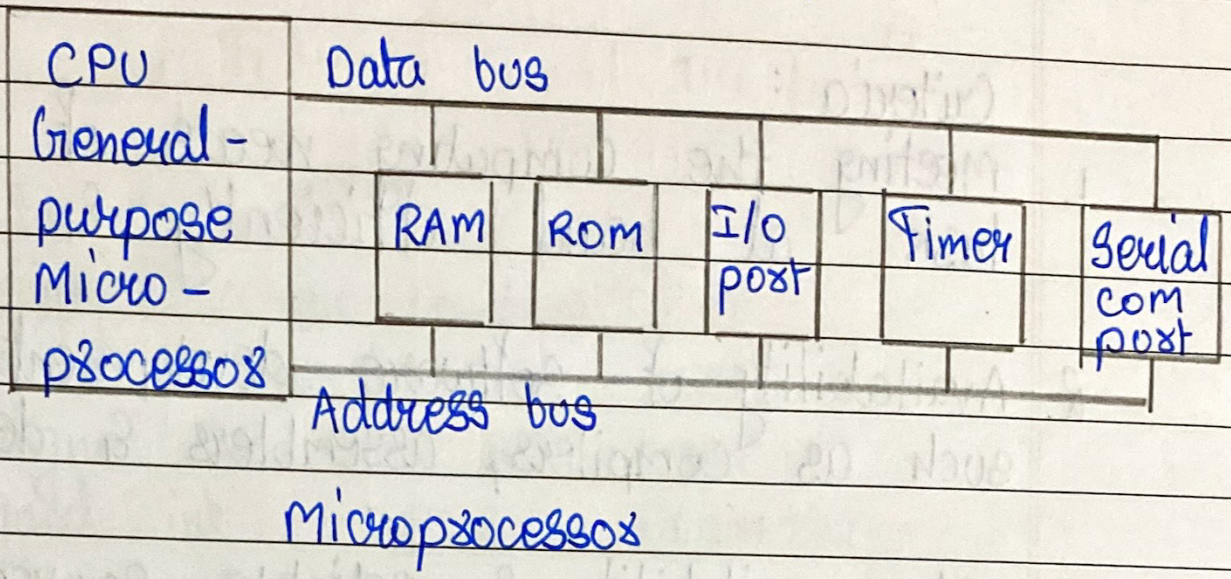
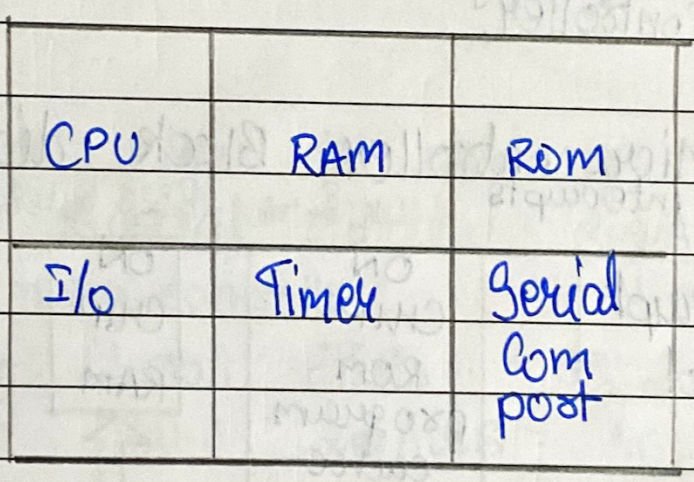
6/3/21. The 8051 Microcontrollers

* Micro controllers & Embedded processors :



Microprocessor



Microcontroller.

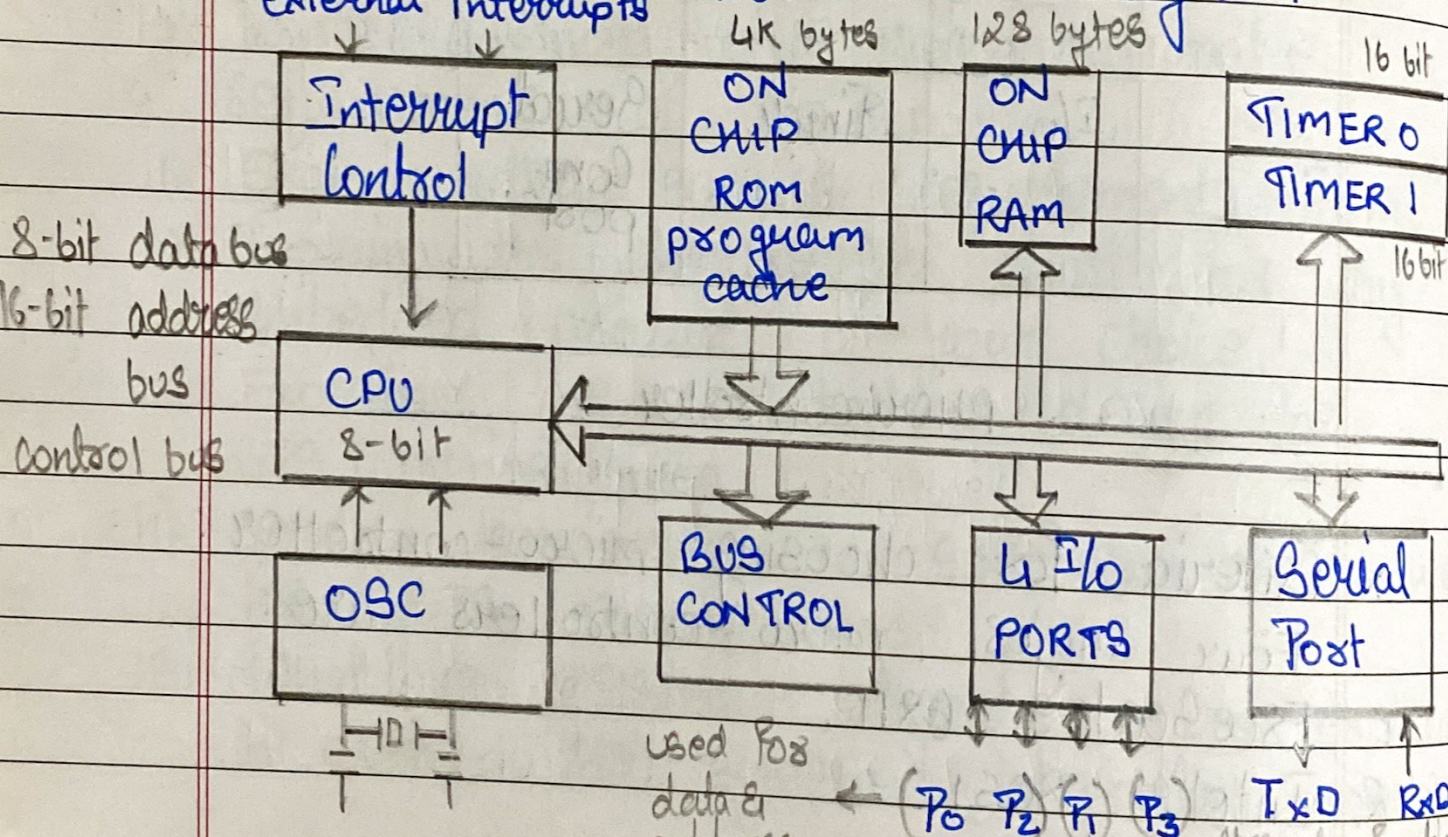
* Criteria for choosing micro-controller :
Four major micro-controllers are :
1. Freescale's 6811.
2. Intel's 8051.

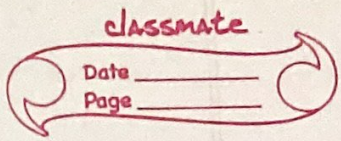
3. Zilog's Z8.
4. PIC 16X.

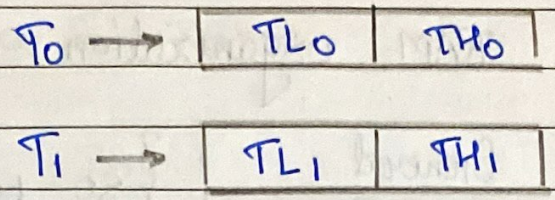
Criteria:

1. Meeting the computing needs of the task at hand efficiently & effectively.
2. Availability of software development tools such as compilers, assemblers & debuggers.
3. Wide availability & reliable sources of micro controller.

* 8051 Microcontroller Block diagram:







* Data types
1. Unsigned char (0-255).
2. Signed char (-128 - +127).
3. unsigned int (0 - 65, 535)
4. Signed int (-32768 - +32767).
5. sbit (1-bit)
6. sfr
7. bit

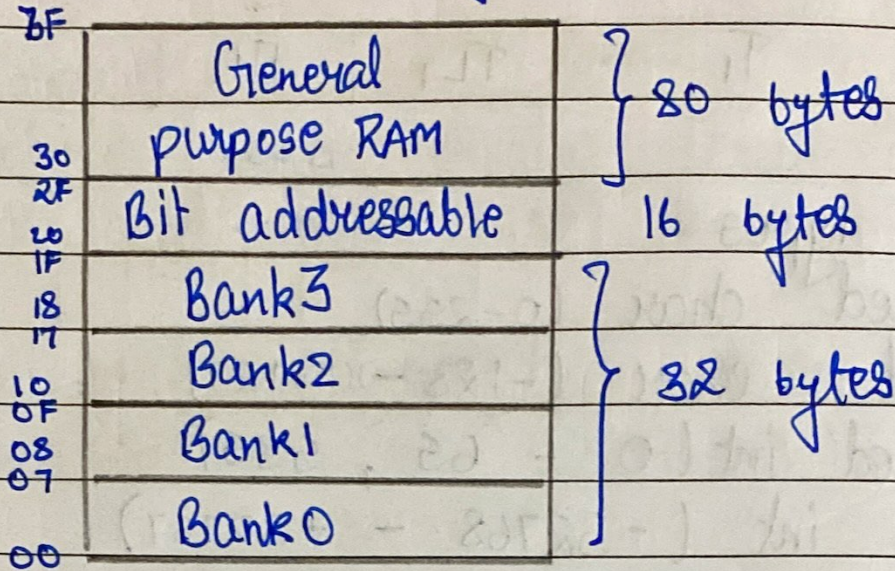
→ sbit data type is used to access single bit of special function registers of 8051 micro controller.

Eg: sbit P0B0 = P0.0
sbit segALSB = ACC.0

→ bit data type is used to access single bit of bit-addressable memory locations
20 - 2F

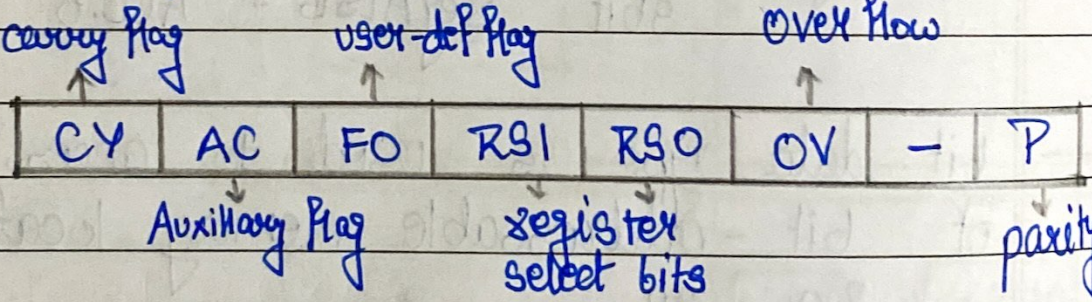
→ sfr to access the byte size special function registers we use sfr data type.

* Internal RAM organization :



21	0F	0E	0D	0C	0B	0A	09	08
20	07	06	05	04	03	02	01	00

* PSW (Program Status Word) :



Eg : Post AC : 0110 1001
0001 1000
0001

For: RS0 & RS1

RS1	RS0	
0	0	Bank 0
0	1	Bank 1
1	0	Bank 2
1	1	Bank 3

* Programs:

Program 1: To introduce delay:

```
#include <reg51.h>
void main (void)
{
   unsigned char x;
   while(1)
   {
      for (x=0; x<=255; x++)
      {
            p1=x;
      }
   }
}
```

* Write 8051 C program to send the ASCII values of hexadecimal values on port 1.

```
#include <reg51.h>
void main (void)
{
   unsigned char mynum[] = "0123456789ABCDEF";

   unsigned char z;
   for (z=0; z<=16; z++)
        p1 = mynum[z];
}
```

* Write a 8051 C program to toggle the of port 1 forever.

```
#include <reg51.h>
void main (void)
{
    for(;;)
    {
        p1 = 0x55;
        P1 = 0xAA;
    }
}
```

* Write a 8051 C program to send the signed numbers 1 to 4 on port 1.

```
#include <reg51.h>
void main (void)
{
  char mynum [] = {+1, -1, +2, -2, +3, -3,
                   +4, -4} ;
  unsigned char x;
  for (x=0 ; x<8 ; x++)
      P1 = mynum [x];
}
```

* Time delay in 8051:
1. Simple for loop [software delay]:
   a. 8051 design.
   b. Crystal frequency.
   c. Compiler used.

2. Timers. [Hardware delay].

* Note:
Machine cycle: it is the no. of clock pulses required to execute one complete instruction. [1 machine cycle = 12 clock pulses]

* Find the time period of the machine cycle for the 8051 microcontrollers operating at:
1. 11.0592 MHz
2. 16 MHz.
3. 22 MHz.

Ans 1. Machine cycle frequency $= \frac{1}{12}$ crystal freq.

$= \frac{11.0592 \times 10^6}{12}$

delay] :

of clock
e complete
12 clock pulses].

achine
rollers

al freq.

$\times 10^6$

Time period = $\frac{1}{\text{Machine cycle frequency}}$

$= \frac{1}{\quad}$

2. MCF $= \frac{1}{12} \times 16 \times 10^6 =$

Time period $= \frac{1}{MFC} = \frac{1}{\quad}$

3. MCF $= \frac{1}{12} \times 22 \times 10^6 =$

Time period $= \frac{1}{MCF} =$

* Write an 8051 C program to toggle bits of port1 continuously with some delay.

```
//toggle P1 with certain delay
#include <reg51.h>
void main (void)
{
    unsigned char x;
    while(1)
```

```
    {
        p1 = 0x00;
        for(x=0; x<400; x++);
        p1 = 0xFF;
        for(x=0; x<400; x++);
    }
}
```

* write a 8051 C program to toggle the bits of P1 continuously with 250 msec delay.

```
//toggle P1 with 250 msec delay
#include <reg51.h>
void MSDelay(unsigned int);

void main(void)
{
    while(1)
    {
        p1 = 0x00;
        MSDelay(250);
        p1 = 0xFF;
        MSDelay(250);
```

```
    }
}

void MSDelay (unsigned int itime)
{
    unsigned int i,j ;
    for(i=0 ; i<itime ; i++)
        for(j=0 ; j<1275 ; j++) ;
}
```

* Write an 8051 C program to toggle all the bits of P0 & P2 continuously with a 250 msec delay.

```
#include <reg51.h>
void MSDelay (unsigned int) ;

void main (void)
{
    while(1)
    {
        P0 = 0x00 ;
        P2 = 0x00 ;
        MSDelay (250) ;
        P0 = 0xFF ;
```

```
        P2 = 0xFF;
        MSDelay(250);
    }
}

void MSDelay(unsigned int itime)
{
  unsigned int i,j;
  for(i=0; i<itime; i++)
      for(j=0; j<1275; j++);
}
```

* LED's are connected to bits P1 & P2, write an 8051 C program that shows the count from 0 to FFH on the LED's

```
#include <reg51.h>
#define LED P2

void main (void)
{
  P1 = 0x00;    //clear P1.
```

```
    LED = 0 ;          // clear P2
    for( ; ; )         // repeat forever
    {
        P1++ ;         // increment P1.
        LED++ ;        // increment P2.
    }
}
```

Imp * Write an 8051 c program to get a byte of data from P1 wait for 0.5 sec and then send it to P2

```
#include <reg51.h>
void delay (unsigned int);

void main (void)
{
  unsigned char mybyte;
  P1 = 0xFF ; // Configure P1 as input port.

  while(1)
  {
     mybyte = P1;
     delay (500);
```

```
        P2 = mybyte ;
    }
}

void delay (unsigned int itime)
{
    ....
}
```

* Write an 8051 C program to get a byte of data from P0. If it is less than 100 send it to P1 otherwise send it to P2.

```
#include <reg51.h>
void delay (unsigned int);

void main()
{
  unsigned char mybyte;
  P0 = 0xFF;          //Configure P0 as
                        input port.
  while(1)
  {
```

```
        if( mybyte < 100)
                P1 = mybyte;
        else
                P2 = mybyte;
    }
}
```

* Write an 8051 C program to toggle only bit P2.4 without disturbing the other bits of Port 2. [continueously toggle]

```
#include <reg51.h>
sbit   mybit = P2^4.

void   delay (unsigned int);

void   main (void)
{
    while (1)
    {
        mybit = 0;
        delay (200);
        mybit = 1;
        delay (200);
```

```
    }
}

void delay ( unsigned int itime)
{
  ....
}
```

* Write an 8051 C program to monitor bit P1.5, if it is high send 55H to port 0 otherwise AA to port 2.

```
#include <reg51.h>
sbit   mybit  =  P1^5 ;

void  main (void)
{
    mybit  =  1 ;  // Configure P1.5 as input.

    while (1)
    {
       if (mybit == 1)
            P0 = 0x55 ;
       else
```

```
            P2 = 0xAA ;
        }
    }
```

* Write an 8051 C program to turn bit P1.5 on & off 50000 times

* Write an 8051 C program to read the status of bit P1.0, save it & send it to P2.7 continuously.

```
#include <reg51.h>
sbit    inbit  = P1^0 ;
sbit    outbit = P2^7 ;
bit     membit ;

void main (void)
{
    inbit = 1 ; // Configure P1.0 as input.
    while (1)
    {
      membit = inbit ;
      outbit = membit ;
    }
```

* A door sensor is connected to $P_{1.1}$ pin and a buzzer is connected to $P_{1.7}$. Write an 8051 C program to monitor the door sensor and when it opens sound the buzzer, by sending a square wave of few Hz.

```
#include <reg51.h>
sbit Dsensor = P1^1;
sbit Buzzer = P1^7;

void delay (unsigned int);

void main (void)
{
   Dsensor = 1 ;

   while ( Dsensor == 1)
   {
       Buzzer = 0 ;
       delay(200) ;
       Buzzer = 1 ;
       delay ( 200) ;
   }
}
```

```
void delay (unsigned int itime)
{
...
}
```

* Logic operations in C:

1. Bitwise operators:

AND	&	NOT	~
OR	|	EX-OR	^

RIGHT SHIFT >>
LEFT SHIFT <<

* Run the following program on your simulator and check the results:

```
P0 = 0x25 & 0x37;
P1 = 0x90 | 0x3F
P2 = ~P1
P3 = 0x80 ^ 0xFF
```

* Write an 8051 C program to toggle all the bits of P0 & P2 continuously with a 250 millisecond delay. Use the inverting operator to toggle the bits.

* Write an 8051 C program to toggle all the bits of port 0 & port 2 continuously with a 250 msec delay. Use the inverting operator to toggle the bits.

```
#include <reg51.h>
void msdelay (unsigned int);

void main (void)
{
 P0 = 0x55;
 P2 = 0x55;
 while(1)
 {
    P0 = ~P0; //inverting operator
    P2 = P2^0XFF; // ex-or operator
    msdelay(250);
 }
}

void msdelay (unsigned int itime)
{
  ...
}
```

* Write an 8051 C program to read a bit from $P_{1.0}$, invert it and send to $P_{2.7}$

```
#include <reg51.h>
sbit inbit = P1^0;
sbit outbit = P2^7;
bit membit;
void main (void)
{
   while(1)
   {
     membit = inbit;      //get bit from P1.0
     outbit = ~membit;    // invert & sent to P2.7
   }
}
```

* Write an 8051 C program to read $P_{1.0}$ & $P_{1.1}$ bits and issue an ascii character to port 0 according to the following table.

$P_{1.1}$	$P_{1.0}$				
0	0	send	0	to	P_0
0	1	"	1	"	"
1	0	"	2	"	"
1	1	"	3	"	"

```
#include <reg51.h>
void main (void)
{
  unsigned char z;
  z = P1;
  z = z & 0x03;
  switch (z)
  {
    case (0):
    {
      P0 = '0';
      break;
    }

    case(1):
    {
      P0 = '1';
      break;
    }

    case(2):
    {
      P0 = '2';
      break;
    }
```

```
        case(3):
        {
            P0 = '3';
            break;
        }
    }
}
```

* Data Conversion in 8051 'C':

BCD : 0-9
ASCII : ↓
0x30

* Write an 8051 C program to convert packed BCD 0x29 to ASCII and display the bytes on P1 & P2.

```
#include <reg51.h>
void main (void)
{
  unsigned char bcdbyte = 0x29;
  unsigned char x,y;
```

```
    x = bedbyte & 0x0F; // mask higher
    y = bedbyte & 0xF0; // mask lower
    y = y >> 4;

    P1 = x | 0x30 ; // P1 = 0x39
    P2 = y | 0x30 ; // P2 = 0x32.
}
```

* Write an 8051 C program to convert ASCII digits of 4 & 7 to packed BCD and display them on P1.

```
#include <reg51.h>
void main (void)
{
    unsigned char bedbyte;
    unsigned char w = '4' ;
    unsigned char z = '7' ;

    w = w & 0x0F ; //mask higher nibble = 0x04
    z = z & 0x0F ; //  "     "      "   = 0x07.

    w = w << 4; //w = 0x40.
```

```
bedbyte = w|z ; //bedbyte = 0x47.
P1 = bedbyte ;
}
```

* For the given 4 bytes of hexa-decimal data 25H, 62H, 3FH & 52H.

1. Find the checksum byte.
2. Perform the check operation to ensure data integrity.
3. If the 2nd byte 62H has been changed to 22H show how checksum detects the errors.

Ans.

1. Checksum byte :

$$\begin{array}{r} 25H \\ +\ 62H \\ 3FH \\ 52H \\ \hline 118H \end{array}$$

drop the carry from 118H => 18H

Now :

1	8	H
0001	1000	

1's Compliment: 1110 0111 H

Add 1 + 1

1110 1000

⇒ E8H (checksum byte)

2. Checksum operation:

25H
62H
+ 3FH
52H
E8H
200

Ignore the carry - 200 ⇒ 00

As result is 00, data is not corrupted.

3.

25H
+ 22H
3FH
52H
E8H
1C0

Now discard carry : 1C0 ⇒ C0

As result is non zero, data is corrupted.

* Write an 8051 C program to calculate the check sum byte for the data given in prev example.

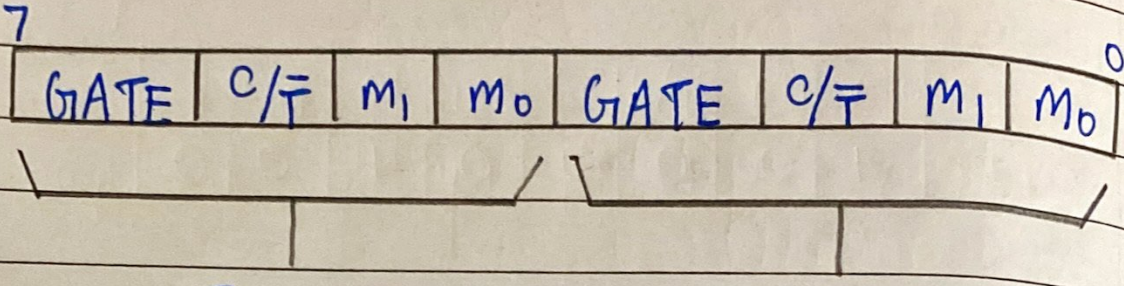
```
#include <reg51.h>
 void main ()
{
   unsigned char mybyte [] = { 0x25, 0x62,
                               0x3F, 0x52 };
   unsigned char sum = 0;
   unsigned char x, checksumbyte;

   for (x=0; x<4; x++)
    {
      sum = sum + mybyte[x]; //checksum byte
    }
    checksumbyte = ~sum +1;
}
```

* Write an 8051 C program to perform checksum operation to ensure data integrity if data is good send ASCII character 'G' to port 0 otherwise send 'B' to port 0.

```
#include <reg51.h>
void main()
{
  unsigned char mybyte[] = {0X25, 0X62,
                            0X3F, 0X52,
                            0XE8};

  unsigned char chksum = 0;
  unsigned char x;

  for(x=0; x<5; x++)
  {
    chksum = chksum + mybyte[x];
  }

  if(chksum == 0)
  {
    P0 = 'G';
  }
  else
  {
    P0 = 'B';
  }
}
```

11 split timer mode

* TMOD :

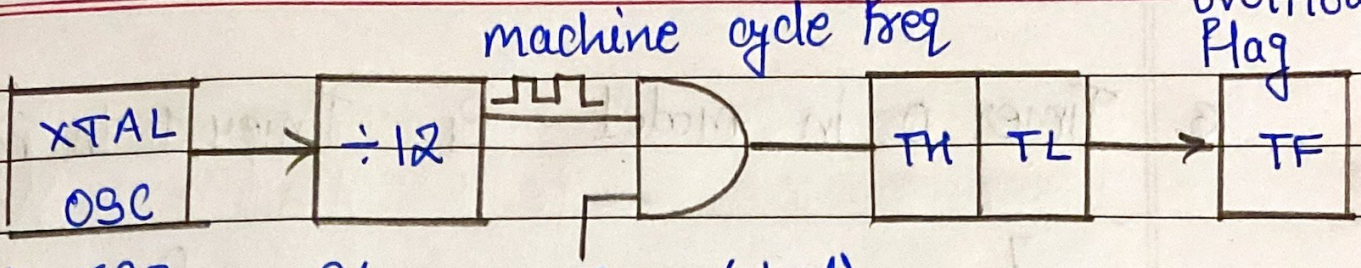


if gate = 1; h/w, INTx → INT0 → Timer 0
INT1 → Timer 1
if gate = 0; s/w, TRx → TR0 → Timer 0
TR1 → Timer 1.

For M1 & M0 :

M1	M0	Timer
0	0	mode 0 (13-bit)
0	1	mode 1 (16-bit)
1	0	mode 2 (8-bit)
1	1	mode 3 (split timer mode)

* Timer MODE1 (16 bit) :



Machine cycle = 12 clock pulses.

* Configure

1. Timer 0 in mode 1.

Ans.

	Timer1				Timer0			
7	G	C/T	M_1	M_0	G	C/T	M_1	M_0	0
	0	0	0	0	0	0	0	1	

TMOD = 0X01

2. Timer 1 in mode 2

Ans

7	G	C/T	M_1	M_0	G	C/T	M_1	M_0	0
	0	0	1	0	0	0	0	0	

TMOD = 0X20

3. Timer 0 in mode1 & Timer 1 in mode 2.

Ans.

7							0
G	C/T	M_1	M_0	G	C/T	M_1	M_0
0	0	1	0	0	0	0	1

TMOD = 0X21.

* Note :

Whenever a single timer is configured in a particular mode by default the other unused timer will be configured in mode0.

* Find the timers clock frequency and its time period for various 8051 based systems with the following crystal frequencies:

1. 11.0592 MHz = XTAL Freq.

Calculate M/c freq & time period.

Ans. M/c cycle frequency = $\frac{\text{crystal freq}}{12}$

$= \frac{11.0592}{12}$

$= 921.6$ KHz.

time period = $t = \frac{1}{\text{m/c cycle freq}}$

$= \frac{1}{921.6\,KHz}$

$= 1.085\,\mu sec.$

2. XTAL freq = 22 MHz.
Calculate m/c frequency & Time period.

Ans. m/c cycle frequency = $\frac{\text{crystal freq}}{12}$

$= \frac{22}{12}$

$= 1.833\,KHz$

Time Period = $\frac{1}{\text{m/c cycle frequency}}$

$= \frac{1}{1.833}$

$= 0.5455\,\mu sec.$

* Questions :

* Generate 25 ms time delay using timer 1 in mode 1. Assume XTAL = 11.0592 MHz.

Ans. Required delay $= [(FFFF - \underbrace{YYXX}_{\text{count to be loaded}}) + 1] \times$ time period

TL ↑ YYXX ↓ TH

→ roll back

Time period $= \frac{1}{\text{m/c cycle Freq.}} = \left(\frac{1}{\left(\frac{11.0592}{12}\right)}\right) = 1.0854\mu s$

$\therefore \quad 25 = [(FFFF - YYXX) + 1] \times 1.085$

$\frac{25}{1.085} = [(FFFF - YYXX) + 1]$

$23041 - 1 = FFFF - YYXX$

$23040 = FFFF - YYXX$

$\therefore YYXX = FFFF - 5A00$

$= A5FF.$

Obtain the maximum possible delay using timer 0 in mode 1. Assume XTAL = 11.0592 MHz.

Required delay = [(FFFF − YYXX) + 1] × 1.085

∴ To get max delay put YYXX as 0000

∴ Required delay = [(FFFF − 0000) + 1] × 1.085
= (65535 + 1) × 1.085 μsec.
= 71106.56 sec
= 71.1 msec

Obtain the values to be loaded into TH & TL register to generate a delay of 10ms.

Required delay = [(FFFF − YYXX) + 1] × time period

10^{-3} / 5×10^{-6}

10 = [(FFFF − YYXX) + 1] × 1.085 μsec.

(9217 − 1) = FFFF − YYXX (decimal to hexa.)

YYXX = FFFF − 2400
= DBFE (OBFF).

mode1 - 16bit mode.
mode2 - 8bit mode.

* Calculate the delay required for the given count 5C00 using timers in mode 1.

Ans. Required delay = [(FFFF - YYXX) + 1] × time period

= [(FFFF - 5C00) + 1] × 1.085 μsec

= (A3FF + 1) × 1.085 μsec

= (41983 + 1) × 1.085

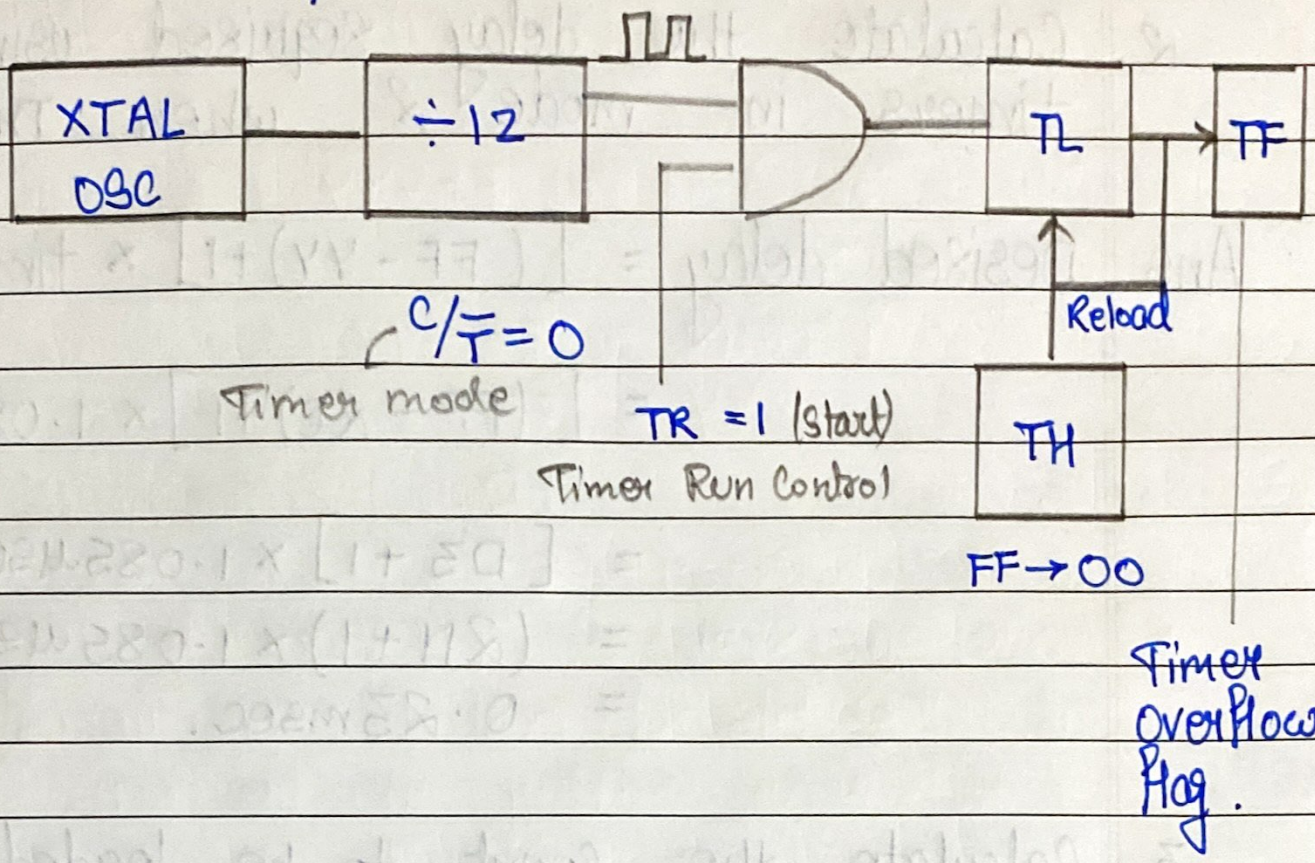
= $\frac{45552.64}{1000}$ msec

= 46.5 msec.

* Note:
Formula to calculate delay in decimal:

delay = [65536 - NNNNN] × time period.

* Mode 2 : (8-bit mode) :
(Auto Reload).



* Problems :

1. Calculate the max. delay obtained, using timers in mode 2.

Ans. Desired delay = [(FF - YY) + 1] × time period.

To obtain max. delay put YY as 00.

Desired delay = [(FF - 00) + 1] × 1.085 μsec.
= (255 + 1) × 1.085 μsec

= 0.27 msec.

2. Calculate the delay sequised using timers in mode 2 when TH = 2C.

Ans. Desired delay = [(FF - YY) + 1] × time period.

= [(FF - 2C) + 1] × 1.085 μsec.

= [D3 + 1] × 1.085 μsec

= (211 + 1) × 1.085 μsec

= 0.23 msec.

3. Calculate the count to be loaded into TH register to obtain a delay of 50 μsec. Use timers in mode 2. Assume XTAL = 11.0592 MHz

Ans. Desired delay = [(FF - YY) + 1] × time period.

$50\,\mu sec = [(FF - YY) + 1] \times 1.085\,\mu sec$

$\frac{50\,\mu sec}{1.085\,\mu sec} = (FF - YY) + 1.$

$46-1 = FF-YY$

$YY = FF-45.$

$= FF-2D$

$= D2.$

$\therefore TH = 0XD2.$

4. Calculate the count to be loaded into TH register to obtain a delay of 500 μsec using timers in mode 2. Assume XTAL = 11.0592 MHz

Ans.

$500\ \mu sec = [(FF-YY)+1] \times 1.085\ \mu sec$

$461-1 = FF-YY$

$YY = FF = 460$

$= FF - 1CC$

$= FFFFFF33$

$\frac{500}{1.085} = 460.82$

> 0.27

$0.27 msec$

$255 - 278\ \mu sec$? Hence not possible.

[Not possible to generate 500μsec delay using mode2].

5. Same abv problem use delay of 3μsec.

FF - 1.76

$\rightarrow FF-2 \Rightarrow FD$

Ans. Desired delay = [(FF - YY) + 1] × time period

$3\mu sec = [(FF - YY) + 1] \times 1.085\mu sec.$

$\left(\frac{3}{1.085}\right) - 1 = FF - YY.$

$1.7649 = FF - YY$

$\therefore YY = FF - 02$

$= FD.$

* Write an 8051 C program to toggle all the bits of port 1 continuously with some delay in between. Use Timer 0 in mode 1 to generate the delay.

```
#include <reg51.h>
void TODelay(void);
void main(void)
{
   while(1)
   {
      P1 = 0x00;
      TODelay();
```

```
        P1 = 0xFF;
        TODelay();
    }
}

void TODelay()
{
    TMOD = 0X01  ; // timer 0 in mode 1.
    TL0  = 0XFE  ;
    TH0  = 0XBD  ; // delay size unknown.
    TR0  = 1     ; // start the timer.
    while (TF0 == 0); // moniter timer overflow
                      flag.
    TR0 = 0;
    TF0 = 0;
}
```

* Write an 8051 C program to toggle only bit $P_{1.5}$ continuously every 50msec, use timer 0 in mode1.

```
#include <reg51.h>
void TODelay(void);
sbit mybit = P1^5;
```

200 msec delay

```
void main (void)
{
  while (1)
  {
   mybit = ~mybit ;  //using NOT operator
   TODelay (1);      //to toggle P1.5.
  }
}

void TODelay (void)
{
  TMOD = 0x01 ;   //timer 0 in mode1.
  TH0  = 0x4B ;   //count for 50msec delay
  TL0  = 0xFE ;
  TR0 = 1 ;       //start the timer.
  while (TF0 == 0) ;
  TR0 = 0;
  TF0 = 0;
}
```

* Write an 8051 C program to toggle all bits of P2 continuously every 25msec use timer 1 in mode 1.