

# Pre Lab: Parallel Port

Group 1b2

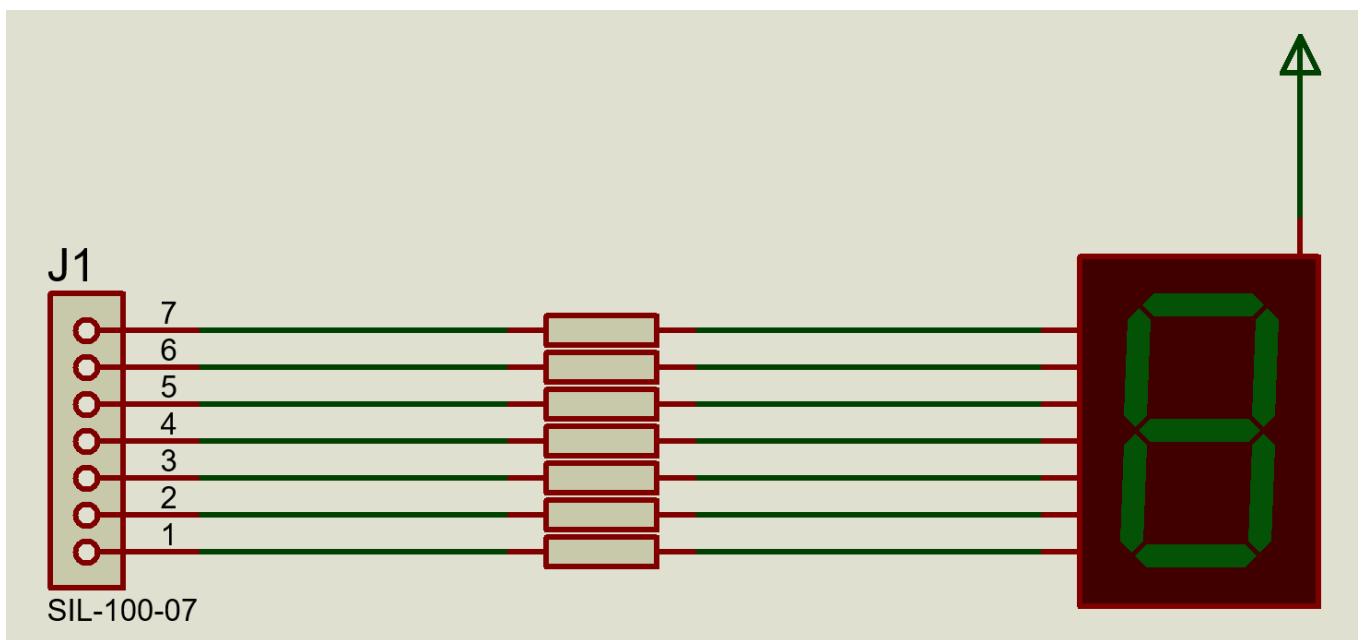
E/20/157 – Janakantha S.M.B.G.

E/20/158 – Jananga T.G.C.

E/20/168 – Jayasinghe B.V.R.R.

## 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.



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

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

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/io.h>
5
6 // Parallel port base address
7 #define DATA_PORT 0x378
8
9 // Port mapping
10 unsigned char segments[] = {
11     0x01, // a
12     0x02, // b
13     0x04, // c
14     0x08, // d
15     0x10, // e
16     0x20, // f
17     0x40 // g
18 };
19
```

```
20
21 // Variable to store write data
22 unsigned char data;
23
24 void main()
25 {
26     // Logging errors
27     if (ioperm(DATA_PORT, 1, 1))
28     {
29         fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
30     }
31
32     // Lighting each segment individually
33     for (int i = 0; i < 7; i++)
34     {
35         // Select segment
36         data = segments[i];
37
38         // Write to the parallel port
39         outb(data, DATA_PORT);
40
41         // Wait 1 second
42         sleep(1);
43     }
44
45     // Turn OFF all segments
46     data = 0x00;
47     outb(data, DATA_PORT);
48
49     return 0;
50 }
```

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

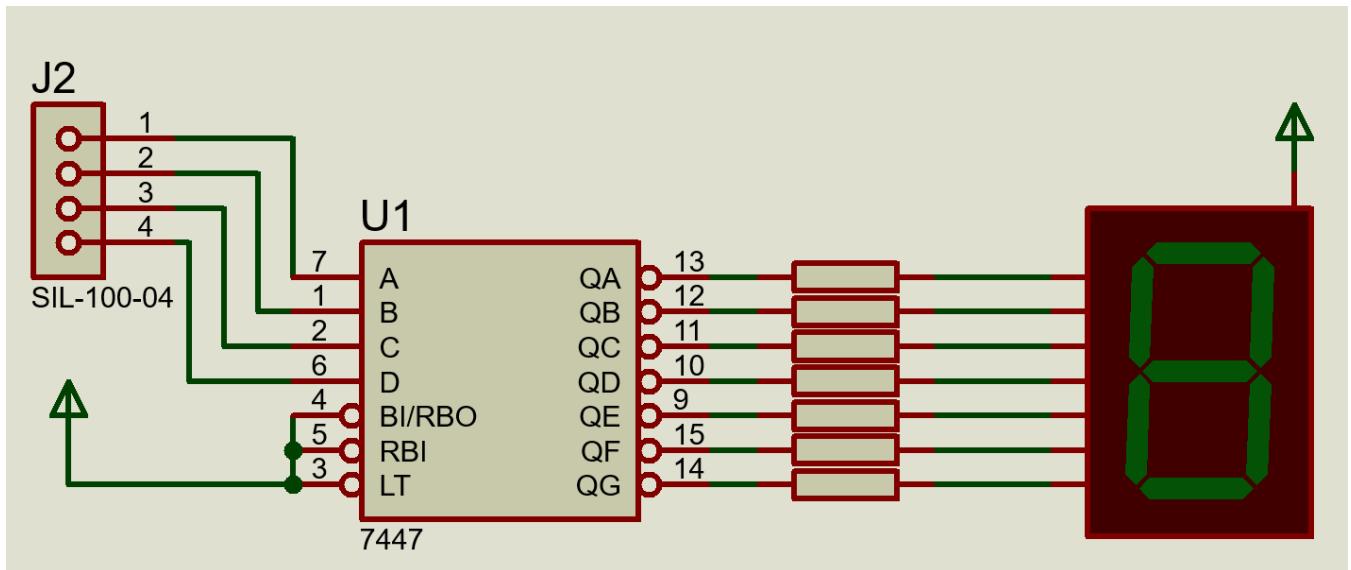
1. Differentiate between the common anode and common cathode 7-segment display.
  - Common Anode (CA):
    - All anodes of the LEDs are connected to +V.
    - A segment glows when its cathode is connected to LOW (0V).
    - Logic LOW = ON, HIGH = OFF.
  - Common Cathode (CC):
    - All cathodes of LEDs 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.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/io.h>
5
6 // Parallel port base address
7 #define DATA_PORT 0x378
8
9 // Character mapping
10 unsigned char characters[] = {
11     0x3F, // 0
12     0x06, // 1
13     0x5B, // 2
14     0x4F, // 3
15     0x66, // 4
16     0x6D, // 5
17     0x7D, // 6
18     0x07, // 7
19     0x7F, // 8
20     0x67, // 9
21 };
22 // Reference: https://hosteng.com/dmdhelp/content/instruction\_set/SEG\_Hex\_BCD\_to\_7\_Segment\_Display.htm
23
24 // Variable to store write data
25 unsigned char data;
```

```
27 void main()
28 {
29     // Logging errors
30     if (ioperm(DATA_PORT, 1, 1))
31     {
32         fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
33     }
34
35     // Displaying each character
36     int i = 0;
37     while (1)
38     {
39         // Select character
40         data = characters[i];
41
42         // Write to the parallel port
43         outb(data, DATA_PORT);
44
45         // Wait 1 second
46         sleep(1);
47
48         // Increment the count
49         i = (i > 8) ? 0 : i + 1;
50     }
51 }
```

### 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)



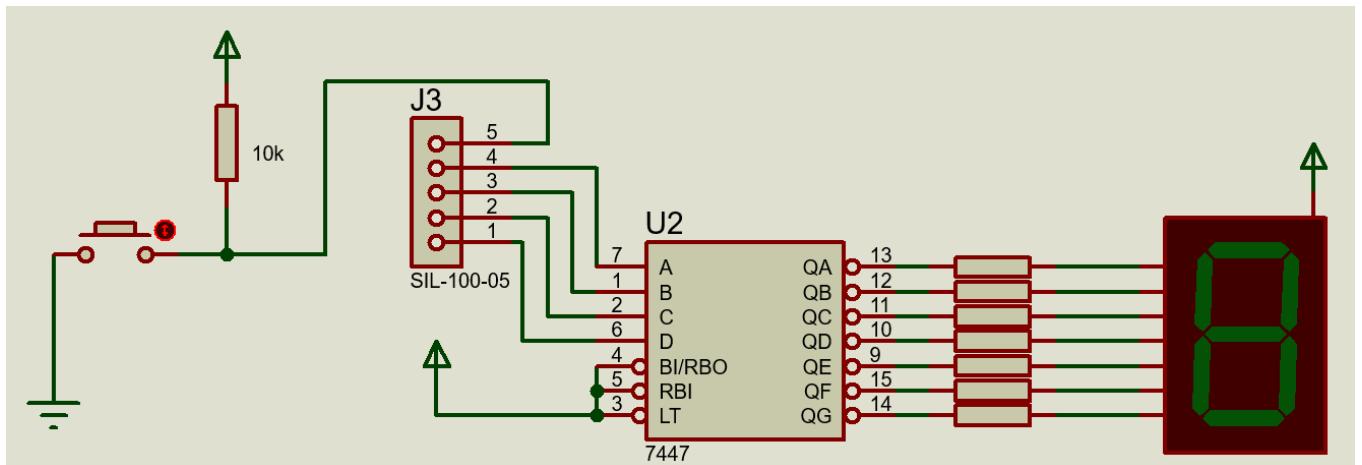
2. Write the program to display characters from 0-9 in an infinite loop with a delay of 1 second between each character.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/io.h>
5
6 // Parallel port base address
7 #define DATA_PORT 0x378
8
9 // Binary mapping
10 unsigned char binaries[] = {
11     0x00, // 0000
12     0x01, // 0001
13     0x02, // 0010
14     0x03, // 0011
15     0x04, // 0100
16     0x05, // 0101
17     0x06, // 0110
18     0x07, // 0111
19     0x08, // 1000
20     0x09 // 1001
21 };
22
```

```
23 // Variable to store write data
24 unsigned char data;
25
26 void main()
27 {
28     // Logging errors
29     if (ioperm(DATA_PORT, 1, 1))
30     {
31         fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1);
32     }
33
34     // Displaying each character
35     int i = 0;
36     while (1)
37     {
38         // Select character
39         data = binaries[i];
40
41         // Write to the parallel port
42         outb(data, DATA_PORT);
43
44         // Wait 1 second
45         sleep(1);
46
47         // Increment the count
48         i = (i > 8) ? 0 : i + 1;
49     }
50 }
```

## 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.



```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/io.h>
5
6 // Parallel port base address
7 #define DATA_PORT 0x378
8 #define STATUS_PORT (DATA_PORT + 1)
9
10 // Binary mapping
11 unsigned char binaries[] = {
12     0x00, // 0000
13     0x01, // 0001
14     0x02, // 0010
15     0x03, // 0011
16     0x04, // 0100
17     0x05, // 0101
18     0x06, // 0110
19     0x07, // 0111
20     0x08, // 1000
21     0x09 // 1001
22 };
23
24 // Variable to store write data
25 unsigned char data, status;
```

```
26
27 int main()
28 {
29     // Logging errors
30     if (ioperm(DATA_PORT, 1, 1))
31     {
32         fprintf(stderr, "Access denied to %x\n", DATA_PORT), exit(1)
33     }
34
35     if (ioperm(STATUS_PORT, 1, 1))
36     {
37         fprintf(stderr, "Access denied to %x\n", STATUS_PORT), exit
38             (1);
39     }
40
41     // Displaying each character
42     int i = 0;
43     while (1)
44     {
45         // Check button press (active LOW)
46         if (inb(STATUS_PORT) == 0x00)
47         {
48             usleep(1000); // debounce delay
49             if (inb(STATUS_PORT) == 0x00)
50             {
51                 // Select character
52                 data = binaries[i];
53
54                 // Write to the parallel port
55                 outb(data, DATA_PORT);
56
57                 // Increment the count
58                 i = (i > 8) ? 0 : i + 1;
59             }
60         }
61
62     return 0;
63 }
```