Saroj Prasad Mainali
"THA075BCT040"

Chapter 9
Microcontrollers in Embedded System.

Q.No 1) What is difference between microprocessor and microcontroller?

	Microprocessor	microcontroller
1)	General purpose processor	1) Specific purpose processor
2)	Only contains CPU.	2) Contains CPU, timers, I/O ports, RAM and ROM.
3)	Clock speed is very high in GHz range	3) Clock speed is low in MHz
4)	Powerfull addressing modes and many instructions are available to move data between memory and CPU	4) It focuse on bit handling instructions along with byte processing instructions.
5)	Designer can select the size of memory, number of I/O ports, timers etc.	5) Size of memory, number of I/O ports, timers etc are fixed.
6)	Systems based of microprocessor are expensive and consumes more power	6) Microcontroller based systems are cheap and consume less power.
7)	Access time is more (external)	7) Access time (on-chip memory) is less

2. What are the factors to be considered in selecting a controller? Explain different addressing modes of 8051 micro controllers.

⇒ The factors to be considered in selecting a controllers are:

1) Computational capabilities
   a) speed
   b) packaging.
   c) power consumption
   d) size of RAM, ROM
   e) no. of I/O pins
   d) timer.

2) Flexibility to develop product.
   Availability of
   a) assembler
   b) debugger
   c) compiler
   d) emulator
   e) technical support.

3) Availability of microcontroller with reliables resources.

Different addressing modes of 8085 microcontrollers are:

1) Immediate addressing mode.
   → It is used to load direct values into registers.
   MOV A, #25H

2) Register Direct Addressing mode.
→ The operand is a register which holds the data to be manupulated
eg. ADD A, R3 (Adds content of A & R3 stores in A)
(A = A+R3)

3) Register indirect Addressing mode.
→ Register holds the addren of data to be manupulated.
Register R0, R1 & DPTR are used as pointer.
eg. MOV A, @R1.

4) Direct addressing mode.
→ The operand represents the address in memory where data is present eg. MOV A, 30H.

5) Relative addressing
→ A offset is added to PC to form actual addres.

6) Absolute Addressing.
→ 11-bit or 16-bit absolute address is specified as operand.
ACALL, AJMP uses 11-bit address
LCALL, LJMP uses 16-bit address

7) Indexed addressing mode
→ Index value ~~or b~~ is added to base address to generate effective addres of operand.
for eg.
MOVC A, @A+DPTR

3. What are the different types of instruction sets used in 8051? Explain in brief with examples.

→ Different types of instruction sets are:-

1) Data transfer instruction
→ It relates with instruction responsible for movement of data in, among register, memory etc. Some of Data transfer instructions are:-
1) MOV : Data movement between registe.
2) MOVX : move data to and from external RAM
3) MOVC : code memory read only data mov.
4) PUSH and POP : Stack operation
5) XCH : Data exchang

2) Arithmetic instruction.
→ It is responsible to calculate arithmetic operation. It includes.
1) ADD : Add value of register, memor
2) ADDC : add with carry
3) SUB : subtract one value form another
4) SUBB : Subtract with borrow
5) INC / DEC : Increase / decrease value by one
6) MUL : Multiply accumulator and register B
7) DIV : divide accumulator by B
8) DA : Decimal Adjust Accumulator.

3) Logical instruction
→ Responsible to solve logic operation
1) ANL : AND operation.

2) ORL : OR operation
3) XRL : XOR operation
4) CLR A : Clear Accumulator
5) CPL A : complement Accumulator
6) RL A : Rotate Accumulator left
7) RLC A : Rotate left through carry
8) RRA : Rotate Right (Accumulator)
9) RRC A : Rotate Accumulator Right through carry
10) SWAP A : Swap nibbles of Accumulator.

4) Bit manipulation instructions.
→ Responsible to do bitwise manipulation
CLR C/bit : clear carry / bit
SETB C/bit : sets carry / bit
CPL C/bit : Compliment
ANL C, bit : And operation
. . . .

5) Program control insuction
→ Responsible for control flow, Jumps, returns etc.
1) ACALL addr 11
2) LCALL addr 16
3) RET
etc.

4) Why 8051 microcontroller is used? Write a assembly program to get data from P0 and send it to P1 and compare with corresponding C program.

→ 8051 microcontroller is used because of following reasons.

[redacted]

4) It

x x x (True reasons)

⭐) Eight bit CPU with registers A and B
⭐) 16 bit PC and DPTR
⭐) Eight bit PSW (program status word)
⭐) 8 bit stack pointer (SP)
⭐) Internal ROM
⭐) Internal RAM
⭐) four 8-bit ports (32 I/O pins)
⭐) two 16-bit timers/counters (T0 and T1)
⭐) Full duplex serial data receiver/transmitter.
⭐) Two external interrupts and 3 internal interrupts
⭐) Operating frequency is 11.0592 MHz. Time period is 1.085 μs

Program to get data from P0 and send it to P1 and compare with corresponding C program.

Assembly code

```
ORG 00H
MOV P0, #0FFH
MOV P1, #00H
Main:
MOV A, P0
mov P1, A
SJMP Main
End
```

C++ code

```
#include <at89x52.h>

void main(){
    P0 = 0xFF;
    P1 = 0x00;
    while(1){
        P1=P0;
    }
}
```

5) List all the special function register of 8051 and explain function of each.

→ The special function register.

1) A (Accumulator)
   → Intermediate storage of arithmetic & logic data.

2) Register B
   → for multiplication & division

3) PSW
   → flag register. (CY, AC, F0, RS1, RS0, OV, -, P)

4) SP (stack pointer)
   → Points to Top of stack

5) DPH / DPL
   → Data pointer Higher byte, Data pointer lower byte. (used for addressing external memory).

6) IE, IP
   → enable interrupt, set priority of interrupt.

7) P0 - P3.
   → Port 0, Port 1, port 2, port 3.

8) PCON, SCON, SBUF
   → Power control, Serial Port control, serial port Data buffer.

9) TMOD
→ Timer / Counter Mode Control

10) TCON
→ Timer / Counter Control

11) TL0, TH0, TL1, TH1
→ Timer 0 & Timer 1 lower and upper byte.

6) Write an assembly program for generating a square wave of 66.67% duty cycle on pin 3.1. Compare it with C program.

```
ORG 00H
  CLR P3.1
main:
  CLR P3.1
  LCALL DELAY
  SETB P3.1
  LCALL DELAY
  LCALL DELAY
  SJMP main:

ORG 300H
DELAY:
  MOV R5, #64H
AGAIN: MOV R4, #FFH
AGAN: MOV R3, #08H
AGA: DJNZ R3, AGA
  DJNZ R4, AGAN
  DJNZ R5, AGAIN
  RET.

END.
```

C program

```
#include <at89x52.h>

void delay(unsigned int x){
    int i,j;
    for(i=0; i<x; i++)
    for(j=0; j<1275; j++);
}

void main(){
    P3.1=0;
    while(1){
        P3.1=0;
        delay(100);
        P3.1=1;
        delay(200);
    }
}
```

Q. No. 7

20ms delay.

ie.

Total MC require = 18433 $\left(\frac{20ms}{1.085\mu s}\right)$

```
DELAY:
    MOV R4, #0BEH      ; 1MC
L1: MOV R3, #2FH       ; 1MC x 190
L2: DJNZ R3, L2        ; 2MC x 190 x 47
    DJNZ R4, L1        ; 2MC x 190
    RET                ; 2MC
```

⇒ 18433MC

= 2ms

delay #

Q.no.

8) Write an assembly code for up counter 0 to 9 controlled by a PUSH butt

8) Write an assembly program for controlling LED connected at port 2 with the switch connected at port 3.

Solution

```
ORG 00H
MOV P2, #00H
MOV P3. #0FFH

Main:
  mov A, P3
  mov P2, A
  SJMP Main
END
```

9) Write an assembly code for up counter 0 to 9 controlled by a Push putton at P1.1 considering a situation that count is increased when a switch is presed.

Solution

```
ORG 00H
  SETB P1.1
  MOV P2, #00H
  MOV R6, #00H
  MOV DPTR, @#DIGITS

MAIN:  MOV A, R6
       MOVC A, @A + DPTR
```

```
mov P2,A
mov C, P1.1
JC MAIN
LCALL DELAY
INC R6
CJNE R2, #0AH, MAIN
MOV R6, #00H
SJMP MAIN

DELAY:
    MOV R3, #0F0H
DEL1: MOV R2, #0FAH
DEL2: DJNZ R2, DEL2
      DJNZ R3, DEL1
RET
END.
```

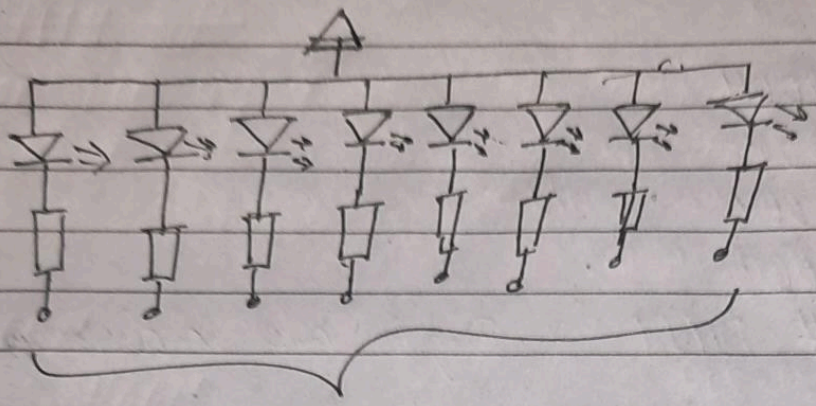
Q.no.10

Explain the modes of Seven segment display? Write an assembly code for segment to count from 99 to 00.

→ There are two modes of seven segment display:

1) Common Anode Configuration.
   → In this mode all Anode is connected to

5V.



from microcontroller.

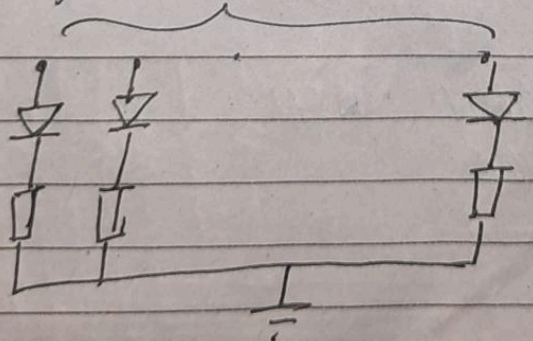
logic 0 is given from microcontroller if LED is to be turned on.

dt.	g	f	e	d	c	b	a	code
1	1	0	0	0	0	0	0	=0x00

to show 0 in display.

Common Cathod mode.

→ Cathodes of all LEDs are connected together to low logic voltage.



logic 1 is given from microcontroller if LED is to be turned on.

For 0 to display "0x3F is send to port."

99 to 00

```
..
ORG 00H
MOV P2, #00H
MOV P1, #00H
MOV R6, #00H
MOV R7, #00H
MOV DPTR, # DIGIT ; Digites are in descendig order 0 to 9

MAIN:
   MOV A, R7
   MOVC A, @A +DPTR
   MOV P2, A
LSB: MOV A, R6
     MOVC A, @ A+DPTR
     MOV P1, A
     LCALL DELAY
     INC R6
     CJNE R6, #0AH, LSB
     MOV R6, #00H
     INC R7
     CJNE R7, #0AH, MAIN
     MOV R7, #00H
     SJMP MAIN

DELAY:
    MOV R3, #0F0H
DEL1: MOV R2, #0FAH
DEL2: DJNZ R2, DEL2
      DJNZ R3, DEL1
RET
```

```
DIGITS:
   DB  6FH, 7FH, 07H, 7DH, 6DH ; (9-5)
   DB  66H, 4FH, 5BH, 06H, 3FH ; (4-0)

END
```

Q.No. 12

Code for transferring data from port2 to port1

```
ORG 00H
MOV P2, #0FFH
MOV P1, #00H
MAIN:
   MOV A, P2
   MOV P1, A
   SJMP MAIN
END.
```

Q.No. 11

Port 0: (Pins 32-39)

→ It is a collection of open drain bidirectional I/O pins. It can be configured as low order address or data bus while accessing external memory

Port 1 (pins 1-8)

→ These 8 pins represt port 1 which can be used as an input / output port. Since it is interally pulled up, it can be used without any external pull up registers configuration.

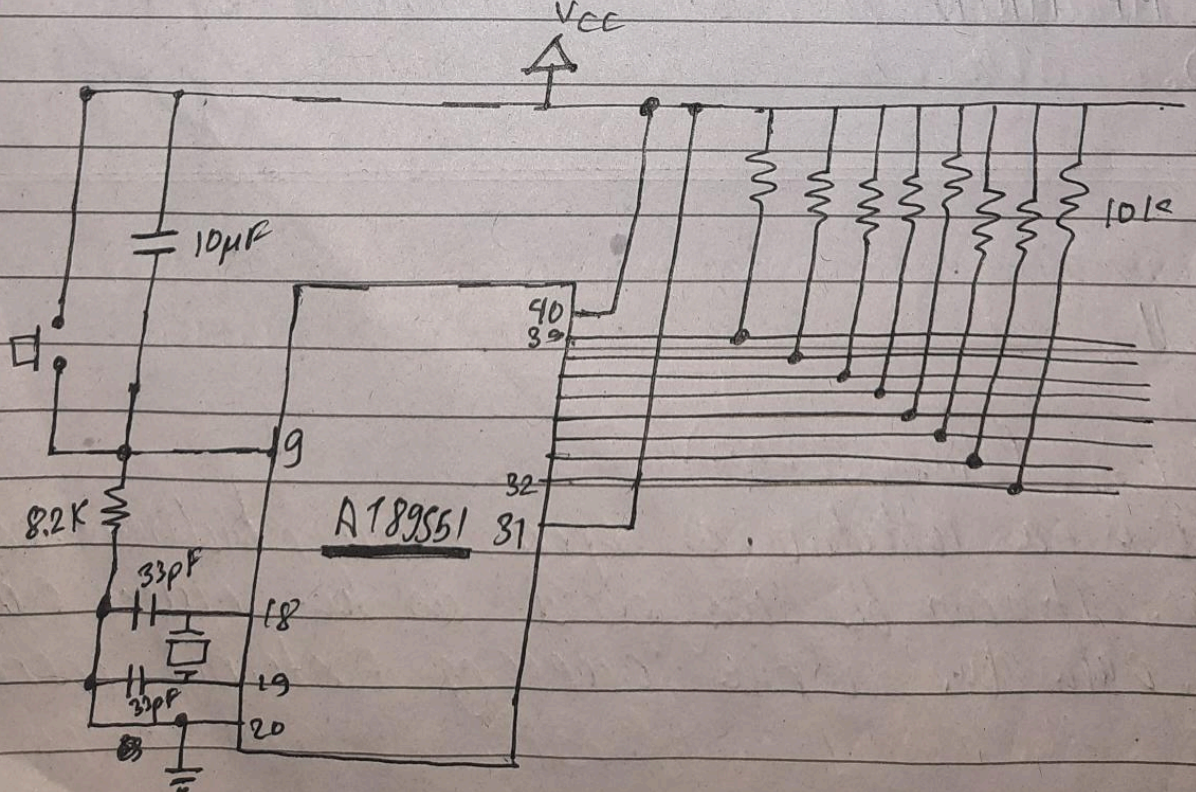
Port 2 (pins 21-28)

→ This is bidirectional and multifunctional in nature. Similiar to port 1. It is also used to provide high-order address in conjuction with port 0 for external addres

Port 3 (pin 10-17)

→ Bidirectional and multifunctional. Similiar to port 1. Biside this it support serial communicator, external interrupts, timers, control signal for external memory.

Minimun Hardware Configuration



2 sec delay $\left(\frac{2}{1.085\times10^{-6}}\right)$ = 1843318MC

```
Delay:
        mov R1, #C8H ;      1X1MC
A:      mov R2, #C8H ;      200x1MC
B:      mov R3, #15H ;      200x200x1MC
C:      djnz R3, C   ;      200x200x21x2MC
        djnz R2, B   ;      200x200x2MC
        djnz R1, A   ;      200x2MC

        mov R1, #C8H ;      1x1MC
D:      mov R2, #69H ;      200x1MC
E:      djnz R2, E   ;      200x105x2MC
        djnz R1, D   ;      200x2MC

        mov R1, #39H ;      1x1MC
F:      djnz R1, F   ;      56x2MC

        nop          ; 1MC

        RET          ; 2MC.
```

= 1 + 200x1 + 200x200x1 + 200x200x21x2 + 200x200x2 + 200x2 + 1 + 200x3
200x105x2 + 200x1 + 56x2 + 3

= 1843318MC

∴ Time = $1843318 \times 1.085\times10^{-6}$ = 2.00000003 sec
≈ 2 sec