میکرو کنترلر ها

مطمئنا همگی شما لزوم وجود کامپیوتر را در زندگی احساس کرده اید .این لزوم شاید تنها به خاطر دریافت صورتحساب های تلفن ، برق ، گاز و ... باشد یا این ینکه این لزوم به خاطر شغل شماست شاید شما یک برنامه نویس سیستم باشید یا یک طراح وب .
به هر حال مهم نیست که روزانه چقدر با کامپیوتر در گیر هستید چون همه لزوم وجود آن را درک کرده اید.می خواهیم پا به دنیای دیگری از شاهکار های دیجیتال پا بگزاریم که شاید تا کنون تمیزی بین آن و کامپیوتر قرار نمی دادید. دنیای میکرو کنترلر ها . در ین مقاله ابتدا ساختمان یک میکرو کامپیوتر را بررسی می کنیم و سپس تفاوت بین میکرو پروسسور و میکرو کنترلر را خواهیم گفت .

مقدمه:

اولین ریز پردازنده ی که با موفقیت عملی به بازار عرضه شد محصول شرکت ینتل در سال 1971 با نام 8080 بود بعد از آن بازار رقابت میکرو پروسسوری گرم شد و شرکت هی دیگری همچون RCA ,ZILOG,MOS technology ریز پردازنده های خود را به بازار عرضه کردند.
کم کم با ورود کامپیوتر به صنعت ،وجود تکنولوژی دیگری نیز احساس شد . تکنولوژیی که از میکرو پروسسور جمع و جور تر و کند تر بود اما هزینه را خیلی پایین می آورد .این تکنولوزی میکرو کنترلر نام گرفت و کار آن نظارت و کنترل یک سری اعمال بود که توسط برنامه نویسی انجام می شد.
امروزه میکرو کنترلر حوزه های وسیعی از زندگی مارا اشغال کرده است بی آنکه بدانیم که سیستم به کار رفته در آنها میکرو کنترلری است و به اشتباه همه ی آنها را کامپیوترمی نامیم.برای مثال ماشین لباسشویی که اتمام کار خود را با نواختن آهنگی اعلام میکند یا اجاق گازی که خاموش شدن شعله را با آهنگ گوشزد می کند و یا خودروی سمندی که باز بودن در خودرو را با جمله ی**" در ب خودرو باز است."**یاد آوری می کند(البته با فرکانس بالاتر از فرکانس صدای من!!!) همگی نمونه هی ملموسی از سیستم هی میکرو کنترلری هستند که روزانه بار ها با آنها سر و کار داریم.

تا ینجا مطمئنا متوجه شده ید که بری چه وقت پر ارزش خودتون را دارید صرف خواندن ین مقاله می کنید؟ بله ! چون آینده صنعت در دست سیستمهی میکرو کنترلری است . وقتی کلمه میکرو کنترلر را به کار می برم منظورم ic 8051 نیست بلکه مجموع خانواده ی میکرو کنترلر ها و تکنولوژی هی بعدی می باشد مانند FPGA,PIC و... اما در ین سری

مقالات به خاطر فقر علمی! فعلا با IC 8051 کار می کنیم.

ملزومات یک سیستم کامپیوتری:

چون میکرو پروسسور و میکرو کنترلر اگر پدر و فرزند نباشند مطمئنا برادر هستند بری آشنایی با میکرو کنترلر ،کمی در مورد میکروپروسسور صحبت می کنیم چون بری اکثر شما ملموس تر است .در پیان شباهت میکرو کنترلر و میکرو پروسسور را خواهیم گفت .

هنگام که **چارلز بابیج(پدر کامپیوتر)** شمای کلی سیستم کامپیوتری را ارائه داد و عنوان کرد که اگر ماشینی خواهد که دستورات انسان را اجرا کند باید شامل این بلوکها باشد (بلو کها را خواهم گفت) اما در زمان آقای بابیج امکانات آن زمان جوابگوی پیاده سازی آن سیستم نبود و به خاطر این آقای بابیج از ینکه به طرح خود جامه عمل بپوشاند باز ماند.بعدها که اولین کامپیوتر ساخته شد از طرح آقای بابیج الگو گرفتند و واحد هایی را برای این سیستم تعریف کردند سازندگان کامپیوتر هم دقیقا این واحد ها را در سیستم خود پیاده سازی می کردند .اولین شرکت سازنده کامپیوتر IBM بود و استاندارد سازی هم بر اساس طرح آنها صورت گرفت لغت "سازگار با IBM" که تا چند سال پیش به کار می رفت به همین دلیل بود.اما واحد های استاندارد یک سیستم کامپیوتری :

1-واحد پردازش مرکزیcpu :

همان مغز سیستم است و فعالیت هی سیستم را کنترل می کند و عملیات هایی را بر روی داده ها انجام می دهد . cpu مجموعه ی از مدارات منطقی است که به طور متناوب دستورات را واکشی و اجرا می کنند.

Cpu خود دارای چند قسمت می باشد:

الف- ALU یا واحد محاسبه و منطق که مسئول انجام اعمال محاسباتی نظیر جمع و تفریق و ... و اعمال منطقی مانند مقیسه And و OR و ... می باشد

ب – واحد کنترل که مسئول رمز گشایی و تعیین نوع عملیاتی است که ALU باید انجام دهد.

ج- ثبات ها جهت ذخیره موقت داده ها قبل از رفتن به ALU و همچنین نگه داری نتایج پردازش به کار میروند.

د – PC یا شمارنده برنامه که آدرس دستورالعمل بعدی که CPU باید از حافظه بخواند را در خود نگه می دارد.

ه- IR یا ثبات دستور العمل که مسئول ذخیره قسمت عملیاتی دستورالعمل فعلی می باشد
بعد از معرفی دیگر واحد ها به تفصیل در مورد واکشی دستور بحث می کنیم.

2. حافظه :

حافظه محل ذخیره اطلاعات است . حافظه ها را بر اساس مشخصه های مختلفی می توان طبقه بندی کرد . معمولترین طبقه بندی ، طبقه بندی بر اساس حافظه هی اولیه و ثانویه است .
حافظه هی اولیه حافظه هایی هستند که در اختیار سیستم هستند و سیستم بری انجام اعمال پردازشی به آنها احتیاج دارد ROM و RAM از حافظه های اولیه هستند .

ROM (رام) یا حافظه فقط خواندنی توسط کارخانه برنامه نویسی شده و محتویات آن توسط کاربر یا برنامه نویس تغییر نمی یابد . در حقیقت ROM یک بار برنامه نویسی شده و بارها و بارها می توان آن را خواند .محتویات ROM ثابت است و با قطع برق از بین نمی رود.

RAM (رم) حافظه با دستیابی تصادفی است که CPU اطلاعات را در آن ذخیره می کند و محتویات آن توسط برنامه نویس هم می تواند تغییر داده شود محتویات آن با قطع جریان برق هم از بین می رود.

حافظه های ثانویه ، حافظه هایی هستند که ما اطلاعات را بر روی آنها ذخیره می کنیم تا بعدا دوباره آن اطلاعات را مورد استفاده قرار دهیم و با قطع برق هم محتویات آنها از بین نمی رود .در حقیقت حافظه های ثانویه ترکیبی از خصوصیات Rom و Ram هستند . پایداری خود در مقابل قطع برق را از ROM و توانایی تغییر محتویات توسط کاربر را از RAM به ارث برده اند .برای حافظه های ثانویه می توان از HARD DISK و FLOPPY نام برد

3- گذرگاه یا باس:

مجموعه ای از سیم ها که اطلاعات را با یک هدف مشترک حمل می کنند.در معماری کامپیوتر از سه باس می توان نام برد:

الف – باس آدرس :برای هر عمل خواندن یا نوشتن CPU آدرس (موقعیت) داده را با قرار دادن یک آدرس روی باس آدرس به حافظه ارسال می کند.

ب - باس کنترل :با قرار دادن آدرس بر روی باس آدرس یک سیگنال کنترلی بر روی گذرگاه کنترل قرار می گیرد که مشخص می کند که عملیات مورد نظر خواندن است یا نوشتن در حافظه.

ج- گذرگاه داده : بری عملیات خواندن یک بیت داده بر روی گذرگاه داده توسط حافظه قرار داده می شود و بری عمل نوشتن یک بیت داده توسط CPU بر روی گذرگاه داده قرار می گیرد .
علاوه بر سیگنالهای کنترلی READ و WRITE سیگنال کنترلی دیگری هم با نام CLOCK وجود دارد که بری هم زمان کردن واحدها به کار می رود.

4 – واحد هی ورودی و خروجی :

ین واحد مسئول ارتباط با دنیای خارج است و توانایی اتصال سیستم ها را به کامپیوتر می دهد .

از کامپیوتر با نام ماشین برنامه پذیر نام بردیم . حال ببینیم برنامه چه تعریفی دارد.

برنامه:

به مجموعه ی از دستور العمل ها که ترتیب اجزای مشخصی دارند و داری نقطه شروع و پیان مشخصی هستند که به منظور انجام عمل خاصی نوشته شده اند یک برنامه می گویند.

از تعریف بالا نتیجه گرفته می شود که واحد ساختمانی برنامه دستور العمل است درست مانند یک ساختمان که به وسیله

آجر ساخته می شود برنامه هم با قرار دادن دستور العمل ها پهلوی هم ساخته می شود.

عمل خواندن دستور العمل ها از حافظه توسط CPU را واکشی گویند.دستور العملها پشت سر هم واکشی شده و توسط CPU اجرا می شوند تا زمانی که برنامه اتمام برسد و آن هدفی را که برنامه نویس مد نظرش بوده است انجام شود.

CPU تنها با داده های باینری کار می کند بنابرین هر دستور العمل نوشته شده توسط برنامه نویس باید به مقدار معادل باینری(همان زبان ماشین) آن تبدیل شود و سپس اجرا شود.
یک سیکل واکشی دستور به صورت زیر می باشد:

1- مقدار PC یا شمارنده برنامه بر روی گذرگاه آدرس قرار می گیرد .
2- سیگنال کنترلی READ بر روی گذرگاه کنترل قرار می گیرد
3- داده (کد عملیاتی دستورالعمل) از حافظه خوانده می شود و روی گذرگاه داده قرار می گیرد
4- کد عملیاتی در ثبات IR قرار می گیرد
5- شمارنده برنامه به ابتدی دستور بعدی اشاره میکند(اغلب گفته می شود که شمارنده برنامه یک واحد افزیش می یابد اما گاهی اوقات 2 واحد و گاهی 3 واحد افزیش پیدا می کند در حقیقت شمارنده برنامه به اندازه طول دستور فعلی افزیش پیدا می کند.)

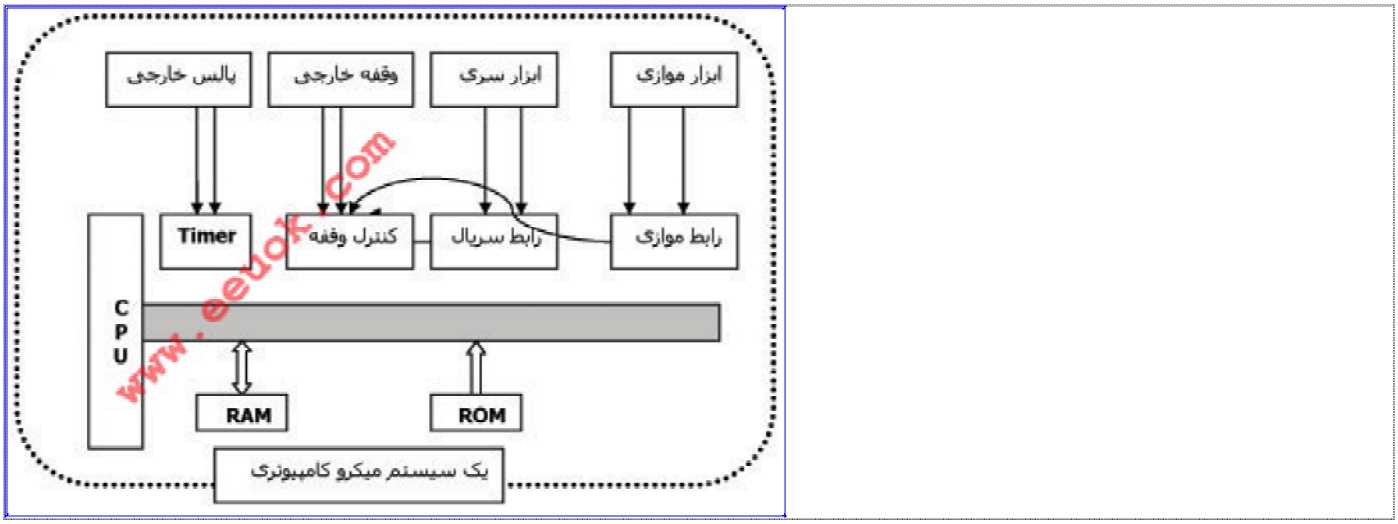
مرحله ی اجرا شامل کد گشایی و ایجاد سیگنالهای کنترلی لازم بری باز کردن ثبات های CPU و قرار دادن محتویات آنها در ALU و مجددا قرار دادن نتیجه عملیات در ثبات ها است.

مقایسه ریز پردازنده و میکرو کنترلر :

ریز پردازنده یک IC است که فقط شامل CPU است اما میکرو کنترلر مدارات اضافی دیگری در داخل همان IC قرار داده شده است که میکرو کنترلر را تبدیل به یک میکرو کامپیوتر کرده است (البته در مقیاس کمتر) مدارات دیگری همانند ROM,RAM,timer, و رابط سریال و پردازش وقفه و...
اما برای ینکه یک میکروپروسسور به عنوان یک میکرو کامپیوتر عمل کند به مدارات ذکر شده بالا در خارج IC نیازمندیم که این مدارات هم حجم و هم هزینه را به طور نمایی افزیش می دهند .

اما ریز پردازنده و میکرو کنترلر از نظر معماری داخلی به هیچ وجه قابل مقایسه نیستند ریز پردازنده بری انجام اعمال پردازشی بسیار پیچیده در سرعت بالا طراحی شده است و نتیجتا معماری آن بسیار پیچیده تر از معماری میکرو کنترلر خواهد بود .

تفاوت دیگر میکرو کنترلر و میکرو پروسسور در مصرف توان آنهاست .میکرو کنترلر توان بسیار کمتر از توان مصرفی میکرو پوسسور لازم دارد چون مدارات آن نسبت به میکروپروسسور بسیار کمتر است و این یک مزیت برای طراحی مدارات می باشد.



میکرو کنترلر ها

اولین خانواده میکرو کنترلر ها با نام MCS-51 توسط شرکت ینتل طراحی و ساخته شد .بعد ها شرکت هی دیگری تحت مجوز ینتل شروع به تولید IC هی میکرو کنترلر کردند . از خانواده MCS-51 اولین عضو 8051 می باشد مشخصات ین IC به صورت زیر است:

1 - 4 کیلو بیت ROM

2- 128 بیت RAM

3 - 4 پورت ورودی و خروجی 8 بیتی

4 - دو تایمر/شمارنده 16 بیتی

5 - رابط سریال بری ارتباط با دیگر وسایل

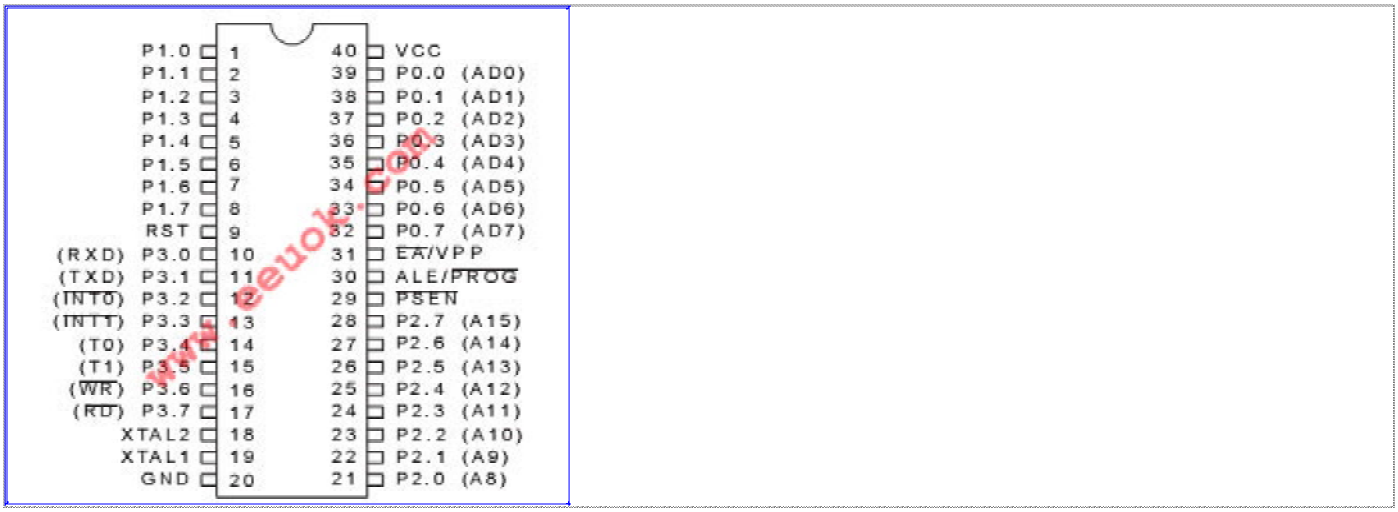
6 - 64 کیلو بیت حافظه کد خارجی و 64 کیلو بیت هم حافظه داده خارجی را می تواند آدرس دهی کند

7 - پردازنده بولی بری انجام اعمال بیتی

8 - 210 مکان بیت آدرس پذیر

پیه ها :

ین Ic همانگونه که در شکل می بینید داری 40 پایه می باشد پایه 20 به زمین و پایه شماره 40 به منبع تغذیه 5 ولت متصل می شود.ین میکرو کنترلر داری 4 پورت I/O (ورودی - خروجی) 8بیتی می باشد که 32 پایه از 40 پایه را به خود اختصاص داده اند به جز پورت شماره 1 سه پورت دیگر دو کاره هستند و علاوه بر عمل ورودی - خروجی کار دیگری را نیز بر عهده دارند.



پورت شماره صفر:

ین پورت پایه های شماره 32 تا 39 را شامل می شود .در برنامه های کوچک عموما وظیفه ورودی و خروجی داده را بر عهده دارد اما در پروژه های بزرگ بیت پیین آدرس و داده را انتقال می دهد . ین پورت به عنوان آدرس و داده مالتی پلکس عمل می کند که در استفاده از حافظه کد و داده خارجی کاربرد دارد .

پورت شماره 2:

(پایه هی شماره 21 تا 28) همانند پورت شماره صفر یک درگاه دو منظوره است و بیت بالی آدرس را در استفاده از حافظه خارجی انتقال می دهد.

پورت شماره 3:

علاوه بر ورودی و خروجی بودن ، هر یک از پایه هی آن داری عملکرد بخصوصی هستند که در جدول زیر آمده است .

:PSEN

(پایه شماره 29) این پایه فعال صفر است (یعنی در صورتی که به این پایه صفر منطقی بدهیم عملکرد تعریف شده برای این پایه انجام میشود).PSEN مخفف Program Storage Enable است در صورتی که از حافظه خارجی برای کد استفاده کنیم این پایه به پایه EPROM (Output Enable) OE حاوی کد متصل می شود و میکرو کنترلر بدین ترتیب می تواند برنامه را از حافظه EPROM بخواند .

:ALE

(پایه شماره 30) این پایه هم فعال صفر است .مخفف Address Latch Enable می باشد در توضیح پورت شماره صفر گفتیم که ین پورت به عنوان آدرس و داده مالتی پلکس استفاده می شود. هنگامی که ALE فعال باشد پورت شماره صفر در نیم سیکل اول آدرس را نگه می دارد و در نیم سیکل بعدی حافظه داده را نگه می دارد به عبارت دیگر ین پورت در نیم سیکل اول به عنوان گذرگاه آدرس و در نیم سیکل بعدی به عنوان گذرگاه داده عمل می کند .

:EA

(پایه 31)فعال صفرو مخفف External Access می باشد این پایه به صفر ولت یا 5 ولت متصل می شود . در صورتی که به 5 ولت وصل شود برنامه از ROM داخلی میکرو کنترلر اجرا می شود و در صورتی که به صفر ولت متصل سود برنامه از EPROM خارجی اجرا می شود.

:RST

(پایه 9) سیستم را ریست میکند . در صورتی که ین پایه 5 ولت به آن اعمال شود برنامه از اولین دستور مجددا اجرا می شود . ین دکمه مشابه دکمه ریست کامپیوتر می باشد .

حافظه ی 8051:

بانک های ثبات :

8051 داری 8 ثبات با نام های R0-R7 می باشد دستوراتی که از این ثبات ها استفاده می کنند نسبت به دستورات مشابه خود که از انواع دیگر آدرس دهی استفاده می کنند دارای تعداد بیت کمتر و سرعت بیشتری هستند بنابرین در صورتی که از داده ی به طور متناوب در برنامه استفاده می کنیم بهتر است که این داده در یکی از اثبات های بانک ثبات قرار گیرد .

در 8051 ، 4 بانک ثبات وجود دارد که در هر لحظه از زمان می توان فقط یکی از ین بانکها را فعال و از آن استفاده نمود .برای فعال کردن یک بانک ثبات از دو بیت با نامهای PSW.3, PSW.4 استفاده می کنیم ان دو بیت اجزائ ثبات PSW هستند.

ثبات PSW:

مخفف Program status word یا کلمه وضعیت سیستم است . این ثبات در هر سیکل ماشین بسته به وضعیت و جواب برنامه تغییر می کند و برنامه نویس می تواند بر اساس این تغییرات منطق برنامه را تغییر دهد .PSW مشابه Flag های ریز پردازنده هی سیستم های میکرو کامپیوتری می باشد.

بیت P:

این بیت بری تنظیم توازن زوج مقدار آن صفر یا یک می شود .در صورتی که تعداد یک هی داخل Accumulator فرد باشد ین بیت یک می شود تا مجموع یک ها زوج باشد این یک روش ساده تشخیص تعداد فرد خطا در انتقال اطلاعات می باشد. بری مثال اگر Accumulator عدد 00000011 را در مبنی دو در خود داشته باشد چون در ین مثال تعداد 1 ها دو عدد می باشد بنابرین بیت توازن صفر می شود تا تعداد یک ها زوج باقی بماند .

بیت OV:

ین بیت با نام Over flow (سر ریز (نام دارد .اگر هنگام جمع یا تفریق حسابی (اعداد علامتدار) نتیجه از +127 بیشتر یا از -127 کمتر شود ین بیت 1 می شود.

بیت RS1,RS0 :

ین دو بیت به منظور فعال کردن یکی از چهار بانک ثبات مورد استفاده قرار می گیرد.مقدار پیش فرض ین دو بیت 00 است که بانک ثبات صفر را فعال می کند .

بیت F0:

ین بیت با نام پرچم صفر هم شناخته شده است .اگر نتیجه ی یک عمل حسابی صفر شود ین بیت یک می شود.

بیت AC:

(بیت نقلی کمکی) در هنگام انجام عملیات حسابی در صورتی که از بیت شماره 3 به شماره 4 رقم نقلی داشته باشیم در این صورت این بیت یک می شود .

بیت CY :

(بیت نقلی) اگر در طول عملیات حسابی از بیت 7 رقم نقلی به بیت بالا تر داشته باشیم ین بیت یک می باشد .

ثبات A:

Accumulator یا انباره نامیده می شود.این ثبات همه منظوره است بیشتر دستورات میکرو کنترلر با ین ثبات کار می کند .بنابرین ابتدا بید داده در ین ثبات قرار گیرد سپس عملیات انجام شود.

ثبات B:

جزو ثبات های عمومی است بری انجام عملیات ضرب و تقسیم به همراه ثبات A به کار می رود.

ثبات DPTR ::

ین ثبات یک ثبات 16 بیتی است که از دو ثبات 8 بیتی با نامهای DPL و DPH ساخته شده است و به عنوان اشاره گر داده به کار می رود در هنگام استفاده از حافظه داده خارجی کاربرد ین ثبات را خواهیم دید.

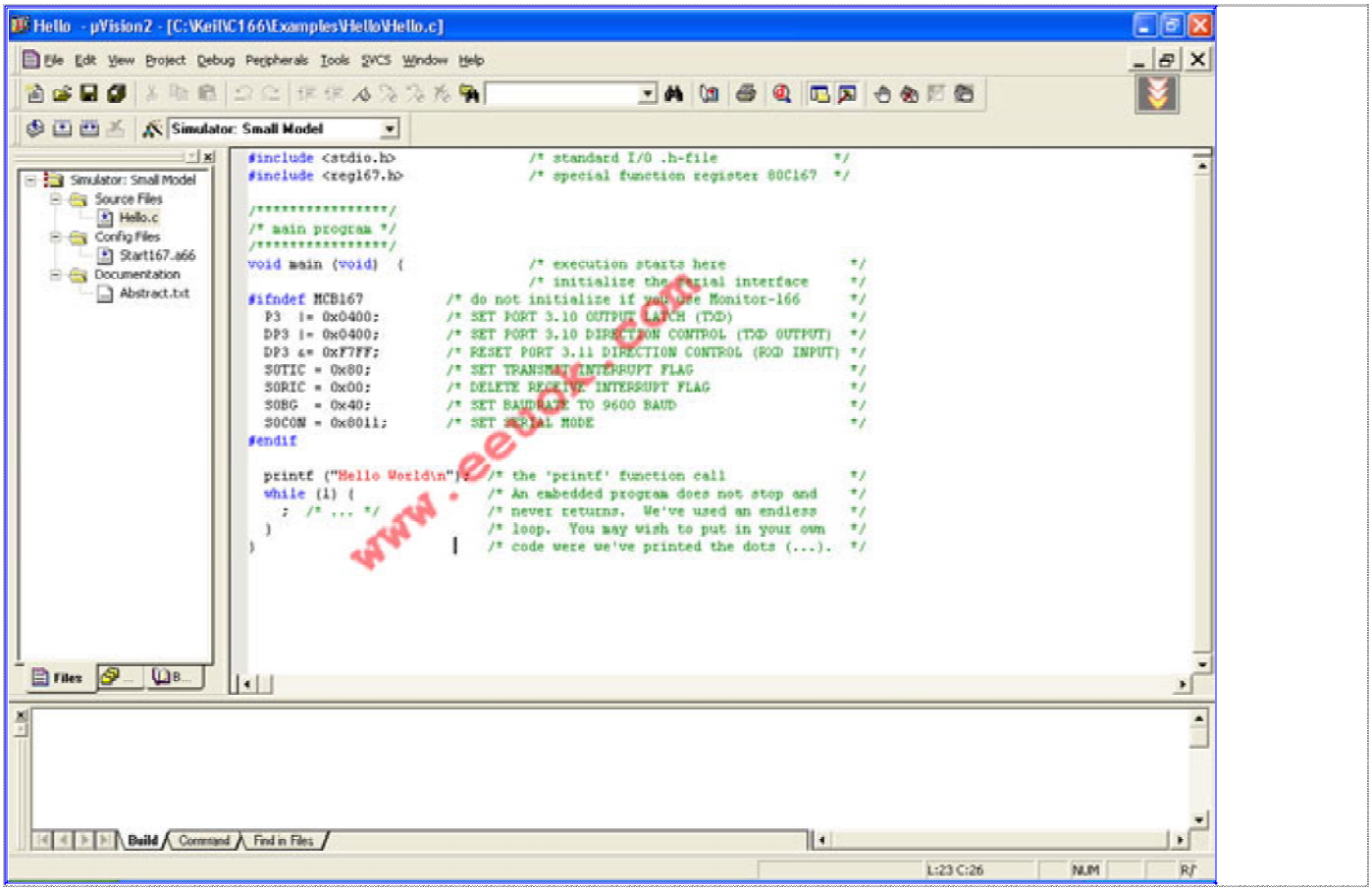
بقیه ثبات ها مانند ثبات هی تاییمر و ثبات هی وقفه و سریال و... در مقاله هی مربوط به مبحث خود جداگانه بررسی می شوند.

بری کار با میکرو کنترلر به چه چیزی احتیاج داریم؟

اولین چیزی که در یک پروژه میکرو کنترلری برجسته تر به نظر می رسد کد نویسی است .یک برنامه نویس میکرو کنترلر

باید با زبان اسمبلی آشنا باشد . البته کلمه " اسمبلی " یک کلمه عمومی است و به کلیه زبانهای سطح پیین (low level) گفته می شود در حالی که هر میکرو کنترلری یا میکرو پروسسوری اسمبلی مختص به خود را دارد یعنی اسمبلی 8051 با اسمبلی Z80 متفاوت است در میکرو پروسسور ها هم تفاوت زیادی بین اسمبلی نسل هی میکرو پروسسور ها وجود دارد اسمبلی Z80 با اسمبلی پنتیوم تفاوت دارد به زبان دیگر چون معماری میکرو ها با هم متفاوت است در نتیجه اسمبلی آنها هم با هم متفاوت است .البته کسی که با اسمبلی یک میکرو آشنا باشد برنامه نویسی در یک میکروی دیگر بریش زیاد دشوار نیست . زبان های دیگری هم بری برنامه نویسی میکرو کنترلر استفاده می شوند مانند C و pascal البته ین زبان ها هر کدام در یک کامپایلر به خصوص کار می کنند بری مثال کامپایلر keil هر دو زبان اسمبلی و C را پشتیبانی می کند .به هر حال آشنایی با زبان برنامه نویسی میکرو کنترلر موضوع مقاله بعدی ما خواهد بود .

شکل زیر نرم افزار keil را نشان می دهد



با فرض اینکه ما با زبان اسمبلی 8051 آشنا هستیم برنامه خود را در ادیتور مانند keil یا Notepad می نویسیم و سپس به وسیله یک کامپیلر مانند ASM51 که کامپایلر پنتل برای 8051 است برنامه را به فایل HEX تبدیل می کنیم .(keil خود داری کامپیلر است و فایل Hex را تولید می کند) فیل Hex درست مانند فایلهی EXE در کامپیوتر است یعنی یک فیل اجرایی برای میکرو کنترلر است . بعد از کامپایل کردن باید برنامه توسط یک پرو گرامر
(وسیله ی که IC8051 را برنامه ریزی می کند) بر روی IC قرار گیرد .پرو گرامر ها به صورت آماده در فروشگاه های قطعات الکترونیک موجود هستند . شما هم می توانید پرو گرامر خود را بسازید .در قسمت دانلود فایل سایت می توانید schematic پرو گرامر) 8052 , 8051 programmer) را download کنید و خودتان programmer را بسازید پیشنهاد می کنم حتما ین کار را انجام دهید چون هم لذت کار عملی را خواهید دید و هم در هزینه صرفه جویی زیادی خواهید کرد (12 تا 15 هزار تومان!).

مبحث مهمی که در برنامه نویسی به زبان اسمبلی وجود دارد شیوه های آدرس دهی است .منظور از آدرس دهی روش دستیابی میکرو پروسسور به اطلاعات است آدرس دهی ، مبدا و مقصد داده را تعیین می کند و اینکه در جریان انتقال داده از مقصد به مبدا چه عملیاتی بیستی انجام شود .قبل از پرداختن به تشریح دستور العمل های اسمبلی 8051 ، انواع شیوه های آدرس دهای را بیان می کنیم.

در 8051 هشت نوع آدرس دهی وجود دارد :

- آدرس دهی ثبات(Register Addressing)
- آدرس دهی مستقیم (Direct Addressing)
- آدرس دهی غیر مستقیم (Indirect Addressing)
- آدرس دهی فوری (Immediate Addressing)
- آدرس دهی نسبی (Relative Addressing)
- آدرس دهی مطلق(Absolute Addressing)
- آدرس دهی طولانی (Long addressing)
- آدرس دهی دهی اندیس دار (Index Addressing)

تک تک این شیوه های آدرس دهای را با هم بررسی خواهیم کرد.

آدرس دهی ثبات:

همانگونه که می دانید در 8051 هشت ثبات با نام بانک ثبات وجود دارند که این ثبات ها به صورت تک تک با نام های R0-R7 شناخته می شوند تعدادی از دستور العمل های 8051 بر روی این ثبات ها کار می کنند ین دستور العمل ها را دستور های با آدرس دهی ثبات گویند.البته چند دستور دیگر هم وجود دارند که فقط بر روی یک ثبات خاص عمل میکنند مانند ثبات های A,B و ثبات DPTR(همان اشاره گر داده) و بیت C (داده نقلی) ثبات PSW(کلمه وضعیت برنامه) این دستور العمل ها هم آدرس دهی ثبات را به کار گرفته اند.

مثالی از این نوع دستورات به صورت زیر است (اگر مفهوم دستورات را متوجه نمی شوید اشکالی ندارد در مقاله های بعدی توضیح کافی بر روی تک تک دستورات داده خواهد شد):

```
ADD A,R0
INC DPTR
MUL AB
DIV AB
```

آدرس دهی مستقیم :

گفتیم که هر بیت (8 بیت) از حافظه میکرو به وسیله عددی منحصر به فرد که شماره ردیف آن بیت است مشخص می شود ین عدد منحصر به فرد **آدرس** نام دارد . از این جهت می گوییم منحصر به فرد زیرا هیچ دو بیت از حافظه ی میکرو داری یک آدرس نیستند!.

با استفاده از این آدرس می توان به کلیه مکانهای حافظه دسترسی داشت .آدرس دهی مستقیم از ین آدرس استفاده می کند .

علاوه بر آدرس ، بیشتر مکان هی حافظه داری نام نیز هستند.برای مثال پورت یک، هم داری آدرس 90H است و هم نام

P1 را دارد . استفاده از هردو (نام یا آدرس) در برنامه نویسی مجاز است .
در برنامه نویسی هنگامی که عددی را بدون هیچ پیشوندی می نویسند نشان دهنده استفاده از آدرس دهی مستقیم است.

آدرس دهی غیر مستقیم :

آدرس دهی غیر مستقیم نیز از آدرس بیات مورد نظر استفاده می کند .تفاوت آن با آدرس دهی مستقیم ین است که در آدرس دهی غیر مستقیم آدرس امکان حافظه ابتدا در یکی از ثبات های بانک ثبات قرار می گیرد وسپس ثبات مذکور به همراه پیشوند @(At sign) در دستور العمل استفاده می شود.

کاربرد این نوع آدرس دهی برای مواقعی است که آدرس یک متغیر بیتی در زمان اجری برنامه ، نه در زمان نوشتن کد ، محاسبه شود و نیز هنگام استفاده از مکانهای متوالی حافظه نیز باید از آدرس دهی غیر مستقیم استفاده کنیم . در موارد بالا نمی توانیم از آدرس دهی مستقیم و ثبات استفاده کنیم زیرا این دو آدرس دهی در هنگام کامپایل برنامه به آدرس امکان حافظه نیاز دارند در حالی که ما آدرس مکان حافظه را نمی دانیم .
برای فهم بیشتر مطلب دو دستور زیر را در نظر بگیرید:

```
MOV R0,#90h
MOV @R0,A
```

دستور اول عدد 90 در مبنی شانزده را در ثبات R0 قرار می دهد – به علامت #(number sign) قبل از عدد توجه کنید مبحث بعدی آدرس دهی ماست- عدد 90h آدرس پورت یک می باشد . دستور دوم به میکرو دستور می دهد تا **محتوی ثبات A را در آدرسی که در R0 است(یا مکانی که R0 به آن اشاره می کند) قرار بده .**
جمله بالا در مفاهیم برنامه نویسی به نام اشاره گر معروف است . اشاره گر متغیری است که آدرس مکانی از حافظه را در خود نگه می دارد .
در این دستور R0@ معادل همان P1 است .
آدرس دهی فوری یا بلافصل :

در صورتی که بخواهیم یک عدد ثابت – نه یک آدرس – را در مکانی از حافظه قرار دهیم از آدرس دهی فوری استفاده می کنیم .دستور اول در مثال قبلی نوعی از آدرس دهی فوری بود . مشخصه ی آدرس دهی فوری استفاده از پیشوند # قبل از عدد است .

در صورتی که در یک دستور از عددی استفاده کنیم اگر عدد بدون پیشوند باشد نوع آدرس دهی مستقیم است و در صورتی که عدد پیشوند # داشته باشد آدرس دهی فوری خواهد بود .به این دو دستور توجه کنید تا ین جمله را بهتر متوجه شوید:

```
MOV A,#90h
MOV A,90h
```

دستور اول عدد 90 در مبنی شانزده را در ثبات A قرار می دهد . بعد از اجرای دستور مطمئن هستیم که محتوی ثبات A ، 90h است . دستور دوم مقداری را که در آدرس 90h حافظه است در ثبات A قرار می دهد یادآوری می کنم که 90h آدرس پورت یک است .بنابرین هر عددی که بر روی پورت یک قرار داشته باشد در درون ثبات A قرار خواهد گرفت .و ما از مقدار این عدد هیچ اطلاعی نداریم .

توجه:

در برنامه نویسی میکرو می توان از مبنا هی عددی مختلف ستفاده کرد . این کار با استفاده از نوشتن عدد و نوشتن مشخصه ی عدد درست بعد از آن ، به عنوان پسوند (در سمت راست عدد) صورت می گیرد .

مشخصه ی مبنا های عددی به صورت زیر است:

B بری مبنی دودویی یا binary

D بری مبنی دهدهی یاdecimal

H بری مبنی شانزدهی یا hexadecimal

در صورتی که قصد استفاده از مبانی ده را داشته باشیم می توانیم از پسوند D صرفنظر کنیم مبنی ده پیش فرض کامپایلر است .

اگر از مبنی شانزدهی در برنامه استفاده کردیم اگر اولین رقم سمت چپ عدد (پر ارزش ترین رقم) رقمی بین A-F در مبنی شانزده بود باید قبل از عدد یک صفر 0 قرار دهیم مثلا عدد #FFh را باید به صورت #0FFh بنویسیم.

آدرس دهی نسبی :

ین نوع آدرس دهی در چند دستورالعمل پرشی به کار رفته است.(مانند sjmp) با استفاده از این نوع آدرس دهی می توان به 127 بیت بالاتر و 127 بیت پینتر از امکان فعلی برنامه پرش کرد . هنگامی که کامپیر به ین نوع دستورات پرشی می رسد آدرس مبدا را از آدرس مقصد تفریق می کند ونتیجه را به شمارنده برنامه اضافه می کند (pc) دستور زیر را در نظر بگیرید:

```
90 sjmp 100
```

اکنون در مکان 90 حافظه هستیم برنامه می خواهد به مکان 100 پرش کوتاه انجام دهد .آدرس مبدا از مقصد کم می شود (90-100) و به 10+ ، pc واحد اضافه می گردد.

مزیت ین نوع آدرس دهی ین است که کد مستقل از امکان یجاد می کند.

آدرس دهی مطلق :

تنها دو دستور العمل Acall,Ajmp از این نوع آدرس دهی استفاده می کنند آدرس مبدا و مقصد هر دو بید در یک صفحه دو کیلو بیتی از حافظه باشند . این دستورات کد وابسته به مکان تولید می کنند .

آدرس دهی طولانی :

دو دستور Lcall, Ljmp از این نوع آدرس دهی استفاده می کنند.مبدا و مقصد می توانند در یک صفحه 64 کیلو بیتی باشند (کل فضی حافظه پوشش داده شده است).

ین نوع آدرس دهی هم وابسته به مکان است.

توجه :

منظور از کد وابسته به مکان بدین صورت است:

فرض کنید برنامه ی نوشته یم که از فضای 100 حافظه شروع می شود .برنامه زیر

```
100 sjmp there
101….
….
….
105 there :Ajmp 110
…..
….
110 here: ------
```

در این برنامه از آدرس 100 به 105 که داری برچسب there است پرش می کند و از there هم به here پرش میکند .حال اگر همین برنامه را روی میکروی دیگری که برنامه از آدرس 50 شروع شده باشد اجرا کنیم منطق برنامه به صورت زیر در می اید:

```
50 sjmp there
101….
….
….
55 there :Ajmp 110
…..
```

```
….
60 here: ------
--- ----- ----
110 :---- ----
```

دستور sjmp که از آدرس دهی نسبی استفاده می کند به مکان درست there پرش می کند اما دستور ajmp که از آدرس دهی مطلق استفاده می کند به جی پرش به here به مکان 10 حافظه پرش می کند که اصلا مورد نظر برنامه نویس نبوده است !!!

بری رفع ین مشکل از دستور ORG در ابتدای برنامه استفاده می کنیم تا به کامپایلر بگوییم که برنامه ما باید از چه مکان حافظه ی بیستی شروع شود .

```
ORG 100
100 sjmp there
101….
….
….
105 there :Ajmp 110
…..
….
110 here: ------
```

آدرس دهی اندیس دار:

ین آدرس دهی از یک ثبات پایه (pc یا DPTR) و یک فاصله نسبی (A) برای دستور العمل هی JMPو MOVC استفاده می کند این نوع آدرس دهی در نوشتن جداول جستجو(LOOKUP Table) استفاده می شود در آینده نمونه ای از ین نوع برنامه ها را خواهیم نوشت .

```
MOVC A,@A+DPTR
```

. موفق باشید

برای شروع برنامه نویسی با میکرو لازم است با دستورات پر کاربرد آن آشنا شویم . البته یک مراجع سریع دستورات زبان برنامه نویسی جزء ابزار های ضروری برای یک برنامه نویس است.اصولا برای یادگیری هر زبان برنامه نویسی علاوه بر آشنایی با دستورات زبان ، تمرین و نوشتن برنامه های گوناگون بسیار لازم است تا برنامه نویس ساختار صحیح دستورات بریش ملکه ذهن شود و از وقت بسیار ارزشمندی که باید به جای ینکه صرف نوشتن کد های جدید تر شود صرف عمل طاقت فرصا و وقت گیر Debugging می شود کاست .پس بعد از آشنایی با هر دستور العملی انواع مودهای آدرس دهی آن را مورد بررسی قرار دهید و مثال های کوتاه اما پر ارزش با محتوا را تمرین کنید.
دستورالعمل هی 8051 را بر اساس نوع عملکرد می توان به پنج گروه تقسیم کرد.

- دستورالعمل های محاسباتی
- دستورالعمل های منطقی
- دستور العمل های انتقال داده
- دستور العمل های بولی
- دستور العمل های انشعاب برنامه

نخست دستور العمل های محاسباتی :

این مجموعه دستورالعمل ها شامل دستورالعمل های جمع و تفریق و کاهش وافزیش و ضرب و تقسیم 8 بیتی می باشند.

دستور العمل ADD برای جمع کردن یک مقدار 8 بیتی با ثبات A مورد استفاده قرار می گیرد . عملوند اول ین دستورالعمل ثبات A (انباره) می باشد . این دستورالعمل در 4 مورد آدرس دهی مورد استفاده قرار می گیرد.
مودهای آدرس دهی را در مقاله قبلی بررسی کردیم حال خودتان بید بتوانید ین مود های آدرس دهی را از همدیگر با استفاده از مشخصه های هر مود آدرس دهی از همدیگر تفکیک کنید.

```
ADD A,Rn
ADD A,direct
ADD A,@Rn
ADD A,#data
```

در دستورالعمل های بالا عملوند هی دوم (Rn,@Rn,#data,direct) سمبل هیی هستند که نشان دهنده نوع آدرس دهی می باشند .منظور از Rn هر کدام از ثبات هی بانک ثبات است ،منظور از data هر عدد ثابت 8 بیتی می تواند باشدمنظور از diret عدد یا نامی است که نشان دهنده آدرس مکانی از حافظه باشد.
مثال:

```
ADD A,R0
ADD A,090h
ADD A,@R0
ADD A,#10
```

دستور اول محتویات R0 را با ثبات A جمع و دوباره در A قرار می دهد دستور دوم محتویات آدرس90h حافظه (پورت یک) را با A جمع می کند و دوباره در A قرار می دهد .دستور سوم محتویات جیی که R0 به آن اشاره می کند را با A جمع کرده و در A قرار می دهد .و دستور چهارم آنچه را که در A قرار دارد با عدد 10 دهدهی جمع می کندو

نتیجه باز هم در A قرار می گیرد.
حتما متوجه شده اید که نوع آدرس دهی در دستور اول ثبات ، دستور دوم مستقیم ، دستور سوم غیر مستقیم ، و در دستور چهارم فوری است.

دستور دیگری برای جمع وجود دارد با سمبل ADC که مخفف ADD with Carry می باشد و به معنی جمع با رقم نقلی است عملکرد آن بدین صورت است که ابتدا مقدار بیت C (نقلی) با ثبات A جمع شده و سپس عملوند دوم هم به ثبات A اضافه می شود . این دستور هم مانند دستور ADD در چهار مود آدرس دهی بالا مورد استفاده قرار می گیرد.

دستورالعمل تقسیم 8 بیتی :

DIV A

ین دستور العمل مقدار موجود در ثبات A را بر مقدار موجود در ثبات B تقسیم می کند . خارج قسمت در ثبات A و باقیمانده عمل تقسیم در ثبات B ذخیره خواهد شد.
فرض کنید که ثبات A حاوی مقدار 11 است و ثبات B حاوی مقدار 4 باشد پس از اجری عمل تقسیم مقدار 2 در ثبات A و مقدار 3 در ثبات B ذخیره خواهد شد.

دستور العمل افزیش :

یک دستور العمل تک عملوندی است ین دستور عملوند خود را یک واحد افزیش می دهد .نکته ی که در موردین دستور العمل بید به آن توجه کرد اینست که هیچ پرچمی بعد از اجری دستور تغییر نخواهد کرد بری مثال اگر ثبات A حاوی

مقدار #offh باشد بعد از اجری ین دستور مقدار A, 00 خواهد شد اما پرچم ov(over flow) تغییر نخواهد کرد.
ین دستور العمل در سه مورد آدرس دهی ثبات ، مستقیم و غیر مستقیم عمل می کند.

```
INC A
INC direct
INC Rn
INC @Ri
INC DPTR
```

دستورالعمل کاهش:

ین دستور العمل یک عملوند دارد .پس از اجری دستور از محتوی عملوند یک واحد کم می شود.

```
DEC A
DEC Rn
DEC direct
DEC @Rn
```

توجه :

دستور DEC DPTR وجود ندارد و باید برای کاهش DPTR دو ثبات DPLو DPH را جداگانه کاهش دهیم .

اگر ثبات A حاوی مقدار 10 باشد پس از اجرای DEC A ثبات A حاوی مقدار 9 خواهد بود .

دستور العمل ضرب :

```
MUL AB
```

ین دستور العمل محتوی ثبات A را در ثبات B ضرب می کند نتیجه ضرب بدین گونه ذخیره می شود .بیت پایین حاصلضرب در ثبات A و بیت بالی حاصلضرب در ثبات B ذخیره می شود.

دستور العمل تفریق:

دستور SUBB برای تفریق یک مقدار 8 بیتی از ثبات A مورد استفاده قرار می گیرد.این دستور مخفف(Sub with Barrow) است و عملوند دوم را به اضافه ی Carry از ثبات A کم می کند اگر در جریان تفریق از بیت هشتم نیاز به رقم قرضی باشد Carry یک می شود و در غیر اینصورت صفر می شود.در زیر مورد های آدرس دهی عمل تفریق آمده اند .

```
SUBB A,Rn
SUBB A, direct
SUBB A, @Rn
SUBB A,#data
```

مجموعه دستورالعمل هی منطقی :

دستور العمل های منطقی بر روی تک تک بیت ها اعمال منطقی ANDو OR وXOR را انجام می دهند دستورات چرخش و شیفت بیتی هم در زمره این دستورات هستند.

در AND کردن ، نتیجه دو بیت زمانی یک است که هر دو بیت یک باشند.در OR کردن دو بیت نتیجه دو بیت زمانی صفر است که هر دو بیت صفر باشند در XOR کردن (یی انحصاری) نتیجه هنگامی صفر است که هر دو بیت یا صفر باشند یا یک.

XOR برای مکمل کردن بیت ها هم به کار می رود .نتیجه XOR یک بیت با بیت 1 مکمل آن خواهد بود و نتیجه XOR یک بیت با بیت 0 خود آن بیت را نتیجه خواهد داد. این دستورات برای مکمل کردن بیت های مشخصی از یک بیت حافظه ، که بیت آدرس پذیر نیستند مفیداست.

دستور اسمبلی که بری And کردن مورداستفاده قرار می گیرد ANL (And Logic) است.

```
ANL A,direct
ANL A,@Ri
ANL A,Rn
ANL A,direct
ANL A,#data
ANL direct,A
ANL direct,#data
ANL c, bit
```

دستور آخر یک دستور بیتی است یعنی بر روی یک بیت به تنهایی عمل می کند بقیه دستورات دستورات بیتی هستند.

مثال:

فرض کنید که پورت یک حاوی مقدار #01001111b است و همچنین در ثبات A هم مقدار بینری #00001010b را داشته باشیم

پس از اجری دستور العمل ANL A,P1 نتیجه به صورت زیر خواهد بود :

```
01001111
AND
00001010
--------------
00001010
```

و ین مقدار در ثبات A قرار خواهد گرفت .

توجه :

ثبات A یک ثبات بیت آدرس پذیر است یعنی می توان به تک تک بیت ها به صورت مستقیم دسترسی داشت .نام دیگری که در برنامه های اسمبلی با آن بر خورد خواهید داشت ACC است که مخفف(Accomulator یا انباره) است با استفاده از ACC به همراه یک نقطه(DOT) و شماره بیت ثبات A می توان مستقیما بر روی آن بیت اعمال منطقی را انجام داد.بری مثال ACC.7 نشانگر آخرین بیت ثبات A است .

ANL C,ACC.7

بیت شماره هفت ثباتA را با AND، Carry کرده و نتیجه در Carry قرار میگیرد .

دستورات OR کردن و XOR کردن هم ساختاری مانند دستور ANL دارند فقط بری OR از ORL و بری XOR از XRL استفاده می کنیم .

دستور دیگری هم با نام CPL وجود دارد که کارش مکمل کردن عملوند خود است .

دستورات چرخشی :

چهار دستور RR A, RL A,RLC A ,RRC A بری عملیات چرخشی مورد استفاده قرار می گیرند.

RL A (Rotate Left A) بیت های ثبات A را یک واحد به طرف چپ چرخش می دهد بیت شماره هفت از

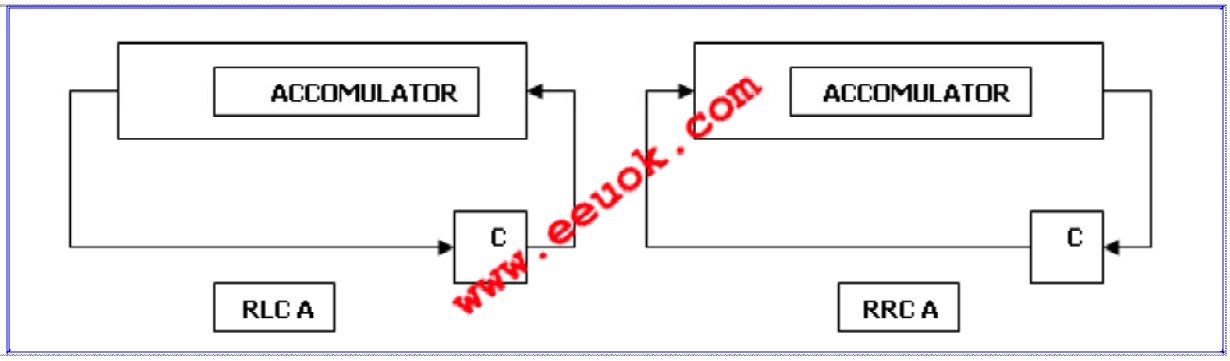
ثبات خارج می و دوباره وارد بیت شماره صفر می شود.اگر ثبات A حاوی #10000001b باشد:

```
10000001 ----------------> 00000011
```

RR A (Rotate right A) ثبات را یک واحد به راست چرخش می دهد در دستورات چرخش بیتی که خارج می شود از طرف دیگر وارد ثبات می شود.

```
10000001 ----------------> 11000000
```

RRC A و RLC A هم مشابه دستورات بالا هستند با این تفاوت که بیتی که از یک طرف خارج می شود وارد بیتCarry می شود و محتوی قبلی Carry از طرف دیگر وارد ثبات می گردد.



در ادامه مبحث بررسی دستور العمل های 8051 می رسیم به مبحث دستور العمل هی انتقال داده .
دستور العمل های انتقال داده به دو دسته دستور های RAM خارجی و دستور های RAM داخلی تقسیم می شوند .برای انتقال داده RAM داخلی از دستور

```
Mov مقصد,مبدا
```

استفاده می شود .ین دستور داده را از مبدا به مقصد انتقال می دهد بدون اینکه نیاز به استفاده از انباره باشد دستیابی به RAM داخلی توسط آدرس دهی غیر مستقیم میسر است.

دستور دیگری که در انتقال داده به کار می رود دستور xch است که باعث می شود محتویات انباره و مکانی از حافظه عوض شود . دستور مشابه ین دستور دستور xchd است که باعث می شود فقط بیت های پایین دو عملوند با هم عوض شوند .

مثال:

فرض کنید ثبات A حاوی مقدار #04fh است و مقدار موجود در مکان 100 حافظه #03EH است :

```
Mov R1,100
Xch A,@R1
XChd A,@R1
```

دستور اول 100 را در R1 ذخیره می کند . میخواهیم که از آدرس دهی غیر مستقیم استفاده کنیم استفاده از آدرس دهی مستقیم مستلزم به کار گیری ثباتی از بانک ثبات ، به عنوان اشاره گر است .

دستور دوم محتوی A و مکان 100 حافظه رابا هم عوض می کند یعنی مقدار 3E در ثبات A و مقدار 4F در مکان 100 حافظه قرار می گیرد . دستور سوم باعث می شود که بیت هیی پایین دو مکان ، ثبات A و مکان 100 حافظه با هم جابه جا شوند پس از اجری دستور مقدار 4E در مکان 100 و 3F در ثبات A ذخیره می شود.

انتقال داده در RAM خارجی:

در هنگام استفاده از RAM خارجی ، برای انتقال داده از آدرس دهی غیر مستقیم استفاده می کنیم . در اجری اینگونه دستورات ثبات A به عنوان عملوند اول (بری ورود داده از RAM) یا عملوند دوم (فرستادن داده به RAM) مورداستفاده قرار می گیرد .
دستور انتقال داده برای RAM خارجی movx است . که در دو سیکل ماشین اجرا می شود.
گاهی اوقات لازم است که در هنگام نوشتن برنامه یک سری اطلاعات را در ROM میکرو قرار دهیم ین اطلاعات چون در داخل ROM (Read Only Memory) هتند فقط می توانند خوانده شوند . ما از دستور mov بری دسترسی به RAM داخلی میکرو استفاده می کردیم و از دستور movx بری دسترسی به RAM خارجی . برای دسترسی به این اطلاعات که در داخل ROM قرار گرفته اند بیدار از دستور movc استفاده کنیم (انتقال ثابت) . برنامه یا کدی که ما می نویسیم در داخل ROM میکرو ذخیره می شود و جزء حافظه ی کد است . حافظه ی کد را فقط در هنگام نوشتن برنامه و قبل از پرو گرام کردن

آی سی ، می توان تغییر داد در هنگام اجرای برنامه فقط می توان اطلاعات این قسمت را خواند .
نمونه ای از استفاده از حافظه کد برای ذخیره داده هنگام استفاده از تکنیک جدول جستجو (lookup Table) است.

دستور العمل های بیتی:

حافظه مجموعه ای از بیت ها است.و بیت از 8 بیت ساخته شده است میکرو کنترلر توانائی به کارگیری دستورات بیتی رابری تغییر بیتها یی که بیت آدرس پذیر هستند را دارد.
ین دستورات شامل دستورات انتقال بیت ودستورات منطقی و ست کردن بیت و پاک کردن بیت و ... هستند.
برای ینکه بیت مشخصی را ست کنیم (مقدار آن را هر چه باشد به یک تغییر دهیم) از دستور SETB استفاده می کنیم.

SETB bit

برای پاک کردن یک بیت(صفر را در آن قرار دهیم) از دستور

```
CLR bit
```

استفاده می کنیم .برای مکمل کردن یک بیت مشخص (اگر مقدار آن یک باشد به صفر تبدیل می شود و برعکس) از دستور

```
CPL bit
```

استفاده می کنیم .دستور انتقال داده بیتی همان دستور mov است . دستور mov را قبلا برای انتقال یک بیت از حافظه نیز به کار بردیم حال چگونه تشخیص دهیم که یک دستور mov ، یک بیت از حافظه را انتقال می دهد یا یک بیت ؟
جواب ین است: در صورتی که عملوند اول دستور یک بیت باشد یا یک ثبات، دستور mov بیتی است و هنگامی که عملوند دستور یک بیت باشد دستور، بیتی است

```
Mov c, p1.0
Mov A,#10
```

دستور های بیتی شامل دستورات منطقی ANLو ORL نیز هستند .که همان AND و OR منطقی می باشند .با این تفاوت که ین دستورات بر روی دو تک بیت عمل می کنند :

```
ANL C,P1.0
```

دستورات بیتی منطقی دستور XRL (XOR) را ندارند.

دستورات انشعاب برنامه :

دستورات برنامه خط به خط بعد از هم اجرا می شوند این اجرای خط به خط ظاهرا برای بعضی برنامه ها مفیداست اما اگر برنامه طوری باشد که مجبور باشیم در یک شریط یک سری دستورات ، و در شریط دیگر یک سری دیگر دستورات دیگر اجرا شوند .مسلما ساختار ترتیبی اجرای برنامه نمی تواند جوابگو باشد به همین دلیل طراحان میکرو ، سری دستورات انشعاب و و پرشی را برای آن در نظر گرفته اند تا نوشتن بر نامه هی پیچیده مقدور باشد .

دستورات پرشی به دو دسته تقسیم می شوند :

1 – پرشه ی غیر شرطی 2- پرشه ی شرطی

انشعاب غیر شرطی :

در این گونه انشعاب ها برنامه بدون چک کردن شرطی ، به قسمت دیگری از برنامه می پرد. این گونه پرشها با استفاده از مشتقات JMP و CALL استفاده می شود.

نوع دستور JMP (jump) وجود دارد . sjmp (short jump) برای پرش کوتاه (پرش نسبی با طول 127 بیت به جلو یا عقب) و ljmp (long Jump) برای پرش طولانی و Ajmp بری پرش با آدرس دهی مطلق . پرش طولانی می تواند به هر کجای برنامه باشد و محدودیت فاصله ندارد (در یک صفحه 64 کیلو بیتی)اما برای پرش مطلق بید هم آدرس مبدا و هم آدرس مقصد در یک صفحه دو کیلو بیتی باشند.

بسته به اختلاف آدرس مبدا و مقصد و همچنین صفحه دو کیلو بیتی که آدرس مبدا در آن قرار دارد از jmp مناسب استفاده می کنیم .فرضا ممکن است که اختلاف آدرس مبدا و قصد کمتر از یک کیلو بیت باشد اما آدرس مبدا در یک صفحه دو کیلو بیتی قرار داشته باشد و آدرس مقصد در یک صفحه دو کیلو بیتی دیگر قرار داشته باشد در ین حالت اگر اختلاف دو آدرس کمتر از 127 باشد از sjmp و در غیر ینصورت از ljmp استفاده می کنیم .

صفحات دو کیلو بیتی مورد بحث از آدرس های زیر شروع می شوند (0000و800hو1000hو1800hوf800h)

نوع دیگر پرشه ی غیر شرطی call است دستور Call بری فراخوانی زیر برنامه ها به کار می رود.زیر برنامه ها را در مقاله ی جدا گانه به زودی برسی خواهیم کرد.

پرشهی شرطی:

پرش هی شرطی ابزار انعطاف پذیری برای انجام عملیات خاصی در شری خاص است .در این دستورات اگر شرط بر قرار باشد به مکانی که آدرس آن مشخص شده پرش می کند و در صورتی که شرط برقرار نباشد دستور بعدی اجرا می شود .

اولین دستورالعملی که بررسی می کنیم دستور JNB است (jump if not bit) پرش در صورتی که بیت مشخص شده یک نباشد .دستور JB هم با شرط مخالف وجود دارد

```
JB bit,address
```

Bit نام یا آدرس یک بیت است و address بر چسب یا آدرس مقصد پرش است از این دستور برای اجرای تاخیر تا یک شدن bit مشخص ، بسیار استفاده می شود(مثلا در عملیات تایمر)

دستور بعدی DJNZ است (decrease jump if not zero) (یک واحد کم کن و در صورتی که صفر نبود بپر.)می توان از این دستوربرای اجری حلقه به تعداد دفعات تکرار مشخص استفاده کرد (مشابه حلقه ی for در زبان های برنامه نویس سطح بالا)

شکل دستور بدین صورت است :

```
DJNZ 1 عملوند,address
```

عملوند1 می تواند یکی از ثبات هی بانک ثبات یا آدرس یک بیت از حافظه باشد.

دستور پرش شرطی پ کاربرد دیگر دستور CJNE(compare jump if not equal)مقایسه کن و بپر در صورتی که دو عملوند برابر نباشند.

```
CJNE 2 عملوند ,1 عملوند, address
```

عملوند 1 و عملوند 2 دو مقداری هستند که با هم مقایسه می شوند.و Address مکانی است که در صورت عدم تساوی دو مقدار برنامه باید از آنجا ادامه پیدا کند .این شرط برای تعیین کوچکتر یا بزرگتر بودن عملوند ها از هم ، به کار می رود .

```
IF 2 عملوند >1 عملوند then
C <----------1
Else
C<-----------0
```

اینها دستورات پر کاربرد زبان اسمبلی 8051 بودند اما تعداد کل دستورات خیلی بیشتر از این تعداد است به یاد داشته باشید که یک مرجع سریع دستورات اسمبلی 8051 جزء ملزومات اولیه یک برنامه نویس است زیرا قرار نیست یک برنامه نویس کلیه روشهای آدرس دهی همه ی دستورات را حفظ کند.بنابرین همین حالا مرجع دستورات اسمبلی 8051 را از قسمت دانلود فایل سایت دانلود کنید

تمرین و حل مسئله برای هر درسی به ماندگاری و فهم آن مطلب بیشتر ین کمک را می کند . بری حل هر مسئله ی راه هی گوناگون و الگوریتم های متفاوتی ممکن است و جود داشته باشد .دیدن راه حل و الگوریتم دیگر برنامه نویسان و تحلیل برنامه آنها یکی از را ه های پرورش قوه " تشخیص راه حل است". در این مقاله یک برنامه مقدماتی و در عین

حال جالب را با هم تحلیل می کنیم.

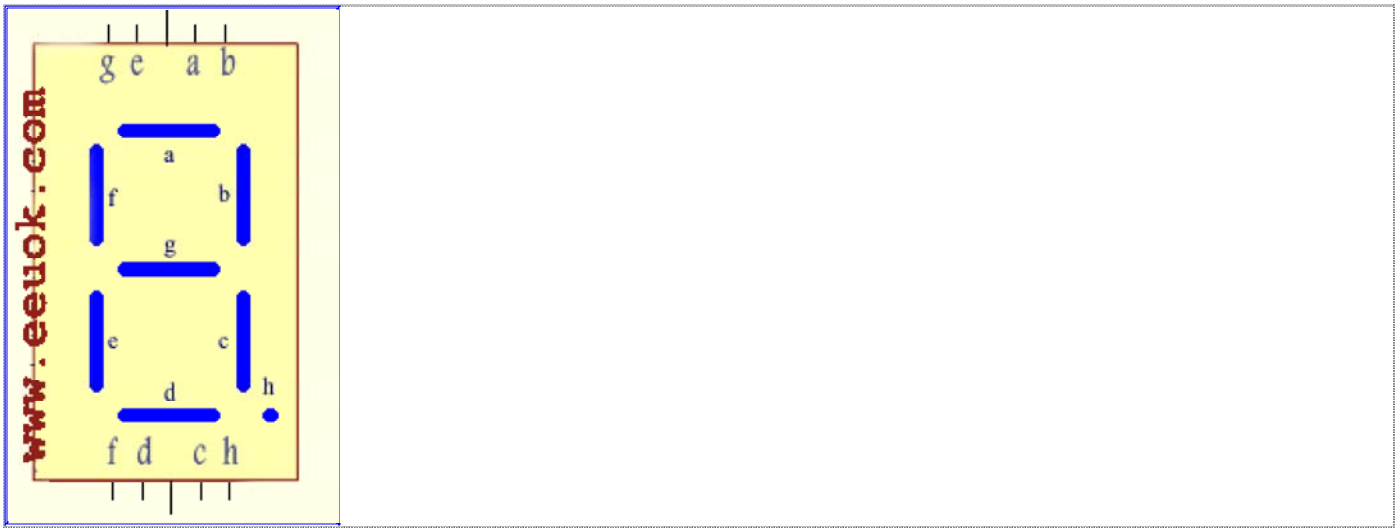
برنامه ای بنویسید که از 4 بیت پیین پورت 3 یک عدد در مبنی دو را دریافت کندو آن را بر روی 7 – segment یا هفت پارچه ی متصل به پورت یک نشان دهد.برای مثال در صورتی که ورودی 0101 باشد خروجی نشان داده شده بر روی 7 –segment عدد 5 باشد.

طراحی و حل چنین مداری مستلزم آشنایی با سخت افزار نیز می باشد .ابتدا سخت افزار لازم را بیان می کنیم. 7-segment از هفت عدد LED خطی و یک عدد LED نقطه ی با نام (dp (decimal Point ساخته شده است .همانطور که می دانید LED یک دیود است که در حالت هدایت جریان الکتریکی از خود نور ساطع می کند .شکل یک LED در مدار به صورت زیر است .

****

segmen ها به دو صورت آندمشترک و کاتد مشترک در بازار موجودند .در صورتی که 7-segment کاتد مشترک د یعنی قسمت منفی LED به همدیگر متصل است و قسمت مثبت یا آند آنها به صورت جدا از هم است . بری ن کردن یک LED معین از seg-7 کاتد مشترک ، باید کاتد به Ground و آند به VCC متصل شود. seg-7 آند رک ، آند آن مشترک بوده و قسمت کاتد LED ها جدا از هم است برای کار کردن با این نوعseg-7 قسمت مشترک VCC متصل می شود و به پایه ی متناظر با LED که می خواهیم روشن کنیم ولتاژ صفر منطقی TTL اعمال می کنیم. segmen داری 10 پایه است که 5 پایه در بالا و 5 پایه در پایین قرار دارد .پایه های وسطی بالا و پایین پایه های

رک هستند و این دو پایه به همدیگر متصلند لذا فقط یکی ازاین پایه ها را به VCC یا GND متصل می کنیم (بسته وع seg-7 که آند مشترک باشد یا کاتد مشترک) و بقیه پایه ها متناظر با LED های داخل PACKAGE می د.هر حرف پایه متناظر با LED با حرف همنام است.



فرض کنید که می خواهیم تا عدد 2 را روی seg-7 نشان دهیم .برای این کار لازم است تا پایه های d,e,g,b,a را روشن کنیم و بقیه پایه ها خاموش باشندحال در صورتی که seg-7 آند مشترک باشد پایه ی مشترک به 5 ولت متصل می شود و پایه ی که می خواهیم روشن کنیم به آن صفر TTL می دهیم .

تا اینجا بید فرق بین seg-7 آند مشترک و کاتد مشترک را فهمیده باشید!

در این مثال ما از seg-7 آند مشترک استفاده می کنیم.

اما قسمت نرم افزاری مسئله :

نکته ی که در این مسئله دیده می شود این است هیچ فرمول و رابطه ای بین عدد باینری وارد شده و مقدار نما یشی آن وجود ندارد . مثلا عدد 0011 برابر 3 و عدد0001111 نمایش عدد 3 بر روی seg-7 است . ما در این مثال از lookup table استفاده می کنیم .

در lookup table برای هر عدد معادل نمیشی آن بر روی seg-7 را در جایی ذخیره می کنیم و سپس با استفاده از برنامه به آن دسترسی پیدا می کنیم و معادل نمایشی آن را را بر روی پورت قرار می دهیم.

جدول را بدین صورت می نویسیم:

g	f	e	d	c	b	a	عدد/پایه
1	0	0	0	0	0	0	0
1	1	1	1	0	0	1	1
0	1	0	0	1	0	0	2
0	1	1	0	0	0	0	3
0	0	1	1	0	0	1	4
0	0	1	0	0	1	0	5
0	0	0	0	0	1	0	6
1	1	1	0	0	0	0	7
0	0	0	0	0	0	0	8
0	0	1	0	0	0	0	9
0	0	0	1	0	0	0	A
0	0	0	0	0	1	1	B
0	1	0	0	1	1	1	C
0	1	0	0	0	0	1	D
0	0	0	0	1	1	0	E
0	0	0	1	1	1	0	F

صفر به معنی روشن بودن LED مربوطه و یک به معنی خاموش بودن LED است.

توجه :

پایه ی a را به کم ارزش ترین بیت پورت متصل کنید و پایه ی g را به پر ارزش ترین بیت پورت متصل کنید.

تحلیل برنامه بدین صورت است :

ورودی :4 بیت که نشان دهنده عدد در مبنی دو است که از پورت 3 وارد می شود.
خروجی :8 بیت پورت یک ، که باید با روشن کردن LED های مربوطه عدد وارد شده را نشان دهد.
پردازش : 4 بیت ورودی با کدام یک از اعداد در داخل ستون اول جدول برابر است مقدار ستون دوم آن ردیف را به پورت یک ارسال کنیم.

کد برنامه :

```
ORG 30h
MOV P1,#0FFh
Loop: mov A,P1
Anl A,#00001111b
Mov DPTR,#Table
Movc A,@A+DPTR
Mov P3,A
Sjmp loop
Table:
Db 1000000b, 1111001b ,0100100b,0110000b, 0011001b,0010010b, 0000010b, 1110000b,
0000000b, 0010000b, 0001000b, 0000011b, 0100111b, 0100001b, 0000110b, 0001110b



End
```

دستور اول محل شروع برنامه را در میکرو کنترلر تعیین می کند .در ین دستور به اسمبلر دستور داده یم تا دستورات بعدی را از آدرس 30h به بعد حافظه کد میکرو قرار دهد.
دستور دوم به اصطلاح پورت را ورودی می کند. در ین دستور کلیه بیت هی پورت یک را set کرده یم . بعد از ین

دستور می توانیم مقدار پورت را بخوانیم.ین کار بری محافظت مدار داخلی پورت است.در یک برنامه یکبار ورودی کردن پورت کافی است .

دستورMov A, P1 مقدار موجود بر روی پورت یک را در ثبات A قرار می دهد .چون در صورت مسئله گفته یم که ورودی ما چهار بیت پیین پورت هستند بنابرین بوسیله دستور ANL A,#00001111b چهار بیت پیین ثبات A که همان عدد خوانده شده از پورت یک است را می کند .بنابرین عدد موجود در ثبات A همان ورودی ما بر روی پورت یک خواهد بود.
دستور mov DPTR,#table آدرس برچسب table را در DPTR قرار می دهد .دو دستور بعدی مربوط به گرفتن داده از حافظه ی کد هستند و داده در A ذخیره می شود و سپس مقدار آن به پورت 3 فرستاده می شود .
دستور sjmp هم باعث می شود میکرو عدد دیگری را بخواند .ین دستور پرش غیر شرطی است و حلقه ی بینهیت را می سازد.

جدول را ینگونه تعریف کرده یم :

```
: Table

Db 1000000b, 1111001b ,0100100b,0110000b, 0011001b,0010010b, 0000010b, 1110000b,
0000000b, 0010000b, 0001000b, 0000011b, 0100111b, 0100001b, 0000110b, 0001110b
```

DB دستوری است که یک بیت حافظه را مقدار دهی می کن‌د.ین دستور بری تعریف داده در حافظه کد به کار می رود (define Byte).
بری دسترسی به داده ی که با استفاده از دستور DB تعریف کرده یم از آدرس آن بیت استفاده می کنیم . درین مثال داده اول که عدد نمیشگر صفر است در آدرس Table قرار دارد و عدد نمیشگر 1 در آدرس table +1 و ... به همین ترتیب عدد نمیشگر F در آدرس Table+15 قرار دارد .
فرض کنید عددی که ما وارد می کنیم 3 باشد ین عدد در ثبات A قرار می گیرد و در DPTR آدرس شروع Table قرار

دارد دستور movc A,@A+DPTR عدد نمیشگر 3 را به پورت 3 ارسال می کند .
بار دیگر یاد آوری می کنم که در جدولی که ماتعریف کرده یم LED ها به وسیله صفر TTL روشن می شوند زیرا که ما از segment-7 آند مشترک را به کار برده یم (ین قسمت اختیاری است اما استفاده از آند مشترک مزیی نیز دارد)

امیدوارم که ین مثال حل شده توانسته باشد مفهوم استفاده از LOOKUP TABLE (جدول جستجو) را تفهیم کند.

آموزش میکرو کنترلر قسمت 7(زیر برنامه ها)

هر گاه در برنامه ی لازم باشد که یک سری عملیات چندین بار اجرا شود ین سری عملیات رابه صورت جدا از برنامه اصلی می نویسیمو بری آن یک نام قرار می دهیم و بری اجری آن عملیاتها ، آن را با نام فرا می خوانیم . تکه برنامه ی که ما به صورت جدا را نوشته یم یک زیر برنامه است.

نوشتن زیر برنامه با نسبت داده یک بر چسب (label) به آن شروع می شود.برچسب اسمی است که به یک آدرس مشخص نسبت داده می شود .آدرس ، عددی است که شماره ردیف هر بیت در حافظه ROM میکرو را مشخص می کند .قبلا گفتیم که با استفاده از دستور ORG می توان ابتدی برنامه را در ROM میکرو مشخص کرد.شید بپرسید که ما آدرس شروع را با استفاده از دستورORG تعیین کرده یم حال با شمردن دستورات می توانیم آدرس هر دستور را بیابیم پس چه نیازی به استفاده از LABEL یا برچسب است ؟

سوال شما در صورتی درست است که همه ی دستورات در حافظه داری طول مشخصی باشند اما در حقیقت بعضی دستورات یک بیتی و بعضی دیگر دو بیتی و بعضی هم سه بیتی هستند بنابرین طول مشخصی ندارند و بر همین مبنا محاسبه آدرس هر دستور کار مشقت باری است بنا برین لزوم استفاده از برچسب امری ضروری است زیرا در بسیاری از دستورات برنامه نویسی به آدرس دستورات قبلی یا بعدی احتیاج داریم مثلا بری پرش به مکان مشخصی در برنامه به آدرس آن مکان احتیاج داریم .

بری برچسب زدن به یک دستور ، در ابتدی دستور یک کلمه یا نام قرار می دهیم و بعد از آن علامت ":"" را قرار می دهیم و آنگاه دستور را می نویسیم .
اسمبلر در چند مرحله برنامه را اسمبل می کند هر مرحله را می توانیم یک فاز بنامیم .در فاز اول اسمبل کردن ،برچسب ها

با آدرس حقیقی دستور جیگزین می شوند. شید تا ینجا فهمیده باشید که برچسب ها جهت سهولت کار ما انسانها به کار می روند زیرا به خاطر سپردن یک نام بسیار راحت تر از به خاطر سپردن یک عدد چند بیتی است .
شبه کد زیر را در نظر بگیرید:

```
50 jmp Main
...
...
100 main: mov A,#10
```

بعد از فاز اول اسمبل کردن به

```
50 Jmp 100
.....
.....
100 mov A,#10
```

تبدیل می شود.برچسب main با آدرس آن یعنی عدد 100 جیگزین می شود.
ین هم از مفهوم و کاربرد برچسب.

گفتیم که زیر برنامه یک تکه از برنامه است . که با استفاده از دستور CALL فراخوانی می شود در انتهی زیر برنامه بیداز دستور RET استفاده گردد .
هر زیر برنامه ی بید دستور RET را داشته باشد . ین دستور برنامه را از دستور بعد از CALL ادامه می دهد .
شبه کد زیر را در نظر بگیرید :

```
Mov A,#10
Call display
Mov b,#11
…

Display:
….
….
….
Ret
```

در برنامه بالا پس از اجری دستور Call زیر برنامه display فراخوانی می شود . دستورات زیر برنامه display تک تک اجرا می شوند تا نوبت به اجری Ret برسد بعد از اجری دستورRET دستور mov B,#11 اجرا می شود .

سوالی که در ینجا ممکن است به ذهن خطور کند ین است که برنامه چگونه تشخیص می دهد که بعد از اجری RET نوبت اجری mov b,#11 است در حالی که آنها در مکانهی متوالی حافظه قرار ندارند؟

در بحث ثبات ها میکرو از ثباتی با نام PC یا شمارنده برنامه نام بردیم و گفتیم که ین ثبات همیشه آدرس دستور العمل بعدی را که بیداجرا شود را در خود نگه می دارد در برنامه بالا فرض کنید که میکرو می خواهد دستور

```
Call display
```

را اجرا کند PC در سیکل نهیی اجری دستور mov A,#10 آدرس دستور CALL را در خود ذخیره خواهد کرد در سیکل انهیی اجری دستور PC ، CALL که آدرس mov B,#11 را در خود دارد در پشته ذخیره می شود اشاره گر پشته یک واحد افزیش می یابد و آدرس معادل display در PC قرار می گیرد و زیر برنامه display دستور به دستور

اجرا می شود تا ینکه به دستور RET برسد در ین هنگام آدرس برگشت برنامه که در پشته ذخیره شده بود از پشته خوانده می شود ودر PC قرار می گیردو سپس از اشاره گر پشته یک واحد کم می شود بنابرین دستور بعدی که اجرا خواهد شد دستور mov B,#11 خواهد بود .

کاربرد دیگر زیر برنامه در برنامه ها بزرگ بیشتر دیده می شود . با استفاده از زیر برنامه می توانیم برنامه اصلی را به چند زیر برنامه بشکنیم و هر برنامه نویس یا تیم برنامه نویسی می تواند یکی از زیر برنامه ها را بنویسد و در آخر می توانیم با match کردن زیر برنامه ها ، برنامه اصلی را باز سازی کنیم . بنا برین با استفاده از زیر برنامه امکان کار گروهی چند برابر می شود.
مزیت دیگر استفاده از زیر برنامه ها ین است که یک برنامه نویس دیگر با مشاهده کد نوشته شده بوسیله ما راحت تر از تکنیک هی برنامه نویسی ما سر در می آورد . برنامه ی رابا 400 خط کد در نظر بگیرید و آن را با برنامه ی متشکل از 10 زیر برنامه با هر کدام 40 خط کد ، که هر دو برنامه عمل یکسانی را انجام می دهند مقیسه کنید . شما از الگوریتم کدام یک از دو برنامه بالا بهتر سر در می آوردید ؟
مسلما بررسی و نقد برنامه دوم که با استفاده از زیر برنامه نوشته شده است بسیار راحت تر است .همچنین در صورتی که بخواهیم برنامه را تغییر دهیم یا یرادی را از آن برطرف کنیم کار کردن با برنامه دوم بسیار راحت تر است و زودتر به نتیجه مورد نظر می رسیم.

فراموش نکنید که استفاده از زیر برنامه یک نوع تکنیک برنامه نویسی است و تشخیص زیر برنامه ی که بید نوشته شود خود فنی دیگر بری موفقیت است پس از همین حالا استفاده از زیر برنامه را بری نوشتن برنامه هی بزرگ تمرین کنید

ین هم کلیاتی در مورد زیر برنامه

آموزش میکرو کنترلر 8051 قسمت هشتم

تیمر ها

در ادامه مقالات آموزش برنامه نویسی 8051، در ین مقاله می خواهیم در مورد تیمر ها بحث کنیم.تیمر ها از اجزاء پر کاربرد پروژه هی 8051 می باشند.کار اصلی تیمر شمارش تعداد سیکل هی کریستال می باشد.چون هر دوره تناوب

کریستال در کسری از ثانیه رخ می دهد بنابرین می توان بری شمارش زمان سپری شده ، از آن استفاده کرد .

8051 داری دو عدد تیمر 16 بیتی می باشد که با نامهی تیمر یک و تیمر صفر شناخته شده اند .هر تیمر در حقیقت از دو ثبات 8 بیتی با نام TH و TL ساخته شده است .از تیمر ها در 4 مود کاری می توان استفاده کرد:

مود 00 که از تیمر به عنوان یک شمارشگر 13 بیتی استفاده می کند .

مود 01 از تیمر به عنوان شمارشگر 16 بیتی استفاده می کند .

مود 02 به عنوان شمارشگر 8 بیتی با راه اندازی خودکار استفاده می کند.(AutoReload)

مود 03 تیمر را به دو تیمر 8 بیتی تقسیم می کند که به آن حالت تسخیر شده می گویند.

در 4 مود کاری بالا مود هی یک و دو بیشترین کاربرد را در انجام پروژه ها دارند .تیمر 16 بیتی مود 01 می تواند از مقدار 0000 تا FFFFH بشمارد پس از رسیدن به مقدار FFFF تیمر پرچم خود را (TF) را یک می کند و مجددا از مقدار 0000 شروع به شمارش می کند.

در مود 2 که همان حالت AutoReload است تیمر به عنوان یک شمارشگر 8 بیتی از مقدار 00 تا FFh می شمارد البته در ین مود می توان بری تیمر تعیین کرد که چه مقدار بشمارد فرضا در برنامه ی احتیاج به ین پیدا می کنیم تا تیمر یک وظیفه ی را هر 100 میکرو ثانیه انجام دهد در ین حالت از مود 2 استفاده می کنیم . مقدار اولیه شمارش را در ثبات TH قرار می دهیم .و سپس پس از راه اندازی تیمر ین مقدار در ثبات TL قرار می گیرد و تیمر از مقدار تعیین شده تا FFh می شمارد . پس از رسیدن TL به FFh تیمر پس از یک کردن TF ، مجددا مقدارموجود در TH را در ،م کپی کرده و به همین ترتیب شمارش ادامه می یابد .

تیمر ها داری دو ثبات با نامهی TCON وTMOD هستندثبات ،TMOD بری مقدار اولیه دادن به تیمر مورد استفاده قرار می گیرند و ثبات TCON بری کنترل تیمر مورد استفاده قرار می گیرد ثبات TMOD همیشه بید قبل از راه اندازی تیمر مقدار دهی شود.شکل ثبات TMOD به صورت زیر است:

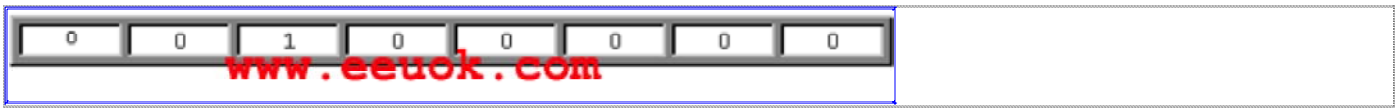
Gate	C/T	M1	M0	Gate	C/T	M1	M0

TMODیک ثبات 8 بیتی است که به دو قسمت 4 بیتی (nible) تقسیم می شود نیبل بالا بری تیمر 1 و نیبل پیین بری

تیمر 0 مورداستفاده قرار می گیرد .اختلاف دو نیبل در همین است و غیر از ین کاملا با هم مشابهند .
بیت Gate ، می توان گفت که یک نوع راه انداز تیمر است . ین بیت مشابه کلید روشن و خاموش عمل می کند در صورتی که مقدار ین بیت را یک قرار دهیم کنترل تیمر از طریق پالس ورودی به پیه ی INT1 صورت می گیرد یکی از کاربرد هی ین بیت اندازه گیری پالس خارجی اعمال شده به پیه ی INT1 می باشد.
بیت C/T نوع کارکرد تیمر را تعیین می کند که یا تیمر بری شمارش تعداد پالس کریستال خارجی متصل به پیه T1 یا T0 مورداستفاده قرار گیرد یا بری شمارش تعداد پالس هی کریستال مدار میکرو که آن هم به پیه هی 18 و 19 متصل است .در حالت اول می گویند که تیمر به عنوان کانتر (شمارنده اتفاقات) عمل می کند و در حالت دوم می گویند که تیمر در حالت شمارش گر زمان کار می کند .در صورتی که بیتc/T را یک کنیم تیمر به عنوان کانتر و در صورتی که ین بیت را صفر کنیم تیمر به عنوان شمارشگر واحد زمان به کار گرفته می شود.

بیتهی M1وM0 هم مود کاری را که قبلا بررسی کردیم تعیین می کند .ابتدا مود مورد نظر را تعیین می کنیم سپس مقدار معادل آن در مبنی دو را در بیتهی M1, M2 قرار می دهیم .

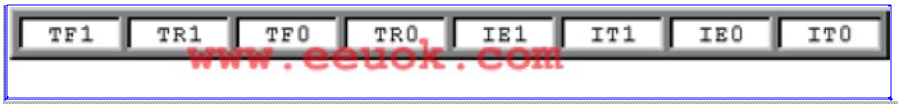
یک مثال بری ینکه تیمر 1در حالت AutoReload و به عنوان تیمر (شمارش گر زمان) مورد استفاده قرار گیرد :



معدل ین کار دستور زیر است:

```
Mov Tmod,#00100000b
```

مرحله بعدی استفاده از تیمر تعیین چگونگی روشن و خاموش کردن تیمر است.
ثبات دیگری که در مورد تیمر مورد استفاده قرار می گیردثبات TCON است بیتهی ین ثبات در شکل زیر مشخص می باشند.



بیت TF:قبلا هم در موردین بیت صحبت کردیم هنگامی که تیمر شروع به شمارش کرد بسته به مودی که بری کار در آن تنظیم شده است پس از رسیدن به مقدار نهیی (FFh یا FFFFh یا1FFFh) قبل از ینکه مجددا از صفر شروع به شمارش کند بیتTF را یک می کند تا نشان دهد که مقدار فعلی در دور دوم شمارش است و قبلا یک بار شمارش کامل شده است .

ین بیت به صورت سخت افزاری یک می شود و ما بید به صورت نرم افزاری قبل از پیان دور دوم شمارش آن را صفر کنیم .

TF1 بری تیمر یک و TF0 پرچم شمارش بری تیمر صفر است.

بیت TR: بیت کنترل راه اندازی نامیده شده است .در صورتی که ین بیت را یک کنیم تیمر شروع به شمارش می کند و در صورتی که ین بیت صفر شود تیمر متوقف می شود. مانند TR1، TF، بری کنترل تیمر 1 و TR0 بری کنترل راه اندازی تیمر 0 مورد استفاده قرار می گیرد.

4 بیت بعدی ین ثبات بری عملیات وقفه کاربرد دارد که انشاالله در مبحث وقفه ها به صورت کامل توضیح خواهیم داد .

اکنون روش کار با تیمر را به صورت الگوریتم در می آوریم تا بهخاطر سپردن آن آسانتر شود.

1.از کدام تیمر استفاده خواهیمکرد؟

2.مود کاری تیمر را تعیین کنید.

3.تیمر مورد نظر شماست یا شمارنده

3.ثبات TMOD را با معادل دودوئی آن بار گذاری کنید.

4.در صورتی که از AutoReload استفاده می کنید TH را با مقدار اولیه مناسب بار کنید.

5.تیمر را راه اندازی کنید .

7.منتظر یک شدن TF بمانید .

مثال:

برنامه ی بنویسید که یک پالس با دوره تناوب 200 میکرو ثانیه بر روی پیه ی P1.0 تولید کند.

حل:

دوره تناوب پالس 200 میکرو ثانیه عنوان شده است یک پالس در نصف دوره تناوب در حالت high و در نصف دیگر در حالت low قرار دارد . نصف دوره تناوب 100 میکرو ثانیه است و چون پالس متقارن است ما فقط به نصف دوره تناوب احتیاج داریم بنابرین مود تیمر را 2 یعنی حالت autoReload و مقدار اولیه را -100 در نظر می گیریم.

بری ینکه بفهمید چرا -100 ،توجه کنید که ما می خواهیم که تیمر 100 واحد بشمارد در حالت AutoReload چون تیمر 8 بیتی است حداکثر مقدار شمارش FFh است بنابرین :

TH=FFh-100d=9Bh

اما -100 هم دقیقا مشابه مقدار بالا است .بنابرین اگر ما به جی -100 مقدار 9Bh را در TH قرار می دادیم نیز نتیجه فرقی نمی کرد.

در ین مثال فرقی نمی کند که از کدام تیمر استفاده کنیم .در حقیقت تیمر ها هیچ تفاوتی با هم ندارند .

```
Mov TMOD,#00100000b

Mov TH1,#-100
Setb TR1
Wait :jnb Tf1,wait
ClR TF1
CPL p1.0
Sjmp wait
```

همانطور که مشاهده می کنید قبل از پیان دور دوم شمارش مقدار TF بید صفر شود.

میکرو کنترلر ها قسمت نهم عملیات سریال

در تراشه میکرو کنترلر 8051 یک پورت سری کار گذاشته شده است که می تواند اطلاعات را بیت به بیت ، پشت سر هم ارسال و دریافت کند پورت سری ابزار قدرتمندی برای ارتباط میکرو کنترلر با دنیای خارج است . ارسال و در یافت اطلاعات پورت سری از طریق پایه های RxDو TxD صورت می گیرد .
همانند دیگر عملیات های میکرو کنترلر استفاده از پوزت سری میکرو کنرترلر نیازکند انجام یک سری setting می باشد این تنظیمات توسط یک ثبات 8 بیتی با نام SCON انجام می گیرد .

ثبات SCON

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

پورت سریال میکرو کنترلر 8051 در 4 حالت یا مد می تواند کار کند که این چهار حالت توسط بیت های Sm0/sm1 انتخاب می شود .

چهار حالت نشان داده شده در بالا دارای فرکانس متغیر یا ثابت هستند حالت های فرکانس متغیر توسط سرریز تایمر یک و حالت های فرکانس ثابت بوسیله اسیلاتور میکرو کنترلر تامین فرکانس می شوند.

حالتهای پورت سریال در میکرو کنترلر 8051

SM0	SM1	حالت	عملکرد	نرخ ارسال
0	0	0	ثبات جابجایی	ثابت (فرکانس اسیلاتور تقسیم بر 12)
0	1	1	8UART بیتی	متغیر(توسط تایمر تعیین می شود)
1	0	2	9 UART بیتی	ثابت(فرکانس نوسان ساز÷32 یا16)
1	1	3	9 UART بیتی	متغیر

ارتباط سریال دارای یک استاندارد بانام UART است که مخفف Universal Asynchron Receive and Transmission است در این استاندارد هر حرف ارسالی بین دو بیت شروع و پایان قرار می گیرد و سپس ارسال می شود.

تشریح مد های ارتباط سریال :

حالت ثبات شیفت دهنده:
گفتیم که پایه های RxD و TxD جهت دریافت و انتقال اطلاعات در ارتباط سریال به کار می روند.اما در حالت ثبات شیفت دهنده فقط پایه ی RxD مورد استفاده قرار می گیرد و به عنوان گیرنده و فرستنده عمل می کند.پایه ی TxD پالس ساعت شیفت دهنده را آماده می کند در این حالت ابتدا کم ارزش ترین بیت ارسال می شود و سپس ارسال بیت به بیت تا پر ارزش ترین بیت انجام می شود . با هر پالس TxD در حالت ارسال اطلاعات یک بیت از داده از ثبات SBUF در RxD قرار می گیرد .

حالت فرکانس متغیر:

در حالت های 1 و3 فرکانس متغیر است و توسط تایمر 1 تعیین می شود.در مد 2 فرکانس اسیلاتور تراشه بر 16 یا 32 تقسیم می شود بسته به اینکه بیت SMOD که در ثبات PCON قرار دارد صفر باشد یا یک فرکانس دو برابر می شود.
در ارسال و دریافت 10، UART بیت اطلاعات ارسال می شود ابتدا یک بیت شروع و سپس 8 بیت داده و بعد از آن یک بیت پایان ارسال می گردد.
اما چگونه تشخیص دهیم که یک کاراکتر دریافت یا ارسال شده است؟
به محض اینکه بیت پایان بر روی خط TxD قرار گرفت پرچم TI که نشان دهنده این است که یک کاراکتر (یک بایت اطلاعات) ارسال شده است یک می شود . اما در دریافت اطلاعات به محض اینکه یک بایت اطلاعات دریافت شد بیت RI که نشان دهنده در یافت یک کاراکتر از پورت سریال است یک خواهد شد.

توجه :

بیت های TI و RI به صورت سخت افزاری یک می شوند و باید به صورت نرم افزاری صفر شوند.

در مباحث اطلاعاتی بایت به عنوان واحد اطلاعاتی قلمداد می شود یک بایت از 8 بیت تشکیل شده است . اما ما دو مد با نامهای UART 9 بیتی از 4 مد ارسال سریال داریم . این بیت اضافی چیست و در کجا قرار می گیرد؟

این بیت یک بیت انعطاف پذیر استکه برنامه نویس خود می تواند وظیفه ای را برای آن تعیین کند و اصطلاحا قابل برنامه ریزی توسط برنامه نویس است .بیشتر برنامه نویسان از این بیت به عنوان بیت توازن استفاده می کنند .
8 بیت اطلاعات ارسالی و یا دریافتی در ثبات SBUF یا بافر سریال قرار می گیرند و سپس پردازش بر روی آن صورت می گیرد ثبات SBUF در حقیقت از دو ثبات 8 بیتی تشکیل شده است که هردو دارای نامSBUF هستندو میکرو کنترلر بر اساس نوع عملیات انجام شده بر روی آن تشخیص می دهد که از کدام SBUF استفاده کند .اگر دستور خواندن از SBUF باشد میکرو کنترلر بر روی SBUF دریافت اطلاعات عملیات انجام می دهد و در صورتی که دستور به کار رفته دستور نوشتن اطلاعات در SBUF باشد میکرو کنترلر بر روی SBUF ارسال اطلاعات عملیات را انجام می دهد.

تا اینجا تکلیف 8 بیت اطلاعات را معلوم کردیم که در SBUF قرار گرفت اما بیت نهم در کجا قرار می گیرد ؟
در داخل ثبات SCON دو بیت با نامهای TB8 وRB8 وجود دارد که مکان قرار گیری بیت نهم می باشد.در صورتی که قصد ارسال اطلاعات را داشته باشیم بیت نهم باید درTB8 و در دریافت اطلاعات بیت نهم به طور خود کار در RB8 قرار می گیرد .پس برای دریافت اطلاعات ما فقط بیت نهم را از RB8 می خوانیم اما در ارسال اطلاعات بیت نهم به طور دستی باید در TB8 قرار گیرد.

ارسال و دریافت اطلاعات با استفاده از دستور mov صورت می گیرد SBUF یکی از اپراند های این دستور خواهد بود .
به محض نوشتن داده در SBUF ارسال اطلاعات آغاز می شود و هنگامی که آخرین بیت بر روی TxD قرار گرفت TI یک می شود . در دریافت اطلاعات با خواندن از SBUF داده در اختیار برنامه قرار می گیرد .
در دریافت اطلاعات باید برنامه منتظر یک شدن بیت RI از ثبات SCON شود و سپس داده را از SBUF بخوانیم .

```
Mov SCON,#01010000b
Wait : jnb RI,wait
Clr RI
Mov A, SBUF
```

دستورات بالا چگونگی دریافت داده را نشان می دهند .دستور اول مقدار دهی اولیه ثباتSCON است که شامل قرار دادن ارتباط در مد یک و فعال کردن بیت REN است .بیت REN مخفف Receive Enable است برای دریافت داده این بیت حتما باید یک باشد.

دستور دوم یک حلقه است که برنامه تا زمانی که بیتRI یک نشود حلقه را تکرار میکند و به محض اینکه یک بایت کامل دریافت شد میکرو به صورت خودکار بیتRI را یک می کند و سپس حلقه پایان می پذیرد .دستور CLR RI بیت RI را مجددا صفر می کند تا در دریافت بایت بعدی دوباره یک شود و آخرین دستور هم داده دریافتی که در ثباتSBUF قرار می گیرد را در ثباتA قرار می دهد.

در UART 9 بیتی نکته ای وجود دارد که باید به آن توجه شود در ارسال اطلاعات، قبل از دستور ارسال (مثلا mov sbuf,A) حتما TB8 باید مقدار دهی شود و در دریافت اطلاعات هم حتما بعد از دستور دریافت (mov a,sbuf) مقدار RI خوانده شود در غیر اینصورت نتیجه خلاف انتظار خواهد بود.

در این تکه کد ماTB8 را با بیت توازن فرد مقدار دهی می کنیم :

```
Mov c,p
Cpl c
Mov TB8,c
Mov SBUF,A
```

بیت p به صورت سخت افزاری برای توازن زوج مقدار دهی می شود با مکمل کردن آن می توان به توازن فرد رسید که اینکار را در دستور cpl c انجام داده ایم . بیت c(carry) در دستورات بیتی به عنوان واسطه عمل می کند در اینجا ما ابتدا مقدار بیت p را در c قرار داده ایم چون دستوری با شکل mov bit,bit در مجموعه دستورالعمل های 8051 نداریم مستقیما نمی توانیم دستور mov TB8,p را به کار بریم .

برای ارسال یک کاراکتر ابتدا چک می کنیم که TI صفر است یا نه؟اگر صفر باشد به معنی اینست که کاراکتر قبلی که ارسال کرده ایم هنوز ارسال نشده است و باید منتظر اتمام عملیات ارسال شویم و سپس TI را صفر کنیم و کاراکتر را ارسال کنیم:

```
Wait:jnb TI,wait
Clr TI
Mov SBUF,A
```

تولید پالس ساعت ارتباط سریال:

تایمر یک برای مد های 1و 3 که فرکانس متغیر است مسئول تامین فرکانس برای ارتباط است در حالت عمومی تایمر یک را در مد 2 یا حالت بار شدن خودکار قرار می دهند.
یک راه دیگر اینست که یک منبع پالس خارجی تنظیم شده را به پایه T1 متصل کنیم .

در هر صورت سرعت تبادل اطلاعات سریال از سرریز تایمر یک تقسیم بر 32(یا 16) صورت می گیرد .
محاسبه مقدار اولیه TH1 بدین صورت می باشد . فرض کنید می خواهیم سرعت ارتباط1200 بیت بر ثانیه(Baud Rate) باشد :

سرریز تایمر1 /32=1200
سرریز تایمر =1200*32 =38.4 KHZ

اگر فرکانس کریستال 12MHZ باشد در اینصورت:

فرکانس تیمر 1=فرکانس کریستال /12=1 MHZ=1000 KHZ

و برای 1200 بیت بر ثانیه فرکانس سرریز باید 38.4 باشد:

1000÷38.4=26.04

پس در هر 26.04 پالس ورودی باید یک سرریز داشته باشیم ما مقدار اولیه TH1 را -26 قرار می دهیم :

```
Mov th1,#-26
```

البته چون به جای 26.04 - ما عدد -26 را قرار داده ایم مطمئنا مقداری خطا خواهیم داشت و سرعت تبادل اطلاعات دقیقا 1200 نخواهد بود.

مثال :

پورت سری را برای ارسال و دریافت 2400 بیت بر ثانیه در حالت UART 8 بیتی مقدار دهی کنید ؟

فرکانس سرریز تایمر 1/32=2400

فرکانس سرریز تایمر 1=2400*32=76.8

12 کریستال MHZ

1000 /76.8=13.02
=? TH1=-13

```
Mov scon ,#01010000b
Mov Tmod , #00100000b
Mov TH1,#-13
SetB TR1
```

یک بیت دیگر در ثبات Scon با نام SM2 باقی مانده است که در ارتباطات مالتی پروسسوری کاربرد دارد که به زودی در مقاله ای جدا گانه به آن خواهیم پرداخت.

آموزش میکرو کنترلر قسمت دهم وقفه ها در 8051

ابزار دیگری که در تراشه ی 8051 پیاده سازی شده است مدارات وقفه می باشد . وقفه به معنی توقف موقتی می باشد و در علوم کامپیوتر وقفه به حالتی گفته می شود که پروسسور عملیات برنامه فعلی (میزبان) را متوقف کرده و برنامه دیگری (مهمان) را اجرا می کند پس از اجری برنامه (مهمان) پروسسور دوباره اجری برنامه میزبان را از جی که متوقف کرده بود از سر می گیرد .

وقفه تکنیکی است بری بهینه سازی و مدیریت زمان میکروپروسسور و همچنین آسان شدن برنامه نویسی و جلوگیری از رشد حجمی یک برنامه پیوسته باشکست آن به زیر برنامه هی کوچکتر که در ین حالت سر در گمی برنامه نویس را در پی نخواهد داشت .

وقفه در اصل زیر برنامه ی است که در شریط خاصی اجرا می شود و برنامه نویس بعضا زمان رخ داد ین شریط را نمی داند به زیر برنامه وقفه روتین سرویس وقفه (Interrupt service Routine) یا ISR می گویند.وقفه ها در میکرو کنترلر 8051 به سه دستهخ وقفه تیمر وقفهی خارجی و وقفه سریال تقسیم می شوند . ین سه دسته در کل شامل 5 منبع وقفه هستند که در ادامه با آنها بیشتر آشنا می شویم .

در بحث مربوط به تیمر ها گفتیم که هنگامی که بیت TFX تیمر یک شود یعنی یک دور شمارش انجام گرفته است و می توانیم دستوراتی را که می خواستیم پس از سپری شدن ین زمان انجام دهیم پس از چک کردن مقدار بیت TFx بنویسیم که شکل عمومی ین دستورات به صورت زیر بود:

```
Wait :jnb TF1,wait
CLR TF1
```

در ین روش میکرو کنترلر غیر از چک کردن بیت TF1 نمی تواند عمل دیگری را در انجام دهد اما در صورتی که از وقفه استفاده شود دیگر میکرو کنترلر زمانی را که در برنامه بالا به انتظار نشسته است را به عملیات دیگری اختصاص خواهد داد .

همچنی در بحث ارتباط سریال هم به همین شکل زمانی را در انتظار یک شدن بیت هی مربوط به عملیات سریال

(TI,RI) منتظر می ماند :

```
Wait : jnb TI,wait
CLR Ti
```

در صورت استفاده از وقفه ین زمان هم به زمان مفیدی تبدیل می شود.

کاربرد وقفه در دو تکه کد بالا را می توان بدین صورت توضیح داد :
فرض کنید در منزلتان نشسته ید تلفن زنگ می زند و دوست شما به شما خبر می دهد که به منزلتان خواهد آمد . دو حالت زیر را در نظر بگیرید :
1. شما هر یک دقیقه از پنجره به کوچه نگاه می کنید که کی دوستتان به پشت در خانه می رسد تا در را بری او باز کنید .
2. دوست شما خود پس از رسیدن به پشت در ، زنگ در خانه را به صدا در می آورد و شما در را بری او باز می کنید .

کدامیک از 2 حالت بالا بهتر است ؟مطمئنا شما هم حالت دوم را ترجیح می دهید.
در حالت اول شما نمی توانید کار دیگری انجام دهید چون بید هر یک دقیقه به کوچه نگاه کنید اما در حالت دوم به راحتی مشغول کاری می شوید و به محض ینکه زنگ خورد در را باز می کنید .
وقفه دقیقا مشابه زنگ در مثال بالا است.

در 8051 5 منبع وقفه وجود دارد دو منبع وقفه بری تیمر ها ، دومنبع وقفه خارجی ، و یک منبع وقفه سریال .
در طول ین چند مقاله آموزشی در مورد میکرو کنترلر ها یاد گرفته ید که بری استفاده از یک امکان در در میکروکنترلر ف مانند ارتباط سریال – تیمر و.. . بید ثبات هی مخصوصی را تنظیم کنید وقفه ها هم داری دو ثبات هستند که مقدار دهی آنها استفاده از وقفه را مهیا می کند.

اولین کاری که در استفاده از وقفه بیدانجام دهیم فعال کردن وقفه می باشد . یک ثبات با نام IE(Interrupt Enable) مسئول فعال کردن وقفه ها می باشد .

ثبات فعال کننده وقفهIE در 8051

EA	--	ET2	ES	ET1	EX1	ET0	EX0

EX0: منبع وقفه ی خارجی 0

ET0: منبع وقفه ی تیمر 0

EX1: منبع وقفه ی خارجی 1

ET1: منبع وقفه تیمر 1

Es: منبع وقفه ی سریال

ET2: منبع وقفه ی تیمر 2(بری 8052)

--: استفاده نشده

EA: فعال کردن کلیه وقفه ها

بیت EA بری فعال یا غیر فعال کردن کلیه وقفه ها به کار می رود در صورتی که بخواهیم وقفه ی را فعال کنیم بید دو بیت را set کنیم بیت اول بیت متناظر با وقفه و بیت دوم بیتEA .

در صورتی که بیت وقفه مورد نظر یک اما بیت EA صفر باشد هیچ وقفه ی روی نخواهد داد بنابرین EA یک شرط لازم بری عملیات وقفه است.

فرض کنید که می خواهیم وقفه تیمر یک را فعال کنیم چون IE بیت آدرس پذیر است(تک تک بیت ها با نام قابل دسترسی هستند) می توان از دو را ین کار را انجام داد :

```
SETB ET1
SETB EA
```

راه دوم می توان از دستور mov استفاده کرد وکلیه بیتهی ثبات را مقدار دهی کرد :

```
Mov IE,#10001000b
```

روش اول فقط دو بیت را تغییر می دهد و بقیه بیت ها را تغییر نمی دهد اما روش دوم بیت هی دیگر IE را صفر می کند.

تقدم وقفه ها :

فرض کنید در یک سیکل ماشین دو وقفه همزمان با هم اتفاق بیفتد مثلا تیمر 1 سرریز کند و یک بیت داده هم در همان زمان انتقال داده شود میکرو کنترلر از کجا بید بداند که مدام وقفه زودتر از دیگری بید انجام گیرد (هر دو بیستی انجام گیرند اما کدام زودتر ؟) در ینجا بحثی با نام تقدم وقفه ها پیش می ید که طراحان IC بری آن هم چاره ی اندیشیده اند ثباتی با نام IP(Interrupt Priority) وجود دارد که دو سطح بالا و پیین تقدم را بری وقفه تعریف می کند . در صورتی که دو وقفه در دو سطح مختلف همزمان روی دهند وقفه سطح بالا تر زودتر انجام می شود .
ثبات IP بیتهی مشابه IE دارد:

ثبات تقدم وقفه IP 8051

PA	--	PT2	PS	PT1	PX1	PT0	PX0

با یک کردن هر کدام از بیتها وقفه متناظر با آن بیت به سطح بالا می رود و وقفه هیی که بیت متناظر آنها صفر است در سطح پیین اولویت قرار می گیرند.
به دستورات زیر توجه کنید :

```
Mov IE,#10011010b
Mov IP,#00010010b
```

دستور اول وقفه هی سریال و تیمر یک و تیمر صفر را فعال می کند . دستور دوم وقفه هی تیمر صفر و سریال را در سطح اولویت بالا قرار می دهد و وقفه تیمر یک در سطح اولویت پیین باقی می ماند . حال اگر دو وقفه ی سریال و تیمر یک همزمان روی دهند وقفه سریال رودتر انجام می شود .

حالت دیگری که ملزم به در نظر گرفتن آن هستیم احتمال روی دادن همزمان دو وقفه با یک سطح اولویت است .فرض کنید در مثال بالا تیمر صفر همزمان با دریافت یک بیت از پورت سریال سر ریز کند .کدامیک زودتر بید سرویس دهی شود ؟ تیمر صفر یا سریال ؟
در چنین مواقعی ترتیب اولویت اجرا بدین گونه تعیین می شود وقفه متناظر با بیت کم ارزشتر زودتر انجام می گیرد . در مثال بالا وقفه تیمر صفر زودتز انجام می شود.
وقفه ها با استفاده از پرچم ها فعال می شوند بیشتر ین پرچم ها را قبلا در عملیات تیمر و سریال به کار گرفته ید(مانند .(Tf1, TF0,TI,RI
بیت هی پرچم در ثبات هی متفاوتی قرار دارند .

وقفه	پرچم	ثبات SFR و موقعیت بیت
0 خارجی	IE0	TCON.1
1 خارجی	IE1	TCON.3
تایمر 1	TF1	TCON.7
تایمر 0	TF0	TCON.5
درگاه سریال	TI	SCON.1
درگاه سریال	RI	SCON.0
تایمر 2	TF2	T2CON.7(8052)
تایمر2	EXF2	T2CON.6

بیتهای پرچم وقفه8051

روتین سرویس دهی وقفه (ISR):

زیر برنامه ی است که می خواهیم در صورتی که وقفه روی داد ین زیر برنامه اجرا شود . فرق ISR با زیر برنامه هی معمولی در ین است که ISR با RETI پیان می پذیرد اما زیر برنامه معمولی با RET.

در نوشتن ISR حتما یادتان باشد که پرچم ها را نرم افزاری با دستور CLR صفر کنید .

```
Timer0_ISR:
CLR TF0
SETB P1.0
RETI
```

بردار وقفه:

هنگامی که یک وقفه روی می دهد آدرس مشخصی بسته به نوع وقفه در PC (Program Counter) ذخیره می شود .ین آدرس بردار وقفه نام دارد . در حقیقت بردار وقفه چندین بیت ابتدی ROM میکرو است که هر 8 بیت مر بوط به یکی از وقفه ها می باشد جدول زیر آدرس وقفه ها را تشریح می کند.

وقفه	پرچم	آدرس بردارک
Resetسیستم	RST	0000H
0 خارجی	IE0	0003H
تایمر0	TF0	000BH
1 خارجی	IE1	0013H
تایمر1	TF1	001BH
درگاه سریال	TI یا RI	00023H
تایمر 2	EXF2یا TF2	002BH

بردار های وقفه 8051

هر بردار وقفه 8 بیت است به بردار مربوط به وقفه RESET که 3 بیت است . در وقفه ی Reset می توانیم Setting هی اولیه ی پروژه را قرار دهیم زمانی که کلد Reset زده می شود ین وقفه انجام می شود.

مسلما در 8 بیت نمی توان زیر برنامه نوشت راه حل ینست که در بردار وقفه فقط دستورات پرش به زیر برنامه ی را که بری وقفه در انتهی برنامه اصلی نوشته یم قرار می دهیم . مثلا بری تیمر صفر ین کار را بدین صورت انجام می دهیم:

```
ORG 0Bh
Ljmp Timer0_ISR
RETI
...
..
...
ORG 100h
Timer0_ISR:
....
.....
```

```
.....
RET
```

در تکه برنامه بالا ، توجه کنید که در انتهی زیر برنامه RET و در انتهی بردار وقفه RETI قرار داده یم در صورتی که RETI را فراموش کنید ممکن است دستورات مربوط به وقفه هی پیین تر هم اجرا شود پس RETI را فراموش نکنید.

وقفه هی خارجی به پالس اعمال شده به پیه هی INT0 وINT1 حساس می باشند و چون ین دو پیه فعال صفر هستند در لبه پیین روند پالس روال وقفه را اجرا می کنند .
وقفه سریال با نتیجه حاصل از OR کردن پرچم هی TI و RI فعال می شود.

نویسنده : مهدی موسوی