دستورات و برنامه ریزی avr به زبان اسمبلی :

راهنمای کامپایلر :

این دستورات برای کامپایلر شناخته شده میباشند . دستور ات راهنمای کامپایلر معنی های خاصی را برای کامپایلر تعریف میکنندو قبل از آنها نقطه گذاشته میشود. این دستورات عبارتند از :

راهنمای کامپایلر #DEF :این دستور یک دستور راهنمای کامپایلر میباشد . بوسیله این دستور میتوان برای ثبات ها اسامی معنی داری تعریف کرد : .DEF نام=Rx

.DEF TEMP=R16

در زبان اسمبلی بهتر است ثبات ها را به نام مستقیمشان بکار نبرید و از اسامی معنی دار استفاده کنید.

راهنمای کامپایلر INCLUDE. : این راهنمای کامپایلر بای وارد کردن یک فایل خارجی تعریف شده در برنامه بکار میرود . این دستور یک دستور سر تیتری میباشد و در ابتدای برنامه بکار میرود .پسوند فایل و محل ذخیره آن در فایل inc میباشد . .INCLUDE "نام فایل"

برای مثال ثبات های همه منظوره و ثباتهای I/O دارای آدرس میباشند آدرس این ثبات ها در فایل inc. میباشد و باید در ابتدای برنامه ضمیمه شود.

.INCLUDE "M32.INC"

دستور ات دسترسی به ثبات های avr :

میکروکنترلر avr دارای 32ث همه منظوره و 64 ثبات I/O میباشد . 32 ثبات همه منظوره را با نامهای R0 تا R31 میخوانند .

برای دسترسی به ثبات ها از دستورات زیر استفاده میکنیم:

Ldi:برای بار کردن مستقیم عدد ثابت 8 بیتی در یک ثبات همه منظوره استفاده میشود.

Ldi Rd,K $16 \le d \le 31$, $0 \le K \le 255$

Mov : بوسیله این دستور مقدار محتوای یک ثبات را به ثبات دیگر کپی میکنیم .

Mov Rd,Rr $0 \le d \le 31$, $0 \le R \le 31$

Movw(copy register word):بوسیله این دستور یک جفت ثبات را در جفت ثبات دیگربه عنوان کلمه(16 بیتی) کپی میکنیم.

Movw Rd+1:Rd,Rr+1:Rr d,r€{0,2,4,...,30}

مثال : Movw R26:25,R1:0

Movw Zh:Zl,Xh:Xl

Movw R17:R16,R0:R1

ثبات های اشاره گر : در avr سه ثبات اشاره گر 16 بیتی موجود است که جفت ثبات های r31:r30(z)-r29:r28(y)-r27:r26(x) تشکیل شده اند. این ثبات ها میتوانند ZH:ZL-YH:YL-XH:XL خوانده شوند.

```
.equ address=ram end
Ldi Yh,high(address)
Ldi Yl,low(address)
```

دستور ات دسترسی به ثباتهای I/O :

In : با این دستور میتوان داده را از فضای I/Oدر ثبات مقصد بار کرد .

In Rd,A $0 \le d \le 31$, $0 \le A \le 63$

In R16,MCUCR

Out : داده را از ثبات مبدا در فضایI/O ذخیره میکند.

Out A,Rr $0 \le r \le 31$, $0 \le A \le 63$

حافظه میکروکنترلر avr :

همانگونه که در بخش قبل گفته شد سازماندهی حافظه ها بصورت زیر میباشد :

الف) حافظه برنامه:

ب) حافظه داده SRAM

ج) حافظه داده EEPROM

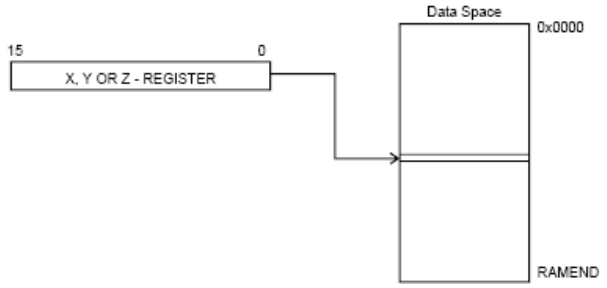
دستور ات کار با حافظه (دسترسی به حافظه)

دستور LD: Load Indirect from Data Space to Register using Index X

بوسیله این دستور میتوان یک بایت را از فضای داده و از آدر س موجود در اشاره گر X بطور غیر مستقیم به درون یک ثبات بارگذاری کرد . برای تراشه های دارای SRAM فضای داده شامل SRAM داخلی و خارجی(در صورت وجود) – فایل ثبات های همه منظوره و فضای ثباتهای I/O میباشد . این دستور برای حافظه داده eeprom نمی باشد .

میتوان به سه شکل زیر از این دستور استفاده کرد :

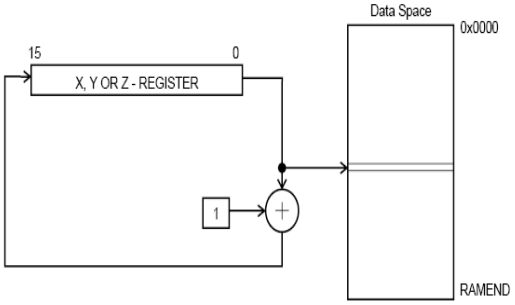
1. محتوای آدرس X را خوانده و در Rd قرار میدهد. مقدار X پس از بار گذاری در ثبات Rd بدون تغییر باقی میماند.



LD Rd, X $0 \le d \le 31$

Rd ← (X)

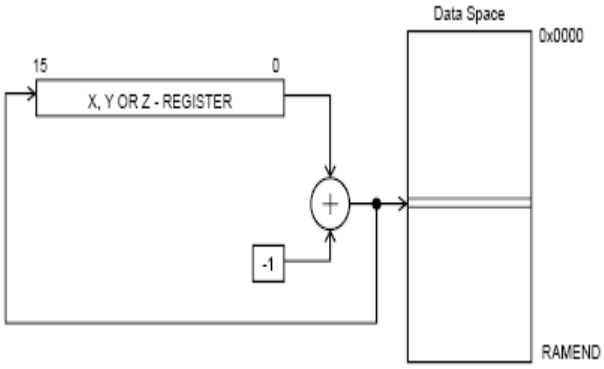
2. محتـوای آدرس X از حافظـه را وانـده و در ثبـات Rdبـار میکنـد سـپس X را یـک واحـد افزایش میدهد.



$Rd \leftarrow (X)$ $X \leftarrow X + 1$

LD Rd, X+ $0 \le d \le 31$

3. ابتدا آدرس X را یک واحد کاهش داده سپس محتـوای موجـود در حافظـه داده در آدرس X-1 را خوانده و در ثبات Rd بار میکند.



$X \leftarrow X - 1$ $Rd \leftarrow (X)$

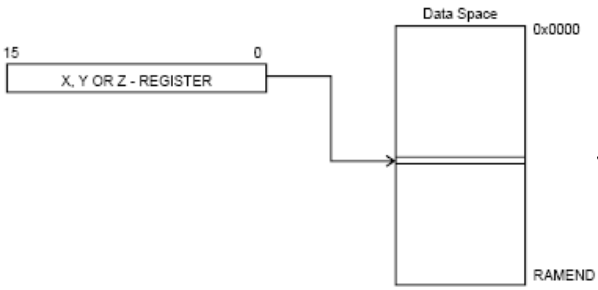
LD Rd, -X $0 \le d \le 31$

دستور LD(LDD):**load indirect from data space to register using index y(z)**

بوسیله این دستور میتوان یک بایت را از فـضای داده و از آدر س موجـود در اشـاره گـر Y(Z) بطور غیر مستقیم به درون یک ثبات بارگذاری کرد . برای تراشه های دارای SRAM فـضای داده شامل SRAM داخلی و خارجی(در صورت وجود) – فایل ثبات های همه منظوره و فـضای ثباتهای I/O میباشد . این دستور برای حافظه داده eeprom نمی باشد .

میتوان به سه شکل زیر از این دستور استفاده کرد :

1. محتوای آدرس Y(Z) را خوانده و در Rd قرار میدهد. مقدار Y(Z) پس از بار گذاری در ثبات Rd بدون تغییر باقی میماند.



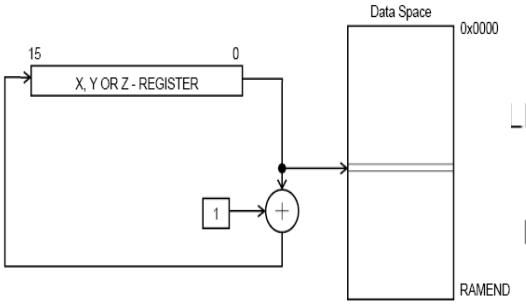
$Rd \leftarrow (Y)$

_D Rd, Y $0 \le d \le 31$

$Rd \leftarrow (Z)$

LD Rd, Z $\quad 0 \le d \le 31$

2. محتوای آدرس Y(Z) از حافظه را وانده و در ثبات Rdبار میکند سپس Y(Z) را یک واحد افزایش میدهد.



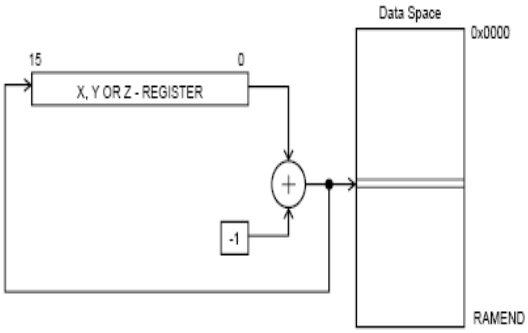
$Rd \leftarrow (Y)$ $\quad Y \leftarrow Y + 1$

LD Rd, Y+ $\quad 0 \le d \le 31$

$Rd \leftarrow (Z)$ $\quad Z \leftarrow Z + 1$

LD Rd, Z+ $\quad 0 \le d \le 31$

3. ابتدا آدرس Y(Z) را یک واحد کاهش داده سپس محتوای موجود در حافظه داده در آدرس Y-1(Z-1) را خوانده و در ثبات Rd بار میکند.



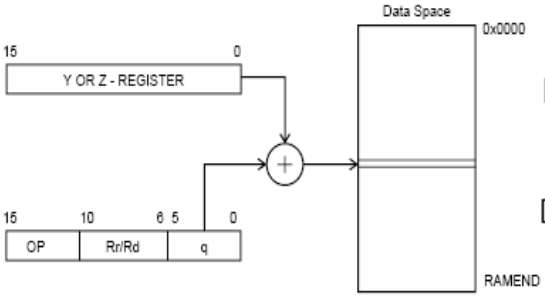
$Y \leftarrow Y - 1$ $\quad Rd \leftarrow (Y)$

LD Rd, -Y $\quad 0 \le d \le 31$

$Z \leftarrow Z - 1$ $\quad Rd \leftarrow (Z)$

LD Rd, -Z $\quad 0 \le d \le 31$

4. محتوای حافظه با آدرس Y+q(Z+q) را در ثبات Rd بار میکند.



$Rd \leftarrow (Y+q)$

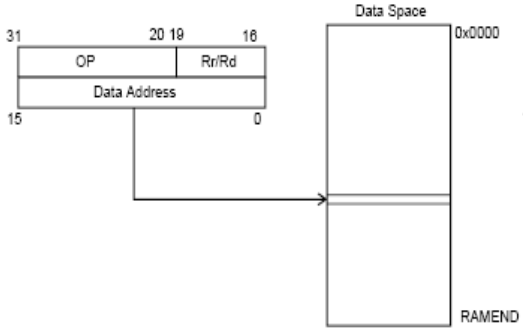
D Rd, Y+q $\quad 0 \le d \le 31,\ 0 \le q \le 63$

$Rd \leftarrow (Z+q)$

D Rd, Z+q $\quad 0 \le d \le 31,\ 0 \le q \le 63$

دستور STS :

محتوای ثبات Rr را در حافظه با آدرس Kبصورت مستقیم ذخیره میکند . که بصورت زیر استفاده میشود:



TS k,Rr $0 \le r \le 31, 0 \le k \le 65535$

$(k) \leftarrow Rr$

Eeprom فضای آدر جدا گانه دارد .

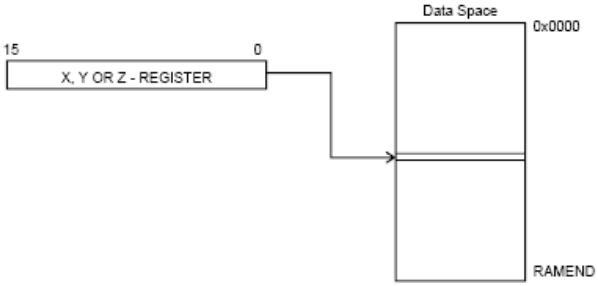
دستور ST: **Store indirect from register to data space using index x**

یک بایت را در آدر س X از حافظه داده (SRAM داخلی و خارجی در صورت وجود) ذخیره میکند.

Eeprom فضای آدرس جداگانه ای دارد.

این دستور بصورتهای زیر استفاده میشود:

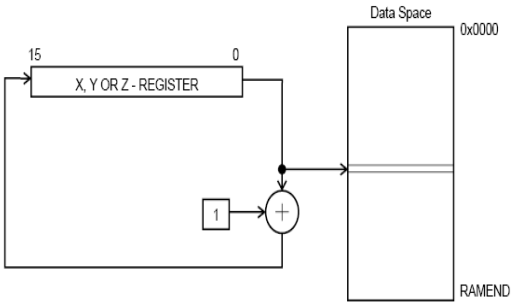
1. محتوای Rr را در حافظه داده با آدرس X ذخیره میکند



$(X) \leftarrow Rr$

ST X, Rr $0 \le r \le 31$

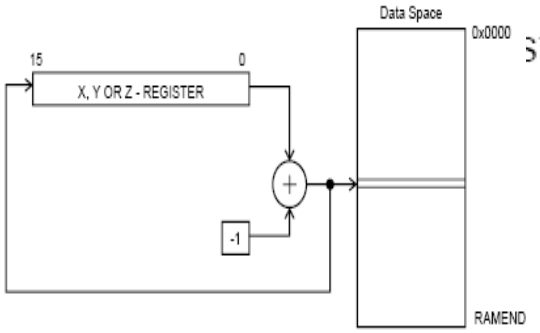
2. محتوای Rr را در حافظه داده در آدرس X ذخیره کرده و سپس X را افزایش میدهد.



$(X) \leftarrow Rr$ $X \leftarrow X+1$

ST X+, Rr $0 \le r \le 31$

3. ابتدا X را کاهش داده و سپس محتوای Rr را در حافظه با آدرس X-1 ذخیره میکند.



$X \leftarrow X - 1$ $\quad$ $(X) \leftarrow Rr$

ST -X, Rr $\quad$ $0 \le r \le 31$

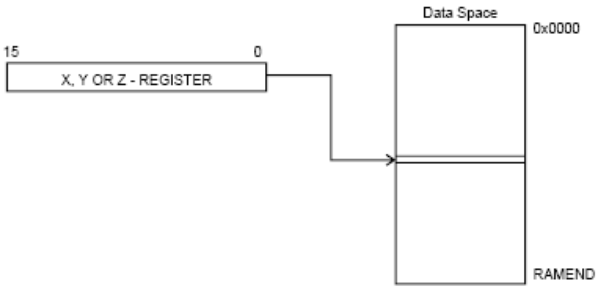
دستور ST(STD) : **Store indirect from register to data space using index y(z)**

یک بایت را در آدر س Y(Z) از حافظه داده (SRAM داخلی و خارجی در صورت وجود) ذخیره میکند .

Eeprom فضای آدرس جداگانه ای دارد.

این دستور بصورتهای زیر استفاده میشود:

1. محتوای Rr را در حافظه داده با آدرس Y(Z) ذخیره میکند.



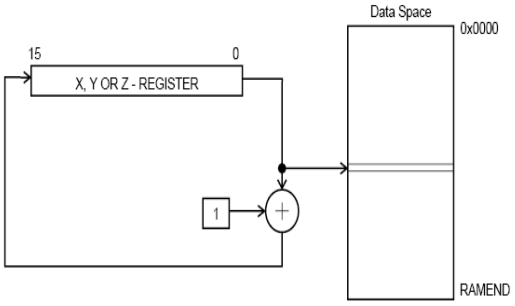
$(Y) \leftarrow Rr$

ST Y, Rr $\quad$ $0 \le r \le 31$

$(Z) \leftarrow Rr$

ST Z, Rr $\quad$ $0 \le r \le 31$

2. محتوای Rr را در حافظه داده در آدرس Y(Z) ذخیره کرده و سپس Y(Z) را افزایش میدهد.



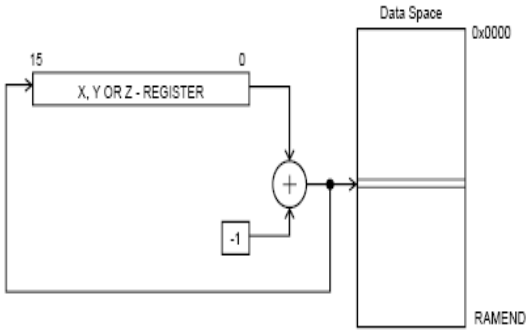
$(Y) \leftarrow Rr$ $\quad$ $Y \leftarrow Y+1$

ST Y+, Rr $\quad$ $0 \le r \le 31$

$(Z) \leftarrow Rr$ $\quad$ $Z \leftarrow Z+1$

ST Z+, Rr $\quad$ $0 \le r \le 31$

3. ابتدا Y(Z) را کاهش داده و سپس محتوای Rr را در حافظه با آدرس Y-1(Z-1) ذخیره میکند.



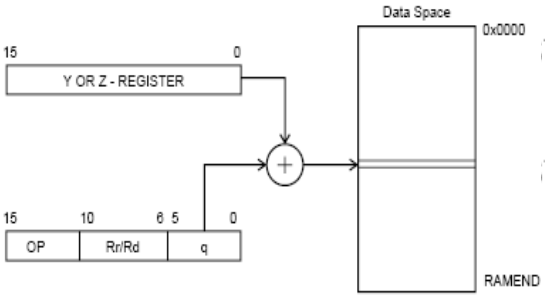
$Y \leftarrow Y - 1$ $\quad$ $(Y) \leftarrow Rr$

ST -Y, Rr $\quad$ $0 \le r \le 31$

$Z \leftarrow Z - 1$ $\quad$ $(Z) \leftarrow Rr$

ST -Z, Rr $\quad$ $0 \le r \le 31$

4. محتوای Rr را در حافظه با آدرس Y+q(Z+q) ذخیره میکند.



$(Y+q) \leftarrow Rr$

TD Y+q, Rr $\quad$ $0 \le r \le 31, 0 \le q \le 63$

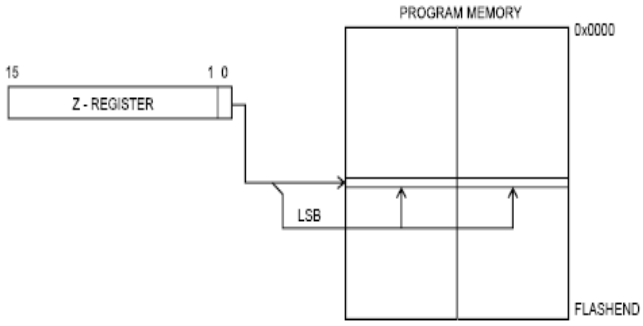
$(Z+q) \leftarrow Rr$

TD Z+q, Rr $\quad$ $0 \le r \le 31, 0 \le q \le 63$

دستور LPM: load program memory

با این دستور آدرس Z از حافظه برنامه FLASH را خوانده و در ثبات R0 قرار میدهد.فقط این دستور برای دستیابی به خواندن در محل ذخیره برنامه وجود دارد که برای جفت اشاره گر Z تعریف شده است .ح برنامه بطور کلمه به کلمه سازماندهی شده است (یک دستور در یک آدرس 16 بیتی یا دو بایتی یا یک کلمه قرار میگیرد . کم ارزش ترین بیت یعنی ZL.0 بایت پایین یا بالا را انتخاب میکند .(بایت پایین =0و بایت بالا=1) این آدرس ابتدایی باید در 2 ضرب شود.

با توجه به 16 بیتی بودن Z این دستور توانایی آدرس دهی 64K ابتدایی فضای حافظه برنامه را دارد .



این دستور را میتوان به شکلهای زیر بکار برد .

1. محتوای حافظه برنامه در آدرس موجود در Z را در R0 ذخیره میکند.

R0 ← (Z)

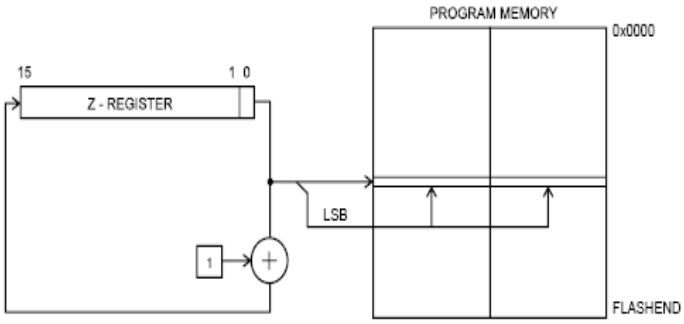
LPM None, R0 implied

2. محتوای حافظه برنامه با آدرس موجود در Z را در Rd ذخیره میکند .

Rd ← (Z)

LPM Rd, Z 0 ≤ d ≤ 31

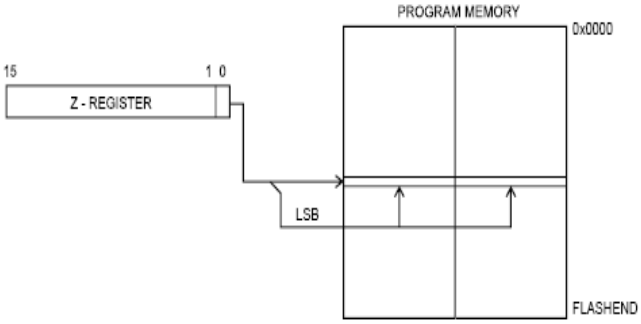
3. محتوای حافظه برنامه با آدرس موجود در Z را درRd ذخیره کرده سپس Z را افزایش میدهد.



– (Z) Z ← Z + 1

Rd, Z+ 0 ≤ d ≤ 31

دستورELPM :



یک بایت را توسط اشاره گر Zو ثبات ورودی خروجی PAMPZ اشاره شده است را میخواند . یک بایتی را که توسط ثبات z و ثبات PAMPZ (در فضایI/O) به آن اشاره شده است را در ثبات Rd بار میکند این دستور توانایی آدرس دهی تمام فضای حافظه برنامه را دارد .این دستور به صورتهای زیر بکار میرود:

1.ELPM :یک بایت را از آدرس موجود در ثبات PAMPZ: Z به ثبات R0 میخواند.

R0 ← (RAMPZ:Z)

ELPM None, R0 implied

2. یک بایت از آدرس PAMPZ : Z از حافظه برنامه را میخواند و در ثبات Rd قرار میدهد.

Rd ← (RAMPZ:Z)

ELPM Rd, Z 0 ≤ d ≤ 31

3. یک بایت را از آدرس PAMPZ : Z از حافظه برنامه خوانده و در ثبات Rd قرار میدهد

سپس PAMPZ: Z را افزایش میدهد

Rd ← (RAMPZ:Z) (RAMPZ:Z) ← (RAMPZ:Z) + 1

ELPM Rd, Z+ 0 ≤ d ≤ 31

دستور SPM: store program memory

با این دستور میتوان یک صفحه را در حافظه برنامه نوشت یا از حافظه برنامه پاک کرد . بوسیله این دستور میتوان یک صفحه را در حافظه برنامه نوشت یا بیتهای قفل boot loader را یک کرد و یا یک صفحه را از برنامه پاک کرد و یا یک کلمه در آن نوشت یا یک صفحه ثبات موقت را در حافظه برنامه نوشت . صفحه یا کلمه در آدرس ثبات اشاره گر Z و ثبات PAMPZ قرار میگیرد .ث های R0,R1 نیز برای داده استفاده میشوند.

1. پاک کردن حافظه برنامه: (RAMPZ:Z) ← $ffff
2. نوشتن کلمه حافظه برنامه: (RAMPZ:Z) ← R1:R0
3. نوشتن بافر صفحه موقت: (RAMPZ:Z) ← R1:R0
4. نوشتن بافر موقتی صفحه به حافظه برنامه: (RAMPZ:Z) ← TEMP
5. یک کردن بیت قفل boot loader : BLBITS ← R1:R0

دستور ات استفاده از پشته :

پشته داده حافظه ای است که داده هایی که در آن ذخیره میشوند به ترتیب در آن ذخیره میشوند سپس هنگام بازیابی آخرین داده ذخیره شده ابتدا بازیافت میشود. آدرس پشته بوسیله ثبات اشاره گر پشته SP (SPL:SPH) اشاره میشود.

نکته : اگر پشته از 256 کمتر باشد SPHتعریف نمیشود.

نکته : مقادیر RAMEND و FLASHEND مقادیر مشخص برای میکروکنترلر میباشند که در فایل INC از هر تراشه موجود است .

استفاده از پشته :

دستور PUSH: (push register on stack)محتوای مورد نظر را در پشته ذخیره میکند . ثبات اشاره گر پشته پس از پوش کاهش می یابد .

STACK ← Rr

PUSH Rr 0 ≤ r ≤ 31

دستور POP: (pop register from stack) آخرین داده ای که در پشته ذخیره شده است را بازیابیمیکند.این دستور ابتدا ثبات اشاره گر پشته را افزایش داده سپس محتوای آن را بازیافت میکند و در ثبات Rd قرار میدهد.

Rd ← STACK

POP Rd $0 \le d \le 31$

همانگونه که در فصل قبل توضیح دادیم با ید در ابتدای برنامه اشاره گـر پـشته را تنظـیم کنـیم این کار را طبق برنامه زیر انجام میدهیم :

```
.def temp=R16
Ldi temp,high(ramend)
Out sph,temp
Ldi temp,low(ramend)
Out spl,temp
```

زمانبندی در طول اجرای برنامه :

زمان اجرای هر دستور العمل بستگی به سیکل فرکانس کاری تراشه مقـدار کـلاک مـورد نیـاز برای اجرای همان دستور دارد.هر کلاک ترشه را یک ماشین سـیکل گوینـد . دسـتورالعمل هـا ممکن است در یک ماشین سیکل یا بیشتر اجرا شوند . در میکروکنترلر avr بیشتر دسـتور ات در یک ماشین سیکل اجرا میشوند.

زمان هر ماشین سیکل = 1/(فرکانس تراشه)

زیر روالها :

استفاده از زیر روالها (برچسبها-زیر برنامه ها) مشکل استفاده زیاد از فـضای ذخیـره برنامـه را جبران میکند . سلسله مراتب یکبار در کد ذخیـره میـشود. زیـر روال بـا یـک برچـسب شـروع میشود.

نام برچسب زیر روال :

کد برنامه زیر روال

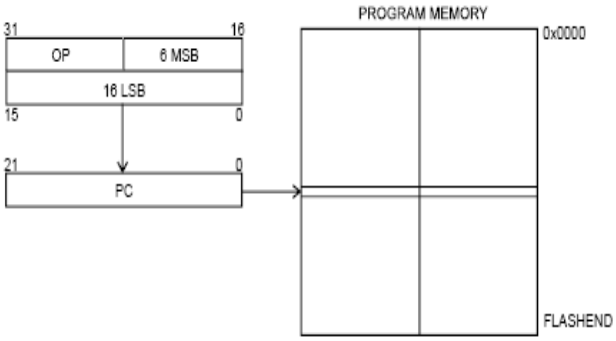
RET

دستور RET برای بازگشت از زیر روال میباشد .

نکته : دقت شود قبل از فراخوانهای زیر روال باید اشاره گر پـشته را تنظـیم کنیـد.(طبـق آنچـه گفته شد)و گر نه آدرس بازگشت روال در مکان فراخوانی نخواهد بود .

دستور ات فراخوانی زیر روال و پرش بصورت زیر میباشند:

CALL,JMP :



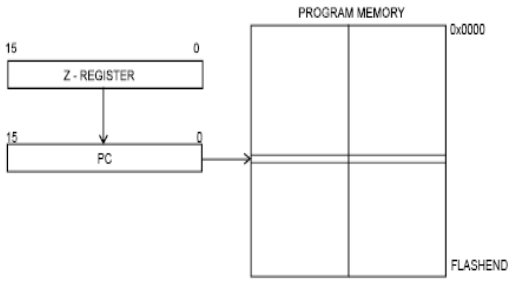
دسـتور ات JMP,CALL فراخـوانی بلن به زیر روال میباشـد.بـرای دسـتور CALL که فراخـوانی زیـر روال اسـت باید آدرس بازگشت در پـشته ذخیـره شود .پس باید در ابتدای برنامـه پـشته

را تنظیم کرد .اگر شمارنده برنامه PC 16 بیتی باشد و حافظه برنامه 128 کیلو بایت باشد آنگاه $0\leq K\leq 64K$ میباشد و اگر PC 22 بیتی باشدآنگاه $0\leq K\leq 4M$ میباشد. برای بازگشت از زیر روال در پایان زیر روال از دستور RET استفاده میشود.

دستور JMP که پرش به زیرروال معین از برنامه میباشد نیازی به دستور بازگشتRET نیست چون پرش است و بازگشت ندارد همچنین نیازی به تنظیم پشته ندارد .

ICALL,IJMP:

ICALL فراخوانی غیر مستقیم زیر روال(SUBROUTIN) میباشد و IJMP پرش مستقیم زیر روال میباشد. آدرس فراخوانی یا پرش از ثبات اشاره گر Z گرفته میشود.



1. اگر PC 16 بیتی باشد و حافظه 128 کیلو بایت باشد آنگاه با دستور ات ICALL,IJMP:

$PC(15:0)\leftarrow Z(15:0)$

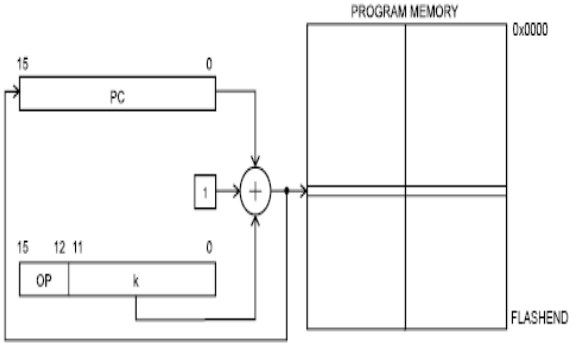
2. اگر PC 22 بیتی باشد و حافظه 8M بایت باشد آنگاه با دستور ات ICALL,IJMP:

$PC(15:0)\leftarrow Z(15:0), PC(21:16)\leftarrow 0$

نکته : برای دستور ICALL باید آدرس بازگشت درپشته ذخیره شود پس باید در ابتدای برنامه پشته را تنظیم کرد.

RCALL,RJMP:

این دو دستور فراخوانی و پرش نسبی به زیر روال میباشد.در این دستور ات در محدوده $-2K\leq K\leq 2K$ فراخوانی و پرش میشوند .که در آن آدرس در محدوده $PC-2K+1, PC-2K$ در شمارنده برنامه ذخیره میشود.در دستور RCALL باید از زیر روال با دستور RET به برنامه بازگشت پس باید آدرس بازگشت در پشته ذخیره شود بنابراین باید ابتدا پشته را تنظیم کنیم. RJMP پرش بدون بازگشت است و نیازی بهRET و تنظیم پشته



ندارد.

EICALL,EIJMP :

در ایـن فراخوانهـا آدرس فراخـوانی بپـسیله ثبـات اشـاره گـر Z و ثبـات بانـامEIND در فضایI/O داده میشود.در EICALL باید آدرس بازگشت در پشته ذخیره شود بنابراین باید ابتدا پشته را تنظیم کنیم. (از دستور RET برای بازگشت از زیر روال استفاده میشود.)

PC(21:16)←EIND,PC(15:0)←Z(15:0)

انشعاب و جهش و مقایسه :

انشعاب و جهش اغلب اوقات بهیک شرط بـستگی دارد کـه انـشعاب شـرطی و جهـش شـرطی نامیـده میـشوند.در انـشعاب شـرطی یـک شـرط بررسـی میـشود و اگـر شـرط برقـرار باشـد میکروکنترلر به زیرروالی انشعاب میکند . در جهش شرطی شرطی یک شرط بررسی میـشود و اگر شرط برقرار باشد میکروکنترلر از روی دستور بعدی جهش میکند (دسـتور بعـدی را اجـرا نمیکند).

دستورات انشعاب و جهش بصورت زیر میباشند:

CPSE : Rr,Rd را باهم مقایسه میکند اگر برابر باشند از روی دستور بعدی جهـش میکنـد و گر نه دستور بعدی را اجرا میکند.

If Rd = Rr then PC ← PC + 2 (or 3) else PC ← PC + 1

CPSE Rd,Rr $0 \le d \le 31,\ 0 \le r \le 31$

CP :این دستور RR,RDرا باهم مقایسه میکنـد . اگـر برابـر باشـندRd=Rrپـرچم z از ثبـات وضــعیت(sreg) را یــک میکنــدو اگــر Rd>Rrباشــد پــرچم c از ثبــات sreg را صــفر میکند(c=0)و اگر Rd<Rr باشد c=1 میشود.(در واقع Rd-Rr میکند)

CP Rd,Rr $0 \le d \le 31,\ 0 \le r \le 31$

CPC : ثبات Rd,Rrرا مقایسه میکند کریر را نیـز در مقایـسه احتـساب میکنـد.(Rd-Rr-c) اگر Rd=Rr+c باشد zرا یک میکند اگر Rd<Rr+c باشد c=1 و اگـر Rd>Rr+c باشـد c=0 میکند.

CPI : Rd را با عدد k مقایسه میکند(Rd-k) اگرRd=k باشدz=1و اگـرRd>k باشـدc=0 و اگر Rd<k باشد c=1میکند.

CPC Rd,Rr $0 \le d \le 31,\ 0 \le r \le 31$

SBRC :ثبات Rr را بررسی میکند اگر بیت b از ثبـات Rrصـفر باشـد از روی دسـتور بعـدی جهش میکند و گر نه دستور بعدی را اجرا میکند.

If Rr(b) = 0 then PC ← PC + 2 (or 3) else PC ← PC + 1

SBRC Rr,b $0 \le r \le 31, 0 \le b \le 7$

TST : ثبات Rd را برای تشخیص صفر بودن یا منفی بودن تست میکند. ثبات را با خودش AND منطقی میکند.

Rd ← Rd • Rd

TST Rd $0 \le d \le 31$

SBRS :اگر بیت b از ثبات Rr یک باشد از روی دستور بعدی جهش میکند و گر نه دستور بعدی را اجرا میکند.

If Rr(b) = 1 then PC ← PC + 2 (or 3) else PC ← PC + 1

SBRS Rr,b $0 \le r \le 31, 0 \le b \le 7$

SBIC:اگر بیت B از ثبات I/O با نام A صفر باشد از روی دستور بعدی جهش میکند و گر نه دستور بعدی را اجرا میکند.

If I/O(A,b) = 0 then PC ← PC + 2 (or 3) else PC ← PC + 1

SBIC A,b $0 \le A \le 31, 0 \le b \le 7$

SBIS :اگر بیت B از ثبات I/O با نام A یک باشد از روی دستور بعدی جهش میکند و گر نه دستور بعدی را اجرا میکند.

If I/O(A,b) = 1 then PC ← PC + 2 (or 3) else PC ← PC + 1

SBIS A,b $0 \le A \le 31, 0 \le b \le 7$

BRBS : اگر پرچم S الز ثبات وضعیت SREG یک باشد از روی دستور بعدی جهش میکند و گر نه دستور بعدی را اجرا میکند.

If SREG(s) = 1 then PC ← PC + k + 1, else PC ← PC + 1

BRBS s,k $0 \le s \le 7, -64 \le k \le +63$

BRBC : اگر پرچم S از ثبات وضعیت SREG صفر باشد از روی دستور بعدی جهش میکند و گر نه دستور بعدی را اجرا میکند.

If SREG(s) = 0 then PC ← PC + k + 1, else PC ← PC + 1

BRBC s,k $0 \le s \le 7, -64 \le k \le +63$

BREQ : این دستور پس از دستور ات CPI,CPC,CP و ... میآید . در آن دستور ات یک عمل مقایسه صورت میگرفت . با این دستور پس از مقایسه توسط دستورات قبل اگر برابر باشند به آدرس K (برچسبK) انشعاب میکند.وگرنه ادامه برنامه را اجرا میکند. این دستور

پرچم Z از ثبات وضعیت SREG را بررسی میکند اگر Z=1 باشد به برچسب K انشعاب میکند.

If Rd = Rr (Z = 1) then PC ← PC + k + 1, else PC ← PC + 1

BREQ k -64 ≤ k ≤ +63

BRNE : این دستور پس از دستور ات CPI,CPC,CP و ... میآید . در آن دستور ات یک عمل مقایسه صورت میگرفت . با این دستور پس از مقایسه توسط دستورات قبل اگر برابر نباشند به آدرس K (برچسبK) انشعاب میکند.وگرنه ادامه برنامه را اجرا میکند. این دستور پرچم Z از ثبات وضعیت SREG را بررسی میکند اگر Z=0 باشد به برچسب K انشعاب میکند.

If Rd ≠ Rr (Z = 0) then PC ← PC + k + 1, else PC ← PC + 1

BRNE k -64 ≤ k ≤ +63

BRCS: اگر نقلی(کریر)C=1 باشد به برچسب K انشعاب میکندوگرنه ادامه برنامه را اجرا میکند.

If C = 1 then PC ← PC + k + 1, else PC ← PC + 1

BRCS k -64 ≤ k ≤ +63

BRCC: اگر نقلی (کریر) C=0 باشد به برچسب K انشعاب میکندوگرنه ادامه برنامه را اجرا میکند.

If C = 0 then PC ← PC + k + 1, else PC ← PC + 1

BRCC k -64 ≤ k ≤ +63

BRSH: : این دستور پس از دستور ات CPI,CPC,CP و ... میآید . در آن دستور ات یک عمل مقایسه صورت میگرفت . با این دستور پس از مقایسه توسط دستورات قبل اگر بزرگتر یا مساوی باشند به آدرس K (برچسبK) انشعاب میکند.وگرنه ادامه برنامه را اجرا میکند. این دستور پرچم C از ثبات وضعیت SREG را بررسی میکند اگر C=0 باشد به برچسب K انشعاب میکند.

If Rd ≥Rr (C = 0) then PC ← PC + k + 1, else PC ← PC + 1

BRSH k -64 ≤ k ≤ +63

BRLO: : این دستور پس از دستور ات CPI,CPC,CP و ... میآید . در آن دستور ات یک عمل مقایسه صورت میگرفت . با این دستور پس از مقایسه توسط دستورات قبل اگرکوچکتر باشند به آدرس K (برچسبK) انشعاب میکند.وگرنه ادامه برنامه را اجرا میکند.

BRMIK: اگر نتیجه یک دستور منفی باشد پرچم N از ثبات وضعیت SREG یک میشود. این دستور پرچم N را بررسی میکند اگر یک باشد انشعاب میکند.

If Rd < Rr (C = 1) then PC ← PC + k + 1, else PC ← PC + 1

BRLO k $-64 \le k \le +63$

BRPL : اگر نتیجه یک دستور مثبت باشد پرچم N از ثبات وضعیت SREG صفر میشود. این دستور پرچم N از ثبات وضعیت SREG را بررسی میکند اگر صفر باشد انشعاب میکند.

If N = 0 then PC ← PC + k + 1, else PC ← PC + 1

BRPL k $-64 \le k \le +63$

BRGE : در دستور مقایسه اگر عبارت مقایسه شده برابر یا بزرگتر باشد به برچسب Kانشعاب میکند و گر نه ادامه برنامه را اجرا میکند . این دستور برای اعداد علامتدار میباشد.

If Rd ≥ Rr (N ⊕ V = 0) then PC ← PC + k + 1, else PC ← PC + 1

BRGE k $-64 \le k \le +63$

BRLT : : در دستور مقایسه اگر عبارت مقایسه شده کوچکتر باشد به برچسب Kانشعاب میکند و گر نه ادامه برنامه را اجرا میکند . این دستور برای اعداد علامتدار میباشد.

If Rd < Rr (N ⊕ V = 1) then PC ← PC + k + 1, else PC ← PC + 1

BRLT k $-64 \le k \le +63$

BRHS :اگر نقلی نیمه (کریر نیمه)H یک باشد به برچسب K انشعاب میکند و گر نه ادامه برنامه را اجرا میکند.

If H = 1 then PC ← PC + k + 1, else PC ← PC + 1

BRHS k $-64 \le k \le +63$

BRHC :اگر نقلی نیمه (کریر نیمه)H صفر باشد به برچسب K انشعاب میکند و گر نه ادامه برنامه را اجرا میکند.

If H = 0 then PC ← PC + k + 1, else PC ← PC + 1

BRHC k $-64 \le k \le +63$

BRTS: اگر پرچم T از ثبات وضعیت یک باشد انشعاب میکند و گرنه ادامه برنامه را اجرا میکند.

If T = 1 then PC ← PC + k + 1, else PC ← PC + 1

BRTS k $-64 \le k \le +63$

BRTC: اگر پرچم T از ثبات وضعیت صفرباشد انشعاب میکند و گرنه ادامه برنامه را اجرا میکند.

If T = 0 then PC ← PC + k + 1, else PC ← PC + 1

BRTC k $-64 \le k \le +63$

BRVS : اگر پرچم سرریزی V از ثبات وضعیت یک باشد انشعاب میکند و گرنه ادامه برنامه را اجرا میکند.

If V = 1 then PC ← PC + k + 1, else PC ← PC + 1

BRVS k $-64 \le k \le +63$

BRVC : اگر پرچم سرریزی V از ثبات وضعیت صفر باشد انشعاب میکند و گرنه ادامه برنامه را اجرا میکند.

If V = 0 then PC ← PC + k + 1, else PC ← PC + 1

BRVC k $-64 \le k \le +63$

BRIE : اگر وقفه سراسری فعال باشد (بیت I از ثبات وضعیت یک باشد) انشعاب میکند و گرنه ادامه برنامه را اجرا میکند.

If I = 1 then PC ← PC + k + 1, else PC ← PC + 1

BRIE k $-64 \le k \le +63$

BRID : اگر وقفه سراسری غیر فعال باشد (بیت I از ثبات وضعیت صفر باشد) انشعاب میکند و گرنه ادامه برنامه را اجرا میکند.

If I = 0 then PC ← PC + k + 1, else PC ← PC + 1

BRID k $-64 \le k \le +63$

دستور ات دستکاری بیتی :

الف)عملگر های منطقی :

OR: این دستور عملOR منطقی را بین ثبات هایRd,Rr انجام میدهد و نتیجه را در Rd میریزد.

Rd ← Rd v Rr

OR Rd,Rr $0 \le d \le 31,\ 0 \le r \le 31$

ORI: Rd را با یک عدد ثابت OR منطقی میکند و نتیجه را در Rd میریزد.

Rd ← Rd v K

ORI Rd,K $16 \le d \le 31,\ 0 \le K \le 255$

AND : این دستور عمل AND منطقی را بین ثبات هایRd,Rr انجام میدهد و نتیجه را در Rd میریزد.

Rd ← Rd • Rr

AND Rd,Rr $0 \le d \le 31,\ 0 \le r \le 31$

ANDI :Rd را با یک عدد ثابت AND منطقی میکند و نتیجه را در Rd میریزد.

$Rd \leftarrow Rd \bullet K$

ANDI Rd,K $16 \le d \le 31,\ 0 \le K \le 255$

EOR : Rd,Rr را xor میکند و نتیجه را در Rd میریزد.

$Rd \leftarrow Rd \oplus Rr$

EOR Rd,Rr $0 \le d \le 31,\ 0 \le r \le 31$

ب)دستور ات دسترسی بیتی :

SBR : بیتهای Rd که در K یک است را یک میکند.(در واقع Rd,k را or منطقی میکند)

$Rd \leftarrow Rd \vee K$

SBR Rd,K $16 \le d \le 31,\ 0 \le K \le 255$

Cbr : بیتهای Rd که در K یک است را صفر میکند.(در واقع Rd و متمم k را and منطقی میکند)

$Rd \leftarrow Rd \bullet (\$FF - K)$

CBR Rd,K $16 \le d \le 31,\ 0 \le K \le 255$

COM : مکمل یک از ثبات Rd را برمیگرداند. (همه بیتهای آن را معکوس میکند.) این دستور ثبات Rd را NOT میکند.

$Rd \leftarrow \$FF - Rd$

COM Rd $0 \le d \le 31$

NEG : مکمل دو از ثبات Rd را برمیگرداند. (عدد مثبت را منفی میکند) تفاوت مکمل دو و مکمل یک این است که با مکمل دو عدد علامتدار مثبت را به منفی آن تبدیل میکنیم .

$Rd \leftarrow \$00 - Rd$

NEG Rd $0 \le d \le 31$

BLD : مقدار پرچم T از ثبات SREG را در بیت b از ثبات Rd بار میکند.

$Rd(b) \leftarrow T$

BLD Rd,b $0 \le d \le 31,\ 0 \le b \le 7$

BST:بیتb از ثبات Rd را در پرچمT از ثبات SREG بار میکند.

$T \leftarrow Rd(b)$

BST Rd,b $0 \le d \le 31,\ 0 \le b \le 7$

INC : ثبات Rd را افزایش میدهد.

$Rd \leftarrow Rd + 1$

INC Rd $\quad 0 \le d \le 31$

DEC : ثبات Rd راکاهش میدهد.

$Rd \leftarrow Rd - 1$

DEC Rd $\quad 0 \le d \le 31$

CLR: ثبات Rd را پاک میکند.

$Rd \leftarrow Rd \oplus Rd$

CLR Rd $\quad 0 \le d \le 31$

SER : ثبات Rd را یک میکند.

$Rd \leftarrow \$FF$

SER Rd $\quad 16 \le d \le 31$

BSET : بیت S از ثبات وضعیت را یک میکند.

$SREG(s) \leftarrow 1$

BSET s $\quad 0 \le s \le 7$

BCLR : بیت S از ثبات وضعیت را صفر(پاک) میکند.

$SREG(s) \leftarrow 0$

BCLR s $\quad 0 \le s \le 7$

SBI : بیتb از ثبات I/O ورودی خروجی A را یک میکند.

$I \leftarrow 0$

SBI A,b $\quad 0 \le A \le 31, 0 \le b \le 7$

CBI : بیتb از ثبات I/O ورودی خروجی A را صفر میکند.

$I/O(A,b) \leftarrow 0$

CBI A,b $\quad 0 \le A \le 31, 0 \le b \le 7$

SEC : پرچم C از ثبات وضعیتSREG را یک میکند.

$C \leftarrow 1$

SEC $\quad$ None

CLC : پرچم C از ثبات وضعیتSREG را صفر (پاک) میکند.

$C \leftarrow 0$

CLC $\quad$ None

SEN : پرچم N از ثبات وضعیتSREG را یک میکند.

$N \leftarrow 1$

SEN None

CLN : پرچم N از ثبات وضعیت SREG را صفر میکند.

$N \leftarrow 0$

CLN None

SEZ : پرچم Z از ثبات وضعیت SREG را یک میکند.

$Z \leftarrow 1$

SEZ None

CLZ : پرچم Z از ثبات وضعیت SREG را صفر میکند.

$Z \leftarrow 0$

CLZ None

SEI : پرچم I از ثبات وضعیت SREG را یک میکند. وقفه سراسری(بیت I از ثبات وضعیت) را فعال میکند.

$I \leftarrow 1$

SEI None

CLI : پرچم I از ثبات وضعیت SREG راصفرمیکند.. وقفه سراسری(بیت I از ثبات وضعیت) را غیر فعال میکند.

$I \leftarrow 0$

CLI None

SEV : پرچم V از ثبات وضعیت SREG را یک میکند.

$V \leftarrow 1$

SEV None

CLV: پرچم V از ثبات وضعیت SREG را صفر میکند.

$V \leftarrow 0$

CLV None

SET : پرچم T از ثبات وضعیت SREG را یک میکند.

$T \leftarrow 1$

SET None

CLT : پرچم T از ثبات وضعیت SREG را صفر میکند.

$T \leftarrow 0$

CLT None

SEH : پرچم H از ثبات وضعیتSREG را یک میکند.

$H \leftarrow 1$

SEH None

CLH : پرچم H از ثبات وضعیتSREG را صفر میکند.

$H \leftarrow 0$

CLH None

TST : ثبات Rd را برای تشخیص صفر بودن یا منفی بودن تست میکند. ثبات را با خودش AND منطقی میکند.

$Rd \leftarrow Rd \bullet Rd$

TST Rd $0 \le d \le 31$

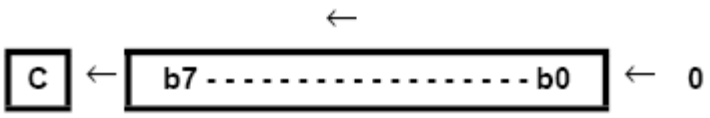
ج)دستور ات شیفت و چرخش:

ما دو نوع شیفت داریم 1. شیفت منطقی 2. شیفت حسابی

LSL : شیفت منطقی به چپ

بیتهای ثبات Rdرا یکواحد به چپ شیفت میدهد . به جای b0 صفر قرار میدهدو بیت هفتمb7 را به پرچم C منتقل میکند.

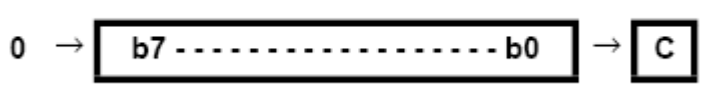
LSL Rd $0 \le d \le 31$



LSR : شیفت منطقی به راست

بیتهای ثبات Rdرا یکواحد به راست شیفت میدهد . به جای b7 صفر قرار میدهدو بیت b0 را به پرچم C منتقل میکند.

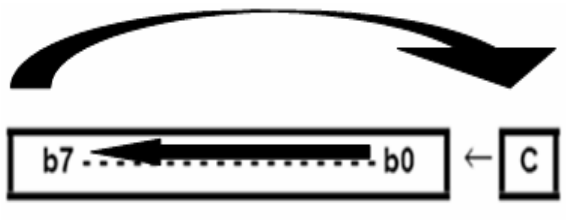
LSR Rd $0 \le d \le 31$



ROL :چرخش به چپ با کمک پرچمC(چرخش منطقی)

بیتهای ثبات Rd را یک واحد به چپ شیفت میدهد و محتوای کریر را به b0 منتقل و b7 را به کریر منتقل میکند.

ROL Rd $0 \le d \le 31$



ROR : چرخش به راست با کریر(چرخش منطقی)

بیتهای ثبات Rd یک واحد به راست شیفت می یابند و محتوای کریر به b7 و b0 به کریر منتقل میشوند.

ROR Rd $0 \le d \le 31$

ASR : شیفت حسابی به راست

بیت علامت تغییر نمیکند و 7 بیت دیگر یکواحد به راست شیفت میشوند. بیت b0 به کریر میـرود و بیت b7 علاوه بر انتقال به b6 صفر نمیشود بلکه مقدار قبلی خود(علامت) را حفظ میکند.

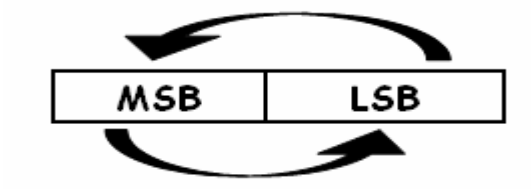
ASR Rd $0 \le d \le 31$

SWAP :

نیمه بالایی Rd با نیمه پایینی آن معاوضه میشود.

$R(7:4) \leftarrow Rd(3:0), R(3:0) \leftarrow Rd(7:4)$

SWAP Rd $0 \le d \le 31$



نکته : شیفت و چرخش اعداد باینری به معنای ضرب آنها در 2و تقسیمشان بر 2 است . ضـرب در 2 با LSL(شیفت منطقی به چپ) و تقسیم بر 2 با LSR (شیفت منطقی به راست) صورت میگیرد.

د) دستور ات محاسباتی:

ADD: جمع بدون احتساب کریر

ثبات های Rd,Rr را باهم جمع کرده و در Rd میریزد.

$Rd \leftarrow Rd + Rr$

ADD Rd,Rr $0 \le d \le 31,\ 0 \le r \le 31$

Adc : جمع با احتساب کریر

ثبات های Rd,Rr و کریر را باهم جمع کرده و در Rd میریزد.

$Rd \leftarrow Rd + Rr + C$

ADC Rd,Rr $\quad 0 \le d \le 31,\ 0 \le r \le 31$

ADIW : جمع فوری با کلمه (16 بیتی)

ثبات 16 بیتی (Rd+1:Rd) و عدد ثابت 16بیتیK (HIGH(K):LOW(K)) را باهم جمع و نتیجه را در ثبات 16 بیتی (Rd+1:Rd) می ریزد.

$Rd+1:Rd \leftarrow Rd+1:Rd + K$

ADIW Rd+1:Rd,K $\quad d \in \{24,26,28,30\},\ 0 \le K \le 63$

SUB : تفریق بدون احتساب کریر

ثبات های Rd,Rr را از هم کم کرده ودر ثبات Rd می ریزد.

$Rd \leftarrow Rd - Rr$

SUB Rd,Rr $\quad 0 \le d \le 31,\ 0 \le r \le 31$

SUBI : تفریق فوری

ثبات RD و عدد ثابت K را از هم کم کرده و در ثبات Rd می ریزد.

$Rd \leftarrow Rd - K$

SUBI Rd,K $\quad 16 \le d \le 31,\ 0 \le K \le 255$

SBC : تفریق با احتساب کریر

ثبات های Rd,Rr و کریر را از هم کم کرده ودر ثبات Rd می ریزد.

$Rd \leftarrow Rd - Rr - C$

SBC Rd,Rr $\quad 0 \le d \le 31,\ 0 \le r \le 31$

SBCI : تفریق فوری با احتساب کریر

ثبات های Rd,K و کریر را از هم کم کرده ودر ثبات Rd می ریزد.

$Rd \leftarrow Rd - K - C$

SBCI Rd,K $\quad 16 \le d \le 31,\ 0 \le K \le 255$

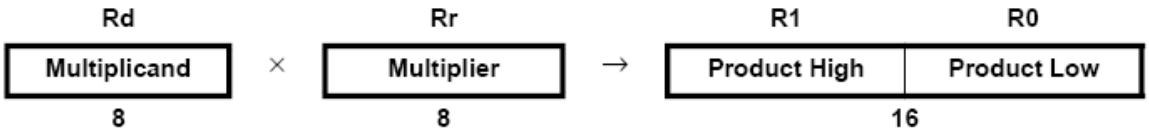
SBIW :تفریق فوری از کلمه (16 بیتی)

ثبات 16 بیتی (Rd+1:Rd) و عدد ثابت 16بیتیK (HIGH(K):LOW(K)) را از هم کرده و نتیجه را در ثبات 16 بیتی (Rd+1:Rd) می ریزد.

$Rd+1:Rd \leftarrow Rd+1:Rd - K$

SBIW Rd+1:Rd,K $\quad d \in \{24,26,28,30\},\ 0 \le K \le 63$

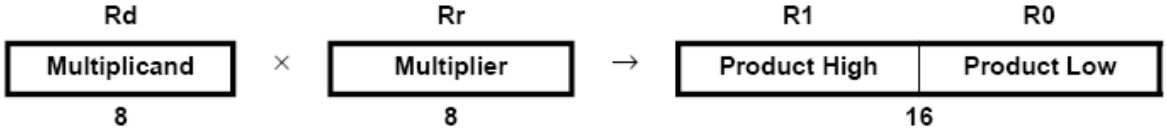
MUL : این دستور دو ثبات 8 بیتی را در هم ضرب میکند و حاصل 16 بیتی را در دو ثبات ذخیـره میکند بطوریکه قسمت پایین نتیجه را در ثبات Rrو قسمت بالایی آنرا در ثبـات Rd قـرار میدهـد. این ضرب بدون علامت است.



R1:R0 ← Rd × Rr (unsigned ← unsigned × unsigned)

MUL Rd,Rr $0 \le d \le 31,\ 0 \le r \le 31$

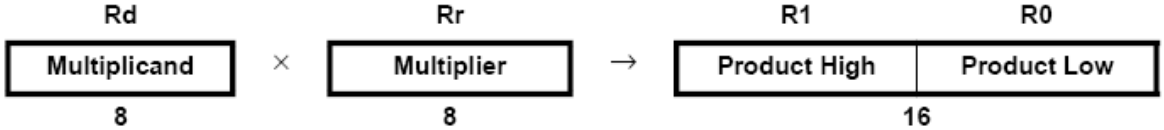
MULS: : این دستور دو ثبات علامتدار 8 بیتی را در هم ضرب میکند و حاصـل 16 بیتـی را در دو ثبات ذخیره میکند بطوریکه قسمت پایین نتیجه را در ثبات Rrو قسمت بـالایی آنـرا در ثبـات Rd قرار میدهد. نتیجه این ضرب علامتدار است.



R1:R0 ← Rd × Rr (signed ← signed × signed)

MULS Rd,Rr $16 \le d \le 31,\ 16 \le r \le 31$

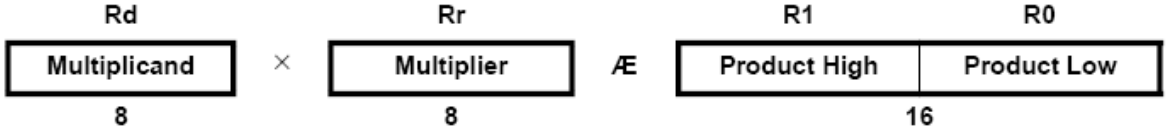
MULSU : این دستور برای ضرب عبارت علامتدار در عبارت بدون علامت میباشد که حاصـل آن علامتدار میباشد و نتیجه در ثبات هایR0وr1 ذخیره میشود.



R1:R0 ← Rd × Rr (signed ← signed × unsigned)

MULSU Rd,Rr $16 \le d \le 23,\ 16 \le r \le 23$

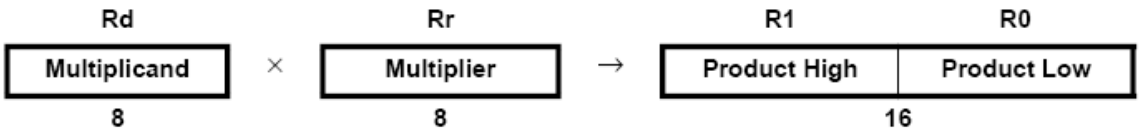
Fmul :این دستور ضرب اعشاری 8 بیتی بـین دو عـدد اعـشاری بـدون علامـت را انجـام میدهـد و حاصل را یک بیت شیفت میدهد . ممیز در ورودی ها بین بیت 6و7 است . عـدد خروجـی 16 بیتـی است که در ثبات های R0,R1 ذخیره میشود ممیز در خروجی بین بیتهای 14و 15 میباشد.



R1:R0 ← Rd × Rr (unsigned (1.15) ← unsigned (1.7) × unsigned (1.7))

FMUL Rd,Rr $16 \le d \le 23,\ 16 \le r \le 23$

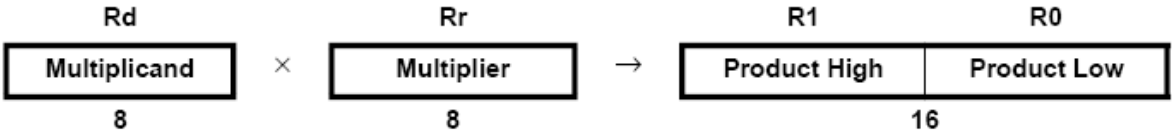
FMULS:این دستور مانند دستور قبل است فقط اینکه ضرب علامتدار میباشد.



R1:R0 ← Rd × Rr (signed (1.15) ← signed (1.7) × signed (1.7))

FMULS Rd,Rr $16 \le d \le 23$, $16 \le r \le 23$

FMULSU: مانند دستور قبل است اما یکی از اعداد علامتدار و دیگری بدون علامت است و نتیجه علامتدار است.



R1:R0 ← Rd × Rr (signed (1.15) ← signed (1.7) × unsigned (1.7))

FMULSU Rd,Rr $16 \le d \le 23$, $16 \le r \le 23$