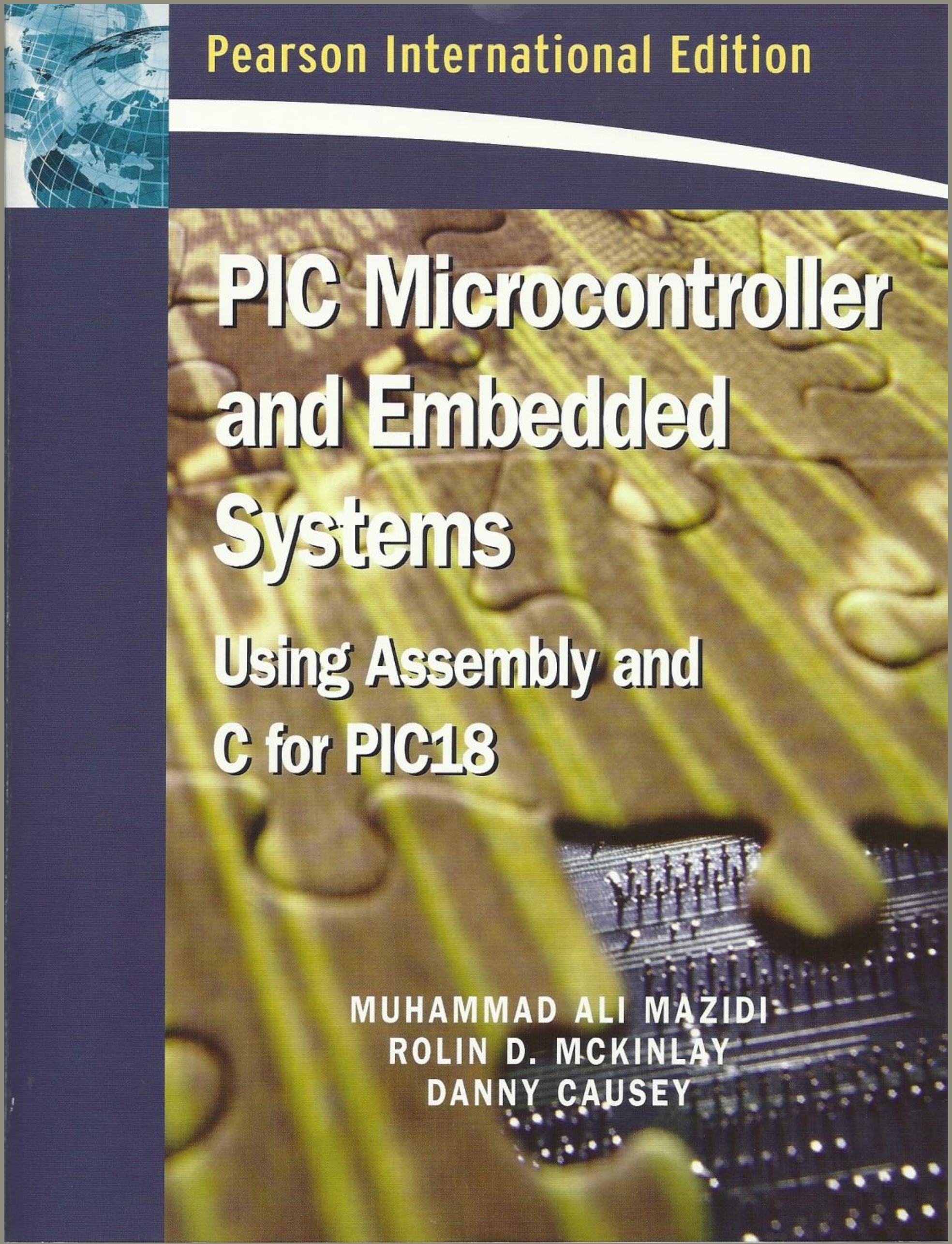
Chapter 5

Arithmetic, Logic Instructions, and Programs



Addition of Unsigned Numbers

- ADDLW K – (WREG = WREG +K.
- ADDWF fileReg, d – to add WREG and individual bytes residing in RAM locations of the file register.

Example 5-1

Show how the flag register is affected by the following instructions.

```
MOVLW 0xF5          ;WREG = F5 hex
ADDLW 0xB           ;WREG = F5 + 0B = 00 and C = 1
```

Solution:

```
  F5H           1111 0101
+ 0BH         + 0000 1011
 100H           0000 0000
```

After the addition, register WREG contains 00 and the flags are as follows:

C = 1 because there is a carry out from D7.

Z = 1 because the result in WREG is zero.

DC = 1 because there is a carry from D3 to D4.

Example 5-2

Assume that file register RAM locations 40–43H have the following hex values. Write a program to find the sum of the values. At the end of the program, location 6 of the file register should contain the low byte and location 7 the high byte of the sum.

```
40 = (7D)
41 = (EB)
42 = (C5)
43 = (5B)
```

Solution:

```
L_Byte EQU  0x6           ;assign RAM location 6 to L_byte of sum
H_Byte EQU  0x7           ;assign RAM location 7 to H_byte of sum

       MOVLW 0            ;clear WREG (WREG = 0)
       MOVWF H_Byte       ;H_Byte = 0
       ADDWF 0x40,W       ;WREG = 0 + 7DH = 7DH , C = 0
       BNC   N_1          ;branch if C = 0
       INCF  H_Byte,F     ;increment (now H_Byte = 0)
N_1    ADDWF 0x41,W       ;WREG = 7D + EB = 68H and C = 1
       BNC   N_2          ;
       INCF  H_Byte,F     ;C = 1, increment (now H_Byte = 1)
N_2    ADDWF 0x42,W       ;WREG = 68 + C5 = 2D and C = 1
       BNC   N_3          ;
       INCF  H_Byte       ;C = 1, increment (now H_Byte = 2)
N_3    ADDWF 0x43,W       ;WREG = 2D + 5B = 88H and C = 0
       BNC   N_4          ;
       INCF  H_Byte,F     ;(H_Byte = 2)
N_4    MOVWF L_Byte       ;now L_Byte = 88h
```

At the end the fileReg location 6 = (8B), and location 7 = (02) because 7D + EB + C5 + 5B + 30 = 28BH. We can use the register indirect addressing mode to do this program much more efficiently. Chapter 6 shows how to do that.

Addition of 16-bit Numbers

- ADDWFC fileReg, d – to add WREG and fileReg with carry.

For example, look at the addition of 3CE7H + 3B8DH, as shown next.

```
          1
        3C  E7
+       3B  8D
        78  74
```

Example 5-3

Write a program to add two 16-bit numbers. The numbers are 3CE7H and 3B8DH. Assume that fileReg location 6 = (8D) and location 7 = (3B). Place the sum in fileReg locations 6 and 7; location 6 should have the lower byte.

Solution:

```
        ;location 6 = (8D)
        ;location 7 = (3B)

        MOVLW  0xE7         ;load the low byte now (WREG = E7H)
        ADDWF  0x6,F        ;F = W + F = E7 + 8D = 74 and CY = 1
        MOVLW  0x3C         ;load the high byte (WREG = 3CH)
        ADDWFC 0x7,F        ;F = W + F + carry, adding the upper byte
                            ;with Carry from lower byte
                            ;F = 3C + 3B + 1 = 78H (all in hex)
```

Notice the use of ADDWF for the lower byte and ADDWFC for the higher byte.

BCD (Binary Coded Decimal) Number System

- Unpacked BCD – the lower 4 bits of the number represent the BCD.
- Packed BCD – a single byte has two BCD numbers in it.

Digit	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

DAW (Decimal Adjust WREG) Instruction

- If the lower nibble is greater than 9, or if DC=1, add 0110 to the lower 4 bits.
- If the upper nibble is greater than 9, or if C=1, add 0110 to the upper 4 bits.

```
MOVLW 0x47   ;WREG = 47H first BCD operand
ADDLW 0x25   ;hex(binary) addition (WREG = 6CH)
DAW          ;adjust for BCD addition (WREG = 72H)
```

Hex	BCD	
57	0101 0111	
+ 77	+ 0111 0111	
CE	1100 1110	
+ 66	+ 0110 0110	
134	1 0011 0100	Note C = 1

Example 5-4

Assume that 5 BCD data items are stored in RAM locations starting at 40H, as shown below. Write a program to find the sum of all the numbers. The result must be in BCD.

```
40 = (71)
41 = (88)
42 = (69)
43 = (97)
```

Solution:

```
L_Byte    EQU    0x6          ;assign RAM loc 6 to L_Byte of sum
H_Byte    EQU    0x7          ;assign RAM loc 7 to H_Byte of sum

          MOVLW    0          ;clear WREG (WREG = 0)
          MOVWF    H_Byte     ;H_Byte = 0
          ADDWF    0x40,W     ;WREG = 0 + 71H = 71H, C = 0
          DAW                 ;WREG = 71H
          BNC      N_1        ;branch if C = 0
          INCF     H_Byte,F   ;
N_1       ADDWF    0x41,W     ;WREG = 71 + 88 = F9H
          DAW                 ;WREG = 59H AND C = 1
          BNC      N_2        ;
          INCF     H_Byte,F   ;C = 1, increment (now H_Byte = 1)
N_2       ADDWF    0x42,W     ;WREG = 59 + 69 = C2 and Carry = 0
          DAW                 ;WREG = 28 and C = 1
          BNC      N_3        ;
          INCF     H_Byte     ;C = 1, increment (now H_Byte = 2)
N_3       ADDWF    0x43,W     ;WREG = 28 + 97 = BFH and C = 0
          DAW                 ;WREG = 25 and C = 1
          BNC      N_4        ;
          INCF     H_Byte,F   ;(now H_Byte = 3)
N_4       MOVWF    L_Byte     ;Now L_Byte = 25H
```

After this code executes, fileReg location 6 = (03), and WREG = 25 because 71 + 88 + 69 + 97 = 325H. We can use the register indirect addressing mode and looping to do this program much more efficiently. Chapter 6 shows how to do that.

Subtraction of Unsigned Numbers

- SUBLW K – (WREG = K – WREG).
- SUBWF fileReg, d –

(Destination = fileReg – WREG).

Example 5-5

Show the steps involved in the following.

```
MOVLW 0x23          ;load 23H into WREG (WREG = 23H)
SUBLW 0x3F          ;WREG = 3F - WREG
```

Solution:

```
   K    = 3F   0011 1111        0011 1111
 - WREG = 23   0010 0011     +  1101 1101  (2's complement)
          1C                  1 0001 1100
                          C = 1, D7 = N = 0 (result is positive)
```

The flags would be set as follows: C = 1, N = 0 (notice that D7 is the negative flag). The programmer must look at the N (or C) flag to determine if the result is positive or negative.

Example 5-6

Write a program to subtract 4C – 6E.

Solution:

```
MYREG EQU 0x20
      MOVLW 0x4C        ;load WREG (WREG = 4CH)
      MOVWF MYREG       ;MYREG = 4CH
      MOVLW 0x6E        ;WREG = 6EH
      SUBWF MYREG,W     ;WREG = MYREG - WREG. 4C - 6E = DE, N = 1
      BNN   NEXT        ;if N = 0 (C = 1), jump to NEXT target
      NEGF  WREG        ;take 2's complement of WREG
NEXT  MOVWF MYREG       ;save the result in MYREG
```

The following are the steps after the SUBWF instruction:

```
  4C     0100 1100                      0100 1100
 -6E     0110 1110    2's comp =        1001 0010
 -22                                    1101 1110
```

After SUBWF, we have N = 1 (or C = 0), and the result is negative, in 2's complement. Then it falls through and NEGF will be executed. The NEGF instruction will take the 2's complement, and we have MYREG = 22H.

Subtraction of Unsigned Numbers

- SUBWFB fileReg, d – (Destination = fileReg – WREG – $\overline{Borrow}$).
- SUBFWB fileReg, d – (Destination = WREG – fileReg – $\overline{Borrow}$).

Example 5-7

Write a program to subtract two 16-bit numbers. The numbers are 2762H – 1296H. Assume fileReg location 6 = (62) and location 7 = (27). Place the difference in fileReg locations 6 and 7; loc 6 should have the lower byte.

Solution:

```
loc 6 = (62)
loc 7 = (27)

MOVLW 0x96             ;load the low byte (WREG = 96H)
SUBWF 0x6,F   ;F = F - W = 62 - 96 = CCH, C = borrow = 0, N = 1
MOVLW 0x12             ;load the high byte (WREG = 12H)
SUBWFB 0x7,F           ;F = F - W - b̄, sub byte with the borrow
                       ;F = 27 - 12 - 1 = 14H
```

After the SUBWF, loc 6 has = 62H – 96H = CCH and the carry flag is set to 0, indicating there is a borrow (notice, N = 1). Because C = 0, when SUBWFB is executed the fileReg location 7 has = 27H – 12H – 1 = 14H. Therefore, we have 2762H – 1296H = 14CCH.

Multiplication of Unsigned Numbers

- MULLW K – After multiplication, the result is in the special function registers PRODH and PRODL.

```
MOVLW  0x25         ;load 25H to WREG (WREG = 25H)
MULLW  0x65         ;25H * 65H = E99 where
                    ;PRODH = 0EH and PRODL = 99H
```

Table 5-1: Unsigned Multiplication Summary (MULLW K)

Multiplication	Byte 1	Byte2	Result
Byte × Byte	WREG	K	PRODH = high byte, PRODL = low byte

Note: Multiplication of operands larger than 8-bit takes some manipulation.

Division of Unsigned Numbers

```
NUM     EQU    0x19        ;set aside fileReg
MYQ     EQU    0x20
MYNMB   EQU    D'95'
MYDEN   EQU    D'10'
        CLRF   MYQ         ;quotient = 0
        MOVLW  MYNMB       ;WREG = 95
        MOVWF  NUM         ;numerator = 95
        MOVLW  MYDEN       ;WREG = denominator = 10
B1      INCF   MYQ,F       ;increment quotient for every 10 subtr
        SUBWF  NUM,F       ;subtract 10 (F = F - W)
        BC     B1          ;keep doing it until C = 0
        DECF   MYQ,F       ;once too many
        ADDWF  NUM,F       ;add 10 back to get remainder
```

Example 5-8

Assume that file register location 0x15 has value FD (hex). Write a program to convert it to decimal. Save the digits in locations 0x22, 0x23, and 0x24, where the least-significant digit is in 0x22

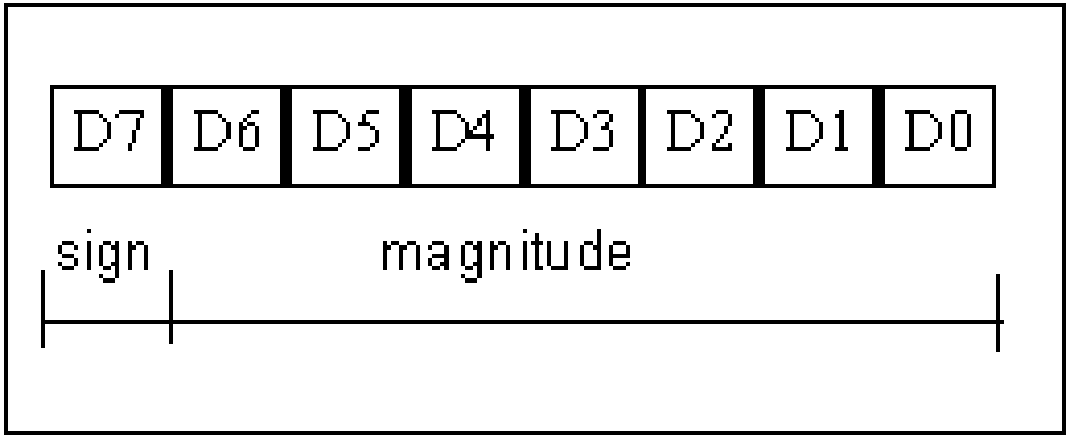
Solution:

```
#include <P18F458.INC>
;PIC Assembly Language Program for division (by repeated subtraction)
;(Byte/Byte)

NUME        EQU    0x15          ;RAM location for NUME
QU          EQU    0x20          ;RAM location for quotient
RMND_L      EQU    0x22
RMND_M      EQU    0x23
RMND_H      EQU    0x24
MYNUM       EQU    0xFD          ;FDH = 253 in decimal
MYDEN       EQU    D'10'         ;253/10
            ORG    0H            ;start at address 0
            MOVLW  MYNUM         ;WREG = 253, the numerator
            MOVWF  NUME          ;load numerator
            MOVLW  MYDEN         ;WREG = 10, the denominator
            CLRF   QU,F          ;clear quotient
D_1         INCF   QU,F          ;increment quotient for every sub
            SUBWF  NUME          ;sub WREG from NUME value
            BC     D_1           ;if positive go back (C = 1 for positive)
            ADDWF  NUME          ;once too many, this is our first digit
            DECF  QU,F           ;once too many for quotient
            MOVFF  NUME,RMND_L   ;save the first digit
            MOVFF  QU,NUME       ;repeat the process one more time
            CLRF   QU            ;clear QU
D_2         INCF   QU,F
            SUBWF  NUME          ;sub WREG from NUME value
            BC     D_2           ;(C = 1 for positive)
            ADDWF  NUME          ;once too many
            DECF  QU,F
            MOVFF  NUME,RMND_M   ;2nd digit
            MOVFF  QU,RMND_H     ;3rd digit
HERE        GOTO  HERE           ;stay here forever
            END                  ;end of asm source file
```

To convert a single decimal digit to ASCII format, we OR it with 30H, as shown in Sections 6.4 and 6.5.

Signed Number Concepts



Example 5-10

Show how the PIC would represent –5.

Solution:

Observe the following steps.

```
1.    0000 0101        5 in 8-bit binary
2.    1111 1010        invert each bit
3     1111 1011        add 1 (which becomes FB in hex)
```

Therefore, –5 = FBH, the signed number representation in 2's complement for –5. The D7 = N = 1 indicates that the number is negative.

Example 5-11

Show how the PIC would represent –34H.

Solution:

Observe the following steps.

```
1.    0011 0100        34H given in binary
2.    1100 1011        invert each bit
3     1100 1100        add 1 (which is CC in hex)
```

Therefore, –34 = CCH, the signed number representation in 2's complement for 34H. The D7 = N = 1 indicates that the number is negative.

Example 5-12

Show how the PIC would represent –128.

Solution:

Observe the following steps.

```
1.     1000 0000        128 in 8-bit binary
2.     0111 1111        invert each bit
3      1000 0000        add 1 (which becomes 80 in hex)
```

Therefore, –128 = 80H, the signed number representation in 2's complement for –128. The D7 = N = 1 indicates that the number is negative. Notice that 128 (binary 10000000) in unsigned representation is the same as signed –128 (binary 10000000).

Overflow Problem

Example 5-13

Examine the following code and analyze the result, including the N and OV flags.

```
MOVLW  +D'96'      ;WREG = 0110 0000
ADDLW  +D'70'      ;WREG = (+96) + (+70) = 1010 0110
                   ;WREG = A6H = -90 decimal, INVALID!!
```

Solution:

```
    +96     0110 0000
  + +70     0100 0110
  + 166     1010 0110   N = 1 (negative) and OV = 1. Sum = -90
```

According to the CPU, the result is negative (N = 1), which is wrong. The CPU sets OV = 1 to indicate the overflow error. Remember that the N flag is the D7 bit. If N = 0, the sum is positive, but if N = 1, the sum is negative.

Overflow Problem

OV is set to 1 if either of the following two conditions occurs.

- There is a carry from D6 to D7 but no carry out of D7 (C=0).
- There is a carry from D7 out (C=1) but no carry from D6 to D7.

Example 5-14

Observe the following, noting the role of the OV and N flags:

```
MOVLW -D'128'       ;WREG = 1000 0000 (WREG = 80H)
ADDLW -D'2'         ;W = (-128) + (-2)
                    ;W = 1000000 + 11111110 = 0111 1110,
                    ;N = 0, W = 7EH = +126, invalid
```

Solution:

```
 -128         1000 0000
+ - 2         1111 1110
- 130         0111 1110  N = 0 (positive) and OV = 1
```

According to the CPU, the result is +126, which is wrong, and OV = 1 indicates that.

Example 5-15

Observe the following, noting the OV and N flags:

```
MOVLW -D'2'          ;WREG = 1111 1110 (WREG = FEH)
ADDLW -D'5'          ;WREG = (-2) + (-5) = -7 or F9H
                     ;correct, since OV = 0
```

Solution:

```
   -2     1111 1110
 + -5     1111 1011
  - 7     1111 1001  and OV = 0 and N = 1. Sum is negative
```

According to the CPU, the result is –7, which is correct, and the OV flag indicates that. (OV = 0).

Logic Instructions

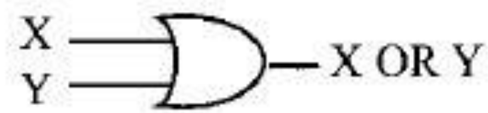
- ANDLW K – (WREG = WREG AND K).
- IORLW K – (WREG = WREG Inclusive-OR K).
- XORLW K – (WREG = WREG XOR K).
- COMF FileReg, d – Complementing.
- NEGF FileReg, d – Negate fileReg.

Logical AND Function

Inputs		Output
X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

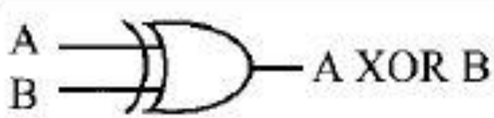
Logical OR Function

Inputs		Output
X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1



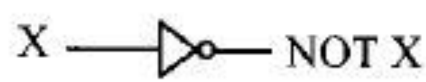
Logical XOR Function

Inputs		Output
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0



Logical Inverter

Input	Output
X	NOT X
0	1
1	0



Example 5-17

Show the results of the following.

```
MOVLW  0x35         ;WREG = 35H
ANDLW  0x0F         ;W = W AND 0FH (now W = 05)
```

Solution:

```
35H   0 0 1 1 0 1 0 1
0FH   0 0 0 0 1 1 1 1
05H   0 0 0 0 0 1 0 1    ;35H AND 0FH = 05H, Z = 0, N = 0
```

Example 5-18

(a) Show the results of the following:

```
MOVLW 0x04          ;WREG = 04
IORLW 0x30          ;now WREG = 34H
```

(b) Assume that Port B bit RB2 is used to control an outdoor light, and bit RB5 to control a light inside a building. Show how to turn "on" the outdoor light and turn "off" the inside one.

Solution:

```
(a)
      04H      0000 0100
      30H      0011 0000
      34H      0011 0100      04 OR 30 = 34H, Z = 0 and N = 0

(b)
      BCF   TRISB,2           ;make RB2 an output
      BCF   TRISB,5           ;make RB5 an output
      MOVLW B'00000100'       ;D2 = 1
      IORWF PORTB,F           ;make RB2 = 1 only
      MOVLW B'11011111'       ;D5 = 0
      ANDWF PORTB,F           ;mask RB5 = 0 only
```

Of course, the above method is unnecessary in PIC, since we can manipulate individual bits using bit-oriented operations. This is shown in Section 6.4.

Example 5-19

Show the results of the following:

```
MOVLW 0x54
XORLW 0x78
```

Solution:

```
54H        0 1 0 1 0 1 0 0
78H        0 1 1 1 1 0 0 0
2CH        0 0 1 0 1 1 0 0    54H XOR 78H = 2CH, Z = 0, N = 0
```

Example 5-21

Read and test PORTB to see whether it has value 45H. If it does, send 99H to PORTC; otherwise, it stays cleared.

Solution:

```
      CLRF  TRISC          ;Port C = output
      CLRF  PORTC          ;Port C = 00
      SETF  TRISB          ;Port B = input
      MOVLW 0x45
      XORWF PORTB,W        ;EX-OR with 0x45, Z = 1 if yes
      BNZ   EXIT           ;branch if PORTB has value other than 0
      MOVLW 0x99
      MOVWF PORTC          ;Port C = 99h
EXIT:...
```

Compare Instructions

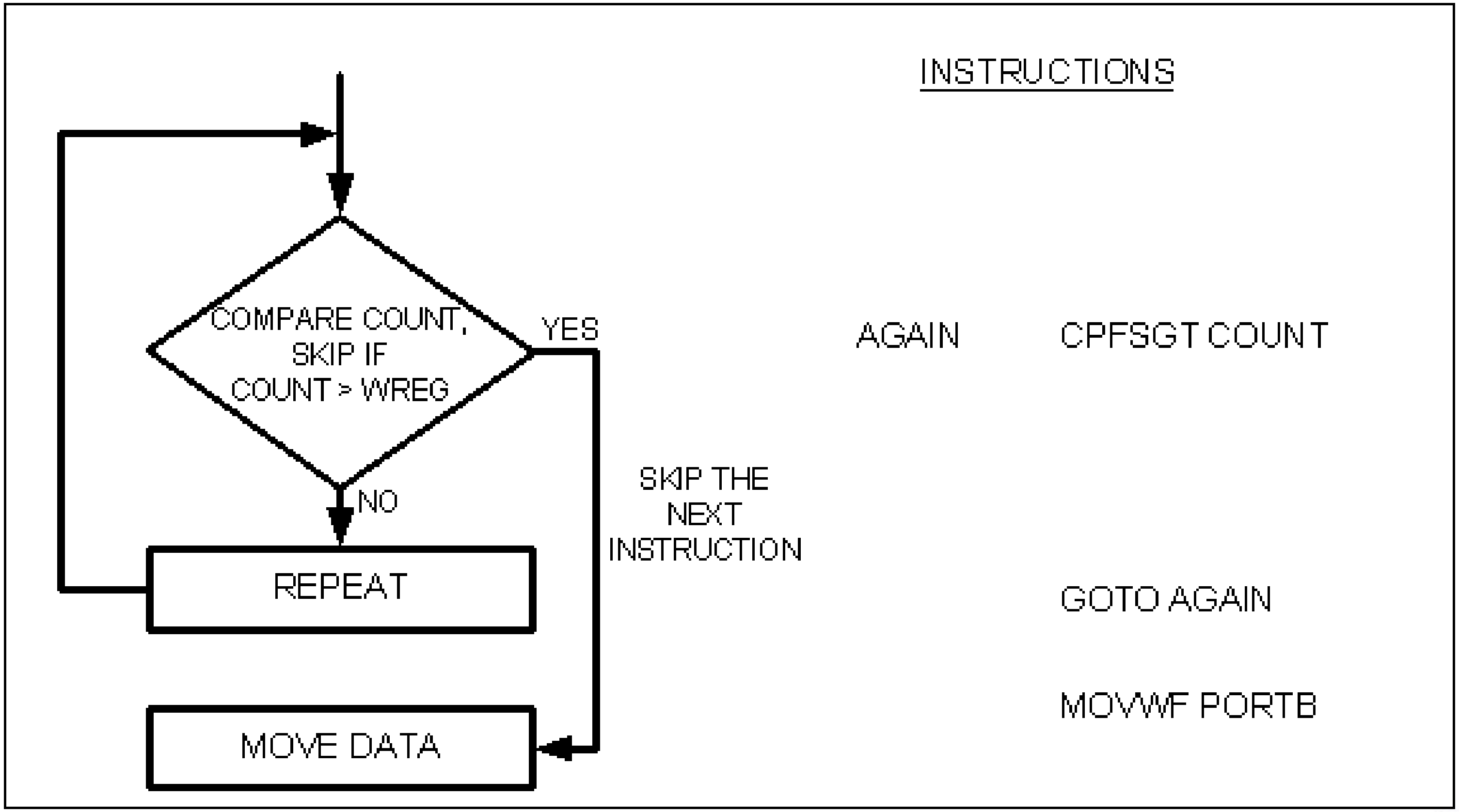
- The PIC18 has three compare instructions, which compare a value in the file register with the contents of the WREG.

Table 5-2: PIC18 Compare Instructions

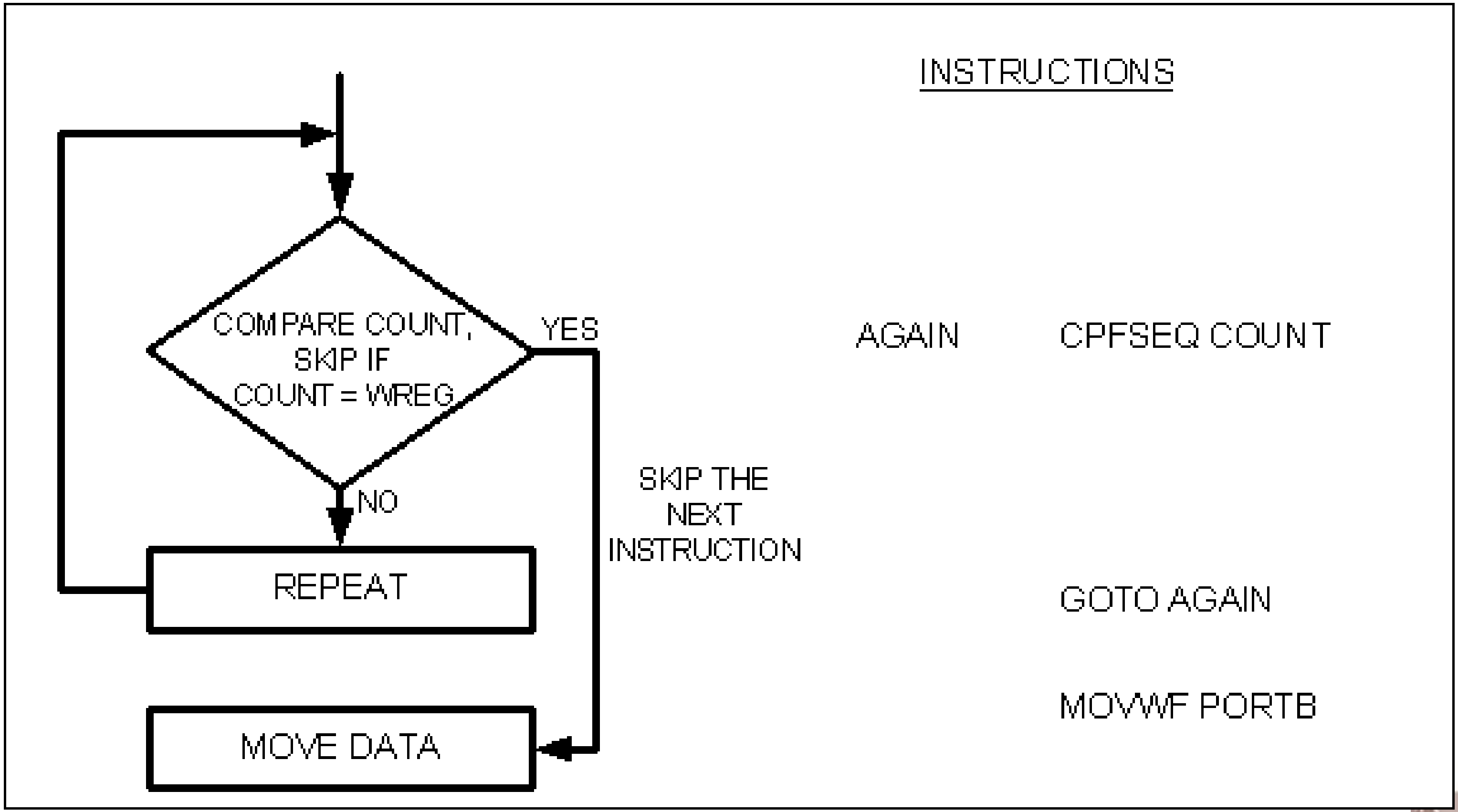
CPFSGT	Compare FileReg with WREG, skip if greater than	FileReg > WREG
CPFSEQ	Compare FileReg with WREG, skip if equal	FileReg = WREG
CPFSLT	Compare fileReg with WREG, skip if less than	FileReg < WREG

Note: These instructions have no effect on the flag bits of the status register. Also the values in fileReg and WREG remain unchanged.

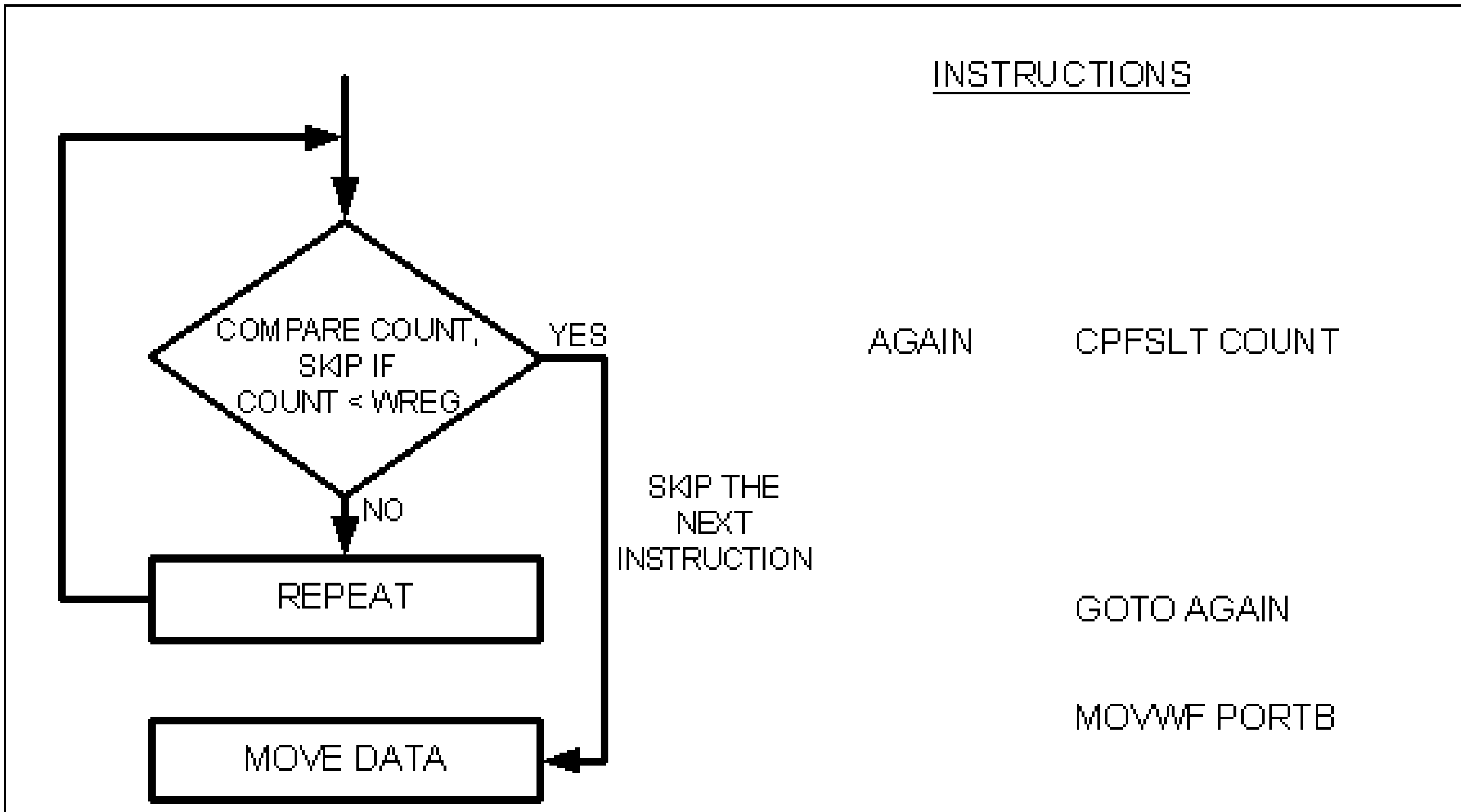
Flowchart for CPFSGT



Flowchart for CPFSEQ



Flowchart for CPFSLT



Example 5-25

Write a program to find the smaller of the two values 27 and 54, and place it in file register location 0x20.

Solution:

```
VAL_1 EQU   D'27'
VAL_2 EQU   D'54'
LREG  EQU   0x20  ;location for smaller of two

MOVLW  VAL_1      ;WREG = 27
MOVWF  LREG       ;LREG = 27
MOVLW  VAL_2      ;WREG = 54
CPFSLT LREG       ;skip if LREG < WREG
MOVWF  LREG       ;place the smaller value in LREG
```

Example 5-26

Assume that Port D is an input port connected to a temperature sensor. Write a program to read the temperature and test it for the value 75. According to the test results, place the temperature value into the registers indicated by the following.

If $T = 75$	then $WREG = 75$
If $T > 75$	then $GREG = T$
If $T < 75$	then $LREG = T$

Solution:

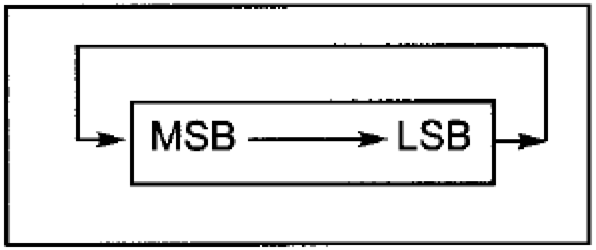
```
LREG EQU 0x20
GREG EQU 0x21
      SETF    TRISD              ;PORTD = input
      MOVLW   D'75'              ;WREG = 75 decimal
      CPFSGT  PORTD              ;skip BRA instruction if PORTD > 75
      BRA     LEQ
      MOVFF   PORTD, GREG
      BRA     OVER
LEQ   CPFSLT  PORTD              ;skip if PORTD < 75
      BRA     OVER
      MOVFF   PORTD, LREG
OVER  .....                      ;it must be equal, WREG = 75
```

Rotate Instruction

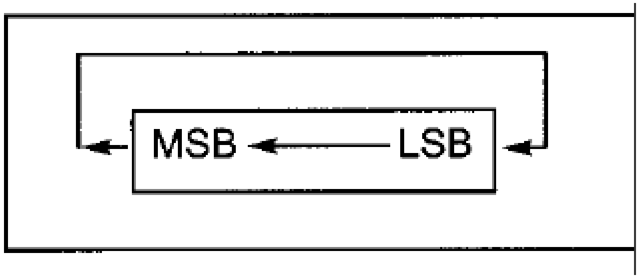
- RRNCF   fileReg, d – Rotate fileReg right with no carry.
- RLNCF   fileReg, d – Rotate fileReg left with no carry.
- RRCF   fileReg, d – Rotate fileReg right with carry.
- RLCF   fileReg, d – Rotate fileReg left with carry.

RRNCF and RLNCF

```
MREG EQU 0x20
      MOVLW  0x36          ;WREG = 0011 0110
      MOVWF  MYREG
      RRNCF  MYREG,F       ;MYREG = 0001 1011
      RRNCF  MYREG,F       ;MYREG = 1000 1101
      RRNCF  MYREG,F       ;MYREG = 1100 0110
      RRNCF  MYREG,F       ;MYREG = 0110 0011
```

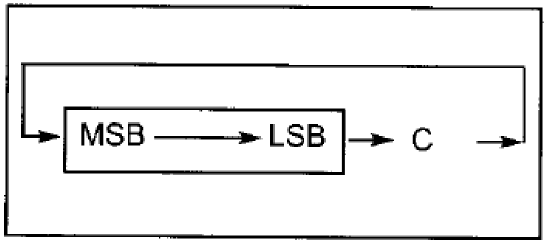



```
MREG EQU 0x20
      MOVLW  0x72          ;WREG = 0111 0010
      MOVWF  MYREG
      RLNCF  MYREG,F       ;MYREG = 1110 0100
      RLNCF  MYREG,F       ;MYREG = 1100 1001
```

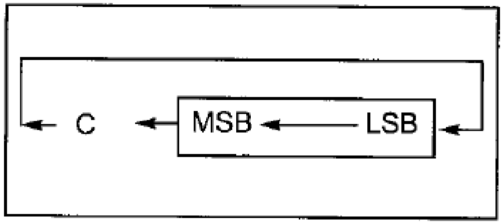


RRCF and RLCF

```
MREG EQU 0x20
       BCF     STATUS,C    ;make C = 0 (carry is D0 of status)
       MOVLW   0x26        ;WREG = 0010 0110
       MOVWF   MYREG
       RRCF    MYREG,F     ;MYREG = 0001 0011 C = 0
       RRCF    MYREG,F     ;MYREG = 0000 1001 C = 1
       RRCF    MYREG,F     ;MYREG = 1000 0100 C = 1
```




```
MREG EQU 0x20
       BSF     STATUS,C    ;make C = 1 (carry is D0 of status)
       MOVLW   0x15        ;WREG = 0001 0101
       MOVWF   MYREG
       RLCF    MYREG,F     ;MYREG = 0010 1011 C = 0
       RLCF    MYREG,F     ;MYREG = 0101 0110 C = 0
       RLCF    MYREG,F     ;MYREG = 1010 1100 C = 0
       RLCF    MYREG,F     ;MYREG = 0101 1000 C = 1
```



Serializing a byte of data

Example 5-28

Write a program to transfer value 41H serially (one bit at a time) via pin RB1. Put one high at the start and end of the data. Send the LSB first.

Solution:

```
      RCNT   EQU   0x20          ;fileReg loc for counter
      MYREG  EQU   0x21          ;fileReg loc for rotate

      BCF    TRISB,1             ;make RB1 an output bit
      MOVLW  0x41                ;WREG = 41
      MOVWF  MYREG               ;value to be serialized
      BCF    STATUS,C            ;C = 0
      MOVLW  0x8                 ;counter
      MOVWF  RCNT                ;load the counter
      BSF    PORTB,1             ;RB1 = high
AGAIN RRCF   MYREG,F             ;rotate right via carry
      BNC    OVER
      BSF    PORTB,1             ;set the carry bit to PB1
      BRA    NEXT
OVER  BCF    PORTB,1
NEXT  DECF   RCNT,F
      BNZ    AGAIN
      BSF    PORTB,1             ;RB1 = high
```

Serializing a byte of data

Example 5-29

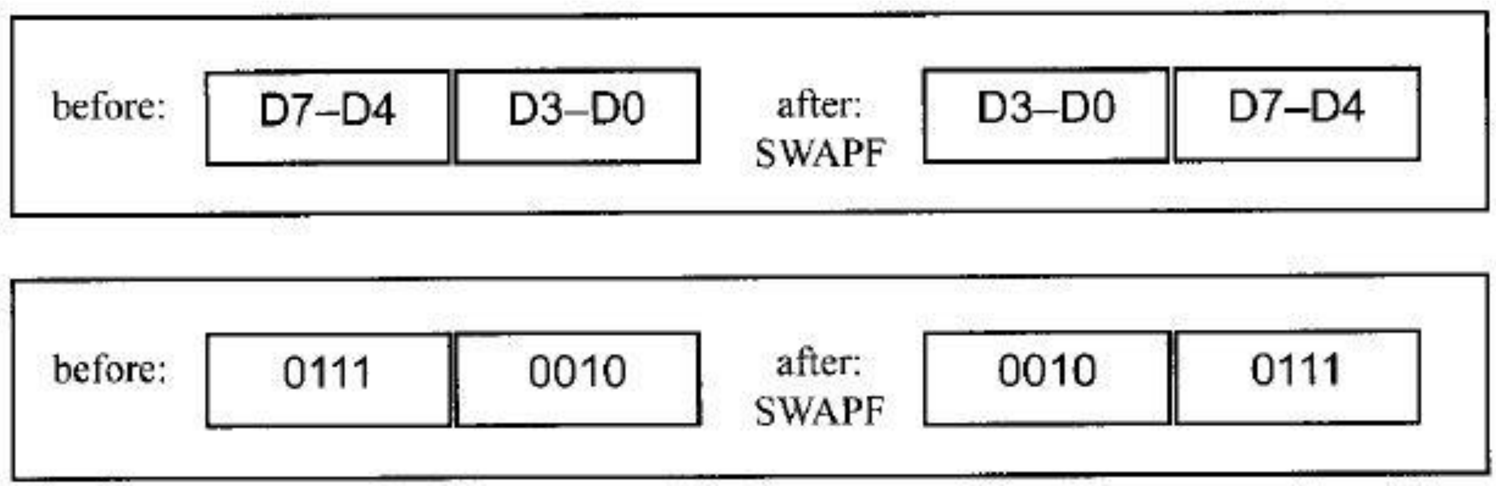
Write a program to bring in a byte of data serially (one bit at a time) via pin RC7 and save it in file register location 0x21. The byte comes in with the LSB first.

Solution:

```
RCNT  EQU   0x20          ;fileReg loc for counter
MYREG EQU   0x21          ;fileReg loc for incoming byte

      BSF   TRISC,7              ;make RC7 an input bit
      MOVLW 0x8                  ;counter
      MOVWF RCNT                 ;load the counter
AGAIN BTFSC PORTC,7              ;skip if RC7 = 0
      BSF   STATUS,C             ;carry = 1
      BTFSS PORTC,7              ;skip if RC7 = 1
      BCF   STATUS,C             ;otherwise carry = 0
      RRCF  MYREG,F              ;rotate right carry into MYREG
      DECF  RCNT,F               ;decrement the counter
      BNZ   AGAIN                ;repeat until RCNT = 0
                                 ;now loc 21H has the byte
```

SWAPF fileReg, d



Example 5-31

(a) Find the contents of the MYREG register in the following code.
(b) In the absence of a SWAPF instruction, how would you exchange the nibbles? Write a simple program to show the process.

Solution:

(a)

```
MYREG EQU 0x20
      MOVLW 0x72          ;WREG = 72H
      MOVWF MYREG         ;MYREG = 72H
      SWAPF MYREG,F       ;MYREG = 27H
```

(b)

```
MYREG EQU 0x20
      MOVLW 0x72          ;WREG = 0111 0010
      MOVWF MYREG         ;MYREG = 0111 0010
      RLNCF MYREG,F       ;MYREG = 1110 0100
      RLNCF MYREG,F       ;MYREG = 1100 1001
      RLNCF MYREG,F       ;MYREG = 1001 0011
      RLNCF MYREG,F       ;MYREG = 0010 0111
```

BCD and ASCII Conversion

Table 5-3: ASCII and BCD Codes for Digits 0–9

Key	ASCII (hex)	Binary	BCD (unpacked)
0	30	011 0000	0000 0000
1	31	011 0001	0000 0001
2	32	011 0010	0000 0010
3	33	011 0011	0000 0011
4	34	011 0100	0000 0100
5	35	011 0101	0000 0101
6	36	011 0110	0000 0110
7	37	011 0111	0000 0111
8	38	011 1000	0000 1000
9	39	011 1001	0000 1001

Packed BCD to ASCII Conversion

Packed BCD	*Unpacked BCD*	*ASCII*
29H	02H & 09H	32H & 39H
0010 1001	0000 0010 &	0011 0010 &
	0000 1001	0011 1001

Example 5-32

Assume that register WREG has packed BCD. Write a program to convert packed BCD to two ASCII numbers and place them in file register locations 6 and 7.

Solution:

```
BCD_VAL EQU 0x29
L_ASC   EQU 0x06  ;set aside file register location
H_ASC   EQU 0x07  ;set aside file register location

      MOVLW BCD_VAL     ;WREG = 29H, packed BCD
      ANDLW 0x0F        ;mask the upper nibble (W = 09)
      IORLW 0x30        ;make it an ASCII, W = 39H ('9')
      MOVWF L_ASC       ;save it (L_ASC = 39H ASCII char)
      MOVLW BCD_VAL     ;W = 29H get BCD data once more
      ANDLW 0xF0        ;mask the lower nibble (W = 20H)
      SWAPF WREG,W      ;swap nibbles (WREG = 02H)
      IORLW 0x30        ;make it an ASCII, W = 32H ('2')
      MOVWF H_ASC       ;save it (H_ASC = 32H ASCII char)
```

ASCII to Packed BCD Conversion

Key	***ASCII***	***Unpacked BCD***	***Packed BCD***
4	34	00000100	
7	37	00000111	01000111 which is 47H

```
MYBCD EQU 0x20      ;set aside location in file register

      MOVLW A'4'          ;WREG = 34H, hex for ASCII char 4
      ANDLW 0x0F          ;mask upper nibble (WREG = 04)
      MOVWF MYBCD         ;save it in MYBCD loc
      SWAPF MYBCD,F       ;MYBCD = 40H
      MOVLW A'7'          ;WREG = 37H, hex for ASCII char 7
      ANDLW 0x0F          ;mask upper nibble (WREG = 07)
      IORWF MYBCD,F       ;MYBCD = 47H, a packed BCD
```