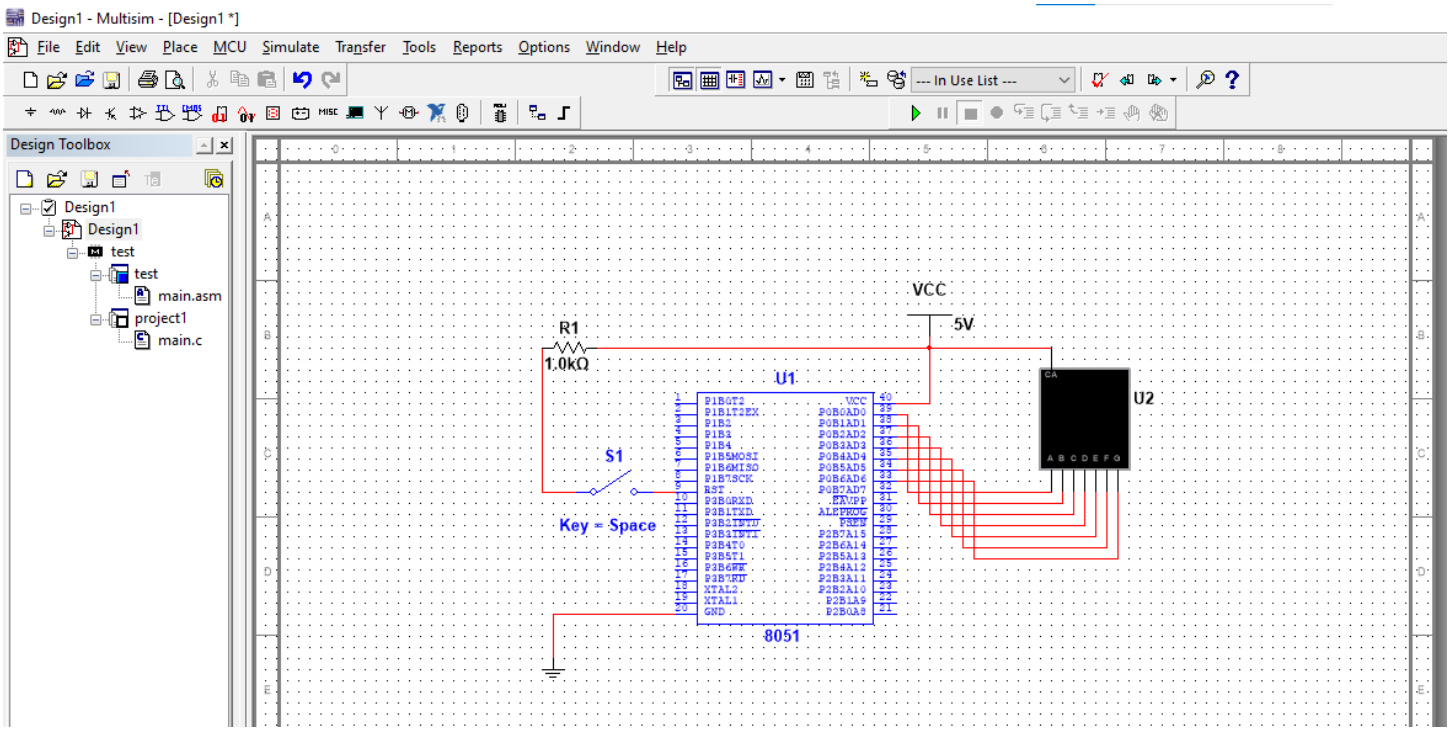
# Bahria University (Karachi Campus)

Department of Software Engineering

**Assignment # 2 – Fall 2023**

| | | | |
|---|---|---|---|
| Course Title: | **Embedded Systems** | Course Code: | **CEN-439** |
| Class: | **BSE-5(B)** | Shift: | **Morning** |
| Name: | **Muhammad Zain** | Reg# | **79321** |
| Course Instructor: | **Dr. Qamaruddin Memon** | Enroll# | **02-131212-077** |

## 1. Draw the Block diagram of seven segment display on Multisim/Proteus and write its code in C and Assembly




**C CODE**

```
#include <htc.h>
#define Port_0 P0
// Define common anode seven-segment display patterns

unsigned char displayPatterns[] = {
    0x40, // 0
    0x79, // 1
    0x36, // 2
    0x48, // 3
    0x19, // 4
    0x12, // 5
    0x02, // 6
    0x78, // 7
    0x00, // 8
    0x10  // 9
};
```

```
void delay(unsigned int milliseconds) {
    unsigned int i, j;
    for (i = 0; i < milliseconds; i++) {
        for (j = 0; j < 110; j++);
    }
}

void main() {
    unsigned char i;
    while (1) {
        for (i = 0; i < 10; i++) {
            Port_0 = displayPatterns[i]; // Display the current digit
            delay(10);            // Delay for a while to display the digit
        }
    }
}
```

## ASSEMBLY CODE

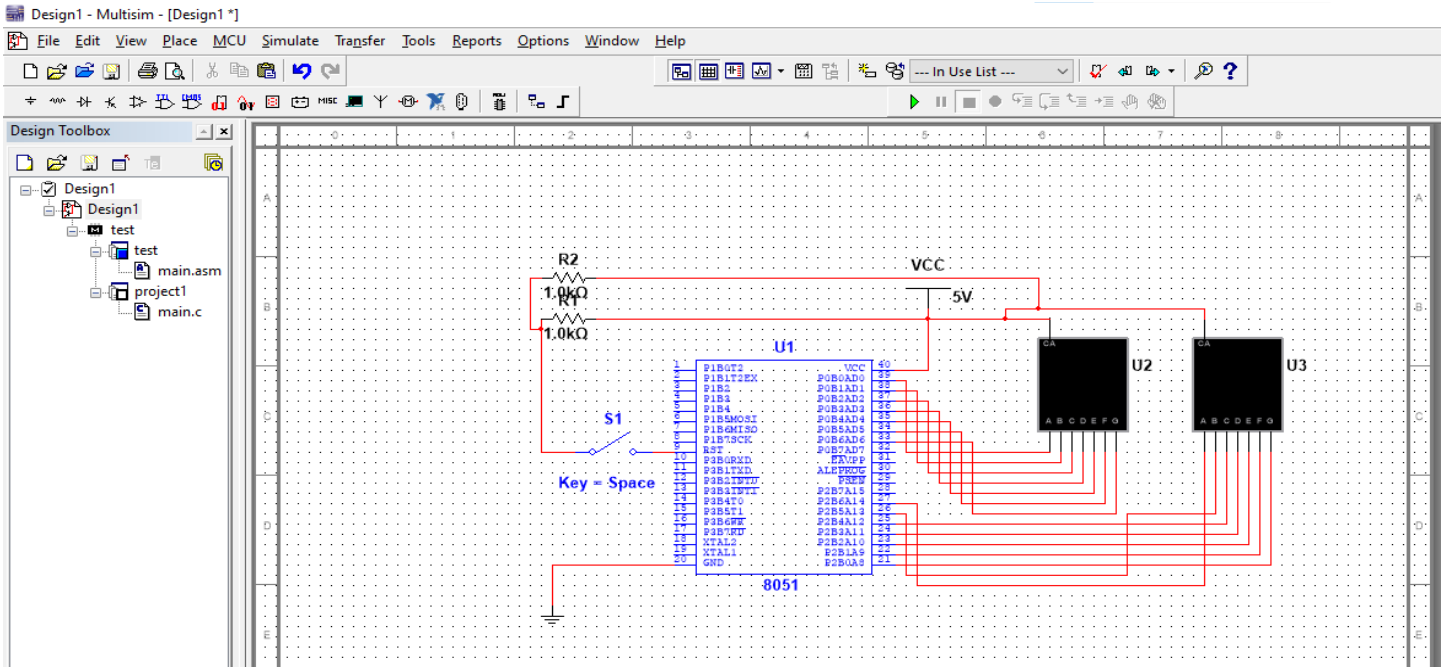
```
$MOD51; This includes 8051 definitions for the Metalink assembler
org 0000h

repeat:
mov p0,#10000001b ; displaying 0
acall delay
mov p0,#11001111b ; displaying 1
acall delay
mov p0,#10010010b ; displaying 2
acall delay
mov p0,#10000110b ; displaying 3
acall delay
mov p0,#11001100b ; displaying 4
acall delay
mov p0,#10100100b ; displaying 5
acall delay
mov p0,#10100000b ; displaying 6
acall delay
mov p0,#10001111b ; displaying 7
acall delay
mov p0,#10000000b ; displaying 8
acall delay
mov p0,#10000100b ; displaying 9
acall delay
sjmp repeat

delay:
mov r3,#010h
l3:mov r2,#0ffh
l2:mov r1,#0ffh
l1:djnz r1,l1
djnz r2,l2
djnz r3,l3
ret

END
```

2. **Draw the Block diagram of seven segment display on Multisim/Proteous and write its code in C and Assembly Write a program in Assembly and C to display a digit from 0 to 9 in both seven segment display.**




## C CODE

```
#include <htc.h>
#define Port_0 P0
#define Port_2 P2
// Define common anode seven-segment display patterns
unsigned char displayPatterns[] = {
    0x40, // 0
    0x79, // 1
    0x36, // 2
    0x48, // 3
    0x19, // 4
    0x12, // 5
    0x02, // 6
    0x78, // 7
    0x00, // 8
    0x10  // 9
};

void delay(unsigned int milliseconds) {
    unsigned int i, j;
    for (i = 0; i < milliseconds; i++) {
        for (j = 0; j < 110; j++);
    }
}

void main() {
    unsigned char i;
    while (1) {
        for (i = 0; i < 10; i++) {
            Port_0 = displayPatterns[i]; // Display the current digit
                  Port_2 = displayPatterns[i]; // Display the current digit
            delay(10);            // Delay for a while to display the digit
        }
    }}
```

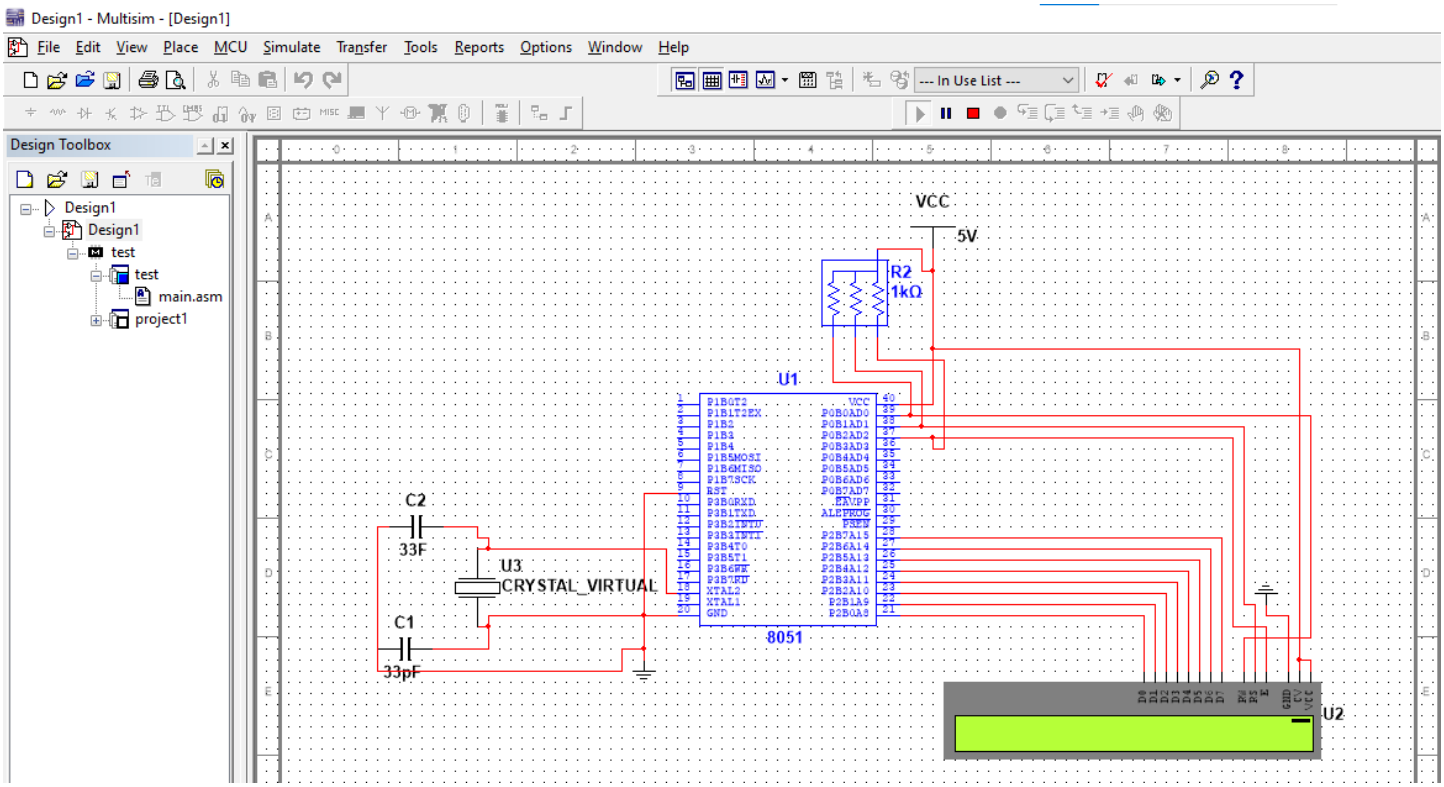
## ASSEMBLY CODE

```
$MOD51; This includes 8051 definitions for the Metalink assembler
org 0000h

repeat:
mov p0,#10000001b ; displaying 0
mov p2,#10000001b ; displaying 0
acall delay
mov p0,#11001111b ; displaying 1
mov p2,#11001111b ; displaying 1
acall delay
mov p0,#10010010b ; displaying 2
mov p2,#10010010b ; displaying 2
acall delay
mov p0,#10000110b ; displaying 3
mov p2,#10000110b ; displaying 3
acall delay
mov p0,#11001100b ; displaying 4
mov p2,#11001100b ; displaying 4
acall delay
mov p0,#10100100b ; displaying 5
mov p2,#10100100b ; displaying 5
acall delay
mov p0,#10100000b ; displaying 6
mov p2,#10100000b ; displaying 6
acall delay
mov p0,#10001111b ; displaying 7
mov p2,#10001111b ; displaying 7
acall delay
mov p0,#10000000b ; displaying 8
mov p2,#10000000b ; displaying 8
acall delay
mov p0,#10000100b ; displaying 9
mov p2,#10000100b ; displaying 9
acall delay
sjmp repeat

delay:
mov r3,#010h
l3:mov r2,#0ffh
l2:mov r1,#0ffh
l1:djnz r1,l1
djnz r2,l2
djnz r3,l3
ret
END
```

## 3. Write a program to display a welcome to bahria in LCD display using 8051 and Arduino.




### C CODE

```
#include <htc.h>
#include <stdio.h>

#define LCD_data P2
#define LCD_D7   P27
#define LCD_rs   P10
#define LCD_rw   P11
#define LCD_en   P12

void LCD_init();
void LCD_busy();
void LCD_command(unsigned char );
void LCD_senddata(unsigned char );
void LCD_sendstring(unsigned char *);

void main()
{
    unsigned char msg[] = "WELCOME TO BAHRIA";
     LCD_init();
     LCD_command(0x80);
     LCD_sendstring(msg);

    while(1);

}

void LCD_init()
{
     LCD_data = 0x38;      //Function set: 2 Line, 8-bit, 5x7 dots
     LCD_rs   = 0;         //Selected command register
     LCD_rw   = 0;         //We are writing in data register
```

```
     LCD_en   = 1;        //Enable H->L
     LCD_en   = 0;
     LCD_busy();          //Wait for LCD to process the command

     LCD_data = 0x0D;     //Display on, Curson blinking command
     LCD_rs   = 0;        //Selected command register
     LCD_rw   = 0;        //We are writing in data register
     LCD_en   = 1;        //Enable H->L
     LCD_en   = 0;
     LCD_busy();          //Wait for LCD to process the command

     LCD_data = 0x01;     //Clear LCD
     LCD_rs   = 0;        //Selected command register
     LCD_rw   = 0;        //We are writing in data register
     LCD_en   = 1;        //Enable H->L
     LCD_en   = 0;
     LCD_busy();          //Wait for LCD to process the command

     LCD_data = 0x06;     //Entry mode, auto increment with no shift
     LCD_rs   = 0;        //Selected command register
     LCD_rw   = 0;        //We are writing in data register
     LCD_en   = 1;        //Enable H->L
     LCD_busy();
}
void LCD_busy()
{
     LCD_D7   = 1;           //Make D7th bit of LCD as i/p
     LCD_en   = 1;           //Make port pin as o/p
     LCD_rs   = 0;           //Selected command register
     LCD_rw   = 1;           //We are reading
     while(LCD_D7)
    {
          LCD_en   = 0;     //Enable H->L
          LCD_en   = 1;
     }
}

void LCD_command(unsigned char var)
{
     LCD_data = var;      //Function set: 2 Line, 8-bit, 5x7 dots
     LCD_rs   = 0;        //Selected command register
     LCD_rw   = 0;        //We are writing in instruction register
     LCD_en   = 1;        //Enable H->L
     LCD_en   = 0;
     LCD_busy();          //Wait for LCD to process the command
}



void LCD_senddata(unsigned char var)
{
     LCD_data = var;      //Function set: 2 Line, 8-bit, 5x7 dots
     LCD_rs   = 1;        //Selected data register
     LCD_rw   = 0;        //We are writing
     LCD_en   = 1;        //Enable H->L
     LCD_en   = 0;
     LCD_busy();          //Wait for LCD to process the command
}
```

```
void LCD_sendstring(unsigned char *var)
{
     while(*var)              //till string ends
       LCD_senddata(*var++);  //send characters one by one
}
```

**ASSEMBLY CODE**

```
$MOD51
RS          EQU   P0.0
RW          EQU   P0.1
EN          EQU   P0.2
DATA_PORT   EQU   P2

NBYTES      EQU   R0
BYTE_IDX    EQU   R1

ORG   0H

; Initialize LCD
INIT:
      MOV   DPTR, #INIT_CMND
      MOV   NBYTES, #3        ; INIT_CMND has 3
commands = 3 bytes
      ACALL SEND_CMND_BYTES         ; Call
SEND_CMND_BYTES subroutine

; Main program
MAIN:
SEND_DATA1:
      MOV   DPTR, #LINE1            ; Load DPTR with
command to begin cursor at line 1
      MOV   NBYTES, #2        ; LINE1 has 2 commands =
2 bytes
      ACALL SEND_CMND_BYTES         ; Call
SEND_CMND_BYTES subroutine
      MOV   DPTR,       #DATA1            ; Load DPTR
with data for line 1
      MOV   NBYTES, #5        ; DATA1 has 5 char = 5
bytes
      ACALL SEND_DATA_BYTES         ; Call
SEND_DATA_BYTES subroutine

SEND_DATA2:
      MOV   DPTR, #LINE2            ; Load DPTR with
command to begin cursor at line 1
      MOV   NBYTES, #1        ; LINE2 has 1 command =
1 byte
      ACALL SEND_CMND_BYTES         ; Call
SEND_CMND_BYTES subroutine
      MOV   DPTR,       #DATA2            ; Load DPTR
with data for line 1
      MOV   NBYTES, #5        ; DATA2 has 5 char = 5
byte
      ACALL SEND_DATA_BYTES         ; Call
SEND_DATA_BYTES subroutine

      SJMP  MAIN              ; Jump back to MAIN
(repeat main program)

; SEND_DATA_BYTES subroutine: write one byte of data
to the LCD at a time
SEND_DATA_BYTES:
```

```
      MOV   BYTE_IDX,   #0          ; Initialize byte
idx at 0
SEND_DATA_BYTE:
      MOV   A,    BYTE_IDX          ; Load A with the
value of BYTE_IDX
      ACALL WRT_DATA                ; Send data to LCD
      ACALL DELAY                   ; Call DELAY
subroutine
      INC   BYTE_IDX                ; Increment byte
idx
      DJNZ  NBYTES, SEND_DATA_BYTE        ; Repeat for
each byte of data (loop until NBYTES = 0)
      RET

; SEND_CMND_BYTES subroutine: write one byte of
command(s) to the LCD at a time
SEND_CMND_BYTES:
      MOV   BYTE_IDX,   #0          ; Initialize byte
idx at 0
SEND_CMND_BYTE:
      MOV   A,    BYTE_IDX          ; Load A with the
value of BYTE_IDX
      ACALL WRT_CMND                ; Send command to
LCD
      ACALL DELAY                   ; Call DELAY
subroutine
      INC   BYTE_IDX                ; Increment byte
idx
      DJNZ  NBYTES, SEND_CMND_BYTE        ; Repeat for
each byte of command (loop until NBYTES = 0)
      RET


; WRT_CMND subroutine: send command to LCD
WRT_CMND:
      MOVC  A,    @A+DPTR           ; Address of the
desired byte in code space is formed by adding A +
DPTR
      MOV   DATA_PORT,  A     ; Load DATA_PORT with
contents of A
      CLR   RS                ; RS = 0 for command
      CLR   RW                ; RW = 0 for write
      SETB  EN                ; EN = 1 for high pulse
      ACALL DELAY             ; Call DELAY subroutine
      CLR   EN                ; EN = 0 for low pulse
      RET

; WRT_DATA subroutine: send data to LCD and display
WRT_DATA:
      MOVC  A,    @A+DPTR           ; Address of the
desired byte in code space is formed by adding A +
DPTR
      MOV   DATA_PORT,  A     ; Load DATA_PORT with
contents of A
      SETB  RS                ; RS = 1 for data
      CLR   RW                ; RW = 0 for write
      SETB  EN                ; EN = 1 for high pulse
      ACALL DELAY             ; Call DELAY subroutine
      CLR   EN                ; EN = 0 for low pulse
      RET
```

```
; DELAY subroutine
DELAY:            MOV   R3,   #255  ; Load R3 with 255
L2:         MOV   R4,   #2    ; Load R4 with 2
L1:         DJNZ  R4,   L1    ; Decrement R4, if not
zero repeat L1
            DJNZ  R3,   L2    ; Decrement R3, if not
zero repeat L1
            RET

; Define commands to initialize LCD display
; 38H: 8-bit, 2 line, 5x7 dots
; 0EH: Display ON cursor, ON
; 06H: Auto increment mode, i.e., when we send char,
cursor position moves right
INIT_CMND:  DB    38H,        0EH,        06H

; Define data to display on lines 1 and 2 of LCD
DATA1:      DB    Welcome To
DATA2:      DB    'Bahria'

; Define commands to go to line 1 of display
; 01H: Clear display
; 80H: Bring cursor to line 1
LINE1:      DB    01H, 80H

; Define command to go to line 2 of display
; 0C0H:     Bring cursor to line 2
LINE2:      DB    0C0H

END
```