به نام خدا

آشنایی با زبان اسمبلی AVR

آدرس دهی بیتی

Dr. Aref Karimiafshar
A.karimiafshar@ec.iut.ac.ir

آدرس دهی بیتی

- Many µP allow programs to access registers in byte size only!
  - To check a single bit
    - Read the entire byte
    - Manipulate the byte with logic instructions

- This is not the case with AVR
  - Bit-addressability options of AVR family

SBR – Set Bits in Register

Manipulating bits of GPR

- Sets specified bits in register Rd.

Operation:

(i) $Rd \leftarrow Rd \vee K$

Syntax:	Operands:	Program Counter:
(i) SBR Rd,K	$16 \le d \le 31$, $0 \le K \le 255$	$PC \leftarrow PC + 1$

16-bit Opcode:

0110	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

Words 1 (2 bytes)

Cycles 1

CBR – Clear Bits in Register

- Clears the specified bits in register Rd.

Operation:

(i) Rd ← Rd • ($FF - K)

	Syntax:	Operands:	Program Counter:
(i)	CBR Rd,K	$16 \leq d \leq 31$, $0 \leq K \leq 255$	PC ← PC + 1

16-bit Opcode:

0111	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	⇔	0	⇔	⇔	–

Words 1 (2 bytes)

Cycles 1

BST – Bit Store from Bit in Register to T Flag in SREG

- Stores bit b from Rd to the T Flag in SREG (Status Register). T → Temporary

Operation:

(i) T ← Rd(b)

	Syntax:	Operands:	Program Counter:
(i)	BST Rd,b	$0 \le d \le 31, 0 \le b \le 7$	PC ← PC + 1

16-bit Opcode:

1111	101d	dddd	0bbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	⇔	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1

BLD – Bit Load from the T Flag in SREG to a Bit in Register

- Copies the T Flag in the SREG (Status Register) to bit b in register Rd.

	Operation:		
(i)	Rd(b) ← T		
	Syntax:	Operands:	Program Counter:
(i)	BLD Rd,b	$0 \le d \le 31$, $0 \le b \le 7$	PC ← PC + 1

16 bit Opcode:

1111	100d	dddd	0bbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1

SBRC – Skip if Bit in Register is Cleared

- This instruction tests a single bit in a register and skips the next instruction if the bit is cleared.

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

	Syntax:	Operands:	Program Counter:
(i)	SBRC Rr,b	0 ≤ r ≤ 31, 0 ≤ b ≤ 7	PC ← PC + 1, Condition false - no skip PC ← PC + 2, Skip a one word instruction PC ← PC + 3, Skip a two word instruction

Words 1 (2 bytes)

Cycles 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

16-bit Opcode:

1111	110r	rrrr	0bbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

SBRS – Skip if Bit in Register is Set

- This instruction tests a single bit in a register and skips the next instruction if the bit is set.

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

	Syntax:	Operands:	Program Counter:
(i)	SBRS Rr,b	$0 \le r \le 31$, $0 \le b \le 7$	PC ← PC + 1, Condition false - no skip
			PC ← PC + 2, Skip a one word instruction
			PC ← PC + 3, Skip a two word instruction

Words 1 (2 bytes)

Cycles 1 if condition is false (no skip)
2 if condition is true (skip is executed) and the instruction skipped is 1 word
3 if condition is true (skip is executed) and the instruction skipped is 2 words

16-bit Opcode:

1111	111r	rrrr	0bbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Manipulating bits of IOR

SBI – Set Bit in I/O Register

- Sets a specified bit in an I/O Register. This instruction operates on the lower 32 I/O Registers addresses 0-31.

(i) $I/O(A,b) \leftarrow 1$

	Syntax:	Operands:	Program Counter:
(i)	SBI A,b	$0 \le A \le 31$, $0 \le b \le 7$	$PC \leftarrow PC + 1$

16-bit Opcode:

1001	1010	AAAA	Abbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	2

Manipulating bits of IOR

CBI – Clear Bit in I/O Register

- Clears a specified bit in an I/O register. This instruction operates on the lower 32 I/O registers addresses 0-31.

(i) I/O(A,b) ← 0

	Syntax:	Operands:	Program Counter:
(i)	CBI A,b	$0 \le A \le 31$, $0 \le b \le 7$	PC ← PC + 1

16-bit Opcode:

1001	1000	AAAA	Abbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	2

Manipulating bits of IOR

SBIC – Skip if Bit in I/O Register is Cleared

- This instruction tests a single bit in an I/O Register and skips the next instruction if the bit is cleared. This instruction operates on the lower 32 I/O Registers addresses 0-31.

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

	Syntax:	Operands:	Program Counter:
(i)	SBIC A,b	$0 \le A \le 31$, $0 \le b \le 7$	PC ← PC + 1, Condition false - no skip
			PC ← PC + 2, Skip a one word instruction
			PC ← PC + 3, Skip a two word instruction

Words 1 (2 bytes)

Cycles 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

16-bit Opcode:

1001	1001	AAAA	Abbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

SBIS – Skip if Bit in I/O Register is Set

- This instruction tests a single bit in an I/O Register and skips the next instruction if the bit is set. This instruction operates on the lower 32 I/O Registers – addresses 0-31.

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

	Syntax:	Operands:	Program Counter:
(i)	SBIS A,b	$0 \le A \le 31$, $0 \le b \le 7$	PC ← PC + 1, Condition false - no skip
			PC ← PC + 2, Skip a one word instruction
			PC ← PC + 3, Skip a two word instruction

Words 1 (2 bytes)

Cycles 1 if condition is false (no skip)

2 if condition is true (skip is executed) and the instruction skipped is 1 word

3 if condition is true (skip is executed) and the instruction skipped is 2 words

16-bit Opcode:

1001	1011	AAAA	Abbb

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

آدرس دهی بیتی

Single-Bit (Bit-Oriented) Instructions for AVR

Instruction		Function
SBI	A,b	Set Bit b in I/O register
CBI	A,b	Clear Bit b in I/O register
SBIC	A,b	Skip next instruction if Bit b in I/O register is Cleared
SBIS	A,b	Skip next instruction if Bit b in I/O register is Set
BST	Rr,b	Bit store from register Rr to T
BLD	Rd,b	Bit load from T to Rd
SBRC	Rr,b	Skip next instruction if Bit b in Register is Cleared
SBRS	Rr,b	Skip next instruction if Bit b in Register is Set
BRBS	s,k	Branch if Bit s in status register is Set
BRBC	s,k	Branch if Bit s in status register is Cleared

BRBC – Branch if Bit in SREG is Cleared

- Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is cleared. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). Parameter k is the offset from PC and is represented in two's complement form.

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

	Syntax:	Operands:	Program Counter:
(i)	BRBC s,k	$0 \le s \le 7$, $-64 \le k \le +63$	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	01kk	kkkk	ksss

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

BRBS – Branch if Bit in SREG is Set

- Conditional relative branch. Tests a single bit in SREG and branches relatively to PC if the bit is set. This instruction branches relatively to PC in either direction (PC - 63 ≤ destination ≤ PC + 64). Parameter k is the offset from PC and is represented in two's complement form.

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

	Syntax:	Operands:	Program Counter:
(i)	BRBS s,k	0 ≤ s ≤ 7, -64 ≤ k ≤ +63	PC ← PC + k + 1 PC ← PC + 1, if condition is false

16-bit Opcode:

1111	00kk	kkkk	ksss

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Words	1 (2 bytes)
Cycles	1 if condition is false
	2 if condition is true

آدرس دهی بیتی

AVR Conditional Branch (Jump) Instructions

Instruction	Action	Instruction	Action
BRCS	Branch if C = 1	BRCC	Branch if C = 0
BRLO	Branch if C = 1	BRSH	Branch if C = 0
BREQ	Branch if Z = 1	BRNE	Branch if Z = 0
BRMI	Branch if N = 1	BRPL	Branch if N = 0
BRVS	Branch if V = 1	BRVC	Branch if V = 0
BRLT	Branch if S = 1	BRGE	Branch if S = 0
BRHS	Branch if H = 1	BRHC	Branch if H = 0
BRTS	Branch if T = 1	BRTC	Branch if T = 0
BRIE	Branch if I = 1	BRID	Branch if I = 0

BSET – Bit Set in SREG

- Sets a single Flag or bit in SREG.

(i) SREG(s) ← 1

	Syntax:	Operands:	Program Counter:
(i)	BSET s	$0 \le s \le 7$	PC ← PC + 1

16-bit Opcode:

1001	0100	0sss	1000

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
⇔	⇔	⇔	⇔	⇔	⇔	⇔	⇔

Words 1 (2 bytes)

Cycles 1

BCLR – Bit Clear in SREG

- Clears a single Flag in SREG.

(i) SREG(s) ← 0

	Syntax:	Operands:	Program Counter:
(i)	BCLR s	$0 \leq s \leq 7$	PC ← PC + 1

16-bit Opcode:

1001	0100	1sss	1000

Status Register (SREG) and Boolean Formula

I	T	H	S	V	N	Z	C
⇔	⇔	⇔	⇔	⇔	⇔	⇔	⇔

Words 1 (2 bytes)

Cycles 1

آدرس دهی بیتی

Manipulating the Flags of the Status Register

Instruction	Action		Instruction	Action	
SEC	Set Carry	C = 1	CLC	Clear Carry	C = 0
SEZ	Set Zero	Z = 1	CLZ	Clear Zero	Z = 0
SEN	Set Negative	N = 1	CLN	Clear Negative	N = 0
SEV	Set overflow	V = 1	CLV	Clear overflow	V = 0
SES	Set Sign	S = 1	CLS	Clear Sign	S = 0
SEH	Set Half carry	H = 1	CLH	Clear Half carry	H = 0
SET	Set Temporary	T = 1	CLT	Clear Temporary	T = 0
SEI	Set Interrupt	I = 1	CLI	Clear Interrupt	I = 0

آدرس دهی بیتی

Manipulating bits of RAM

- The internal RAM is not bit-addressable
- In order to manipulate a bit of internal RAM location
  - Bring it into GPR and manipulate

Write a program to see if the internal RAM location $195 contains an even value. If so, send it to Port B. If not, make it even and then send it to Port B.

```
.EQU   MYREG = 0x195            ;set aside loc 0x195
       LDI    R16,0xFF
       OUT    DDRB,R16          ;make Port B an output port
AGAIN:LDS    R16,MYREG
       SBRS   R16,0             ;bit test D0, skip if set
       RJMP   OVER              ;it must be LOW
       CBR    R16,0b00000001    ;clear bit D0 = 0
OVER:  OUT    PORTB,R16         ;copy it to Port B
       JMP    AGAIN             ;we can use RJMP too
```

ماکروها

- A group of instructions performs a task
  - Used repeatedly
- Does not make sense to rewrite this code every it is needed

⇩

Macros

- Macros allow
  - To write the task once only, and to invoke it whenever it is needed
  - Reduce the time to write code and possibility of errors

تعریف ماکروها

- Macro definition

.MACRO name → Beginning of macro definition

... → Body

.ENDMACRO → End of macro definition

- A macro can take up to 10 parameters
  - Parameters can be referred to as @0 to @9
- After the macro has been written, it can be invoked by its name

ماکروها نحوه استفاده

For example, moving immediate data into I/O register data RAM is a widely used service, but there is no instruction for that. We can use a macro to do the job as shown in the following code:

```
.MACRO      LOADIO
      LDI   R20,@1
      OUT   @0,R20
.ENDMACRO
```

The following are three examples of how to use the above macro:

```
1. LOADIO   PORTA, 0x20      ;send value 0x20 to PORTA

2. .EQU     VAL_1 = 0xFF
   LOADIO   DDRC, VAL_1
3. LOADIO   SPL, 0x55        ;send value $55 to SPL
```

ماکروها نحوه استفاده

Assume that several macros are used in every program. Must they be rewritten every time? The answer is no, if the concept of the .INCLUDE directive is known. The .INCLUDE directive allows a programmer to write macros and save them in a file, and later bring them into any program file. For example, assume that the following widely used macros were written and then saved under the filename "MYMACRO1.MAC".

```
 toggling Port B using macros
.INCLUDE "M32DEF.INC"
.INCLUDE "MYMACRO1.MAC" ;get macros from macro file
;----------------------------program starts
      .ORG  0
      LOADIO   DDRB,0xFF
L1:   LOADIO   PORTB,0x55
      DELAY    R18,0x70
      LOADIO   PORTB,0xAA
      DELAY    R18,0x70
      RJMP     L1
```

پایان

موفق و پیروز باشید