

La...  [Try HackMD \(https://hackmd.io?utm\\_source=view-page&utm\\_medium=logo-nav\)](https://hackmd.io?utm_source=view-page&utm_medium=logo-nav)

# Lab 3 Instruction set

---

Reference Materials:


[datasheet](https://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf) (<https://ww1.microchip.com/downloads/en/DeviceDoc/39631E.pdf>).

[intruction set](https://technology.niagarac.on.ca/staff/mboldin/18F_Instruction_Set/) ([https://technology.niagarac.on.ca/staff/mboldin/18F\\_Instruction\\_Set/](https://technology.niagarac.on.ca/staff/mboldin/18F_Instruction_Set/)).

- Lab 3 Instruction set
  - status register
  - Bank Select Register (BSR)
- Overview
  - Instruction Structure
  - Instruction Categories
- Instruction Overview
  - Byte-oriented operations
  - Bit-oriented operations
  - Literal operations
  - Control operations

## status register

Records the status after an operation:

- N Negative
- OV Overflow
- Z Zero
- DC Digit Carry (0×0F  0×10)

- C Carry

## REGISTER 5-2: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC <sup>(1)</sup>	C <sup>(2)</sup>
bit 7						bit 0	

### Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit

This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).

1 = Result was negative

0 = Result was positive

bit 3 **OV:** Overflow bit

This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7) to change state.

1 = Overflow occurred for signed arithmetic (in this arithmetic operation)

0 = No overflow occurred

bit 2 **Z:** Zero bit

1 = The result of an arithmetic or logic operation is zero

0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit Carry/borrow bit<sup>(1)</sup>

For ADDWF, ADDLW, SUBLW and SUBWF instructions:

1 = A carry-out from the 4th low-order bit of the result occurred

0 = No carry-out from the 4th low-order bit of the result

bit 0 **C:** Carry/borrow bit<sup>(2)</sup>

For ADDWF, ADDLW, SUBLW and SUBWF instructions:

1 = A carry-out from the Most Significant bit of the result occurred

0 = No carry-out from the Most Significant bit of the result occurred

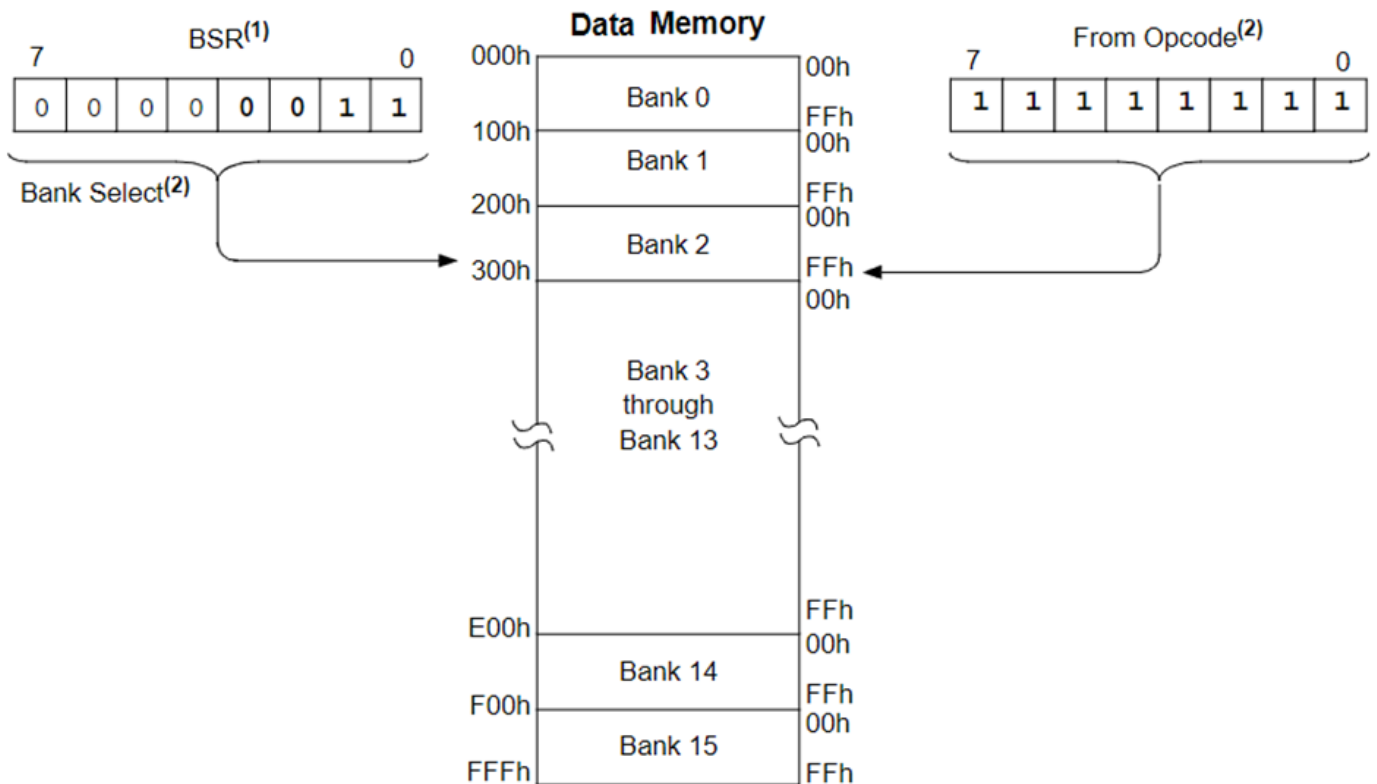
**Note 1:** For borrow, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either bit 4 or bit 3 of the source register.

**2:** For borrow, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low-order bit of the source register.

## Bank Select Register (BSR)

- Data Memory: 12-bit addressing space.
- In most instructions, the operand used for addressing is 8-bit.
- The BSR is used to record the bank:

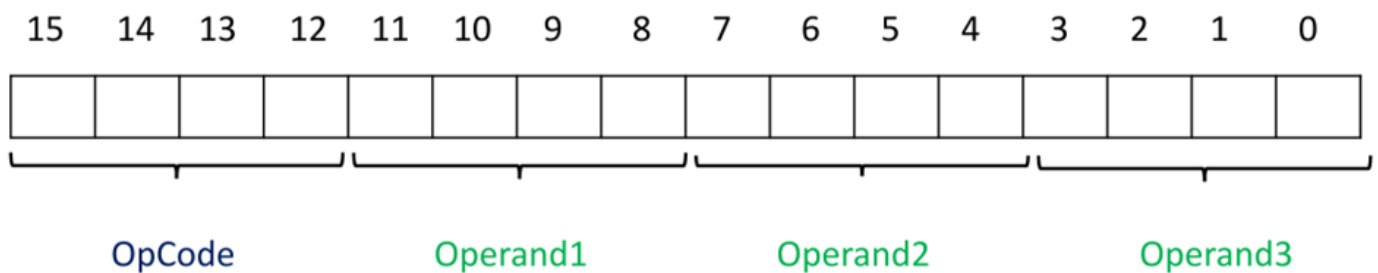
- Determine the bank
- Determine the offset



## Overview

### Instruction Structure

- Most instructions in the PIC18F4520 consist of 1 word (2 bytes).
- Each instruction can be divided into OpCode and Operands.
- Operands can be addresses, parameters, or constants.



## Instruction Categories

- Byte-oriented
  - Operates on whole bytes. This category includes arithmetic, logic operations, and conditional skip instructions.
- Bit-oriented
  - Operates on specific bits within a byte, including clear, set, and flip operations.
- Literal
  - Uses constants as operands, directly operating with registers.
- Control
  - Controls the flow of the program, primarily through various branch instructions.

## Instruction Overview

---

### Byte-oriented operations

- MOVF: Move data from a file register to the WREG or back to the original file register.
- SUBWF: Subtract the WREG contents from a file register. (F - W)
- COMF: Complement the contents of a file register.

BYTE-ORIENTED 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, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	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 <sub>s</sub> , f <sub>d</sub>	Move f <sub>s</sub> (source) to 1st word f <sub>d</sub> (destination) 2nd word	2	1100	ffff	ffff	ffff	None	
				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	



ADDWF	ADD W to f				
Syntax:	ADDWF f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(W) + (f) \rightarrow \text{dest}$				
Status Affected:	N, OV, C, DC, Z				
Encoding:	<table><tr><td>0010</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>	0010	01da	ffff	ffff
0010	01da	ffff	ffff		
Description:	<p>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' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>				
Words:	1				
Cycles:	1				

ANDWF	AND W with f				
Syntax:	ANDWF f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	(W) .AND. (f) $\rightarrow$ dest				
Status Affected:	N, Z				
Encoding:	<table><tr><td>0001</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table>	0001	01da	ffff	ffff
0001	01da	ffff	ffff		
Description:	<p>The contents of W are ANDed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>				
Words:	1				
Cycles:	1				

DECFSZ	Decrement f, skip if 0
Syntax:	[label] DECFSZ f [,d [,a]]
Operands:	$0 \leq f \leq 255$

ADDWFC	ADD W and Carry bit to f				
Syntax:	ADDWFC f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	$(W) + (f) + (C) \rightarrow \text{dest}$				
Status Affected:	N,OV, C, DC, Z				
Encoding:	<table border="1"><tr><td>0010</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>	0010	00da	ffff	ffff
0010	00da	ffff	ffff		
Description:	<p>Add W, the Carry flag and data memory 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 (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>				
Words:	1				
Cycles:	1				

XORWF	Exclusive OR W with f				
Syntax:	XORWF f {,d {,a}}				
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$				
Operation:	(W) .XOR. (f) → dest				
Status Affected:	N, Z				
Encoding:	<table border="1"><tr><td>0001</td><td>10da</td><td>ffff</td><td>ffff</td></tr></table>	0001	10da	ffff	ffff
0001	10da	ffff	ffff		
Description:	<p>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' (default).</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>				
Words:	1				
Cycles:	1				

CPFSEQ	Compare f with W, skip if f = W
Syntax:	[label] CPFSEQ f [,a]
Operands:	$0 \leq f \leq 255$



```

Example:      HERE      DECFSZ      CNT, 1, 1
              GOTO      LOOP
              CONTINUE

Before Instruction
      PC      =      Address (HERE)

After Instruction
      CNT      =      CNT - 1
      If CNT   =      0;
      PC      =      Address (CONTINUE)
      If CNT   ≠      0;
      PC      =      Address (HERE+2)

```

Example:            HERE        CPFSEQ REG, 0  
                  NEQUAL        :  
                  EQUAL        :

Before Instruction		
PC Address	=	HERE
W	=	?
REG	=	?
After Instruction		
If REG	=	W;
PC	=	Address (EQUAL)
If REG	≠	W;
PC	=	Address (UNEQUAL)



**MOVWF****Move W to f**

Syntax:	MOVWF	f {,a}				
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$					
Operation:	$(W) \rightarrow f$					
Status Affected:	None					
Encoding:	<table border="1"><tr><td>0110</td><td>111a</td><td>ffff</td><td>ffff</td></tr></table>		0110	111a	ffff	ffff
0110	111a	ffff	ffff			
Description:	<p>Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>					
Words:	1					
Cycles:	1					

**MULWF****Multiply W with f**

Syntax:	MULWF f{,a}				
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$				
Operation:	$(W) \times (f) \rightarrow \text{PRODH:PRODL}$				
Status Affected:	None				
Encoding:	<table border="1"><tr><td>0000</td><td>001a</td><td>ffff</td><td>ffff</td></tr></table>	0000	001a	ffff	ffff
0000	001a	ffff	ffff		
Description:	<p>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.</p> <p>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.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.</p>				
Words:	1				
Cycles:	1				

**RLNCF Rotate Left f (No Carry)**

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

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$ Operation:  $(f < n) \rightarrow \text{dest} < n + 1 >$ ,  
 $(f < 7 >) \rightarrow \text{dest} < 0 >$ 

Status Affected: N, Z

Encoding: 

0100	01da	ffff	ffff
------	------	------	------

Description: 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' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). 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 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

**RLCF Rotate Left f through Carry**

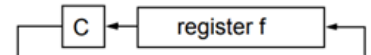
Syntax: RLCF f{,d{,a}}

Operands:  $0 \leq f \leq 255$   
 $d \in [0,1]$   
 $a \in [0,1]$ Operation:  $(f < n) \rightarrow \text{dest} < n + 1 >$ ,  
 $(f < 7 >) \rightarrow C$ ,  
 $(C) \rightarrow \text{dest} < 0 >$ 

Status Affected: C, N, Z

Encoding: 

0011	01da	ffff	ffff
------	------	------	------

Description: The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). 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 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

## Bit-oriented operations

- BCF: Bit Clear f, clears a specific bit in a file register.
- BSF: Bit Set f, sets a specific bit in a file register.
- BTFSC: Bit Test f, Skip if Clear; tests a bit and skips the next instruction if the bit is clear.
- BTFSS: Bit Test f, Skip if Set; tests a bit and skips the next instruction if the bit is set.

**BIT-ORIENTED OPERATIONS**

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, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2

BCF	Bit Clear f				
Syntax:	BCF f, b {,a}				
Operands:	$0 \leq f \leq 255$ $0 \leq b \leq 7$ $a \in [0,1]$				
Operation:	$0 \rightarrow f[b]$				
Status Affected:	None				
Encoding:	<table><tr><td>1001</td><td>bbba</td><td>ffff</td><td>ffff</td></tr></table>	1001	bbba	ffff	ffff
1001	bbba	ffff	ffff		
Description:	Bit 'b' in register 'f' is cleared. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default). 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 <b>Section 24.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"</b> for details.				
Words:	1				
Cycles:	1				

BSF		Bit Set f							
Syntax:	BSF f, b {,a}								
Operands:	$0 \leq f \leq 255$								
	$0 \leq b \leq 7$								
	$a \in [0,1]$								
Operation:	$1 \rightarrow f[b]$								
Status Affected:	None								
Encoding:	<table border="1"><tr><td>1000</td><td>bbba</td><td>ffff</td><td>ffff</td></tr></table>					1000	bbba	ffff	ffff
1000	bbba	ffff	ffff						
Description:	<p>Bit 'b' in register 'f' is set.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh). See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>								
Words:	1								
Cycles:	1								

BTFSC	Bit Test File, Skip if Clear				
Syntax:	BTFSC f, b {,a}				
Operands:	$0 \leq f \leq 255$ $0 \leq b \leq 7$ $a \in [0,1]$				
Operation:	skip if $(f[b]) = 0$				
Status Affected:	None				
Encoding:	<table><tr><td>1011</td><td>bbba</td><td>ffff</td><td>ffff</td></tr></table>	1011	bbba	ffff	ffff
1011	bbba	ffff	ffff		
Description:	<p>If bit 'b' in register 'f' is '0', then the next instruction is skipped. If bit 'b' is '0', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh).</p> <p>See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>				
Words:	1				
Cycles:	1(2)				

BTFSS	Bit Test File, Skip if Set				
Syntax:	BTFSS f, b {,a}				
Operands:	$0 \leq f \leq 255$ $0 \leq b < 7$ $a \in [0,1]$				
Operation:	skip if $(f[b]) = 1$				
Status Affected:	None				
Encoding:	<table border="1"><tr><td>1010</td><td>bbba</td><td>ffff</td><td>ffff</td></tr></table>	1010	bbba	ffff	ffff
1010	bbba	ffff	ffff		
Description:	<p>If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a two-cycle instruction.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever <math>f \leq 95</math> (5Fh).</p> <p>See <b>Section 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”</b> for details.</p>				
Words:	1				
Cycles:	1(2)				

## BTG Bit Toggle f

Syntax: BTG f, b {,a}

Operands:  $0 \leq f \leq 255$   
 $0 \leq b < 7$   
 $a \in [0,1]$

Operation:  $\overline{(f \ll b)} \rightarrow f \ll b$

Status Affected: None

Encoding:

0111	bbba	ffff	ffff
------	------	------	------

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 (default).

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 24.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

## Literal operations

- ADDLW     Add Literal and WREG
- LFSR       Move Literal (12-bit) 2nd word to FSR(f) 1st word

LITERAL OPERATIONS									
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 to FSR(f) 1st word	2	1110	1110	00ff	kkkk	None	
				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	

### MOVLW Move literal to W

Syntax:	[ <i>label</i> ] MOVLW k				
Operands:	$0 \leq k \leq 255$				
Operation:	$k \rightarrow W$				
Status Affected:	None				
Encoding:	<table><tr><td>0000</td><td>1110</td><td>kkkk</td><td>kkkk</td></tr></table>	0000	1110	kkkk	kkkk
0000	1110	kkkk	kkkk		
Description:	The eight-bit literal 'k' is loaded into W.				
Words:	1				
Cycles:	1				

### MOVLB Move Literal to Low Nibble in BSR

Syntax:	MOVLW k				
Operands:	$0 \leq k \leq 255$				
Operation:	$k \rightarrow \text{BSR}$				
Status Affected:	None				
Encoding:	<table border="1"><tr><td>0000</td><td>0001</td><td>kkkk</td><td>kkkk</td></tr></table>	0000	0001	kkkk	kkkk
0000	0001	kkkk	kkkk		
Description:	The 8-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '0', regardless of the value of $k_7:k_4$ .				
Words:	1				
Cycles:	1				

## Control operations

- Unconditional Branch
  - BRA Branch Unconditionally
  - GOTO Go to Address 1st word 2nd word
- Conditional Branch
  - BC Branch if Carry
  - BN Branch if Negative
  - BNC Branch if Not Carry
  - BNOV Branch if Not Overflow
  - BNZ Branch if Not Zero
  - BOV Branch if Overflow
  - BZ Branch if Zero





CONTROL OPERATIONS									
BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None	4
BN	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	
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None	
CALL	n, s	Call Subroutine 1st word	2	1110	110s	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk	None	
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	0100	TO, PD	
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	0111	C	
GOTO	n	Go to Address 1st word	2	1110	1111	kkkk	kkkk	None	
		2nd word		1111	kkkk	kkkk	kkkk	None	
NOP	—	No Operation	1	0000	0000	0000	0000	None	
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None	
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	

### BRA Unconditional Branch

Syntax:	BRA    n			
Operands:	$-1024 \leq n \leq 1023$			
Operation:	$(PC) + 2 + 2n \rightarrow PC$			
Status Affected:	None			
Encoding:	1101	0nnn	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			

### GOTO Unconditional Branch

Syntax:	GOTO k								
Operands:	$0 \leq k \leq 1048575$								
Operation:	$k \rightarrow PC<20:1>$								
Status Affected:	None								
Encoding:	<table border="1"><tr><td>1110</td><td>1111</td><td><math>k_7</math>kkk</td><td>kkkk<sub>0</sub></td></tr><tr><td>1111</td><td><math>k_{19}</math>kkk</td><td>kkkk</td><td>kkkk<sub>8</sub></td></tr></table>	1110	1111	$k_7$ kkk	kkkk <sub>0</sub>	1111	$k_{19}$ kkk	kkkk	kkkk <sub>8</sub>
1110	1111	$k_7$ kkk	kkkk <sub>0</sub>						
1111	$k_{19}$ kkk	kkkk	kkkk <sub>8</sub>						
Description:	<p>GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC&lt;20:1&gt;.</p> <p>GOTO is always a two-cycle instruction.</p>								
Words:	2								
Cycles:	2								

### BC Branch if Carry

Syntax:	BC    n				
Operands:	$-128 \leq n \leq 127$				
Operation:	if Carry bit is '1', $(PC) + 2 + 2n \rightarrow PC$				
Status Affected:	None				
Encoding:	<table border="1"><tr><td>1110</td><td>0010</td><td>nnnn</td><td>nnnn</td></tr></table>	1110	0010	nnnn	nnnn
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				

Q Cycle Activity:  
If Jump:

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

If No Jump:

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

Example:                      HERE                      BC                      5



incremented to return the next instruction, the new address will be  $PC + 2 + 2n$ . This instruction is then a two-cycle instruction.

Words: 1  
Cycles: 1(2)

Before instruction  
PC = address (HERE)  
After Instruction  
If Carry = 1;  
PC = address (HERE + 12)  
If Carry = 0;  
PC = address (HERE + 2)