

Jeu d'Instructions Des Microcontrôleurs PIC

9.0 INSTRUCTION SET SUMMARY

Each PIC16CXX instruction is a 14-bit word divided into an OPCODE which specifies the instruction type and one or more operands which further specify the operation of the instruction. The PIC16CXX instruction set summary in Table 9-2 lists **byte-oriented**, **bit-oriented**, and **literal and control** operations. Table 9-1 shows the opcode field descriptions.

For **byte-oriented** instructions, 'f' represents a file register designator and 'd' represents a destination designator. The file register designator specifies which file register is to be used by the instruction.

The destination designator specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the W register. If 'd' is one, the result is placed in the file register specified in the instruction.

For **bit-oriented** instructions, 'b' represents a bit field designator which selects the number of the bit affected by the operation, while 'f' represents the number of the file in which the bit is located.

For **literal and control** operations, 'k' represents an eight or eleven bit constant or literal value.

TABLE 9-1 OPCODE FIELD DESCRIPTIONS

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
label	Label name
TOS	Top of Stack
PC	Program Counter
PCLATH	Program Counter High Latch
GIE	Global Interrupt Enable bit
WDT	Watchdog Timer/Counter
TO	Time-out bit
PD	Power-down bit
dest	Destination either the W register or the specified register file location
[]	Options
()	Contents
\rightarrow	Assigned to
<>	Register bit field
E	In the set of
italics	User defined term (font is courier)

The instruction set is highly orthogonal and is grouped into three basic categories:

- Byte-oriented operations
- Bit-oriented operations
- Literal and control operations

All instructions are executed within one single instruction cycle, unless a conditional test is true or the program counter is changed as a result of an instruction. In this case, the execution takes two instruction cycles with the second cycle executed as a NOP. One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μ s. If a conditional test is true or the program counter is changed as a result of an instruction, the instruction execution time is 2 μ s.

Table 9-2 lists the instructions recognized by the MPASM assembler.

Figure 9-1 shows the general formats that the instructions can have.

Note: To maintain upward compatibility with future PIC16CXX products, <u>do not use</u> the OPTION and TRIS instructions.

All examples use the following format to represent a hexadecimal number:

0xhh

where h signifies a hexadecimal digit.

FIGURE 9-1: GENERAL FORMAT FOR INSTRUCTIONS

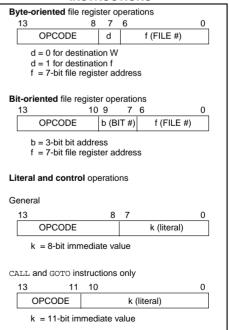


TABLE 9-2 PIC16FXX INSTRUCTION	SET
--------------------------------	-----

Mnemo		Description	Cycles		14-Bit	Opcode	e	Status	Notes
Operar	nds			MSb			LSb	Affected	
		BYTE-ORIENTED FILE REGI	STER OPE	RATIC	ONS				
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	С	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	С	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 REGIS	TER OPER	RATIO	NS				
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 CONTRO	L OPERAT	IONS					
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	TO,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	TO,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	

Note 1: When an I/O register is modified as a function of itself (e.g., MOVF PORTB, 1), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned to the Timer0 Module.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

9.1 Instruction Descriptions

ADDLW	Add Lite	ral and \	N	
Syntax:	[<i>label</i>] Al	DDLW	k	
Operands:	$0 \le k \le 2\xi$	55		
Operation:	(W) + k –	→ (W)		
Status Affected:	C, DC, Z			
Encoding:	11	111x	kkkk	kkkk
Description:	The conter added to the result is play	ne eight b	it literal 'k'	and the
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read literal 'k'	Process data	Write to W
Example:	ADDLW	0x15		
	Before In	W =	0x10	
		W =	0x25	

ADDWF	Add W a	nd f		
Syntax:	[<i>label</i>] A	DDWF	f,d	
Operands:	$\begin{array}{l} 0 \leq f \leq 12 \\ d \in \left[0,1\right] \end{array}$	27		
Operation:	(W) + (f)	ightarrow (desti	nation)	
Status Affected:	C, DC, Z			
Encoding:	00	0111	dfff	ffff
Description:	Add the co register 'f'. in the W re stored bac	If 'd' is 0 egister. If '	the result i d' is 1 the	is stored
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process data	Write to destination
Example	ADDWF	FSR,	0	
	Before In	struction	1	
		W =	0x17	
	After Inst	FSR = ruction	0xC2	
		W =	0xD9	
		FSR =	0xC2	

	AND Lite		W	
Syntax:	[<i>label</i>] A	NDLW	k	
Operands:	$0 \le k \le 2$	55		
Operation:	(W) .AND	D. (k) \rightarrow (W)	
Status Affected:	Z			
Encoding:	11	1001	kkkk	kkkk
Description:	The conte AND'ed wi result is pl	ith the eig	ht bit litera	l 'k'. The
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read literal "k"	Process data	Write to W
Example	ANDLW	0x5F		
	Before In			
	After Inst	W =	0xA3	
		W =	0x03	
ANDWF Syntax:	AND W v [label] A		f,d	
Operands:	$0 \le f \le 12$	27		
	$d \in [0,1]$			
Operation:	d ∈ [0,1] (W) .ANE	D. (f) \rightarrow (c	destinatio	n)
Operation: Status Affected:		D. (f) \rightarrow (d	destinatio	n)
•	(W) .ANE	$0. (f) \rightarrow (d)$	destinatio	n)
Status Affected:	(W) .ANE Z AND the V is 0 the res ter. If 'd' is	0101 V register sult is stor 1 the res	dfff with regist red in the V	ffff er 'f'. If 'd' <i>N</i> regis-
Status Affected: Encoding: Description:	(W) .ANE Z 00 AND the V is 0 the re- ter. If 'd' is register 'f'.	0101 V register sult is stor 1 the res	dfff with regist red in the V	ffff er 'f'. If 'd' <i>N</i> regis-
Status Affected: Encoding: Description: Words:	(W) .ANE Z AND the V is 0 the re- ter. If 'd' is register 'f'. 1	0101 V register sult is stor 1 the res	dfff with regist red in the V	ffff er 'f'. If 'd' <i>N</i> regis-
Status Affected: Encoding: Description: Words: Cycles:	(W) .ANE Z AND the V is 0 the re- ter. If 'd' is register 'f'. 1	0101 V register sult is stor 1 the rest	dfff with regist red in the V ult is store	ffff er 'f'. If 'd' <i>N</i> regis- d back in
Status Affected: Encoding: Description: Words:	(W) .ANE Z AND the V is 0 the re- ter. If 'd' is register 'f'. 1 1 Q1	0101 V register sult is stor 1 the resu	dfff with regist red in the V ult is store	ffff er 'f'. If 'd' <i>W</i> regis- d back in Q4
Status Affected: Encoding: Description: Words: Cycles:	(W) .ANE Z AND the V is 0 the re- ter. If 'd' is register 'f'. 1	0101 V register sult is stor 1 the rest	dfff with regist red in the V ult is store	ffff er 'f. If 'd' W regis- d back in Q4 Write to
Status Affected: Encoding: Description: Words: Cycles:	(W) .ANE Z AND the V is 0 the re- ter. If 'd' is register 'f'. 1 1 Q1	0101 V register sult is stor 1 the resu Q2 Read register T	dfff with regist red in the V ult is store Q3 Process	ffff er 'f'. If 'd' W regis- d back in Q4
Status Affected: Encoding: Description: Words: Cycles: Q Cycle Activity:	(W) .ANE Z 00 AND the V is 0 the re- ter. If 'd' is register 'f'. 1 1 Q1 Decode	0101 V register sult is stor 1 the resu Q2 Read register "f" FSR, struction	dfff with regist red in the V ult is store Q3 Process data	ffff er 'f'. If 'd' W regis- d back in Q4 Write to
Status Affected: Encoding: Description: Words: Cycles: Q Cycle Activity:	(W) .ANE Z 00 AND the V is 0 the re- ter. If 'd' is register 'f'. 1 1 Q1 Decode ANDWF Before In	0101 V register sult is stor 1 the resu Q2 Read register T FSR, struction W =	dfff with regist red in the V ult is store Q3 Process data 1 0x17	ffff er 'f'. If 'd' W regis- d back in Q4 Write to
Status Affected: Encoding: Description: Words: Cycles: Q Cycle Activity:	(W) .ANE Z 00 AND the V is 0 the re- ter. If 'd' is register 'f'. 1 1 Q1 Decode ANDWF Before In	0101 V register sult is stor 1 the resu Q2 Read register T FSR, struction W = FSR =	dfff with regist red in the V ult is store Q3 Process data	ffff er 'f'. If 'd' W regis- d back in Q4 Write to

BCF	Bit Clear	f			BTFSC
Syntax:	[<i>label</i>] B0	CF f,b			Syntax:
Operands:	$0 \le f \le 12$ $0 \le b \le 7$				Operand
Operation:	$0 \rightarrow (f < b)$	>)			Operatio
Status Affected:	None				Status A
Encoding:	01	00bb	bfff	ffff	Encodin
Description:	Bit 'b' in re	gister 'f' is	s cleared.		Descript
Words:	1				
Cycles:	1				
Q Cycle Activity:	Q1	Q2	Q3	Q4	
	Decode	Read register 'f'	Process data	Write register 'f'	Words: Cycles:
Example	BCF	FLAG_	REG, 7		Q Cycle
	Before In After Inst	FLAG_RE	EG = 0xC7		
		FLAG_RE	EG = 0x47		

BTFSC	Bit Test,	Skip if Cl	ear	
Syntax:	[<i>label</i>] BT	FSC f,b		
Operands:	$0 \le f \le 12$ $0 \le b \le 7$?7		
Operation:	skip if (f<	b>) = 0		
Status Affected:	None	.,		
Encoding:	01	10bb	bfff	ffff
Description:	instruction If bit 'b', in instruction	register 'f' is is executed register 'f', is discarde nstead, ma	d. is '0' then i ed, and a N	the next IOP is
Words:	1			
Cycles:	1(2)			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process data	No-Operat ion
If Skip:	(2nd Cyc	le)		
	Q1	Q2	Q3	Q4
	No-Operat ion	No-Operati on	No-Opera tion	No-Operat ion
Example	HERE FALSE TRUE	BTFSC GOTO •	FLAG,1 PROCESS_	_CODE
	After Inst	PC = a ruction if FLAG<1> PC = a if FLAG<1>	= 0, address T	

BSF	Bit Set f			
Syntax:	[<i>label</i>] BS	SF f,b		
Operands:	$0 \le f \le 12$ $0 \le b \le 7$.7		
Operation:	$1 \rightarrow (f < b;$	>)		
Status Affected:	None			
Encoding:	01	01bb	bfff	ffff
Description:	Bit 'b' in re	gister 'f' is	s set.	,
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process data	Write register 'f'
Example	BSF	FLAG_F	REG, 7	
	Before In		EG = 0x0A	Ą

BTFSS	Bit Test f, Skip if Set		CALL	Call Sul	proutine		
Syntax:	[<i>label</i>] BTFSS f,b		Syntax:	[label]	CALL I	<	
Operands:	$0 \le f \le 127$		Operands:	$0 \le k \le 2$	047		
	0 ≤ b < 7		Operation:	(PC)+ 1-	→ TOS,		
Operation:	skip if (f) = 1			$k \rightarrow PC$,	50 / 0	
Status Affected:	None				1<4:3>) -	\rightarrow PC<12	:11>
Encoding:	01 11bb bfff	ffff	Status Affected:	None	1	1	
Description:	If bit 'b' in register 'f' is '0' then th instruction is executed.	ne next	Encoding:	10	0kkk	kkkk	kkkk
Words:	If bit 'b' is '1', then the next instru discarded and a NOP is execute instead, making this a 2TCY inst 1	ed	Description:	(PC+1) is eleven bit into PC b the PC ar	pushed or immediate ts <10:0>.	st, return ac nto the stac address i The upper rom PCLA ction.	ck. The s loaded bits of
Cycles:	1(2)		Words:	1			
Q Cycle Activity:	Q1 Q2 Q3	Q4	Cycles:	2			
	Decode Read Process register 'f' data	No-Operat ion	Q Cycle Activity:	Q1	Q2	Q3	Q4
If Skip:	(2nd Cycle)		1st Cycle	Decode	Read literal 'k',	Process data	Write to PC
	Q1 Q2 Q3	Q4			Push PC to Stack		
	No-Operati No-Operati No-Operati on tion	No-Operat ion	2nd Cycle	No-Opera tion	No-Opera tion	No-Opera tion	No-Operat ion
Example	HERE BTFSC FLAG,1 FALSE GOTO PROCESS_	CODE	Example	HERE	CALL	THERE	
	TRUE •			Before I	nstruction		
	•			After Ins		Address HE	RE
	Before Instruction					ddress TH	IERE
	PC = address H	ERE			TOS = A	Address HE	RE+1
	After Instruction						
	if FLAG<1> = 0, PC = address FA	LSE					
	if $FLAG<1>=1$,						
	PC = address TR	UE					

CLRF	Clear f			
Syntax:	[<i>label</i>] C	LRF f		
Operands:	$0 \le f \le 12$	7		
Operation:	$\begin{array}{l} 00h \rightarrow (f) \\ 1 \rightarrow Z \end{array}$			
Status Affected:	Z			
Encoding:	00	0001	lfff	ffff
Description:	The conter and the Z		ster 'f' are	cleared
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process data	Write register 'f'
Example	CLRF	FLAG	_REG	
	Before In			
	After Inst	FLAG_RE	EG =	0x5A
		FLAG RE	- 	0x00
		Ζ	=	1

CLRW	Clear W
Syntax:	[label] CLRW
Operands:	None
Operation:	$\begin{array}{l} 00h \rightarrow (W) \\ 1 \rightarrow Z \end{array}$
Status Affected:	Z
Encoding:	00 0001 0xxx xxxx
Description:	W register is cleared. Zero bit (Z) is set.
Words:	1
Cycles:	1
Q Cycle Activity:	Q1 Q2 Q3 Q4
	Decode No-Opera Process Write to data W
Example	CLRW
	Before Instruction
	W = 0x5A After Instruction
	W = 0x00
	Z = 1
CLRWDT	Clear Watchdog Timer
Curatava	
Syntax:	[label] CLRWDT
Operands:	[<i>label</i>] CLRWD1 None
5	None $00h \rightarrow WDT$
Operands:	None $00h \rightarrow WDT$ $0 \rightarrow WDT$ prescaler,
Operands:	None $00h \rightarrow WDT$
Operands:	None $00h \rightarrow WDT$ $0 \rightarrow WDT$ prescaler, $1 \rightarrow \overline{TO}$
Operands: Operation:	None $00h \rightarrow WDT$ $0 \rightarrow WDT$ prescaler, $1 \rightarrow \overline{TO}$ $1 \rightarrow \overline{PD}$
Operands: Operation: Status Affected:	None $00h \rightarrow WDT$ $0 \rightarrow WDT prescaler,$ $1 \rightarrow TO$ $1 \rightarrow PD$ TO, PD CLRWDT instruction resets the Watch- dog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are
Operands: Operation: Status Affected: Encoding:	None $00h \rightarrow WDT$ $0 \rightarrow WDT prescaler,$ $1 \rightarrow T\overline{O}$ $1 \rightarrow \overline{PD}$ $\overline{TO}, \overline{PD}$ $\boxed{00 0000 0110 0100}$ CLRWDT instruction resets the Watch- dog Timer. It also resets the prescaler
Operands: Operation: Status Affected: Encoding: Description:	None $00h \rightarrow WDT$ $0 \rightarrow WDT prescaler,$ $1 \rightarrow TO$ $1 \rightarrow PD$ TO, PD CLRWDT instruction resets the Watch- dog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are set.
Operands: Operation: Status Affected: Encoding: Description: Words:	None $\begin{array}{l} 00h \rightarrow WDT \\ 0 \rightarrow WDT prescaler, \\ 1 \rightarrow TO \\ 1 \rightarrow PD \\ \hline TO, PD \\ \hline \hline O \\ CLRWDT instruction resets the Watch-dog Timer. It also resets the Prescaler of the WDT. Status bits TO and PD are set. \\ 1 \end{array}$
Operands: Operation: Status Affected: Encoding: Description: Words: Cycles:	None $\begin{array}{c} 00h \rightarrow WDT \\ 0 \rightarrow WDT \ prescaler, \\ 1 \rightarrow TO \\ 1 \rightarrow PD \\ \hline TO, \ PD \\ \hline \hline 00 0000 0110 0100 \\ \hline CLRWDT \ instruction \ resets the \ Watch-dog \ Timer. It also \ resets the \ Prescaler \ of the \ WDT. \ Status \ bits \ TO \ and \ PD \ are \ set. \\ \hline 1 \\ 1 \\ \hline Q1 Q2 Q3 Q4 \\ \hline \hline Decode No-Opera Process Clear \\ \end{array}$
Operands: Operation: Status Affected: Encoding: Description: Words: Cycles:	None $00h \rightarrow WDT$ $0 \rightarrow WDT prescaler,$ $1 \rightarrow T\overline{O}$ $1 \rightarrow P\overline{D}$ $T\overline{O}, P\overline{D}$ CLRWDT instruction resets the Watch- dog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are set. 1 1 Q1 Q2 Q3 Q4
Operands: Operation: Status Affected: Encoding: Description: Words: Cycles:	None $\begin{array}{c} \text{Ooh} \rightarrow \text{WDT} \\ \text{O} \rightarrow \text{WDT prescaler,} \\ 1 \rightarrow \text{TO} \\ 1 \rightarrow \text{PD} \\ \hline \text{TO, PD} \\ \hline \hline \text{O0} & 0000 & 0110 & 0100 \\ \hline \\ \hline \text{CLRWDT instruction resets the Watch-dog Timer. It also resets the Prescaler of the WDT. Status bits TO and PD are set.} \\ 1 \\ 1 \\ \hline \\ \hline \text{Q1} & \text{Q2} & \text{Q3} & \text{Q4} \\ \hline \hline \\ \hline \\ \hline \text{Decode} & \text{No-Opera} & \text{Process} & \text{Clear WDT} \\ \hline \\ \text{CLRWDT} \\ \hline \end{array}$
Operands: Operation: Status Affected: Encoding: Description: Words: Cycles: Q Cycle Activity:	None $\begin{array}{r} \text{None} \\ 00h \rightarrow \text{WDT} \\ 0 \rightarrow \text{WDT prescaler,} \\ 1 \rightarrow \overline{\text{TO}} \\ 1 \rightarrow \overline{\text{PD}} \\ \hline \overline{\text{TO, PD}} \\ \hline \hline 00 & 0000 & 0110 & 0100 \\ \hline \\ \hline CLRWDT instruction resets the Watch-dog Timer. It also resets the Watch-dog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are set. \\ 1 \\ 1 \\ \hline \\ Q1 & Q2 & Q3 & Q4 \\ \hline \hline \\ \hline \\ Q1 & Q2 & Q3 & Q4 \\ \hline \hline \\ \hline \\ \hline \\ CLRWDT \\ \hline \\ \\ \hline \\ CLRWDT \\ \hline \\ \\ \\ \hline \\ \\ \hline \\ \\ \hline \\ \\ \\ \hline \\ \\ \\ \hline \\ \\ \\ \hline \\ \hline \\ \\ \hline \\ \hline \\ \hline \\ \\ \hline \\ \\ \hline \\ \hline \\ \hline \\ \\ \hline \\ \hline \\ \hline \\ \\ \hline \\ \hline \\ \hline \\ \hline \\ \\ \hline \\ \hline \\ \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \\ \hline \\ \hline \\ \\ \hline \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline \hline \\ \hline \hline \\ \hline \\ \hline \\ \hline \\ \hline \hline \\ \hline \\ \hline \hline \\ \hline \\ \hline \\ \hline \hline \\ \hline \hline \\ \hline \\ \hline \\ \hline \\ \hline \hline \\ \hline \\ \hline \\ \hline \\ \hline \hline \\ \hline \\ \hline \hline \\ \hline \\ \hline \\ \hline \\ \hline \hline \\ \hline \\ \hline \\ \hline \hline \\ \hline \hline \\ \hline \\ \hline \\ \hline \\ \hline \\ \hline $
Operands: Operation: Status Affected: Encoding: Description: Words: Cycles: Q Cycle Activity:	None $\begin{array}{c} 00h \rightarrow WDT \\ 0 \rightarrow WDT prescaler, \\ 1 \rightarrow TO \\ 1 \rightarrow PD \\ \hline TO, PD \\ \hline 00 0000 0110 0100 \\ \hline CLRWDT instruction resets the Watch-dog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are set. \\ 1 \\ 1 \\ \hline Q1 Q2 Q3 Q4 \\ \hline \hline Decode No-Opera \\ tion Process \\ data Clear \\ WDT \\ Cutry DT \\ \hline CLRWDT \\ \hline Before Instruction \\ WDT counter = ? \\ After Instruction \\ \hline \end{array}$
Operands: Operation: Status Affected: Encoding: Description: Words: Cycles: Q Cycle Activity:	None $\begin{array}{c} 00h \rightarrow WDT\\ 0 \rightarrow WDT prescaler,\\ 1 \rightarrow TO\\ 1 \rightarrow PD\\ \hline TO, PD\\ \hline \hline 00 0000 0110 0100\\ \hline CLRWDT instruction resets the Watch-dog Timer. It also resets the Vatch-dog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are set.\\ 1\\ 1\\ \hline Q1 Q2 Q3 Q4\\ \hline \hline Decode No-Opera \\ tion Process \\ data WDT \\ Counter\\ \hline \\ CLRWDT\\ \hline \\ Before Instruction \\ WDT counter = ?\\ After Instruction \\ WDT counter = 0 x00\\ WDT prescaler = 0\\ \hline \end{array}$
Operands: Operation: Status Affected: Encoding: Description: Words: Cycles: Q Cycle Activity:	None $\begin{array}{r} 00h \rightarrow WDT \\ 0 \rightarrow WDT prescaler, \\ 1 \rightarrow TO \\ 1 \rightarrow PD \\ \hline TO, PD \\ \hline \hline 0 \\ 000 \\ 0110 \\ 0100 \\ \hline \end{array}$ CLRWDT instruction resets the Watch-dog Timer. It also resets the prescaler of the WDT. Status bits TO and PD are set. 1 1 1 Q1 Q2 Q3 Q4 \hline Decode No-Opera Process Clear WDT counter CLRWDT Before Instruction WDT counter = ? After Instruction WDT counter = 0x00

COMF	Compler	nent f		
Syntax:	[label]	COMF	f,d	
Operands:	$\begin{array}{l} 0 \leq f \leq 12 \\ d \in \ [0,1] \end{array}$	27		
Operation:	$(\bar{f}) \to (de$	stination)	
Status Affected:	Z			
Encoding:	0 0	1001	dfff	ffff
Description:	The conte mented. If W. If 'd' is register 'f'.	1 the resu	ister 'f' are e result is Ilt is stored	comple- stored in back in
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process data	Write to destination
Example	COMF	סדר	£1,0	
Lixample				
	Before In	REG1	= 0x13	3
	After Inst			
		REG1 W	= 0x13 = 0xE0	
DECF	Decreme	ent f		
DECF Syntax:	Decreme [label] [
_		ECF f,d		
Syntax:	[<i>label</i>] □ 0 ≤ f ≤ 12	ECF f,d	ion)	
Syntax: Operands:	[<i>label</i>] □ 0 ≤ f ≤ 12 d ∈ [0,1]	ECF f,d	ion)	
Syntax: Operands: Operation:	$[label] \square 0 \le f \le 12$ d \in [0,1] (f) - 1 \rightarrow	ECF f,d	ion)	ffff
Syntax: Operands: Operation: Status Affected:	$[label] \square \\ 0 \le f \le 12 \\ d \in [0,1] \\ (f) - 1 \rightarrow \\ Z$	DECF f,d 7 (destinat 0011 tt register ored in th	dfff 'f'. If 'd' is (e W registe	0 the er. If 'd' is
Syntax: Operands: Operation: Status Affected: Encoding:	[<i>label</i>] C $0 \le f \le 12$ $d \in [0,1]$ (f) - 1 \rightarrow Z Decrement result is st	DECF f,d 7 (destinat 0011 tt register ored in th	dfff 'f'. If 'd' is (e W registe	0 the er. If 'd' is
Syntax: Operands: Operation: Status Affected: Encoding: Description:	[<i>label</i>] D $0 \le f \le 12$ $d \in [0,1]$ (f) - 1 \rightarrow Z Decrement result is standard the result is standard the	DECF f,d 7 (destinat 0011 tt register ored in th	dfff 'f'. If 'd' is (e W registe	0 the er. If 'd' is
Syntax: Operands: Operation: Status Affected: Encoding: Description: Words:	[<i>label</i>] C $0 \le f \le 12$ $d \in [0,1]$ (f) - 1 \rightarrow Z 00 Decrement result is st 1 the result	DECF f,d 7 (destinat 0011 tt register ored in th	dfff 'f'. If 'd' is (e W registe	0 the er. If 'd' is
Syntax: Operands: Operation: Status Affected: Encoding: Description: Words: Cycles:	[<i>label</i>] C $0 \le f \le 12$ $d \in [0,1]$ (f) - 1 \rightarrow Z Decrement result is st 1 the result 1	DECF f,d 7 (destinat 0011 t register ored in th It is stored	dfff 'f'. If 'd' is (e W registr d back in re	0 the er. If 'd' is egister 'f'.
Syntax: Operands: Operation: Status Affected: Encoding: Description: Words: Cycles:	$ \begin{bmatrix} label \end{bmatrix} \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $	QECF f,d (destinat 0011 t register ored in th t is stored Q2 Read register	dfff 'f'. If 'd' is o e W registe d back in re Q3 Process	Q4 Write to
Syntax: Operands: Operation: Status Affected: Encoding: Description: Words: Cycles: Q Cycle Activity:	$ \begin{bmatrix} label \end{bmatrix} \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ $	QECF f,d (destinat 0011 tregister ored in th tt is stored Q2 Read register r	dfff 'f'. If 'd' is t e W regist d back in n Q3 Process data	Q4 Write to destination

DECFSZ	Decreme	ent f, Skip	o if O		
Syntax:	[label]	[label] DECFSZ f,d			
Operands:	$\begin{array}{l} 0 \leq f \leq 12 \\ d \in \ [0,1] \end{array}$	7			
Operation:	(f) - 1 \rightarrow skip if res		on);		
Status Affected:	None				
Encoding:	0.0	1011	dfff	ffff	
Description:	The contents of register 'f' are decre- mented. 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 1, the next instruction, is executed. If the result is 0, then a NOP is executed instead making it a 2Tcy instruc- tion.				
Words:	1				
Cycles:	1(2)				
Q Cycle Activity:	Q1	Q2	Q3	Q4	
	Decode	Read register 'f'	Process data	Write to destination	
If Skip:	(2nd Cyc	le)			
	Q1	Q2	Q3	Q4	
	No-Operat ion	No-Opera tion	No-Operat ion	No-Operati on	
Example	HERE	DECFS GOTO JE • •	Z CNT, LOOF		
	Before In PC After Inst CNT if CNT PC if CNT PC	= addr ruction = CNT = 0, = addr ≠ 0,	ess here - 1 ess conti ess here-		

GOTO	Unconditional Bra	nch		INCF	Increment f
Syntax:	[label] GOTO k			Syntax:	[label] INCF f,d
Operands:	$0 \leq k \leq 2047$			Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	$k \rightarrow PC<10:0>$ PCLATH<4:3> $\rightarrow P$	C<12:11	>	Operation:	(f) + 1 \rightarrow (destination)
Status Affected:	None			Status Affected:	Z
Encoding:	10 1kkk	kkkk	kkkk	Encoding:	00 1010 dfff ffff
Description:	GOTO is an uncondition eleven bit immediate v into PC bits <10:0>. TI PC are loaded from P GOTO is a two cycle in	value is loa he upper l CLATH<4	aded bits of	Description:	The contents of register 'f' are incre- mented. 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			Words:	1
Cycles:	2			Cycles:	1
Q Cycle Activity:	Q1 Q2	Q3	Q4	Q Cycle Activity:	Q1 Q2 Q3 Q4
1st Cycle	Decode Read literal 'k'	Process data	Write to PC		Decode Read register data Vite to destination
2nd Cycle	No-Operat I ion	No-Opera tion	No-Operat ion	Example	INCF CNT, 1
Example	GOTO THERE				Before Instruction CNT = 0xFF
	After Instruction PC = A	ddress I	HERE		Z = 0 After Instruction $CNT = 0x00$ $Z = 1$

INCFSZ	Increment f, Skip if 0	IORLW	Inclusive OR Literal with W
Syntax:	[label] INCFSZ f,d	Syntax:	[<i>label</i>] IORLW k
Operands:	$0 \le f \le 127$	Operands:	$0 \le k \le 255$
	d ∈ [0,1]	Operation:	(W) .OR. $k \rightarrow$ (W)
Operation:	(f) + 1 \rightarrow (destination), skip if result = 0	Status Affected:	Z
Status Affected:	None	Encoding:	11 1000 kkkk kkkk
Encoding:	00 1111 dfff ffff	Description:	The contents of the W register is
Description:	The contents of register 'f' are incre-		OR'ed with the eight bit literal 'k'. The result is placed in the W register.
Description.	mented. If 'd' is 0 the result is placed in	Words:	1
	the W register. If 'd' is 1 the result is placed back in register 'f'.	Cycles:	1
	If the result is 1, the next instruction is executed. If the result is 0, a NOP is exe- cuted instead making it a 2TCY instruc- tion.	Q Cycle Activity:	Q1 Q2 Q3 Q4
Words:	1		Decode Read Process Write to literal 'k' data W
Cycles:	1(2)		
Q Cycle Activity:	Q1 Q2 Q3 Q4	Example	IORLW 0x35
	Decode Read register 'f' Process Write to destination		Before Instruction W = 0x9A After Instruction
If Skip:	(2nd Cycle)		W = 0xBF
	Q1 Q2 Q3 Q4		Z = 1
	No-Operat ion No-Opera No-Operati tion tion on		
Example	HERE INCFSZ CNT, 1 GOTO LOOP CONTINUE • • Before Instruction PC = address HERE After Instruction CNT = CNT + 1 if CNT = 0,		
	PC = address CONTINUE if $CNT \neq 0$, PC = address HERE +1		

IORWF	Inclusive OR W with f			
Syntax:	[label]	IORWF	f,d	
Operands:	$\begin{array}{l} 0 \leq f \leq 12 \\ d \in \left[0,1\right] \end{array}$	27		
Operation:	(W) .OR.	(f) \rightarrow (de	estination)
Status Affected:	Z			
Encoding:	00	0100	dfff	ffff
Description:	Inclusive C ter 'f'. If 'd' W register back in reg	is 0 the re . If 'd' is 1	sult is place	ced in the
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process data	Write to destination
Example	IORWF		RESULT,	0
		struction RESULT W		
	After Inst	ruction	0X0	
		RESULT W		-
		Z	= 0x93 = 1	D

MOVLW	Move Literal to W
Syntax:	[<i>label</i>] 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
Q Cycle Activity:	Q1 Q2 Q3 Q4
	Decode Read literal 'k' Process Write to data W
Example	MOVLW 0x5A After Instruction
	W = 0x5A

MOVF	Move f				
Syntax:	[label]	[<i>label</i>] MOVF f,d			
Operands:	$\begin{array}{l} 0 \leq f \leq 12 \\ d \in \left[0,1\right] \end{array}$.7			
Operation:	(f) \rightarrow (de	stination)		
Status Affected:	Z				
Encoding:	00	1000	dfff	ffff	
Description:	The contendestination of d. If $d =$ d = 1, the itself. $d = 1$ ter since s	n dependa 0, destina destinatio I is useful	ant upon th ition is W r n is file reg to test a f	ne status egister. If gister f ile regis-	
Words:	1				
Cycles:	1				
Q Cycle Activity:	Q1	Q2	Q3	Q4	
	Decode	Read register 'f'	Process data	Write to destination	
Example		ruction	0 le in FSR I	register	

MOVWF	Move W	to f		
Syntax:	[label]	MOVW	= f	
Operands:	$0 \le f \le 12$	27		
Operation:	$(W) \rightarrow (f)$			
Status Affected:	None			
Encoding:	00	0000	lfff	ffff
Description:	Move data 'f'.	from W r	egister to	register
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process data	Write register 'f'
Example	MOVWF	OPTIC	DN_REG	·,
	Before In			_
		OPTION W	= 0xFl = 0x4f	
	After Inst	ruction	0,11	
		OPTION	•••••	
		W	= 0x4	

NOP	No Operation			
Syntax:	[label]	NOP		
Operands:	None			
Operation:	No opera	ition		
Status Affected:	None			
Encoding:	00	0000	0xx0	0000
Description:	No operat	ion.		
Words:	1			
Cycles:	1			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	No-Opera tion	No-Opera tion	No-Operat ion
Example	NOP			

RETFIE	Return fi	om Inter	rupt	
Syntax:	[label]	RETFIE		
Operands:	None			
Operation:	$\begin{array}{l} TOS \rightarrow F \\ 1 \rightarrow GIE \end{array}$	PC,		
Status Affected:	None			
Encoding:	00	0000	0000	1001
	and Top of PC. Interru Global Inte (INTCON< instruction	upts are er errupt Ena (7>). This i	habled by s ble bit, GIE	setting
Words:	1			
Cycles:	2			
Q Cycle Activity:	Q1	Q2	Q3	Q4
1st Cycle	Decode	No-Opera tion	Set the GIE bit	Pop from the Stack
2nd Cycle	No-Operat ion	No-Opera tion	No-Opera tion	No-Operat ion
Example	RETFIE			

After Interrupt	
PC =	TOS
GIE =	1

OPTION	Load Option Register
Syntax:	[label] OPTION
Operands:	None
Operation:	$(W) \to OPTION$
Status Affected:	None
Encoding:	00 0000 0110 0010
Words: Cycles: Example	The contents of the W register are loaded in the OPTION register. This instruction is supported for code com- patibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it. 1
	To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

RETLW	Return with Literal in W			
Syntax:	[label]	RETLW	k	
Operands:	$0 \le k \le 255$			
Operation:	$\begin{array}{l} k \rightarrow (W); \\ TOS \rightarrow PC \end{array}$			
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 from the top of the stack (the return address). This is a two cycle instruction.			
Words:	1			
Cycles:	2			
Q Cycle Activity:	Q1	Q2	Q3	Q4
1st Cycle	Decode	Read literal 'k'	No-Opera tion	Write to W, Pop from the Stack
2nd Cycle	No-Operat ion	No-Opera tion	No-Opera tion	No-Operat ion
Example	CALL TABLE ;W contains table ;offset value ;W now has table value ADDWF PC ;W = offset RETLW k1 ;Begin table RETLW k2 ;			
TABLE				
	• RETLW kn ; End of table			
	Before Instruction W = 0x07			
	After Inst		value of k8	3

RETURN	Return from Subroutine			
Syntax:	[label] RETURN			
Operands:	None			
Operation:	$TOS\toPC$			
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			
Q Cycle Activity:	Q1	Q2	Q3	Q4
1st Cycle	Decode	No-Opera tion	No-Opera tion	Pop from the Stack
2nd Cycle	No-Operat ion	No-Opera tion	No-Opera tion	No-Opera tion
Example	RETURN After Inte	•	TOS	

RLF	Rotate Left f through Carry	RRF	Rotate Right f through Carry
Syntax:	[<i>label</i>] RLF f,d	Syntax:	[<i>label</i>] RRF f,d
Operands:	$0 \le f \le 127$ $d \in [0,1]$	Operands:	$\begin{array}{l} 0 \leq f \leq 127 \\ d \in \left[0,1\right] \end{array}$
Operation:	See description below	Operation:	See description below
Status Affected:	С	Status Affected:	С
Encoding:	00 1101 dfff ffff	Encoding:	00 1100 dfff ffff
Description:	The contents of register 'f' are rotated Description: 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'.		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'.
	C Register f		C Register f
Words:	1	Words:	1
Cycles:	1	Cycles:	1
Q Cycle Activity:	Q1 Q2 Q3 Q4	Q Cycle Activity:	Q1 Q2 Q3 Q4
	Decode Read Process Write to destination		Decode Read register data Write to destination
Example	RLF REG1,0	Example	RRF REG1,0
	Before Instruction REG1 = 1110 0110		Before Instruction REG1 = 1110 0110
	C = 0 After Instruction		C = 0 After Instruction
	REG1 = 1110 0110 W = 1100 1100 C = 1		REG1 = 1110 0110 W = 0111 0011 C = 0

SLEEP SUBLW Subtract W from Literal Syntax: [label] SUBLW k Syntax: [label] SLEEP Operands: $0 \le k \le 255$ Operands: None Operation: $k - (W) \rightarrow (W)$ Operation: $00h \rightarrow WDT$, $0 \rightarrow WDT$ prescaler, C, DC, Z Status Affected: $1 \rightarrow \overline{TO}$. Encoding: 11 110x kkkk kkkk $0 \rightarrow \overline{PD}$ The W register is subtracted (2's comple-Description: TO. PD Status Affected: ment method) from the eight bit literal 'k'. Encoding: 00 0000 0110 0011 The result is placed in the W register. The power-down status bit, PD is Words: 1 Description: cleared. Time-out status bit, TO is Cycles: 1 set. Watchdog Timer and its prescaler Q Cycle Activity: Q1 Q2 Q3 Q4 are cleared. The processor is put into SLEEP Write to W Decode Read Process mode with the oscillator stopped. See literal 'k' data Section 14.8 for more details. Words: 1 Example 1: SUBLW 0x02 Cycles: 1 Before Instruction Q Cycle Activity: Q1 Q2 Q3 Q4 w 1 -С ? -Decode No-Opera No-Opera Go to tion Z tion Sleep 2 After Instruction Example: SLEEP W = 1 С 1; result is positive = 7 0 = Example 2: **Before Instruction** W 2 -С 2

=

= 7 _ 1

= С =

?

? = Ζ

? =

0 =

0xFF

1; result is zero

0; result is nega-

Ζ

С

С

tive Ζ

After Instruction W = 0

Before Instruction W = 3

After Instruction W

Example 3:

SUBWF	Subtract	W from f		
Syntax:	[label]	SUBWF	f,d	
Operands:	$\begin{array}{l} 0 \leq f \leq 12 \\ d \in \ [0,1] \end{array}$	7		
Operation:	(f) - (W) –	→ (destina	ition)	
Status Affected:	C, DC, Z			
Encoding:	00	0010	dfff	ffff
Description:	Subtract (2's complement method) W reg- ister 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			
Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process data	Write to destination
Example 1:	SUBWF		reg1,1	
	Before Ins	struction		
	REG1	=	3	
	W C	=	2 ?	
	z	=	?	
	After Instr	ruction		
	REG1	=	1	
	W C	=	2 1; result is	positivo
	z	=	0	positive
Example 2:	Before Instruction			
	REG1	=	2	
	W	=	2	
	C Z	=	? ?	
	After Instr	ruction		
	REG1	=	0	
	W	=	2	
	C Z	=	1; result is	zero
Example 3:	Before Ins	struction		
	REG1	=	1	
	W C	=	2 ?	
	z	=	? ?	
	After Instr	ruction		
	REG1	=	0xFF	
	W	=	2	
	C Z	=	0; result is 0	negative
	-		-	

SWAPF	Swap Ni	bbles in t	f		
Syntax:	[label] SWAPF f,d				
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]				
Operation:	$(f<3:0>) \rightarrow (destination<7:4>),$ $(f<7:4>) \rightarrow (destination<3:0>)$				
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				
Q Cycle Activity:	Q1	Q2	Q3 Q4		
	Decode	Read register 'f'	Process data	Write to destination	
Example	SWAPF	REG,	0		
	Before Instruction				
		REG1	= 0xA	\ 5	
	After Instruction				
		REG1 W	= 0xA = 0x5	.0	

TRIS	Load TRIS Register		
Syntax:	[<i>label</i>] TRIS f		
Operands:	$5 \le f \le 7$		
Operation:	(W) \rightarrow TRIS register f;		
Status Affected:	None		
Encoding:	00 0000 0110 0fff		
Description:	The instruction is supported for code compatibility with the PIC16C5X prod- ucts. Since TRIS registers are read- able and writable, the user can directly address them.		
Words:	1		
Cycles:	1		
Example			
	To maintain upward compatibility with future PIC16CXX products, do not use this instruction.		

XORLW	Exclusive OR Literal with W	XORWF	Exclusive OR W with f
Syntax:	[<i>label</i>] XORLW k	Syntax:	[<i>label</i>] XORWF f,d
Operands:	0 ≤ k ≤ 255	Operands:	$0 \le f \le 127$
Operation:	(W) .XOR. $k \rightarrow (W)$	Operation:	$d \in [0,1]$ (W) .XOR. (f) \rightarrow (destination)
Status Affected:	Z	Status Affected:	Z
Encoding:	11 1010 kkkk kkkk	Encoding:	00 0110 dfff ffff
Description:	The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W regis- ter.	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	Words:	1
Cycles:	1	Cycles:	1
Q Cycle Activity:	Q1 Q2 Q3 Q4	Q Cycle Activity:	Q1 Q2 Q3 Q4
	Decode Read Process Write to literal 'k' data W	Q Oyole Activity.	Decode Read Process Write to destination
Example:	XORLW 0xAF		
	Before Instruction	Example	XORWF REG 1
	W = 0xB5		Before Instruction
	After Instruction W = 0x1A		REG = 0xAF W = 0xB5
			After Instruction
			REG = 0x1A

 $\begin{array}{rcl} \mathsf{REG} &=& 0\mathsf{x}\mathsf{1}\mathsf{A} \\ \mathsf{W} &=& 0\mathsf{x}\mathsf{B5} \end{array}$