SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

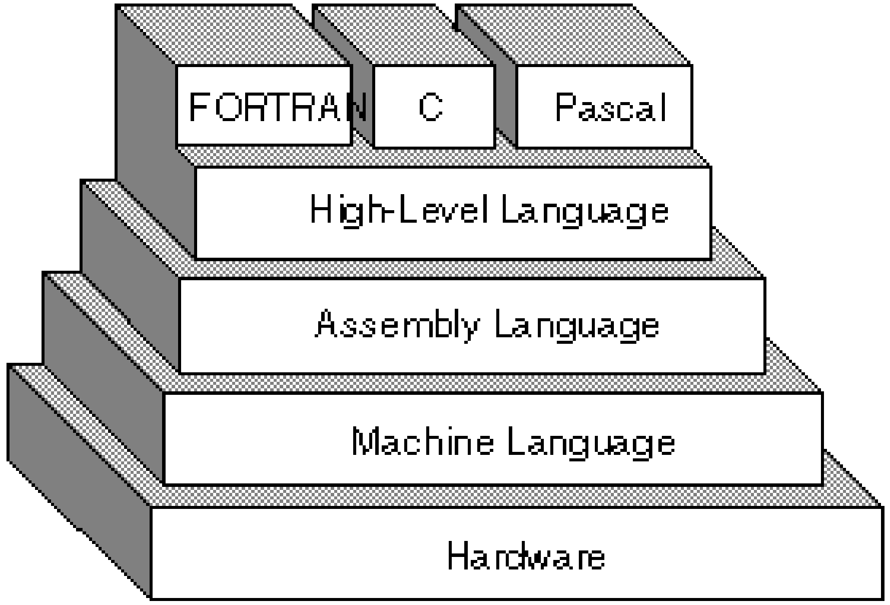
DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

PIC16F877-Instruction Set

Dr.G.Arthy
Assistant Professor
Department of EEE
SNS College of Engineering

Programming Language

- The lowest-level language is called **Machine languages**. It means those languages which are closer to the understanding of machine rather than human beings. A machine language thus comprises a string of binary 0's and 1's.
- Machine language is actually a coded set of instructions for a particular CPU (Central Processing Unit), and it is also known as a **machine code**.
- A machine language is designed to be used by a computer without the need of translation.



Machine Language

Disadvantage :

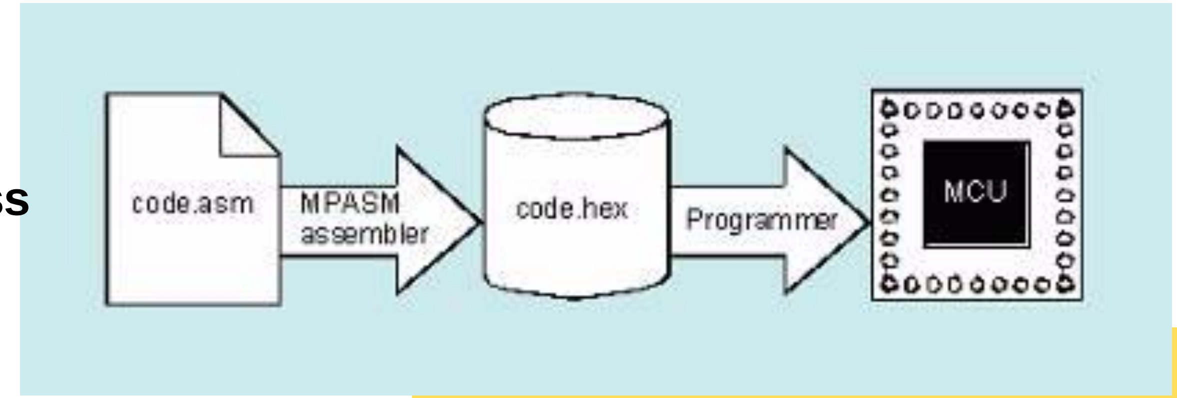
1. It is a machine dependent programming language. Machine dependent means the program designed in one type of machine or computer could not be run on other type of computer or machine. So programs designed in the machine language in one computer are not easily portable to other computers.

2. It is a very difficult language to understand and learn. If there is any problem in the program, written in machine language, then it is very difficult to find out the correct mistake.

Assembly language

- Assembly language is one level above the machine language. (Both Machine and Assembly language are considered low-level language)
- It uses certain predefined symbolic codes instead of binary codes. These symbolic codes are called **mnemonics**.
- Assembly language programs are translated into machine language by a program called an **assembler**.

Assembler Process



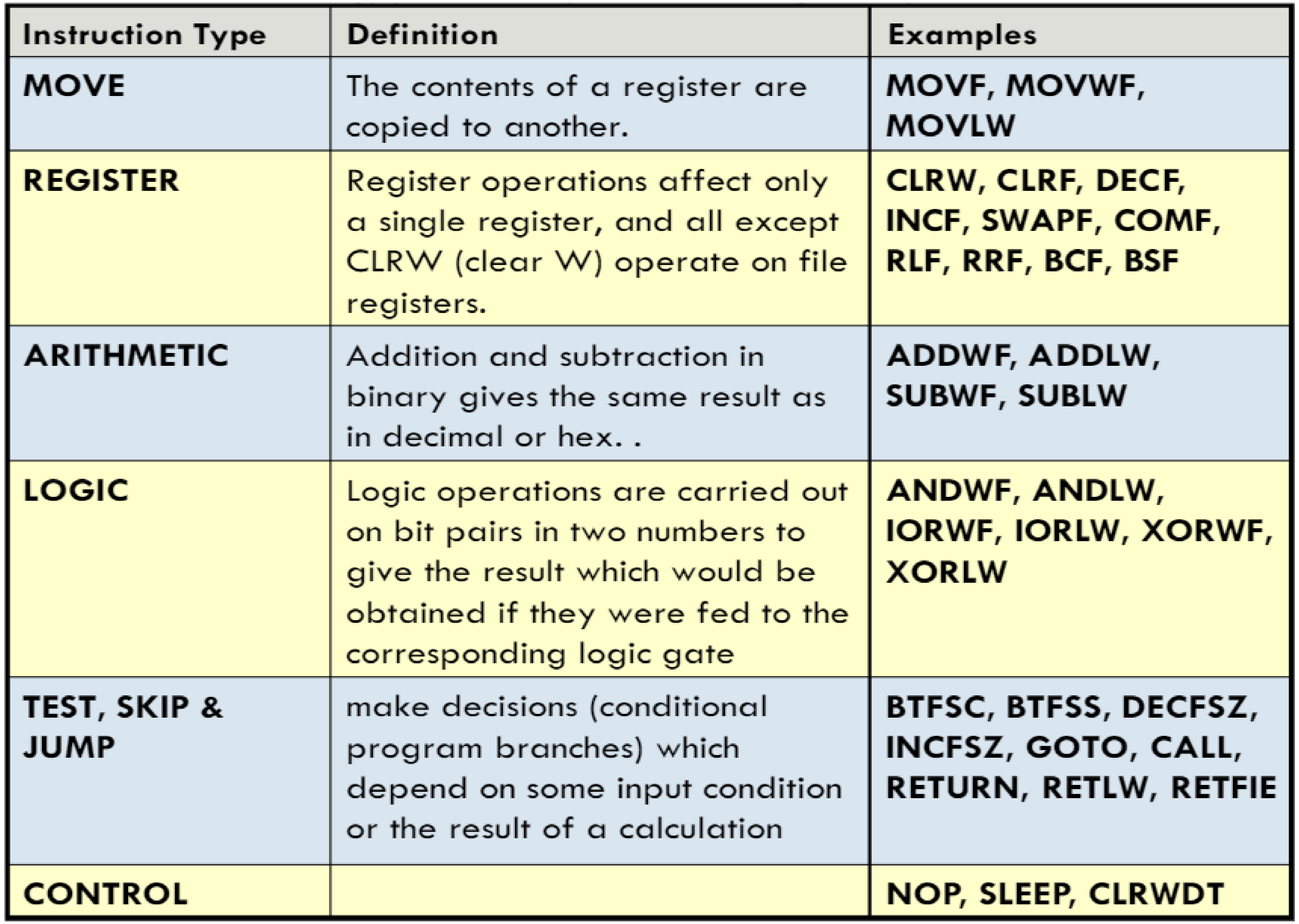
High-level language

- High-Level Language overcomes the limitation of writing a program in Machine and Assembly language as it is difficult and time consuming.
- In High-Level Language, the programs can be written using simple English words. Examples of High-Level Language are BASIC, Fortran, COBOL, C, C++.
- Programs written in high-level languages are translated into machine language by a **compiler**.

Instructions of PIC

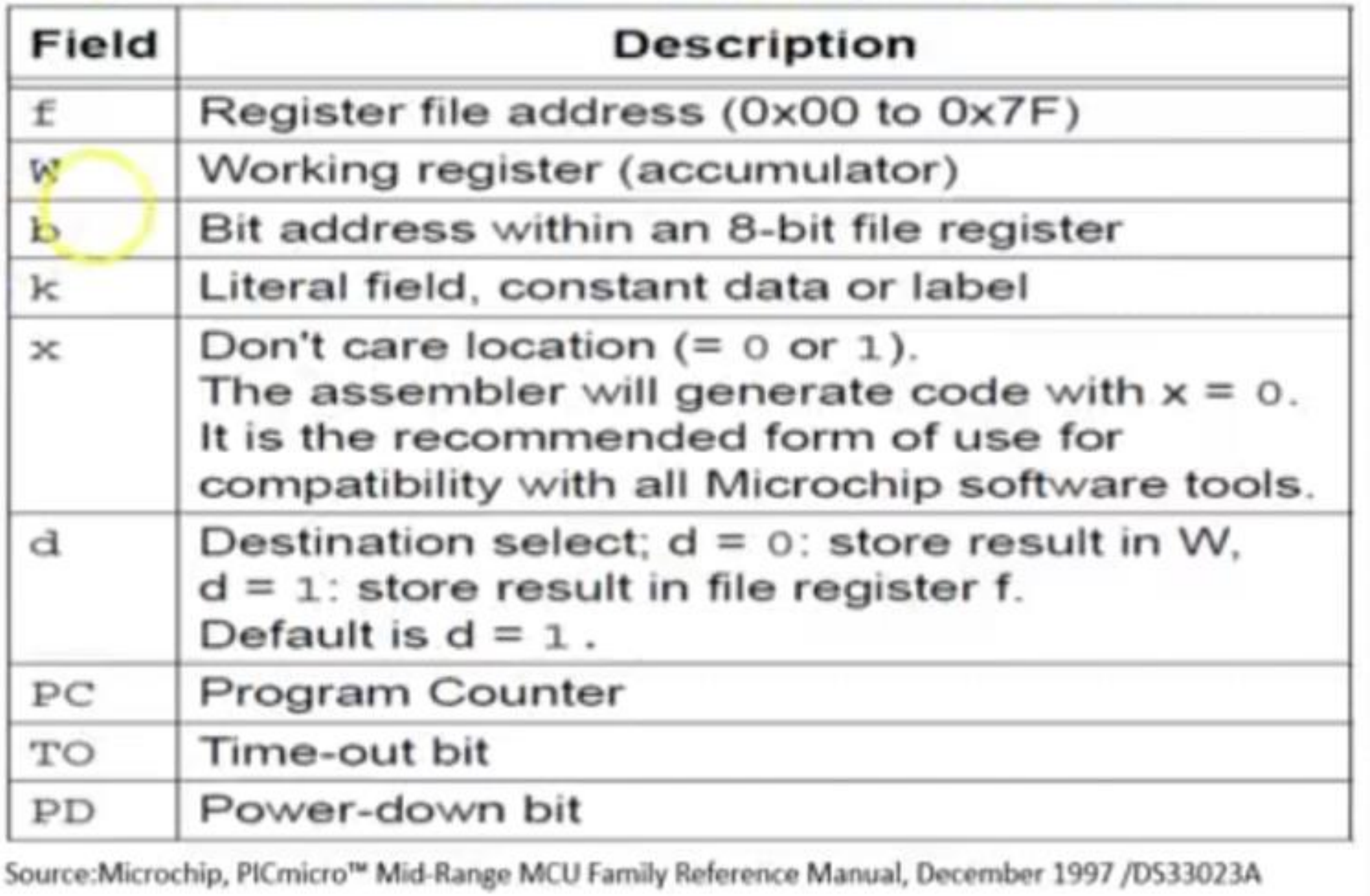
- PIC16F877 has 35 instructions.
- Each instruction if 14 bit words.
- RISC architecture

Instruction Set



Instruction Type	Definition	Examples
MOVE	The contents of a register are copied to another.	**MOVF, MOVWF, MOVLW**
REGISTER	Register operations affect only a single register, and all except CLRW (clear W) operate on file registers.	**CLRW, CLRF, DECF, INCF, SWAPF, COMF, RLF, RRF, BCF, BSF**
ARITHMETIC	Addition and subtraction in binary gives the same result as in decimal or hex. .	**ADDWF, ADDLW, SUBWF, SUBLW**
LOGIC	Logic operations are carried out on bit pairs in two numbers to give the result which would be obtained if they were fed to the corresponding logic gate	**ANDWF, ANDLW, IORWF, IORLW, XORWF, XORLW**
TEST, SKIP & JUMP	make decisions (conditional program branches) which depend on some input condition or the result of a calculation	**BTFSC, BTFSS, DECFSZ, INCFSZ, GOTO, CALL, RETURN, RETLW, RETFIE**
CONTROL		**NOP, SLEEP, CLRWDT**

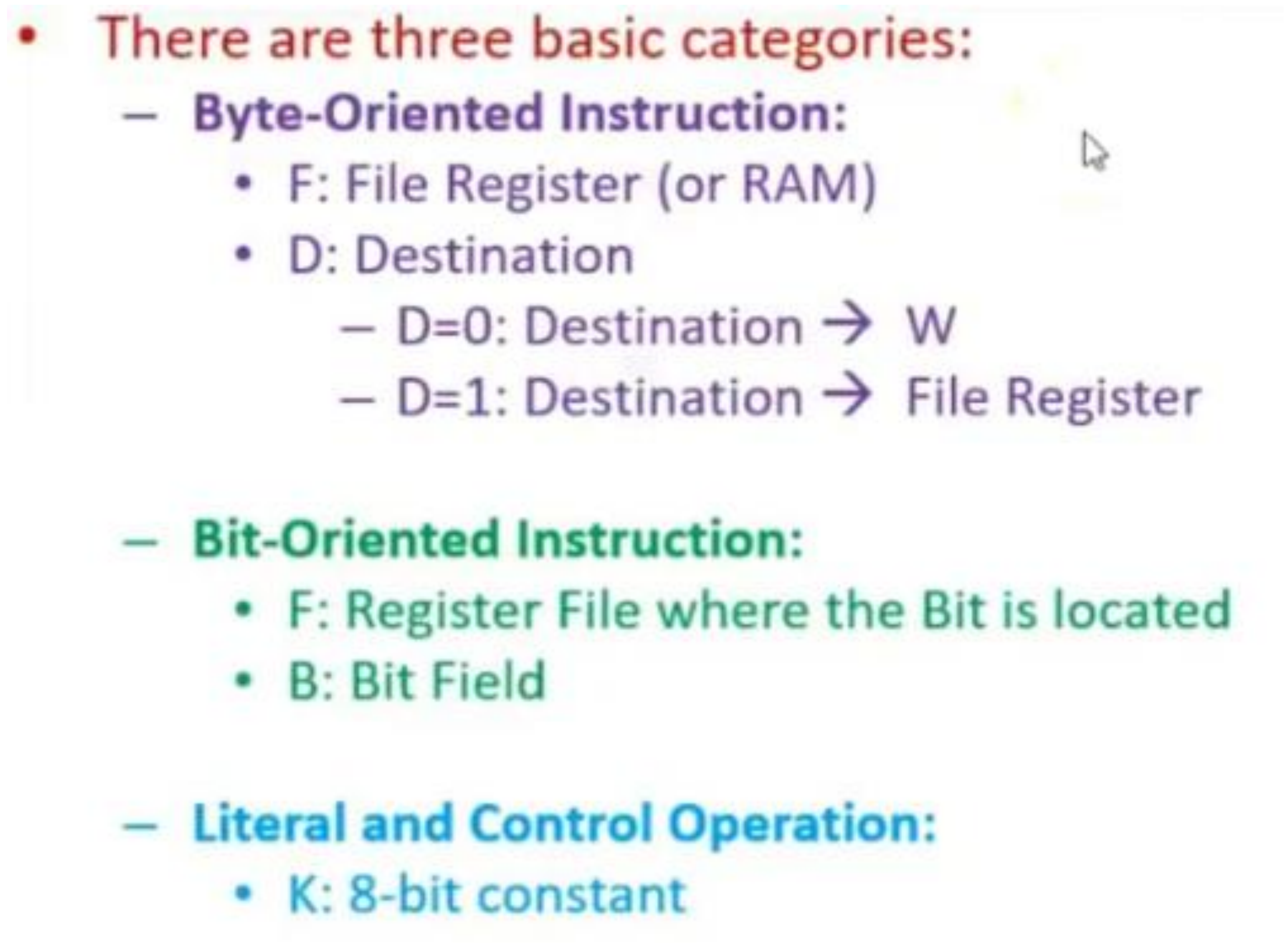
Opcode field description



Field	Description
f	Register file address (0x00 to 0x7F)
W	Working register (accumulator)
b	Bit address within an 8-bit file register
k	Literal field, constant data or label
x	Don't care location (= 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. Default is d = 1 .
PC	Program Counter
TO	Time-out bit
PD	Power-down bit

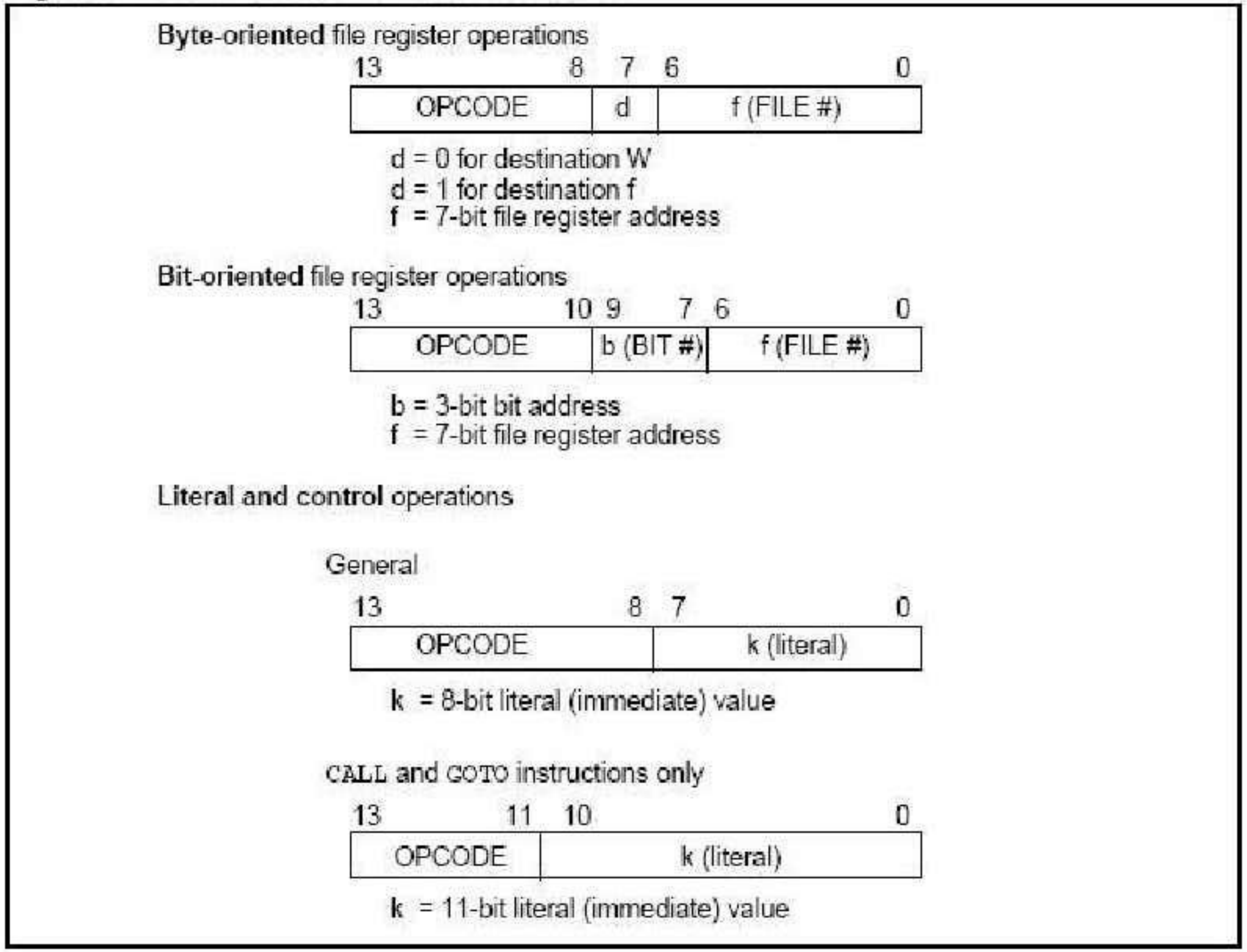
Source:Microchip, PICmicro™ Mid-Range MCU Family Reference Manual, December 1997 /DS33023A

Basic Categories of Instructions



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

Figure 29-1: General Format for Instructions

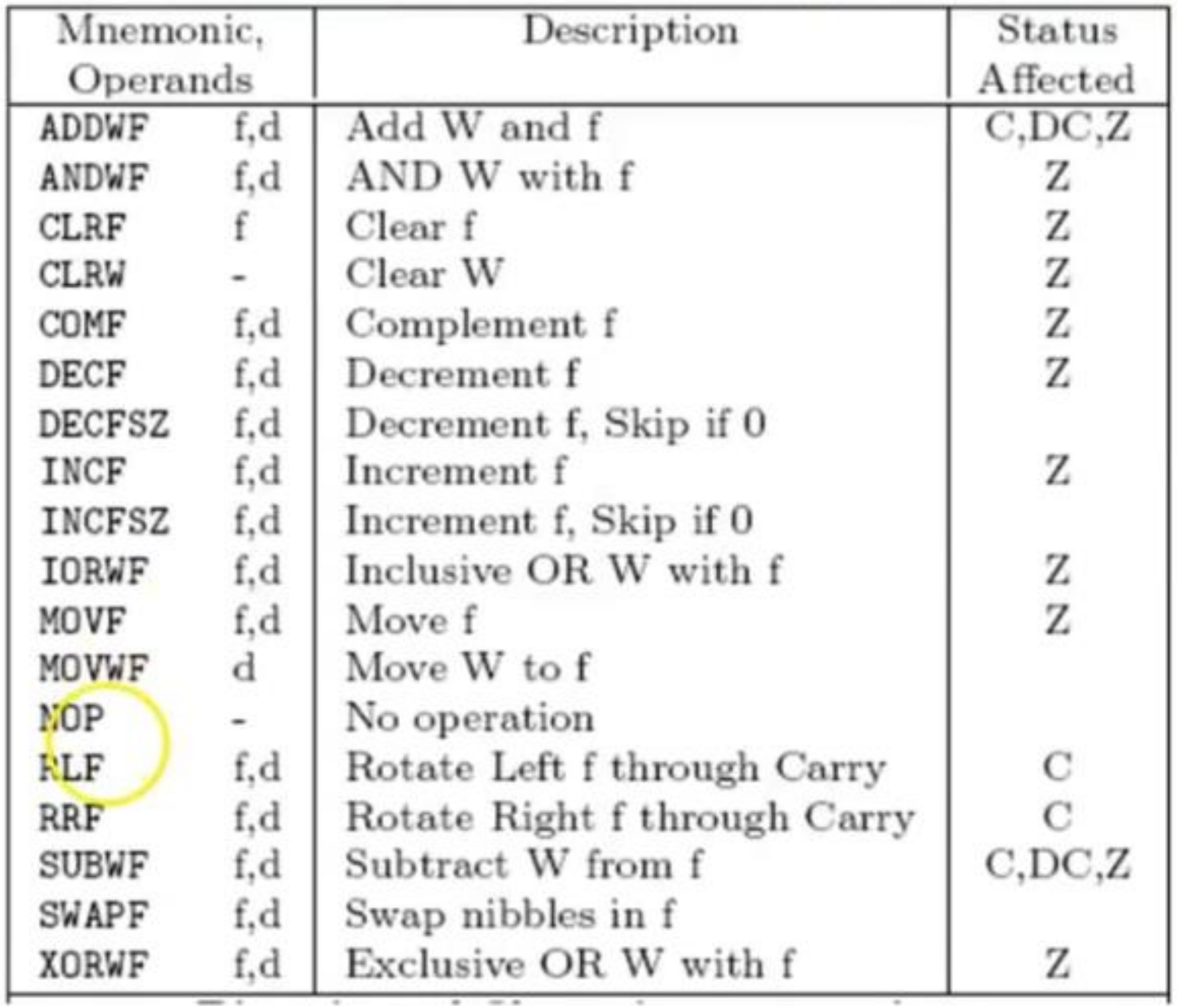

Byte-oriented file register operations
13
8
7
6
0
OPCODE
d
f (FILE #)
d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address
Bit-oriented file register operations
13
10
9
7
6
0
OPCODE
b (BIT #)
f (FILE #)
b = 3-bit bit address
f = 7-bit file register address
Literal and control operations
General
13
8
7
0
OPCODE
k (literal)
k = 8-bit literal (immediate) value
CALL and GOTO instructions only
13
11
10
0
OPCODE
k (literal)
k = 11-bit literal (immediate) value

PIC16F877A Instruction set

- Some instructions with alternate result destinations. **The default destination for the result** of an operation is the **file register**, but the working register **W is sometimes an option**.

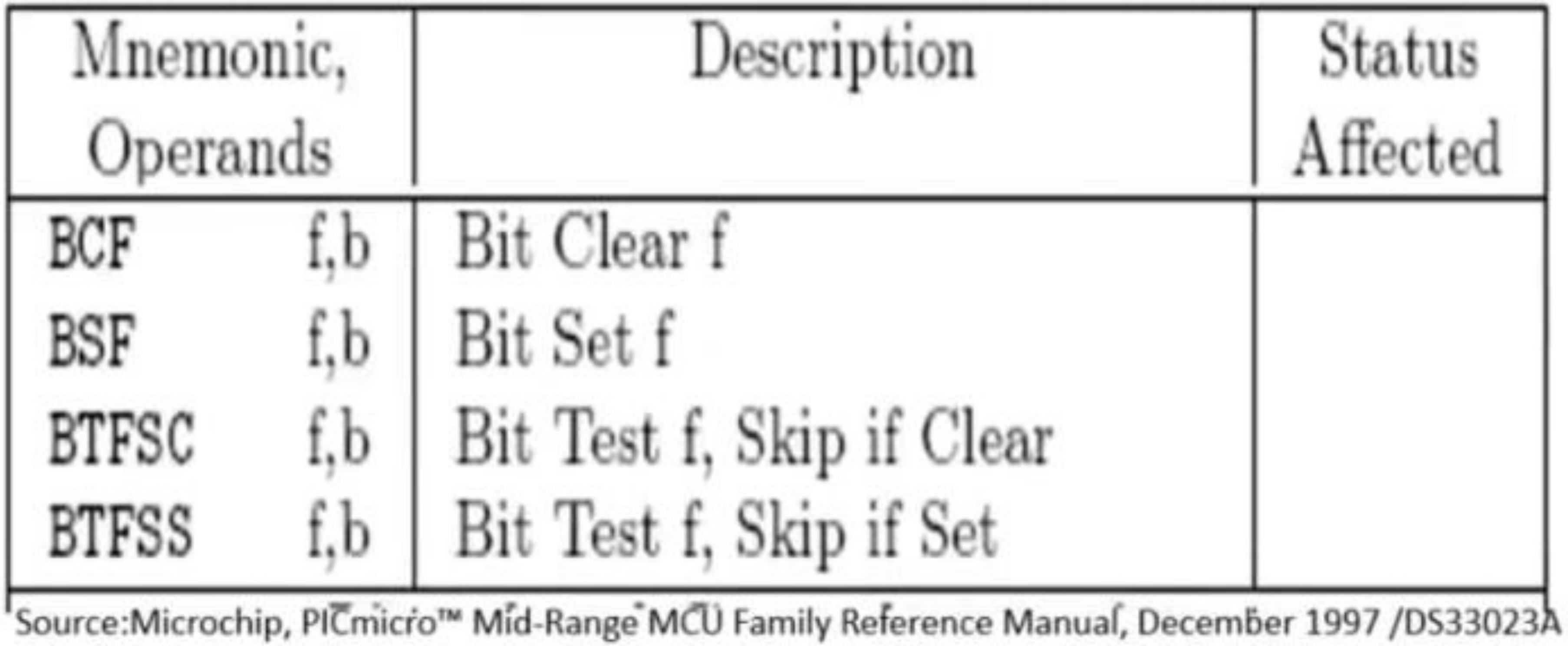
- There are three basic categories:
  - **Byte-Oriented Instruction:**
    - F: File Register (or RAM)
    - D: Destination
      - D=0: Destination → W
      - D=1: Destination → File Register
  - **Bit-Oriented Instruction:**
    - F: Register File where the Bit is located
    - B: Bit Field
  - **Literal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

Byte oriented file register Operation



Mnemonic, Operands		Description	Status Affected
ADDWF	f,d	Add W and f	C,DC,Z
ANDWF	f,d	AND W with f	Z
CLRF	f	Clear f	Z
CLRW	-	Clear W	Z
COMF	f,d	Complement f	Z
DECF	f,d	Decrement f	Z
DECFSZ	f,d	Decrement f, Skip if 0	
INCF	f,d	Increment f	Z
INCFSZ	f,d	Increment f, Skip if 0	
IORWF	f,d	Inclusive OR W with f	Z
MOVF	f,d	Move f	Z
MOVWF	d	Move W to f	
NOP	-	No operation	
RLF	f,d	Rotate Left f through Carry	C
RRF	f,d	Rotate Right f through Carry	C
SUBWF	f,d	Subtract W from f	C,DC,Z
SWAPF	f,d	Swap nibbles in f	
XORWF	f,d	Exclusive OR W with f	Z

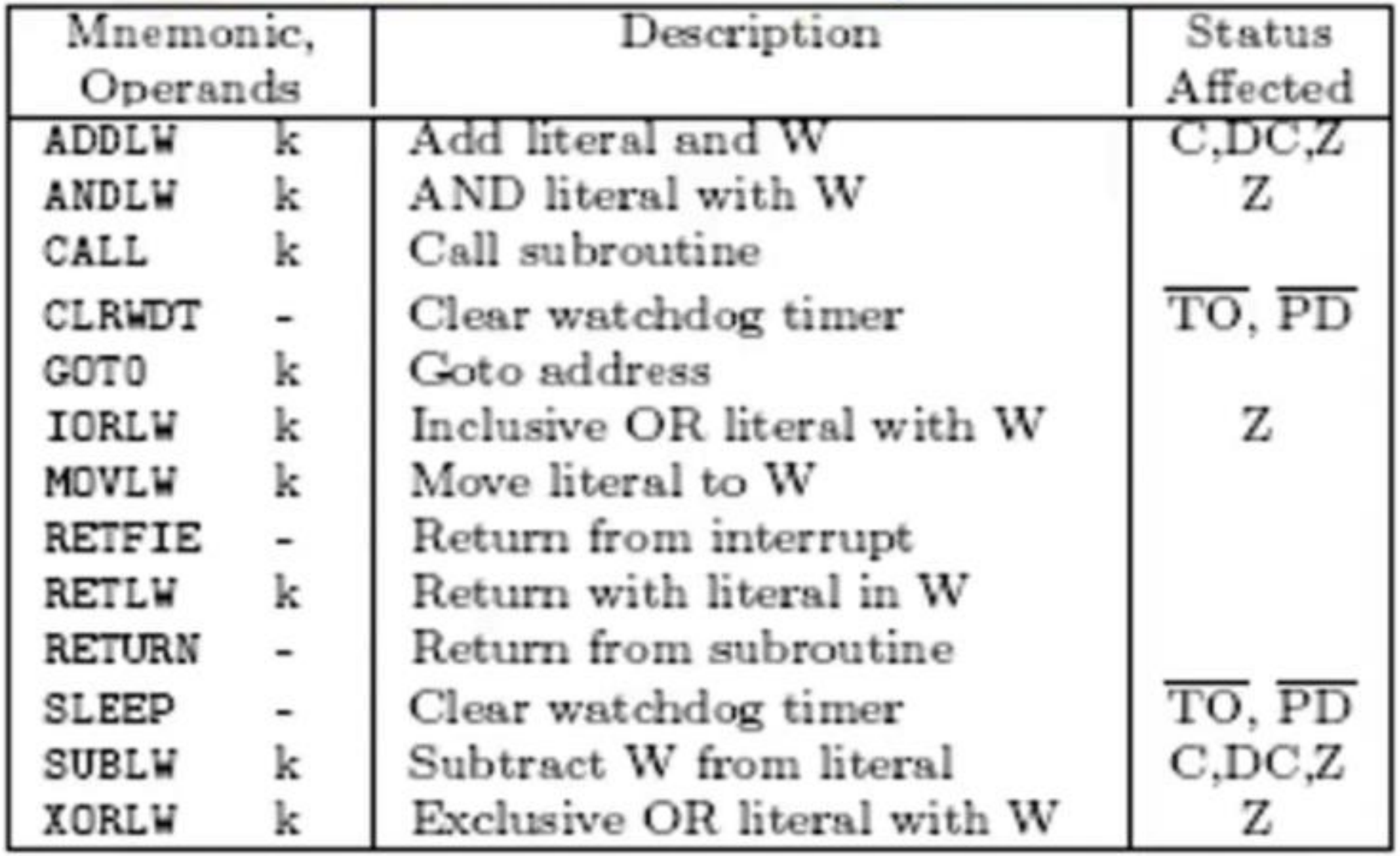
Bit oriented file register Operation



Mnemonic, Operands	Description	Status Affected
BCF f,b	Bit Clear f	
BSF f,b	Bit Set f	
BTFSC f,b	Bit Test f, Skip if Clear	
BTFSS f,b	Bit Test f, Skip if Set	

Source:Microchip, PICmicro™ Mid-Range MCU Family Reference Manual, December 1997 /DS33023A

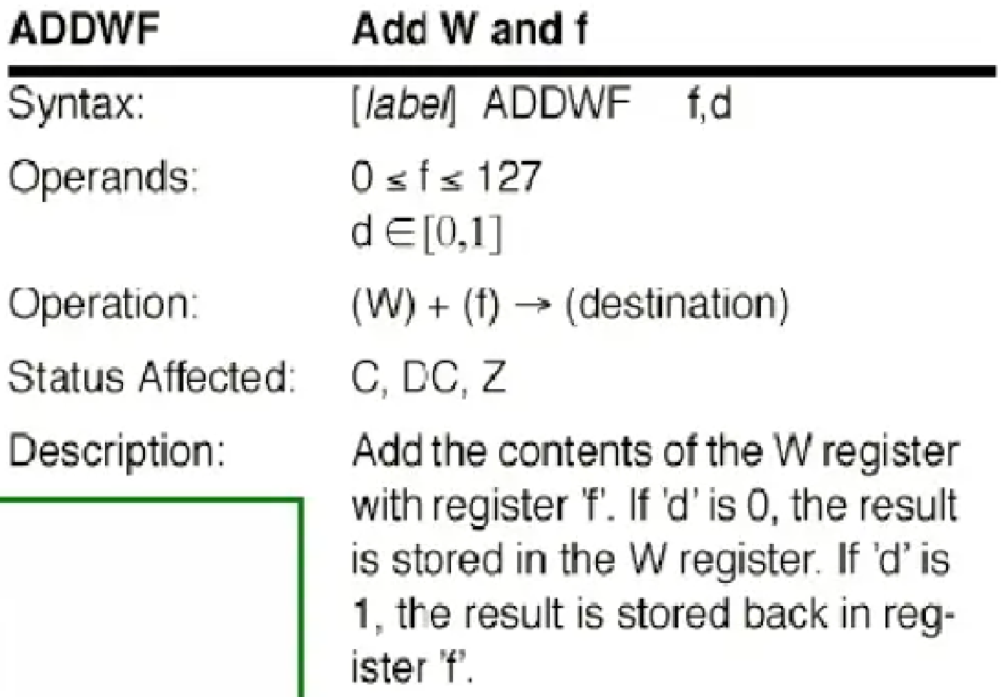
Literal and Control operations



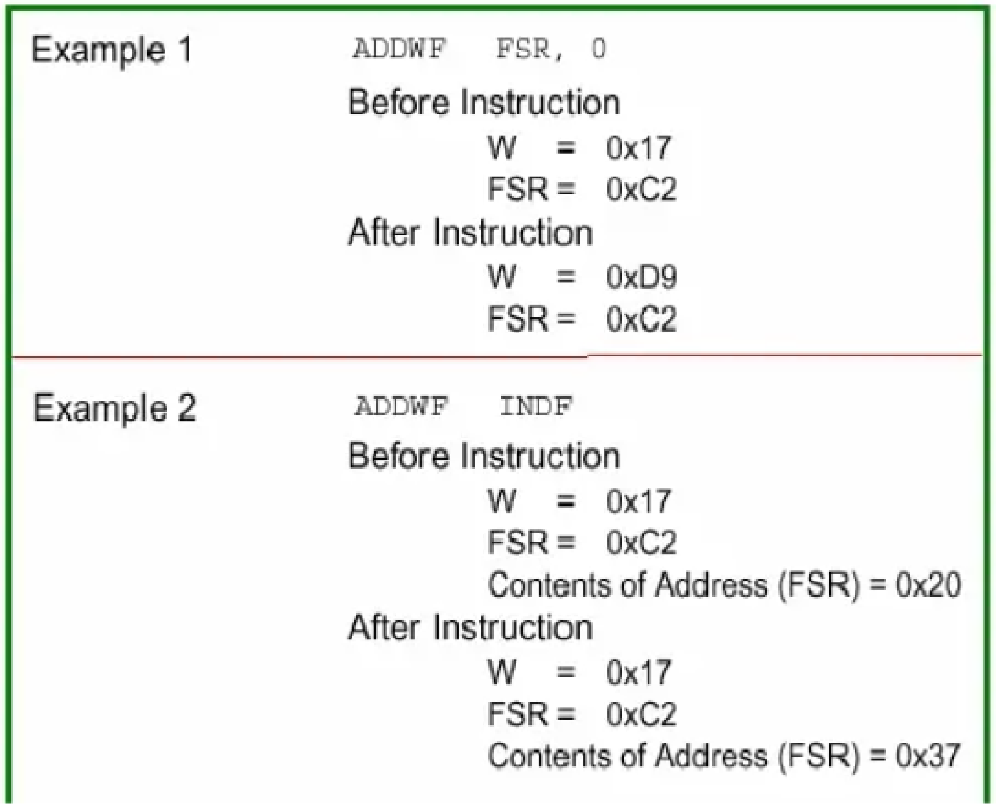
Mnemonic, Operands		Description	Status Affected
ADDLW	k	Add literal and W	C,DC,Z
ANDLW	k	AND literal with W	Z
CALL	k	Call subroutine	
CLRWDT	-	Clear watchdog timer	$\overline{TO}$, $\overline{PD}$
GOTO	k	Goto address	
IORLW	k	Inclusive OR literal with W	Z
MOVLW	k	Move literal to W	
RETFIE	-	Return from interrupt	
RETLW	k	Return with literal in W	
RETURN	-	Return from subroutine	
SLEEP	-	Clear watchdog timer	$\overline{TO}$, $\overline{PD}$
SUBLW	k	Subtract W from literal	C,DC,Z
XORLW	k	Exclusive OR literal with W	Z

Source:Microchip, PICmicro™ Mid-Range MCU Family Reference Manual, December 1997 /DS33023A

ADDWF



ADDWF	Add W and f
Syntax:	[*label*] ADDWF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	$(W) + (f) \rightarrow$ (destination)
Status Affected:	C, DC, Z
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'.



Example 1 `ADDWF FSR, 0`

Before Instruction

```
W   =  0x17
FSR =  0xC2
```

After Instruction

```
W   =  0xD9
FSR =  0xC2
```

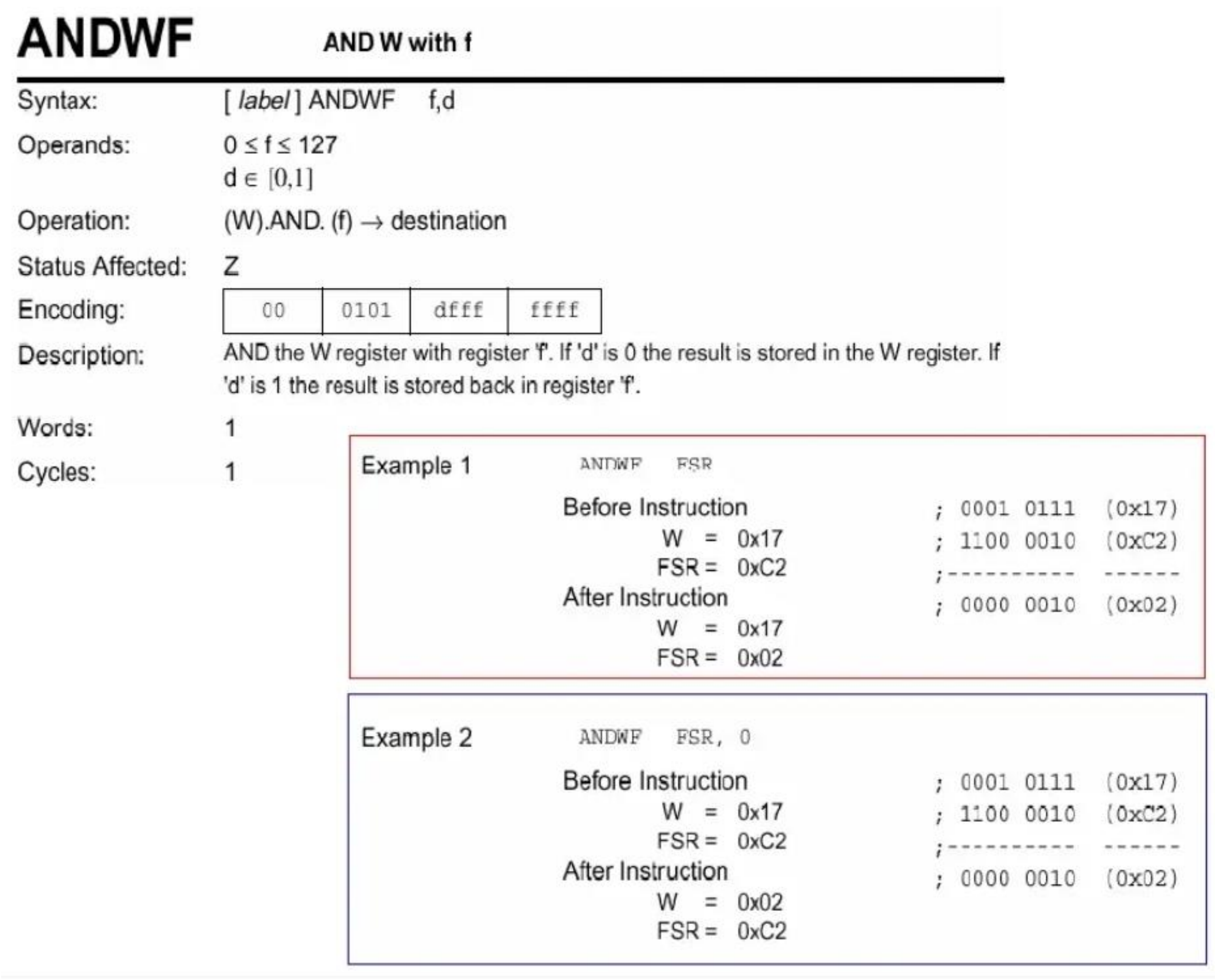
Example 2 `ADDWF INDF`

Before Instruction

```
W   =  0x17
FSR =  0xC2
Contents of Address (FSR) = 0x20
```

After Instruction

```
W   =  0x17
FSR =  0xC2
Contents of Address (FSR) = 0x37
```



ANDWF AND W with f

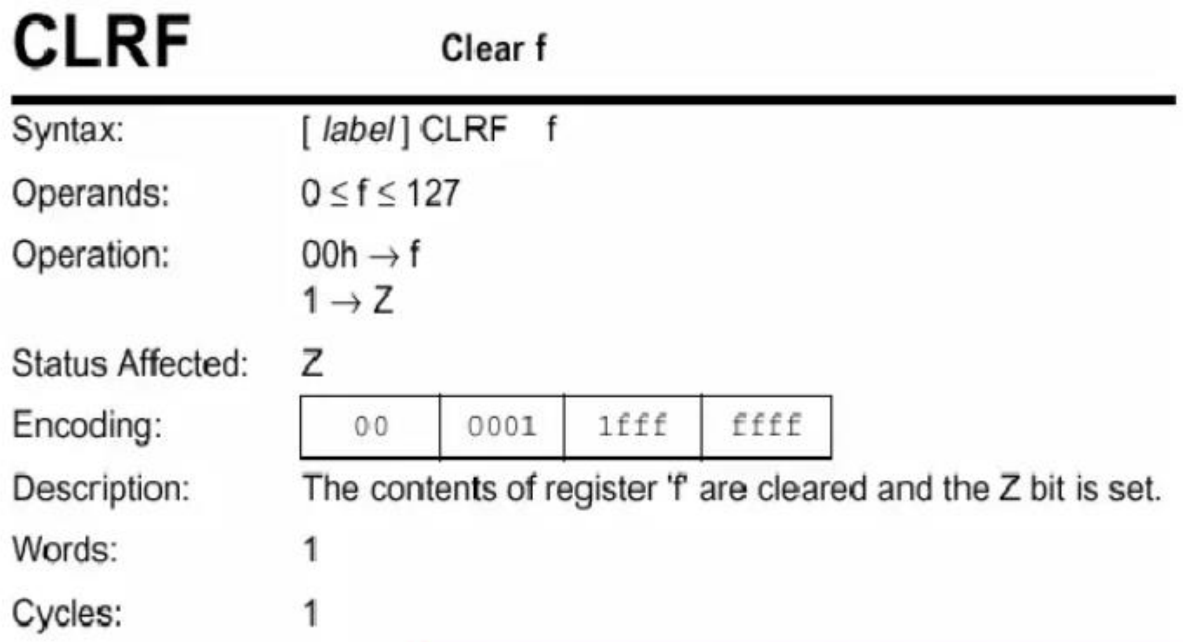
Syntax:	[*label*] ANDWF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
Operation:	(W).AND. (f) → destination
Status Affected:	Z
Encoding:	00 | 0101 | dfff | ffff
Description:	AND the W register with register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back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

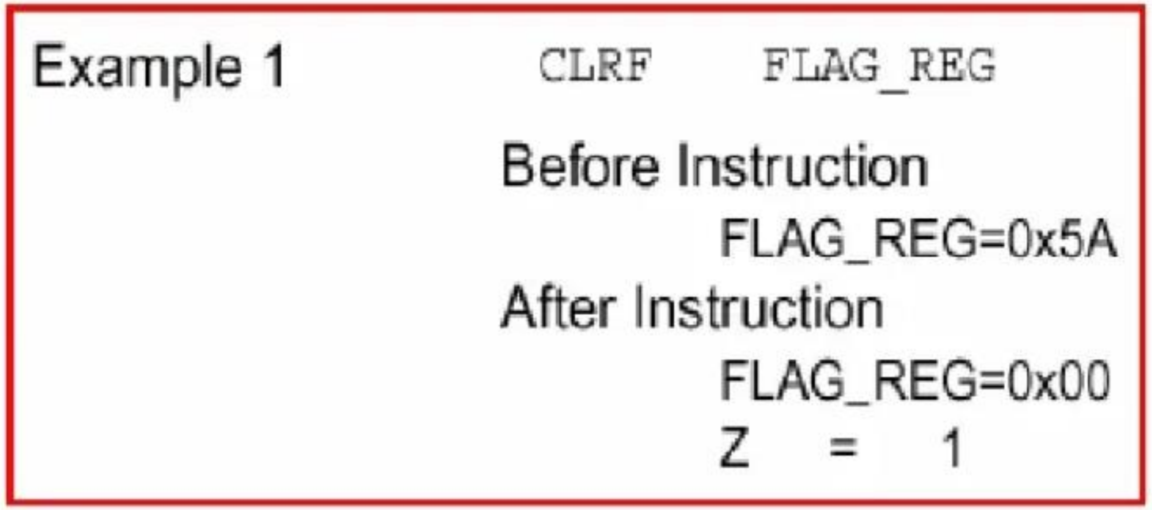
```
Example 2        ANDWF   FSR, 0

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



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`

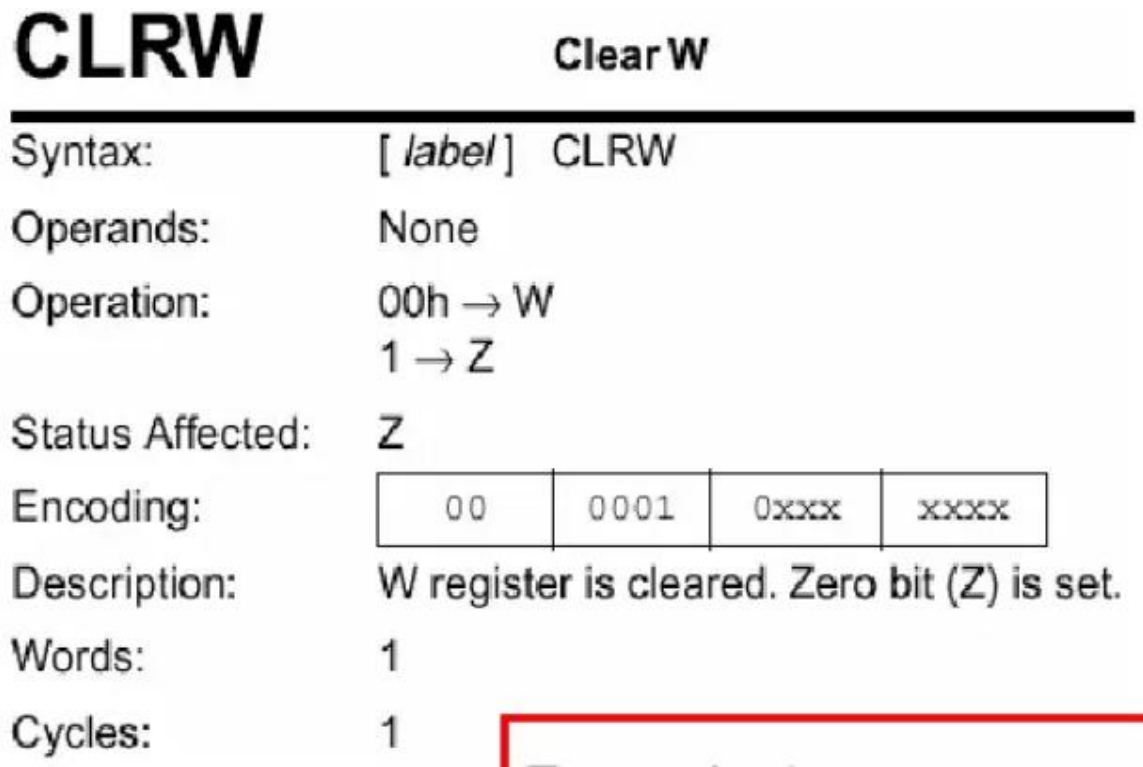
Before Instruction

FLAG_REG=0x5A

After Instruction

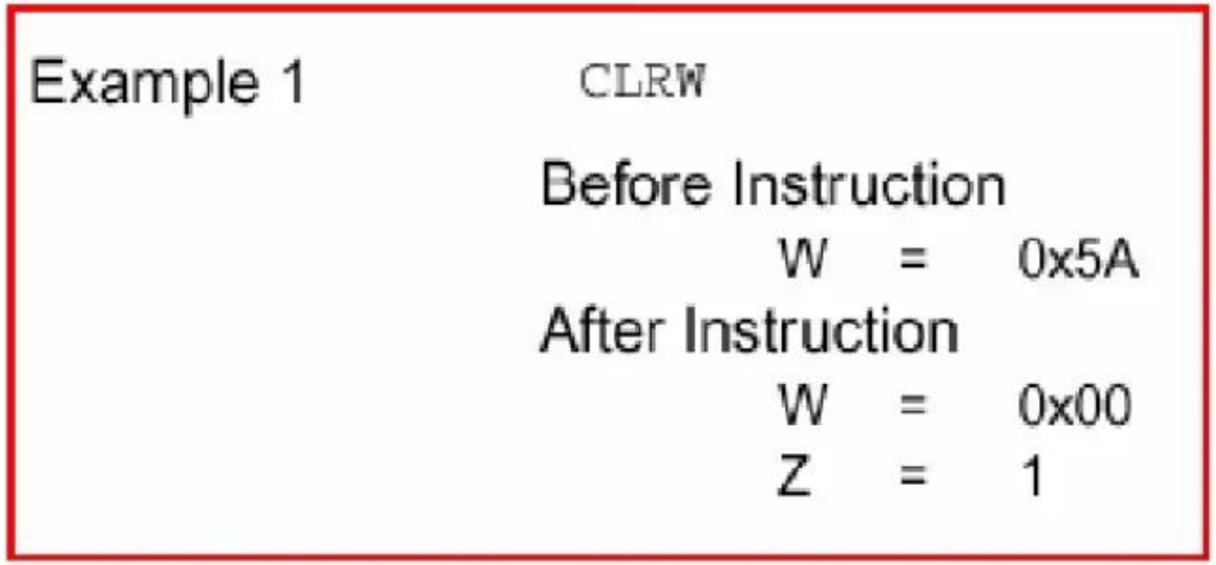
FLAG_REG=0x00

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`

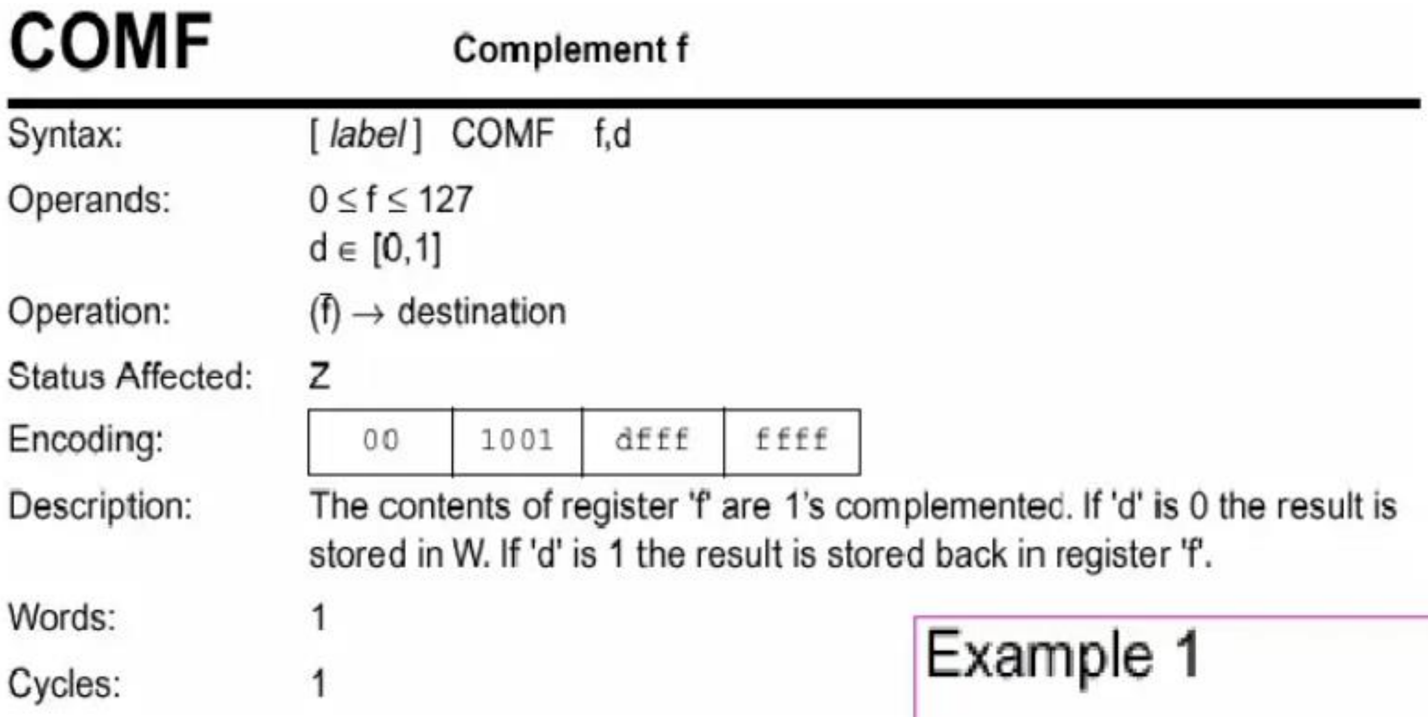
Before Instruction

W = 0x5A

After Instruction

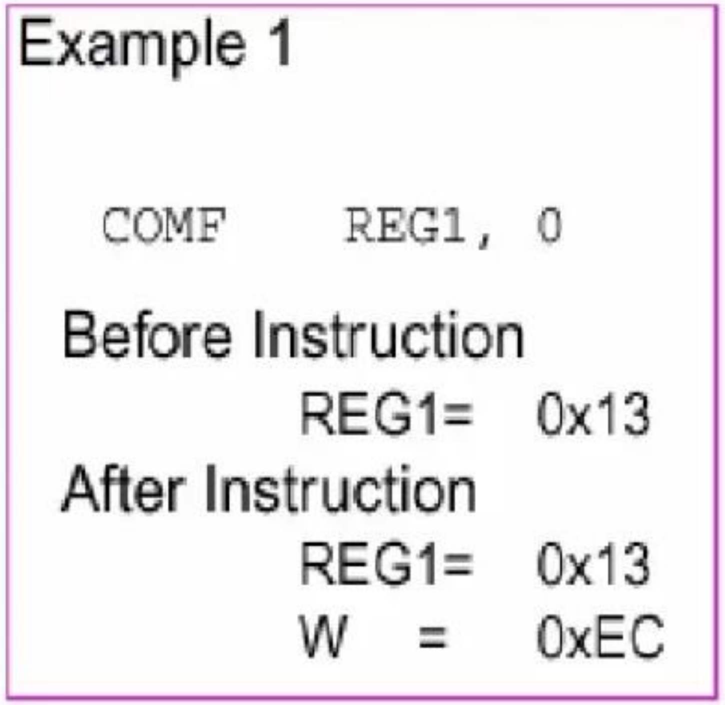
W = 0x00

Z = 1



COMF — Complement f

Syntax:	[*label*] COMF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
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
```

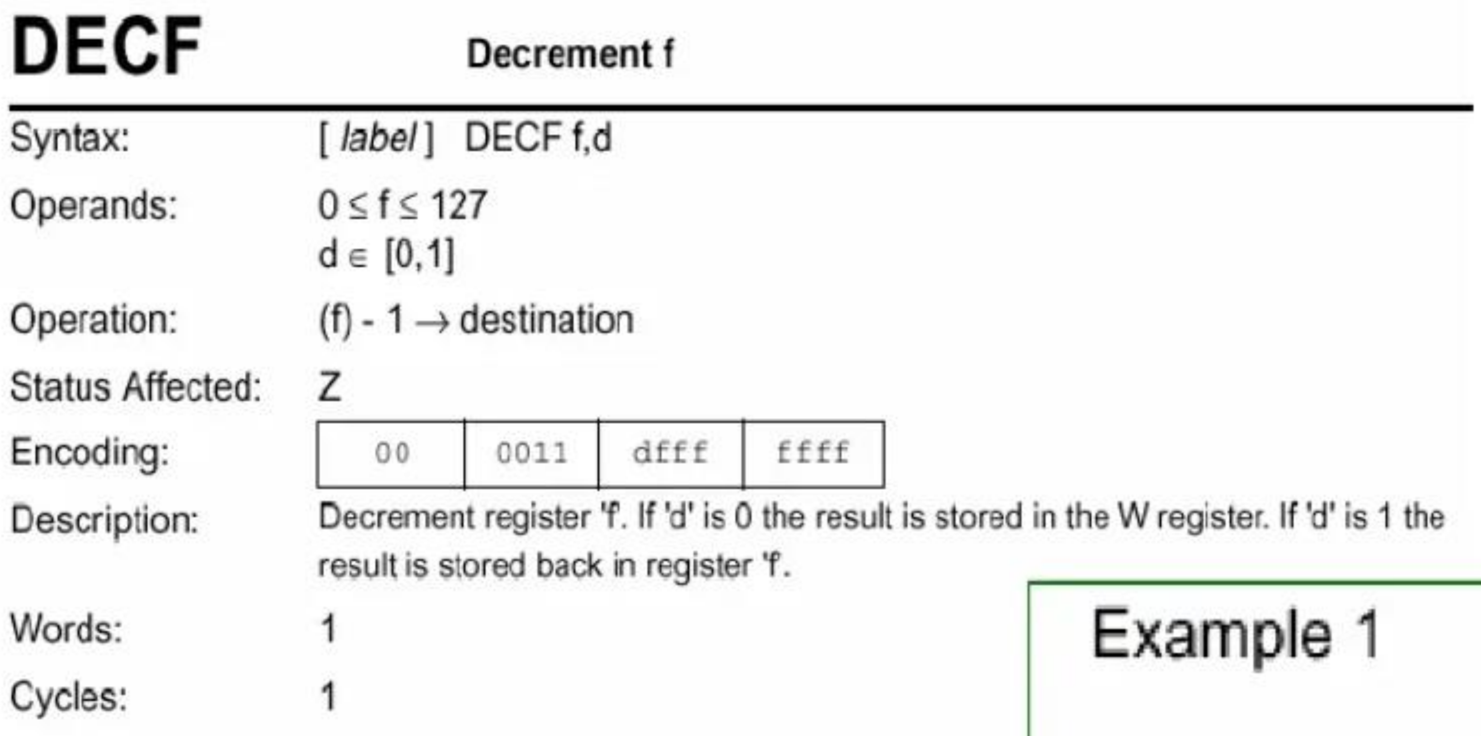
Before Instruction

REG1= 0x13

After Instruction

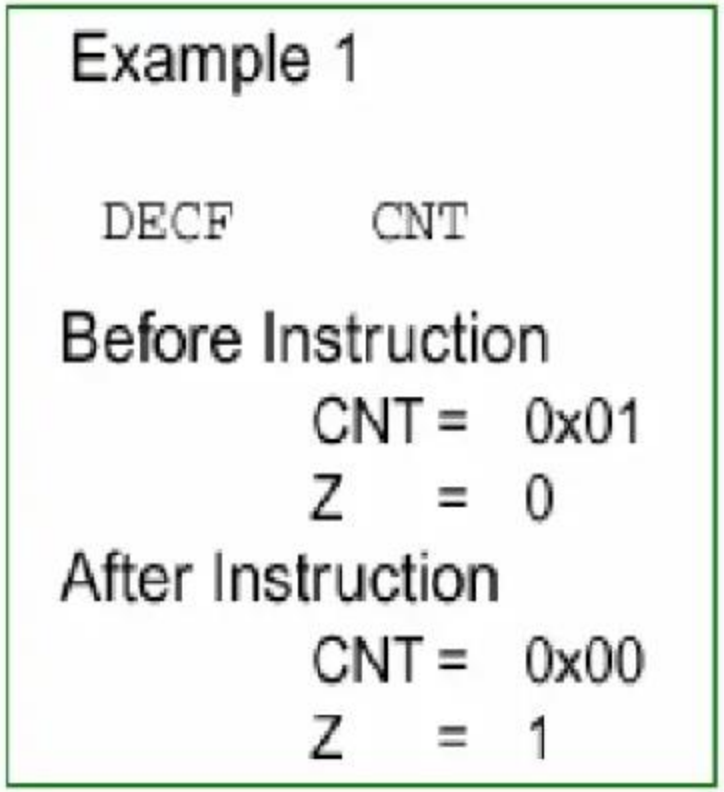
REG1= 0x13

W = 0xEC



DECF Decrement f

Syntax:	[*label*] DECF f,d
Operands:	$0 \leq f \leq 127$ $d \in [0,1]$
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:	$(f) - 1 \rightarrow$ 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
```