Microprocessor Interfacing & Programming

LECTURE 16

Multiplication of unsigned numbers

- PIC supports byte-by-byte multiplication only.
- The bytes are assumed to be unsigned data.

SYNTAX: MULLW K ; W x K and 16 bit is result is in PRODH : PRODL

- The result is in the special function registers PRODH and PRODL.

Example:

```
MOVLW  0x25
MULLW  0x65
```

NOTE: Multiplication of operands larger than 8 bits takes some manipulation

Division of unsigned numbers

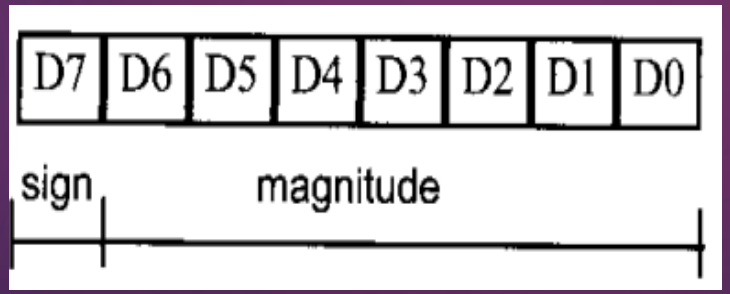
- There is no single instruction for the division of byte/byte numbers in PIC18.
- We write a program to perform division by repeated subtraction.
- In dividing a byte by a byte, the numerator is placed in a fileReg and the denominator is subtracted from it repeatedly.
- The quotient is the number of times we subtracted and the remainder is in fileReg upon completion.

Example

```
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
```

Signed numbers

- In signed byte operands, D7 (MSB) is the sign and D0 to D6 are set aside for the magnitude of the number.
- If D7= 0, the operand is +ve, and if D7=1, it is negative.
- The N flag in the status register is the D7 bit.



Positive and Negative numbers

- The range of positive numbers is 0 to +127.
- The range of negative numbers is -1 to -128.
- For negative numbers, D7 is 1 and the magnitude is represented in it's 2's complement.
- To convert to negative number representation (2's complement), these steps are followed:

1. Write the magnitude of the number in 8 bit binary (no sign).
2. Invert each bit.
3. Add 1 to it.

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.

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.

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).

Decimal	*Binary*	*Hex*
-128	1000 0000	80
-127	1000 0001	81
-126	1000 0010	82
...	.....	..
-2	1111 1110	FE
-1	1111 1111	FF
0	0000 0000	00
+1	0000 0001	01
+2	0000 0010	02
..	.....	..
+127	0111 1111	7F

Overflow Problem

- When using signed numbers, the problem occurs known as **overflow**.
- The PIC indicates the existence of this error by raising the OV flag.

OVERFLOW: If the result of an operation on signed numbers is too large for the register, an overflow has occurred.

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.

When is the OV flag set?

- In 8-bit signed number operations, OV is set to 1 if either of the following two conditions occurs:

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

This means, if there is a carry both from D6 to D7 and from D7 out, OV = 0.

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.

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).

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

```
MOVLW +D'7'        ;WREG = 0000 0111
ADDLW +D'18'       ;W = (+7) + (+18)
                   ;W = 00000111 + 00010010 = 0001 1001
                   ;W = (+7) + (+18) = +25, N = 0, positive and
                   ;correct, OV = 0
```

Solution:

```
   + 7 0000 0111
+ +18 0001 0010
  +25 0001 1001   N = 0 (positive 25) and OV = 0
```

According to the CPU, this is +25, which is correct and OV = 0 indicates that.

NOTE: In unsigned addition, we must monitor the status of C flag, and in signed number addition, the OV flag must be monitored.

As in unsigned, we have BNC and BC. In signed numbers, there are BOV, BNOV, BN and BNN instructions.

Logic instructions

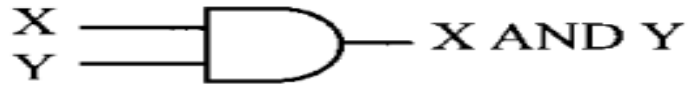
AND instruction ----- ANDLW K ; WREG = WREG AND K

ANDWF fileReg, d

- The AND instruction will affect the Z and N flags.
- This instruction is often used to mask (set to 0) cetain bits of an operand.

Logical AND Function

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



Example

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
```

OR instruction

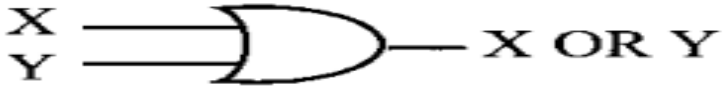
SYNTAX: IORLW K ; WREG = WREG INCLUSIVE-OR K

IORWF fileReg, d

- This instruction will also affect the Z and N flags.
- OR instruction is used to set certain bits of an operand to 1.

Logical OR Function

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



Example

(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.

EX-OR

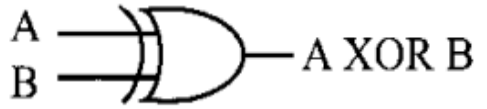
SYNTAX: XORLW K ;WREG = WREG XOR K

XORWF fileReg, d

- This instruction will also affect the Z and N flags.
- EX-OR can also be used to see if two registers have the same value.
- If both registers have the same value, 00 is placed in WREG, then we can use the BZ instruction to make a decision based on the result.

Logical XOR Function

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



- Another widely used application of EX-OR is to toggle the bits of an operand.

```
MOVLW 0xFF          ;WREG = FFH
XORWF PORTC,F       ;EX-OR PORTC with 1111 1111 will
                    ;change all the bits of Port C to
                    ;opposite
```

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
```

EX-OR

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:...
```

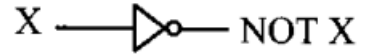
COMF and NEGF

- COMF (complement fileReg) instruction complements the contents of a file register. This instruction will perform 1's complement.

```
CLRF   TRISB          ;Port B = Output
MOVLW  0x55
MOVWF  PORTB
COMF   PORTB,F        ;now PORTB = AAH
```

Logical Inverter

Input	Output
X	NOT X
0	1
1	0



- NEGF (negate fleReg) intruction takes the 2's complement of a file register.

NEGF

Find the 2's complement of the value 85H. Note that 85H is –123.

Solution:

```
MYREG EQU 0x10
      MOVLW 0x85          85H = 1000 0101
      MOVWF MYREG         1'S = 0111 1010
      NEGF  MYREG                    + 1
                  2's comp    0111 1011 = 7BH
```

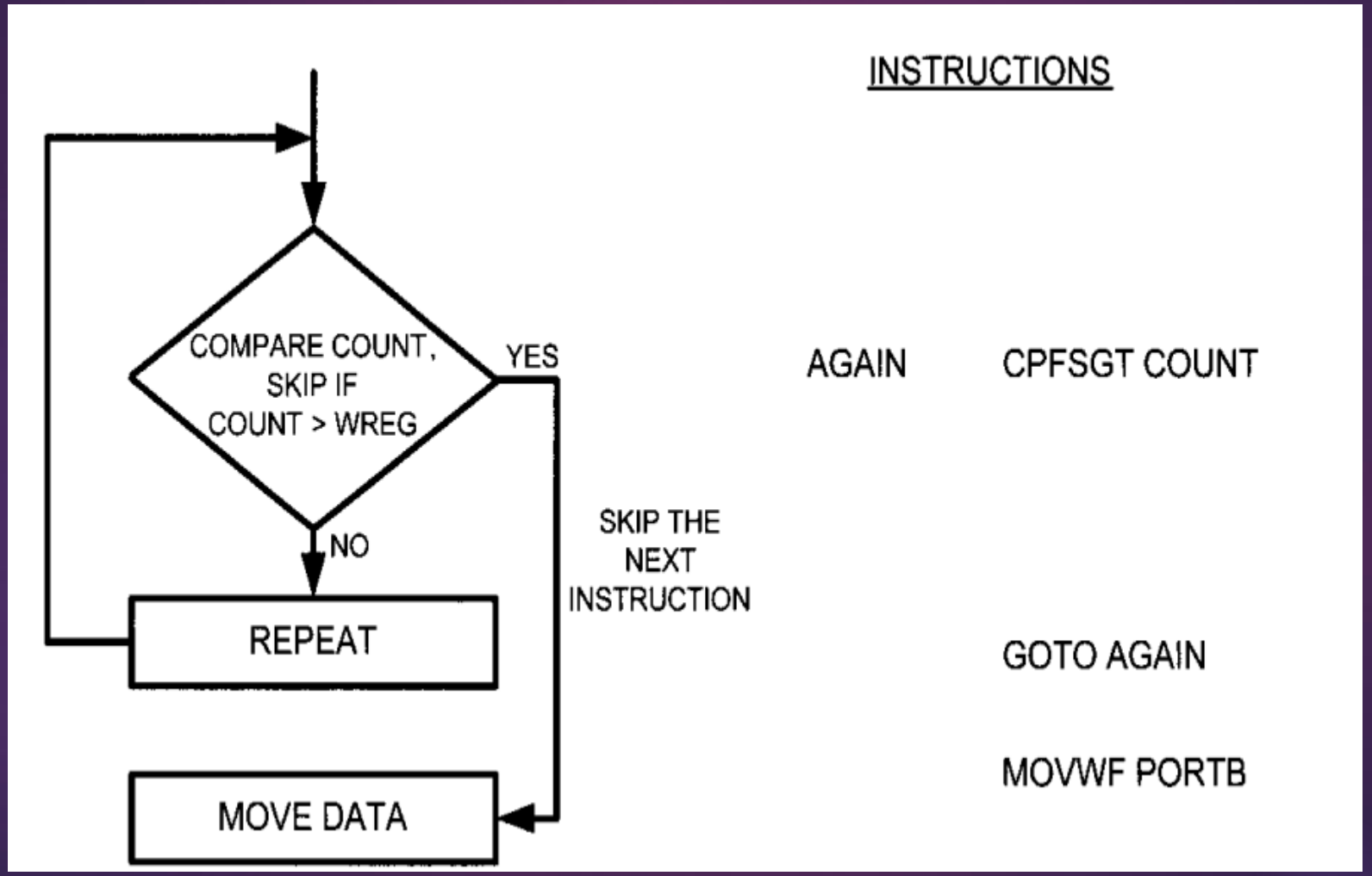
Compare instructions

- PIC18 has 3 intructions for the compare operation.
- These instructions compare a value in the file register with the contents of the WREG register and make decisions based on whether fileReg is greater than, equal to or less than WREG.
- The compare instruction is really a subtraction, except that the values of the operands do not change.
- In PIC18, flags are not changed either after the compare instruction.

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.

CPFSGT instruction



Example

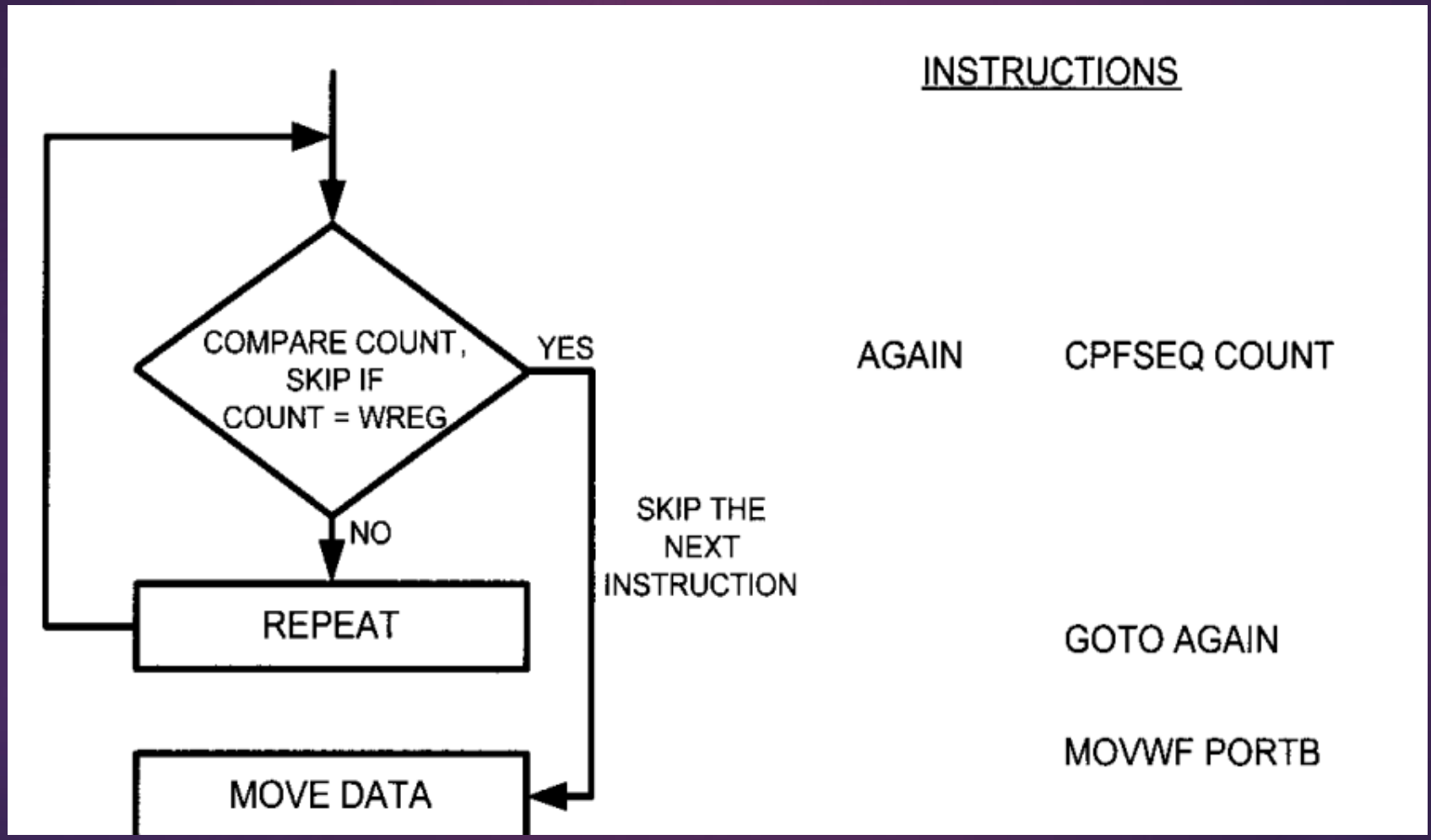
Write a program to find the greater of the two values 27 and 54, and place it in file register location 0x20.

Solution:

```
VAL_1 EQU   D'27'
VAL_2 EQU   D'54'
GREG  EQU   0x20

MOVLW  VAL_1      ;WREG = 27
MOVWF  GREG       ;GREG = 27
MOVLW  VAL_2      ;WREG = 54
CPFSGT GREG       ;skip if GREG > WREG
MOVWF  GREG       ;place the greater in GREG
```

CPFSEQ instruction



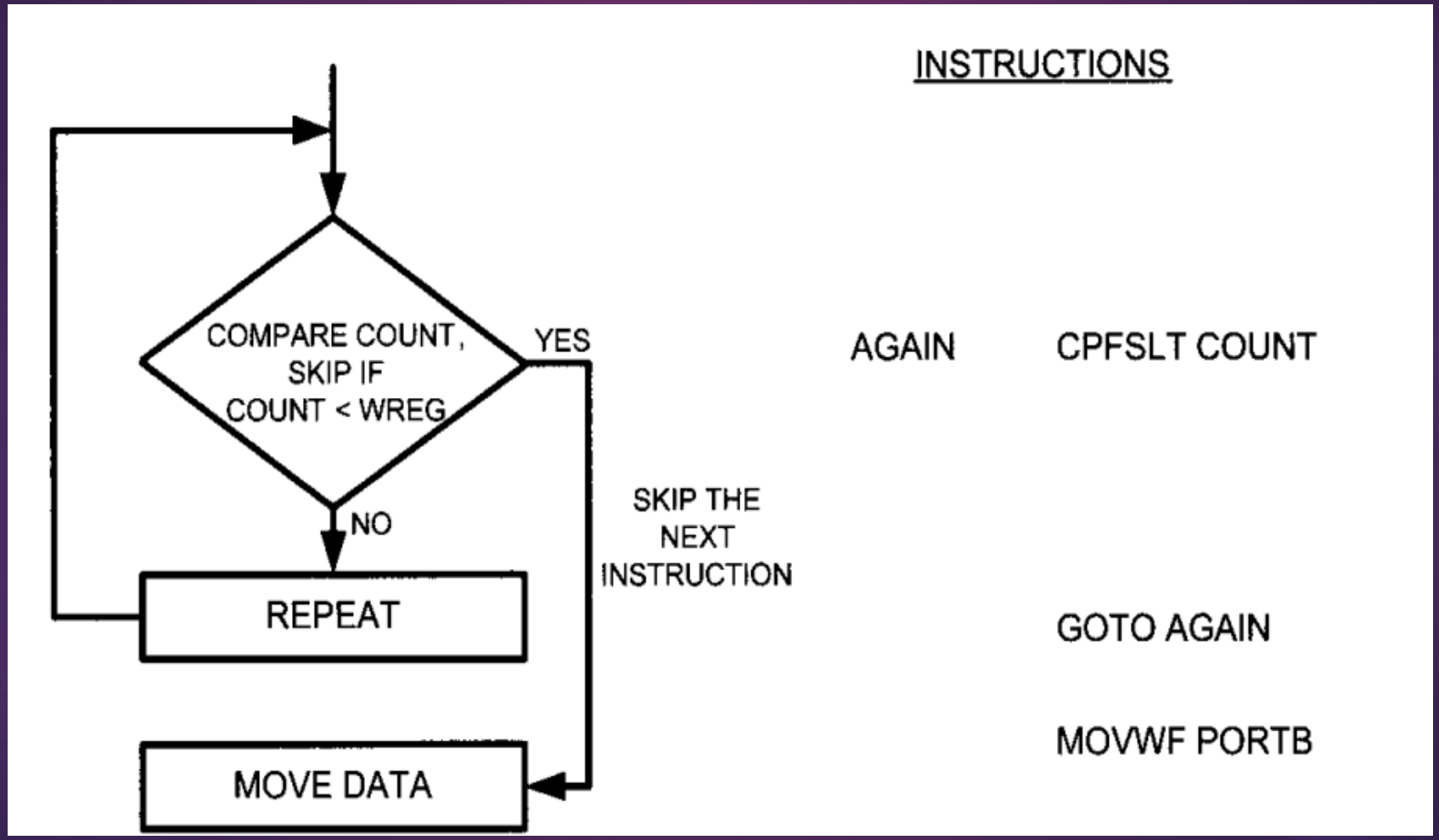
Example

Write a program to monitor PORTD continuously for the value 63H. It should stop monitoring only if PORTD = 63H.

Solution:

```
        SETF   TRISD      ;PORTD = input
        MOVLW  0x63       ;WREG = 63H
BACK    CPFSEQ PORTD      ;skip BRA instruction if PORTD = 63H
        BRA    BACK
        .....
```

CPFSLT instruction



Example

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
```