# ECE 376 - Homework #2

Assembler & Flow Charts - Due Monday, January 22nd

## Assembler Programming

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

| Command | W | A | B |
|---|---|---|---|
| ; Start | 12 | 9 | 3 |
| addwf A,W | **21** | **9** | **3** |
| addwf B,F | **21** | **9** | **24** |
| iorwf A,W | **29** | **9** | **24** |
| andwf B,F | **29** | **9** | **24** |
| movlw 6 | **6** | **9** | **24** |
| subwf A,F | **6** | **3** | **24** |

Note: 21 or 9:

```
21 = 0001 0101
 9 = 0000 1001
----------------
29 = 0001 1101
```

29 and 24

```
29 = 0001 1101
24 = 0001 1000
------------------
24 = 0001 1000
```

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

*note: There are multiple solutions*

*Option #1: 7 instructions*

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

A    equ 0
B    equ 1
C    equ 2

; A = 2*B + 3*C + 4;

     movf    B,W
     addwf   B,W

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

     addlw   4

     movwf   A
```

*Option #2: Using the MUL command*

- 9 instructions

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

A    equ 0
B    equ 1
C    equ 2
X   equ 3

; A = 2*B + 3*C + 4;

    movlw    2
     mulwf B
     movff   PRODL, A

    movlw    3
     mulwf   C
     movf    PRODL,W
     addwf   A,F

     movlw   4
     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 = 2*B + 3*C + 4;

    movff   B,A
    movff   B+1,A+1

    movf    B,W
    addwf   A,F
    movf    B+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

    movlw   4
    addwf   A,F
    movlw   0
    addwfc  A+1,F
```

*note: With 16 bit operations, you need to do operations on the low byte then the high byte*
*16-bit operations are a lot harder than 8-bit operations with an 8-bit processor*

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

```
; unsigned char A, B;
A    equ     0
B    equ     1

;A = A & 0x07;
    movlw   0x07
    andwf   A,F

; if(A == 0) B = B + 1;
    movlw   0
    cpfseq  A
    goto    L1
    incf    B,F

; if(A == 1) B = B + 3;
L1:
    movlw   1
    cpfseq  A
    goto    L2
    movlw   3
    addwf   B,F

; if(A == 2) B = B + 5;
L2:
    movlw   2
    cpfseq  A
    goto    L3
    movlw   5
    addwf   B,F

; if(A == 3) B = B + 7;
L3:
    movlw   3
    cpfseq  A
    goto    L4
    movlw   7
    addwf   B,F

L4:
    nop
```

*note: With this processor, if-statements are usually implemented by*

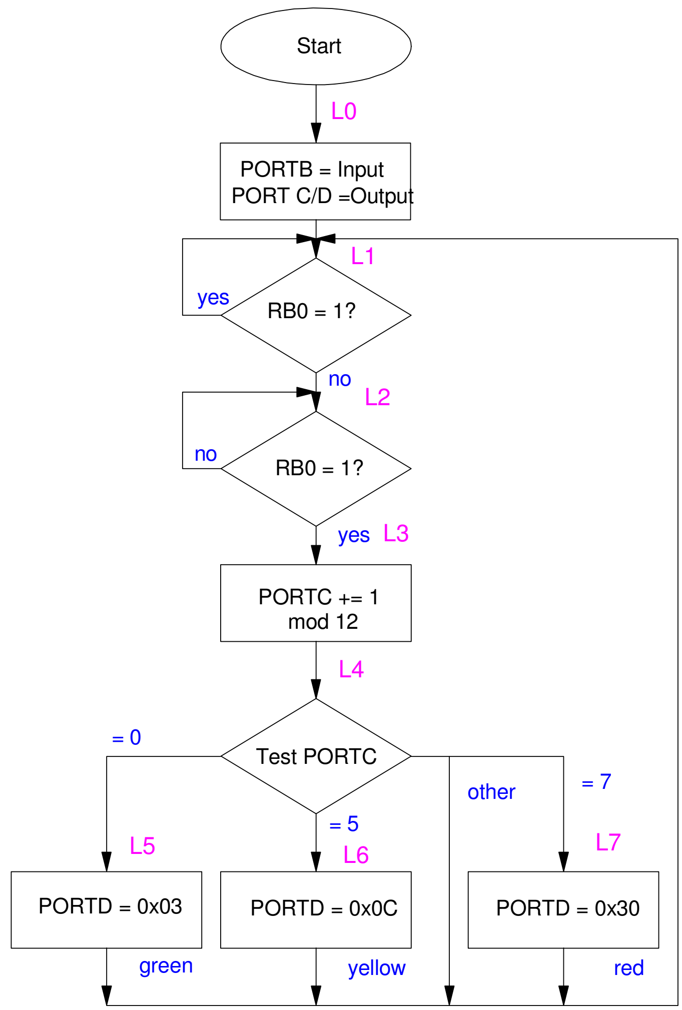
- *set up the cpfxxx command (set up W)*
- *execute the cpfxxx command*
- *then follow that command with a pair of goto-statements*
- *You can eliminate one of the goto statements with the code from one of the branches*

5) The flow chart on the left is for turning your PIC into a stoplight

- Every second press RB0 (keeps track of timing)
- For five counts, the stoplight is green (PORTB = 0x03)
- For the next two counts, the stoplight is yellow (PORTB = 0x0C)
- For the last five counts, the stoplight is red (PORTB = 0x30)
- The process then repeats every 12 button presses.

Write the corresponding assembly code

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

L1:

    btfsc   PORTB,0
    goto    L1

L2:
    btfss   PORTB,0
    goto    L2

L3:
    incf    PORTC,F
    movlw   11
    cpfsgt  PORTC
    goto    L4
    clrf    PORTC

L4:
    movlw   0
    cpfseq  PORTC
    goto    L4a
    goto    L5
L4a:
    movlw   5
    cpfseq  PORTC
    goto    L4b
    goto    L6
L4b:
    movlw   7
    cpfseq  PORTC
    goto    L1
    goto    L7
L5:
    movlw   0x03
    movwf   PORTD
    goto    L1
L6:
    movlw   0x0C
    movwf   PORTD
    goto    L1
L7:
    movlw   0x30
    movwf   PORTD
    goto    L1
```



6) The flow chart to the right has a PIC receive data using SPI protocol:

- The PIC waits for a rising edge on RB0 (CLK)
- Once detected, it checks Chip Select (RB1)
- If CS=0, 4then PORTC is shifted left with
- RC0 being determined by the DATA line (RB2)

Write the corresponding assembly code

```
    movlw   0xFF
    movwf   TRISB
    clrf    TRISC

    clrf    PORTC

L1:
    btfsc   PORTB,0
    goto    L1

L2:
    btfss   PORTB,0
    goto    L2
L3:
    btfsc   PORTB,1
    goto    L1
L4:
    rlcf    PORTC,F
L5:
    btfsc   PORTB,2
    goto    L6
    goto    L7
L6:
    bsf     PORTC,0
    goto    L1
L7:
    bcf     PORTC,0
    goto    L1
```