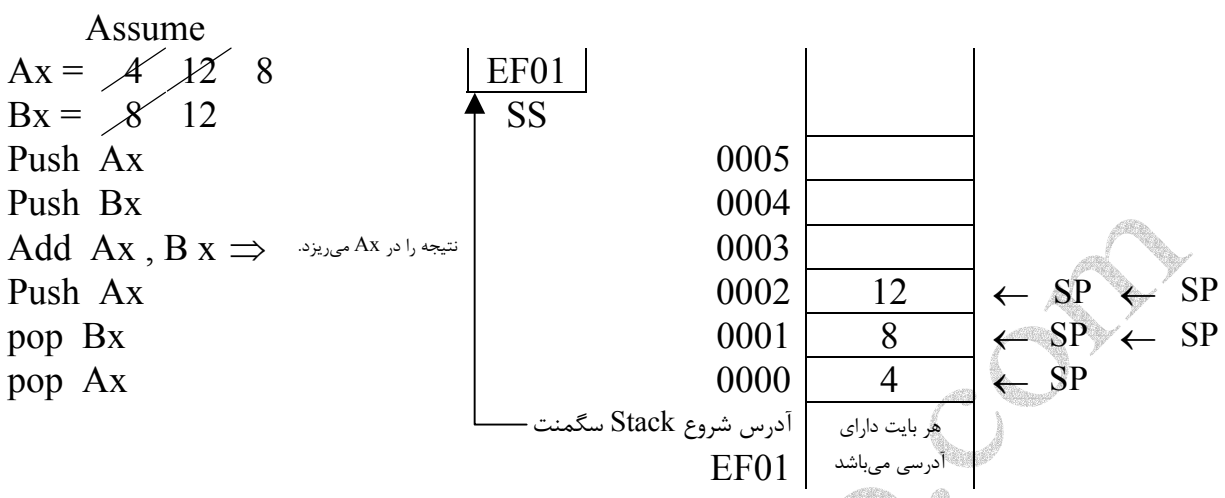
۶- همه حافظه‌ها باید بتوانند اطلاعات و داده‌ها را بطور پایدار در خود نگهداری نمایند.

آشنایی جزیی‌تر با Stack

Stack همانند یک حافظه عمل می‌کند. بطوریکه در آن می‌توانیم اطلاعات را به طور موقت ذخیره نمائیم. این سگمنت دارای یک آدرس شروع می‌باشد که برای بدست آوردن آدرس فیزیکی با مقدار SP ترکیب می‌گردد.

تذکر ۱ : تنها دستوری که اطلاعات و داده‌ها را به درون Stack می‌ریزد دستور Push می‌باشد و تنها دستوری که از Stack خارج می‌کند دستور pop می‌باشد.

تذکر ۲ : ثبات SP نشان دهنده محلی است که در آن اطلاعات می‌ریزیم و یا اطلاعات بر می‌داریم.



+ ثبات سگمنت	SS +	EF010 +	EF010 +
Offset	SP	0000	0001
آدرس فیزیکی		(آدرس واقعی) EF010	EF011

تذکر : دو ثبات سگمنت دیگر وجود دارد بنامهای GS و FS که این ثبات‌ها در موارد مورد نیاز استفاده می‌شوند و می‌توانید از آنها جهت ثبات‌های سگمنت Code و یا ثبات‌های اضافه استفاده نمائید.

۸. **ثبات Stack Pointer (SP) (ثبات اشاره‌گر پشته) :** این ثبات 16 بیتی می‌باشد که برای افست در Stack Segment استفاده می‌شود. این ثبات در کامپیوترهای جدید بصورت 32 بیتی می‌باشد که حرف اول آن E قرار می‌گیرد (ESP).

تذکر : این ثبات جزو ثبات‌های اشاره‌گر می‌باشد.

۹. **ثبات Base Pointer (BP) :** این ثبات 16 بیتی می‌باشد که برای انتقال اطلاعات به درون Stack استفاده می‌شود. داده‌هایی که به درون Stack منتقل می‌شوند می‌توانند داده و یا آدرس باشند.

تذکر ۱ : این ثبات به عنوان افست یا ثبات SS ترکیب می‌گردد.

تذکر ۲ : ثبات BP به عنوان ثبات پایه و اصلی می‌تواند با DI و SI ترکیب شود.

آشنایی با Interrupt ها (وقفه‌ها)

وقفه چیست؟

برنامه‌ای است که از چند دستورالعمل تشکیل شده است که یک هدفی را دنبال می‌کند. در واقع وقفه‌ها برنامه‌هایی حاضر و آماده می‌باشند که باید یک عمل خاصی را اجرا نمایند.

چند نوع وقفه وجود دارد؟

دو نوع وقفه وجود دارد :

۱- وقفه‌های سخت افزاری

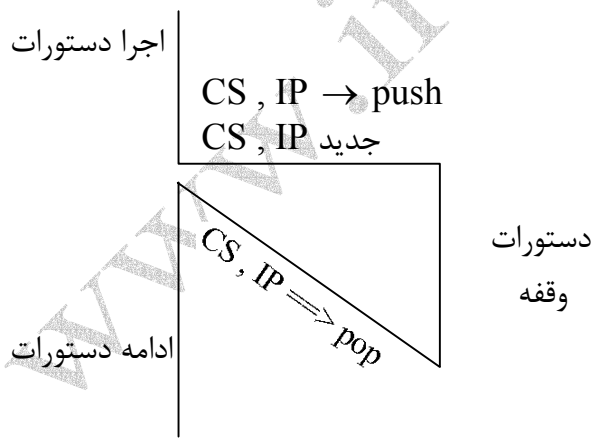
۲- وقفه‌های نرم افزاری

سؤال : وقفه‌های سخت افزاری چیست؟

جواب : برنامه‌هایی هستند که توسط CPU اجرا می‌گردند و اجرا شدن آنها منوط به دریافت سیگنالی توسط CPU می‌باشد.

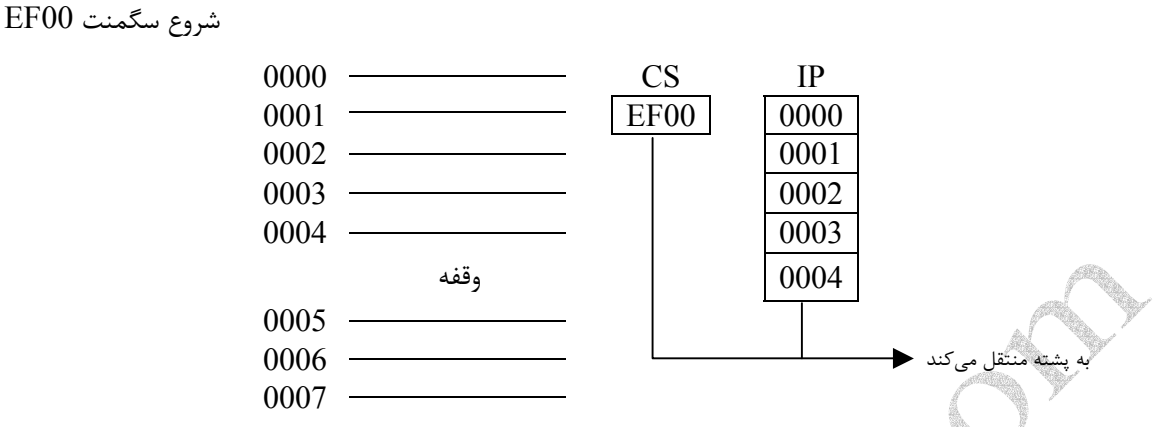
تذکر : پالس‌هایی که برای انجام وقفه‌ها می‌باشند از مسیر IRQ به CPU منتقل می‌شوند.

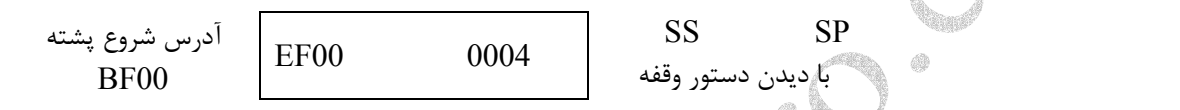
تذکر مهم : هرگاه وقفه‌ای را از CPU تقاضا می‌کنیم ، CPU اجرای برنامه‌های خود را متوقف کرده و دستورات درون وقفه را تا رسیدن به دستور IRET اجرا می‌کند و سپس به درون دستورات در حال اجرا باز می‌گردد. همانند الگوی زیر :

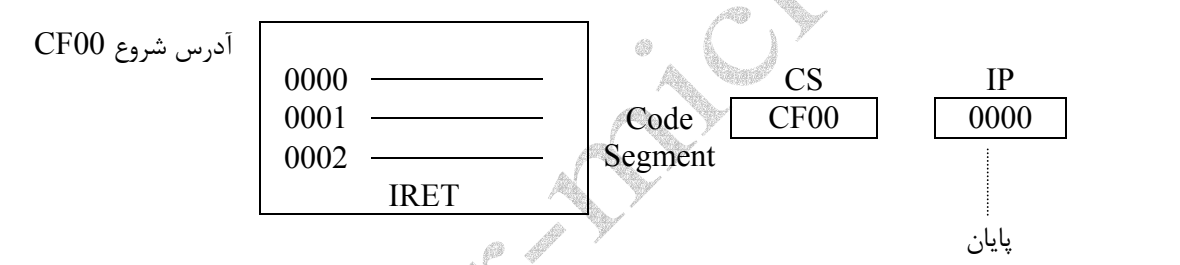


تذکر : هرگاه CPU و مجری (سیستم عامل) برنامه‌های زبان ماشین به دستور وقفه می‌رسند مقدار CS و IP را در Stack ذخیره نموده و محل وقفه را در حافظه RAM شناسایی می‌کند. چونکه متن برنامه‌های وقفه در یک سگمنت دیگری قرار دارد سیستم مترجم این آدرس‌های جدید را به ثبات‌های CS و IP منتقل می‌کند تا سیستم در مکان‌یابی اجرای دستورات دچار مشکل نشود. پس از اجرای دستورات درون وقفه و با رسیدن به دستور IRET دوباره سیستم مترجم مقادیر CS و IP که در

Stack ذخـیره شده است را برداشته و به درون دو ثبات CS و IP می‌ریزد تا اجرای دستورات برنامه اصلی ادامه یابد.







هـرگاه بـه دسـتور IRET رسید باید ابتدا محتوای درون Stack را به CS و IP منتقل کند تا بتواند ادامه برنامه را اجرا نماید.

دستور فراخوانی وقفه‌ها بصورت زیر می‌باشد.

Int شماره وقفه H

تذکر ۱ : بر خلاف هِدرِ فایل‌ها که دارای اسم هستند وقفه‌ها شماره‌بندی شده هستند.

تذکـر ۲ : بـه زیربـرنامه و امکاناتـی که درون هر وقفه وجود دارد سرویس می‌گویند و سرویس‌ها نیز شماره‌بندی شده هستند.

تذکـر ۳ : هـرگاه بـه وقفـه‌ای نیاز داشته باشیم (در بین دستورات) باید با استفاده از دستور int آنرا فراخوانی نمائیم.

تذکر ۴ : شماره سرویس را باید با دستور Mov درون ثبات AH قرار دهیم.

تذکـر ۵ : سـاختار و عملکـرد وقفه‌ها همانند توابع کتابخانه‌ای می‌باشند. یعنی اینک برنامه‌نویس باید

شـماره سـرویس ، اطلاعـات ورودی بـه درون سـرویس و عمل فراخوانی را انجام دهد و ***همه وقفه‌ها اطلاعات خروجی خود را درون ثبات‌ها می‌ریزند.***

int شماره وقفه H

Mov AH , 01 H مشخص کردن شماره سرویس

محل تعیین اطلاعات ورود به سرویس

int 21 H

اطلاعات خروجی را درون ثبات‌ها می‌ریزد.

مثال)

Scanf

Mov AH , 01 H

یک کاراکتر از صفحه کلید دریافت می‌کند و کد اسکی آنرا درون ثبات AL می‌ریزد

int 21 H

برای نمایش دادن کاراکتر و داده‌ها روی صفحه مانیتور

Mov AH , 02 H ← شماره سرویس

Mov DL , 3 ← اطلاعات ورودی به سرویس

int 21 H ← فراخوانی وقفه

پروژه اول : برنامه‌ای بنویسید که ۱۳ عدد را از ورودی بخواند (یک رقمی) و میانگین آنرا چاپ کند.

پـروژه دوم : مربع گرافیکی ترسیم شده را با زدن کلید Enter حرکت دهید و با زدن کلید Space متوقف کنید.

پروژه سوم : هفت کاراکتر را از ورودی بخوانید و آنرا مرتب چاپ کنید.