ECE 376 - Homework #2

Assembler, Flow Charts, Binary Inputs. Due Wednesday, September 7th

Assembler Programming

1) Determine the contents of registers W, A, and B after each assembler command:

Command	W	A	B
; Start	5	6	7
incf A,F	5	**7**	7
decf B,W	**6**	7	7
addwf A,F	6	**13**	7
sublw 3	**-3 = 253** W = 3 - W	13	7
andwf A,F	253	**13** 13 = 0000 1101 253 = 1111 1101 ---------- and = 0000 1101	7
iorwf B,F	253	13	**255** 253 = 1111 1101 7 = 0000 0111 ---------- OR = 1111 1111

2) Convert the following C code to assembler (8-bit operations)

```
; unsigned char A, B, C;

A       equ       0
B       equ       1
C       equ       2

; A = 4*B + 5*C + 6;

        movlw     6

        addwf     B,W
        addwf     B,W
        addwf     B,W
        addwf     B,W

        addwf     C,W
        addwf     C,W
        addwf     C,W
        addwf     C,W
        addwf     C,W

        movwf     A
```

Not stylish, but things that seem dumb actually work in assembler. (there are other solutions)

Another solution....

```
        movlw     6
        movwf     A

        movf      B,W
        mullw     4
        movf      PRODL,W
        addwf     A,F

        movf      C,W
        mullw     5
        movwf     PRODL,W
        addwf     A,F
```

3) Convert the following C code to assembler: (16-bit operations)

```
; unsigned int A, B, C;

A       equ       0
B       equ       2
C       equ       4

;A = 4*B + 5*C + 6;

; A = 6
        movlw     6
        movwf     A
        clrf      A+1

; A = A + 4B

        movf      B,W
        addwf     A,F
        movf      B+1,W
        addwfc    A+1,F

        movf      B,W
        addwf     A,F
        movf      B+1,W
        addwfc    A+1,F

        movf      B,W
        addwf     A,F
        movf      B+1,W
        addwfc    A+1,F

        movf      B,W
        addwf     A,F
        movf      B+1,W
        addwfc    A+1,F

;A = A + 5C

        movf      C,W
        addwf     A,F
        movf      C+1,W
        addwfc    A+1,F

        movf      C,W
        addwf     A,F
        movf      C+1,W
        addwfc    A+1,F

        movf      C,W
        addwf     A,F
        movf      C+1,W
        addwfc    A+1,F

        movf      C,W
        addwf     A,F
        movf      C+1,W
        addwfc    A+1,F

        movf      C,W
        addwf     A,F
        movf      C+1,W
        addwfc    A+1,F
```

4) Convert the following C code to assembler (bar chart - if-statements)

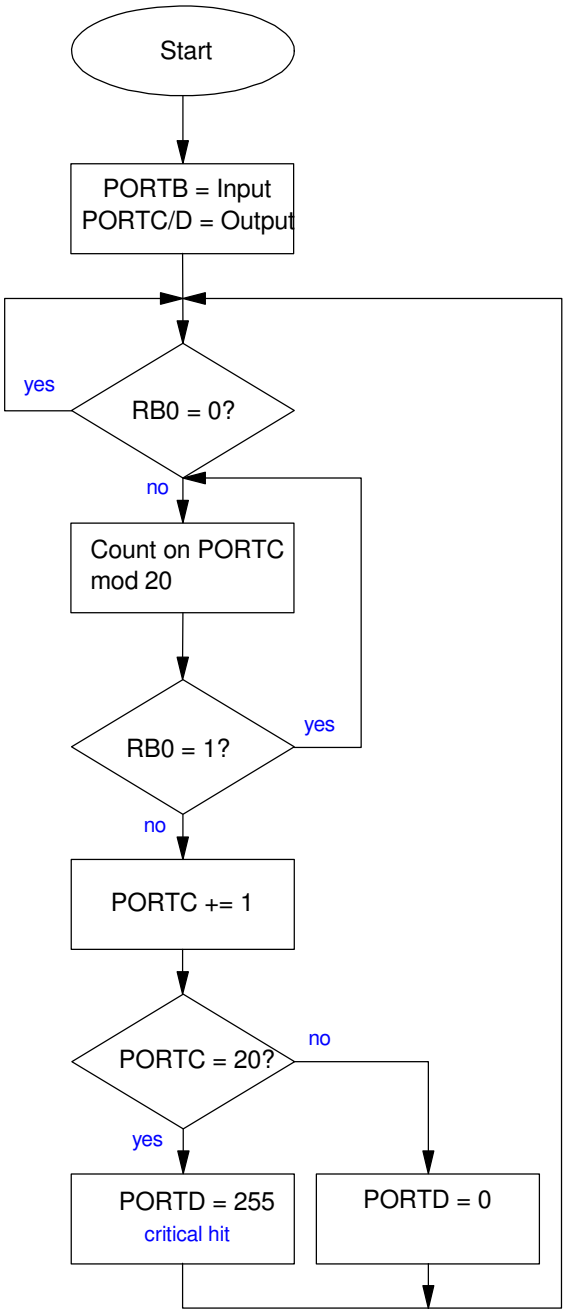
```
;unsigned char A, B;

A       equ       0
B       equ       1

;if(A == 0) B = 0;

        movlw     0
        cpfseq    A
        goto      L1
        clrf      B

; if(A == 1) B = 1;
L1:
        movlw     1
        cpfseq    A
        goto      L2
        movlw     1
        movwf     B

; if(A == 2) B = 3;
L2:
        movlw     2
        cpfseq    A
        goto      L3
        movlw     3
        movwf     B

; if(A == 3) B = 7;
L3:
        movlw     3
        cpfseq    A
        goto      L4
        movlw     7
        movwf     B
L4:
        nop
```

5) The flow chart below turns your PIC into a Dungeon's and Dragon's 20-sided die:

- Press & release RB0 to roll the die
- If you roll a 20, light up PORTD (critical hit)

Write the corresponding assembler code.

```
        movlw     0xFF
        movwf     TRISB
        clrf      TRISC
        clrf      TRISD

        movlw     0x0F
        movwf     ADCON1

L1:
        btfss     PORTB,0
        goto      L1

L2:
        movlw     19
        cpfseq    PORTC
        goto      L3
        clrf      PORTC
        goto      L4
L3:     incf      PORTC,F
L4:     btfsc     PORTB,1
        goto      L2

L5:
        incf      PORTC,F

        movlw     20
        cpfseq    PORTC
        goto      L6
        movlw     0xFF
        movwf     PORTD
        goto      L1
L6:
        clrf      PORTD
        goto      L1
```



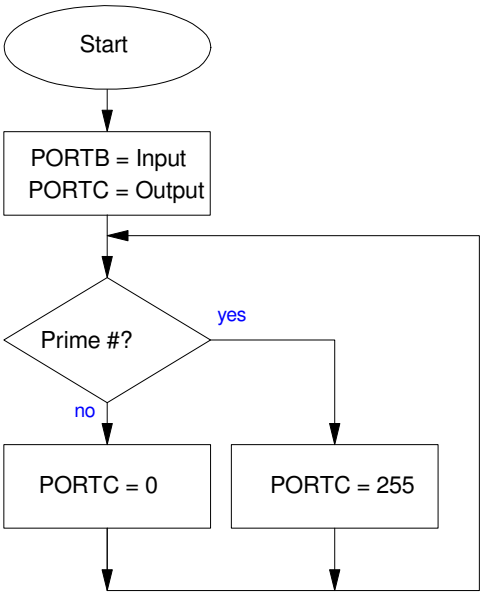
D&D 20-sided die

6) The flow chart below turns your PIC into a prime-number detector

- If the buttons on PORTB are a 4-bit prime number {1, 2, 3, 5, 7, 9, 11, 13), PORTC lights up
- Otherwise, PORTC = 0

Write the corresponding assembly code

```
        movlw     0xFF
        movwf     TRISB
        clrf      TRISC

        movlw     0x0F
        movwf     ADCON1
L1:
        movlw     1
        cpfseq    PORTB
        goto      L2
        goto      Prime
L2:
        movlw     2
        cpfseq    PORTB
        goto      L3
        goto      Prime
L3:
        movlw     3
        cpfseq    PORTB
        goto      L4
        goto      Prime
L4:
        movlw     5
        cpfseq    PORTB
        goto      L5
        goto      Prime
L5:
        movlw     7
        cpfseq    PORTB
        goto      L6
        goto      Prime
L6:
        movlw     11
        cpfseq    PORTB
        goto      L7
        goto      Prime
L7:
        movlw     13
        cpfseq    PORTB
        goto      NotPrime
Prime:
        movlw     0xFF
        movwf     PORTC
        goto      L1
NotPrime:
        clrf      PORTC
        goto      L1
```



Problem #6: 4-Bit Prime #

Binary Inputs (hardware)

Assume a thermistor has a resistance-temperature relationship of

$$R = 1000 \exp\left(\frac{3905}{T+273} - \frac{3905}{298}\right)\Omega$$

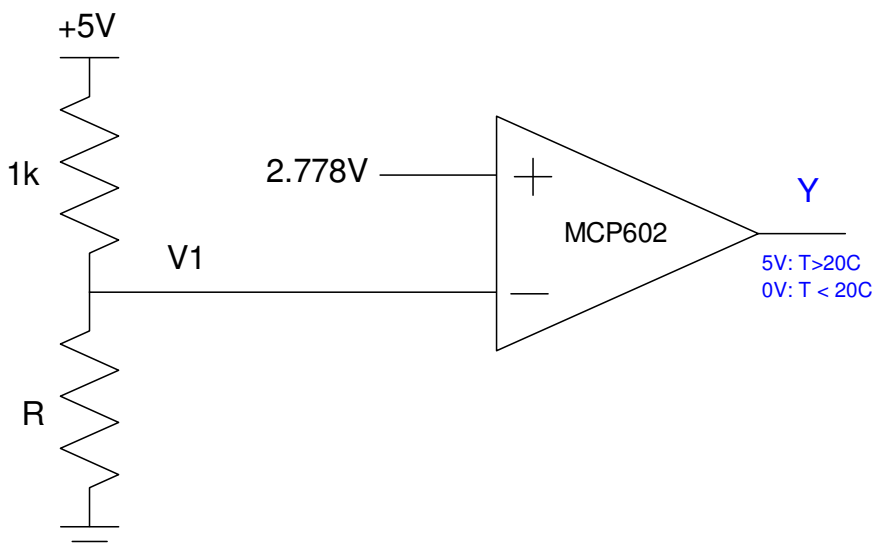
7) Design a circuit which outputs

- 0V when T < 20C
- 5V when T > 20C

At 20C, R = 1250 Ohms

Assuming a 1k resistor and a voltage divider,

V1 = 2.778V



8) Design a circuit which outputs

- 0V when T < 20C
- 5V when T > 25C

Assume a 1k resistor and a voltage divider

20C (Y=0V)

- R = 1250 Ohms
- V1 = 2.778V

25C (Y=5V)

- R = 1000 Ohms
- V1 = 2.500V

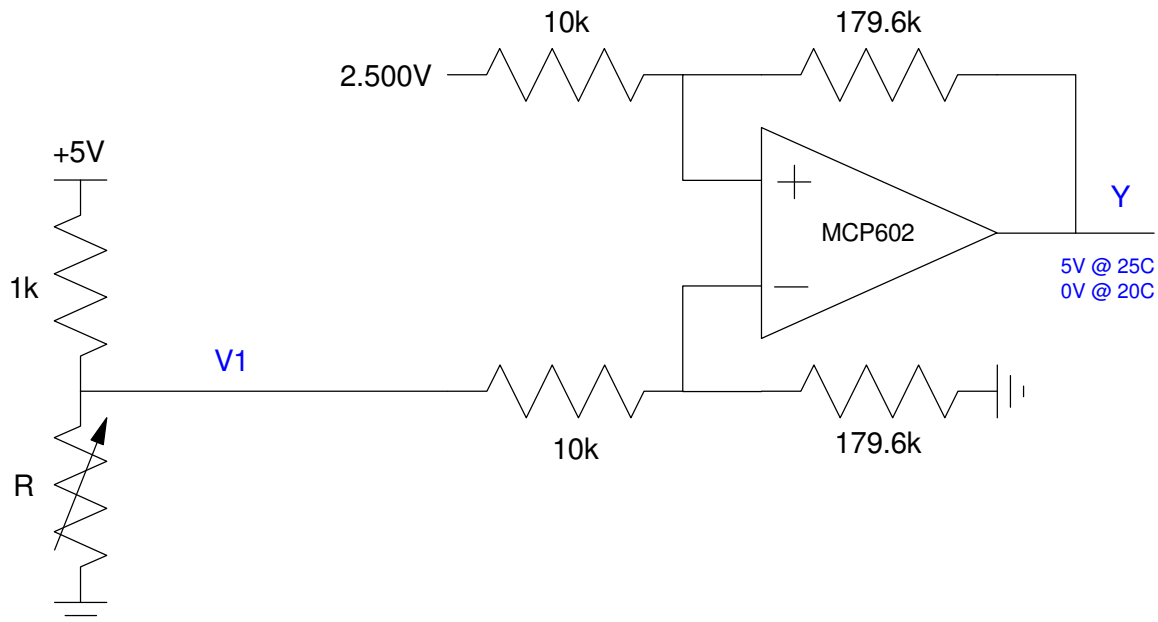
When V1 goes down, Y goes up. Connect to the minus input.

Y is set when V1 = 2.500V. Make the offset 2.500V.

The gain needed is

$$gain = \left(\frac{\text{change in output}}{\text{change in input}}\right) = \left(\frac{5V-0V}{2.778V-2.500V}\right) = 17.96$$

Make the resistors 17.96 : 1



Assume three momentary switches are used: {A, B, C}. These switches are

- open when not pressed
- shorted (0 ohms) when pressed

9) Design a circuit which outputs the function Y = AB + C

- 5V when (A and B) is pressed,
- 5V when C is pressed, and
- 0V otherwise.

One solution is to use switches in series and parallel

- AND is series
- OR is parallel