SYLLABUS

Unit 3:

The 8051 Microcontrollers: Microcontrollers and Embedded processors, Overview of 8051 family. 8051 Microcontroller hardware, Input/output pins, Ports, and Circuits, External Memory.

8051 Programming in C:
Data Types and time delay in 8051 C, I/O Programming, Logic operations, Data conversion Programs.

Reference book:

1. The 8051 Microcontroller and Embedded Systems by Muhammad Ali Mazidi
2. Embedded Systems by Rajkamal

The Difference Between an Embedded Processor & Microcontroller:

As computer systems become more and more ubiquitous in everyday items, the workings of those items will require some sort of digital controlling. Controllers that once handled large mechanical systems now work inside smaller digital and mechanical devices. These microcontrollers are not the same as embedded computer processors, however. While sharing similar functionality, both integrate into their roles in computers in a different way, and for different reasons.

Controllers: Controllers are special pieces of digital equipment that control some aspect of their environment, according to responses from that environment. Typical types of controllers include those for home temperature regulation systems or security systems. A controller can be thought of as a self-contained computer that controls another system. Often, older controllers were large pieces of machinery contained externally from the system they controlled.

Microcontrollers : As computer miniaturization progressed through the mid to late 1900s, controllers became smaller and smaller. All the parts of a controller, including memory and I/O devices, became integrated with the controller as a standard. Finally, when the entire controller apparatus was able to fit on single chips, they became known as "microcontrollers." A microcontroller contains everything required to control an external system, and nothing else. This limiting of microcontroller functionality to the basic requirements of a control unit makes implementing microcontrollers cheap and easy.

Embedded Processors

An "embedded" processor is simply a computing device placed inside a system it controls. A processor embedded into a system handles all the computation and logical operation of a computer. The embedded processor also handles such tasks as storing and retrieving data from memory, and

processing data from any inputs or outputs. Embedded processors often work as part of a computer system, alongside memory and I/O devices.

Differences

The primary difference between microntrollers and embedded processors is makeup and integration. Embedded processors, while in a sense "controlling" the system they are a part of, require external resources such as RAM and registers in order to do so. A processor is not a control "system." Microcontrollers, on the other hand, contain everything required to control a system in a single chip. A microcontroller might contain an embedded processor as part of its makeup, but also combines other computer parts, such as memory and signal registers, in a single chip.

An embedded product uses a microprocessor (or microcontroller) to do one task and one task only

There is only one application software that is typically burned into ROM

A PC, in contrast with the embedded system, can be used for any number of applications

It has RAM memory and an operating system that loads a variety of applications into RAM and lets the CPU run them

A PC contains or is connected to various embedded products
Each one peripheral has a microcontroller inside it that performs only one task.

<u>Types of Microcontroller :</u>

A. 8-bit microcontrollers:
    i. Motorola’s 6811
    ii. Intel’s 8051
    iii. Zilog’s Z8
    iv. Microchip’s PIC
B. There are also 16-bit and 32-bit microcontrollers made by various chip makers.

Criteria for Selecting Microcontroller:

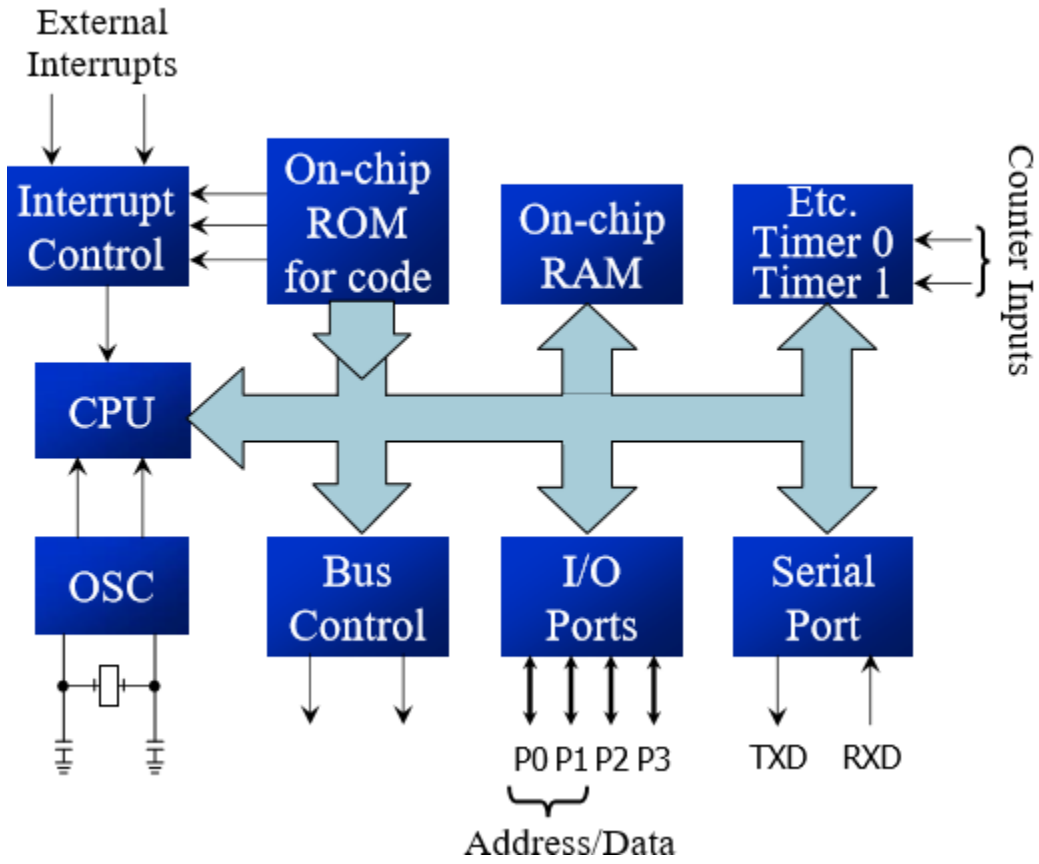
1. Meeting the computing needs of the task at hand efficiently and cost effectively. This includes:
   i. Speed
   ii. Packaging
   iii. Power consumption
   iv. Amount of RAM and ROM on chip.
   v. Number of I/O pins and timer on the chip.
   vi. Upgradability
   vii. Cost per unit.
2. Availability of software development tools such as compiler, assemblers and debuggers.
3. Wide availability and reliable source of the microcontroller.

Overview of 8051 Family:

- Intel introduced 8051, referred as MCS- 51, in 1981 which was a 8 bit microprocessor which had following features:
  The CPU can work on only 8 bits of data at a time
  The 8051 had
  i. 128 bytes of RAM
  ii. 4K bytes of on-chip ROM
  iii. Two timers
  iv. One serial port
  v. Four I/O ports, each 8 bits wide
  vi. 6 interrupt sources

- The 8051 became widely popular after allowing other manufactures to make and market any flavour of the 8051, but remaining code-compatible

- Below diagram shows overview architecture diagram of 8051



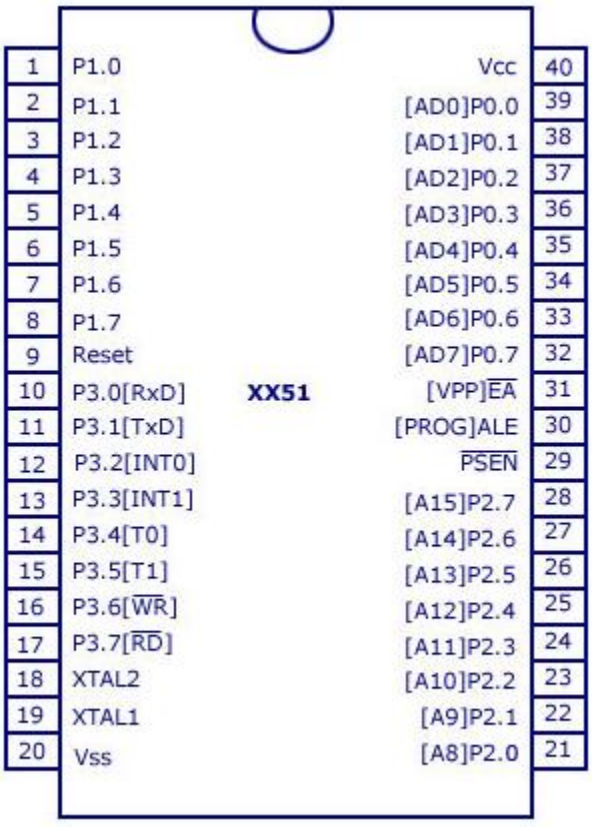
8051 Microcontroller Hardware:

Referring the above architecture diagram of 8051 it consist of the following hardware units:

1. It has 12 MHz clock. Processor instruction cycle time is 1 µs.
2. It has an 8 bit ALU with internal bus width of 8 bit.
3. It has CISC architecture.
4. It has special instruction to manipulate individual bits.
5. It has program counter. Initial default reset value of PC defined in the processor is 0x0000
6. It has stack pointer. Initial default reset value of SP defined in the processor is 0x07.

7. It has Harvard memory architecture. Program memory and data memory have separate address spaces from 0x0000 and separate control signals.
8. It has on-chip RAM of 128b bytes. 8052 version provides RAM of 256 bytes. 32 bytes of RAM are also used as four bank(sets) of registers. Each register – set (bank) thus has eight registers.
9. Special Function Register(SFR's) are the special registers present in 8051 family. SFR's function as PSW
10. It has two external interrupt pins INT0 and INT1.
11. It has four ports of 8 bits each in single chip mode. It has two timers T0 and T1. It also ha serial interface(SI)
12. Classic version of 8051 does not consist of PWM, MODEM ,WATCHDOG timer and ADC.

8051 Inputs/Outputs Pins,Ports and Circuit:



Pin Diagram of 8051

Function of the 8051 Pin:

- **Pin-40 :** Named as Vcc is the main power source. Usually its +5V DC.
- **Pins 32-39:** Known as Port 0 (P0.0 to P0.7) – In addition to serving as I/O port, lower order address and data bus signals are multiplexed with this port (to serve the purpose of external memory interfacing). This is a bi directional I/O port (the only one in 8051) and external pull up resistors are required to function this port as I/O.
- **Pin-30:-** ALE aka Address Latch Enable is used to demultiplex the address-data signal of port 0 (for external memory interfacing.) 2 ALE pulses are available for each machine cycle.
- **Pin-31:-** EA/ External Access input is used to enable or disallow external memory interfacing. If there is no external memory requirement, this pin is pulled high by connecting it to Vcc.
- **Pin- 29:-** PSEN or Program Store Enable is used to read signal from external program memory.
- **Pins- 21-28:-** Known as Port 2 (P 2.0 to P 2.7) – in addition to serving as I/O port, higher order address bus signals are multiplexed with this quasi bi directional port.
- **Pin 20:-** Named as Vss – it represents ground (0 V) connection.
- **Pins 18 and 19:-** Used for interfacing an external crystal to provide system clock.
- **Pins 10 – 17:-** Known as Port 3. This port also serves some other functions like interrupts, timer input, control signals for external memory interfacing RD and WR , serial communication signals RxD and TxD etc. This is a quasi bi directional port with internal pull up.
- **Pin 9:-** RESET pin is used to set the 8051 microcontroller to its initial values, while the microcontroller is working or at the initial start of application. The RESET pin must be set high for 2 machine cycles.
- **Pins 1 – 8:-** Known as Port 1. Unlike other ports, this port does not serve any other functions. Port 1 is an internally pulled up, quasi bi directional I/O port.

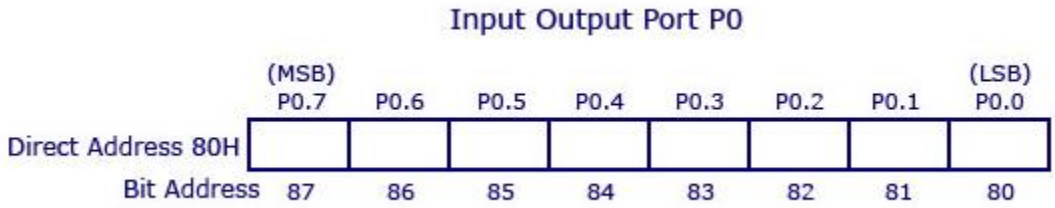
Function of the 8051 Ports:

8051 has 4 Input/Output ports named P0, P1, P2 and P3 has got four corresponding port registers with same name P0, P1, P2 and P3. Data must be written into port registers first to send it out to any other external device

through ports. Similarly any data received through ports must be read from port registers for performing any operation.

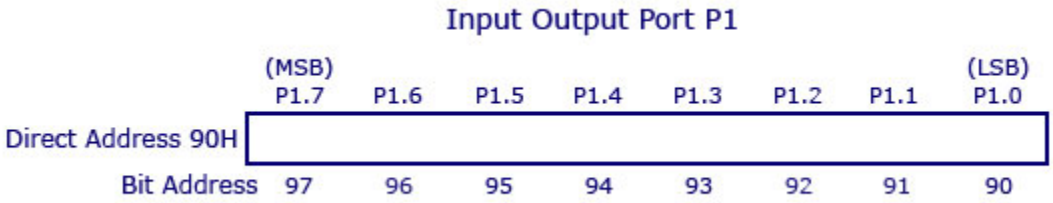
i. Port 0 (P0.0 to P0.7) :

- In addition to serving as I/O port, lower order address and data bus signals are multiplexed with this port (to serve the purpose of external memory interfacing). This is a bi directional I/O port (the only one in 8051) and external pull up resistors are required to function this port as I/O.



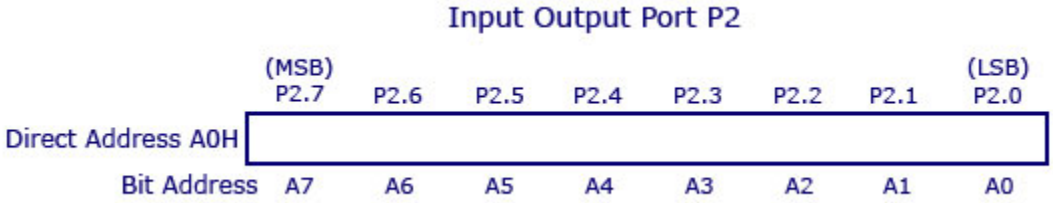
ii. **Port 1 (P1.0 to P1.7) :**

- It is 8-bit bi-directional I/O port. It is bit/ byte addressable. When logic '1' is written into port latch then it works as input mode. It functions as simply I/O port and it does not have any alternative function.



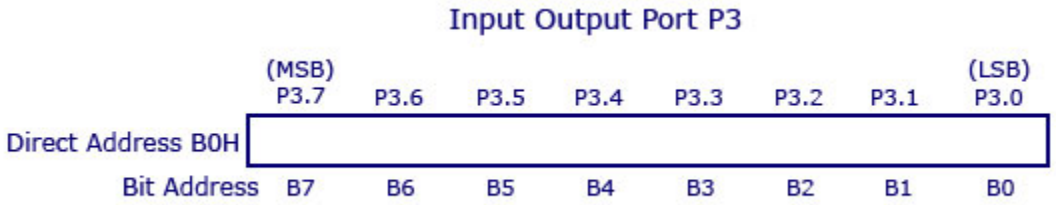
iii. **Port 2 (P2.0 to P2.7) :**

- It is 8-bit bi-directional I/O port. It is bit/ byte addressable. During external memory access it functions as higher order address bus (A8-A15).



iv. **Port 3 (P3.0 to P3.7) :**

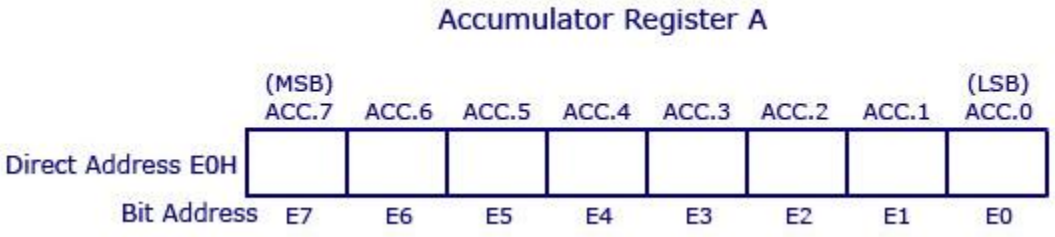
- This port also serves some other functions like interrupts, timer input, control signals for external memory interfacing RD and WR , serial communication signals RxD and TxD etc. This is a quasi bi directional port with internal pull up.



Function of the 8051 Special Function Registers(SFR):

i. Register A/Accumulator:

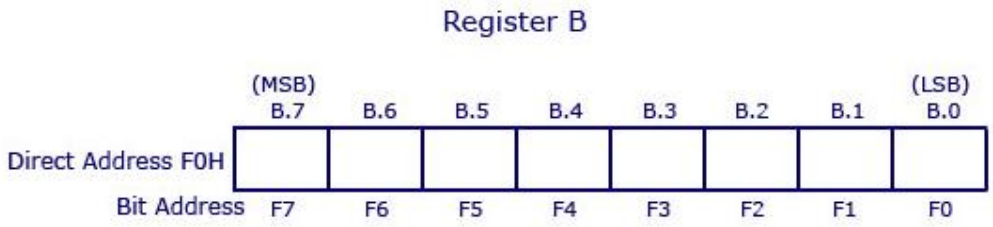
- The most important of all special function registers is **Accumulator** which is also known as **ACC** or **A.**
- The Accumulator (sometimes referred to as Register A also) holds the result of most of arithmetic and logic operations. ACC is usually accessed by direct addressing and its physical address is **E0H.**
- Accumulator is ***both byte and bit addressable(refer diagram below)***
- To access the first bit (i.e bit 0) or to access accumulator as a single byte (all 8 bits at once), you may use the same physical address E0H. Now if you want to access the second bit (i.e bit 1), you may use E1H and for third bit E2H and so on.



ii. Register B:

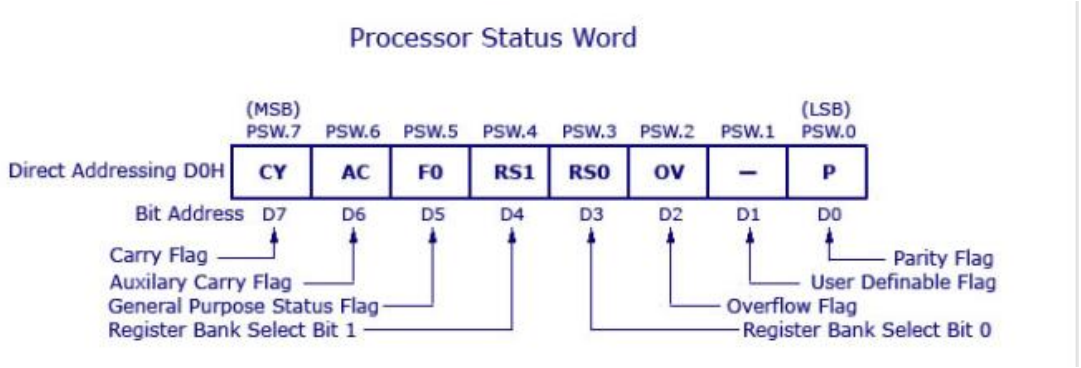
- The major purpose of this register is in executing multiplication and division. The 8051 micro controller has a single instruction for multiplication **(MUL)** and division **(DIV)**.

- If you are familiar with 8085, you may now know that multiplication is repeated addition, where as division is repeated subtraction. While programming 8085, you may have written a loop to execute repeated addition/subtraction to perform multiplication and division. Now here in 8051 you can do this with a single instruction.
- Register B is also byte addressable and bit addressable. To access bit o or to access all 8 bits (as a single byte), physical address F0 is used. To access bit 1 you may use F1 and so on. Please take a look at the picture below.

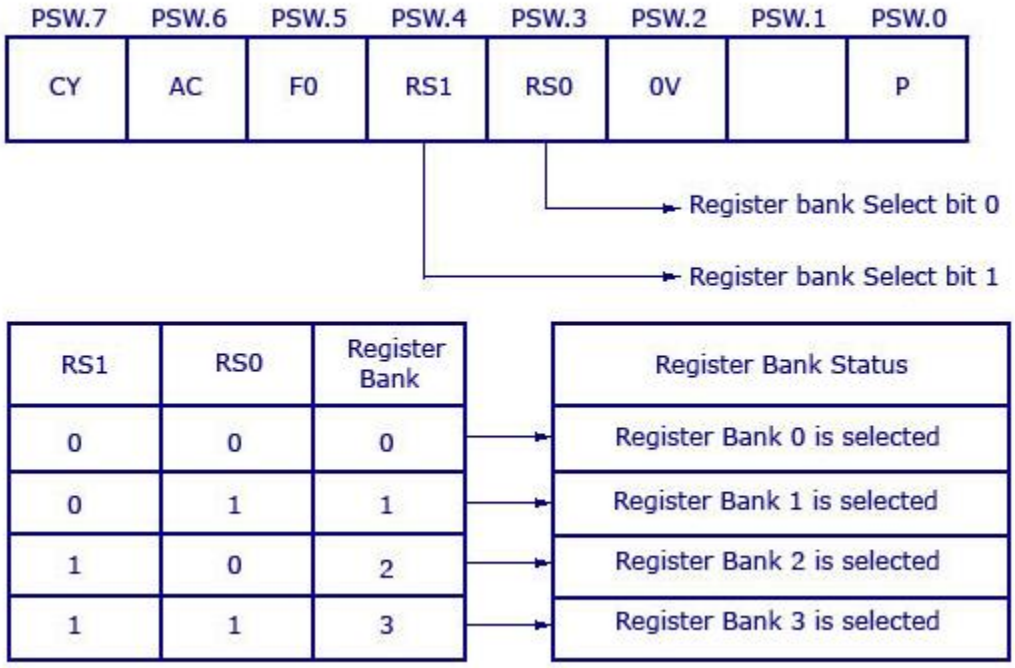


iii. Processor Status Word (PSW)

- Commonly known as the PSW register, this is a vital SFR in the functioning of micro controller.
- This register reflects the status of the operation that is being carried out in the processor.
- The diagram below shows PSW register and the way register banks are selected using PSW register bits – RS1 and RS0. PSW register is both bit and byte addressable.
- The physical address of PSW starts from D0H. The individual bits are then accessed using D1, D2 … D7. The various individual bits are explained below.



- At a time registers can take value from R0,R1…to R7. You may already know there are 32 such registers.
- ***So how you access 32 registers with just 8 variables to address registers?*** Here comes the use of register banks. There are 4 register banks named 0,1,2 and 3.
- Each bank has 8 registers named from R0 to R7. At a time only one register bank can be selected. Selection of register bank is made possible through PSW register bits PSW.3 and PSW.4, named as RS0 and RS1.These two bits are known as register bank select bits as they are used to select register banks.

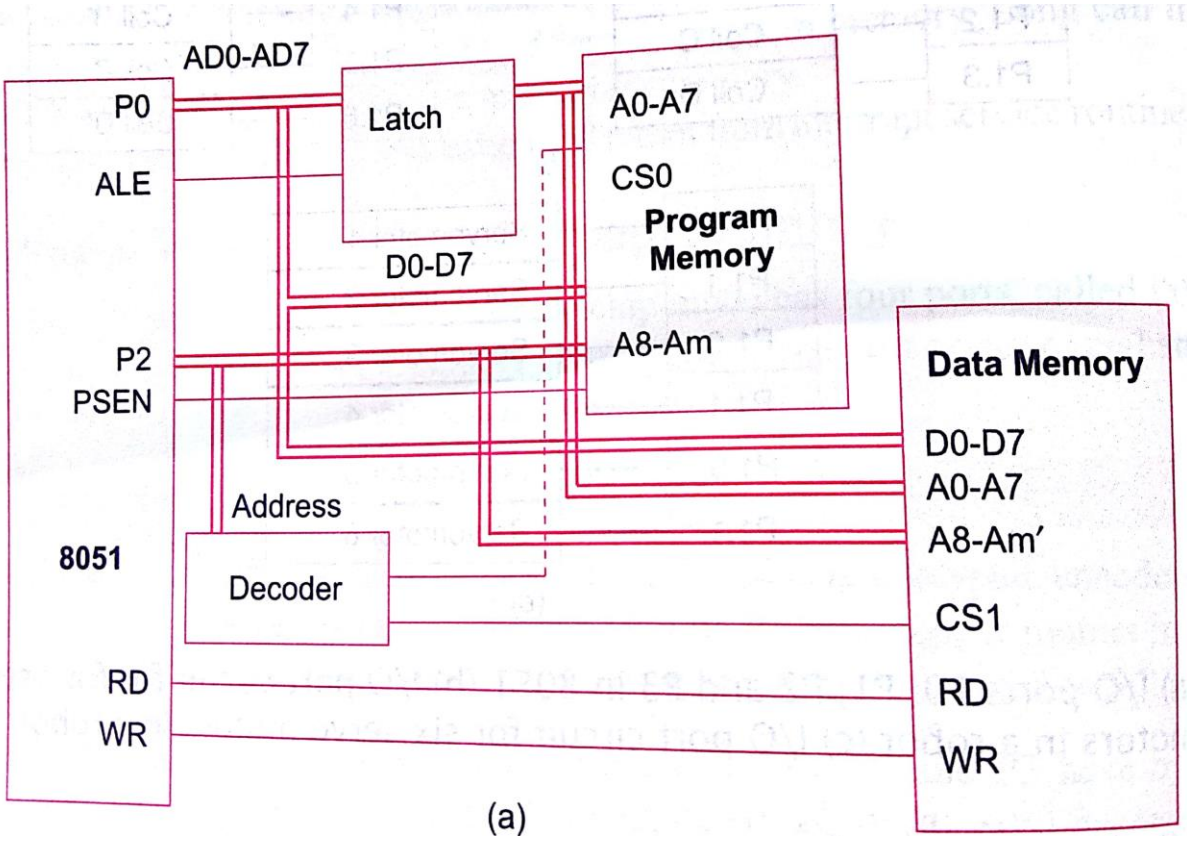


RS1	RS0	Register Bank	Register Bank Status
0	0	0	Register Bank 0 is selected
0	1	1	Register Bank 1 is selected
1	0	2	Register Bank 2 is selected
1	1	3	Register Bank 3 is selected

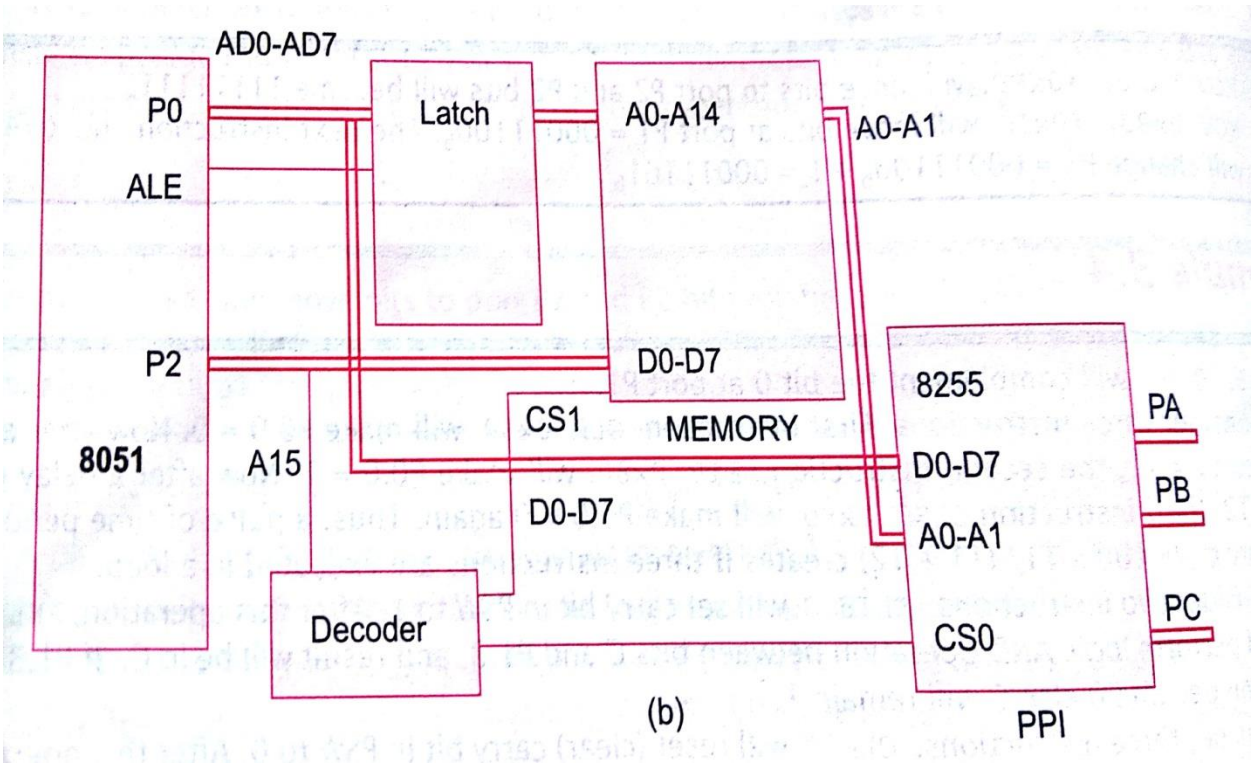
External Memory:

- There are two sets of memory:
  - **a.** Program Memory
  - **b.** Data Memory.
- Figure (a) shows interfacing the external program memory and data memory in 8051.

- Processor has two control signal PSEN and RD to control read from program or data memory.
- Processor has control signal ALE to control use of AD0 – AD7 as address or data at given instances.
- 8051 also has memory mapped I/O. It means memory and ports are assigned the addresses such that each have distinct range of address in the data memory address space.
- Therefore interfacing circuit design is identical to that for the memory when 8051 connects to external ports and parallel peripheral interface(PPI) chip Intel 8255.
- External memory and ports are assigned separate distinct address in 8051.
- Fig(b) shows interfacing of PPI intel 8255 with 8051.



(a)



Data Types in 8051 C:

- The main aim to use C language in writing program for 8051 is to create smaller hex files.
- Hence data types for 8051 is same as data types of C programming.

Sr .No	Data Type	Size in bits	Data Range/Usage
1	unsigned char	8 bit	0 to 255
2	signed char	8 bit	-128 to +127
3	unsigned int	16 bit	0 to 65535
4	signed int	16 bit	-32,768 to +32767
5	sbit	1- Bit	SFR bit addressable only
6	bit	1- Bit	RAM bit addressable only
7	sfr	8 bit	RAM addresses 80 – FF H only

Program on Data Types:

Q1. WAP to send values 00 to FF to port P1

Program:

```
#include <reg51.h>
 void main(void)
      {
            unsigned char z;
            for ( z = 0 ; z < = 255; z++)
            P1 = z;
      }
```

Q2. WAP to send hex values for ASCII characters of 0,1,2,3,4,5,A,B,C and D to port P1.

Program:

```
#include <reg51.h>
 void main(void)
      {
            unsigned char mynum [] = "012345ABCD";
            unsigned char z;
            for ( z = 0 ; z < = 10; z++)
            P1 = mynum[z];
      }
```

Output:

30H, 31H, 32H, 33H, 34H, 35H, 41H, 42H, 43H, 44H

(the above values are hex values of 0,1,2,3,4,5,A,B,C and D)

Q3. WAP to send values of -4 to +4 to port P1

Program:

```
// signed numbers
#include <reg51.h>
 void main(void)
      {
            char mynum [] = {+1, -1, +2, -2, +3, -3, +4, -4};
```

```
        signed char z;
        for ( z = 0 ; z < = 8; z++)
        P1 = mynum[z];
    }
```

Output:

1, FFH, 2, FEH, 3, FDH, 4, and FCH
(the above values are hex values of +1, -1, +2, -2, +3, -3, +4, and -4)

Q4. WAP to toggle bit D0 of the port P1(P1.0) 50,000 times

Program:

```
#include <reg51.h>
sbit MYBIT = P1^ 0;     // sbit is declared outside of main
void main(void)
    {
        unsigned int z;
        for ( z = 0 ; z < = 50000; z++)
        {
            MYBIT = 0;
            MYBIT = 1:
        }
    }
```

Time Delay in 8051:

There are two ways to create a time delay in 8051 C

i. Using a simple "for" loop
ii. Using 8051 timers

Time delay using "for" loop:

While using "for" loop to create time delay following factors should be considered to have accurate delay:

i. 8051 design is one of the important consideration while using "for" loop.

The 8051 designed in 1990 used 12 clock periods per machine cycle however new generation uses fewer clocks per machine cycle.
e.g. DS5000 uses 4 clock periods per machine cycle while DS89C420 uses only one clock per machine cycle.

ii. The crystal frequency connected to the X1 – X2 inputs pins. The duration of the clock period for the machine cycle is a function of this crystal frequency.

iii. Compiler choice is another important factor that needs to consider. In case C program , compiler converts the C statement and functions to assembly language instruction . As a result different compiler produces different code.

Program on Time Delay using "for" loop:

Q1. WAP to toggle bits of P1 continuously forever with some time delay.

Program:

// Toggle P1 forever with some time delay between "ON" and "OFF"

```
#include <reg51.h>
void main(void)
    {
        unsigned int x;
        for( ; ;)       //repeat forever
        {
            P1 = 0x55;
            for ( x = 0; x < 40000; x++) ; // delay size unkown
            P1 = 0xAA;
            for ( x = 0; x < 40000; x++) ;
```

```
        }
    }
```

Q2. WAP to toggle al the bits of P0 and P2 continuously with 250 ms time delay.

Program:

// This program is tested for the DS89C420 with XTAL = 11.0592 Mhz

```
#include <reg51.h>
void MSDelay(unsigned int);
void main(void)
    {
        while(1)            //another way to do forever delay
        {
            P0 = 0x55;
            P2 = 0x55;
            MSDelay(250);
            P0 = 0xAA;
            P2 = 0xAA;
            MSDelay(250);
        }
    }
void MSDelay(unsigned int itime)
```

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

Time delay using 8051 Timer:

In 8051 C we can access the timer register TH,TL and TMOD directly using the reg51.h header file.

Program on Time Delay using Timer:

WAP to toggle all bits of port P1 continuously with some time delay in between. Use Timer 0,16 bit mode to generate the delay

Program:

```
#include <reg51.h>
void T0 Delay (void);
void main(void)
{
    while(1)            //repeat forever
    {
        P1 = 0x55;   //toggle all the bits of P1
        T0Delay( );  //delay size unknown
        P1 = 0xAA; //toggle all the bits of P1
    }
}
```

```
void T0Delay( )
{
        TMOD = 0x01;      // Timer 0, Mode 1
        TL0 = 0x00;       // load TL0
        TH0 = 0x35;       // load TH0
        TR0 = 1;          // turn ON T0
        while(TF0 = = 0); // wait for TF0 to roll over
        TR0 = 0;          // turn OFF T0
        TF0 = 0;          // clear TF0
}
```

Source: https://www.youtube.com/watch?v=pAKm7TCp7es

I/O Programming in 8051 C:

Q1. WAP to get a byte of data from P1, wait ½ second and then send it to P2.

Program:

```
#include <reg51.h>

void MSDelay (unsigned int);

void main (void)

{

        unsigned char mybyte;

        P1 = 0xFF;              //make P1 an input port

        while(1)

        {

                mybyte = P1;          //get a byte from P1

                MSDelay(500);

                P2 = mybyte;          //send it to P2

        }

}

void MSDelay(unsigned int itime);

        {

        unsigned int i , j;

        for ( i = 0; i < itime; i++) ;

        for ( j = 0; i < 1275; j++) ;

        }
```

Q2. WAP to get a byte of data from P0.If it is less than 100, send it to P1,otherwise send it to P2

Program:

```
#include <reg51.h>

void main (void)

{

        unsigned char mybyte;

        P0 = 0xFF;              //make P0 an input port

        while(1)

        {

                mybyte = P0;          //get a byte from P0

                if (mybyte < 100)

                        P1 = mybyte;          //send it to P1 if less than 100

                else

                        P2 = mybyte;          //send it to P2 if more than 100

        }

}
```

Q3. WAP to toggle only bit P2.4 continuously without disturbing the rest of the bits of P2.

Program:

```
#include <reg51.h>

sbit mybit = P2^4              //notice the way single bit is declared

void main(void)
```

```
{
        while(1)
        {
                mybit = 1;              //turn ON P2.4
                mybit = 0;              //turn OFF P2.4
        }
}
```

Logical Operation in 8051 C:

1. Bitwise Logical Operator in C:

		AND	OR	EX-OR	INVERTER
A	B	A & B	A | B	A ^ B	Y = ~ B
0	0	0	0	0	1
0	1	0	1	1	0
1	0	0	1	1	
1	1	1	1	0	

Example: WAP to toggle all the bits of P0 and P2 continuously with 250 ms delay. Using the inverting operator.

Program:

```
#include <reg51.h>
```

```
void MSDelay(unsigned int);
void main(void)
    {
        P0 = 0x55;
        P2 = 0x55;
        while(1)            //another way to do forever delay
        {
            P0 = ~P0 ;
            P2 = ~P2 ;
            MSDelay(250);
         }
        }

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

2. **Bitwise Logical Shift Operator in C:**
   There are two bit wise shift operator in C
   a. Shift right (>>)
   b. Shift left (<<)

   Following are the example of shift operator in C

a. 0x9A >> 3 = 0x13 /*shifting right 3 times*/
b. 0x77 >> 4 = 0x07 /*shifting right 4 times*/
c. 0x6 << 4 = 0x60 /*shifting left 4 times*/

Data Conversion Program:

Q1. WAP to convert packed BCD 0x29 to ASCII and display the bytes on P1 and P2.

Program:

```
#include<reg51.h>

void main(void)

{

    unsigned char x , y ;

    unsigned char mybyte = 0x29;

    x = mybyte & 0x0F ;       // mask upper 4 bits

    P1 = x | 0x30 ;           // make it ASCII

    y = mybyte & 0xF0 ;       // mask lower 4 bits

    y = y >> 4 ;              // shift it to lower 4 bits
    P2 = y | 0x30 ;           // make its ASCII

}
```

Q2. WAP to convert ASCII digits of 4 and 7 to packed BCD and display them on P1.

Program:

```
#include<reg51.h>

void main(void)
```

```
{
        unsigned char bcdbyte ;
        unsigned char w = '4' ;
        unsigned char z = '7' ;
        w = w & 0x0F ;      // mask 3
        w = w << 4 ;        // shift left to make upper BCD digit
        z = z & 0x0F ;      // mask 3
        bcdbyte = w | z ;   // combine to make packed BCD
        P1 = bcdbyte ;
}
```

Q3. WAP to convert 11111101(FD hex) to decimal and display the digits on P0,P1 and P2.

Program:

```
#include <reg51.h>
void main(void)
{
        unsigned char x , binbyte , d1 , d2 , d3 ;
        binbyte = 0xFD ;            //binary (hex) byte
        x = binbyte / 10 ;          // divide by 10
        d1 = binbyte % 10 ;         // find remainder(LSD)
        d2 = x % 10 ;               // middle digit
        d3 = x / 10 ;               // find MSD
        P0 = d1 ;
```

```
    P1 = d2 ;

    P2 = d3 ;

}
```

Graded Question:

1. Write an 8051 C-program to read the P1.0 and P1.1 bits and issue an ASCII character to P0 to according to the following table:

P1.1	P1.0	
0	0	Send '0' to P0
0	1	Send '1' to P0
1	0	Send '2' to P0
1	1	Send '3' to P0

2. Write an 8051 C program to calculate the checksum byte for the data given below:
   25H,62H,3FH and 52H.
3. Explain interfacing of external program memory and data memory in 8051.
4. Explain PSW in details.
5. WAP to toggle bit D0 of the port P1(P1.0) 50,000 times.

MCQ:

1. 8051 has _____ on chip ROM
   a. 4kb
   b. 2kb
   c. 8kb
   d. 16kb

2. 8051 is _____ architecture.
   a. Harvard
   b. Von-Neuman
   c. Both
   d. None of the above.
3. Pin no. of ALE is _____
   a. 15
   b. 30
   c. 29
   d. 26
4. Direct address of reg.A is _____
   a. 40H
   b. 80H
   c. 90H
   d. A0H
5. Sbit is _____ size bit.
   a. 1bit
   b. 2bit
   c. 4bit
   d. 8bit.