INDEX

SR. NO.	Experiment Description	Page No. / Experiment Date	Submission Date	Remarks / Signature
1.	Programming 8051 Microcontroller using ASM and C, and implementation in flash 8051 microcontroller.	1 to 3		
2.	Programming with Arithmetic Logic instructions.	4 to 5		
3.	Programming using constructs (sorting an array)	6 to 7		
4.	Programming using ports. (Assembly and C)	8 to 9		
5.	Delay generation using timer (Assembly & C)	10 to 11		

INDEX

SR. NO.	Experiment Description	Page No. / Experiment Date	Submission Date	Remarks
6.	Programming interrupts (Assembly and C)	12 to 13		
7.	Interfacing LCD display (Assembly & C)	14 to 15		
8.	Interfacing with keypad (Assembly & C)	16 to 20		
9.	Programming ADC/ DAC (Assembly & C)	21 to 22		
10.	Interfacing with stepper motor (Assembly & C)	23 to 24		

A Gayal Product

Date : Page : 01

Topic : EXPERIMENT - 1

AIM :- Programming 8051 Microcontroller using ASM and C, and implementation in Flash 8051 microcontroller.

THEORY:-

An assembly language program consists of, among other things, a series of lines of assembly language instructions. An assembly language instruct consists of a mnemonic, optionally followed by one or more operands.

A given assembly language program (See prog. 2-1) is a series of statements, or lines which are either assembly language instructions such as ADD and MOV, or statements called directives.

While instructions tell the CPU what to do, directives give directions to the assembler.

Eg. In the below program while the MOV and ADD instructs are commands to the CPU, ORG and END are directives to the assembler.
ORG tells the assembler to place the opcode at memory location while END indicates to the assembler the end of source code..

```
ORG 0H   ; start (origin) at location 0
MOV R5,#25H ; Load 25H into R5
MOV R7,#34H ; Load 34H into R7
MOV A,#0 ;  Load 0 into A
ADD A, R5 ;  add contents of R5 to A
             ; Now A = A+R5.
ADD A, R7 ;  add contents of R7 to A
             ; Now A = A+R7.
ADD A,#12H ; add to A value 12H
             ; Now A = A+12H
HERE: SJMP HERE ; Stay in this Loop
END ;  End of .asm source file.
```

(Sample of a Assembly Language Program)

Compilers produce hex files that we download into the ROM of the microcontroller.
The size of the hex file produced by the compiler is one of the main concerns of microcontroller programmers for two reasons.

(i) Microcontroller have limited on chip ROM.
(ii) The code space for the 8051 is limited to 64 bytes.

C programming on the other hand, is less time consuming and much easier to write but the hex file size produced is much larger than if we used Assembly language. The following are some of major advantages —

(i) It is easier and less time consuming to write in C than assembly.
(ii) C is easier to modify and update.
(iii) You can use code available in function libraries.

```
// use Timer 0 to create a square wave on P1.0
# include <REG52.H>
  Sbit Portbit = P1^0;
Void main ()
 {
   TMOD = 1; // 16-bit mode on timer 0
   while (1) // Loop forever
 {
   TH0 = 0XFE;  // start timer at 2^16 -500
   TL0 = 0XFE;
   TR0 = 1; // start timer 0
   while (TF0 != 1) // wait for overflow
     ;
   TR0 = 0; // stop timer
   TF0 = 0; // clear timer overflow flag
   Port bit = ! Portbit; // Toggle Port bit
 }
}
```

A Gayal Product

Date: Page: 04

Topic: EXPERIMENT - 2

AIM:- Programming with Arithmetic logic Instructions.

16-Bit Addition

Program :-

```
MOV R0, #51H      // Initialize input 1 memory pointer
MOV R1, #61H      //* Initialize input 2 memory pointer
                     and store output also same. */
MOV R2, #02H      // Initialize interation count
CLR C
BACK: MOV A, @R0  /* Get lower bytes data in
                     first generation, upper bytes
                     data in record interation,
                     add them with carry and
                     store in memory pointer 2 */
ADDC A, @R1
MOV @R1, A
DEC R0            // Increment memory pointer.
DEC R1
DJNZ R2, BACK     /* Decrement interation count
                     and if it is not zero, go
                     to relative address and
                     repeat the same process */
```

```
JNC FINISH
MOV @RL, #01H
FINISH : SJMP $
END.
```

Memory Window:-

Before Execution-

D: 0x50H: FD 07 00 00 00 00
D: 0x60H: FF 5F 00 00 00 00

After Execution-

D: 0x50H: FD 07 00 00 00 00
D: 0x60H: 01 FC 06 00 00 00

A Gayal Product

Date : Page : 06

Topic : EXPERIMENT-3

AIM:- Program using constructs (sorting an array);

OBJECTIVE:-
To arrange N 8-Bit Numbers in Ascending Order

Program -

```
MOV R2, #05H        // initialize the iteration counter
DEC R2              // decrement the iteration count
BACK1 : MOV R0, #50H // Initialize memory pointer 1
MOV R1, #51H        // initialize memory pointer 2
MOV A, R2           // store outer loop count
MOV R3, A           // store inner loop count
BACK : MOV A, @R0   // Get the data from memory pointer 1
MOV B, @R1          // Get the data from pointer 2
CJNE A, B, LOOP     /* compare if not equal go to
                       relative address LOOP */
LOOP: JC LOOP1      /* If carry generated, go to
                       relative address (LOOP1) */
MOV @R0, B
LOOP1 : INC R0
INC R1
DJNZ R3, BACK
back
DJNZ R2, BACK1
back1
END
```

A Good Product

Date _____ Page 07

Topic EXPERIMENT-3

Memory Window:-

Before Execution

D:0X50H: 06 04 03 07 02 01

After Execution.

D:0X50H: 01 02 03 04 05 07

A Gayal Product

Date _____ Page 08

Topic EXPERIMENT - 04

ASM:- Programming using ports (Assembly and C)

* Write an ALP to transmit characters to a PC hyper terminal using the serial port and display on the serial window.

Label	Opcode and operands	Comments
	ORG 0000H	
	LJMP 8000H	
	ORG 8000H	
	MOV TMOD, #20H	; TMOD = 20H
	MOV TH1, #-3	; TH1 = -3H
	MOV SCON, #50H	; SCON = 50H
	SETB TR0	; TR0 = 1
AGAIN;	MOV A, #'H'	; Load letter 'H' in A
	ACALL TRANS	; call transmit program
	MOV A, #'T'	; Load letter 'T' in A
	ACALL TRANS	; call transmit program
AGAIN;	SJMP AGAIN	; short JUMP to again
TRANS;	MOV SBUF, A	; load SBUF with letter stored in A
HERE;	JNB TI, HERE	; if TI is not equal 1 JUMP here
	RET	; Return
	END	; END

A Gayal Product

Date :

Page : 09

Topic : Experiment - 04

* Write a C program for the 8051 to transfer the letter 'A' serially at 4800 baud continuously. Use 8-bit data and 1 stop bit.

```
#include <reg51.h>
void main (void)
{
TMOD = 0X20;  //use time 1, 8-Bit autoreload

TH1 = 0XFA;  // 4800 baud rate.
SCON = 0X50;
TR1 = 1;
While (1)
 {
  SBUF = 'A';  //place value in buffer.

  While (TI==0);
  TI =0
  }
}
```

A Gogal Product

Date _____ Page 10

Topic: Experiment - 5

AIM:- Delay generation using Timer (Assembly and C).

* Write an ALP to toggle the content of Port 0 continuously using timer delay in between.

	ORG 0000H	
	LJMP 8000H	
	ORG 8000H	
	MOV R0, #0AH	; R0 = 0AH
40;	MOV A, #55H	; A = 55H
	MOV P0, A	; Move the content of A in P0
	ACALL DELAY	; call delay program
	MOV A, #0AAH	; A = AAH
	MOV P0, A	; Move the content of A in P0
	ACALL DELAY	; call delay program
	DJNZ R0, 40	; R0 = R0 - 1.
	LCALL 0003H	; End of main program
DELAY;	MOV TMOD, #01H	; Load TMOD
START;	MOV TL0, #00H	; Load TL0
	MOV TH0, #00H	; Load TH0
	SETB TR0	; TR0 = 1
HERE;	JNB TF0, HERE	; If TF0 is not equal to 1 jump here
	CLR TR0	; TR0 = 0
	CLR TF0	; TF0 = 0
	RET	; Return to main program

A Gopal Product

Date Page 11

Topic Experiment - 5

X Write a C program to generate 1ms delay using 8051 timer 0

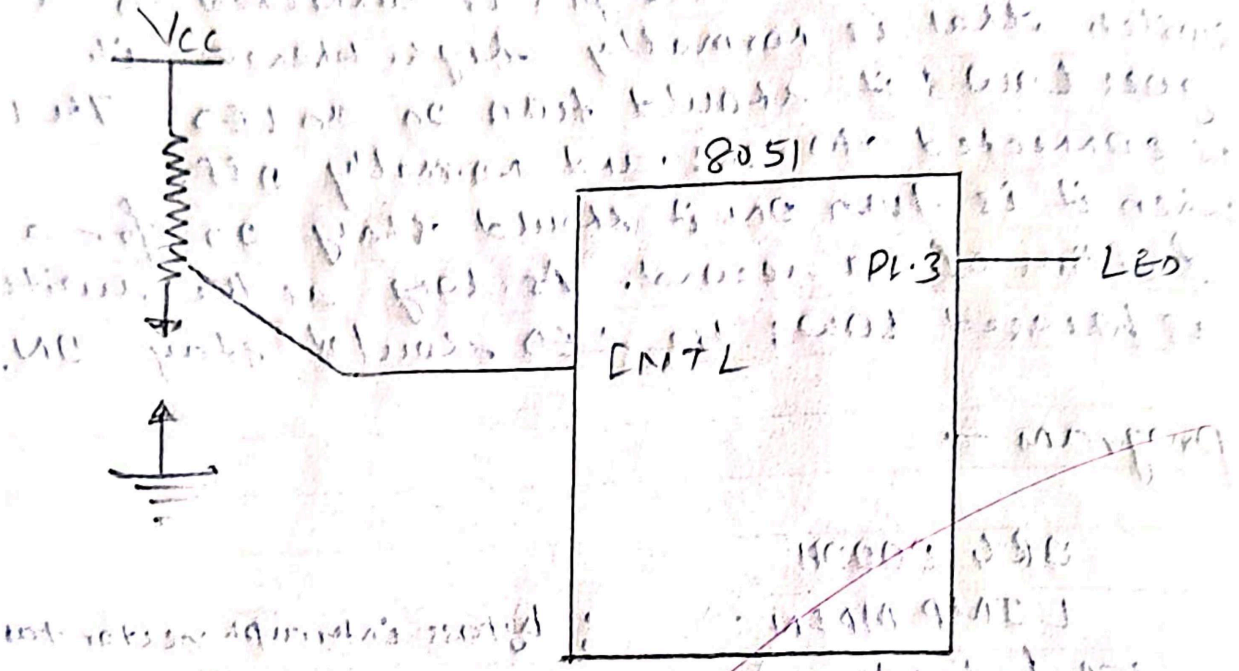
```
void delay()
{
  TMOD = 0x01; //timer 0 mode 1
  TH0 = 0xFC; //initial value for 1ms
  TL0 = 0x66;
  TR0 = 1;    //time start
  while(TF0 == 0); //check overflow condition

  TR0 = 0; //stop timer
  TF0 = 0; //clear flag.
}
```

AIM:- Programming Interrupts (Assembly and C).

- Assume that the INTL pin is connected to a switch that is normally high. Whenever it goes LOW, it should turn on an LED. The LED is connected to P1.3 and normally off. When it is turn ON it should stay on for a fraction of a second. As long as the switch is pressed LOW; the LED should stay ON.

Program -

```
        ORG 0000H
        LJMP MAIN.          ; bypass interrupt vector table
;--- ISR for hardware interrupt INTL to turn ---
        ORG 0013H           ; INTL ISR
        SETB P1.3           ; turn on LED
        MOV R3,#255         ; load counter
BACK:   DJNZ R3,BACK        ; keep LED on for a while
        CLR R3;BACK         ; turn off LED.
        RETL                ; return from ISR
;--- Main program for initialization ---
        ORG 30H
MAIN:   MOV IE,#10000100B   ; enable internal INTL
HERE:   SJMP HERE           ; stay here wit interrupted
        END
```

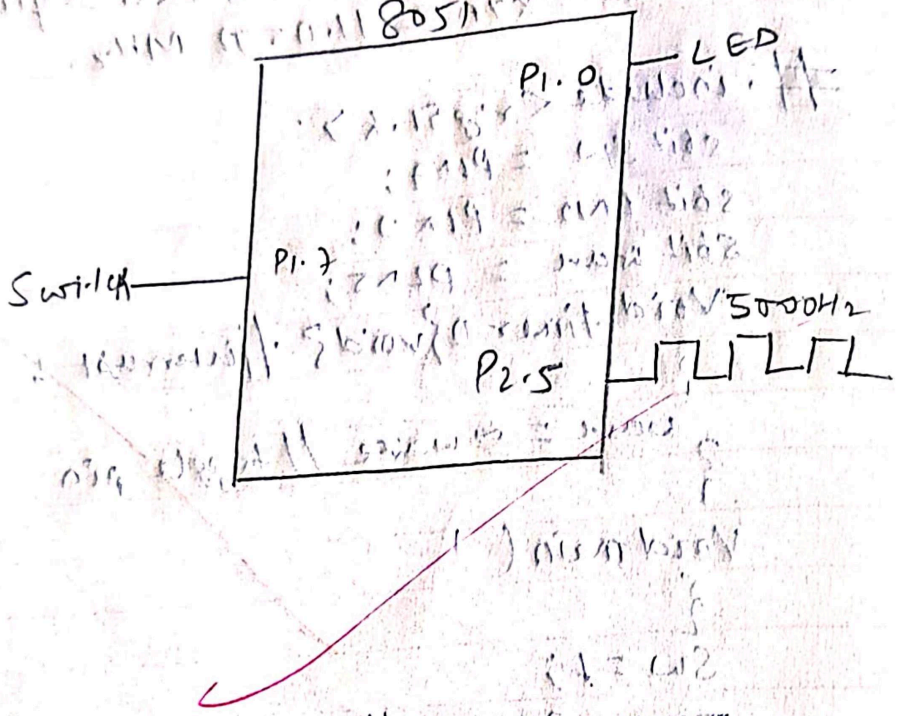

Vcc
8051
P1.3
LED
INT1

Date: A Great Product
Topic: Assignment - 6 Page: 13

* Write a C program that continuously gets a single bit of data from P1.7 and sends it to P1.0. While simultaneously creating a square wave of 200 μs period on pin P2.5 (use time 0 to 25). Create the square wave. Assume that XTAL = 11.0592 MHz.

```
#include <reg51.h>
sbit SW = P1^7;
sbit LED = P1^0;
sbit wave = P2^5;
void timer0() void ISR
{
wave = ~wave // toggle pin
}
void main()
{
SW = 1;
TMOD = 0x02; // make with timer
TH0 = 0xA4; // TH0 = -92
IE = 0x82; // make interrupt for timer0
while(1)
{
LED = SW; // send switch to LED.
}
}
```

200 μs / 2 = 100 μs
100 μs / 1.085 μs = 92


LED
P1.0
P1.7
Switch
P2.5
5000Hz

A Gogal Product

Date : Page : 14

Topic : Experiment - 7

AIM:- Interfacing LCD Display (Assembly & C).

ALP for LCD Interfacing.

```
        ORG  0H
        SJMP START
        ORG  30H
START : ACALL INT..LCD
        MOV A,#'H'
        ACALL . DATAWRT
        MOV A, #'E'
        ACALL. DATAWRT
        MOV A,#'L'
        ACALL . DATAWRT
        MOV A, #'L'
        ACALL DATAWRT
        MOVA , #'O'
        ACALL . DATAWRT
HERE: SJMP HERE     ; wait here
```

* C program for LCD interfacing to 8051

```
#include <reg51.h>
void main()
{
 intilled();
 lcd data('H');
 lcd data('E');
 lcd data('Y');
 lcd data('L');
 lcd data('O');
while(1);
}
```

A Gogal Product

Date : Page : 16

Topic : Subroutine - 8

AIM:- Interfacing with keypad (Assembly and C).

ALP for interfacing keypad

To check whether any key is pressed,

```
Start : mov a, #00h
        mov P1, a
        mov a, #0Fh
        mov P1, a

Press : mov 0, P2
        JZ press
```

After making sure that any key is pressed

```
        mov a, #0Ah
Next :  mov a, r4
        mov P1, a
        mov P2, a
        JNZ column
        mov a, r4
        r1, a
        mov r4, a
        mov a, r3
        add a, #08h
        mov r3, a
        SJMP Next
```

A Goyal Product

Date : Page : 17

Topic : Experiment - 8

After identifying the row to check the column following steps are followed.

```
calscan : mov r5, #00h
in : rrc a
     JC out
     inc r3
     Jmp in
out : mov a, r3
      da a
      mov P2, a
      Jmp start
```

* C program for interfacing keypad.

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

sbit rs = P3^2;
sbit rw = P3^3;
sbit e = P3^4;
sbit c4 = P1^0;
sbit c3 = P1^1;
sbit c2 = P1^2;
sbit c1 = P1^3;
```

```
sbit r4 = P1^4;
sbit r3 = P1^5;
sbit r2 = P1^6;
sbit r1 = P1^7;

void msdelay (unsigned int time)
  // function for creating delay in milliseconds.
  {
  unsigned i,j;
  for(i=0; i<time ; i++)
  for(j=0; j<1275; j++)
  {
  }

void lcd_cmd (unsigned char command)
  // function to send command instruction to LCD
  {
  display_port = command;
  rs=0
  rw=0;
  e=1;
  ms delay(1);
  e=0;
  }
```

```
void Lcd_data(unsigned char disp_data)
// function to send display data to LCD
{
  display_port = disp_data;
  rs = 1
  rw = 0;
  e = 1;
  ms delay(1);
  e = 0
}

void Lcd_init()
{
  LCD_cmd(0x38);
  ms delay(10);
  Lcd_cmd(0x0F);
  ms delay(10);
  Lcd_cmd(0x01);
  ms delay(10);
  Lcd_cmd(0x81);
  ms delay(10);
}
```

```
void row_finder4()
{
    R1=R2=R3=R4=1;
    C1=C2=C3=C4=0;
    if (R1==0);
    lcd_data('A');
    if (R2==0);
    lcd_data('B');
    if (R3==0);
    lcd_data('C');
    if (R4==0);
    lcd_data('D');
}

void main()
{
    lcd_init();
    while(1)
    {
    ms delay(30);
    C1=C2=C3=C4=1;
    R1=R2=R3=R4=0;
    if(C1==0)
    row_finder1();
    else if(C2==0);
    row_finder2();
    else if(C3==0);
}}
```

A Gayal Product

Date :

Page : 21

Topic : Experiment - 9

AIM:- Programming ADC/DAC [Assembly & C]

programming ADC 0804 in assembly

```
RD   BIT P1.5   ; RD
WR   BIT P1.6   ; WR (start conversion)
INTR BIT P1.7   ; end of conversion
Mydata EQU P1
  MOV P1, #0FFH

    SETB INTR
BACK: CLR  WR
    SETB WR

HERE: JB INTR; HERE
    CLR RD
    MOV A, Mydata
    ACALL CONVERSION

    ACALL DATA_display
    SETB RD
    SJMP BACK
```

programming ADC 0804 in C

The 8051 C version of the above program is given below-

```
#include <reg51.h>
    sbit RD = P2^5;
    sbit WR = P2^6;
    sbit INTR = P2^7;
    sfr Mydata = P1;
    void main();
     {
      unsigned char value;
      MYDATA = 0XFF;
      INTR = 1;
      WR = 1;
     while(1)
      {
       WR = 0;
       WR = 1;
      while(INTR = 1);
       RD = 0;
       value = MYDATA
      conversion display (value);
       RD = 1;
      }
     }
```

AIM:- Interfacing with stepper motor (Assembly & C)

Program in assembly

```
        MOV A, #66H
BACK:   MOV P1, A
        RR A
        ACALL DELAY
        SJMP BACK
        ...
DELAY
        MOV R2, #100
        MOV R3, #255
H1:     DJNZ R3, H2
H2:     DJNZ R2, H1
        RET
```

Date: A Gogal Product

Topic: Experiment 10 Page: 24

Program in C

```
#include <reg51.h>
void main ()
{
  while (1)
  {
    P1 = 0X60;
    Msdelay (100);
    P1 = 0XCC;
    ms delay (100);
    P1 = 0X99;
    msdelay (100);
    P1 = 0X33;
    msdelay (100);
  }
}
```