8051 µC (Microcontroller)

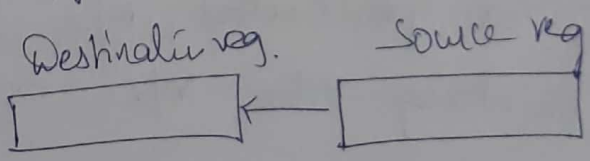
Addressing Modes

→ Addressing mode refers to the way in which the operand of an instruction is specified.

(1) Register addressing
(2) Direct byte addressing
(3) Register indirect addressing
(4) Immediate addressing.
(5) ~~Register specific~~
(5) Index add
(6) stack addressing mode.

(1) Register addressing

8051 access eight 'working registers' (R_0-R_7). The programmer can select a register bank by modifying the bits 4 & 3 in the PSW.



Eg:

MOV A, R_3

This addressing mode involves the use of registers to hold the data to be manipulated.

Ex MOV PSW, # 00001000 B
MOV A, R3
ADD A, R4
} to add the contents of reg R3 & R4 from bank 2.

Direct addressing

In this type the operand is specified with in instruction as a direct m/r location.

Ex MOV A, 50 H : load byte from add 50 H into A.

Register Indirect addressing

In this addressing mode R0 & R1 of the each register bank can be used as an index (or) pointer reg. R0 & R1 point to the contents in the RAM. The instruction with indirect addressing uses the @ sign

In this type the operand is specified with in instruction as a register that points a particular m/r location (indirectly pointing the m/r location).

Ex MOV A, @ R0 : load the contents pointed by R0 in A.

Immediate Addressing mode

In this type, the operand is specified with in instruction as an immediate data sign '#' indicates it is a immediate add mode.

Eg: MOV A, #52

Index add

only program mem can be accessed in the index addressing. Either the DPTR or PC can be used as an index register.

Eg: Read data from the program mem

MOV CA, @ A + DPTR

This instruction adds the unsigned 8-bit & accumulator contents into to 16-bit data pointer & uses the sum as an address from which byte to be moved into accumulator.

Stack addressing

It is a type of direct addressing mode in which stack instructions PUSH & POP are used.

PUSH 04 : PUSH R4 on to stack

Instruction set of 8051

- → Data transfer instruction
- → Arithmatic Instruction
- → logical Instruction
- → boolean Instruction
- → branching instruction

(1) Data transfer group

MOV <dest>, <source>

(1) MOV A, Rn (copy the content of register bank to accumulator)
Ex MOV A, R0

(2) MOV Rn, A
Ex MOV R0, A.

(3) MOV A, direct (copy the contents of add specified with instruction to A.
Ex MOV A, 30H.

(4) MOV direct, A
Ex MOV 20H, A.

(5) MOV A, @Ri (copy the contents of the add in Ri to A)

MOV A, @R1 : This instruction copies the content of mem location whose add is specified in Reg R1 from selected reg bank.

(6) MOV A, #data
load the given data in to A.
Eg MOV A, #30H.

(7) MOV Rn, direct
Eg MOV R1, 40H
copy the contents of 40H mem add in to R1 reg.

(8) MOV Rn, #data
MOV R2, #20H

(9) MOV direct, A.
Eg MOV 20H, A

(10) MOV direct, Rn
Eg MOV 30H, R2

(11) MOV direct, direct
MOV 20H, 30H

(12) MOV direct, @Ri
Eg MOV 20H, @R3
copy the content of the mem locate whose add is given by reg Ri (i.e R3) in to mem locatin whose address 20H.

(13) MOV direct, # data

Eg MOV 30H, # 12H

(14) MOV @Ri, A

Eg MOV @ R1, A

copies the content of A to mem loc'n whose add is given by reg R1

(15) MOV @ Ri, direct

Eg MOV @R2, 30H.

(16) MOV @ Ri, # data

Eg MOV @R2, # 30H.

(17) MOV DPTR, # data 16

Eg MOV DPTR, # 1234 H

the data pointer is loaded with the 16-bit.

the second by (DPH) is the high-order byte, while the 3rd byte (DPL) holds low-order byte.

Eg this instruction loads the value 1234 H in to data pointer

DPH ← 12 H

DPL ← 34 H.

Instruction to access external data mem

1) MOVX A, @Ri
copy the contents of the external address in Ri to A.

Ex: MOVX A, @R0, this instruction copies data from the 8-bit address in R0 to A.

2) MOVX A, @DPTR
copies data from the 16-bit address in DPTR to A.

3) MOVX @DPTR, A.

Instructions to Access External ROM/ Pgm mem

1) MOVC A, @A+DPTR.
copy the contents of external ROM address formed by adding A and the DPTR to A.

2) MOVC A, @A+PC
copy the contents of external ROM address formed by adding A and the PC, to A.

Stack instructions

1) PUSH direct : push onto stack
2) POP direct : POP from stack.

Data Exchange instructions

1) XCH A, Rn ; Exchange data bytes b/n Eg XCH A, R0 Reg Rn and A

2) XCH A, direct ; Exchange data bytes b/n Eg XCH A, 20H address directly given within instruction and A.

3) XCH A, @Ri ; Exchange data bytes between Eg ~~MOV~~ A and address in Ri

4) XCHD A, @Ri ; XCHD exchanges the low-order nibble of the Accumulator (3-0) with that of the internal RAM location indirectly addressed by the specified register. The high-order nibble (bits 7-4) of each reg are not affected.

- MOV DPTR # data

immediate data is moved to data pointer.

Arithmatic Instructions

1) ~~ADD~~
~~ADD A, B~~

2) ~~ADD~~

1) Addition

ADD

1) ADD A, Rn
   Ex ADD A, R2 : adds contents of A & R2

2) ADD A, direct
   Ex ADD A, 20H

3) ADD A, @Ri ; add the contents of A mem whose address is given by reg R2
   Ex ADD A, @R2

4) ADD A, # data.

Addition with carry

1) ADDC A, Rn
2) ADDC A, direct
3) ADDC A, @Ri
4) ADDC A, # data.

Substraction

SUB

1) SUBB A, Rn
2) SUBB A, direct
3) SUBB A, @Ri
4) SUBB A, # data

Multiplication

1) MUL AB

multiply the byte in A & B, high order byte of result is stored in B, the lower order byte is stored in A: the overflow or flag is set if the result exceeds (1s bit)

Division

DIV AB

divide the byte in A by the byte B. put the quotient in A & the remainder in B set the ov flag to 1 if B=00h before the division.

(1) INC Source

Add 1 to source

INC source

source ← source + 1

(2) DEC source

source ← source − 1

INC

1) INC A
2) INC Rn
3) INC direct
4) INC DPTR.

INC @Ri

DEC

1) DEC A
2) DEC Rn
3) DEC direct
4) DEC @Ri

DEC @Ri

DAA Decimal-adjust accumulator for addition

Logical Instructions

1) ANL – AND logic

ANL destination, source

Eg: ANL A, B

2) ORL – OR logic

ORL D, S

Eg: ORL A, B

3) XRL – XOR logic

XRL destination, source

Eg: XRL A, B

4) CPL – complement logic

CPL A

5) CLR – clear logic

CLR A

6) RRA Rotate right

7) RLA Rotate left

8) RRC – Rotate right with carry.

9) RLC - Rotate left with carry.

Boolean Instructions

1) CLR :- clear a bit /cy flag.

2) SET B - set a bit /cy flag

3) CPL - complement a bit /cy flag

4) JC /JNC - Jump on carry / Jump on No carry

Jump to a relative addres if cy is set /cleared.

5) JB /JNB - bit is set / cleared

6) JBC - Jump to a relative addres if a bit set & clear the bit

7) ORL/ANL ——> OR/AND operation of a bit with cy flag.

8) MOV — data transfer b/n a bit & cy flag.

Branching Instructions

CALL & RETURN :-

calls may be short & long range addressing returns are to any long range addres in memory

Relative (short) range :- +127 to -128
(+7F to -80H)

Absolute range : 0000H to 07FF H

long range : 0000H to FFFFH

There are two subroutine-call instruction LCALL (long call) & ACALL (Absolute or short call)

ACALL S add ;-

call the routine located at absolute short address.

L CALL Ladd r

call the routine located at absolute long address.

RET - Return to

Return :-

RET :- Return to anywhere in the program at the address found on the top two bytes of stack.

RET I :- Return from a routine called by hardware interrupt & reset the interrupt logic.

Branching Instructions :-

JUMP :-

JUMPS alter program flow by replacing the PC contents with the address of JUMP addres.

JUMPS have the following ranges.

Relative: Up to PC + 127 byte, PC - 128 bytes away from the PC.

Absolute short: anywhere on a 2k - byte

Absolute long: " in a program mem

JUMP opcodes can test an individual bit / byte.

The bit Jumps are shown below.

JC radd - Jump relative if carry flag is set.

JNC radd - Jump " " " " "
cleared.

JB b, radd - Jump relative if addressable bit is set to 1.

JNB b, radd - " " " "
bit is cleared to 0.

JBC b, r add - Jump relative if addressa -ble bit is set to 1 & cleare bit to 0.

JZ rel : Jump if Accumulator is zero

JNZ rel : " " " " not zero

JC rel : Jump if carry flag is set

JNC rel : " " " " cleared

JB b, rel : Jump if direct bit set.

Description : If the indicated bit is one Jump to address indicated.

JNB b, rel : Jump if direct bit not set.

JBC bit, rel : Jump if direct bit is set & clear bit.

Description : If the indicated bit is one branch to the address indicated The bit will not be cleared if it is already zero.

Byte Jumps

CJNE : CJNE destination, source, address compares destination & source ; Jump to address if not equal.

DJNZ : DJNZ destination, address.

Decrement destination by one. Jump to address if result is not zero.

Unconditional Jumps

JMP @A+DPTR – Jump to 16-bit add formed by adding A to DPTR.

AJMP S add

L JMP & add

S JMP r add

NOP :- No operation

Interrupts of 8051 :-

It provides 5 vectored interrupts

(1) $\overline{INT0}$ → External interrupt - 0

(2) TF0 - Timer - 0 overflow interrupt

(3) TF1 - Timer - 1 overflow "

(4) $\overline{INT1}$ - External interrupt - 1

(5) RI / TI - Serial port interrupt.

AVR RISC µC

Timer/counter 1 control Reg A – TCCR1A

COM1A1, COM1A0 : compare o/p mode 1 Bits & 0

COM1A 1	COM1A 0	Description
0	0	T/C 1 disconnected from o/p pin OC1 (o/p compare)
0	1	Toggle the OC1 o/p line
1	0	clear the OC1 o/p line (0)
1	1	Set the OC1 o/p line (1)

PWM11, PWM10 : pulse width modulator select bits

which selects the PWM operation of T/C 1

PWM11	PWM10	Description
0	0	PWM operation of T/C disabled.
0	1	Timer/counter 1 is an 8-bit PWM
1	0	T/C 1 is 9-bit PWM
1	1	T/C 1 is 10-bit PWM

Timer/counter 1 control Reg B

TCCR1B

ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10
R/W	R/W			R/W	R/W	R/W	R/W

ICNC1 : i/p capture 1 Noise canceller when this bit is 3e10 the i/p capture trigger noise canceller function disabled.

ICES1 : i/p capture edge select

If this bit 0, T/c 1 contents are transferred to the i/p capture register (ICR) on falling edge of the i/p capture pin (ICP).

If this bit 1, T/c 1 contents are transferred to the ICR on raising edge of the i/p capture pin (ICP)

CTC : clear Timer/counter 1 on compare match

If it is set T/C-1 is reset to 0000 H in the clock cycle after a compare match.

If it is cleared T/C 1 continues counting & unaffected by a compare match.

CS12, CS11, CS10 clock select Bits.

these bits defines the prescalling source of T/C 1.

CS12	CS11	CS10	Description
0	0	0	T/C-1 is stopped
0	0	1	No prescaller
0	1	0	prescaller i.e divide clk by 8
0	1	1	divide clk by 64
1	0	0	" " 256
1	0	1	" " 1024
1	1	0	Increment timer 1 on T1 pin falling edge
1	1	1	Increment timer 1 on T1 pin rising edge.

Timer/counter o/p compare reg h

→ It is a 16-bit R/W registor

→ It contains the data to be continuously compared with Timer/counter.

Timer / counter 1 i/p capture regr

It is a 16-bit registr. & Read only registr

→ when the raising or falling edge of the signal at the i/p capture pin is detected, the current value of the Timer / counter 1 is transferred to the I/p capture registr.

Interrupts (Interrupt Vector table)

S.No	program Vector	source	Interrupt definition
1	000 H	RESET	H/w pin, power on Reset and watch dog Reset.
2	001 H	INT0	External Interrupt Request 0
3	002 H	INT1	External Interrupt Request 1
4	003 H	Timer1 cap1	Timer / counter 1 capture event
5	004 H	Timer1 comp1	Timer / counter 1 compare match
6	005 H	Timer1 OVF1	Timer / counter 1 overflow
7	006 H	Timer0 OVF0	Timer / counter 0 overflow
8	007 H	UART, RX	UART, RX complete
9	008 H	UART, UDRE	UART, Data Reg empty
10	009 H	UART, TX	UART, TX complete
11	00A H	ANA_comp	Analog comparator

Interrupt structure :—

→ AVR provides 10 different interrupt sources. These interrupts & the reset vector each have a separate program vector in program memory space.

→ Each interrupt is assigned individual enable bits that must be set together with the I-bit in the status register to enable the interrupt.

→ The lowest address in the program memory space are automatically defined as the Reset & Interrupt Vectors.

→ priority levels of diff interrupts are given by the interrupt vector table.

→ The lower the address, the higher the priority level

The AVR has three sources of reset:

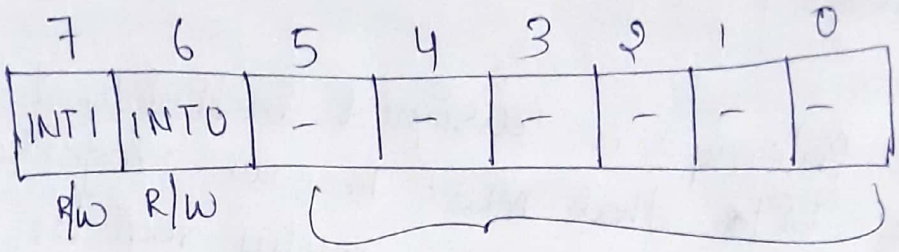
→ power-on Reset

→ External Reset

→ watch dog reset.

Registers in AVR (AT90S2313)

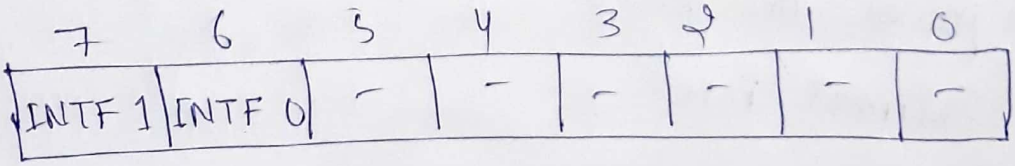
1) General Interrupt Mask Register - (GIMSK)
8-bit Reg



INT1 :— External interrupt request 1
If INT1 = 1 & I = 1 in Status Reg the external interrupt 1 is enabled.

INT0 :— External interrupt request 0.
If INT0 = 1 & I = 1 in Status Reg external interrupt 0 is enabled.

General Interrupt flag Register — (GIFR)
8-bit Reg

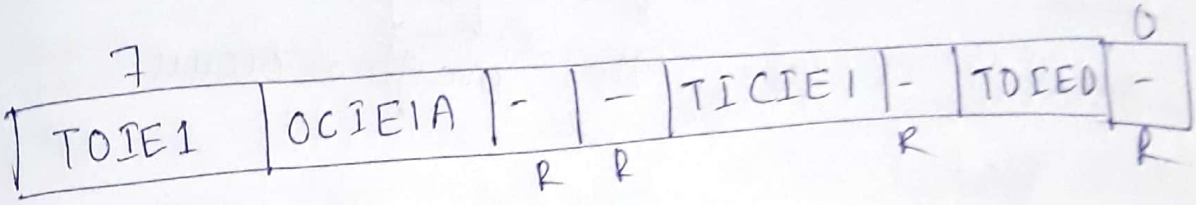


INTF 1/0 :— External Interrupt Flag 1/0
→ when an edge on the INT1/0 pin triggers an interrupt Req, the corresponding flag INTF 1/0 becomes Set.

→ If I=1 & INT1/0 =1 in GIMSK, then the uc will jump to the interrupt vector. the flag is cleared when the interrupt routine is executed.

Timer/counter Interrupt mask register

TIMSK — 8-bit Reg



TOIE1 :- Timer/counter 1 overflow Interrupt enable

→ when TOIE1 = 1, I=1 in SReg, the Timer/counter-1 overflow interrupt is enabled.

OCIE1A :- Timer/counter-1 o/p compare match Interrupt enable

→ when OCIE1A=1, I=1 in SReg T/C-1 compare match interrupt is enabled.

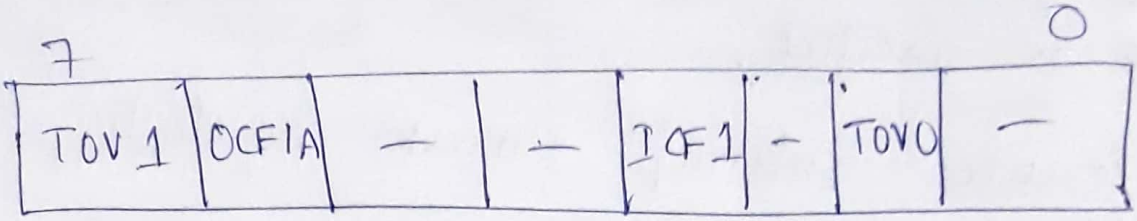
TICIE1 :- T/C-1 I/p capture IE :-

→ when TICIE1=1, I=1 in SReg, the T/C-1 i/p capture event interrupt is enabled

TOIE0 :- (Timer/counter 0 overflow IE)

when TOIE0 =1 & I=1 in SReg the Timer/counter overflow interrupt is enabled.

Timer/counter Interrupt Flag Reg (TIFR)



TOV 1 – T/C-1 overflow flag.
when it is set, the overflow occurs in T/C-1

OCF1A – O/p compare Flag 1A.
→ when it is set, the compare match occurs b/n the T/C-1 & the data in O/p compare Reg 1A.

ICF 1 :– I/p capture Flag 1
If ICF1 = 1, it indicates the T/C-1 value has been transferred to ICR 1.

TOV0 :– T/C-0 overflow flag
when it is set, the overflow occurs in T/C-0.

MCU control register :- MCUCR

8-bit Reg

The MCU control Reg contains control bits for general MCU functions.

7	6	5	4	3			
-	-	SE	SM	ISC11	ISC10	ISC01	ISC00

SE :- Sleep enable.

The SE bit must be set to make the MCU enter the sleep mode. when sleep instruction is executed.

SM :- Sleep mode.

This bit selects b/w two available sleep modes.

SM = 0 Idle mode

SM = 1 power down mode.

ISC11, ISC10 :- Interrupt Sense control 1 bit 1 & bit 0

ISC11	ISC10	Description
0	0	The lower level of INT1 generates an interrupt req
0	1	Reserved
1	0	The falling edge of INT1 generates an interrupt Req
1	1	The raising edge of INT1 generates an interrupt Req

ISC01, ISC00 ⟵ Interrupt Sense control 0 bit 1 & bit 0

ISC01	ISC00	description
0	0	The low level of INT0 generates interrupt Req
0	1	Reserved
1	0	The falling edge of INT0 gener Interrupt Req
1	1	The raising edge of INT0 generat Interrupt Req.