

Part I: (20 pts)

a. Give the machine code in HEX for the instruction *bcf 0x17A, 3*

```
bcf 17A,3    1001 bbba ffff ffff,    bbb = 011 (3), ffff ffff = 0x7A
              1001 0111 0111 1010 → 0x977A
```

b. Circle any statement below that is true about the PIC18 Program memory. More than one statement can be circled.

1.  Is NON-VOLATILE
2.  Is VOLATILE.
3.  Address supports a maximum of 4K bytes.
4.  Address supports a maximum of 2M bytes.
5.  Contains instructions.
6.  Contains the Special Function Registers (SFRs).

c. The machine code 0x1BAF represents instruction? (use 'w' or 'f' for the destination, and 'ACCESS' or BANKED to represent the value of the *a* bit). You need to give the complete instruction including all of the operands.

```
0x1BAF → 0001 1011 1010 1111 , first SIX bits indicate the XORWF instruction
xorwf    1001 10da ffff ffff, → xorwf 0xAF,f, BANKED (a =1)
```

d. How many instruction cycles does it take to execute the following instructions? How many clock cycles? With a 20 MHz clock, how long does it take to execute the following instructions? (give the answer in **nanoseconds**). For reference, 1 MHz has a period = 1 us = 1000 ns.

```
incf 0x3A0,f
goto 0x0130
```

|                                  | inst cycles | clk cycles |
|----------------------------------|-------------|------------|
| <b>incf</b>                      | 1           | 4          |
| <b>goto</b>                      | 2           | 8          |
|                                  |             | -----      |
|                                  |             | 12 clks    |
| 20 MHz clock has 1/20 period of  |             |            |
| 1 MHz clock, so 20 MHz period is |             |            |
| 1000ns/20 = 50 ns. Total time is |             |            |
| 12 clks * 50 ns = 600 ns.        |             |            |

e. Assume that you could add another instruction to the PIC18 that had the format:

```
addff srcA, srcB, dest ; dest ← (srcA) + (srcB)
```

where dst, srcA, srcB are all locations in data memory. What is the minimum number of instruction WORDS (1 word = 2 bytes = 16 bits) would it take to encode this instruction? You MUST defend your answer or you will get no credit.

**Using the MOVFF instruction as a guide, we can encode this instruction as:**

```
addwf    ???? ffff ffff ffff (srcA)
          1111 ffff ffff ffff (srcB)
          1111 ffff ffff ffff (dest)
```

**Will take three instruction words. The first four bits ???? would be the opcode, the last 12 bits in each would be the data memory address. The Bank Select Register would not be used, the operands can be in any bank.**

| Location | Contents |
|----------|----------|
| 0x061    | 0x02     |
| 0x062    | 0x30     |
| 0x063    | 0x91     |
| 0x064    | 0x01     |

Assume the W register has the value 0xC9 in it, and that initial values of C, Z are both '0'.

Part II. (35 pts) Assume the above memory contents, W register value, initial C,Z values at the START of each instruction.

a. bsf 0x063,3

```

          7654 3210
[0x63] = 0x91= 1001 0001 (set bit 3)
new value [0x63]=1001 1001 = 0x99

```

Circle one: W dest. Reg. file dest.

New value (hex) \_0x99\_ C\_flag : \_0\_ , Z flag: \_0\_

b. subwf 0x061, f

```

[0x61] = 0x02
W      = - 0xC9
new [0x61] = 0x39
C = 0 because of borrow out of MSB

```

Circle one: W dest. Reg. file dest.

New value (hex) \_0x39\_ C\_flag : \_0\_ , Z flag: \_0\_

c. andwf 0x062, f

```

[0x62] = 0x30 = 0011 0000 AND operation
W      = 0xC9 = & 1100 1001
new f  =          0000 0000 = 0x00

```

Circle one: W dest. Reg. file dest.

New value (hex) \_0x00\_ C\_flag : \_0\_ , Z flag: \_1\_

d. rrcf 0x064, w

Rotate right thru carry W ← contents of (0x064)  
0x01 = old C flag (0) → 0000 0001 → new C flag  
after right shift, new value is 0x00, C flag = 1,  
Z = 1 since result is zero and this affects the Z flag.

Circle one: W dest. Reg. file dest.

New value (hex) \_0x00\_ C\_flag : \_1\_ , Z flag: \_1\_

e. movlw 0x63

```

move literal 0x63 to W reg
W ← 0x63      W = 0x63

```

Circle one: W dest. Reg. file dest.

New value (hex) \_0x63\_ C\_flag : \_0\_ , Z flag: \_0\_

(45 pts) PART III. Convert the following C code fragments to PIC18 assembly.

UNLESS otherwise stated in a particular problem, assume all variables are in locations 0x000 to 0x07F.

If you use a temporary memory location, use temp and assume it is in bank 0. When writing code, you **must use** symbolic names for variable names, register names, and bit names for (i.e, use: bsf STATUS, C instead of bsf 0xFD8, 0x0). You do not have to show the CBLOCK declaration for variables.

Hint: A common mistake in these problems is to write code that modifies variables to the right of the '=' sign (i.e, for 'a = b - c;' the code you write somehow modifies b, or c, as well as a). This is incorrect; make sure that your code only modifies variables to the left of the '=' sign.

Also, recall that 'k++' is the same as 'k=k+1;', 'j--' is the same as 'j=j-1', that "i==j" is true if i is equal to j, that "i!=j" is true if i is not equal to j, "<<" is a left shift, ">>" is a right shift, '|' is bitwise logical OR, '&' is a bitwise logic AND, '^' is a bitwise logical XOR.

unsigned char i,j,k,p,q,r,s,t;

- a. (7 pts)  
j = p + q + 2;

```
movf  p,w      ;w = p
addwf q,w      ;w = q+p
addlw 2        ;w = w + 2
movwf j        ;j = w
```

- b. (9 pts)  
if ( (q == 5) && (j != 0) {  
 //if-body – just write a placeholder here  
} else {  
 //else-body – just write a placeholder here  
}

```
Basic structure:
    test q == 5
    branch false to else
    test j != 0
    branch false to else
if_body
    ...instr..
    ...instr..
    bra end_if
else_body
    ...instr..
    ...instr..
end_if
```

```
Assembly:
    movlw 5      ;w = 5
    subwf q,w    ;w = q - 5
    bnz  else_body
    movf j,f     ;test j
    bz   else_body
if_body
    ...instr..
    ...instr..
    bra end_if
else_body
    ...instr..
    ...instr..
end_if
```

- c. (6 pts) Write the following in assembly language  
 $i = (k | 0xF0) \ll 1;$

```

movlw 0xF0      ;w = 0xF0
iorwf k,w       ;w = k | w
bcf    STATUS,C ;clear carry for shift
rlcf   WREG,w   ;w = w << 1
movwf  i

```

- d. (10 pts)

```

while ( (p > q) || (r == 0) ) {
//loop-body – just write a placeholder here
}

```

Basic structure:

```

loop_top:
    test p>q
    branch true to loop_body
    test r==0
    branch false to end_loop
loop_body:
    ...instr..
    ...instr..
    bra loop_top
loop_end:

```

Assembly:

```

movf  p,w      ;w = q
subwf q,w      ;w = q - p
bnc   loop_body
movf  r,f      ; test r
bnz   loop_end
loop_body:
    ...instr..
    ...instr..
    bra loop_top
loop_end:

```

|         | Operation | True          | False              |
|---------|-----------|---------------|--------------------|
| $p > q$ | $q - p$   | Borrow<br>C=0 | No borrow<br>C = 1 |

- e. (6 pts) Assume that  $r$  is in bank 1,  $s$  is in bank 2, and  $t$  is in bank 3. Implement the following code in assembly language:

```
t = r - s ;
```

```
movlb 2      ;select bank 2
movf  s,w    ;w = s
movlb 1      ;select bank 1
subwf r,w    ;w = r- w
movlb 3      ;select bank 3
movwf t      ;t = w
```

- f. (7 pts) Write a PIC18 instruction sequence that does the following.

```
do {
    //loop body, place holder
} while (p >= q)
```

```
Basic structure:
loop_top:
    ...instr..
    ...instr..
    test p>=q
    branch true to loop_top
loop_end:
```

```
Assembly:
loop_top:
    ...instr..
    ...instr..
    movf q,w      ;w = q
    subwf p,w     ;w = p-q
    bc loop_top
loop_end:
```

|            | Operation | True             | False         |
|------------|-----------|------------------|---------------|
| $p \geq q$ | $p - q$   | no borrow<br>C=1 | borrow<br>C=0 |