Chapter 5. Problems

"All programming problems should include design pseudo code either as a separate design document on embedded comments in the code."

1S. Prior to execution of the following code segment, PC was "0x0120", WREG was "0x02" and register file 0x81 was "0x35".

```
ADDWF     0x81,0,0
MOVWF     0x82
ADDWF     0x81,1,0
RRNCF     0x81
GOTO      0x029A
```

After the execution of the above the code segment, determine:

a) The content of register files "0x81", "0x82" and WREG.
b) The content of PC.

Solution

a)

Register 0x81 ← 0x36
Register 0x82 ← 0x37
WREG ← 0x37

b)

PC ← 0x029A

1U. Prior to execution of the following code segment, PC was "0x0220", WREG was "0xA2" and register file 0x83 was "0x45".

```
ADDWF     0x83,0,0
MOVWF     0x82
ADDWF     0x83,1,0
RRNCF     0x83
GOTO      0x029A
```

After the execution of the above the code segment, determine:

a) The content of register files "0x83", "0x82" and WREG.
b) The content of PC.

Solution

2S. Write an assembly code segment that implements the following C code, where the value of op1 and op2 are in the register files 0x92 and 0xA0.

```
If (op1 > op2) {
        op1 = 23;
        op2 = 29;
}
```

Solution

```
Op1     equ   0x92
Op2     equ   0xA0
        MOVF      Op1, 0
        CPFSLT    Op2
```

```
        BRA     Next
        MOVLW   23
        MOVWF   Op1
        MOVLW   29
        MOVWF   Op2
Next:
```

2U. Write an assembly code segment that implements the following C code, where the value of op1 and op2 are in the register files 0x81 and 0x82.

```
If (op1 ≤ op2) {
      op1 = op1 + 93;
      op2 = 0;
}
```

Solution

3S. Write an assembly code segment that implements the following C code, where the value of index and count in memory locations 0xC2 and 0xC5.

```
for (index=0 ; index<26; index++){
        Count = Count + 2
}
```

Solution

```
index   equ   0xC2
count   equ  0xC5
        MOVLW   26
        CLRF    index
        CLRF    count
floop:  INCF    0xC2
        INCF    0XC5
        INCF    0XC5
        CPFSEQ  0xC2
        BRA floop
```

3U. Write an assembly code segment that implements the following C code, where the value of index and count in register files 0xA2 and 0xA3.

```
for (index=5 ; index<25; index=index+2){
     Count = Count + 4;
}
```

Solution

4S. Write an assembly code that set the contents of registers 0xA1 through 0xA8 (unique value in each register) such that the logical AND of the registers results in 0x00 and logical OR results in 0xFF.

Solution

```
// there are eight registers with 8-bits each so simply setting only one unique bit
// in each register will satisfy the requirements
MOVLW   0x01
MOVWF   0xA1
MOVFF   0xA1, 0xA2
RLNCF   0xA2
```

```
MOVFF    0xA2, 0xA3
RLNCF    0xA3
MOVFF    0xA3, 0xA4
RLNCF    0xA4
MOVFF    0xA4, 0xA5
RLNCF    0xA5
MOVFF    0xA5, 0xA6
RLNCF    0xA6
MOVFF    0xA6, 0xA7
RLNCF    0xA7
MOVFF    0xA7, 0xA8
RLNCF    0xA8
```

4U. Write an assembly code that set the contents of register files 0xA1 through 0xA8 (unique value in each register) such that the logical XOR of all the registers results in 0x00.

Solution

5S. Write an assembly code that reads the value of register 0xD1, doubles it and saves it in register 0xD2, quadruple of the value and save it in register file 0xD3 and half the read value to register file 0xD4.

Solution

```
MOVF     0xD1,0          // read the value
MOVWF    0xD2
RLNCF    0xD2            // multiply by 2 <shift left>
BCF      0xD2,0          // just to clear carry
MOVFF    0xD2, 0xD3
RLNCF    0xD3            // multiply by 2 <shift left>
BCF      0xD3,0          // just to clear carry

MOVWF    0xD4
RRNCF    0xD4
BCF      0xD4,7          // Divide by 2 <shift right>
```

5U. Write an assembly code that reads the value of register file 0xA8, writes the bits in reverse order in register 0x8A. Upon execution of the code, bit 0 of register 0xA8 will be in bit 7 of register 0x8A and bit 7 of register 0xA8 will be in bit 0 of register 0x8A.

Solution

6S. Write an assembly code that clears (0x00) registers 0x80 through 0x91 and then sets the value of register listed on the right column to 0xFF if the corresponding value in the left column is equal to the value in the register 0xA1. For example if the value in 0xA1 register is 4, only register 0x83 should be set to oxFF.

Content of register 0xA1	Register file to be set to 0xff
1	0x81
2	0x86
3	0x87
4	0x83
8	0x84
9	0x91
All other values	0x80

Solution

```
            ;  Clear Registers -  You can use indirect addressing (Chap6) to set the values in a loop
            CLRF    0x80
            CLRF    0x81
            CLRF    0x82
            CLRF    0x83
            CLRF    0x84
            CLRF    0x85
            CLRF    0x86
            CLRF    0x87
            CLRF    0x88
            CLRF    0x89
            CLRF    0x8A
            CLRF    0x8B
            CLRF    0x8C
            CLRF    0x8D
            CLRF    0x8E
            CLRF    0x8F
            CLRF    0x90
            CLRF    0x91

            ; Setting the values
            MOVLW   1
            CPFSEQ  0xA1
            BRA Cont2
            BRA     L81
Cont2:      MOVLW   2
            CPFSEQ  0xA1
            BRA Cont3
            BRA     L86
Cont3:      MOVLW   3
            CPFSEQ  0xA1
            BRA Cont4
            BRA     L87
Cont4:      MOVLW   4
            CPFSEQ  0xA1
            BRA Cont8
            BRA     L83
Cont8:      MOVLW   8
            CPFSEQ  0xA1
            BRA Cont9
            BRA     L84
Cont9:      MOVLW   9
            CPFSEQ  0xA1
            BRA Other
            BRA     L91

L81:        MOVLW   0xFF
            MOVWF   0x81
            BRA Done
L86:        MOVLW   0xFF
            MOVWF   0x86
            BRA Done
L87:        MOVLW   0xFF
            MOVWF   0x87
            BRA Done
L83:        MOVLW   0xFF
            MOVWF   0x83
            BRA Done
```

```
L84:    MOVLW   0xFF
        MOVWF   0x84
        BRA Done
L91:    MOVLW   0xFF
        MOVWF   0x91
        BRA Done
Other:  MOVLW   0xFF
        MOVWF   0x80
Done:   BRA Done
```

6U. Write an assembly code that clears (0x00) registers 0x81 through 0x8A and then sets the value for each of these registers to sum of the address of its digits if the sum is even. For example location 0x82 will be set to 0xA but content of location 0x83 will not be changed.

Solution

7S. Write an assembly code that sorts the values stored in register files 0x90 through 0x94 from the largest to smallest.

Solution

```
                list p=18F1220
                radix hex

                temp  equ 0x80
                temp1 equ 0x90
                temp2 equ 0x91
                temp3 equ 0x92
                temp4 equ 0x93
                temp5 equ 0x94

                org 0x0000

COMP12:         MOVF temp1,0
                CPFSGT temp2; If temp1 > tempt2, then skip next line
                GOTO COMP13

                ; Swap values
                MOVFF temp1, temp
                MOVFF temp2, temp1
                MOVFF temp, temp2

COMP13:         CPFSGT temp3; If temp1 > temp3, then skip next line
                GOTO COMP14; To skip

                MOVFF temp1, temp
                MOVFF temp3, temp1
                MOVFF temp, temp3

COMP14:         CPFSGT temp4; If temp1 > temp4, then skip next line
                GOTO COMP15

                ; swap values
                MOVFF temp1, temp
                MOVFF temp4, temp1
                MOVFF temp, temp4
```

```
COMP15:     CPFSGT temp5; If temp1 > temp5 skip next line.
            GOTO COMP23

            ; Swap values
            MOVFF temp1, temp
            MOVFF temp5, temp1
            MOVFF temp, temp5

COMP23:     MOVF temp2,0
            CPFSGT temp3; If temp2 > temp3, skip next line
            GOTO COMP24   ; Skipped line

            MOVFF temp2, temp
            MOVFF temp3, temp2
            MOVFF temp, temp3

COMP24:     CPFSGT temp4; If temp2 > temp4, skip a next line
            GOTO COMP25                 ; Skipped line

            MOVFF temp2, temp
            MOVFF temp4, temp2
            MOVFF temp, temp4

COMP25:     CPFSGT temp5; If temp2 > temp5, skip a next line
            GOTO COMP34                 ; skipped line

            ; Swap values
            MOVFF temp2, temp
            MOVFF temp5, temp2
            MOVFF temp, temp5

COMP34:     MOVF temp3,0
            CPFSGT temp4; If temp3 > temp4, skip a next line
            GOTO COMP35         ; skipped line

            ; Swap values
            MOVFF temp3, temp
            MOVFF temp4, temp3
            MOVFF temp, temp4

COMP35:     CPFSGT temp5; If temp3 > temp5, skip a next line
            GOTO COMP45; skipped line

            ; Swap values
            MOVFF temp3, temp
            MOVFF temp5, temp3
            MOVFF temp, temp5

COMP45:     MOVF temp4,0
            CPFSGT temp5; If temp4 > temp5, skip a next line
            GOTO endPro

            ; Swap values
            MOVFF temp4, temp
            MOVFF temp5, temp4
            MOVFF temp, temp5
endPro:

            end
```

7U. Write an assembly code that sorts the values stored in register files 0x90 through 0x94 from the smallest to largest.

Solution

8S. What is the real number represented by the following single precision floating point representation:

Bit 31																															Bit 0
1	1	1	1	0	1	0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Solution

Sign bit is 1 → Negative number
Power = Exponent – Bias = $(11101010)_2 - 127 = 234 - 127 = 107$
Fraction = $2^{-1} + 2^{-3} + 2^{-4} + 2^{-5} = 0.71875$

Real Number = $- 1.71875 \times 2^{107} = - 2.788831 \times 10^{32}$

8U. What is the real number represented by the following single precision floating point representation:

Bit 31																															Bit 0
0	1	1	1	0	1	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Solution

9S. Write the real number 1.375 x 16 in double precision floating point format.

Solution

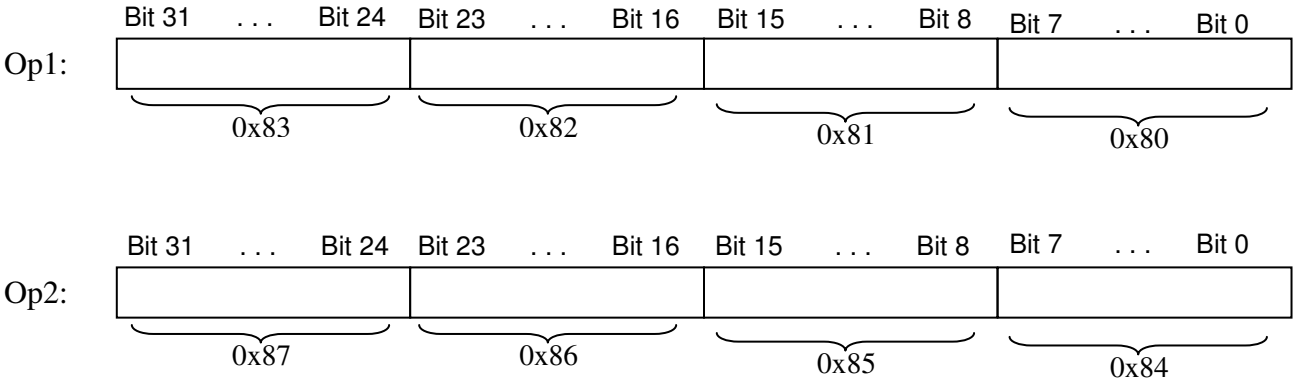
Real Number = 1.375×2^4 → Convert to nearest 1.ffff x 2^{power} with the available precision
Positive → Sign bit is 0
Exponent = Power + Bias

Double Precision Floating Point (64 bits):
[0] [1000 0000 011] [0110 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000]

9U. Write the real number {1.875 / 16} in double precision floating point format.

Solution

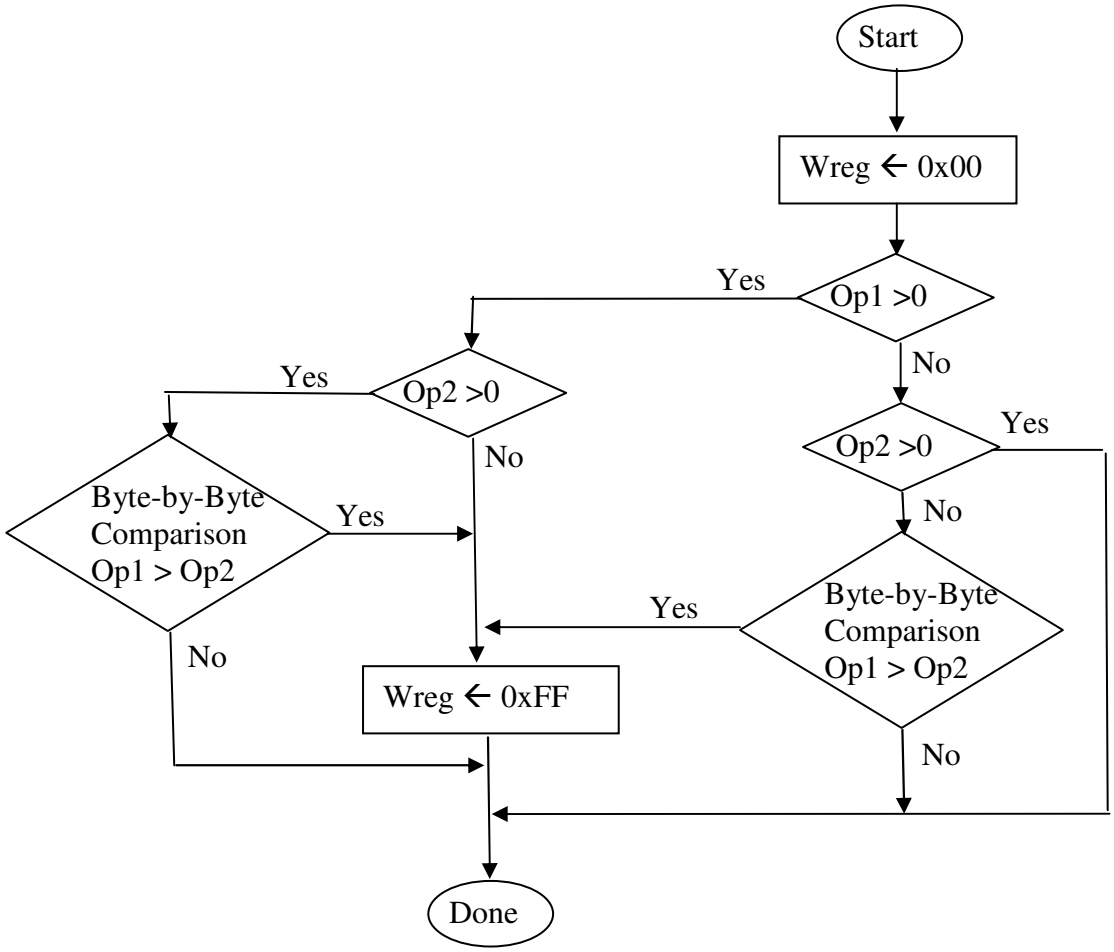
10S. Write an Assembly code that compare two single float numbers where one of number, op1, is stored in register files 0x80 through 0x83 and the second number, op2, is stored in register files 0x84 through 0x87 as shown below:.



If Op1 is larger than Op2 set Wreg to 0xFF.

Solution

Flow Chart




```
; Code to implement the comparison
;

    SIGN        equ     0x90
    OP1GTOP2    equ     0x91
```

```
.               CLRF    WREG
                CLRF    SIGN
                CLRF    OP1GTOP2

.               ; Determine the signs
                BTFSC   0x83,7
                BRA     OP1N
                BRA     OP1P
OP1P:           BTFSC   0x87,7
                BRA     OP1P_OP2N
                BRA     OP1P_OP2P
OP1N:           BTFSC   0x87,7
                BRA     OP1N_OP2N
                BRA     OP1N_OP2P

OP1P_OP2N:      ; OP1 is positive and OP2 is negative
                SETF    WREG
                BRA     DONE

OP1N_OP2P:      ; OP1 is negative and OP2 is positive
                CLRF    WREG
.               BRA     DONE

OP1N_OP2N:      CALL    COMP_BYTES
                MOVLW   1
.               CPFSEQ  OP1GTOP2
.               SETF    WREG
                BRA     DONE

OP1P_OP2P:      CALL    COMP_BYTES
                CLRF    WREG
.               CPFSEQ  OP1GTOP2
.               SETF    WREG
                BRA     DONE

.               ; wait loop
Done:           BRA     Done


COMP_BYTES:     ; Compare bytes and return 1 if OP1 is larger and 0 otherwise
BYTE3:          MOVF
                CPFSEQ  0xA1
                BRA     OP1BN
OP1BP:          MOVLW   0xFF
                BRA     Done
OP1BN:          MOVLW   0
                BRA     Done
OP1NB:          MOVLW   0
                CPFSEQ  0xA1
                BRA     OP1NBN
OP1NBP:         MOVLW   0
                BRA     Done
OP1NB:          MOVLW   0xFF
                RETURN
```

10U. Write an assembly code that compare two double precision float numbers where one is in op1 (registers A0 to A7) and the other is in op2 (registers B0 to B7). If op1 is less than op2 set register 0x80 to 0x00, otherwise set register 0x80 to 0xFF.

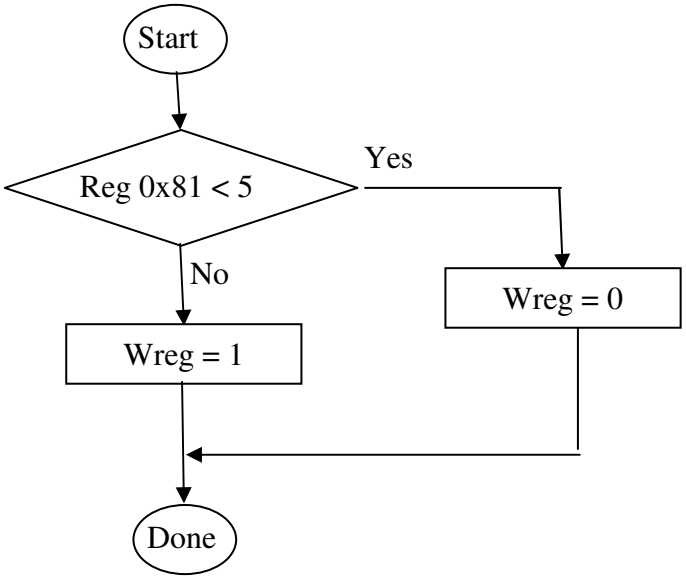
Solution

11S. Use CPFSLT instruction to implement the following C code functionality:

```
//  high is in Wreg and temp is in register file 0x81
high =0
If (temp ≥ 5 ) {
        high = 1;
}
```

Solution

Flow Chart



Code

```
        MOVLW   5
        CPFSLT  0x81
        BR      WR_One
        MOVLW   0
        BRA     Done
WR_One: MOVLW   1
Done:   BR      Done
```

11U. Use CPFSLT instruction to implement the following C code functionality:

```
//  high is in Wreg and temp is in register file 0x84
high =1
If (temp < 15 ) {
        high = 0;
}
```

Solution

12U. Write a PICmicro Assembly code that does the following:

- Accepts temperature in Centigrade ranging from 0 to 127 oC on Port A.
- Output the temperature in Fahrenheit ranging from 1 to 126 on Port B. “0” output represent too low to show temperature and “127” represents too high to show.
  Note: use Tf = (2Tc + 32) for conversion.
- Every hour, the temperature in oC is stored in registers starting at 0xA0 and will keep the data for 24 hours.
  Note: first temp is stored in 0xA0 and the 24th is stored in 0xB7 and 25th is stored in 0xA0.
- All temperature reading in location 0xA0 through 0xB7 will be cleared when INT0 pin transitions from High to low.

Solution