### 27.0 INSTRUCTION SET SUMMARY

PIC18(L)F2X/45K50 devices incorporate the standard set of 75 PIC18 core instructions, as well as an extended set of eight new instructions, for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

#### 27.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC® MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are four instructions that require two program memory locations.

Each single-word instruction is a 16-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 instruction set is highly orthogonal and is grouped into four basic categories:

- · Byte-oriented operations
- · Bit-oriented operations
- · Literal operations
- · Control operations

The PIC18 instruction set summary in Table 27-2 lists byte-oriented, bit-oriented, literal and control operations. Table 27-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

- 1. The file register (specified by 'f')
- The destination of the result (specified by 'd')
- 3. The accessed memory (specified by 'a')

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

All **bit-oriented** instructions have three operands:

- 1. The file register (specified by 'f')
- 2. The bit in the file register (specified by 'b')
- 3. The accessed memory (specified by 'a')

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by 'k')
- The desired FSR register to load the literal value into (specified by 'f')
- No operand required (specified by '—')

The **control** instructions may use some of the following operands:

- A program memory address (specified by 'n')
- The mode of the CALL or RETURN instructions (specified by 's')
- The mode of the table read and table write instructions (specified by 'm')
- No operand required (specified by '—')

All instructions are a single word, except for four double-word instructions. These instructions were made double-word to contain the required information in 32 bits. In the second word, the 4 MSbs are '1's. If this second word is executed as an instruction (by itself), it will execute as a NOP.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a NOP.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1  $\mu s$ . If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2  $\mu s$ . Two-word branch instructions (if true) would take 3  $\mu s$ .

Figure 27-1 shows the general formats that the instructions can have. All examples use the convention 'nnh' to represent a hexadecimal number.

The Instruction Set Summary, shown in Table 27-2, lists the standard instructions recognized by the Microchip Assembler (MPASM $^{TM}$ ).

Section 27.1.1 "Standard Instruction Set" provides a description of each instruction.

### TABLE 27-1: OPCODE FIELD DESCRIPTIONS

Field	Description
	RAM access bit
a	a = 0: RAM location in Access RAM (BSR register is ignored)
	a = 1: RAM bank is specified by BSR register
bbb	Bit address within an 8-bit file register (0 to 7).
BSR	Bank Select Register. Used to select the current RAM bank.
C, DC, Z, OV, N	
d	Destination select bit
	d = 0: store result in WREG
	d = 1: store result in file register f
dest	Destination: either the WREG register or the specified register file location.
f	8-bit Register file address (00h to FFh) or 2-bit FSR designator (0h to 3h).
f <sub>s</sub>	12-bit Register file address (000h to FFFh). This is the source address.
f <sub>d</sub>	12-bit Register file address (000h to FFFh). This is the destination address.
GIE	Global Interrupt Enable bit.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
label	Label name.
mm	The mode of the TBLPTR register for the table read and table write instructions.
	Only used with table read and table write instructions:
*	No change to register (such as TBLPTR with table reads and writes)
*+	Post-Increment register (such as TBLPTR with table reads and writes)
*-	Post-Decrement register (such as TBLPTR with table reads and writes)
+*	Pre-Increment register (such as TBLPTR with table reads and writes)
n	The relative address (2's complement number) for relative branch instructions or the direct address for
	CALL/BRANCH and RETURN instructions.
PC	Program Counter.
PCL	Program Counter Low Byte.
PCH	Program Counter High Byte.
PCLATH	Program Counter High Byte Latch.
PCLATU	Program Counter Upper Byte Latch.
PD	Power-down bit.
PRODH	Product of Multiply High Byte.
PRODL	Product of Multiply Low Byte.
s	Fast Call/Return mode select bit
	s = 0: do not update into/from shadow registers
	s = 1: certain registers loaded into/from shadow registers (Fast mode)
TBLPTR	21-bit Table Pointer (points to a Program Memory location).
TABLAT	8-bit Table Latch.
TO	Time-out bit.
TOS	Top-of-Stack.
u	Unused or unchanged.
WDT	Watchdog Timer.
WREG	Working register (accumulator).
х	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.
Z <sub>S</sub>	7-bit offset value for indirect addressing of register files (source).
z <sub>d</sub>	7-bit offset value for indirect addressing of register files (destination).
[	Optional argument.
[text]	Indicates an indexed address.
(text)	The contents of text.
[expr] <n></n>	Specifies bit n of the register indicated by the pointer expr.
$\rightarrow$	Assigned to.
< >	Register bit field.
€	In the set of.
italics	User defined term (font is Courier).

### FIGURE 27-1: GENERAL FORMAT FOR INSTRUCTIONS

#### Byte-oriented file register operations **Example Instruction** 10 9 8 7 15 OPCODE d a f (FILE #) ADDWF MYREG, W, B d = 0 for result destination to be WREG register d = 1 for result destination to be file register (f) a = 0 to force Access Bank a = 1 for BSR to select bank f = 8-bit file register address Byte to Byte move operations (2-word) 12 11 0 OPCODE f (Source FILE #) MOVFF MYREG1, MYREG2 15 12 11 0 f (Destination FILE #) 1111 f = 12-bit file register address Bit-oriented file register operations 12 11 987 OPCODE b (BIT #) a f (FILE #) BSF MYREG, bit, B b = 3-bit position of bit in file register (f) a = 0 to force Access Bank a = 1 for BSR to select bank f = 8-bit file register address Literal operations 15 **OPCODE** k (literal) MOVLW 7Fh k = 8-bit immediate value **Control** operations CALL, GOTO and Branch operations 15 0 **OPCODE** n<7:0> (literal) GOTO Label 12 11 0 15 1111 n<19:8> (literal) n = 20-bit immediate value 15 **OPCODE** n<7:0> (literal) CALL MYFUNC 0 15 12 11 1111 n<19:8> (literal) S = Fast bit 11 10 0 OPCODE n<10:0> (literal) BRA MYFUNC 15 8 7 n OPCODE BC MYFUNC n<7:0> (literal)

TABLE 27-2: PIC18 INSTRUCTION SET

Mnemonic, Operands				16-	Bit Instr	uction W	/ord	Status	
		Description	Cycles	MSb			LSb	Affected	Notes
BYTE-ORIE	ENTED (	OPERATIONS							
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and CARRY bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3,
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
NCF	f, d, a	Increment f	1 ` ´	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3,
NCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
NFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1 ` ′	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	$f_s, f_d$	Move f <sub>s</sub> (source) to 1st word	2	1100	ffff	ffff	ffff	None	
	3, u	f <sub>d</sub> (destination) 2nd word		1111	ffff	ffff	ffff		
MOVWF	f. a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	1, 2
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	,
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	1, 2
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	,
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	1, 2
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	,
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	,-
SWAPF	f, d, a	Swap nibbles in f	1	0011	10da	ffff	ffff	None	4
TSTFSZ	f, a	Test f, skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None	1, 2
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N	

Note 1: When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), 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'.

<sup>2:</sup> If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

<sup>3:</sup> 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.

**<sup>4:</sup>** Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

TABLE 27-2: PIC18 INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cualas	16-Bit Instruction Word			Status	Natas	
		Description	Cycles	MSb			LSb	Affected	Notes
BIT-ORIEN	TED OP	ERATIONS							
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, b, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2
CONTROL	OPERA	TIONS	•	•					•
3C	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	
3N	n	Branch if Negative	1 (2)	1110	0110	nnnn	nnnn	None	
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None	
BNN	n	Branch if Not Negative	1 (2)	1110	0111	nnnn	nnnn	None	
BNOV	n	Branch if Not Overflow	1 (2)	1110	0101	nnnn	nnnn	None	
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None	
BOV	n	Branch if Overflow	1 (2)	1110	0100	nnnn	nnnn	None	
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None	
3Z	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	k, s	Call subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
CLRWDT	_	Clear Watchdog Timer	1	0000	0000	0000	0100	TO, PD	
DAW	_	Decimal Adjust WREG	1	0000	0000	0000	0111	С	
GOTO	k	Go to address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk		
NOP	_	No Operation	1	0000	0000	0000	0000	None	
NOP	_	No Operation	1	1111	xxxx	XXXX	xxxx	None	4
POP	_	Pop top of return stack (TOS)	1	0000	0000	0000	0110	None	
PUSH	_	Push top of return stack (TOS)	1	0000	0000	0000	0101	None	
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None	
RESET		Software device Reset	1	0000	0000	1111	1111	All	
RETFIE	S	Return from interrupt enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None	
SLEEP	_	Go into Standby mode	1	0000	0000	0000	0011	TO, PD	

Note 1: When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), 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'.

<sup>2:</sup> If this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.

<sup>3:</sup> 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.

<sup>4:</sup> Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

TABLE 27-2: PIC18 INSTRUCTION SET (CONTINUED)

Mnemonic,		December 1	Cuala a	16-Bit Instruction Word			Word	Status	Natas
Opera	ınds	Description	Cycles	MSb			LSb	Affected	Notes
LITERAL (	OPERATI	ONS							
ADDLW	k	Add literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW	k	AND literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	Inclusive OR literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	Move literal (12-bit) 2nd word	2	1110	1110	OOff	kkkk	None	
		to FSR(f) 1st word		1111	0000	kkkk	kkkk		
MOVLB	k	Move literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW	k	Move literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW	k	Multiply literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW	k	Return with literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW	k	Subtract WREG from literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW	k	Exclusive OR literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEN	MORY ↔	PROGRAM MEMORY OPERATION	IS						
TBLRD*		Table Read	2	0000	0000	0000	1000	None	
TBLRD*+		Table Read with post-increment		0000	0000	0000	1001	None	
TBLRD*-		Table Read with post-decrement		0000	0000	0000	1010	None	
TBLRD+*		Table Read with pre-increment		0000	0000	0000	1011	None	
TBLWT*		Table Write	2	0000	0000	0000	1100	None	
TBLWT*+		Table Write with post-increment		0000	0000	0000	1101	None	
TBLWT*-		Table Write with post-decrement		0000	0000	0000	1110	None	
TBLWT+*		Table Write with pre-increment		0000	0000	0000	1111	None	

- Note 1: When a PORT register is modified as a function of itself (e.g., MOVF PORTB, 1, 0), 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.
  - 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.
  - 4: Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

### 27.1.1 STANDARD INSTRUCTION SET

ADD	LW	ADD liter	al to W					
Synta	ax:	ADDLW	k					
Oper	ands:	$0 \le k \le 255$	$0 \leq k \leq 255$					
Oper	ation:	$(W) + k \rightarrow V$	W					
Statu	s Affected:	N, OV, C, E	C, Z					
Enco	ding:	0000	1111	kkkk	kkkk			
Desc	ription:	The conten 8-bit literal W.						
Word	ls:	1	1					
Cycle	es:	1						
QC	ycle Activity:							
	Q1	Q2	Q3	3	Q4			
	Decode	Read literal 'k'	Proce Data		ite to W			

Example: ADDLW 15h

Before Instruction W = 10hAfter Instruction W = 25h

ADDWF	ADD W to f					
Syntax:	ADDWF f {,d {,a}	}				
Operands:	$0 \le f \le 255$ d $\in [0,1]$ a $\in [0,1]$					
Operation:	(W) + (f) $\rightarrow$ dest					
Status Affected:	N, OV, C, DC, Z					
Encoding:	0010 01da	ffff	ffff			
Description:	Add W to register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See  Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.					
Words:	1					
Cycles:	1					

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

Example: ADDWF REG, 0, 0

Before Instruction

W = 17h REG = 0C2h

After Instruction

W = 0D9hREG = 0C2h

**Note:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

#### **ADDWFC** ADD W and CARRY bit to f

Syntax: **ADDWFC** f {,d {,a}} Operands:  $0 \le f \le 255$  $d \in [0,1]$  $a\in \left[ 0,1\right]$ Operation:  $(W) + (f) + (C) \rightarrow dest$ Status Affected: N,OV, C, DC, Z Encoding: 0010 ffff ffff 00da Description: Add W, the CARRY flag and data mem-

ory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \le 95$  (5Fh). See Section 27.2.3 "Byte-Oriented and **Bit-Oriented Instructions in Indexed** Literal Offset Mode" for details.

Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

Example: ADDWFC REG, 0, 1

Before Instruction

CARRY bit = 02h REG 4Dh

After Instruction

CARRY bit = 0 REG 02h 50h **ANDLW** AND literal with W

ANDLW Syntax: Operands:  $0 \le k \le 255$ Operation: (W) .AND.  $k \rightarrow W$ 

Status Affected: N, Z

Encoding: 0000 1011 kkkk kkkk

Description: The contents of W are AND'ed with the 8-bit literal 'k'. The result is placed in W.

Words: Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write to W
	'k'	Data	

Example: ANDLW 05Fh

Before Instruction

A3h

After Instruction

W 03h ANDWF AND W with f

Syntax: ANDWF  $f \{,d \{,a\}\}$ 

 $0 \le f \le 255$   $d \in [0,1]$  $a \in [0,1]$ 

Operation: (W) .AND. (f)  $\rightarrow$  dest

Status Affected: N, Z

Operands:

Encoding: 0001 01da ffff ffff

Description: The contents of W are AND'ed with register 'f'. If 'd' is '0', the result is stored

in register 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

in W. If 'd' is '1', the result is stored back

GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

Example: ANDWF REG, 0, 0

Before Instruction

W = 17h REG = C2h

After Instruction

W = 02hREG = C2h BC Branch if Carry

Syntax: BC n

Operands:  $-128 \le n \le 127$ Operation: if CARRY bit is '1'

 $(PC) + 2 + 2n \rightarrow PC$ 

Status Affected: None

Encoding: 1110 0010 nnnn nnnn

Description: If the CARRY bit is '1', then the program

will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have

incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a

two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity: If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write to PC
	ʻn'	Data	
No	No	No	No
operation	operation	operation	operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	ʻn'	Data	operation

Example: HERE BC 5

Before Instruction

PC = address (HERE)

After Instruction

If CARRY = 1;

PC = address (HERE + 12)

If CARRY = 0;

PC = address (HERE + 2)

BCF	Bit Clear	f		
Syntax:	BCF f, b	(,a)		
Operands:	$0 \le f \le 255$ $0 \le b \le 7$ $a \in [0,1]$			
Operation:	$0 \rightarrow f < b >$			
Status Affected:	None			
Encoding:	1001	bbba	ffff	ffff
Description:	Bit 'b' in re	•		selected.

If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \le 95$  (5Fh). See Section 27.2.3 "Byte-Oriented and

**Bit-Oriented Instructions in Indexed** Literal Offset Mode" for details.

Words: Cycles: Q Cycle Activity:

> Q1 Q2 Q3 Q4 Decode Write Read **Process** register 'f' Data register 'f'

Example: BCF FLAG\_REG, 7, 0

Before Instruction FLAG\_REG = C7h After Instruction FLAG\_REG = 47h

BN **Branch if Negative** 

BN n

Syntax: Operands:  $\text{-}128 \leq n \leq 127$ 

Operation: if NEGATIVE bit is '1'  $(PC) + 2 + 2n \rightarrow PC$ 

Status Affected: None

Encoding: 1110 0110 nnnn nnnn

Description: If the NEGATIVE bit is '1', then the

program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have

incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a

two-cycle instruction.

Words: 1 Cycles: 1(2)

Q Cycle Activity: If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write to PC
	ʻn'	Data	
No	No	No	No
operation	operation	operation	operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example: HERE BN Jump

Before Instruction

PC address (HERE)

After Instruction

If NEGATIVE

address (Jump)

If NEGATIVE

address (HERE + 2)

BNC Branch if Not Carry

Syntax: BNC n

Operands:  $-128 \le n \le 127$ Operation: if CARRY bit is '0'

 $(PC) + 2 + 2n \rightarrow PC$ 

Status Affected: None

Encoding: 1110 0011 nnnn nnnn

Description: If the CARRY bit is '0', then the program

will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have

incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a

two-cycle instruction.

Words: 1 Cycles: 1(2)

Q Cycle Activity: If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write to PC
	ʻn'	Data	
No	No	No	No
operation	operation	operation	operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	ʻn'	Data	operation

Example: HERE BNC Jump

Before Instruction

PC = address (HERE)

After Instruction

If CARRY = 0;

PC = address (Jump)

If CARRY = 1:

PC = address (HERE + 2)

BNN Branch if Not Negative

Syntax: BNN n

Operands:  $-128 \le n \le 127$ 

Operation: if NEGATIVE bit is '0'  $(PC) + 2 + 2n \rightarrow PC$ 

Status Affected: None

Encoding: 1110 0111 nnnn nnnn

Description: If the NEGATIVE bit is '0', then the

program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have

incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a

two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity: If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write to PC
	ʻn'	Data	
No	No	No	No
operation	operation	operation	operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	ʻn'	Data	operation

Example: HERE BNN Jump

Before Instruction

PC = address (HERE)

After Instruction

If NEGATIVE = 0;

PC = address (Jump)

If NEGATIVE = 1;

PC = address (HERE + 2)

BNOV	Branch if Not Overflow
Syntax:	BNOV n
Operands:	$-128 \leq n \leq 127$
Operation:	if OVERFLOW bit is '0' (PC) + 2 + 2n $\rightarrow$ PC
Status Affected:	None

1110 Description: If the OVERFLOW bit is '0', then the

program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next

nnnn

0101

instruction, the new address will be PC + 2 + 2n. This instruction is then a

two-cycle instruction.

Words: 1 Cycles: 1(2)

Q Cycle Activity: If Jump:

Encoding:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write to PC
	ʻn'	Data	
No	No	No	No
operation	operation	operation	operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example: HERE BNOV Jump

Before Instruction

PC address (HERE)

After Instruction

If OVERFLOW =

address (Jump)

OVERFLOW =

address (HERE + 2)

**BNZ Branch if Not Zero** 

Syntax: BNZ n Operands:  $\text{-}128 \leq n \leq 127$ 

Operation: if ZERO bit is '0'

 $(PC) + 2 + 2n \rightarrow PC$ 

Status Affected: None

Description:

Encoding: 1110 0001 nnnn nnnn

> If the ZERO bit is '0', then the program will branch.

> > The 2's complement number '2n' is added to the PC. Since the PC will have

incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a

No

operation

Nο

operation

two-cycle instruction.

Words: 1 Cycles: 1(2)

No

Q Cycle Activity: If Jump:

Q1 Q2 Q3 Q4 Decode Read literal **Process** Write to PC 'n' Data

No

operation

operation If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	'n'	Data	operation

Example: HERE BNZ Jump

Before Instruction

PC address (HERE)

After Instruction

If ZERO

address (Jump)

address (HERE + 2)

### BRA Unconditional Branch

Syntax: BRA n

Operands:  $-1024 \le n \le 1023$ Operation:  $(PC) + 2 + 2n \rightarrow PC$ 

Status Affected: None

Encoding: 1101 Onnn nnnn nnnn

Description: Add the 2's complement number '2n' to

the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction.

Words: 1 Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write to PC
	ʻn'	Data	
No	No	No	No
operation	operation	operation	operation

Example: HERE BRA Jump

Before Instruction

PC = address (HERE)

After Instruction

PC = address (Jump)

BSF	Bit Set f
Syntax:	BSF f, b {,a}
Operands:	$0 \le f \le 255$ $0 \le b \le 7$ $a \in [0,1]$
Operation:	$1 \rightarrow f < b >$
Status Affected:	None
Encoding:	1000 bbba ffff ffff

Description: Bit 'b' in register 'f' is set.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f  $\leq$  95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words: 1

Cycles: 1

Q Cycle Activity:

	Q1	Q2	Q3	Q4
[	Decode	Read	Process	Write
		register 'f'	Data	register 'f'

Example: BSF FLAG\_REG, 7, 1

Before Instruction

FLAG\_REG = 0Ah

After Instruction

FLAG\_REG = 8Ah

BTFS	SC .	Bit Test Fil	le, Skip if C	lear	BTFSS	Bit Test Fil	e, Skip if Se	t
Syntax	X:	BTFSC f, b	{,a}		Syntax:	BTFSS f, b	{,a}	
Opera	nds:	$\begin{array}{l} 0 \leq f \leq 255 \\ 0 \leq b \leq 7 \\ a \in [0,1] \end{array}$			Operands:	$0 \le f \le 255$ $0 \le b < 7$ $a \in [0,1]$		
Opera	tion:	skip if (f <b>)</b>	= 0		Operation:	skip if (f <b>)</b>	= 1	
Status	Affected:	None			Status Affected:	None		
Encod	ling:	1011	bbba fi	fff ffff	Encoding:	1010	bbba ff	ff ffff
Descri	iption:	instruction is the next instru- and a NOP is this a two-cy If 'a' is '0', th 'a' is '1', the GPR bank. If 'a' is '0' an set is enable Indexed Lite mode where See Section Bit-Oriented	skipped. If bit ruction fetcher uction execution execution executed instruction execus Ban BSR is used to the extended, this instructional Offset Addever f ≤ 95 (5F	on is discarded stead, making it. It is selected. If to select the id instruction tion operates in ressing it. It is coriented and is in Indexed	Description:	If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' 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 two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.		
Words	s:	1			Words:	1		
Cycles			e cycles if ski 2-word instru	p and followed ction.	Cycles:	by a	e cycles if skip 2-word instruc	and followed ction.
Q Cy	cle Activity:	00	00	0.4	Q Cycle Activity:		00	0.4
Г	Q1 Decode	Q2 Read	Q3 Process	Q4 No	Q1 Decode	Q2 Read	Q3 Process	Q4 No
	Decode	register 'f'	Data	operation	Decode	register 'f'	Data	operation
If skip	):	3			If skip:	- 3	1	
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
	No	No	No	No	No	No	No	No
	operation	operation	operation	operation	operation		operation	operation
If skip		by 2-word ins			If skip and follow	-		
Г	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
	No operation	No operation	No operation	No operation	No operation	No operation	No operation	No operation
F	No	No	No	No	No	No	No	No
	operation	operation	operation	operation	operation	_	operation	operation
Example: HERE BTFSC FLAG, 1, 0  FALSE: TRUE:  Before Instruction PC = address (HERE)  After Instruction  If FLAG<1> = 0; PC = address (TRUE)  If FLAG<1> = 1; PC = address (FALSE)			Example:  Before Instruc PC After Instruc If FLAC PC If FLAC	FALSE TRUE  uction = action S<1> = 0; C = ac6<1> = 1;	BTFSS FLF: : : Idress (HERE Idress (FALS	E )		

**BTG** Bit Toggle f Syntax: BTG f, b {,a} Operands:  $0 \le f \le 255$  $0 \le b < 7$  $a \in [0,1]$  $(\overline{f < b >}) \rightarrow f < b >$ Operation: Status Affected: None Encoding: ffff ffff 0111 bbba Description: Bit 'b' in data memory location 'f' is inverted.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write
	register 'f'	Data	register 'f'

Example: BTG PORTC, 4, 0

Before Instruction:

PORTC = 0111 0101 [75h]

After Instruction:

PORTC = 0110 0101 [65h]

BOV Branch if Overflow

Syntax: BOV n

Operands:  $-128 \le n \le 127$ 

Operation: if OVERFLOW bit is '1'

 $(PC) + 2 + 2n \rightarrow PC$ 

Status Affected: None

Encoding: 1110 0100 nnnn nnnn

Description: If the OVERFLOW bit is '1', then the program will branch.

The 2's complement number '2n' is added to the PC. Since the PC will have

incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a

two-cycle instruction.

Words: 1 Cycles: 1(2)

Q Cycle Activity: If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write to PC
	ʻn'	Data	
No	No	No	No
operation	operation	operation	operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	ʻn'	Data	operation

Example: HERE BOV Jump

Before Instruction

PC = address (HERE)

After Instruction

If OVERFLOW = 1;

PC = address (Jump)

If OVERFLOW = 0;

PC = address (HERE + 2)

BZ n Syntax:

Operands:  $-128 \le n \le 127$ Operation: if ZERO bit is '1'

 $(PC) + 2 + 2n \rightarrow PC$ 

Status Affected: None

Encoding: 1110 0000 nnnn nnnn If the ZERO bit is '1', then the program

Description: will branch.

> The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a

two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity: If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write to PC
	ʻn'	Data	
No	No	No	No
operation	operation	operation	operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	No
	ʻn'	Data	operation

Example: HERE ΒZ Jump

Before Instruction

PC address (HERE)

After Instruction

If ZERO

address (Jump)

address (HERE + 2)

**CALL Subroutine Call** 

Syntax: CALL k {,s}

Operands:  $0 \leq k \leq 1048575$ 

 $s \in \left[0,1\right]$ 

Operation:  $(PC) + 4 \rightarrow TOS$ ,

 $k \rightarrow PC < 20:1>$ , if s = 1

 $(W) \rightarrow WS$ ,

 $(Status) \rightarrow STATUSS$ ,

 $(BSR) \rightarrow BSRS$ 

Status Affected: None

Encoding: 1st word (k<7:0>)

2nd word(k<19:8>)

1110	110s	k <sub>7</sub> kkk	kkkk <sub>0</sub>
1111	k <sub>19</sub> kkk	kkkk	kkkk <sub>8</sub>

Description: Subroutine call of entire 2-Mbyte

> memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no

update occurs. Then, the

20-bit value 'k' is loaded into PC<20:1>.

CALL is a two-cycle instruction.

Words: 2 2 Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	PUSH PC to	Read literal
	'k'<7:0>,	stack	'k'<19:8>,
			Write to PC
No	No	No	No
operation	operation	operation	operation

Example: HERE CALL THERE, 1

Before Instruction

PC address (HERE)

After Instruction

PC TOS address (THERE) = address (HERE + 4)

WS **BSR** STATUSS = Status

**CLRF** Clear f Syntax: CLRF f {,a} Operands:  $0 \le f \le 255$  $a \in \left[0,1\right]$  $000h \rightarrow f$ Operation:  $1 \rightarrow Z$ Status Affected: Ζ 101a Encoding: 0110 ffff ffff Description: Clears the contents of the specified

register.

If 'a' is '0' the Access Bank is selecte

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write
	register 'f'	Data	register 'f'

Example: CLRF FLAG\_REG, 1

Before Instruction

FLAG\_REG = 5Ah After Instruction

FLAG\_REG = 00h

CLRWDT Clear Watchdog Timer

Syntax: CLRWDT Operands: None

Operation:  $000h \rightarrow WDT$ ,

 $000h \xrightarrow{\rightarrow} WDT \text{ postscaler,}$   $1 \xrightarrow{} TO,$ 

 $1 \to 10,$   $1 \to \overline{PD}$   $\overline{TO}, \overline{PD}$ 

Encoding: 0000 0000 0000

Description: CLRWDT instruction resets the

Watchdog Timer. It also resets  $\underline{\text{the}}$  post-scaler of the WDT. Status bits,  $\overline{\text{TO}}$  and

0100

PD, are set.

Words: 1
Cycles: 1

Q Cycle Activity:

Status Affected:

Q1	Q2	Q3	Q4
Decode	No	Process	No
	operation	Data	operation

Example: CLRWDT

Before Instruction

WDT Counter = ?

After Instruction

 WDT Counter
 =
 00h

 WDT Postscaler
 =
 0

 TO
 =
 1

 PD
 =
 1

COMF	Complement f			
Syntax:	COMF 1	f {,d {,a}}		
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$	5		
Operation:	$(\overline{f}) \rightarrow dest$	t		
Status Affected:	N, Z			
Encoding:	0001	11da	ffff	ffff
Description:	The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.			
Words:	1			
Cycles:	1			
Q Cycle Activity:				
Q1	Q2	Q3	3	Q4

Example:	C	OMF	REG,	0,	0
Before Instru	ıction				
REG	=	13h			
After Instruct	tion				
REG	=	13h			

ECh

Read

register 'f'

Process

Data

Write to

destination

Decode

CPF	SEQ	Compare f with W, skip if f = W			
Synta	ax:	CPFSEQ	f {,a}		
Oper	ands:	$\begin{array}{l} 0 \leq f \leq 255 \\ a \in [0,1] \end{array}$			
Oper	ation:	(f) – (W), skip if (f) = (W) (unsigned comparison)			
Statu	s Affected:	None			
Enco	ding:	0110	001a ff	ff ffff	
Desc	ription:	location 'f' t performing If 'f' = W, th discarded a instead, ma instruction. If 'a' is '0', tl If 'a' is '1', tl GPR bank. If 'a' is '0' a set is enabl in Indexed I mode when Section 27	Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed		
Word	s:	1			
Cycles: 1(2)  Note: Three cycles if skip and followed by a 2-word instruction.			•		
Q C	ycle Activity:				
	Q1	Q2	Q3	Q4	
	Decode	Read register 'f'	Process Data	No operation	

		register 'f'	Data	operation
lf sk	ip:			
	Q1	Q2	Q3	Q4
	No	No	No	No
	operation	operation	operation	operation
If sk	ip and followed	by 2-word ins	struction:	

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation
No	No	No	No
operation	operation	operation	operation

	operation	operation	operation	operation
	No operation	No operation	No operation	No operation
Exan	nple:	HERE NEQUAL	CPFSEQ REG	, 0

Before Instruction PC Address HERE W ? REG After Instruction If REG W;

EQUAL

Address (EQUAL)

If REG

Address (NEQUAL)

#### **CPFSGT** Compare f with W, skip if f > W

Syntax: CPFSGT f {,a} Operands:  $0 \le f \le 255$  $a \in [0,1]$ 

(f) - (W),

Operation: skip if (f) > (W)

(unsigned comparison)

Status Affected: None

Encoding: Description:

0110 010a ffff ffff Compares the contents of data memory

location 'f' to the contents of the W by performing an unsigned subtraction. If the contents of 'f' are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a

two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \le 95$  (5Fh). See Section 27.2.3 "Byte-Oriented and **Bit-Oriented Instructions in Indexed** Literal Offset Mode" for details.

Words: 1 Cycles: 1(2)

Note: Three cycles if skip and followed

by a 2-word instruction.

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	No
	register 'f'	Data	operation

If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation
No	No	No	No
operation	operation	operation	operation

Example: HERE CPFSGT REG, 0

> NGREATER GREATER

Before Instruction

PC Address (HERE)

W ?

After Instruction

If REG W; >

> PC = Address (GREATER)

If REG ≤ W.

PC Address (NGREATER)

#### **CPFSLT** Compare f with W, skip if f < W

CPFSLT f {,a} Syntax: Operands:  $0 \leq f \leq 255$ 

 $a \in [0,1]$ 

Operation: (f) - (W),skip if (f) < (W)

(unsigned comparison)

Status Affected: None

Encoding: 0110 000a ffff ffff

Description: Compares the contents of data memory

> location 'f' to the contents of W by performing an unsigned subtraction. If the contents of 'f' are less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a

two-cycle instruction.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

GPR bank.

1 Words: Cycles: 1(2)

> Note: Three cycles if skip and followed by a 2-word instruction.

### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	No
	register 'f'	Data	operation

### If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation
No	No	No	No
operation	operation	operation	operation

Example: HERE CPFSLT REG, 1

NLESS LESS

Before Instruction

After Instruction

PC W = Address (HERE)

=

If REG < W:

PC Address (LESS)

If REG ≥

PC Address (NLESS)

DAV	v	Decimal Adjust W Register					
Synt	ax:	D	AW				
Ope	rands:	Ν	one				
Ope	ration:	(V el	[W<3:0> : V<3:0>) + se V<3:0>) -	6 → W<	3:0>;	then	
			[W<7:4> - W<7:4>) + = 1; se V<7:4>) +	6 + DC -	→ W<		
	ıs Affected:	C		1			
Enco	oding:		0000	0000	0000		
Description:			sulting fro	m the ea	rlier ac	alue in W, ddition of tw BCD format cked BCD	
Word	ds:	1					
Cycl	es:	1					
QC	ycle Activity:						
	Q1		Q2	Q3		Q4	
	Decode		Read	Proce		Write	
Ever	nnlo1:	re	gister W	Data	a	W	
<u> Exai</u>	nple1:	ח	ΑW				
	Before Instruc		-111				
	W	=	A5h				
	С	=	0				
	DC After Instruction	= on	0				
W : C : DC :		= = =	05h 1 0				
Example 2:  Before Instruction		tion					
	W	=	CEh				
	C DC	=	0				
	After Instruction		0.41				
	W C DC	= = =	34h 1 0				

DEC	F	Decremer	Decrement f				
Synta	IX:	DECF f{,c	l {,a}}				
Opera	ands:	$\begin{array}{l} 0 \leq f \leq 255 \\ d \in [0,1] \\ a \in [0,1] \end{array}$	d ∈ [0,1]				
Opera	ation:	$(f) - 1 \rightarrow de$	$(f) - 1 \rightarrow dest$				
Statu	s Affected:	C, DC, N, OV, Z					
Enco	ding:	0000	01da ffi	ff ffff			
	ription:	Decrement register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'. If 'a' is '0', the Access Bank is selecte If 'a' is '0', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operate in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.					
Word	s:	1					
Cycles:		1					
Q Cycle Activity:							
r	Q1	Q2	Q3	Q4			
	Decode	Read register 'f'	Process Data	Write to destination			

Example:	DECF		CNT,	1, 0
Before Instruc CNT 7	ction = =	01h 0		
After Instruction				
CNT Z	=	00h 1		

DEC	FSZ	Decreme	nt f, skip if (	)	DCF	SNZ	Decreme	nt f, skip if n	ot 0
Synta	ax:	DECFSZ	f {,d {,a}}		Synt	ax:	DCFSNZ	f {,d {,a}}	
Oper	ands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$		Ope	rands:	$\begin{array}{l} 0 \leq f \leq 255 \\ d \in [0,1] \\ a \in [0,1] \end{array}$	$d \in [0,1]$		
Oper	ation:	(f) $-1 \rightarrow de$ skip if resul			Ope	ration:	(f) $-1 \rightarrow de$ skip if result		
Statu	s Affected:	None			Statu	us Affected:	None		
Enco	ding:	0010	11da ff	f ffff	Enco	oding:	0100	11da fff	f ffff
Desc	Description:  The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'.  If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction.  If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See  Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.		Desc	cription:	decrements placed in V placed bac If the result instruction, discarded a instead, ma instruction. If 'a' is '0', t If 'a' is '1', t GPR bank. If 'a' is '0' a set is enab in Indexed mode wher Section 27 Bit-Oriente	nd the extendind, this instruction of the control	the result is the result in the result is the result is the result is the result is the result in the result is the result in the result is the result in the result is the result is the result in the result is the result is the result is the result is the result in the result is the result is the result is the result in the result is the result is the result in the result in the result is the result in the result in the result in the result is the result in th		
Word	ls:	1			\A/	-l		set Mode" for	details.
Cycle	es:		ee cycles if sk a 2-word inst	•	Word Cycl			ree cycles if s	•
Q C	ycle Activity:				0.0		followed by	a 2-word insti	uction.
ĺ	Q1	Q2	Q3	Q4	Q C	Cycle Activity:	00	00	04
	Decode	Read register 'f'	Process Data	Write to destination		Q1 Decode	Q2 Read	Q3 Process	Q4 Write to
lf sk	ip:	register i	Data	acstination	J	Decode	register 'f'	Data	destination
	Q1	Q2	Q3	Q4	If sk	kip:			
	No	No	No	No		Q1	Q2	Q3	Q4
	operation	operation	operation	operation		No	No	No	No
If sk	ip and followe	d by 2-word in	struction:			operation	operation	operation	operation
ı	Q1	Q2	Q3	Q4	IT SH	kip and followe	,		0.4
	No	No	No	No		Q1 No	Q2	Q3	Q4
	operation No	operation No	operation No	operation No		operation	No operation	No operation	No operation
	operation	operation	operation	operation		No	No	No	No
	'				1	operation	operation	operation	operation
Exam	nple:	HERE CONTINUE	DECFSZ GOTO	CNT, 1, 1 LOOP	<u>Exar</u>	nple:	ZERO	:	MP, 1, 0
	Before Instruc	ction						:	
	PC After Instruction CNT If CNT PC	on = CNT - 1 = 0;	S (HERE)  S (CONTINUE	.)		TEMP After Instruction TEMP If TEMP	=	? TEMP – 1, 0;	
	If CNT PC	<b>≠</b> 0;	S (HERE + 2			PC If TEMP PC	- = ≠ =	Address (: 0; Address (:	

#### **GOTO Unconditional Branch**

Syntax: GOTO k Operands:  $0 \leq k \leq 1048575$ Operation:  $k \rightarrow PC < 20:1 >$ 

Status Affected: None

Encoding: 1st word (k<7:0>) 1110 1111 k<sub>7</sub>kkk kkkk<sub>0</sub> 2nd word(k<19:8>) 1111  $kkkk_8$ k<sub>19</sub>kkk kkkk

Description: GOTO allows an unconditional branch

anywhere within entire

2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle

instruction.

2 Words: Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	No	Read literal
	'k'<7:0>,	operation	'k'<19:8>,
			Write to PC
No	No	No	No
operation	operation	operation	operation

Example: GOTO THERE

After Instruction

PC = Address (THERE)

#### **INCF** Increment f

Syntax: INCF f {,d {,a}} Operands:  $0 \leq f \leq 255$  $d \in \left[0,1\right]$  $a \in \left[0,1\right]$ Operation: (f) + 1  $\rightarrow$  dest Status Affected: C, DC, N, OV, Z

Encoding: 0010 ffff ffff 10da

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is

placed back in register 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

If 'a' is '0' and the extended instruction

GPR bank.

set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \le 95$  (5Fh). See Section 27.2.3 "Byte-Oriented and **Bit-Oriented Instructions in Indexed** Literal Offset Mode" for details.

Words: 1

Q Cycle Activity:

Cycles:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

Example: INCF CNT, 1, 0

1

Before Instruction

? =

FFh = 0 =

ĎC After Instruction

CNT

Z C

CNT Z C DC 00h

INCI	-SZ	Increment	f, skip if 0		IN	FSNZ	li	ncremen	t f, skip if no	ot 0
Synta	ax:	INCFSZ f	{,d {,a}}		Sy	ntax:	II	NFSNZ f	{,d {,a}}	
Operands: $ 0 \le f \le 255 $ $ d \in [0,1] $ $ a \in [0,1] $		Op	perands:	d	$\leq f \leq 255$ $\in [0,1]$ $\in [0,1]$					
Oper	ation:	` '			Op	peration:	,	(f) + 1 $\rightarrow$ dest, skip if result $\neq$ 0		
Statu	s Affected:	None			Sta	atus Affected:	Ν	lone		
Enco	dina.	г т	11da fff	f ffff	Er	ncoding:		0100	10da fff	ff ffff
Description:  The contents incremented. placed in W. I placed back in If the result is which is alread and a NOP is it a two-cycle If 'a' is '0', the If 'a' is '1', the GPR bank. If 'a' is '0' and set is enabled in Indexed Litt mode wheneved Section 27.2. Bit-Oriented		Described by the control of the cont		escription:	ir p p Iff ir d d ir ir If If S s ir	cremente laced in W laced back the result estruction, iscarded a estead, ma estruction. 'a' is '0', t 'a' is '1', t EPR bank. 'a' is '0' a et is enable indexed in node when ection 27 ist-Oriente	ts of register 'f d. If 'd' is '0', th /. If 'd' is '1', th k in register 'f'. is not '0', the i which is alreading a NOP is ex iking it a two-c the Access Bar he BSR is used and the extende ed, this instruct Literal Offset A lever f ≤ 95 (5F 2.3 "Byte-Ori d Instruction set Mode" for	ne result is e result is e result is next dy fetched, is eccuted ycle nk is selected. d to select the ed instruction etion operates addressing Fh). See iented and is in Indexed		
Word		1				ords:	1			
Cycle	es:		cles if skip and 2-word instruc		Су	/cles:			cycles if skip a a 2-word instr	
Q C	ycle Activity:				Q	Cycle Activity	<i>r</i> :			
	Q1	Q2	Q3	Q4	-	Q1		Q2	Q3	Q4
	Decode	Read	Process	Write to		Decode		Read	Process	Write to
16 -1-		register 'f'	Data	destination			re	gister 'f'	Data	destination
If sk	•	00	00	04	If	skip:		00	00	0.4
	Q1 No	Q2 No	Q3 No	Q4 No	]	Q1		Q2	Q3	Q4
	operation	operation	operation	operation		No operation	,   ,	No peration	No operation	No operation
If sk	ip and followed	d by 2-word ins	struction:	<u> </u>	ı If	skip and follow				000.000
	Q1	Q2	Q3	Q4		Q1	,	Q2	Q3	Q4
	No	No	No	No		No		No	No	No
	operation	operation	operation	operation		operation	op	eration	operation	operation
	No	No	No	No		No		No	No	No
	operation	operation	operation	operation		operation	op	eration	operation	operation
Example: HERE INCFSZ CNT, 1, 0 NZERO : ZERO :		<u>Ex</u>	cample:	Z N	ERE : ERO ZERO	INFSNZ REG	, 1, 0			
	Before Instruc PC After Instructic CNT If CNT PC If CNT PC CNT PC CONT	= Address on = CNT + 1 = 0; = Address ≠ 0;	(HERE)			Before Instructure PC After Instructure REG If REG PC If REG PC	= ction = 6 ≠ =	REG + 0; Address 0;	(HERE)  (NZERO)  (ZERO)	

#### **IORLW** Inclusive OR literal with W

Syntax: IORLW k Operands:  $0 \le k \le 255$ Operation: (W) .OR.  $k \rightarrow W$ 

N, Z Status Affected:

Encoding: 0000 1001 kkkk kkkk

Description: The contents of W are ORed with the eight-bit literal 'k'. The result is placed in

W.

Words: 1 Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to W
	literal 'k'	Data	

Example: IORLW 35h

Before Instruction

W 9Ah

After Instruction

W BFh **IORWF** Inclusive OR W with f

Syntax: IORWF f {,d {,a}} Operands:  $0 \leq f \leq 255$ 

 $d \in \left[0,1\right]$  $a \in \left[0,1\right]$ 

Operation: (W) .OR. (f)  $\rightarrow$  dest

Status Affected: N, Z

Encoding: 0001 00da ffff ffff

Description: Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \le 95$  (5Fh). See Section 27.2.3 "Byte-Oriented and **Bit-Oriented Instructions in Indexed** 

Literal Offset Mode" for details.

Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

Example: IORWF RESULT, 0, 1

Before Instruction

RESULT = 13h W 91h After Instruction

> RESULT = 13h W 93h

**LFSR** Load FSR

Syntax: LFSR f. k Operands:  $0 \leq f \leq 2$ 

 $0 \le k \le 4095$ 

 $k \to FSRf$ Operation:

Status Affected: None

Encoding: 1110 1110 00ff  $k_{11}kkk$ 0000 1111 k<sub>7</sub>kkk kkkk

Description: The 12-bit literal 'k' is loaded into the

File Select Register pointed to by 'f'.

2 Words: Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal	Process	Write
	'k' MSB	Data	literal 'k'
			MSB to
			FSRfH
Decode	Read literal 'k' LSB	Process Data	Write literal 'k' to FSRfL

Example: LFSR 2, 3ABh

After Instruction

FSR2H FSR2L 03h ABh

MOVF	Move f
Syntax:	MOVF f {,d {,a}}
Operands:	$0 \le f \le 255$ d $\in [0,1]$ a $\in [0,1]$
Operation:	$f \rightarrow dest$
Status Affected:	N, Z
Encoding:	0101 00da ffff ffff
Description:	The contents of register 'f' are moved to

Description: The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f'. Location 'f'

can be anywhere in the

256-byte bank. If 'a' is '0', the Access Bank is selected.

If 'a' is '1', the BSR is used to select the

GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \le 95$  (5Fh). See Section 27.2.3 "Byte-Oriented and **Bit-Oriented Instructions in Indexed** 

Literal Offset Mode" for details.

Words: 1 Cycles:

Q Cycle Activity:

	Q1	Q2	Q3	Q4	
De	ecode	Read register 'f'	Process Data	Write W	

Example: MOVF REG, 0, 0

Before Instruction

**REG** 22h W FFh

After Instruction

REG 22h W 22h

MOVFF	Move f to f			
Syntax:	MOVFF f <sub>s</sub> ,f <sub>d</sub>			
Operands:	$0 \le f_s \le 4095$ $0 \le f_d \le 4095$			
Operation:	$(f_s) \rightarrow f_d$			
Status Affected:	None			
Encoding: 1st word (source) 2nd word (destin.)	1100 ffff ffff ffff <sub>s</sub> 1111 ffff ffff ffff <sub>d</sub>			
Description:	The contents of source register 'fs' are			

moved to destination register ' $f_d$ '. Location of source 'fs' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'fd' can also be anywhere from 000h to

Either source or destination can be W (a useful special situation). MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit

buffer or an I/O port). The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the

destination register.

2 Words: Cycles: 2 (3)

Q Cycle Activity:

Q1	Q2	Q3	Q4	
Decode	Read register 'f' (src)	Process Data	No operation	
Decode	No operation No dummy read	No operation	Write register 'f' (dest)	

Example: MOVFF REG1, REG2

Before Instruction

REG1 REG2 33h 11h

After Instruction

33h 33h REG1 REG2

MΟ\	/LB	Move literal to low nibble in BSR				
Synta	ax:	MOVLB k				
Oper	ands:	$0 \le k \le 255$	5			
Oper	ration:	$k \to BSR$				
Statu	s Affected:	None				
Enco	ding:	0000	0000 0001 kkk		kkkk	
Description:		The eight-l Bank Selection of BSR<7: regardless	ct Registe 4> always	er (BSR). s remains	The value '0',	
Word	is:	1				
Cycles:		1				
Q C	ycle Activity:					
	Q1	Q2	Q3	3	Q4	
	Decode	Read	Proce	ess W	rite literal	

Example: MOVLB 5

Before Instruction

BSR Register = 02h

literal 'k'

'k' to BSR

Data

After Instruction

05h BSR Register =

Description: The eight-bit literal 'k' is loaded into W.

Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4	
Decode	Read literal 'k'	Process Data	Write to W	

Example: MOVLW 5Ah

After Instruction

W = 5Ah

MOVWF	Move W	to f		
Syntax:	MOVWF	f {,a}		
Operands:	$0 \le f \le 255$ $a \in [0,1]$	5		
Operation:	$(W) \rightarrow f$			
Status Affected:	None			
Encoding:	0110	111a	ffff	ffff
Description:	Move data		-	

Location 'f' can be anywhere in the 256-byte bank.

If 'a' is '0', the Access Bank is selected.

If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction

set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4	
Decode	Read	Process	Write	
	register 'f'	Data	register 'f'	

Example: MOVWF REG, 0

Before Instruction

W = 4FhREG = FFh

After Instruction

W = 4Fh REG = 4Fh

#### MULLW Multiply literal with W

Syntax: MULLW k Operands:  $0 \le k \le 255$ 

Operation: (W)  $x k \rightarrow PRODH:PRODL$ 

Status Affected: None

Encoding: 0000 1101 kkkk kkkk

Description:

An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in the PRODH:PRODL register

pair. PRODH contains the high byte. W is unchanged.

None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result

is possible but not detected.

Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write
	literal 'k'	Data	registers
			PRODH:
			PRODL

Example: MULLW 0C4h

Before Instruction

W = E2h PRODH = ? PRODL = ?

W = E2h PRODH = ADh

PRODL = 08h

### MULWF Multiply W with f

Syntax: MULWF  $f \{,a\}$ Operands:  $0 \le f \le 255$  $a \in [0,1]$ 

Operation: (W) x (f)  $\rightarrow$  PRODH:PRODL

Status Affected: None

Encoding: 0000 001a ffff ffff

Description:

An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are

unchanged.

None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used

to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset

Mode" for details.

Words: 1
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write
	register 'f'	Data	registers
			PRODH:
			PRODL

Example: MULWF REG, 1

Before Instruction

W = C4h
REG = B5h
PRODH = ?
PRODL = ?

After Instruction

W = C4h REG = B5h PRODH = 8Ah PRODL = 94h

**NEGF** Negate f NEGF f {,a} Syntax:  $0 \le f \le 255$ Operands:  $a \in [0,1]$  $(\overline{f}) + 1 \rightarrow f$ Operation: Status Affected: N, OV, C, DC, Z Encoding: 0110 ffff ffff 110a Description: Location 'f' is negated using two's

complement. The result is placed in the data memory location 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \le 95$  (5Fh). See Section 27.2.3 "Byte-Oriented and **Bit-Oriented Instructions in Indexed** Literal Offset Mode" for details.

Words: 1 1 Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4	
Decode	Read	Process	Write	
	register 'f'	Data	register 'f'	

Example: NEGF REG, 1

Before Instruction

REG 0011 1010 [3Ah]

After Instruction

1100 0110 [C6h] REG

NOF	•	No Operation					
Synta	ax:	NOP					
Oper	ands:	Ν	lone				
Oper	ation:	Ν	lo operatio	n			
Statu	s Affected:	Ν	lone				
Enco	ding:		0000	0000	000	0	0000
			1111	xxxx	XXX	X	XXXX
Desc	Description:		lo operation	n.			
Word	ls:	1					
Cycle	es:	1					
QC	ycle Activity:						
	Q1		Q2	Q3	3		Q4
	Decode		No	No	)		No
		0	peration	opera	tion	O	peration

Example:

None.

Top of Return Stack
)

Syntax: POP
Operands: None

Operation:  $(TOS) \rightarrow bit bucket$ 

Status Affected: None

Encoding: 0000 0000 0000 0110

Description:

The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack.
This instruction is provided to enable

the user to properly manage the return stack to incorporate a software stack.

Words: 1
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No	POP TOS	No
	operation	value	operation

Example: POP

GOTO NEW

Before Instruction

TOS = 0031A2h Stack (1 level down) = 014332h

After Instruction

TOS = 014332h PC = NEW PUSH Push Top of Return Stack

Syntax: PUSH Operands: None

Operation:  $(PC + 2) \rightarrow TOS$ 

Status Affected: None

**Encoding**: 0000 0000 0000 0101

The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.

Words: 1
Cycles: 1

Q Cycle Activity:

Description:

Q1	Q2	Q3	Q4
Decode	PUSH	No	No
	PC + 2 onto	operation	operation
	return stack		

Example: PUSH

Before Instruction

TOS = 345Ah PC = 0124h

After Instruction

PC = 0126h TOS = 0126h Stack (1 level down) = 345Ah

RCALL	Relative Call				
Syntax:	RCALL 1	RCALL n			
Operands:	-1024 ≤ n	$-1024 \le n \le 1023$			
Operation:	$(PC) + 2 \rightarrow TOS,$ $(PC) + 2 + 2n \rightarrow PC$				
Status Affected:	None				
Encoding:	1101 1nnn nnnn nnnn				
Description:	Subroutine call with a jump up to 1K from the current location. First, return				

address (PC + 2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a

two-cycle instruction.

Words: 1 Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
	PUSH PC to stack		
No	No	No	No
operation	operation	operation	operation

Example: HERE RCALL Jump

Before Instruction

PC = Address (HERE)

After Instruction

PC = Address (Jump) TOS = Address (HERE + 2)

RES	ET	Reset					
Synta	ax:	RESET					
Oper	ands:	None					
Oper	ation:		Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset.				
Statu	s Affected:	All					
Enco	ding:	0000	0000	111	L1	1111	
Desc	ription:	This instruction				•	
Word	ls:	1					
Cycles:		1	1				
Q C	ycle Activity:						
	Q1	Q2	Q3	1		Q4	
	Decode	Start	No	1		No	

Example: RESET

After Instruction

Registers = Reset Value Flags\* = Reset Value

Reset

operation

operation

### RETFIE Return from Interrupt

Syntax: RETFIE {s}

Operands:  $s \in [0,1]$ Operation:  $(TOS) \rightarrow PC$ ,

 $1 \rightarrow GIE/GIEH$  or PEIE/GIEL,

if s = 1 (WS)  $\rightarrow$  W,

 $(STATUSS) \rightarrow Status,$  $(BSRS) \rightarrow BSR,$ 

PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding: 0000 0000 0001 000s

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into

the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers accurate.

of these registers occurs.

Words: 1 Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No	No	POP PC
	operation	operation	from stack
			Set GIEH or
			GIEL
No	No	No	No
operation	operation	operation	operation

Example: RETFIE 1

After Interrupt

RETLW Return literal to W

Syntax: RETLW k

Operands:  $0 \le k \le 255$ Operation:  $k \to W$ ,  $(TOS) \to PC$ ,

PCLATU, PCLATH are unchanged

Status Affected: None

Encoding: 0000 1100 kkkk kkkk

Description: W is loaded with the eight-bit literal 'k'.

The program counter is loaded from the

top of the stack (the return address). The high address latch (PCLATH)

remains unchanged.

Words: 1
Cycles: 2

Q Cycle Activity:

_	Q1	Q2	Q3	Q4
	Decode	Read	Process	POP PC
		literal 'k'	Data	from stack,
				Write to W
	No	No	No	No
	operation	operation	operation	operation

#### Example:

CALL TABLE ; W contains table ; offset value ; W now has

; table value :

TABLE

ADDWF PCL ; W = offset
RETLW k0 ; Begin table

RETLW k1 ;

RETLW kn ; End of table

Before Instruction

W = 07h

After Instruction

W = value of kn

Dotate Laft & through Carry

### RETURN Return from Subroutine

 $\label{eq:syntax:} Syntax: RETURN \ \{s\}$   $Operands: \quad s \in [0,1]$   $Operation: \quad (TOS) \to PC, \\ if \ s = 1 \\ (WS) \to W, \\ (STATUSS) \to Status, \\ (BSRS) \to BSR, \\ PCLATU, PCLATH \ are \ unchanged$ 

Status Affected: None

Encoding:

Description:

Return from subroutine. The stack is popped and the top of the stack (TOS) is leaded into the program country. If

is loaded into the program counter. If 's'= 1, the contents of the shadow registers, WS, STATUSS and BSRS, are loaded into their corresponding registers, W, STATUS and BSR. If 's' = 0, no update of these registers

occurs.

Words: 1 Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No	Process	POP PC
	operation	Data	from stack
No	No	No	No
operation	operation	operation	operation

Example: RETURN

After Instruction: PC = TOS

RLCF	Rotate Left f through Carry					
Syntax:	RLCF f {,d {,a}}					
Operands:	$0 \le f \le 255$ d $\in [0,1]$ a $\in [0,1]$					
Operation:	(f <n>) → dest<n +="" 1="">, (f&lt;7&gt;) → C, (C) → dest&lt;0&gt;</n></n>					
Status Affected:	C, N, Z					
Encoding:	0011 01da ffff ffff					
	one bit to the left through the CARRY flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'.  If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.  C register f					

Words: 1
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

Example: RLCF REG, 0, 0

Before Instruction

REG = 1110 0110 C = 0

After Instruction

REG = 1110 0110 W = 1100 1100

C = 1

RLNCF	Rotate Left f (No Carry)					
Syntax:	RLNCF	f {,d {,a}}	}			
Operands:	$0 \le f \le 255$ d $\in [0,1]$ a $\in [0,1]$					
Operation:	, ,	$(f) \rightarrow dest,$ $(f<7>) \rightarrow dest<0>$				
Status Affected:	N, Z					
Encoding:	0100	01da	ffff	ffff		
Description:	one bit to t is placed ir stored bac If 'a' is '0', 1 If 'a' is '0', 1 GPR bank If 'a' is '0' a set is enab in Indexed mode when Section 27 Bit-Oriento	The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f'.  If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.  If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See  Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.				
		ieg	ister f			
Words:	1					
Cycles:	1					
Q Cycle Activity:						
Q1	Q2	Q3		Q4		
Decode	Read register 'f'	Proce Data		Vrite to stination		
Example:  Before Instruc	RLNCF	REG,	1, 0			
REG After Instruction	= 1010 1	.011				
After instruction	ווע					

RRCF	CF Rotate Right f through Carry					
Syntax:	RRCF f{,	d {,a}}				
Operands:	$\begin{array}{l} 0 \leq f \leq 255 \\ d \in [0,1] \\ a \in [0,1] \end{array}$	$d \in [0,1]$				
Operation:	$(f) \rightarrow dest,$ $(f<0>) \rightarrow C,$ $(C) \rightarrow dest<7>$					
Status Affected:	C, N, Z					
Encoding:	0011	00da	ffff	ffff		
Description:	one bit to the flag. If 'd' is '1', the register 'f'. If 'a' is '0', the fa' is '1', the GPR bank. If 'a' is '0' at set is enable in Indexed mode where Section 27 Bit-Oriente.	If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the				
	C	re	gister f			
Words:	1					
Cycles:	1					
Q Cycle Activity:						
Q1	Q2	Q3		Q4		
Decode	Read register 'f'	Proce Dat		Write to destination		
Example:	RRCF	REG,	0, 0			
Before Instruc						
REG C	= 1110 0 = 0	110				
After Instruction						

REG = 1110 0110 W = 0111 0011 C = 0

REG = 0101 0111

DS30684A-page 438

### Syntax: RRNCF $f \{ d \{,a\} \}$ Operands: $0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$ Operation: $(f \le n \ge ) \rightarrow dest \le n - 1 \ge n$

Rotate Right f (No Carry)

00da

Operation:  $(f < n >) \rightarrow dest < n - 1 >$ ,  $(f < 0 >) \rightarrow dest < 7 >$ 

Status Affected: N, Z

Encoding: 0100

Description:

**RRNCF** 

The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is

ffff

ffff

placed back in register 'f'.

If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as

per the BSR value.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.



Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

Example 1: RRNCF REG, 1, 0

Before Instruction

REG = 1101 0111

After Instruction

REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction

W = ?

REG = 1101 0111

After Instruction

W = 1110 1011 REG = 1101 0111

SETF	Set f
Syntax:	SETF f {,a}
Operands:	$0 \le f \le 255$ $a \in [0,1]$
Operation:	$FFh \to f$
Status Affected:	None
Encoding:	0110 100a ffff ffff
Description:	The contents of the specified register

are set to FFh.

If 'a' is '0', the Access Bank is selected.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words: 1
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write
	register 'f'	Data	register 'f'

Example: SETF REG, 1

Before Instruction

REG = 5Ah After Instruction

REG = FFh

#### **SLEEP Enter Sleep mode**

SLEEP Syntax: Operands: None  $00h \rightarrow WDT$ , Operation:  $0 \rightarrow WDT$  postscaler,  $1 \rightarrow \overline{TO}$ 

 $0 \rightarrow \overline{PD}$ TO, PD

Status Affected:

Encoding: 0000 0000 0000 0011 The Power-down Status bit  $(\overline{PD})$  is Description:

> cleared. The Time-out Status bit (TO) is set. The Watchdog Timer and its

postscaler are cleared.

The processor is put into Sleep mode

with the oscillator stopped.

Words: Cycles:

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No	Process	Go to
	operation	Data	Sleep

Example: SLEEP

> Before Instruction TO =

PD = ? After Instruction

TO = 1 + PD =

† If WDT causes wake-up, this bit is cleared.

SUBFWB	Subtract f	from W w	ith borrow
Syntax:	SUBFWB	f {,d {,a}}	

f {,d {,a}} Operands:  $0 \leq f \leq 255$  $d\,\in\,[0,1]$  $a \in \left[0,1\right]$ 

Operation:  $(W) - (f) - (\overline{C}) \rightarrow dest$ Status Affected: N, OV, C, DC, Z

Encoding: 01da ffff ffff 0101

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

register 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used

to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \le 95$  (5Fh). See **Section 27.2.3** "Byte-Oriented and Bit-Oriented **Instructions in Indexed Literal Offset** Mode" for details.

1 1

Cycles:

Q Cycle Activity:

Words:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

Example 1: SUBFWB REG, 1, 0 Before Instruction

3 2 W 1 After Instruction REG FF W 2 0 =

REG

C Z Ν ; result is negative

Example 2: SUBFWB REG, 0, 0

Before Instruction REG W = 5 After Instruction 2 REG 3 W = C = = 0

Ν ; result is positive = 0 Example 3: SUBFWB REG, 1, 0

Before Instruction REG W

2 С ō After Instruction 0 REG = 2

W C Z = ; result is zero = N

SUBLW	Subtract W from literal	SUBWF	Subtract W from f
Syntax:	SUBLW k	Syntax:	SUBWF f {,d {,a}}
Operands:	$0 \leq k \leq 255$	Operands:	$0 \leq f \leq 255$
Operation:	$k - (W) \rightarrow W$		$d \in [0,1]$
Status Affected:	N, OV, C, DC, Z	Operation	$a \in [0,1]$
Encoding:	0000 1000 kkkk kkkk	Operation:	$(f) - (W) \rightarrow dest$
Description	W is subtracted from the eight-bit	Status Affected:	N, OV, C, DC, Z
	literal 'k'. The result is placed in W.	Encoding:	0101   11da   ffff   ffff
Words:	1	Description:	Subtract W from register 'f' (2's complement method). If 'd' is '0', the
Cycles:	1		result is stored in W. If 'd' is '1', the
Q Cycle Activity:			result is stored back in register 'f'.
Q1	Q2 Q3 Q4		If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used
Decode	Read Process Write to W		to select the GPR bank.
	literal 'k' Data		If 'a' is '0' and the extended instruction
Example 1:	SUBLW 02h		set is enabled, this instruction operates in Indexed Literal Offset
Before Instruc			Addressing mode whenever
W C	= 01h = ?		f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented
After Instruction	on = 01h		Instructions in Indexed Literal Offse
С	= 1 ; result is positive		Mode" for details.
Z N	= 0 = 0	Words:	1
Example 2:	SUBLW 02h	Cycles:	1
Before Instruc	tion	Q Cycle Activity:	
W C	= 02h = ?	Q1	Q2 Q3 Q4
After Instruction	on	Decode	Read Process Write to
W C	= 00h = 1 ; result is zero		register 'f' Data destination
Ž N	= 1 = 0	Example 1:	SUBWF REG, 1, 0
Example 3:	SUBLW 02h	Before Instru REG	= 3
Before Instruc		W C	= 2 = ?
W	= 03h	After Instruct	ion
C After Instruction	= ?	REG W	= 1 = 2
W	= FFh ; (2's complement)	C Z	= 1 ; result is positive = 0
C Z	= 0 ; result is negative = 0	N	= 0
N	= 1	Example 2:	SUBWF REG, 0, 0
		Before Instru REG	ction = 2
		W	= 2
		C After Instruct	= ?
		REG W	= 2 = 0
		Ċ	= 1 ; result is zero
		Z N	= 1 = 0
		Example 3:	SUBWF REG, 1, 0
		Before Instru	
		REG W	= 1 = 2
		C After Instruct	= ?
		After Instruct REG W	= FFh ;(2's complement)

; result is negative

SUBWFB	Subtract	W from f with	Borrow	SWA	<b>\PF</b>	Swap f		
Syntax: SUBWFB f {,d {,a}}		Synt	ax:	SWAPF f	{,d {,a}}			
Operands: $ 0 \le f \le 255 $ $ d \in [0,1] $ $ a \in [0,1] $		Oper	rands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation: Status Affected:	(f) – (W) – ( N, OV, C, E	• •		Oper	ration:	$(f<3:0>) \rightarrow (f<7:4>) \rightarrow$		
Encoding:	0101	10da fff	f ffff	Statu	s Affected:	None		
Description:			•	Enco	oding:	0011	10da ff	ff ffff
Description:  Subtract W and the CAF (borrow) from register 'f' ment method). If 'd' is '0 stored in W. If 'd' is '1', t stored back in register 'f If 'a' is '0', the Access Bit 'a' is '1', the BSR is us GPR bank.  If 'a' is '0' and the extend set is enabled, this in Indexed Literal Offset mode whenever f ≤ 95 (section 27.2.3 "Byte-O Bit-Oriented Instructio Literal Offset Mode" for		od). If 'd' is '0', f'. If 'd' is '1', the k in register 'f'. the Access Banche BSR is used and the extende led, this instructions f ≤ 95 (5F'.2.3 "Byte-Oriced Instructions	he result is result is result is k is selected. to select the d instruction cion operates ddressing h). See ented and in Indexed		Description:		The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in register 'f'.  If 'a' is '0', the Access Bank is selecter If 'a' is '1', the BSR is used to select the GPR bank.  If 'a' is '0' and the extended instruction set is enabled, this instruction operater in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.	
Words:	1			Word	is.	1	101	dotallo.
Cycles:	1			Cycle	es:	1		
Q Cycle Activity:	00	02	04	•	ycle Activity:			
Q1 Decode	Q2 Read	Q3 Process	Q4 Write to	4.0	Q1	Q2	Q3	Q4
200040	register 'f'	Data	destination		Decode	Read	Process	Write to
Example 1:	SUBWFB	REG, 1, 0				register 'f'	Data	destination
Before Instruc								
REG W	= 19h = 0Dh	(0001 100 (0000 110		Exar	nple:	SWAPF R	EG, 1, 0	
C	= 1				Before Instruc			
After Instruction	on = 0Ch	(0000 110	0)		REG After Instruction	= 53h		
W C	= 0Dh = 1	(0000 110	1)		REG	= 35h		
Z	= 0							
N Example 2:	= 0	; result is po	sitive					
Example 2:  Before Instruc		REG, 0, 0						
REG W C	= 1Bh = 1Ah = 0	(0001 101 (0001 101						
After Instruction REG W C	on = 1Bh = 00h = 1	(0001 101	1)					
Z N	= 1 = 0	; result is ze	ro					
Example 3:	SUBWFB	REG, 1, 0						
Before Instruc REG W C	etion = 03h = 0Eh = 1	(0000 001 (0000 111						
After Instruction REG	on = F5h = 0Eh	(1111 010 ; <b>[2's comp]</b> (0000 111						
Č Z N	= 0 = 0 = 1	; result is ne						

#### **TBLRD Table Read**

Syntax: TBLRD ( \*; \*+; \*-; +\*)

Operands: None Operation: if TBLRD \*,

 $(Prog Mem (TBLPTR)) \rightarrow TABLAT;$ 

TBLPTR - No Change;

if TBLRD \*+,

 $(Prog Mem (TBLPTR)) \rightarrow TABLAT;$ 

 $(TBLPTR) + 1 \rightarrow TBLPTR;$ 

if TBLRD \*-

(Prog Mem (TBLPTR)) → TABLAT;

 $(TBLPTR) - 1 \rightarrow TBLPTR;$ 

if TBLRD +\*,

 $(TBLPTR) + 1 \rightarrow TBLPTR;$ 

 $(Prog Mem (TBLPTR)) \rightarrow TABLAT;$ 

Status Affected: None

Encoding:

0000	0000	0000	10nr	1
			nn=0	*
			=1	*+
			=2	* _
			=3	+*

Description:

This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table

Pointer (TBLPTR) is used.

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR

has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte

of Program Memory

Word

Most Significant Byte TBLPTR[0] = 1:

of Program Memory Word

The TBLRD instruction can modify the value

of TBLPTR as follows:

· no change

post-increment

post-decrement

pre-increment

Words: Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No	No	No
	operation	operation	operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TAB- LAT)

#### **TBLRD** Table Read (Continued)

Example1: TBLRD

Before Instruction

**TABLAT** 55h = 00A356h **TBLPTR** = MEMORY (00A356h) = 34h

After Instruction

**TABLAT** 34h **TBLPTR** 00A357h

Example2: TBLRD +\*

Before Instruction

TABLAT TBLPTR MEMORY (01A357h) MEMORY (01A358h) = AAh 01A357h = = 12h = 34h

After Instruction

**TABLAT** 34h **TBLPTR** 01A358h

TBLWT	Table Write				
Syntax:	TBLWT ( *; *+; *-; +*)				
Operands:	None				
Operation:	if TBLWT*,  (TABLAT) → Holding Register;  TBLPTR – No Change;  if TBLWT*+,  (TABLAT) → Holding Register;  (TBLPTR) + 1 → TBLPTR;  if TBLWT*-,  (TABLAT) → Holding Register;  (TBLPTR) - 1 → TBLPTR;				
	if TBLWT+*, (TBLPTR) + 1 → TBLPTR; (TABLAT) → Holding Register;				
Status Affected:	None				
Encoding:	0000 0000 11nn nn=0 * =1 *+ =2 *- =3 +*				
Description:	This instruction uses the three LSBs of TBLPTR to determine which of the eight holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to Section 7.0 "Flash Program Memory" for additional details on programming Flash memory.) The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-MByte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.  TBLPTR[0] = 0: Least Significant Byte of Program Memory Word TBLPTR[0] = 1: Most Significant Byte of Program Memory Word TBLPTR[0] = 1: TBLPTR as follows:				
	<ul><li>no change</li><li>post-increment</li><li>post-decrement</li><li>pre-increment</li></ul>				
Words:	1				
Cycles:	2				
Q Cycle Activity:	_				

Q1	Q2	Q3	Q4
Decode	No	No	No
	operation	operation	operation
No	No	No	No
operation	operation	operation	operation
	(Read		(Write to
	TABLAT)		Holding
			Register)

```
TBLWT
                  Table Write (Continued)
Example1:
                  TBLWT *+;
     Before Instruction
          TABLAT
TBLPTR
                                        55h
                                        00A356h
          HOLDING REGISTER
           (00A356h)
                                        FFh
     After Instructions (table write completion)
          TABLAT
                                        55h
          TBLPTR
HOLDING REGISTER
                                        00A357h
           (00A356h)
                                        55h
Example 2:
                  TBLWT +*;
     Before Instruction
          TABLAT
TBLPTR
HOLDING REGISTER
                                        34h
                                        01389Ah
                                        FFh
           (01389Ah)
          HOLDING REGISTER
                                        FFh
           (01389Bh)
     After Instruction (table write completion)
          TABLAT
                                        34h
          TBLPTR
HOLDING REGISTER
(01389Ah)
HOLDING REGISTER
(01389Bh)
                                        01389Bh
                                        FFh
                                        34h
```

#### **TSTFSZ** Test f, skip if 0

Syntax: TSTFSZ f {,a} Operands:  $0 \le f \le 255$  $a \in [0,1]$ 

Operation: skip if f = 0

Status Affected: None Encoding:

If 'f' = 0, the next instruction fetched Description:

0110

during the current instruction execution is discarded and a NOP is executed, making this a two-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

ffff

011a

ffff

GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever  $f \le 95$  (5Fh). See Section 27.2.3 "Byte-Oriented and **Bit-Oriented Instructions in Indexed** Literal Offset Mode" for details.

Words: 1 1(2) Cycles:

Note: Three cycles if skip and

followed by a 2-word

instruction.

#### Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	No
	register 'f'	Data	operation

If skip:

Q1	Q2	Q3	Q4
No	No	No	No
operation	operation	operation	operation

If skip and followed by 2-word instruction:

	Q1	Q2	Q3	Q4
ſ	No	No	No	No
	operation	operation	operation	operation
ſ	No	No	No	No
	operation	operation	operation	operation

Example: HERE TSTFSZ CNT, 1

> NZERO ZERO

Before Instruction

PC. Address (HERE)

After Instruction

If CNT 00h,

Address (ZERO)

00h,

If CNT PC Address (NZERO)

#### **XORLW** Exclusive OR literal with W

XORLW k Syntax: Operands:  $0 \le k \le 255$ Operation: (W) .XOR.  $k \rightarrow W$ 

Status Affected: N, Z

Encoding: 0000 1010 kkkk kkkk

Description: The contents of W are XORed with

the 8-bit literal 'k'. The result is placed

in W.

Words: Cycles:

Q Cycle Activity:

	Q1	Q2	Q3	Q4
Ī	Decode	Read	Process	Write to W
		literal 'k'	Data	

Example: XORLW 0AFh

Before Instruction

W B5h

After Instruction

W 1Ah

#### XORWF Exclusive OR W with f

Syntax: XORWF f {,d {,a}}

 $0 \le f \le 255$   $d \in [0,1]$  $a \in [0,1]$ 

Operation: (W) .XOR. (f)  $\rightarrow$  dest

Status Affected: N, Z

Operands:

Encoding: 0001 10da ffff ffff

Description: Exclusive OR the contents of W with

register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back

in the register 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the

GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Section 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.

Words: 1 Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read	Process	Write to
	register 'f'	Data	destination

Example: XORWF REG, 1, 0

Before Instruction

REG = AFh W = B5h

After Instruction

REG = 1AhW = B5h