Assembly Language Instructions

ELEC 330

Digital Systems Engineering

Dr. Ron Hayne

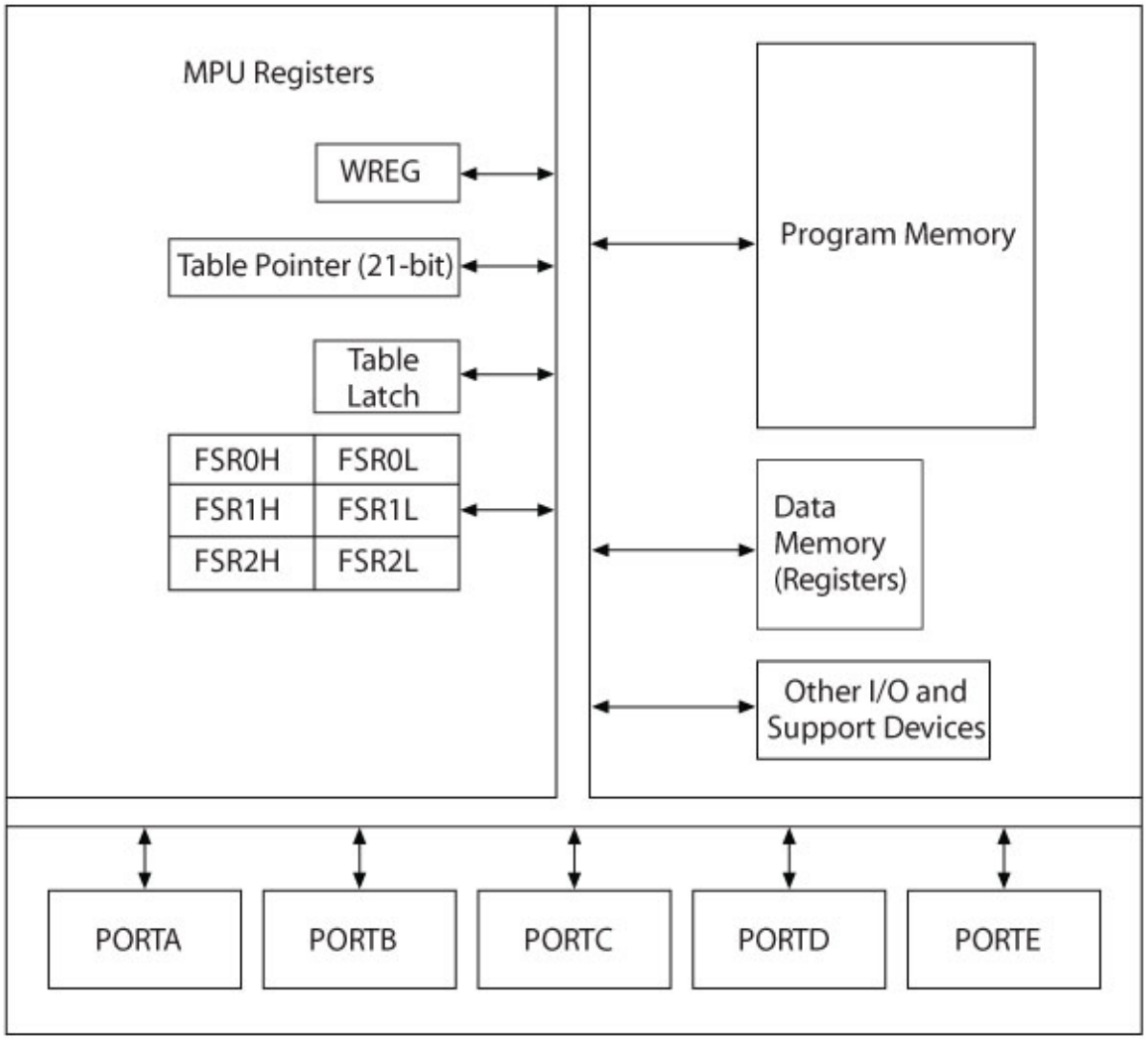
Images Courtesy of Ramesh Gaonkar and Delmar Learning

THE CITADEL
DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

Data Copy Operations

- Load 8-bit data directly in WREG
- Copy data between WREG and data (file) register including I/O ports
- Copy data from one data (file) register to another data (file) register
- Clear or set all data bits in data (file) register
- Exchang low-order four bits (nibble) with high-order four bits in data (file) register

Frequently Used Registers



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

MOV (Copy) Operations

Instructions

- MOVLW 8-bit
- MOVWF F,a
- MOVF F,d,a
- MOVFF Fs,Fd

Examples

- MOVLW 0xF2
  Load F2H into WREG
- MOVWF REG1
  Copy WREG into REG1
- MOVF REG1,F
  Copy REG1 into REG1
- MOVF REG1,W
  Copy REG1 into WREG
- MOVFF REG1,REG2
  Copy REG1 into REG2

SET/CLR Instructions

Instructions

- CLRF F,a
- SETF F,a
- SWAPF F,d,a

Examples

- CLRF REG1

  Clear REG1
- SETF REG1

  Set all bits in REG1
- SWAPF REG1,F

  Exchange low and high nibbles in REG1

Points to Remember

- 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 F
- The letter “d” represents the destination
  - If d = 0 or W, result saved in WREG
  - If d = 1 or F, result saved in File (data) register

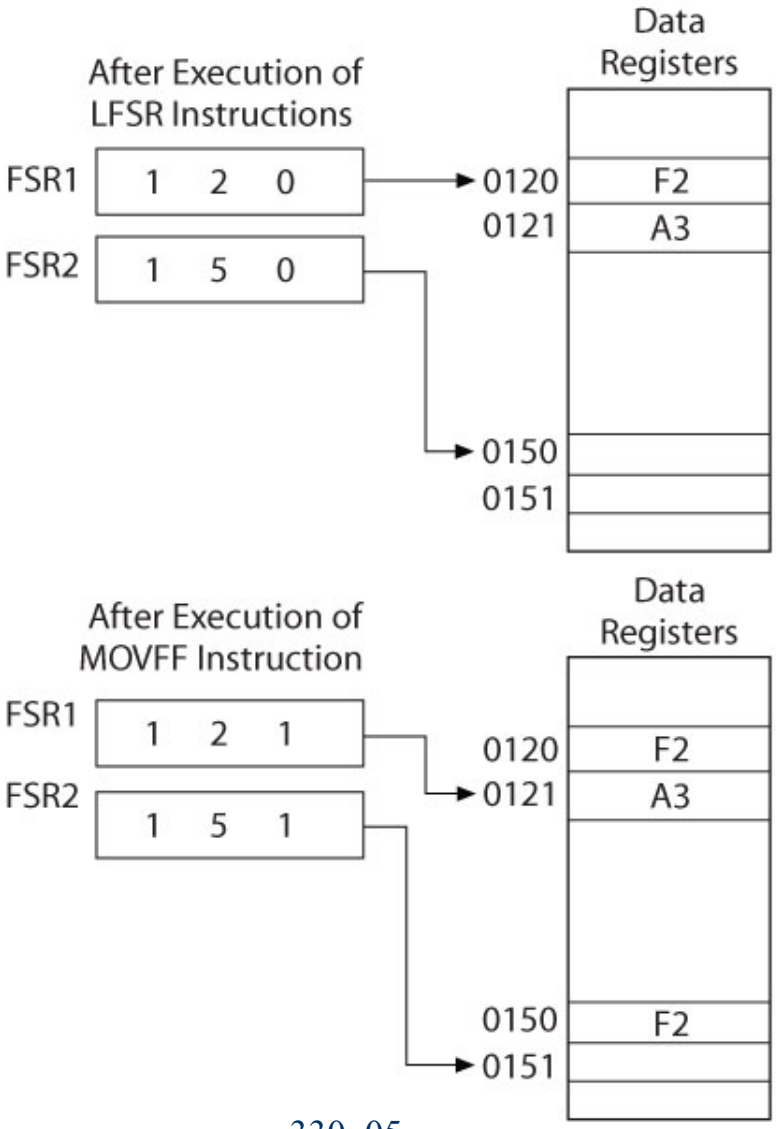
File Select Registers as Pointers

- Three registers: FSR0, FSR1, and FSR2
  - LFSR F,12-bit Load 12-bit address into FSR
- Each can be used in five different formats
  - INDF0 Use FSR0 as pointer
  - POSTINC0 Use FSR0 as pointer and increment FSR0
  - POSTDEC0 Use FSR0 as pointer and decrement FSR0
  - PREINC0 Increment FSR0 first and use as pointer
  - PLUSW0 Add W to FSR0 and use as pointer

Pointer Example

Opcode	Operands	Comments
LFSR	FSR1,0x0120	;Load 120H into FSR1
LFSR	FSR2,0x0150	;Load 150H into FSR2
MOVFF	POSTINC1,POSTINC2	;Copy data in register 120H into register 150H and increment both FSRs

Indirect Addressing

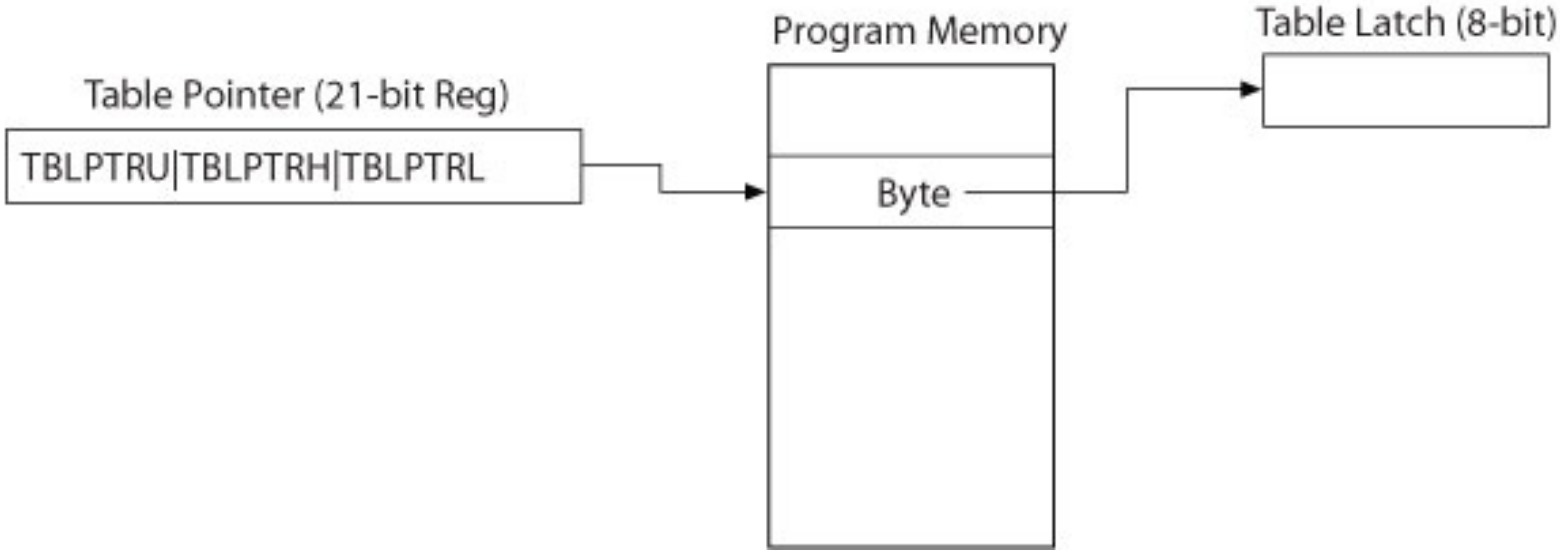


Using Table Pointers to Copy Data

- TBLRD*
  - Copy from Program Memory into Table Latch Using Table Pointer
- TBLRD*+
  - Copy from Program Memory into Table Latch and Increment Table Pointer
- TBLRD*-
  - Copy from Program Memory into Table Latch and Decrement Table Pointer
- TBLRD+*
  - Increment Table Pointer first and then copy from Program Memory into Table Latch

Copying Data

Instruction TBLRD* Copies Data From Program Memory to Table Latch



Example 5.3

- The program memory location BUFFER (at address 000040_H) holds the byte $F6_H$.
- Write the assembly language instructions to copy the byte from BUFFER to the data register REG10 (at address 010_H)

Table Pointer Example

Label	Opcode	Operand	Comment
REG10	EQU	0x10	;Data Register
	MOVLW	UPPER BUFFER	;Load upper bits of BUFFER
	MOVWF	TBLPTRU	
	MOVLW	HIGH BUFFER	;Load high byte of BUFFER
	MOVWF	TBLPTRH	
	MOVLW	LOW BUFFER	;Load low byte of BUFFER
	MOVWF	TBLPTRL	
	TBLRD*		;Copy data to Table Latch
	MOVF	TABLAT,W	;Copy Table Latch to WREG
	MOVWF	REG10	;Copy WREG to REG10
	SLEEP		
	ORG	0x40	
BUFFER	DB	0xF6	;Data Byte
	END		

Arithmetic Operations

- PIC18F MPU
  - Add
  - Subtract
  - Multiply
  - Negate (2s complement)
  - Increment
  - Decrement

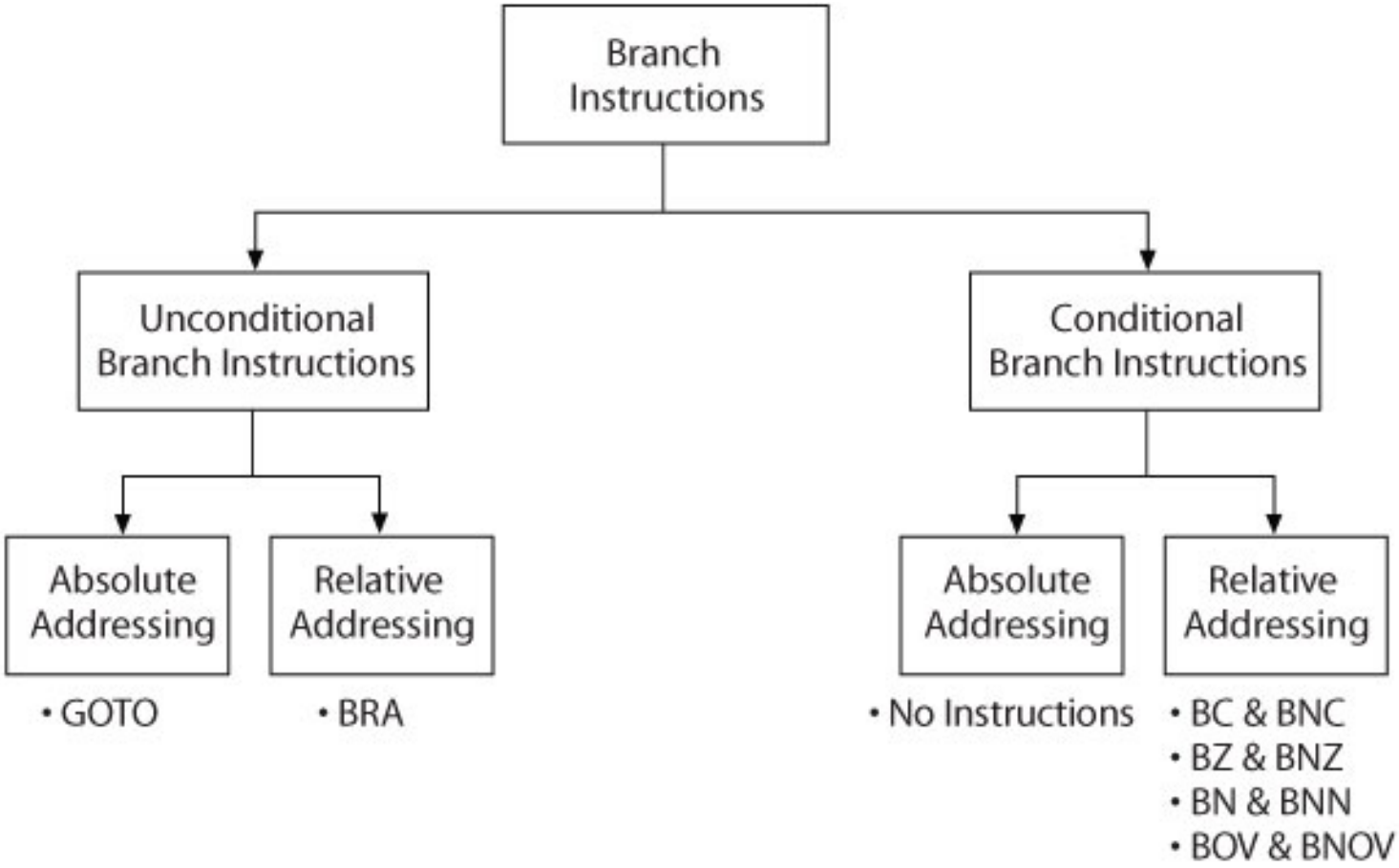
Points to Remember

- 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

Redirection of Program Execution

- Three groups of instructions that change the direction of execution
  - Branch
  - Skip
  - Call (discussed in Chapter 7)

Branch Instructions



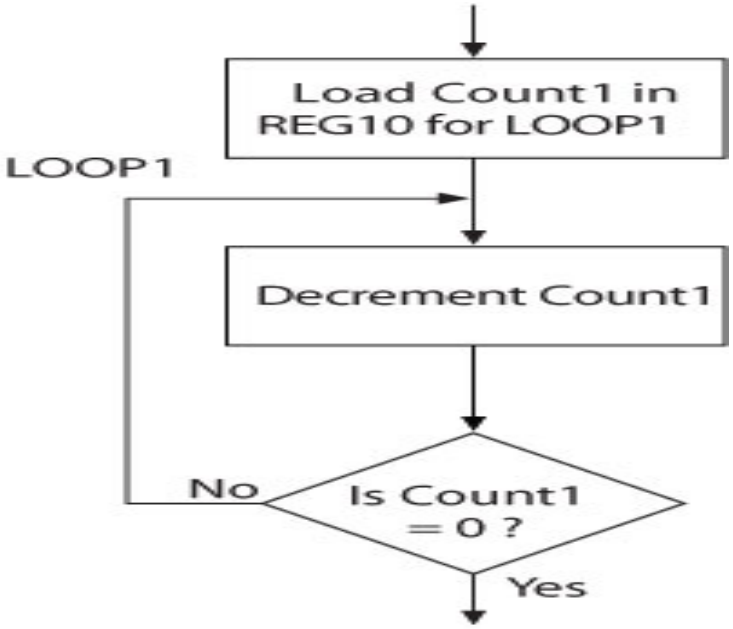
Points to Remember

- Branch instructions use relative addressing
  - If the operand is positive, the jump is forward
  - If negative, the jump is backward
- These instructions do not affect flags
  - Flags set by previous instructions used to make decisions

Arithmetic Instructions with Skip

- Compare File (Data) Register with W:
  - CPFSEQ F,a ;Skip next instruction if F = W
  - CPFSGT F,a ;Skip next instruction if F > W
  - CPFSLTF,a ;Skip next instruction if F < W
- Increment File (Data) Register:
  - INCFSZF,d,a ;Skip next instruction if F = 0
  - INFSNZF,d,a ;Skip next instruction if F ≠ 0
- Decrement File (Data) Register:
  - DECFSZ F,d,a ;Skip next instruction if F = 0
  - DCFSNZ F,d,a ;Skip next instruction if F ≠ 0

Flowchart for Loop



Loop Example

Label	Opcode	Operand	Comments
COUNT1	EQU	0x01	;Counter is REG01
	ORG	0x20	
START:	MOVLW	0x05	;Initialize Counter to 5
	MOVWF	COUNT1	
LOOP1:	DECF	COUNT1,F	;Decrement Counter
	BNZ	LOOP1	;Count1 = 0?
	SLEEP		;Done
	END		

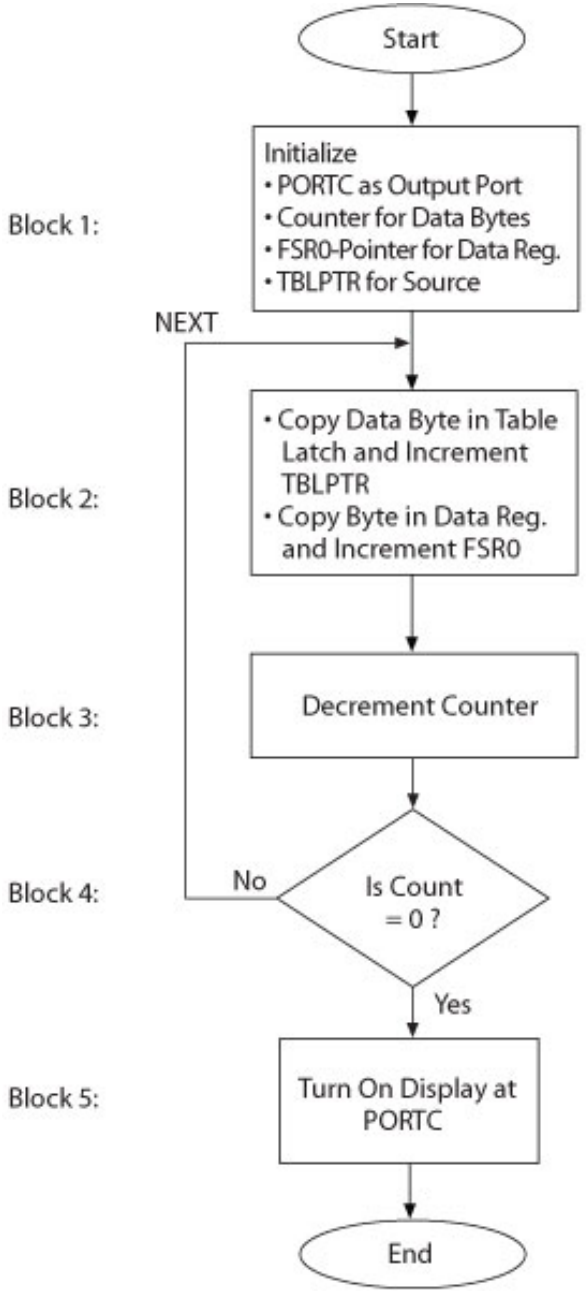
Loop Example2

Label	Opcode	Operand	Comments
COUNT1	EQU	0x01	;Counter is REG01
	ORG	0x20	
START:	MOVLW	0x05	;Initialize Counter to 5
	MOVWF	COUNT1	
LOOP1:	DECFSZ	COUNT1,F	;Decrement Counter, Skip if 0
	BRA	LOOP1	
	SLEEP		;Done
	END		

Illustrative Program

- Problem Statement
  - Write a program to copy five data bytes from program memory (with the starting location 000050_H called SOURCE) to data registers (with beginning address 010_H called BUFFER).
  - When the copying process is complete, turn on all LEDs at PORTC.
  - Data Bytes: $F6_H$, 67_H, $7F_H$, $A9_H$, 72_H

Copy Data Program



Copy Data Program

Label	Opcode	Operand	Comments
BUFFER	EQU	0x10	;Begin Data Registers
COUNTER	EQU	0x01	;Counter is REG01
	ORG	0x00	;Reset Vector
	GOTO	START	

Copy Data Program

Label	Opcode	Operand	Comments
	ORG	0x20	
START:	MOVLW	0x00	;Init PORTC as Output
	MOVWF	TRISC	
	MOVLW	0x05	;Init COUNTER = 5
	MOVWF	COUNTER	
	LFSR	FSR0, BUFFER	;Init FSR0 Pointer
	MOVLW	UPPER SOURCE	;Init Table Pointer
	MOVWF	TBLPTRU	
	MOVLW	HIGH SOURCE	
	MOVWF	TBLPTRH	
	MOVLW	LOW SOURCE	
	MOVWF	TBLPTRL	

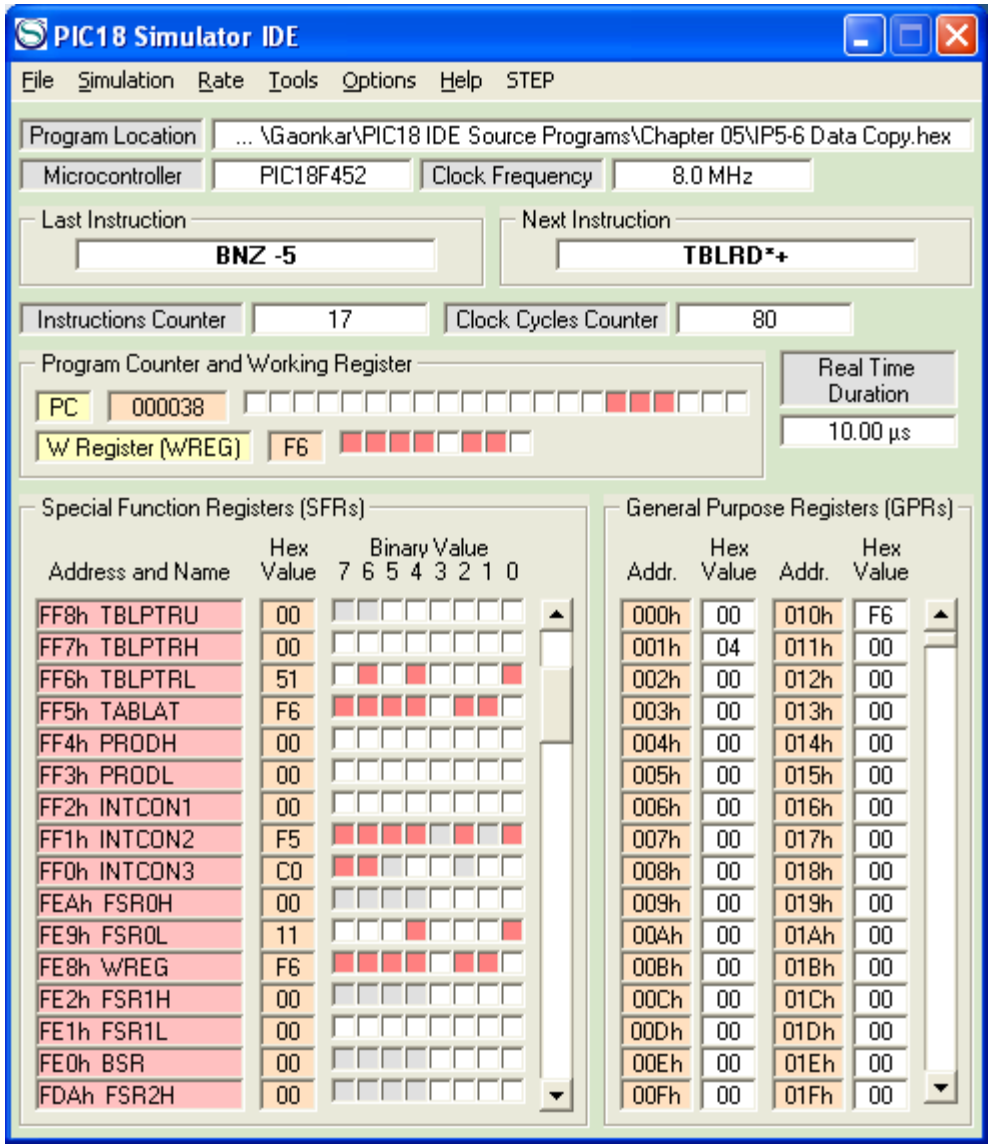
Copy Data Program

Label	Opcode	Operand	Comments
NEXT:	TBLRD*+		;Copy to Table Latch
			;Inc Table Pointer
	MOVF	TABLAT,W	;Copy data to W
	MOVWF	POSTINC0	;Copy to Register
			;Inc FSR0 Pointer
	DECF	COUNTER,F	;Dec COUNTER
	BNZ	NEXT	;COUNTER = 0?
	MOVLW	0xFF	;Set W
	MOVWF	PORTC	;Turn ON LEDs
	SLEEP		

Copy Data Program

Label	Opcode	Operand	Comments
	ORG	0x50	
SOURCE	DB	0xF6,0x67,0x7F,0xA9,0x72	
	END		

Program Execution and Troubleshooting (Debugging)



Logic Operations

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

Application Example

- The W register holds a packed BCD number 68_H.
- Write the instructions to mask the high-order four bits (7-4), preserve the low-order bits (3-0), and save the result in REG1.

Label	Opcode	Operand	Comments
	ANDLW	B'00001111'	;And with Mask
	MOVWF	REG1	;Save Result

Points to Remember

- Each bit (7-0) of W is logically operated with the corresponding bit of the operand
- When the operand is a File (data) register, the result can be saved in either W or F using the “d” parameter
- These instructions affect only the N and Z flags

Bit Set, Clear, and Toggle

Instructions

- BCF F,b,a
- BSF F,b,a
- BTG F,b,a
- These instructions can set, reset, or toggle any (single) bit in a data register.

Examples

- BCF REG1,7
  Clear Bit7 in REG1
- BSF REG2,4
  Set Bit4 in REG2
- BTG REG5,0
  Toggle Bit0 in REG5

Bit Test and Skip Instructions

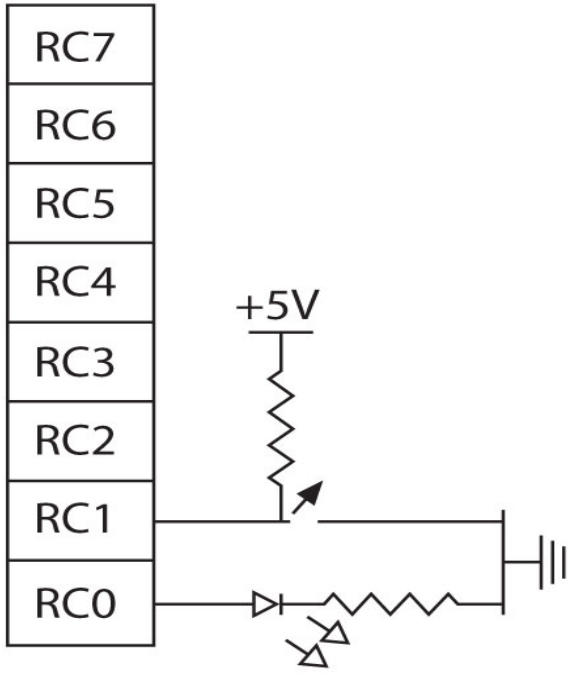
- Instructions test a bit in a data register for set or reset condition; if conditions met, MPU skips next instruction.

Instructions	Examples
◆ BTFSC F,b,a	BTFSC REG1,7 Test BIT7 in REG1 and if the bit is zero, skip the next instruction
◆ BTFSS F,b,a	BTFSS REG1,5 Test Bit5 in REG1 and if the bit is one, skip the next instruction

Application Example

- The MPU checks RC1 (Bit1) at PORTC.
  - If the switch is open (RC1=1), it stays in the loop and continues to check the switch.
  - When the switch is closed (RC1=0), the MPU skips the branch instruction, and turns on the LED.

Label	Opcode	Operand
CHECK:	BTFSC	PORTC,1
	BRA	CHECK
	BSF	PORTC,0



Bit Rotation

- The instruction set includes four instructions that can shift a bit to the adjacent position
  - Left or right
- The instructions are further classified as 8-bit rotation and 9-bit rotation
  - In 9-bit rotation, carry flag becomes the ninth bit

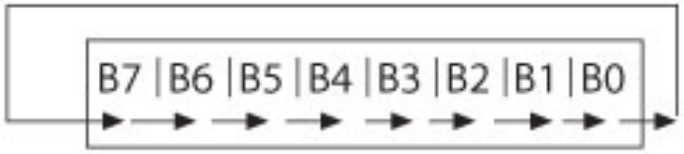
Rotate Instructions

RLCF F, d, a — Rotate Left through Carry
If d = 1, save result in F, and
if d = 0, save in W

RLNCF F, d, a — Rotate Left with No Carry
If d = 1, save result in F, and
if d = 0, save in W

RRCF F, d, a — Rotate Right through Carry
If d = 1, save result in F, and
if d = 0, save in W

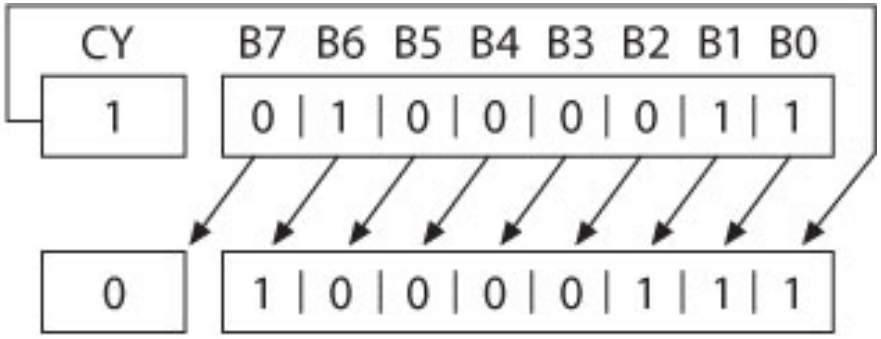
RRNCF F, d, a — Rotate Right with No Carry
If d = 1, save result in F, and
if d = 0, save in W



Rotate Left Example

REG1: Initial Contents

REG1 = 43H

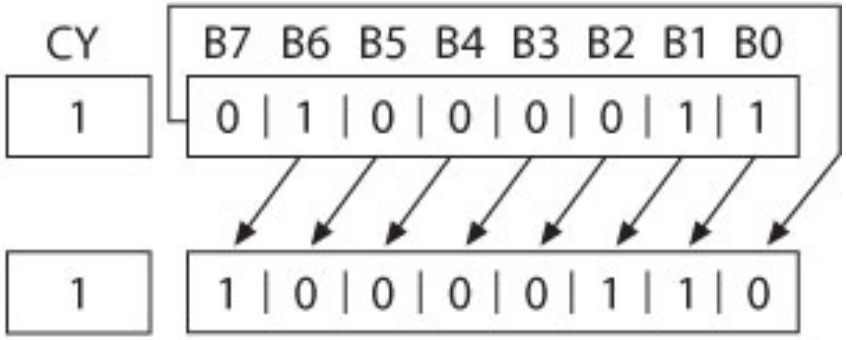


RLCF REG1, 1, 0
Rotate Left through Carry
Save in REG1 because d = 1

REG1 = 87H

REG1: Initial Contents

REG1 = 43H



RLNCF REG1, 0, 0
Rotate Left With No Carry
Save in W because d = 0

W = 86H

Unpacking Example

- Write a program to unpack a packed BCD byte into two buffers, so it can be displayed on two seven-segment displays.

Unpacking Example

Label	Opcode	Operand	Comments
BCD0	EQU	0x10	;Define Data Registers
BCD1	EQU	0x11	
REG1	EQU	0x01	
	ORG	0x00	;Reset Vector
	GOTO	START	

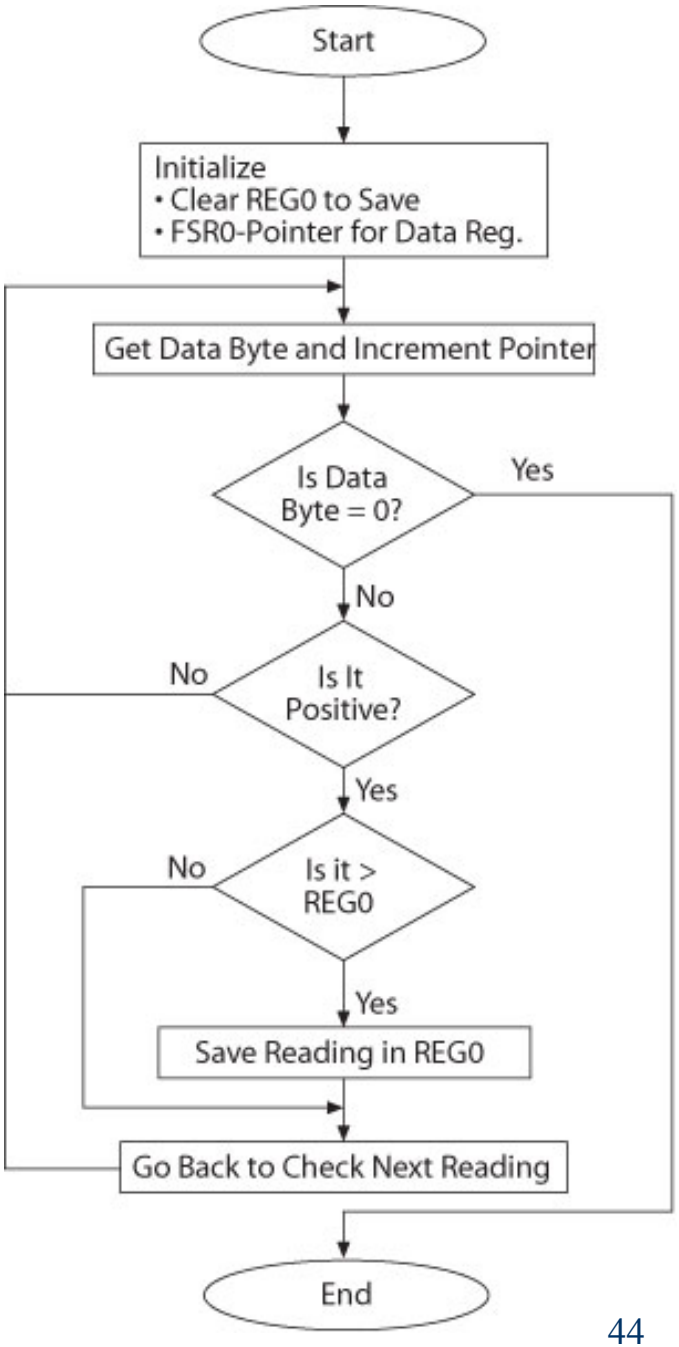
Unpacking Example

Label	Opcode	Operand	Comments
	ORG	0x20	
START:	MOVLW	0x37	;Load packed byte
	MOVWF	REG1	
	ANDLW	0x0F	;Mask high order digit
	MOVWF	BCD0	;Save low order digit
	MOVF	REG1,W	;Reload byte
	ANDLW	0xF0	;Mask low order digit
	RRNCF	WREG,W	;Rotate 4 times
	RRNCF	WREG,W	
	RRNCF	WREG,W	
	RRNCF	WREG,W	
	MOVWF	BCD1	;Save high order digit
	SLEEP		

Illustrative Program

- Find the Highest Temperature in a Data String
  - Data string includes positive and negative 8-bit readings.
  - Terminated in null character 00.
  - To find the highest temperature.
    - Get a reading and check whether it is zero.
    - Check whether the byte is negative.
    - Is the byte larger than the previously saved data?
      - If yes, replace the existing byte.
    - Go back to get the next byte.

Flow Chart


Start
Initialize
• Clear REG0 to Save
• FSR0-Pointer for Data Reg.
Get Data Byte and Increment Pointer
Is Data Byte = 0?
Yes
No
Is It Positive?
No
Yes
Is it > REG0
No
Yes
Save Reading in REG0
Go Back to Check Next Reading
End

Highest Temperature

Label	Opcode	Operand	Comments
REG0	EQU	0x00	;Define Data Registers
BUFFER	EQU	0x10	
	ORG	0x00	;Reset Vector
	GOTO	START	

Highest Temperature

Label	Opcode	Operand	Comments
	ORG	0x20	
START:	CLRF	REG0	;Init REG0
	LFSR	FSR0,BUFFER	;Init Pointer
NEXT:	MOVF	POSTINC0,W	;Get Data & Inc Pointer
	BZ	FINISH	;Data = 0?
	BTFSC	WREG,7	;Data > 0?
	BRA	NEXT	
	CPFSLT	REG0	;REG0 < Data?
	BRA	NEXT	
	MOVWF	REG0	;Save Larger Data
	BRA	NEXT	
FINISH	SLEEP		

Program Execution