The University of Texas at Arlington

Lecture 2
PIC Overview

CSE 3442/5442

Embedded Systems I

Based heavily on slides by Dr. Gergely Záruba and Dr. Roger Walker

Overview of PIC18 Family

- 1989, Microchip Technology introduced 8-bit microcontroller called PIC (Peripheral Interface Controller) – see web site at www.Microchip.com

- PIC18F452 Data sheets

http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010296

Lab Module and Online Book

- **Book that describes the PIC module used in the lab and other information on the PIC18: link on the class website.**
- **Kit used in lab: QwikFlash Development Kit No. 3 (Expanded Kit with QwikBreadboard 400 and Stand)** http://www.microdesignsinc.com/qwikflash/index.html

8-bit Microchip Families

- Microchip is currently the number one supplier of 8-bit microcontrollers.
- PIC families include 10xxx, 12xxx,14xxx, 16xxx, 17xxx, and 18xxx.
- PIC families are not upward compatible. All 8-bit processors.
- 12xxx/16xxx have 12-bit & 14-bit instructions
- PIC18xxx have 16-bit instructions

PIC RAM/ROM Components

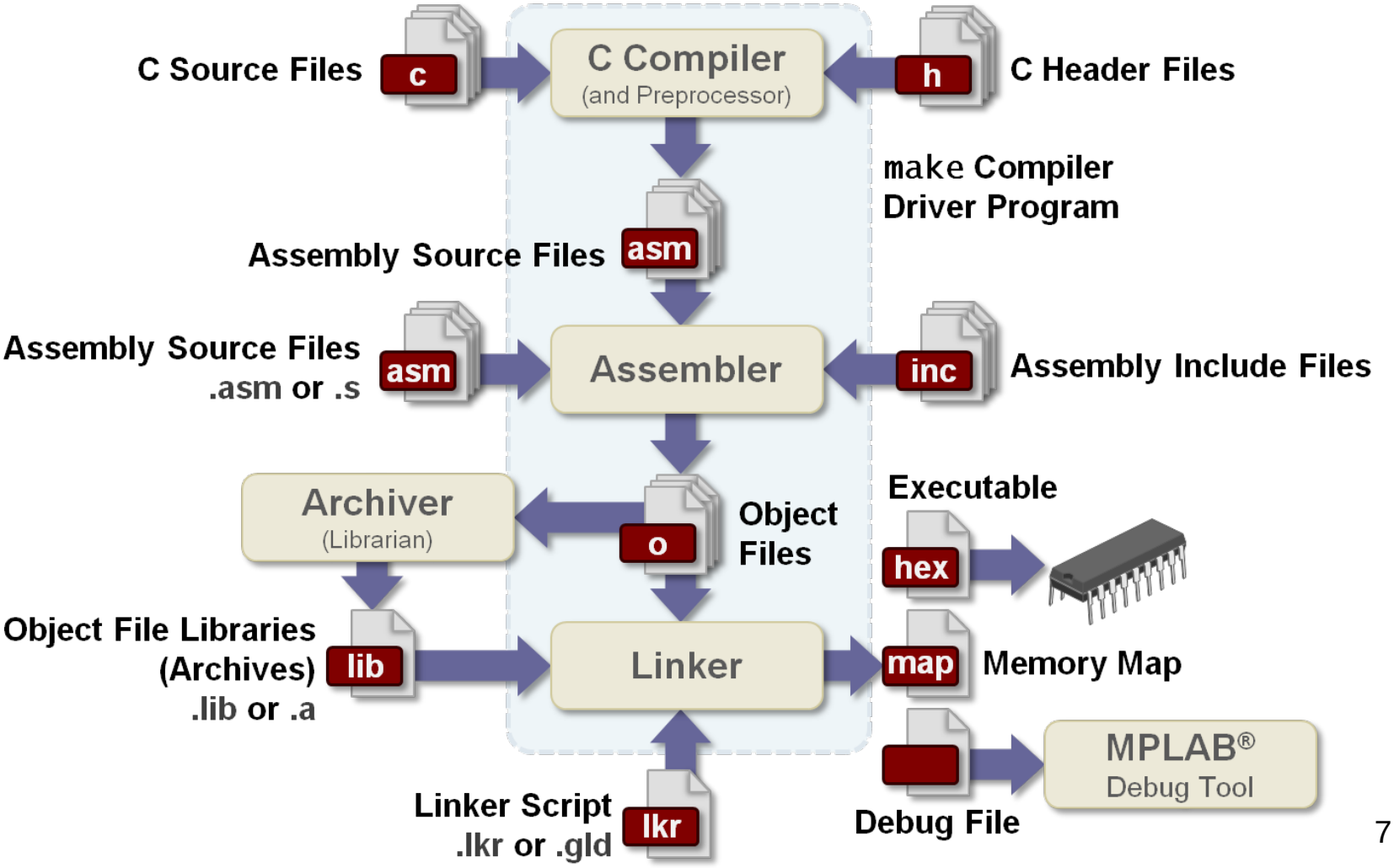
- **ROM – program or code ROM** (PIC18 can have up to 2 megabytes (2M) of ROM
- UV-EPROM for program memory– must have UV-EPROM eraser/programmer (burner ~20 minutes to erase)
- **Flash memory** PIC18**F**458 use for program development (EEPROM – electrically erasable PROM) – use PICkit 3 (from Microchip) using USB and PC.

PIC Components cont.

- **OTP** version of PIC – **one time programmable** (ROM) versions available from Microchip (PIC16C452) – use for final production version – programmed at Microchip

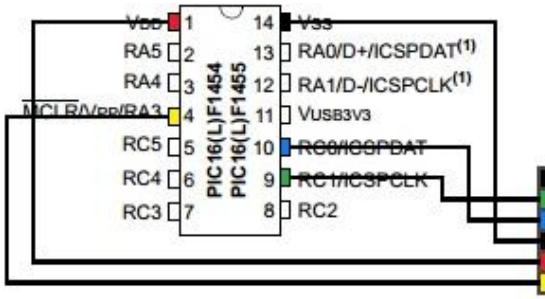
- Masked versions of PIC – Provides a means of chip fabrication with program built in – minimum order but cost per IC cheapest of all methods.

Software Program Flow

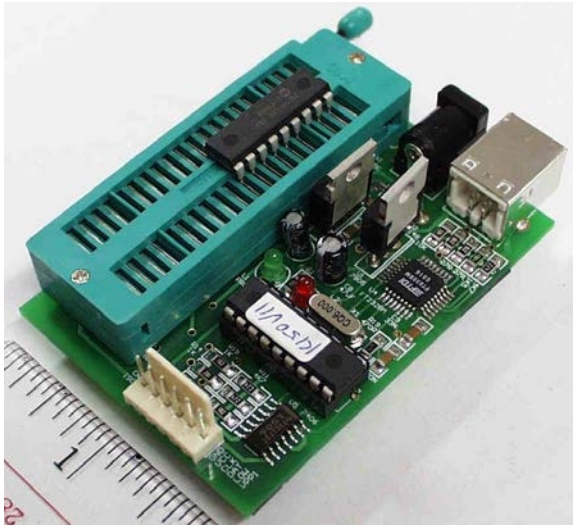


Programming a PIC

- PicKit 3 (ICSP)



- Socket Programmer



Software Program Flow

CSE@UTA

```
67
68      //Main routine, forever in an infin
69      while(1)
70      {
71          //Your main code goes here
72          Print_To_LCD(Str_1);
73          Print_To_LCD(Str_2);
74
75          Toggle_LEDs();
76
77          Str_2[8]++;
78
79          if(Str_2[8] > '9')
80              Str_2[8] = '0';
81      }
```

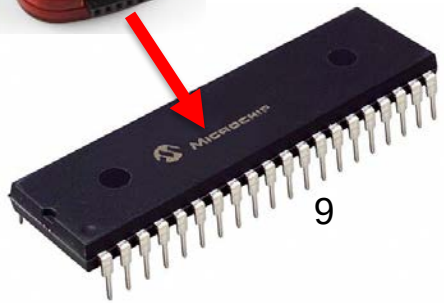
.c

```
 movlw   244
 movwf   ??_Initialize_LCD& (0+255),c
 movlw   168
u57:
 dw  65535   ; errata NOP
 decfsz  wreg,f,c
 bra u57
 decfsz  ??_Initialize_LCD& (0+255),f,c
 bra u57
 decfsz  (??_Initialize_LCD+1)& (0+255),
 bra u57
 nop2
```

.asm

```
:100760000FEC03F01200FFFFFFFF9EEC03F00B0EF7
:10077000156E060E166E57EC03F0010E156E060E82
:10078000166E57EC03F0FFFFFED7C0202020207C
:1007900020202000802020202020202000FFFF7B
:020000040020DA
:08000000FFFFFFFFFFFFFF00
:020000040030CA
:0E000000FF220D0EFF0181FF0FC00FE00F4029
:00000001FF
```

.hex

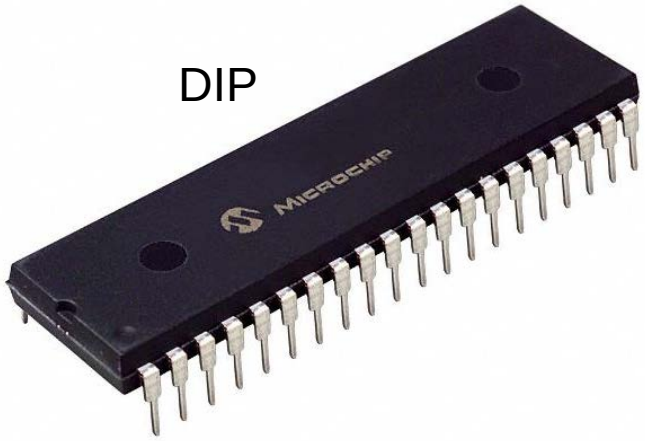


PIC18 family

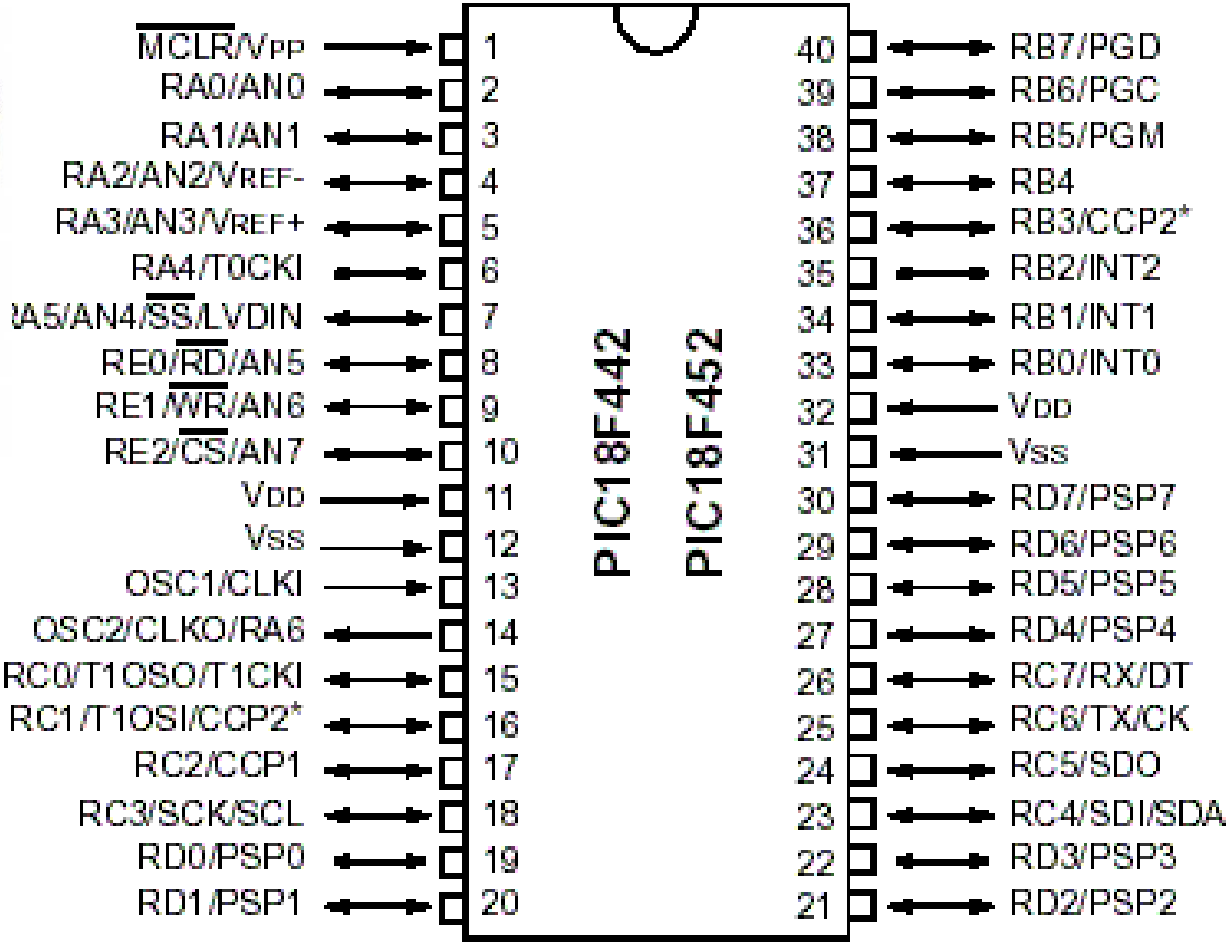
- PIC18 one of the higher performers of the Microchip's PIC families. There is now both a 32 bit PIC32 family and DSPIC (16 bit) with high performance.
- PIC families come in 18-to 80 pin packages.
- Select family based on performance, footprint, etc., needed, use selection guide: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2661

PIC18F452

DIP



PLCC, SMD, SMT

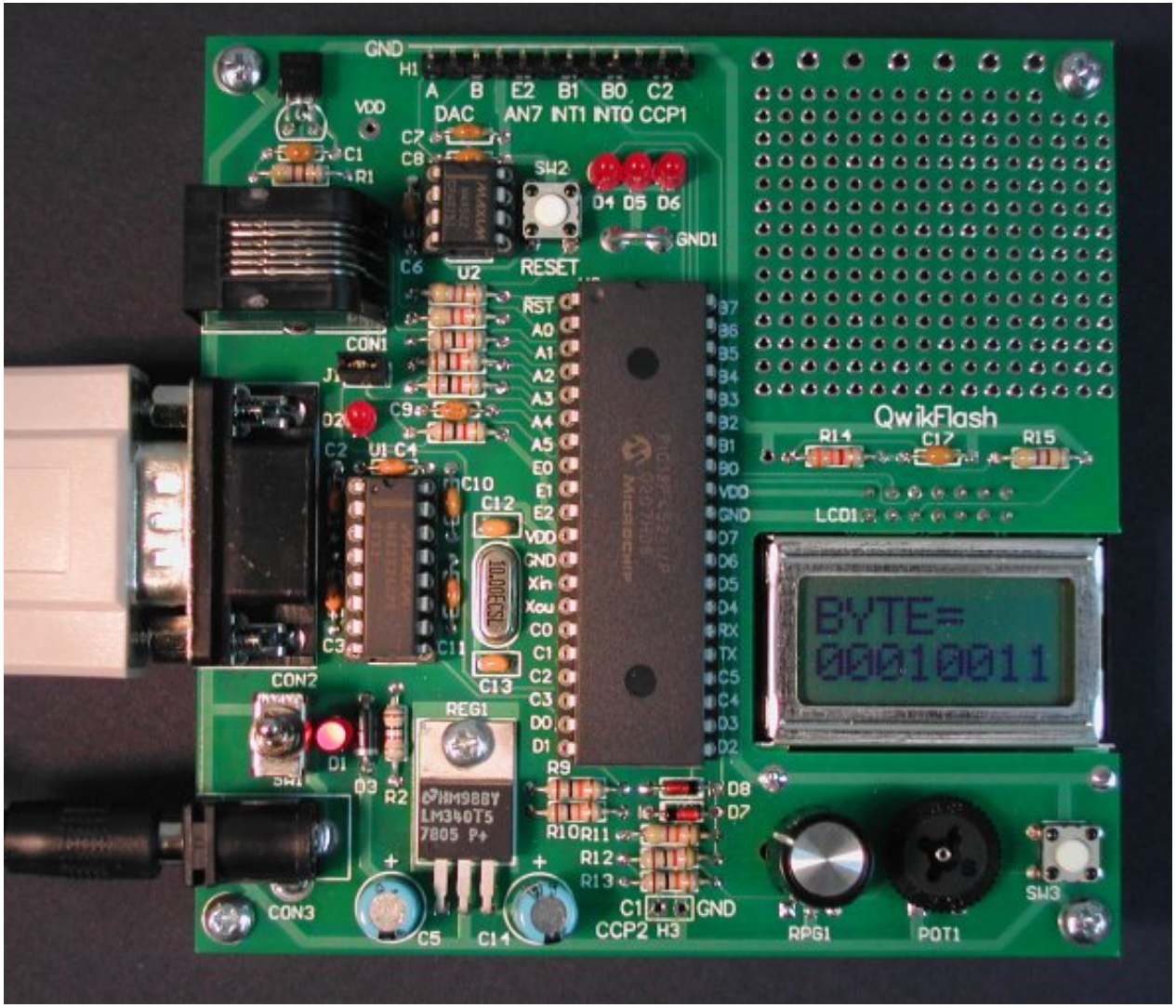


PIC18 features

- RISC architecture
- On-chip program ROM, data RAM, data EEPROM, timers, ADC,USART, and I/O Ports

- ROM, data RAM, data EEPROM, and I/O ports sizes varies within PIC18 family

QwikFlash


QwikFlash
BYTE=
00010011
GND
VDD
RESET
CON1
CON2
CON3
REG1
LCD1
SW1
SW2
SW3
RPG1
POT1
CCP2 H3

QwikFlash

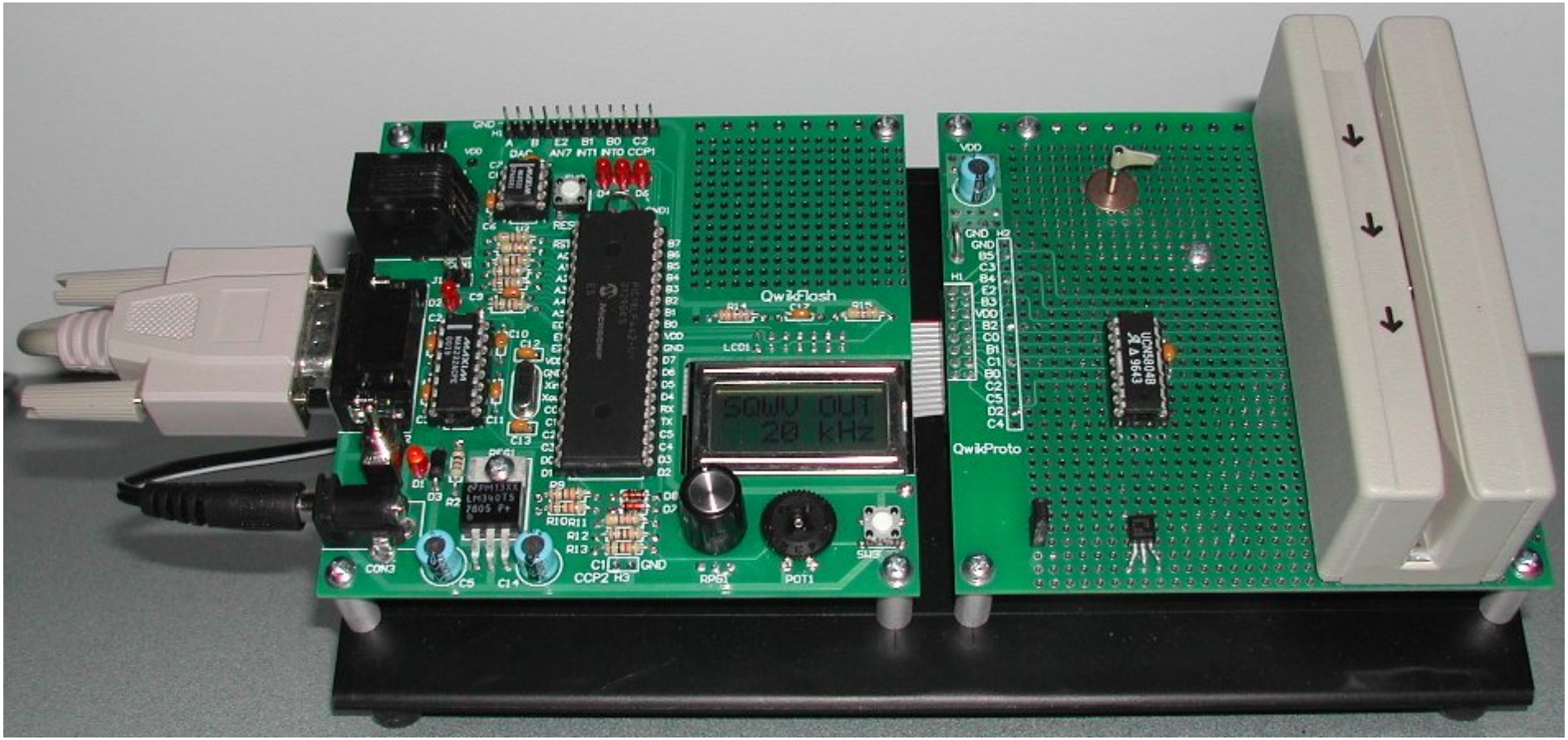


Figure 1-2. Simplified View of a PIC Microcontroller

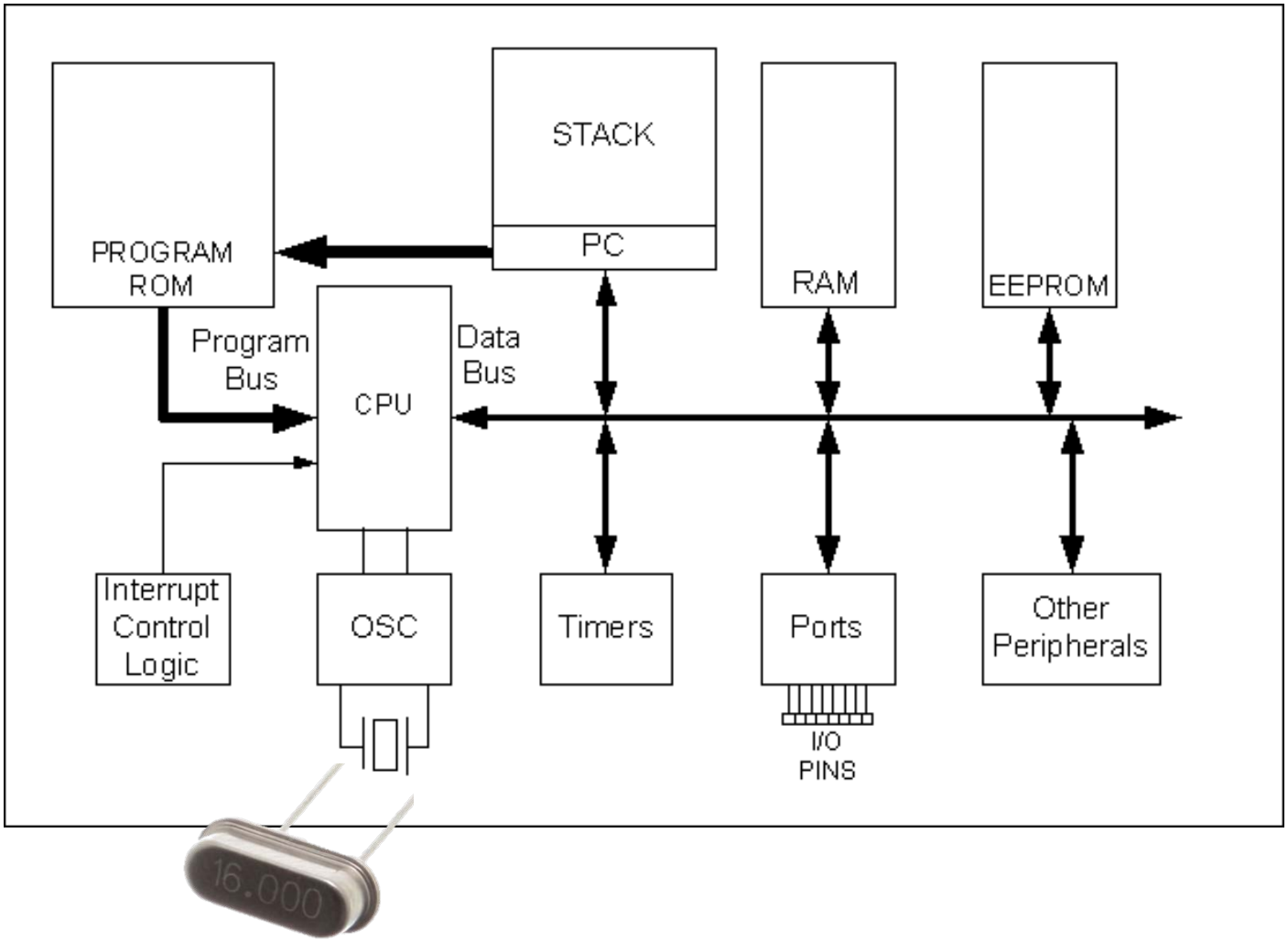


Figure 1-3. PIC18 Block Diagram

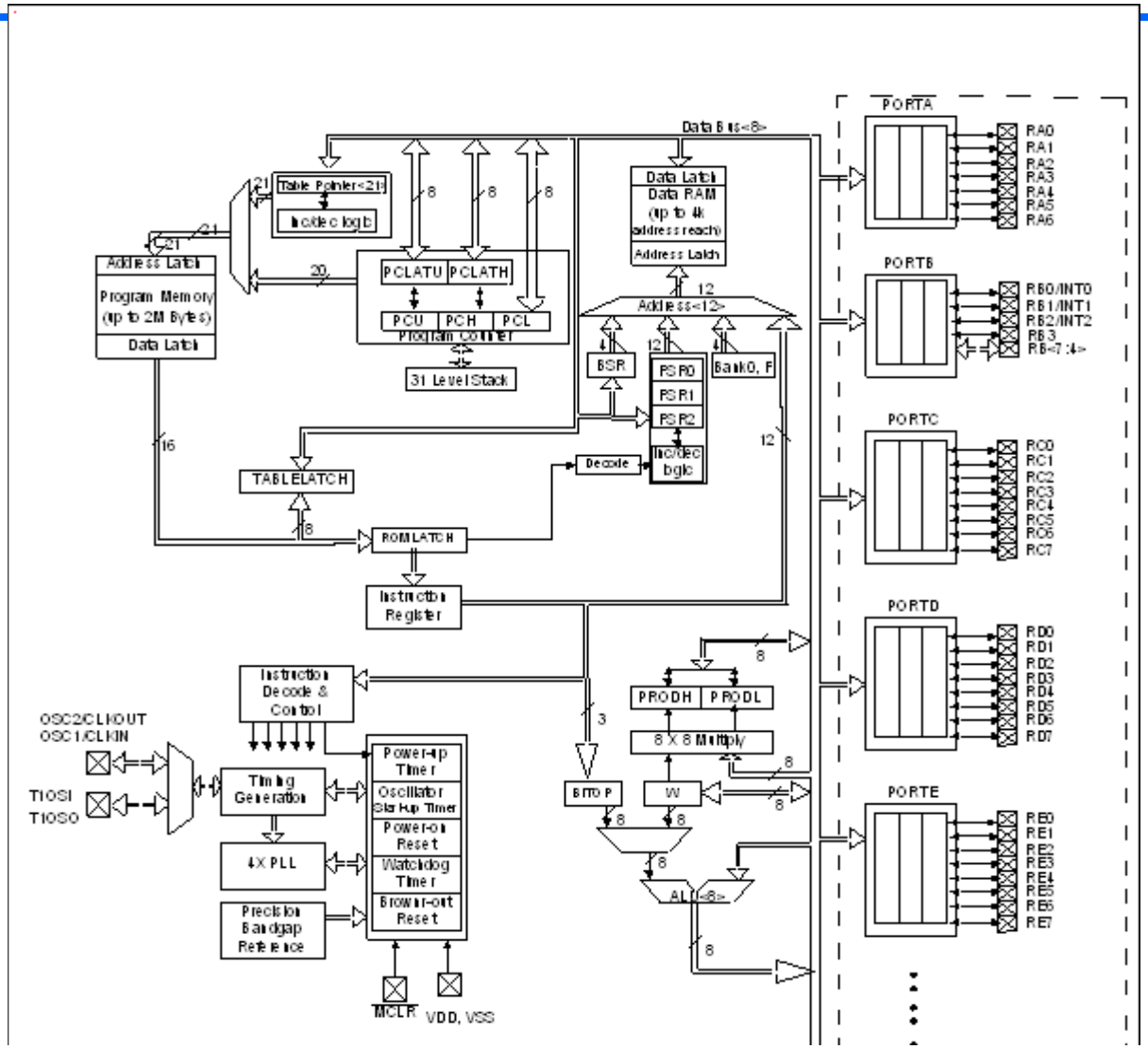
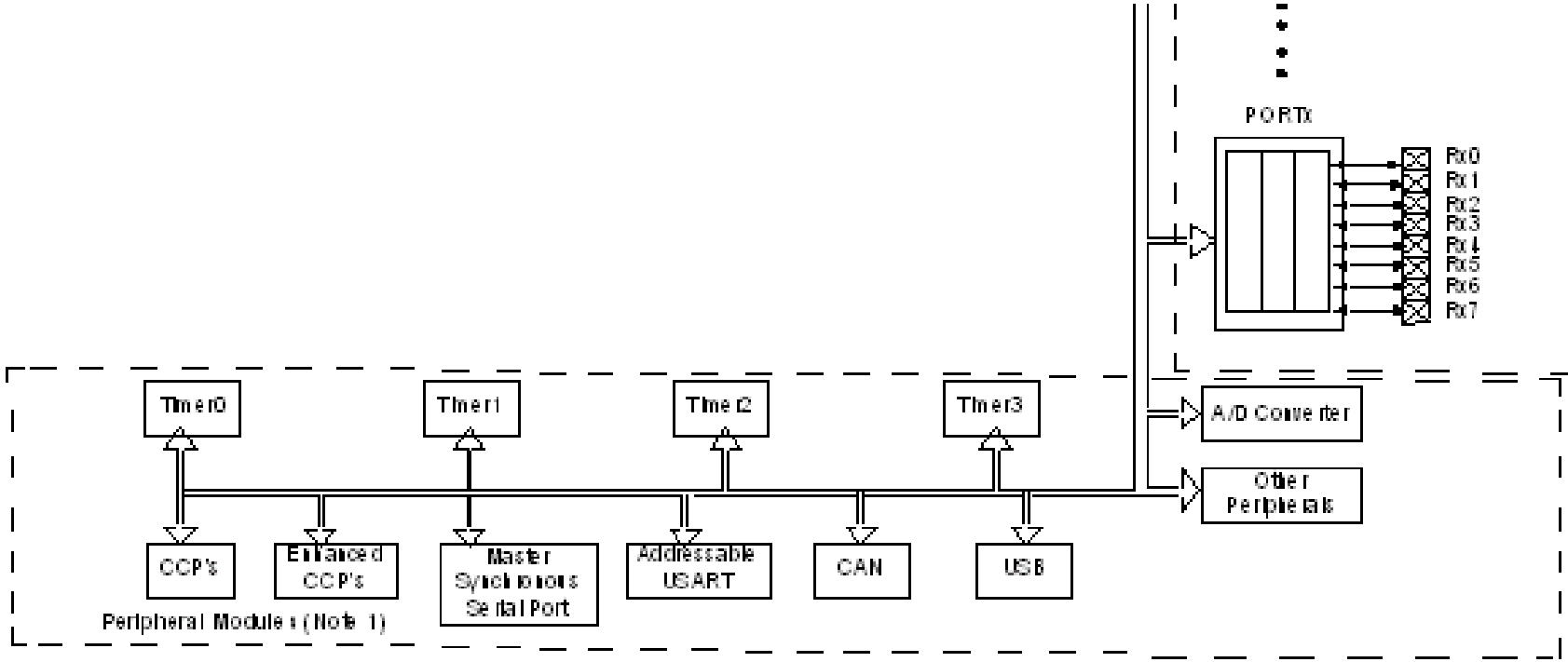


Figure 1-3. PIC18 Block Diagram (continued)



Note 1: Many of the general purpose I/O pins are multiplexed with one or more peripheral module functions. The multiplexing combinations are device dependent.

What is a Register?

- **Register**
  - A place inside the PIC that can be written to, read from, or both (8-bit numbers)

TABLE 9-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

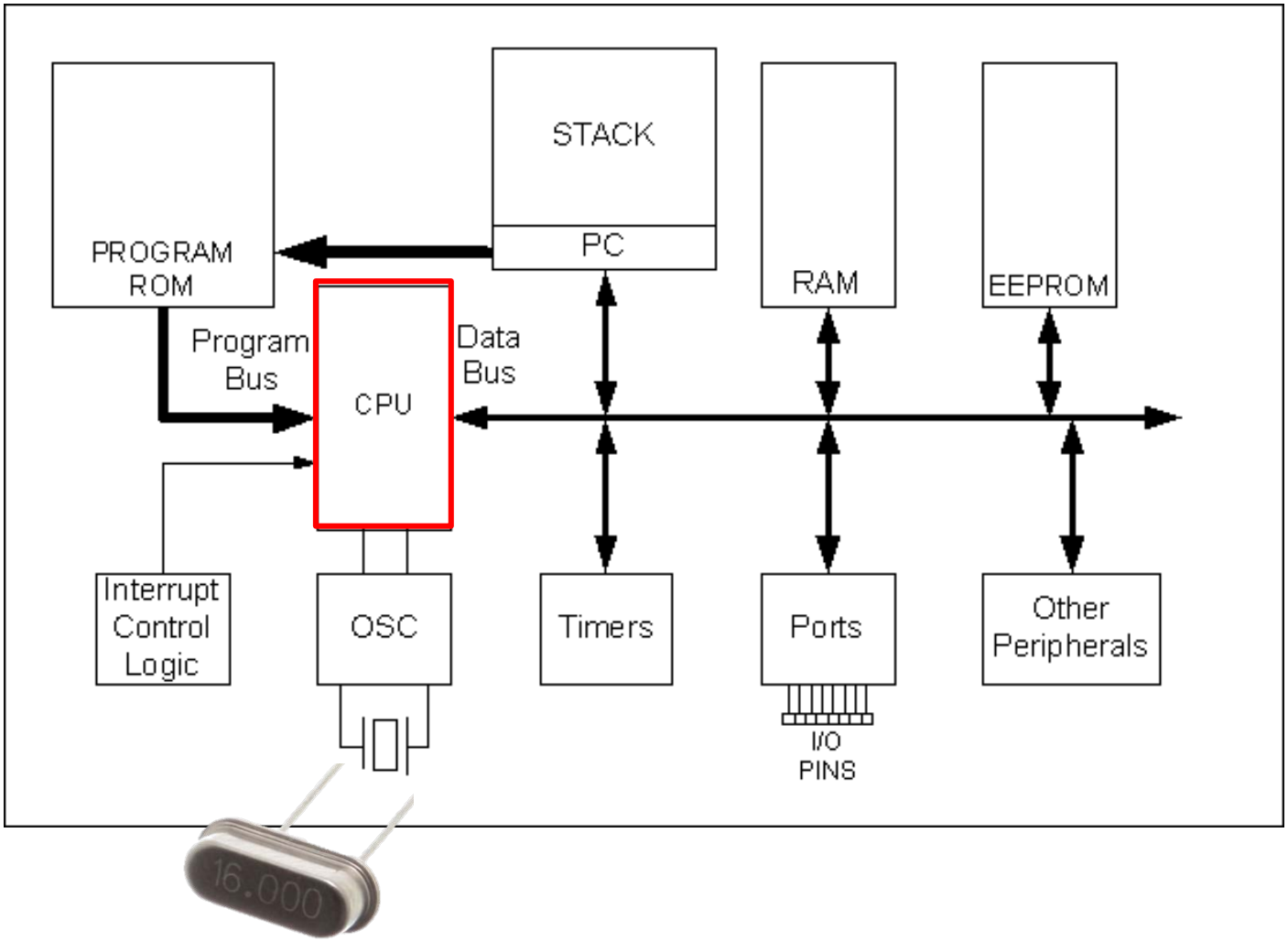
Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on All Other RESETS
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
LATB	LATB Data Output Register								xxxx xxxx	uuuu uuuu
TRISB	PORTB Data Direction Register								1111 1111	1111 1111
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1	1111 -1-1
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	11-0 0-00

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

WREG

- **W**orking **Reg**ister is the same as the **accumulator** in other microprocessors
- Used for all arithmetic/logic instructions
  - Avoids use of main memory
  - Close as possible to the ALU within the CPU
- **Can only hold 0-255 dec (0-FFh)**
  - Truncates larger values and causes warning
    - 1001 1101 1100 = 2,524dec = 9DCh
    - 0000 1101 1100 = 220dec = DCh

WREG is in the CPU



Chapter 2 PIC Architecture & Assembly Language Programming

- WREG – 8 bit register in PIC (Working Register) – used the most

MOVLW K

Move (“MOV”) the number (“L” for “literal”) K (example 0xA), or 10 in decimal) - into the working register (“W”).

MOVLW 0xA

That is, load W with the value 0xA

Note: ‘WREG’ sometimes shortened to ‘W’

Moving to WREG

- **MOVLW K**; move literal value K into WREG
- Once again K is an 8 value 0-255 decimal or 00-FF in hex

- Eg.

  MOVLW 25H; move 25H (0x25) into WREG

  (WREG = 25H)

ALU is in the CPU

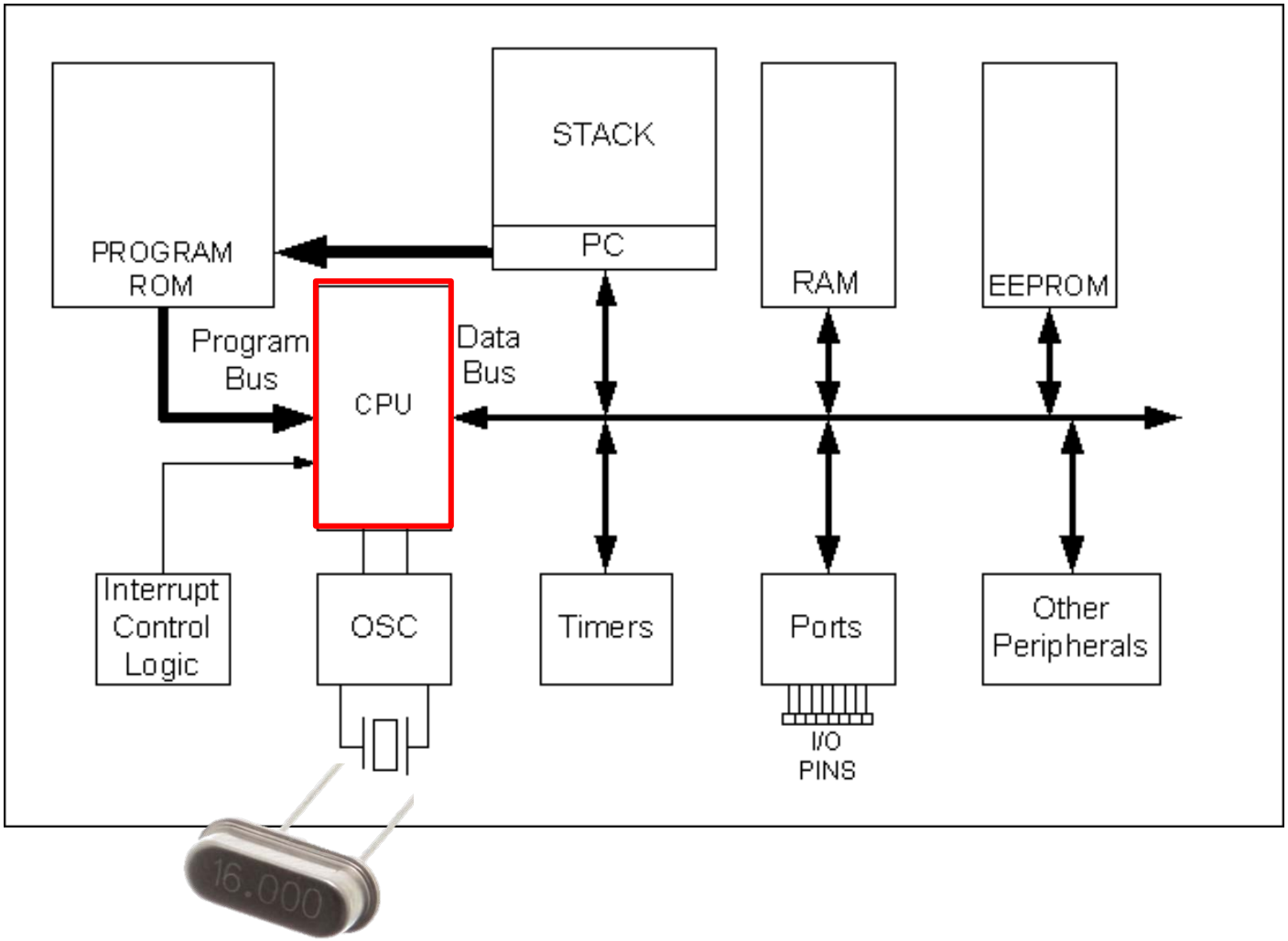
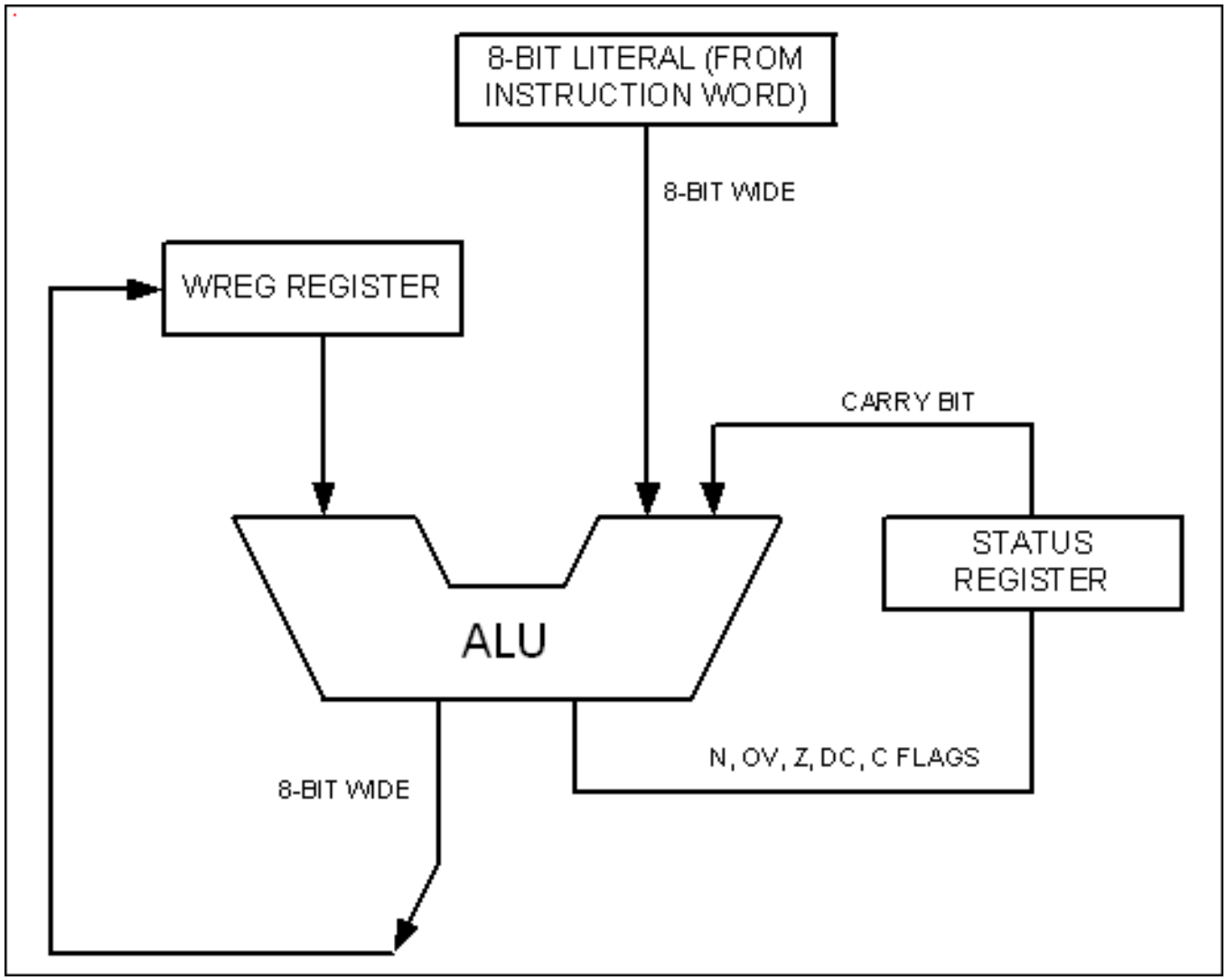


Figure 2-1. PIC WREG and ALU Using Literal Value



Move and Add Instructions

- **MOVLW 12H** ;load value 12H -> WREG
- **ADDLW 16H** ;add 16H to WREG
- **ADDLW 11H** ;add 11H to WREG
- **ADDLW 43H** ;add 43H to WREG

FILE REGISTER

- **File Register = Data Memory (RAM)**
  - Read/write memory used by CPU for data storage
  - Varies from 32 bytes to thousands depending on chip size (family)
  - Can perform arithmetic/logic operations on many locations of File Register data
- Divided into two sections
  - Special Function Registers (**SFR**)
  - General Purpose Registers (**GPR**) or (GP RAM)

File Registers = Data RAM

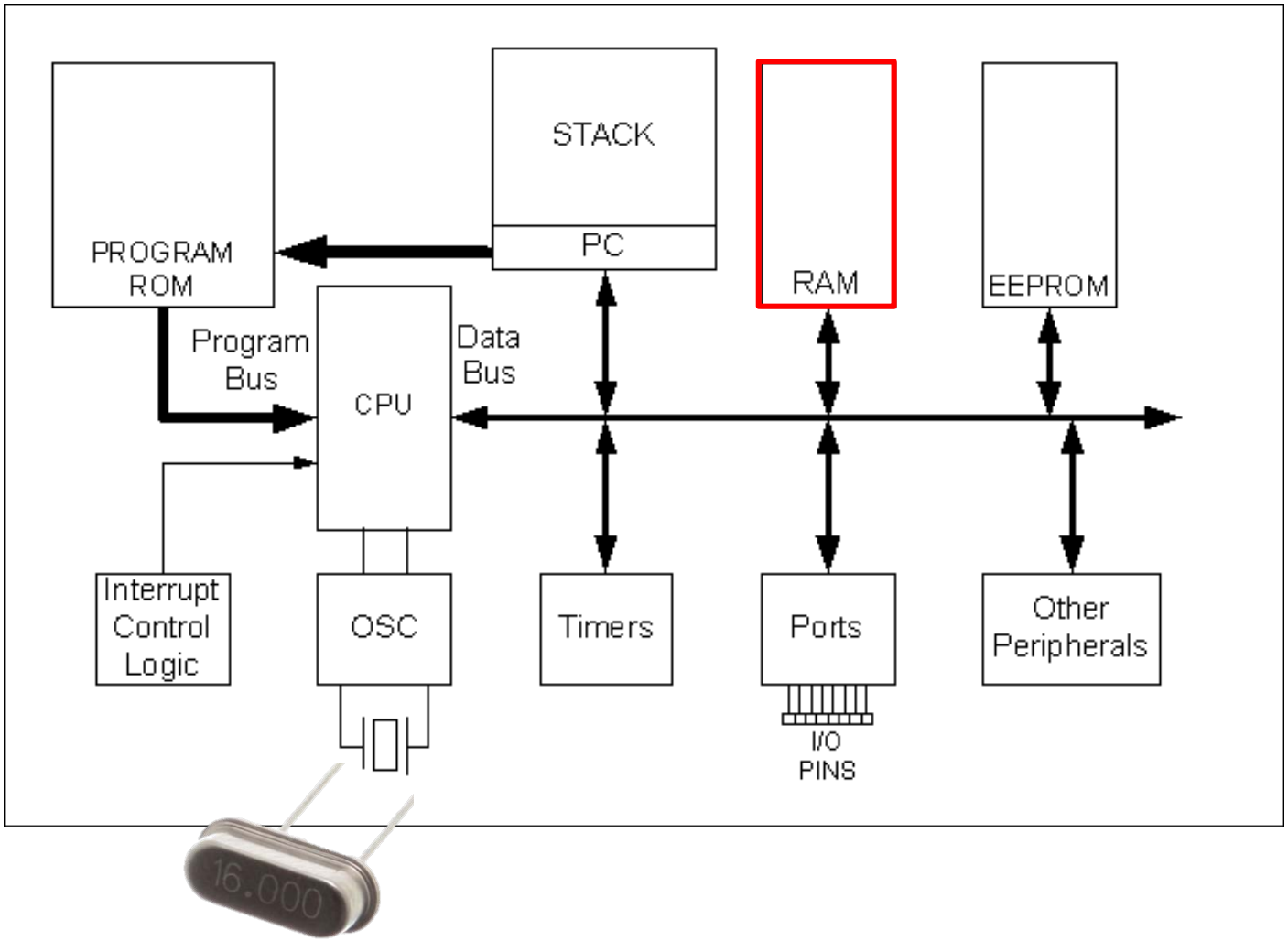


Figure 2-2. File Registers of PIC12, PIC16, and PIC18

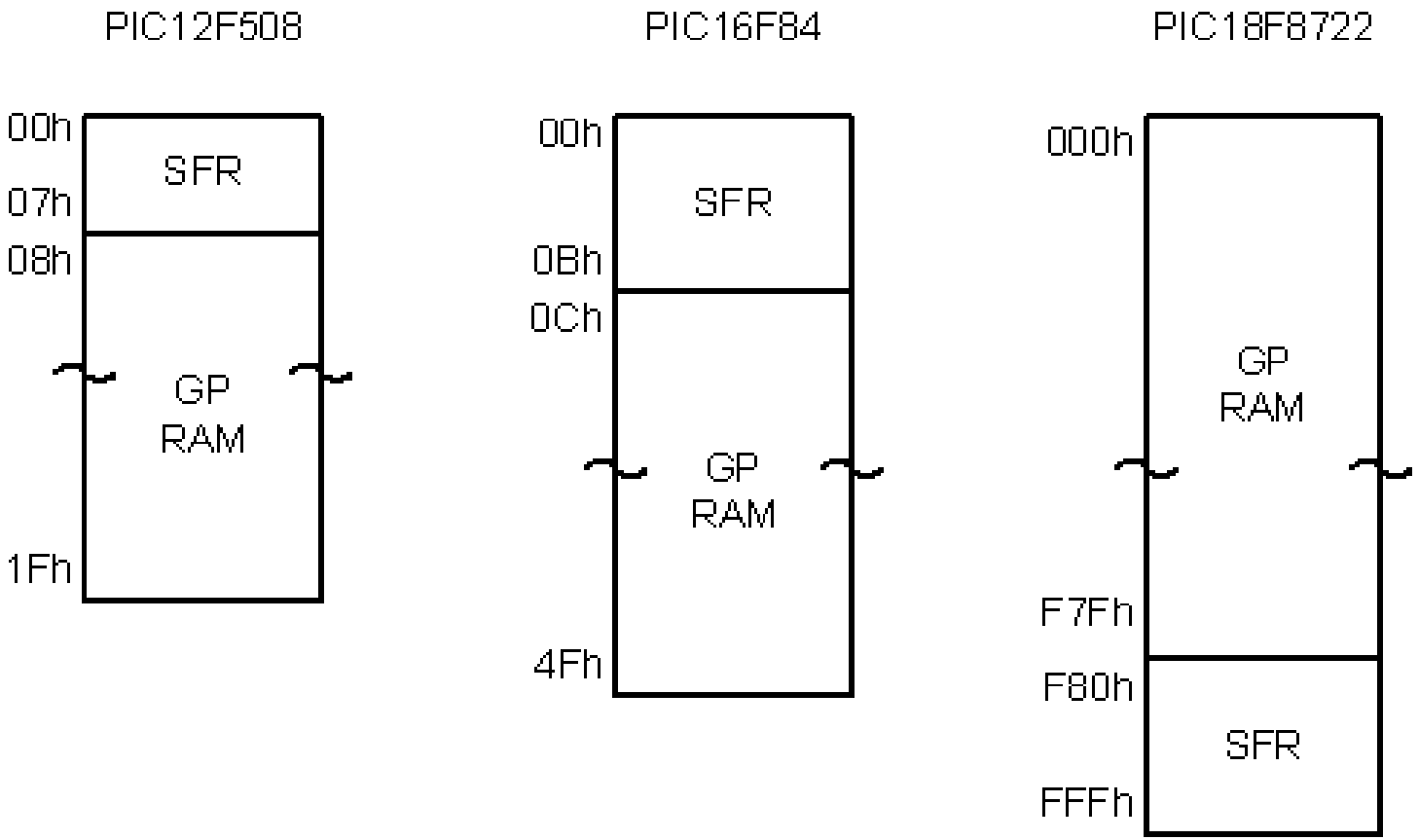
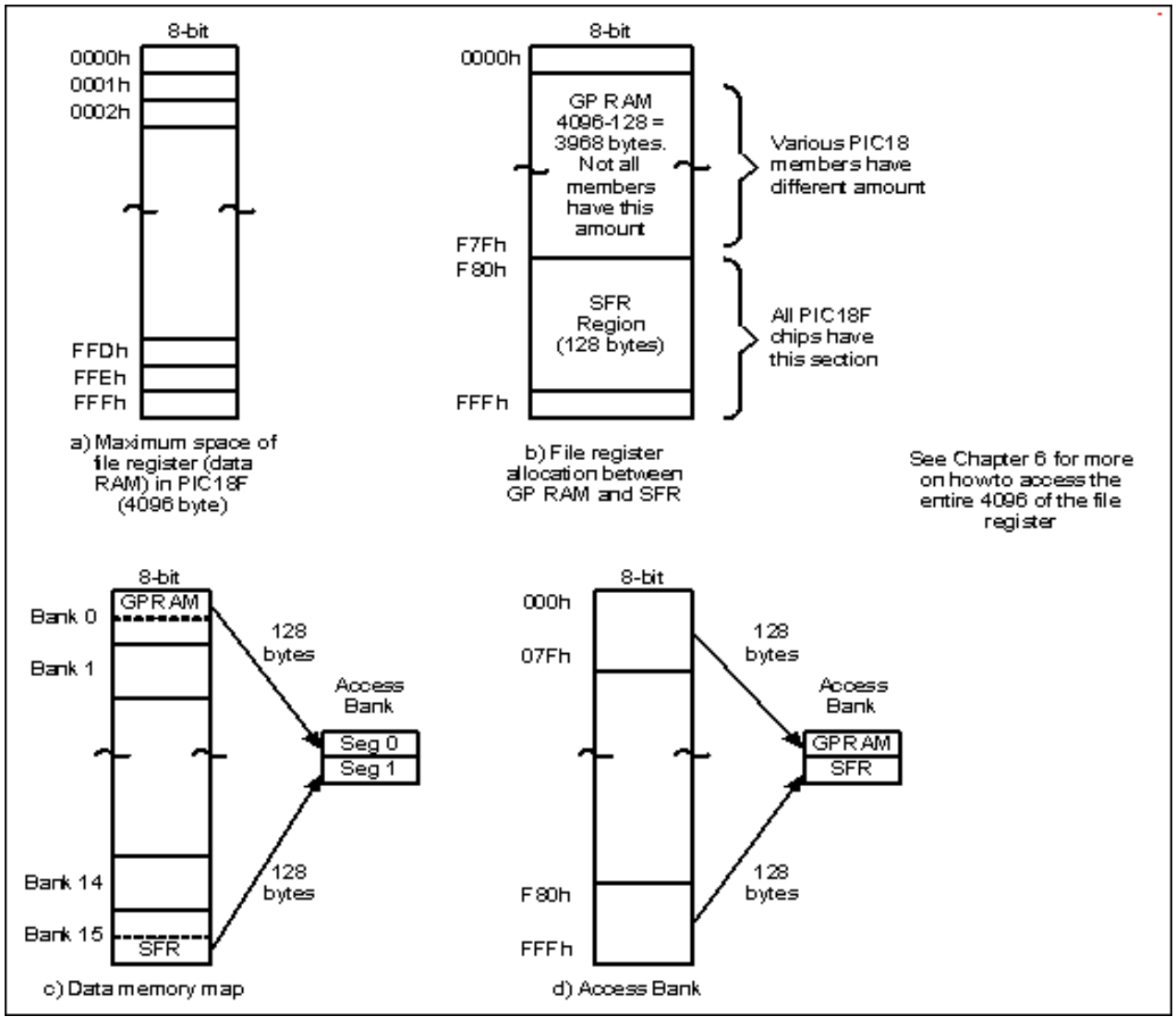


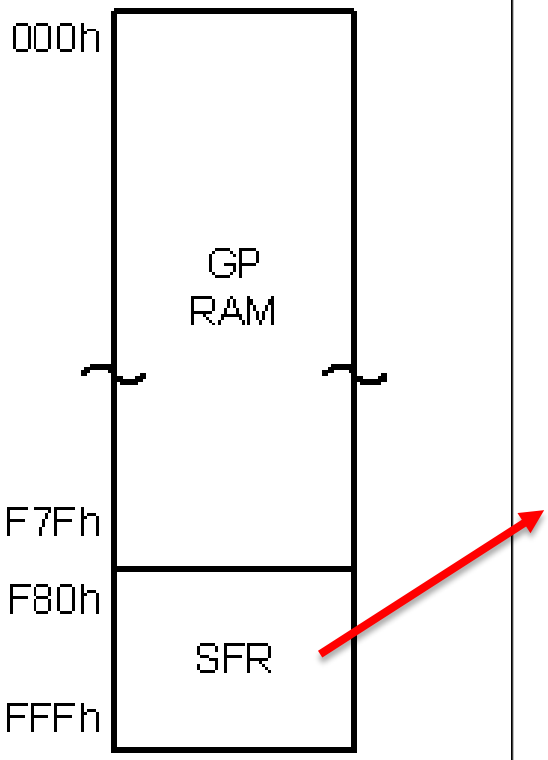
Figure 2-3. File Register for PIC18 Family



FILE REGISTER cont.

- **SFR – special functions** (8-bit wide regs)
  - Functionality of each is fixed in design
  - Usually for all of the peripherals
    - ALU status
    - Timers
      - Ex: the more timers the more SFR in a PIC
    - Serial communications
    - I/O Ports
    - A/D

Special Function Registers of the PIC18 Family



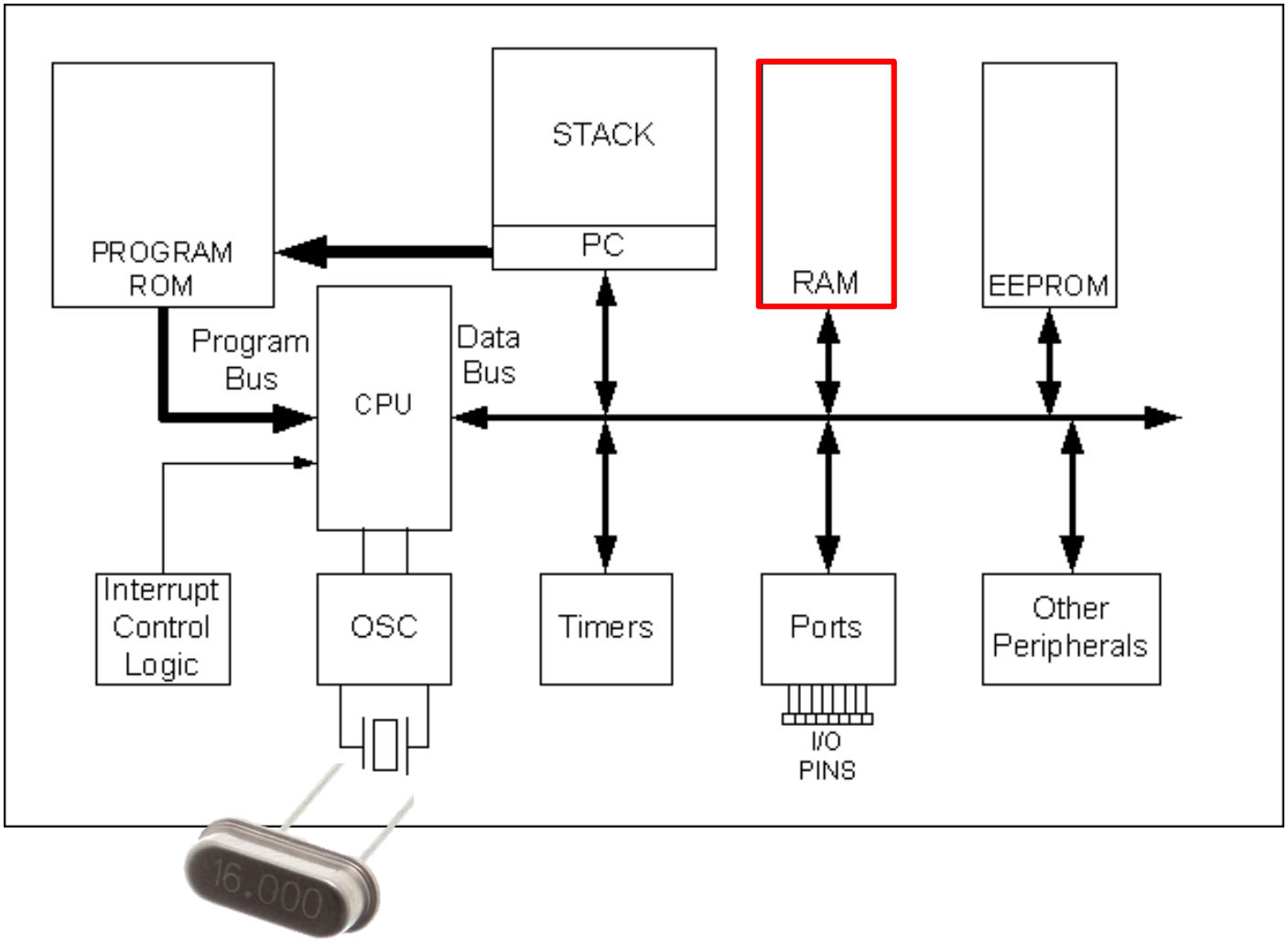
F80h	PORTA	FA0h	PIE2	FC0h	----	FE0h	BSR
F81h	PORTB	FA1h	PIR2	FC1h	ADCON1	FE1h	FSR1L
F82h	PORTC	FA2h	IPR2	FC2h	ADCON0	FE2h	FSR1H
F83h	PORTD	FA3h	----	FC3h	ADRESL	FE3h	PLUSW1 *
F84h	PORTE	FA4h	----	FC4h	ADRESH	FE4h	PREINC1 *
F85h	----	FA5h	----	FC5h	SSPCON2	FE5h	POSTDEC1 *
F86h	----	FA6h	----	FC6h	SSPCON1	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.

FILE REGISTER – cont.

- **GPR** – **general-purpose** (8-bit wide regs)
  - RAM locations used for ANY data storage and scratch pad as long as it's 8 bits (data RAM size same as GFR size)
  - Larger GPR = more difficult to manage in ASM
  - C compilers now handle management/addressing
- GPR ≠ EEPROM
  - GPR used by CPU for "internal" data storage
  - EEPROM considered "add-on" memory

File Registers = Data RAM



File Registers, cont.

- PIC File Register has max 4K or 000-FFFH
  - Addresses (locations) are 12-bit wide
- Divided into 16 banks of 256 bytes
- All PIC's have at least one bank…
  - **Access Bank**
- The access bank divided into 2 sections of 128 bytes, SFR and GPR

PIC18 File Register and Access Bank

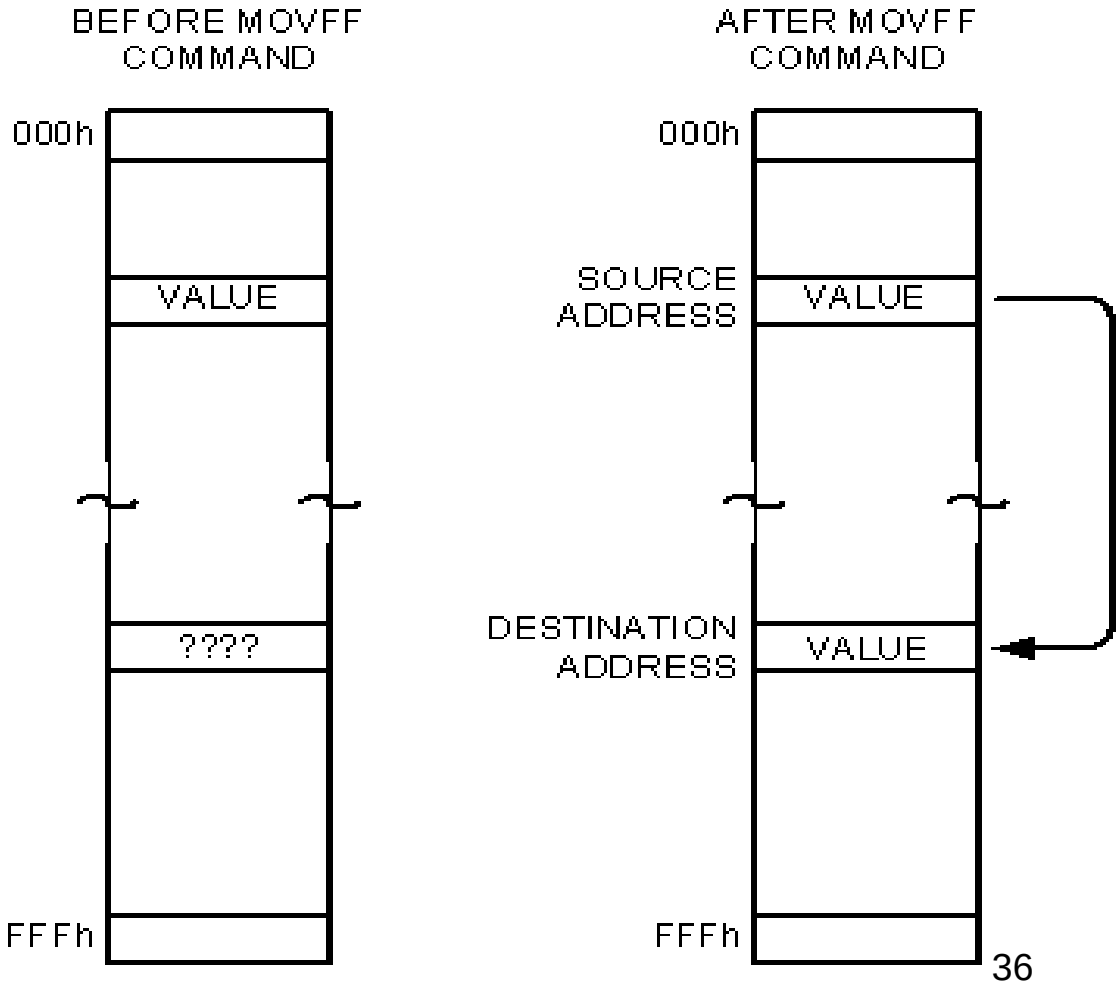
- Bank switching required (as only 256 bytes are addressable) in File Registers if using more than 256 bytes:
- MOV***W***F used to copy from work register into file register:
  - MOVLW 12H ; 12H -> WREG
  - MOVWF 16H ; (WREG) -> File Register 16H
  - MOVWF PORTC ; (WREG) -> PORTC (F8BH)

Figure 2-6. Moving Data Directly Among the fileReg Locations

MOVFF FileRegS, FileRegD

MOVFF 05H, 09H

MOVFF 09H, LATC



WREG and Access Bank Instructions

- **ADDWF fileReg, D** ; (D = Destination Bit)
  - Contents of WREG added to contents of fileReg address location
    - If D = 0, result placed in WREG
    - If D = 1, result placed in fileReg location

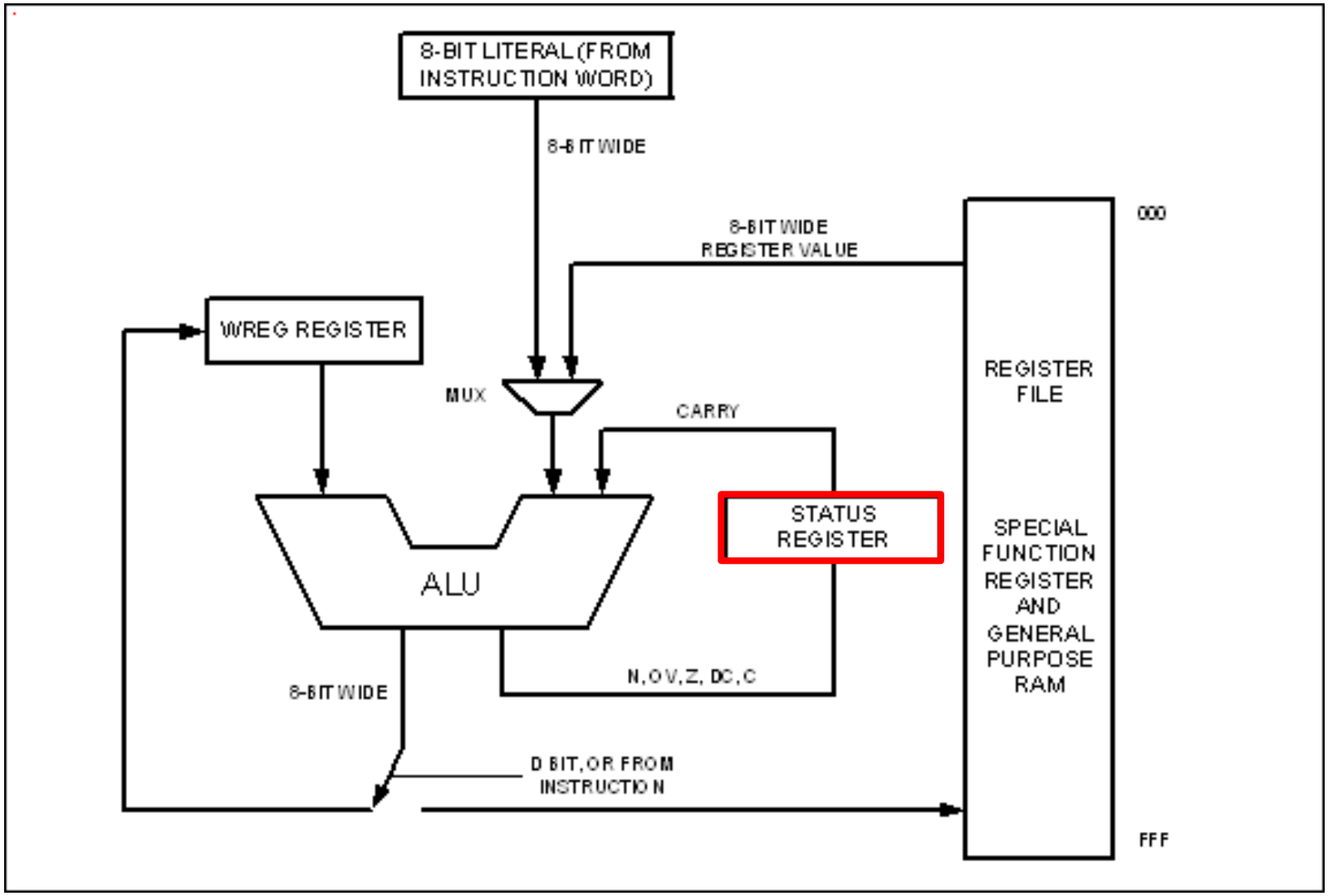
  e.g., **ADDWF 16H, 0** ; add the value contained in 16H to the value of W (thus store in W)

  e.g., **ADDWF PORTC, 1** ; add the value contained in W to the value of PORTC/F82H (thus store in F82H/PORTC's IO Pins)

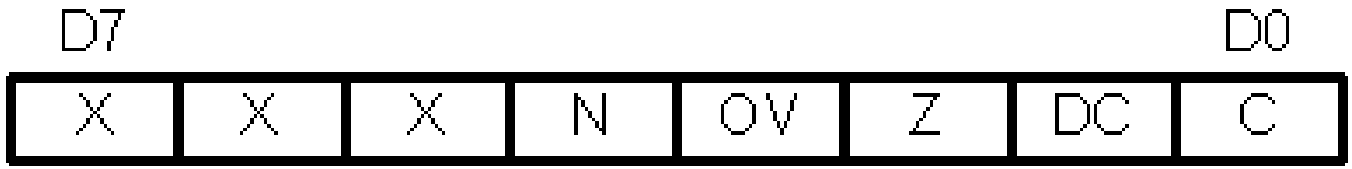
ALU Instructions WREG & fileReg (Table 2-2)

- **ADDWF fileReg, d** ;ADD WREG & fileReg
  - W + F → d
    - If d = 0, result put in WREG
    - If d = 1, result put in F (location)
- **ADDWFC fileReg, d** ;ADD WREG & fileReg & Carry
  - W + F + C → d
    - If d = 0, result put in WREG
    - If d = 1, result put in F (location)

STATUS Register in the CPU



Bits of Status Register



C – Carry flag

DC – Digital Carry flag

Z – Zero flag

OV – Overflow flag

N – Negative flag

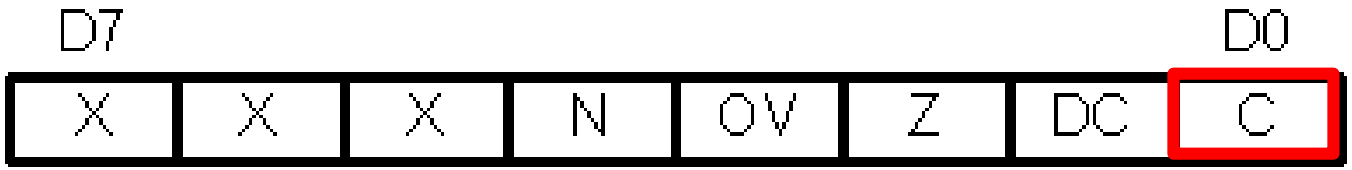
X – D5, D6, and D7 are not implemented, and reserved for future use.

Figure 2-7

Carry Flag

- C Flag is set (1) when there is a “carry out” from the D7 bit
  - Can be set by an ADD or SUB

```
   1101 1010
 + 1010 1111
 = 1 1000 1001
```



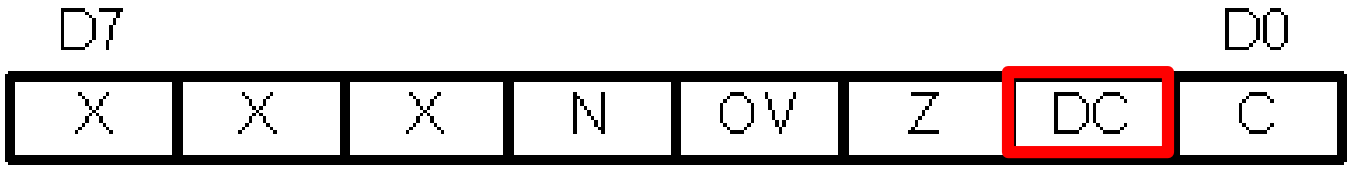
Digital Carry Flag

- DC Flag is set (1) when there is a “carry” from the D3 to D4 bits
  - Can be set by an ADD or SUB

    1101 1010

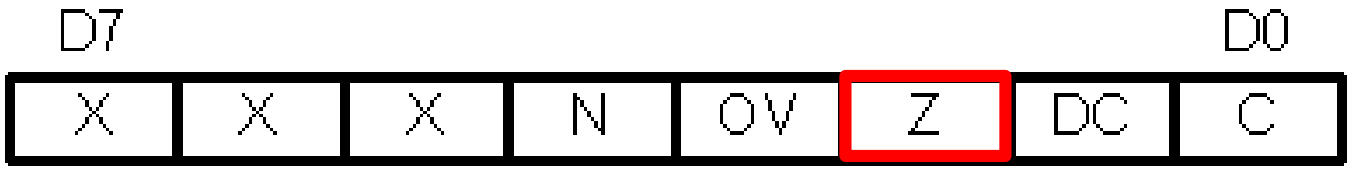
    + 1010 1111

    = 1 1000 1001
  - Used by instructions that perform BCD arithmetic



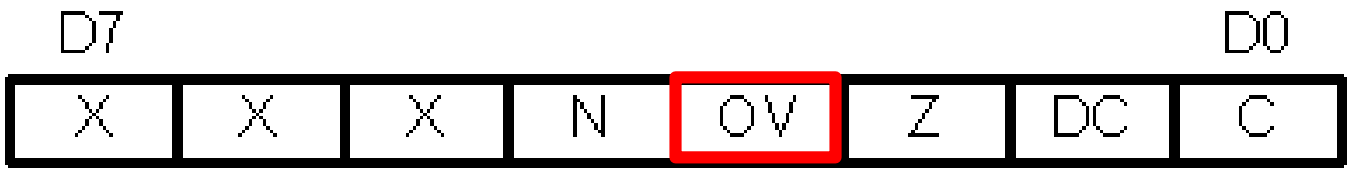
Zero Flag

- Z Flag indicates if the the result of an arithmetic or logic operation is 0
  - Result = 0; Z = 1
  - Result ≠ 0; Z = 0



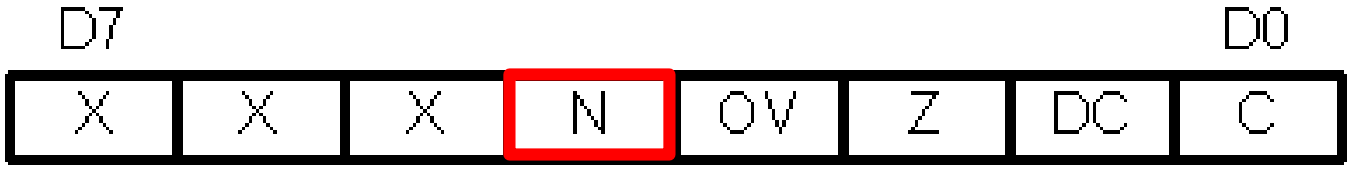
Overflow Flag

- OV Flag is set (1) when the result of a signed number operation is too large
  - The numerical result overflows/overtakes the sign bit of the number
- Usually used to detect errors in signed operations



Negative Flag

- N Flag is set (1) when the result of an arithmetic operation is less than zero
  - If D7 bit = 0, N = 0, positive result
  - If D7 bit = 1, N = 1, negative result



Flags Affected Following Execution of Most Instructions

- See Table 2-4. on page 60 of textbook
- ADDLW can affect C, DC, Z, OV N
- ANDLW can affect Z
- MOVF can affect Z
- Move instructions (except for MOVF) will not affect any status bits

ADDLW Example

- **MOVLW 38H**
- **ADDLW 2FH**

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

C = ?

DC = ?

Z = ?

ADDLW Example

- **MOVLW 38H**
- **ADDLW 2FH**

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

C = 0 no carry out from bit 7

DC = 1 carry out from bit 3 to 4

Z = 0 WREG has value other than zero after addition

ADDLW Example cont.

- **MOVLW 9CH**
- **ADDLW 64H**

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

C = ?

DC = ?

Z = ?

ADDLW Example cont.

- **MOVLW 9CH**
- **ADDLW 64H**

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

C = 1 carry out from bit 7

DC = 1 carry out from bit 3 to 4

Z = 1 WREG has value of zero after addition

Flag Bits used in Branching

- BC — Branch if C = 1 (carry, positive)
- BNC — Branch if C ≠ 1
- BZ — Branch if Z = 1 (zero)
- BNZ — Branch if Z ≠ 1
- BN — Branch if N = 1 (negative)
- BNN — Branch if N ≠ 1
- BOV — Branch if OV = 1 (overflow, 2s cmp)
- BNOV — Branch if OV ≠ 1

Assignment for This Week

- Read Chapter 0-2 of textbook

- **Download:**
  - MPLAB X IDE (v4.05)
    - http://www.microchip.com/mplab/mplab-x-ide
  - XC8 Compiler (v1.45)
    - http://www.microchip.com/mplab/compilers

Scroll to the bottom of the pages and click the "Downloads" tab. Pick the install for your OS.