به نام خدا

معماری میکروکنترلرهای AVR

AVR Architecture

مقدمه:

تکنولوژی AVR برای اولین بار در سال 1997 توسط شرکت Atmel ارائه شد و بعد از آن جزء تولیدات محبوب این شرکت قرار گرفت.مزیت اصلی این تکنولوژی داشتن هسته RISC همراه با تعداد زیادی ثبات کاری یا Working Register است.این ثباتها به ALU مرتبط هستند و توسط آنها می توان تعداد زیادی ریز دستورالعمل را در مدت زمان یک پالس ساعت اجرا کرد به عبارتی دیگر اجرای هر دستورالعمل یک پالس ساعت لازم دارد در حالیکه اجرای این ریز دستورالعملها در میکروکنترلرهای دیگر در تعداد زیادتری از پالس ساعت اجرا می شوند بنابراین AVR ها می توانند بسیار سریعتر عمل کنند و همچنین کدهای با حجم بالایی را اجرا کنند.به عنوان مثال کارایی یک AVR که با سرعت 4MHz کار می کند با کارایی میکروی PIC با سرعت 16MHz و همچنین میکروی 8051 با سرعت 48MHz برابر است!

ثباتهای AVR

- میکروکنترلرهای ۸ بیتی AVR ۳۲ ثبات ۸ بیتی همه منظوره دارند یعنی r0 تا r31 .
- سه ثبات آدرس شانزده بیتی با نام مستعار X و Y و Z که هر کدام از این سه ثبات دو ثبات از همان ۳۲ ثبات ۸ بیتی هستند یعنی X(r27:r26), Y(r29:r28), Z(r31:r30) .
- یک ثبات ۱۶ بیتی به منظور اشاره گر پشته که در آدرسهای ورودی/خروجی: 0x3e(SPH) و 0x3d(SPL) قرارگرفته اند.همچنین این آدرسها در حافظه داده با آدرسهای 0x5e و 0x5d هستند
- یک ثبات ۸بیتی به منظور سنجش وضعیت یا همان ثبات پرچم با نام SREG .

I	T	H	S	V	N	Z	C

- I : فعال ساز و غیرفعال ساز عمومی وقفه SREG7 یا Global Interrupt Enable/Disable Flag
- T : بیت انتقالی مورد استفاده دستورالعملهای BLD وBST با نام SREG6
- H : Half Carry Flag, SREG5
- S : بیت علامت یا Signed tests Instruction Set, SREG4
- V : سرریزنما برای مکمل دو یا Two's Complement Overflow Indicator, SREG3
- N : بیت منفی یا Negative Flag, SREG2
- Z : بیت صفر یا Zero Flag, SREG1
- C : Carry Flag, SREG0

بر طبق معماری Harvard همراه با حافظه ی کد فلش و حافظه داده استاتیک یا SRAM که حجم حافظه ی کد آنها از 1k تا 128k بایت و حجم حافظه ی داده ی آنها از 32 بایت تا 4k بایت متغیر است یاد آورم شویم که این مقادیر حافظه همراه با گذشت زمان پیوسته در حال افزایش است .

حافظه داده و ثباتهای AVR

- ۳۲ آدرس اول حافظه یعنی (0x0000 تا 0x001f) متعلق به ثباتهای r0 تا r31 هستند.البته در برخی MCU (MicroController Unit) ها برای ثباتها از فضای حافظه ی داده استفاده می شود.
- آدرسهای (0x0020 تا 0x005f) از حافظه ی داده در دسترس آدرسهای ورودی/خروجی (0x00 تا 0x3f) است.
- از آدرس 0x0060 حافظه ی داده به بعد فقط شامل حافظه استاتیک است یعنی SRAM .

دو ثبات برای واحد ریاضی منطقی ALU

تعداد زیادی از دستورالعملهای ALU شامل دو ثبات هستند یکی مقصد یا Destination(Rd) و یکی منبع یا Source(Rr) که نحوه کدگشایی دستورالعمل را در زیر می بینید:

i	i	i	i	i	i	r	d	d	d	d	d	r	r	r	r

بیتهایی که در آن حرف i قرار گرفته دستورالعملند و حرف d بیتهای مقصد هستند و حرف r بیتهای منبع هستند ثبات منبع از بهم پیوستن بیتهای (r9 : r3 : r2 : r1 : r0) و ثبات مقصد از بهم پیوستن بیتهای (d8 : d7 : d6 : d5 : d4) مشخص می شوند همچنین بیتهای باقی مانده (i15 : i14 : i13 : i12 : i11 : i10) خود دستورالعمل را مشخص می کنند.
به عنوان مثال حاصل جمع r17 و r2 که همان Add r17 , r2 است به صورت زیر کدگشایی یا Encode می شود:

0	0	0	0	1	1	0	1	0	0	0	1	0	0	1	0

که در این صورت خروجی AVR-OBJDUMP از این قرار است Hex:

```
0:    12 0d            add     r17, r2
```

توجه داشته باشید که یک کلمه ی ۱۶ بیتی از هشت بیت کم ارزش آن در حافظه ذخیره می شود.

دستورالعملهای Encode شده با قالب دو ثباتی را در جدول زیر مشاهده می کنید:

000111rdddddrrrr	adc Rd, Rr
000111ddddddddd	rol Rd
000011rdddddrrrr	add Rd, Rr
000011ddddddddd	lsl Rd
001000rdddddrrrr	and Rd, Rr
001000ddddddddd	tst Rd
000101rdddddrrrr	cp Rd, Rr
000001rdddddrrrr	cpc Rd, Rr
000100rdddddrrrr	cpse Rd, Rr
001001rdddddrrrr	eor Rd, Rr
001001ddddddddd	clr Rd
001011rdddddrrrr	mov Rd, Rr
100111rdddddrrrr	mul Rd, Rr
001010rdddddrrrr	or Rd, Rr
000010rdddddrrrr	sbc Rd, Rr
000110rdddddrrrr	sub Rd, Rr

توجه: دستور rol Rd همان دستور adc Rd , Rd است و همچنین دستور lsl Rd همان دستورالعمل add Rd , Rd و همچنین tst Rd معادل با دستور and Rd , Rd و همینطور دستور clr Rd معادل است با دستور
. eor Rd , Rd

دستورالعملهای منطقی و ریاضی

دستور العمل	عملیات	تاثیر روی ثبات وضعیت
ADD Rd, Rr	Rd = Rd + Rr	Z,C,N,V,H
ADC Rd, Rr	Rd = Rd + Rr + C	Z,C,N,V,H
ADIW Rdl,K	Rdh:Rdl = Rdh:Rdl + K	Z,C,N,V,S
SUB Rd, Rr	Rd = Rd - Rr	Z,C,N,V,H
SUBI Rd, K	Rd = Rd - K	Z,C,N,V,H
SBC Rd, Rr	Rd = Rd - Rr - C	Z,C,N,V,H
SBCI Rd, K	Rd = Rd - K - C	Z,C,N,V,H
SBIW Rdl,K	Rdh:Rdl = Rdh:Rdl - K	Z,C,N,V,S
AND Rd, Rr	Rd = Rd & Rr	Z,N,V
ANDI Rd, K	Rd = Rd & K	Z,N,V
OR Rd, Rr	Rd = Rd | Rr	Z,N,V
ORI Rd, K	Rd = Rd | K	Z,N,V
EOR Rd, Rr	Rd = Rd ^ Rr	Z,N,V
COM Rd	Rd = $FF - Rd	Z,C,N,V
NEG Rd	Rd = $00 - Rd	Z,C,N,V,H
SBR Rd,K	Rd = Rd | K	Z,N,V
CBR Rd,K	Rd = Rd & ($FF - K)	Z,N,V
INC Rd	Rd = Rd + 1	Z,N,V
DEC Rd	Rd = Rd - 1	Z,N,V
TST Rd	Rd = Rd & Rd	Z,N,V
CLR Rd	Rd = Rd ^ Rd	Z,N,V
SER Rd	Rd = $FF	None
MUL Rd, Rr	R1:R0 = Rd*Rr	Z,C
MULS Rd, Rr	R1:R0 = Rd*Rr	Z,C
MULSU Rd, Rr	R1:R0 = Rd*Rr	Z,C
FMUL Rd, Rr	R1:R0 = (Rd*Rr) << 1	Z,C
FMULS Rd, Rr	R1:R0 = (Rd*Rr) << 1	Z,C
FMULSU Rd, Rr	R1:R0 = (Rd*Rr) << 1	Z,C

یک برنامه به زبان اسمبلی برای ضرب r3 در عدد 7 :

```
.arch atmega32     ; the mcu to simulate

    .text              ; specifies instruction memory
    .org 0x0           ; start off at memory location 0
start:
    clr   r2           ; alias for eor r2, r2
    mov   r4, r3       ; r4 = r3
    add   r3, r3       ; double r3
    add   r4, r3       ; r4 = r3 + 2*r3
    add   r3, r3       ; double r3 again
    add   r4, r3       ; r4 = r3 + 2*r3 + 4*r3 = 7*r3
```

نتیجه زبان ماشین برنامه فوق برای AVR-OBJDUMP –d از این قرار است:

```
00000000 :
   0:   22 24          eor     r2, r2
   2:   43 2c          mov     r4, r3
   4:   33 0c          add     r3, r3
   6:   43 0c          add     r4, r3
   8:   33 0c          add     r3, r3
   a:   43 0c          add     r4, r3
```

Immediate ALU Encoding

کدگشایی به روش فوری به صورت قالب زیر صورت می گیرد:

i	i	i	i	K	K	K	K	d	d	d	d	K	K	K	K

ثباتهای R16 تا R31 توسط چهار بیت نوع d قابل دسترسی هستند.بیتهای نوع I شامل Opcode هستند و هشت بیت نوع k شامل داده های فوری می شوند که از این هشت بیت ، چهار بیت با ارزش بالا یا High nybble آن عبارت است از (k12 , k11 , k10 , k9) و همچنین چهار بیت با ارزش پایین یا Low nibble آن برابر است با (k3 , k2 , k1 , k0) .

به عنوان مثال دستور العمل LDI r27, 0xac به صورت زیر کدگشایی خواهد شد:

1	1	1	0	1	0	1	0	1	0	1	1	1	1	0	0

```
avr-objdump:

0:   bc ea            ldi     r27, 0xAC
```

آپ کد (Opcode) در این مثال برابر است با 1110 که معادل است با 0xe HEX .

در جدول زیر تعدادی دستورالعمل همراه با قالب کدگشایی به روش فوری Immediate آمده است:

0111KKKKddddKKKK	andi Rd, K
0111KKKKddddKKKK	cbr Rd, K
0011KKKKddddKKKK	cpi Rd, K
1110KKKKddddKKKK	ldi Rd, K
0110KKKKddddKKKK	ori Rd, K
0110KKKKddddKKKK	sbr Rd, K
0100KKKKddddKKKK	sbci Rd, K
0101KKKKddddKKKK	subi Rd, K

دستورالعمل cbr rd, k معادل است با andi rd, (~k) .
دستورالعمل sbr rd, k معادل است با ori rd, k .

دستورالعملهای پرشی:

میکروهای AVR می توانند به اندازه 63+ ≥ k ≥ 64- کلمه از محل دستورالعمل جاری با حفظ وابستگی به یکی از هشت بیت ثبات پرچم پرش نمایند به جدول زیر توجه فرمایید:

BRBS s, k	if (SREG(s) = 1) then PC = PC+k+1
BRBC s, k	if (SREG(s) = 0) then PC = PC+k+1
BREQ k	if (Z = 1) then PC = PC+k+1
BRNE k	if (Z = 0) then PC = PC+k+1
BRCS k	if (C = 1) then PC = PC+k+1
BRCC k	if (C = 0) then PC = PC+k+1
BRSH k	if (C = 0) then PC = PC+k+1
BRLO k	if (C = 1) then PC = PC+k+1
BRMI k	if (N = 1) then PC = PC+k+1
BRPL k	if (N = 0) then PC = PC+k+1
BRGE k	if (S = 0) then PC = PC+k+1
BRLT k	if (S = 1) then PC = PC+k+1
BRHS k	if (H = 1) then PC = PC+k+1
BRHC k	if (H = 0) then PC = PC+k+1
BRTS k	if (T = 1) then PC = PC+k+1
BRTC k	if (T = 0) then PC = PC+k+1
BRVS k	if (V = 1) then PC = PC+k+1
BRVC k	if (V = 0) then PC = PC+k+1
BRIE k	if (I = 1) then PC = PC+k+1
BRID k	if (I = 0) then PC = PC+k+1

به برنامه اسمبلی زیر که به if-then else مربوط می شود توجه فرمایید:

```
.arch atmega32
    .text
    .org 0x0          ; start off at memory location 0
start:
    ldi  r16, 0x34    ; r16 = 0x34
    ldi  r17, 0x56    ; r17 = 0x56
    cp   r17, r16     ; r17 - r16 set cc
    breq else         ; if ( z = 1 ) goto 1f
    inc  r2
    nop
    brne endif        ; only for demo
    .org 0x20
else:
    dec  r2
endif:
    sleep
```

نتایج کدهای ترجمه شده این برنامه به زبان ماشین از این قرار است:

```
00000000 :
   0:   04 e3      ldi     r16, 0x34   ; 52
   2:   16 e5      ldi     r17, 0x56   ; 86
   4:   10 17      cp      r17, r16
   6:   61 f0      breq    .+24         ; 0x20
   8:   23 94      inc     r2
   a:   00 00      nop
   c:   51 f4      brne    .+20         ; 0x22
        ...
00000020 :
  20:   2a 94       dec     r2
00000022 :
  22:   88 9        sleep
```

نحوه کدگشایی دستورالعملهای پرشی:

دستورالعمل BRBS s, k به صورت زیر کدگشایی می شود:

1	1	1	1	0	0	k	k	k	k	k	k	k	s	s	s

هفت بیت نوع k مکمل دو آدرس پرش هستند.
سه بیت نوع s انتخابگر یکی از شرایط هشت گانه ثبات پرچم هستند.

به عنوان مثال دستورالعمل BRBC s, k به شکل زیر کدگشایی می شوند:

1	1	1	1	0	1	k	k	k	k	k	k	k	s	s	s

دستورالعملهای انتقال داده:

MOV Rd, Rr	Rd = Rr
MOVW Rd, Rr	Rd+1:Rd = Rr+1:Rr
LDI Rd, K	Rd = K
LD Rd, X	Rd = (X)
LD Rd, X+	Rd = (X), X = X + 1
LD Rd, - X	X = X - 1, Rd = (X)
LD Rd, Y	Rd = (Y)
LD Rd, Y+	Rd = (Y), Y = Y + 1
LD Rd, - Y	Y = Y - 1, Rd = (Y)
LDD Rd,Y+q	Rd = (Y + q)
LD Rd, Z	Rd = (Z)
LD Rd, Z+	Rd = (Z), Z = Z+1
LD Rd, -Z	Z = Z - 1, Rd = (Z)
LDD Rd, Z+q	Rd = (Z + q)
LDS Rd, k	Rd = (k)
ST X, Rr	(X) = Rr
ST X+, Rr	(X) = Rr, X X + 1
ST - X, Rr	X = X - 1, (X) Rr
ST Y, Rr	(Y) = Rr
ST Y+, Rr	(Y) = Rr, Y = Y + 1
ST - Y, Rr	None 2
STD Y+q,Rr	(Y + q) = Rr
ST Z, Rr	(Z) = Rr
ST Z+, Rr	(Z) = Rr, Z = Z + 1
ST -Z, Rr	Z = Z - 1, (Z) = Rr
STD Z+q,Rr	(Z + q) = Rr
STS k, Rr	(k) = Rr
LPM	R0 = (Z)
LPM Rd, Z	Rd = (Z)
LPM Rd, Z+	Rd = (Z), Z = Z+1
SPM	(Z) = R1:R0
IN Rd, P	Rd = P
OUT P, Rr	P = Rr
PUSH Rr	Stack = Rr
POP Rd	Rd = Stack

نگارنده : فرشید سفیدگران
کارشناسی کامپیوتر سخت افزار
خرداد ۱۳۸۳
Sefidgaran@gmail.com
http://Sefidgaran.blogfa.com