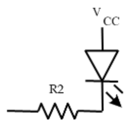
HW 2 Solution

1. A 3:8 decoder converts 3 bit input into a 8 bit output. E.g.
   if input = 3 (011b), output pin 3 is HI, others are LO (00001000b);
   if input = 5 (101b), output pin 5 is HI, others are LO (00100000b).
   The decoder input is driven by port B of the microcontroller, and its output is connected to 8 LEDs in reverse-drive orientation. Write a code to make the LEDs light sequentially from top to bottom and back to top in an infinite loop.

   Reverse 

drive

   To light an LED output 0. To turn off an LED output 1.

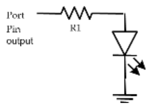
   PORTB -> 3:8 decoder -> LED

   PORTB2-0 to connect the 3:8 decoder

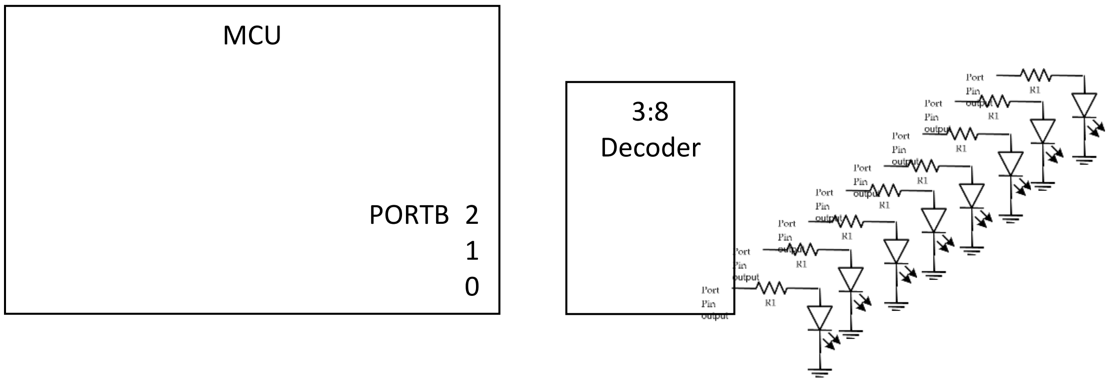
   To light 1st LED then PORTB = 1,2,3,4,5,6,7 (skip 0, skip the number that you want to lit)
   To light 2nd LED then PORTB = 0,2,3,4,5,6,7 (skip 1)
   To light 3rd LED then PORTB = 0,1,3,4,5,6,7 (skip 2)

```
DDRB = 0xFF;
while(1)
{
    for(int i=0; i<8; i++)
        for(int j=i+1; j < i+8; j++)
        {   PORTB = j%8;            //remainder of j/8
            delay1ms(1);
        }
    }

    for(int i=7; i=0; i++)
        for(int j=i+1; j < i+8; j++)
        {   PORTB = j%8;            //remainder of j/8
            delay1ms(1);
        }
    }
}
```

Alternatively, redefine problem so that LED orientation is positive-drive orientation.
To turn LED on output a 1.



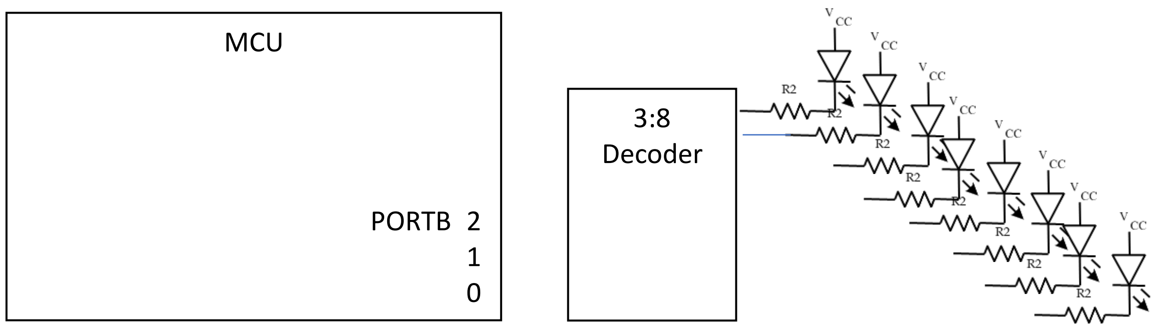
```
DDRB = 0xFF;
while(1)
{
    for(int i=0; i<8; i++)
        PORTB = i;
        delay1ms(100);
    }

    for(int i=7; i=0; i++)
        PORTB = i;
        delay1ms(100);
    }
}
```



Alternatively, use reverse-drive but the 3:8 decoder is such that
if input = 3 (011b), output pin 3 is HI,
others are LO (11110111b);
if input = 5 (101b), output pin 5 is HI, others are LO (11011111b).

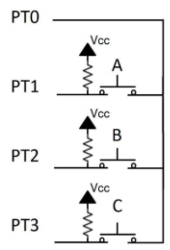
```
DDRB = 0xFF;
while(1)
{
    for(int i=0; i<8; i++)
        PORTB = i;
        delay1ms(100);
    }

    for(int i=7; i=0; i++)
        PORTB = i;
```

```
            delay1ms(100);
        }
}
```



2. Given the circuit in Figure 2 (see attached file). Select a port and write a program that will determine which button is pressed.

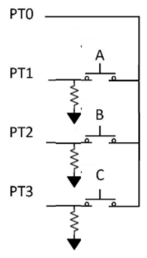
   Output a LO on PT0 and scan PT1-3. A “1” input => button not pressed. A “0” input => button pressed.



PT0-3 to PORTB0-3

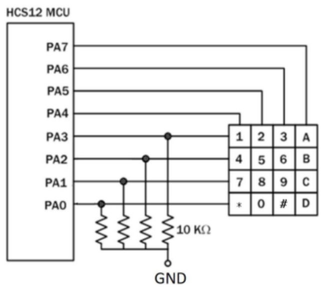
```
char i,button;

DDRB = 0x01;                              //PTB0 is output; PTB1-3 is input
PORTB &= 0xFE;                            //PTB0=0 (PORTB & 0b11111110)
switch(~PINB & 0x0E)                      //grab PTB3-1
{   case(0b0000010): button = “A”; break;
    case(0b0000100): button = “B”; break;
    case(0b0001000): button = “C”; break;
    default:
}
```

Alternatively, if the diagram shown input being GND instead of VCC

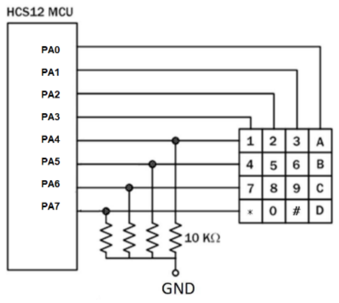


```
char i,button;

DDRB = 0x01;                              //PTB0 is output; PTB1-3 is input

PORTB |= 0x01;                            //PTB0=1 (PORTB | 0b00000001)
switch(PINB & 0x0E)                       //grab PTB3-1
{   case(0b0000010): button = “A”; break;
    case(0b0000100): button = “B”; break;
    case(0b0001000): button = “C”; break;
    default:
}
```

3. For the circuit in figure 3, derive a table to determine the values in PORTA for each key.

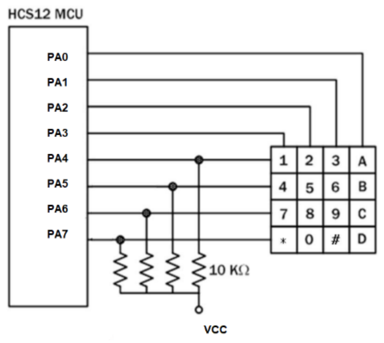


Key	PORTA7-4	PINA3-0
1	0b0001xxxx	0bxxxx1000
2	0b0010xxxx	0bxxxx1000
3	0b0100xxxx	0bxxxx1000
A	0b1000xxxx	0bxxxx1000
4	0b0001xxxx	0bxxxx0100
5	0b0010xxxx	0bxxxx0100
6	0b0100xxxx	0bxxxx0100
B	0b1000xxxx	0bxxxx0100
7	0b0001xxxx	0bxxxx0010
8	0b0010xxxx	0bxxxx0010
9	0b0100xxxx	0bxxxx0010
C	0b1000xxxx	0bxxxx0010
*	0b0001xxxx	0bxxxx0001
0	0b0010xxxx	0bxxxx0001
#	0b0100xxxx	0bxxxx0001
D	0b1000xxxx	0bxxxx0001



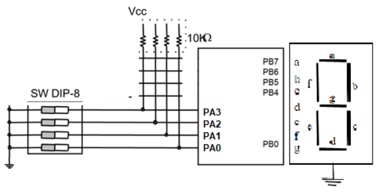
Key	PORTA3-0	PINA7-4
1	0bxxxx1000	0b0001xxxx
2	0bxxxx0100	0b0001xxxx
3	0bxxxx0010	0b0001xxxx
A	0bxxxx0001	0b0001xxxx
4	0bxxxx1000	0b0010xxxx
5	0bxxxx0100	0b0010xxxx
6	0bxxxx0010	0b0010xxxx
B	0bxxxx0001	0b0010xxxx
7	0bxxxx1000	0b0100xxxx
8	0bxxxx0100	0b0100xxxx
9	0bxxxx0010	0b0100xxxx
C	0bxxxx0001	0b0100xxxx
*	0bxxxx1000	0b1000xxxx
0	0bxxxx0100	0b1000xxxx
#	0bxxxx0010	0b1000xxxx
D	0bxxxx0001	0b1000xxxx

Alternatively, use VCC for inputs.



Key	PORTA3-0	PINA7-4
1	0bxxxx0111	0b1110xxxx
2	0bxxxx1011	0b1110xxxx
3	0bxxxx1101	0b1110xxxx
A	0bxxxx1110	0b1110xxxx
4	0bxxxx0111	0b1101xxxx
5	0bxxxx1011	0b1101xxxx
6	0bxxxx1101	0b1101xxxx
B	0bxxxx1110	0b1101xxxx
7	0bxxxx0111	0b1011xxxx
8	0bxxxx1011	0b1011xxxx
9	0bxxxx1101	0b1011xxxx
C	0bxxxx1110	0b1011xxxx
*	0bxxxx0111	0b0111xxxx
0	0bxxxx1011	0b0111xxxx
#	0bxxxx1101	0b0111xxxx
D	0bxxxx1110	0b0111xxxx

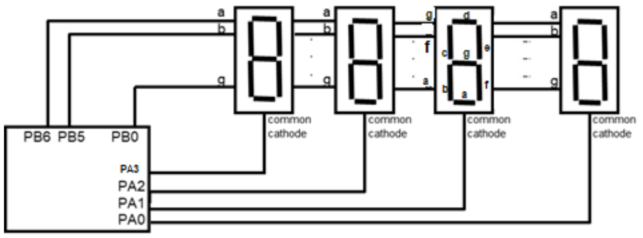
4. A quad DIPS and a common cathode 7SD are interfaced to the microcontroller via PORTA and PORTB respectively. Write a program that will read the DIP switches and convert their settings to hex digit then display that digit on the 7SD. Assume that the 7SD is grounded. Draw the diagram for the circuit and explain where all pins are connected.



PA3-0 connected to 4 input DIPS. PB7-0 connected to segments a,b,c,…,g

```
char SSD[] = {0x7e, 0x30, 0x6d, 0x79, 0x33, 0x5b, 0x5f, 0x70, 0x7f, 0x7b,
              0b111101, 0b0011111, 0b0001101, 0b0111101, 0x1101111, 0x1000111};
unsigned int i;

DDRA &= 0xF0;       //PTA3-0 input
DDRB |= 0x7F;       //PTB7-0 output

while(1)
{    i = (unsigned int) (~PINA & 0x0F);
     PORTB = SSD[i];
}
```

5. The variable time is a string of containing the two digits for the hours and 2 digits for the minutes, e.g. "08:30". The inputs of four common cathode 7SDs are connected to PORTB. The GND pins are connected to PORTA pins3-0. Write a program that will display the time on the 7SDs. Draw a circuit diagram. Be sure to use the dot segment on 2 and 3 7SD.



PA7 to connect to the dot segment.
PB6-0 connect to the segments a-g.

```
unsigned char SSD[] = {0x7e, 0x30, 0x6d, 0x79, 0x33, 0x5b, 0x5f, 0x70, 0x7f, 0x7b}
unsigned char SSD2[] = {0b0111111, 0b1011011, 0b1111001, 0b1110100, 0b1101101, 0b1101111,
            0b0111000, 0b1111111, 0b1111101}
unsigned time[] = "08:30"
int i;
DDRB = 0xFF;
DDA &= 0x0F;

while(1)
{    i = (unsigned int) (time[0] - 0x30);
     PORTB = SSD[i];
     PORTA = 0b0111;
     delay1ms(10);

     i = (unsigned int) (time[1] - 0x30);
     PORTB = SSD[i] | 0x80;
     PORTA = 0b1011;
     delay1ms(10);

     i = (unsigned int) (time[3] - 0x30);
     PORTB = SSD2[i] | 0x80;
     PORTA = 0b1101;
     delay1ms(10);

     i = (unsigned int) (time[4] - 0x30);
     PORTB = SSD[i] ;
     PORTA = 0b1110;
     delay1ms(10);

}
```

6. Rewrite str2lcd: must check for 16 characters and 32 characters using a character count. When the count = 16 then go to the next line using a command that changes the address to 0x40. When the count = 32 create a 5sec delay before erasing the LCD and.

```
void str2LCD (char * ptr)
{    int count=0;
     while (*ptr ) //ascii Z strings (null terminated)
     {
          byte2LCD(* ptr , LCD_RS);
          ptr++;
          count++;
          if (count%16)                    //=1 when count is a multiple of 16
               byte2LCD (0xC0, ~LCD_RS);
          if (count%32)                    //=1 when count is a multiple of 32
               byte2LCD (0x01, ~LCD_RS);
     }
}
```