فصل اول

میکروکنترلر ۸۰۵۱

مقدمه:

به طور ساده می توان گفت که میکروپروسسور یا CPU مقدار پیچیده الکترونیکی است که می تواند انواع عملیات ریاضی و منطقی و کنترلی و... را انجام دهد اما از بین این مدارات موجود کدام عمل را انجام دهد را استفاده کننده باید به وسیله کدهای متناظر با این عملیات به میکروپروسسور اعمال کند. کدهای متناظر با عملیات مورد درخواست استفاده کننده مطابق شکل زیر به ترتیب در خانه حافظه برنامه که معمولاً از جنس ROM یا اجزای آن می باشد قرار گرفته و سپس با روشن شدن دستگاه میکروپروسسور با رجوع متوالی به این خانه ها و از روی کد مربوط به ان عمل آن را انجام داده و سپس به سراغ کد دستور بعدی و ... می رود.

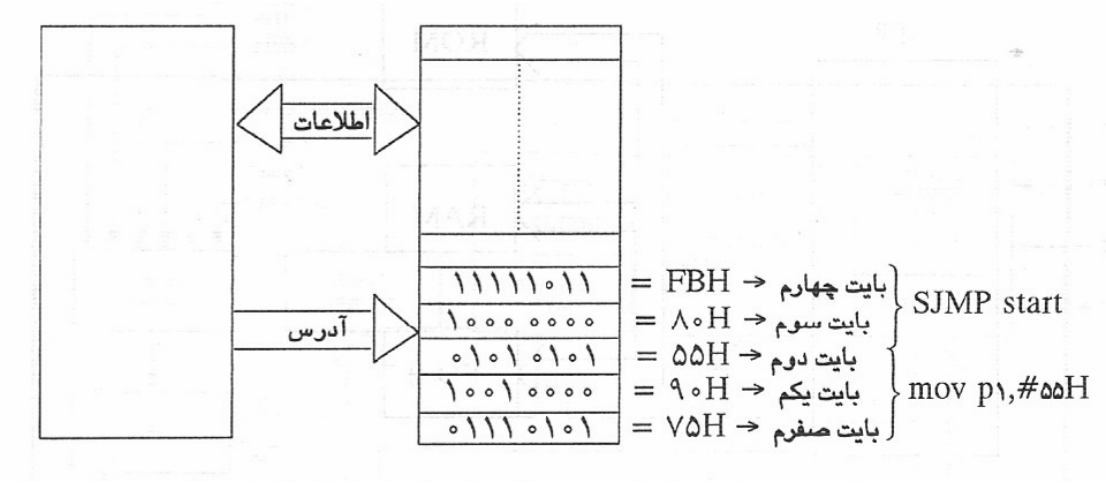
شکل ۱ نحوه قرار گرفتن کد دستورات یک برنامه ساده دو خطی به صورت زیر را نشان می دهد.

Start:MOV P1. ≠ 55H

SJMP Start

همچنانکه در فصلهای بعدی می آید خط اول به معنای قرار دادن عدد 55H در پورت P1 و خط دوم پرش به خط اول و تکرار عمل فوق می باشد. چنانچه دیده می شود دستور خط اول یا 55H ≠ و MOV P1 به صورت ۳ بایت کد شامل 75H و 55H می باشد که در

بایت های صفر و یک و دو قرار می گیرد و میکروپروسسور هنگامی که دستگاه روشن مـی شود با رجوع به این سه خانه دستور مذکور را انجام داده و سپس به سراغ خانـه بعـدی مـی رود آنگاه با دریافت دو بایت سوم و چهارم که مقادیر 80H (کد دستور SJMP) و FBH (مکمل دو تعداد خانه هایی که باید به عقب بر گردد یـا مکمـل ۲ عـدد ۵) در آن قـرار دارد مجدداً به خانه صفر رجوع و عملیات فوق را تکرار می کند.

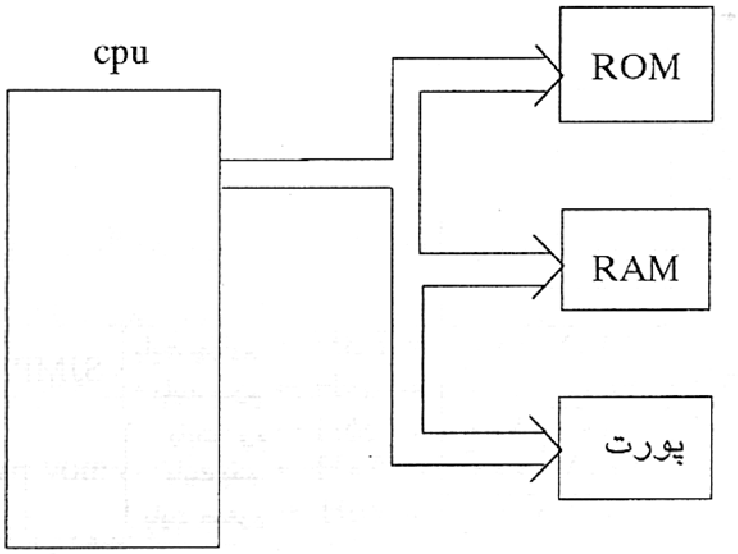


شکل ۱ قرار گرفتن کدهای برنامه در حافظه برنامه

چگونگی قرار گرفتن کدهای برنامه در حافظه برنامه و همچنین تولید کد برنامه ... و نیاز بـه نرم افزارهای اسمبلر و دستگاه پر کننده حافظه برنامه (programmer) و نرم افزار مربـوط به آن جهت ارتباط کامپیوتر با این دستگاه را دارد که توضیح کامل آن را در ضـمیمه ۲ آمـده است.

اگرچه حافظه برنامه به همراه CPU برای اجرای یک برنامه ساده کافی می باشد اما برای یک برنامه پیچیده تر نیاز به مکان هایی برای ذخیره موقت اطلاعات می باشد که برای همین در یک سیستم میکروپروسسوری علاوه بر حافظ برنامه (که معمولاً از جنس) ROM و یا اجزای آن می باشد.) به حافظه اطلاعات (که معمولاً از جنس RAM می باشد) نیز نیاز است. به علاوه برای اعمال ورودی به یک سیستم میکروپوسسوری و همچنین نمایش نتیجه برنامه در خروجی نیاز به یک سری پورتهای ارتباطی بین CPU و وسیله خارجی می باشد تا اجزایی نظیر صفحه کلید، کلیدهای قطع و وصل و ... و یا سون سگمنت، LCD و ... بتوانند با CPU ارتباط برقرار کنند.

میکروکنترلر ۸۰۵۱ تمام قطعات ذکر شده ا یکجا در داخل خودش دارد که به وسیله ٤٠ پایه بیرونی استفاده کننده می تواند با آن ارتباط برقرار کند بدون آنکه به RAM یا ROM و یا پورت بیرونی نیاز داشته باشد.



شکل ۲ شمای اتصال میکرو پروسسور با دیگر اجزای سیستم

۱-۱- مدار داخلی میکروکنترلر ۸۰۵۱

شکل ۱-۱ شمای داخلی آی سی میکروکنترلر ۸۰۵۱ که یک آی سی ٤۰ پایه اس را نشان می دهد. چنانچه دیده می شود در داخل این آی سی علاوه بر cpu اجزای زیر نیز وجود دارد.

۱- ٤^k بابت ROM (برای ذخیره کد برنامه)

۲- ۱۲۸ بابت RAM (برای نوشتن و خواندن اطلاعات)

۳- چهار پورت ورودی- خروجی هشت بیتی(پورت موازی)

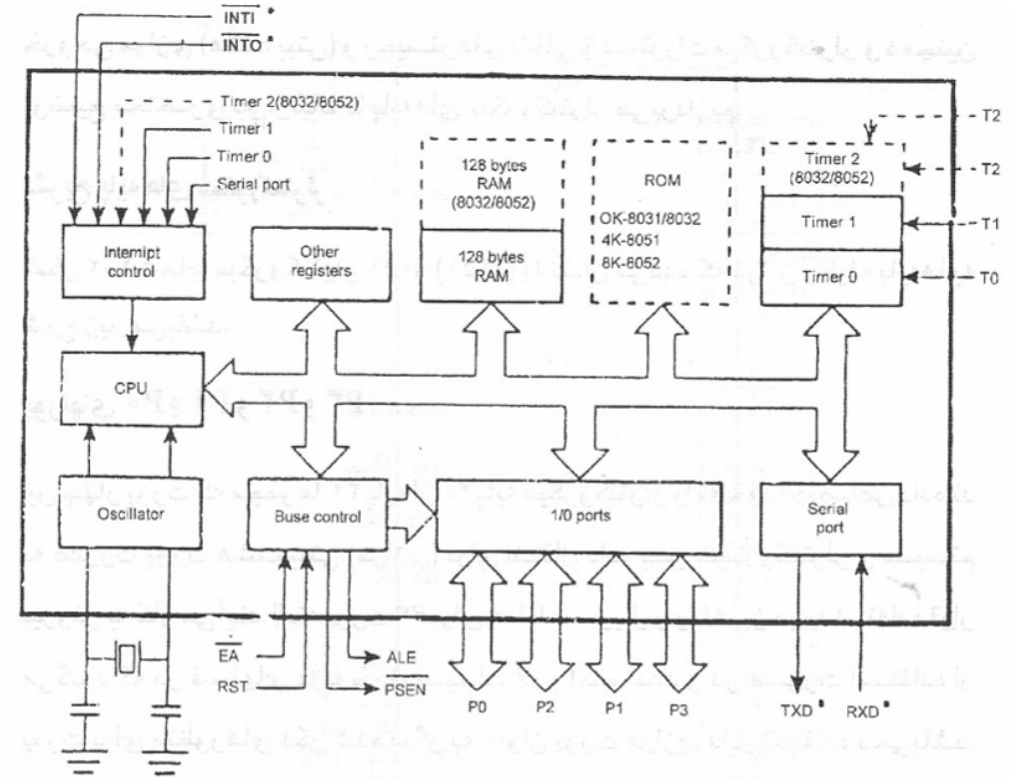
٤- پورت سریال

٥- دو عدد تایمر (شمارنده) ١٦ بیتی

٦- کنترل کننده وقفه

وجود چنین اجزایی در داخل این آی سی باعث شده است تا طراحی این مدار میکروپوروسسری برای حالاتی که حافظه زیادی نیاز ندارد به راحتی و با کمترین اتصالات بیرونی امکان پذیر گردد. اگرچه برای مدارهایی که نیاز به حافظه بیشتری باشد امکان اتصال حافظه RAM , ROM خارجی (تا ٦٤ کیلو بایت) نیز می باشد. همین سادگی اتصالات سخت افزاری باعث شده است تا این میکروکنترلر در بین دانشجویان و افراد متخصص در صنعت برای طراحی مدارهای میکروپروسسوری طرفداران فراوانی داشت باشد البته در عمل

به جای استفاده از ۸۰۵۱ که در حافظه ROM می باشد بهتراست از میکروکنترلر ۸۷۵۱ کـه در آن EPROM (ROM قابل برنامه ریزی و قابل پاک شدن با اشعه ماوراء بـنفش) قـرار دارد استفاده شود و از آن بهتر به کار بردن ۸۹۵۱ است که دقیقاً شبیه ۸۰۵۱ و ۸۷۵۱ بـوده و فقط به جـای ROM (۸۰۵۱) و یـا EPROM (۸۷۵۱) از EEPROM یـا ROM قابـل برنامه نویسی و قابل پاک شدن با سیگنال الکتریکی در آن استفاده شده است.



شکل ۱-۱ شمای داخلی میکرو کنترلر ۸۰۵۱

در مسائل این کتاب نیز از آی سی میکروکنترلر ۸۹۵۱ استفاده شده است که بلوک دیاگرام و مدار داخلی و پایه های آن همان شکل نشان داده شده برای ۸۰۵۱ مـی باشـد بـا توجـه بـه توضیحات بیان شده برای این بلوک نکات مرتبط با پورت سریال و تایمر (شمارنده) وقفه و RAM داخلی و خارجی در فصلهای مربوطه به طور کامل بررسی می گردنـد امـا در ابتـدا

برای شروع به پورت های ورودی- خروجی موازی (هشت بیتی) و رجیسترهای داخلی و دستورات میکروکنترلر و همچنین توضیح مختصری در ارتباط با پایه های میکروکنترلر می پردازیم.

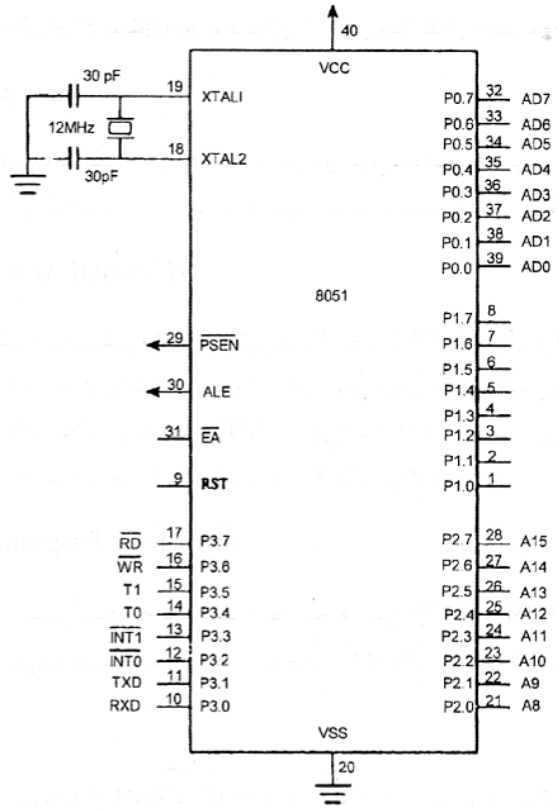
تشریح پایه های میکروکنترلر

شکل ۲-۱ شمای میکروکنترلر ۸۰۵۱ (۸۹۵۱) را نشان می دهد که در آن وظیفه پایه ها به شرح زیر می باشد.

پورتهای P3,P2,P1,P0

این چهار پورت که مجموعاً ۳۲ از ٤٠ پایه میکروکنترلر را به خود اختصاص داده اند به صورت پورت هشت بیتی (موازی) برای انقال داده بین میکروکنترلر و سیستم بیرونی به کار می آیند. البته پورت P3 برای عملیات سریال و وقفه نیز مورد استفاده قرار می گیرد که در فصلهای مربوطه توضیحات لازم آمده است و در صورت استفاده از پورت برای منظورهای ذکر شده دیگر به عنوان پورت موازی قابل استفاده نمی باشد. همچنین پورتهای P0 , P1 نیز علاوه بر پورت هشت بیتی می توانند به عنوان خطوط آدرس ١٦ رقمی در دسترسی به حافظه های RAM و ROM خارجی مورد استفاده قرار گیرد. که در فصل مربوط به RAM خارجی توضیح کامل آن آمده است و طبیعتاً اگر برای چنین مقصودی استفاده گردند در این صورت به عنوان پورت انتقال داده هشت بیتی قابل استفاده نمی باشند. به هر

حال استفاده معمولی از این پورتها به عنوان پورت هشت بیتی برای انتقال داده می باشد که زودی ان را نشان می دهیم.



شکل ۲-۱ شمای پایه های میکروکنترلر ۸۰۵۱

البته برای استفاده از پورتها به عنوان I/O برای پورت P0 با توجه به ساختار داخلی ان لازم است که مقاومت های KQ ۱۰ (برای هر پایه از پورت) از پایه های این پورت به VCD وصل گردند و نکته دیگر آنکه در صورتی که از این پورتها به صورت دو منظوره ورودی و هم خروجی استفاده شود باید در آنها یک نوشت که درمدارهای ۹۶ و ۹۷ و ۱۲۸ و ۱۲۹ آمده است.

این پورت ها هم به صورت بایتی و هم بیتی قابل دسترسی می باشند که در مدارهای مختلف چگونگی آن نشان داده شده است.

پایه های XTAL2 , XTAL1

این دو پایه مطابق شکل برای تأمین فرکانس لازم میکروکنترلر مـی باشـد کـه سـخت افـزار خارجی مورد نیاز و اتصال آن (کریستال و دو عدد خازن ۳۰PF) در شکل نـشان داده شـده است.

پایه RST **(ری ست)**

با فعال شدن این پایه (یک شدن) به مدت حداقل دو سیکل ماشین و سپس غیر فعال شـدن آن میکروکنترلر کد دستور را از آدرس صفرم حافظه برنامه واکشی می کند.

پایه EA (External Access)

اگر این پایه یک گردد میکروکنترلردستورات را از حافظه ROM داخلی و اگر ایـن پایـه بـه زمین وصل گردد. میکروکنترلر دستورات را از حافظه بیرونی واکشی می کند برای مـدارهای این کتاب چون از حافظه ROM خارجی استفاده نشده اسـت. ایـن پایـه بـه VCD وصـل گردید توضیح بیشتر آن درمبحث RAM خارجی آمده است.

پایه PSE (Enable ?! Program)

این پایه برای زمانی است که بخـواهیم از حافظـه خـارجی (ROM , RAM خـارجی) در مدار استفاده کنیم و توضیح کامل آن در مبحث RAM خارجی آمده است.

پایه ALE

این پایه برای استفاده از RAM و ROM و پورت خارجی جهت ذخیره کردن ادرس در پورت P0 می باشد و توضیح کامل ان در مبحث RAM خارجی آمده است.

پایه Vcc و Vss :

پایه Vcc به $+5^{v}$ و پایه Vss به زمین مدار وصل می گردد.

حال پس از تشریح پایه های میکروکنترلر به امکانات داخلی آن از نظر برنامه نویس می پردازیم.

ثبات های کاربرد خاص (Special function registers)

هر میکروپروسسور دارای یک سری رجیستر داخلی است که معمولاً دستورات برروی این رجیسترها قابل اجرا است و به اضافه اینکه از این رجیسترها به عنوان ذخیره کننده موقت اطلاعات نیز می توان استفاده کرد. در میکروکنترلر این رجیسترها به عنوان بخشی از RAM روی تراشه میکروکنترلر پیکر بندی می شود که شکل ۳-۱ آن را نشان می دهد. در این شکل رجیسترهایی که با شماره هایی بیت در داخل آن مشخص شده اند علاوه بر دستورات بایتی به صورت بیتی نیز قابل آدرس دهی می باشند (می توان هر بیت آن را مستقیماً با دستور صفر و یک کرد و یا دستور شرطی بر روی آن اجرا کرد و...) و آنهایی که

خالی است برای تولیدات بعدی میکروکنترلر از آن استفاده می گردد و نباید چیـزی در آنهـا نوشته شود و بقیه بایتهای نشان داده شده فقط به صورت بایتی قابل دسترسی اند (not bit addressable) البته از آنجایی که این رجیسترها به صورت RAM طراحی شده اند علاوه بر اسم خاص (نظیر A , B و PSW و...) به صورت شماره بایت نیز می توان به آن اسـتناد کرد. چنانچه دیده می شود شروع این رجیـسترها از خانـه ۱۲۸ 80H الـی ۲۵۵ FFH مـی باشـد کـه مـثلاً در آن خانـه 80H همـان پـورت P0 و 82H همـان DPL همـان A یـا آکومولاتور می باشد (۱۲۸ بایت اول یعنی از 0 الی ۱۲۷ از RAM چنانچه در فـصل شـشم می آید به عنوان RAM عمومی استفاده می گردند) به عنوان نمونه دو دسـتور زیـر هـر دو باعث می گردند تا عدد ۲۷ در آکومولاتور قرار گیرد.

MOV A, ≠27

MOV E0H,≠27

FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	D1	D0	PSW
B8	--	--	--	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A8	AF	--	--	AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	not bit addressable								SBUF
98	9E	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	not bit addressable								TH1
8C	not bit addressable								TH0
8B	not bit addressable								TL1
8A	not bit addressable								TL0
89	not bit addressable								TMOD
88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	not bit addressable								PCON
83	not bit addressable								DPH
82	not bit addressable								DPL
81	not bit addressable								SP
80	87	86	85	84	83	82	81	80	P0

Special Function Registers

شکل ۳-۱ رجیستر های کاربرد خاص(SFR)

و همچنین آنهایی که بیت آدرس پذیرند هم به صورت شماره بایت و شماره بیت و هم مستقیماً به صورت شماره بیت نشان داده شده قابل آدرس دهی هستند مثلاً دو دستور زیر هر دو به معنای یک کردن بیت دوم رجیستر PSW می باشد.

Setb psw.2

Setb D2H

در حالت کلی می توان گفت هر یک از این رجیسترها به منظور زیر در نظر گرفته شده اند.

آکومولاتور یا A(ACC): برای عموم دستورات انتقال و ریاضی و منطقی و مقایسه و شیفت و ... استفاده می گردد.

رجیسترB : برای دستورات ضرب و تقسیم (به همراه آکومولاتور) و دستوراتی که درضمیمه (۱) با عنوان direct مشخص شده است.

SP (Stka pointer) : به عنوان اشاره گر به فضای پشته

DRTA (Data pointer) : به عنوان اشاره گر به RAM خارجی و حافظه برنامه ROM که این رجیستر ۱٦ بیتی و به صورت دو بایت DPL و DPH نشان داده شده است.

P3,P2,P1,P0 : به عنوان پورتهای ورودی- خروجی هشت بیتی موازی البته پورت P3 برای وقفه خارجی و عملیات سریال و ... و پورتهای P2,P0 به عنوان آدرس , ROM RAM خارجی نیز قابل استفاده اند که در بخش های مربوطه بررسی می گردد.

(Interrupt Priority) , IE (Interrupt Enable) به عنوان رجیسترهای کنترل کننده عملکرد وقفه استفاده می شود و توضیح کامل در فصل های ۱۵و ۱٦ آمده است.

SCON : برای ارتباط سریال که در فصل شانزدهم بررسی می گردد.

SBUF : برای ارتباط سریال که در فصل شانزدهم بررسی می گردد.

TCON و TMOD : برای عملیات تایمر (کانتر) و سریال و وقفه مورد استفاده قرار می گیرد که در فصلهای ۱٤ به بعد بررسی می گردد.

TH0 , TL0 : رجیسترهای تایمر صفر می باشد که در مبحث تایمر بررسی می گردد.

TH1 , TL1 : رجیسترهای تایمر یک می باشد و در مبحث تایمر بررسی می گردد.

رجیستر PSW یا کلمه وضعیت برنامه

چنانچه دیده می شود این رجیستر بیت آدرس پذیر بوده و شماره آن D0H می باشد که در آن بیت های این رجیستر بیان کننده نکات زیر می باشند.

P یا بیت هفتم (PSW.7) : بیان کننده توازن زوج است هر برای هر سیکل ماشین تعداد بیهای آکومولاتور با این بیت به تعداد یکهای زوج منجر می گردد به عنوان مثال اگر آکومولاتور ۰۰۰۰۱۰۰۱ باشد (۲ عدد یک) آنگاه بیت هفتم یا P برابر صفر می گردد تا مجموعاً تعداد یک ها زوج باشد (۲ عدد) اما اگر ACC= ۰۰۰۰۱۰۱۱ باشد آنگاه P برابر یک می گردد. این بیت معمولاً برای ارسال و دریافت سریال به همراه بیت توازن به کار می آید که در فصل سریال مثال آن آورده شده است.

بیت ششم: رزرو شده است.

OV یا بیت پنجم (PSW.5) : در هنگام جمع و تفریق اعداد علامت دار می توان برای صحت جواب بیت سر ریز یا OV (Overflow) را تست کرد. در مدار ۲۲ توضیح آن آمده است.

PS1 یا بیت چهارم (PSW.4) و RS0 یا بیت سوم (PSW.3) این دو بیت برای انتخاب یکی از چهار بانک ثبات RAM داخلی می باشد که در فصل ششم در مورد آن بحث شده است. AC یا بیت یکم (PSW.1) : این بیت که به پرچم نقلی کمکی (Auxilary) معروف است. برای تصحیح نتیجه حاصل جمع دو عدد دهدهی به کار می آید. یعنی اگر در جمع دو عددی که به صورت دهدهی (BCD) در یک بایت قرار دارند نتیجه حاصل جمع موجود در آکومولاتور دهدهی نباشد آنگاه این بیت یک و در غیر این صورت صفر می گردد که با تست این بیت و با استفاده از دستور تنظیم دهدهی (DAA) جواب حاصل را می توان مجدداً به فرم دهدهی درآورد. در مدار ۳۱ این مضوع نشان داده شده است.

C یا بیت صفرم (PSW.7) این بیت که به پرچم نقلی (Carry) معروف است بیان کننده رقم نقلی خارج شده از بیت با ارزش هفتم در دستورات جمع و تفریق می باشد که در دستورات مختلف با همین نماد C از آن استفاده می گردد.

در این فصل رجیسترهای داخلی ۸۰۵۱ بحث و بررسی شده در فصلهای بعدی به وسیله برنامه های ساده به بررسی دستورات میکروکنترلر می پردازیم.

۲-۱- دستورات انتقال و ریاضی :

۱-۲-۱- دستورات انتقال و پرش غیر شرطی

دستور انتقال :

این دستور که بیشترین کاربرد را در برنامه نویسی اسمبلی میکروکنترلـر دارد دارای بیـشترین تنوع مد آدرس دهی نیز می باشد و فرم کلی این دستور که به وسـیله MOV یـا MOVX یا MOVC بیان می گردد به صورت زیر است:

الف) MOV Destination,source

ب) MOVC A,@A+DPTR

MOVC A,@A+PC

ج) MOVX A, @DPTR

MOVX @DPTR,A

MOVX A,@Ri

MOVX@Ri,A

از سه مورد ذکر شده دستور MOVC (Move code) یا حالت (ب) که برای انتقال کد از حافظه برنامه می باشد در فصل هفتم و در برنامه مربوط به جدول جستجو و دستور MOVX یا حالت (ج) که برای انتقال داده از حافظه RAM خارجی به آکومولاتور و بالعکس می باشد در فصل هجدهم و در مبحث RAM خارجی بررسی می گردد.

اما دستور حالت (الف) که برای انتقال داده بین رجیسترهای SFR و یا RAM داخلی میکروکنترلر می باشد به صورت MOVC Destinational, Source بیان می گردد که برای حالت بیتی بیت منبع را در داخل بیت مقصد و در حالت بایتی محتوای بایت مبع (Source) را در بایت مقصد (Destinational) قرار می دهد و تنها استثناء در این دستور رجیستر DPTR است که می توان یک عدد ۱۶ رقمی (۲بایتی) را در داخل آن قرار داد دستورات بیتی در فصل سوم بحث می گردد و در این قسمت به بررسی دستورات بایتی می پردازیم.

اثر بر روی پرچم	مد آدرس‌دهی	مثال	توضیح	فرمت دستور MOV

ندارد	آدرس‌دهی ثباتی (بانک رجیستری)	MOV A,R۰
		MOV A,R۵
		MOV ۲۴H,R۲
		MOV B,R۶
			R۱ یا R۰=R_n
			R۲ و R۲ یا ... R۷ یا
				MOV A,R_n
				MOV Direct,R_n

ندارد	آدرس‌دهی مستقیم (Direct Addressing)	MOV A,۳۰H
		MOV A,B
		MOV R۲,۴۰H
		MOV R۲,DPH
		MOV ۲۰H,۶۳H
		MOV ۱۰H,B
		MOV @R۰,۴AH
		MOV @R۱,A
			Direct یکی از رجیسترهای SFR و یا یکی از بایت‌های RAM داخلی از شماره ۰۰H الی ۷FH می‌باشد.	MOV A,Direct
				MOV R_n,Direct
				MOV Direct,Direct
				MOV @R_i,Direct

ندارد	آدرس‌دهی غیرمستقیم (indirect Addressing)	MOV A,@R۰
		MOV B,@R۱
		MOV ۵EH,@R۰
		MOV ۱۲H,@R۱
		MOV TL۰,@R۰
			Ri یکی از رجیسترهای R۰ یا R۱ می‌باشد (برای R۲ الی R۷ این دستور مجاز نمی‌باشد)	MOV A,@R_i
				MOV Direct,@R_i

ندارد	(آدرس‌دهی بلافصل یا فوری (Immediate Addressing)	MOV A,#۱۱A
		MOV R۳,#۲۷H
		MOV ۴۰H,# ۴۰H
		MOV B,# ۵FH
		MOV @Ri,# ۶۶
		MOV @R۰,# EEH
			data عددی یک بایتی از ۰۰H الی FFH می‌باشد.	MOV A, #data
				MOV R_n,# data
				MOV Direct, # data
				MOV @R_i,# data

			data۱۶ عدد دو بایتی یا ۱۶ بیتی	

جدول ۱-۲ حالت های ممکن برای دستور انتقال MOV بـایتی (و یـک حالـت خـاص دو بایتی و برای رجیستر DPTR) را نشان می دهد. حالت های دستور MOV در جدول کـه نشان دهنده مدهای مختلـف آدرس دهـی در میکروکنترلـر ۸۰۵۱ مـی باشـد بـه تناسـب در فصلهای مختلف توضیح داده خواهد شد و در این قسمت پس از توضیح مختصری در مورد جـدول بـه اسـتفاده از دسـتورات MOV در دو مـورد آدرس دهـی مـستقیم (Direct) و بلافصل (Immediate) از جدول مذکور می پردازیم.

الف) آدرس دهی ثباتی

۱- دستور Rn و MOV A :

در این دستور منبع Rn بیان کلی برای یکی از رجیسترهای R0 الی R7 می باشد و توضیح کامل آن در فصل ششم در تشریح بیان RAM میکروکنترلر امده است. که در نتیجه اجـرای این دستور محتوای رجیستر ذکر شده در دستور در داخل رجیستر مقصد یا آکومولاتور قـرار می گیرد به عنوان مثال دستور R2 و MOV A باعث می گردد تا محتوای رجیستر R2 در آکومولاتور قرار می گیرد.

۲- دستور MOV Direct , Rn

در این دستور Rn یا منبع همان رجیسترهای R0 الی R7 می باشد اما مقصد یا Drect یا بیان کننده ی یکی از رجیسترهای SFR نظیر A,B و DPL و IE و... (که در فصل اول بحث شد) می باشد و یا بیان کننده یکی از بایتهای RAM داخلی از شماره 00H الی FH7 که در فصل ششم بررسی می گردد می باشد.

به عنوان مثال دستور R4 و mov B باعث می گردد تا محتوای رجیستر R3 در بایت 69H از RAM داخلی میکروکنترلر قرار گیرد.

تذکر: چون در دستور Rn و mov Direct منبع محتوای یکی از رجیسترهای بانک ثبات (R0 الی R7) می باشد که در رجیستر مقصد قرار می گیرد به آن آدرس دهی ثباتی می گویند.

ب) آدرس دهی مستقیم

۱- دستور MOV A , Direct

محتوای منبع که یکی از رجیسترهای SFR و یا یکی از بایتهای RAM داخلی که درقسمت قبل بیان شد را در مقصد یا آکومولاتور قرار می دهد.

۲- دستور MOV Rn , Direct

محتوای منبع که یکی از رجیسترهای SFR و یا یکی از بایت های RAM داخلی است را در مقصد که یکی از رجیسترهای R0 الی R7 است قرار می دهد. به عنوان مثال دستور

3EH و MOV R2 محتوای بایت 3EH از RAM داخلی را در رجیستر R2 قرار می دهد.

۳- دستور MOV Direct , Direct

محتوای یکی از رجیسترهای مستقیم (Direct) بیان شده در قسمت قبل را در یکی از رجیسترهای Direct قرار می دهد. به عنوان مثال دستور 63H و MOV 15H محتوای 30H hc RAM داخلی را در رجیستر B از SFR قرار می دهد.

٤- دستور MOV @ Ri , Direct

محتوای یکی از رجیسترهای مستقیم منبع را در مقصد که همان ادرس اشاره شده به وسیله Ri (R0 یا R1) می باشد قرار می دهد. به عنوان مثال در دستور TL0 و MOV @ R0 اگر محتوای R0 برابر 35H باشد آنگاه محتوای رجیستر TL0 در بایتی از RAM داخلی که R0 به آن اشاره می کند (یعنی 35H) قرار می گیرد.

تذکر: چون در تمام دستورات بیان شده مقدار منبع به وسیله بیان مستقیم (Direct) رجیستر مورد نظر ذکر گردیده است به آن آدرس دهی مستقیم گفته می شود. یعنی در دستور منبع به وسیله نام بایت نظیر B , A و DPL و TL0 و... (رجیسترهای SFR) و یا به نظیر شماره بایت نظیر 20H و 3EH و 75H و... (بایت های RAM داخلی) ذکر می گردد.

ج) آدرس دهی غیر مستقیم

۱- دستور MOV A , @Ri

محتوای منبع را که به وسیله رجیستر Ri (R0 یا R1) به شماره آدرس آن اشاره می گـردد در داخل رجیستر مقصد یا آکومولاتور قرار می دهـد. بـه عنـوان مثـال دسـتور MOV A , @R1 با فرض آنکه R1 برابر 53H باشد محتوای بایـت 53Hکـه بـه وسـیله R2 بـه آن اشاره می گردد را در آکومولاتور قرار می دهد.

۲- دستور MOV Direct , @ Ri

محتوای منبع را که به وسیله رجیستر Ri (R0 یا R1) به شماره آدرس آن اشاره می گردد را در داخل رجیستر مقصد که به صورت مستقیم (Direct) در دستور ذکر مـی گـردد قـرار می دهد. به عنوان مثال دستور MOV 7CH , @ R0 با فرض آنکه R0=12. باشد باعث می گردد تا محتوای بایت ۱۲ از حافظه RAM داخلـی در بایـت 7CH از RAM داخلـی قرار گیرد.

تذکر: چون در دستورات بیان شده برای این حالت مقدار منبع در دسـتور بـه وسـیله آدرس دهی یکی از دو رجیستر R0 یا R1 مشخص می گردد به آن آدرس دهی غیر مستقیم گفتـه میشود. به عبارت دیگر در این روش به جای آنکه مستقیماً جایگاه منبع نظیـر حالـت آدرس دهی مستقیم مشخص گردد به صورت غیر مستقیم و به وسـیله رجیـسترهای R0 , R1 بـه جایگاه منبع اشاره می گردد. این نوع ادرس دهی برای عملیات رشته ای مناسب می باشد.

د) آدرس دهی بلافصل یا فوری

۱- دستور MOV A , # data

محتوای منبع را که یک عدد مشخص یک بایتی از صفر الی ۲۵۵ (H ۰۰ الـی FFH) اسـت در رجیستر مقصد یا آکومولاتور قرار می دهد. به عنـوان مثـال دسـتور MOV A , # ۱۰۳ عدد ۱۰۳ را در داخل آکومولاتور قرار می دهد.

۲- دستور MOV Rn , data

محتوای منبع که یک عدد مشخص یک بایتی است را در یکی از رجیستر R0 الی R7 قـرار می دهد به عنوان مثال دستور MOV R3 , #10H عدد 10H را در رجیستر R3 قرار می دهد.

۳- دستور MOV Direct , #data

محتوای منبع که یک عدد مشخص شده یک بایتی است را در رجیستر مقصد که به صـورت مستقیم (Direct) نام آن در دستور بیان می گردد قـرار مـی دهـد. بـه عنـوان مثـال دسـتور MOV 4CH , # 0 عدد صفر را در بایت 4CH از RAM داخلی قرار می دهد.

٤- دستور MPV @ Ri , # data

مقدار عدد را که یک عدد مشخص شده یک بـایتی اسـت را در مکـانی از حافظـه (RAM داخلی) که رجیستر Ri (R0 یا R1) به آن اشاره می کند قرار می دهد.

توجه: دستور 20H عدد 30H را در داخل آکومولاتور قرار مـی دهـد امـا دسـتور 20H و MOV A محتوای بایت 20H از RAM داخلی را در آکومولاتور قرار می دهد.

تذکر: چون برای دستورات ذکر شده در این حالت مقدار منبع یک عدد مشخص مـی باشـد که مستقیماً و بی واسطه در دستور ذکر می گردد به آن آدرس دهی بلافـصل یـا فـوری مـی گویند.

ح) دستور انتقال ۱۶ بیتی به وسیله رجیستر DPTR

ایـن رجیـستر کـه بـه عنـوان اشـاره گـر بـه مکانهـای حافظـه برنامـه ROM و اطلاعـات (ROM خارجی) به کار می آید یک رجیستر ١٦ رقمی بـوده کـه در میکروکـونترلر امکـان عدد دهی مستقیم آن وجود دارد یعنی دستور ۱۲۳۵ #, MOV DRTR باعث می گردد تـا عدد ۱۲۳۵ در آن قرار گیرد. لازم به ذکر است که عدد دهی رجیستر DPTR به صورت ١٦ رقمی فقط به وسیله آدرس دهی بلافصل امکان پذیر است.

بحث کاملتر دستورات انتقال و مدهای مختلف آدرس دهی و مثـال هـی مربـوط بـه آن بـه تناسب در فصلهای ششم و به بعد می آید و در این فصل به دو مورد از ایـن دسـتور کـه در مدارهای این فصل مورد نیاز است می پردازیم.

همچنانکه ذکر شد فرم کلی دستور انتقال داده بین RAM و رجیسترهای SFR به صـورت MOV Destination می باشد که با اجـرای ایـن دسـتور محتـوای منبـع (Source) در داخل مقصد (Destinational) قرار می گیرد اما در دستورات این فصل حـالتی کـه در آن منبع بایتی یعنی بین صفر الی ۲۵۵ (00H الی FFH) می باشد و رجیستر مقصد نیز یکی از

رجیسترهای SFR می باشد بررسی می گردد و دستوراتی که در آن رجیسترهای منبع و مقصد می توانند حالات دیگری را نیز به خود بگیرند در فصل های ششم و به بعد آمده است. مثال های زیر وضعیت رجیسترها را پس از اجرای دستور MOV نشان می دهد.

مثال ۱: دستور MOV A , #37H که بیان کننده آدرس دهی بلافصل می باشد باعث می گردد تا عدد37H در داخل آکومولاتور (از رجیستر SFR) قرار گیرد. به عبارت دیگر پس از اجرای این دستور محتوای آکومولاتور یا ACC یا A برابر است با:

Acc=

0	0	1	1	0	1	1	1

مثال ۲: دو دستور مقابل باعث می گردد تا ابتدا عدد MOV B , #5EH

در داخل رجیستر B قرار گرفته (آدرس دهی بلافصل) MOV A,B

و سپس در دستور بعد یعنی MOV A,B مقدار رجیستر B (که برابر 5EH می باشد)

در داخل آکومولاتور (A) قرار گیرد (آدرس دهی مستقیم) یعنی بعد از اجرای این دو دستور محتوای دو رجیستر A , B به صورت زیر می باشد:

A=B=

0	1	0	1	1	1	1	0

مثال ۳: در دستور A و MOV P2 اگر محتوای رجیستر A برابر A2H باشد آنگاه پس از اجرای دستور محتوای P2 برابر است با :

P2=

1	0	1	0	0	0	1	0

۲-۲-۱- دستورات جا به جایی

الف) دستور SWAP A

این دستور که به صورت SWAP A نوشته می شود باعث می شود تا محتوای نیم بایت کم ارزش آکومولاتور با محتوای نیم بایت با ارزش آکومولاتور با همدیگر عوض گردند.

تذکر: دستور SWAP محتوای آکومولاتور یا ACC یا A قبل از اجرای دستور مطابق شکل زیر برابر 94H باشد.

A=Acc=

1	0	0	1	0	1	0	0

آنگاه پس از اجرای دستور مقدار آن برابر است با :

A=Acc=

0	1	0	0	1	0	0	1

یعنی جای دو نیم بایت عوض می گردد.

ب) دستور XCH

این دستور به یکی از سه صورت زیر قابل بیان است:

۱- XCH A, Direct

۲- XCH A, Rn

۳- XCH A, Ri

بیان کامل حالتهای ۱و۲و۳و مفهوم آن در فصل ششم و پس از توضیح RAM داخل میکروکنترلر آمده است و در این فصل فقط به دستور شماره ۱ و حالت خاص آنکه در آن Direct بیان کننده ی رجیسترهای SFR می باشد اشاره می کنیم.

اما اجرای دستور XCH باعث می گردد تا محتوای آکومولاتور (A) با بایت مشخص شده به وسیله منبع با یکدیگر عوض گردند.

مثال : در دستور XCH A , B اگر محتوای رجیستر B قبل از اجرای دستور برابر B8H و محتوای آکومولاتور برابر 16H باشد آنگاه پس از اجرای دستور محتوای آکومولاتور برابر B8H و مقدار رجیستر B برابر 16H می گردد. شکل زیر مقادیر این رجیسترها را قبل و بعد از اجرای دستور نشان می دهد.

A=	0	0	0	1	0	1	1	0
B=	1	0	1	1	1	0	0	0
A=	1	0	1	1	1	0	0	0
B=	1	0	1	1	1	0	0	0

ج) دستور XCHD

این دستور که به صورت XCHD A , @Ri نوشته می شود فقط نیم بایت کم ارزش آکومولاتور را با محتوای نیم بایت کم ارزش بایتی که رجیستر Ri(R0 یا R1) به آن اشاره می کند عوض می کند. مفهوم کامل این دستور که به آدرس دهی غیر مستقیم معروف است در فصل ششم و پس از تشریح RAM داخلی میکروکنترلر می آید و در اینجا فقط با ذکر یک مثال آن را نشان می دهیم.

مثال : در دستور XCHD A , @ R0 اگر محتوای رجیستر R0 برابر 37H و محتوای بایت 37H از RAM داخلی برابر 46H و مقدار آکومولاتور برابر 5CH باشد آنگاه محتوای بایت 37H و آکومولاتورقبل و بعد از اجرای دستور به شرحزیر می باشد.

A=	0	1	0	1	1	1	0	0
37H=	0	1	0	0	0	1	1	0

A=	0	1	0	1	0	1	1	0
37H=	0	1	0	0	1	1	0	0

به عبارت دیگر جای نیم بایت آکومولاتور(1100B=CH) با نیم بایت، بایت (0110B=6H) عوض گردید.

۳-۱-۲- دستور مکمل کردن

این دستور که به صورت CPLA نوشته می شود جزء دستورات منطقی می باشد که در فصل سوم آمده است اما چون درمدارهای این فصل از آن استفاده شده است به توضیح ان می پردازیم:

دستورCPLA محتوای بایت آکومولاتور را به صورت مکمل (یکهای) آن در می آورد به عبارت دیگر بیتی از آکومولاتور که برابر صفر است یک و بیتی از آکومولاتور که برابر یک است صفر می گردد.

مثال: اگر در دستور CPLA مقدار آکومولاتور قبل از اجرای دستور برابر 3DH باشد آنگاه پس از اجرای دستور مقدار آن برابرC2H می باشد شکل زیر محتوای آکومولاتور را قبل و بعد از اجرای دستور نشان می دهد.

A=	0	0	1	1	1	1	0	1
A=	1	1	0	0	0	0	1	0

۳-۲-۱- دستور پرش غیر شرطی(Unconditional Jump)

دستور پرش غیر شرطی که به آن پرش مطلق نیز گفته می شود برنامه را بـدون هـیچ قیـد و شرطی به برچسب مشخص شده در دستور منتقل می کند. یکی از کاربردهـای ایـن دسـتور برای تکرار دائمی برنامه می باشد یعنی از آنجاییکه می خواهیم میکروکنترلر به طـور مـداوم یک عمل را پیگیری کند بنابراین پس از پایان یک دور اجرای برنامه مجـدداً آن را بـه نقطـه شروع عملیات بر می گردانیم که این توسط دستور پرش مطلق بـه نقطـه شـروع انجـام مـی گیرد و در تمام برنامه های کتاب از آن استفاده شده است. در حالت کلی پرش مطلق بـه دو صورت پرش کوتاه و بلند می تواند باشد.

دستور پرش کوتاه یا SJMP Lable

این دستور باعث پرش برنامه به برچـسب مـشخص شـده (Lable) در دسـتور مـی گـردد محدود پرش می تواند ۱۲۷ بایت رو به جلو و ۱۲۸ بایت رو به عقب باشد.

دستور پرش بلند یا LJMP Lable

این دستور باعث پرش برنامه به برچسب مشخص شده (Lable) در دستور می گردد.

محدود پرش در هر نقطه از حافظه ٤ کیلو بایتی ROM داخلی و یا ٦٤ کیلو بـایتی ROM خارجی می توانید باشد به عبارت دیگر برای پرش بلند مستقیماً شماره آدرس خانه پرش در داخل رجیستر شمارنده برنامه یا PC قرار می گیرد.

برتری دستور پرش بلند نسبت به پرش کوتاه محدوده آن می باشـد کـه بـه هـر نقطـه ای از برنامه می تواند پرش کند اما عیب آن در این است که کد دستور پرش بلند ۳ بایتی است اما

برای پرش کوتاه کد آن دو بایتی می باشد که در نتیجه یک بایت در حافظه برنامه صرفه جویی می گردد. توجه شود که به علت حجم کم حافظه ROM میکروکنترلر (٤ کیلو بایت حافظه داخلی) در صورت بزرگ بودن حجم برنامه این نکته می تواند اهمیت داشته باشد اما برای برنامه های کوچک این موضوع اهمیت زیادی ندارد.

البته علاوه بر دستورات ذکر شده دستور AJMP نیز برای میکروکنترلر وجود دارد که در آن اگرچه کد دستور دو بایتی است محدوده پرش می تواند دو کیلو بایت باشد یعنی برای این دستور کد رمز به صورت aaa00001aaaaaaaa می باشد که در آن ۳ بیت با ارزش و ۸ بیت کم ارزش به عنوان یازده رقم کم ارزش آدرس و پنج رقم با ارزش آدرس همان مقدار موجود در PC می باشد (پس از دو واحد افزایش) که مجموعاً یک عدد ١٦ رقمی آدرس را تشکیل می دهند از آنجاییکه پنج رقم PC توسط برنامه نویس تعیین نمی گردد بنابراین محدوده پرش توسط ۱۱ رقم کم ارزش صورت می گیرد و حداکثر می تواند تا ۲ کیلو بایت بالاتر از مکان فعلی باشد که اصطلاحاً می گویند با این روش حافظه برنامه به صورت صفحات ۲ کیلو بایتی در می آید.