EECE416: Microcomputer Fundamentals and Design

source: www.mwftr.com

PIC Instruction Set and Some Tips for Programming

Dr. Charles J. Kim

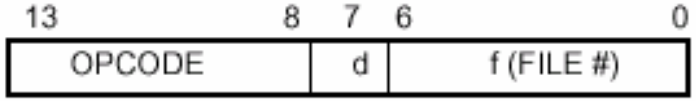
Howard University

F877 Instruction Set

- 14-Bit Word
- **Byte-Oriented Instruction**
  - F: File Register (or RAM)
  - D: Destination
    - D=0: Destination → W
    - D=1: Destination → File Register
- **Bit-Oriented Instruction**
  - B: Bit Field
  - F: Register File where the Bit is located
- **Literal and Control Operation**
  - K: 8-bit constant

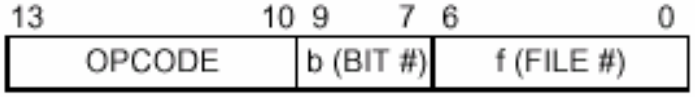
General Form of Instruction

Byte-oriented file register operations



d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

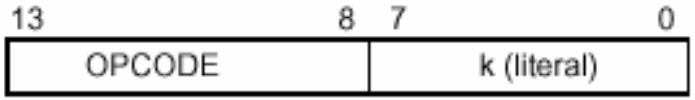
Bit-oriented file register operations



b = 3-bit bit address
f = 7-bit file register address

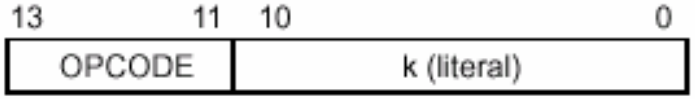
Literal and control operations

General



k = 8-bit immediate value

CALL and GOTO instructions only



k = 11-bit immediate value

Instruction List

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff	Z	1,2
CLRF	f	Clear f	1	00 0001 lfff ffff	Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx	Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff	Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff		1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff	Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff	Z	1,2
MOVWF	f	Move W to f	1	00 0000 lfff ffff		
NOP	-	No Operation	1	00 0000 0xx0 0000		
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k	Add literal and W	1	11 111x kkkk kkkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11 1001 kkkk kkkk	Z	
CALL	k	Call subroutine	2	10 0kkk kkkk kkkk		
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100	$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to address	2	10 1kkk kkkk kkkk		
IORLW	k	Inclusive OR literal with W	1	11 1000 kkkk kkkk	Z	
MOVLW	k	Move literal to W	1	11 00xx kkkk kkkk		
RETFIE	-	Return from interrupt	2	00 0000 0000 1001		
RETLW	k	Return with literal in W	2	11 01xx kkkk kkkk		
RETURN	-	Return from Subroutine	2	00 0000 0000 1000		
SLEEP	-	Go into standby mode	1	00 0000 0110 0011	$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from literal	1	11 110x kkkk kkkk	C,DC,Z	
XORLW	k	Exclusive OR literal with W	1	11 1010 kkkk kkkk	Z	

Destination of the result & Machine Code

- D=0: Destination → W
- D=1: Destination → File Register (Default)

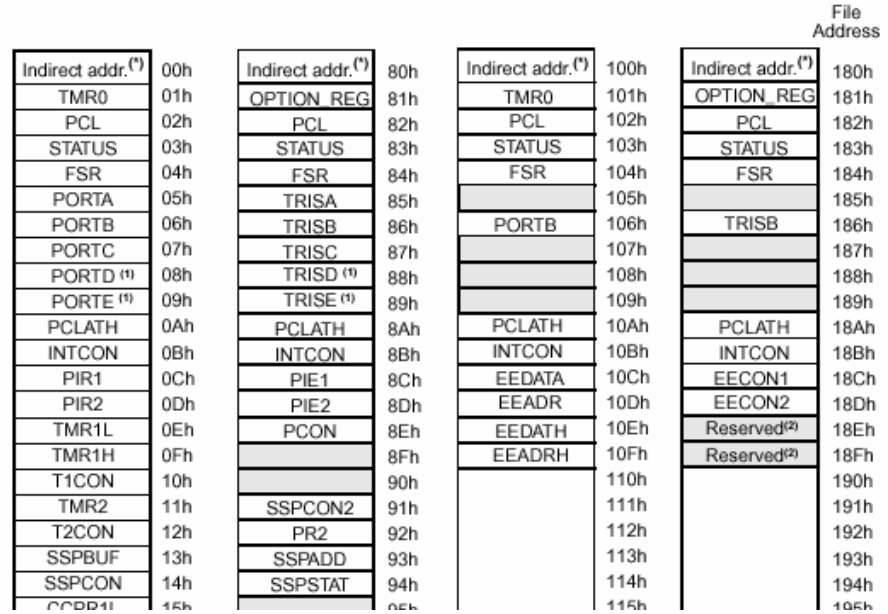
  - **addwf PORTD ;**
    - Add content of PORTD to content of the W and store the result back into PORTD

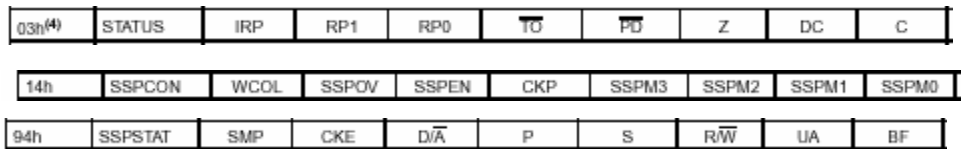
  - **addwf PORTD, 0 ;**
    - Add content of PORTD to content of the W and store the result into W

Register Addressing Modes

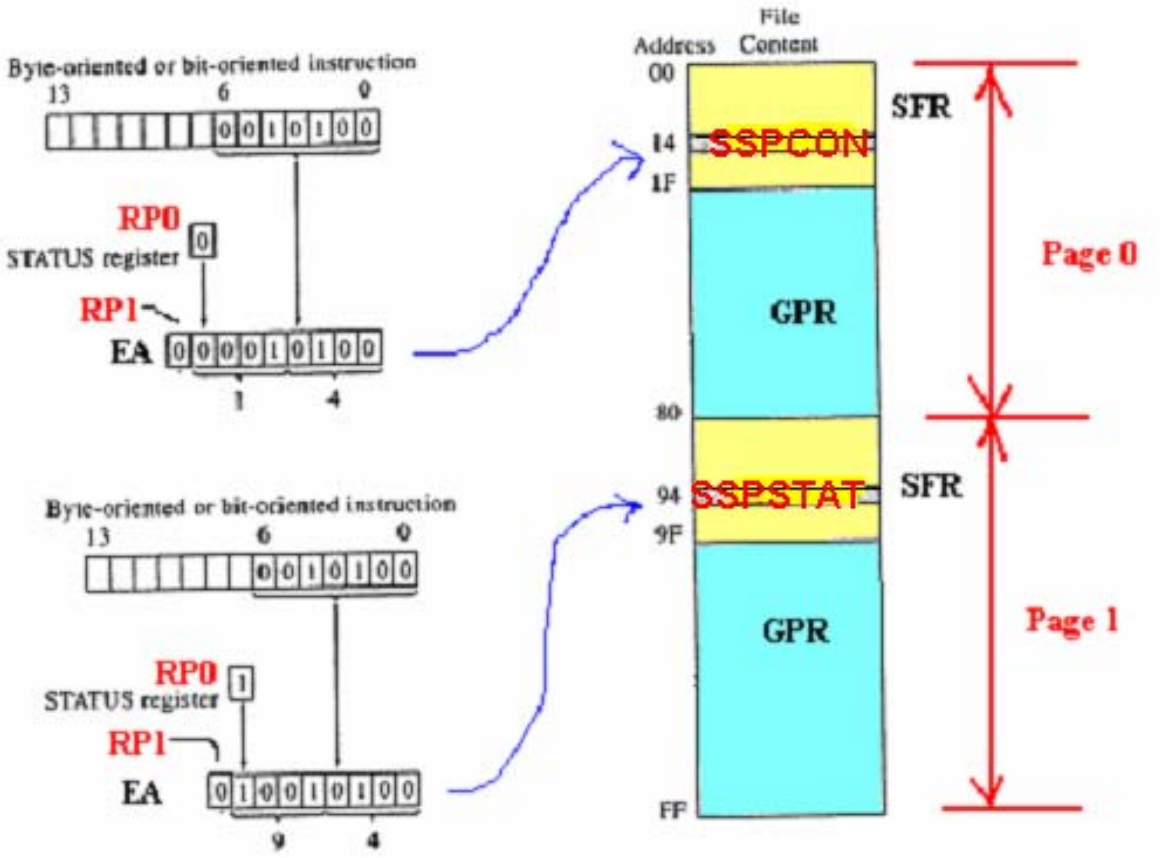
- Immediate Addressing
  - (ex) MOVLW 0x0F
- Direct Addressing
  - Uses 7 bits of 14 bit instruction to identify a register file address
  - 8th and 9th bit comes from RP0 and RP1 bits of STATUS register.
  - Initial condition for RP0 and RP1: RP1=RP0=0
  - (ex)

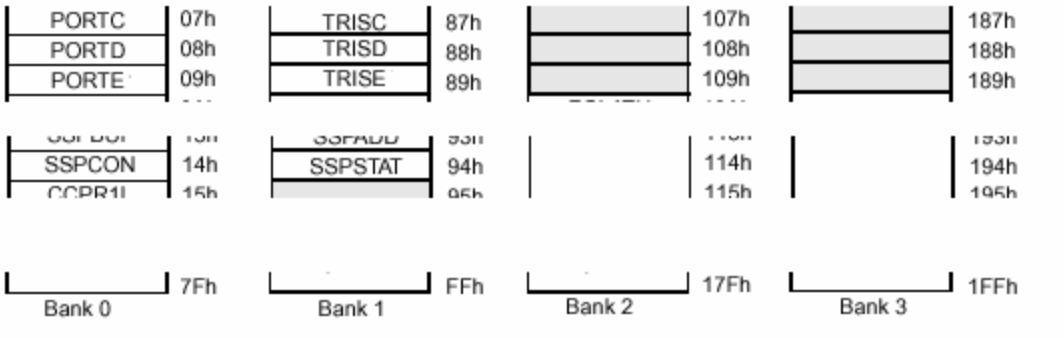
```
SSPCON  EQU  0x14
STATUS  EQU  0x03
SSPSTAT EQU  0x94
BCF      STATUS, 0x05
BCF      SSPCON, 0x01
BSF      STATUS, 0x05
BCF      SSPSTAT, 0x02
```





Direct Addressing - Recap

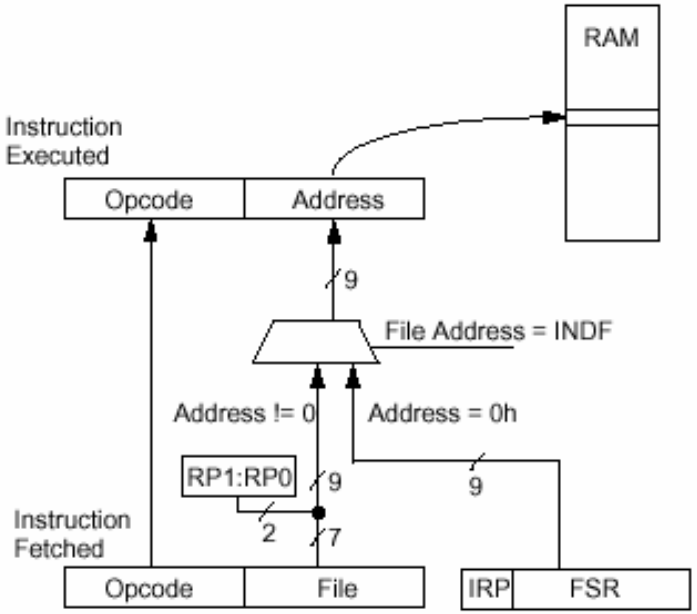




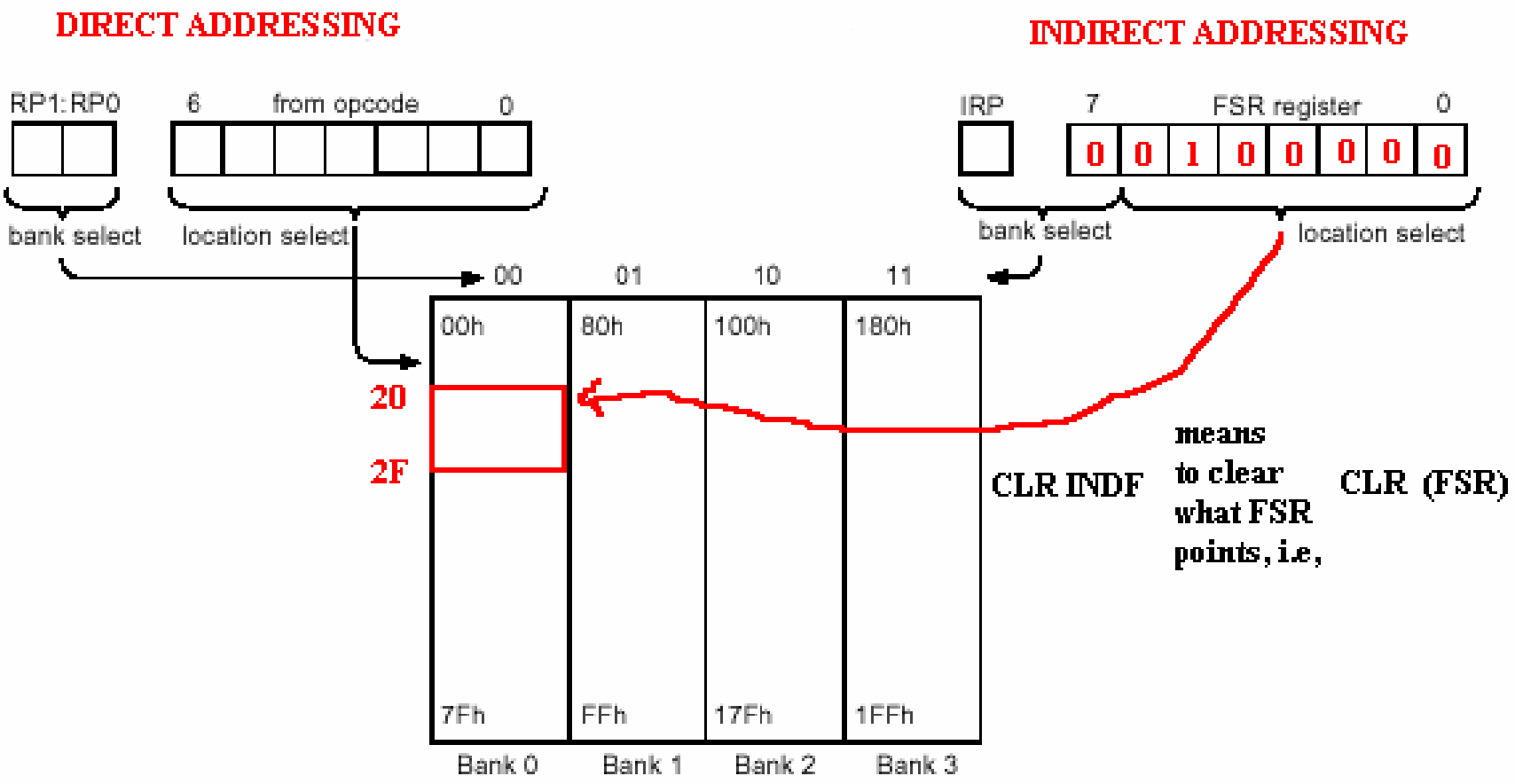
Indirect Addressing

- INDF register
  - Any instruction using the INDF actually accesses the register pointed to by the File Select Register (FSR).
- A 9-bit EA is obtained by concatenating the 8-bit FSR register and the IRP bit(STATUS<7>)
- Example: Erase the RAM section of 0x20-0x2F

```
        Movlw 0x20; pointer
        Movwf FSR
Next    clrf  INDF
        incf  FSR
        btfss FSR, 4
        goto  next
              ......
```



Direct vs. Indirect Addressing



Instruction Sets –description convention

Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register (0 to 7)
k	Literal field, constant data or label (may be either an 8-bit or an 11-bit value)
x	Don't care (0 or 1) The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
d	Destination select; d = 0: store result in W, d = 1: store result in file register f.
dest	Destination either the W register or the specified register file location
label	Label name
TOS	Top of Stack
PC	Program Counter
PCLATH	Program Counter High Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer
$\overline{TO}$	Time-out bit
$\overline{PD}$	Power-down bit
[ ]	Optional
( )	Contents
→	Assigned to
< >	Register bit field
∈	In the set of
italics	User defined term (font is courier)

ADDLW — Add Literal and W

Syntax:	[*label*] ADDLW k
Operands:	$0 \le k \le 255$
Operation:	$(W) + k \rightarrow W$
Status Affected:	C, DC, Z

Encoding:

11	111x	kkkk	kkkk

Description: The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.

Words: 1

Cycles: 1

Example1 `ADDLW 0x15`

Before Instruction
W = 0x10
After Instruction
W = 0x25

Example 2 `ADDLW MYREG`

Before Instruction
W = 0x10
Address of `MYREG` † = 0x37
† `MYREG` is a symbol for a data memory location
After Instruction
W = 0x47

ADDWF — Add W and f

Syntax:	[*label*] ADDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W) + (f) → destination
Status Affected:	C, DC, Z
Encoding:	00 | 0111 | dfff | ffff
Description:	Add the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example 1

```
ADDWF   FSR, 0
```

Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0xD9
FSR = 0xC2

Example 2

```
ADDWF   INDF
```

Before Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x20
After Instruction
W = 0x17
FSR = 0xC2
Contents of Address (FSR) = 0x37

ANDLW — And Literal with W

Syntax:	[*label*] ANDLW k
Operands:	$0 \le k \le 255$
Operation:	(W).AND. (k) $\rightarrow$ W
Status Affected:	Z
Encoding:	11 | 1001 | kkkk | kkkk
Description:	The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1

Example 1

```
ANDLW   0x5F
```

Before Instruction
W = 0xA3
After Instruction
W = 0x03

```
; 0101 1111   (0x5F)
; 1010 0011   (0xA3)
;----------   ------
; 0000 0011   (0x03)
```

ANDWF — AND W with f

Syntax:	[*label*] ANDWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W).AND. (f) → destination
Status Affected:	Z
Encoding:	00 | 0101 | dfff | ffff
Description:	AND the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example 1

```
ANDWF   FSR
Before Instruction                  ; 0001 0111   (0x17)
        W   =  0x17                 ; 1100 0010   (0xC2)
        FSR =  0xC2                 ;----------   ------
After Instruction                   ; 0000 0010   (0x02)
        W   =  0x17
        FSR =  0x02
```

Example 2

```
ANDWF   FSR, 0
Before Instruction                  ; 0001 0111   (0x17)
        W   =  0x17                 ; 1100 0010   (0xC2)
        FSR =  0xC2                 ;----------   ------
After Instruction                   ; 0000 0010   (0x02)
        W   =  0x02
        FSR =  0xC2
```

BCF — Bit Clear f

Syntax:	[*label*] BCF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	0 → f<b>
Status Affected:	None
Encoding:	01 | 00bb | bfff | ffff
Description:	Bit 'b' in register 'f' is cleared.
Words:	1
Cycles:	1

Example 1

```
BCF     FLAG_REG, 7
```

Before Instruction

```
FLAG_REG = 0xC7        ; 1100 0111
```

After Instruction

```
FLAG_REG = 0x47        ; 0100 0111
```

BSF — Bit Set f

Syntax:	[*label*] BSF f,b
Operands:	0 ≤ f ≤ 127 0 ≤ b ≤ 7
Operation:	1 → f<b>
Status Affected:	None
Encoding:	01 | 01bb | bfff | ffff
Description:	Bit 'b' in register 'f' is set.
Words:	1
Cycles:	1

```
BSF     FLAG_REG, 7
```

Before Instruction

```
FLAG_REG =0x0A         ; 0000 1010
```

After Instruction

```
FLAG_REG =0x8A         ; 1000 1010
```

BTFSC

Bit Test, Skip if Clear

Syntax: [*label*] BTFSC f,b

Operands: $0 \le f \le 127$
$0 \le b \le 7$

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

01	10bb	bfff	ffff

Description: If bit 'b' in register 'f' is '0' then the next instruction is skipped.
If bit 'b' is '0' then the next instruction (fetched during the current instruction execution) is discarded, and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Example 1

```
HERE    BTFSC   FLAG, 4
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

Case 1: Before Instruction
PC = address HERE
FLAG= xxx0 xxxx
After Instruction
Since FLAG<4>= 0,
PC = address TRUE

Case 2: Before Instruction
PC = address HERE
FLAG= xxx1 xxxx
After Instruction
Since FLAG<4>=1,
PC = address FALSE

BTFSS

Bit Test f, Skip if Set

Syntax: [*label*] BTFSS f,b

Operands: $0 \le f \le 127$
$0 \le b < 7$

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

01	11bb	bfff	ffff

Description: If bit 'b' in register 'f' is '1' then the next instruction is skipped.
If bit 'b' is '1', then the next instruction (fetched during the current instruction execution) is discarded and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Example 1

```
HERE    BTFSS   FLAG, 4
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```

Case 1: Before Instruction
PC = address HERE
FLAG= xxx0 xxxx
After Instruction
Since FLAG<4>= 0,
PC = address FALSE

Case 2: Before Instruction
PC = address HERE
FLAG= xxx1 xxxx
After Instruction
Since FLAG<4>=1,
PC = address TRUE

CALL — Call Subroutine

Syntax:	[*label*] CALL k
Operands:	$0 \le k \le 2047$
Operation:	(PC)+ 1→ TOS, k → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Status Affected:	None
Encoding:	10 | 0kkk | kkkk | kkkk
Description:	Call Subroutine. First, the 13-bit return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH<4:3>. CALL is a two cycle instruction.
Words:	1
Cycles:	2

Example 1

```
HERE    CALL   THERE
```

Before Instruction
PC = Address HERE
After Instruction
TOS = Address HERE+1
PC = Address THERE

CLRF — Clear f

Syntax: [*label*] CLRF f

Operands: $0 \le f \le 127$

Operation: $00h \rightarrow f$
$1 \rightarrow Z$

Status Affected: Z

Encoding:

00	0001	1fff	ffff

Description: The contents of register 'f' are cleared and the Z bit is set.

Words: 1

Cycles: 1

Example 1

```
CLRF    FLAG_REG
```

Before Instruction
FLAG_REG=0x5A
After Instruction
FLAG_REG=0x00
Z = 1

CLRW — Clear W

Syntax: [*label*] CLRW

Operands: None

Operation: $00h \rightarrow W$
$1 \rightarrow Z$

Status Affected: Z

Encoding:

00	0001	0xxx	xxxx

Description: W register is cleared. Zero bit (Z) is set.

Words: 1

Cycles: 1

Example 1

```
CLRW
```

Before Instruction
W = 0x5A
After Instruction
W = 0x00
Z = 1

COMF

Complement f

Syntax: [*label*] COMF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: $(\bar{f}) \rightarrow$ destination

Status Affected: Z

Encoding:

00	1001	dfff	ffff

Description: The contents of register 'f' are 1's complemented. If 'd' is 0 the result is stored in W. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example 1

```
COMF    REG1, 0
```

Before Instruction
REG1= 0x13
After Instruction
REG1= 0x13
W = 0xEC

DECF

Decrement f

Syntax: [*label*] DECF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) - 1 $\rightarrow$ destination

Status Affected: Z

Encoding:

00	0011	dfff	ffff

Description: Decrement register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example 1

```
DECF    CNT
```

Before Instruction
CNT = 0x01
Z = 0
After Instruction
CNT = 0x00
Z = 1

DECFSZ

Decrement f, Skip if 0

Syntax:	[*label*] DECFSZ f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) - 1 → destination; skip if result = 0
Status Affected:	None
Encoding:	00 | 1011 | dfff | ffff
Description:	The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'. If the result is 0, then the next instruction (fetched during the current instruction execution) is discarded and a NOP is executed instead, making this a 2 cycle instruction.
Words:	1
Cycles:	1(2)

Example

```
HERE      DECFSZ   CNT, 1
          GOTO     LOOP
CONTINUE  •
          •
          •
```

Case 1:
Before Instruction
PC = address HERE
CNT = 0x01
After Instruction
CNT = 0x00
PC = address CONTINUE

Case 2:
Before Instruction
PC = address HERE
CNT = 0x02
After Instruction
CNT = 0x01
PC = address HERE + 1

GOTO

Unconditional Branch

Syntax:	[*label*] GOTO k
Operands:	$0 \le k \le 2047$
Operation:	k → PC<10:0> PCLATH<4:3> → PC<12:11>
Status Affected:	None
Encoding:	10 | 1kkk | kkkk | kkkk
Description:	GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two cycle instruction.
Words:	1
Cycles:	2

INCF

Increment f

Syntax:	[*label*] INCF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) + 1 → destination
Status Affected:	Z
Encoding:	00 | 1010 | dfff | ffff
Description:	The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
Words:	1
Cycles:	1

Example

```
INCF    CNT,  1
```

Before Instruction

CNT = 0xFF

Z = 0

After Instruction

CNT = 0x00

Z = 1

INCFSZ

Increment f, Skip if 0

Syntax: [*label*] INCFSZ f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f) + 1 → destination, skip if result = 0

Status Affected: None

Encoding:

00	1111	dfff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is 0, then the next instruction (fetched during the current instruction execution) is discarded and a NOP is executed instead, making this a 2 cycle instruction.

Words: 1

Cycles: 1(2)

Example

```
HERE      INCFSZ   CNT
          GOTO     LOOP
CONTINUE  •
          •
          •
```

Case 1:
Before Instruction
PC = address HERE
CNT = 0xFF
After Instruction
CNT = 0x00
PC = address CONTINUE

Case 2:
Before Instruction
PC = address HERE
CNT = 0x00
After Instruction
CNT = 0x01
PC = address HERE + 1

IORLW — Inclusive OR Literal with W

Syntax:	[*label*] IORLW k
Operands:	$0 \le k \le 255$
Operation:	(W).OR. k → W
Status Affected:	Z
Encoding:	11 | 1000 | kkkk | kkkk
Description:	The contents of the W register is OR'ed with the eight bit literal 'k'. The result is placed in the W register.
Words:	1
Cycles:	1

Example

```
IORLW   0x35
```

Before Instruction
W = 0x9A
After Instruction
W = 0xBF
Z = 0

IORWF — Inclusive OR W with f

Syntax:	[*label*] IORWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W).OR. (f) → destination
Status Affected:	$\overline{Z}$
Encoding:	00 | 0100 | dfff | ffff
Description:	Inclusive OR the W register with register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
Words:	1
Cycles:	1

Example

```
IORWF   RESULT, 0
```

Before Instruction
RESULT=0x13
W = 0x91
After Instruction
RESULT=0x13
W = 0x93
Z = 0

MOVLW

Move Literal to W

Syntax:	[*label*] MOVLW k
Operands:	$0 \le k \le 255$
Operation:	$k \rightarrow W$
Status Affected:	None
Encoding:	11 | 00xx | kkkk | kkkk
Description:	The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.
Words:	1
Cycles:	1

Example

```
MOVLW   0x5A
```

After Instruction

W = 0x5A

MOVF

Move f

Syntax:	[*label*] MOVF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(f) → destination
Status Affected:	Z
Encoding:	00 | 1000 | dfff | ffff
Description:	The contents of register 'f' is moved to a destination dependent upon the status of 'd'. If 'd' = 0, destination is W register. If 'd' = 1, the destination is file register 'f' itself. 'd' = 1 is useful to test a file register since status flag Z is affected.
Words:	1
Cycles:	1

Example

```
MOVF    FSR, 0
```

Before Instruction

W = 0x00

FSR = 0xC2

After Instruction

W = 0xC2

Z = 0

MOVWF — Move W to f

Syntax:	[*label*] MOVWF f
Operands:	$0 \le f \le 127$
Operation:	$(W) \rightarrow f$
Status Affected:	None
Encoding:	00 | 0000 | 1fff | ffff
Description:	Move data from W register to register 'f'.
Words:	1
Cycles:	1

& NOP

Example

```
MOVWF   OPTION_REG
```

Before Instruction
OPTION_REG=0xFF
W = 0x4F
After Instruction
OPTION_REG=0x4F
W = 0x4F

NOP — No Operation

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	00 | 0000 | 0xx0 | 0000
Description:	No operation.
Words:	1
Cycles:	1

Example

```
HERE    NOP
```

Before Instruction
PC = address HERE
After Instruction
PC = address HERE + 1

RETFIE — Return from Interrupt

Syntax:	[*label*] RETFIE
Operands:	None
Operation:	TOS → PC, 1 → GIE
Status Affected:	None
Encoding:	00 | 0000 | 0000 | 1001
Description:	Return from Interrupt. The 13-bit address at the Top of Stack (TOS) is loaded in the PC. The Global Interrupt Enable bit, GIE (INTCON<7>), is automatically set, enabling Interrupts. This is a two cycle instruction.
Words:	1
Cycles:	2

Example

```
HERE    CALL TABLE    ; W contains table
                      ; offset value
        •             ; W now has table value
        •
        •
TABLE   ADDWF PC      ;W = offset
        RETLW k1      ;Begin table
        RETLW k2      ;
        •
        •
        •
        RETLW kn      ; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of k8
PC = TOS = Address Here + 1

RETLW — Return with Literal in W

Syntax:	[*label*] RETLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → W; TOS → PC
Status Affected:	None
Encoding:	11 | 01xx | kkkk | kkkk
Description:	The W register is loaded with the eight bit literal 'k'. The program counter is loaded 13-bit address at the Top of Stack (the return address). This is a two cycle instruction.
Words:	1
Cycles:	2

RETURN

Return from Subroutine

Syntax:	[*label*] RETURN
Operands:	None
Operation:	TOS → PC
Status Affected:	None
Encoding:	00 | 0000 | 0000 | 1000
Description:	Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.
Words:	1
Cycles:	2

RLF

Rotate Left f through Carry

Syntax: [*label*] RLF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

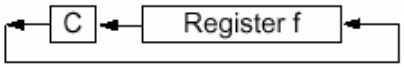
Operation: See description below

Status Affected: C

Encoding:

00	1101	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is stored back in register 'f'.



Words: 1

Cycles: 1

Example

```
RLF     REG1,0
```

Before Instruction

```
REG1=  1110 0110
C   =  0
```

After Instruction

```
REG1=1110 0110
W   =1100 1100
C   =1
```

RRF

Rotate Right f through Carry

Syntax: [*label*] RRF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

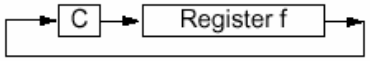
Operation: See description below

Status Affected: C

Encoding:

00	1100	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.



Words: 1

Cycles: 1

Example

```
RRF             REG1,0
```

Before Instruction

```
REG1= 1110 0110
W   = xxxx xxxx
C   = 0
```

After Instruction

```
REG1= 1110 0110
W   = 0111 0011
C   = 0
```

SUBLW

Subtract W from Literal

Syntax: [*label*] SUBLW k

Operands: $0 \le k \le 255$

Operation: $k - (W) \rightarrow W$

Status Affected: C, DC, Z

Encoding:

11	110x	kkkk	kkkk

Description: The W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example

```
SUBLW   0x02
```

Before Instruction

W = 0x01
C = x
Z = x

After Instruction ; result is positive

W = 0x01
C = 1
Z = 0

SUBWF

Subtract W from f

Syntax: [*label*] SUBWF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: $(f) - (W) \rightarrow$ destination

Status Affected: C, DC, Z

Encoding:

00	0010	dfff	ffff

Description: Subtract (2's complement method) W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Example

```
SUBWF   REG1,1
```

Before Instruction

REG1= 3
W = 2
C = x
Z = x

After Instruction

REG1= 1
W = 2
C = 1
Z = 0

SWAPF

Swap Nibbles in f

Syntax: [*label*] SWAPF f,d

Operands: $0 \le f \le 127$
$d \in [0,1]$

Operation: (f<3:0>) → destination<7:4>,
(f<7:4>) → destination<3:0>

Status Affected: None

Encoding:

00	1110	dfff	ffff

Description: The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.

Words: 1

Cycles: 1

Example

```
SWAPF   REG, 0
```

Before Instruction

REG1= 0xA5

After Instruction

REG1= 0xA5
W = 0x5A

XORLW

Exclusive OR Literal with W

Syntax: [*label*] XORLW k

Operands: $0 \le k \le 255$

Operation: (W).XOR. k → W

Status Affected: Z

Encoding:

11	1010	kkkk	kkkk

Description: The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Example

```
XORLW   0xAF            ; 1010 1111   (0xAF)
Before Instruction      ; 1011 0101   (0xB5)
        W = 0xB5        ; ---------   ------
After Instruction       ; 0001 1010   (0x1A)
        W = 0x1A
        Z = 0
```

XORWF — Exclusive OR W with f

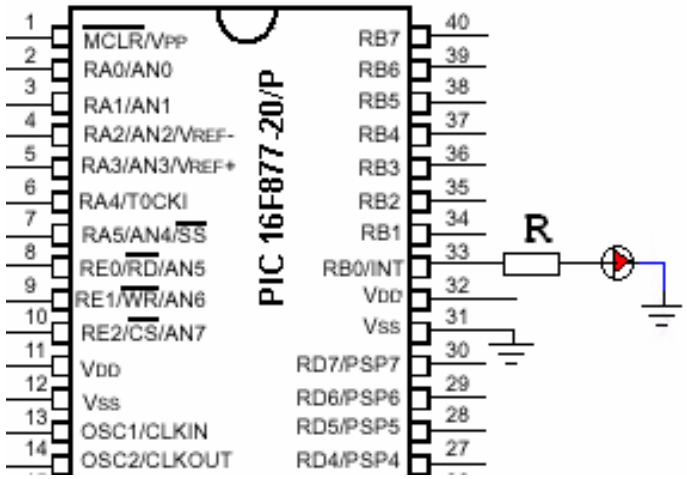
Syntax:	[*label*] XORWF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$
Operation:	(W).XOR. (f) → destination
Status Affected:	Z
Encoding:	00 | 0110 | dfff | ffff
Description:	Exclusive OR the contents of the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words:	1
Cycles:	1

Example

```
XORWF    REG                    ; 1010 1111    (0xAF)
Before Instruction              ; 1011 0101    (0xB5)
        REG=  0xAF              ; ---------    ------
        W   =  0xB5             ; 0001 1010    (0x1A)
After Instruction
        REG=  0x1A
        W   =  0xB5
```

F877 Instruction – Programming Tips

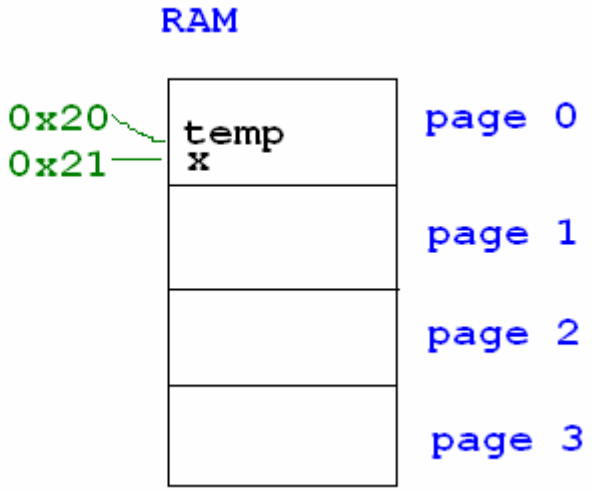
- Tips on Instruction
- Skills and Tricks
- Frequently Met Situations
- (a) Turn on/off an LED
  - An LED is connected to a pin at one, say, PORTB<0>, and to a ground at the other.
  - Use a bit-oriented file register operation:
    - `bsf PORTB, 0x00` ;to turn on and
    - BSF PORTB, 0 ;same effect
    - `bcf PORTB, 0x00` ;to turn off.
    - BCF PORTB, 0 ;same OFF



Tips – Variable Declaration

- **(b)Variable Declaration**
  - In C:
    - `byte temp`
    - `int x`
  - In PIC:
    - Variables must occupy physical RAM space
    - CBLOCK ...ENDC pair
    - Example:

```
CBLOCK  0x20
        temp
        x
ENDC
```



Tips – I/O designation

- (c) I/O designation to the I/O Port
  - I/O Ports: PORTA, PORTB, PORTC, PORTD, PORTE
  - Bi-directional
  - Input or Output
  - I/O selection file register
    - TRISA for PORTA; TRISB for PORTB; TRISC for PORTC
    - TRISD for PORTD; TRISE for PORTE
    - Each pin can be selected as
      - 1: Input
      - 0: Output

```
BSF     STATUS, RP1     ;move to page 1
CLRF    TRISB           ;PROTB all output
BSF     TRISB, 1        ;PORTB<1> is input
                        ;all others, outputs

MOVLW   0xF0
MOVWF   TRISB           ;PORTB<0-3> outputs
                        ;PORTB<4-7> inputs
```

Tips – Reading Sensor Input

- (e) Reading sensor (digital) input and decision-making based on the input
- PIR motion detector & buzzer example
- PIR output
  - 1: No motion detected
  - 0: Motion detected
- Buzzer
  - 1: Buzzer On
  - 0: Buzzer Off
- Use BTFSS or BTFSC

```
STATUS   EQU      0x03
PORTB    EQU      0x06
TRISB    EQU      0x86
PORTD    EQU      0x08
TRISD    EQU      0x88
P1       EQU      0x06
P0       EQU      0x05
PIR      EQU      0x01     ;PIR is connected to PORTB<1>
BUZ      EQU      0x02     ;Buzzer is connected to PORTD<2>
         BCF      STATUS, P1
         BSF      STATUS, P0       ;page 1
         BSF      TRISB, PIR
         BCf      TRISD, BUZ
MONITOR  BCF      STATUS, P0       ;page 0
         BCF      PORTD, BUZ       ;Buz Off
AGAIN    BTFSC    PORTB,PIR
         GOTO     AGAIN
         BSF      PORTB, BUZ
         GOTO     MONITOR
         END
```