DEPARTMENT

OF

ELECTRONICS & COMMUNICATION ENGINEERING

MICROCONTROLLERS

(Theory Notes)

Autonomous Course

Prepared by

PROF Vibha T G

Module – 3 Contents
8051 Programming: Assembly language programs, Software delay calculations, Software delay programming. **8051 Interrupts:** Introduction, programming external interrupts in assembly. **Text Book-1**

Dayananda Sagar College of Engineering

Shavige Malleshwara Hills, Kumaraswamy Layout,

Banashankari, Bangalore-560078, Karnataka

Tel : +91 80 26662226 26661104 Extn : 2731 Fax : +90 80 2666 0789

Web - http://www.dayanandasagar.edu Email : hod-ece@dayanandasagar.edu

(An Autonomous Institute Affiliated to VTU, Approved by AICTE & ISO 9001:2008 Certified)

(Accredited by NBA, National Assessment & Accreditation Council (NAAC) with 'A' grade)

3.1 Assembly Language Programs.

1.Write an 8051-assembly language program to find the 2's complement of a number stored in location D:20h and store the result in x:2000h?

```
ORG 00H
MOV A,20H
XRL A, #0FFH
ADD A, #01H
MOV DPTR, #2000H
MOVX @DPTR, A
SJMP $
END
```

2.WAP to perform NAND operation of the two numbers stored in location 37h and 38h and store the result in X:50h(starting address).

```
ORG 00H
MOV R2,37H
MOV R1,38H
MOV A, R2
AND A, R1
CPL A //FOR NAND
MOV R0, #50H
MOVX R0, A
SJMP $
END
```

3.WAP to check whether the given number is a positive or negative number

```
ORG 00H
MOV R1, #20H
MOV A, @R1
RL A
JC NEG
CLR A
SJMP LAST
NEG:INC A //if number is negative A is incremented
LAST: SJMP $
END
```

4.Set of 10 numbers are stored in the memory location starting from D:20h and check whether the number is positive or negative and also check odd or even. Also keep count of +ve and –ve number.

```
ORG 00H
MOV R2, #00H //Even count
MOV R3, #00H //odd count
MOV R1, #20H //Source address
MOV R0, #0AH //counter variable
BACK:MOV A, @R1
RL A
```

```
JC NEG
INC R2
SJMP LAST
NEG:INC R3
LAST:INC R1
DJNZ R0, BACK
SJMP $
END
```

5. **Set of 10 numbers are stored in the memory location starting from D:20h. Store the odd numbers in location starting from 30h and even numbers starting from location 40h.**

```
ORG 00H
MOV R3, #30H //storing location for odd no
MOV R2, #40H //storing location for even no
MOV R1, #20H //source address
MOV R0, #0AH //counter
BACK:MOV A, @R1
RR A
JC ODD
MOV @R2,A
INC R2
SJMP LAST
ODD:MOV @R3,A
INC R3
LAST:INC R1
DJNZ R0,BACK
SJMP $
END
```

6.**Port 1 is read and compaired to the value 65**

If P1==65,then A has the result

P1<65,R2 has the result

P1>65 ,R3 has the result

```
ORG 00H
CLR C
MOV A, P1
CJNE A, #65, NEXT
SJMP EXIT
NEXT: JC NEXT1
MOV R3, A
SJMP EXIT
NEXT1:MOV R2, A
EXIT: SJMP $
END
```

7.Add five numbers stored in an array starting from the location 20H.

```
ORG 00H
MOV R1,#00
MOV R2,#04
MOV R0,#20H
```

```
MOV A,@R0
DAA
JNC NEXT
INC R1
NEXT: DJNZ R2, BACK
SJMP $
END
```

8.Search RAM locations 40h to 4fh to find how many of them are zeroes.

```
ORG 00H
MOV R2, #00
MOV R0, #40H
MOV R1, #0FH
BACK:MOV A, @R0
JNZ L1
INC R2
L1:INC R0
DJNZ R1, BACK
SJMP $
END
```

9.Monitor the bit P1.5 continuously when it becomes low set 55H to port 2.

```
SETB P1.5
HERE: JB P1.5, HERE
MOV P2, #55H
```

10.Write an 8051 ALP to find Factorial of a number (upto 5)

```
ORG 00H
MOV R0,25H
MOV A, R0
DEC R0
BACK:MOV B, R0
MUL AB
DJNZ R0, BACK
MOV 26H, A
SJMP $
END
```

11.Write an ALP to find Fibonacci series till 12.

```
ORG 00H
MOV 12H,#01
MOV 13H,#01
MOV R1,#0AH
MOV A,12H
BACK:ADD A,13H
MOV 13H,A
MOV A,R0
DJNZ R1,BACK
SJMP $
END
```

12. Write an 8051 ALP to Add two 32 bit numbers.

```
        ORG 00H
        MOV R0, 04H
        MOV R1, #12H
        MOV R2, #16H
        MOV R3, #1AH
        CLR C
BACK: MOV A, @R1
        ADDC A, @R2
        MOV @R3, A
        INC R1
        INC R2
        INC R3
        DJNZ R0, BACK
      SJMP $
       END
```

13.Write an ALP to find the address of 1st two internal ram location b/w 20h to 60h which contains consecutive number?

```
      ORG 00H
        MOV R1, #20H
        MOV R0, #41H
BACK: MOV A, @R1
        INC R1
        SUB A, R1
        MOV R0, A
        CJNE R0, #00, LAST
LAST:   DJNZ R0, BACK
        MOV A, R1
        MOV R2, A
        DEC R1
        MOV A, R1
        MOV R3, A
        SJMP $
        END
```

14.Write an 8051 ALP to find LCM of two numbers?

```
        ORG 00H
        MOV R0, 20H
        MOV R1, 21H
        MOV R2, #01H
        MOV A, R0
        MOV B, R1
        SUBB A, R1
        JC NEXT
BACK:   MOV A, R0
        MOV B, R2
        MUL AB
        MOV R3, A
```

```
        MOV B, R1
        DIV AB
        INC R2
        XCH A,B
CJNE A, #00H, BACK
        SJMP LAST
NEXT: MOV A, R0
        XCH A, R1
        MOV R0, A
        SJMP BACK
LAST: MOV 22H, R3
        SJMP $
        END
```

15. Write an 8051 ALP to check whether given number is nibble wise Palindrome? (66h,99h, CCh are examples).

```
      ORG 00H
        MOV R2, #00H
        MOV R3, #00H
        MOV A, 25H
        MOV R0, A
        SWAP A
        MOV R1, A
        CLR
        MOV A, R0
        SUBB A, R1
        CJNE A, #00, NP
        INC R2
        SJMP NEXT
NP:     INC R3
NEXT: SJMP $
        END
```

16. Write an 8051 ALP to find the Cube of a number.

```
      ORG 00H
        MOV A, #0FFH
        MOV B, #0FFH
        MOV R0, #0FFH
        MUL AB
        MOV R1, B
        MOV B, R0
        MUL AB
        MOV 50H, A
        MOV R5, 50H
        MOV R2, B
        MOV A, R1
        MOV B, R0
        MUL AB
        ADD A, R2
```

```
        JNC SKIP
        MOV R3, #00H
        INC R3
SKIP:MOV 51H, A
        MOV R6, 51H
        MOV A, B
        ADD A, R3
        MOV 52H, A
        MOV R7, 52H
        SJMP $
        END
```

17. Cube of a number using Push and Pop instructions.

```
        ORG 00H
        MOV R0, #20H
        MOV A, @R0
        MOV R1, A
        MOV B, A
        INC R0
        MOV A, @R0
        MUL AB
        MOV SP, #30H
        PUSH 0F0H
        MOV B, A    //B = 01H
        MOV A, R1  //A = FFH
        MUL AB
        INC R0
        MOV @R0, A
        MOV A, B
        MOV R2, A
        POP 0F0H
        MOV A, R1
        MUL AB
        ADD A, R2
        INC R0
        MOV @R0, A
        MOV A, B
        ADDC A, #00H
        INC R0
        MOV @R0, A
        MOV A, B
        ADDC A, #00H
        INC R0
        MOV @R0, A
        SJMP $
        END
```

3.2: Software Delay Programs:

Time delay:

- The number of clock cycle taken to excute an instruction is called machine cylae.
- The length of a machine cycle depends on frequency of an crystal oscillator.
- Frequency can vary from 4MHz to 30MHz
- 11.0592MHz crystal oscillator is used to to make 8051 system compatible with serial port of IBM PC.
- In 8051 systems 12 clock cycles combination is called as one machine cycle.
- So the time taken to complete one machine cycle is 1/12*clock frequency.

Example:1

The following shows the crystal frequency of 3 different 8051 based systems. Find the period of machine cycle in each case.

a. 11.0592 MHz
b. 16MHz
c. 20MHz

Ans.

a. 11.0592MHz/12=921.6KHz=1/921.6KHz=1.085µ
b. 16MHz/12=1.33MHz;MC cyce=1/1.333MHz=0.75 µ Sec
c. 20MHZ/12=1.66MHz;Machine cyle=1/1.666MHz=060 µSec

Example:2

For 8051 system of 11.0592 MHz, find how long it takes to execute each instruction.

(a) MOV R3,#55
(b) DEC R3
(c) DJNZ R2 target
(d) LJMP
(e) SJMP
(f) NOP
(g) MUL AB

Solution: Delay calculation for 8051:

	Machine cycles	Time to execute
A	1	1x1.085µs = 1.085µs
B	1	1x1.085µs = 1.085µs
C	2	2x1.085µs = 2.17µs
D	2	2x1.085µs = 2.17µs
E	2	2x1.085µs = 2.17µs
F	1	1x1.085µs = 1.085µs
G	4	4x1.085µs = 4.34µs

Find the size of the delay in following program, if the crystal frequency is 11.0592MHz.

```
MOV A,#55H
AGAIN: MOV P1,A
ACALL DELAY
CPL A
SJMP AGAIN
;---time delay-------
DELAY: MOV R3,#200
HERE: DJNZ R3,HERE
RET
```

Solution:

	Machine cycle
DELAY: MOV R3,#200	1
HERE: DJNZ R3,HERE	2
RET	2

Therefore, [(200x2)+1+2]x1.085µs = 436.255µs

Loop inside loop:

Another way to get higher delay is nested loop.

Example 3 : Find the size of the delay in following program, if the crystal frequency is 11.0592MHz

	Machine cycle
DELAY: MOV R3,#250	1
HERE: NOP	1
NOP	1
NOP	1
NOP	1
DJNZ R3,HERE	2
RET	2

Solution:

The time delay inside HERE loop is [250(1+1+1+1+2)]x1.085µs = 1627.5µs.

Adding the two instructions outside loop we have 1627.5µs + 3 x 1.085µs = 1630.755µs

Find the size of the delay in following program, if the crystal frequency is 11.0592MHz.

	Machine cycle
DELAY: MOV R2,#200	1
AGAIN: MOV R3,#250	1
HERE: NOP	1
NOP	1
DJNZ R3,HERE	2
DJNZ R2,AGAIN	2
RET	2

Solution:
For HERE loop, we have (4x250) x1.085µs＝1085µs.
For AGAIN loop repeats HERE loop 200 times, so
we have 200x1085µs＝217000µs. But "MOV
R3,#250" and "DJNZ R2,AGAIN" at the start and
end of the AGAIN loop add (3x200x1.805)=651µs.
As a result we have 217000+651=217651µs.

Example:4

Write an ALP to toggle all the pins of P1 every 200m Sec. assume crystal frequency is 11.0592MHz

```
        Mov ,#55h
Again:  Mov P1,a
        Acall Delay
        Cpl a
        Sjmp again
Dealy:  mov R5,#2
Here1:  mov R4,#180
Here2:  mov R3,#255
Here3:  Djnz R3,here3
        Djnz R4,here2
        Djnz R5,here1
        Ret
```

3.3 8051 Interrupts:

During program execution if peripheral devices needs service from microcontroller, device will generate interrupt and gets the service from microcontroller. When peripheral device activates the interrupt signal, the processor branches to a program called interrupt service routine. After executing the interrupt service routine, the processor returns to the main program.

3.3.1 Interrupt Service Routine (ISR):

- For every interrupt there must be an ISR or interrupt handler. When an interrupt is invoked controller runs the ISR.
- For every interrupt there is a fixed location in the memory that holds the address of the ISR.The group of memory locations set aside to hold the addresses of ISR's is called the **Interrupt Vector Table (IVT)**

3.3.2Steps taken while processing an interrupt:

- It completes the execution of the current instruction.
- PSW is pushed to stack.
- PC content is pushed to stack.
- Interrupt flag is reset.
- PC is loaded with ISR address.
- ISR will always ends with RETI instruction. The execution of RETI instruction results in the
- following.
- POP the current stack top to the PC.
- POP the current stack top to PSW.

3.3.3 8051 Interrupt Structure:

8051 has five interrupts. They are maskable and vectored interrupts. Out of these five, two are external interrupt and three are internal interrupts.8051 interrupt vector table is shown below as table 3.3.3.1

Interrupt source	*Type*	*Vector address*	*Priority*
External interrupt 0	External	0003	Highest
Timer 0 interrupt	Internal	000B	
External interrupt 1	External	0013	
Timer 1 interrupt	Internal	001B	
Serial interrupt	Internal	0023	Lowest

Table 3.3.3.1:8051 Interrupt Vector Table(IVT)

8051 makes use of two registers to deal with interrupts.

1. **IE Register (Interrupt Enable)**

This is an 8-bit register used for enabling or disabling the interrupts. Its bit addressable. The structure of IE register is shown below fig 3.3.3.1

EA	-	-	ES	ET1	EX1	ET0	EX0

Fig 3.3.3.1:8051 IE register format

EA-Enable or disable all interrupts. If EA=0 it disables all interrupts. If EA=1,all interrupts can be individually enabled or disabled by setting or clearing its enabling bit.

- Reserved for future use
- Reserved for future use

ES-Enable or disable serial port interrupt
ET1-Enable or Disable Timer 1 Overflow Interrupt.
EX1- Enable or Disable External Interrupt 1
ET0- Enable or Disable Timer 0 Overflow Interrupt.
EX0- Enable or Disable External Interrupt 0

Example: Show the instructions to

a) enable the serial interrupt, Timer 0 interrupt and external hardware interrupt 1.
b) Disable(mask) Timer1 interrupt.
c) Disable all interrupts.
   a) MOV IE,#10010110B
   b) CLR IE.1
   c) CLR IE.7

2. IP Register (Interrupt Priority Register):

This is an 8-bit register used for setting the priority of the interrupts. If the bit is 0, the corresponding interrupt has a lower priority and if the bit is 1 the corresponding interrupt has a highest priority. IP register format is as shown in below fig 3.2

-	-	-	PS	PT1	PX1	PT0	PX0

Fig 3.3.3.2: IP register format.

PS-Defines the serial port interrupt priority level

PT1-Defines the Timer1 Interrupt Priority level
PX1-Defines External Interrupt 1 Priority level.
PT0- Defines the Timer0 Interrupt Priority level
PX0-Defines External Interrupt 0 Priority level.

Example1:

a) Program the IP register to assign the highest priority to INT1 (external interrupt 1),
MOV IP, #00000100B or SETB IP.2

b) Discuss what happens if INT0, INT1, and TF0 are activated at the same time.
First INT0 is serviced followed by TF0 and then INT1 as per default priority of interrupts.

Example 2:

Assume that after reset, the interrupt priority is set by the instruction
"MOV IP, #00001100B". Discuss the sequence in which the interrupts are serviced.

Solution: The sequence in which the interrupts are serviced is shown below.

Highest Priority	External Interrupt 1 (INT1)
	Timer Interrupt 1(TF1)
	External Interrupt 0(INT0)
	Timer Interrupt 0(TF0)
Lowest Priority	Serial Communication (RI+TI)

3.3.4Programming External Hardware Interrupts:

- 8051 has two external hardware interrupts. Pins P3.2 and P3.3 of 8051 named as INT0 and INT1.
- On activation of these pins,8051gets interrupted in whatever it is doing and jumps to vector table to perform the ISR.
- There are two types of activations for external hardware interrupts
  - Level triggered
  - Edge triggered

Activation of INT0 and INT1 interrupts using the 2 methods is diagrammatically shown in below figure 3.3.4.1

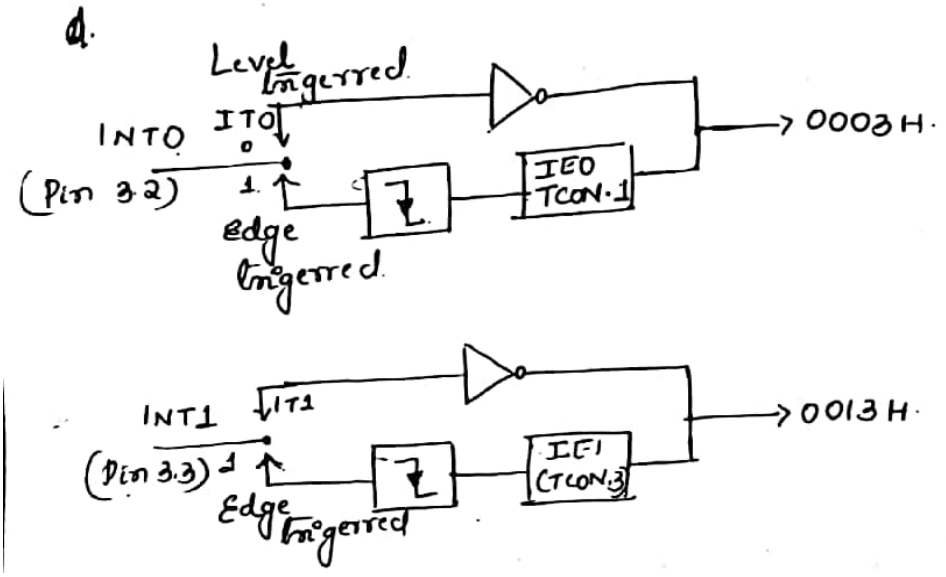


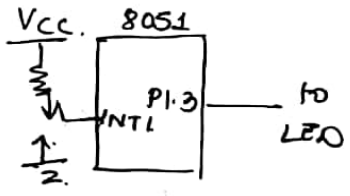
Figure 3.3.4.1 Activation of INT0 and INT1 pins

3.3.4.1 Level Triggered Interrupt:

- In level triggered mode, INT0 and INT1 pins are normally high and if low level signal is applied to them, it triggers the input interrupt.
- The controller then stops whatever it is doing and jumps to the ISR to service the interrupts. This is called level triggered or level activated interrupt and is the default mode upon reset of 8051.
- This low-level signal at the INT pin must be removed before the execution of ISR, RETI, else another interrupt will be generated.

Program 1

Assume that INT1 pin is connected to a switch that is normally high, whenever it goes low, it should turn on the LED. The LED is connected to P1.3 and is normally off. When it is turned on it should stay on for fraction of a second.




```
        ORG 00h
        LJMP MAIN
        ORG 0013h
        SETB P1.3
        MOV R3, #255
BACK: DJNZ R3, BACK
        CLR P1.3
        RETI
        ORG 30H
        MOV IE, #10000100B
HERE: SJMP HERE
        END
```

Sampling the low-level triggered interrupt:

- Once the hardware interrupts in IE register are enabled, the controller keeps sampling the INTx pin for a low-level signal once each machine cycle.
- To ensure the activation of hardware interrupt at the INTx pin, the duration of low level signal should be around 4 machine cycles and is as shown in fig 3.3.4.1.1 below.

Fig 3.3.4.1.1.:Sampling low level interrupt

3.3.4.2 Edge Triggered interrupts:

- To make interrupts edge triggered, we must program the bits of TCON register.
- In TCON register we have 4 bits related to interrupts i.e., IE1, IT1, IE0, IT0.
  IT1 and IT0=0, Implies interrupts are low level triggered.
  IT1 and IT0=1, Implies interrupts are edge triggered and are to be done by the programmer.
- When there is a high to low signal applied on pin P3.3 the collector will be interrupted and forced to jump to the location 0013H in vector table. Similarly, for P3.2.

Program 2:

Assuming P3.3(INT1) is connected to pulse generator, write an 8051 ALP in which falling edge of pulse will send a high to P1.3 which is connected to buzzer. In other words, LED is turned on and off at the same rate as pulses are applied to INT1.

```
        ORG 00H
        LJMP MAIN
        ORG 0013H
        SETB P1.3
        MOV R3, #255
BACK:   DJNZ R3, BACK
        CLR P1.3
        RETI
        ORG 30H
MAIN:   SETB TCON.2
        MOV IE, #10000100B
        SJMP $
        END
```