MECE336 Microprocessors I
Logical Instructions

Dr. Kurtuluş Erinç Akdoğan

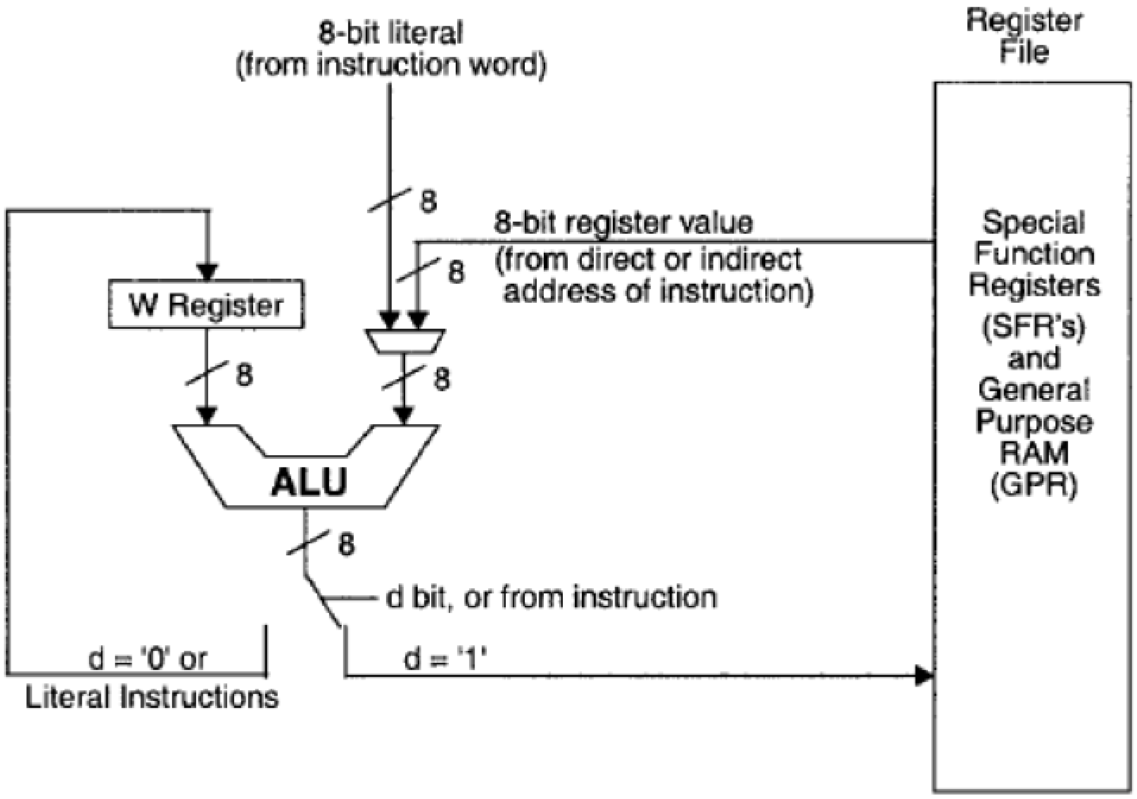
kurtuluserinc@cankaya.edu.tr

Course Webpage: http://MECE336.cankaya.edu.tr

ÇANKAYA ÜNİVERSİTESİ
MEKATRONİK MÜHENDİSLİĞİ BÖLÜMÜ

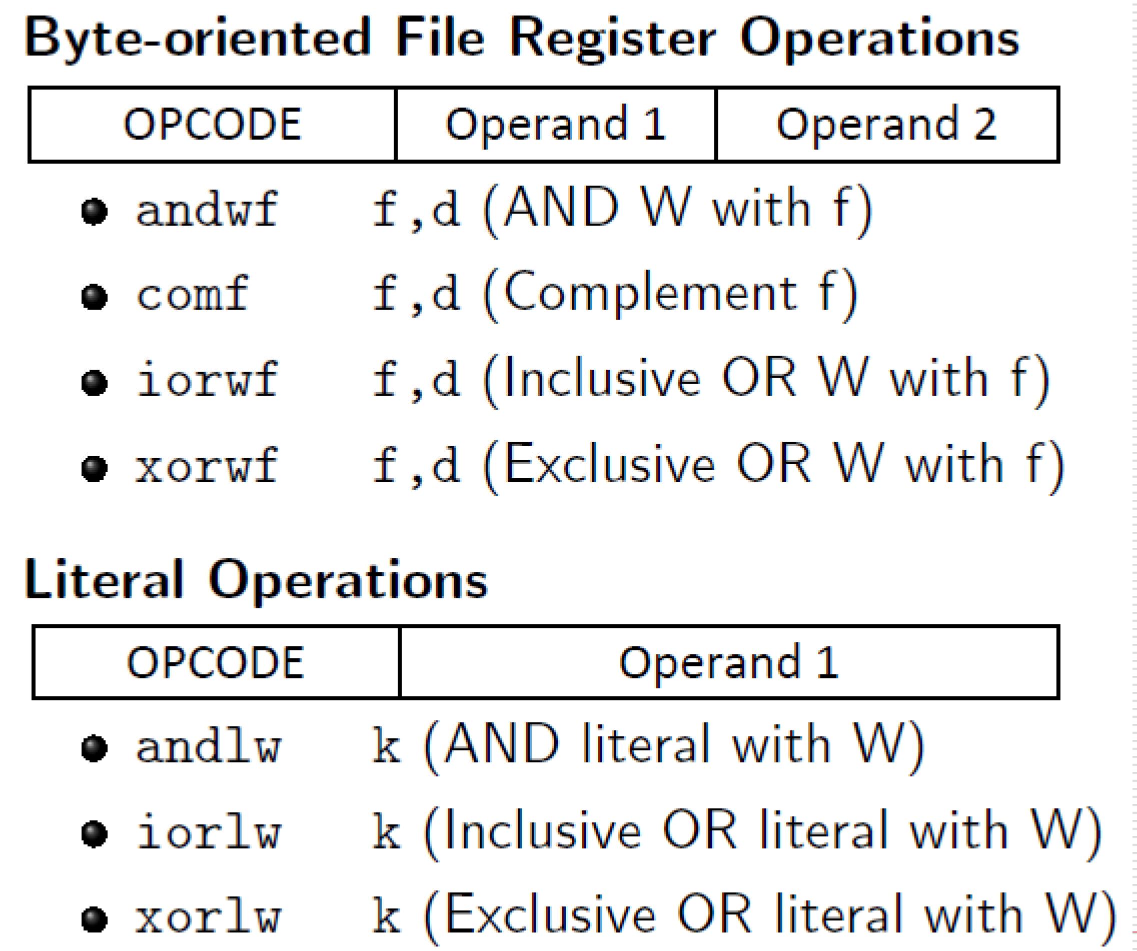
Arithmetic Logic Unit: Basics

- ALU Operates on Data from two Sources
  - Working Register W
  - Literal value or value from data memory
- Literal
  - One byte of data the programmer writes in the program
- Data Memory
  - Memory location is called "register files"by Microchip
- There will be instructions using data memory or literals
- Depending on instruction, result of an Operation is stored
  - Working register
  - Data memory



8-bit literal (from instruction word)
Register File
8
8-bit register value (from direct or indirect address of instruction)
8
W Register
8
8
ALU
8
d bit, or from instruction
d = '0' or Literal Instructions
d = '1'
Special Function Registers (SFR's) and General Purpose RAM (GPR)

Arithmetic Logic Unit: Logical Instructions



Byte-oriented File Register Operations

OPCODE	Operand 1	Operand 2

- andwf f,d (AND W with f)
- comf f,d (Complement f)
- iorwf f,d (Inclusive OR W with f)
- xorwf f,d (Exclusive OR W with f)

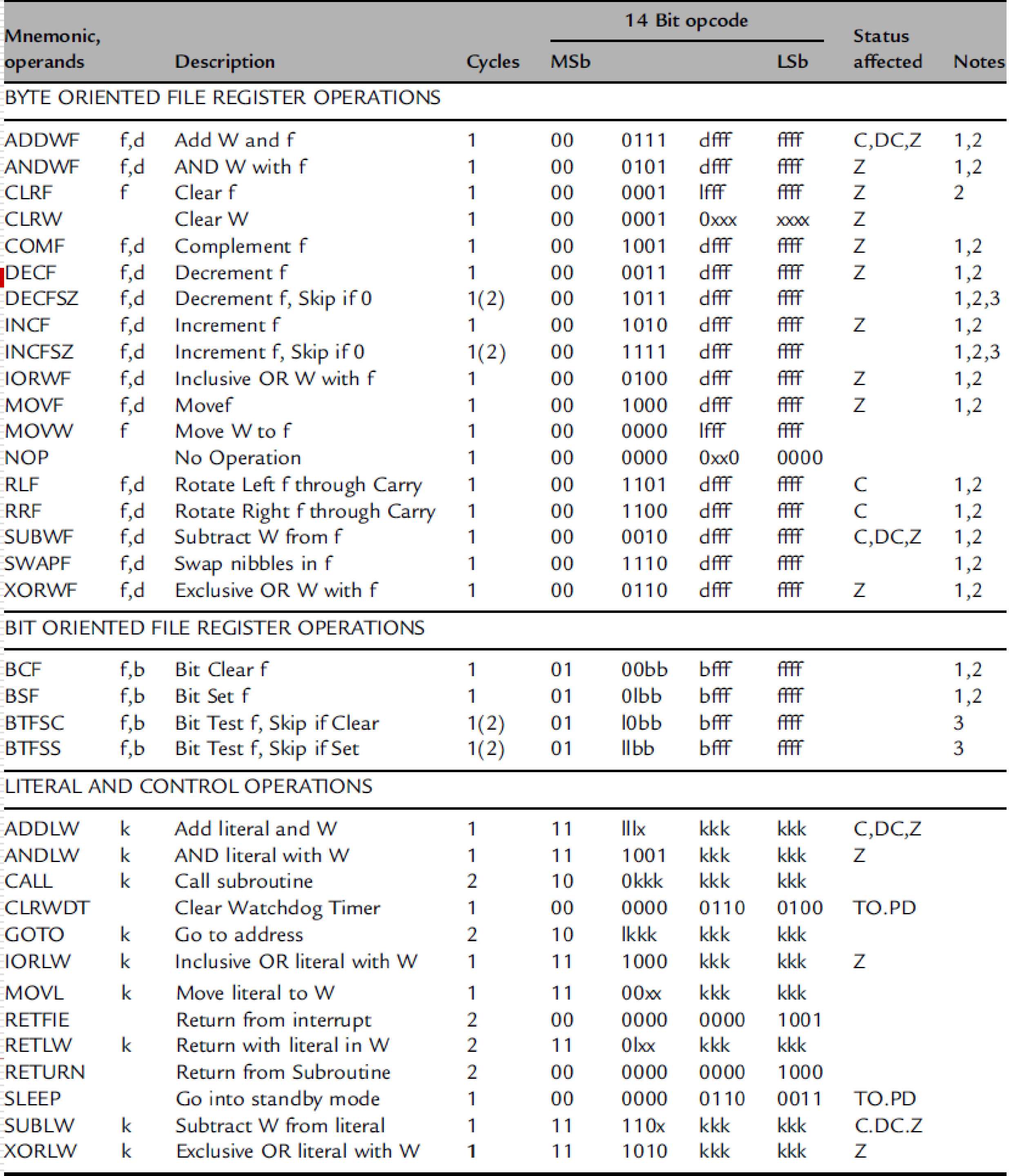
Literal Operations

OPCODE	Operand 1

- andlw k (AND literal with W)
- iorlw k (Inclusive OR literal with W)
- xorlw k (Exclusive OR literal with W)

Arithmetic Logic Unit: COMF

- **Complement f**
  - comf f,d: Bitwise complement of the contents of memory location f
    - Write the result to the W register if the d bit is set to 0
    - Write the result to the memory location f if the d bit is set to 1
- Example
    - Write B'00101010' to the memory location 0x0D. Compute the complement of the memory location 0x0D. Write the result to 0x0D.



Mnemonic, operands		Description	Cycles	14 Bit opcode				Status affected	Notes
				MSb			LSb		
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	Movef	1	00	1000	dfff	ffff	Z	1,2
MOVW	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	0lbb	bfff	ffff		1,2
BTFSC	f,b	Bit Test f, Skip if Clear	1(2)	01	l0bb	bfff	ffff		3
BTFSS	f,b	Bit Test f, Skip if Set	1(2)	01	llbb	bfff	ffff		3
LITERAL AND CONTROL OPERATIONS									
ADDLW	k	Add literal and W	1	11	lllx	kkk	kkk	C,DC,Z	
ANDLW	k	AND literal with W	1	11	1001	kkk	kkk	Z	
CALL	k	Call subroutine	2	10	0kkk	kkk	kkk		
CLRWDT		Clear Watchdog Timer	1	00	0000	0110	0100	TO.PD	
GOTO	k	Go to address	2	10	lkkk	kkk	kkk		
IORLW	k	Inclusive OR literal with W	1	11	1000	kkk	kkk	Z	
MOVL	k	Move literal to W	1	11	00xx	kkk	kkk		
RETFIE		Return from interrupt	2	00	0000	0000	1001		
RETLW	k	Return with literal in W	2	11	0lxx	kkk	kkk		
RETURN		Return from Subroutine	2	00	0000	0000	1000		
SLEEP		Go into standby mode	1	00	0000	0110	0011	TO.PD	
SUBLW	k	Subtract W from literal	1	11	110x	kkk	kkk	C.DC.Z	
XORLW	k	Exclusive OR literal with W	1	11	1010	kkk	kkk	Z	

Complement

- **Complement f**
  - comf f,d: Bitwise complement of the contents of memory location f
    - Write the result to the W register if the d bit is set to 0
    - Write the result to the memory location f if the d bit is set to 1
- Example
  - Write B'00101010' to the memory location 0x0D. Compute the complement of the memory location 0x0D. Write the result to 0x0D.

```
movlw   b'00101010';
movwf   0x0D;
comf 0D,1;
```

Result is 11010101

And

- The AND function gives a true output whenever all inputs are true.
- Another way of looking at this truth table is to regard the X input as a Control input and the Y input as a data input.
- IF the Control input is 0 THEN the output is always 0.
- IF the Control input is 1 THEN:
  - IF the Data input Y is 0 THEN the output is 0.
  - IF the Data input Y is 1 THEN the output is 1. That is f = Y.
- ANDing an input with a 0 always gives a 0 output.
- ANDing an input with a 1 does not change the logic value of the other input.
- We can therefore use the AND function to zero bits. For example: 10101010 · 00001111 = 00001010

AND instructions: ANDLW

- AND Literal with W
  - andlw k: Bitwise AND of the contents of the W register and the literal k
  - Write the result to the W register
- Example
  - Write B'00101010' to the W register. AND W and B'10110110'.

```
movlw   b'00101010';
andlw   b'10110110';
```

Result is 00100010

ANDWF

- AND W with f
  - andwf f,d: Bitwise AND of the contents of the W register and the memory location f
    - Write the result to the W register if the d bit is set to 0
    - Write the result to the memory location f if the d bit is set to 1
- Example
  - Compute B'00101010' AND B'10110110'. Write the result to 0x0C.

```
movlw   b'00101010';
movwf   0x0C;
movlw   b'10110110';
andwf 0x0C, 1
```

Result is 00100010

Inclusive OR

- The Inclusive-OR function gives a true output whenever any input is true.
- Assume X input as a Control input and the Y input as a data input,
  - IF the Control input is 0 THEN, f = Y :
    - IF the Data input Y is 0 THEN the output is 0.
    - IF the Data input Y is 1 THEN the output is 1.
  - IF the Control input is 1 THEN the output is always 1.
- For example:

10101010 +

11111000 =

11111010

Inclusive OR: IORLW

- Inclusive OR Literal with W
  - iorlw k: Bitwise XOR of the contents of the W register and the literal k
  - Write the result in the W register
- Example: Write B'00101010' to the W register. OR W and B'10110110'.

```
movlw   b'00101010';
iorlw    b'10110110';
```

Result is 10111110

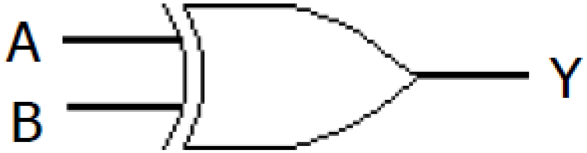
Inclusive OR: IORWF

- Inclusive OR W with f
  - iorwf f,d: Bitwise OR of the contents of the W register and the memory location f
    - Write the result to the W register if the d bit is set to 0
    - Write the result to the memory location f if the d bit is set to 1
- Example: Compute B'00101010' OR B'10110110'. Write the result to 0x0E.

```
movlw   b'00101010';
movwf   0x0E;
movlw   b'10110110';
iorwf 0x0E, 1
```

Result is 10111110

The XOR (Exclusive-OR)



- This is a XOR gate.
- XOR gates assert their output when exactly one of the inputs is asserted, hence the name.
- The switching algebra symbol for this operation is ⊕, i.e. 1 ⊕ 1 = 0 and 1 ⊕ 0 = 1.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

XOR instructions: XORWF

- Exclusive OR W with f
  - xorwf f,d: Bitwise XOR of the contents of the W register and the memory location f
    - Write the result to the W register if the d bit is set to 0
    - Write the result to the memory location f if the d bit is set to 1
- Example: Compute B'00101010' XOR B'10110110'. Write the result to 0x0F.

```
movlw   b'00101010';
movwf   0x0F;
movlw   b'10110110';
xorwf 0x0F, 1
```

Result is 10011100

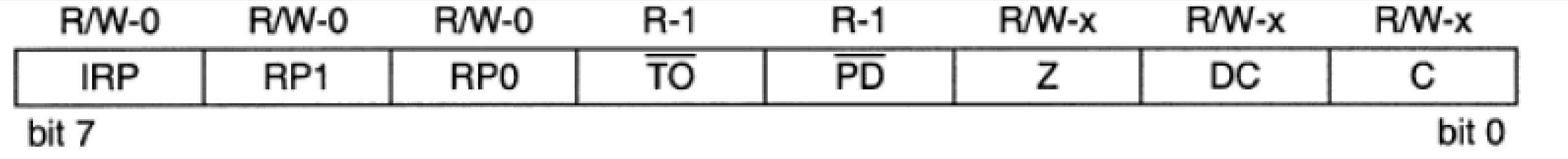
XOR instructions: XORLW

- Exclusive OR Literal with W
  - xorlw k: Bitwise XOR of the contents of the W register and the literal k
  - Write the result to the W register
- Example: Write B'00101010' to the W register. XOR W and B'10110110'.

```
movlw   b'00101010';
xorlw    b'10110110';
```

Result is 10011100

Status Register and AND Instructions

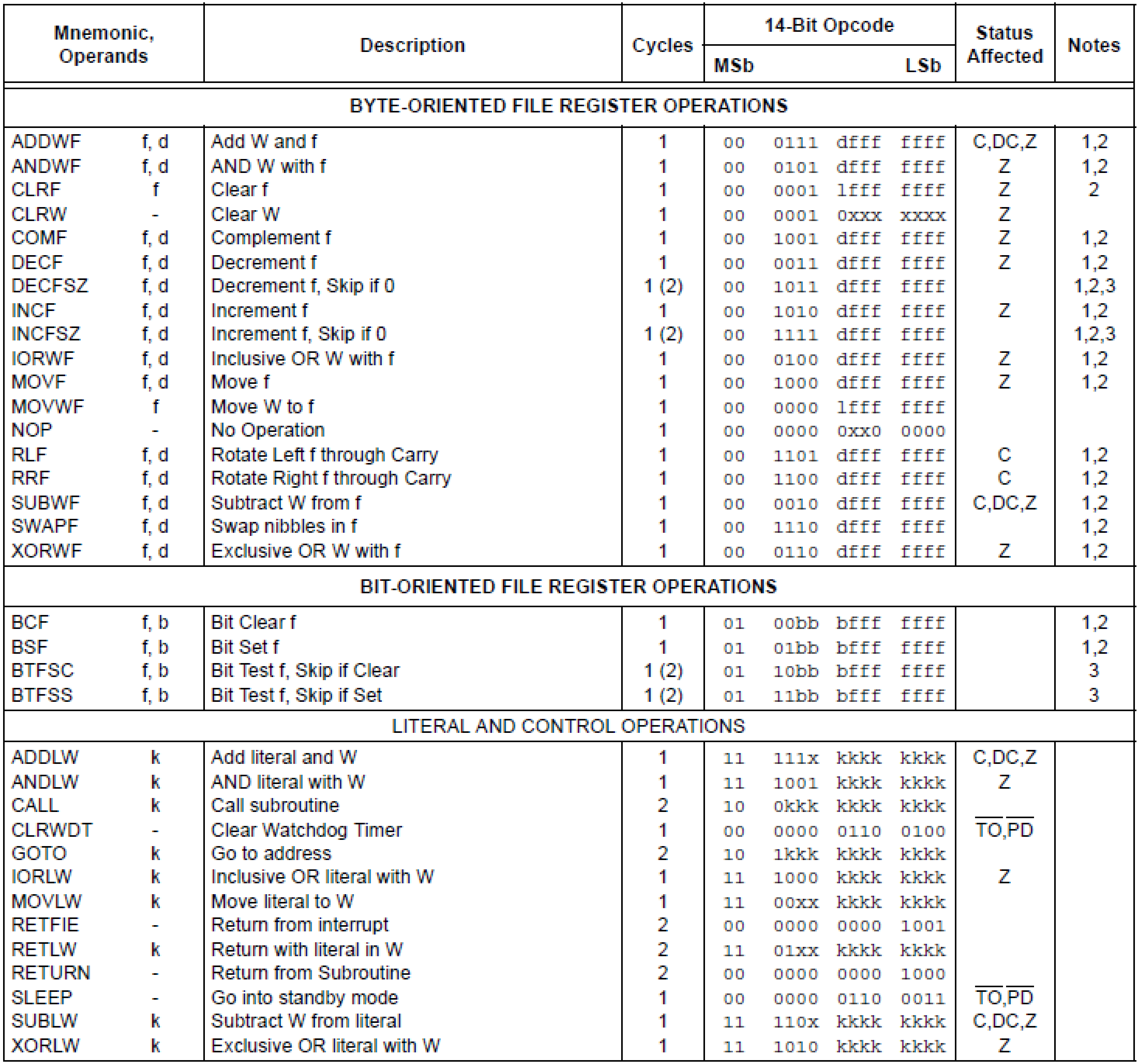


R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

- Flag Z in the Status Register (bit 2) is set if the result of any logic operation is zero
- Another use of ANDing is to check the state of any bit or bits in a data; for example:

```
andlw b'00000011' ; Check bits 0 & 1
btfsc STATUS,Z ; IF not both zero THEN skip
goto ALL_ZERO ; ELSE == 00, so go to ALL_ZERO routine
```

- **Q: Write program that will put a byte in PORTB showing which bits in both Files h'30' and h'31' which are 1.**



Mnemonic, Operands		Description	Cycles	14-Bit Opcode				Status Affected	Notes
				MSb			LSb		
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	

Example

- □ Write program that will put a byte in PORTB showing which bits in both Files h’30’ and h’31’ which are 1.

```
LIST P=16F84A
INCLUDE ‘’P16F84A.INC’’
        CLRF PORTB
        BSF STATUS, 5 ; in BANK1
        CLRF TRISB ;PORTB is output
        BCF STATUS, 5 ; in BANK0
TEST_PORTA
        MOVWF h’30’;
        ANDWF h’30’,0 ;
        MOVWF PORTB ;
        END
```

What Does The **XOR** Instruction Do? **TOGGLING**

- □ We can write an instruction that selects a particular output bit (LED) and changes the state of only this bit. This will turn a LED on or turn it off.
- □ If the bit is ON, it will be turned OFF. And if the instruction is processed again, the bit will be turned ON. This is called TOGGLING. With this instruction we do not have to know the state of the bit. It can be "1" or "0." The instruction will simply change the state.

```
toggle movlw   10h       ;Put 0001 0000 into w to toggle GP4
       xorwf  gpio,f    ;the only bit that will change is bit4
```

```
reverse  movlw    b'00000011'    ;this is the MASK to reverse the two lowest bits
                                ;fileA contains b'11111101'  (before)
         xorwf     fileA,f      ; fileA contains  b'11111110'  (after)
```

http://www.talkingelectronics.com/projects/PICK%20A%20PIC%20Project/XOR-Trick-1.html

What Does The **XOR** Instruction Do?
MATCHING TWO FILES = COMPARISON = COMPARE

- We can also use the XOR function to detect a MATCH between two files.
- To find out if two numbers are the same, they are XOR-ed together. Since each binary digit (bit) will be the same, the result will be (0000 0000).
- For example, if we have two files: b'00010011' and b'00010011' bit0 in each file is '1' bit1 in each file is "1" bit2 in each file is "0" etc etc etc. In fact all bits are the same. When all bits are the same, this will SET the zero flag in the Status (03) file and by testing bit 2 (the z flag) you can include an instruction in your program to skip the next instruction when the z bit is set.

```
match  movlw   0Ch        ; load 0Chex into w
       xorwf   motor      ; see if "motor" file holds 0Chex
       btfss   status,2   ; test the z flag to see if it is SET
       goto    notsame    ;z flag is not set
       goto    same       ;z flag set = files are both 0Chex
```

http://www.talkingelectronics.com/projects/PICK%20A%20PIC%20Project/XOR-Trick-1.html

What Does The **XOR** Instruction Do?

EXCHANGE THE CONTENTS OF A FILE WITH W

- This code exchanges the contents of a file with the w register. It doesn't require the use of another file to store an intermediate value.
- file holds: b'0001 1100'   w holds:b'0000 1111'

```
f_x_w xorwf  file,f   ;before: file=b'0001 1100' w=b'0000 1111'
                      ;after: file=b'0001 0011'  w=b'0000 1111'
      xorwf  file,w   ;after: file=b'0001 0011' w=b'0001 1100'
      xorwf  file,f   ;after: file=b'0000 1111' w=b'0001 1100'
```

http://www.talkingelectronics.com/projects/PICK%20A%20PIC%20Project/XOR-Trick-1.html

What Does The **XOR** Instruction Do?

EXCHANGE THE CONTENTS OF TWO FILES

- This code exchanges the contents of two files using the w register. It doesn't require the use of another file to store an intermediate value.

```
exch  movf    file2,w   ;file1=b'0001 0011' file2=b'0000 1111'
                        ;w=b'0000 1111' (after execution)
      xorwf   file1,f   ;file1=b'0001 1100' w=b'0000 1111'
      xorwf   file1,w   ;file1=b'0001 1100' w=b'0001 0011'
      xorwf   file1,f   ;file1=b'0000 1111' w=b'0001 0011'
      movwf  file2       ;file2=b'0001 0011'
```

http://www.talkingelectronics.com/projects/PICK%20A%20PIC%20Project/XOR-Trick-1.html

Example

- □ Check if PORTA contains B'01010101`

```
match  movlw   b’01010101’ ; load a byte into w
       xorwf    05h,0      ; see if “PORTA" file holds byte
       btfss   status,2    ; test the z flag to see if it is SET
       goto    notsame     ;z flag is not set
       goto    same        ;z flag set = files are the same
```

Example

- Check if bit 4 of PORTA is 1 using **andlw.**

```
loop  movf porta,0 ; move portA to W register
      andlw b’00010000’ ; Check bit 4
      btfss   status,2   ; test the z flag to see if it is SET
      goto    loop   ;z flag is not set
      goto    same           ;z flag set = files are the same
```

Example

- □ Write a program that turns on a LED at RB0 if the buttons connected to RA1 & RA2 are pressed. Use **xorlw**.

```
          list p=16f84a
          include “p16f84a.inc”
          clrf portb
          bsf status, 5 ; in bank1
          clrf trisb ; portb is output
          movlw h’ff’
          movwf trisa ; porta is input
          bcf status, 5 ; in bank0

loop   movf porta,0 ; move portA to W register
          xorlw b’00000110’ ; Check bits 1 & 2
          btfss    status,2    ; test the z flag to see if it is SET
          goto     loop    ;z flag is not set
          movlw h’01’ ;if yes (z flag set = files are the same)
          movwf portb ;portb=01
```

Example

- □ Write a program that turns on a LED at RB0 if the buttons connected to RA1 & RA2 are pressed. Use **iorlw**.

```
     list p=16f84a
     include “p16f84a.inc”
     clrf portb
     bsf status, 5 ; in bank1
     clrf trisb ; portb is output
     movlw b’00000110’
     movwf trisa ; RA1 & RA2 are input
     bcf status, 5 ; in bank0
     clrf porta
loop  movf porta,0 ; move portA to W register
      iorlw b’11111001’ ; Check bits 1 & 2
      movwf 0x0C; move W register to 0x0C
      comf 0x0C,0; complement byte and put result to W
      btfss   status,2   ; test the z flag to see if it is SET
      goto    loop   ;z flag is not set
      movlw h’01’ ;if yes (z flag set = files are the same)
      movwf portb ;portb=01
```

Example

- □ Turn on a LED at RB1 if either RA3 or RB5 are 1.

```
        list p=16f84a
        include “p16f84a.inc”
        bsf status, 5 ; in bank1
        movlw b‘00100000’;
        movwf trisb; RB5 is input, rest is output.
        movlw b‘00001000’;
        movwf trisa ; RA3 is input
        bcf status, 5 ; in bank0
        clrf porta;
        clrf portb;

loop    movf porta,0 ; move portA to W register
        iorwf portb,0; b’00000110’ ; inclusive or RA3 and RB5
        xorwl b’00101000’; check bits RA3 and RB5
        btfss   status,2    ; test the z flag to see if it is SET
        goto    loop    ;z flag is not set
        movlw h’02’ ;if yes (z flag set = files are the same)
        movwf portb ;portb=02
```

Example

- If the buttons connected RA0, RA2 and RA3 of PORTA are pressed, all leds of PORTB are turned on.

```
LIST P=16F84A
INCLUDE ‘’P16F84A.INC’’
        CLRF PORTB
        BSF STATUS, 5 ; in BANK1
        CLRF TRISB ;PORTB is output
        MOVLW h’FF’
        MOVWF TRISA ; PORTA is input
        BCF STATUS, 5 ; in BANK0
TEST_PORTA
        MOVLW b’00001101’ ; W=b’ 00001101’
        XORWF PORTA,W ;W=PORTA xor W
        BTFSS STATUS,2 ; Z=1?
        GOTO TEST_PORTA ; if NO
ON
        MOVLW h’FF’ ;if YES
        MOVWF PORTB ;PORTB=FF
        END
```