BILKENT UNIVERSITY

COMPUTER SCIENCE

CS 224 : COMPUTER ORGANIZATION

PRELIMINARY DESIGN REPORT

LAB 7

SECTION 3 & 4, respectively

BATUHAN ÖZÇÖMLEKÇİ / 21703297

ALPEREN CAN / 21601740

25.12.2019

PRELIMINARY REPORT PART B

The I/O Ports module consists of Special Function Registers (SFRs) such as TRISx, PORTx, Latx and ODCx.

TRISx refers to the data direction register for the module x. Tri-State registers configure the data direction flow through port I/O pins. Moreover, TRIS register bits determine whether a PORT I/O pin is an input or output. TRIS bit set which is equal to 1 configures the I/O port pin as an input, where 0 configures as an output. The last value written to the TRIS register is read during the read operation. After a power-on reset, all I/O pins are defined as inputs.

PORTx refers to PORT register for the module x. Port registers allow I/O pins to be read. A write to a port register writes to the corresponding LAT register, acting same as write to a LAT register. However, a read from a PORT register reads the synchronized signal applied to the port I/O pins.

LATx refers to Latch register for the module x. LAT registers (PORT data latch) hold data written to port I/O pins. Write to a LAT register latches data to corresponding port I/O pins, where a read from LAT register reads the data held in the PORT data latch, <u>NOT</u> from the port I/O pins.(Unlike PORTx)

ODCx refers to Open-Drain control register for the module x. Each I/O pin can be individually configured for normal digital output or open-drain output. This is controlled by ODC register ODCx, associated with each I/O pin. If the ODC bit for an I/O pin equals to 1, pin acts as an open-drain output. If the pin equals to 0, then the pin is configured for a normal digital output. ODC bit is valid only for output pins. Also after a reset, all the bits of the ODCx register is set to 0.

For part c), we used TRISD, TRISA, PORTA, PORTD.

For part d), we used TRISD, TRISE, PORTD, PORTE.

PRELIMINARY REPORT PART C

```
// This code shows and rotates the pattern (10001000) right or stops based on the input coming from
the user. The pattern is to be shown on the LEDs.

int stop = 0;

int initial = 0b10001000;           // Initial pattern. Note that 0 means on, while 1 means off.

int right = 1;

int DIR = 0;                        // Input DIR

int EN = 1;                         // Input EN

void main() {

   TRISD = 0x0;            // All bits of PORTD are output. ~0 means output~

   // Three bits of PORTA are inputs

   TRISA = 0b111;

   // From PORTD, outputs will be sent to LEDs.

   // Initial pattern is sent to the LEDs through PORTD.

   PORTD = initial;

   while (1) {

         if( DIR == 0 && EN == 1 )

         {

           int lsb;          // the least significant bit

           int mask;

           // Stop button is the push-button which is labeled as 1 on the board.

           if (PORTABits.RA1 == 0) {       // If stop button clicked

                  stop = !stop;

                  if (!stop){

                  // If process restarted, copy initial pattern into PORTD.

                  PORTD = initial;

                  }

           }

           if ( !stop ) {

              //Rotate right
```

```
      lsb = PORTD & 0x1;            // Extract least significant bit
      mask = lsb << 7;   // Least significant bit will be the msb of the shifted pattern
      PORTD = (PORTD >> 1) | mask;   // Paste lsb to the leftmost bit the right shifted portd
   }  else {
     //Do not shift anything, that is, stop.
     PORTD = 0b11111111;
   }
 }
 else if ( DIR == 1 && EN ==1 )
{
   int msb;         // the most significant bit
   int mask;
   // Stop button is the push-button which is labeled as 1 on the board.
   if( PORTABits.RA1 == 0 ){        // If stop button clicked
            stop = !stop;
            if( !stop ){
            // If process restarted, copy initial pattern into PORTD.
            PORTD = initial;
            }
   }
   if ( !stop ){
       //Rotate left
      msb = PORTD & 0b10000000;     // Extract most significant bit
      mask = msb >> 7;  // Most significant bit will be the lsb of the shifted pattern
      PORTD = (PORTD << 1) | mask;   // Paste msb to the rightmost bit the left shifted portd
   }  else {
     //Do not shift anything, that is, stop.
     PORTD = 0b11111111;
   }
 }
```

```
        else if ( EN == 0 )            // If EN equals to zero
        {
            PORTD = PORTD;      // The position is "frozen"
        }
        // DIR button is the push button which is labeled as 2 on the board.
        if ( PORTABits.RA2 == 0 ) {         // If DIR button is clicked
            DIR = !DIR;
        }
        // EN button is the push button which is labeled as 3 on the board.
        if ( PORTABits.RA3 == 0 ) {         // If EN button is clicked
              EN = !EN;
        }
        delay_ms(1000); // Wait 1 second.
    }
}
// Rotation ends here
```

PRELIMINARY REPORT PART D

```
int x = 1;
int value = 1;     // value is f(x)
int first;         // rightmost digit of f(x)
int second;        // second rightmost digit of f(x)
int third;         // third rightmost digit of f(x)
int fourth;        // leftmost digit of f(x)
int increment;  // counter for the loop
void main () {
   TRISD = 0x0; // All bits of PORTB are output. ~0 means output~
   TRISE = 0x0; // All bits of PORTB are output. ~0 means output~
   // From PORTD, outputs will be sent to 7-Segment
   While (1) {
         //Display
         first = value % 10;          // takes the rightmost digit
         value = value /10;
         second = value % 10;     // takes the second rightmost digit
         value = value /10;
         third = value % 10;        // takes the third rightmost digit
         value  = value /10;
         fourth = value %10;       // takes the leftmost digit
         // Since there are four 1ms delays among following switch statements, it should loop 250
         //times in order to delay a total of 1 second between f(x) values.
         for ( increment = 0; increment< 250; increment++ )
         {
            switch( first ){          // Following case statements display the digit at 7-Segment
            case 0:   PORTD  =   0x3F; break;
            case 1:   PORTD  =   0x06; break;
            case 2:   PORTD  =   0x5B; break;
            case 3:   PORTD  =   0x4F; break;
```

```
case 4:   PORTD  =   0x66; break;
case 5:   PORTD  =   0x6D; break;
case 6:   PORTD =   0x7D; break;
case 7:   PORTD  =  0x07; break;
case 8:   PORTD  = 0x7F; break;
case 9:   PORTD = 0x6F; break;
default:   0x00; break;
}
PORTE = 0x8;                    // Choose the appropriate AN input (4th)
delay_ms(1);                    // Delay 1 ms
switch( second ){               // Following case statements display the digit at 7-Segment
case 0:   PORTD  =   0x3F; break;
case 1:   PORTD  =   0x06; break;
case 2:   PORTD  =   0x5B; break;
case 3:   PORTD  =   0x4F; break;
case 4:   PORTD  =   0x66; break;
case 5:   PORTD  =   0x6D; break;
case 6:   PORTD =   0x7D; break;
case 7:   PORTD  =  0x07; break;
case 8:   PORTD  = 0x7F; break;
case 9:   PORTD = 0x6F; break;
default:   0x00; break;
}
PORTE = 0x4;                    // Choose the appropriate AN input (3th)
delay_ms(1);                    // Delay 1 ms
 switch( third ){               // Following case statements display the digit at 7-Segment
 case 0:   PORTD  =   0x3F; break;
 case 1:   PORTD  =   0x06; break;
 case 2:   PORTD  =   0x5B; break;
 case 3:   PORTD  =   0x4F; break;
 case 4:   PORTD  =   0x66; break;
```

```
    case 5:   PORTD  =   0x6D; break;
    case 6:   PORTD =   0x7D; break;
    case 7:   PORTD  =  0x07; break;
    case 8:   PORTD  = 0x7F; break;
    case 9:   PORTD = 0x6F; break;
    default:   0x00; break;
    }
    PORTE = 0x2;                    // Choose the appropriate AN input (2nd)
    delay_ms(1);                    // Delay 1 ms
    switch( fourth ){               // Following case statements display the digit at 7-Segment
    case 0:   PORTD  =   0x3F; break;
    case 1:   PORTD  =   0x06; break;
    case 2:   PORTD  =   0x5B; break;
    case 3:   PORTD  =   0x4F; break;
    case 4:   PORTD  =   0x66; break;
    case 5:   PORTD  =   0x6D; break;
    case 6:   PORTD =   0x7D; break;
    case 7:   PORTD  =  0x07; break;
    case 8:   PORTD  = 0x7F; break;
    case 9:   PORTD = 0x6F; break;
    default:   0x00; break;
    }
    PORTE = 0x1;                    // Choose the appropriate AN input (1st)
    delay_ms(1);                    // Delay 1 ms
}
// return to one if the limit is reached
if( x == 21 )
{
  x = 1;
  value = 1;
}
```

```
        else
        {
           x = (x+1);              // increment x
           value = x*x*x;          // take its cube
        }
    }
}
```