IOT DEVELOPMENT USING EMBEDDED C WITH 8051 MC

CSE DEPT

IGDTUW

CONTENTS:

CHAPTER-1: INTRODUCTION

Computer is a device which takes raw data as input from users, processes them through a set of instructions and gives the output

Internet of Things is an embedded system which is connected to the internet world.

Embedded system is a large system which has a particular function to perform in a large system. It has computer hardware and software embedded in it. It is a sub-system in a large system. The hart of the embedded system is a Microcontroller[MC]. A MC is a computer on a chip. Ex, 8051, ARM M0, M1 etc

COMPONENTS of MC:-

- Central processing unit(CPU)
- Random Access Memory)(RAM)-store data temporarily for operation
- Read Only Memory(ROM)-store program of microcontroller
- Input/output ports
- Timers and Counters-measurement of intervals
- Interrupt Controls-providing delay for working programs and allow another program to work which is more important
- Analog to digital converters
- Digital to analog converters
- Bus-collection of wires for transfer of data(16 bit address bus and 8 bit data bus)
- Oscillator-provide clock pulses for operation

BLOCK DIAGRAM of MC

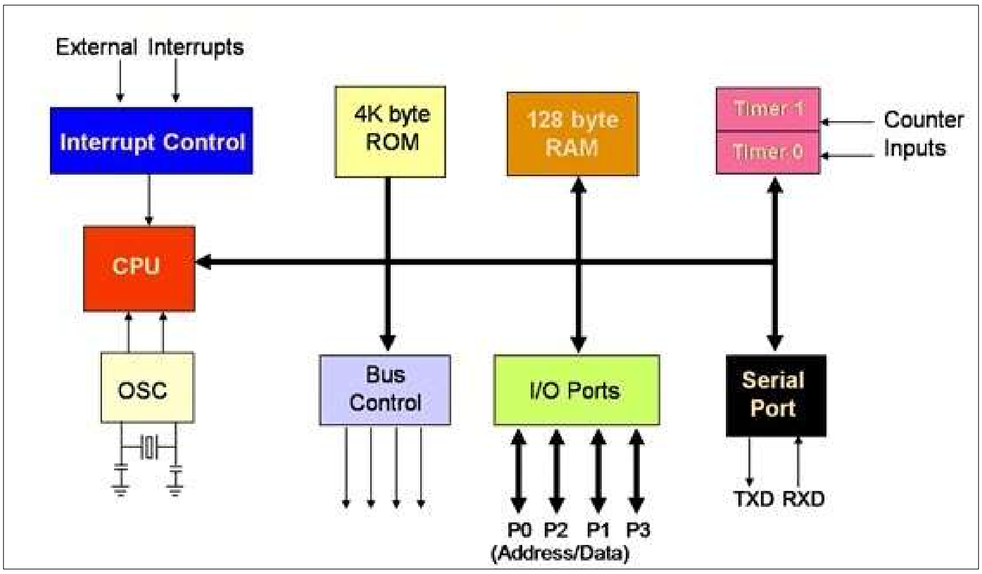


Fig1. 8051 MC block diagramSource:www.tutorialspoint.com

Programming the Embedded Systems: An Integrated Development Environment (IDE) [also known as tool chain] is required to program the MC and develop the IoT and embedded system. An IDE is an integration of several tools such as compilers, assemblers, linkers, editor, simulator etc as a single software. Here , Keil IDE is used to develop the applications.

Compiler converts source code written in high level language to a assembly language which can be understood by a computer.

Assembler converts code in assembly language to machine language which can be understood by a computer.

Cross compiler converts source code written in high level language into assembly language for a platform other than on which it is running i.e. when the CPU is different

Cross assembler converts assembly program into object code for a platform other than one on which it is running

Host computer is connected to other computers and terminals and provides data and other computer services to them. The program is compiled or assembled on this computer. Eg. PC, Laptop, Server etc

Target computer is the one on which program is loaded and run. Eg- Washing MC, traffic light control system

High level language is close to human language and allow to write programs which is independent of the computer used. It is different from middle level language in the way that it consists of classes as well. Ex-C++

Assembly language is a symbolic representation of machine code and depends on computer architecture. It consists of mnemonics.

Machine language consists of binary and hexadecimal instructions which the computer can understand directly.

A **simple software development tool chain** consists of a compiler and linker, libraries and debugger etc.

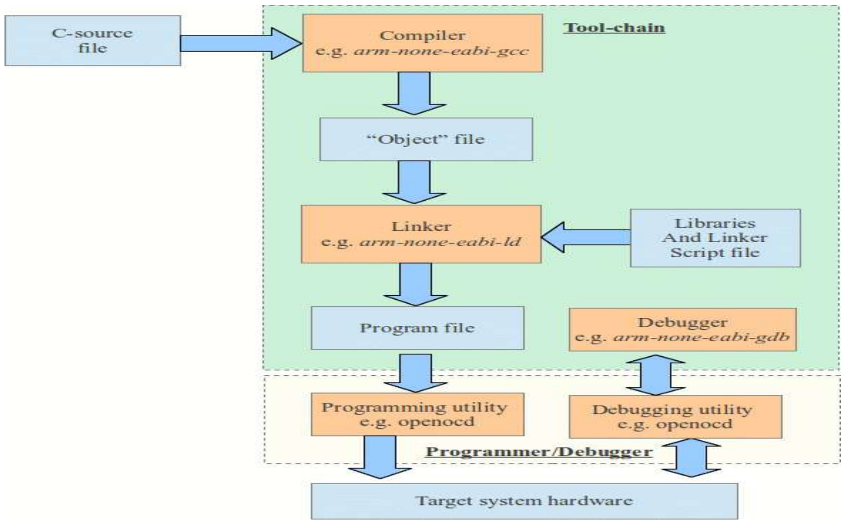


Fig2. Tool- Chain for IoT/ ESD Source: ioprog.com

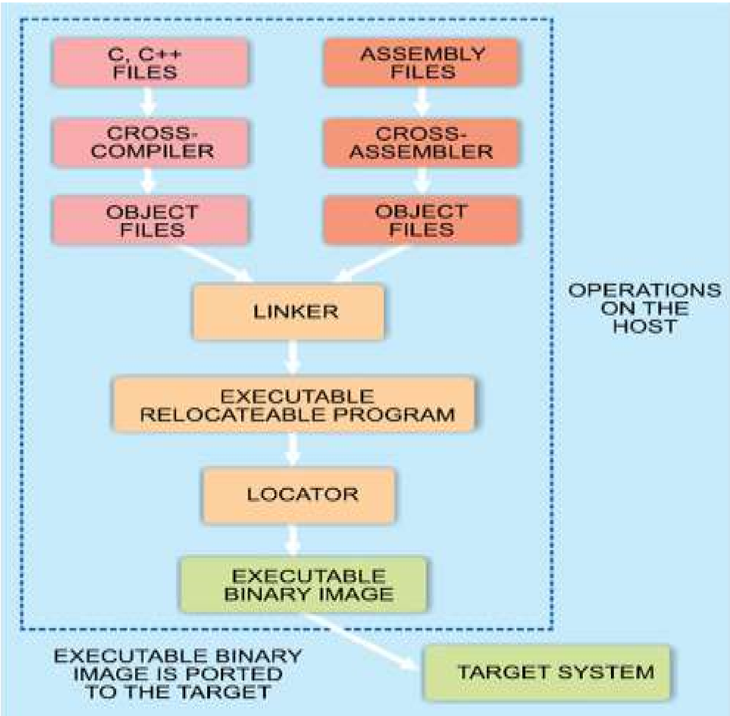
An **embedded system development tool chain** consists of an editor, compiler, linker, debugger

Editor edits the source code for embedded systems and speeds up the input for source code

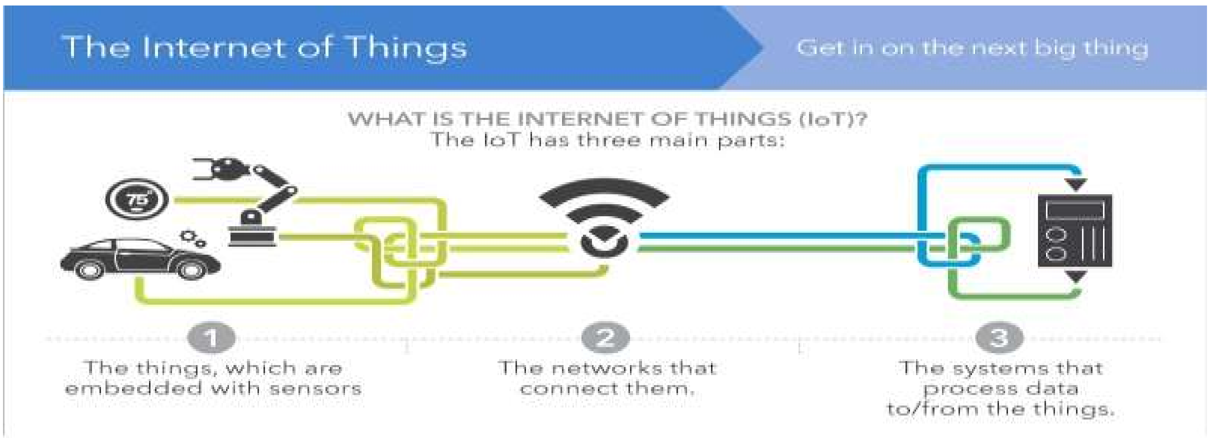
Linker links one or more object files into a single executable file

Debugger tests the program

Source: electronicsforu.com



Source: internet-things-iot-dummies-rajat-kochhar



CHAPTER-2: BASICS OF EMBEDDED C

EXPERIMENT 2.1: To blink an LED
REQUIREMENTS: 1. HARWARE: microcontroller 8051 and LEDs on a Embedded Development Board, keyboard, mouse 2. SOFTWARE: Keil uvision5, nuvoton utility
INTRODUCTION: The LEDs are connected to pin 2 and pin 3 of port 3 of the microcontroller(P3.2 and P3.3). Input 0 and 1 are given to these pins and status of LED is observed.
CODE:

```
//to blink a LED
#include<reg51.h>      //including this library to use the various registers
sbit LED1=P3^2;                    //variable LED1 has the value of pin 2 of port 3
sbit LED2=P3^3;     //variable LED2 has the value of pin 3 of port 3
void delay(unsigned int i);       //prototype of delay function

void main(void)
{
         while(1)                    //for infinite loop
         {

         LED1=0; LED2=1;                  //LED1 is ON and LED2 is OFF
         delay(75);                       //calling the delay function with 75 input
         LED1=1; LED2=0;      //LED1 is OFF and LED2 is ON
         delay(75);
         }
}
//definition of delay function
void delay(unsigned int i)
         {
                  unsigned int k;
                  unsigned int j;
                  for(j=0;j<i;j++)       //loop goes from 0 to input(75)
                  for(k=0;k<750;k++);     //loop goes from 0 to 750
         }
```

OBSERVATIONS:			
0 and 1 are given as input to LEDs and output noted.			
INPUT (LED1)	**INPUT (LED2)**	**STATUS (LED1 AT P3.2)**	**STATUS (LED2 AT P3.3)**
0	1	ON	OFF
1	0	OFF	ON

EXPERIMENT 2.2: To explore data types in embedded c (or c) using LED interface.

REQUIREMENTS:

1. HARWARE: microcontroller 8051 and LEDs on a Embedded Development Board, keyboard, mouse
2. SOFTWARE: Keil uvision5, nuvoton utility

INTRODUCTION:

The LEDs are connected to pin 2 and pin 3 of port 3 of the microcontroller(P3.2 and P3.3). Different inputs are given using various data types and the outputs are noted by observing the status of LEDs. The LED is on when data in its pin is 0 and if off when it is 1.

	
BOTH LEDs OFF	BOTH LEDs ON
	
BOTH LEDs ON	LED1 ON

CODE:

```
//to explore data types in embedded c (or c)  using LED interface
#include<reg51.h>      //including this library to use the various registers
int b,f;         //integer data type
float a;                    //float data type
double d;               // double data type
char c;                 //character data type
sfr j=0x80;          //special function register data type
sbit LED1=P3^2;   //single bit data type
sbit LED2=P3^3;
void delay(unsigned int i);      //prototype of function with definition later int the program

void main(void)
{
                    // sbit data type
            LED1=0;       LED2=1;                    //LED1 is ON and LED2 is OFF
            delay(150);                         //calling the delay function with 150  input
            LED1=1;       LED2=0;                    //LED1 is OFF and LED2 is ON
            delay(150);                         //calling the delay function with 150 input
                    //integer data type
            for(b=0;b<=7;b++)                //for loop to give values to variable b from 0-7
            {
```

```
                P3=61;                  //both the LEDs on port 3 are off
                delay(75);              //calling the delay function with 75 input
                P3=b;                   //binary equivalent of decimal nos. stored in port
                delay(150);             //calling the delay function with 150 input
        }
        //character data type-CAPITAL LETTERS
               for(c='A';c<='Z';c++)    //for loop to give values to variable c from 'A' to 'Z'
        {
                P3=61;
                delay(75);
                P3=c;                   //binary equivalent of ASCII code stored in port 3
                delay(150);
        }
        //character data type-SMALL LETTERS
        for(c='a';c<='z';c++)           //for loop to give values to variable c from 'a' to 'z'
        {
                P3=61;
                delay(75);
                P3=c;                   //binary equivalent of ASCII code stored in to port 3
                delay(150);
        }
        //float data type
      for(a=0.6;a<=7.6;a++)             //for loop to give values to variable a from 0.6 to 7.6
        {
                P3=61; delay(75);
                P3=a;                           //binary code stored in port 3
                delay(150);
        }
        //double data type
        for(d=0.2;d<=7.2;d++)
        {
                P3=61;  delay(75);      P3=d;                   //binary code stored in port 3
                delay(150);
        }
        //sfr data type
        for(f=0;f<=61;f+=61)
        {
          P0=f;    P3=j;    delay(150);
        }
}
//delay function to cause some delay to observe LED glowing
void delay(unsigned int i)
{
        unsigned int k;
        unsigned int;
        for(j=0;j<i;j++)                //for loop till the number passed to function
        for(k=0;k<7500;k++);            //for loop to cause delay of 7500
}
```

OBSERVATIONS:

1. SBIT data type

From line 14-18 in the code, 0 and 1 are given as input to LEDs and output noted.

INPUT (LED1)	INPUT (LED2)	STATUS (LED1 AT P3.2)	STATUS (LED2 AT P3.3)
0	1	ON	OFF
1	0	OFF	ON

2. INT data type

From line 19-26 in the code, numbers 0-7 are given as input through a for loop and output noted. The output is as per the binary code stored corresponding to the number. Some examples are:

DECIMAL NUMBER (INPUT)	BINARY NUMBER (STORED)	STATUS (LED1 AT P3.2)	STATUS (LED2 AT P3.3)
0	0000	ON	ON
1	0001	ON	ON
6	0110	OFF	ON
7	0111	OFF	ON

3. CHAR data type

TYPE 1:CAPITAL LETTERS

From line 27-34 in the code, the alphabets in capitals are given as input through a loop and output noted. The output is according to the binary equivalent of the ASCII code of the letter. Some examples are:

CHARACTER	ASCII CODE	BINARY CODE	STATUS (LED1 AT P3.2)	STATUS (LED2 AT P3.3)
A	65	1000001	ON	ON
D	68	1000100	OFF	ON
H	72	1001000	ON	OFF
L	76	1001100	OFF	OFF
P	80	1010000	ON	ON
T	84	1010100	OFF	ON
Z	90	1011010	ON	OFF

TYPE 2: SMALL LETTERS

From line 35-42, the alphabets in small are given as input through a loop and output noted. The output is according to the binary equivalent of the ASCII code of the letter. Some examples are:

CHARACTER	ASCII CODE	BINARY CODE	STATUS (LED1 AT P3.2)	STATUS (LED2 AT P3.3)
b	98	01100010	ON	ON
g	103	01100111	OFF	ON
k	107	01101011	ON	OFF
o	111	01101111	OFF	OFF
s	115	01110011	ON	ON
w	119	01110111	OFF	ON
z	122	01111010	ON	OFF

4. FLOAT data type

From line 43-50, numbers with a decimal point were given as input. But the output was according to the binary equivalent of the whole number part of the number only. Some examples are:

NUMBER (INPUT)	WHOLE NUMBER OF INPUT	BINARY NUMBER (STORED)	STATUS (LED1 AT P3.2)	STATUS (LED2 AT P3.3)
2.6	2	0010	ON	ON
3.6	3	0011	ON	ON
4.6	4	0100	OFF	ON

5.6	5	0101	OFF	ON

5. DOUBLE data type

From line 51-58, numbers with a decimal point were given as input. But the output was according to the binary equivalent of the whole number part of the number only.

NUMBER (INPUT)	WHOLE NUMBER OF INPUT	BINARY NUMBER (STORED)	STATUS (LED1 AT P3.2)	STATUS (LED2 AT P3.3)
1.2	1	0001	ON	ON
2.2	2	0010	ON	ON
5.2	5	0101	OFF	ON
6.2	6	0110	OFF	ON

6.SPECIAL FUNCTION REGISTER (SFR) data type

From line 59-65, variable j of sfr data type was given direct address of port 0. Port 3 consisting of the LEDs was pointed to this variable(P3=j). Then port 0 was given values 0 and 1 and the corresponding changes were observed in LED.

INPUT	STATUS (LED1 AT P3.2)	STATUS (LED2 AT P3.3)
0	ON	ON
61	OFF	OFF

EXERCISE 2.2:

Fill the table below:

INPUT IN P3	BINARY CODE	STATUS OF LED 1 AT P3.2	STATUS OF LED2 AT P3.3
3			
5			
8			
9			
Q			
S			
G			
V			
I			
r			
x			
t			
6.9			
3.5			
1.0			
2.5			
59	00111011		
61	00111101		
35	00100011		
47	00101111		

EXPERIMENT 2.3: To demonstrate status of LEDs using expressions in embedded c (or c)

REQUIREMENTS:

1. HARWARE: microcontroller 8051 and LEDs on a Embedded Development Board, keyboard, mouse
2. SOFTWARE: Keil uvision5, nuvoton utility

INTRODUCTION:

The LEDs are connected to pin 2 and pin 3 of port 3 of the microcontroller(P3.2 and P3.3). Arithmetic(+, -,* ,/ ,% ,++, --), logical (AND(&&), OR(||), NOT(!)), boolean(XOR(^), AND(&), OR(|)) and relational (>,>=,<=,<,==,!=) operators are used to form expressions and observe the status of the LEDs.

CODE:

```
//to observe status of LEDs using expressions
#include<reg51.h>  //include this library to use various registers
sbit LED1=P3^2;    //LED1 variable holds data of pin 2 of port 3
sbit LED2=P3^3;   //LED2 variable holds data of pin 3 of port 3
int a,b;  //integer data type variables
void delay(unsigned int i);  //prototype of delay function
void main()  //main function starts here
{
        for(a=0;a<=1;a++)   //for loop gives value 0 and 1 to 'a' variable
        {
                for(b=0;b<=1;b++)  //for loop gives value 0 and 1 to 'b' variable
                {
                        //LOGICAL AND OPERATOR
                        LED1=(a && b);                          //LED1 assigned value of result a&&b
                        //following code so as to observe variations in LED1 clearly
                        LED2=0;      //LED2 ON
                        delay(150);  //delay function called with input 150
                        LED2=61;     //LED2 OFF
                        delay(75);  //delay function called with input 75
                        //LOGICAL OR OPERATOR
                        LED2=a||b;  //LED2 assigned value of result a||b
                        LED1=0;
                        delay(150);
                        LED1=61;
                        delay(75);
                        //LOGICAL NOT OPERATOR
                        LED1=!(a&&b);
                        LED2=0;
                        delay(150);
                        LED2=61;
                        delay(75);
                        //BOOLEAN AND OPERATOR
                        LED1=a&b;    //LED1 assigned value of result a&b as per truth table
                        //following code so as to observe variations in LED1 clearly
                        LED2=0;      //LED2 ON
                        delay(150);  //delay function called with input 150
                        LED2=61;     //LED2 OFF
                        delay(75);  //delay function called with input 75
                        //BOOLEAN OR OPERATOR
                        LED2=a
                        |b;  //LED2 assigned value of result a|b as per truth table
                        LED1=0;
                        delay(150);
                        LED1=61;
                        delay(75);
                        //BOOLEAN XOR OPERATOR
                        LED1=a^b;  //LED1 assigned value of result a^b as per truth table
                        LED2=0;
                        delay(150);
                        LED2=61;
                        delay(75);
```

```
                //ARITHMETIC '-' OPERATOR
                LED1=a-b;  //all other values except 0 are taken as OFF
                LED2=0;
                delay(150);
                LED2=61;
                delay(75);
                //ARITHMETIC '/' OPERATOR
                LED2=a/b;  //all other values except 0 are taken as OFF
                LED1=0;
                delay(150);
                LED1=61;
                delay(75);
                //RELATIONAL > OPERATOR
                LED1=0;
                if(a>b)
                    LED2=b;
                else
                    LED2=a;
                delay(150);
                LED1=61;
                delay(75);
                //RELATIONAL >= OPERATOR
                LED2=0;
                if(a>=b)
                    LED1=a;
                else
                    LED1=b;
                delay(150);
                LED2=61;
                delay(75);
            }  //for loop of 'b' variable ends
        }  //for loop of 'a' variable ends
    }  //main function ends
    //definition of delay function
    void delay(unsigned int i)
    {
        unsigned int j,k;
        for(j=0;j<=i;j++)    // for loop goes from 0 to number passed to the function
        for(k=0;k<=7500;k++);  //for loop goes from 0 to 7500
    }
```

OBSERVATIONS:

1. LOGICAL AND OPERATOR - true if both operands are non zero. It passes value 1 to the variable if true and 0 if false.

VALUE OF VARIABLE 'a'	VALUE OF VARIABLE 'b'	a && b	VALUE IN LED1 (a && b)	STATUS OF LED1
0	0	FALSE	0	ON
0	1	FALSE	0	ON
1	0	FALSE	0	ON
1	1	TRUE	1	OFF

2. LOGICAL OR OPERATOR - true if one of the operands is non zero

VALUE OF VARIABLE 'a'	VALUE OF VARIABLE 'b'	a || b	VALUE IN LED2 (a | |b)	STATUS OF LED2
0	0	FALSE	0	ON
0	1	TRUE	1	OFF
1	0	TRUE	1	OFF
1	1	TRUE	1	OFF

3.LOGICAL NOT OPERATOR

VALUE OF VARIABLE 'a'	VALUE OF VARIABLE 'b'	!(a && b)	VALUE IN LED1 !(a && b)	STATUS OF LED1
0	0	TRUE	1	OFF
0	1	TRUE	1	OFF
1	0	TRUE	1	OFF
1	1	FALSE	0	ON

4.BOOLEAN AND OPERATOR

VALUE OF VARIABLE 'a'	VALUE OF VARIABLE 'b'	VALUE IN LED1 (a & b)	STATUS OF LED1
0	0	0	ON
0	1	0	ON
1	0	0	ON
1	1	1	OFF

5.BOOLEAN OR OPERATOR

VALUE OF VARIABLE 'a'	VALUE OF VARIABLE 'b'	VALUE IN LED1 (a | b)	STATUS OF LED1
0	0	0	ON
0	1	1	OFF
1	0	1	OFF
1	1	1	OFF

6.BOOLEAN XOR OPERATOR

VALUE OF VARIABLE 'a'	VALUE OF VARIABLE 'b'	VALUE IN LED1 (a ^ b)	STATUS OF LED1
0	0	0	ON
0	1	1	OFF
1	0	1	OFF
1	1	0	ON

4. ARITHMETIC '-' OPERATOR

VALUE OF VARIABLE 'a'	VALUE OF VARIABLE 'b'	VALUE IN LED1 (a-b)	STATUS OF LED1
0	0	0	ON
0	1	-1	OFF
1	0	1	OFF
1	1	0	ON

5. ARITHMETIC '/' OPERATOR

VALUE OF VARIABLE 'a'	VALUE OF VARIABLE 'b'	VALUE IN LED2 (a/b)	STATUS OF LED2
0	0	0/0	OFF
0	1	0	ON

1	0	1/0	OFF
1	1	1	OFF

6. RELATIONAL '>' OPERATOR

VALUE OF VARIABLE 'a'	**VALUE OF VARIABLE 'b'**	**VALUE IN LED2 (b if a>b)**	**STATUS OF LED2**
0	0	0	ON
0	1	0	ON
1	0	0	ON
1	1	1	OFF

7.RELATIONAL '>=' OPERATOR

VALUE OF VARIABLE 'a'	**VALUE OF VARIABLE 'b'**	**VALUE IN LED1 (a if a>=b)**	**STATUS OF LED1**
0	0	0	ON
0	1	1	OFF
1	0	1	OFF
1	1	1	OFF

EXERCISE 2.3:

Write the status of the LED for the expression given:

VALUE OF VARIABLE 'a'	**VALUE OF VARIABLE 'b'**	**EXPRESSION**	**STATUS OF LED (P3.2)**
0	0	a&&b	
1	1	!a	
0	1	a||b	
0	0	!b	
1	1	a-b	
1	0	a*b	
0	0	a+b	
1	1	a/b	
0	1	if(a<=b) LED=b;	
1	0	if(a>b) LED=a;	
1	0	if(a<b) LED=a;	
1	1	if(a>=b) LED=b;	
0	0	if(a==b) LED=a;	
0	1	if(a!=b) LED=b;	
0	1	if(a!=b) LED=a;	

EXPERIMENT 2.4: To explore control structures in embedded c (or c)

REQUIREMENTS:
1. HARWARE: microcontroller 8051 and LEDs on a Embedded Development Board, keyboard, mouse
2. SOFTWARE: Keil uvision5, nuvoton utility

INTRODUCTION:

The LEDs are connected to pin 2 and pin 3 of port 3 of the microcontroller(P3.2 and P3.3). Control structures-for loop, while loop, switch case and If-Else ladder are used to assign the values to port 3 where LED is connected and status of LEDs is observed.

CODE:

```
//to explore control structures in embedded c (or c)
#include<reg51.h>  //library included so as to use registers defined in it
int a,b; //integer data type
void delay( unsigned int i);  //prototype of delay function
void main()  //main function starts here
{
  //for loop
        for(a=0;a<=5;a++)
        {
                P3=61;  //both LEDs on port 3 are OFF
                delay(75);  //calling the delay function with input 75
          P3=a;  //assigning number to the port
                delay(150);  //calling the delay function with input 150
        }
        //while loop
        a=0;
        while(a<=5)     //till a is less than or equal to 5
        {
                P3=61;
                delay(75);
                P3=a;
                delay(150);
                a++;  //incrementing value of a by 1
        }
        //switch
        for(b=0;b<=3;b++)   //for loop for different cases
        {
          switch(b)
          {
                case 0:
                  {
                        P3=61;
                          delay(75);
                        P3=b;
                        delay(150);
                  }
                  case 1:
                  {
                          P3=61;
                          delay(75);
                          P3=b;
                          delay(150);
                  }
                  case 2:
                  {
                          P3=61;
                          delay(75);
                          P3=b;
                          delay(150);
```

```
			}
			case 3:
			{
				P3=61;
				delay(75);
				P3=b;
				delay(150);
			}
			default: P3=0;
		}
	}
//IF-ELSE
 for(a=0;a<=1;a++)
 {
	if(a==0)
	{
		P3=61;
		delay(75);
		 P3=a;
		 delay(150);
	}
	else
	{
		P3=61;
		delay(75);
		 P3=0;
		delay(150);
	}
 }
}
//definition of delay function
void delay(unsigned int i)
{
	unsigned int j;
	unsigned int k;
	for(j=0;j<=i;j++)   //for loop goes from 0 to number passed to function
	for(k=0;k<=7500;k++);  //for loop goes from 0 to 7500
}
```

OBSERVATIONS:

1. for loop

From line 8 to 14,the loop goes from 0 to 5 and value is given to port 3. The status of LEDs is then noted.

VALUE OF VARIABLE 'a'	BINARY CODE	STATUS OF LED1 (AT P3.2)	STATUS OF LED2 (AT P3.3)
0	0000	ON	ON
1	0001	ON	ON
2	0010	ON	ON
3	0011	ON	ON
4	0100	OFF	ON
5	0101	OFF	ON

2.while loop

from line 16 to 24,the loop starts from 0 and incremented each time till it reaches 5. The number is passed to the port and status of LEDs noted.

VALUE OF VARIABLE 'a'	BINARY CODE	STATUS OF LED1 (AT P3.2)	STATUS OF LED2 (AT P3.3)
0	0000	ON	ON
1	0001	ON	ON
2	0010	ON	ON
3	0011	ON	ON
4	0100	OFF	ON
5	0101	OFF	ON

3. switch case

From line 26 to 60, the loop goes from 0 to 3 and switch case is tested for each iteration. A value is passed to the port corresponding to each case and status of LEDs is noted.

VALUE OF VARIABLE 'a'	BINARY CODE	CASE CALLED	STATUS OF LED1 (AT P3.2)	STATUS OF LED2 (AT P3.3)
0	0000	0	ON	ON
1	0001	1	ON	ON
2	0010	2	ON	ON
3	0011	3	ON	ON

4. If-Else ladder

From line 62 to 77, condition is checked in if statement and if true the statements in 'if' block are executed otherwise statements in 'else' block are executed.

VALUE OF VARIABLE 'a'	VALUE PASSED TO P3	STATUS OF LED1 (AT P3.2)	STATUS OF LED2 (AT P3.3)
0	a i.e.0	ON	ON
1	0	ON	ON

EXERCISE 2.4:

1. What is the status of LED after each iteration? (LED is at P3.2)

```
 int b=20,a=50;
while(a!=0)
 {
  if (a>=b)
     LED =0;
     else
     LED=1;
     a-=5;
  }
```

2. What is the status of LEDs ?

```
for(b=0;b<=10;b++)
        {
          switch(b)
          {
```

```
                case 0:
                 {
                         P3=61;
                           delay(75);
                         P3=b;
                         delay(150);
                 }
                 case 2:
                 {
                          P3=61;
                          delay(75);
                          P3=b;
                          delay(150);
                 }
                 case 4:
                 {
                          P3=61;
                          delay(75);
                          P3=b;
                          delay(150);
                 }
                 case 6:
                 {
                          P3=61;
                          delay(75);
                          P3=b;
                          delay(150);
                 }
                 default: P3=0;
     }
  }
```

EXPERIMENT 2.5: To explore functions in embedded c (or c)

REQUIREMENTS:

1. HARWARE: microcontroller 8051 and LEDs on a Embedded Development Board, keyboard, mouse
2. SOFTWARE: Keil uvision5, nuvoton utility

INTRODUCTION:

The LEDs are connected to pin 2 and pin 3 of port 3 of the microcontroller(P3.2 and P3.3). Different operations such as add, subtract, multiply, divide and recursion are performed using functions. The output is observed by observing the status of LEDs.

CODE:

```
//to explore functions in embedded c
#include<reg51.h>    //include library to access registers defined in it
int a=1,b=3,c,d; //global integer variable
//function prototypes
void delay(unsigned int i);  //prototype of delay function
int add(); //no parameters passed but returns an integer value
int subtract(int, int);   //integer data types passed to the function and returns an integer value
void multiply(); //no parameters passed and returns no value
void divide(); //no parameters passed and returns no value
int recursive(int);  //an integer parameter passed and returns an integer value
void main()  //main function starts
{
	c=add();     //calling function add() and giving the value to variable c
	P3=0;  //LEds ON
	delay(150); //delay function called with input 150
	P3=c;  //return value of add() function given to port
	delay(150);
	d=subtract(c,a);    //calling function subtract() by passing parameters and assigning return value to variable d
	P3=61;  //LEDs OFF
	delay(150);
	P3=d;  //return value of subtract() function given to port
	delay(150);
	multiply();  //multiply() function called
	divide();  //divide() function called
	c=recursive(a);   //recursive() function called by passing a parameter and return value assigned to variable c
	P3=61;
	delay(150);
	P3=c;  //return value of recursive() function given to port
	delay(75);
}
int add()   //definition of add() function
{
	int c;   //local variable to this function
	c=a+b;
	return c;
}
int subtract(int a,int b)  //definition of subtract() function
{
	int c;
	c=a-b;
	return c;
}
void multiply()  //definition of multiply() function
{
	int a=1,b=1,c;
	c=a*b;
	P3=61;
	delay(150);
	P3=c;
	delay(150);
}\
void divide()  //definition of divide() function
```

```
{
        int a=1,b=1,c;
        c=a/b;
        P3=61;
        delay(150);
        P3=c;
        delay(150);
}
int recursive(int a)  //definition of recursive() function
{
        int c;
        while(a>=0)   //loop works till value of a is greater or equal to 0
        {
          c = a*1;
          recursive(--a); //calling the recursive function from its body by decremented value of a
        }
        return c;  //returning the value to the calling parameter
}
void delay(unsigned int i)
{
        unsigned int j;
        unsigned int k;
        for(j=0;j<=i;j++)
        for(k=0;k<=7500;k++);
}
```

OBSERVATIONS:

FUNCTION	VALUE GIVEN TO PORT 3	BINARY CODE	STATUS OF LED1 (AT P3.2)	STATUS OF LED2 (AT P3.3)
ADD()	4	0100	OFF	ON
SUBTRACT()	3	0011	ON	ON
MULTIPLY()	1	0001	ON	ON
DIVIDE()	1	0001	ON	ON
RECURSIVE()	0	0000	ON	ON

EXERCISE 2.5:
What will be the status of LEDs?

```
void main()
{
        int a=5,b=1,c;
        c=add(a,b);
        subtract(c,b);
        multiply(c,b);
        divide(a,b);
        c=recursive(a);
        P3=c;
}
int add(int a , int b)
{
        int c;
        c=a+b;
        P3=c;
        return c;
}
```

```
void subtract(int a , int b)
{
        int c;
        c=a-b;
        P3=c;
}
void multiply(int a, int b)
{
        int c;
        c=a*b;
        P3=c;
}
void divide(int a, int b)
{
        int c;
        c=a/b;
        P3=c;
}
int recursive(int a)
{
        int c;
        while(a>0)
        {
          c=a*a;
          recursive(a--);
        }
        return c;
}
```

EXPERIMENT 2.6: To demonstrate status of LEDs using arrays in embedded c (or c)

REQUIREMENTS:

1. HARWARE: microcontroller 8051 and LEDs on a Embedded Development Board, keyboard, mouse
2. SOFTWARE: Keil uvision5, nuvoton utility

INTRODUCTION:

The LEDs are connected to pin 2 and pin 3 of port 3 of the microcontroller(P3.2 and P3.3). Various values are stored in single dimensional and multi dimensional arrays. Then values of array is given to port 3.

CODE:

```
//to demonstrate arrays by observing status of LEDs
#include<reg51.h>  //include library to use registers defined in it
int a[5], b[2][3]; //a[] is a single dimensional array and b[][] is a multi dimensional array
int i,j,k;
void delay(unsigned int i);  //prototype of delay function
void main()  //main() starts
{
for(i=0;i<5;i++)
a[i]=i;          //assigning values to array
for(j=0;j<2;j++)
{
        for(k=0;k<3;k++)
        {
                b[j][k]=j*k;  //assigning values to array
        }
}
for(i=0;i<5;i++)
{
        P3=61;  //both LEDs OFF
        delay(75);  //calling delay() function with input 75
```

```
    P3=a[i];  //assigning values to port 3
        delay(150);  //calling delay() function with input 150
    }
    for(j=0;j<2;j++)
    {
        for(k=0;k<3;k++)
        {
            P3=61;
            delay(75);
            P3=b[j][k];  //assigning values to port 3
            delay(150);
        }
    }
    }
    //definition of delay function
    void delay(unsigned int i)
    {
        unsigned int j,k;
        for(j=0;j<=i;j++)
        for(k=0;k<=7500;k++);
    }
```

OBSERVATIONS:

1. single dimensional array (a[ ])

VALUE GIVEN TO PORT 3	BINARY CODE	STATUS OF LED1 (AT P3.2)	STATUS OF LED2 (AT P3.3)
a[0]=0	0000	ON	ON
a[1]=1	0001	ON	ON
a[2]=2	0010	ON	ON
a[3]=3	0011	ON	ON
a[4]=4	0100	OFF	ON

2.multi dimensional array (b[ ][ ])

VALUE GIVEN TO PORT 3	BINARY CODE	STATUS OF LED1 (AT P3.2)	STATUS OF LED2 (AT P3.3)
b[0][0]=0	0000	ON	ON
b[0][1]=0	0000	ON	ON
b[0][2]=0	0000	ON	ON
b[1][0]=0	0000	ON	ON
b[1][1]=1	0001	ON	ON
b[1][2]=2	0010	ON	ON

EXERCISE 2.6:

What is the status of LEDs?

```
void main()
{
for(i=0;i<2;i++)
a[i]=i*i;
for(j=0;j<3;j++)
  for(k=0;k<3;k++)
          b[j][k]=j*k;
for(i=0;i<2;i++)
  P3=a[i];
for(j=0;j<3;j++)
    for(k=0;k<3;k++)
            P3=b[j][k];
}
```

EXPERIMENT 2.7: To explore pointers in embedded c (or c)

REQUIREMENTS:

1. HARWARE: microcontroller 8051 and LEDs on a Embedded Development Board, keyboard, mouse
2. SOFTWARE: Keil uvision5, nuvoton utility

INTRODUCTION:

The LEDs are connected to pin 2 and pin 3 of port 3 of the microcontroller(P3.2 and P3.3). A pointer variable is declared which points to the address of the integer variable. The pointer is dereferenced to give its value to port 3

.

CODE:

```
//to explore pointers in embedded c
#include<reg51.h>  //include library to use registers defined in it
void delay(unsigned int i);
void main()
```

```
{
    int i=1,j=6,k=72;  //integer variable declaration
    int *a;  //pointer variable declaration
    a=&i;  //pointer stores the address of variable i
    P3=61;  //both LEds OFF
    delay(75); //calling delay() function with input 75
    P3=*a;  //the port is given the value of pointer
    delay(150);  //calling delay() function with input 150
    a=&j; //pointer stores the address of variable j
    P3=61;
    delay(75);
    P3=*a;
    delay(150);
    a=&k;  //pointer stores the address of variable k
    P3=61;
    delay(75);
    P3=*a;
    delay(150);
}
void delay(unsigned int i)  //definition of delay() function
{
    unsigned int j,k;
    for(j=0;j<=i;j++)
    for(k=0;k<=7500;k++);
}
```

OBSERVATIONS:

VALUE GIVEN TO PORT 3	BINARY CODE	STATUS OF LED1 (AT P3.2)	STATUS OF LED2 (AT P3.3)
1	00000001	ON	ON
6	00000110	OFF	ON
72	01001000	ON	OFF

EXERCISE 2.7:
What will be status of LEDs?

```
void main()
{
        int i=4,j=8,k;
        int *a;
        for(k=0;k<4;k++)
        {
              if(i<5)
                 a=&i;
              else
               a=&j;
               ++i;
         }
        P3=*a;
}
```

APPENDIX:

1. Binary codes of decimal numbers

DECIMAL NUMBER (INPUT)	BINARY NUMBER (STORED)
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

2. Binary codes of alphabets

- CAPITAL LETTERS

CHARACTER	ASCII CODE	BINARY CODE
A	65	1000001
B	66	1000010
C	67	1000011
D	68	1000100
E	69	1000101
F	70	1000110

G	71	1000111
H	72	1001000
I	73	1001001
J	74	1001010
K	75	1001011
L	76	1001100
M	77	1001101
N	78	1001110
O	79	1001111
P	80	1010000
Q	81	1010001
R	82	1010010
S	83	1010011
T	84	1010100
U	85	1010101
V	86	1010110
W	87	1010111
X	88	1011000
Y	89	1011001
Z	90	1011010

- SMALL LETTERS:

CHARACTER	ASCII CODE	BINARY CODE
a	97	01100001
b	98	01100010
c	99	01100011
d	100	01100100
e	101	01100101
f	102	01100110
g	103	01100111
h	104	01101000
i	105	01101001
j	106	01101010
k	107	01101011
l	108	01101100
m	109	01101101
n	110	01101110
o	111	01101111
p	112	01110000
q	113	01110001
r	114	01110010
s	115	01110011
t	116	01110100
u	117	01110101
v	118	01110110
w	119	01110111
x	120	01111000
y	121	01111001
z	122	01111010

3. TRUTH TABLE-BOOLEAN OPERATORS

a	b	a&b (AND)	a|b (OR)	a^b (XOR)
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0