PIC18F Programming Model and Instruction Set

ELEC 330

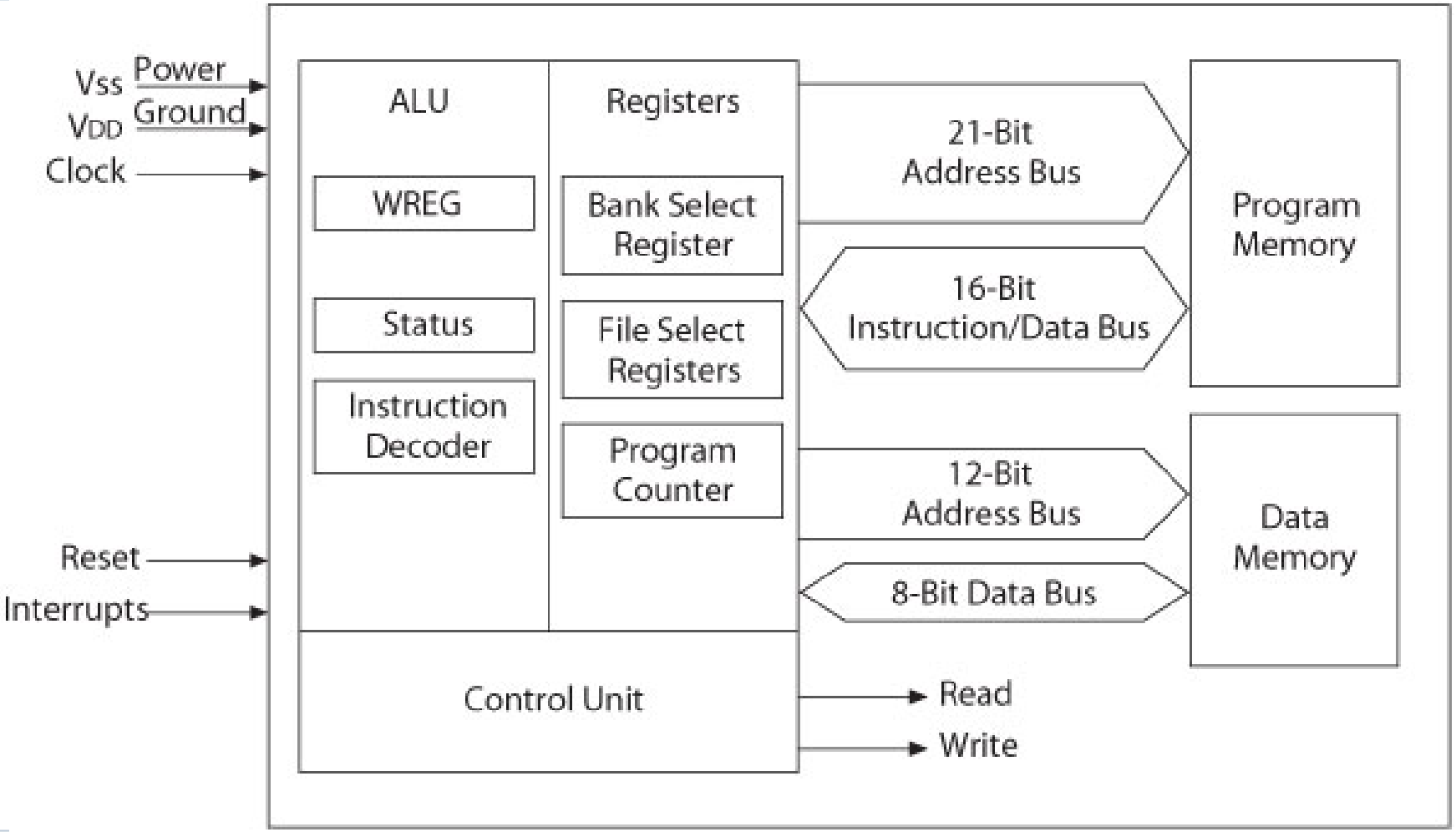
Digital Systems Engineering

Dr. Ron Hayne

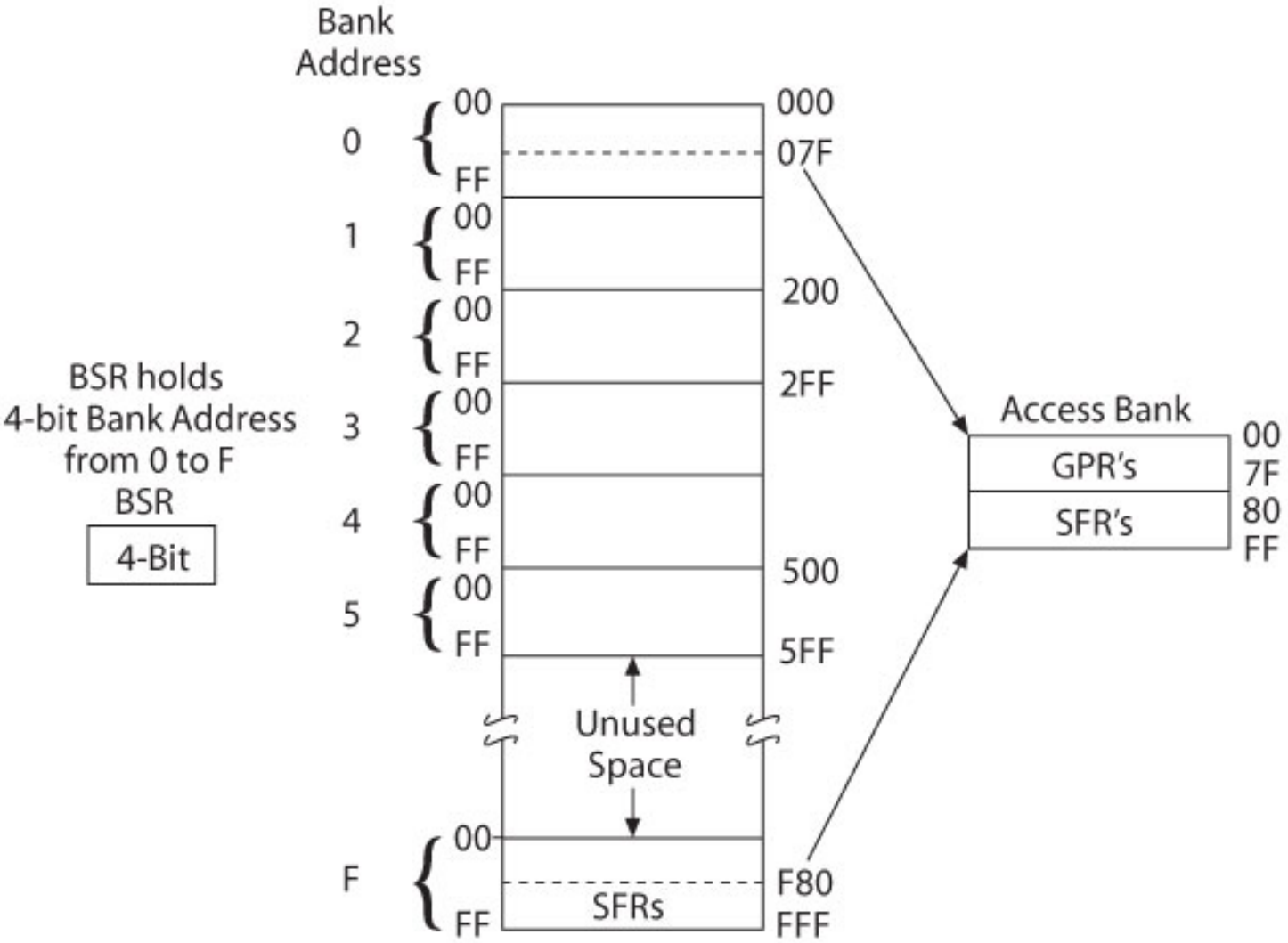
Images Courtesy of Ramesh Gaonkar and Delmar Learning

THE CITADEL
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Review: MPU and Memory



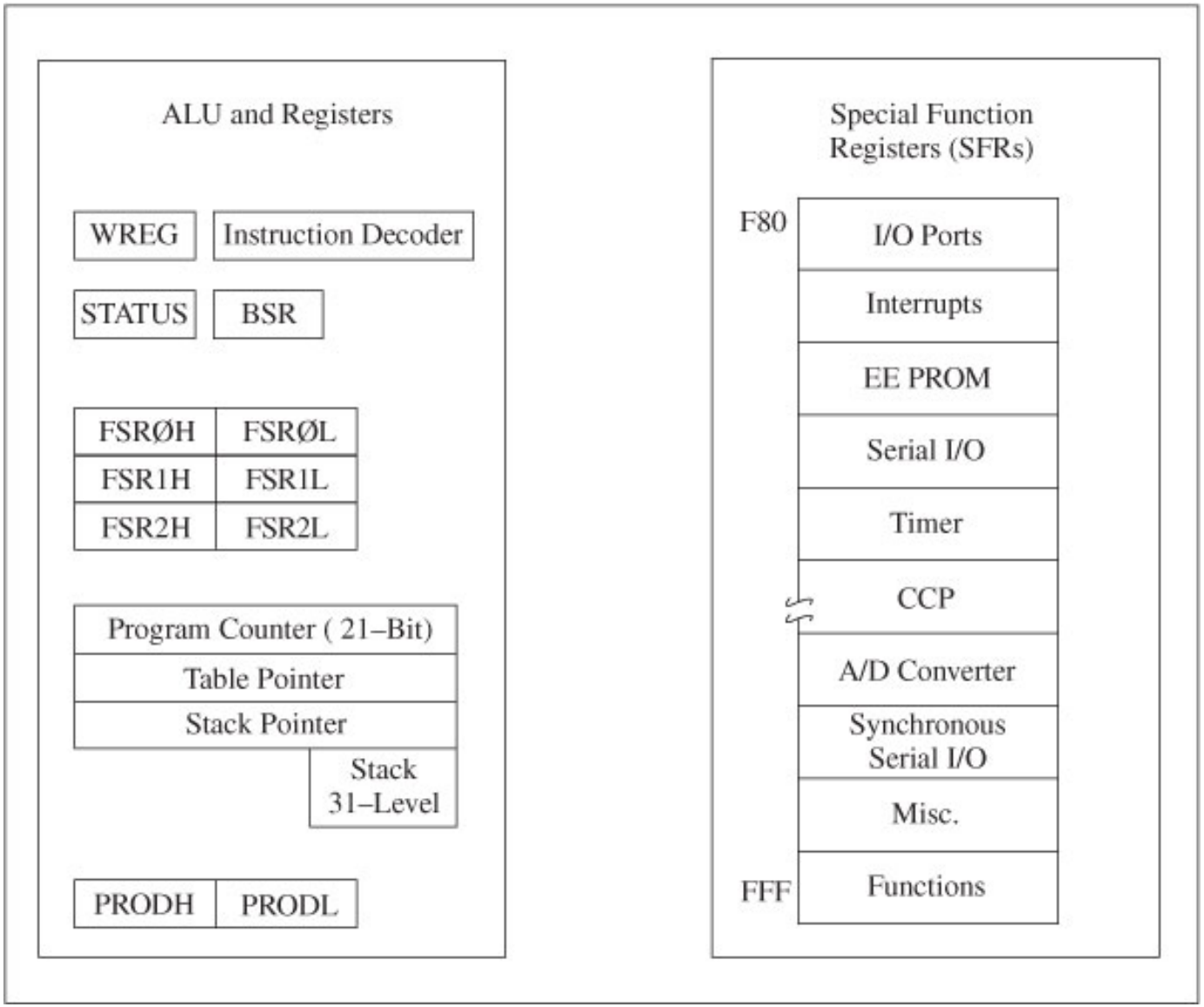
Review: Data Memory



PIC18F Programming Model

- Representation of the internal architecture necessary to write assembly language programs
- Divided into two groups
  - Arithmetic Logic Unit (ALU) and Registers
    - From Microprocessor Unit (MPU)
  - Special Function Registers (SFRs)
    - From Data (File) Memory

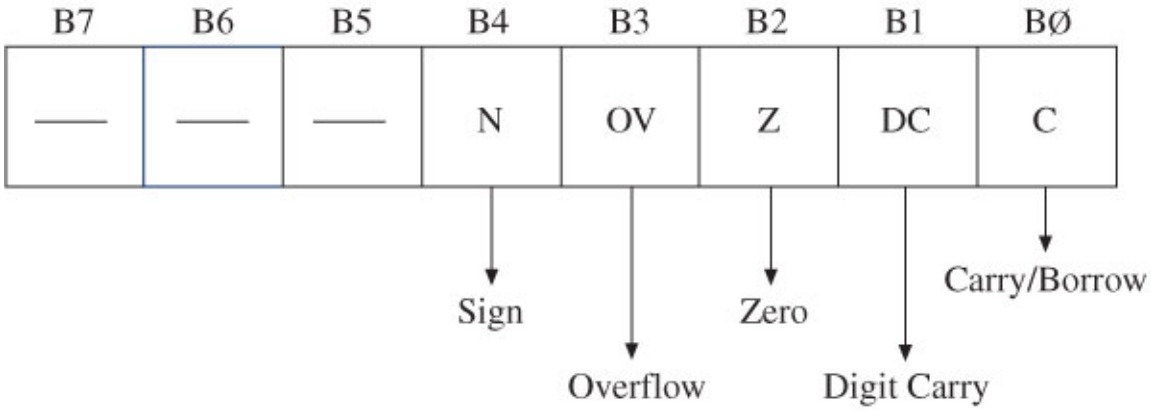
PIC18F Programming Model



ALU

- Instruction Decoder
  - 16-bit Instructions
- Table Latch
  - 8-bit Data
- STATUS: Flag Register
  - 5 individual bits called flags
- WREG (W): Working Register
  - 8-bit Accumulator
- Product
  - 16-bit Product of 8-bit by 8-bit Multiply

Flags in Status Register



- N (Negative Flag)
  - Set when bit B7 is one as the result of an arithmetic/logic operation
- OV (Overflow Flag)
  - Set when result of an operation of signed numbers goes beyond 7-bits
- Z (Zero Flag)
  - Set when result of an operation is zero
- DC (Digit Carry Flag) (Half Carry)
  - Set when carry generated from Bit3 to Bit4 in an arithmetic operation
- C (Carry Flag)
  - Set when an addition generates a carry (out)

Registers

- Program Counter (PC)
  - 21-bit register used as a pointer to Program Memory during program execution
- Table Pointer
  - 21-bit register used as a pointer to copy bytes between Program Memory and data registers
- Stack Pointer (SP)
  - 5-bit register used to point to the stack
- Stack
  - 31 registers used for temporary storage of memory addresses during execution of a subroutines

Registers

- BSR: Bank Select Register (0_H to F_H)
  - 4-bit Register
    - Provides upper 4-bits of 12-bit address of Data Memory
- FSR: File Select Registers
  - FSR0, FSR1, and FSR2
  - FSR: composed of two 8-bit registers
    - FSRH and FSRL
  - Used as pointers for Data Memory
    - Holds 12-bit address of data register

Special Function Registers

- SFRs: Table 3-1
  - Data registers associated with I/O ports, support devices, and processes of data transfer
    - I/O Ports (A to E)
    - Interrupts
    - EEPROM
    - Serial I/O
    - Timers
    - Capture/Compare/PWM (CCP)
    - Analog-to-Digital (A/D) Converter

PIC18 Instruction Set

- Includes 77 instructions
  - 73 one word (16-bit) long
  - 4 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

Addressing Modes

- Method of specifying of an operand
  - Immediate (Literal) addressing
    - The operand is a number that follows the opcode
  - Direct addressing
    - The address of the operand is a part of the instruction
  - Indirect addressing
    - An address is specified in a register (pointer) and the MPU looks up the address in that register

Move and Load Instructions

- MOVLW 8-bit ;Load an 8-bit Literal into WREG
  MOVLW 0xF2 ;Load $F2_H$ into W

- MOVWF F,a ;Copy WREG into File (Data) Reg.
  ;If a = 0, F is in Access Bank
  ;If a = 1, Bank is specified by BSR
  MOVWF 0x25,0 ;Copy W into F $Reg25_H$
  MOVWF 0x25 ;Alternate format

- MOVFF fs,fd ;Copy from one File Reg. to ;another File Reg.
  MOVFF 0x20,0x30 ;Copy F $Reg20_H$ into $Reg30_H$

Move and Load Instructions

- MOVF F,d,a ;Copy F into itself or W
  ;If d = 0 (or W), destination is W
  ;If d = 1 (or F), destination is F
  ;Affects N & Z flags
  MOVF 0x25,0,0 ;Copy F $Reg25_H$ into W
  MOVF 0x25,W ;Alternate format

- CLRF F,a ;Clear F Reg.
  ;Sets Z flag
  CLRF 0x25 ;Clear $Reg25_H$

- SETF F,a ;Sets all bits to 1 in F Reg.

Points to Remember

- Each instruction has two parts
  - Opcode and Operand
- When instructions copy data from one register to another, the source is not modified
- In general, these instructions do not affect flags
  - Except CLRF and MOVF

Arithmetic Instructions

- ADDLW 8-bit ;Add 8-bit number to W & set flags
  ADDLW 0x32 ;Add 32_H to W

- ADDWF F,d,a ;Add W to F & set flags
  ;Save result in W if d = 0 (or W)
  ;Save result in F if d = 1 (or F)
  ADDWF 0x20,0 ;Add W to $REG20_H$ and ;save result in W
  ADDWF 0x20,W ;Alternate format

  ADDWF 0x20,1 ;Add W to $REG20_H$ and ;save result in $REG20_H$
  ADDWF 0x20,F ;Alternate format

Arithmetic Instructions

- ADDWFC F,d,a ;Add W to F with carry
;and save result in W or F
- SUBLW 8-bit ;Subtract W from literal
- SUBWF F,d,a ;Subtract W from F
- SUBWFB F,d,a ;Subtract W from F with borrow
- INCF F,d,a ;Increment F
- DECF F,d,a ;Decrement F
- NEGF F,a ;Take 2's Complement of F

Arithmetic Instructions

- MULLW 8-bit ;Multiply 8-bit Literal and W
  ;Save result in PRODH:PRODL
- MULWF F,a ;Multiply W and F
  ;Save result in PRODH:PRODL
- DAW ;Decimal adjust W for BCD
  ;Addition

Points to Remember

- Arithmetic instructions
  - Can perform operations on W and 8-bit literals
    - Save the result in W
  - Can perform operations an W and F
    - Save the result in W or F
  - In general, affect all flags

Logic Instructions

- COMF F,d,a ;Complement (NOT) F
  ;and save result in W or F
- ANDLW 8-bit ;AND Literal with W
- ANDWF F,d,a ;AND W with F and ;save result in W or F
- IORLW 8-bit ;Inclusive OR Literal with W
- IORWF F,d,a ;Inclusive OR W with F ;and save result in W or F

  IORWF 0x12,F ;OR W with $REG12_H$ and
  ;save result in $REG12_H$
- XORLW 8-bit ;Exclusive OR Literal with W
- XORWF F,d,a ;Exclusive OR W w/ F ;and save result in W or F

Points to Remember

- Logic instructions
  - Can perform operations on W and 8-bit literals
    - Save the result in W
  - Can perform operations an W and F
    - Save the result in W or F
  - In general, affect only two flags: N and Z

Branch Instructions

- BC n ;Branch if C flag = 1, + or – 64 Words
  ;to PC+2+2n
  BC 5 ;Branch on Carry to PC+2+10
  BC Label ;Alternate: Branch to Label
- BNC n ;Branch if C flag = 0
- BZ n ;Branch if Z flag = 1
- BNZ n ;Branch if Z flag = 0
- BN n ;Branch if N flag = 1
- BNN n ;Branch if N flag = 0
- BOV n ;Branch if OV flag = 1
- BNOV n ;Branch if OV flag = 0
- BRA nn ;Branch always, + or – 512 Words

Branch Example

Address	Label	Opcode	Operand	Comment
000020	START:	MOVLW	BYTE1	;Load BYTE1 into W
000022		MOVWF	REG0	;Save into REG0
000024		MOVLW	BYTE2	;Load BYTE2 into W
000026		MOVWF	REG1	;Save into REG1
000028		ADDWF	REG0,W	;Add REG0 to REG1
00002A		BNC	SAVE	;Branch if no carry
00002C		MOVLW	0x00	;Clear W
00002E	SAVE:	MOVWF	REG2	;Save Result
000030		SLEEP		

Call and Return Instructions

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

Points to Remember

- Eight conditional relative branch instructions
  - Based on four flags
  - Range is ± 64 words
- Unconditional relative branch instruction
  - Range is ± 512 words
- If the operand is positive, the jump is forward
- If negative, the jump is backward

Bit Manipulation Instructions

- BCF F,b,a ;Clear bit b of F, b = 0 to 7
  BCF 0x2,7 ;Clear bit 7 of Reg2

- BSF F,b,a ;Set bit b of F, b = 0 to 7
- BTG F,b,a ;Toggle bit b of F, b = 0 to 7
- RLCF F,d,a ;Rotate bits left in F through
  ;carry and save in W or F
- RLNCF F,d,a ;Rotate bits left in F
  ;and save in W or F
- RRCF F,d,a ;Rotate bits right in F through
- ;carry and save in W or F
- RRNCF F,d,a ;Rotate bits right in F
  ;and save in W or F

Points to Remember

- Any bit in a File (data) register
  - Set, reset, or complemented
- There are two types of rotate instructions
  - 8-bit and 9-bit (include C)
  - Any file (data) register can be rotated left or right
    - Saved in W or F

Test and Skip Instructions

- BTFSC F,b,a ;Test bit b in F and skip the
  ;next instruction if bit is cleared (bit=0)

  BTFSC 0x2,7 ;Test bit B7 in REG2

  ;if B7=0 then skip next instruction
- BTFSS F,b,a ;Test bit b in F and skip the
  ;next instruction if bit is set (bit=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

Increment/Decrement and Skip Next Instruction

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

Points to Remember

- Any File (data) register or single bit in a File (data) register can be tested for 0
- A File (data) register can be compared with W for equality, greater than, and less than
- A File (data) register can be incremented or decremented and tested for 0
- If a condition is met, the next instruction is skipped (no flags are affected)

Table Read/Write Instructions

- 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

Table Read/Write Instructions

- 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

Machine Control Instructions

- CLRWDT ;Clear Watchdog Timer
- RESET ;Reset all registers and flags
- SLEEP ;Go into standby mode
- NOP ;No operation

Instruction Format

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

Instruction Format

- Byte-oriented instruction – ADDWF F, d, a

ADDWF 0x1,F ;Add W to REG1, save in REG1

B15 B10	B9	B8	B7 B0
6-bit Opcode	d	A	F = 8-bit File Register Address
0 0 1 0 0 1	1	0	0000 0001

- Bit-oriented instruction – BCF F, b, a

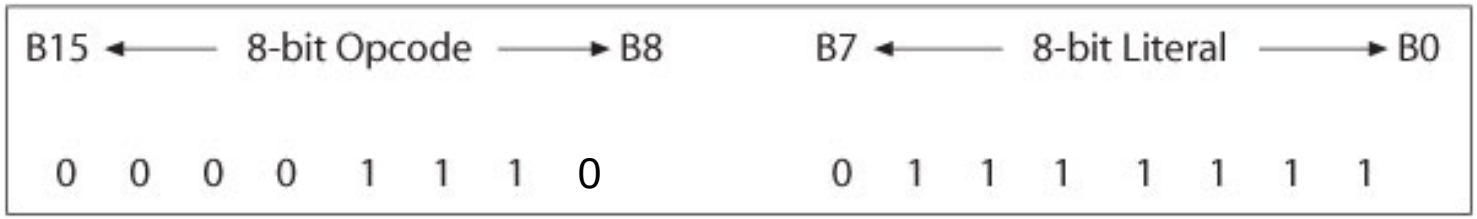
BCF 0x15,7 ;Clear bit7 in $REG15_H$

B15 B12	B11 B9	B8	B7 B0
4-bit Opcode	b b b	a	F = 8-bit File Register Address
1 0 0 1	1 1 1	0	0 0 0 1 0 1 0 1

Instruction Format

- Literal instruction — MOVLW k

  MOVLW 0x7F ;Load $7F_H$ into W



- Branch instruction — BC n

  BC 0x15 ;Branch if carry $+15_H$ words

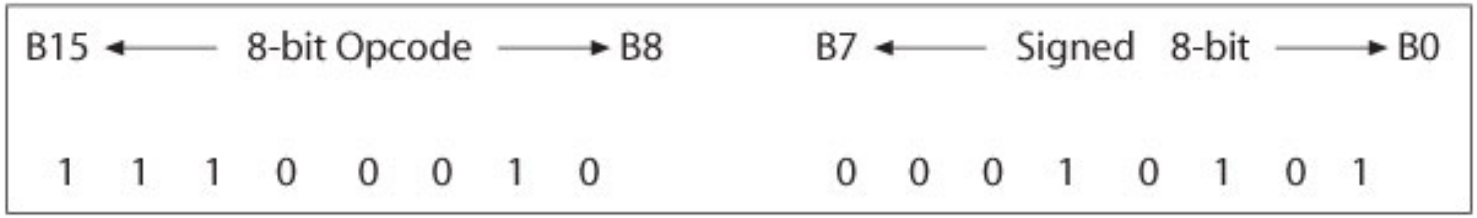


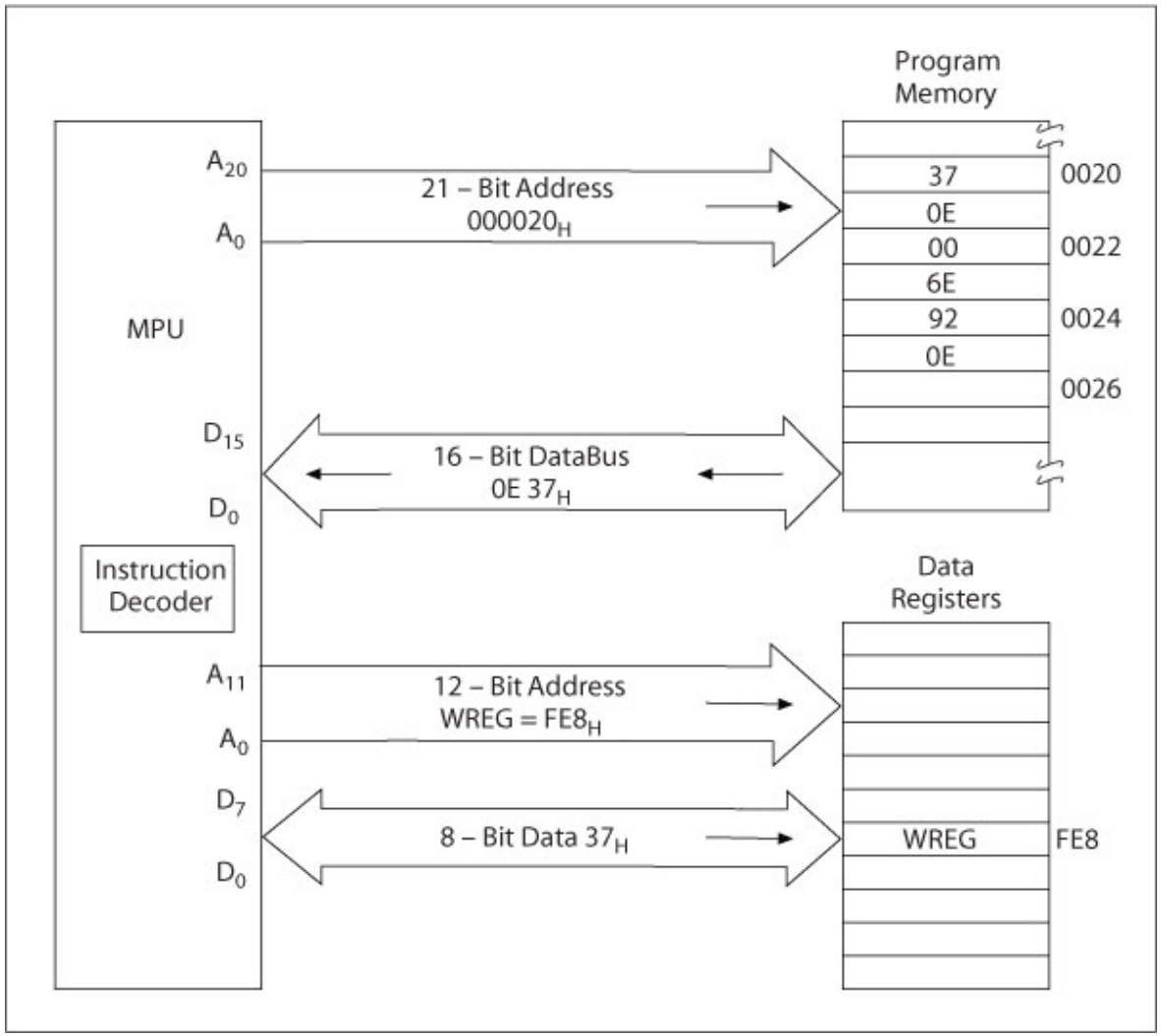
Illustration: Addition

- Problem Statement
  - Load two bytes (37_H and 92_H) in registers REG0 and REG1
  - Add the bytes and store the sum in REG2

Address	Hex	Opcode	Operand	Comments
0020	0E37	MOVLW	0x37	;Load first byte in W
0022	6E00	MOVWF	REG0	;Save first byte in REG0
0024	0E92	MOVLW	0x92	;Load second byte in W
0026	6E01	MOVWF	REG1	;Save second byte in REG1
0028	2400	ADDWF	REG0,W	;Add bytes and save sum in W
002A	6E02	MOVWF	REG2	;Save sum in REG2
002C	0003	SLEEP		;Power Down

Bus Contents

- Execution of the instruction: MOVLW 0x37



Pipeline Fetch and Execution

Instruction Cycles:

	Instruction Cycle 0	Instruction Cycle 1	Instruction Cycle 2	Instruction Cycle 3
	$Q_1 Q_2 Q_3 Q_4$	$Q_1 Q_2 Q_3 Q_4$	$Q_1 Q_2 Q_3 Q_4$	$Q_1 Q_2 Q_3 Q_4$
Fetch	$0E\ 37_H$	$6E\ 00_H$	$0E\ 92_H$	
Execute		$37_H \longrightarrow W$	$W \longrightarrow Reg0$	$92_H \longrightarrow W$
	Fetch 1	Fetch 2	Fetch 3	
		Execute 1	Execute 2	Execute 3