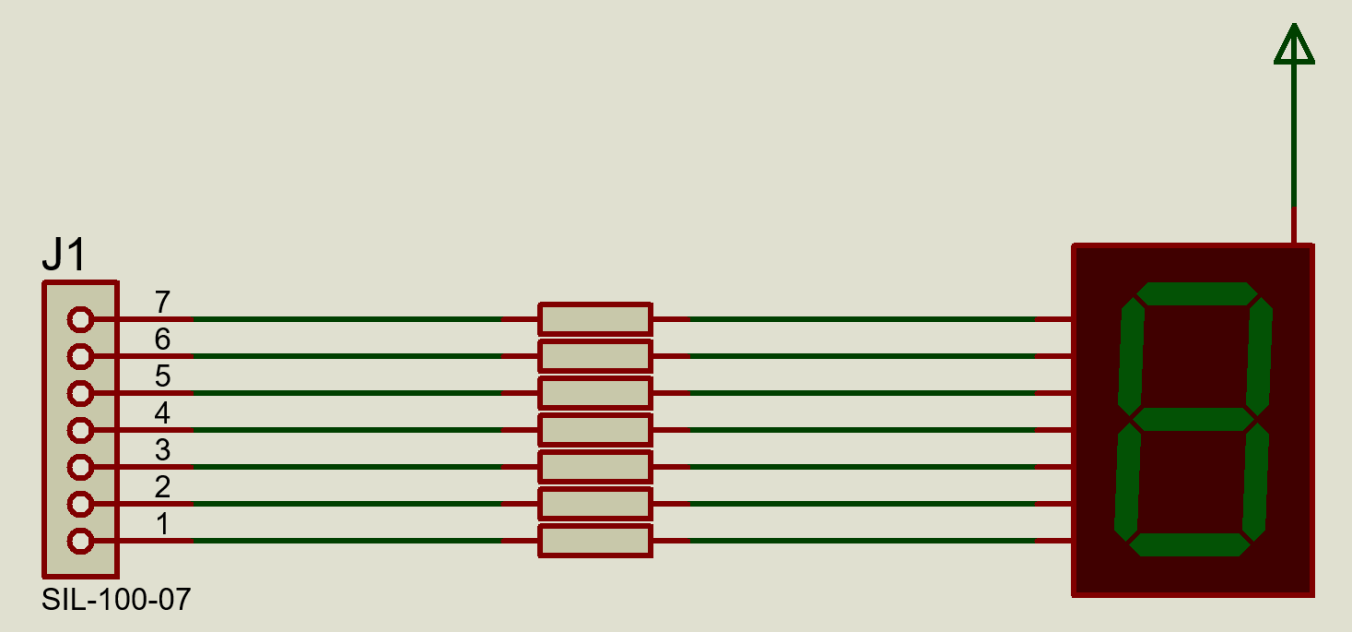
Pre Lab: Parallel Port

Group 1b2
E/20/157 – Janakantha S.M.B.G.
E/20/158 – Jananga T.G.C.

Part 01

1. Draw the circuit diagram that includes a 7-segment display and the data port of the parallel port. **Make sure that you have to connect separate resistors in series for each segment of SSD!**
   Calculate the resistance of the resistors that need to be connected.



$V_{HIGH} = 5V, i_{max} = 20\ mA, R_{min} = 250\ \Omega$
$\therefore R \geq 250\ \Omega$

2. Write a program to light up each segment of the SSD one by one.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

// Parallel port base address
#define DATA_PORT 0x378

// Port mapping
unsigned char segments[] = {
    0xFE, // a
    0xFD, // b
    0xFB, // c
    0xF7, // d
    0xEF, // e
    0xDF, // f
    0xBF  // g
};

```

```
// Variable to store write data
unsigned char data;

int main()
{
    // Logging errors
    if (ioperm(DATA_PORT, 1, 1))
    {
        fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
    }

    // Lighting each segment individually
    int i = 0;
    while (1)
    {
        // Select segment
        data = segments[i];

        // Write to the parallel port
        outb(data, DATA_PORT);
        printf("Lighting segment %x\n", data);

        // Wait 1 second
        sleep(1);

        // Move to the next segment
        i = (i + 1) % 7;
    }

    return 0;
```

Part 2: Display 0-9 numbers on a single 7 segment display

1. Differentiate between the common anode and common cathode 7-segment display.

   - Common Anode (CA):
     - All anodes of the LEDs are connected to +V.
     - A segment glows when its cathode is connected to LOW (0V).
     - Logic LOW = ON, HIGH = OFF.

   - Common Cathode (CC):
     - All cathodes of LEDs are connected to GND.
     - A segment glows when its anode is connected to HIGH (+V).
     - Logic HIGH = ON, LOW = OFF.

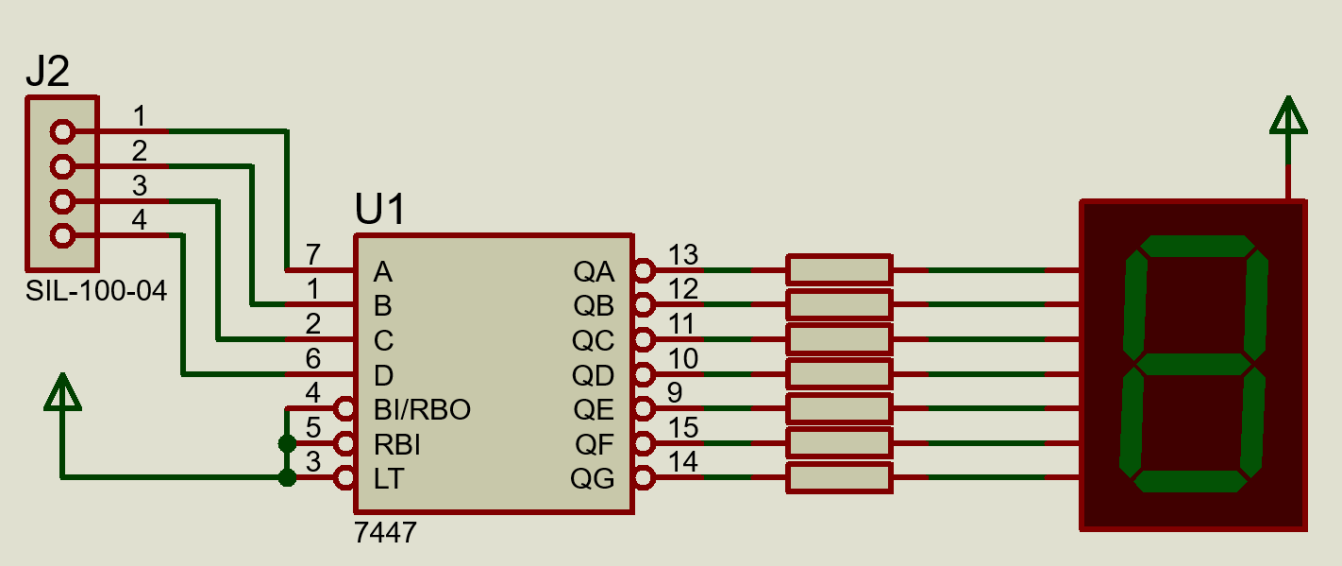
2. Write a program to display characters from 0-9 in an infinite loop with a delay of 1 second between each character.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

// Parallel port base address
#define DATA_PORT 0x378

// Character mapping
unsigned char characters[] = {
    0xC0, // 0
    0xF9, // 1
    0xA4, // 2
    0xB0, // 3
    0x99, // 4
    0x92, // 5
    0x82, // 6
    0xF8, // 7
    0x80, // 8
    0x90  // 9
};
// Reference: https://www.electronicsforu.com/resources/7-segment-display-pinout-understanding

// Variable to store write data
unsigned char data;

```

```c
void main()
{
    // Logging errors
    if (ioperm(DATA_PORT, 1, 1))
    {
        fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
    }

    // Displaying each character
    int i = 0;
    while (1)
    {
        // Select character
        data = characters[i];

        // Write to the parallel port
        outb(data, DATA_PORT);
        printf("Displaying: %d -> Hex: %02X\n", i, data);

        // Wait 1 second
        sleep(1);

        // Increment the count
        i = (i + 1) % 10;
    }
}
```

Part 3: Display 0-9 numbers on a single 7 segment display using 74LS47 IC

1. Draw the circuit diagram that includes a 7-segment display, 74LS47 IC, and the parallel port. Refer to the datasheet of the 74LS47 IC to find the least significant bit of the output. (Use a common anode)



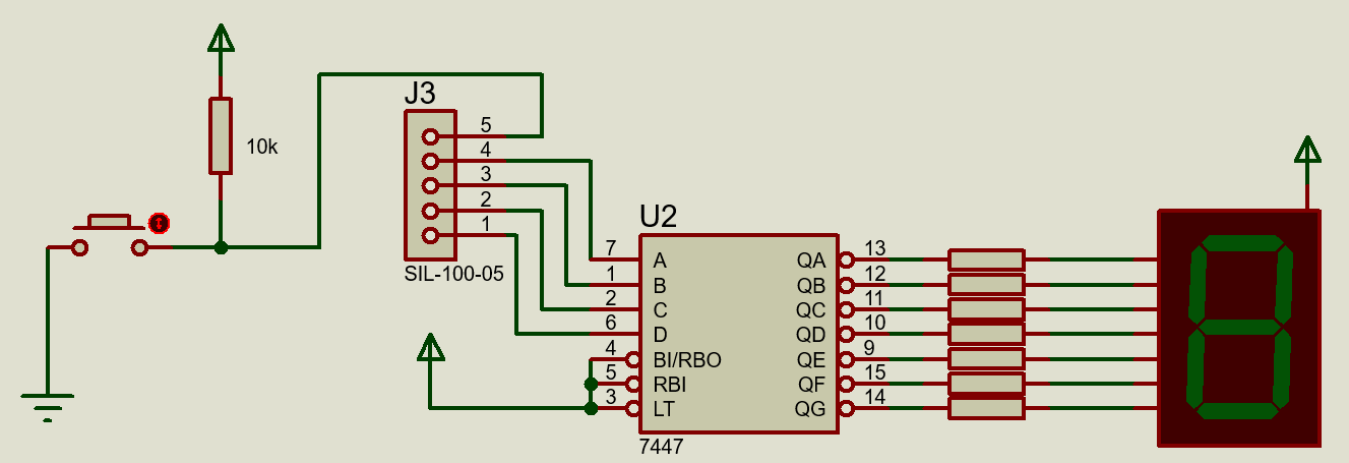
2. Write the program to display characters from 0-9 in an infinite loop with a delay of 1 second between each character.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

// Parallel port base address
#define DATA_PORT 0x378

// Binary mapping
unsigned char binaries[] = {
    0x10, // 0001 0000
    0x11, // 0001 0001
    0x12, // 0001 0010
    0x13, // 0001 0011
    0x14, // 0001 0100
    0x15, // 0001 0101
    0x16, // 0001 0110
    0x17, // 0001 0111
    0x18, // 0001 1000
    0x19  // 0001 1001
};

```

```c
// Variable to store write data
unsigned char data;

void main()
{
    // Logging errors
    if (ioperm(DATA_PORT, 1, 1))
    {
        fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
    }

    // Displaying each character
    int i = 0;
    while (1)
    {
        // Select character
        data = binaries[i];

        // Write to the parallel port
        outb(data, DATA_PORT);
        printf("Displaying: %d -> Input: %02X\n", i, data);

        // Wait 1 second
        sleep(1);

        // Increment the count
        i = (i + 1) % 10;
    }
}
```

Part 4: Change the numbers displayed in the SSD with a push button

1. **Draw the circuit diagram** that includes a push button to take inputs, a 7-segment display, a **74LS47 IC** to show outputs and the parallel port. Make sure you use **proper resistors (pull-up/pull-down)** when taking inputs through the push button.




```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/io.h>

// Parallel port base address
#define DATA_PORT 0x378
#define STATUS_PORT (DATA_PORT + 1)

// Note: In default state, status port's last 3 bits are internally HIGH.
// When the button is pressed, it pulls one bit LOW, resulting in 0x07.
// Make sure to check the default state of your status port.
#define BUTTON_PRESSED 0x07

// Binary mapping
unsigned char binaries[] = {
    0x10, // 0001 0000
    0x11, // 0001 0001
    0x12, // 0001 0010
    0x13, // 0001 0011
    0x14, // 0001 0100
    0x15, // 0001 0101
    0x16, // 0001 0110
    0x17, // 0001 0111
    0x18, // 0001 1000
    0x19  // 0001 1001
};
```

```
28
29  // Variable to store write data
30  unsigned char data, status;
31
32  int main()
33  {
34      // Logging errors
35      if (ioperm(DATA_PORT, 1, 1))
36      {
37          fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
38      }
39
40      if (ioperm(STATUS_PORT, 1, 1))
41      {
42          fprintf(stderr, "Access denied to %x\n", STATUS_PORT), exit(1);
43      }
44
45      // Displaying each character
46      int i = 0;
47      while (1)
48      {
49          // Check button press (active LOW)
50          status = inb(STATUS_PORT);
51          printf("Ready for button press. \t Default STATUS PORT: %02X\n", status);
52          if (status == BUTTON_PRESSED)
53          {
54              usleep(1000); // debounce delay
55              if (inb(STATUS_PORT) == BUTTON_PRESSED)
56              {
57                  // Select character
58                  data = binaries[i];
59
60                  // Write to the parallel port
61                  outb(data, DATA_PORT);
62
63                  // Increment the count
64                  i = (i + 1) % 10;
65              }
66          }
67      }
68
69      return 0;
70  }
```

Lab Setup: