فصل چهارم: دستورات اسمبلی و برخی از تنظیمات اولیه

عناوین مطالب:

- زبان اسمبلی در AVR
- دستورات انتقال داده
- مدهای آدرس دهی
- پورت‌ها و تنظیمات اولیه
- پشته و تنظیمات اولیه
- خواندن و نوشتن در حافظه برنامه
- برخی از دستورات راهنما

* زبان اسمبلی در AVR *

برای برنامه ریزی میکروکنترلرهای AVR، از زبان‌های مختلفی می توان استفاده نمود. زبان‌های برنامه نویسی C، Basic و اسمبلی سه زبان رایج برای برنامه ریزی میکرو کنترلرهای AVR است.

به طور کلی زبان‌های برنامه نویسی به دو دسته سطح بالا و سطح پایین تقسیم می‌شوند. زبان سطح پایین، زبانیست که هر یک از دستورالعمل‌هایش، معادل یک عمل واقعی در پردازنده یا CPU باشد. به عبارت دیگر هر خط از دستورات نوشته شده با یک زبان سطح پایین، معادل یک عملیات سخت افزاری در پردازنده است. اما در زبان های سطح بالا، یک دستورالعمل، می‌تواند معادل چندین عملیات سخت افزاری باشد. برنامه نویسی در هر یک از این دو سطح، مزایا و معایبی دارد. به عنوان مثال، برنامه نویسی با زبان های سطح بالا ساده تر است چرا که این زبان ها به زبان تکلم ما نزدیک تر هستند اما برنامه نویسی با زبان های سطح پایین قدری دشوارتر است به این دلیل که این زبان ها، از مجموعه‌ای از دستورات قابل فهم برای پردازنده تشکیل شده است و تسلط به آنها نیازمند تأمل بیشتری نسبت به دستورات زبان های سطح بالاست. در مقابل، زبان های سطح پایین به برنامه نویس اجازه می دهند که مستقیماً با پردازنده ارتباط برقرار کند و به آن فرمان دهد. این ویژگی باعث می شود که برنامه نویس از جزئیات عملیات پردازنده آگاه باشد و بتواند برنامه هایی دقیق تر و سریع تر بنویسد و در صورت نیاز به قسمت های مختلف حافظه برنامه و حافظه داده دسترسی داشته باشد.

مشهورترین زبان سطح پایین، زبان اسمبلی است که برای برنامه ریزی پردازنده های گوناگونی مورد استفاده قرار می‌گیرد. توجه داشه باشید که زبان اسمبلی در پردازنده های مختلف، شکل های مختلفی دارد. مثلاً زبان اسمبلی برای میکروهای AVR با زبان اسمبلی برای پردازنده های intel متفاوت است. اما از آنجایی که این تفاوت ها اندک هستند و دستورات، یک قالب کلی واحد دارند، به همه این زبان ها، اسمبلی گفته می شود.

فراگیری زبان اسمبلی، ضمن اینکه ما را با نحوه برنامه نویسی و مقدمات کار عملی با میکرو کنترلرهای AVR آشنا می کند، کمک شایانی به فهم عملکرد CPU و نحوه ارتباط آن با حافظه ها و سایر بخش‌های میکرو می‌کند. به همین دلیل در این فصل به بررسی دستورات اصلی زبان اسمبلی در AVR خواهیم پرداخت.

به صورت کلی دستورات اسمبلی در AVR به ۵ دسته زیر تقسیم می شوند:

- دستورات انتقال داده
- دستورات محاسباتی و منطقی
- دستورات انشعابی
- دستورات بیتی
- دستورات کنترلی

برای مشاهده لیست کامل این دستورات درمیکروکنترلر ATMega32، به پیوست ب مراجعه فرمایید.

در بخش ابتدایی این فصل به بررسی دستورات انتقال داده می‌پردازیم.

* دستورات انتقال داده *

غالب این دستورات، برای انتقال اطلاعات از حافظه داده به رجیسترهای عمومی و یا بالعکس تعبیه شده‌اند. سایر دستورات انتقال داده برای جابجایی اطلاعات میان رجیسترهای عمومی، خواندن از حافظه برنامه و نوشتن در آن مورد استفاده قرار می گیرند. عمده این دستورات عبارتند از:

- بارگذاری داده در رجیسترهای عمومی (Load):

LDI LDS LD LDD

- انتقال داده از رجیسترهای عمومی به حافظه داده (Store):

STS ST STD

- جابه‌جایی داده میان رجیسترهای عمومی:

MOV

- تبادل داده میان رجیسترهای عمومی و رجیسترهای فضای I/O:

IN OUT

- کار با پشته:

PUSH POP

- دسترسی به اطلاعات حافظه برنامه:

LPM

دستور LDI

- فرم نوشتن: LDI Rd,k[1]
- بررسی عملوندها: R16<Rd<R31 k یک عدد صحیح مثبت است
- مفهوم دستور: انتقال یک مقدار عددی به یکی از رجیسترهای عمومی (Load Immediate)
- خلاصه عملیات: Rd<---k
- نوع آدرس دهی: فوری (شرح مفهوم آدرس دهی، در ادامه همین فصل آمده است)
- توضیحات: به طور کلی دستورات Load، دستوراتی هستند که داده‌ای را به یکی از رجیسترهای عمومی منتقل می‌کنند. در دستور LDI، عدد k به رجیستر مورد نظر منتقل می شود.
- مثال:

LDI R16,0x55[2]

پس از نوشتن این دستور، مقدار رجیستر R16 برابر با ۵۵ در مبنای ۱۶ که معادل ۰۱۰۱۰۱۰۱ در مبنای ۲ است، می‌شود.

برای تبدیل مستقیم اعداد مبنای ۱۶ به مبنای ۲، کافیست که برای هر یک از ارقام عدد مورد نظر، چهار بیت در نظر بگیریم و سپس معادل دودویی (مبنای دو) هر رقم را به جای آن قرار دهیم. به عنوان مثال عدد D7 در مبنای ۱۶ برابر است با 11010111 در مبنای ۲. همان طور که مشاهده می کنید چهار رقم سمت چپ این عدد یعنی 1101، معادل مبنای دوی رقم D و چهار رقم سمت راستی عدد، معادل دودویی رقم 7 می باشد.

---

[1] Rd به معنای R destination یا رجیستر مقصد است.

[2] نوشتن عبارت 0x قبل از عدد به این معناست که عدد ذکر شده در مبنای ۱۶ می‌باشد. می‌توان به جای عبارت 0x از علامت $ نیز استفاده نمود. همچنین برای بیان اعداد در مبنای ۲، از عبارت 0b استفاده می کنیم و برای بیان اعداد در مبنای ۱۰، خود اعداد را بدون نوشتن علامت ذکر می‌نماییم.

دستور LDS

- فرم نوشتن: LDS Rd,k
- بررسی عملوندها: R0<Rd<R31

k آدرس یکی از خانه های حافظه داده است که بسته به حجم حافظه داده می تواند مقادیر مختلفی اختیار کند. مثلاً اگر حجم حافظه داده، معادل ۲ کیلو بایت باشد، k می تواند مقادیر بین 0 تا 800H را اختیار نماید.

- مفهوم دستور: انتقال محتویات یکی از خانه های حافظه داده به یکی از رجیسترهای عمومی (Load Direct from Data Space)
- خلاصه عملیات: Rd<---(k)[۳]
- نوع آدرس دهی: مستقیم
- مثال:

LDS R5,0x60

با اجرای این دستور، محتویات خانه 60H که اولین خانه از فضای SRAM می‌باشد، به رجیستر R5 منتقل می‌شود.

[۳] Rd <---(k) یعنی محتویات خانه‌ای از حافظه با آدرس k به رجیستر Rd منتقل شود.

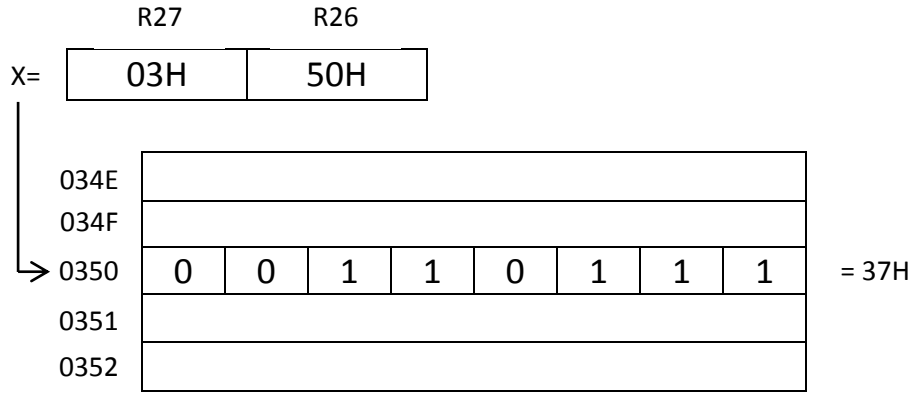
دستور LD

- فرم نوشتن: LD Rd,P[۴]
- بررسی عملوندها: R0<Rd<R31

P یکی از رجیسترهای اشاره‌گر X، Y یا Z می‌باشد. در ادامه در بخش مدهای آدرس دهی، به بررسی این رجیسترها خواهیم پرداخت.

- مفهوم دستور: انتقال محتویات آدرس ذخیره شده در رجیستر اشاره‌گر به یکی از رجیسترهای عمومی (Load Indirect)
- خلاصه عملیات: Rd<---(P)
- نوع آدرس دهی: غیر مستقیم
- مثال: فرض کنیم که خانه 350H از حافظه داده حاوی عدد 37H باشد و رجیستر X نیز حاوی عدد 350H باشد. با نوشتن دستور زیر محتویات خانه 350H از حافظه داده که همان عدد 37H است به رجیستر R16 منتقل می‌شود.

LD R16,X



[۴] این دستور به دو فرم دیگر LD Rd,P+ و LD Rd,-P نیز قابل نوشتن است. در فرم اول، پس از انتقال داده به Rd، رجیستر P یک واحد افزوده می‌شود. به عبارت دیگر، P جدید به خانه بعدی از حافظه داده اشاره خواهد نمود. در فرم دوم، قبل از انجام عملیات انتقال داده، مقدار رجیستر P یک واحد کاهش می‌یابد و سپس محتویات آدرسی که در P ذخیره شده است به Rd منتقل خواهد شد.

* مدهای آدرس دهی *

قبل از اینکه به ادامه دستورات بپردازیم، مفهوم آدرس دهی و انواع آن را بررسی می نماییم.
آدرس دهی، نحوه دسترسی CPU به عملوند مورد نظر را مشخص می‌کند.

انواع آدرس دهی عبارتند از:

- آدرس دهی فوری یا Immediate Addressing
- آدرس دهی مستقیم یا Direct Addressing
- آدرس دهی غیر مستقیم یا Indirect Addressing

– در آدرس دهی فوری، عملوند نوشته شده در دستور، مستقیماً به CPU منتقل می شود. به عنوان نمونه، دستور LDI R16,0x20، عدد ۲۰ در مبنای ۱۶ که معادل ۳۲ در مبنای ۱۰ است را به رجیستر R16 منتقل می کند.

– در دستوراتی که از آدرس دهی مستقیم استفاده می کنند، عملوند، معادل آدرس یکی از خانه های حافظه داده است. CPU با دسترسی به حافظه داده، محتویات آدرس مذکور را به رجیسترهای عمومی یا واحد محاسبه و منطق منتقل می کند و یا محتویات یکی از رجیسترهای عمومی را در آدرس مذکور ذخیره می نماید. برای مثال، دستور LDS یکی از دستوراتیست که از آدرس دهی مستقیم استفاده می کند. فرض کنیم عدد A5 در مبنای ۱۶ در آدرس 250H[۵] از حافظه داده ذخیره شده باشد، پس از نوشتن دستور LDS R10,0x250، عدد A5H به رجیستر R10 منتقل می‌شود.

– اما در آدرس دهی غیر مستقیم، عملوند یکی از رجیسترهای اشاره گر است. رجیسترهای اشاره گر عبارتند از رجیستر X، رجیستر Y و رجیستر Z. هر کدام از این رجیسترها شامل دو رجیستر هشت بیتی هستند که برای نگه داری آدرس‌های ۱۶ بیتی حافظه داده، مورد استفاده قرار می گیرند. مثلاً رجیستر اشاره گر X، از دو رجیستر هشت بیتی به نام‌های XL و XH تشکیل شده است.[۶] اگر بخواهیم یک آدرس ۱۶ بیتی را در این رجیستر ذخیره کنیم، ۸ بیت پرارزش آدرس را در XH و ۸ بیت کم ارزش را در XL قرار می‌دهیم.
بنابراین رجیسترهای اشاره گر در مجموع از شش رجیستر ۸ بیتی تشکیل شده‌اند. این رجیسترها به ترتیب، شش بایت انتهایی از رجیسترهای عمومی، یعنی R26 تا R31 هستند.[۷]
هنگام نوشتن دستوراتی که از این مد آدرس دهی استفاده می‌کنند، ابتداً آدرس داده مورد نظر را در یکی از رجیسترهای اشاره گر قرار می دهیم. سپس به جای اینکه خود آدرس را مستقیماً در دل دستور بنویسیم، از اشاره گر مقدار دهی شده استفاده می کنیم. به عبارت دیگر زمانی که از دستورات با آدرس دهی غیر مستقیم استفاده می کنیم، توسط این رجیسترها به یکی از خانه های حافظه داده اشاره می کنیم.

---

[۵] 250H به معنای ۲۵۰ در مبنای هگزا دسیمال (Hex Decimal) یا همان مبنای ۱۶ است.

[۶] H و L، حروف اول کلمات High و Low هستند.

[۷] توجه داشته باشید که این ۶ رجیستر، علاوه بر اینکه می‌توانند مانند سایر رجیسترهای عمومی برای ذخیره سازی اطلاعات مورد استفاده قرار گیرند، می توانند در نقش اشاره گر برای دستوراتی که از آدرس دهی غیر مستقیم استفاده می‌کنند، ظاهر شوند.

دستور LD، یکی از دستوراتیست که با استفاده از آدرس دهی غیر مستقیم، محتویات حافظه داده را به رجیسترهای عمومی منتقل می کند. فرض کنیم در رجیستر R26، عدد 50H و در رجیستر R27، عدد 02H ذخیره شده باشد. با نوشتن دستور LD R16,X، محتویات خانه 250H از حافظه داده به R16 منتقل می‌شود.

*** ادامه دستورات انتقال داده ***

دستور LDD

- فرم نوشتن: LDD Rd,Y+q یا LDD Rd,Z+q
- بررسی عملوندها: R0<Rd<R31

Y و Z دو رجیستر آخر از رجیسترهای اشاره‌گر هستند.

q یک عدد صحیح بین ۰ تا ۶۳ است.

مفهوم دستور: انتقال محتویات آدرس محاسبه شده به یکی از رجیسترهای عمومی. این آدرس عبارت است از حاصل جمع آدرس ذخیره شده در رجیستر اشاره‌گر به علاوه عددی بین ۰ تا ۶۳

(Load Indirect with Displacement)

- خلاصه عملیات: Rd<---(Y) یا Rd<---(Z)
- نوع آدرس دهی: غیر مستقیم
- توضیحات: هدف از استفاده این دستور پیمایش حافظه داده یا به عبارت دیگر خواندن خانه‌های مختلف آن است بدون اینکه مجبور باشیم برای خواندن این خانه‌ها، آدرس دقیق آنها را در هر عملیات load، ذکر کنیم.

- مثال: در این مثال ابتدا رجیستر Z که عبارت است از رجیسترهای R30 و R31 را مقدار دهی می کنیم. سپس با استفاده از آن، محتویات چندین خانه از حافظه داده را به رجیسترهای عمومی منتقل می کنیم.

توجه داشته باشید که مقداری که در رجیستر Z قرار می گیرد، همان آدرسی از حافظه داده است که می‌خواهیم محتویات درون آن را بخوانیم.

```
LDI R30,0x00
LDI R31,0x04
LDD R0,Z+0
LDD R1,Z+1
LDD R10,Z+5
LDD R16,Z+2
```

پس از اجرای این دستورات محتویات خانه‌های حافظه داده به صورت زیر در رجیسترهای مورد نظر قرار می‌گیرند:

R0 <---(450) R1 <---(451) R10<---(455) R16<---(452)

دستور STS

- فرم نوشتن: STS k,Rr
- بررسی عملوندها: R0<Rd<R31

k یک مقدار عددیست که به یکی از خانه‌های حافظه داده اشاره دارد.

- مفهوم دستور: ذخیره مقدار یکی از رجیسترهای عمومی در یکی از خانه‌های حافظه داده (Store Direct to Data Space)
- خلاصه عملیات: Rr--->(k)
- نوع آدرس دهی: مستقیم
- توضیحات: دستورات Store که همگی با حروف ST آغاز می شوند، بر عکس دستورات Load، مقدار ذخیره شده در یکی از رجیسترهای عمومی را به یکی از خانه‌های حافظه داده منتقل می‌کنند.
- مثال:

```
LDI R10,0b10100100
STS 0x250,R10
```

پس از اجرای این دستورات مقدار A4 در خانه 250H از حافظه داده ذخیره می گردد.

دستور ST

- فرم نوشتن: ST P,Rr[۸]
- بررسی عملوندها: R0<Rd<R31

P یکی از رجیسترهای اشاره‌گر X، Y یا Z می باشد.

- مفهوم دستور: ذخیره مقدار یکی از رجیسترهای عمومی در خانه‌ای از حافظه داده که رجیستر اشاره‌گر به آن اشاره می کند (Store Indirect)
- خلاصه عملیات: Rr--->(P)
- نوع آدرس دهی: غیر مستقیم
- توضیحات: این دستور مشابه دستور LD عمل می کند با این تفاوت که جهت انتقال اطلاعات از رجیسترهای عمومی به حافظه داده است.
- مثال: پس از اجرای دستورات زیر، عدد 95H در خانه300H از حافظه داده ذخیره می گردد.

```
LDI R16,0x95
LDI R28,0x01
LDI R29,0x03
ST –Y,R16
```

[۸] این دستور هم مشابه دستور LD به دو فرم دیگر ST P+,Rr و ST –P,Rr نوشته می شود. برای توضیحات بیشتر به دستور LD مراجعه فرمایید.

دستور STD

- فرم نوشتن: STD Y+q,Rr یا STD Z+q,Rr
- بررسی عملوندها: R0<Rd<R31

Y و Z دو رجیستر آخر از رجیسترهای اشاره‌گر هستند.

q یک عدد صحیح بین ۰ تا ۶۳ است.

مفهوم دستور: ذخیره مقدار یکی از رجیسترهای عمومی در یکی از خانه‌های حافظه داده که آدرس آن عبارت است از حاصل جمع آدرس ذخیره شده در رجیستر اشاره‌گر به علاوه عددی بین ۰ تا ۶۳

(Store Indirect with Displacement)

- خلاصه عملیات: Rr---> (Y+q) یا Rr---> (Z+q)
- نوع آدرس دهی: غیر مستقیم
- مثال:

```
LDI R16,0x66
LDI R17,0x77
LDI R18,0x88
LDI R30,0x00
LDI R31,0x01
STD Z+0,R16
STD Z+1,R17
STD Z+2,R18
```

پس از اجرای این دستورات، خانه‌های 100H، 101H و 102H از حافظه داده به ترتیب مقادیر 66H، 77H و 88H را اختیار می‌کنند.

دستور MOV

- فرم نوشتن: MOV Rd,Rr
- بررسی عملوندها: R0<Rd<R31 R0<Rr<R31
- مفهوم دستور: انتقال محتویات رجیستر Rr به رجیستر Rd
- خلاصه عملیات: Rd<---Rr
- توضیحات: در این دستور، محتوای رجیستر مبدأ تغییری نمی‌کند و فقط مقدار آن در رجیستر مقصد کپی می‌شود.
- مثال: با فرض اینکه مقدار 0XFF در رجیستر R16 ذخیره شده باشد، پس از اجرای دستور زیر، محتویات R17 نیز معادل 0XFF خواهد بود.

```
MOV R17,R16
```

همان طور که می دانید، ۶۴ بایت از خانه های حافظه داده در ATMega32 به رجیسترهای I/O اختصاص دارند. تقریباً تمامی تنظیمات مربوط به واحدهای مختلف AVR، با استفاده از این رجیسترها انجام می شود. به همین دلیل این رجیسترها، در هنگام برنامه نویسی، مکرراً مورد استفاده قرار می گیرند.[۹] برای سهولت در برنامه نویسی، دو دستور IN و OUT مخصوص کار با رجیسترهای I/O در نظر گرفته شده است که دسترسی به آنها را در زمان نوشتن برنامه آسان‌تر می کند. شیوه استفاده از این دستورات به صورت زیر می باشد:

دستور IN

- فرم نوشتن: IN Rd,A
- بررسی عملوندها: R0<Rd<R31
  A آدرس یکی از رجیسترهای I/O است که می‌تواند عددی بین ۰ تا ۶۳ باشد.
- مفهوم دستور: انتقال محتویات یکی از رجیسترهای I/O به یکی از رجیسترهای عمومی
- خلاصه عملیات: Rd<---(A)
- توضیحات: آدرس رجیسترهای I/O در این دستور نسبی هستند یا به عبارت دیگر نسبت به اولین خانه از فضای I/O آدرس دهی می شوند. به عنوان مثال، اولین رجیستر از فضای I/O با آدرس 0 شناخته می شود، حال آنکه آدرس واقعی این رجیستر در حافظه داده، 0x20 است که دقیقاً پشت سر رجیستر R31 با آدرس 0x1F قرار می‌گیرد.
- مثال:

```
IN R16,0x00
IN R17,0x3F
```

با اجرای این دستورات، محتویات اولین خانه از رجیسترهای I/O که مربوط به رجیستر TWBR است به رجیستر R16 و محتویات شصت و چهارمین یا آخرین خانه از این رجیسترها که به مربوط به رجیستر SREG می‌باشد، به R17 انتقال پیدا می کند.

روشی دیگری که برای دسترسی به رجیسترهای I/O می توان ذکر نمود، استفاده از دستورات Load و Store است. در این روش با استفاده از آدرس واقعی رجیسترهای I/O، عملیات خواندن و نوشتن، پیاده سازی می‌شوند. البته اجرای هر یک از این دستورات، نیازمند دو سیکل ساعت است که این زمان برای دستورات IN و OUT معادل یک سیکل می‌باشد.

مثال: دستورات زیر مشابه مثال قبل، محتوای رجیسترهای اول و آخر فضای I/O را به R16 و R17 منتقل می‌نمایند.

```
LDS R16,0x20
LDS R17,0x5F
```

[۹] برای مشاهده لیست کامل رجیسترهای I/O در میکروکنترلر ATMega32، به پیوست الف مراجعه فرمایید.

دستور OUT

- فرم نوشتن: OUT A,Rd
- بررسی عملوندها: R0<Rd<R31

A آدرس یکی از رجیسترهای I/O است که می‌تواند عددی بین ۰ تا ۶۳ باشد.

- مفهوم دستور: انتقال محتویات یکی از ۳۱ رجیستر عمومی به یکی از رجیسترهای I/O
- خلاصه عملیات: Rd--->(A)
- توضیحات: این دستور دقیقاً مشابه دستور IN عمل می نماید با این تفاوت که انتقال اطلاعات، از رجیسترهای عمومی به رجیسترهای I/O می باشد.

دستور راهنمای EQU.

برخی از دستورات در زبان اسمبلی هیچ گاه به کد ماشین تبدیل نشده و در حافظه برنامه ذخیره نمی‌شوند. این دستورات که اصطلاحاً دستورات راهنما[۱۰] نام دارند، وظیفه راهنمایی اسمبلر[۱۱] به انجام یک کار به خصوص را به عهده دارند.

همان طور که در مثال‌های بالا مشاهده کردید، استفاده از دستورات IN و OUT با نوشتن آدرس یکی از ۶۴ رجیستر فضای I/O همراه است. این موضوع با غرض اصلی دستورات IN و OUT که ایجاد سهولت در برنامه نویسی بود در تناقض است چرا که برنامه نویس برای خواندن یا نوشتن در رجیسترهای مورد نظر خود، باید دائماً به برگه اطلاعاتی مربوط به رجیسترهای I/O مراجعه کرده و آدرس رجیستر مورد نیاز خود را استخراج نماید. دستور راهنمای EQU. برای جلوگیری از همین مسئله تعبیه شده است. با استفاده از این دستور می توان نامی برای هر یک از رجیسترهای فضای I/O در نظر گرفت و پس از آن با استفاده از این نام‌ها به رجیسترها دسترسی پیدا کرد. برای آشنایی با نحوه استفاده از این دستور، به مثال زیر توجه فرمایید.

مثال: در این مثال، ابتداً نامی برای رجیستر PORTA با آدرس 1B در نظر گرفته شده و سپس با استفاده از دستور OUT، مقدار 55H به این رجیستر منتقل شده است.

```
.EQU PORTA=1B
LDI R16,0x55
OUT PORTA,R16
```

در این مثال نام مورد نظر برای رجیستر PORTA خود عبارت PORTA انتخاب شده است. گر چه در انتخاب نام برای رجیسترها آزاد هستیم و لزومی به انتخاب این نام نداریم، اما با این کار، برای استفاده از رجیسترهای I/O در زمان برنامه نویسی، عیناً از نام اصلی آنها استفاده خواهیم نمود.

در انتهای این فصل به برخی از دستورات راهنای پرکاربرد اشاره خواهیم نمود.

---

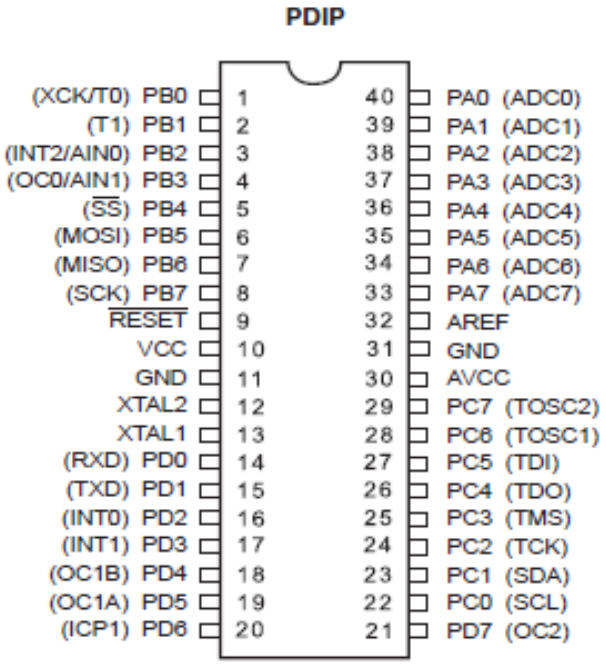
[۱۰] Directive

[۱۱] اسمبلر مترجمیست که برنامه‌های نوشته شده به زبان اسمبلی را به کدهای زبان ماشین تبدیل می نماید.

* پورت‌ها و تنظیمات اولیه *

پورت در اصطلاح به معنای گذرگاه ورودی یا خروجی است و در میکرو کنترلر به درگاه ورودی و خروجی اطلاعات اطلاق می شود به این معنا که توسط پورت‌ها می توانیم اطلاعاتی را از میکرو کنترلر به بیرون ارسال نماییم یا از خارج میکرو کنترلر، اطلاعاتی را دریافت کرده و درون رجیسترهای داخلی میکرو ذخیره نماییم.

میکرو کنترلر ATMega32 به ۴ پورت به نام های A، B، C و D مجهز می باشد. هر کدام ار این پورت ها دارای ۸ پایه مخصوص به خود هستند که جمعاً ۳۲ پایه را به خود اختصاص می دهند. در شکل زیر مکان و نام پایه‌های مربوط به پورت‌ها مشخص شده است.[۱۲]



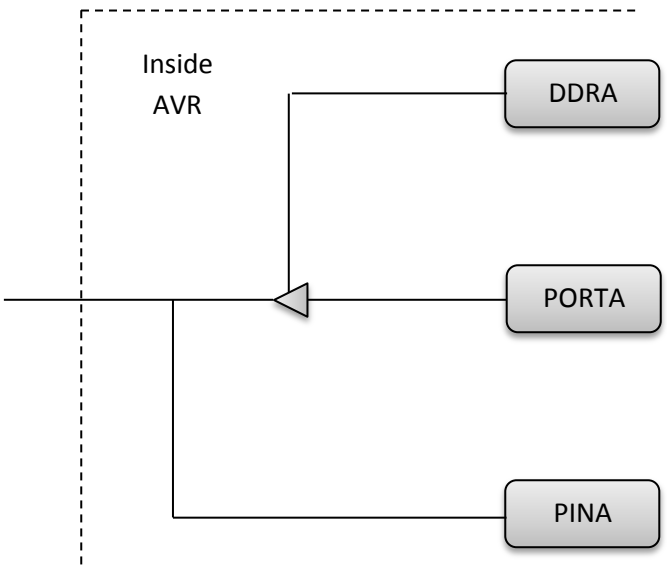
نمای کلی میکرو کنترلر ATMega32

برای آشنایی با نحوه کار با پورت ها ابتدا به بررسی ساختان داخلی آنها می پردازیم.

هر یک از پورت‌های ۴ گانه در ATMega32 دارای سه رجیستر ۸ بیتی با نام های DDR، PORT و PIN هستند. برای مثال سه رجیستر DDRA، PORTA و PINA رجیسترهای تعبیه شده برای پورت A در ATMega32 می‌باشند. این رجیسترها، عضوی از رجیسترهای ۶۴ گانه فضای I/O می باشند.

[۱۲] در واقع از لحاظ مفهومی تمامی پایه‌های میکرو کنترلر که امکان ورود یا خروج ولتاژ (۰ و ۱ منطقی) را برای آن فراهم می کنند، نوعی پورت محسوب می شوند. اما از لحاظ اصطلاحی، پورت ها در میکروکنترلر به پایه های مخصوصی گفته می شود که با استفاده از تنظیمات اولیه، می توانیم آنها را به صورت ورودی یا خروجی تعریف نماییم و پس از آن از طریق آنها اقدام به دیافت یا ارسال اطلاعات کنیم.

شکل زیر نمایش دهنده بخشی از ساختار درونی یک پورت و نحوه اتصال آن به رجیسترهای مربوطه است.[۱۳]



بخشی از ساختار درونی یک پورت

توجه داشته باشید که هر یک از رجیسترهای نمایش داده شده در شکل، تنها بیانگر یک بیت از یک رجیستر کامل هستند. همان طور که مشاهده می کنید، رجیستر DDR[۱۴]، تعیین کننده جهت ورود و خروج داده است. با یک بودن مقدار این رجیستر، بافر متصل به پورت، وصل شده و اطلاعات از رجیستر PORT به پایه انتقال پیدا می کند. همچنین با صفر شدن مقدار این رجیستر، بافر در حالت قطع قرار می گیرد و هیچ داده ای از رجیستر پورت اجازه بیرون رفتن پیدا نمی کند. در این حالت، اگر ولتاژی ار بیرون روی پایه میکرو قرار گیرد، مسقیماً وارد رجیستر پین می شود.

به این نکته توجه داشته باشید که رجیستر DDR، تأثیری بر روی جهت انتقال اطلاعات به PIN ندارد و محتوای رجیستر PIN همواره معادل با مقدار ولتاژی است که روی پایه میکرو قرار دارد. حتی در حالتی که مقدار DDR برای یک پایه ۱ منطقی باشد یا به عبارت دیگر، پورت در حالت خروجی قرار داشته باشد، محتوای رجیستر PIN، معادل اطلاعات قرار گرفته روی پایه توسط رجیستر PORT خواهد بود.

---

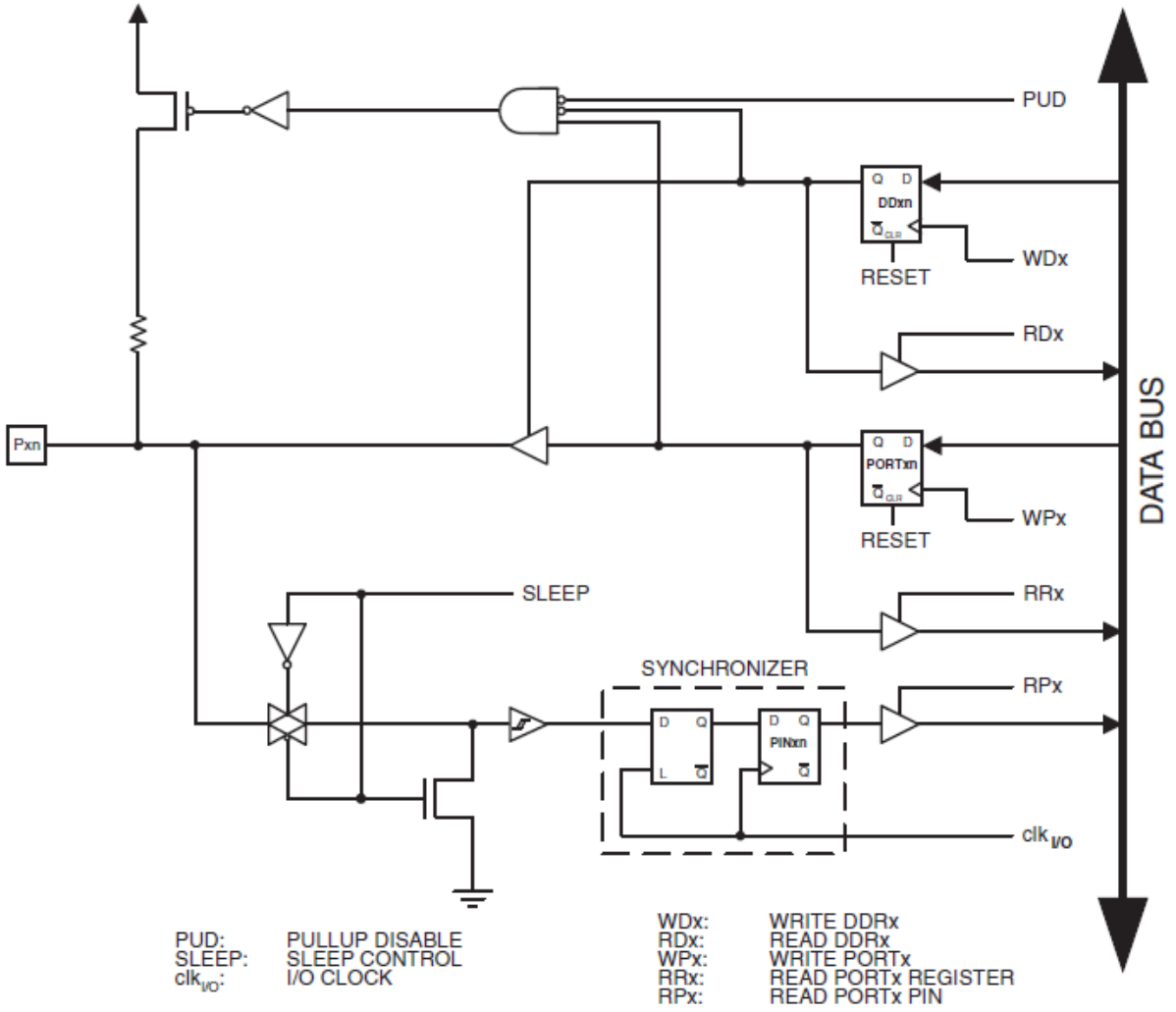
[۱۳] در هنگام استفاده از کلمه پورت، گاه منظور گذرگاه اطلاعات یا همان پایه های در نظر گرفته شده برای ورود و خروج اطلاعات است و گاه منظور، رجیستر ۸ بیتی PORT است.

همچنین واژه پین، گاه به معنای پایه و گاه به معنای رجیستر ۸ بیتی PIN مورد استفاده قرار می گیرد.

معمولاً هنگام استفاده از واژه های پورت و پین به معنای رجیسترهای PORT و PIN، واژه رجیستر هم به همراه آنها به کار برده می شود. مثلاً می گوییم خواندن از رجیستر PIN

[۱۴] Data Direction یا جهت داده

شکل زیر، نمایش دهنده ساختان کامل یک پورت در ATMega32 است.



همان طور که مشاهده می فرمایید سه بیت از رجیسترهای DDR، PORT و PIN با نام های DDxn، PORTxn و PINxn به ترتیب از بالا به پایین قرار گرفته اند.[۱۵] دو رجیستر DDRx و PORTx دارای دو پایه با نام های WDx و WPx برای نوشته شدن اطلاعات در آنها هستند. با فعال شدن هر یک از این پایه‌ها، اطلاعات روی گذرگاه داده[۱۶] به رجیستر مربوطه انتقال می‌یابد. هنگام نوشته شدن دستور OUT در برنامه نویسی، همین عملیات پیاده سازی می شود.

مثال: در قطعه کد زیر با اجرای دستور خط دوم، اطلاعات رجیستر R16 روی باس قرار گرفته و سپس پایه WDA که مربوط به نوشتن در رجیستر DDRA می باشد فعال می شود. بدین ترتیب محتویات R16 به DDRA انتقال می یابد.

```
LDI R16,0xFF
OUT DDRA,R16
```

[۱۵] n در انتهای نام گذاری، به معنای بیت شماره n و x به معنای یکی از حروف A، B، C یا D می‌باشد.

[۱۶] DATA BUS

علاوه بر دو پایه مذکور، سه پایه دیگر با نام های RDx، RRx و RPx به ترتیب مربوط به خواندن اطلاعات از رجیسترهای DDRx، PORTx و PINx می باشند. با فعال شدن هر یک از این پایه ها بافر متصل به رجیستر مربوطه در حالت وصل قرار گرفته و اطلاعات از داخل رجیستر به سمت باس هدایت می شوند. پیاده سازی این عملیات توسط دستور IN انجام می گیرد.

مثال:

```
LDI R16,0x55
OUT PORTA,R16
IN R17, PORTA
IN R18, PINA
```

در این برنامه فرض بر این است که پورت A به صورت ورودی تعریف شده است. همان طور که می دانیم در زمان تعریف یک پورت به صورت ورودی، محتویات رجیستر PORT اجازه قرار گرفتن روی پایه های میکرو را ندارد. در ابتدا با اجرای خطوط اول و دوم، عدد 55H به رجیستر PORTA منتقل شده و سپس با اجرای دستورات IN، پایه های RRA و RPA به ترتیب فعال شده و محتویات رجیسترهای PORTA و PINA به R17 و R18 منتقل می شوند. پس از اتمام برنامه، مقدار رجیستر R17، عدد 55H و مقدار رجیستر R18 معادل ولتاژ خارجی قرار گرفته روی پایه‌های پورت A می باشد.

توجه داشته باشید که پایه های WDx، WPx، RDx، RRx و RPx میان تمام پایه های یک پورت مشترک می‌باشند. به عبارت دیگر هر یک از پورت های ۴ گانه، دارای دو پایه برای نوشتن در رجیسترهای DDR و PORT و سه پایه برای خواندن مقادیر رجیسترهای DDR، PORT و PIN مربوط به خود می باشند. به عنوان مثال پایه WPD مخصوص نوشتن اطلاعات در رجیستر PORTD است و به تمامی بیت های این رجیستر متصل می باشد.

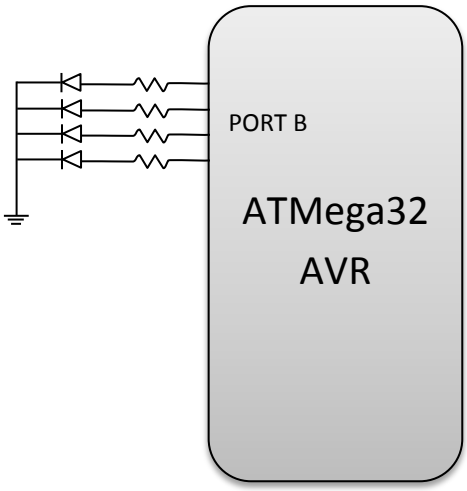
توضیحات بیشتر در مورد ساختار پورت ها به مرور در طول مباحث این بخش بیان خواهد شد. در ادامه با ذکر چند مثال به نحوه برنامه ریزی پورت ها در AVR خواهیم پرداخت.

مثال ۱: فرض کنید قصد داریم محتویات رجیستر R16 را روی پایه های مربوط به پورت A قرار دهیم. از آنجایی که قصد داریم تمامی بیت های R16 را به خروجی منتقل کنیم، به تمام ۸ پایه پورت A احتیاج داریم. بنابراین ابتداً باید پورت A را به صورت خروجی تعریف نماییم. برای این کار کافیست محتویات رجیستر DDRA را برابر عدد 11111111 در مبنای ۲ قرار دهیم. بدین ترتیب تمامی بافرهای متصل به رجیستر پورت A در حالت فعال قرار گرفته و محتوای پورت به پایه های PA0 تا PA7 منتقل می شوند.

```
LDI R17,0xFF
OUT DDRA,R17
OUT PORTA,R16
```

پس از اجرای این دستورات، محتویات رجیستر PINA نیز معادل محتویات رجیستر R16 خواهد بود.

مثال ۲: فرض کنید مطابق شکل، ۴ پایه اول از پورت B را به ۴ دیود نوری متصل کرده باشیم.



حال می خواهیم این دیودها را به ترتیب روشن و خاموش نماییم به طوری که در مرحله اول، هر ۴ دیود خاموش باشند. در مرحله بعد دیود اول روشن و باقی دیودها خاموش باشند. در مرحله بعدی دیود دوم روشن و باقی دیودها خاموش باشند و این روند همین طور تا رسیدن به مرحله اول ادامه بیاید.

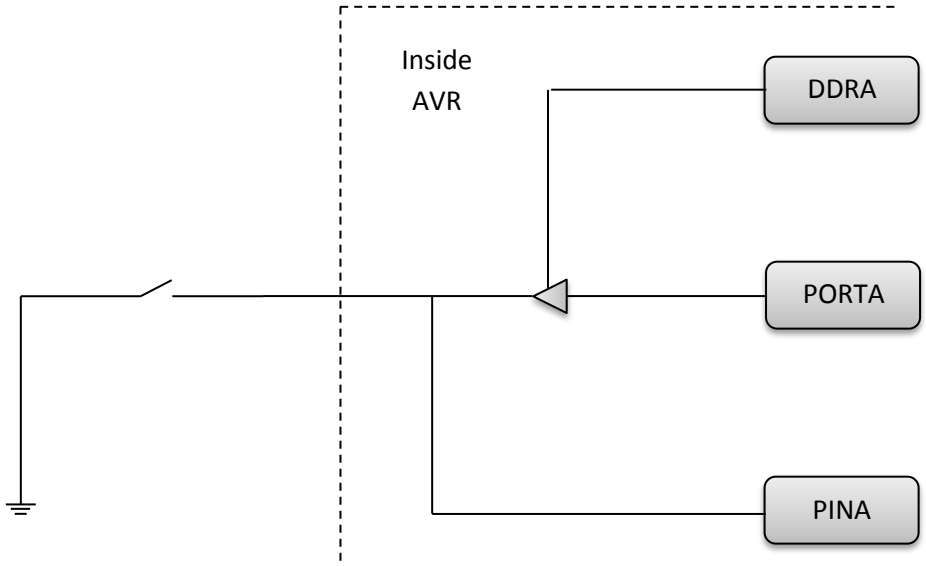
برای این منظور ابتداً ۴ بیت اول از پورت B را به صورت خروجی معرفی می نماییم، سپس اعداد باینری 0000، 0001، 0010، 0100، 1000 و 0000 را به ترتیب در ۴ بیت اول از رجیستر پورت B قرار می دهیم تا خروجی‌های مورد نظر به همان ترتیبی که می خواهیم روی پایه های پورت B ظاهر شوند.

قطعه برنامه زیر این عملیات را پیاده سازی می نماید.[۱۷]

```
LDI R16,0x0F
OUT DDRB,R16
LDI R16,0x00
OUT PORTB,R16
LDI R16,0x01
OUT PORTB,R16
LDI R16,0x02
OUT PORTB,R16
LDI R16,0x04
OUT PORTB,R16
LDI R16,0x08
OUT PORTB,R16
LDI R16,0x00
OUT PORTB,R16
```

[۱۷] توجه فرمایید که این برنامه حالت آموزشی دارد و در عمل سرعت اجرای این دستورات به قدری بالاست که تمام مراحل در کسری از ثانیه انجام می‌گیرند و چشم ما قابلیت تشخیص این مراحل را نخواهد داشت. در فصل ۵ به نمونه‌ای عملیاتی از این مثال خواهیم پرداخت.

مثال ۳: فرض کنید یک کلید مطابق شکل زیر به پایه آخر از پورت C متصل شده است.



می‌خواهیم وضعیت قطع یا وصل بودن این کلید را بررسی نماییم. برای این منظور کافیست که پایه شماره ۷ از پورتC را به صورت ورودی تعریف نماییم و سپس مقدار رجیستر PINC را بخوانیم. محتوای بیت شماره ۷ از این رجیستر، تعیین کننده وضعیت کلید خواهد بود. همان طور که در شکل پیداست در صورت وصل بودن کلید، ولتاژ زمین به پایه یا پین آخر از رجیستر پورت C منتقل می شود.

با استفاده از دستورات زیر می توان این عملیات را انجام داد.

```
LDI R16,0b01111111
OUT DDRC,R16
IN R16,PINC
```

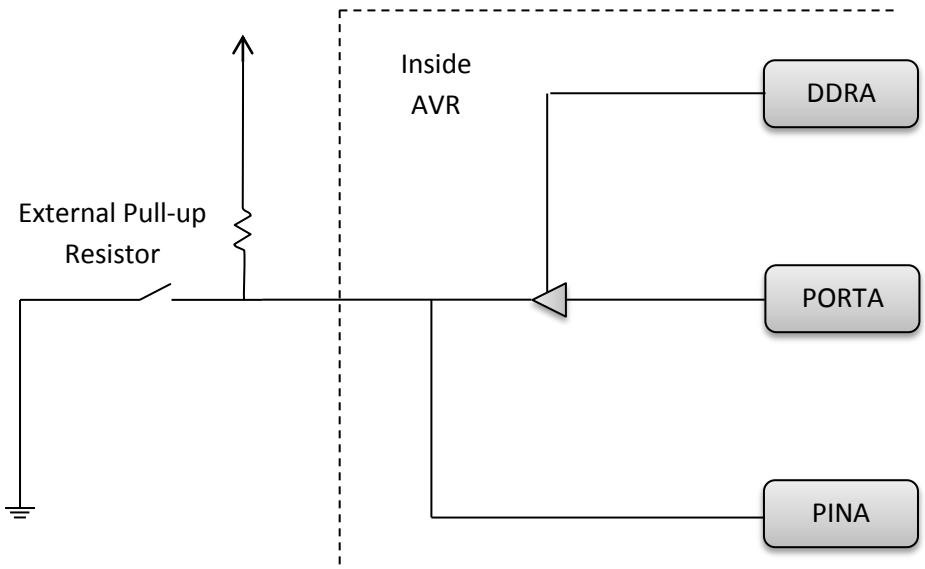
همان طور که مشاهده می کنید در دو خط اول برنامه، پین آخر از پورت C در حالت ورودی و باقی پین ها در حالت خروجی قرار گرفته اند. البته لزومی به خروجی کردن باقی پین ها در این مثال نیست ولی برای متمایز شدن بیت آخر رجیستر DDRC، باقی بیت ها را معادل ۱ منطقی قرار دادیم.

نکته مهمی که در این مثال باید به آن توجه نمود، مقدار ولتاژ روی پایه متصل به کلید در حالت باز بودن آن است. در این حالت از طرفی پایه شماره ۷ به صورت ورودی تعریف شده و از درون میکرو مقدار دهی نمی شود و از طرف دیگر سر دیگر پایه آزاد بوده و به جایی متصل نمی باشد. در این حالت اصطلاحاً پایه در حالت شناور یا float قرار دارد و مقدار ولتاژ روی آن معلوم نیست.[۱۸] نویزهای محیطی یا حتی نویز پایه های مجاور، می تواند روی ولتاژ این پایه اثر گذار باشد. بنابراین در صورت باز بودن کلید، مقدار ولتاژ روی پایه نامعتبر خواهد بود و ما را در تشخیص وضعیت کلید دچار خطا خواهد نمود.

[۱۸] High Impedance یا Z اصطلاحات دیگری است که برای این وضعیت به کار برده می شود.

برای جلوگیری از رخ دادن این حالت، می توانیم با استفاده از یک مقاومت بالاکش یا pull-up، پایه مورد نظر را به Vcc متصل نماییم.

شکل زیر نمایش دهنده چگونگی اتصال کلید به میکرو با در نظر گرفتن یک مقاومت pull-up است.



بدین ترتیب در زمان قطع بودن کلید، مقدار ۱ منطقی و در زمان وصل بودن آن، مقدار ۰ منطقی روی پایه قرار خواهد گرفت چرا که مقاومت بین پایه و Vcc، مانع از قرار گرفتن ولتاژ منبع روی پایه می شود.

روش صحیح دیگری نیز برای اتصال کلید به میکرو وجود دارد که عبارت است از متصل کردن کلید به Vcc و اتصال پایه میکرو به زمین با استفاده از یک مقاومت pull-down. شکل زیر این نحوه اتصال را نمایش می دهد.