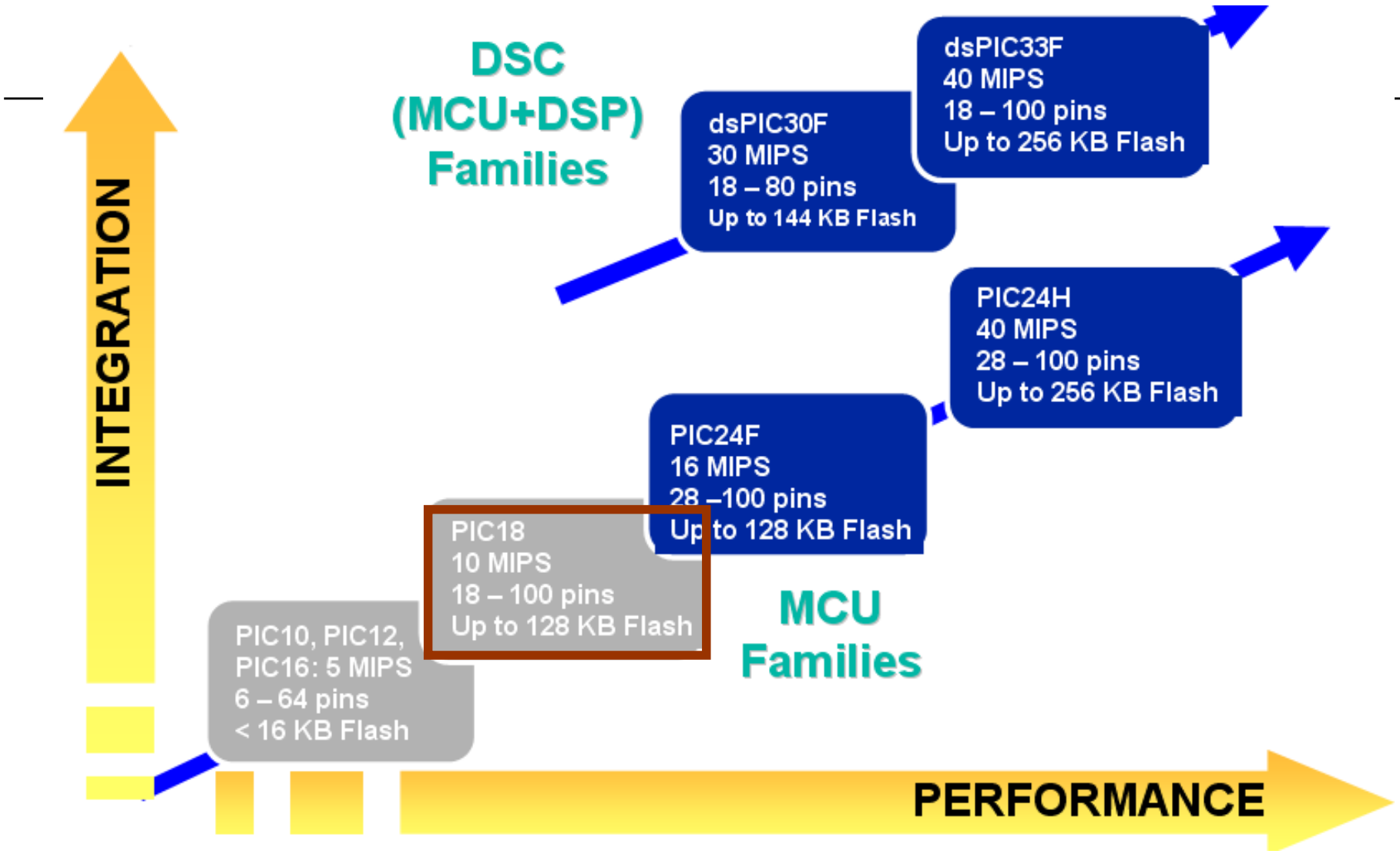
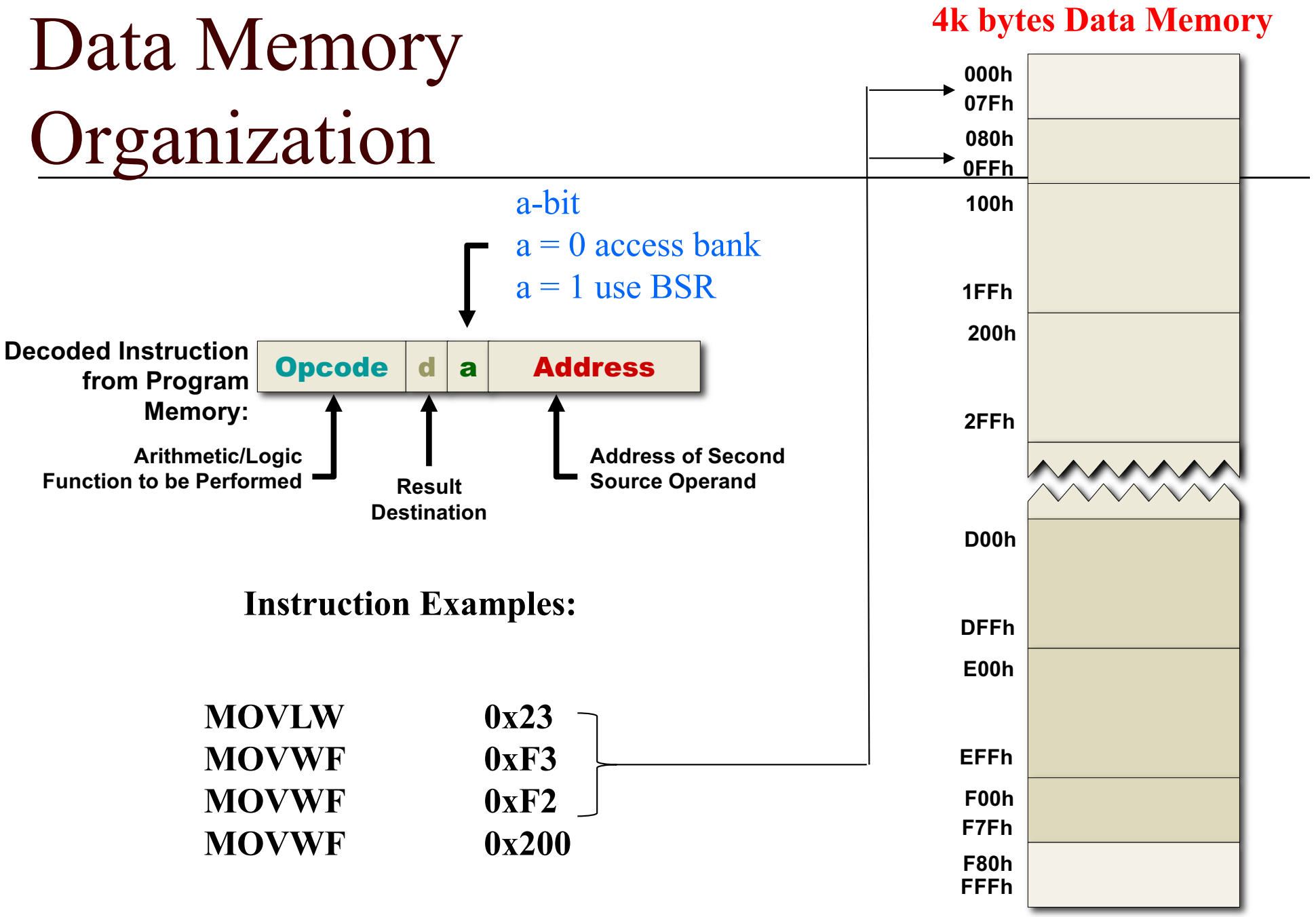
Chapter 3

PIC18F Programming Model and Its Instruction Set

Updated: 2/10/2019


INTEGRATION
DSC
(MCU+DSP)
Families
dsPIC33F
40 MIPS
18 – 100 pins
Up to 256 KB Flash
dsPIC30F
30 MIPS
18 – 80 pins
Up to 144 KB Flash
PIC24H
40 MIPS
28 – 100 pins
Up to 256 KB Flash
PIC24F
16 MIPS
28 –100 pins
Up to 128 KB Flash
PIC18
10 MIPS
18 – 100 pins
Up to 128 KB Flash
MCU
Families
PIC10, PIC12,
PIC16: 5 MIPS
6 – 64 pins
< 16 KB Flash
PERFORMANCE

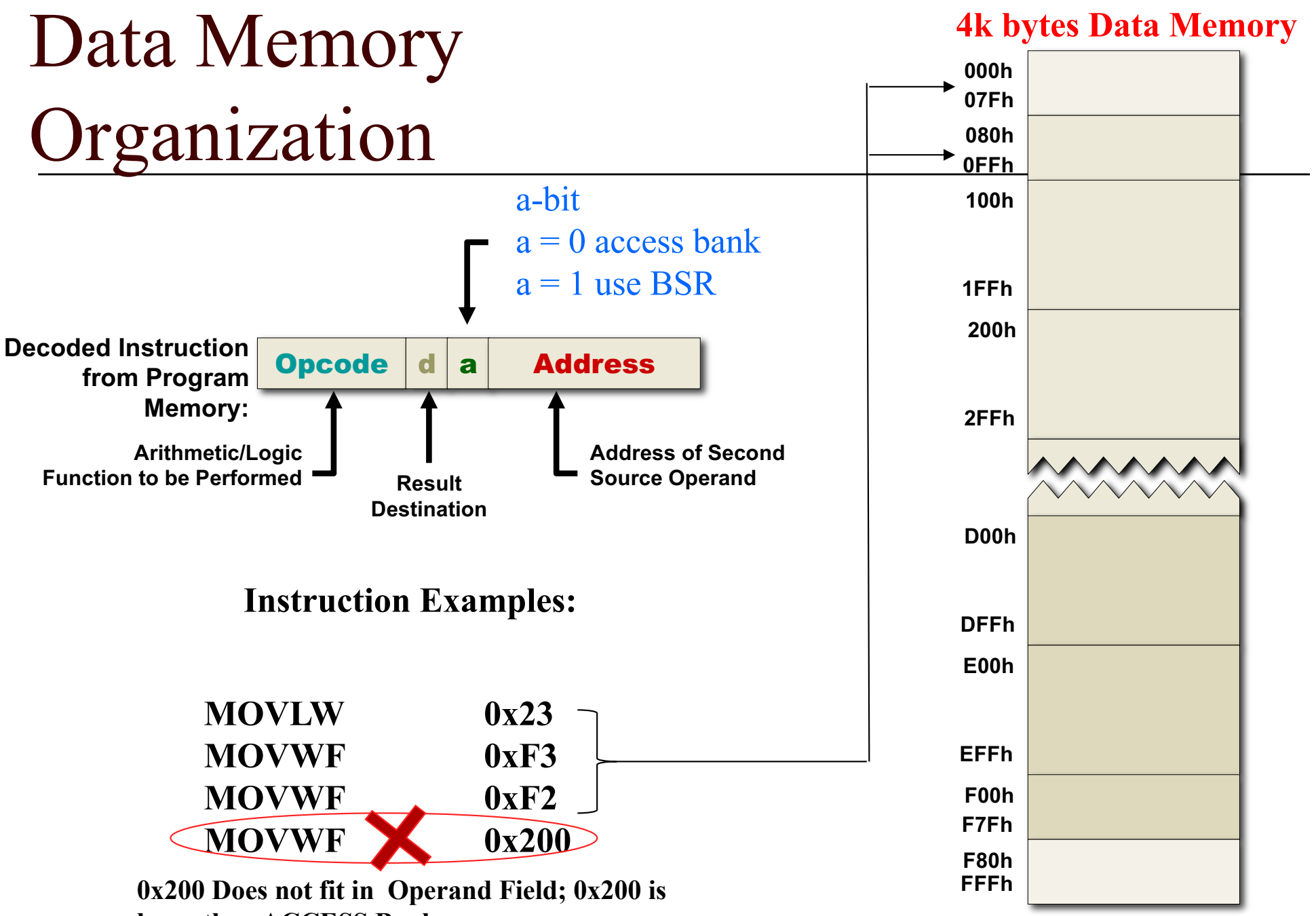
Data Memory Organization



Instruction Examples:

```
MOVLW   0x23
MOVWF   0xF3
MOVWF   0xF2
MOVWF   0x200
```

Data Memory Organization

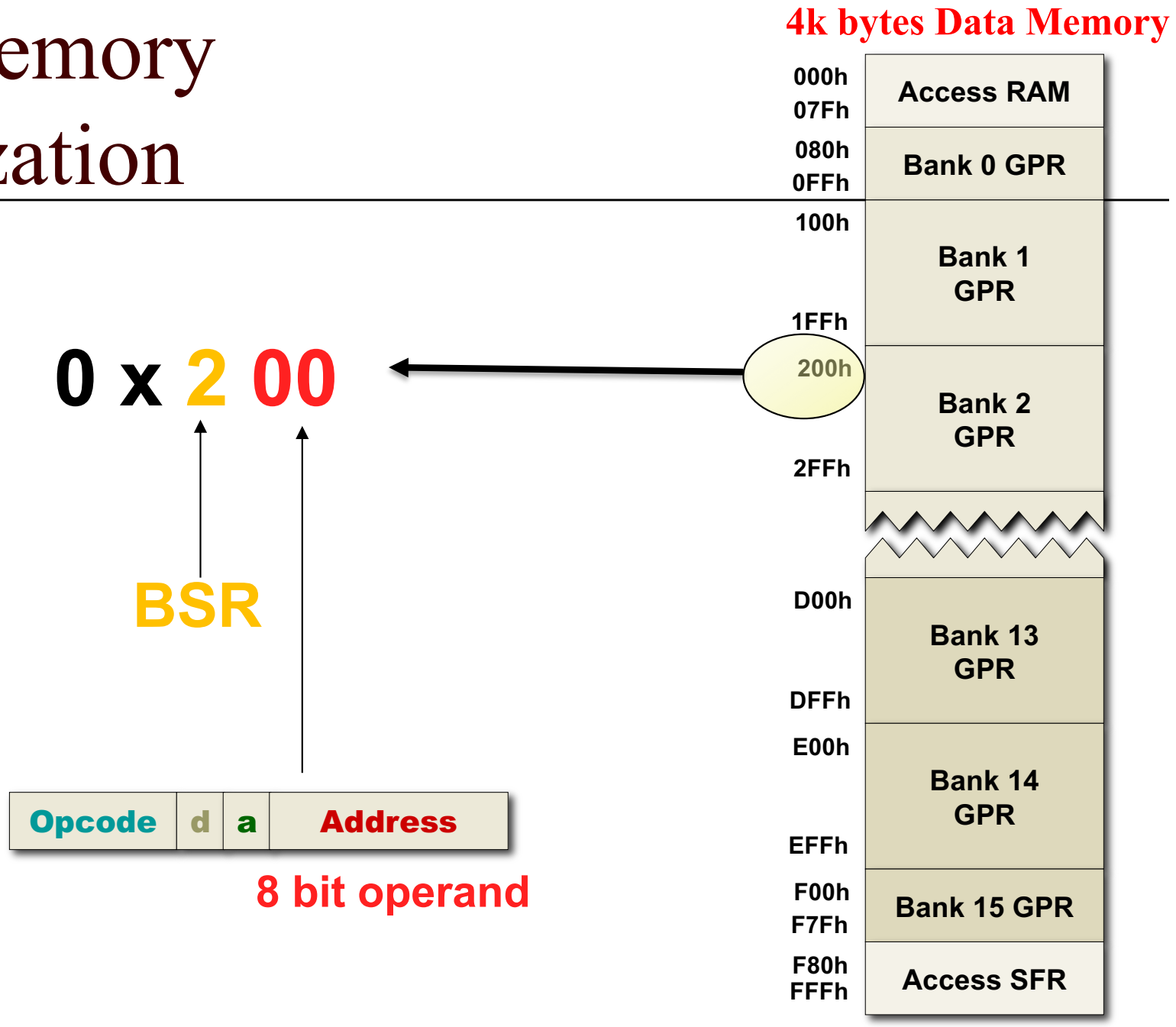


Instruction Examples:

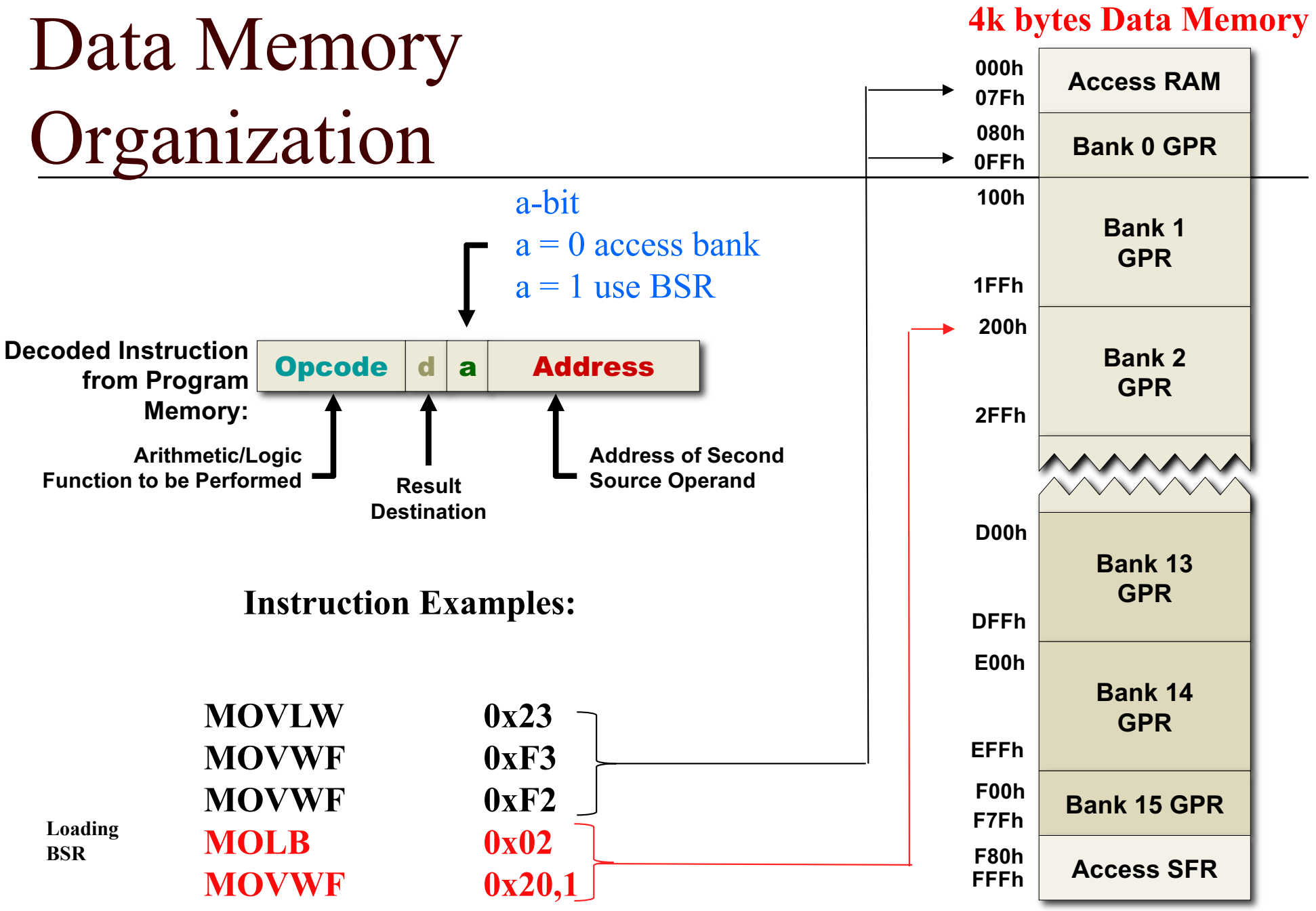
```
MOVLW    0x23
MOVWF    0xF3
MOVWF    0xF2
MOVWF    0x200
```

0x200 Does not fit in Operand Field; 0x200 is large than ACCESS Bank

Data Memory Organization



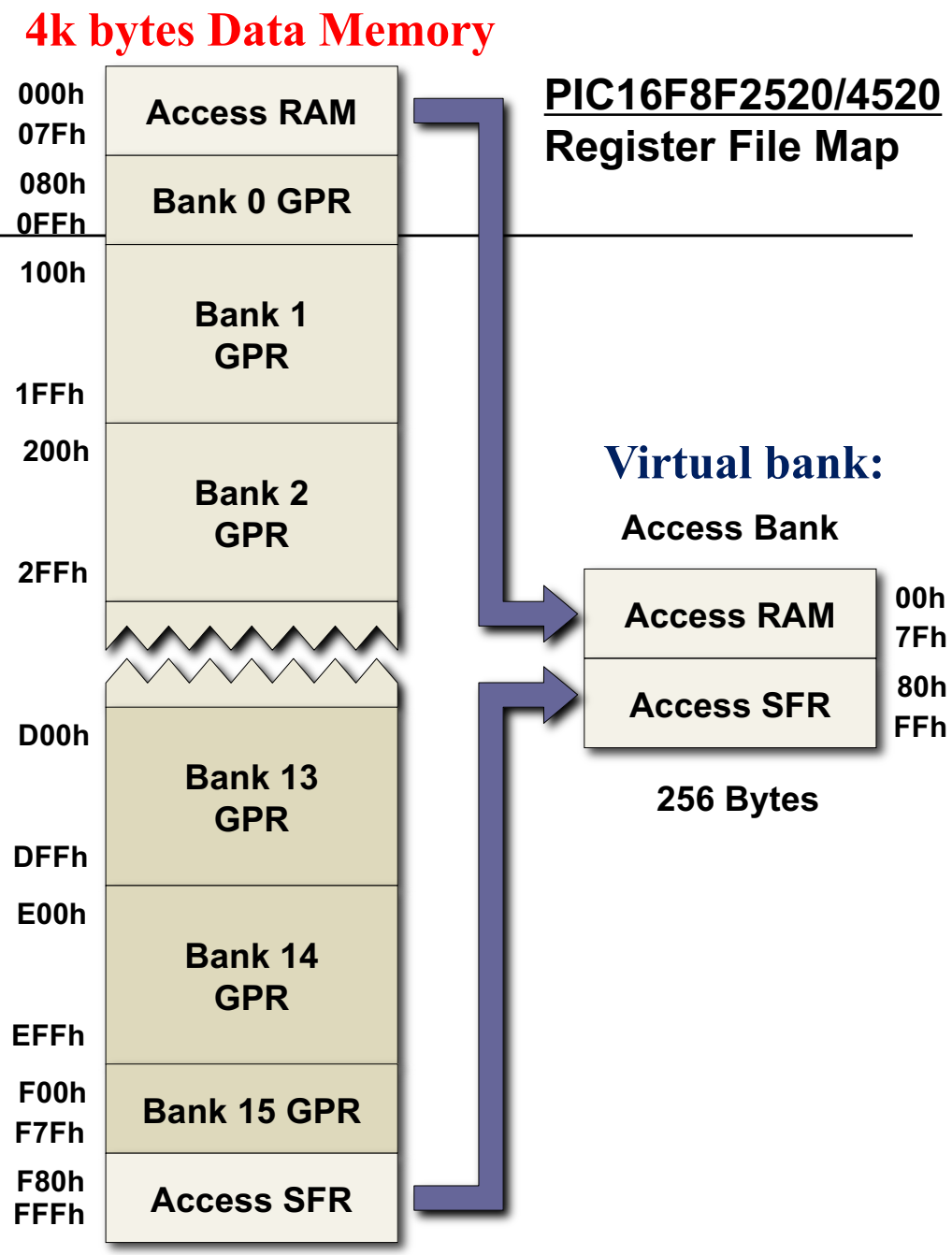
Data Memory Organization



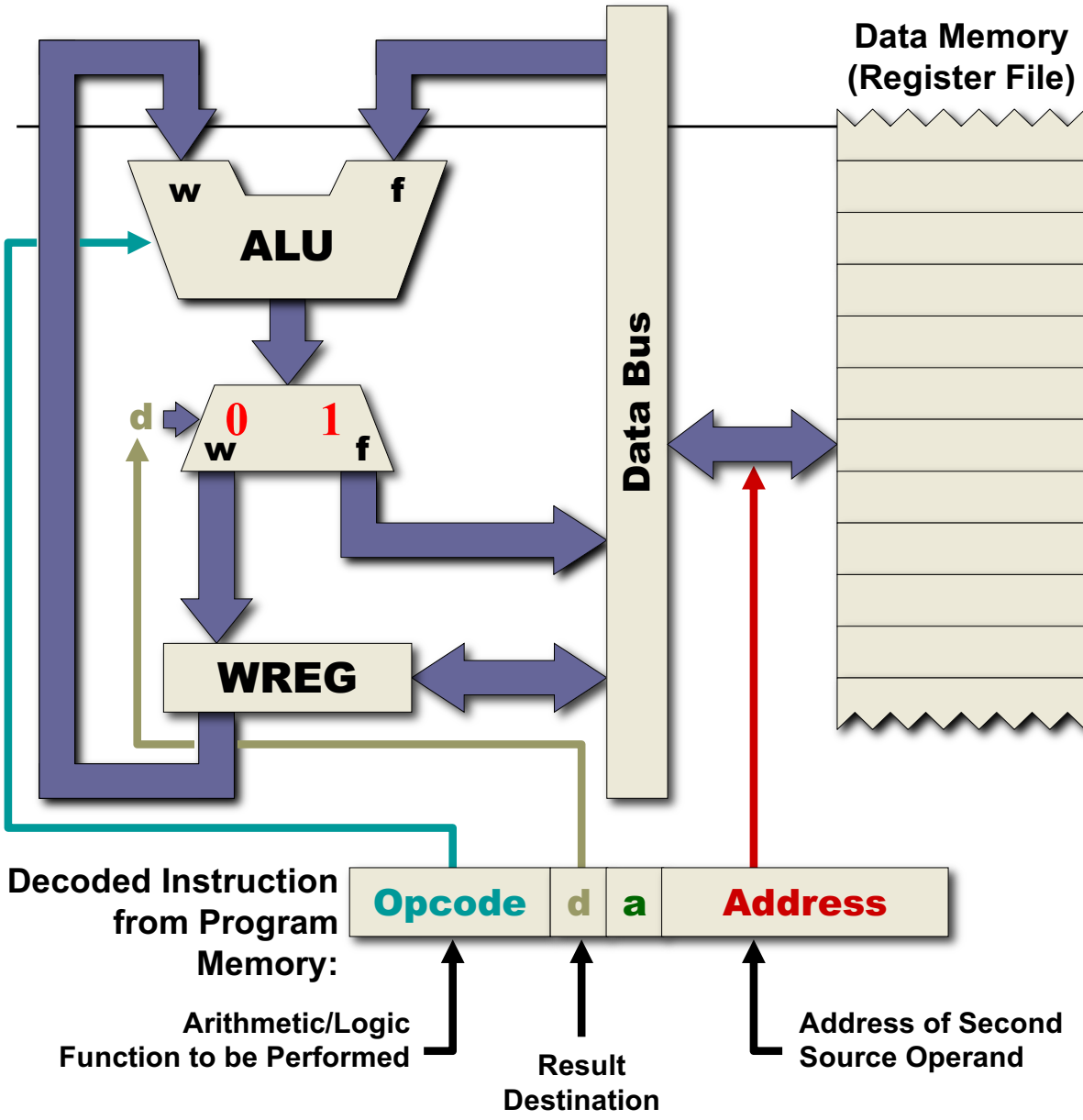
Data Memory Organization

Remember:

- Data Memory up to 4k bytes
- Divided into 256 byte banks
- Half of bank 0 and half of bank 15 form a **virtual bank** that is accessible no matter which bank is selected



Register File Concept



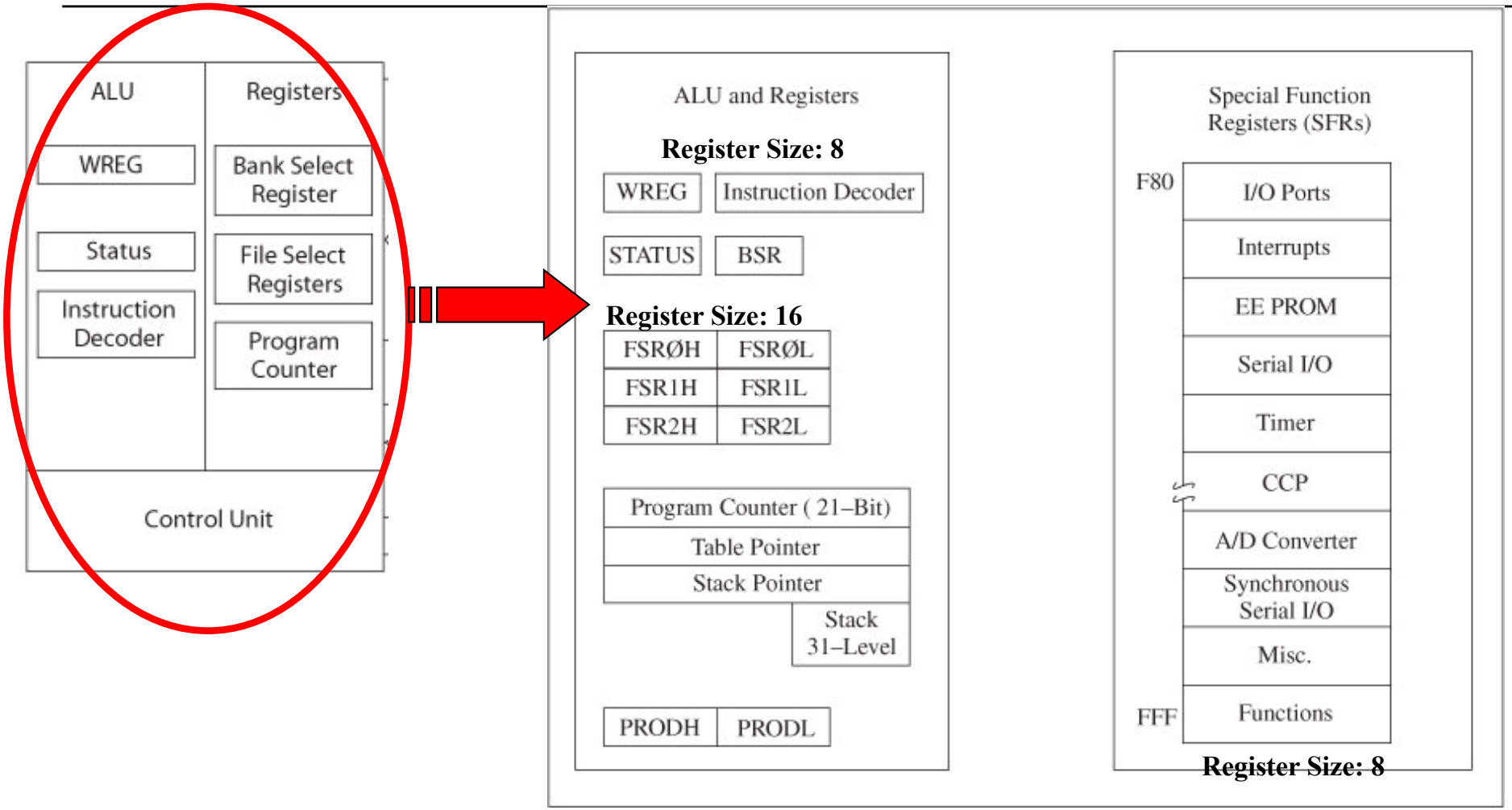
- Register File Concept: All of data memory is part of the register file, so any location in data memory may be operated on directly
- All peripherals are mapped into data memory as a series of registers
- Orthogonal Instruction Set: ALL instructions can operate on ANY data memory location

a-bit
a = 0 access bank
a = 1 use BSR

PIC18F Programming Model (1 of 2)

- The representation of the internal architecture of a microprocessor, necessary to write assembly language programs
  - Programming Model
- Two Groups of Registers in PIC16 8-bit Programming Model (all in SRAM)
  - ALU Arithmetic Logic Unit (ALU)
  - Special Function Registers (SFRs)

PIC18F Programming Model (2 of 2)

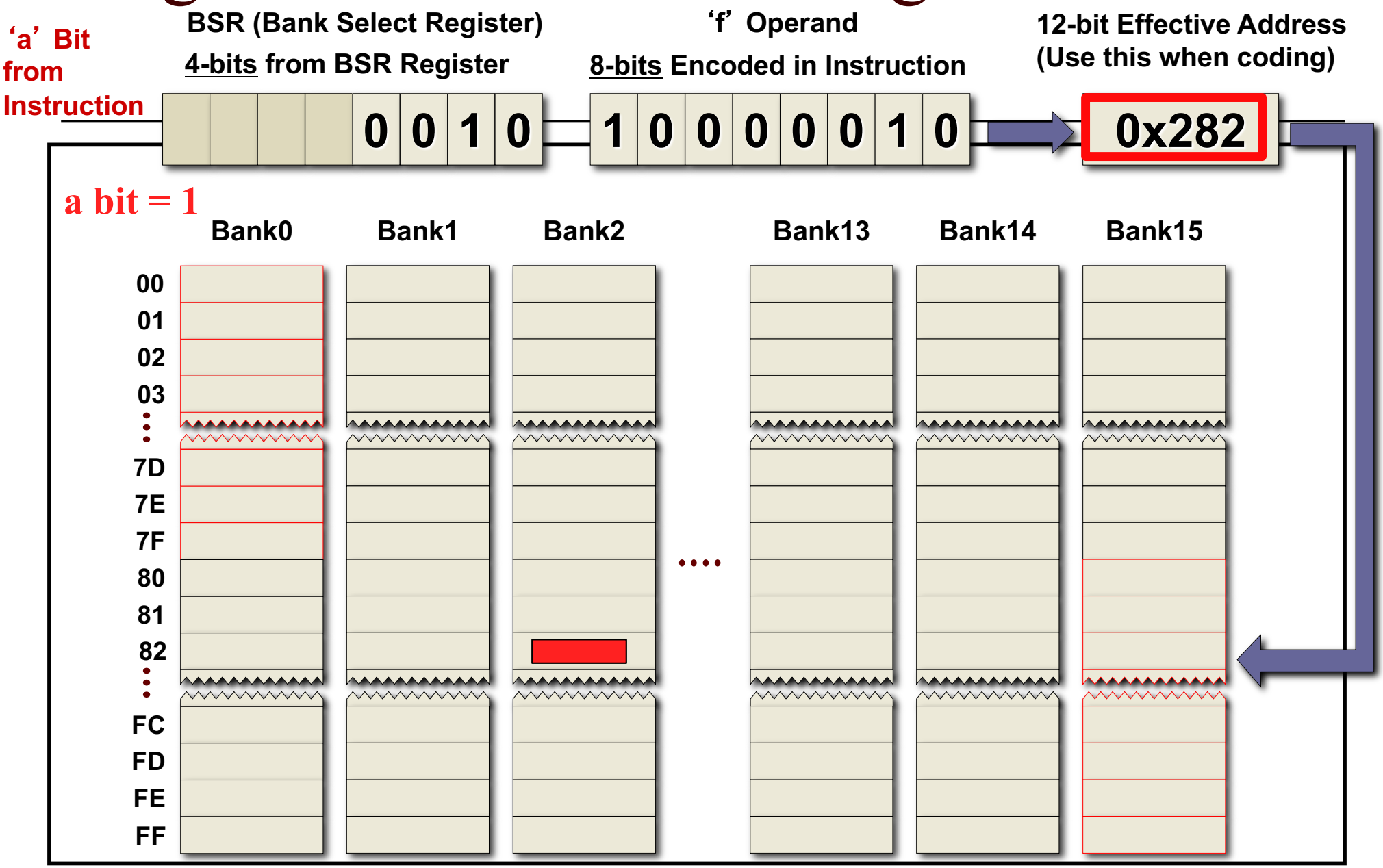


Two Groups of Registers in PIC16 8-bit Programming Model

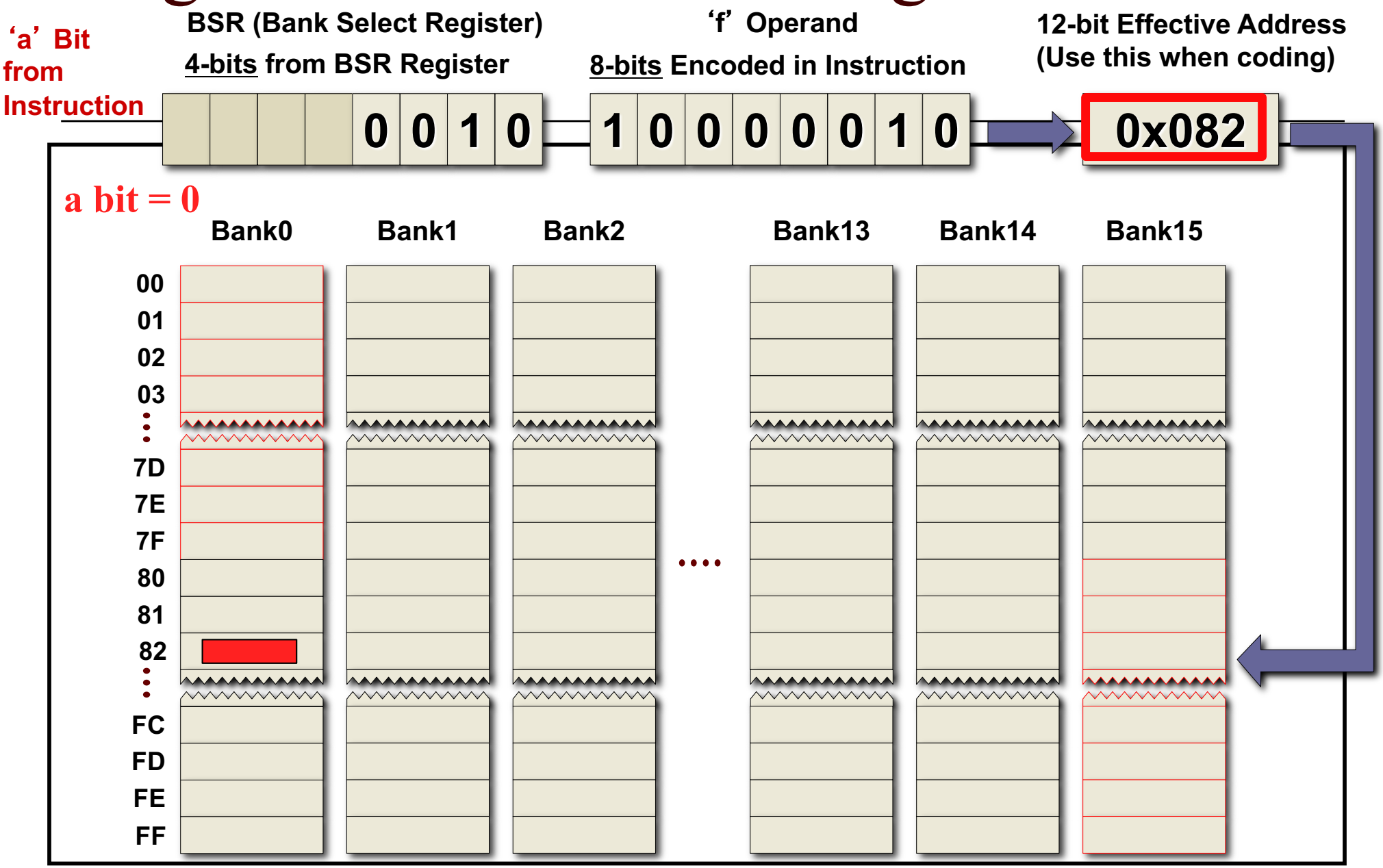
Registers

- WREG
  - 8-bit Working Register (equivalent to an accumulator)
  - Used for arithmetic and logic operations
- BSR: Bank Select Register (0 to F)
  - 4-bit Register
    - Only low-order four bits are used to provide MSB four bits of a12-bit address of data memory.

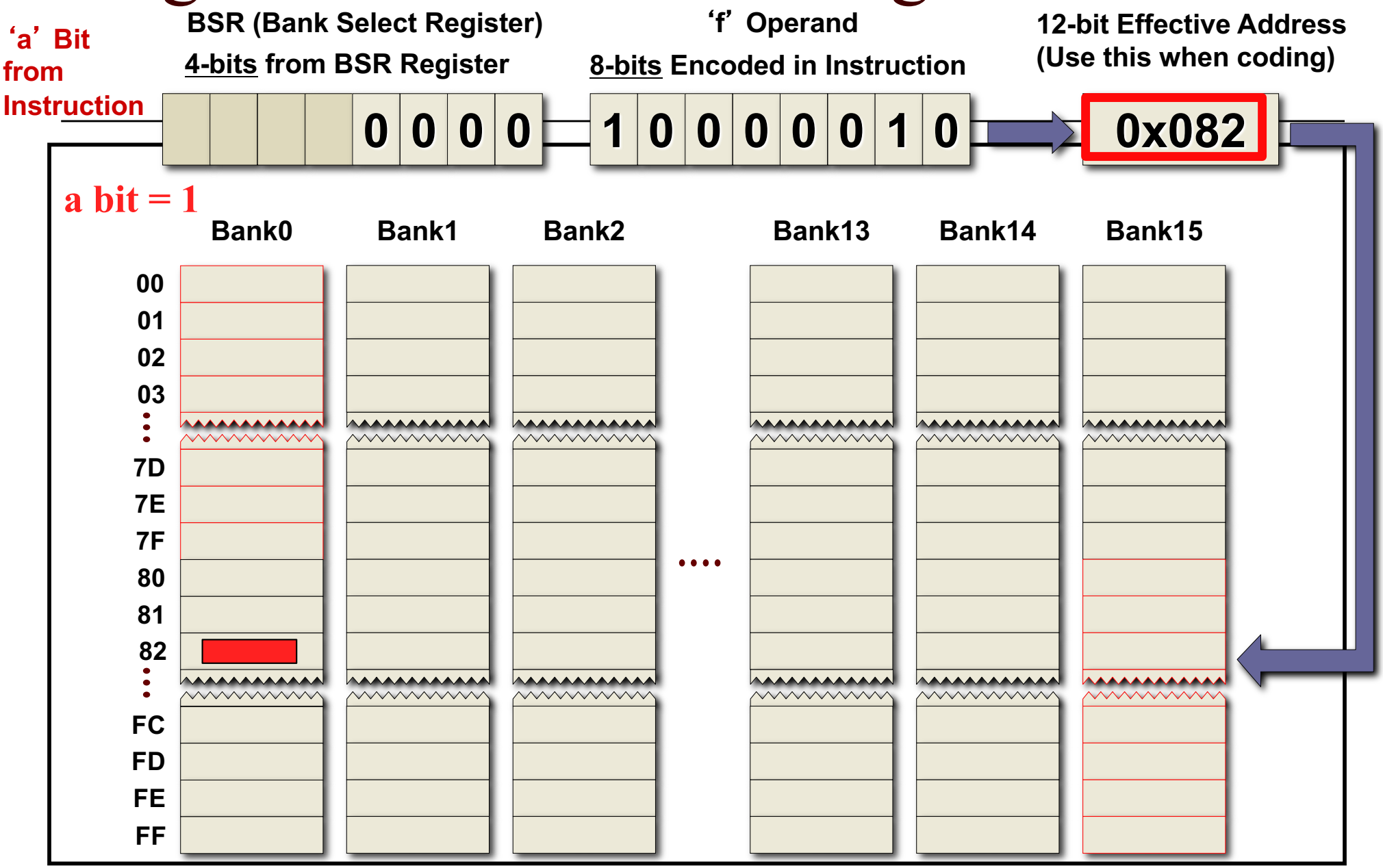
Register Direct Addressing



Register Direct Addressing

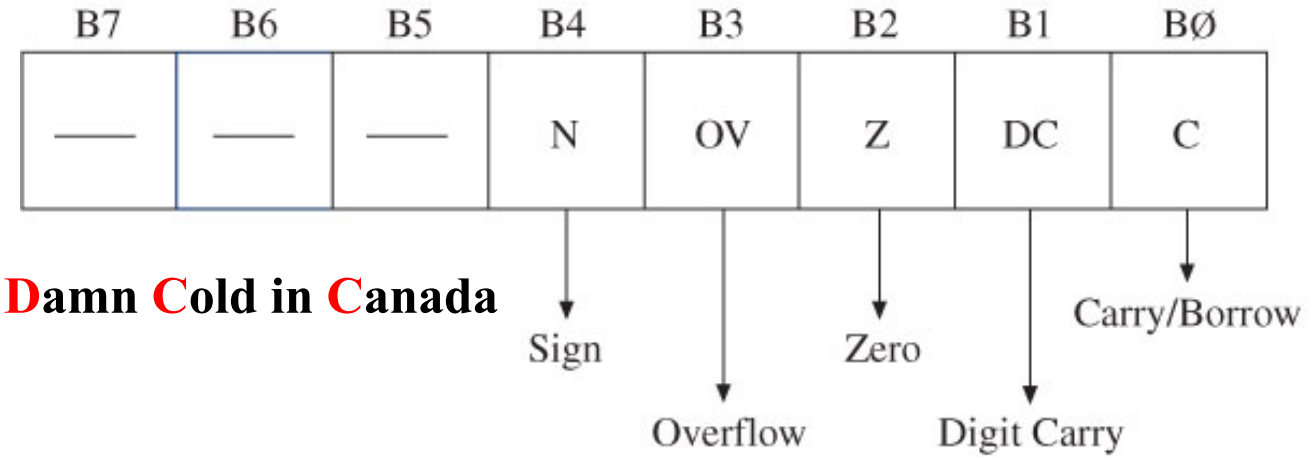


Register Direct Addressing



STATUS: Flag Register

Flags in Status Register



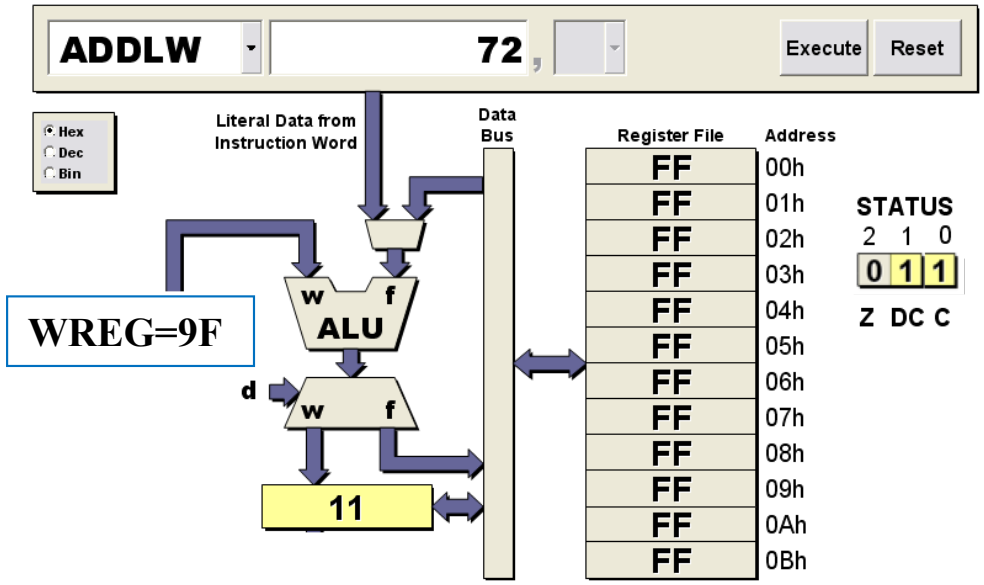
Remember:
NOVember iZ Damn Cold in Canada

- C (Carry/Borrow Flag): set when an addition generates a carry and a subtraction generates a borrow
- DC (Digit Carry Flag): also called Half Carry flag; set when carry generated from Bit3 to Bit4 in an arithmetic operation
  - Used for BCD representation
- Z (Zero Flag): set when result of an operation is zero
- OV (Overflow Flag): set when result of an operation of **signed numbers** goes beyond seven bits – if the results fall outside 127 (0x7F) and -128 (0x80)
- N (Negative Flag): set when bit B7 is one

Example:
PIC18 Visual Interpreter

ADD: WREG=9F and L=72. Which flags will be set?

```
    1001 1111     N=0
    0111 0010     OV=0 // not signed numbers
    ----------    Z=0
1   0001 0001     DC=1
    =0x11         C= 1
```

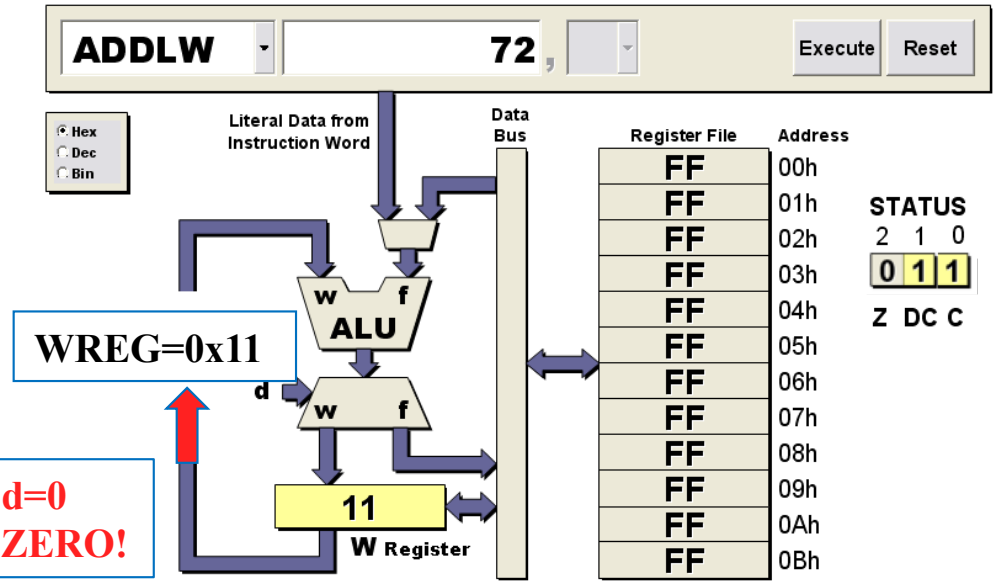



```
0111 0010
1001 1111
----------
0001 0001
=0x11
```

Example:
PIC18 Visual Interpreter

ADD: WREG=9F and L=72. Which flags will be set?

```
  1001 1111     N=0
  0111 0010     OV=0 // not signed numbers
  ----------    Z=0
1 0001 0001     DC=1
  =0x11         C= 1
```



The results will be directed back to WREG since d=0
Note that for ADDLW instruction the d bit is always ZERO!

```
0111 0010
1001 1111
----------
0001 0001
=0x11
```

Example:
PIC18 Visual Interpreter

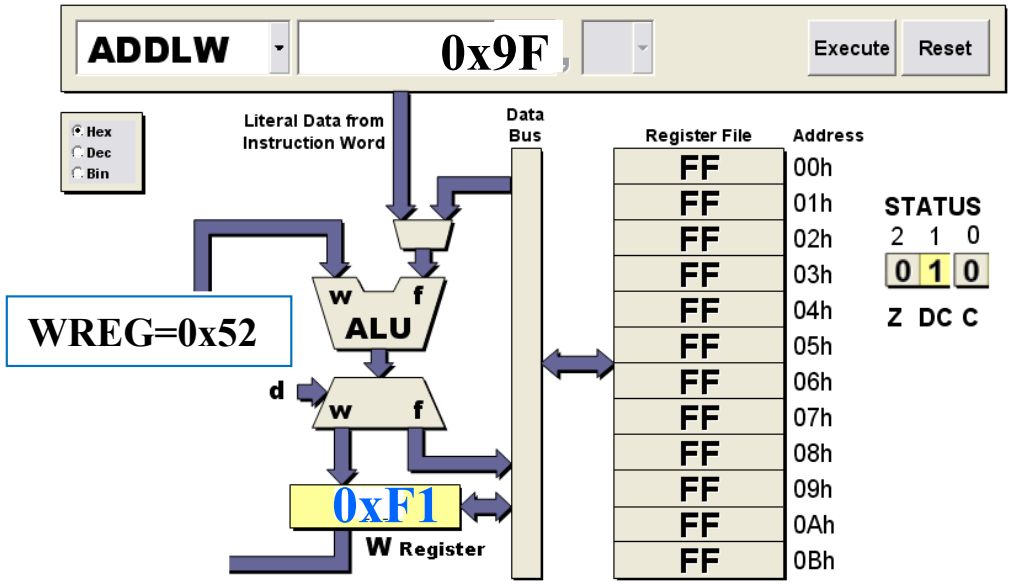
ADD: Literal=0x9F and WREG=0x52.
Which flags will be set?

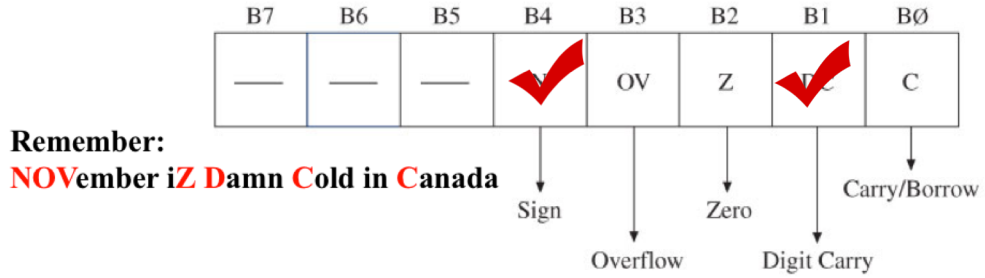
1

1001 1111
0101 0010

1111 0001
= 0xF1

N=1
OV=0
Z=0
DC=1
C=0





Examples

Practice These Instructions!!

```
ORG     0x20        ;Begin assembly at 0000H

MOVLW   0X00
MOVFF   W,STATUS
MOVLW   0x88
ADDLW   0X88

MOVLW   0X00
MOVFF   W,STATUS
MOVLW   0x7F
ADDLW   0X7F

MOVLW   0X00
MOVFF   W,STATUS
MOVLW   0xC0
ADDLW   0X70
-----------------------------
MOVLW   0X00
MOVFF   W,STATUS
MOVLW   0x50
SUBLW   0XF0

MOVLW   0X00
MOVFF   W,STATUS
MOVLW   0xF0
SUBLW   0X50

MOVLW   0X00
MOVFF   W,STATUS
MOVLW   0xFF
SUBLW   0X1

MOVLW   0X00
MOVFF   W,STATUS
MOVLW   0x80
SUBLW   0X1         ;--> STATUS: N,O,DC
```

Clearing STATUS Register

W=110; Status: DC,C,OV

W=FE; Status: DC,N,OV

W=130; Status: C

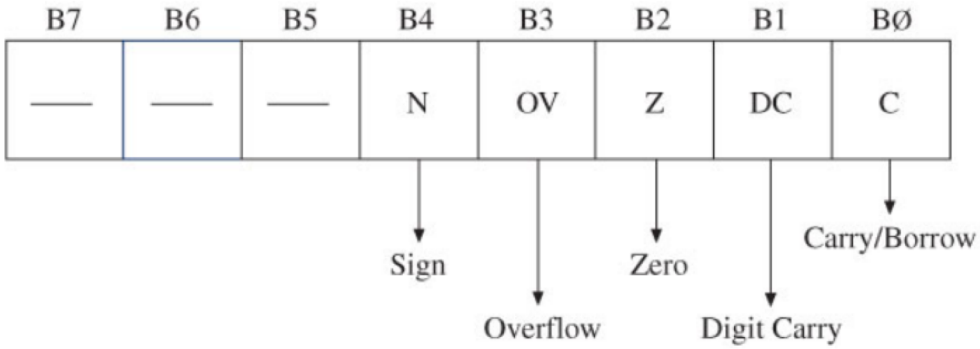
L-W→W

W=A0; Status: ?

W=?; Status: ?

W=?; Status: ?

L-W→W
Note: 0x01-0x80→81/ Note that this can be interpreted as 1-(-128)=+129→ Overflow!

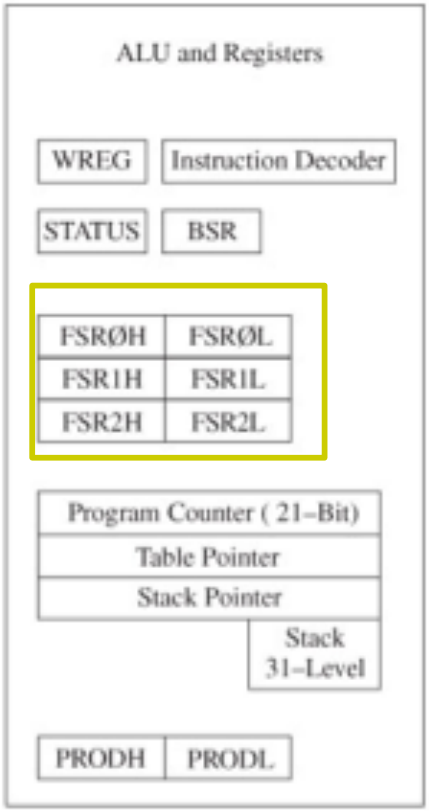


How OV Is Calculated

- Read this for a very good description as to how OV works:
  - http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt
- We compare carry into 7th bit and carry out of the 7th bit; if they are EQUAL then no overflow, else there is an overflow. Basic steps:
  - When ADDING just add the two numbers; THEN check Carry-IN and Carry-OUT
  - When SUBTRACTING (X-Y); FIRST convert Y to 2's complement THEN calculate X+Y_2'sComp FINALLY check Carry-IN and Carry-OUT
- Try the following (assuming all numbers are in HEX):
  - 0xFF-0x12 ; no overflow because 0xFF+0xEE→ Carry into bit 7=Carry out from 7th bit
  - 0x12-0xFF ; no overflow because 0x12+0x1 → Carry into bit 7=Carry out
  - 0x80-0xFF : Carry into bit 7=Carry out → no overflow
  - 0xFF-0x81; Carry into bit 7 = Carry out → No overflow
  - 0xA-0x81: Carry into bit 7 NOT = Carry out → overflow
  - 0x1-0xFA: Carry into bit 7 = Carry out → No overflow
  - 0xEF-FB: Carry into bit 7 = Carry out from 7th bit→ No overflow

File Select Registers (FSR)

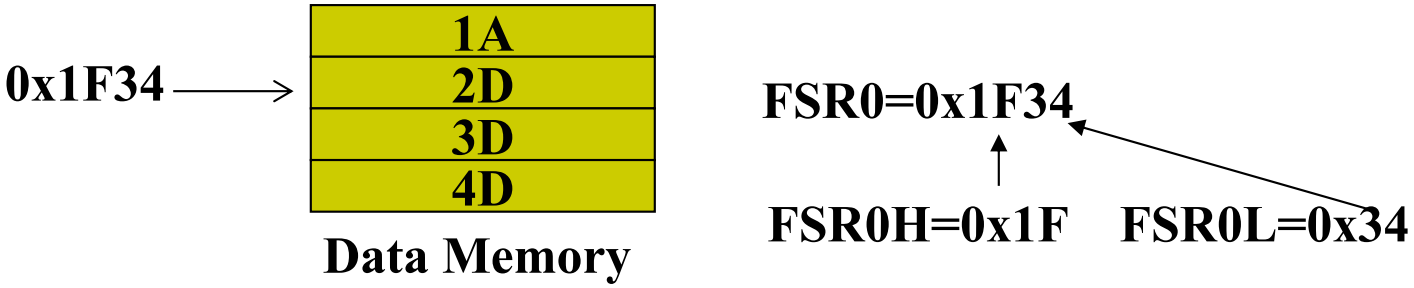
- Three registers holding 12-bit address of data registers
  - FSR0, FSR1, and FSR2
- File Select Registers composed of two 8-bit registers (FSRH and FSRL)
- Used as pointers for data registers for indirect addressing
  - Associated with index (INDF) registers



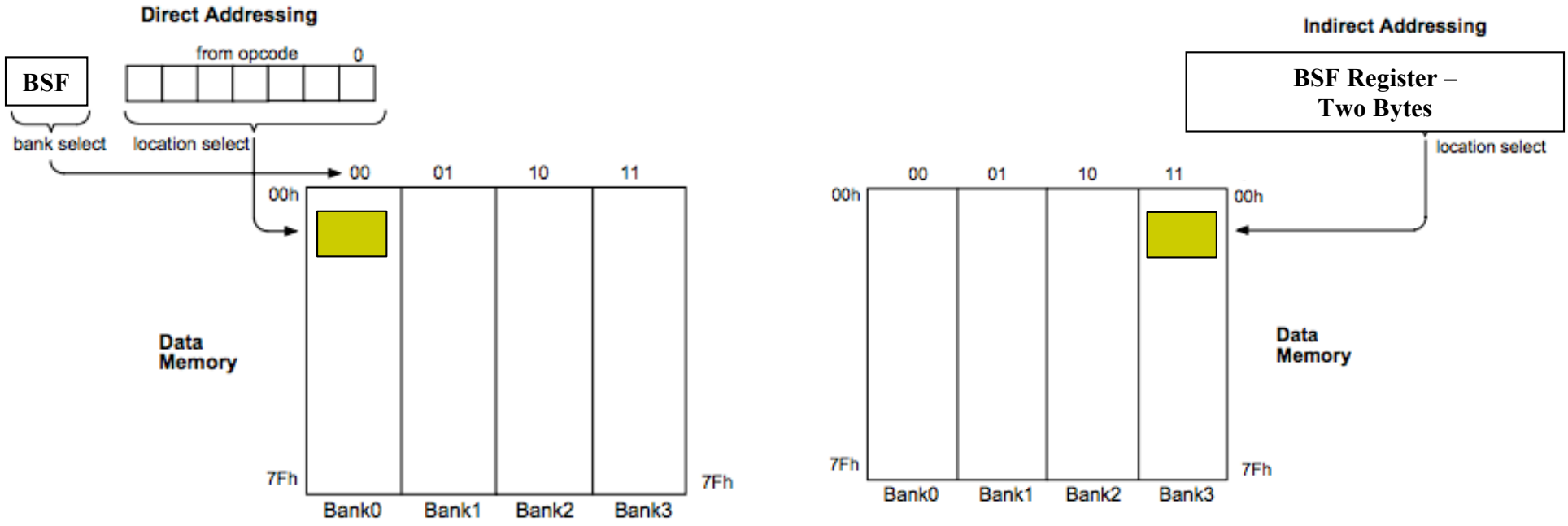
Find FSR0-FSR2 in Special Function Register – What are the File addresses for each? / How many INDF do you find?

File Select Registers (FSR) –
Indirect Addressing

- The main application of FSR is **Indirect Addressing**
  - FSRs will be pointing at the address of the data file and they can be incremented
  - This is much easier than using direct addressing



Direct and Indirect Addressing



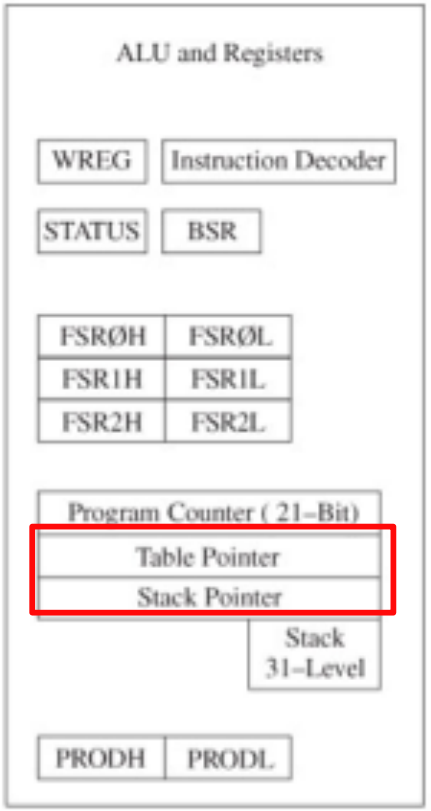
Remember: We are taking about DATA MEMORY!

WE WILL DISCUSS THIS IN MORE DETAILS WHEN WE LEARN MORE AOUT COMMANDS!

http://ww1.microchip.com/downloads/en/DeviceDoc/31006a.pdf

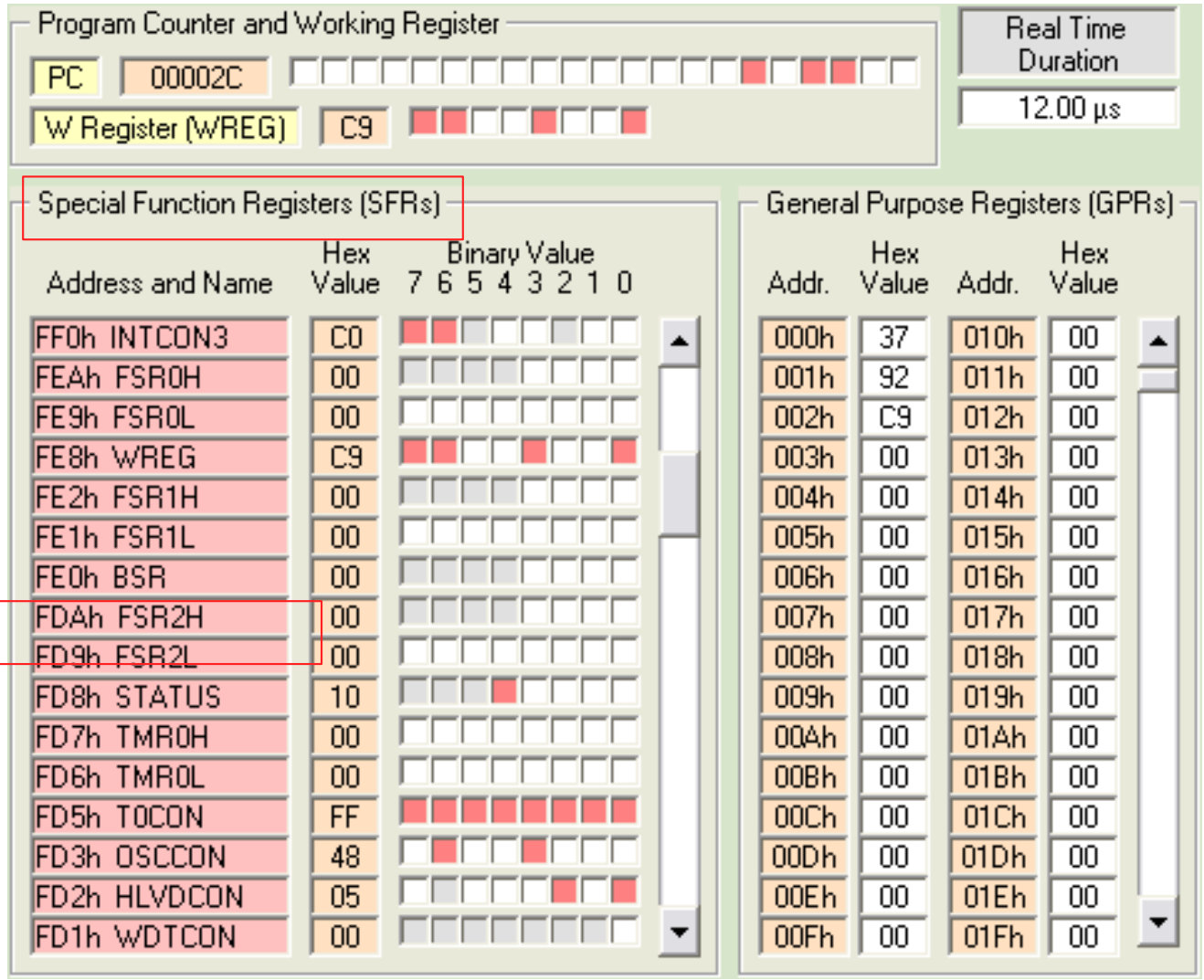
Stack and Table Pointers

- Table Pointer
  - 21-bit register used as a memory pointer to copy bytes between program memory and data registers
- Stack Pointer (SP)
  - Stack is a group of 31 word-size registers used for temporary storage of memory address during execution
  - Used to store the return address
  - Requires 5-bit address
  - Saved in STKPTR in SFR
  - Used primarily for saving PC for next program address prior to entering subroutine



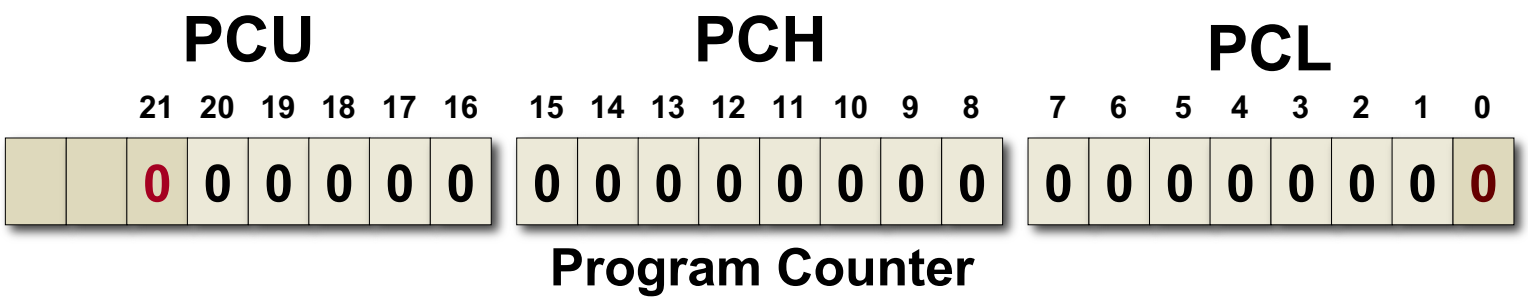
FSR

Don’t confuse SFRs and FSRs (file Select Registers)



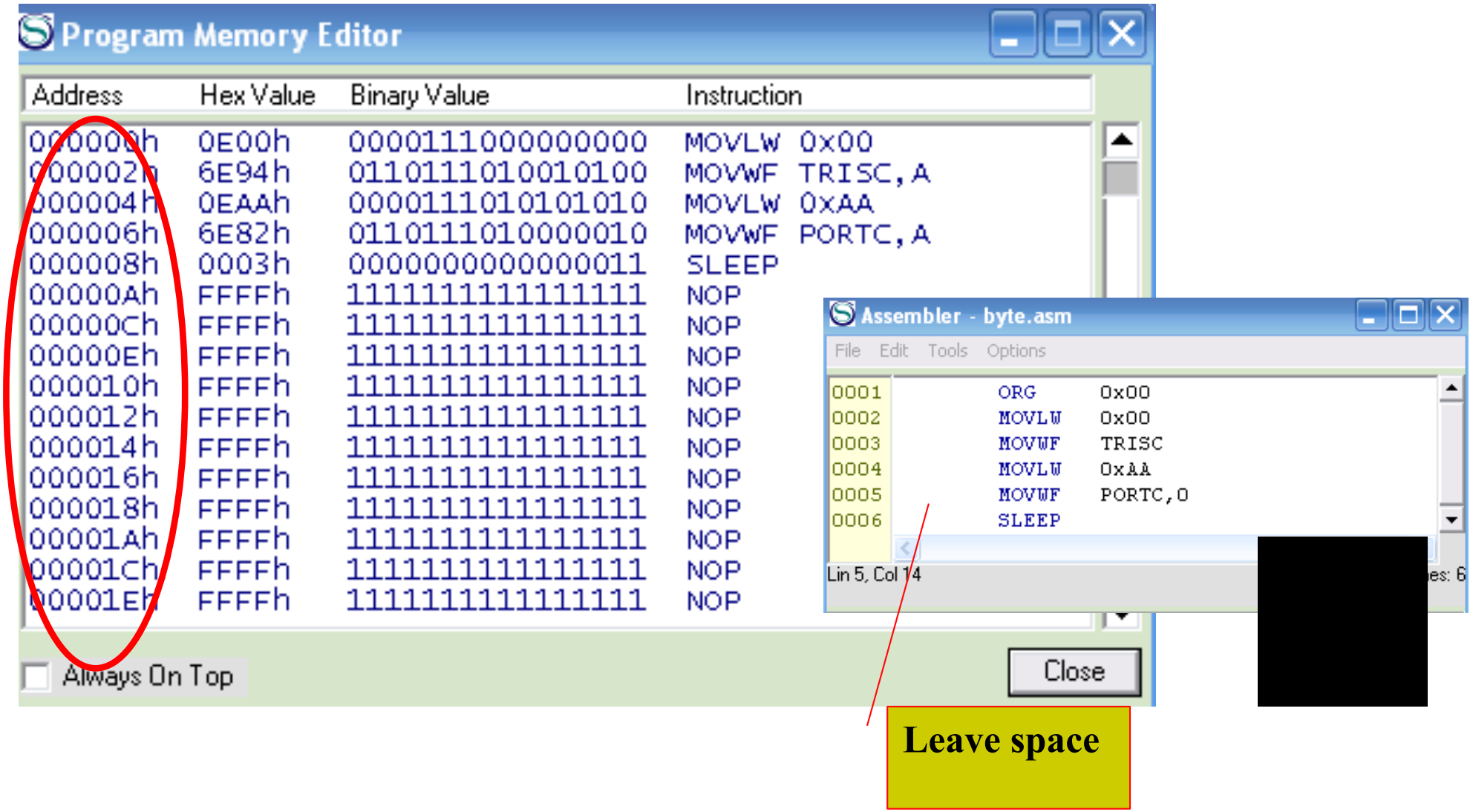
Program Counter

21-bit register functions as a pointer to **program memory** during program execution



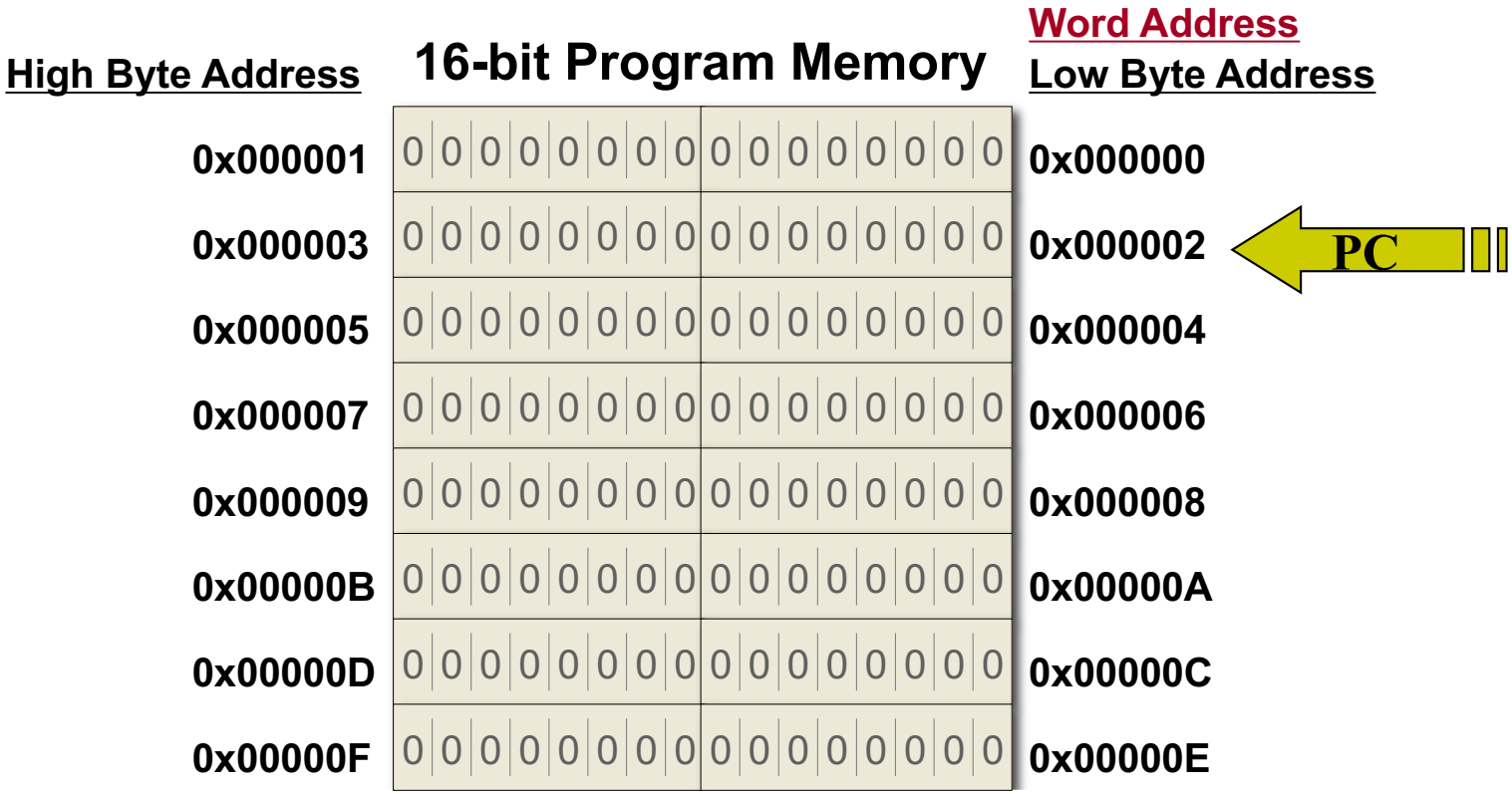
- 21-bit PC can access up to 2^{21} = 2MB (1MWord)
- 22nd bit used to access **configuration memory** at program time or via table reads & writes
- Contains address of NEXT instruction (pipelining)
- Lower byte accessible in data memory as PCL
- Upper bytes indirectly accessible via PCLATH/PCLATU
- Bit 0 of PC is always '0' except when reading or writing program memory via table read/write mechanism

21-Bit PC Example & Program Memory



Program Memory is Byte Addressable

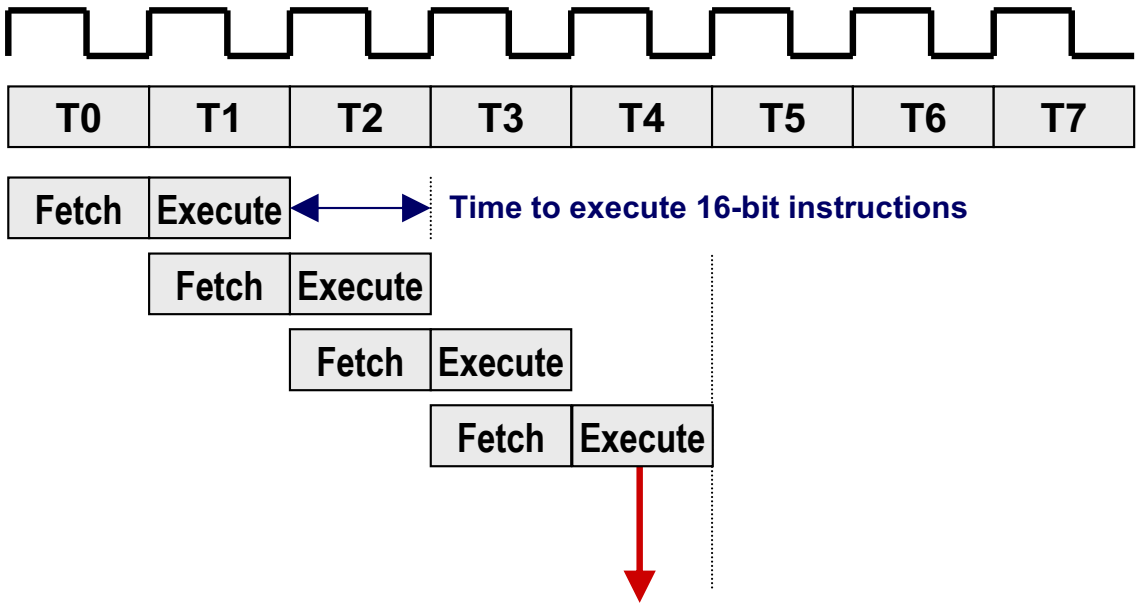
- Low byte has even address, high byte has odd address
- Addresses of instructions are always even
- 16-bit wide program memory is byte addressable
- All program instructions will start at an even address
- So if we are jumping 4 instructions ahead, we are actually jumping 8-bytes (or 8 word addresses) ahead



Instruction Pipelining

- Instruction fetch is overlapped with execution of previously fetched instruction

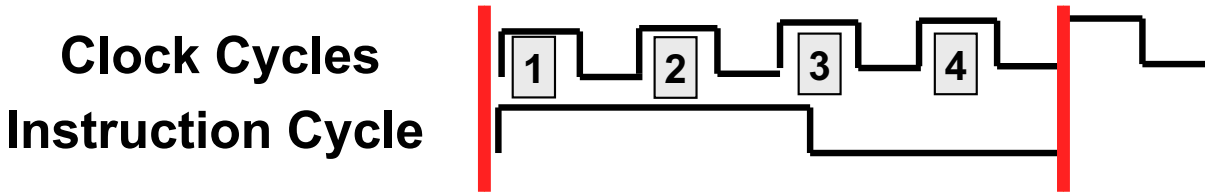
Instruction Cycles



Example Program

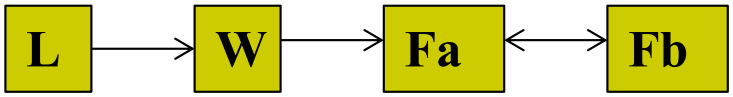
```
1   MAIN  movlw   0x37
2         movwf   REG0
3         movlw   0x92
4         movwf   REG1
          :
```

Remember: In PIC ONE Instruction Cycle takes 4 Clock Cycle:



Introduction to PIC18 Instruction Set

- Includes 77 instructions;
  - 73 one-word (16-bit) long
  - Four two-words (32-bit) long
- Divided into seven groups
  - Move (Data Copy) and Load
  - Arithmetic
  - Logic
  - Program Redirection (Branch/Jump)
  - Bit Manipulation
  - Table Read/Write
  - Machine Control



Move and Load Instructions

- MOVLW 8-bit ;Load an 8-bit literal in WREG
- MOVLW 0 x F2

- MOVWF F, a ;Copy WREG in File (Data) Reg.
  ; If a = 0, F is in Access Bank
  ;If a = 1, Bank is specified by BSR
- MOVWF 0x25, 0 ;Copy W in Data Reg.25H

- MOVFF fs, fd ;Copy from one Data Reg. to
  ;another Data Reg.
- MOVFF 0x20,0x30 ;Copy Data Reg. 20 into Reg.30

Arithmetic Instructions (1 of 3)

- ADDLW 8-bit ;Add 8-bit number to WREG
- ADDLW 0x32 ;Add 32H to WREG

- ADDWF F, d, a ;Add WREG to File (Data) Reg.
  ;Save result in W if d =0
  ;Save result in F if d = 1
- ADDWF 0x20, 1 ;Add WREG to REG20 and
  ;save result in REG20
- ADDWF 0x20, 0 ;Add WREG to REG20 and
  ;save result in WREG

Fa
W
C

Arithmetic Instructions (2 of 3)

- ADDWFC F, d, a ;Add WREG to File Reg. with ;Carry and save result in W or F
- SUBLW 8-bit ;Subtract WREG from literal
- SUBWF F, d, a ;Subtract WREG from File Reg.
- SUBWFB F, d, a ;Subtract WREG from File Reg. ;with Borrow
- INCF F, d, a ;Increment File Reg.
- DECF F, d, a ;Decrement File Reg.
- COMF F, d, a ;Complement File Reg.
- NEGF F, a ;Take 2's Complement-File Reg.

L-W→W

F-W→Dest.

Arithmetic Instructions (3 of 3)

- MULLW 8-bit ;Multiply 8-bit and WREG **L x W→ PROD**
  ;Save result in PRODH-PRODL
- MULWF F, a ;Multiply WREG and File Reg.
  ;Save result in PRODH-PRODL
- DAW ;Decimal adjust WREG for BCD

;Operations

Example:

```
MOVLW   0xA   ;W=A
DAW           ;W=10
```

Logic Instructions

- ANDLW 8-bit ;AND literal with WREG
- ANDWF F, d, a ;AND WREG with File Reg. and
  ;save result in WREG/ File Reg.

- IORLW 8-bit ;Inclusive OR literal with WREG
- IORWF F, d, a ;Inclusive OR WREG with File Reg.
  ;and save result in WREG/File Reg.

- XORLW 8-bit ;Exclusive OR literal with WREG
- XORWF F, d, a ;Exclusive OR WREG with File Reg.
  ;and save result in WREG/File Reg.

And, XOR, and IOR

A	B	T
0	0	0
0	1	0
1	0	0
1	1	1

($T = A \cdot B$)

AND

```
X X X X   X X X X
0 0 0 0   1 1 1 1
---------------------
0 0 0 0   X X X X
```

Cleared to zero

XOR

```
X X X X   X X X X
0 0 0 0   1 1 1 1
---------------------
X X X X   X X X X
```

If they are the same→0
If they are different→1

Toggled

IOR

```
X X X X   X X X X
0 0 0 0   1 1 1 1
---------------------
X X X X   1 1 1 1
```

Set to one

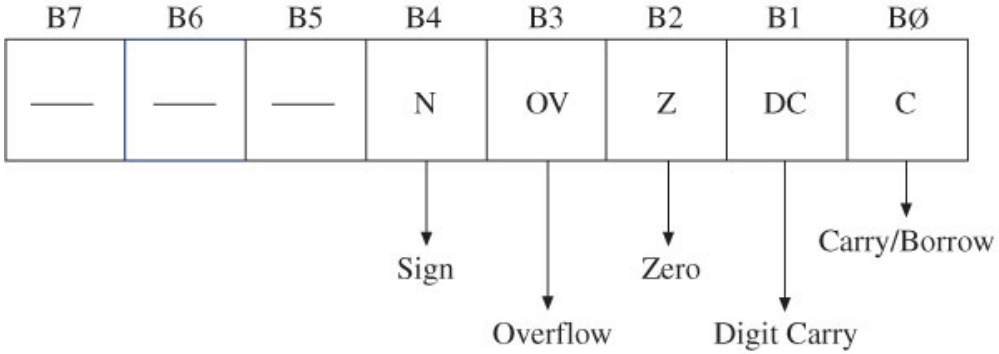
Examples

```
        MOVLW 0x1F
        ANDLW 0xFC     ;clear bits 0 and 1
        IORLW  0xC0    ;set bits 6 and 7
Stop:   GOTO   Stop
```

```
        MOVLW 0x90
        XORLW 0xE0     ;invert left 3 bits
Stop:   GOTO   Stop
```

Branch Instructions

- BC n ;Branch if C flag = 1 within + or – 64 Words
- BNC n ;Branch if C flag = 0 within + or – 64 Words (NO CARRY)
- BZ n ;Branch if Z flag = 1 within + or – 64 Words
- BNZ n ;Branch if Z flag = 0 within + or – 64 Words
- BN n ;Branch if N flag = 1 within + or – 64 Words
- BNN n ;Branch if N flag = 0 within + or – 64 Words
- BOV n ;Branch if OV flag = 1 within + or – 64 Words
- BNOV n ;Branch if OV flag = 0 within + or – 64 Words
- GOTO Address: Branch to 20-bit address unconditionally



Branch Example

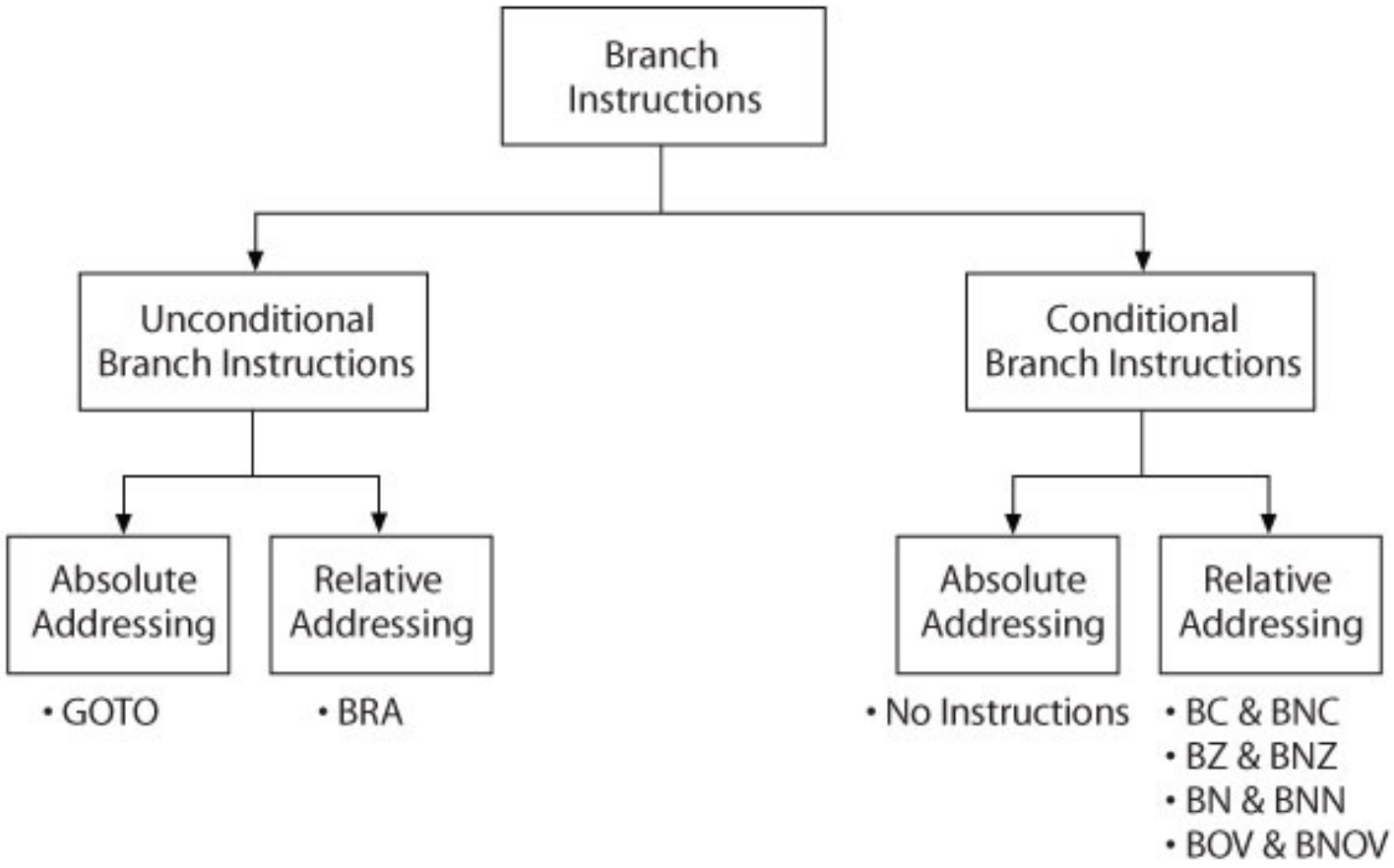
Check the table in The TEXT (p78)!

```
BCN     0xFA    ;Brant-If-No-Carry  to location:
                ; → PC(current_Decimal) + 2 + 2x Decimal(0xFA)
                ; Note 0xFA is signed!
                ;→ PC + 2 + 2(-6)
```

Remember:

1 Word Instruction / 1 Instruction Cycle / 4 Clock Cycles

Branch Instructions



Call and Return Instructions

- RCALL nn ;Call subroutine within +or – 512 words
- CALL 20-bit, s ;Call subroutine
  ;If s = 1, save W, STATUS, and BSR
- RETURN, s ;Return subroutine
- ;If s = 1, retrieve W, STATUS, and BSR
- RETFIE, s ;Return from interrupt
- ;If s = 1, retrieve W, STATUS, and BSR

Review

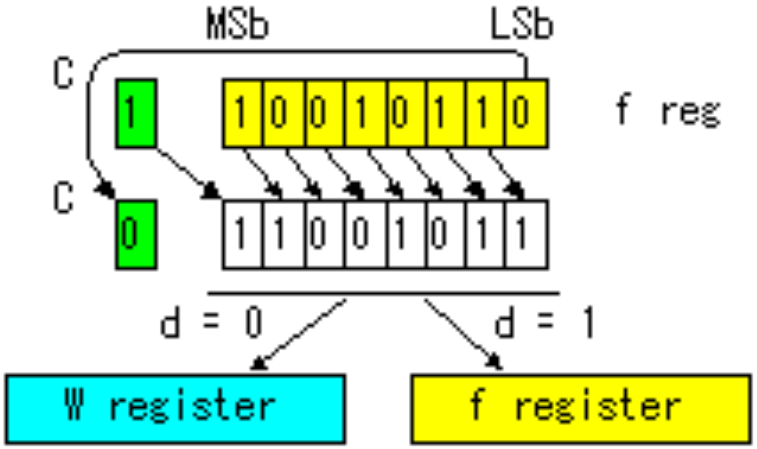
Bit Manipulation Instructions

- BCF F, b, a ;Clear bit b of file register. b = 0 to 7
- BSF F, b, a ;Set bit b of file register. b = 0 to 7
- BTG F, b, a ;Toggle bit b of file register. b = 0 to 7

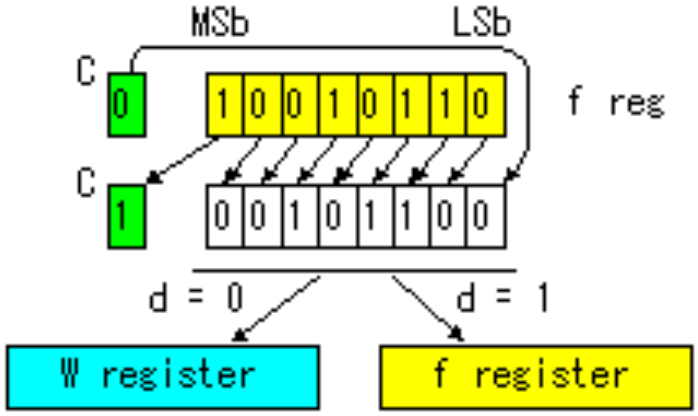
- RLCF F, d, a ;Rotate bits left in file register through
  ; carry and save in W or F register
- RLNCF F, d, a ;Rotate bits left in file register
  ; and save in W or F register
- RRCF F, d, a ;Rotate bits right in file register through
- ; carry and save in W or F register
- RRNCF F, d, a ;Rotate bits right in file register
  ; and save in W or F register

Rotations

Rotate Right through Carry
RRCF



Rotate LEFT through Carry
RLCF



Test and Skip Instructions

- BTFSC F, b, a ;Test bit b in file register and skip the ;next instruction if bit is cleared (b =0)
- BTFSS F, b, a ;Test bit b in file register and skip the ;next instruction if bit is set (b =1)
- CPFSEQ F, a ;Compare F with W, skip if F = W
- CPFSGT F, a ;Compare F with W, skip if F > W
- CPFSLT F, a ;Compare F with W, skip if F < W
- TSTFSZ F, a ;Test F; skip if F = 0

Example

```
        MOVLW   0x1F
        BCF     WREG, 0         ;clear bit 0
        BCF     WREG, 1         ;clear bit 1
        BSF     WREG, 6         ;set bit 6
        BSF     WREG, 7         ;set bit 7
Stop:   GOTO    Stop
```

```
        MOVLW  0x7F             ;load test data
        BTFSS  WREG, 7
        BCF    WREG, 0          ;clear bit 0
Stop:   GOTO   Stop
```

Increment/Decrement and Skip Next Instruction

- DECFSZ F, d, a ;Decrement file register and skip the ;next instruction if F = 0
- DECFSNZ F, d, a ;Decrement file register and skip the ;next instruction if F ≠ 0
- INCFSZ F, d, a ;Increment file register and skip the ;next instruction if F = 0
- INCFSNZ F, d, a ;Increment file register and skip the ;next instruction if F ≠ 0

Table Read/Write Instructions (1 of 2)

- TBLRD* ;Read Program Memory pointed by TBLPTR
  ;into TABLAT
- TBLRD*+ ;Read Program Memory pointed by TBLPTR
  ;into TABLAT and increment TBLPTR
- TBLRD*- ;Read Program Memory pointed by TBLPTR
  ;into TABLAT and decrement TBLPTR
- TBLRD+* ; Increment TBLPTR and Read Program
  ; Memory pointed by TBLPTR into TABLAT

Review

Table Read/Write Instructions (2 of 2)

- TBLWT* ;Write TABLAT into Program Memory pointed ;by TBLPTR
- TBLWT*+ ; Write TABLAT into Program Memory pointed ;by TBLPTR and increment TBLPTR
- TBLWT*- ; Write TABLAT into Program Memory pointed ;by TBLPTR and decrement TBLPTR
- TBLWT+* ; Increment TBLPTR and Write TABLAT into ; Program Memory pointed by TBLPTR

Review

Machine Control Instructions

- CLRWDT ;Clear Watchdog Timer
  - Helps recover from software malfunction
  - Uses its own free-running on-chip RC oscillator
  - WDT is cleared by CLRWDT instruction
- RESET ;Reset all registers and flags
  - When voltage < a particular threshold, the device is held in reset
  - Prevents erratic or unexpected operation
- SLEEP ;Go into standby mode
- NOP ;No operation

Sleep Mode

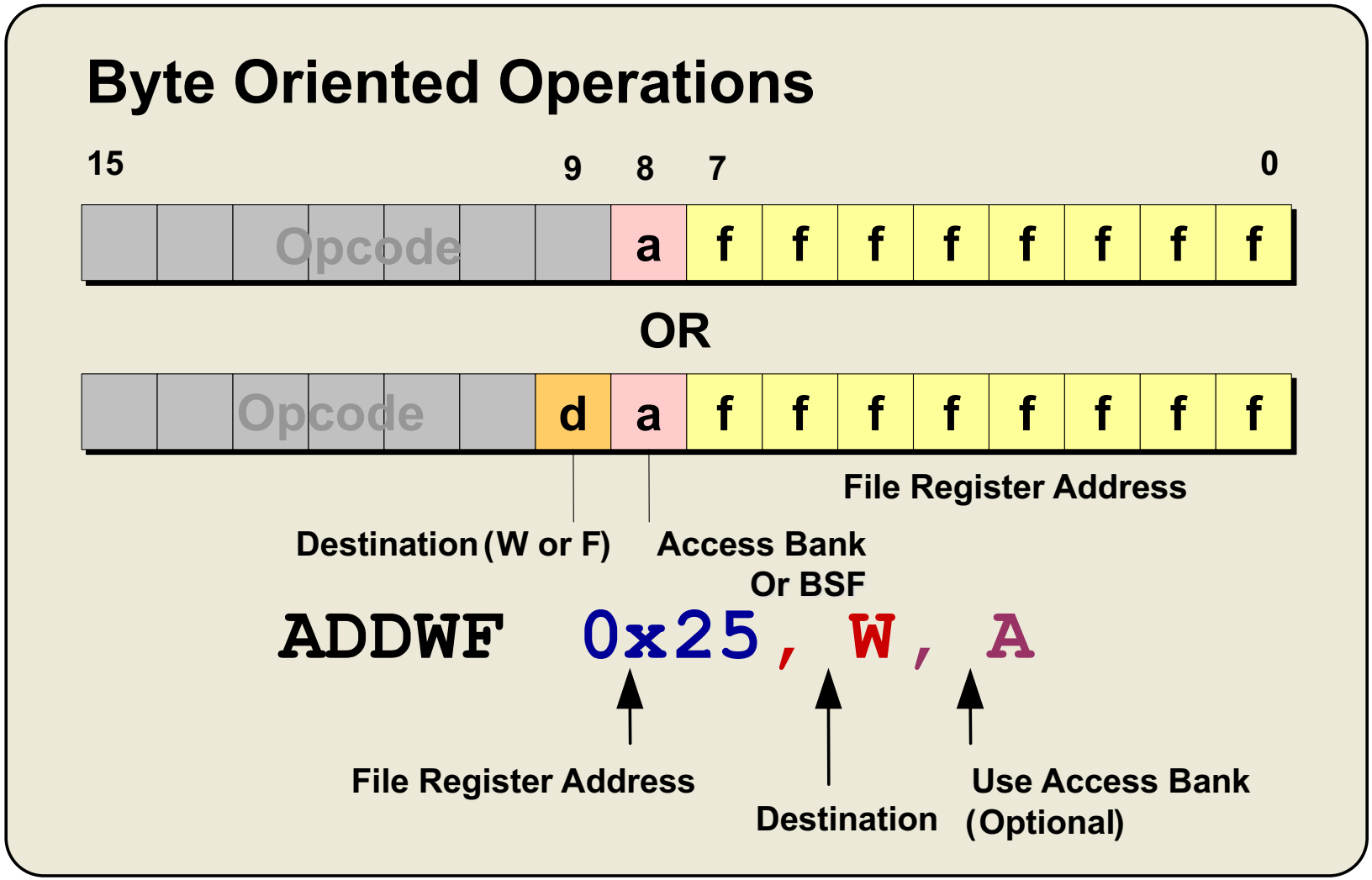
- The processor can be put into a power-down mode by executing the SLEEP instruction
  - System oscillator is stopped
  - Processor status is maintained (static design)
  - Watchdog timer continues to run, if enabled
  - Minimal supply current is drawn - mostly due to leakage (0.1 - 2.0µA typical)

Events that wake processor from sleep	
MCLR	Master Clear Pin Asserted (pulled low)
WDT	Watchdog Timer Timeout
INT	INT Pin Interrupt
TMR1	Timer 1 Interrupt (or also TMR3 on PIC18)
ADC	A/D Conversion Complete Interrupt
CMP	Comparator Output Change Interrupt
CCP	Input Capture Event
PORTB	PORTB Interrupt on Change
SSP	Synchronous Serial Port (I^2C Mode) Start / Stop Bit Detect Interrupt
PSP	Parallel Slave Port Read or Write

Instruction Format (1 of 3)

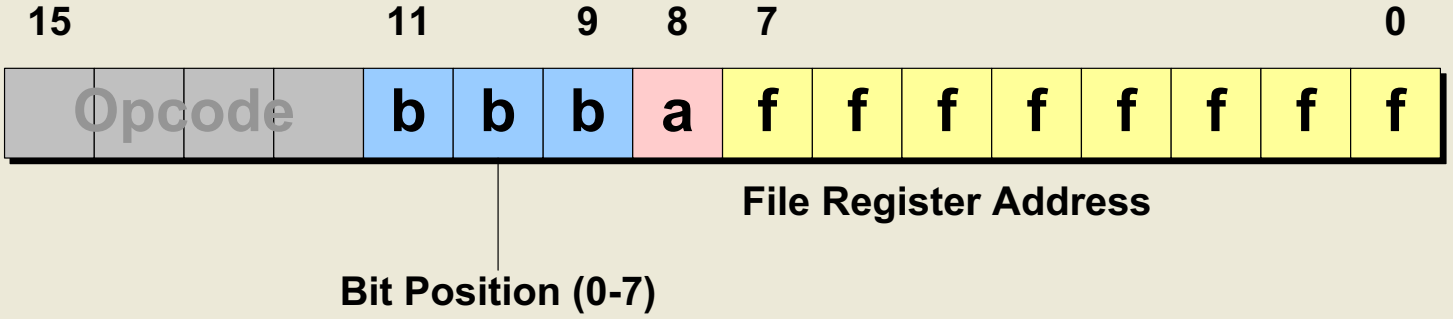
- The PIC18F instruction format divided into four groups
  - Byte-Oriented operations
  - Bit-Oriented operations
  - Literal operations
  - Branch operations

PIC18 Instruction Set Overview –



Instruction Set Overview

Bit Oriented Operations




```
BSF  0x25, 3, A
```

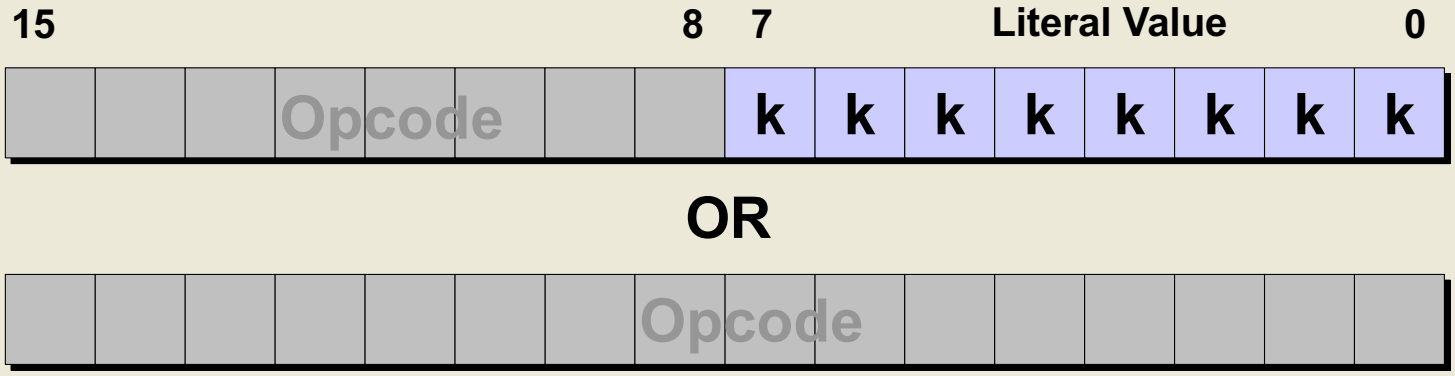
File Register Address

Bit Position

Access Bank (Optional)

Instruction Set Overview

Literal and Control Operations



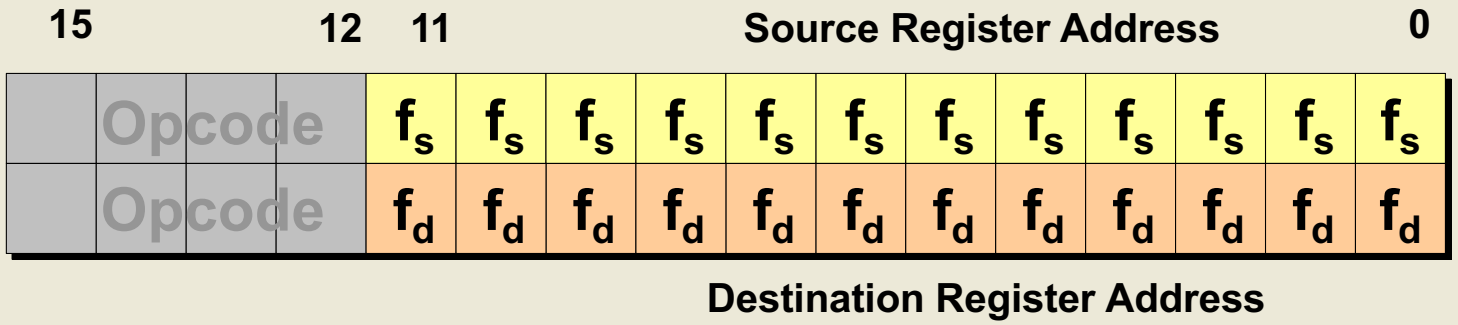

```
MOVLW   0x25
```

Literal Value

Instruction Set Overview

- Two-word instruction

Byte to Byte Move Operations(2 Words)



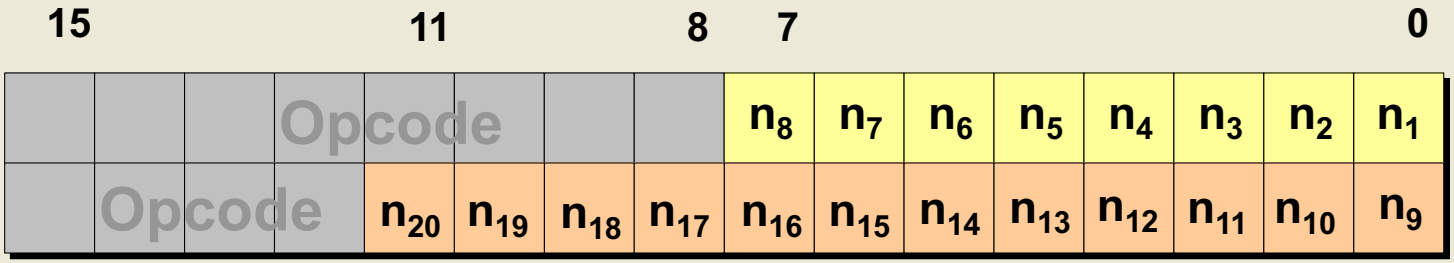

```
MOVFF   0x125, 0x140
```

Source Address

Destination Address

Instruction Set Overview

Call and Goto Operations (2 Words)




```
CALL    0x1125
```

Subroutine Address

Example (do it in class!)

```
ORG       0x20
REG0      EQU    0x00
REG1      EQU    0x01
REG2      EQU    0x02

MOVLW     0x37
MOVWF     REG0,0
MOVLW     0x92
MOVWF     REG1,0
ADDWF     REG0,0
MOVWF     REG2, 0
SLEEP
```

Explain what this program does, specify PC value for each line, which flags are changed as the program is executed.

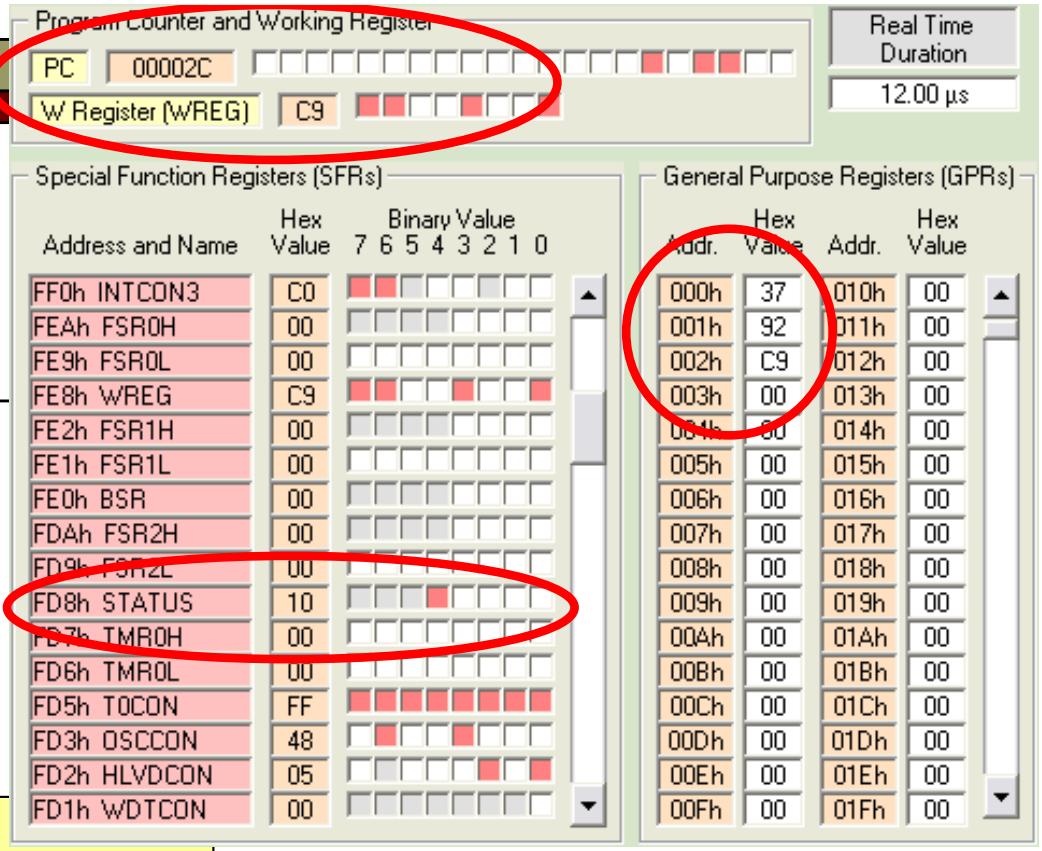
Command	PC	REG0,1,2	STATUS	Time

Example

```
ORG       0x20
REG0      EQU    0x00
REG1      EQU    0x01
REG2      EQU    0x02

MOVLW     0x37
MOVWF     REG0,0
MOVLW     0x92
MOVWF     REG1,0
ADDWF     REG0,0
MOVWF     REG2, 0
SLEEP
```

→W=0x37
→REG0=0x37
→W=0x92
→REG1=0x92
→W=37+92=C9
→REG2=C9



NOTES:
Each 1W instruction take 4 clock periods
Use the STOPWATCH in the simulator!

Next QUIZ: Review the Following

- Arithmetic commands (ADDLW, ADDWF, ADDWFC, SUBLW, SUBWF, SUBWB, INC, DEC, MULLW, NEGF, COMPF)
- Logical commands, ANDLW, XOR, IOR, AND)
- MOVE & Copy (MOVLW, MOVFF, MOVWF, CLR, SETF)
- Branches (BC, BNC, BZ, BNZ, BOV, BRA, GOTO)
- Bit manipulations (BCF, BSF, BTG, RLCF, RRCF)
- Make sure you know about flags.
- Make sure you can do the homework assignment