حالت‌های آدرس‌دهی
و
مجموعه دستورالعمل‌ها

در میکروکنترلرهای AVR

فهرست مطالب

- حالت‌های آدرس‌دهی (Addressing modes)
- مجموعه دستورالعمل‌ها

اسمبلی میکروکنترلرهای ۸ بیتی AVR

• میکروکنترلر RISC تقویت شده AVR، حالت‌های آدرس‌دهی قدرتمند و موثری را برای دسترسی به حافظه برنامه (فلش) و حافظه داده (SRAM)، فایل ثبات، حافظه I/O و حافظه توسعه‌یافته I/O فراهم می‌نماید.

پرچم‌ها (خلاصه)

علائم و اختصارات ثبات وضعیت (SREG) حاوی بیت های وضعیت (پرچم‌ها)

نام پرچم	علامت پرچم
پرچم نقلی	C
پرچم صفر	Z
پرچم منفی	N
پرچم سرریز	V
پرچم تست مقادیر علامت‌دار	S: N ⊕ V
پرچم نیمه‌توازن	H
بیت انتقال (مورد استفاده در دستورالعمل‌های BST و BLD)	T
پرچم وقفه سراسری	I

علائم و اختصارات ثبات‌ها و عملوندها

علائم و اختصارات ثبات‌ها و عملوندها و منظور از استفاده از آنها در دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

نام ثبات	عملیات
Rd	ثبات مقصد (و منبع) در فایل ثبات
Rr	ثبات منبع در فایل ثبات
R	نتیجه بعد از اجرای دستورالعمل
K	داده ثابت
k	آدرس ثابت
b	بیت در فایل ثبات یا ثبات I/O (سه بیت)
s	بیت در ثبات وضعیت (سه بیت)
X,Y,Z	ثبات آدرس‌دهی غیر مستقیم (X=R27:R26, Y=R29:R28 and Z=R31:R30)
A	آدرس محل I/O

علائم و اختصارات ثبات‌ها و عملوندها

علائم و اختصارات ثبات‌های ورودی/خروجی و منظور از استفاده از آنها در دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR:

• ثبات های RAMPX، RAMPY و RAMPZ ثبات‌های متصل به ثبات های X، Y و Z هستند که آدرس‌دهی غیرمستقیم کل فضای داده میکروکنترلرهای با بیش از 64K بایت فضای داده و واکنشی داده ثابت در میکروکنترلرهای خانواده ATmega با بیش از 64K بایت حافظه برنامه را امکان پذیر می‌سازند.

• ثبات RAMPD ثبات متصل به ثبات Z است که آدرس‌دهی مستقیم کل فضای داده میکروکنترلرهای با بیش از 64K بایت فضای داده را امکان‌پذیر می‌سازد.

علائم و اختصارات ثبات‌ها و عملوندها

- ثبات EIND ثبات متصل به ثبات Z است که پرش و فراخوانی غیرمستقیم به کل فضای داده میکروکنترلرهای با بیش از 64K کلمه (128K بایت) و فضای برنامه را امکان‌پذیر می‌سازد.

- پشته (STACK) برای ذخیره‌سازی آدرس بازگشت و ثبات‌های پوش شده، و ثبات SP ثبات اشاره‌گر پشته می‌باشد.

علائم و اختصارات وضعیت پرچم‌ها

علائم مورد استفاده جهت بیان چگونگی تغییر وضعیت پرچم‌ها بازاء اجراء هر دستورالعمل:

⇔: پرچم توسط دستورالعمل تغییر کرده است.
0 : پرچم توسط دستورالعمل صفر شده است.
1 : پرچم توسط دستورالعمل یک شده است.
- : پرچم توسط دستورالعمل تغییر نکرده است.

توجه: در مباحث و جداولی که در ادامه می‌آیند، OP به معنای بخش کد عملیاتی کلمه دستورالعمل است.

کلمات RAMEND و FLASHEND بالاترین محل در فضای داده و برنامه را نشان می‌دهند.

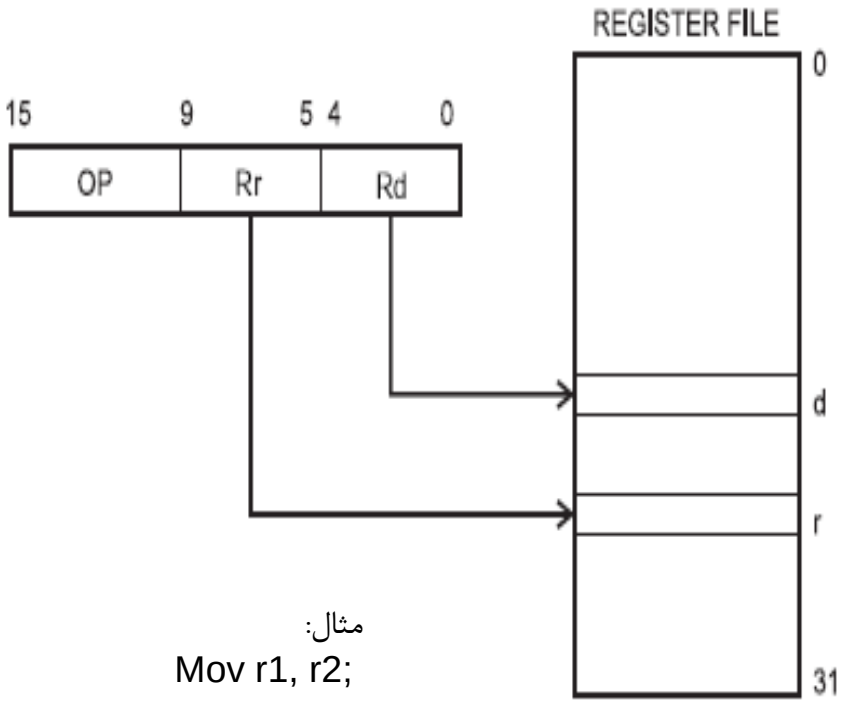
ATMega16 ← RAMEND
1120

ATMega16 ← FLASHEND
8k word

حالت‌های آدرس‌دهی

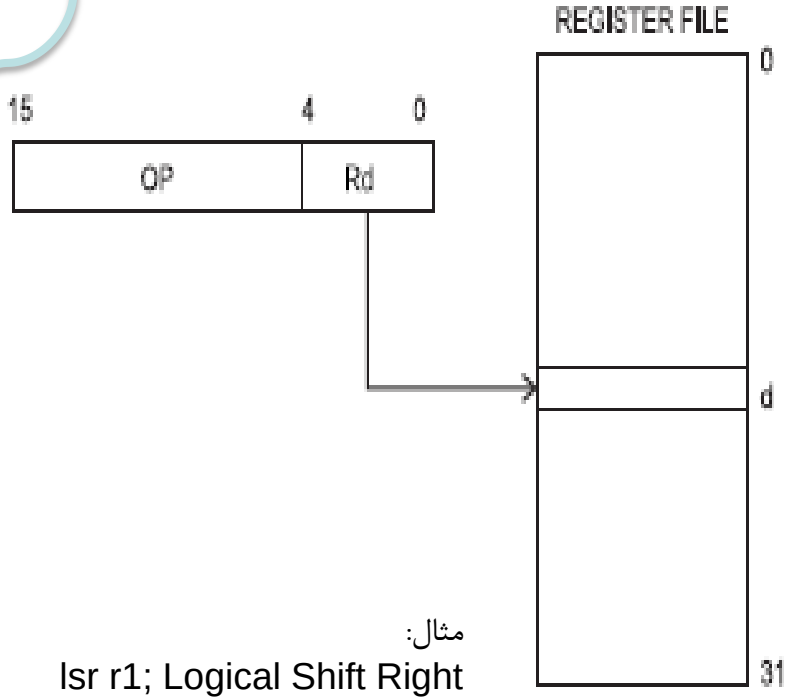
ثبات ها : ثبات های register file اند نه ثبات های I/O

ب) حالت آدرس دهی مستقیم توسط ثبات (با دو ثبات Rd و Rr)



مثال:

Mov r1, r2;



مثال:

lsr r1; Logical Shift Right

اف) حالت آدرس‌دهی مستقیم توسط ثبات (با تنها یک ثبات Rd)

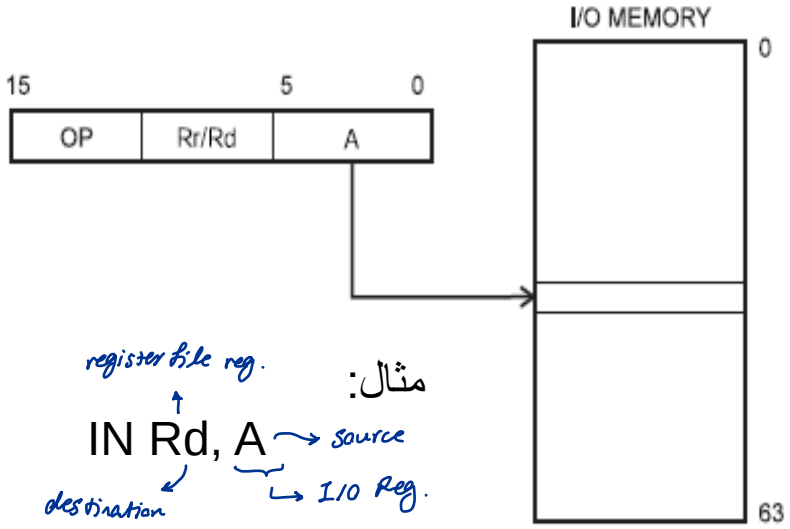
حالت‌های آدرس‌دهی

الف) حالت آدرس‌دهی مستقیم توسط ثبات (با تنها یک ثبات Rd). در این مود، عملوند دستورالعمل در ثبات Rd قرار می‌گیرد.

ب) حالت آدرس‌دهی مستقیم توسط ثبات (با دو ثبات Rd و Rr). در این مود، عملوندها در ثبات های Rd و Rr قرار می‌گیرند. نتیجه در ثبات Rd گذاشته می‌شود

حالت‌های آدرس‌دهی

ج) حالت آدرس دهی مستقیم I/O

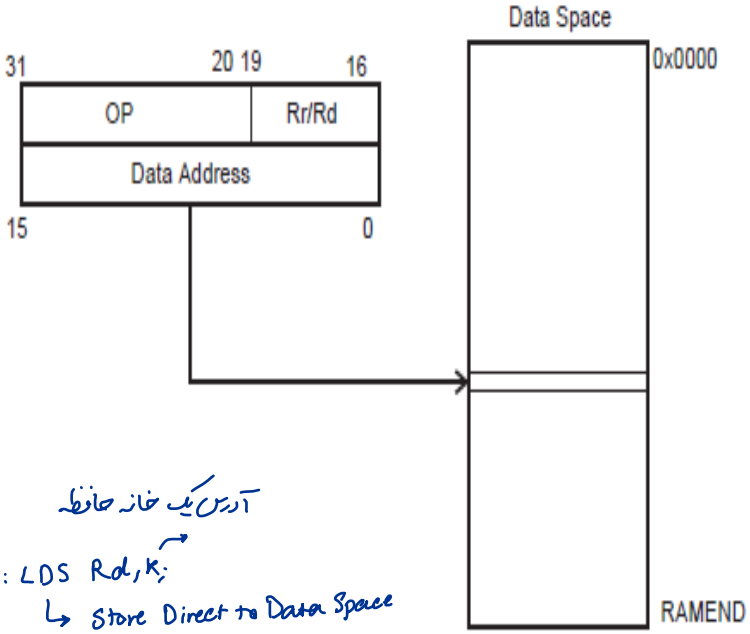


مثال:

IN Rd, A

ج) حالت آدرس‌دهی مستقیم ورود/خروجی. در این مود، آدرس عملوند در ۶ بیت از کلمه دستورالعمل قرار می‌گیرد. A آدرس ثبات مبداء یا مقصد است.

د) حالت آدرس دهی مستقیم داده

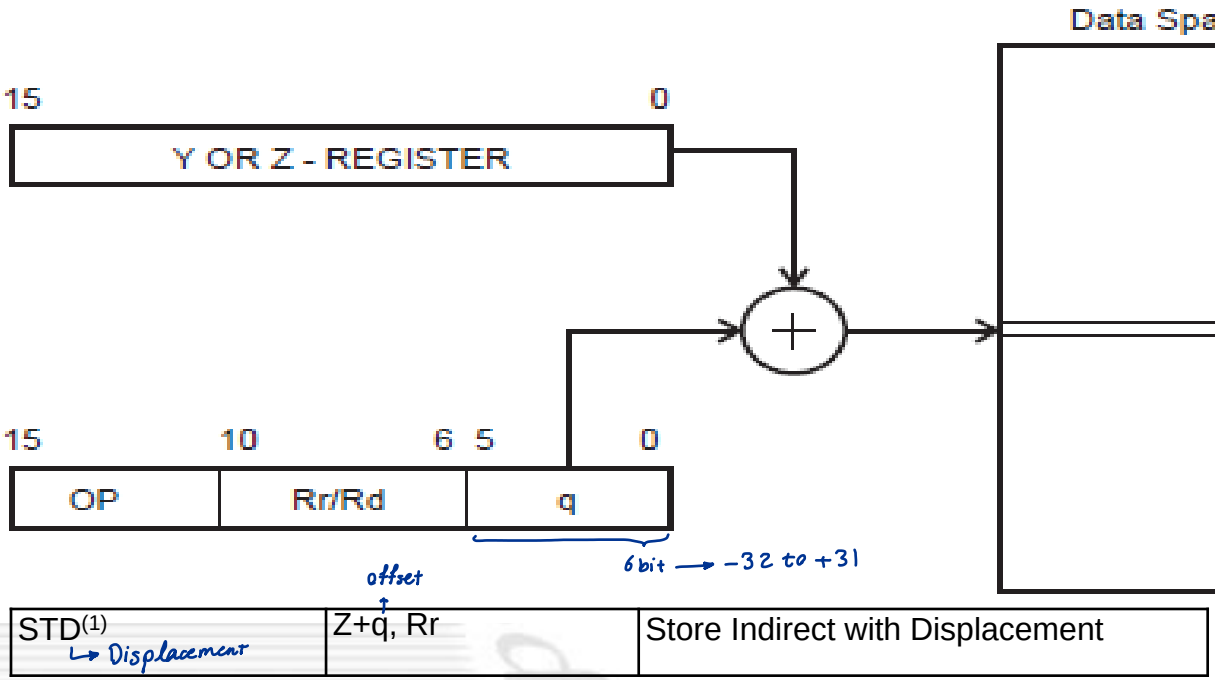


e.g.: LDS Rd, k;

e.g.: STD k, Rr;

د) حالت آدرس‌دهی مستقیم داده. در این مود، یک آدرس داده ۱۶ بیتی در ۱۶ بیت کم ارزش یک دستورالعمل دو کلمه‌ای قرار دارد. Rd/Rr ثبات مبداء یا مقصد را تعریف می‌کنند.

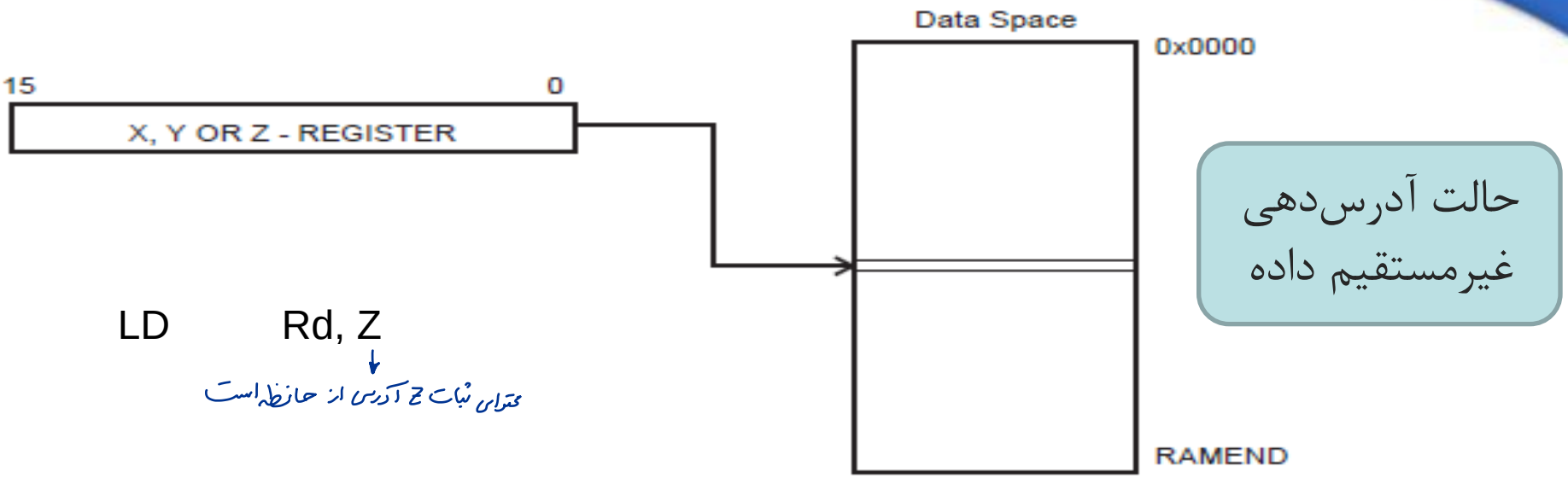
حالت‌های آدرس‌دهی



حالت آدرس دهی غیرمستقیم داده با جابجایی

• در این مود، آدرس عملوند نتیجه حاصل جمع محتوای ثبات‌های Y یا Z با ۶ بیت آدرس موجود در کلمه دستورالعمل می‌باشد. Rd و Rr ثبات‌های مبدا و مقصد را مشخص می‌کنند

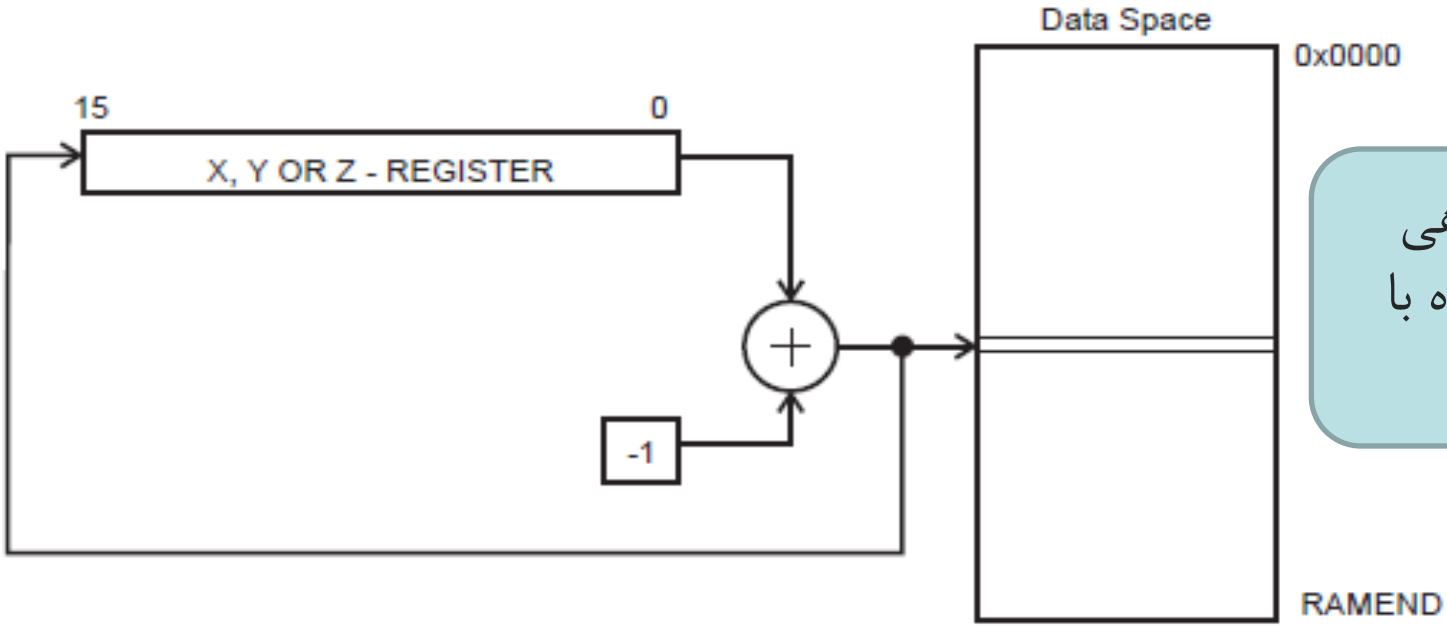
حالت‌های آدرس دهی



حالت آدرس‌دهی غیرمستقیم داده

- در این مود، آدرس عملوند، محتوای یکی از ثبات های X،Y یا Z است.
- آدرس دهی غیرمستقیم ثبات زیرمجموعه‌ای از حالت آدرس‌دهی غیرمستقیم داده است، چرا که فضای داده از ۰ تا ۳۱ همان فایل ثبات را تشکیل می‌دهد.
- در میکروکنترلرهای AVR بدون حافظه SRAM، حالت آدرس‌دهی غیرمستقیم داده را حالت آدرس‌دهی غیرمستقیم ثبات می‌نامند.

حالت‌های آدرس دهی

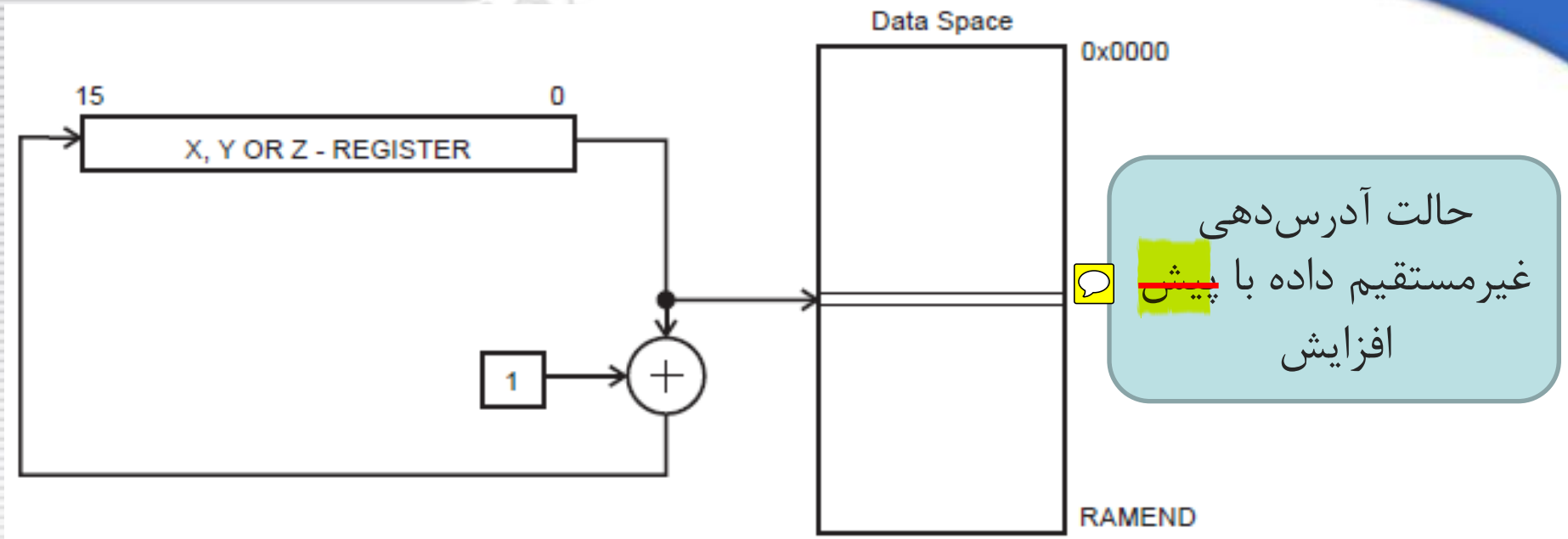


حالت آدرس‌دهی غیرمستقیم داده با پیش کاهش

• در این حالت ثبات X، Yیا Z قبل از عملیات، یک واحد کاهش می یابد. آدرس عملوند کاهش یافته محتوای یکی از ثبات‌های X، Y یا Z است.

ST(2)	-Z, Rr	Store Indirect and Pre-Decrement

حالت‌های آدرس دهی

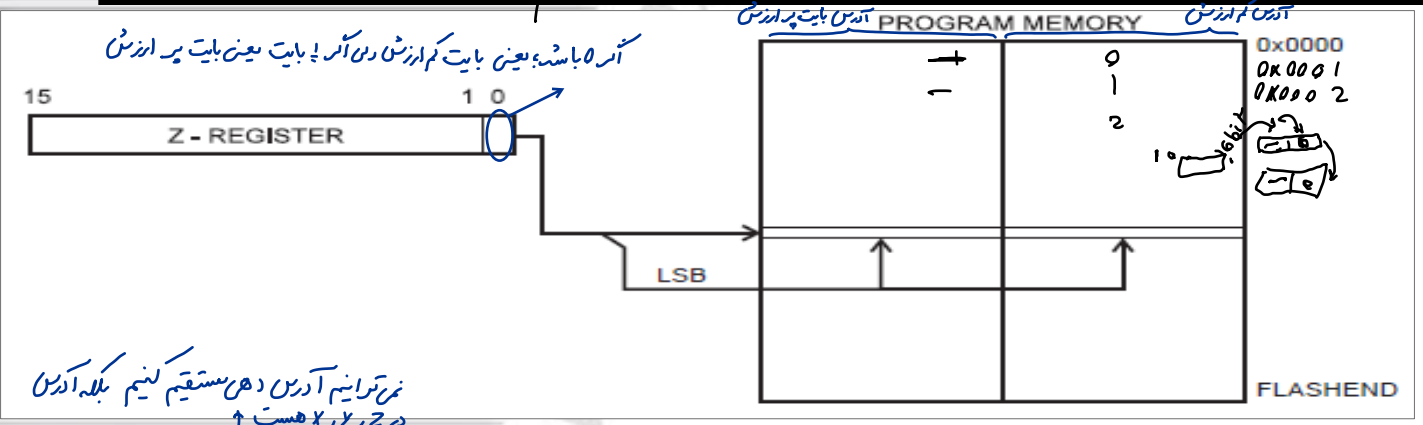


• در این حالت ثبات X، Y یا Z قبل از عملیات، یک واحد افزایش می‌یابد. آدرس عملوند افزایش یافته محتوای یکی از ثبات‌های X، Y یا Z است.

برای اینکه در کلاک انجام شود از reg های موقت استفاده می شود

ST(2)	Z+, Rr	Store Indirect and Post-Increment

حالت‌های آدرس دهی



حالت آدرس‌دهی حافظه برنامه با آدرس ثابت

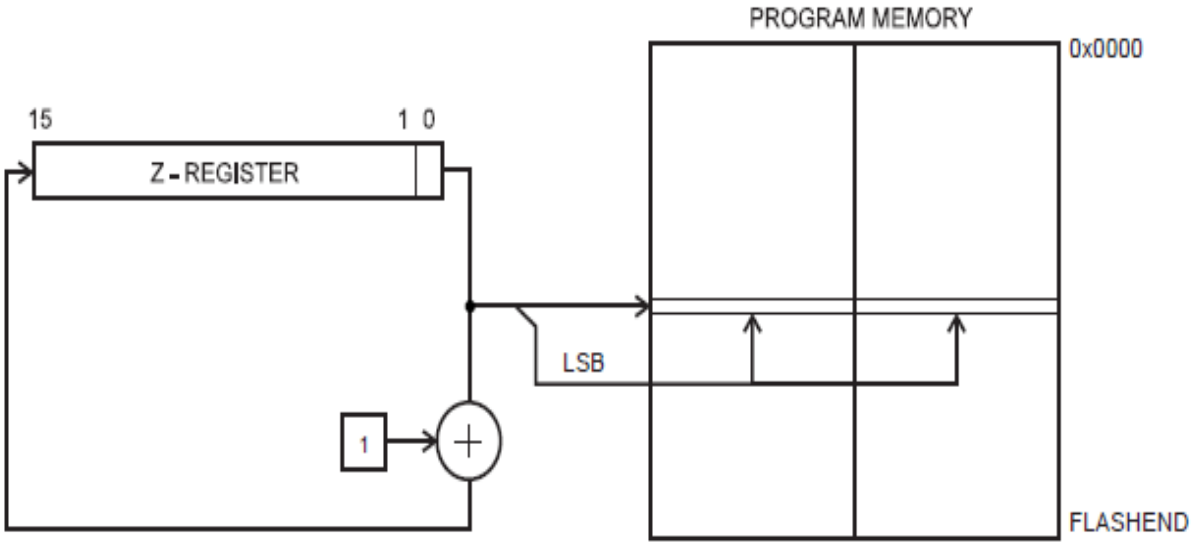
LPM Rd, Z; Load Program memory; Rd ← (Z)

- در این مود، آدرس بایتی معینی در حافظه برنامه توسط محتوای ثبات Z تعریف می شود.
- ۱۵ بیت پر ارزشِ آدرسِ کلمه را انتخاب می کنند.
- برای LPM، اگر بیت کم ارزش 0 شود، بایت پائین انتخاب و اگر 1 باشد، بایت پرارزش انتخاب می شود. اگر ELPM بکار رود، ثبات RAMPZ برای توسعه ثبات Z بکار می رود.

ELPM Rd, Z; Extended Load Program memory; Rd ← (RAMPZ: Z)

SPM z+; Store Program Memory and Post- Increment by 2; (RAMPZ:Z) ← R1:R0 Z ←Z+2

حالت‌های آدرس دهی

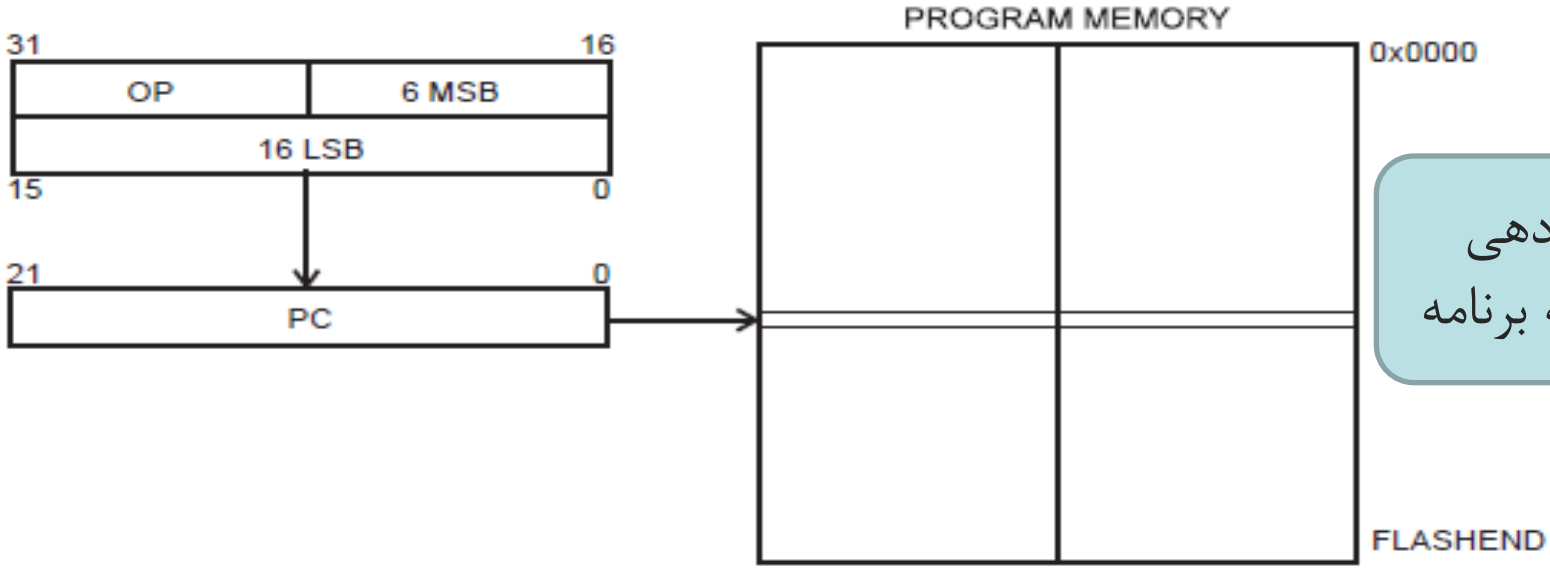


Post Increament

حالت آدرس‌دهی حافظه برنامه با پس‌افزایش با استفاده از LPM Z+ و ELPM Z+

- در این حالت آدرس بایتی ثابت توسط محتوای ثبات Z تعریف می‌شود.
- ۱۵ بیت پرارزش آدرس کلمه را انتخاب می‌کنند.
- اگر بیت کم ارزش صفر شود، بایت پائین انتخاب و اگر یک باشد، بایت پرارزش انتخاب می‌شود.
- اگر ELPM Z+ بکار رود، ثبات RAMPZ برای توسعه ثبات Z بکار می‌رود.

حالت‌های آدرس‌دهی



حالت آدرس‌دهی
مستقیم حافظه برنامه

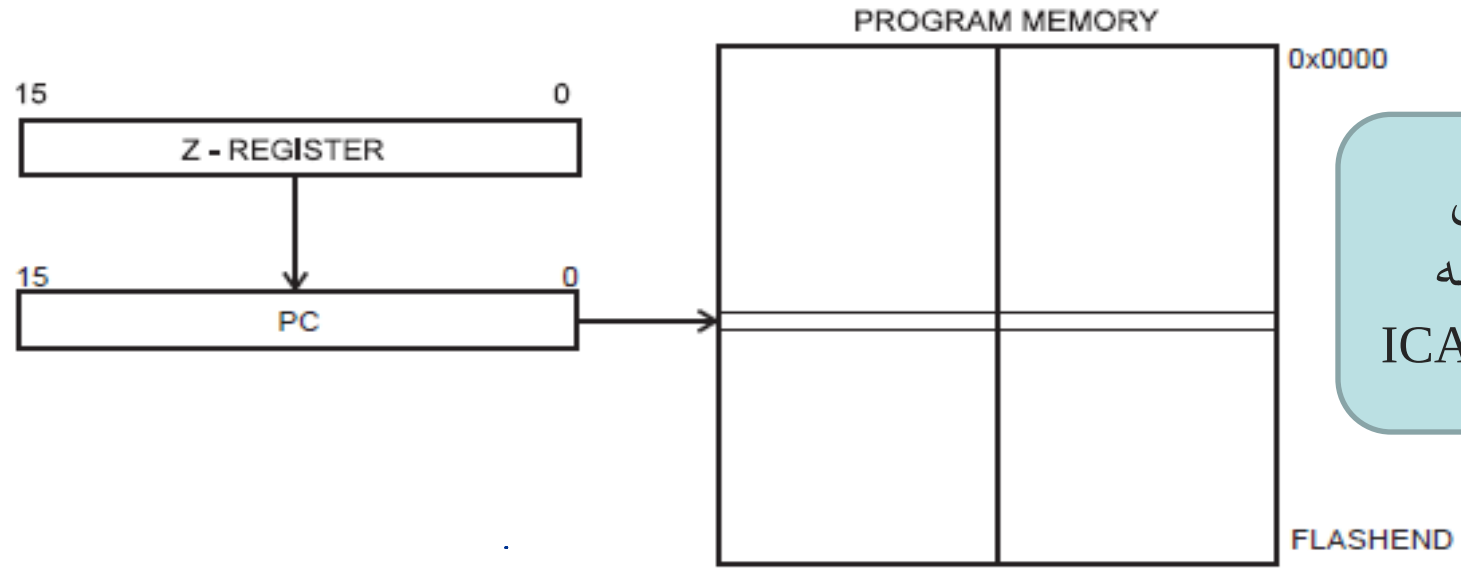
آدرس یک خانه حافظه

e.g.: CALL k → فضای آدرس‌دهی بسیار بزرگ → 22 بیت است

pc ← k

* نقل و انتقال داده انجام نمی‌شود

حالت‌های آدرس دهی



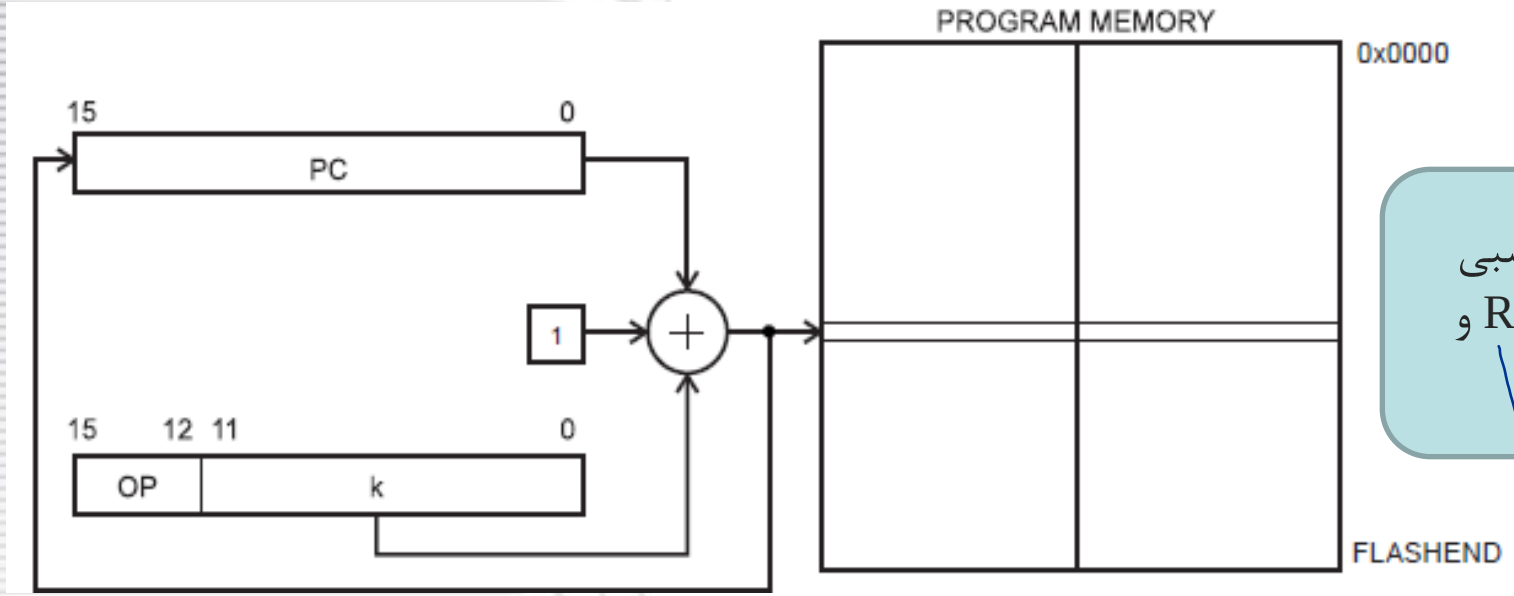
حالت آدرس‌دهی غیرمستقیم حافظه برنامه، IJMP و ICALL

بصورت پیش فرض

- در این حالت اجرای برنامه از آدرس موجود در ثبات Z ادامه می‌یابد (یعنی اینکه مقدار موجود در ثبات Z در PC قرار می‌گیرد).

فقط ۱۶ بیت اول Z مورد بقیه‌اش صفر می‌رود ← حداکثر 64KB آدرس دهی میکند

حالت‌های آدرس‌دهی



حالت آدرس‌دهی نسبی حافظه برنامه، RJMP و RCALL

Relative

• در این حالت اجرای برنامه از آدرس PC+K+1 ادامه می‌یابد. آدرس نسبی k از 2048- تا 2048 تغییر می‌نماید.

e.g.: RCALL K

PC ← PC+K+1

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

- دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR شامل گروه‌های زیر هستند:
  - الف- دستورالعمل‌های حسابی و منطقی
  - ب- دستورالعمل‌های انشعاب
  - ج- دستورالعمل‌های انتقال داده
  - د- دستورالعمل‌های بیتی و تست بیت
  - ه- دستورالعمل‌های کنترل میکروکنترلر

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

توجه: معانی اعداد نوشته شده در داخل پرانتزها در ستون‌های ۶ و ۷ در اسلایدهای بعد به شرح زیر است:

۱) این دستورالعمل ها در تمام میکروکنترلرهای ۸ بیتی AVR وجود ندارد.
۲) تمام گونه‌های این دستورالعمل در تمام میکروکنترلرهای ۸ بیتی AVR فراهم نیستند
۳) زمان‌های چرخه برای تمام دسترسی‌های به حافظه داده فرض بر دسترسی به حافظه RAM داخلی دارند و برای دسترسی از طریق واسطه RAM خارجی معتبر نیستند.
۴) در صورت دسترسی به SRAM داخلی یک چرخه ساعت دیگر اضافه شود.
۵) تعداد چرخه‌های ساعت برای میکروکنترلر ATtiny10.

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks-XMEGA
ADD	Rd, Rr	Add without Carry	Rd ← Rd + Rr	Z,C,N,V,S,H	1	
ADC	Rd, Rr	Add with Carry	Rd ← Rd + Rr + C	Z,C,N,V,S,H	1	
ADIW(1)	Rd, K	Add Immediate to Word	Rd + 1:Rd ← Rd + 1:Rd + K	Z,C,N,V,S	2	
SUB	Rd, Rr	Subtract without Carry	Rd ← Rd - Rr	Z,C,N,V,S,H	1	
SUBI	Rd, K	Subtract Immediate	Rd ← Rd - K	Z,C,N,V,S,H	1	
SBC	Rd, Rr	Subtract with Carry	Rd ← Rd - Rr - C	Z,C,N,V,S,H	1	
SBCI	Rd, K	Subtract Immediate with Carry	Rd ← Rd - K - C	Z,C,N,V,S,H	1	
SBIW(1)	Rd, K	Subtract Immediate from Word	Rd + 1:Rd ← Rd + 1:Rd - K	Z,C,N,V,S	2	
AND	Rd, Rr	Logical AND	Rd ← Rd • Rr	Z,N,V,S	1	
ANDI	Rd, K	Logical AND with Immediate	Rd ← Rd • K	Z,N,V,S	1	
OR	Rd, Rr	Logical OR	Rd ← Rd v Rr	Z,N,V,S	1	
ORI	Rd, K	Logical OR with Immediate	Rd ← Rd v K	Z,N,V,S	1	
EOR	Rd, Rr	Exclusive OR	Rd ← Rd ⊕ Rr	Z,N,V,S	1	

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

COM	Rd	One's Complement	Rd ← $FF - Rd	Z,C,N,V,S	1	
NEG	Rd	Two's Complement	Rd ← $00 - Rd	Z,C,N,V,S,H	1	
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V,S	1	
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FFh - K)	Z,N,V,S	1	
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V,S	1	
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V,S	1	
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V,S	1	
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V,S	1	
SER	Rd	Set Register	Rd ← $FF	None	1	
MUL(1)	Rd,Rr	Multiply Unsigned	R1:R0 ← Rd x Rr (UU)	Z,C	2	
MULS(1)	Rd,Rr	Multiply Signed	R1:R0 ← Rd x Rr (SS)	Z,C	2	
MULSU(1)	Rd,Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr (SU)	Z,C	2	
FMUL(1)	Rd,Rr	Fractional Multiply Unsigned	R1:R0 ← Rd x Rr<<1 (UU)	Z,C	2	
FMULS(1)	Rd,Rr	Fractional Multiply Signed	R1:R0 ← Rd x Rr<<1 (SS)	Z,C	2	
FMULSU(1)	Rd,Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← Rd x Rr<<1 (SU)	Z,C	2	
DES	K	Data Encryption	if (H = 0) then R15:R0 ←Encrypt(R15:R0, K) else if (H = 1) then R15:R0 ←Decrypt(R15:R0, K)			1/2

DES Instruction Description:

The module is an instruction set extension to the AVR CPU, performing DES iterations. The 64-bit data block (plaintext or cipher text) is placed in the CPU register file, registers R0-R7, where LSB of data is placed in LSB of R0 and MSB of data is placed in MSB of R7. The full 64-bit key (including parity bits) is placed in registers R8-R15, organized in the register file with LSB of key in LSB of R8 and MSB of key in MSB of R15. Executing one DES instruction performs one round in the DES algorithm. Sixteen rounds must be executed in increasing order to form the correct DES cipher text or plaintext. Intermediate results are stored in the register file (R0-R15) after each DES instruction. The instruction's operand (K) determines which round is executed, and the half carry flag (H) determines whether encryption or decryption is performed.

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks-XMEGA
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2	
IJMP(1)		Indirect Jump to (Z)	PC(15:0) ←Z PC(21:16) ← 0	None	2	
EIJMP(1)		Extended Indirect Jump to (Z)	PC(15:0) ← Z PC(21:16) ← EIND	None	2	
JMP(1)	k	Jump	PC ← k	None	3	
RCALL	k	Relative Call Subroutine	PC ← PC + k + 1	None	3/4(3) (5)	2/3(3)
ICALL(1)		Indirect Call to (Z)	PC(15:0) ← Z, PC(21:16) ← 0	None	3/4(3)	2/3(3)
EICALL(1)		Extended Indirect Call to (Z)	PC(15:0)← Z, PC(21:16) ← EIND	None	4(3)	3(3)
CALL(1)	k	call Subroutine	PC ← k	None	4 / 5 (3)	3/4(3)
RET		Subroutine Return	PC ← STACK	None	4 / 5 (3)	
RETI		Interrupt Return	PC ← STACK	I	4 / 5 (3)	
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3	
CP	Rd,Rr	Compare	Rd - Rr	Z,C,N,V,S,H	1	
CPC	Rd,Rr	Compare with Carry	Rd - Rr -C	Z,C,N,V,S,H	1	
CPI	Rd,K	Compare with Immediate	Rd - K	Z,C,N,V,S,H	1	
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b) = 0) PC ← PC + 2 or 3	None	1/2/3	
SBRS	Rr, b	Skip if Bit in Register Set	if (Rr(b) = 1) PC ← PC + 2 or 3	None	1/2/3	
SBIC	A, b	Skip if Bit in I/O Register Cleared	if (I/O(A,b) = 0) PC ← PC + 2 or 3	None	1/2/3	2/3/4
SBIS	A, b	Skip if Bit in I/O Register Set	If (I/O(A,b) =1) PC ← PC + 2 or 3	None	1/2/3	2/3/4
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC ← PC + k + 1	None	½	

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

جک کردن Flag ها :

e.g. : BRNE label ← درست است

BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC ← PC + k + 1	None	½	
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	½	
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	½	
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	½	
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	½	
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	½	
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	½	
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	½	
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	½	
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	½	
BRLT	k	Branch if Less Than, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	½	
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	½	
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	½	
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	½	
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	½	
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	½	
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	½	
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	½	
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	½	

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
MOV	Rd, Rr	Copy Register	Rd ← Rr	None	1	
MOVW[1]	Rd, Rr	Copy Register Pair	Rd+1:Rd ← Rr+1:Rr	None	1	
LDI	Rd, K	Load Immediate	Rd ← K	None	1	
LDS[1]	Rd, k	Load Direct from data space	Rd ← (k)	None	1[5]/2[3]	2[3][4]
LD[2]	Rd, X	Load Indirect	Rd ← (X)	None	1[5]/2[3]	1[3][4]
LD[2]	Rd, X+	Load Indirect and Post-Increment	Rd ← (x) X ← X+1	None	2[3]	1[3][4]
LD[2]	Rd, -X	Load Indirect and Pre-Decrement	X ← X – 1, Rd ← (X)	None	2[3]/3[5]	2[3][4]
LD[2]	Rd, Y	Load Indirect	Rd ← (Y)	None	1[5]/2[3]	1[3][4]
LD[2]	Rd, Y+	Load Indirect and Post-Increment	Rd ← (Y) Y ← Y+1	None	2[3]	1[3][4]
LD [2]	Rd, -Y	Load Indirect and Pre-Decrement Y	Y ← Y-1 Rd ← (Y)	None	2[3]/3[5]	2[3][4]
LDD[1]	Rd, Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2[3]	2[3][4]

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

LD(2)	Rd, Z	Load Indirect	Rd ← (Z)	None	1(5)/2(3)	1(3)(4)
LD(2)	Rd, Z+	Load Indirect and Post-Increment	Rd ← (Z) Z← Z+1	None	2(3)	1(3)(4)
LD(2)	Rd, -Z	Load Indirect and Pre-Decrement	Z← Z-1 Rd ← (Z)	None	2(3)/3(5)	2(3)(4)
LDD(1)	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2(3)	2(3)(4)
STS(1)	k, Rr	Store Direct to Data Space	(k) ← Rd	None	1(5)/2(3)	2(3)
ST(2)	X, Rr	Store Indirect	(X) ← Rr	None	1(5)/2(3)	1(3)
ST(2)	X+, Rr	Store Indirect and Post-Increment	(X) ← Rr X ← X+1	None	1(5)/2(3)	1(3)
ST(2)	-X, Rr	Store Indirect and Pre-Decrement	X ← X-1 (X) ← Rr	None	2(3)	2(3)
ST(2)	Y, Rr	Store Indirect	(Y) ← Rr	None	1(5)/2(3)	1(3)
ST(2)	Y+, Rr	Store Indirect and Post-Increment	(Y) ← Rr Y ← Y+1	None	1(5)/2(3)	1(3)
ST(2)	-Y, Rr	Store Indirect and Pre-Decrement	Y ← Y-1 (Y) ← Rr	None	2(3)	2(3)
STD(1)	Y+q, Rr	Store Indirect with Displacement	(Y + q) ← Rr $0 \le q \le 63$	None	2(3)	2(3)
ST(2)	Z, Rr	Store Indirec	(Z) ← Rr	None	1(5)/2(3)	1(3)

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

ST[2]	Z+, Rr	Store Indirect and Post-Increment	(Z) ← Rr Z ← Z+1	None	1[5]/2[3]	1[3]
ST[2]	-Z, Rr	Store Indirect and Pre-Decrement	Z ← Z-1	None	2[3]	2[3]
STD[1]	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2[3]	2[3]
LP[1] [2]	با حافظه برنامه	Load Program Memory	R0 ← (Z)	None	3	3
LPM[1] [2]	Rd, Z	Load Program Memory	Rd ← (Z)	None	3	3
LPM[1] [2]	Rd, Z+	Load Program Memory and Post- Increment	Rr ← (Z) Z ← Z+1	None	3	3
ELPM[1]		Extended Load Program Memory	R0 ← (RAMPZ:Z)	None	3	
ELPM[1]	Rd, Z	Extended Load Program Memory	Rd ← (RAMPZ:Z)	None	3	
ELPM[1]	Rd, Z+	Extended Load Program Memory and Post-Increment	Rd ← (RAMPZ:Z) Z ← Z+1	None	3	

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

SPM(1)		Store Program Memory	(RAMPZ:Z) ← R1:R0	None	-	-
SPM(1)	Z+	Store Program Memory and Post- Increment by 2	(RAMPZ:Z) ← R1:R0 Z ←Z+2	None	-	-
IN	Rd, A	In From I/O Location	Rd ← I/O(A)	None	1	
OUT	A, Rr	Out To I/O Location	I/O(A) ← Rr	None	1	
PUSH(1)	Rr	Push Register on Stack	STACK ← Rr	None	2	1(3)
POP(1)	Rd	Pop Register from Stack	Rd ← STACK	None	2	2(3)

ارتباط با ثبات های Register file یا I/O

نمی‌توانیم I/O رو push کنیم؛ چکار کنیم؟ ابتدا از دستور IN استفاده کنیم بعد از دستور PUSH ؟

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n) Rd(0) ← 0 C ← Rd(7)	Z,C,N,V,H	1	
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1) Rd(7) ← 0 C ← Rd(0)	Z,C,N,V	1	
ROL	Rd	Rotate Left Through Carry	Rd(0) ← C Rd(n+1) ←Rd(n) C ← Rd(7)	Z,C,N,V,H	1	
ROR	Rd	Rotate Right Through Carry	Rd(7) ← C Rd(n) ←Rd(n+1) C ← Rd(0)	Z,C,N,V	1	
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1	
SWAP	Rd	Swap Nibbles	Rd(3..0) ↔ Rd(7..4)	None	1	
BSET	s ⇝ 3bit	Flag Set	SREG(s) ← 1	SREG(s)	1	
BCLR	A, b	Flag Clear	SREG(s) ← 0	SREG(s)	1(5)/2	1
SBI	A, b	Set Bit in I/O Register	I/O(A, b) ← 1	None	1(5)/2	1
CBI	Rr, b	Clear Bit in I/O Register	I/O(A, b) ← 0	None	1	
BST	Rd, b	Bit Store from Register to T	T ← Rr(b)	T	1	

Stack pointer به آدرس بالای حافظه Stack اشاره می‌کند و بعد به سمت آدرس های کم ارزش تر می رود

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

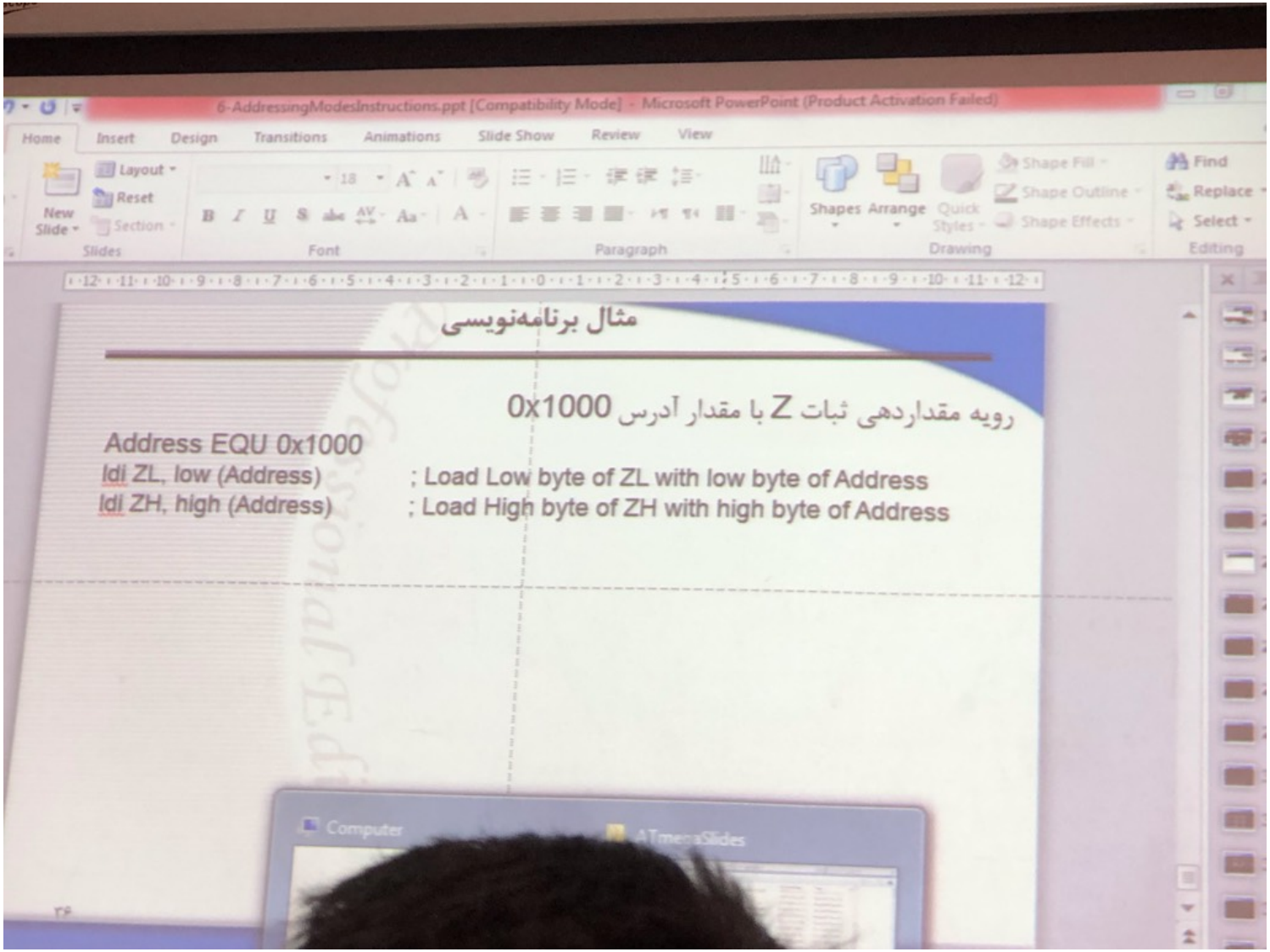
BLD	B, Rd	Bit load from T to Register	Rd(b) ← T	None	1	
SEC		Set Carry	C ← 1	C	1	
CLC		Clear Carry	C ← 0	C	1	
SEN		Set Negative Flag	N ← 1	N	1	
CLN		Clear Negative Flag	N ← 0	N	1	
SEZ		Set Zero Flag	Z ← 1	Z	1	
CLZ		Clear Zero Flag	Z ← 0	Z	1	
SEI		Global Interrupt Disable	I ← 1	I	1	
CLI		Global Interrupt Disable	I ← 0	I	1	
SES		Set Signed Test Flag	S ← 1	S	1	
CLS		Clear Signed Test Flag	S ← 0	S	1	
SEV		Set Two’s Complement Overflow	V ← 1	V	1	
CLV		Clear Two’s Complement Overflow	V ← 0	V	1	
SET		Set T in SREG	T ← 1	T	1	
CLT		Clear T in SREG	T ← 0	T	1	
SEH		Set Half Carry Flag in SREG	H ← 1	H	1	

مجموعه دستورالعمل‌های میکروکنترلرهای ۸ بیتی AVR

Mnemonics	Operands	Description	Operation	Flags	#Clocks	#Clocks XMEGA
BREAK(1)		Break برای debug استفاده می‌شود (همان break point است)	(See specific descr. for BREAK)	None	1	
NOP		No Operation		None	1	
SLEEP		Sleep	(see specific descr. for Sleep)	None	1	
WDR		Sleep	(see specific descr. for WDR)	None	1	

مجموعه دستورالعمل های کنترل میکروکنترلر در میکروکنترلرهای ۸ بیتی AVR

توجه ۱: دستورالعمل BREAK توسط سیستم debug سوار بر تراشه استفاده شده و توسط برنامه های کاربردی قابل استفاده نیست.


6-AddressingModesInstructions.ppt [Compatibility Mode] - Microsoft PowerPoint (Product Activation Failed)
Home
Insert
Design
Transitions
Animations
Slide Show
Review
View
مثال برنامه‌نویسی
رویه مقداردهی ثبات Z با مقدار آدرس 0x1000
Address EQU 0x1000
ldi ZL, low (Address) ; Load Low byte of ZL with low byte of Address
ldi ZH, high (Address) ; Load High byte of ZH with high byte of Address
Computer
ATmegaSlides