EC-310 Microprocessor and Microcontroller Based Design

Chapter - 2

PIC Architecture and Assembly Language Programming

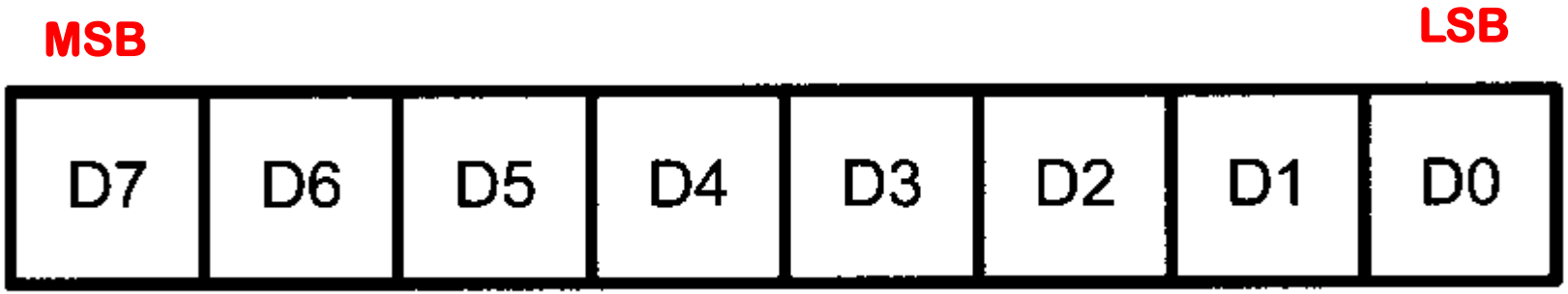
Nazar Abbas Saqib
nazar.abbas@ceme.nust.edu.pk

Outline

1. WREG Register in PIC Microcontrollers
2. The PIC File Register
3. Using Instructions with Default Access Bank
4. PIC Status Register
5. PIC Data Format and Directives
6. Introduction to PIC Assembly Programming
7. Assembling and Linking a PIC Program
8. The Program Counter and Program ROM Space in the PIC
9. RISC Architecture in the PIC

WREG Register in PIC Microcontrollers

- In CPU, registers are used to store information **temporarily**
- That information can be the data itself or an address, pointing to the data.
- Most of the PIC registers are 8-bit registers.



- In PIC there is only one data type and that is 8-bit. For data larger than 8-bit, data is divided into 8-bit chunks before processing.

WREG Register in PIC Microcontrollers

- WREG stands for **working register**.
- WREG is an 8-bit, most widely used register in PIC microcontroller
- Same as accumulator in other microprocessors.
- Used for all arithmetic and logical instructions.

WREG Register in PIC Microcontrollers

- **MOVLW Instruction**
  - **Moves the 8-bit data into the WREG register.**
  - **It has following format**

MOVLW K (Moves the literal value K into WREG)

  - **Here K is a literal i.e. it is the actual value. It cannot be an address of some memory location.**
  - **Literal K may belong to any of the following categories of number systems:**
    - Hexadecimal
    - Binary
    - ASCII (for characters)
    - Decimal
  - **All arithmetic and logical instructions involve WREG register.**

WREG Register in PIC Microcontrollers

- **MOVLW**

```
MOVLW 25H    ;move value 25H into WREG (WREG = 25H)

MOVLW 87H    ;load 87H into WREG (WREG = 87H)

MOVLW 15H    ;load 15H into WREG (WREG = 15H)
```

WREG Register in PIC Microcontrollers

- **MOVLW**

For writing a <u>hexadecimal number</u> into WREG register:

1. **MOVLW 02**
   Moves hexadecimal 2 into WREG Register

2. **MOVLW 02H**
   Moves hexadecimal 2 int0 WREG Register

3. **MOVLW 0x02**
   Moves hexadecimal 2 into WREG Register

WREG Register in PIC Microcontrollers

- **MOVLW**

For writing a binary number into WREG register:

1. MOVLW B '11001010'

Moves binary number into WREG Register

For writing a ASCII into WREG register:

1. MOVLW A 'd'

Moves an ASCII character into WREG Register

For writing a Decimal into WREG register:

1. MOVLW D '100'

Moves a decimal number into WREG Register

WREG Register in PIC Microcontrollers

- **Testing your knowledge**

Tell whether the following instructions are valid or not. Justify you answer.

- **MOVLW 0 x AB**

Answer: Valid

- **MOVLW D '100'**

Answer: Valid

- **MOVLW 0 x EFF**

Answer: Invalid: because 0xEFF is a 12-bit num whereas WREG is 8-bit num.

- **MOVLW D '328'**

Answer: Invalid: : because 8-bit WREG can only accommodate a decimal number less than 256.

WREG Register in PIC Microcontrollers

- **ADDLW**
  - This instruction is used to add two literal and place the result in WREG.
  - **It has following format**

```
ADDLW    K          ; WREG = WREG + K
```

  - **To add two numbers X and Y:**

```
MOVLW X             ;WREG = X
ADDLW Y             ;WREG = X + Y
```

WREG Register in PIC Microcontrollers

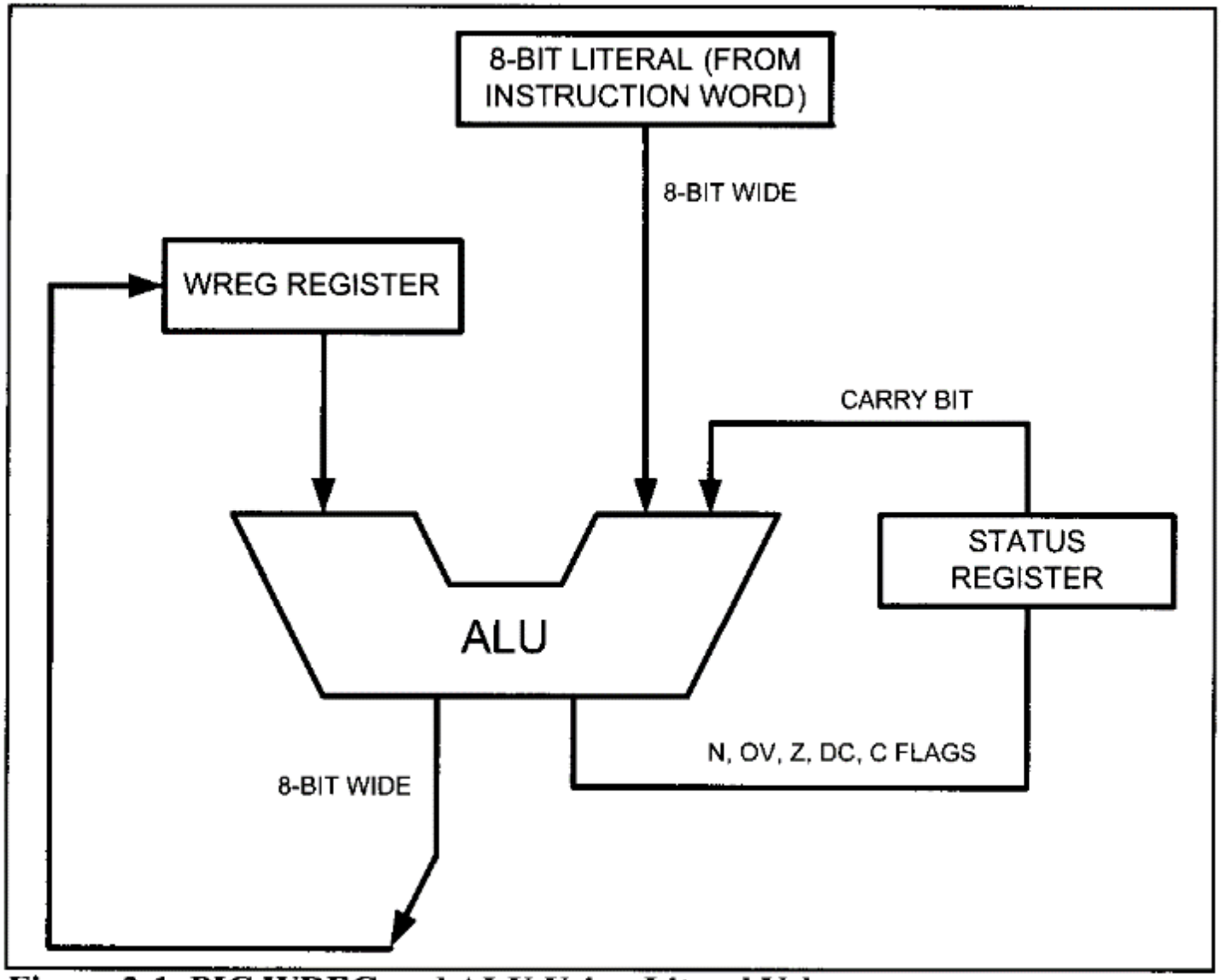


Figure 2-1. PIC WREG and ALU Using Literal Value

WREG Register in PIC Microcontrollers

- **ADDLW**
- **Instructions to add 25H and 34H**

```
MOVLW 25H    ;load 25H into WREG
ADDLW 34H    ;add value 34 to W(W = W + 34H)
```

Executing the above lines results in WREG = 59H (25H + 34H = 59H)

WREG Register in PIC Microcontrollers

- **ADDLW**

The following program will add values 12H, 16H, 31H, and 43H:

```
MOVLW 12H    ;load value 12H into WREG (WREG = 12H)
ADDLW 16H    ;add 16 to WREG (WREG = 28H)
ADDLW 11H    ;add 11 to WREG (WREG = 39H)
ADDLW 43H    ;add 43 to WREG (WREG = 7CH)
```

The PIC File Register

The PIC File Register

- **PIC microcontroller has many other registers other than WREG, called as**
  - Program (code) memory space
  - Data memory space
- **The data memory space is also called File register** (letter F represent it in programs)
- **File Register Data RAM is divided into two sections**
  - General Purpose Registers (GP-RAM)
  - Special Function Registers (SFRs)

The PIC File Register

Special Function Registers (SFRs)

Dedicated to Special Functions such as: ALU, Status, Timers, ADC, Serial Communication, I/O Ports.

PIC SFRs are 8-bit registers

Function of SFRs is fixed by the manufacturers

Used for control of microcontroller or peripherals

Number of SRFs depends upon the number of peripherals

The PIC File Register

- **General Purpose Ram (GP – RAM)**

Data Storage and Scratch Pad

Each Location is 8-bit wide

GP – RAM is usually greater than SFRs

Greater the GP-RAM more difficult is memory management

The PIC File Register

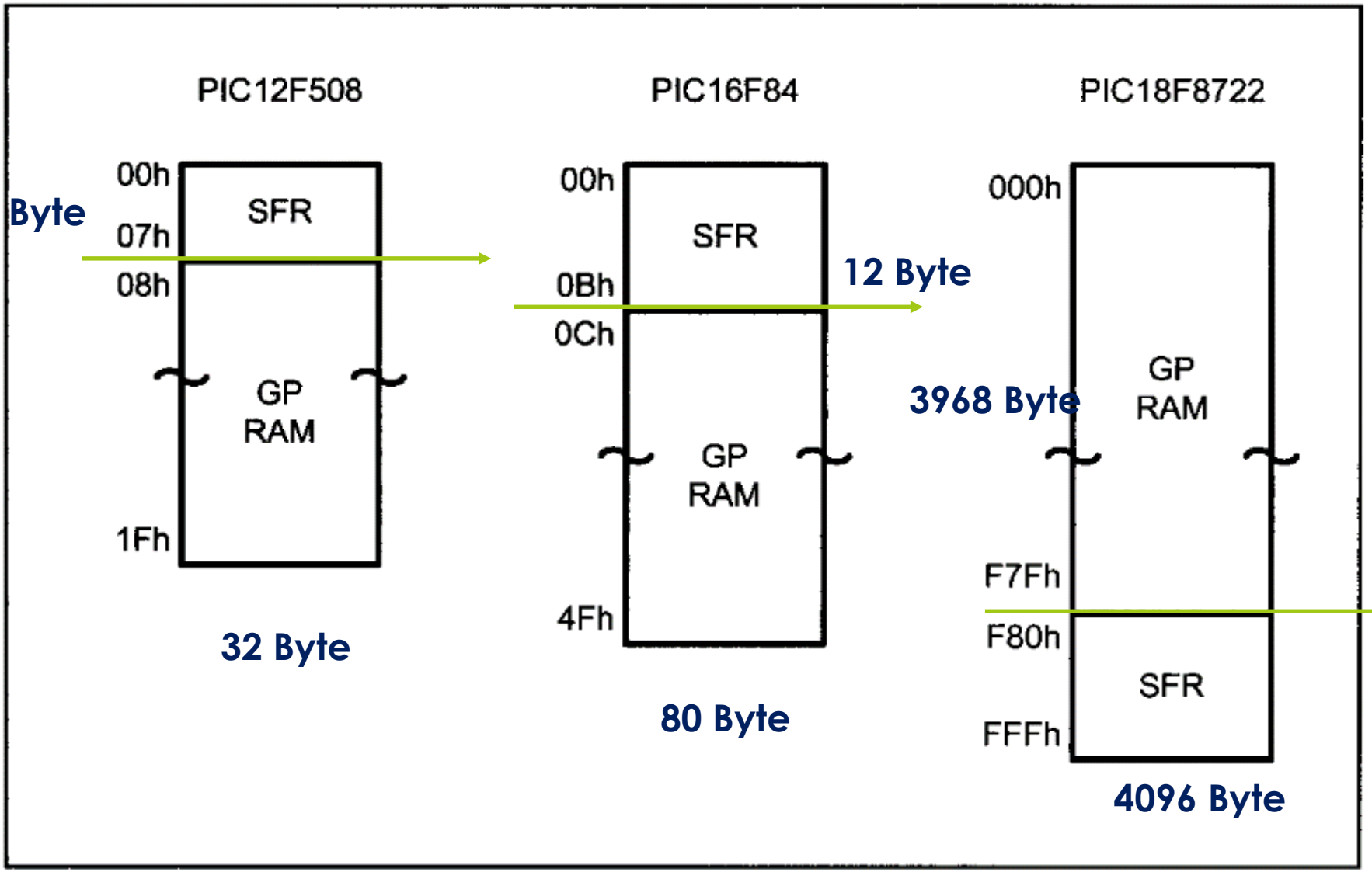


Figure 2-2. File Registers of PIC12, PIC16, and PIC18

The PIC File Register

- **File Register Sizes of PIC Chips**

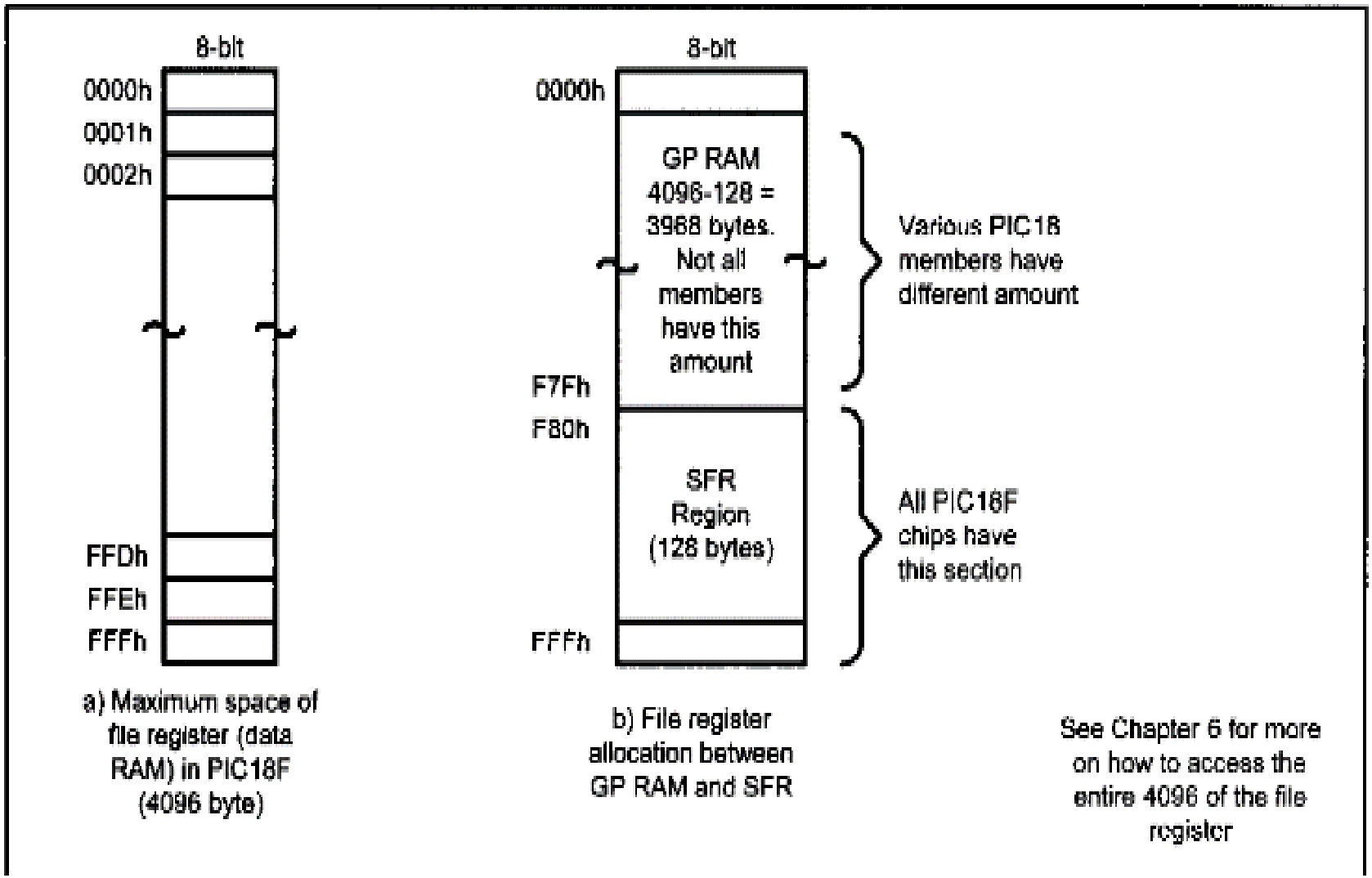
	File Register (Bytes)	=	SFR (Bytes)	+	Available space for GPR (Bytes)
PIC12F508	32		7		25
PIC16F84	80		12		68
PIC18F1220	512		256		256
PIC18F452	1792		256		1536
PIC18F2220	768		256		512
PIC18F458	1792		256		1536
PIC18F8722	4096		128		3968

The PIC File Register

- **File Register and access bank in PIC18**
- **File Register of PIC18 can have a maximum of 4096 bytes from 000H to FFFH memory address locations.**
- **File register of PIC18 is divided into 256-bytes banks. For 4096 bytes there exist 16 banks (16 x 256 bytes)**
- The minimum bank that every PIC has is called as access bank and is made of 128 bytes of lower addresses and 128 bytes of higher addresses
  - **The memory from 00H to 7FH (128-bytes) is reserved for Access-RAM**
  - **The memory in between 7FH and F80H is reserved for GP-RAM for various PIC18**
  - **The memory from F80H to FFH (128-bytes) is reserved for SFRs**

The PIC18 File Register

- File Register and access bank in PIC18.



The PIC File Register

- File Register and access bank in PIC18

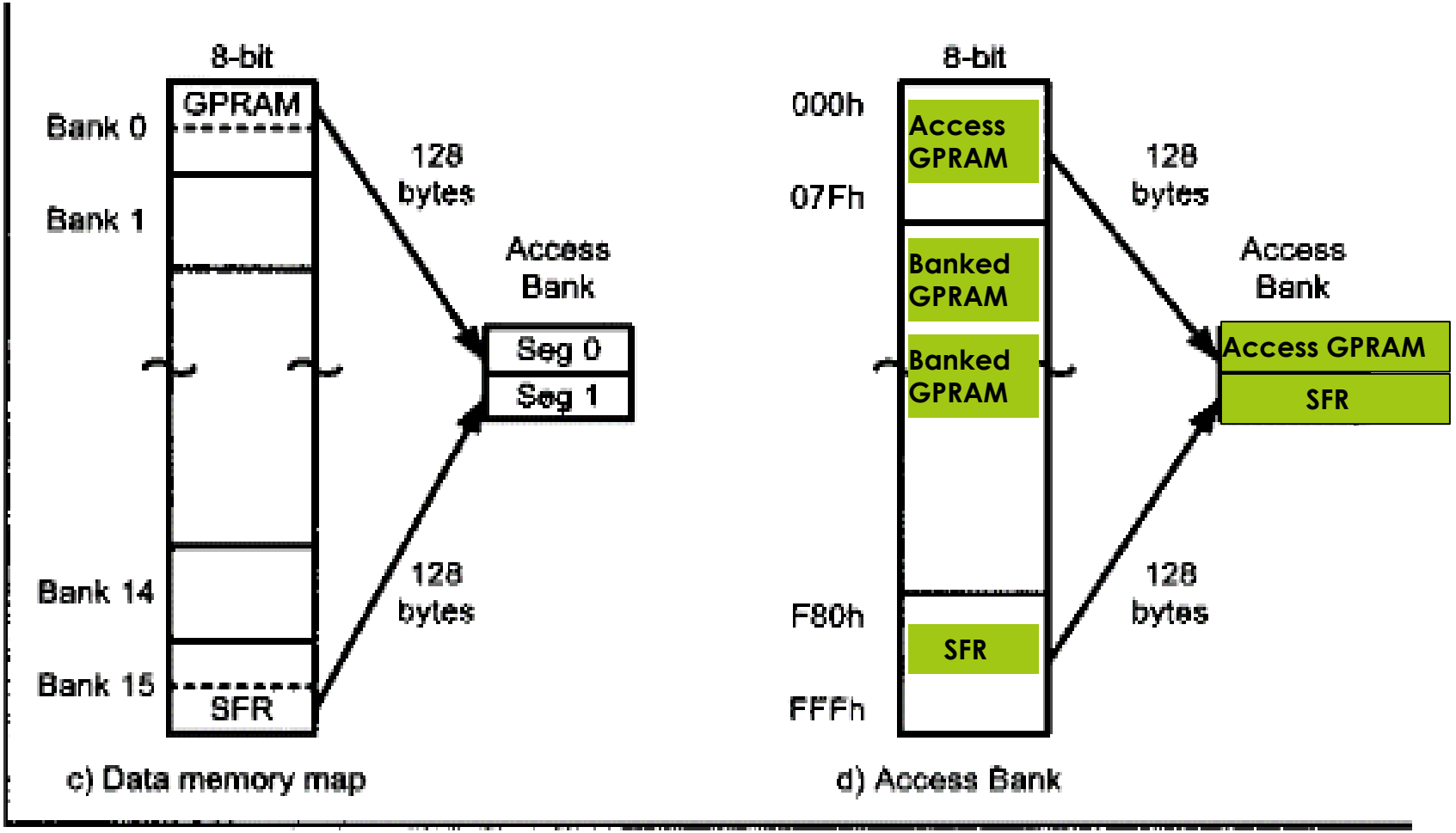


Figure 2-3. File Register for PIC18 Family

F80h	PORTA	FA0h	PIE2	FC0h	----	FE0h	BSR
F81h	PORTB	FA1h	PIR2	FC1h	ADCON1	FE1h	FSR1L
F82h	PORTC	FA2h	IPR2	FC2h	ADCON0	FE2h	FSR1H
F83h	PORTD	FA3h	----	FC3h	ADRESL	FE3h	PLUSW1 *
F84h	PORTE	FA4h	----	FC4h	ADRESH	FE4h	PREINC1 *
F85h	----	FA5h	----	FC5h	SSPCON2	FE5h	POSTDEC1 *
F86h	----	FA6h	----	FC6h	SSPCON1	FE6h	POSTINC1 *
F87h	----	FA7h	----	FC7h	SSPSTAT	FE7h	INDF1 *
F88h	----	FA8h	----	FC8h	SSPADD	FE8h	WREG
F89h	LATA	FA9h	----	FC9h	SSPBUF	FE9h	FSR0L
F8Ah	LATB	FAAh	----	FCAh	T2CON	FEAh	FSR0H
F8Bh	LATC	FABh	RCSTA	FCBh	PR2	FEBh	PLUSW0 *
F8Ch	LATD	FACh	TXSTA	FCCh	TMR2	FECh	PREINC0 *
F8Dh	LATE	FADh	TXREG	FCDh	T1CON	FEDh	POSTDEC0 *
F8Eh	----	FAEh	RCREG	FCEh	TMR1L	FEEh	POSTINC0 *
F8Fh	----	FAFh	SPBRG	FCFh	TMR1H	FEFh	INDF0 *
F90h	----	FB0h	----	FD0h	RCON	FF0h	INTCON3
F91h	----	FB1h	T3CON	FD1h	WDTCON	FF1h	INTCON2
F92h	TRISA	FB2h	TMR3L	FD2h	LVDCON	FF2h	INTCON
F93h	TRISB	FB3h	TMR3H	FD3h	OSCCON	FF3h	PRODL
F94h	TRISC	FB4h	----	FD4h	----	FF4h	PRODH
F95h	TRISD	FB5h	----	FD5h	T0CON	FF5h	TABLAT
F96h	TRISE	FB6h	----	FD6h	TMR0L	FF6h	TBLPTRL
F97h	----	FB7h	----	FD7h	TMR0H	FF7h	TBLPTRH
F98h	----	FB8h	----	FD8h	STATUS	FF8h	TBLPTRU
F99h	----	FB9h	----	FD9h	FSR2L	FF9h	PCL
F9Ah	----	FBAh	CCP2CON	FDAh	FSR2H	FFAh	PCLATH
F9Bh	----	FBBh	CCPR2L	FDBh	PLUSW2 *	FFBh	PCLATU
F9Ch	----	FBCh	CCPR2H	FDCh	PREINC2 *	FFCh	STKPTR
F9Dh	PIE1	FBDh	CCP1CON	FDDh	POSTDEC2 *	FFDh	TOSL
F9Eh	PIR1	FBEh	CCPR1L	FDEh	POSTINC2 *	FFEh	TOSH
F9Fh	IPR1	FBFh	CCPR1H	FDFh	INDF2 *	FFFh	TOSU

* - These are not physical registers.

Figure 2-4. Special Function Registers of the PIC18 Family.

Using Instructions with Default Access Bank

- **MOVWF Instruction**
  - **Moves/copies the contents of WREG register into fileReg**

Ex:

- **MOVWF 02H** (Moves the contents of WREG to 02H memory location)
- **MOVWF PORTA** (Moves the contents of WREG to PORTA SFR)
- **To move any value into the SFR or GPRAM, first it must be placed in WREG**

```
MOVLW 55H          ;WREG = 55H
MOVWF PORTB        ;copy WREG to Port B (Port B = 55H)
MOVWF PORTC        ;copy WREG to Port C (Port C = 55H)
MOVWF PORTD        ;copy WREG to Port D (Port D = 55H)
```

Example 2-1

State the contents of file register RAM locations after the following program:

```
MOVLW 99H          ;load WREG with value 99H
MOVWF 12H
MOVLW 85H          ;load WREG with value 85H
MOVWF 13H
MOVLW 3FH          ;load WREG with value 3FH
MOVWF 14H
MOVLW 63H          ;load WREG with value 63H
MOVWF 15H
MOVLW 12H          ;load WREG with value 12H
MOVWF 16H
```

Solution:

After the execution of `MOVWF 12H` fileReg RAM location 12H has value 99H;
After the execution of `MOVWF 13H` fileReg RAM location 13H has value 85H;
After the execution of `MOVWF 14H` fileReg RAM location 14H has value 3FH;
After the execution of `MOVWF 15H` fileReg RAM location 15H has value 63H;
And so on, as shown in the chart.

Address	Data
012	99
013	85
014	3F
015	63
016	12

Using Instructions with Default Access Bank

- **ADDWF Instruction**
  - Adds the contents of WREG register with fileReg and places the result in specified destination.
  - Format:

ADDWF fileReg, D

fileReg = any location of GP-RAM or any SFR

D = 0 or W for WREG

D = 1 or F for fileReg

  - If D is not specified, then by default the result is placed in **fileReg**

The following will first put value 22H into GP RAM locations 5, 6, and 7, then add them together and put the result in WREG:

```
MOVLW 22H    ;WREG = 22H
MOVWF 5H     ;move(copy) WREG contents to location 5H
MOVWF 6H     ;move(copy) WREG contents to location 6H
MOVWF 7H     ;move(copy) WREG contents to location 7H
ADDWF 5H, 0 ;add W and loc 5, result in WREG (W = 44H)
ADDWF 6H, 0 ;add W and loc 6, result in WREG (W = 66H)
ADDWF 7H, 0 ;add W and loc 7, result in WREG (W = 88H)
```

Address	Data
005	22
006	22
007	22

Address	Data
005	22
006	22
007	22

Now look at the same program where the result is put into file register location 7:

```
MOVLW 22H    ;WREG = 22H
MOVWF 5H     ;move (copy) WREG contents to location 5H
MOVWF 6H     ;move (copy) WREG contents to location 6H
MOVWF 7H     ;move (copy) WREG contents to location 7H
ADDWF 5, 0   ;add W and loc 5, result in WREG (W = 44H)
ADDWF 6, 0   ;add W and loc 6, result in WREG (W = 66H)
ADDWF 7, 1   ;add W and loc 7, result in location 7H
             ;now location 7 has 88H and WREG = 66H
```

Address	Data
005	22
006	22
007	22

Address	Data
005	22
006	22
007	88

The same program with W & F, much easier

```
MOVLW 22H    ;WREG = 22H
MOVWF 5H     ;move (copy) WREG contents to location 5H
MOVWF 6H     ;move (copy) WREG contents to location 6H
MOVWF 7H     ;move (copy) WREG contents to location 7H
ADDWF 5H,W   ;add W and loc 5, result in WREG (W = 44H)
ADDWF 6H,W   ;add W and loc 6, result in WREG (W = 66H)
ADDWF 7H,F   ;add W and loc 7, result in location 7
             ;now location 7 has 88H and WREG = 66H
```

Address	Data
005	22
006	22
007	22

Address	Data
005	22
006	22
007	88

WREG, fileReg & ALU in PIC18

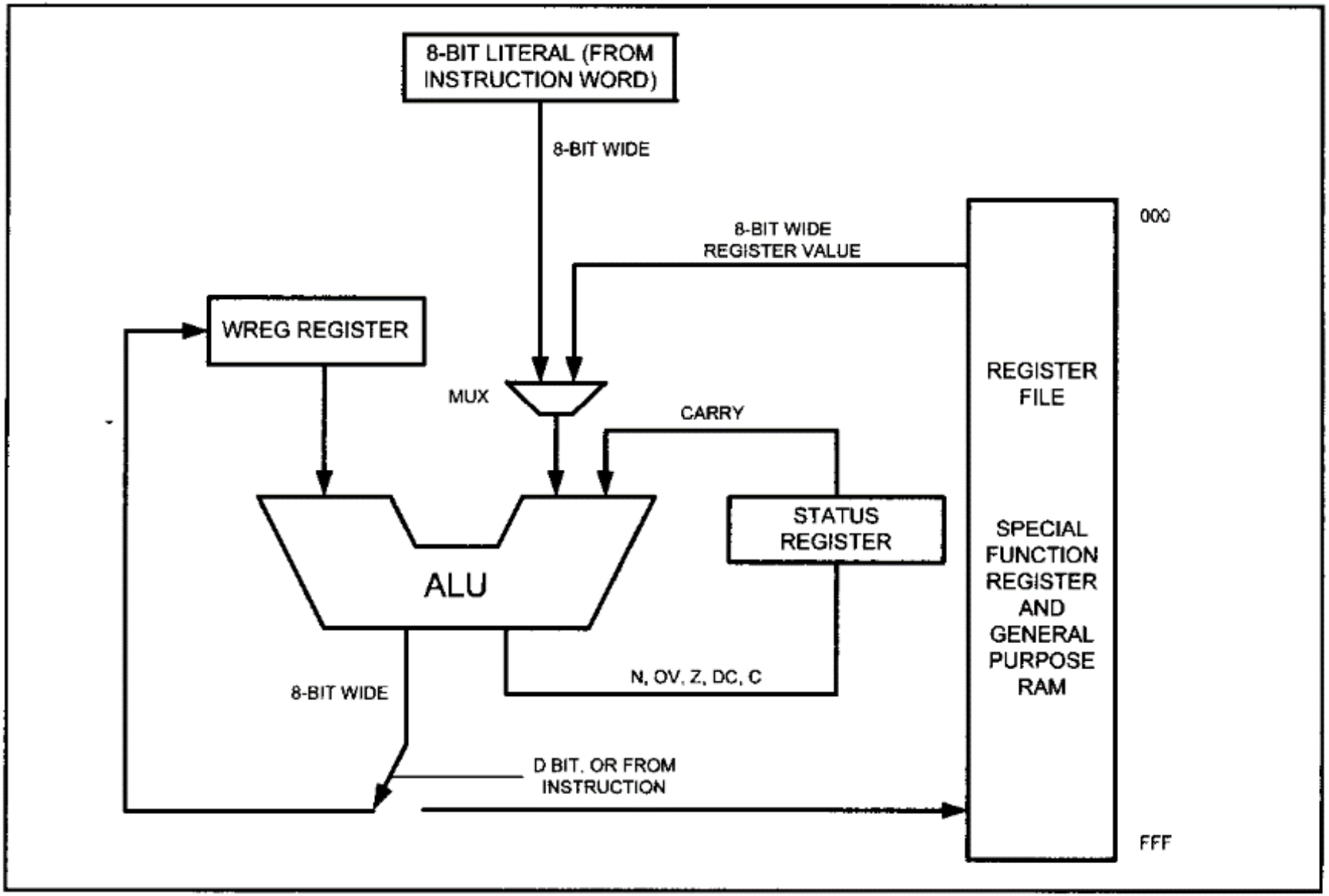


Figure 2-5. WREG, fileReg, and ALU in PIC18

ALU Instructions using both WREG & fileReg

Instruction		
ADDWF	fileReg, d	ADD WREG and fileReg
ADDWFC	fileReg, d	ADD WREG and fileReg with Carry
ANDWF	fileReg, d	AND WREG with fileReg
IORWF	fileReg, d	OR WREG with fileReg
SUBFWB	fileReg, d	Subtract fileReg from WREG with borrow
SUBWF	fileReg, d	Subtract WREG from fileReg
SUBWFB	fileReg, d	Subtract WREG from fileReg with borrow
XORWF	fileReg, d	Exclusive-OR WREG with fileReg

File Register Instructions using WREG or fileReg as destination

Instruction		
COMF	fileReg, d	Complement fileReg
DECF	fileReg, d	Decrement fileReg
DECFSZ	fileReg, d	Decrement fileReg and skip if zero
DECFSNZ	fileReg, d	Decrement fileReg and skip if not zero
INCF	fileReg, d	Increment fileReg
INCFSZ	fileReg, d	Increment fileReg and skip if zero
INCSNZ	fileReg, d	Increment fileReg and skip if not zero
MOVF	fileReg, d	Move fileReg
NEGF	fileReg, d	Negative fileReg
RLCF	fileReg, d	Rotate left fileReg through carry
RLNCF	fileReg, d	Rotate left fileReg (No carry)
RRCF	fileReg, d	Rotate right fileReg through carry
RRNCF	fileReg, d	Rotate right fileReg (No carry)
SWAPF	fileReg, d	Swap nibbles in fileReg
BTG	fileReg, d	Bit Toggle fileReg

Using Instructions with Default Access Bank

- **COMF Instruction**
  - Complements the content of **fileReg** and places the result in WREG or fileReg
  - Format:

COMF fileReg, D

fileReg = any location of GP-RAM or any SFR

D = 0 or W for WREG

D = 1 or F for fileReg

Code to toggle PORTB between values of 55H and AAH

```
MOVLW 55H           ;WREG = 55h
MOVWF PORTB         ;Move WREG to Port B SFR (PB = 55h)
COMF  PORTB, F      ;complement Port B (PB = AAh)
```

Using Instructions with Default Access Bank

- COMF

Example 2-4

Write a simple program to toggle the SFR of PORT B continuously forever.

Solution:

```
        MOVLW  55H          ;WREG = 55h
        MOVWF  PORTB        ;move WREG to Port B SFR (PB = 55h)
B1      COMF   PORTB, F     ;complement Port B and place it in Port B
        GOTO   B1           ;repeat forever (See Chapter 3 for GOTO)
```

Using Instructions with Default Access Bank

- **DECF Instruction**
  - Decreases the content of **fileReg** by 1 and places the result in WREG or fileReg
  - Format:

DECF fileReg, D

fileReg = any location of GP-RAM or any SFR

D = 0 or W for WREG

D = 1 or F for fileReg

  - Frequently used in loops, conditional branches and pointer manipulation.

Using Instructions with Default Access Bank

- DECF

```
MOVLW 3              ;WREG = 3
MOVWF 20H            ;move WREG to loc 20H (loc 20H = 3)
DECF  0x20, F        ;loc 20H has 2
DECF  0x20, F        ;loc 20H has 1
DECF  0x20, F        ;loc 20H has 0 and WREG = 3
```

Now, contrast the above code with the following:

```
MOVLW 3              ;WREG = 3
MOVWF 20H            ;move WREG to loc 20H (loc 20H = 3)
DECF  0x20, W        ;loc 20H has 3 (WREG = 2)
DECF  0x20, W        ;loc 20H has 3 (WREG = 2)
DECF  0x20, W        ;loc 20H has 3 (WREG = 2)
```

Using Instructions with Default Access Bank

- **MOVF Instruction**
  - Moves the content of **fileReg** to specified destination.
  - Format:

MOVF fileReg, D

fileReg = any location of GP-RAM or any SFR

D = 0 or W for WREG

D = 1 or F for fileReg

  - In this case D=1 or F will copy the content into the same location i.e source copied at source location. This is often done to affect the flags of status registers.

Using Instructions with Default Access Bank

Example 2-5

Write a program to get data from the SFRs of Port B and send it to the SFRs of PORT C continuously.

Solution:

```
AGAIN MOVF   PORTB, W     ;bring data from PortB into WREG
      MOVWF  PORTC        ;send it to Port C
      GOTO   AGAIN        ;keep doing it forever
```

Using Instructions with Default Access Bank

Example 2-6

Write a program to get data from the SFRs of Port B. Add the value 5 to it and send it to the SFRs of Port C.

Solution:

```
MOVF   PORTB,W      ;bring data from Port B into WREG
ADDLW  05H          ;add 5 to WREG
MOVWF  PORTC        ;copy WREG to Port C
```

Using Instructions with Default Access Bank

- **MOVFF Instruction**
  - Moves the content of one **fileReg** location to another **fileReg** location.
  - **Format:**

  - **MOVFF source, destination**

    MOVFF PORTB, PORTC

    MOVFF 0x20, 0x30

Using Instructions with Default Access Bank

Example 2-7

Write a program to get data from the SFRs of Port B and send it to the SFRs of PORT C continuously using MOVFF. Compare this to Example 2-5 and explain the difference.

Solution:

```
AGAIN MOVFF PORTB, PORTC      ;copy data from Port B to Port C
      GOTO AGAIN              ;keep doing it forever
```

In Example 2-5 we have:

```
AGAIN MOVF  PORTB, W          ;bring data from Port B into WREG
      MOVWF PORTC             ;send it to Port C
      GOTO  AGAIN             ;keep doing it forever
```

Using **MOVFF** we simply copy data from one location to another location. But when we use WREG we can perform arithmetic and logic operations on data before it is moved.

PIC Status Register

PIC Status Register

- The status register is an 8-bit register.
- Also known as **flag register**
- First 5 bits (starting from LSB) are used. Rest 3 bits are unimplemented.
- These 5 flags are also know as **conditional flags**, meaning that these flags indicate some conditions that result after an instruction execution.

PIC Status Register

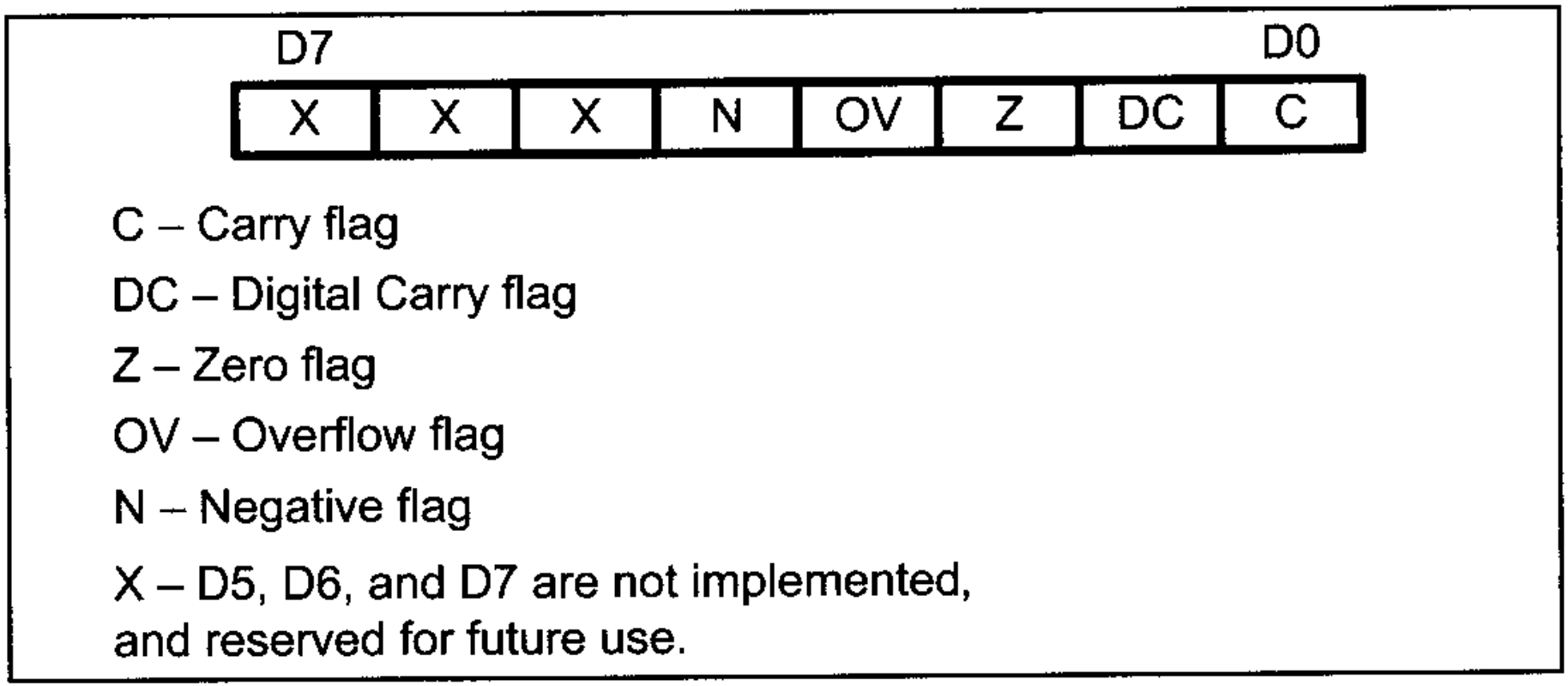


Figure 2-7. Bits of Status Register

PIC Status Register

- C:
  - Carry Flag
  - Flag is set whenever there is a carry out from D7 bit.
  - Affected by 8-bit addition or subtraction
- DC:
  - Digital Carry Flag
  - Set when there is a carry from D3 to D4 i.e lower nibble to higher nibble.
  - Used in BCD arithmetic
- Z:
  - Zero flag
  - If result of any arithmetic or logical instruction is zero then Z=1 else Z=0.

PIC Status Register

- OV:
  - Overflow Flag
  - Flag is set whenever the result of signed number operation is too large.
  - Used to detect errors in signed arithmetic operations.

- N:
  - Negative Flag
  - Represents a negative number.
  - Is set when result of an arithmetic operation results in a negative number.

PIC Status Register

Example 2-8

Show the status of the C, DC, and Z flags after the addition of 38H and 2FH in the following instructions:

```
MOVLW 38H
ADDLW 2FH          ;add 2FH to WREG
```

Solution:

38H	0011 1000	
+ 2FH	0010 1111	
67H	0110 0111	WREG = 67H

C = 0 because there is no carry beyond the D7 bit.
DC = 1 because there is a carry from the D3 to the D4 bit.
Z = 0 because the WREG has a value other than 0 after the addition.

PIC Status Register

Example 2-9

Show the status of the C, DC, and Z flags after the addition of 9CH and 64H in the following instructions:

```
MOVLW 9CH
ADDLW 64H          ;add 64H to WREG
```

Solution:

9CH	1001 1100	
+ 64H	0110 0100	
100H	0000 0000	WREG = 00

C = 1 because there is a carry beyond the D7 bit.
DC = 1 because there is a carry from the D3 to the D4 bit.
Z = 1 because the WREG has a value 0 in it after the addition.

PIC Status Register

Example 2-10

Show the status of the C, DC, and Z flags after the addition of 88H and 93H in the following instructions:

```
MOVLW 88H
ADDLW 93H          ;add 93H to WREG
```

Solution:

88H	1000 1000	
+ 93H	1001 0011	
11BH	0001 1011	WREG = 1BH

C = 1 because there is a carry beyond the D7 bit.
DC = 0 because there is no carry from the D3 to the D4 bit.
Z = 0 because the WREG has a value other than 0 after the addition.

Flag Bits and Decision Making

Table 2-5: PIC18 Branch (Jump) Instructions Using Flag Bits

Instruction	Action
BC	Branch if C = 1
BNC	Branch if C ≠ 0
BZ	Branch if Z = 1
BNZ	Branch if Z ≠ 0
BN	Branch if N = 1
BNC	Branch if N ≠ 0
BOV	Branch if OV = 1
BNOV	Branch if OV ≠ 0

Will be discussed in detail in chapter 3