16.317: Microprocessor Systems Design I
Fall 2013
Exam 3 Solution

1. (20 points, 5 points per part) ***Multiple choice***

For each of the multiple choice questions below, clearly indicate your response by circling or underlining the single choice you think best answers the question.

a. Given the PIC assembly code below, what is the final value of the variable x?

```
    clrf      x
    movlw     0x0E
    movwf     COUNT
L:  incf      x
    decfsz    COUNT, F
    goto      L
```

i. 0

ii. 1

iii. ***14 (0xE)***

iv. 15 (0xF)

v. 241 (0xF1)

b. Under what conditions will the following code jump to the label L1?

```
    btfss     STATUS, C
    goto      END
    movlw     0x11
    subwf     x, F
    btfsc     STATUS, Z
    goto      L1
END:
```

i. C = 0

ii. C = 0 and x = 0x11

iii. C = 1

iv. ***C = 1 and x = 0x11***

v. C = 1 and x ≠ 0x11

1 (continued)

c. Which of the following statements about shift and rotate operations using 16-bit values (and therefore multiple registers) on the PIC16F684 are **<u>true</u>**? Note that these statements assume the C bit is modified before performing any rotate operations to ensure the correct bit is shifted in.

   a. In a 16-bit right shift, the most significant byte should be shifted first to ensure that the correct data is transferred between bytes.

   b. In a 16-bit rotate operation that does not include the carry, the initial value of the C bit should always be 1.

   c. Every 16-bit shift or rotate operation requires at least three PIC instructions, even if the shift amount is 1.

   d. In a 16-bit arithmetic right shift, the initial value of the C bit depends on the most significant bit (bit 15) of the value being shifted.

   i. Only A

   ii. B and C

   iii. A and C

   iv. B and D

   v. ***<u>A and D</u>***

d. Circle one (or more) of the choices below that you feel best "answers" this "question."

   i. "Thanks for the free points."

   ii. "I don't REALLY have to answer the last three questions, do I?"

   iii. "It's about time we have a test that doesn't start at 8:00 AM."

   iv. None of the above.

<u>Any of the above are "correct."</u>

2. (16 points) ***Reading PIC assembly***

Show the result of each PIC 16F684 instruction in the sequences below. Be sure to show not only the state of updated registers, but also the carry (C) and zero (Z) bits.

a.
```
cblock 0x20
    x
endc
```

movlw	0x10	**W = $\underline{\text{0x10}}$**
sublw	0x31	**W = 0x31 – W = 0x31 – 0x10 = $\underline{\text{0x21}}$**
clrf	x	**x = $\underline{\text{0x00}}$**
decf	x, F	**x = x – 1 = 0x00 – 1 = $\underline{\text{0xFF}}$**
xorwf	x, F	**x = x XOR W = 0xFF XOR 0x21 = $\underline{\text{0xDE}}$**
swapf	x, W	**W = value in x with nibbles swapped = $\underline{\text{0xED}}$**
btfsc	x, 7	**Test bit 7 of x and skip next instruction if bit is 0 $\rightarrow$ x = 0xDE = $\underline{1}101\ 1110_2$ $\rightarrow$ since bit 7 = 1, $\underline{\text{do not skip}}$**
bcf	x, 3	**Clear bit 3 of x $\rightarrow$ x = $1101\ \underline{1}110_2$ before clear x = $1101\ \underline{0}110_2$ after clear = $\underline{\text{0xD6}}$**

3. (24 points) ***PIC programming sequences***
Complete each program below by writing the appropriate PIC instruction into each of the blank spaces. The purpose of each instruction is described in a comment to the right of the blank.

a. (12 points)
This short program uses an 8-bit variable, COUNT, to count from 0 to 100.

```
        clrf        COUNT       ; Set COUNT = 0


L:      incf        COUNT, F    ; Increment COUNT


        movlw       100         ; W = 100


        subwf       COUNT, W    ; Use two instructions
                                ;   to compare current
                                ;   COUNT value to 100
        btfss       STATUS, Z   ; If COUNT is 100, skip
                                ;   goto instruction and
                                ;   exit loop

        goto    L               ; Return to start of loop
```

3 (continued)
b.   (12 points)
This program snippet calls a function, SetBits, which reads the current state of PORTA and isolates the lowest bits to produce a value between 0 and 3. Based on that value, SetBits returns a value that can be used to set 0, 1, 2, or 3 bits in register PORTC without changing any other bits. The short snippet below shows how the function is called; you have to determine how the return value is used.

```
        ; Code below is short snippet from main program

        call     SetBits          ; Call SetBits function

        iorwf  PORTC, F           ; Use return value from SetBits
                                  ;   to set appropriate bits
                                  ;   in PORTC

        . . .                     ; Remainder of main program
                                  ;   (not shown)

SetBits:
        movf   PORTA, W           ; Read state of PORTA into W


        andlw 0x03                ; Isolate lowest bits of PORTA to
                                  ;   get a value between 0 and 3

        addwf PCL, F              ; Add that value to PCL
                                  ;   to pick instruction below

        retlw   b'00000000'       ; Return value used to set no
                                  ;   bits in PORTC


        retlw   b'00000001'       ; Return value that will set
                                  ;   lowest bit in PORTC


        retlw   b'00000011'       ; Return value used to set
                                  ;   lowest two bits in PORTC


        retlw b'00000111'         ; Return value used to set
                                  ;   lowest three bits in PORTC
```

4. (40 points, 20 points per part) ***<u>Complex operations</u>***

For each of the following complex operations, write a sequence of PIC 16F684 instructions that performs an equivalent operation. **<u>CHOOSE ANY TWO OF THE THREE PARTS</u>** and fill in the space provided with appropriate code. **<u>You may complete all three parts for up to 10 points of extra credit, but must clearly indicate which part is the extra one—I will assume it is part (c) if you mark none of them.</u>**

Assume that 8-bit variables "TEMP" and "COUNT" have been defined for cases where you may need extra variables.

Finally, please note that <u>you are not required to write comments describing each instruction</u>. You are certainly welcome to do so if you feel it will make your solution clearer to the instructor.

<u>In all cases, multiple solutions may be acceptable.</u>

a. You are given three 16-bit values X, Y, and Z. You can access individual bytes within each value—"X" contains bytes XH and XL (XL is the least-significant byte), "Y" contains bytes YH and YL, and "Z" contains bytes ZH and ZL. Perform the following 16-bit addition:

X = Y + Z *(Add Y and Z and store the result in X, without changing Y or Z)*

<u>Solution:</u>

```
movf    YL, W       ; Copy YL to XL
movwf   XL
movf    YH, W       ; Copy YH to XH
movwf   XH
movf    ZL, W       ; Add low bytes
addwf   XL, F
btfsc   STATUS, C   ; Account for carry
incf    XH, F
movf    ZH, W       ; Add high bytes
addwf   XH, F
```

4 (continued)

b. Given two 8-bit variables, X and B, copy the value of bit position B within variable X into the carry flag. For example:

- If X = 0x03 and B = 0x00, set C to the value of bit 0 within X.
  - Since X = 0x03 = 0000 001<u>1</u>$_2$, C = 1
- If X = 0xC2 and B = 0x04, set C to the value of bit 4 within X.
  - Since X = 0xC2 = 110<u>0</u> 0011$_2$, C = 0

Note that:

- This operation is very similar to the bit test (BT) instruction in the x86 architecture.
- Since B is not a constant, you cannot use the value of B directly in any of the PIC bit test instructions (for example, `btfsc X, B` is <u>not</u> a valid instruction).
- Your code should not modify either X or B.

<u>Solution</u>

```
        movlw   0x01        ; TEMP will hold bit mask used
        movwf   TEMP        ;       to isolate bit B within X
        movf    B, W        ; Copy B to W—determines # of times to shift temp
L:      btfsc   STATUS, Z   ; Once W hits 0, end loop—bit mask is set
        goto    L2          ; Must test this first for case where B == 0
        rlf     TEMP, F     ; TEMP will eventually be 1 << B
        addlw   -1          ; Decrement W
        goto    L
L2:     bcf     STATUS, C   ; Clear C
        movf    TEMP, W     ; AND temp with X to mask out all but bit B
        andwf   X, W
        btfss   STATUS, Z   ; If result is non-zero, set C bit; otherwise, leave as 0
        bsf     STATUS, C
```

4 (continued)

c. Given an 8-bit variable, Y, perform the multiplication:

Y = Y * 10

Hint: Note that multiplication by a constant amount can be broken into a series of shift and add operations. For example:

- X * 2 can be implemented by shifting X to the left by 1 (X << 1)
- X * 5 can be implemented as (X * 4) + X = (X << 2) + X

Solution: *Recognize that Y * 10 = (Y * 8) + (Y * 2) = (Y << 3) + Y + Y*

```
        movf      Y, W         ; Copy original value of Y into TEMP
        movwf     TEMP
        movlw     3            ; Set W = 3—use as loop counter for left shift
L:      bcf       STATUS, C    ; C must be 0 for left shift
        rlf       Y, F
        addlw     -1           ; Decrement loop counter
        btfss     STATUS, Z    ;   and exit once it reaches 0
        goto      L
        movf      TEMP, W      ; W = TEMP = original value of Y
                               ; At this point, Y = (original Y) << 3 = (original Y) * 8
        addwf     Y, F         ; Y = (original Y) * 9
        addwf     Y, F         ; Y = (original Y) * 10
```