به نام خداوند بخشنده مهربان

میکروپروسسور ۸۰۸۶

استاد قرائی

تهیه کننده : حامد مظاهری

دانشگاه آزاد اسلامی - واحد تهران جنوب
دانشکده فنی

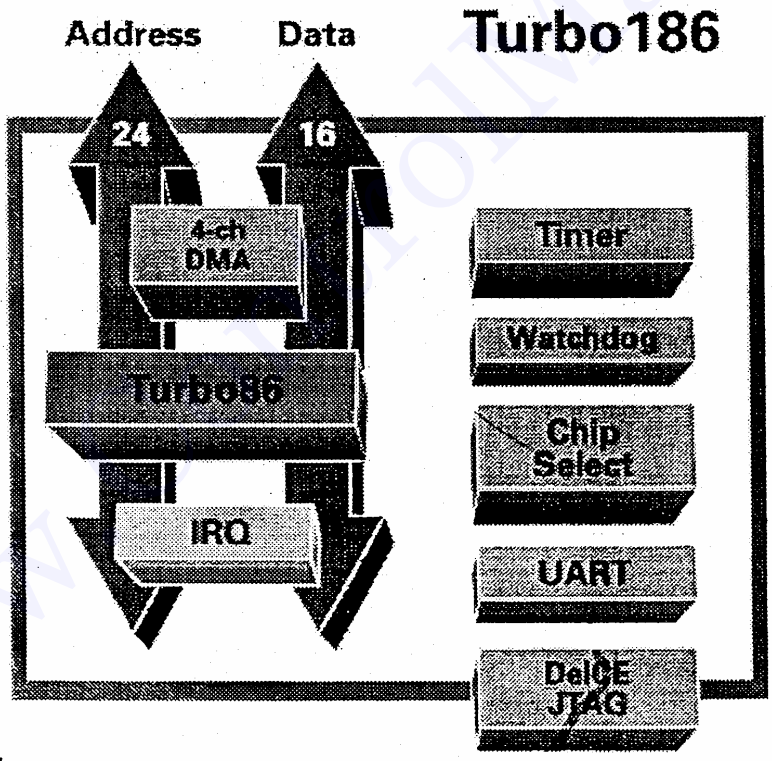
شما هم می توانید مقالات ، جزوات و مطالب خود را برای ما ارسال کنید
تا با نام خودتان و برای استفاده دیگر دوستان در سایت قرار داده شود .

>> Hamed.Mazaheri@Gmail.com <<

(بخش اول)

کتاب میکروپروسسور

8086



تهیه و تنظیم : مهندس محمد حسن قرائی

مقدمه

پیشرفت تکنولوژی الکترونیک در طی سه دهه گذشته چهره کامپیوترها را ابتدا سال به سال و سپس ماه به ماه، اخیراً روز به روز تغییر داده است.

اولین کامپیوتر بنام ENIAC معروف بوده و در سال ۱۹۴۶ میلادی بوسیله اکرت (EKKERT) و ماچلی (MAUCHLY) از مدرسه مهندسی برق دانشگاه پنسیلوانیا ساخته شد، که بیشتر شبیه یک ماشین حساب بود حتی یک ماشین حساب با کاربرد وسیع هم نبود بلکه بیشتر برای محاسبه جداول مربوط به پرتاب توپ از آن استفاده می کردند. در این ماشین از ۱۸۰۰ لامپ خلاء استفاده شده بود که ۴۰ قفسه بزرگ را پر می کرد. و این قفسه ها در یک اتاق ۱۳×۱۰ متر قرار گرفته بودند. همزمان با ENIAC شرکت IBM نیز اولین کامپیوتر خود را بنام IBM مدل603 به بازار عرضه کرد، که البته کامپیوتر الکترونیکی کوچکی بود. پس از آن کامپیوترهای بزرگتر IBM مثل 604 در سال ۱۹۴۸ (همان سال اختراع ترانزیستور) و IBM 605 در سال ۱۹۵۴ به بازار عرضه شد. عنصر اصلی در ساخت همگی این کامپیوترها همان لامپ خلاء بود و بنام کامپیوترهای نسل اول (FIRST GENERATION) معروف می باشند.

دومین نسل از کامپیوترها آنهایی بودند که از ترانزیستور بعنوان عنصر اصلی الکترونیکی استفاده کردند. مثل CDC6600, IBM7094, IBM7090 نیز از این دسته است. پیدایش تکنولوژی مدارهای مجتمع در اواخر سالهای ۶۰ و اوایل دهه ۶۰ دوره جدیدی را در صنعت کامپیوتر بوجود آورد که باعث تولید سومین نسل کامپیوتر شد. در نسل سوم از مدارهای مجتمع در مقیاس کوچک (SSI) (یک تا ۱۰ گیت در یک تراشه) و بعضاً مدارهای در مقیاس متوسط (MSI ده تا صد گیت در یک تراشه) استفاده می شد. و از سالهای ۱۹۶۶ به بعد کامپیوترهای نسل سوم به بازار عرضه شدند (IBM360-370) از جمله کامپیوترهای معروف این نسل هستند. تا قبل از پیدایش مدارهای مجتمع گرانی قیمت کامپیوترها باعث می شد که کامپیوترهای بزرگ برای چند محل ساخته شوند (Main Frame) و ترمینالهای آنها در اختیار استفاده کننده بصورت اجاره ای قرار می گرفت. با پیدایش و پیشرفت مدارهای مجتمع کامپیوترهای کوچکتری بنام (Mini Computer) ساخته شد که شرکتها و مؤسسات نسبتاً بزرگ مثل دانشگاهها و شرکت بیمه بتوانند خریداری کنند. از مینی کامپیوترهای معروف PDP-11 را می توان نام برد که در دهه ۱۹۷۰ به بازار عرضه شد. پیشرفت روزافزون تکنولوژی و پیدایش تکنولوژی MOS امکان ساخت مدارهای مجتمع با مقیاس بزرگ (LSI بیش از هزار دروازه در IC) را فراهم نمود و این تکنولوژی باعث

شد که بتوانند میکروپروسسور طراحی نمایند. یعنی یک مدار مجتمع قابل برنامه ریزی که با گرفتن دستور می تواند عملیات جمع تفریق و مقایسه و یا عملیات منطقی را انجام دهد. اولین ریزپردازنده که به بازار ارائه شد تراشه 4004 بود که در سال ۱۹۷۱ شرکت اینتل (INTEL) البته نه به عنوان ریزپردازنده بلکه بعنوان تراشه قابل برنامه ریزی برای ماشین حساب آن را ارائه نمود. این مسئله که تولد ریزپردازنده نیز به آن گفته اند در صنعت الکترونیک باعث انقلابی شد. چون از یک تراشه تنها با تغییر برنامه آن، می توان در بخشهای مختلف صنعت استفاده کرد. عمده هزینه تراشه طراحی آن است، هزینه ساخت برای یک تراشه بسیار ناچیز است، بکارگیری یک تراشه در جاهای مختلف باعث ارزان شدن قیمت سیستمهای ساخته شده می گردد. لذا ریزپردازنده جای خود را در صنعت (مثل کنترل ماشینهای صنعتی) در مصارف خانگی (مثل فرهای میکروویو) در مصارف علمی (مثل ماشین های حساب و وسایل کمک آموزشی) باز کردند.

پیدایش ریزپردازنده ها طراحی سیستمهای دیجیتالی را بطور چشمگیری تغییر داد. طراحان امروزه بیشتر سراغ میکروپروسسور می روند و از نرم افزارهای مختلف بگونه ای استفاده می کنند که یک مدار طراحی شده در چند جای مختلف بکار گرفته شود. با استفاده از ریزپردازنده ها می توان کامپیوترهای کوچک و ارزان ساخت که به آنها ریزکامپیوتر (Micro Computer) گفتند. ریز کامپیوتر امروزه با قیمتی کمتر از ۳۰۰ دلار از نظر قدرت محاسباتی بالاتر از اولین کامپیوتر ساخته شده (ENIAC) است. این کامپیوتر ۳۰ برابر سریعتر از آن عمل می کند، خیلی بیشتر از آن حافظه دارد، هزاران بار بیشتر قابل اطمینان است. توان مصرفی آن در حد یک لامپ روشنائی است. کامپیوترهای جدیدی که با استفاده از تکنولوژی LSI ساخته شدند را نسل چهارم کامپیوتر نامیدند. از ریزپردازنده های این نسل Intel8080, PDP11 را برای میکروکامپیوتر و CRAY را برای کامپیوترهای بزرگ می توان نام برد. در نسل پنجم کامپیوترها از مدارات مجتمع VLSI استفاده شد مثال هایی از این نوع 8086 شرکت اینتل (ریزکامپیوتر) VX-11/780 (برای مینی کامپیوترها و کامپیوترهای بزرگ) می توان نام برد. در این کتاب ما به هیچ وجه قصد پرداختن به کامپیوترهای مختلف را نداریم. موضوع این بحث تنها به بررسی ریزپردازنده ها و کاربرد آنها و بصورت یک مثال 8086 مورد بررسی جزئی تر و دقیق تر قرار داده می شود.

فصل اول

۱-۱- تعاریف متداول

۱-۱-۱- بیت(BIT)

بیت (BIT)، منظور از یک بیت یک رقم از اعداد مبنای ۲ می باشد کلمه بیت خلاصه ای از ترکیب دو واژه Digital و Binary بدست آمده است. واضح است تنها مقادیر 1,0 را بخود اختصاص می دهد. این دو رقم را باید با ارقام ۰ تا ۹ پایه ده مقایسه نمود (درس مدار منطقی) تا به چگونگی انجام محاسبات در کامپیوتر واقف شوید. علت انتخاب پایه دو در کامپیوتر سادگی ایجاد (ساخت) این اعداد است. به ۸ بیت یک بایت می گویند و به ۴ بیت یک نیبل (Nibble) گفته می شود. طول کلمه یک ریزپردازنده عبارت است از تعداد بیتی که ریزپردازنده می تواند بطور همزمان پردازش نماید (Word Length). طول کلمه ریزپردازنده 4004 اینتل ۴ بیتی بود. طول کلمه ریزپردازنده های 8080 اینتل و M68000 موتورولا و Z-80 شرکت زایلوگ ۸ بیتی است و طول کلمه 8086 اینتل. M6800 موتورولا و Z8000 دارای ۱۶ بیت است و 80186 اینتل و Z-80000 دارای ۳۲ می باشند. نکته ای که باید یادآور شد اینکه محاسباتی که میکرو انجام می دهد. بطول کلمه آن محدود نمی شود. یعنی میکروپروسسور Z-80 می تواند دو عدد ۱۶ بیتی را نیز جمع کند (البته تحت شرایطی).

۱-۲- واحد محاسبات عددی و منطقی (ALU = Arithmetic Logic Unit)

ALU یا واحد محاسبات عددی و منطقی یک ریزپردازنده، عبارت است از بخشی از سیستم که در آن کلیه محاسبات عددی و منطقی صورت می گیرد. طول لغتی که در این واحد پردازش می شود برابر طول کلمه ریزپردازنده می باشد.

۱-۳-انواع حافظه (MEMORY)

تشکیل شده از تعدادی سلولهای دیجیتالی که در آنها می توان اطلاعات باینری را ثبت کرد و در موقع لزوم مورد استفاده قرار داد. حافظه ها از نظر کاربرد به دو دسته تقسیم می شوند. حافظه با قابلیت خواندن و نوشتن و حافظه فقط با قابلیت خواندن.

۱-۳-۱- ثبـــات (REGISTER)

عبارت است از یک واحد حافظه ای که می توان یک کلمه ریزپردازنده را در آن وارد نمود و نگهداری کرد. هر ریزپردازنده دارای چند ثبات می باشد. ثبات ها به دو دسته تقسیم می شوند.

(الف) ثبات های عمومی، که در کارهای مختلف از آنها می توان استفاده کرد.

(ب) ثبات ها برای کاربرد خاص، همانطور که از نام آنها پیداست برای کارهای خاصی استفاده می شوند.

۲-۳-۱- حافظه های فقط خواندنی (ROM=Read Only Memory)

این حافظه امروزه، در حد وسیعی و در انواع مختلف با استفاده از تکنولوژی LSI, VLSI ساخته می شوند. اطلاعات آنها غیر قابل تغییر است. و در بیرون کامپیوتر برنامه ریزی می شوند. به آنها حافظه پاک نشدنی (NON VOLATILE) می گویند. گاهاً هنگام ساخت توسط کارخانه سازنده پر می شوند (ROM) انواع دیگر دارند (E^2PROM, EPROM, PROM) که توسط استفاده کننده پر می شوند.

۳-۳-۱- حافظه ها با قابلیت خواندن و نوشتن (R/W MEM)

اصطلاحاً به این حافظه های RAM یعنی (Random Access Memory) حافظه با دسترسی تصادفی می گویند. یعنی زمان دسترسی به کلیه اطلاعات یا بعبارت دیگر به کلیه مکانهای این حافظه یکسان است و مستقل از محل قرار گرفتن آن محل مورد نظر در ابتدا یا آخر حافظه است. در این حافظه ها هم می توان نوشت و هم می توان هر محلی که مورد نظر است خواند، یعنی اطلاعات آنرا برداشت.

۱-۴-۱-آدرس باس AddressBus

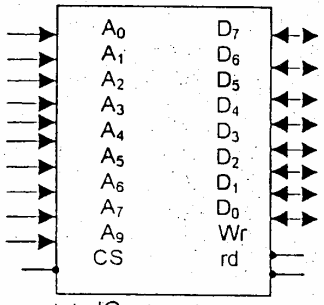
در تراشه های حافظه تعدادی پایه وجود دارد که توسط آنها مشخص می شود با کدام محل حافظه کار دارید این پایه ها را، پایه آدرس (ADDRESS BUS) می گویند اگر ۳ پایه آدرس داشته باشید، حافظه شما 2^3 =8 محل دارد اگر بیش از ۸ محل نیاز داشته باشد باید 2^4 =16 محلی باشد بهمین ترتیب ظرفیت حافظه زیاد می شود اگر حافظه ۱۰ پایه آدرس دارد 2^{10}=1024 محل که نزدیکترین عدد به ۱۰۰۰ است به آن یک کیلو حافظه گفته می شود.

۲-۴-۱- دیتا باس (DATA BUS)

پایه های دیگری هم روی تراشه حافظه ملاحظه می کنید بنام مسیر داده یا (DATA BUS) که آنچه را که می خواهید بنویسید یا از حافظه بخوانید روی این پایه ها عبور می کند. تعداد این پایه برابر با تعداد بیت های کلمه سیستم (Word Length) می باشد اگر سیستمی ۸ بیتی باشد حافظه های آن هم ۸بیتی و تعداد خطوط داده آن نیز ۸ عدد می باشد. شکل صفحه بعد یک حافظه با قابلیت خواندن و نوشتن بظرفیت 1K×8 را نشان می دهد.

خطوط آدرس فقط ورودی به حافظه و خطوط دیتا دو طرفه است.

۱۰خط آدرس ۱۰۲۴=2^{10} محل و ۸ خط دیتا ۸×۱۰۲۴ یعنی هر محل ۸ بیت اطلاعات را ذخیره می کند.



شکل ۱-۱ : نمایش یک IC حافظه

۵-۱- باب های ورودی و خروجی (INPUT/OUTPUT PORTS)

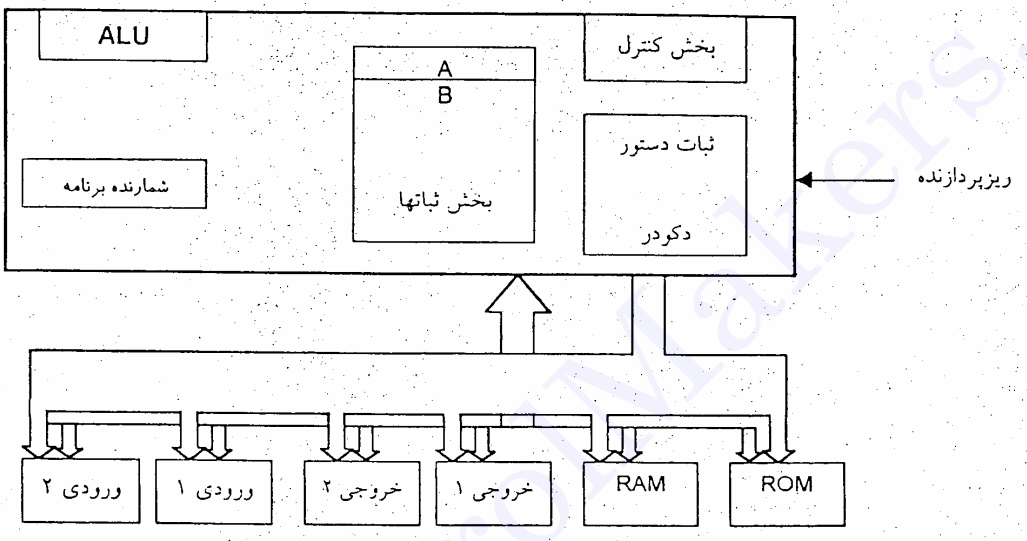
برای اینکه ریزپردازنده ها مفید واقع شوند باید بتوان از خارج به آنها برنامه و داده های مختلفی داد تا کارهای متعددی انجام دهند و نتایج کارها را از آنها بازستانیم این ارتباط از محل هائی صورت می گیرد بنام دروازه یا بابهای ورودی و خروجی

۶-۱- ساختار کلی ریزپردازنده

ریزپردازنده بعنوان بخش کلی و فعالی از ریز کامپیوتر تمام امور مربوط به پردازش اطلاعات، کنترل بخش های مختلف و تولید پالس ساعت سیستم را بعهده دارد. شکل ۲-۱ نمای داخلی و ارتباط ریزپردازنده، با دستگاههای دیگر سیستم را نشان می دهد.

برای اتصال حافظه RAM یا ROM، باب های ورودی و خروجی به ریزپردازنده از یک سری مسیرهای عمومی (BUS) استفاده می شود که عیناً این پایه ها در ریزپردازنده هم وجود دارد. برای درک چگونگی استفاده از BUS ها و برقراری ارتباط بین ریزپردازنده با المانهای جانبی به ذکر مثال زیر می پردازیم : فرض کنید میکروپروسسور بخواهد از محل ۱۲۱ حافظه RAM اطلاعاتی را بخواند. ابتدا آدرس (یعنی عدد ۱۲۱) را روی مسیر آدرس قرار می دهد سیگنال کنترلی را فعال می کند که چیپ RAM به آدرس جواب دهد. بعد سیگنال RD=Read را فعال

می کند که چیپ اطلاعات محل ۱۲۱ خود را به بیرون بفرستد، پس از آن دیتای موجود بوسیله مسیر دیتا درون ثبات مورد نظر در میکروپروسسور قرار می گیرد. ممکن است آنچه که خوانده شد یک دستور باشد، در اینصورت آدرس این محل باید توسط PC=Program Counter داده شود بعد از آنکه وارد میکروپروسسور شد حال باید ترجمه شود که چه دستوری است پس از مشخص شدن باید اجرا گردد.



شکل ۲-۱ یک شمای کلی از ساختمان ریزکامپیوتر

۷-۱-کاربرد ریزپردازنده

استفاده روزافزون از ریزپردازنده در زمینه های مختلف در سه دهه گذشته نظر افراد بسیاری را بخود جلب کرده است. بطوریکه در عرض کمتر از چهار سال پس از عرضه اولین ریزپردازنده 4004 (در سال ۱۹۷۱) جای خود را به عنوان عناصر اصلی سیستم های مختلف باز کرده اند. ریزپردازنده های مختلف با ویژگی های خاص به زودی به بازار عرضه شد و در زمینه های مختلف مثل دستگاههای کنترل بسیار دقیق و پیچیده، ترمینال های باهوش (INTELEGENT)، ماشین حسابهای فروشگاهها، انتقال اطلاعات با سرعت های کم و متوسط و پردازش سیگنالها بکار گرفته شدند. با وجود استفاده وسیعی که ریزپردازنده ها در قسمت های مختلف پیدا کرده اند، ولی همیشه یک عامل باعث محدودیت استفاده از ریزپردازنده ها شده است. این محدودیت سرعت ریزپردازنده است، یک ریزپردازنده معمولاً در یک ثانیه یک میلیون عمل (جمع، تفریق، عمل منطقی یا وارد و خارج کردن اطلاعات) انجام می دهد، فرض کنید در یک سیستم عمل پردازش سیگنال توسط ریزپردازنده انجام بگیرد. ریزپردازنده سیگنال را دریافت می کند روی هر نمونه فرض کنید ۲۰ دستور پردازش نماید و

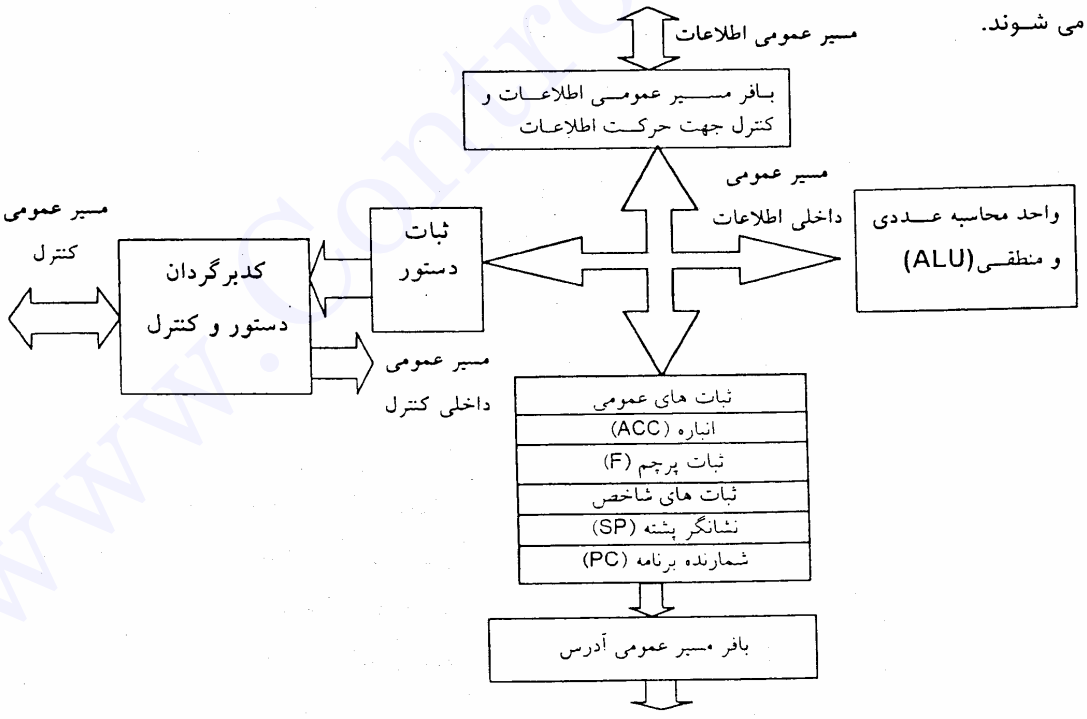
اطلاعات پردازش شده را به خروجی ارسال نماید. در اینصورت در هر ثانیه بتواند ۵۰۰۰۰ نمونه از سیگنال ورودی دریافت کرده و بدون اشکال پردازش نماید. حال اگر سرعت نمونه برداری بیش از این مقدار باشد، ریزپردازنده عقب خواهد ماند و کار سیستم مختل می شود.

۸-۱-ساختمان ریزپردازنده

در بخش قبل با شمای کلی ساختمان ریزپردازنده آشنا شدید. توضیح دقیقتری بصورت کلی درباره ریزپردازنده در این قسمت ارائه می شود و انشاءالله چگونگی برنامه ریزی ریزپردازنده ها و تکنیک مختلف برنامه ریزی به زبان ماشین (با زبان اسمبلی) موضوع فصول بعدی خواهد بود.

۱-۸-۱- ریزپردازنده یا واحد پردازش مرکزی ریزکامپیوتر

قبلاً نیز اشاره شد که ریزپردازنده واحد پردازش مرکزی کامپیوتر است و لذا به آن CPU(CenteraL Processing Unit) گفته می شود. در شکل ۳-۱ یک شمای کلی نشان داده شده است. در داخل CPU نیز مسیرهای عمومی برای مبادله اطلاعات وجود دارد(BUS). کنترل بخش ها نیز توسط سیگنالهایی که از طریق همین مسیرها ارسال می شود اجرا می گردد. این سیگنالهای کنترلی توسط بخش کد برگردان دستور، و بخش کنترل تولید می شوند.



شکل ۳-۱- شمای کلی از ساختمان داخلی یک ریزپردازنده

در دنباله این بحث به توضیح بخش های مختلف شکل ۳-۱ می پردازیم.

۲-۸-۱- ثبات های عمومی (General Purpose Register)

قبلاً ذکر شد که درون یک ریزپردازنده تعدادی ثبات داریم که برای نگهداری بخشی از اطلاعات موجود در سیستم می توانند مورد استفاده قرار بگیرند. ظرفیت هر ثبات به اندازه طول کلمه سیستم است. دسترسی به محتوای ثبات ها بمراتب ساده تر و سریعتر از دسترسی به محتوای حافظه کامپیوتر است. لذا ریزپردازنده در ثبات های عمومی اطلاعاتی را نگهداری می کند. که بیش از هر اطلاعات دیگری به آن نیاز دارد.

ریزپردازنده ثبات های دیگری دارد که هر کدام کار خاصی را انجام می دهند و به آنها ثبات با اهداف خاص (Special Purpose Reg) گویند. که در ادامه به بحث تک تک آنها پرداخته خواهد شد.

۳-۸-۱- انباره (Accumulator)

در اکثر ریزپردازنده ها یکی از ثبات های عمومی بیش از ثباتهای دیگر مورد استفاده قرار می گیرد. این ثبات را معمولاً انباره یا آکومولاتور می نامند. همیشه یکی از داده های مورد پردازش درون آکومولاتور است و حاصل هم به آن ریخته می شود. البته ریزپردازنده هایی هم هستند که تعداد انباره های آنها بیش از یکی است. بعنوان مثال ریزپردازنده M6800 دارای دو انباره A,B می باشد.

۴-۸-۱- ثبات پرچم (Flag Register)

این ثبات با سایر ثباتها فرق دارد، هر یک از بیتهای این ثبات مبین اطلاعات خاصی است و ممکن است تمام بیتهای آن قابل استفاده نباشد. اطلاعاتی که این ثبات در اختیار ریزپردازنده قرار می دهد به شرح زیر است :

۱-۴-۸-۱- پرچم صفر (ZeroFlag)

در صورتیکه نتیجه حاصل از عملی که ریزپردازنده انجام می دهد صفر شود پرچم صفر (Z) مساوی یک می شود و در غیر اینصورت برابر صفر است.

۲-۴-۸-۱- پرچم نقلی (CarryFlag)

در صورتیکه آخرین عمل انجام شده دارای یک بیت نقلی باشد پرچم C مساوی یک می شود و الا صفر خواهد بود.

۳-۴-۸-۱- پرچم سرریز (OverflowFlag)

در صورتیکه حاصل آخرین عمل انجام شده از تعداد بیتهای سیستم بیشتر شود بعبارت دیگر سرریز شود این بیت (V) برابر یک و در غیر اینصورت صفر خواهد بود.

۴-۴-۸-۱- پرچم علامت (SignFlag)

آخرین بیت ظرف سیستم (آکومولاتور) را بیت علامت می نامند اگر عدد مورد نظر مثبت باشد بیت علامت صفر است و اگر عدد مورد نظر منفی باشد بیت علامت یک است و عدد بصورت متمم ۲ خواهد بود، حال اگر حاصل آخرین عمل انجام شده توسط ریزپردازنده منفی شود بیت علامت (S) یک می شود و اگر مثبت باشد این بیت صفر است.

۵-۴-۸-۱- پرچم توازن (ParityFlag)

در صورتیکه تعداد بیت های ۱ موجود در نتیجه حاصل آخرین عمل ریزپردازنده عدد زوجی باشد (توازن زوج) و یا عدد فردی باشد (توازن فرد) پرچم توازن (P) مساوی یک و در غیر اینصورت برابر صفر می شود. معمولاً در سیستمها از توازن زوج استفاده می شود. علاوه بر بیتهای پرچم مذکور، دو بیت دیگر وجود دارد که کاربرد آنها در محاسبات BCD است و آنها عبارتند از :

۶-۴-۸-۱- پرچم نقلی میانی (HalfCarryFlag)

پرچم رقم نقلی میان (HC) که در ریزپردازنده های ۸ بیتی کاربرد دارد، در صورتی مساوی ۱ خواهد شد که آخرین عمل انجام شده دارای یک رقم نقلی بین دو رقم میانی خود باشد.

۷-۴-۸-۱- پرچم تفریق (SubtractFlag)

اگر عمل انجام شده یک عمل تفریق باشد این پرچم (N) مساوی ۱ خواهد شد، در غیر اینصورت صفر است. کاربرد عمده این بیت ها، بغیر از اینکه اعلام وضعیت سیستم را بعهده دارند، در پرشهای شرطی در برنامه نویسی نیز کاربرد دارند، که به تفصیل صحبت خواهد شد.

۵-۸-۱- شمارنده برنامه PC (ProgramCounter)

شمارنده برنامه همیشه حاوی آدرس کلمه ای از حافظه است که در اولین فرصت باید به داخل ریزپردازنده خوانده شود، کلمه خوانده شده می تواند که متناظر با یک دستور باشد و یا اطلاعاتی که ریزپردازنده برای اجرای دستور موجود به آن نیاز دارد. در بعضی از ریزپردازنده ها به این ثبات اشاره کننده به دستورالعمل می گویند (InstructionPointer)

۶-۸-۱- ثبات شاخص (IndexRegister)

در بعضی از ریزپردازنده ها، از ثبات شاخص بمنظور نگهداری اطلاعات لازم جهت تعیین آدرس استفاده می شود. محتوی این ثبات با اطلاعاتی که در دستورات مربوطه وجود دارد جمع می شود تا آدرس مورد نظر بدست آید.

۷-۸-۱- ثبات نشانه گر پشته (SP=StackPointer)

پشته چیست؟ پشته (Stack) عبارت است از حافظه ای که در هر لحظه تنها به آخرین اطلاعات وارد شده به آن می توان دست یافت، بهمین جهت می گویند دارای سازمان LIFO(Last In First Out) است. یعنی اولین اطلاعاتی که می توانید بردارید، آخرین اطلاعاتی است که در آن قرار داده اید. برای قرار دادن اطلاعات در پشته از دستور PUSH و برای برداشتن اطلاعات از دستور POP استفاده می شود. پشته (Stack) معمولاً بخشی در حافظه RAM سیستم است که برای این کار اختصاص داده می شود. برای قرار دادن اطلاعات در پشته یا برداشتن اطلاعات از پشته نیاز به آدرس محل مورد نظر داریم که نشانگر پشته (SP) آدرس مورد نظر را در اختیار ریزپردازنده قرار می دهد.

۸-۸-۱- ثبات دستور (InstructionRegister)

کدهای مربوط به دستورات متوالی که باید توسط ریزپردازنده اجرا شوند، در ابتدا به این بخش از ریزپردازنده وارد می شوند، و از آنجا به عنوان ورودی بخش کدبرگردان دستور بکار برده می شوند.

۹-۱- کدبرگردان دستور (InstructionDecoder)

این بخش از ریزپردازنده که دستوری که باید اجرا شود را از ثبات دستور می گیرد و با توجه به آن سیگنالهای لازم جهت کنترل بخش های مختلف ریزپردازنده را تولید می کند. علاوه بر این، آن سیگنالهائی که به منظور کنترل بخش های داخلی ریزپردازنده بکار می روند، سیگنالهائی جهت کنترل حافظه ها و باب های ورودی و خروجی، که در اطراف ریزپردازنده قرار دارند، نیز تولید می شوند. این سیگنالها توسط مسیر عمومی کنترل به قسمتهای مختلف ریز کامپیوتر انتقال می یابند. همچنین سیگنالهای کنترلی از خارج ریزپردازنده به بخش کنترلی آن وارد می شوند. این سیگنالها به منظورهای خاص که بعداً گفته خواهد شد بکار برده می شود.

۱۰-۱- واحد محاسبات عددی و منطقی (ALU=Arithmetic Logic Unit)

کلیه محاسبات عددی (از قبیل جمع، تفریق و احیاناً ضرب و تقسیم) و منطقی از قبیل (...-AND-OR) در این بخش انجام می گیرند. همچنین عملیاتی از قبیل انتقال چرخش به راست و به چپ، مقایسه دو کلمه در این بخش از ریزپردازنده صورت می گیرد، بطور خلاصه می توان گفت که کلیه عملیات پردازش در ALU انجام می شود و بخش های دیگر ریزپردازنده تنها به منظور حفظ اطلاعات یا کنترل مسیر حرکت اطلاعات بکار می روند. نتایج حاصل از عملیات ALU، در حالت کلی، به انباره می ریزد.

سؤالات دوره ای فصل اول

۱-پایه های یک IC حافظه یا یک IC میکروپروسسور معمولاً به سه دسته تقسیم می شوند آنها را نام ببرید.

۲-اگر تعداد پایه های آدرس یک IC حافظه ۱۲ عدد باشد، چند محل قابل برنامه ریزی دارد؟

۳-اندازه لغت یک IC حافظه چگونه بیان می شود، و به کدام دسته از پایه های آن مربوط می شود؟

۴- قسمت های مختلف داخلی یک ریزپردازنده کدامند؟

۵-انواع حافظه ها کدامند؟

۶-واحد پردازش مرکزی (CPU) به چه قسمت هائی از یک سیستم کامپیوتری گفته می شود؟

۷-میکروپروسسور شامل چه قسمت هائی از یک سیستم کامپیوتری می باشد.

۸-ثبات های معروف یک کامپیوتر کدامند؟

۹-پرچم های (Flags) معروف یک سیستم کامپیوتری کدامند؟

۱۰- راجع به ثبات شمارنده برنامه (PC) و یا اشاره گر به دستورالعمل (IP) چه میدانید؟

۱۱- پشته (Stack) چیست؟ و وظیفه اشاره گر به پشته (SP) کدام است؟

فصل دوم

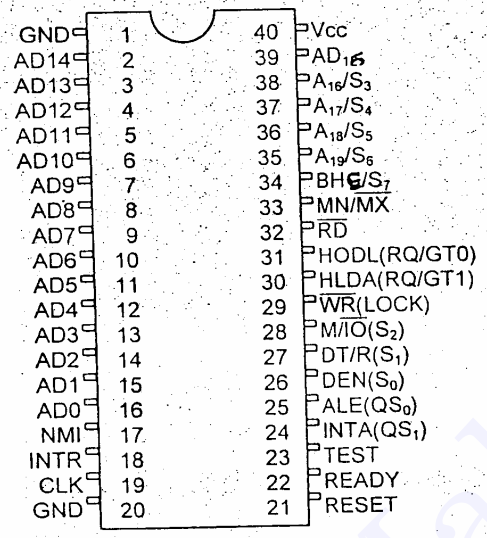
«میکروپروسسور ۸۰۸۶»

یکی از مهمترین ریزپردازنده های ۱۶ بیتی که امروزه کاربرد بسیاری پیدا کرده است بخصوص در صنعت استفاده زیادی دارد، ریزپردازنده ۸۰۸۶ می باشد. ۸۰۸۶ اولین ریزپردازنده ۱۶ بیتی است که ساخت کارخانه اینتل است با ۲۹۰۰۰ ترانزیستور و با تکنولوژی H-MOS در سال ۱۹۷۸ ساخته شد. این ریزپردازنده فرم توسعه یافته 8080 ,8085 است که کلیه دستورالعمل های آنها را اجرا می کند. این ریزپردازنده دارای ۲۰ خط آدرس بوده و قادر است 2^{20}=1M بایت حافظه را آدرس دهی کند بعلاوه دارای ۱۶ خط دیتاست این تعداد خطوط آدرس و دیتا، تسهیلاتی جهت اجرای چند برنامه (Multy Programming) در آن واحد را فراهم می سازد. همچین با بکارگیری ریزپردازنده کمکی اش (8087) سیستم دارای قدرت چند پردازش نیز می شود (Multiy Proccessing).

شرکت اینتل در سال ۱۹۸۲ با متمرکز کردن بعضی از IC های جانبی، ریزپردازنده های 80188, 80186 را با تعدادی دستورالعمل اضافی نسبت به ۸۰۸۶ به بازار عرضه کرد، یک سال بعد ۸۰۲۸۶ را با کارآئی بیشتر ساخت که پردازشگر کامپیوترهای AT شد. بعد از آن ۸۰۳۸۶ را برای آنکه تمام ریزپردازنده های بعدی بر پایه ۸۰۸۶ ساخته شده اند، لذا مطالعه و بررسی ۸۰۸۶ نسبت به سایر ریزپردازنده ها ضروری خواهد بود.

۱-۲- معماری (8086-Architecture)

شکل ۱-۲ تعداد و نام پایه های ۸۰۸۶ را نشان می دهد. ۲۰ خط آدرس و ۱۶ خط دیتا دارد. که ۱۶ خط آدرس با ۱۶ خط دیتا مشترک و بنام AD_0 تا AD_{15} نشان داده شده است. جهت صرفه جویی از پایه های مشترک استفاده شده در واقع عمل مولتی پلکس روی آدرس و دیتا انجام شده است. به نظر می رسد امکان کاهش سرعت ریزپردازنده وجود دارد. ولی پس از مطالعه تایمینگ ریزپردازنده متوجه خواهید شد که سرعت تغییر نخواهد کرد. ریزپردازنده دارای ۱۶ خط کنترلی است که کلیه سیگنالهای مورد نیاز CPU را جهت انتقال دیتا از باس و ارتباط با دنیای بیرون CPU فراهم می کنند.



شکل ۱-۲-نمای ظاهری IC ریزپردازنده۸۰۸۶

۲-۲- شرح پایه های (Pin Allocation)

پایه ۴۰ ولتاژ تغذیه (Vcc) سیستم و پایه ۱ و ۲۰ (GND) زمین منبع تغذیه است که این ریزپردازنده با Vcc=5V کار می کند هر دو GND باید وصل شود.

پایه های ۲ تا ۱۶ و ۳۹ برای آدرس و دیتا استفاده شده است (AD_0~AD_{15}). پایه های ۳۵، ۳۶، ۳۷ و ۳۸ برای A_{16} تاA_{19} که این پایه ها با S_3، S_4، S_5، S_6 مالتی پلکس شده اند.

پایه ۱۷ NMI (non-MaskableIntrupt) : پایه درخواست کننده وقفه است که با لبه (از صفر به یک) فعال می شود، نمی توان جلوی آن را گرفت (نمی توان آنرا پوشاند)

راجع به چگونه سرویس دادن به آن بعداً صحبت خواهیم کرد. این پایه ورودی به سیستم است.

پایه ۱۸ INTR(Intrupt Request) ورودی به سیستم است، درخواست کننده وقفه ای است که اگر پرچم اینتراپت یک باشد (IF=1) می تواند سرویس بگیرد.

پایه ۱۹ CLK ورودی به سیستم است که تنظیم کننده زمان بندی سیستم نیز همین CLK ورودی است.

پایه ۲۱ RESET ورودی به سیستم است. اگر لول صفر منطقی برای مدت چهار پالس ساعت فعال باشد ریزپردازنده را به شرح زیر به اول برمی گرداند. چهار ثبات در سیستم وجود دارد : Extra SEG، Code SEG، Data SEG و Stack SEG که DS=0000H، SS=0000H، ES=0000H، CS=FFFFH و ثبات اشاره گر IP=0000H نیز خواهد شد. وضع سایر ثبات ها تعریف نشده است.

راجع به شرح وظایف ثباتها بعداً صحبت می شود. چون آدرس مؤثر هر دستور از جمع CS شیفت داده شده با IP بدست می آید ریزپردازنده به محل FFFF0H رفته و از آنجا شروع به اجرای برنامه می کند. یعنی با RESET کردن سیستم از محل FFFF0H شروع به خواندن دستورالعمل و اجرای آن می کند.

پایه ۲۲ (READY) ورودی به سیستم است، برای تطبیق سرعت ریزپردازنده با حافظه یا ورودی و خروجی که کندتر از ریزپردازنده هستند بکار می رود قرار گرفتن ولتاژ صفر منطقی روی این پایه بمنزله عدم آمادگی حافظه یا ورودی و خروجی مربوطه است و ریزپردازنده در انتظار می ماند.

پایه ۲۳ (TEST) ورودی به سیستم است. زمانی که ریزپردازنده یک دستور Wait دریافت می کند (دستور نرم افزاری در برنامه بنام WAIT). مادامیکه روی این پایه صفر باشد سیستم در حالت WAIT می ماند یعنی دستور دیگری اجرا نمی شود. بمحض HIGH شدن این پایه از WAIT خارج شده و به اجرای دستورات بعدی می پردازد.

پایه ۲۴ (INTA) پایه خروجی از سیستم است. پاسخ به درخواست وقفه (INTR) است. هدف ریزپردازنده از این پاسخ بدست آوردن آدرس برنامه ای است که قرار است به اجرای آن بپردازد.

پایه ۲۵ ALE (Address Latch Enable) خروجی از سستم است برای بافر کردن آدرس در Latchها و جدا کردن آن از دیتا بکار می رود در ابتدا هر ماشین سیکل، سیستم آدرس را روی خط آدرس باس قرار می دهد و توسط ALE در مدارات که برای این منظور فراهم شده، بافر می شود. در پالس های بعدی دیتا روی باس وجود خواهد داشت که مسیر آنها جدا خواهد شد.

پایه ۲۶ (DEN) خروجی است. Activelow می باشد و 3-State است. (Data Enable) این پایه وقتی که ریزپردازنده در مد مینیمم اگر از درایور دیتا استفاده شود خروجی آنرا فعال می کند.

پایه ۲۷ DT/R (DataTransmit/Resive) خروجی از سیستم و 3-State است. وقتی که یک منطقی باشد ریزپردازنده دیتا ارسال می کند و زمانیکه صفر است ریزپردازنده دیتا دریافت می کند. لذا برای آماده کردن درایور دیتا بکار می رود تا جهت ارسال و دریافت را برایش مشخص نماید.

پایه ۲۸ M/IO خروجی و 3-State است. بوسیله آن مشخص می کند که ریزپردازنده با حافظه سروکار دارد یا با ورودی و خروجی، اگر M/IO=0 باشد با ورودی و خروجی و اگر M/IO=1 باشد با حافظه سروکار دارد.

پایه ۲۹ Wr (Write) خروجی از سیستم و 3-State است. برای نوشتن در حافظه یا خروجی استفاده می شود که توسط پایه ۲۸ (M/IO) حافظه یا خروجی مشخص می شود.

پایه ۳۱ (HOLD) ورودی به سیستم و پایه ۳۰ (HLDA) خروجی از سیستم و پاسخ به HOLD است. وقتی که واحدی از بیرون بخواهد به حافظه سیستم دسترسی پیدا کند سیگنال HOLD را می فرستد. زمانی که ریزپردازنده HDLA را فعال کند به او اجازه استفاده از باس را می دهد و پایه های خروجی خودش به 3-State می رود به اینگونه دسترسی به حافظه تکنیک DMA می گویند (Direct Memory Access)

پایه ۳۲ $\overline{RD}$(Read)، این پایه برای خواندن از حافظه یا ورودی فعال می شود که باز از روی $\overline{M/IO}$ می توان حافظه یا ورودی را انتخاب کرد.

پایه ۳۳ (Minimum/Maximum) ریزپردازنده ۸۰۸۶ دارای دو مد کاری است اگر $MN/\overline{MX}=1$ باشد ریزپردازنده در مد ماکزیمم است و کار بعضی از پایه ها عوض می شود مثل : پایه های ۲۴ تا ۳۱ که کارهای دیگری در داخل پرانتز ذکر شده است. شرح کار این پایه ها بعداً ذکر خواهد شد.

پایه ۳۴ $\overline{BHE_{57}}$(BUS High Enable) پایه خروجی و دارای وضعیت 3-State است. با صفر فعال می شود و بایت بالای دیتای را فعال می کند. بعبارتی انتخاب کننده آدرس های فرد حافظه است. بهمراه پایه A_0 آدرس باس زمینه دسترسی به بخش های فرد و زوج حافظه را فراهم می آوردند که بعداً شرح داده خواهد شد. این پایه نیز با S_7 مالتی پلکس شده که شرح پایه های وضعیت سیستم تشریح خواهد شد.

پایه های ۳۵، ۳۶، ۳۷، و ۳۸ که S_3, S_4, S_5, S_6 با خطوط A_{19}, A_{18}, A_{17}, A_{16} آدرس باس مالتی پلکس شده اند. پس از آنکه ALE غیر فعال شد بیان گر وضعیت سیستم هستند که به شرح زیر است :

S_6 همیشه مقدار صفر خواهد داشت.

S_5 نشان دهنده وضعیت پرچم اینترایت است.

S_3 و S_4 بیانگر این مطلب هستند که کدام سگمنت در این سیکل باس مورد دسترسی قرار گرفته اند. جدول ۱-۲ این مطلب را نشان می دهد.

۱-۲-جدول حالات بیت های وضعیت S_3 و S_4

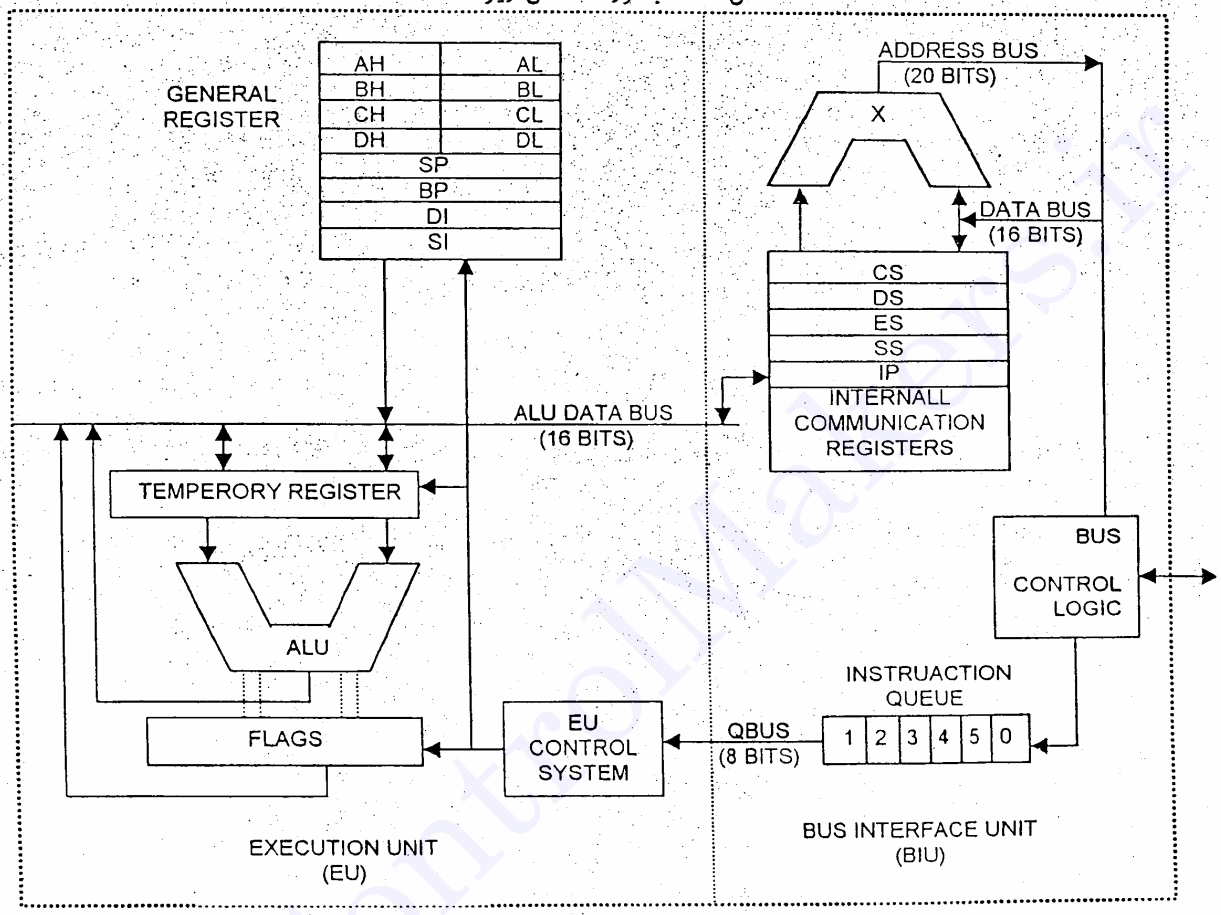
S_4	S_3	FUNCTION
0	0	Extra Segment
0	1	Stack Segment
1	0	Code Segment
1	1	Data Segment

لازم به ذکر است که با استفاده از این دو بیت می توان 4MB حافظه به میکروپروسسور ۸۰۸۶ وصل کرد که هر یک مگا بایت برای یک سگمنت باشد.

سه بیت S_0, S_1, S_2 در مد ماکزیمم ۸۰۸۶ مورد استفاده قرار می گیرد، که بعداً توضیح داده خواهد شد.

۳-۲- ساختار داخلی CPU :

داخل CPU بصورت شکل زیر است :



داخل ۸۰۸۶ را به دو قسمت مستقل از هم به نام های BIU=BusInterfaceUnit و EU=ExecutionUnit تقسیم می کنند، BIU شامل ثبات های سگمنت و IP و صف ۶ بایتی دستورالعمل است. EU نیز شامل بخش کنترل، ALU و پرچم های (FLAGS) و ثبات های چهارگانه AX,BX,CX,DX و ثبات های اشاره گر مثل DI,SI,BP,SP می شود.

این تقسیم بندی صرفاً بمنظور بالا رفتن سرعت میکروپرسسور صورت گرفته است. وظیفه BIU عبارت است از ارسال آدرس روی آدرس باس و خواندن دستورالعمل از حافظه و یا دیتا از حافظه و ورودی و نوشتن اطلاعات در حافظه و خروجی. بعبارت دیگر کلیه ارسال و دریافت اطلاعات به حافظه یا I/O وظیفه BIU است. صف دستورالعمل که در این ریزپردازنده پیش بینی شده برای افزایش سرعت است. زیرا همواره ۶ بایت دستورالعمل از پیش Fetch شده و آماده ترجمه و اجرا است. این بخش بصورت FIFO=First In First Out کار می کند. زمانی که بخش EU مشغول ترجمه و تفسیر و اجرای یک دستورالعمل فچ شده است بخش BIU به خواندن

دستورالعمل بعدی و قرار دادن آنها در صف می پردازد، لذا به غیر از اولین دستورالعمل مابقی زمان خواندن ندارند یعنی صرفه جویی در زمان شده فقط کافی است بخش EU از صف دستورالعمل بعدی را بردارد. بدیهی است اگر دستوری که ترجمه و تفسیر شد از نوع JUMP یاCALL باشد باید صف پاک شود و مجدداً دستوری را که دستور جاری تعیین می کند Fetch نماید، یعنی این بار زمان خواندن دستور خواهد داشت. اما باز دستورات بعد از اولین دستور در صف آماده خواهند شد. به این عمل صف نوعی پردازش خط لوله Pipe Line Processing می گویند.

۱-۳-۲-ثبات های ۸۰۸۶

ثبات های ۸۰۸۶ به سه گروه تقسیم می شوند :

الف) ثبات های گروه دیتا، که یک مجموعه ثبات محاسباتی هستند.

ب) ثبات های اشاره گر: که شامل ثبات های پایه و شاخص (INDEX)می باشند، همچنین شمارنده برنامه و اشاره گر به پشته در این گروه جای دارند.

ج) ثبات های سگمنت : که یک مجموعه ثبات پایه برای اهداف خاصی هستند، تمامی این ثبات ها ۱۶ بیتی هستند.

ثبات های گروه دیتا شامل چهار ثبات است بنام های AX, BX, CX, DX این ثبات ها قادرند هم اپرند عملیات را داشته باشند هم می توانند حاصل عملیات را ذخیره کنند. و هر کدام از آنها می توانند بصورت ۱۶ بیتی یا بصورت دو ثبات ۸ بیتی مورد استفاده قرار بگیرند. بعنوان مثال ثبات AX هم می تواند ۱۶ بیت دیتا را در خود ذخیره کند و هم اینکه می توان آن را بصورت دو ثبات ۸ بیتی کاملاً مجزا و مستقل از هم مثل AL, AH مورد استفاده قرار داد. زیرا گفته شد این میکروپروسسور می تواند جانشین ۸۰۸۵ شود و با سیستم های ۸ بیتی بکار رود. بصورت زیر می تواند جانشین ۸۰۸۰ یا ۸۰۸۵ باشد.

8086	8080
AL	A
BH	H
BL	L
CH	B
CL	C
DH	D
DL	E

مضافاً اینکه سه ثبات DX, CX, BX علاوه بر سرویسی که در محاسبات می دهند می توانندکاربرد خاصی مثل آدرس دهی - شماره نده ای و نقش آدرس دهی ورودی و خروجی به شرح زیر را بعهده بگیرند :

۱- AX در بعضی موارد بعنوان آکومولاتور مورد استفاده قرار می گیرد، دریافت و ارسال دیتای I/O از طریق این ثبات است.

۲-BX می تواند بعنوان ثبات پایه (BASE REGISTER) در محاسبات آدرس مورد استفاده قرار بگیرد.

۳-CX را می توان بعنوان شمارنده (COUNTER) در دستوراتی که نیاز به شمارش است استفاده کرد (مثل عملیات رشته ای)

۴-DX را می توان بعنوان نگهدارنده آدرس I/O در عملیات مورد استفاده قرار داد.

ثبات های اشاره گر: شامل DI, SI, BP, SP, IP هستند. ثبات های SP, IP همان شمارنده برنامه و اشاره گر به پشته می باشند. البته برای بدست آوردن آدرس تکمیلی ۲۰ بیتی باید محتوای آنها تحت شرایطی با SS, CS جمع شود که در زیر شرح داده خواهد شد.

BP بعنوان ثبات پایه برای دسترسی به پشته و ممکن است با سایر ثبات ها و یا با محتوای میدان آدرس (Displacement) که بخشی از دستورالعمل محسوب می شود، بکار برده شود.

دو ثبات DI, SI دو ثبات شاخص هستند. اگر چه ممکن است آنها به تنهایی مورد استفاده قرار بگیرند، ولی معمولاً آنها با ثبات های BP, BX و محتوای میدان آدرس (Displacement) بکار می روند. بجز IP که استثناست، یک ثبات اشاره گر می تواند حاوی اپرند نیز باشد. در اینصورت باید به تنهائی در دستور قرار بگیرد.

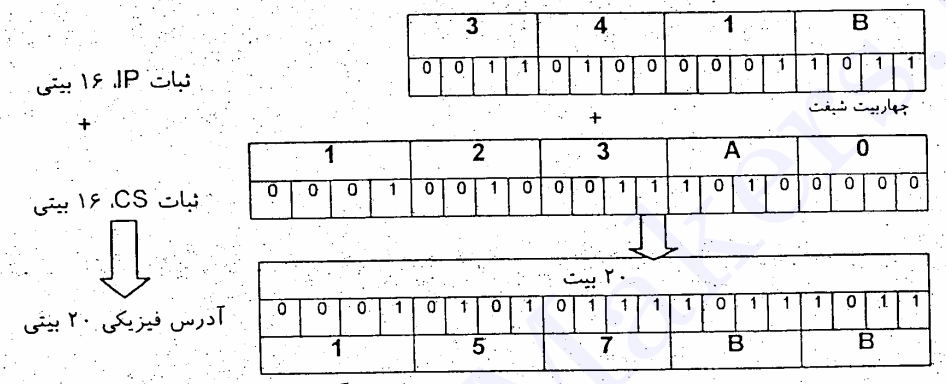
در شیوه های آدرس دهی خواهید دید که برای داشتن مانور در آدرس دهی، آدرس یک دیتا را ممکن است با جمع کردن ترکیبی از BX یا SI, DI, BP و محتوای میدان آدرس دستورالعمل بدست آورد. حاصل چنین محاسبه ای را آدرس مؤثر می گویند (EA = Effective Address) یا Offset می گویند. در کتب Intel (Manuals) وقتی درباره زبان ماشین صحبت می کند واژه آدرس مؤثر (Effective Address) بکار برده شده است. وقتی درباره زبان اسمبلی صحبت می کند واژه Offset بکار می برد. و واژه Displacement برای بیان مقداری که به محتوای ثبات ها اضافه می شود تا آدرس مؤثر بدست آید بکار می رود. البته توجه دارید که بحث آدرس دهی مخصوص دسترسی به دیتا است و برای بدست آوردن آدرس فیزیکی (۲۰ بیتی) باید آدرس مؤثر (EA) با یکی از ثبات های سگمنت جمع شود.

ثبات های سگمنت: عبارتند از ES, DS, SS, CS همانگونه که تاکنون بحث شد سیستم ۱۶ بیتی است، آدرس مؤثر نیز ۱۶ بیتی است ثبات های سگمنت هم ۱۶ بیتی هستند، اما آدرس فیزیکی یا آن مقداری که قرار است روی آدرس باس قرار داده شود ۲۰ بیتی است، ۴ بیت اضافه از جمع آدرس مؤثر با یکی از ثبات های سگمنت بصورت شکل ۳-۲ بدست می آید. بدین ترتیب که ثبات سگمنت مربوطه چهار بیت به سمت چپ شیفت داده می شود بعدچهار

صفر به آن وارد می شود و با آدرس مؤثر جمع می شود. فرض کنید (CS)=123A باشد و (IP)=341B آدرس فیزیکی برای فیچ کردن دستورالعمل مورد نظر بصورت زیر است.

341B	EA
123A0	CS
157BB	آدرس فیزیکی است

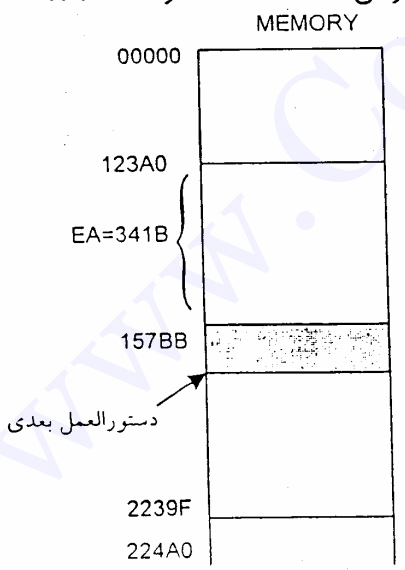
پرانتزی که دور IP و CS قرار داده شده علامت استانداردی است و معنی آن محتوای IP و CS مورد نظر است.



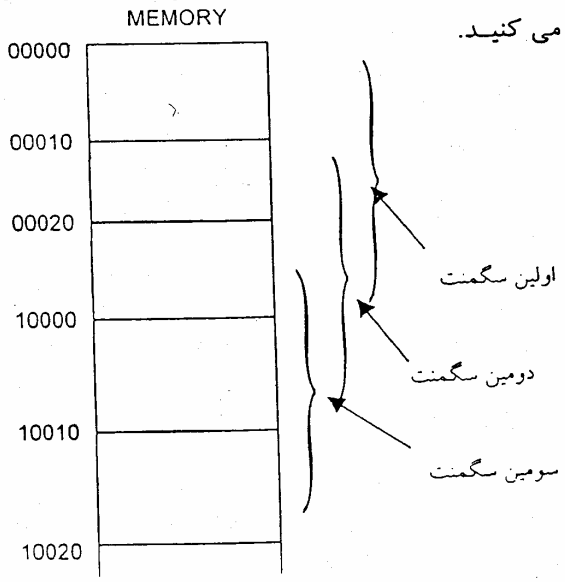
شکل ۳-۲-نحوه شکل دهی آدرس فیزیکی

اساساً بکار گرفتن ثبات های سگمنت باعث می شود که فضای حافظه به قسمت هایی تقسیم شوند که با هم تداخل دارند. هر سگمنت دارای 64 KB طول خواهد داشت که شروع آن از محلی است که محتوای ثبات سگمنت ضربدر ۱۶ نشان می دهد. برای مثال نمایش بخش کد حافظه مثال قبلی بصورت زیر می باشد. (شکل a-۴-۲)

در شکل b-۴-۲ طرز قرار گرفتن سگمنت های مختلف با شرحی که گذشت را مشاهده می کنید.



شکل a-٤-۲- نمایش یک سگمنت دستورالعمل



شکل b-٤-۲-سگمنت هائی که بخش مشترک دارند

سؤالی که اینجا قابل طرح است این است که آیا سگمنت کردن حافظه مزیتی هم دارد یا نه؟
پاسخ مثبت است مزایای تقسیم حافظه به بخش های مختلف به شرح زیر است :

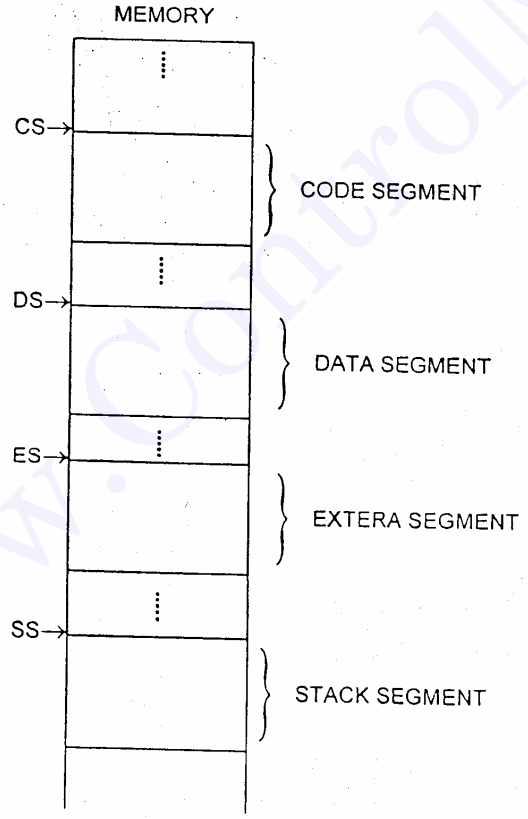
۱-اجازه می دهد که یک ریزپردازنده ۱۶ بیتی قادر شود تا یک مگا محل حافظه را آدرس دهدی کند.

۲-اجازه می دهد که بخش های دستورالعمل، دیتا و پشته در یک برنامه بیش از 64KB باشد.

۳-تسهیلات بکارگیری نواحی جداگانه ای را برای دستورات و دیتا و پشته یک برنامه فراهم می کند.

۴-اجازه می دهد برای هر بار اجرای یک برنامه محل جدیدی را برای ذخیره دیتای آن در نظر بگیرید.

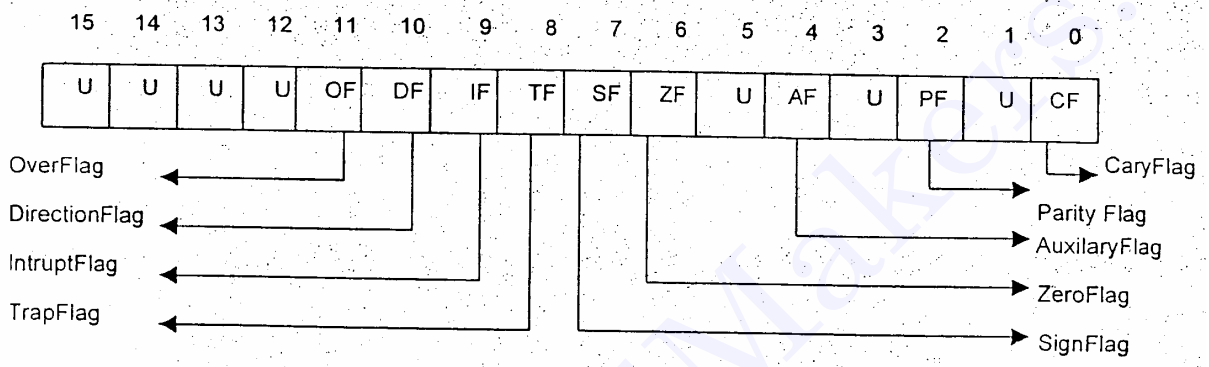
حتی شما می توانید برای یک برنامه ناحیه دستورالعمل و دیتا و پشته را بدون داشتن تداخلی با یکدیگر بصورت کاملاً مجزا انتخاب نمائید.(شکل ۵-۲) این اختصاص خیلی مناسب و ساده است حتی می توانید ناحیه پشته را همیشه در یک محل ثابت انتخاب نمائید، مثل شروع پشته همیشه آدرس 08000 می باشد.



شکل ۵-۲- جدا بودن کد سگمنت، دیتا سگمنت، اکسترا سگمنت و استک سگمنت

۲-۳-۲- ثبات پرچم(FLAGS REGISTER)

ثبات پرچم ها FlagRegister : یک پرچم یک فیلیپ فلاپ است که نشان دهنده بعضی شرایط ایجاد شده بوسیله اجرای یک دستور یا کنترل کننده یک عمل مطمئن در واحد اجرائی است. ثبات پرچم ۸۰۸۶ یک ثبات ۱۶ بیتی است که ۹ بیت آن فعال است. شکل زیر بیت های فعال را نشان می دهد و بیت های غیر فعال با U معرفی شده بمعنی Undefined است.



شکل ۶-۲ ثبات پرچم ها

در ۸۰۸۶ پرچم ها به دو دسته تقسیم می شوند، اول پرچم های شرایط (UnditionalFlags) که بیانگر نتیجه عمل قبلی است که در ALU انجام شده است. دومین گروه پرچم های کنترلی (ControlFlags) هستند که کنترل کننده عملیات خاصی هستند.

شش تا از آنها برای نمایش وضع ایجاد شده از اجرای یک دستور است مثل CF بیت اضافه شده به تعداد ۱۶ بیت سیستم در عمل جمع اگر یگ بیت اضافه شود CF=1 خواهد شد والا CF=0 است. با یک شدن CF سیستم اعلام می کند که بیتی بعنوان با ارزشترین بیت به مقدار اضافه شده است. PF بیت دیگری است که در اثر عملیات فعال می شود. PF یک می شود، اگر ۸ بیت کم ارزشتر عملیات دارای تعداد زوجی یک باشد در غیر این صورت صفر است.

AF=1 اگر در عملیات جمع در چهار بیت اول Cary ایجاد شود یا در عملیات تفریق Borrow نیاز شود والا صفر خواهد بود.

ZF=1 اگر حاصل اجرای عمل قبلی صفر شود، در غیر اینصورت صفر خواهد بود. SF برابر است با MSB نتیجه عملیات یعنی اگر یک شود پس MSB=1 است و عدد حاصله منفی است، یعنی بصورت متمم ۲ خواهد بود.

OF=1 اگر حاصل عملیات دچار سرریزی گردد. یعنی نتیجه از رنج مجاز سیستم بیشتر شود. سه پرچم دیگر بیت های کنترلی ریزپردازنده هستند :

DF جهت حرکت عملیات را در پردازش های رشته ای تعیین می کند اگر DF=0 باشد اجرای رشته دیتا از آدرس ابتدای المانها شروع و تا آخر ادامه می یابد و اگر DF=1 باشد رشته دیتا از آدرس بالائی شروع و اجرا و به اولین آدرس ختم می شود.

IF پرچم فعال ساز اینترایپت است. اگر IF=1 باشد اجازه ورود اینترایپت قابل پوشش (MaskableIntrupt) داده شده است. والا باید از اینترایپت چشم پوشی گردد.

TF پرچمی است که اگر TF=1 شود بعد از اجرای هر دستورالعمل سیستم متوقف می شود یعنی برنامه قدم به قدم اجرا می گردد. لازم به ذکر است که ۸ بیت اول این پرچم ها شبیه ۸ بیت پرچم 8080 و 8085 است.

۲-۴-عملکرد داخل میکروپروسسور

عملکرد عمومی یک کامپیوتر به شرح زیر است:

۱- خواندن دستورالعمل (Fetch) از حافظه که آدرس آن با توجه به محتوای کد سگمنت و IP محاسبه می شود

آدرس فیزیکی ۲۰ بیتی = CS*16+IP

۲- قرارد دادن دستورالعمل خوانده شده در ثبات دستورالعمل و دکود شدن آن در زمانیکه به محتوای IP یکی اضافه می شود تا آدرس دستورالعمل بعدی را تولید کند.

۳- اجرای دستورالعمل مذکور و در صورتیکه دستور فوق یک انشعاب باشد پر کردن IP با آدرس محلی که قرار است به آنجا منشعب شویم.

۵-تکرار قدم یک تا سه .....

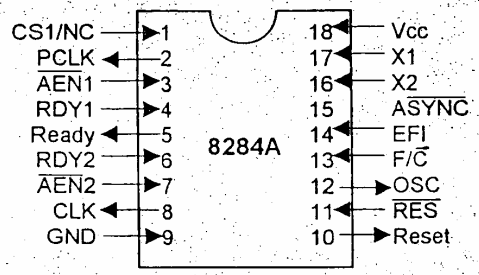
علاوه بر عملیات مذکور ممکن است کارهای دیگری در هنگام اجرای کارهای فوق اجرا شود مثل پرشدن صف ۶ بایتی دستورالعمل هر وقت که سیستم باس برای کار دیگری مورد استفاده قرار نگیرد. بدیهی است که زمان لازم برای خواندن دستورالعمل بعدی ذخیره می شود. اگر یک دستور انشعاب خوانده شده باشد دستوراتی که در صف قرار داده اید مفید نخواهد بود باید دستور جدیدی که در این انشعاب تعریف شده خوانده شود، بدون اینکه در زمان صرفه جوئی شود البته اینگونه دستورات خیلی کم است.

۲-۵- مولد سیگنال کلاک برای 8086/8088

در این قسمت به معرفی IC مولد کلاک (8284A)، سیگنال RESET و تأمین کننده سیگنال Ready پرداخته می شود.

IC شماره 8284 یک المان مناسبی برای میکروپروسسورهای 8086/8088 می باشد، بدون این مولد کلاک، به مدارات اضافی زیادی نیاز خواهید داشت، لذا مناسب ترین المان IC مذکور است که سیگنالهای زیر را تولید می کند :

تولید کلاک، ایجاد همزمانی برای سیگنال RESET و ایجاد همزمانی برای ورود سیگنال Ready بعهده این IC است. این IC دارای ۱۸ پایه بشکل و شرح زیر می باشد:



شکل ۷-۲-IC تولیدکننده پالس ساعت

دو پایه ۹و و ۱۸ که مشخص است، Vcc و GND خواهد بود و با و لتاژ +5V با تلورانس ±10% کار می کند.

دو پایه 16, 17 بنام های X1,X2 به یک کریستالی که تأمین کننده کلاک سیستم است وصل می شود، در اینصورت باید پایه شماره ۱۳ که (Frequency/Crystar) نام دارد به زمین وصل شود.

نحوه دیگر که می توان کلاک سیستم را تأمین کرد، استفاده از یک کلاک خارجی آماده است که باید به ۱۴ وصل شود (ExternalFrequencyInput) در اینصورت پایه ۱۳ باید به Vcc وصل شود. اگر از کریستال داخلی استفاده شود باید EFI نیز گراند شود.

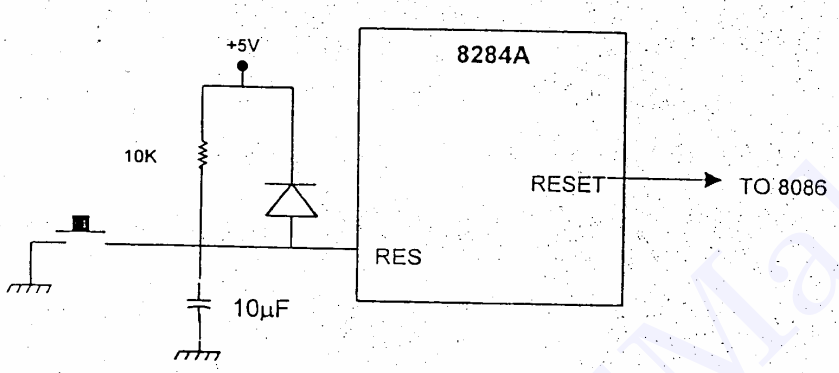
پایه شماره ۸ کلاک ساخته شده توسط 8284A برای میکروپروسسور سایر IC های سیستم است، این CLK یک خروجی دارد که 1/3 فرکانس کریستال یا فرکانس ورودی EFI می باشد. دارای زمان دوام پالس (Dutycycle) 33% می باشد.

پایه شماره ۲، PCLOCK : این پایه کلاک جانبی (PeripheralClock) نام دارد، یک سیگنال ارائه می دهد که فرکانس آن 1/6 فرکانس کریستال یا فرکانس ورودی از پایه EFI می باشد. و زمان دوام پالس آن (DutyCycle) 50% است. این سیگنال بعنوان کلاک برای دستگاههای جانبی سیستم مورد استفاده قرار می گیرد.

پایه شماره ۱۲، OSC نام دارد (OscilatorOutput). این پایه سیگنالی ارائه می دهد که دارای سطح ولتاژ TTL و فرکانس برابر فرکانس کریستال یا برابر فرکانس ورودی EFI می باشد. هدف از این سیگنال تهیه سیگنالی برای ورودی به EFI، ICهای 8284A دیگری که برای استفاده از چند پردازشگر مورد نیاز می باشد.

پایه شماره یک CSYNC نام دارد. همزمان کننده کلاک است (ClockSynchronization) در سیستمی که چند پردازشگر مورد نیاز است و فرکانس CLK از EFI استفاده می شود، این پایه برای همزمانی کلاکهای سیستمها بکار می رود. اما اگر از کریستال استفاده می کنید باید به GND وصل شود.

پایه شماره ۱۱ ($\overline{RES}$)، ورودی Reset است که از دکمه فشاری سیستم یک زمین دریافت می کند، معمولاً از یک مدار RC مطابق شکل زیر استفاده می شود:

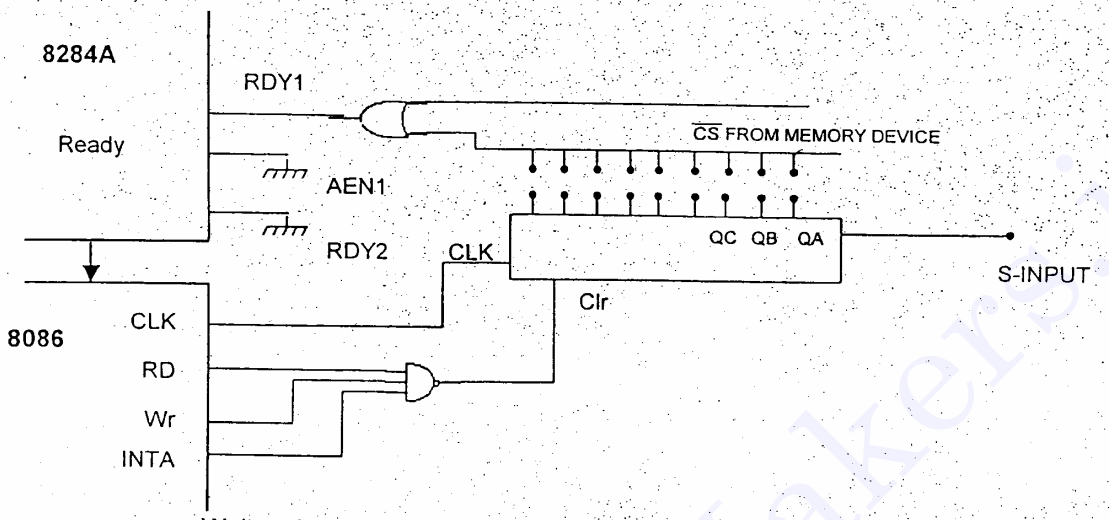


شکل ۸-۲-طریقه وصل کردن دکمهٔ RESET به IC شماره 8284

۱-۵-۲-سیگنال Ready و Wait-State

بطوریکه قبلاً گفته شد پایه Ready در میکروپروسسور سبب می شود که سرعت میکرو بخاطر حافظه ها یا I/O ها کندتر شود. برای تحقق این امر یک یا چند پالس ساعت (Wait-State) TW، بین T4,T3 اعمال می شود. اگر یک TW اعمال شود زمان دسترسی به حافظه که بطور معمول 460nSec است با فرکانس کاری میکروپروسسور که 5 MHz فرض می شود به 660nSec افزایش یافته است. پایه ورودی Ready در لبه پایین رونده T_2 هر سیکل ماشین نمونه برداری می شود. اگر صفر منطقی بود یک TW ساخته می شود و تولید T_4 به تأخیر می افتد و یک TW جایگزین آن می شود، مجدداً در اواسط TW عمل نمونه برداری تکرار می شود و هر وقت Ready دارای سطوح یک منطقی شد تولید TW متوقف و T_4 تولید می شود. توسط 8284A IC دو دستگاه (مثل حافظه و I/O) می توانند تقاضای Wait کنند. پایه های 3,7 ، AEN2,AEN1(AddressEnable) همراه با پایه های 6,4 (AEN1,AEN2) این IC، یک سیگنال Ready مناسب برای ۸۰۸۶ تهیه می کنند. AEN1,AEN2 با صفر فعال می شوند و با آمدن RDY مربوطه در پایه ۵ این IC یک سیگنال Ready با دامنه ۵ ولت برای میکروپروسسور ارسال می نمایند که به پایه Ready میکروپروسسور وصل می شود، RDY1 و RDY2 از طرف دستگاه جانبی ای که نیاز به TW دارد ارسال می شوند (شکل ۹-۲).

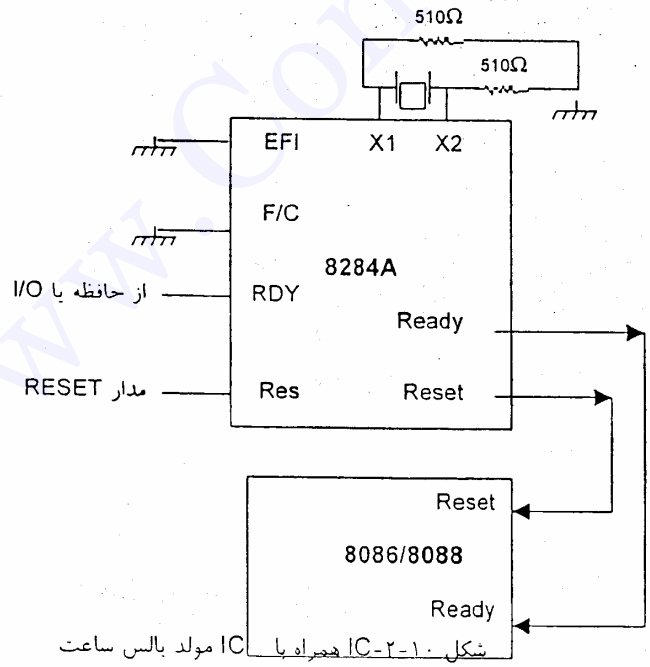
مدار زیر مولد مناسبی بـــرای Ready مـی باشـد.



شکل۹-۲-تولید سیگنال

در مدار بالا در سه موقع نیاز به TW داریـم، خوانـدن – نوشـتن و پاسـخ بـه اینـتراپـت لـذا ابتـدا شـیفت رجیسـتر مربوطـه (IC شـماره 74LS164) پـاک (Clr) مـی شـود بـا CLK بعـدی یـک بداخـل آن شـیفت پیدامـــی کنـــد و در QA قـــرار مـــی گـــیرد پـــس از یـــک کـــلاک بـــه QB میرســـد و یـــک بـــه ورودی RDY1 8284A می فرسـتد. یعنـی یـک TW بیـن T_3 و T_4 اعمـال مـی کنـد.

لازم بـه ذکـر اسـت کـه میکـروپروسسـور 8086 دارای انـــواع مختلفـــی اســـت کـــه رنـــج CLK آنـــها از 5MHz تـا 10MHz مـی توانـد تغـییر کنـد. شـکل ۱۰-۲ میکـروپرسسـور را همـراه بـا IC مولـــد CLK نشان می دهـد.



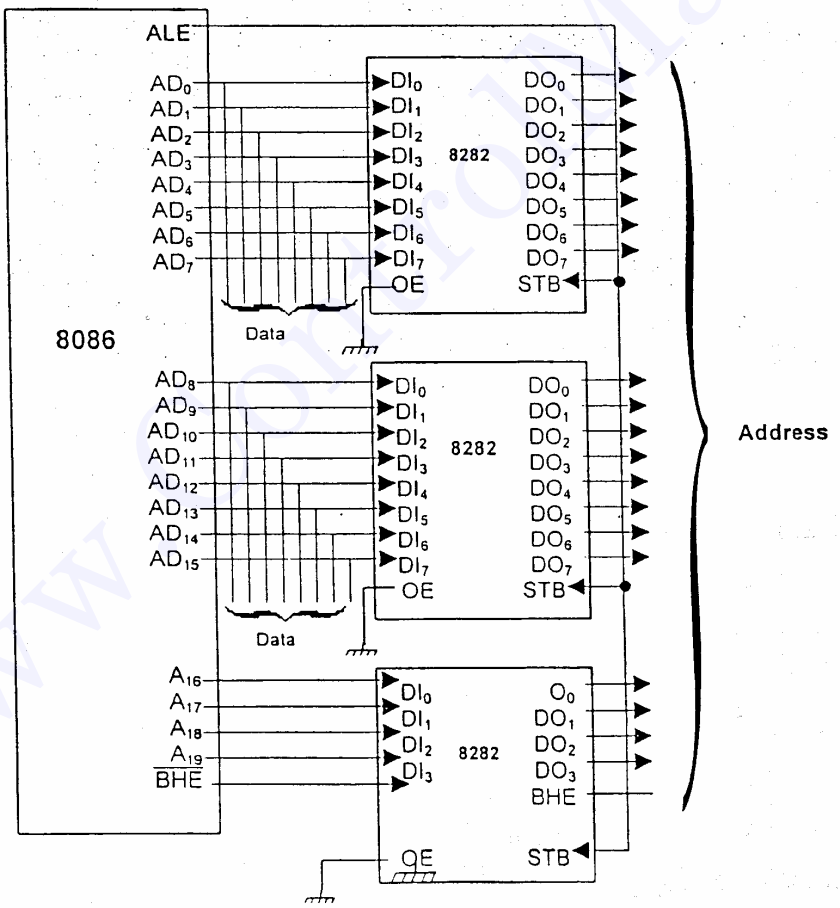
شکل ۱۰-۲-IC همراه با

۶-۲-جداسازی آدرس بـاس از دیتـا بـاس

علت جداسازی آدرس از دیتا اینسـت کـه هـر دو روی بـاس مشـترک ارسـال مـی شـوند، بـرای اینکـار از IC شـمار 8282 بنـام AddressLatch اسـتفاده مـی شـود. البتـه مـی توانیـد از IC شـماره 74Ls373 نـیز اسـتفاده نمـائید هـر کـدام دارای ۸ عـدد Latch مـی باشـند. در واقـع مـی تـوان گفـت ۸ ورودی ۸ خروجـی و یـک پایـه STB کـه بـازکننده ورودی Latch بـرای دریـافت اطلاعـات هسـتند و یـک پایـه هم OutputEnable که OE نامیده مـی شـود بعـلاوه GND,Vcc از پایـه هـای دیگـر ایـن IC اسـت.

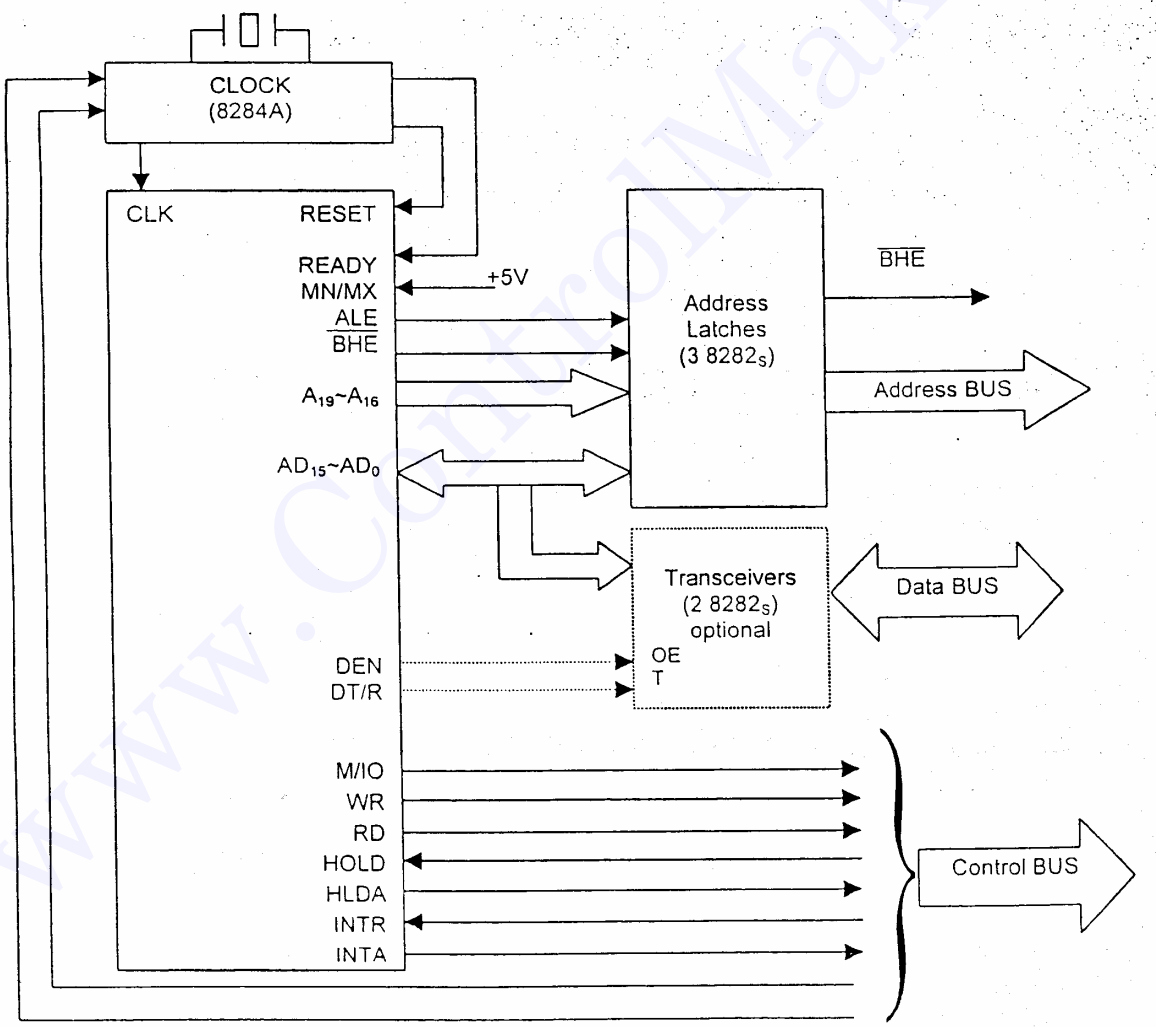
خطـوط AD_0 و AD_7 بـه IC اول و AD_8 تـا AD_{15} بـه IC دوم و AD_{16} تـا AD_{19} و BHE بـه IC سـوم داده می شـود.

میکـروپروسسـور در اولیـن پـالس هـر عملیـاتی آدرس را روی AD_0~AD_{19} قـرار مـی دهـد بعـد از آن ALE (پـایه ۲۵) را فعـال مـی کنـد. پایـه ۲۵ را بـه STB هـر سـه IC لـچ مذکـور وصـل مـی کنیـم تا AD_0~AD_{19} در لچ ها ذخـیره شـود خروجـی آنـها A_0~A_{19} اسـت. مطـابق شـکل ۱۱-۲ :



شکل۱۱-۲-میکروپروسسور به همراه IC های Latch

OE مربوط به ICهـای 8282 را همـواره فعـال مـی کنیـم تـا آدرس همیشـه در خروجـی آنـها یعنـی روی آدرس باس باشد. البته اگر قرار باشد ریـزپردازنـده بـا ریـزپردازنـده هـای دیگـر کـار کنـد و بخواهیـم از تکنیک DMA استفاده کنیـم نبـاید همیشـه OE را بـه زمیـن وصـل کـرد. اگـر بـه زمیـن وصـل نباشـد یعنی Vcc داشته باشد خروجـی بافرهـا بـه امپدانـس بـالا مـی رود (Hiz). در مسـیر دیتـا بـاس نـیز اگـر تعداد دستگاه زیاد نباشد نیاز بـه بـافر کـردن دیتـا بـاس نیسـت امـا اگـر قـرار باشـد سیسـتم تعـدادی زیادی IC حافظه یا ورودی و خروجـی داشـته باشـد یا نیـاز بـه تکنیـک DMA باشـد (بـا چنـد پردازنـده کار کند) باید سـر راه دیتـا بـاس نـیز بـافر گذاشـته شـود. اینتـل IC، 8286 را معرفـی مـی کنـد کـه دو عـددIC مـورد نیـاز اسـت. AD_0~AD_{19} بـه ورودی آنـها وصـل مـی شـود. ایـن IC نـیز دو پایـه کنـترلی دارد، OE و پایـه T کـه OE را بـه پایـه ۲۶ یعنـی DEN وصـل مـی شـود و پایـه T بـه پایـه ۲۷ یعنـی DT/R وصل می شـود مطـابق شـکل ۱۲-۲:

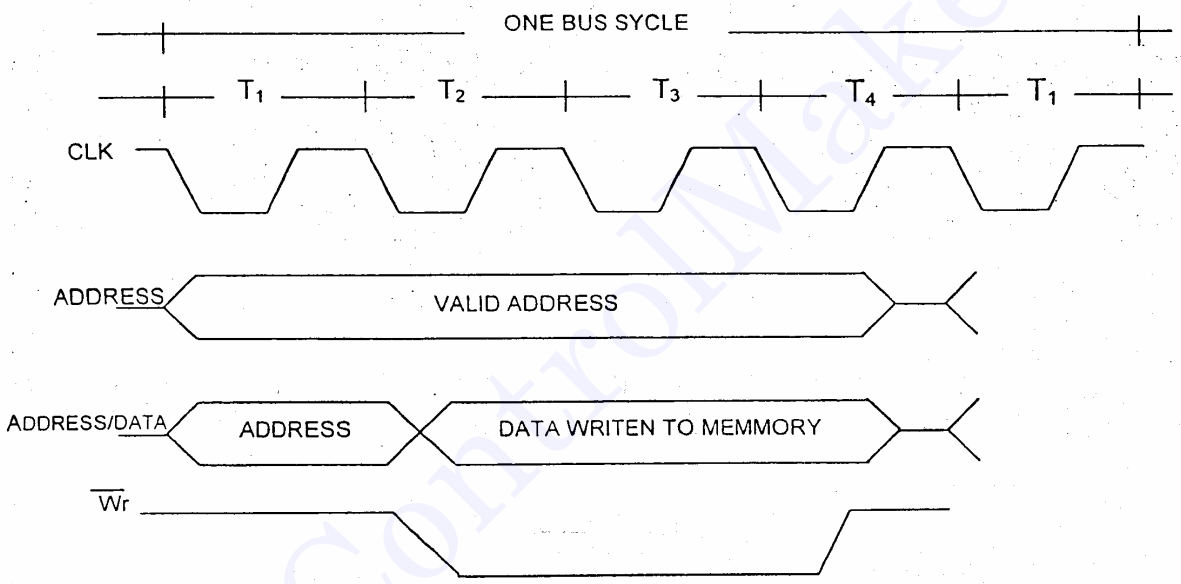


شکل ۱۲-۲-بلوک دیاگرام مینیمم سیستم 8086 در مد مینیمم

۷-۲- زمان بندی باس های ۸۰۸۶ (BUS Timing)

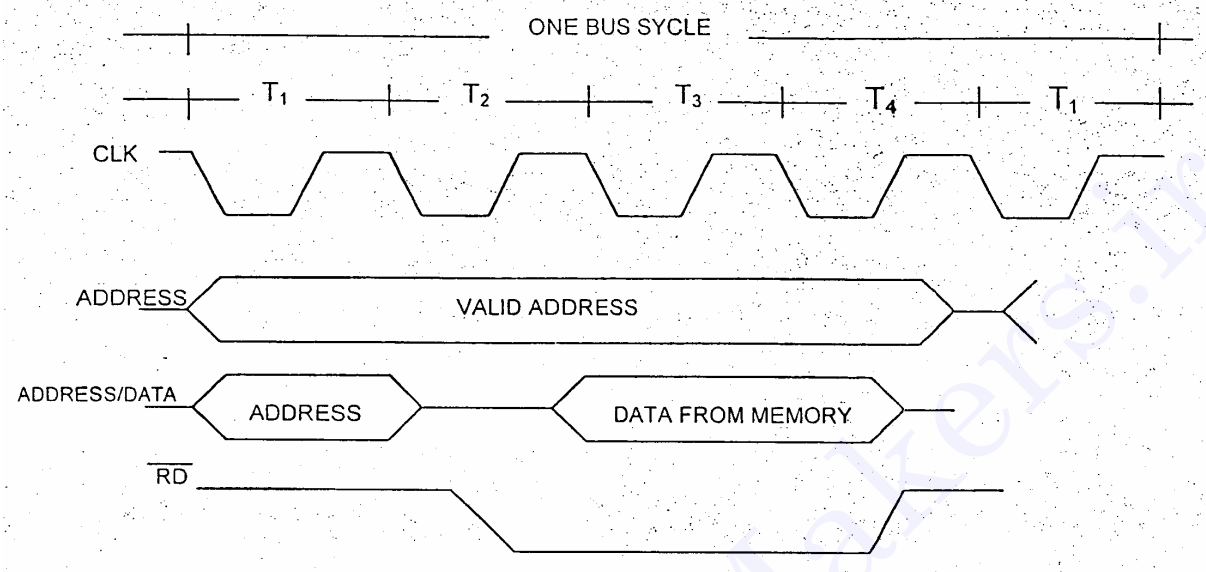
قبل از اینکه حافظه ای یا ورودی و خروجی برای 8086 یا 8088 انتخاب کنید باید از وضعیت زمان بندی باس آن مطلع باشید. در این قسمت نگاهی به زمان بندی باس هنگام دو عمل خواندن و نوشتن خواهیم داشت.

سه باس دیتا، آدرس و کنترل 8086 دقیقاً مثل باس های سایر میکروپروسسورها عمل می کند. برای نوشتن دیتا در حافظه به فرم ساده زمان بندی شکل ۱۳-۲ نگاه کنید. میکروپروسسور آدرس را روی آدرس باس قرار می دهد، سپس دیتائی که قرار است در حافظه نوشته شود روی دیتا باس قرار می دهد. سیگنال کنترلی $\overline{Wr}$ را صادر می کند. (در ۸۰۸۶ خط $M/\overline{IO}=1$ و در ۸۰۸۸ $IO/\overline{M}=0$ می شود)



شکل ۱۳-۲ سیکل باس برای نوشتن در حافظه بطور خلاصه

اگر قرار است دیتا از حافظه خوانده شود شکل ۱۴-۲ را ملاحظه فرمائید. میکروپروسسور آدرس را روی خطوط آدرس قرار می دهد، یک سیگنال RD صادر می کند، سپس دیتای موجود در دیتا باس که از حافظه خارج شده است را قبول می کند.



شکل ۱۴-۲-سیکل باس برای خواندن از حافظه بطور خلاصه

۱-۷-۲- زمان بندی عمومی

۸۰۸۶ حافظه و I/O را بصورت پریودیک بکار می گیرد که این زمان سیکل باس (BUS CYCLE) نامیده می شود، که برابر ۴ پالس ساعت است. اگر فرکانس پالس ساعت ۵ مگاهرتز باشد، زمان تکمیل عملیات خواندن و نوشتن این میکروپروسسور ۸۰۰ نانوثانیه می شود. یعنی میکروپروسسور در هر ثانیه ۱/۵ میلیون دفعه بین I/O و یا حافظه می خواند و می نویسد. صف داخل میکروپروسسور می تواند ۲/۵ میلیون دستورالعمل را در ثانیه اجرا نماید. (با این واحد سرعت میکروپروسسورها مقایسه می شود MIPS) البته انواع دیگر این میکروپروسسور با سرعتی بالاتر از این مقدار عمل می کنند. در اولین پالس ساعت هر سیکل که T_1 نام دارد آدرس روی خطوط آدرس یا بعبارت دیگر روی خطوط دیتا و آدرس مالتی پلکس شده ارسال می گردد.

در پایان حالت T_1 سیگنالهای کنترلی ALE, $DT/\overline{R}$, $M/\overline{IO}$ صادر می شوند. در زمان دومین پالس(T_2) میکرو سیگنالهای کنترلی $\overline{Wr}$, $\overline{DEN}$ یا $\overline{RD}$ را صادر می کند. اگر سیکل نوشتن است دیتائی که قرار است در حافظه یا I/O نوشته شود روی خطوط دیتا باس قرار داده می شود. سیگنال کنترلی $\overline{DEN}$ بافر دیتا را باز می کند و حافظه یا I/O دیتای ارسال شده را دریافت می کنند.

در خلال T_3 حافظه و I/O به اندازه کافی وقت دارند که دیتای روی دیتا باس را دریافت کنند. یک اتفاق مهمی که در خلال T_3 رخ می دهد سیگنال Ready نمونه برداری می شود اگر Ready

دارای ولتاژ یک منطقی باشد بعد از T_3، سیگنال T_4 صادر می شود اما اگر Low باشد پالس ساعت بعدی TW خواهد بود.

در T_4 تمام سیگنالها غیر فعال می شوند و آماده تدارک سیکل بعدی می شوند، همچنین زمان نمونه برداری از اتصال دیتا باس توسط میکروپروسسور است که دیتای خوانده شده از حافظه یا I/O را بردارد. بعلاوه در این لحظه سیگنال $\overline{Wr}$ که دارای یک لبه بالا رونده است (زیرا به یک منطقی برمی گردد) دیتای ارسالی میکرو را به حافظه یا I/O منتقل می کند.

با توجه به این زمانها شما باید برای میکروپروسسور حافظه ای انتخاب کنید که از نظر زمان دسترسی با میکروپروسسور مشکلی نداشته باشد. در واقع بعد از T_1 تا پایان T_4 زمان مناسبی برای دسترسی به حافظه می تواند باشد تقریباً ۶۰۰ نانوثانیه البته ۱۱۰ نانوثانیه بعد از T_1 زمان لازم است تا آدرس ظاهر شود که باید از ۶۰۰ نانوثانیه کم شود. مضافاً اینکه ۳۰ نانوثانیه قبل از اتمام T_4 باید دیتا در خروجی باشد، یعنی زمان دسترسی به حافظه ۴۶۰=۳۰-۱۱۰-۶۰۰ نانوثانیه می شود.

سؤالات دوره ای فصل دوم

۱-میکروپروسسور۸۰۸۶ دارای چند خط آدرس و چند خط دیتا است؟

۲-میکروپروسسور کمکی ۸۰۸۶ کدام است؟

۳-میکروپروسسور۸۰۸۶ چه مقدار حافظه را آدرس دهی می کند؟

۴-میکروپروسسور ۸۰۸۶ دارای چند خط کنترلی است؟

۵-سگمنت رجیسترهای ۸۰۸۶ چه نام دارند؟

۶-وقتی که میکروپروسسور POWER ON و یا RESET می شود از چه آدرسی شروع به اجرای برنامه می کند ؟ چرا؟

۷-چیپ ۸۰۸۶ چند پایه دارد؟

۸-پایه READY به چه منظوری تدارک دیده شده است؟

۹-ALE چه عملی انجام می دهد؟

۱۰- سیگنالهای وضعیت S_4,S_3 چه اطلاعاتی در اختیار می گذارند؟

۱۱- با استفاده از سیگنالهای وضعیت S_4,S_3 چگونه می توان ۴ مگا بایت حافظه به میکروپروسسور وصل کرد؟

۱۲- BIU چه وظایفی بعهده دارد؟

۱۳- پیش واکشی (Prefetch) در میکروپروسسور ۸۰۸۶ یعنی چه؟

۱۴- سه گروه ثبات های ۸۰۸۶ را شرح دهید؟

۱۵- مزایای سگمنت بندی حافظه چیست؟

۱۶- آیا می توان ۴ سگمنت حافظه در فضای 64 KB باشد؟

۱۷- چند پرچم برای بیان وضعیت سیستم در میکروپروسسور ۸۰۸۶ وجود دارد نام ببرید؟

۱۸- سه پرچمی که بیت های کنترلی ریزپردازنده هستند چه نام دارند؟ نام ببرید.

۱۹- سه وظیفه IC شماره 8284 که مولد پالس ساعت سیستم است کدامند؟

۲۰- تکنیک DMA یعنی چه؟ و چه پایه هائی از ۸۰۸۶ در این تکنیک نقش دارند؟

فصل سوم

«واسط حافظه و ورودی/خروجی MEMORY& I/O INTERFACE»

هر سیستم کامپیوتری اعم از اینکه ساده یا پیچیده باشد نیاز به حافظه دارد. ۸۰۸۶ نیز از این نیاز مثتثنی نیست. برای طراحی یک مینیمم سیستم معمولاً دو نوع حافظه بعنوان حافظه اصلی مورد استفاده قرار می گیرند، حافظه فقط با قابلیت خواندن (ROM) و حافظه با قابلیت خواندن و نوشتن (R/W MEM) که اصطلاحاً به آن حافظه با دسترسی تصادفی (RAM) گفته می شود. در این فصل چگونگی متصل کردن حافظه های مذکور به ریزپردازنده مورد بررسی قرار می گیرد.

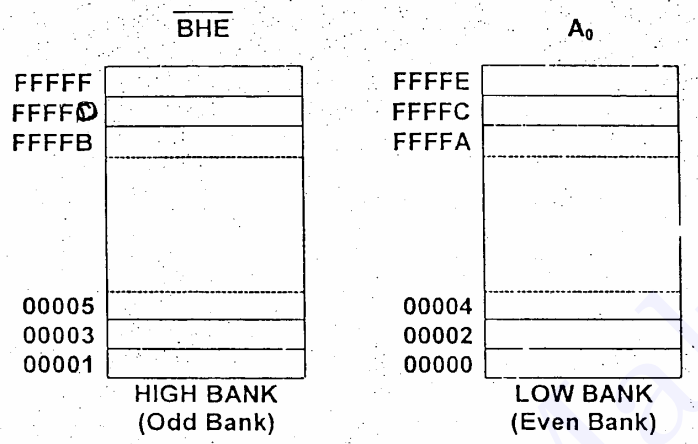
البته در اینجا فرض ما بر این است که خواننده محترم (دانشجویان عزیز) با انواع حافظه های ROM و DRAM و SRAM آشنائی کامل دارند.

۱-۳-دکودکردن آدرس :

برای دسترسی به یک محل معین یک حافظه توسط ریزپردازنده لازم است که آدرس میکرو دکود شود تا برای یک محل یک عدد معینی اختصاص داده شود. اگر از روش دکود کردن آدرس استفاده نشود، فقط یک IC حافظه می توان به ریزپردازنده وصل کرد. که استفاده از تمام محل های قابل آدرس دهی توسط ریزپردازنده غیر ممکن می شود.

قبل از اینکه به اتصال حافظه به ریزپردازنده پرداخته شود لازم است که ذکر گردد اتصال حافظه به ریزپردازنده ۸۰۸۶ متفاوت از وصل کردن حافظه به ریزپردازنده ۸۰۸۸ است زیرا این دو ریزپردازنده در سه مطلب با هم اختلاف دارند، اول اینکه دیتا باس ۸۰۸۶ شانزده بیتی است در حالیکه دیتا باس ۸۰۸۸ هشت بیتی است، دوم اینکه پایه $\overline{M/IO}$ که در ۸۰۸۶ بحث شد در ۸۰۸۸ بصورت $\overline{IO/M}$ می باشد و سوم اینکه پایه $\overline{BHE}$ که در ۸۰۸۶ وجود دارد در ۸۰۸۸ چنین پایه وجود ندارد.

چون ریزپردازنده ۸۰۸۶ دارای ۱۶ خط دیتاست و اینکه هم قادر است بصورت ۸ بیتی کار کند و هم بصورت ۱۶ بیتی لذا باید حافظه آن به دو بانک فرد و زوج و یا بانک بالا (HIGH) و بانک پایین (LOW) تقسیم شود. مطابق شکل ۱-۳



شکل ۱-۳- نمایش بانک بالا(فرد) و بانک پایین (زوج) در ریزپردازنده ۸۰۸۶

ریزپردازنده با استفاده از پایه A_0 و $\overline{BHE}$ مطابق جدول ۱-۳ قادر است به هر یک از بانکها به تنهایی (وقتی که ۸ بیتی کار می کند) و یا به هر دو بانک (وقتی که بصورت ۱۶ بیتی کار می کند) دسترسی پیدا کند.

جدول ۱-۳ : جدول صحت A_0 و $\overline{BHE}$

BHE	A_0	حالت دسترسی
0	0	هر دو بانک فعال است
0	1	بانک بالا(فرد) فعال است
1	0	پانک پایین (زوج) فعال است
1	1	هیچکدام فعال نیستند

ملاحظه می فرمائید که این ریزپردازنده 2^{20}=1M بایت محل حافظه را آدرس دهی می کند بعبارت دیگر 16× 512K مقدار حافظه را آدرس دهی می کند.

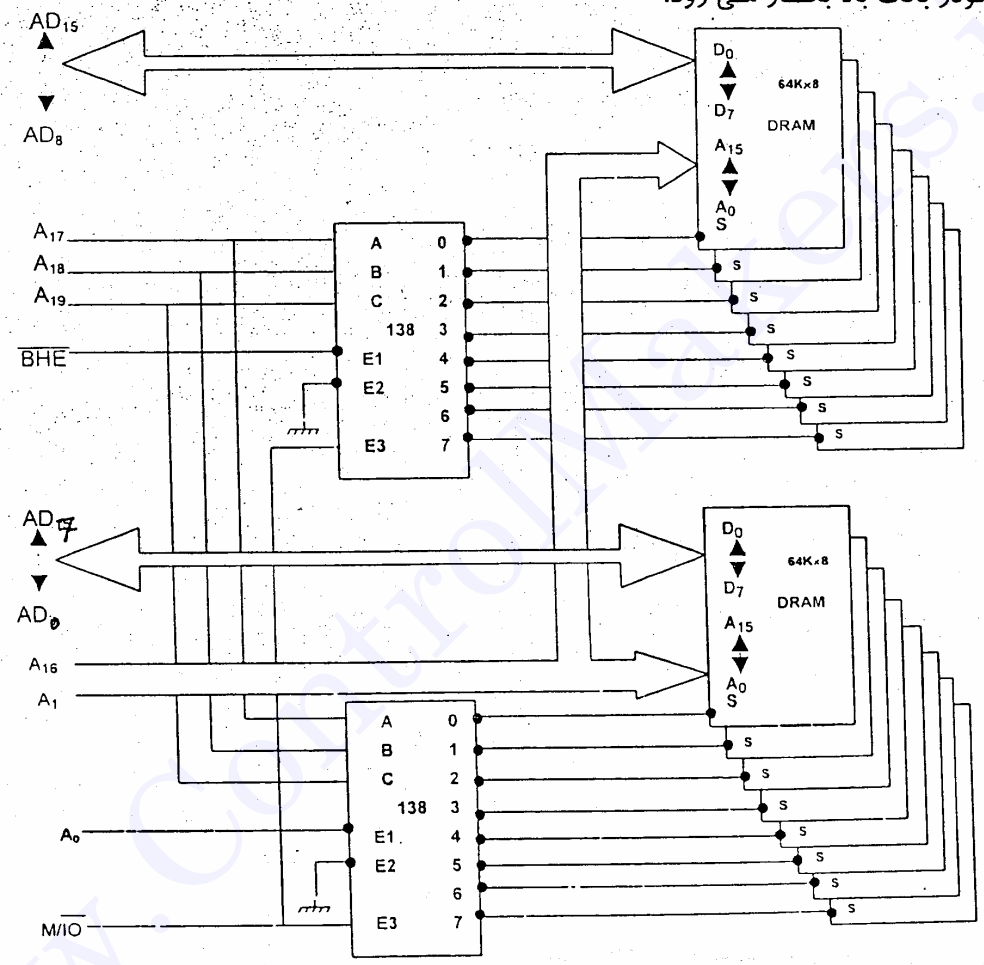
به دو روش می توان بانکها را انتخاب کرد :

۱-یک سیگنال $\overline{Wr}$ جداگانه برای هر بانک اختصاص داد.

۲-یک دکودر جداگانه برای هر بانک اختصاص داد.

روش دوم توصیه می شود. زیرا مطمئن ترین راه برای آدرس دهی است و قابل فهم تر نیز می باشد. لذا با استفاده از دو دکودر 74LS138 دو بانک به ظرفیت 1M×8 به ریزپردازنده مطابق شکل ۲-۳ متصل می کنیم.

هر بانک ۸ IC، 64K×8 دارد که هر IC نیاز به ۱۶ خط آدرس و ۸ خط دیتا دارد، زیرا 2^{16}=64K از A_1 تا A_{16} شانزده خط آدرس است که وارد هر IC می شود هم در بانک بالا و هم در بانک پایین و سه خط A_{17}، A_{18}، A_{19} وارد دکودرهای بانکها شده که هر کدام از دکودرها ۸ خروجی دارند که ۸ IC را انتخاب می کنند. A_0 برای انتخاب دکودر بانک پایین و $\overline{BHE}$ برای انتخاب دکودر بانک بالا بکار می رود.

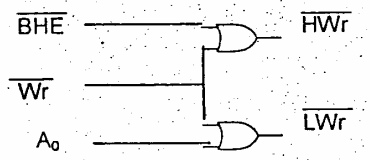


شکل ۲-۳- متصل کردن یک مگا بایت حافظه به ریزپردازنده ۸۰۸۶

توجه داشته باشید که A_0 ریزپردازنده داخل ICهای حافظه نمی رود بلکه به E دکودر وصل می شود، پس باید A_1 ریزپردازنده، به IC A_0 های حافظه و A_2 به A_1 و الی آخر وصل نمود.

روش اول مؤثرتر از روش قبلی است که تشریح شد. این تکنیک فقط به یک دکودر نیاز دارد و یک محل ۱۶ بیتی را انتخاب می کند، روشی است که از نظر اقتصادی بصرفه و تعداد المان کمتری بکار می گیرد.

شکل ۳-۳ چگونگی تولید دو سیگنال کنترلی خواندن را نشان می دهد که با استفاده از دو گیت منطقی OR با استفاده از IC، 74LS32 طرح شده است. که یکی برای انتخاب بانک بالا و دیگری برای انتخاب بانک پایین کاربرد دارد.



شکل ۳-۳ تولید در سیگنال کنترلی خواندن برای بانک بالاو پایین

حال با ذکر یک مثال بحث را دنبال می کنیم. فرض کنید بخواهیم به یک ریزپردازنده ۸۰۸۶ حافظه با قابلیت فقط خواندن (ROM) به میزان 32K×16 از آدرس F0000 تا FFFFF که از ICهای 2764 که هر کدام 8K×8 حافظه EPROM هستند وصل نمائیم، بعلاوه 8K×16 حافظه از نوع RAM که از آدرس 00000H تا 03FFFH را اشغال نماید با استفاده از ICهای 4016 که هر کدام 2K×8 هستند وصل نمائیم مطلوبست رسم بلوک دیاگرام مدار درصورتیکه از تکنیک یک سیگنال کنترلی خواندن برای هر بانک استفاده کنیم.

شکل ۴-۳ بلوک دیاگرام رسم شده مدار را نشان می دهد. بانک بلای RAM تشکیل شده از ۴ عدد IC، 4016 که 2K×8 هستند مجموعاً 8K×8 می شودعیناً به همین ترتیب 8K×8 هم بانک پایین حافظه RAM می باشد. ICهای 2K×8 یازده خط آدرس دارد از A_0 تا A_{10} که به A_1 تا A_{11} خطوط ریزپردازنده وصل می شوند همزمان بانک بالا و پایین را با دکودر 2×4، 74LS139 IC فعال می کنیم خطوط A_{12} و A_{13} میکرو وارد دکودر می شود چهار خروجی دکودر برای 8K×16 حافظه مذکور زمانی فعال می شود که بقیه خطوط آدرس (A_{19}, A_{18}, A_{17}, A_{16}, A_{15}, A_{14}) که همگی در صفر هستند با NOT کردن آنها و NAND کردن آنها با سیگنال $M/\overline{IO}$ تولید یک صفر منطقی می کند تا دکودر مذکور فعال شود، لذا از آدرس00000H~03FFFH (یعنی تمامی خطوط A_0 تا A_{13} یک می شوند) حافظه RAM مورد نظر تکمیل می شود.

حال به بخش ROM نظر افکنید ۴ عدد IC، 2764 برای بانک بالا (هر کدام 8K×8) و ۴ عدد از همان IC برای بانک پایین انتخاب شده است، یعنی 32K×8 بانک بالا و 32K×8 بانک پایین که با هم تشکیل حافظه 32K×16 را می دهند، پایه های آدرس هر IC از A_0 تا A_{12} می باشد که به پایه های A_1 تا A_{13} ریزپردازنده وصل می شوند، A_{14} و A_{15} به یک دکودر دیگر 74LS139 وصل می شوند تا ICهای8K×8 را فعال کند. چون قرار است این حافظه ها از آدرس F0000 شروع شوند پس چهار خط باقیمانده آدرس یعنی A_{19}, A_{18}, A_{17}, A_{16} باید یک باشند تا با $M/\overline{IO}$ بتوانند دکودر مربوط به ROM را فعال کنند.

به این ترتیب حافظه ها همانگونه که صورت مسئله خواسته بود در آدرسهای مورد نظر قرار گرفته اند ولی هر وقت آدرس دهی شوند یک محل ۱۶ بیتی فعال می شود. ممکن است نیاز باشد ما حافظه را بصورت ۱۶ بیتی مورد استفاده قرار دهیم که در این حالت باید هم A_0 و هم $\overline{BHE}$ فعال شوند یعنی $\overline{Lwr}$, $\overline{Hwr}$ هر دو فعال شوند، ممکن هم است که حافظه بصورت ۸ بیتی مورد استفاده قرار بگیرد که این مشکل با تهیه $\overline{HWr}$ و $\overline{LWr}$ حل می شود. هر کدام فعال شوند درآن بانک مربوطه نوشته می شود.

سوالی که ممکن است به ذهن برسد اینکه ، ..... هنگام خواندن خافظه چه باید کرد ؟ هنگام خواندن مشکل ندارید زیرا ریزپردازنده در هر سیکل باس فقط بایت مورد نظر را از روی دیتا باس برمی دارد. حتی اگر ۱۶ بیت دیتا روی دیتا باس باشد بایت مورد نظر خوانده می شود. مگر اینکه هدف ریزپردازنده خواندن دیتای ۱۶ بیتی باشد.

نکته دیگری که باید توجه کرد سیگنال فعال کننده دکودر EPROM به منزله سیگنال RDY به ریزپردازنده ارسال می شود. زیرا معمولاً EPROM به یک TW نیاز دارد تا اطلاعات در خروجی اش ظاهر شود شکل ۴-۳ کلیه موارد صحبت شده را بوضوح نشان می دهد.

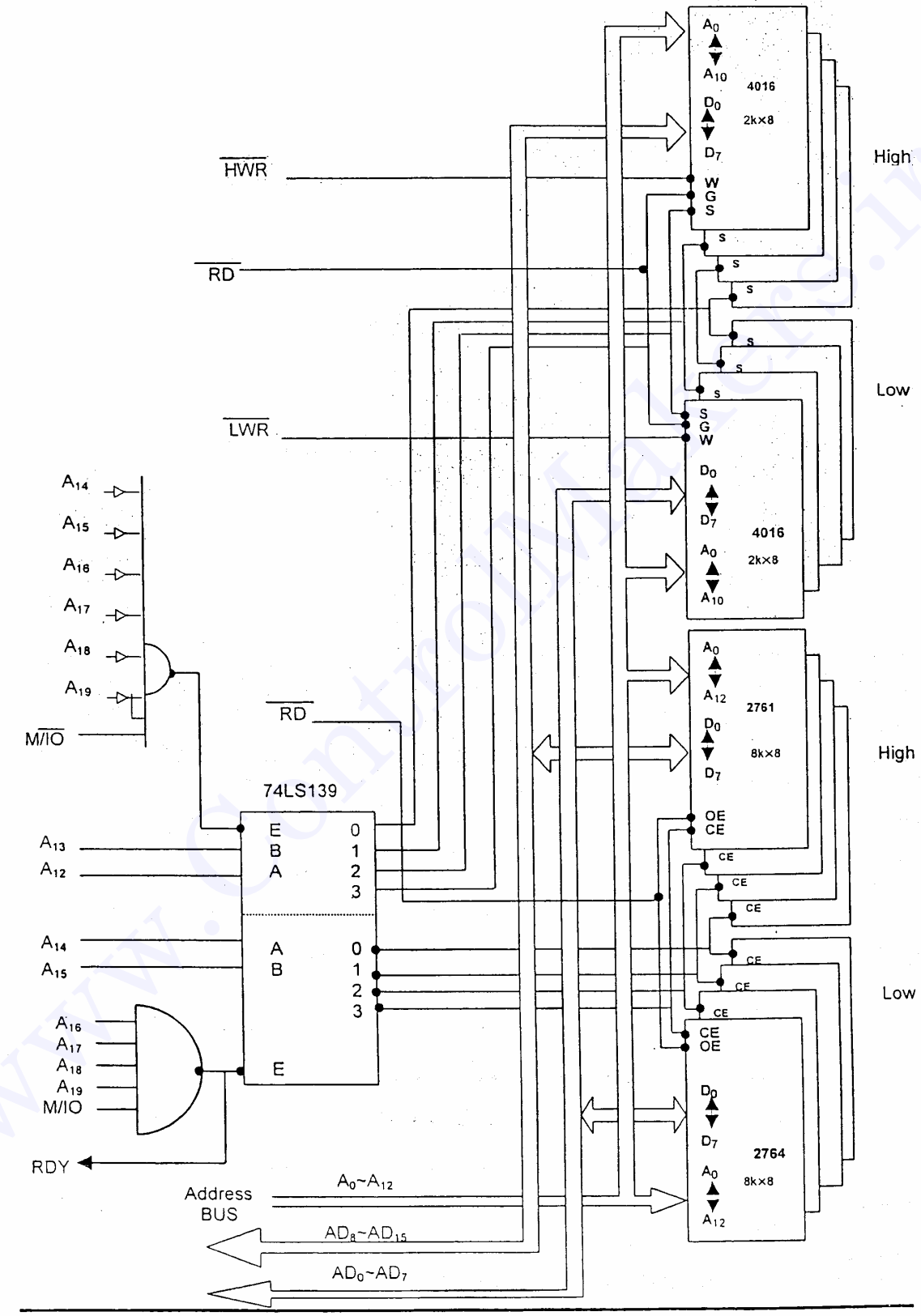
۲-۳-ارتباط یک ریزپردازنده با دنیای بیرون

یک ریزپردازنده ساخته شده تا بوسیله اطلاعاتی که دریافت می کند مسائلی را حل نماید بدیهی است باید قادر باشد با بیرون ارتباط برقرار کند. در این بخش روش های اصلی ارتباط موازی و سری بین انسان و ریزپردازنده معرفی می شوند. ابتدا واسط های اصلی I/O معرفی می شوند و درباره دکود کردن آدرسهای ورودی و خروجی بحث می شود سپس راجع به جزئیات ارسال و دریافت بصورت موازی و سری صحبت خواهد شد.

در ابتدا کلیه دستورات ورودی و خروجی ریزپردازنده ۸۰۸۶ معرفی می شوند. مفهوم I/O مستقیم یا ایزوله شده بیان می گردد و اتصال I/O به سیستم در قالب حافظه (MEMORY-MAPPED I/O) را مطرح خواهیم کرد.

۳-۳-دستورات ورودی و خروجی ۸۰۸۶

ریزپردازنده ۸۰۸۶ فقط یک دستور برای وارد کردن اطلاعات داردبنام IN و یک دستور برای خارج کردن اطلاعات از ریزپردازنده دارد بنام OUT ، هر کدام از دستورات OUT, IN به چهار صورت بکار گرفته می شوند. دو صورت از آن آدرس ۸ بیتی است و دیتا ۸ بیتی یا ۱۶ بیتی منتقل می شود و دو صورت دیگر آدرس ۱۶ بیتی و دیتا۸ یا ۱۶ بیتی منتقل می شود. جدول شماره ۲-۳ هشت دستور این ریزپردازنده لیست شده است.



جدول ۲-۳ انواع مختلف کاردبرد دستورات ورودی و خروجی

INSTRUCTION	عرض دیتا	شرح دستور
IN AL, D8	۸ بیت	یک بایت از یک I/O که آدرس ۸ بیتی دارد می خواند
IN AL, DX	۸ بیت	یک بایت از یک I/O که DX آدرس آنرا می دهد می خواند
IN AX, D8	۱۶ بیت	۱۶ بیت از یک I/O که آدرس ۸ بیتی دارد می خواند
IN AX, DX	۱۶ بیت	۱۶ بیت از یک I/O که DX آدرس آنرا می دهد می خواند
OUT D8, AL	۸ بیت	یک بایت را در I/O یی که آدرس ۸ بیتی دارد می نویسد
OUT DX, AL	۸ بیت	یک بایت را در I/O یی که DX آدرس آنرا می دهد می نویسد
OUT D8, AX	۱۶ بیت	۱۶ بیت را در I/O یی که آدرس ۸ بیتی دارد می نویسد
OUT DX, AX	۱۶ بیت	۱۶ بیت را در I/O یی که DX آدرس آنرا می دهد می نویسد

هر دو دستور فوق دیتا را بین ریزپردازنده و دستگاه ورودی و خروجی مبادله می نمایند. آدرس پورت یا بصورت فوری (Immediate Addressing) یا در داخل رجیستر ۱۶ بیتی DX قرار دارد. شرکت اینتل آدرس ۸ بیتی را آدرس ثابت (Fixed) شده می گویند چون هنگام نوشتن برنامه تعیین شده ولی آدرس ۱۶ بیتی که درون رجیستر DX است هر لحظه امکان تغییر دارد.

وقتی که دیتا توسط دستورات OUT, IN منتقل می شود عدد آدرسی که ریزپردازنده روی خط آدرس می گذارد، شماره پورت نیز گفته می شود. دستگاه خارجی عدد روی آدرس باس را دکود می کند تا آن پورت مورد نظر فعال شود. اگر عدد آدرس ۸ بیتی باشد روی A_0 تا A_7 قرار می گیرد، اگر ۱۶ بیت باشد روی A_0 تا A_{15} قرار می گیرد.

۴-۳- I/O ایزوله شده و I/O در قالب حافظه (MEMORY-MAPPED I/O)

دو روش کاملاً مستقلی برای اتصال I/O به ریزپردازنده ۸۰۸۶ وجود دارد I/O ایزوله شده و I/O در قالب حافظه. اگر ورودی و خروجی در قالب I/O ایزوله شده معرفی شوند ما می توانیم از تمام ظرفیت ریزپردازنده برای حافظه استفاده کنیم. سیگنال M/IO معنی خودش را دارد. مطالبی که در بالا برای I/O گفته شد صادق خواهد بود از دستورات OUT, IN استفاده می کنیم و عدد آدرس شماره پورت خواهد بود.

اما اگر I/O در قالب حافظه معرفی شود. دیگر مجاز نیستید از دستورات OUT, IN استفاده کنید، باید تصور کنید که اطلاعات بین ریزپردازنده و حافظه مبادله می شود یعنی از دستورات Write, Read باید استفاده کرد. شما نمی توانید از تمام ظرفیت حافظه استفاده کنید باید آن محلهائی که آدرس آنها را به پورت اختصاص داده اید بلاستفاده بمانند، زیرا I/O شما بجای آن محل ها معرفی شده یعنی ریزپردازنده I/O نمی شناسد، بلکه تصور می کند با حافظه سروکار دارد.

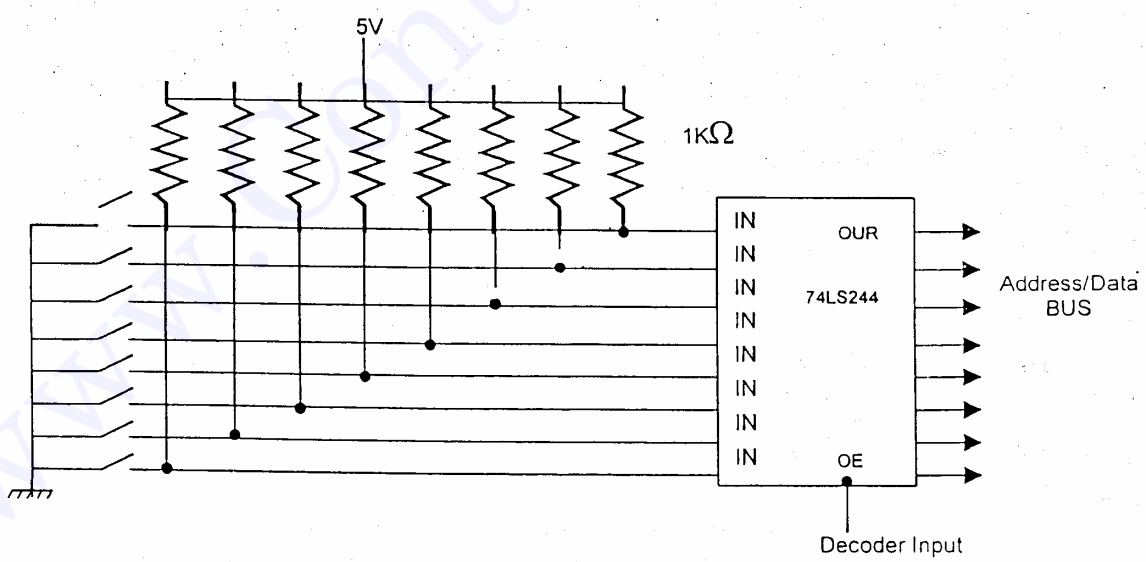
یک ورودی که در قالب I/O ایزوله شده به سیستم معرفی می شود، می تواند یک Latch با خروجی Tri-Stat باشد (۸ بیتی یا ۱۶ بیتی) مطابق شکل ۵-۳

توسط ۸ سوئیچ می توانید هر گونه دیتائی که بخواهید تنظیم کنید با ارسال آدرس پورت خروجی بافرهای Tri-State باز شده و دیتای تنظیم شده روی خط دیتا باس قرار می گیرد که ریزپردازنده دیتا را دریافت می کند.

دروازه خروجی دیتا از ریزپردازنده دریافت می کند. معمولاً باید آنرا توسط بافر برای دستگاهی نگهداری کند.

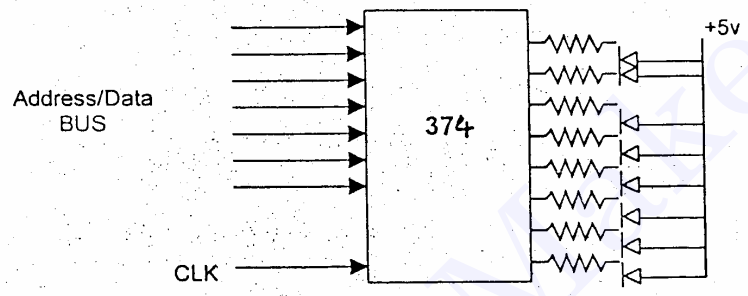
شکل ۶-۳ یک پورت خروجی را نشان می دهد. فرض کنید دیتا در بافر 74LS374 ارسال شود خروجی آن برای نمایش به ۸ لامپ LED وصل شده باشد.

زمانی که دستور OUT اجرای می شود محتوای AL یا AX روی دیتا باس ارسال می شود که ما توسط IC، 74LS374 که ۸ عدد Latch است دریافت می کنیم که در این جا فرض شده دکودر کلاک لازم را ارسال می کند و دیتا وارد Latch می شود. البته در این مدار هر دیتائی که صفر منطقی باشد LED مربوطه آن روشن می شود.



شکل ۵-۳- یک ورودی ۸ بیتی با استفاده از ۸ سوئیچ و ۸ بیت Latch با خروجی Tri-State

همانطور که مشاهده می کنید دکود کردن آدرس برای ورودی و خروجی شبیه دکود کردن آدرس برای حافظه است علی الخصوص زمانی که I/O در قالب حافظه معرفی شده است. اختلاف اصلی بین دکود کردن آدرس I/O و آدرس حافظه در تعداد خطوط آدرس است. زمانی که ریزپردازنده آدرس حافظه را می دهد از ۲۰ خط آدرس استفاده می کند (A_0 تا A_{19}) ولی وقتیکه ریزپردازنده I/O را آدرس دهی می کند از ۱۶ خط آدرس (A_0 تا A_{15}) و یا از ۸ خط آدرس (A_0 تا A_7) استفاده می کند. اختلاف دیگر در پایه $\overline{M/IO}$ می باشد.



شکل ۶-۳- یک دروازه خروجی

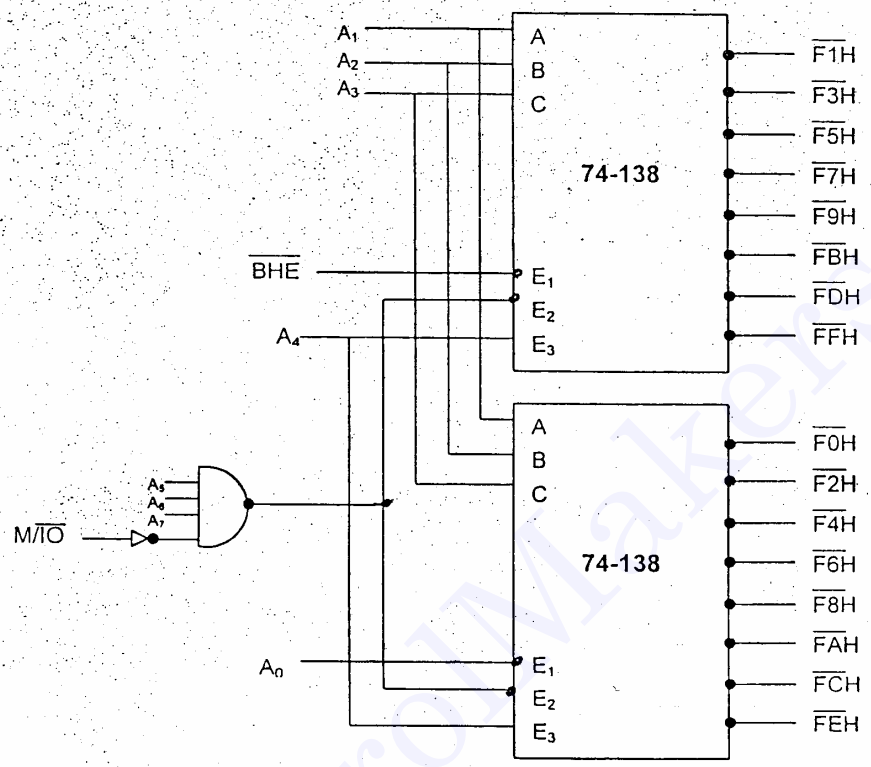
۱-۴-۳- ۸ بیت I/O با دو بانک

I/O هم می تواند مثل حافظه دارای دو بانک بالا و پایین باشد شکل ۷-۳ اتصال یک I/O را بصورت دو بانک بالا و پایین نشان می دهد.

زمانی که A_7, A_6, A_5 همگی یک باشند و $M/\overline{IO}=0$ باشد، پایه E2 دکودرهای 74138 را فعال می کند خط A_4 نیز E_3 را فعال می کند، پایه E_1 بانک بالا توسط $\overline{BHE}$ و بانک پایین توسط A_0 فعال می شود. با تغییرات A_1, A_2, A_3 شانزده آدرس F0 تا FF فعالی می شوند.

۲-۴-۳- اتصال I/O ۱۶ بیتی به 8086

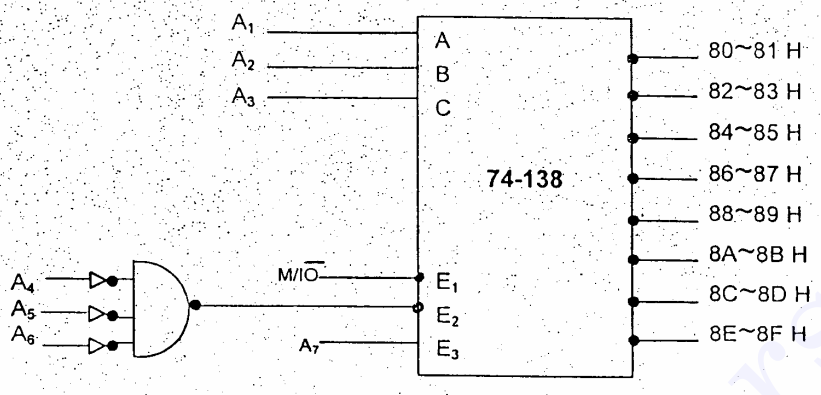
اگر قرار باشد I/O ۱۶ بیتی به 8086 متصل شود باید از هر دو BHE و A_0 چشم پوشی کرد زیرا آنها هر کدام برای ۸ بیت فعال می شوند. ورودی و خروجی ۱۶ بیتی هم زمانی مورد نیاز است که بخواهیم اطلاعات یک دستگاه مبدل آنالوگ به دیجیتال (ADC) و یا اطلاعات یک دستگاه مبدل دیجیتال به آنالوگ (DAC) را بخوانیم. زیرا این سیستم ها معمولاً ۱۰ تا ۱۲ بیت عرض دیتا دارند. شکل ۸-۳ دکود کردن ۸ پورت ۱۶ بیتی را نشان می دهد.



شکل ۷-۳- مداری که در بانک بالا و پایین را برابر ریزپردازنده با ۸ خط آدرس برای آدرس دهی I/O طراحی شده است

با اندکی دقت به فعال ساز دکودر 3×8 ، IC شماره 74LS138 متوجه می شوید که A_4 تا A_6 باید دارای سطح صفر منطقی و A_7 دارای سطح یک منطقی باشد از A_0 هم که قرار شد چشم پوشی شود هشت ترکیب $A_3A_2A_1$ آدرسهای مورد نظر را دکود می کنند.

تا این حدآشنائی با نحوه اتصال ورودی و خروجی به سیستم کافی است انشاءالله بعداً راجع به ICهای مناسب برای ۸۰۸۶ که بعنوان پورت ورودی و خروجی کاربرد دارند صحبت خواهد شد. لازم به ذکر است که گاهی I/O بصورت اینتراپت از ریزپردازنده سرویس می گیرند. که پس از بحث در باره اینتراپت و چگونگی سرویس دهی آن در این مورد هم صحبت خواهد شد.



شکل ۸-۳- مداری که قادر است ۸ پورت ۱۶ بیتی را آدرس دهی کند

سوالات و مسائل دوره ای :

۱-هدف از پایه های CE و یا CS که در ICهای حافظه تعبیه شده است چیست ؟

۲-هدف از پایه های OE که در ICهای حافظه تعبیه شده است چیست؟

۳-اختلاف در ریزپردازنده ۸۰۸۶ و ۸۰۸۸ را بیان کنید.

۴-میزان حافظه ای که ۸۰۸۶ آدرس دهی می کند چقدر است؟

۵-با استفاده از دو دکودر جداگانه برای هر بانک یک می نیمم سیستم طراحی کنید که به اندازه کافی حافظه داشته باشد.

۶-با استفاده از تهیه دو سیگنال Wr برای هر بانک بالا و پایین حداقل حافظه مورد نیاز یک مینیم سیستم را به ریزپردازنده ۸۰۸۶ وصل نمائید.

۷-روش های اتصال I/O به ریزپردازنده را شرح دهید.

۸-برای یک ریزپردازنده ۸۰۸۶ یک پورت ورودی و یک پورت خروجی وصل نمائید.

۹-دستورات وارد و خارج کردن اطلاعات در ۸۰۸۶ کدامند.

۱۰- به چند صورت دستور وارد کردن اطلاعات بکار می رود.

۱۱- به چند صورت دستور خارج کردن اطلاعات مورد استفاده قرار می گیرد.

۱۲- زمانیکه ریزپردازنده از آدرس ۱۶ بیتی برای یک دروازه ورودی یا خروجی استفاده می کند کدام رجیستر آدرس را دارد.

۱۳- توضیح دهید چرا شرکت اینتل آدرس های ۸ بیتی آدرس ثابت شده می گوید.

۱۴- چرا خروجی حافظه ها باید دارای وضعیت سوم (Hi-Z) باشند.

۱۵- آیا ورودی و خروجی را در ۸۰۸۶ نیز بصورت بانک بالا و پایین به سیستم معرفی می کنند؟

۱۶- معمولاً در چه مواقعی از ورودی و خروجی ۱۶ بیتی استفاده می شود.

فصل چهارم

«مدهای آدرس دهی ۸۰۸۶ (8086 ADDRESSING MODES)»

کار کردن موثر و مفید با یک سیستم و یا نوشتن برنامه خوب برای یک سیستم نیازمند آشنا بودن با مدهای آدرس دهی آن ریزپردازنده برای هر دستور می باشد، در این فصل بحث را با بکار گرفتن دستور MOV که یکی از ساده ترین دستورات ۸۰۸۶ می باشد شروع می کنیم، دستور MOV یک بایت یا یک کلمه ۱۶ بیتی را بین ثبات ها و یا یک ثبات و حافظه انتقال می دهد.

۱-۴- مدهای آدرس دهی دیتا

با استفاده از دستور MOV مدهای رجیستری-فوری-مستقیم-رجیستری غیرمستقیم-پایه بعلاوه شاخص-رجیستری نسبی و پایه نسبی بعلاوه شاخص را مورد بررسی قرار می دهیم. در شکل ۱-۴ دستور MOV جهت انتقال دیتا از منبع به مقصد برای تمامی مدها مشخص شده است. چون جهت انتقال دیتا در ریزپردازنده های مختلف متفاوت است، توضیحاً یادآور می شویم که در دستور MOV AX, BX ثبات BX منبع است و AX مقصد یعنی :

AX ◄———— BX

ضمناً بخاط داشته باشدی که در این گونه عملهای انتقال دیتا، منبع دیتا هرگز تغییر نخواهد کرد. و محتوای مقصد همواره تغییر می کند.

۱-۱-۴-آدرس دهی مد رجیستری

در این مد یک بایت یا یک کلمه ۱۶ بیتی اطلاعات را از ثبات منبع به ثبات مقصد انتقال می دهد. بعنوان مثال دستور MOV AX, CX محتوای ثبات CX را به ثبات AX کپی می کند.

نوع مد	دستور	منبع	مقصد
		BX	AX
۱-رجیستری	MOV AX, BX	رجیستر	رجیستر
۲-فوری	MOV BL,3AH	3AH دیتا	BL رجیستر
۳- مستقیم	MOV 1434H,AX	AX رجیستر DS×10H+DI	1434H حافظه
۴- رجیستری غیر مستقیم	MOV [BX], AX	AX رجیستر DS×10H+BX	[BX] حافظه
۵- پایه بعلاوه شاخص	MOV [BX+SI], AX	AX رجیستر DS×10H+BX+SI	[BX+SI] حافظه
۶- نسبی رجیستری	MOV [BX+4], AX	AX رجیستر DS×10H+BX+4	[BX+4] حافظه
۷- رجیستری نسبی بعلاوه شاخص	MOV Array [BX+SI], AX	AX رجیستر DS×10H+Array+BX+SI	Array[BX+4] حافظه

شکل ۱-۴- مدهای آدرس دهی دیتا

۲-۱-۴- آدرس دهی سریع (فوری):

یک بایت یا یک کلمه ۱۶ بیتی دیتا را به رجیستر مقصد منتقل می کند. بعنوان مثال MOV AX, 1234H دیتائی که مقدار آن 1234 در مبنای ۱۶ را به ثبات AX منتقل می کند. توجه دارید که حرف H بدنبال هر عددی که آمد بیانگر آن است که دیتا در مبنای ۱۶ است.

۳-۱-۴- آدرس دهی مستقیم:

یک بایت یا یک کلمه ۱۶ بیتی را بین حافظه و یک ثبات انتقال می دهد که آدرس مورد نظر حافظه با دستورالعمل ذخیره می شود، بعنوان مثال دستور MOV AX, List محلی از حافظه که ۱۶ بیتی است و آدرس آن در List است به ثبات AX منتقل می کند.

۴-۱-۴-آدرس دهی رجیستری غیر مستقیم :

یک بایت یا یک کلمه ۱۶ بیتی را بین حافظه و یک ثبات انتقال می دهد که آدرس مورد نظر حافظه در ثبات دیگری است مثل دستور MOV AX, [BX] ، ۱۶ بیت اطلاعات را از محلی که آدرس آن در ثبات BX است به ثبات AX منتقل می کند.

۵-۱-۴- آدرس دهی پایه بعلاوه شاخص:

یک بایت یا یک کلمه ۱۶ بیتی بین حافظه و یک ثبات منتقل می شود که آدرس محل حافظه از جمع دو ثبات پایه و شاخص بدست می آید. مثل دستور MOV AX,[BX+SI] که ۱۶ بیت اطلاعات از محلی از حافظه که آدرس آن بصورت زیر محاسبه می شود به ثبات AX منتقل می نماید. AX ← M [DS×10H+BX+SI]

۶-۱-۴-آدرس دهی نسبی رجیستری:

در این مد نیز یک بایت یا یک کلمه بین رجیستر و حافظه منتقل می شود که آدرس محل مورد نظر حافظه از جمع یک رجیستر و عدد دیگری که از Displacement است بدست می آید.

مثل MOV AX, [BX+4] که محلی از حافظه به آدرس AX ← M [DS×10H+BX+4]

یا MOV AX, Array[SI] یعنی در AX بریزد محلی از حافظه که از رابطه زیر بدست می آید :

AX ← M [DS×10H+Array+SI]

۷-۱-۴-آدرس دهی نسبی پایه بعلاوه شاخص:

یک بایت یا یک کلمه بین رجیستر و حافظه انتقال داده می شود که آدرس حافظه از جمع محتوای رجیستر پایه و رجیستر ایندکس و عدد داده شده بدست می آید، مثل دستور :

MOV AX, Array[BX+DI]
AX ← M [DS×10H+Array+BX+DI]

۲-۴-آدرس دهی رجیستری (Register Addressing)

آدرس دهی رجیستری یکی از ساده ترین مدهای آدرس دهی در ۸۰۸۶ یا ۸۰۸۸ می باشد، این ریزپردازنده دارای ۸ ثبات ۸ بیتی است بنام های AH, AL, BH, BL, CH, CL, DH, DL و ثبات های ۱۶ بیتی به نام های AX, BX, CX, DX, SP, BP, DI, SI, CS, DS, SS, ES می باشد.

به غیر از ثبات های سگمنت که فقط در چند دستور خاص مثل MOV, PUSH, POP کاربرد دارند، مابقی ثبات ها می توانند هر کدام منبع یا مقصد باشند. نکته دیگر اینکه ظرفیت منبع و مقصد باید برابر باشد والا اسمبلر اعلام خطا می کند. در جدول ۱-۴ بعضی از دستورات از انواع

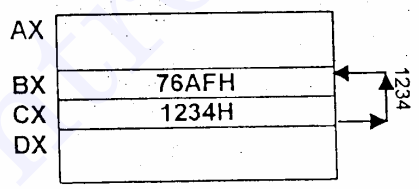
مختلف MOV لیست شده است. بدیهی است لیست کردن تمامی دستورات MOV که ۲۵۶ دستور می شود کار ساده نیست.

جدول ۱-۴- دستورات آدرس دهی رجیستری

Assembly Language	Operation
MOV AL,BL	BL → AL
MOV CH,CL	CL → CH
MOV AX,CX	CX → AX
MOV SP,BP	BP → SP
MOV DS,AX	AX → DS
MOV SI,DI	DI → SI
MOV DI,SI	SI → DI
MOV BX,ES	ES → BX
MOV CS,DS	Not allowed (segment-to-segment)
MOV BL,BX	Not allowed (Mixed Size)

در جدول ۱-۴- دقت کنید دو دستور آخر اجازه داده نشده است.

شکل ۲-۴ نحوه مبادله اطلاعات توسط دستور MOV BX, CX را نشان می دهد. توجه دارید که اطلاعات منبع تغییر نمی کند، اما اطلاعات قبلی مقصد از بین می رود و عدد هگز 1234h از محتوای CX بداخل BX کپی می شود.



شکل ۲-۴ انتقال دیتا از ثبات منبع به ثبات مقصد

۳-۴-آدرس دهی فوری (ImmediateAddressing)

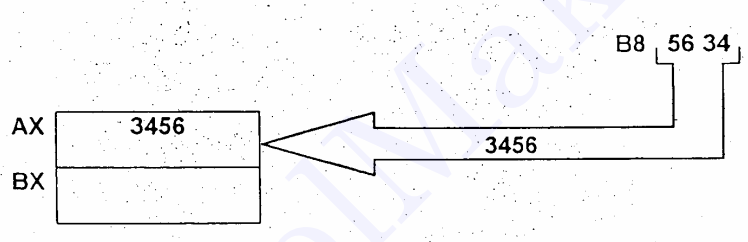
مد آدرس دهی دیگری که نسبتاً ساده است، زیرا دیتا بصورت یک بایت یا یک کلمه ۱۶ بیتی بلافاصله بعد از Op-Code در داخل حافظه ذخیره می شود. در بعضی از زبان های اسمبلی معمولاً قبل از دیتا علامت # را قرار می دهند مثل دستور MOV AX,#1654H .

البته در ریزپردازنده ۸۰۸۶ اگر علامت # را نیز نگذاریم صحیح است و مترجم اسمبلی آنرا ترجمه می کند. دیتا را به هر دو صورت اعشار (پایه ۱۰) و هگز (پایه ۱۶) در این دستور معرفی می کند. اگر در پایه ۱۰ بود بعد از دیتا هیچ علامتی ندارد اما اگر در پایه ۱۶ باشد باید حرف H آورده شود. جدول ۲-۴ مثالهایی از مد آدرس دهی فوری را با دستور MOV نشان می دهد.

Assembly Language	Operation
MOV BL,44	BL ← 2CH
MOV AX,44H	AX ← 0044H
MOV SI,0	SI ← 0000H
MOV CH,100	CH ← 64H
MOV SP,3000H	SP ← 3000H

جدول ۲-۴ مثالهایی از مد آدرس دهی فوری با دستور MOV

بعنوان مثال اگر دستور MOV AX,3456H را در نظر بگیرید، کد عملیاتی MOV AX عدد B8 می شود و این دستور در حافظه به شکل ۳-۴ ذخیره می شود. یعنی دستور سه بایت جا نیاز دارد.



شکل ۳-۴- ذخیره سازی دیتا در ثبات

با اجرای دستور عدد 3456 در AX کپی می شود.

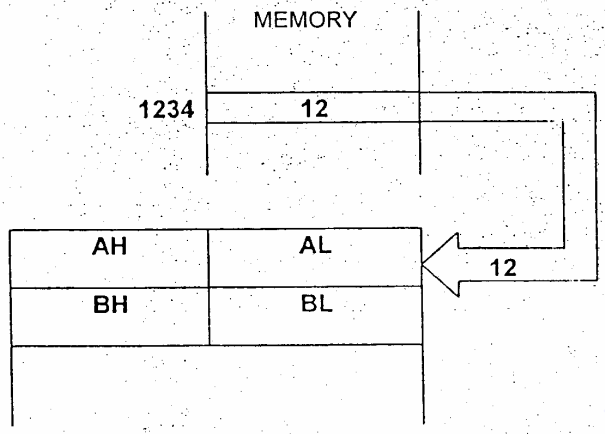
۴-۴- مد آدرس دهی مستقیم (Direct Addressing)

این مد آدرس دهی دو فرم کلی دارد. یکی آدرس دهی مستقیمی که فقط در دستور MOV بکار می رود و یا بطور اخص در دستور MOV بین حافظه و AX یا AL بکار می رود و فرم دیگر آنرا Displacement Addressing گویند و تقریباً با هر دستوری در ۸۰۸۶ یا ۸۰۸۸ بکار می رود.

الف : آدرس دهی مستقیم

وقتی که صحبت از آدرس دهی مستقیم می شود، دستورات LDA, STA که در ۸۰۸۵ معرفی شده اند و ۸۰۸۶ نیز آنها را براحتی اجرا می کند را در بر می گیرد. هر زمانی که دیتا بین حافظه و AL و یا حافظه AX مبادله شود این مد کاربرد دارد. دستور MOV AL,Data که در ۸۰۸۶ کاربرد دارد یک بایت دیتا را از یک محل حافظه که Data آدرس آن محل می باشد بداخل AL کپی می کند این دستور سه بایت طول دارد بعضی وقت بشکل ۴-۴ نوشته می شود.

با اجـرای دسـتور MOV AL,[1234H] اگـر فـرض کنیـم در آدرس ۱۲۳۴ حافظـه عـدد ۱۲ باشـد ایـن مقدار در AL کپی می شـود. مطـابق شـکل زیـر :



شکل ۴-۴- طریقه انتقال دیتا از محل حافظه به AL

جدول ۳-۴ چهار صـورت ممکنـه ایـن مـد را نمـایش مـی دهـد. مجـدداً یـادآور مـی شـود کـه دسـتور MOV فقط بین AL یـا AX بـا حافظـه کـاربرد دارد.

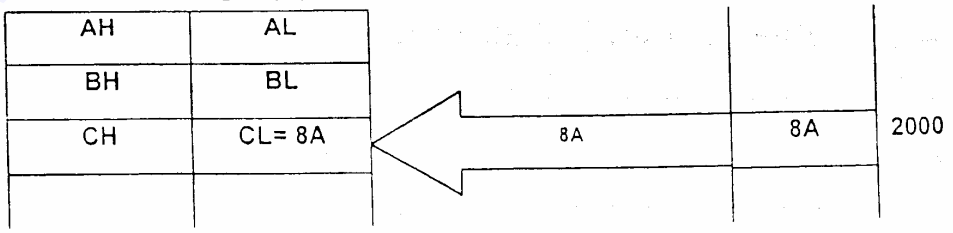
جدول ۳-۴- چهار صورت ممکنه دستور MOV برای انتقال اطلاعات بین حافظه و AL یا AX

Assemble Language		Operation
MOV	AL,NUMB	یک بایت از حافظه بـه آدرس NUMB بـه AL کپی مـی شـود.
MOV	AX,COW	یک کلمه ۱۶ بیتـی از حافظـه بـه آدرس COW در AX کپـی مـی شـود.
MOV	NEW,AL	محتوای AL در محلـی از حافظـه بـه آدرس NEW ذخـیره مـی شـود.
MOV	THERE,AX	محتوای AX در محلـی از حافظـه بـه آدرس THERE ذخـیره مـی شـود.

ب: Displacement Addressing

در این مد دستور برخـلاف آدرس دهـی مسـتقیم کـه صحبـت شـد و ۳ بـایت طـول داشـت دارای طـول چـهار بـایتی اسـت. و اسـتفاده از آن خیلـی سـاده و اکـثراً ۸۰۸۶ یـا ۸۰۸۸ از آن اسـتفاده مـی کننـد. دسـتور MOV CL,[2000H] را در نظـر بگـیرید خیلـی شـبیه دسـتور MOV AL,[1234] است با این تفاوت که آن سـه بـایت بـود و ایـن ۴ بـایت طـول دارد.

دستور قبلی مخصوص AL بـود در حالیکـه ایـن دسـتور بـرای تمـام ثبـات هـا کـاربرد دارد. اگـر فـرض کنیم محتوای محل ۲۰۰۰ حافظـه دارای دیتـای 8A باشـد. آنـرا در CL کپـی مـی کنـد.



مقایسه دو دستور فوق از نظر تعداد بایت ها و Opcode به شکل زیر می باشد.

MOV AL,[1230H] A03412 سه بایت

MOV CL,[2000H] BA0E0020 چهار بایت

جدول ۴-۴ تعدادی از دستورات MOV با فرم Displacement را نشان می دهد. البته تمام اینگونه دستورات 512 عدد می شود که امکان لیست کردن تمامی آنها وجود ندارد.

جدول ۴-۴- لیست تعدادی از دستورات MOV Displacement

Assembly Language	Operation
MOV CH,DOG	محتوای محلی از حافظه که با DOG در دیتا سگمنت مشخص شده در CH کپی می شود (مقدار عملی DOG توسط اسمبلی تعیین می شود)
MOV CH,[1000H]	محتوای محلی از حافظه که با آدرس 1000H مشخص شده در CH کپی می شود (البته سگمنت مربوطه DS خواهد بود)
MOV DATA,BP	محتوای ثبات BP کپی می شود در محلهائی از حافظه به آدرس DATA و DATA+1 (در بخش DS)
MOV NUMBER,SP	محتوای ثبات SP در محلهائی به آدرس NUMBER و NUMBER+1 کپی می شود (ناحیه DS)

۵-۴- آدرس دهی رجیستری غیر مستقیم (Register InDirect Addressing)

در این مد یک محل از حافظه توسط یکی از ثبات های BX ,BP ,SI ,DI آدرس دهی می شود. بعنوان مثال اگر محتوای ثبات BX عدد 1000H باشد، دستور MOV AX,[BX] محتوای محل ۱۰۰۰ حافظه در DS را در AX کپی می کند.

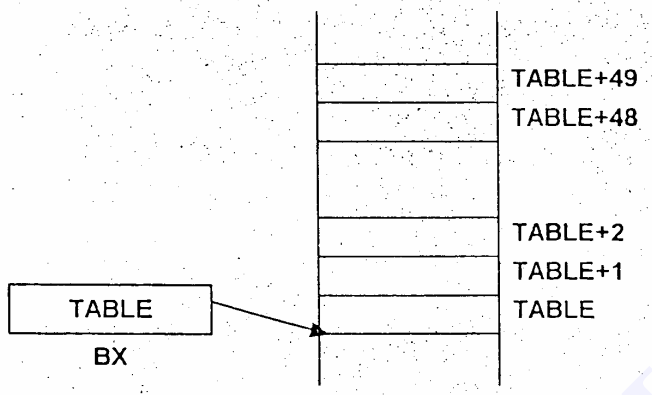
بعضی از دستوراتی که از مد آدرس دهی رجیستری غیر مستقیم استفاده می کند در جدول شماره ۵-۴ لیست شده اند.

جدول ۵-۴- لیست بعضی از دستورات آدرس دهی رجیستری غیر مستقیم

Assembly Language	Operation
MOV CX,[BX]	یک کلمه ۱۶ بیتی از محلی از حافظه در DS در CX کپی می کند
MOV [BP],DL	محتوای DL را در محلی از حافظه در SS که با BP آدرس دهی شده ذخیره می کند
MOV [DI],BH	محتوای BH را در محلی از حافظه در DS که با DI آدرس دهی شده کپی می کند
MOV [DI],[BX]	محتوای حافظه به حافظه اجازه داده نشده، مگر اینکه دستور رشته ای استفاده شود

نکته ای که باید توجه کرد این است که وقتی از ثبات های BX ,SI ,DI استفاده می کنید آدرس محلی از حافظه در ناحیه دیتا سگمنت DS را می دهید و هنگامیکه از ثبات BP استفاده می کنید آدرس محلی از حافظه در ناحیه استک سگمنت SS را می دهید.

نکته دیگر اینکه معمولاً از مد غیر مستقیم رجیستری وقتی استفاده می شود که ارجاع به یک جدول از داده ها باشد. فرض کنید شما مجبورید یک جدول از دیتائی که از یک ولت متر دیجیتالی می خوانید تشکیل دهید. مثلاً ۵۰ نمونه خوانده شده و مطابق شکل (۴-۵) که هم ۵۰ دیتا را نشان می دهد و هم ثبات BX برای مشخص کردن هر محل جدول استفاده شده است.



شکل ۵-۴

برای تکمیل کردن مثال شما نیاز دارید با دستور MOV محل TABLE را در BX بریزید بعد با استفاده از مد آدرس دهی رجیستری غیر مستقیم مقادیر را به ترتیب در جدول وارد کنید، مثال زیر این مطلب را نشان می دهد.

```
;EXAMPLE 4-1
;Instructions That Read 50 Bytes of Data From DATA-PORT and Stores Them in a TABLE

BEHIN  :  MOV    BX,Offset TABLE  ;  Address TABLE
          MOV    CX,50            ;  Load Loop Connt
AGAIN  :  IN     AL,DATA-PORT     ;  Get DATA
          MOV    [BX],AL          ;  Save DATA
          INC    BX               ;  BX ←——— BX+1
          LOOP   AGAIN            ;  Repeat Until CX=0
            ⋮      ⋮
```

در مثال فوق کلمه Offset که به Directive معروف است به اسمبلر می گوید که BX را با افست آدرس TABLE پر کن و همچنین شماره CX هم با عدد۵۰ پر می شود، از خصایص دستور LOOP است که به برچسب مورد نظر برود تا CX=0 شود.

۶-۴-آدرس دهی پایه بعلاوه شاخص (Base Plus Index Addressing)

این مد خیلی شبیه مد آدرس دهی رجیستری غیر مستقیم است. زیرا آدرس حافظه با جمع یکی از ثبات های پایه مثل BX یا BP با یک ثبات شاخص مثل DI یا SI بدست می آید. غالباً ثبات

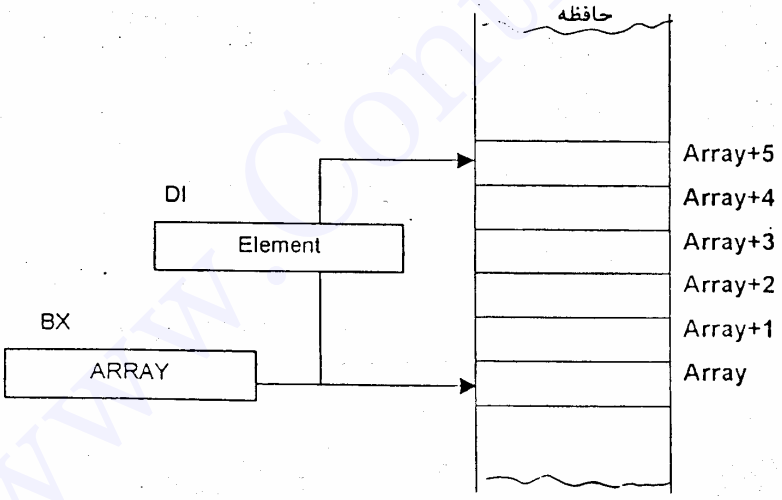
پایه آدرس شروع یک حافظه آرایه ای را دارد و ثبات شاخص آدرس نسبی را در آرایه نگهداری می کند مثل دستور : MOV DX, [BX+DI]

فرض کنید که : BX=1000H , DI=10H , DS=0100H باشد آدرس فیزیکی تولید شده عبارت است از 0100×10 +100+10=2010H لذا دیتای محل های 02010H و 02011H وارد ثبات DX می شوند.

جدول ۶-۴- مثال هائی از دستورات با مد آدرس دهی ثبات پایه بعلاوه ثبات شاخص

Assembly Language	Operation
MOV CX,[BX+DI]	ثبات CX از محلی از حافظه در DS با آدرس عملیاتی که از جمع BX+DI بدست می آید. پر می شود
MOV CH,[BP+SI]	ثبات CH از محلی از حافظه در SS با آدرس عملیاتی که از جمع BP+SI بدست می آید. پر می شود
MOV [BX+SI],SP	SP در محلی از حافظه در DS با آدرسی که از BX+SI بدست می آید ذخیره می شود
MOV [BP+DI],CS	CS در محلی از حافظه در SS با آدرسی که از BP+SI بدست می آید ذخیره می شود

مزیت عمده کاربرد این مد در آدرس دهی المان های یک آرایه ای از دیتاست. فرض کنید یک آرایه ای از دیتا در بخش DS در محل ARRAY واقع شده را بخواهیم آدرس دهی کنیم. برای اینکار نیاز داریم BX را با آدرس ARRAY پر کنیم و DI ردیف المان های آرایه را داشته باشد. مطابق شکل ۶-۴ بخوبی مشخص است که چگونه BX ابتدای آرایه و DI شمارش آنرا بعهده دارد.



شکل ۶-۴- مثالی که نشان دهنده آدرس دهی ثبات پایه بعلاوه ثبات شاخص باشد.

در مثال ۲-۴ توسط قطعه برنامه ای چگونگی استفاده از این مد آدرس دهی نشان داده شده است، در این برنامه عنصر محل ۱۰ آرایه به محل ۲۰ آرایه منتقل می شود.

```
'Example 4-2
'Using The Base Plus Index Addressing Mode
   MOV   BX,Offset Array    'Address Array
   MOV   DI,10H             'Address Element 10H
   MOV   AL,[BX+DI]         'Get Data in Element 10
   MOV   DI,20H             'Address Element 20H
   MOV   [BX+DI],AL         'Save Data at Element 20H
```

۷-۴- آدرس دهی نسبی رجیستری (Register Relative Addressing)

این مد نیز شبیه آدرس دهی ثبات پایه بعلاوه ثبات شاخص است و همچنین شبیه Displacement Addressing است. در این مد، دیتای موجود در یک سگمنت حافظه از روش جمع کردن عدد Displacement با محتوای یکی از ثبات های پایه یا شاخص (BX, BP, SI, DI) آدرس دهی می شود، دستور MOV AX, [BX+1000] را در نظر بگیرید با فرض اینکه DS=200H و BX=100H و محتوای محل هایی از حافظه به آدرس 03100H و 03101H درون AX قرار می گیردچونکه : آدرس فیزیکی=200×10+100+1000=3100H

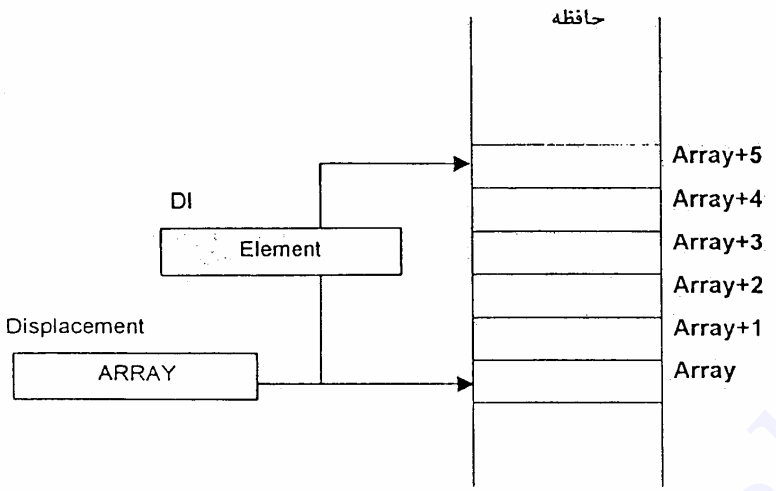
یادآور می شود که اگر از ثبات BP استفاده شود دیتا در بخش SS آدرس دهی خواهد شد.

جدول ۷-۴ لیست چند دستور در این مد را نشان می دهد.

جدول ۷-۴- مثالهائی از آدرس دهی رجیستری نسبی

Assembly Language	Operation
MOV AX,[DI+100H]	دیتا از بخش DS با آدرس DI+100 به AX منتقل می شود.
MOV Array[SI],BL	BL در بخش DS به آدرس Array+SI ذخیره می شود.
MOV LIST[BP]+CL	CL در بخش SS به آدرس LIST+BP ذخیره می شود.
MOV DI,SET[BX]	دیتا از بخش DS با آدرس SET+BX به DI منتقل می شود

همانگونه که در مد آدرس دهی ثبات پایه بعلاوه ثبات شاخص گفته شد این مد نیز توانائی خوبی برای آدرس دهی آرایه ای دارد. شکل ۶-۴ همانند شکل ۷-۴ چگونگی این مدر ا با این تفاوت که نقش ثبات پایه به Displacement داده شده است را نشان می دهد.



شکل ۷-۴

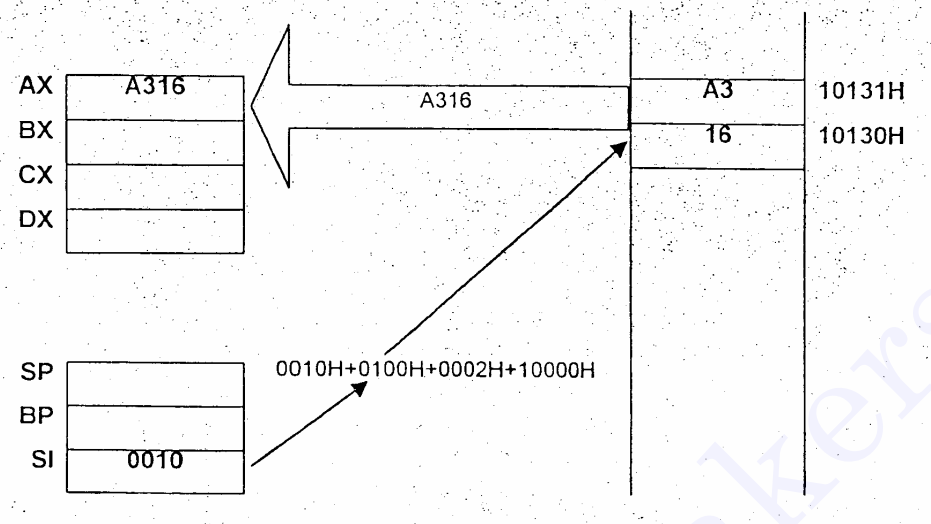
همچنین مثال ۳-۴ نیز عیناً قطعه برنامه ای را در رابطه با مد آدرس دهی رجیستری نسبی نشان می دهد.

```
'Example 4-3
'Using Register Relative Addressing
   MOV    DI,10H              'Address Element 10H
   MOV    AL,Array[Di]        'Get Data From Element 10H
   MOV    DI,20H              'Address Array Element 20H
   MOV    Array[DI],AL        'Save Data at Element 20H
```

۸-۴- آدرس دهی نسبی پایه بعلاوه شاخص (Base Relative-Plus-Index Addressing)

آخرین مد کاربردی در ۸۰۸۶ این مد است. این مد نیز شبیه ثبات پایه بعلاوه ثبات شاخص است، با این تفاوت که به حاصل جمع ثبات پایه و شاخص باید محتوای Displacement را نیز افزود تا آدرس مورد نظر تولید شود. این نوع آدرس دهی معمولاً برای آدرس دهی آرایه های دو بعدی بکار می رود دستور زیر را در نظر بگیرید MOV AX, [BX+SI+100H]

با فرض اینکه SI=0010H, BX=0020H, DIS=100H, DS=1000 باشد آدرس فیزیکی و مطابق شکل ۸-۴ محتوای محلهای 1013H و 10131H به درون AX می ریزد DS×10+DIS+BX+SI=0010H=



شکل ۸-۴- نشان دهند آدرس دهی نسبی پایه بعلاوع شاخص

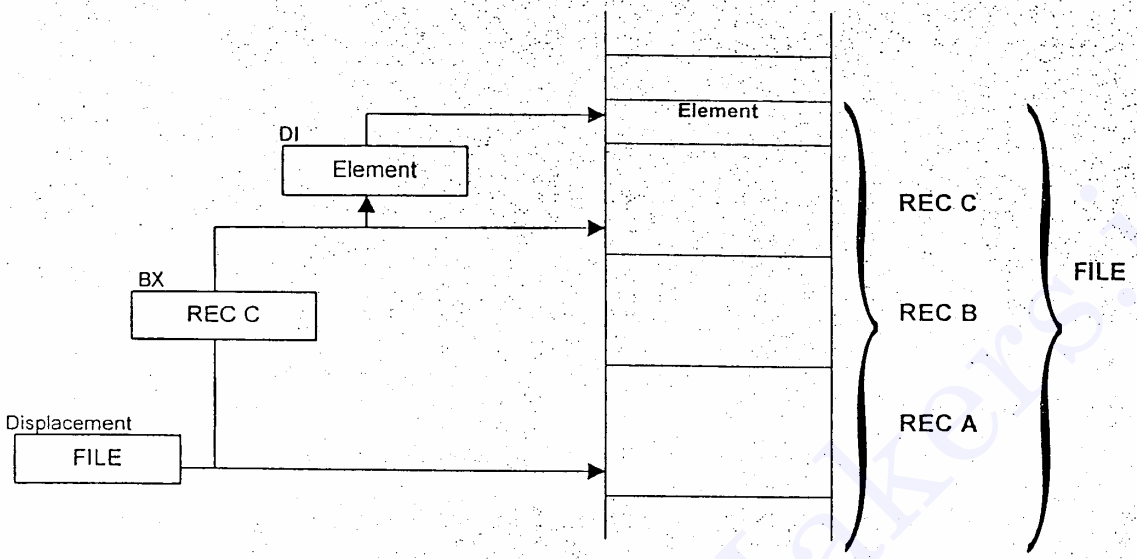
واضح است که محاسبه آدرس فیزیکی در این مد کاملاً پیچیده است. بعضی از دستوراتی که از این مد استفاده می کنند در جدول ۸-۴ لیست شده اند.

جدول ۸-۴- بعضی از دستوراتی که از مد آدرس دهی نسبی پایه بعلاوه شاخص استفاده می کنند.

Assembly Language	Operation
MOV DH,[BX+DI+20H]	محتوای محلی از حافظه در DS به آدرس BX+DI+2 در DH کپی می شود.
MOV AX,FILE[BX+DI]	محتوای محلی از حافظه در DS به آدرس BX+DI+FILE در AX کپی می شود.
MOV LIST[BP+DI],CL	محتوای CL در محلی از حافظه که آدرس آن LIST+BP+DI است ذخیره می شود.

فرض کنید فایلی داریم شامل تعدادی رکورد و هر رکورد شامل تعدادی عنصر (المان) باشد. Displacement آدرس فایل را بدهد و ثبات پایه آدرس یک رکورد در فایل را بدهد و ثبات شاخص آدرس هر عنصر در رکورد را بدهد. این مد آدرس دهی در شکل ۸-۴ نشان داده شده است.

مثال شماره ۴-۴ یک برنامه ایست که انتقال اطلاعات از المان صفر رکورد A به المان ۲ رکورد C را با استفاده از مد آدرس دهی نسبی بعلاوه شاخص نشان می دهد.



شکل ۸-۴- آدرس دهی نسبی پایه بعلاوع شاخص که آدرس یک المان از یک رکورد در یک فایل را نمایش می دهد

```
'Example 4-4
'
'Using The Base Relative Plus Index Addressing Mode
'
MOV     BX,Offset-REC A
MOV     DI,0
MOV     AL,FILE[BX+DI]
MOV     BX,Offset-REC C
MOV     DI,2
MOV     FILE[BX+DI],AL
```

۹-۴-مدهای آدرس دهی برنامه

مدهای آدرس دهی برنامه با دستورات JMP ,CALL دارای سه شکل مستقیم، نسبی و غیر مستقیم می باشد، که در این بخش با بکارگیری دستور JMP به تشریح هر سه نوع پرداخته می شود.

۱-۹-۴- مد آدرس دهی برنامه بصورت مستقیم

مد مستقیم، همان صورتی است که ریزپردازنده ۸۰۸۵ هم برای تمام دستورات JMP و CALL مورد استفاده قرار می دهد. (البته تقریباً تمام ریزپردازنده های ۸ بیتی هم استفاده می کنند) ۸۰۸۶ نیز از این مد استفاده می کند البته نه مثل مدهای نسبی و غیر مستقیم دستوراتی که از مد مستقیم استفاده می کنند آدرس مورد نظر را با Op-Code ذخیره می کنند بعنوان مثال اگر

در یک برنامه ا ی قرار باشد به محل 10000H پرش داشته باشیم عدد آدرس باید با Op-Code ذخیره شود. شکل ۹-۴ نحوه آدرس دهی مستقیم را نشان می دهد. ملاحظه می فرمائید که ۴ بایت برای ذخیره کردن سگمنت بعدی و آفست بعدی مورد نیاز است. یعنی برای JMP to 10000 باید یک عددی مثل 1000 بعنوان سگمنت و عددی مثل 0000 بعنوان افست ذخیره شده که اولی در CS و دومی در IP ریخته خواهد شد.

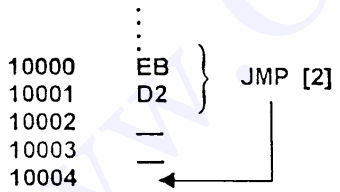
OP-CODE	OFFSET-LOW	OFFSET-HIGH	SEGMENT-LOW	SEGMENT-HIGH
EA	00	00	00	10

شکل۹-۴- پنج بایت زبان ماشین برای دستور JMP به محل 10000H حافظه

تنها دستور دیگری که در مد آدرس دهی مستقیم برنامه کاربرد دارد دستور CALL داخل همان سگمنت است. و اغلب از یک اسمی بعنوان LABEL برای آدرس دهی استفاده می شود.

۲-۹-۴- مد ادرس دهی برنامه بصورت نسبی :

این مد در ۸۰۸۵ وجود ندارد، اما بعضی از ریزپردازنده های ۸ بیتی همانند ۸۰۸۶ دارند که به شرح آن می پردازیم. کلمه نسبی (Relative) عملاً به موقعیت ثبات IP برمی گردد یعنی نسبت به اشاره گر به دستورالعمل آدرس دهی می کند. بعنوان مثال اگر دستور JMP اشاره به دو محل بعد دارد، یعنی دو بایت بعد از آنچه که IP اشاره دارد. برای بدست آوردن چنین آدرسی باید IP+2 شود. به قطعه برنامه زیر توجه کنید :



ملاحظه می کنید که Op-Code دستور JMP یک بایت است و یک بایت هم عدد مورد نظر جمعاً دستور دو بایت فضا اشغال می کند. بهمین دلیل است که کاربرد این مد بیش از آدرس دهی مستقیم برنامه است.

دستورات با مد نسبی دو دستور JMP یا CALL هستند و شامل یک محل Displacement ۸ بیتی یا ۱۶ بیتی هستند. اغلب اسمبلرها دستور Far JMP نیز دارند و بطور اتوماتیک فضای مناسب

برای Displacement خود انتخاب می کنند. اگر میزان پرش خیلی بزرگ باشد بعضی اسمبلرها JMP مستقیم را ترجیح می دهند. با یک بایت فضا شما می توانید تا 127+ محل به بعد و یا 128- محل به قبل برنامه برگردید. اگر از ۱۶ بیت فضا برای Displacement استفاده شود مقدار پرش به ±32k می رسد.

۳-۹-۴- مد آدرس دهی برنامه بصورت غیر مستقیم :

ریزپردازنده ۸۰۸۶ چند فرم از پرش ها و یا صدا زدن زیربرنامه ها را بصورت غیرمستقیم اجازه می دهد در جدول شماره ۹-۴ چند دستور JMP بصورت غیر مستقیم لیست شده است که هر کدام می توانند از هر یک از ثبات های ۱۶ بیتی (AX, BX, CX, DX, SP, BP, SI, DI و هر ثبات نسبی ا ی مثل [BX], [BP], [SI], [DI] و هر ثبات نسبی با Displacement) را بکار بگیرند.

جدول شماره ۹-۴- مثالهائی از آدرس دهی غیرمستقیم برنامه

Assembly Language	Operation
JMP AX	به محلی از حافظه که آدرس آن در AX است در همان سگمنت پرش می کند.
JMP CX	به محلی از حافظه که آدرس آن در CX است در همان سگمنت پرش می کند.
JMP [BX]	به محلی از حافظه که آدرس آن در محلی از حافظه است و BX آدرس آن محل در DS را مشخص می کند در همان سگمنت پرش می کند.
JMP [DI]	به محلی از حافظه در همان سگمنت که آدرس آن محلی در حافظه در DS بوسیله DI مشخص شده پرش می کند.
JMP TABLE[BX]	به محلی از حافظه در همان سگمنت که آدرس آن محل در حافظه بوسیله BX+TABLE مشخص می شود پرش می کند.

برای تشریح دستور JMP TABLE[BX] به مثال زیر توجه کنید :

```
'Example 4-5
'
'Using Indirect Addressing for a JMP
'
MOV     BX,#4          '
JMP     TABLE[BX]      '
TABLE   DW    LOC0
        DW    LOC1
        DW    LOC2
        DW    LOC3
```

فرض کنید TABLE نیز بصورت زیر معرفی شده باشد. که استفاده از این TABLE امکان پرش به جاهای مختلف را می دهد.

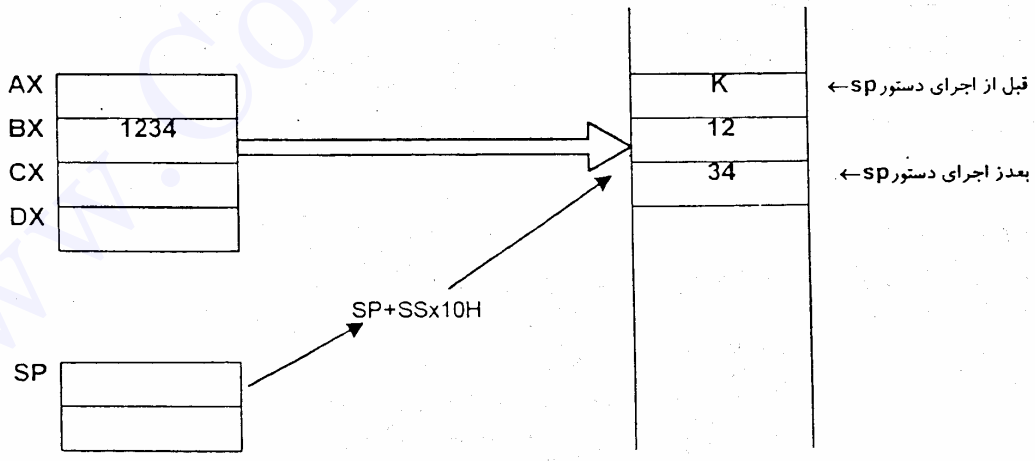
در TABLE ما آدرس هائی برای چهار برنامه مختلف داریم که هر آدرس دارای ۲ بایت Offset می باشد. وقتی که شما از دستور JMP TABLE[BX] استفاده می کنید، وجود TABLE ما را به اول TABLE هدایت می کند به شرطی که BX=0 باشد. هر (DeffineWord)=DW دو بایت فضای حافظه را اشغال می کند لذا وقتی که در BX عدد ۴ را قرار می دهید TABLE+[BX] آدرس LOC2 را بوجود می آورد.

۱۰-۴- آدرس دهی ناحیه پشته (StackMemoryAddressing)

ناحیه پشته در تمام ریزپردازنده ها از اهمیت خاصی برخوردار است. در این حافظه اطلاعات موقتی و آدرس های مراجعه از زیربرنامه را نگهداری می کنند. پشته دارای سازمانی بفرم LIFO(Last in First out) می باشد. دیتا توسط دستور PUSH در پشته قرار داده می شود. البته اگر از دستور CALL استفاده کنید بطور اتوماتیک آدرس برگشت در پشته قرار داده می شود. برداشت دیتا از پشته توسط دستور POP یا اجرای دستور RET در آخر زیربرنامه صورت می گیرد.

اصطلاحاً می گویند SP همیشه به بالای پشته اشاره می کند یعنی به آخرین دیتائی که در پشته قرار داده اید. در ریزپردازنده ۸۰۸۶ حافظه پشته از بالا پر می شود شکل ۱۰-۴ را ببینید. فرض کنید K آخرین دیتائی است که در پشته ذخیره کرده اید. حال اگر قرار باشد دیتای جدیدی در پشته PUSH کنید ابتدا SP←SP-1 می شود

اولین بایت شما ذخیره می شود مجدداً این عمل تکرار می شود تا بایت دوم را ذخیره کنید.



شکل ۱۰-۴- چگونگی قراردادن ثبات BX را در ناحیه پشته نشان می دهد.

نحوه اجرای دستور POPهم بهمین ترتیب است. ابتدا بایت اول برداشته می شود پس از آن SP←SP+1 می شود آنوقت بایت دوم را از پشته برمیدارد. جدول شماره ۱۰-۴ لیست بعضی از دستورات POP, PUSH قابل استفاده در ۸۰۸۶ را نشان می دهد.

توجه داشته باشید که دستورات POP, PUSH همیشه یک کلمه ۱۶ بیتی را به پشته وارد و یا از آن خارج می کنند یعنی هرگز بصورت بایت عمل نمی کنند.

جدول شماره ۱۰-۴- مثالهایی از دستورات PUSH و POP

Assembly Language	Operation
PUSHF	ثبات پرچمها (FLAGS) را در پشته ذخیره می کند.
POPF	ثبات پرچمها (FLAGS) را از پشته برمیدارد.
PUSH AX	ثبات AX را در پشته ذخیره می کند.
POP BX	ثبات BX را از پشته پر می کند
PUSH DS	ثبات DS را در پشته ذخیره می کند.
POP CS	اجرای این دستور امکان پذیر نیست
PUSH [BX]	یک کلمه از حافظه که آدرس آن در BX درون پشته ذخیره می شود.

جدول شماره ۱۱-۴-تمام مودهای آدرس دهی مورد استفاده در ۸۰۸۶ لیست شده است

MOV	AL,BL	آدرس دهی رجیستری
MOV	AL,LIST	آدرس فیزیکی=DS × 10+LIST
MOV	AL,12	آدرس دهی فوری
MOV	AL,[BX]	آدرس فیزیکی=DS × 10+BX
MOV	AL,[BP]	آدرس فیزیکی=SS × 10+BP
MOV	AL,[SI]	آدرس فیزیکی=DS × 10+SI
MOV	AL,[DI]	آدرس فیزیکی=DS × 10+DI
MOV	AL,LIST[BX]	آدرس فیزیکی=DS × 10+BX+LIST
MOV	AL,LIST[BP]	آدرس فیزیکی=SS × 10+BP+LIST
MOV	AL,LIST[SI]	آدرس فیزیکی=DS × 10+SI+LIST
MOV	AL,LIST[DI]	آدرس فیزیکی=DS × 10+DI+LIST
MOV	AL,[BX+SI]	آدرس فیزیکی=DS × 10+BX+SI
MOV	AL,[BX+DI]	آدرس فیزیکی=DS × 10+BX+DI
MOV	AL,[BP+SI]	آدرس فیزیکی=SS × 10+BP+SI
MOV	AL,[BP+DI]	آدرس فیزیکی=SS × 10+BP+DI
MOV	AL,LIST[BX+SI]	آدرس فیزیکی= SS × 10+BX+SI+LIST
MOV	AL,LIST[BX+DI]	آدرس فیزیکی= SS × 10+BX+DI+LIST
MOV	AL,LSIT[BP+SI]	آدرس فیزیکی= DS × 10+BP+SI+LIST
MOV	AL,LIST[BP+DI]	آدرس فیزیکی= DS × 10+BP+DI+LIST

سوالات دوره ای :

۱-ثبات های ۸ بیتی ۸۰۸۶ را نام ببرید.

۲- ثبات های ۱۶ بیتی ۸۰۸۶ کدامند ؟

۳-کدامیک از دستورات زیر غلط است ؟ چرا ؟

```
MOV    AX,BX        MOV    SP,DI        MOV    CS,AX        MOV    CS,SS
```

۴-علامت [ ] نشان دهنده چیست ؟

۵- فرض کنید که DI=0300H, BX=0200H, DS=0200H باشد، آدرس دسترسی به دیتا در حافظه توسط هر کدام از دستورات زیر را تعیین کنید.

```
MOV    AL,[2000H]        MOV    AL,[BX]        MOV    [DI],AL
```

۶- آیا دستور زیر صحیح است ؟ چرا ؟

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

۷-توضیح دهید فرق دو دستور زیر در چیست ؟

```
MOV    BX,Data
MOV    BX,OffsetData
```

۸-اگر DS=1000H, SS=2000H, BP=1000H, DI=0100H را داشته باشیم آدرس دیتا را در هر یک از دستور زیر تعیین کنید ؟

```
MOV    AL,[BP+DI]
MOV    DX,[BP]
MOV    CX,[DI]
```

۹-سه مد آدرس دهی برنامه را نام ببرید.

۱۰- یک دستور JMP چند بایت طول دارد و محتوای هر بایت چیست؟

۱۱- شرح عملکرد PUSH [DI] را بیان کنید.

فصل پنجـــم

«دستورات انتقال داده ها در ۸۰۸۶»

مقدمه : دستورات هر ریزپردازنده به سه دسته تقسیم می شوند، این تقسیم بندی بمنظور سادگی در مطالعه و بررسی انواع دستورات می باشد :

الف- دستورات انتقال داده ها (Data Movement Instruction)

این دستورات فقط عمل انتقال داده از منبع به مقصد را بعهده دارند بدون اینکه هیچ تغییری در داده داشته باشد.

ب- دستورات محاسباتی ریاضی و منطقی (Arithmetic & Logic Instruction)

این دستورات محاسبات ریاضی و منطقی را بعهده دارند بعبارت دیگر پردازشهای مورد نظر توسط این دستورات انجام می گیرد.

ج- دستورات کنترل برنامه (Program Control Instruction)

دستوراتی که در این گروه مورد بررسی قرار می گیرند هیچ عملی و یا ربطی به دیتا ندارند بلکه فقط کنترل برنامه را بعهده دارند.

در این فصل سعی می شود تمام دستوراتی که انتقال و یا جابجائی دیتا در ریزپردازنده ۸۰۸۶ را بعهده دارند معرفی کنیم. این دستورات عبارتند از MOV, PUSH, POP, XCHG, XLAT, IN, OUT, LEA, LDS, LES, LAHF, SAHF و انتقال رشته ای بنام MOVS

علت اینکه اول این دستورات معرفی می شوند این است که اولاً کاربری بیشتری دارند بعلاوه ساده تر از سایر دستورات از نظر درک دانشجو می باشد.

و در خاتمه ما غالب دستورالعمل ها (Format Instructions) مورد تجزیه و تحلیل قرار داده تا بتوانید به زبان ماشین نیز برنامه بنویسید و یا برنامه ای را تحلیل نمائید.

۱-۵- مرور مجدد بر دستور MOV

در فصل چهارم راجع به دستورات MOV در بحث مدهای آدرس دهی صحبت شده، در اینجا ما این دستورات را بصورت زبان ماشین مورد بررسی قرار می دهیم ؛ زیرا ما باید قادر به تفسیر یک برنامه نوشته شده به زبان اسمبلی باشیم. علی الخصوص زمانی که بخواهیم آنرا اشکال زدایی یا تغییری بدهیم.

۱-۱-۵- زبان ماشین

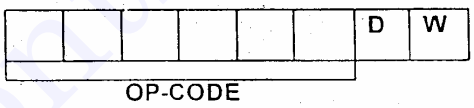
دستورات ۸۰۸۶ به زبان ماشین دارای طول متغیری از یک بایت تا ۶ بایت می باشند، اولین بایت هر دستور کد عملیاتی (OP-CODE) آن است که بیانگر نوع عملی است که باید انجام دهد. شکل ۱-۵ فرم کلی یک کد عملیاتی ریزپردازنده ۸۰۸۶ برای تعدادی از دستورات را نشان می دهد، ۶ بیت اولیه بایت اول هر دستور OP-CODE آن است و دو بیت باقیمانده نشان دهنده جهت حرکت دیتا (D) و اینکه دیتا بصورت ۸ بیتی است یا ۱۶ بیتی می باشد(W).

اگر D=1 است یعنی دیتا به ثباتی که در بایت دوم معرفی می شود وارد خواهد شد.

اگر D=0 است یعنی دیتا از ثباتی که در بایت دوم معرفی می شود خارج خواهد شد.

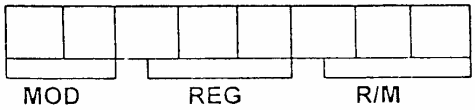
اگر W=0 است یعنی عمل انجام شده روی یک بایت دیتا خواهد بود.

اگر W=1 است عمل انجام شده روی دیتای ۱۶ بیتی خواهد بود.



شکل ۱-۵- بایت اول اغلب دستورات زبان ماشین ریزپردازنده ۸۰۸۶

شکل ۲-۵ بایت دوم دستورالعمل را در صورتیکه داشته باشد نشان می دهد. این بایت بیانگر مد (MOD) آدرس دهی دیتا است. دو بیت برای MOD، سه بیت برای معرفی ثبات (REG) و سه بیت برای معرفی حافظه یا ثبات دیگر (R/M) که یکی برای منبع و دیگری برای مقصد مورد استفاده قرار می گیرد.



شکل ۲-۵- بایت دوم یک دستورالعمل زبان ماشین ریزپردازنده ۸۰۸۶

بخش مد یکی از انواع مختلف آدرس دهی و Displacement را طبق جدول ۱-۵ برای دستورات انتخاب می کند.

جدول ۱-۵- بیانگر مدهای مورد استفاده

MOD	عملیات
00	NODISPLACEMENT
01	یک DISPLACEMENT ۸ بیتی دارد.
10	یک DISPLACEMENT ۱۶ بیتی دارد.
11	یک ثبات است.

طبق جدول اگر MOD=11 باشد بخش R/M بایت دوم، یک ثبات را معرفی می کند اما در سه حالت دیگر MOD، موضوع R/M به سایر مدهای عملیاتی برمی گردد. اگر در مد آدرس دهی از 00 استفاده شده باشد در محاسبه آدرس Displacement وجود ندارد. و دو حالت 01 و 10 یک DISP هشت بیتی و یا ۱۶ بیتی دارد. وقتی که بایت استفاده می شود 00~7F مقدار DISP با احتساب بیت علامت خواهد بود، اگر این مقدار منفی باشد 80~FF خواهد شد. در صورتی که DISP شانزده بیتی استفاده شود (توسعه یافته) مقدار آن 0000~007F و در حالتی که ۱۶ بیتی باشد مقدار آن نیز منفی باشد FF80~FFFF خواهد شد.

۲-۱-۵- تخصیص کد به ثبات ها :

جدول شماره ۲-۵ کد اختصاص داده شده به ثبات ها (REG) و یا حافظه ثبات (R/M) را وقتی که MOD=11 می باشد نشان می دهد.

جدول شماره ۲-۵- تخصیص کد به ثبات (REG) و ثبات حافظه (R/M) که MOD=11 است.

CODE	W=0(BYTE)	W=1(WORD)
000	AL	AX
001	CL	CX
010	DL	DX
011	BL	BX
100	AH	SP
101	CH	BP
110	DH	SI
111	BH	DI

توجه دارید که این جدول دارای دو لیست به کدها اختصاص داده شده است یکی به ازای W=0 که دیتا انتقالی بایت است و لیست دوم به ازای W=1 که دیتای انتقالی ۱۶ بیتی است.

فرض کنید که یک دستور دو بایتی که مقدار آن 8BECH است داشته باشیم. آنرا بصورت باینری بطور مجزا بایت ۱ و بایت ۲ مشخص کنید شکل ۳-۵ بدست می آید. این دستور MOV BP, SP است

OP-CODE						D	W	MOD		REG			R/W		
1	0	0	0	1	0	1	1	1	1	1	0	1	1	0	0

MOV	OP-CODE=100010
جهت به ثبات	D=1
دیتای ۱۶ بیتی	W=1
R/W یک ثبات است	MOD=11
ثبات BPاست	REG=101
ثبات SP است	R/M=100

شکل ۳-۵- نمایش باینری دستور 8BECH و تفسیر بیت های آن

با مراجعه به ضمیمه یک این کتاب ملاحظه می فرمائید که OP-CODE مذکور متعلق به دستور MOV است توجه دارید که هر دو بیت W, D یک هستند، یعنی اینکه یک کلمه ۱۶ بیتی به ثبات انتقال داده می شود. و ثبات آن 101 می باشد که BP خواهد بود. در واقع دیتای ۱۶ بیتی به ثبات BP منتقل می کند. چون MOD=11 است پس R/M نیز یک ثبات است که طبق جدول ۲-۵ کد 100 ثبات SP می باشد. یعنی دیتا از SP به BP منتقل می شود فرم دستور آن MOV BP, SP است.

۳-۱-۵- آدرس دهی حافظه برای R/M

اگر بخش MOD دارای مقادیر 00 یا 01 یا 10 باشد، میدان R/M معنی جدیدی بخود می گیرد که همانا محلی از حافظه خواهد بود. در اینصورت به این بخش مدهای آدرس دهی مختلفی اختصاص داده می شود که توسط سه بیت مربوطه مطابق جدول ۳-۵ تعیین می شود. البته دقت فرمائید تمامی این مدها در فصل چهار صحبت شده است. با توجه به جدول اگر MOD=0 باشد و R/M=101 باشد مد آدرس دهی [DI] خواهد بود. اما اگر MOD=01 یا 10 باشد آنوقت مد بعنوان مثال DISP[DI] یا DATA[DI] خواهد شد.

شکل ۴-۵ فرمت دستورالعمل و زبان ماشین دستور MOV DL,[DI] را نشان می دهد. این دستور دارای دو بایت طول است و کد عملیاتی آن OP-CODE=100010 است. D=1 است (انتقال به

ثبات) و W=0 (بایت منتقل می شود) و MOD=00 (DISP ندارد) و REG=010 (بیانگر ثبات DL) و R/M=101 (یعنی محلی از حافظه که آدرس آن در DI است)

جدول ۳-۵- آدرس دهی مدهای R/M

CODE	شرح عملیات
000	[BX+SI]
001	[BX+DI]
010	[BP+SI]
011	[BP+DI]
100	[SI]
101	[DI]
110	[BP] *
111	[BX]

* در مورد [BP] تحت عنوان "مد آدرس دهی خاص" دقیق مطالعه فرمائید.

OP-CODE						D	W
1	0	0	0	1	0	1	0

MOD		REG			R/M		
0	0	0	1	0	1	0	1

OP-CODE بیانگر MOV

D جهت انتقال به REG

W یک بایت منتقل می شود

MOD بیانگر Displacement ندارد

REG بیانگر ثبات DL

R/M بیانگر ثبات [DI]

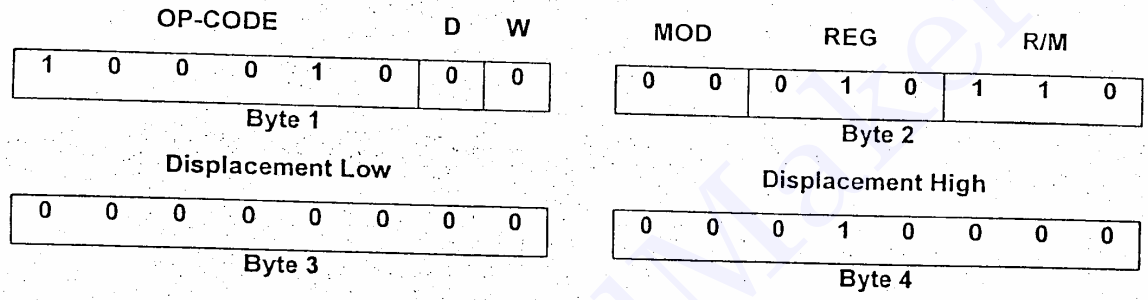
شکل ۴-۵- بیان و نمایش میدان های دستور MOV DL,[DI]

۴-۱-۵- مد آدرس دهی خاص

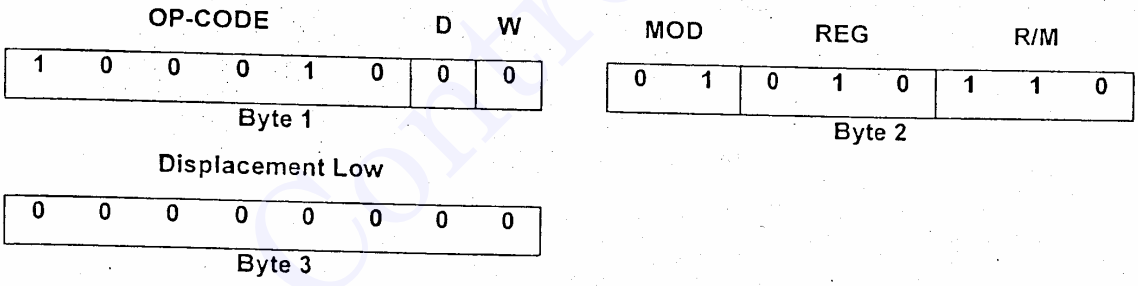
یک مد آدرس دهی خاص داریم که در جداول ۱-۵ و ۲-۵ و ۳-۵ ظاهر نشده است. این مد زمانی کاربرد دارد که آدرس دیتا فقط با Displacement مشخص شده باشد. بعنوان مثال دستورات MOV [1000H],DL و MOV DATA,DL را در نظر بگیرید، دستور اول محتوای ثبات DL را در حافظه ای که آدرس آن DS×10+1000 است ذخیره می کند. هر وقت که یک دستور فقط یک Displacement داشته باشد، میدان مد آن همیشه MOD=00 خواهد بود و میدان R/M همیشه کد 110 را دارد. این ترکیب باعث می شودتصور کنیدکه Displacement ندارد و از مد آدرس دهی [BP] بدون Displacement استفاده می کنید اما شما عملاً نمی توانید مد [BP] را بدون Displacement استفاده کنید، زبان اسمبلی یک بایت Displacement بطور

اتوماتیک برای آن قرار می دهد ولی مقدار آن را 00 می گذارد و از مد MOD=01 استفاده می کند، لذا هنگام استفاده از [BP] عملاً از دستور [BP+00H] استفاده خواهد شد، شکل ۵-۵ میدان و بیت های باینری دستور MOV [1000H],DL را تشریح می کند، اگر چه به تنهایی این مد شناخته شده نیست این دستور بطور غیر صحیحی فرض شده که بصورت MOV [BP],DL معرفی شود.

شکل ۶-۵ فرم اسمبلی و شرح بیت های دستور MOV [BP],DL را نشان می دهد ملاحظه می کنید که سه بایت است و بایت سوم 00H است.



شکل ۵-۵- تعداد بایت ها و فرمت دستور MOV [1000H],DL



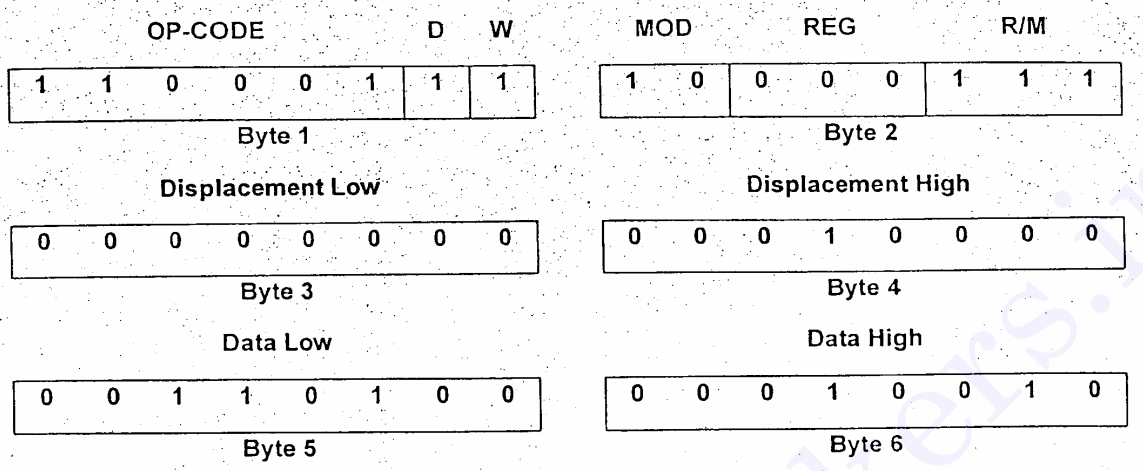
شکل ۶-۵- تعداد بایت ها و فرمت دستور MOV [BP],DL با سومین بایت که 00Hاست

۵-۱-۵- دستور MOV با آدرس دهی فوری

دستور MOV [BX+1000],1234H را در نظر بگیرید. این دستور عدد 1234H را در محلی از حافظه که آدرس آن بصورت زیر محاسبه می شود ذخیره می کند :

آدرس فیزیکی = DS×10H+BX+1000H

این دستور ۶ بایت طول دارد دو بایت برای کد عملیاتی D, W, MOD, REG و R/M مثل دستورات قبلی، دو بایت برای ذخیره کردن دیتا 1234 و دو بایت Displacement برای ذخیره کردن عدد 1000H شکل ۷-۵ بایت ها و فرمت دستور را نشان می دهد:



شکل ۷-۵- شش بایت و فرمت دستور MOV [BX+1000],1234H

۶-۱-۵- دستور MOV بر انتقال ثبات های سگمنت

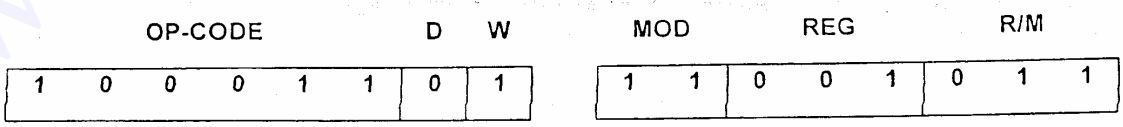
اگر بخواهیم محتوای یک ثبات سگمنت را بوسیله دستورات PUSH, POP, MOV انتقال دهیم در میدان REG بایت دوم کد عملیاتی کدهای مخصوصی باید در نظر گرفت که در جدول ۴-۵ ثبات های مختلف سگمنت را معرفی می کند.

جدول ۴-۵- میدان REG برای ثبات های سگمنت

ثبات سگمنت	کد
ES	000
CS *	001
SS	010
DS	011

* توجه داشته باشید که MOV CS,XX و POP CS اجازه داده نشده است

شکل ۸-۵ دستور MOV BX,CS را نشان می دهد. دقت داشته باشید که OP-CODE این MOV با MOVهای دیگر تفاوت دارد و REG براساس ثبات های سگمنت انتخاب می شود.



شکل ۸-۵- دستور MOV BX,CS بصورت زبان ماشین

اگر چه در ایـــن مبحـث مـا بطـور کـامل راجـع بـه زبـان ماشـین صحبـت نمـی کنیـم، ولـی ایـن مقـدار می تواند شروع خوبی برای علاقمنـدان بـه برنامـه نویسـی بـه زبـان ماشـین باشـد. بخـاطر داشـته باشـید که معمولاً برنامـه بـاید بصـورت اسـمبلی نوشـته مـی شـود و مـترجم آن اسـمبلر نـام دارد کـه آن را بـه زبان ماشین تبدیـل مـی کنـد. ندرتـاً ممکـن اسـت شـما نیـاز پیـدا کنیـد کـه زبـان ماشـین را بخواهیـد بصورت مسـتقیم بنویسـید.

۲-۵- دســـتورات PUSH/POP

دو دسـتور POP, PUSH از جملـه دسـتورات مـهمی در ریـزپردازنـده ۸۰۸۶ جـهت قـرار دادن و برداشـتن دیتـا در سـازمان LIFO پشـته (Stack) مـی باشـند. در ایـن ریـزپردازنـده چـهار فـرم دستور POP, PUSH بـه شـرح زیـر داردیـم :

الـف- ثبـات (REGISTER)

ب- حافظـه / ثبـات (Register/Memory)

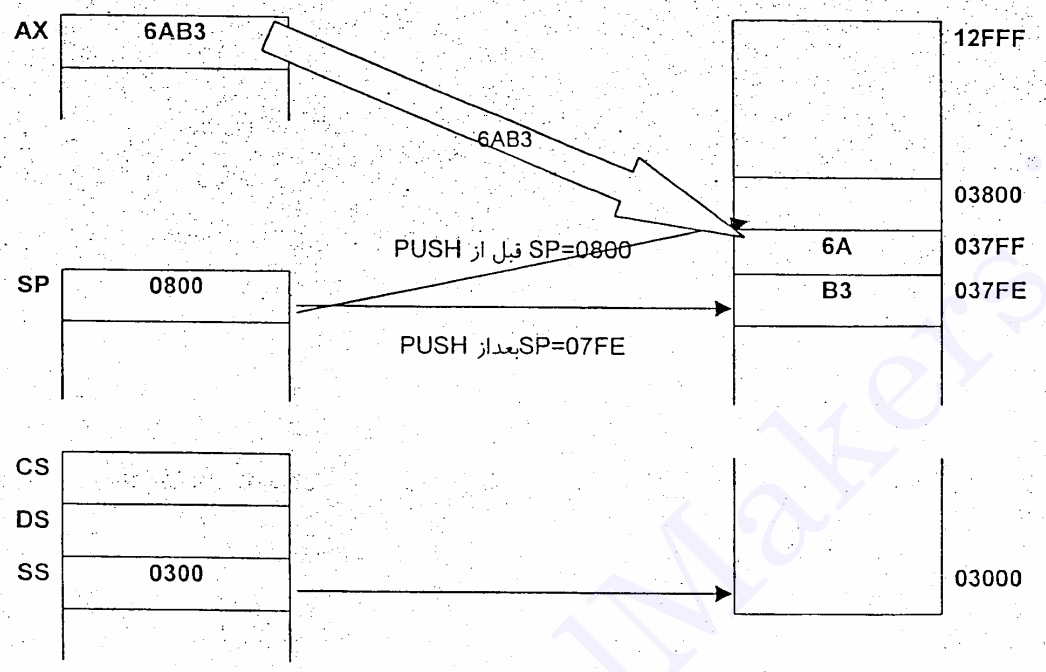
ج- ثبـات سـگمنت (SegmentRegister)

د- پـرچمـها (Flags)

در فـرم ثبـات اجـازه داریـد هـر ثبـات ۱۶ بیتـی را بـه پشـته PUSH و یـا از آن POP نمـائید. در فـرم حافظه / ثبات مجـازید محتـوای یـک ثبـات ۱۶ بیتـی یـا یـک محـل حافظـه را در پشـته ذخـیره کنیـد. در فرم ثبـات هـای سـگمنت مجـازید هـر ثبـات سـگمنت را در پشـته PUSH نمـائید. امـا کـد سـگمنت را نمـی تـوان از پشـته POP کـرد بـاید بصـورت دیگـری مقـدار PUSH شـده را بازیـافت. در مـورد آخرین فرم، پرچمها (Flags) را می تـوان بیـن پشـته و ثبـات موقعیـت سیسـتم مبادلـه نمـود.

۱-۲-۵- دســـتور PUSH

دستور PUSH همیشـه دو بـایت دیتـا بداخـل پشـته منتقـل مـی کنـد. منبـع ایـن دیتـا مـی توانـد ثبـات پرچم، هر ثبات ۱۶ بیتی داخلی ریزپردازنـده و یـا ۲ بـایت دیتـا از حافظـه باشـد. هـر وقـت قـرار باشـد دیتا وارد پشته شود ابتـداً بـا ارزشـترین بـایت وارد حافظـه پشـته در محلـی SP-1 قـرار مـی گـیرد پـس از آن بـایت دوم (کـم ارزشـتر) وارد پشـته شـده و در محـل SP-2 ذخـیره مـی گـردد. یعنـی در هـر دسـتور PUSH بـاید ثبـات SP دو عـدد کـم شـود. شـکل ۹-۵ عملیـات اجرائـی دسـتور PUSH AX را نشـان مـی دهـد. ملاحظـه مـی فرمـائید کـه محتـوای ثبـات AX بصـورت زیـر در پشـته ذخـیره مـی شـود [SP-1]=AH, [SP-2]=AL



شکل ۹-۵- نمایش نحوه اجرای دستور PUSH AX

شکل ۵-۵ لیست چهار فرم دستور PUSH ممکنه در ریزپردازنده ۸۰۸۸/۸۰۸۶ را نمایش می دهد. بایت یک که OP-CODE می باشد نیز بصورت باینری مشخص شده است.

جدول ۵-۵- چهار صورت دستورات PUSH

SYMBOLIC	BYTE1	BYTE2	مثال
PUSH reg PUSH mem PUSH seg PUSHF	01010rrr 11111111 000ss110 10011100	mm110aaa	PUSH BX PUSH [BX] PUSH DS PUSHF

توجه : در جدول بالا

aaa = یعنی هر مد آدرس دهی حافظه

mm= همان کد میدان mod در بایت دوم است

rrr= هر یک از ثبات های ۱۶ بیتی داخلی سیستم

ss= هر کدام از ثبات های سگمنت می تواند باشد

۲-۲-۵- دستور POP

دستور POP عکس دستور PUSH عمل می کند. این دستور دیتا را از پشته برمی دارد و آنرا در محلی که تعیین کرده اید مثل یک ثبات یا پرچمها یا یک محل ۱۶ بیتی حافظه که آدرس دهی کرده اید قرار می دهد. POP هم به چهار فرم ذکر شده برای PUSH عمل می کند (ثبات، حافظه / ثبات، ثبات سگمنت و پرچمها)

تصور کنید که یک دستور مثل POP BX اگر اجرا شود ابتدا اولین بایت دیتا از پشته (از محلی که SP اشاره به آن دارد) آورده می شود و در BL قرار داده خواهد شد بایت دوم دیتا از پشته از محلی که SP+1 نشان می دهد آورده و در BH قرار داده می شود مجدداً SP افزوده می شود. در واقع پس از انتقال هر دو بایت SP دو بار افزوده شده است. جدول ۶-۵ کدهای عملیاتی و ترکیباتی باینری و یک مثال برای هر نوع POP را لیست کرده است. توجه داشته باشید که دستور POP CS مجاز نیست.

جدول شماره ۶-۵- کدهای عملیاتی و ترکیبات باینری دستور POP را با ذکر مثال نشان می دهد.

شکل سمبلیک	BYTE1	BYTE2	یک مثال
POP reg	01011rrr		POP DI
POP mem	10001111	mm000aaa	POP[DI+2]
POP seg	000ss111		POP ES
POPF	10011101		POPF

توجه aaa = هر نوع در آدرس دهی می تواند باشد

mm = کد میدان mod در بایت دوم

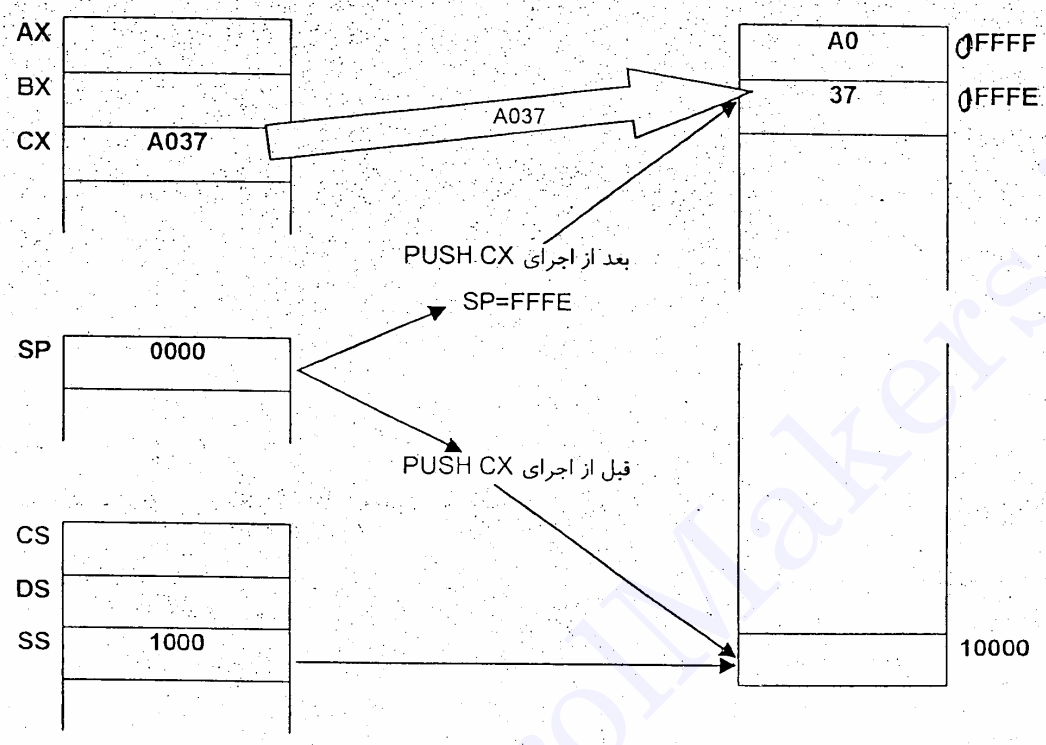
rrr =هر ثبات ۱۶ بیتی می تواند باشد

ss=هر ثبات سگمنت می تواند باشد به غیر از CS

۳-۲-۵- برقراری پشته (Initializing the Stack)

وقتی که ناحیه پشته برقرار می شود. هر دو ثبات مربوطه یعنی ثبات سگمنت پشته (SS) و اشاره گر به پشته (SP) پر می شوند. در اغلب حالات معمول آن است که ناحیه ای از حافظه ای را به پشته اختصاص بدهیم و SS را با مقدار مربوطه پر کنیم. بعنوان مثال اگر بخواهیم پشته از ناحیه 10000H شروع شود و تا 1FFFFH ادامه داشته باشد. آنوقت باید SS=1000H باشد که با 10برابر شدن برابر آدرس شروع مورد نظر شود.

در اینصورت باید SP=0000H باشد. شکل ۱۰-۵ نشان می دهد که چگونه هنگام اجرای دستور PUSH دیتا در بالای (TOP) پشته قرار می گیرد. برای نشان دادن مطلب فوق از دستور PUSH CX استفاده شده است، بخاطر داشته باشید که تمام سگمنت ها طبیعتاً دوره ای (CYCLIC) هستند یعنی پس از پر شدن پایین ترین محل دوباره بالاترین محل (TOP) پشته پر می شود.



شکل ۱۱-۵- نمایش اینکه ناحیه پشته طبیعتاً دوره ای است

۳-۵- دستور LEA (Load Effective Address)

سه دستور داریم که برای پر کردن یک ثبات و یا یک ثبات معمولی و یک ثبات سگمنت از آدرس موثر استفاده می شود. دقت کنید یک آدرس در ثبات ریخته می شود نه یک دیتا جدول شماره ۷-۵ هر سه فرم این دستورات را که در ۸۰۸۶/۸۰۸۸ کاربرد دارند نشان می دهد.

جدول ۷-۵- دستورات ممکنه LEA ها در ۸۰۸۶

فرم سمبلیک	شرح عمل
LEA AX,DATA	ثبات AX با یک آدرس از محل DATA پر می شود.
LDS DI,LIST	DI و DS از آدرس های موجود در LIST پر می شوند.
LES BX,CAT	BX و ES از آدرس های موجود در CAT پر می شوند.

دستور LEA معمولاً یک ثبات را با یک آدرسی که بصورت اپرند معرفی شده پر می کند مانند مثال اول جدول شماره ۷-۵ که اپرند محتوای دیتا در ثبات AX ریخته می شود نه محتوای محلی که دیتا آدرس آن باشد.

اگر LEA را با یک دستور MOV مقایسه کنید. قطعاً متوجه خواهید شد دو دستور داریم MOV BX,[DI], LEA BX,[DI] خیلی شبیه هم هستند در LEA محتوای DI درون BX کپی می شود اما در دستور MOV محتوای محلی از حافظه که DI دارای آدرس آن محل است درون BX کپی می شود، در واقع LEA BX,LIST شبیه MOV BX,OFFSET LIST می باشد. هر دو دستور OFFSET آدرس محل حافظه بنام LIST را در BX می ریزند. چرا ما شبه دستور OFFSET را بکار می بریم در صورتی که در LEA نیازی به آن نداریم. جواب این است که ما OFFSET را برای اپرندهای ساده بکار می بریم مانند DATA, LIST و غیره و امکان ندارد آنرا برای اپرندهائی مثل LIST[SI], [DI] بکار برد. لذا برای اپرندهای ساده خیلی مناسب تر است از دستور LEA استفاده شود. در ریزپردازنده ۸۰۸۶ اجرای دستور LEA BX,LIST از نظر زمانی طولانی تر از دستور MOV BX,OFFSET LIST می باشد. اولی ۸ پالس ساعت زمان می برد در صورتیکه دستور MOV ۴ پالس ساعت طول می کشد. در واقع دستور MOV BX, OFFSET LIST دو بار اجرا می شود تا دستور LEA BX,LIST یک بار اجرا شود و همچنین دستور MOV BX,DI از دستور LEA BX,[DI] خیلی مناسب تر است.

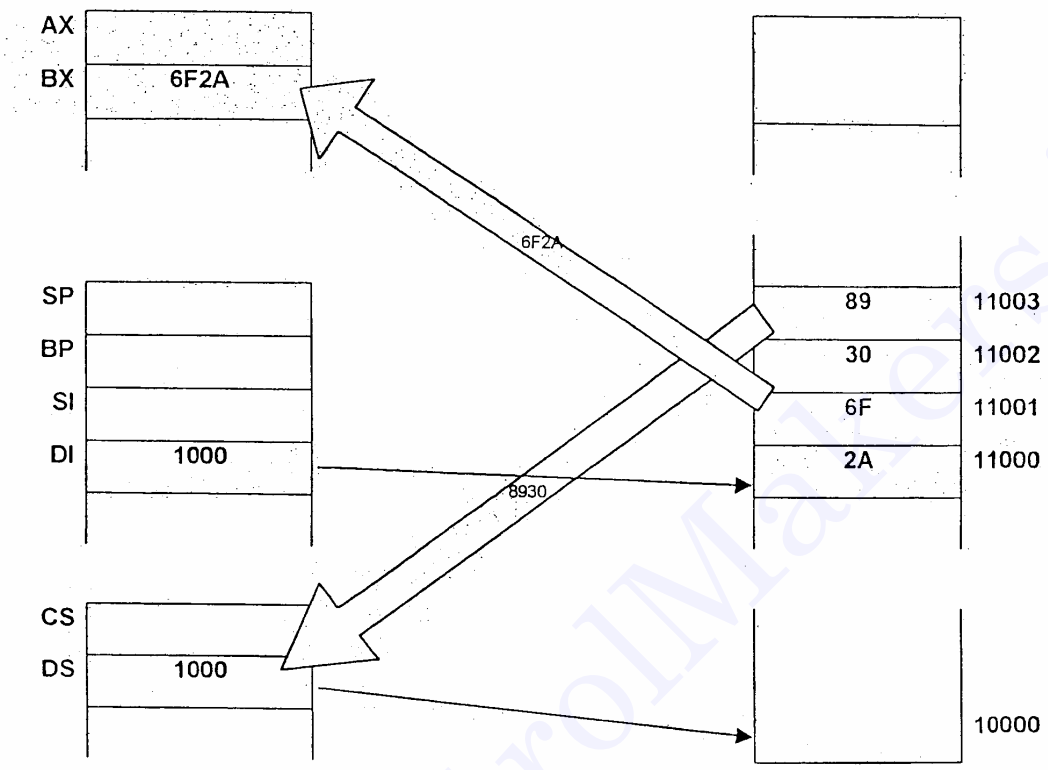
مثال دیگری که در این مورد می تواند جالب باشد بصورت زیر است به دستور LEA CX,[BX+DI] توجه کنید باید محتوای BX با DI جمع شود بعنوان آدرس درون ثبات CX ریخته شود. در این مثال مشکلی که پیش می آید جمع مدول 64K خواهد بود، یعنی اگر بعنوان مثال BX=FF00 باشد و DI=1000 باشد آنچه در CX ریخته می شود مقدار 0F00 خواهد بود. یعنی از CARRY حاصله از جمع صرف نظر شده است.

۱-۳-۵- دستورات LES, LDS

دستورات LES, LDS یک رجیستر ۱۶ بیتی را با آدرس Offset بعلاوع DS, ES را نیز با آدرس جدید سگمنت پر می کنند. این دستورات نمی توانند دارای MOD=11 باشند. بعنوان مثال همانگونه که در شکل ۱۲-۵ مشخص شده است دستور LDS BX, [DI] یک آدرس ۳۲ بیتی را از محلی از حافظه که DI مشخص به ثبات BX, DS منتقل می کند. دستورات LES, LDS اشاره به یک دیتا سگمنت یا اکسترا سگمنت جدید دارند، همچنین Offset جدید را نیز در ثباتی که در دستور مشخص شده می ریزند.

در ادامه این فصل هنگامیکه ما راجع به دستورات رشته ای صحبت کنیم مجدداً این دستورات را نیز بررسی خواهیم کرد.

توجه کنید که آدرسی که در ثبات ها ریخته می شود باید قبلاً در حافظه ذخیره شده باشد.



شکل۱۲-۵- نمایش دستور LDS BX,[DI] ثبات BX از محل 11000 و 11001 پر می شود و ثبات DS از محلهای 11002 و 11003 پر می شود.

۴-۵- انتقال دیتای رشته ای : (String Data Transfer)

سه دستور برای انتقال دیتا بصورت رشته ای وجود دارد. LODS, STOS, MOVS هر کدام اجازه می دهند که دیتا بصورت گروهی، بلوکی یا تک بایتی و یا تک کلمه ای انتقال یابد، قبل از اینکه وارد بحث این دستورات شویم، اجازه دهید تا درباره پرچم D (Direction Flag) و ثبات های سگمنتی که به ثبات های SI, DI اختصاص داده می شود تا بتوان عملیات رشته ای را انجام دهیم صبحت کنیم.

۱-۴-۵- پرچم D

پرچم تعیین کننده جهت (D)، نحوه خود افزایشی (به ازای D=0) و خود کاهشی (به ازای D=1) را برای ثبات های SI, DI در خلال عملیات رشته ای انتخاب می کند، این پرچم توسط دو

دستور STD, CLD دارای مقدار صفر و یک می شود لذا دستور CLD برای ثبات های DI, SI حالت خود افزایشی و دستور STD حالت خود کاهشی انتخاب می کنند.

اگر هدف انتقال اطلاعات بصورت بایتی باشد میزان خود افزایشی و خود کاهشی یکی یکی خواهد بود. اما اگر قرار باشد کلمه ۱۶ بیتی انتقال دهیم میزان افزودن و یا کاستن از ثبات های SI, DI باید بصورت دوتا، دوتا می باشد. اگر در عملیاتی فقط یک ثبات را مورد استفاده قرار دهید (بعنوان مثال از DI استفاده شود) فقط همان ثبات افزوده و یا کاسته می شود.

۲-۴-۵- ثبات های SI, DI در عملیات رشته ای

در خلال اجرای دستور رشته ای ما می توانیم به ثبات های DI یا SI و یا هر دوی آنها دسترسی داشته باشیم. در DI آدرس شروع بلوکی را قرار می دهیم که در سگمنت اکسترا (ES) وجود دارد. و در SI آدرس شروع بلوکی را قرار می دهیم که در سگمنت دیتا (DS) وجود دارد. سگمنت اختصاص داده شده به ثبات SI را می توانید عوض کنید، که این کار توسط افزودن بخش سگمنت به دستور عملی می شود و در آینده راجع به آن صبحت خواهد شد. اما DI همواره در بخش اکسترا سگمنت (ES) قرار خواهد داشت و امکان عوض کردن سگمنت DI وجود ندارد.

۳-۴-۵- دستور LODS

این دستور AL یا AX را از محلی از حافظه که توسط ثبات SI آدرس دهی می شود پر می کند. جدول ۸-۵ لیست دستورات LODS را به دو صورت LODSB یا LODSW که باعث می شود یک بایت یا یک کلمه ۱۶ بیتی از حافظه درون AL یا AX ریخته شود را نشان می دهد. معمولاً بدنبال دستور LODS انتخاب کننده بایت یا کلمه ۱۶ بیتی خواهد بود. معمولاً اپرندها بمنزله بایت توسط DB (Define Byte) و به منزله یک کلمه ۱۶ بیتی توسط DW (Define Word) معرفی می شوند. بعبارت دیگر DB یک شبه دستورالعمل است بمعنی تعریف بایت (Defie Byte) و DW یک شبه دستورالعمل است به معنی تعریف یک کلمه ۱۶ بیتی (Define Word).

جدول ۸-۵- فرم دستورهای LODS

فرم سمبلیک	شرح عمل
LODS B	محتوای حافظه ای که SI آدرس می دهد = AL
LODS W	محتوای حافظه ای که SI و SI+1 آدرس می دهد = AX
LODS BYTE	به شرطی که BYTE به منزله یک بایت معرفی شده باشد AL = M[SI]
LODS WORD	به شرطی که WORD به منزله یک کلمه معرفی شده باشد AX = M[SI]

۵-۴-۴- دستور STOS

این دستور AL یا AX را در محلی از حافظه که توسط DI در بخش اکسترا سگمنت (ES) مشخص شده است ذخیره می کند. جدول ۹-۵ لیست دستورات قابل اجرای STOS را نشان می دهد. مانند دستور LODS با حروف B و W مقدار ۸ بیتی یا ۱۶ بیتی بودن دیتا معین می شود.

جدول ۹-۵-نمایش دستورات STOS

فرم سمبلیک	شرح عمل
STOS B	M [DI] ← AL
STOS W	M [DI] ← AX
STOS BYTE	M [DI] ← AL به شرطی که BYTE به منزله ۸ بیت معرفی شده باشد
STOS WORD	M [DI] ← AX به شرطی که WORD به منزله ۱۶ بیت معرفی شده باشد

۵-۴-۵- دستورات STOS با REP

کلمه REP بمنزله تکرار کردن (Repeat) ممکن است بعنوان پیشوند به هر دستور رشته ای اضافه شود. این اضافه شده باعث می شود که اجرای دستور مذکور ادامه یابد تا ثبات CX (بمنزله شمارنده) برابر صفر شود.

تصور کنید تعداد ۱۰ بایت دیتا در یک منطقه ای از حافظه داشته باشیم و بخواهیم آنها را پاک کنیم یعنی آن ۱۰ محل را صفر کنیم. این عمل را می توانیم با یک سری از دستورات STOS (یعنی با ۱۰ تا از آن) تکمیل کنیم یا اینکه با یک دستور STOS که با پیشوند REP بیاید به مثال ۱-۵ دقت کنید :

```
Example    5-1
           LES    DI,BUFFER     ;
           MOV    CX,10         ;
           CLD                  ;
           MOV    AL,0          ;
           REP    STOSB         ;
```

ملاحظه می کنید از دستور LES استفاده کرده تا از حافظه دو بایت بنام Offset آدرس در DI بریزد و دو بایت بنام سگمنت آدرس در ES بریزد. از CLD استفاده کرده تا D=0 شود و DI بطور اتوماتیک افزوده شود.

در بعضی از اسمبلی ها باید برچسب آدرسی که در دستور LES استفاده می کند بصورت شبه دستور DD (Define Double-Word) معرفی می شوند. کار این شبه دستور این است که اعلام می کند حاوی آدرس ۳۲ بیتی می باشد.

دستور دیگر در AL عدد صفر را قرار می دهد و آنرا توسط STOSB در حافظه ذخیره می کند. بخاطر وجود REP هر بار به DI افزوده می شود و CX کاسته می شود و این عمل تکرار می شود تا CX=0 گردد.

راه سریعتر از آن این است که از دستور REP STOSW استفاده کنید. مانند مثال ۲-۵ :

Example 5-2 :

```
LES   DI,BUFFER   ;
MOV   CX,5        ;
CLD               ;
MOV   AX,0        ;
REP               ;
STOSW             ;
```

۶-۴-۵- دستور MOVS

قوی ترین دستور رشته ای انتقال دیتا MOVS است زیرا این دستور یک بایت یا یک کلمه از محلی از حافظه را به محل دیگری از حافظه انتقال می دهد. دستور MOVS یک دیتا را از محلی که توسط DS و SI آدرس می دهد به محل دیگری که ES و DI تعیین می کند منتقل می نماید. جدول ۱۰-۵ لیست دستورات قابل اجرا توسط MOVS را نشان می دهد. لازم به ذکر است که DS را توسط دستوری که در آینده راجع به آن صحبت خواهیم کرد می توان به سگمنت دیگری تغییر داد

جدول ۱-۵- لیست دستورات MOVS

فرم سمبلیک	شرح عمل	
MOVSB	M [DI] ← M[SI]	یک بایت از محلی از حافظه به محل دیگر می ریزد
MOVSW	M [DI] ← M[SI]	یک کلمه از محلی از حافظه به محل دیگر می ریزد
MOVS BYTE1, BYTE2	M [DI] ← M[SI]	اگر BYTE1 و BYTE2 به منزله یک بایت باشد
MOVS WORD1, WORD2	M [DI] ← M[SI]	اگر WORD1 و WORD2 به منزله یک کلمه باشد

تصور کنید یک منطقه ای از حافظه میزان ۱۰۰ بایت را بخواهیم به منطقه دیگری منتقل کنیم استفاده از دستور MOVSB با پیشوند REP دستور مناسبی خواهد بود، به مثال ۳-۵ توجه کنید :

```
Example   5-3          ;
          LES   DI,List1   ;
          LDS   SI,List2   ;
          CLD              ;
          MOV   CX,100     ;
          REP   MOVSB      ;
```

دو دستور LES و LDS بکار برده شده اند تا سگمنت منبع اطلاعات و سگمنت مقصد اطلاعات و SI و DI را از آدرس مربوطه پر کنند. این آدرسها در محل هائی بنام LIST1 و LIST2 قرار دارند اگر DS و ES مقدار مناسب را داشته باشند بجای آن دو دستور از MOV می توان استفاده کرد و فقط OFFSET آدرس را در DI و SI قرار داد. شمارنده را هم (CX) با تعداد دیتائی که باید منتقل شود پر می کنیم دستور REP MOVSB مادامیکه CX برابر صفر نشده است تکرار می شود.

۵-۵- سایر دستورات انتقال دیتا

این دستورات عبارتند از OUT, IN, XLAT, SAHF, LAHF, XCHG که مهمترین دستورات انتقال هستند، کاربرد آنها کمتر از سایر دستورات است. و راجع به IN و OUT قبلاً مفصل صحبت شده است.

۱-۵-۵- دستور XCHG

این دستور محتوای یک ثبات را با محتوای ثبات دیگر یا حافظه عوض می کند، این دستور نمی تواند محتوای ثبات های سگمنت را و یا محتوای دو محل از حافظه را عوض کند. اما برای تعویض اطلاعات یک ثبات با یک محل از حافظه می تواند از تمام مدهای آدرس دهی ۸۰۸۶ استفاده نماید. جدول شمار ۱۱-۵ لیست تمام صورتهای دستور XCHG را نشان می دهد. حتی ترکیب بیت های باینری هر بایت را نشان می دهد. آنطور که از جدول برمی آید بهترین و مناسب ترین فرم این دستور تعویض محتوای AX با هر ثبات ۱۶ بیتی دیگر است، زیرا این دستور فقط یک بایت طول دارد و برای ذخیره کردن این دستور در حافظه یک محل ۸ بیتی کافی است.

جدول ۱۱-۵- فرم های مختلف دستور XCHG

فرم سمبلیک	BYTE1	BYTE2
XCHG AX,reg	10010rrr	-------
XCHG reg,reg	1000011w	11rraaa
XCHG reg,mem	1000011w	mmrrraaa

rrr=هر ثباتی می تواند باشد بجز ثبات های سگمنت

mm=مد آدرس دهی است (mod)

aaa=هر نوع آدرس دهی یا هر مدی می تواند باشد

w=بیانگر بایت یا Word بودن دیتای است

۲-۵-۵- دستورات LAHF, SAHF

علت استفاده از این دو دستور برای ۸۰۸۶ از این جهت است که این ریزپردازنده باید قادر باشد برنامه های نوشته شده برای ۸۰۸۵ را نیز اجرا کند. لذا در نوشتن برنامه برای ۸۰۸۶ کاربردی ندارند. عملکرد دو دستور SAHF, LAHF بطور کلی این است که بایت کم ارزش Flag، که شبیه FLAG ۸۰۸۵ است درون AH می ریزد یا از AH به درون ثبات Flag می ریزد این دستورات در ارتباط با PUSH AX, POP AX دو دستور PUSH PSW و POP PSW را در ۸۰۸۵ معادل سازی می کنند. ترتیب اجرای دستورات مثال ۴-۵ چگونگی این معادل سازی را نشان می دهند.

```
Example 5-4                 ;
        LAHF                ;   Flagرا درAHقرارمی دهد
        XCHG    AL,AH       ;   محتوایAH و AL را عوض می کند
        PUSH    AX          ;   A و F را در حافظه پشته ذخیره می کند
```

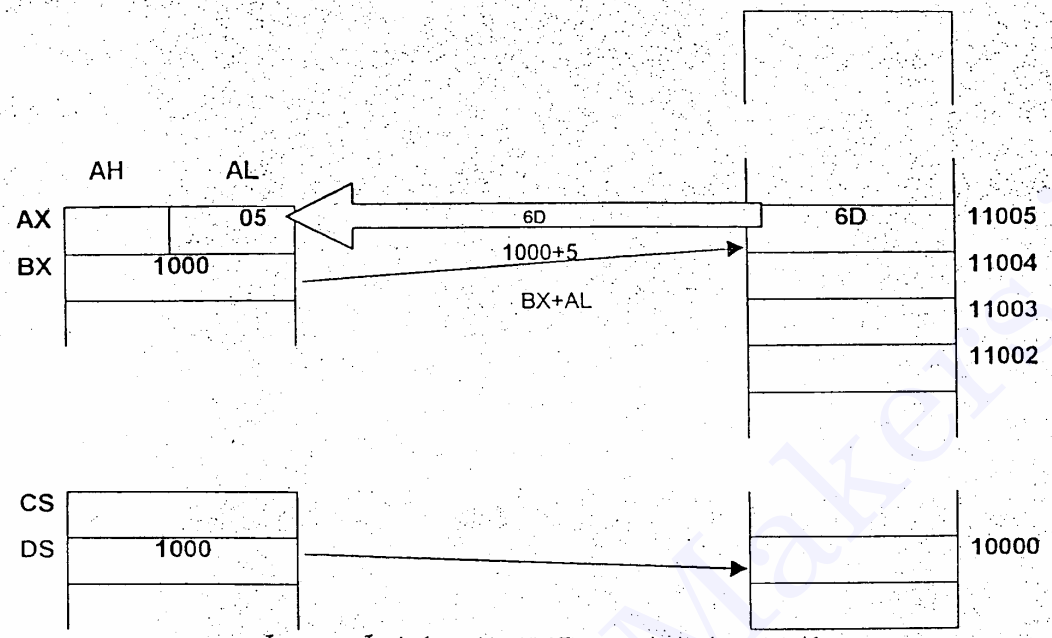
۳-۵-۵- دستور XLAT

این دستور محتوای AL را به یک عددی که در جدولی ذخیره شده است تبدیل می کند. در واقع این دستور با تکنیک LOOKUP TABLE کدی را به کد دیگر تبدیل می کند. یک دستور XLAT ابتدا محتوای AL را به محتوای ثبات BX اضافه می کند تا آدرس یک محل حافظه در DS را بدست آورد. سپس دیتای ذخیره شده در این آدرس را به ثبات AL منتقل می کند. فرض کنید بخواهیم کد BCD را به 7-SEG تبدیل کنیم و کدهای 7-SEG بصورت LOOKUP-TABLE در جدولی که آدرس اول آن در محلی بنام TABLE ذخیره شده باشد. مثال ۵-۵ این عمل را انجام می دهد و شکل ۱۳-۵ نحوه عملکرد این دستورات را نشان می دهد. اگر TABLE=1000H و DS=4000H و مقدار اولیه AL=05 (یعنی عدد ۵ در BCD) طبق شکل جواب بدست آمده برابر 6D می شود.

Example :

```
MOV     BX,OFFSET     TABLE
XLAT
```

بکار گرفتن دستوراتی که ترجمه یا تبدیل کد BCD به 7-SEG باشد.



شکل ۱۳-۵- تاثیر اجرای دستور XLAT و نمایش چگونگی آدرس دهی آن

۶-۵- عوض کردن سگمنت در یک دستور (Segment Override Prefix)

عوض کردن ثبات سگمنت تقریباً در تمامی دستور ۸۰۸۶ با هر مد آدرس دهی اجازه داده شده است. این عمل بوسیله اضافه کردن سگمنت بصورت پیشوند به آدرس امکان پذیر می باشد. و در زبان ماشین هم یک بایت به کد عملیاتی اضافه می شود. بعنوان مثال در دستور MOV AX,[DI] معمولاً در بخش دیتا سگمنت (DS) آدرس دهی می شود می توان این دستور را به شکل MOV AX,ES:[DI] تغییر و عوض کرد. که در این صورت آدرس دهی در بخش اکسترا سگمنت انجام می پذیرد یعنی :

آدرس فیزیکی = ES×10H+DI

جدول ۱۲-۵ لیست بعضی از دستوراتی که سگمنت را عوض می کنند. دقت داشته باشید که در اغلب این دستورات دیتا را در هر سگمنتی می توان یافت.

جدول ۱۲-۵- دستوراتی که سگمنت را عوض می کنند

سگمنتی که معمولاً باید در اختیار قرار می گرفت	سگمنتی که در اختیار قرار می گیرد	فرم سمبلیک
سگمنت پشته بود SS	دیتا سگمنت DS	MOV AX, DS:[BP]
سگمنت پشته بود SS	اکسترا سگمنت ES	MOV AX, ES:[BP]
دیتا سگمنت DS	سگمنت پشته SS	MOV AX, SS:[DI]
دیتا سگمنت DS	کد سگمنت CS	MOV AX, CS:[SI]
دیتا سگمنت DS	اکسترا سگمنت ES	MOV AX, ES:LIST

۷-۵- احکام اسمبلر (Assembler Directive)

احکام اسمبلر دستوراتی هستند که هدایت کننده اسمبلی بمنظور جهت دهی و پیشبرد برنامه اسمبلی هستند، در این قسمت، متن نشان می دهد که چگونه احکام اسمبلر را بکار بگیرید و هم اینکه چگونه برنامه به زبان اسمبلی بنویسید. این احکام زیاد هستند جدول زیر را در نظر بگیرید.

جدول ۱۳-۵- راهنماهای اسمبلی

WORD	FUNCTION
ALIGN	از یک محل حافظه به آدرس زوج شروع می کند.
ASSUMME	نشان می دهد که سگمنت کجا واقع شده است.
AT	سگمنت را در حافظه ذخیره می کند.
BYTE	همانند یک اپرند بایتی برخورد می کند.
DB	تعریف یک بایت (۸ بیتی) است.
DD	تعریف یک دیتا دو کلمه ای (32bit) است.
DQ	تعریف چهار کلمه ای است (64bit).
DT	تعریف دیتای ۱۰ بایتی است (80bit).
DUP	کاراکترهای زیری دو برابر شده هستند.
DW	تعریف word کلمه (۱۶ بیتی) است.
END	نشان دهنده پایان لینست است.
ENDP	نشان دهنده پایان زیربرنامه است.
ENDS	نشان دهنده پایان سگمنت است.
EQU	معادل یا مساوی است.
FAR	بیان عوض شدن سگمنت است.
NEAR	وجود دیتا در همین سگمنت است.
ORG	مشخص کردن آفست آدرس است.
PROC	بمنزله شروع یا لول سگمنت است.
PTR	تعریف شروع زیر برنامه است.
SEGMENT	بمنزله اشاره گر به حافظه است.
STACK	طرح شروع سگمنت است.
THIS	نشان دهند قسمت پشته است (SS)
	بکار می رود برای بیان THIS BYTE یا THIS WORD

توجه داشته باشدی که اغلب این کلمات کلیدی (KEY WORD) توسط اغلب اسمبلی های ۸۰۸۶/۸۸ پذیرفته شده اند.

۱-۷-۵- راهنماها :

اغلب اسمبلی های ۸۰۸۶ اغلب کلمات جدول ۱۳-۵ را تشخیص می دهند و متوجه می شوند که با یک اپرند چگونه برخورد نمایند یا یک برنامه را چگونه اجرا نمایند. در مثال ۶-۵ استفاده از DD, DW, DB بیان شده است.

```
Example   5-6                                  ;
        DATA-ONE   DB    1,2,3        ;   سه بایت تعریف شده بمنزله ۱و۲و۳
                   DB    45H          ;   یک بایت عدد ۴۵ را در پایه ۱۶ نشان می دهد
                   DB    'A'          ;   کد اسکیAرا اعلام می دارد
                   DB    11110000B    ;   یک بایت را بصورت باینری معرفی شده
                   DW    12,13        ;   دو کلمه ۱۶ بیتی معرفی شده ۱۲و۱۳
        DATA-TWO   DW    LIST         ;   آفست آدرسی بنامLISTتعریف شده
                   DW    3456H        ;   یک کلمه ۱۶ بیتی با مقدار3456Hمعرفی شده
                   DD    0000FFFFH    ;   دو کلمه ۱۶ بیتی معرفی شده
```

همانگونه که در مثال ۶-۵ نشان داده شده است مشخصه هائی مثل DB و DW یا DD قادرند یک بایت یا یک کلمه ۱۶ بیتی یا دو کلمه ۱۶ بیتی از دیتا را در حافظه ذخیره کنند. حافظه را می توان حتی برای دیتاهائی که در آتی بدست می آوریم توسط DB ,DW ,DD بوسیله راهنمایی DUP رزرو کرد.

DUP به اسمبلر نشان می دهد که یک کارکتری که درون () گذاشته شده است باید دو برابر شود. عدد قبل از پرانتز تعداد دفعات دو برابر شدن را نشان می دهد. اگر درون پرانتز علامت ؟ قرار داده شود، حالا دیگر معنی دو برابر شدن کارکتر را نمی دهد بلکه می گوید حافظه بدون تغییر ترک شود به مثال ۷-۵ توجه کنید بعضی از مثال های راهنمای DUP را نشان می دهد.

```
Example    5-7                                 ;
           LIST-A    DB      ?          ;   یک بایت برایLIST-Aذخیره می کند
           LIST-B    DB      10DUP(?)   ;   ۲۰بایت ذخیره می کند
                     ALIGN              ;   یک آدرس زوج می سازد
           LIST-C    DW      10DUP(?)   ;   ۲۰ کلمه ذخیره می کند
           LIST-D    DD      22DUP(?)   ;   ۲۲ تا دو کلمه ذخیره می کند
           ZEROS     DB      100DUP(0)  ;   ۱۰۰ عدد صفر درZEROSذخیره می کند
```

۲-۷-۵- راهنماهای EQU و THIS

عبارت EQU (Equavte) برای ارزیابی دو داده عددی که ASCII یا ارزیابی یک برچسب (Lable) با برچسب دیگر بکار می رود. دستور EQU اسمبلر برنامه را مجبور می کند تا پس از زمانی آنرا اشکال زدائی کند مثال ۸-۵ چند نوع مختلف عبارت EQU را نشان می دهد :

```
Example 5-8
                                        ;
TEN          EQU   10                   ;   برچسب هایTENرا مساوی 10می کند
LETTER-A     EQU   'A'                  ;   برچسبLETTER-A رامساوی 'A' می کند
COUNT        EQU   99                   ;   برچسبCOUNT را مساوی 99می کند
NUMBER-10    EQU   TEN                  ;   برچسبNUMBER-10 را مساوی10می کند
```

راهنمایTHIS یک نام برچسب را بدون تعریف یک محل حافظه جدیدی نشان می دهد. بعنوان مثال وقتی که یک پشته برقرار می شود ثبات اشاره گر به پشته (SP) دارای مقدار آدرس TOP پشته می شود این کار توسط راهنماهای EQU, THIS انجام می شود مثال ۹-۵ را با دقت ببینید، چگونه واژه WORD را با THIS و EQU نشانده TOP پشته می باشند.

```
Example  5-9
                                        ;
STACK-MEM    DW    50 DUP(?)             ;   ۵۰کلمه رزرو می کند
STACK-TOP    EQU   THIS WORD             ;   TOP OF STACKاست
             DB    10                    ;   این ۱۰ را اینجا ذخیره کن
DATA         EQU   THIS BYTE             ;   DATA=10است
```

۳-۷-۵- سازمان حافظه

اسمبلر برای خبر دادن از وجود سگمنت های مختلف راهنماهای خاص را بکار می گیرد. اسامی راهنماهای سگمنت ها شروع هر سگمنت و کلمه های ENDS نشان دهنده پایان آنهاست. وقتی که یکSEGMENT و ENDS بیان گر شروع و پایان سگمنت باشد کلمه ASSUME به اسمبلر اطلاع می دهد که کدام یک از DS یا CS یا ES و SS ساخته شده است.

مثال ۱۰-۵ چگونگی کاربرد این راهنماها را نشان می دهد. سیستم عامل معمولاً ثبات های کد سگمنت (DS) و اکسترا سگمنت (ES) را پر نمی کند. این ها باید با شروع هر برنامه پر شوند. که در این صورت ثبات AX با آدرس سگمنت دیتا پر می شود و سپس با دستور MOV به DS ریخته خواهد شد. توجه داشته باشید که راهنمای OFFSET در اینجا کاربرد ندارد. اسمبلر بطور اتوماتیک OFFSET را برای هر برچسب تعریف شده بعنوان اسم سگمنت اعمال می کند. ES نیز به همین طریق که ذکر شد پر می شود. اگر مجبور باشیم که سگمنت را در هر بخش جزئی از حافظه قرار داده شود از راهنمای AT استفاده می شود.

مثال ۱۱-۵ نشان می دهد که چگونه عبارت AT سگمنت دیتا را بوسیله پر کردن ثبات سگمنت از 0100H در 0100H قرار می دهد. و عملاً محل حافظه آدرس 01000H خواهد شد.

مثال ۱۰-۵ نشان دهنده چگونگی استفاده از ENDS-SEGMENT و ASSUME

```
Example   5-10                          ;
          DATA    SEGMENT               ; اسم سگمنت
          ;
          LIST-A  DB    10   DUP(?)     ; ۱۰ بایت رزرو می کند
          LIST-B  DB    12   DUP(6)     ; از ۶تا۱۲کلمه ذخیره می کند
          ;
          DATA    ENDS                  ; پایان سگمنت
          ;
          EXTRA   SEGMENT               ;
          ;
          STRING  DB    'HELLO'         ; HELLOرا ذخیره می کند
          ;
          EXTR    ENDS
          ;
          STACK   SEGMENT
          ;
                  DW    50   DUP(?)     ; ۵۰ کلمه رزرو می کند
          ;
          STACK   ENDS
          ;
          CODE    SEGMENT
          ;
                  ASSUME    CS : CODE   DS : DATA
                  ASSUME    EX : EXTRA  SS : STACK

          BEGIN :
                  MOV   AX,DATA         ; DS رااز DATAپرمی کند
                  MOV   DS,AX           ;
                  MOV   AX,EXTRA        ; ES رااز EXTRA پرمی کند
                  MOV   ES,AX           ;
                  ----  ---
                  ----  ---
                  ----  ---
          CODE    ENDS
                  END   BEGIN
```

یک نمونه برنامه برای نشان دادن راهنماهای ORG, AT

```
Example   5-11                          ;
;
DATA      SEGMENT   AT    0100H         ; اختصاص دادن آدرس سگمنت
;
          ORG       100H                ; اختصاص دادنOFFSETبه آدرس
;
LIST-A    DW        10 DUP(?)           ; ۱۰کلمه ۱۶ بیتی رزرو کردن
;
DATA      ENDS
```

در مثال بالا نه تنها دیتا در آدرس سگمنت ذخیره شد. بلکه LIST-A همچنین شروعش را در آدرس0100H قرار می دهد. که با قرار دادن DS=100H محل فیزیکی حافظه 01100H خواهد شد. در اغلب برنامه ها واژه های ORG, AT بکار برده نمی شود.

۴-۷-۵- زیربرنامه (SUBROOTIN/PROCEDURE)

زیربرنامه بوسیله راهنماهای PROC و ENDP نشان داده می شود.PROC نام شروع زیربرنامه همراه با نوع زیربرنامه (FARیاNEAR) را نشان می دهد. ENDS نشان دهنده آخر زیربرنامه است. بعنوان مثال تصور کنید که محتوای ثبات های DX, CX, BX باید با هم جمع شود و حاصل جمع در AX ریخته شود. این کار توسط زیربرنامه ADDEM در مثال ۱۲-۵ بطور کامل نشان داده شده است.

زیربرنامه ای که محتوای DX, CX, BX را جمع می کند و حاصل جمع را در AX ذخیره می کند.

```
;
ADDEM       PROC    FAR         ;       معرفی زیربرنامه
;
            MOV     AX,BX       ;
            ADD     AX,CX       ;
            ADD     AX,DX       ;
            RET                 ;       مراجعه از زیربرنامه به برنامه اصلی
;
ADDEM       ENDP                ;       پایان زیربرنامه
```

همانگونه که ملاحظه می فرمائید زیربرنامه با نام ADDEM و از نوع FAR معرفی شده است. معنی آن این است که در هر محلی از حافظه می تواند قرار داده شود. یعنی هم باید آدرس OFFSET و هم آدرس سگمنت تعیین شود. اگر بصورت NEAR معرفی شود باید در همان سگمنت قرار داده شود و فقط آدرس OFFSET آن مشخص گردد. اگر دنبال LABLE دو نقطه : (COLON) ظاهر شود، زیربرنامه از نوع NEAR است ولی اگر : ظاهر نشود از نوع FAR است. طبیعی است که اجرای زیربرنامه NEAR راحتر و سریعتر صورت می گیرد.

۵-۷-۵- آدرس شروع و اشاره گر (OFFSET and PTR)

راهنمای OFFSET نشان دهنده یک آدرس شروع (OFFSET) به اسمبلر است. راهنمای PTR که مخفف POINTER است نوع دیتا (BYTE یا WORD) را به اسمبلر معرفی می کند.

بعنوان مثال آدرس DATA را توسط دستور MOV SI, OFFSET DATA در SI ریخته، در صورتیکه دستور MOV SI, DATA محتوای محلی از حافظه که آدرسش DATA است درون SI می ریزد ولی OFFSET، آدرس دیتا را درون SI می ریزد.

زمانیکه نوع (TYPE) دیتا برای اسمبلر مشخص نباشد از دستور PTR استفاده می شود، بعنوان مثال تصور کنید که دستور زیر در یک برنامه ظاهر شود MOV [BX],2 این دستور یک بایت یا ۱۶ بیت دیتا که مقدرا آن 2 است در حافظه ذخیره می کند. راهنمای PTR با کلمه BYTE یا WORD این موضوع را روشن خواهد کرد. لذا دستور خواهد شد MOV BYTE PTR[BX],2 که یک بایت حافظه را برای ذخیره کردن عدد 2 بکار می گیرد. اما MOV WORD PTR[BX],2 یک محل ۱۶ بیتی از حافظه را که BX آدرس آنرا می دهد برای ذخیره کردن عدد 2 بکار می گیرد.

۶-۷-۵- یک نمونه برنامه

مثال ۱۳-۵ یک نمونه برنامه است که یک BLOCK اطلاعات را به BLOCK دیگری منتقل می کند. برای اینکار از دستور MOVSB استفاده می کند. این مثال بکارگیری خیلی از راهنماهائی که در این بخش صحبت شد را نشان می دهد. توجه دارید که COUNT مساوی 100 است که با استفاده از دستور MOV CX,COUNT در شمارنده قرار می گیرد. همچنین توجه دارید که آخرین دستور INT 20H. این دستور نوعی وقفه است که کنترل کامپیوتر را به سیستم عامل DOS برمی گرداند. در آینده راجع وقفه های مختلف در ۸۰۸۶ صحبت خواهیم کرد.

مثالی از زیربرنامه نویسی که ۱۰۰ بایت از بلوک ۱ را به بلوک ۲ منتقل می کند.

```
Example     5-13                                  ;
COUNT          EQU      100                       ;   تعریف شمارنده
DATA        SEGMENT
BLOCK-1     DB       100 DUP(?)                   ;   ۲۰۰ بایت رزرو می کند
DATA        ENDS
EXTRA          SEGMENT
BLOCK-2     DB       100 DUP(?)                   ;   ۲۰۰بایت رزرو می کند
EXTRA        EDNS
CODE        SEGMENT
            ASSUME      CS : CODE,  DS : DATA, ES : EXTRA
BEGIN
            MOV      AX,DATA                      ;   پرکردنDSاز آدرس مربوطه
            MOV      DS,AX                        ;
            MOV      AX,EXTRA                     ;   پرکردنESاز آدرس مربوطه
            MOV      ES,AX                        ;
;
            MOV      SI,OFFSET BLOCK-1            ;   آدرس دیتا منبع
            MOV      DI,OFFSET BLOCK-2            ;   آدرس مقصد
            CLD                                   ;   انتخاب خودافزایشی آدرسها
            MOV      CX,COUNT                     ;   شمارنده را پر می کند
            REP
            MOVSB                                 ;   ۱۰۰بایت انتقال می دهد
            INT         20H                       ;   برمی گردد بهDOS
CODE        ENDS
            END      BEGIN
```

سوالات و مســائل

۱-اولین بایت هر دستور ۸۰۸۶ چه نام دارد؟

۲-هدف از بیت های W, D چیست؟ شرح دهید.

۳-میدان MOD در یک دستور به زبان ماشین چه اطلاعاتی دارد؟

۴-اگر میدان REG یک دستور 010 و W=0 چه ثباتی توسط دستور انتخاب می شود؟

۵-اگر مقدار R/M=001 و MOD=00 باشد چه مد آدرس دهی تعریف شده است؟

۶-بیان کنید سگمنت معمولی (Default Segment) برای هر یک از ثبات های زیر کدام است ؟

الف: SP ب: BX ج: BP د: DI هـ : SI

۷-دستور 8B07H را به زبان اسمبلی تبدیل کنید.

۸-دستور 8B1E9004CH را به زبان اسمبلی تبدیل کنید.

۹- دستور MOV SI,[BX+2] را به زبام ماشین تبدیل کنید.

۱۰- آیا دستور MIOV CS,AX مجاز است پاسخ خود را شرح دهید.

۱۱- چه نوع دستور MOV دارای ۶ بایت طول خواهد بود.

۱۲- دستورات POP, PUSH همیشه چند بیت در حافظه قرار می دهند؟

۱۳- چه ثبات سگمنتی را نمی توان از حافظه POP کرد؟

۱۴- شرح دهید هنگام اجرای دستور PUSH BX چه عملی اتفاق می افتد، فرض کنید که SS=0200H, SP=100H باشد BL, BH در چه آدرسهائی ذخیره می شود؟

۱۵- عملکرد دستور MOV BX, DATA را با دستور LEA BX, DATA مقایسه کنید؟

۱۶- شرح دهید دستور LDS BX, DATA چگونه عمل می کند؟

۱۷- نقش فیلیپ فلاپ D چیست ؟

۱۸- دستوراتی که باعث تغییر وضعیت D می شوند کدام است؟

۱۹- برنامه ای بنویسید که ۱۲ بایت اطلاعات را از محلی از حافظه به آدرس SOURCE به محل دیگر از حافظه به آدرس DEST که هر دوی آنها دارای ۳۲ بیت آدرس دارند منتقل نماید؟

۲۰- با استفاده از راهنمای اسمبلی ۳۰ بایت از حافظه را برای LIST1 رزرو کنید.

۲۱- توضیح دهید چرا PTR در دستوراتی مثل MOV WORD PTR[DI],3 ضروری است.