9.1 Instruction Descriptions

ADDLW — **Add Literal and W**

Syntax: [label] ADDLW k
Operands: 0 ≤ k ≤ 255
Operation: (W) + k → (W)
Status Affected: C, DC, Z
Encoding:

11	111x	kkkk	kkkk

Description: The contents of the W register are added to the eight bit literal 'k' and the result is placed in the W register.
Words: 1
Cycles: 1
Q Cycle Activity:

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

Example:
```
ADDLW   0x15
```
Before Instruction
W = 0x10
After Instruction
W = 0x25

ANDLW — **AND Literal with W**

Syntax: [label] ANDLW k
Operands: 0 ≤ k ≤ 255
Operation: (W) .AND. (k) → (W)
Status Affected: Z
Encoding:

11	1001	kkkk	kkkk

Description: The contents of W register are AND'ed with the eight bit literal 'k'. The result is placed in the W register.
Words: 1
Cycles: 1
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal "k"	Process data	Write to W

Example
```
ANDLW   0x5F
```
Before Instruction
W = 0xA3
After Instruction
W = 0x03

ADDWF — **Add W and f**

Syntax: [label] ADDWF f,d
Operands: 0 ≤ f ≤ 127
d ∈ [0,1]
Operation: (W) + (f) → (destination)
Status Affected: C, DC, Z
Encoding:

00	0111	dfff	ffff

Description: Add the contents of the W register with the contents of register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words: 1
Cycles: 1
Q Cycle Activity:

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

Example
```
ADDWF   FSR, 0
```
Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0xD9
FSR = 0xC2

ANDWF — **AND W with f**

Syntax: [label] ANDWF f,d
Operands: 0 ≤ f ≤ 127
d ∈ [0,1]
Operation: (W) .AND. (f) → (destination)
Status Affected: Z
Encoding:

00	0101	dfff	ffff

Description: AND the W register with contents of register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.
Words: 1
Cycles: 1
Q Cycle Activity:

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

Example
```
ANDWF   FSR, 1
```
Before Instruction
W = 0x17
FSR = 0xC2
After Instruction
W = 0x17
FSR = 0x02

BCF — **Bit Clear f**

Syntax: [label] BCF f,b
Operands: 0 ≤ f ≤ 127
0 ≤ b ≤ 7
Operation: 0 → (f<b>)
Status Affected: None
Encoding:

01	00bb	bfff	ffff

Description: Bit 'b' in register 'f' is cleared.
Words: 1
Cycles: 1
Q Cycle Activity:

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

Example
```
BCF     FLAG_REG, 7
```
Before Instruction
FLAG_REG = 0xC7
After Instruction
FLAG_REG = 0x47

BSF — **Bit Set f**

Syntax: [label] BSF f,b
Operands: 0 ≤ f ≤ 127
0 ≤ b ≤ 7
Operation: 1 → (f<b>)
Status Affected: None
Encoding:

01	01bb	bfff	ffff

Description: Bit 'b' in register 'f' is set.
Words: 1
Cycles: 1
Q Cycle Activity:

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

Example
```
BSF     FLAG_REG,   7
```
Before Instruction
FLAG_REG = 0x0A
After Instruction
FLAG_REG = 0x8A

BTFSC — **Bit Test, Skip if Clear**

Syntax: [label] BTFSC f,b
Operands: 0 ≤ f ≤ 127
0 ≤ b ≤ 7
Operation: skip if (f<b>) = 0
Status Affected: None
Encoding:

01	10bb	bfff	ffff

Description: If bit 'b' in register 'f' is '1' then the next instruction is executed.
If bit 'b', in register 'f', is '0' then the next instruction is discarded, and a NOP is executed instead, making this a 2TCY instruction.
Words: 1
Cycles: 1(2)
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	No-Operation

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

Example
```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```
Before Instruction
PC = address HERE
After Instruction
if FLAG<1> = 0,
PC = address TRUE
if FLAG<1>=1,
PC = address FALSE

BTFSS — **Bit Test f, Skip if Set**

Syntax: [label] BTFSS f,b
Operands: 0 ≤ f ≤ 127
0 ≤ b < 7
Operation: skip if (f<b>) = 1
Status Affected: None
Encoding:

01	11bb	bfff	ffff

Description: If bit 'b' in register 'f' is '0' then the next instruction is executed.
If bit 'b' is '1', then the next instruction is discarded and a NOP is executed instead, making this a 2TCY instruction.
Words: 1
Cycles: 1(2)
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process data	No-Operation

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

Example
```
HERE    BTFSC   FLAG,1
FALSE   GOTO    PROCESS_CODE
TRUE    •
        •
        •
```
Before Instruction
PC = address HERE
After Instruction
if FLAG<1> = 0,
PC = address FALSE
if FLAG<1> = 1,
PC = address TRUE

CALL — **Call Subroutine**

Syntax: [label] CALL k
Operands: 0 ≤ k ≤ 2047
Operation: (PC)+ 1→ TOS,
k → PC<10:0>,
(PCLATH<4:3>)→ PC<12:11>
Status Affected: None
Encoding:

10	0kkk	kkkk	kkkk

Description: Call Subroutine. First, return address (PC+1) is pushed onto the stack. The eleven bit immediate address is loaded into PC bits <10:0>. The upper bits of the PC are loaded from PCLATH.CALL is a two cycle instruction.
Words: 1
Cycles: 2
Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	Read literal 'k', Push PC to Stack	Process data	Write to PC
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example
```
HERE    CALL    THERE
```
Before Instruction
PC = Address HERE
After Instruction
PC = Address THERE
TOS = Address HERE+1

CLRF — **Clear f**

Syntax: [label] CLRF f
Operands: 0 ≤ f ≤ 127
Operation: 00h → (f)
1 → Z
Status Affected: Z
Encoding:

00	0001	lfff	ffff

Description: The contents of register 'f' are cleared and the Z bit is set.
Words: 1
Cycles: 1
Q Cycle Activity:

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

Example
```
CLRF    FLAG_REG
```
Before Instruction
FLAG_REG = 0x5A
After Instruction
FLAG_REG = 0x00
Z = 1

CLRW — **Clear W**

Syntax: [label] CLRW
Operands: None
Operation: 00h → (W)
1 → Z
Status Affected: Z
Encoding:

00	0001	0xxx	xxxx

Description: W register is cleared. Zero bit (Z) is set.
Words: 1
Cycles: 1
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No-Operation	Process data	Write to W

Example
```
CLRW
```
Before Instruction
W = 0x5A
After Instruction
W = 0x00
Z = 1

CLRWDT — **Clear Watchdog Timer**

Syntax: [label] CLRWDT
Operands: None
Operation: 00h → WDT
0 → WDT prescaler,
1 → $\overline{TO}$
1 → $\overline{PD}$
Status Affected: $\overline{TO}$, $\overline{PD}$
Encoding:

00	0000	0110	0100

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the prescaler of the WDT. Status bits $\overline{TO}$ and $\overline{PD}$ are set.
Words: 1
Cycles: 1
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No-Operation	Process data	Clear WDT Counter

Example
```
CLRWDT
```
Before Instruction
WDT counter = ?
After Instruction
WDT counter = 0x00
WDT prescaler= 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF **Complement f**

Syntax: [label] COMF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f̄) → (destination)

Status Affected: Z

Encoding:

00	1001	dfff	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'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example
```
COMF    REG1,0
```
Before Instruction
REG1 = 0x13
After Instruction
REG1 = 0x13
W = 0xEC

DECF **Decrement f**

Syntax: [label] DECF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) - 1 → (destination)

Status Affected: Z

Encoding:

00	0011	dfff	ffff

Description: Decrement contents of register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example
```
DECF    CNT, 1
```
Before Instruction
CNT = 0x01
Z = 0
After Instruction
CNT = 0x00
Z = 1

DECFSZ **Decrement f, Skip if 0**

Syntax: [label] DECFSZ f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) - 1 → (destination); skip if result = 0

Status Affected: None

Encoding:

00	1011	dfff	ffff

Description: The contents of register 'f' are decremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is not 0
If the result is 1, the next instruction, is executed. If the result is 0, then a NOP is executed instead making it a 2TCY instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

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

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

Example
```
HERE    DECFSZ  CNT, 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+1

GOTO **Unconditional Branch**

Syntax: [label] GOTO k

Operands: 0 ≤ k ≤ 2047

Operation: k → PC<10:0>
PCLATH<4:3> → PC<12:11>

Status Affected: None

Encoding:

10	1kkk	kkkk	kkkk

Description: GOTO is an unconditional branch. The eleven bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH<4:3>. GOTO is a two cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	Read literal 'k'	Process data	Write to PC
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example
```
GOTO THERE
```
After Instruction
PC = Address THERE

INCF **Increment f**

Syntax: [label] INCF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) + 1 → (destination)

Status Affected: Z

Encoding:

00	1010	dfff	ffff

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example
```
INCF    CNT, 1
```
Before Instruction
CNT = 0xFF
Z = 0
After Instruction
CNT = 0x00
Z = 1

INCFSZ **Increment f, Skip if 0**

Syntax: [label] INCFSZ f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) + 1 → (destination), skip if result = 0

Status Affected: None

Encoding:

00	1111	dfff	ffff

Description: The contents of register 'f' are incremented. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.
If the result is not 0
If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead making it a 2TCY instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

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

If Skip: (2nd Cycle)

Q1	Q2	Q3	Q4
No-Operation	No-Operation	No-Operation	No-Operation

Example
```
HERE    INCFSZ  CNT, 1
        GOTO    LOOP
CONTINUE •
         •
         •
```
Before Instruction
PC = address HERE
After Instruction
CNT = CNT + 1
if CNT = 0,
PC = address CONTINUE
if CNT ≠ 0,
PC = address HERE +1

IORLW **Inclusive OR Literal with W**

Syntax: [label] IORLW k

Operands: 0 ≤ k ≤ 255

Operation: (W) .OR. k → (W)

Status Affected: Z

Encoding:

11	1000	kkkk	kkkk

Description: The contents of the W register is OR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example
```
IORLW   0x35
```
Before Instruction
W = 0x9A
After Instruction
W = 0xBF
Z = 1

Z = 0 !!!

IORWF **Inclusive OR W with f**

Syntax: [label] IORWF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (W) .OR. (f) → (destination)

Status Affected: Z Z !!!

Encoding:

00	0100	dfff	ffff

Description: Inclusive OR the W register with contents of register 'f'. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example
```
IORWF   RESULT, 0
```
Before Instruction
RESULT = 0x13
W = 0x91
After Instruction
RESULT = 0x13
W = 0x93
Z = 1

Z = 0 !!!

MOVF **Move f**

Syntax: [label] MOVF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) → (destination)

Status Affected: Z

Encoding:

00	1000	dfff	ffff

Description: The contents of register f is moved to a destination dependant upon the status of d. If d = 0, destination is W register. If d = 1, the destination is file register f itself. d = 1 is useful to test a file register since status flag Z is affected.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example
```
MOVF    FSR, 0
```
After Instruction
W = value in FSR register
Z = 1

MOVLW **Move Literal to W**

Syntax: [label] MOVLW k

Operands: 0 ≤ k ≤ 255

Operation: k → (W)

Status Affected: None

Encoding:

11	00xx	kkkk	kkkk

Description: The eight bit literal 'k' is loaded into W register. The don't cares will assemble as 0's.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example
```
MOVLW   0x5A
```
After Instruction
W = 0x5A

MOVWF **Move W to f**

Syntax: [label] MOVWF f

Operands: 0 ≤ f ≤ 127

Operation: (W) → (f)

Status Affected: None

Encoding:

00	0000	1fff	ffff

Description: Move data from W register to register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example
```
MOVWF   OPTION_REG
```
Before Instruction
OPTION = 0xFF
W = 0x4F
After Instruction
OPTION = 0x4F
W = 0x4F

NOP — **No Operation**

Syntax: [label] NOP
Operands: None
Operation: No operation
Status Affected: None
Encoding:

00	0000	0xx0	0000

Description: No operation.
Words: 1
Cycles: 1
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No-Operation	No-Operation	No-Operation

Example

```
NOP
```

RETFIE — **Return from Interrupt**

Syntax: [label] RETFIE
Operands: None
Operation: TOS → PC,
1 → GIE
Status Affected: None
Encoding:

00	0000	0000	1001

Description: Return from Interrupt. Stack is POPed and Top of Stack (TOS) is loaded in the PC. Interrupts are enabled by setting Global Interrupt Enable bit, GIE (INTCON<7>). This is a two cycle instruction.
Words: 1
Cycles: 2
Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	No-Operation	Set the GIE bit	Pop from the Stack
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example

```
RETFIE
```

After Interrupt
PC = TOS
GIE = 1

OPTION — **Load Option Register**

Syntax: [label] OPTION
Operands: None
Operation: (W) → OPTION
Status Affected: None
Encoding:

00	0000	0110	0010

Description: The contents of the W register are loaded in the OPTION register. This instruction is supported for code compatibility with PIC16C5X products. Since OPTION is a readable/writable register, the user can directly address it.
Words: 1
Cycles: 1
Example

To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

RETLW — **Return with Literal in W**

Syntax: [label] RETLW k
Operands: 0 ≤ k ≤ 255
Operation: k → (W);
TOS → PC
Status Affected: None
Encoding:

11	01xx	kkkk	kkkk

Description: The W register is loaded with the eight bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two cycle instruction.
Words: 1
Cycles: 2
Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	Read literal 'k'	No-Operation	Write to W, Pop from the Stack
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example

```
CALL TABLE  ;W contains table
            ;offset value
•           ;W now has table value
•
•
TABLE ADDWF PC   ;W = offset
      RETLW k1   ;Begin table
      RETLW k2   ;
•
•
•
      RETLW kn   ; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of k8

RETURN — **Return from Subroutine**

Syntax: [label] RETURN
Operands: None
Operation: TOS → PC
Status Affected: None
Encoding:

00	0000	0000	1000

Description: Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two cycle instruction.
Words: 1
Cycles: 2
Q Cycle Activity:

	Q1	Q2	Q3	Q4
1st Cycle	Decode	No-Operation	No-Operation	Pop from the Stack
2nd Cycle	No-Operation	No-Operation	No-Operation	No-Operation

Example

```
RETURN
```

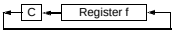
After Interrupt
PC = TOS

RLF — **Rotate Left f through Carry**

Syntax: [label] RLF f,d
Operands: 0 ≤ f ≤ 127
d ∈ [0,1]
Operation: See description below
Status Affected: C
Encoding:

00	1101	dfff	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 the W register. If 'd' is 1 the result is stored back in register 'f'.



Words: 1
Cycles: 1
Q Cycle Activity:

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

Example

```
RLF     REG1,0
```

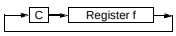
Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 1100 1100
C = 1

RRF — **Rotate Right f through Carry**

Syntax: [label] RRF f,d
Operands: 0 ≤ f ≤ 127
d ∈ [0,1]
Operation: See description below
Status Affected: C
Encoding:

00	1100	dfff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the W register. If 'd' is 1 the result is placed back in register 'f'.



Words: 1
Cycles: 1
Q Cycle Activity:

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

Example

```
RRF     REG1,0
```

Before Instruction
REG1 = 1110 0110
C = 0
After Instruction
REG1 = 1110 0110
W = 0111 0011
C = 0

SLEEP

Syntax: [label] SLEEP
Operands: None
Operation: 00h → WDT,
0 → WDT prescaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$
Status Affected: $\overline{TO}$, $\overline{PD}$
Encoding:

00	0000	0110	0011

Description: The power-down status bit, $\overline{PD}$ is cleared. Time-out status bit, $\overline{TO}$ is set. Watchdog Timer and its prescaler are cleared.
The processor is put into SLEEP mode with the oscillator stopped. See Section 14.8 for more details.
Words: 1
Cycles: 1
Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No-Operation	No-Operation	Go to Sleep

Example:

```
SLEEP
```

SUBLW — **Subtract W from Literal**

Syntax: [label] SUBLW k
Operands: 0 ≤ k ≤ 255
Operation: k - (W) → (W)
Status Affected: C, DC, Z
Encoding:

11	110x	kkkk	kkkk

Description: The contents of W register is subtracted (2's complement method) from the eight bit literal 'k'. The result is placed in the W register.
Words: 1
Cycles: 1
Q Cycle Activity:

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

Example 1:

```
SUBLW           0x02
```

Before Instruction
W = 1
C = ?
Z = ?
After Instruction
W = 1
C = 1; result is positive
Z = 0

Example 2: Before Instruction
W = 2
C = ?
Z = ?
After Instruction
W = 0
C = 1; result is zero
Z = 1

Example 3: Before Instruction
W = 3
C = ?
Z = ?
After Instruction
W = 0xFF
C = 0; result is negative
Z = 0

SUBWF **Subtract W from f**

Syntax: [label] SUBWF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f) - (W) → (destination)

Status Affected: C, DC, Z

Encoding:

00	0010	dfff	ffff

Description: Subtract (2's complement method) contents of W register from register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: `SUBWF REG1,1`

Before Instruction
REG1 = 3
W = 2
C = ?
Z = ?

After Instruction
REG1 = 1
W = 2
C = 1; result is positive
Z = 0

Example 2: Before Instruction
REG1 = 2
W = 2
C = ?
Z = ?

After Instruction
REG1 = 0
W = 2
C = 1; result is zero
Z = 1

Example 3: Before Instruction
REG1 = 1
W = 2
C = ?
Z = ?

After Instruction
REG1 = 0xFF
W = 2
C = 0; result is negative
Z = 0

SWAPF **Swap Nibbles in f**

Syntax: [label] SWAPF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (f<3:0>) → (destination<7:4>),
(f<7:4>) → (destination<3:0>)

Status Affected: None

Encoding:

00	1110	dfff	ffff

Description: The upper and lower nibbles of contents of register 'f' are exchanged. If 'd' is 0 the result is placed in W register. If 'd' is 1 the result is placed in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example `SWAPF REG, 0`

Before Instruction
REG1 = 0xA5

After Instruction
REG1 = 0xA5
W = 0x5A

TRIS **Load TRIS Register**

Syntax: [label] TRIS f

Operands: 5 ≤ f ≤ 7

Operation: (W) → TRIS register f;

Status Affected: None

Encoding:

00	0000	0110	0fff

Description: The instruction is supported for code compatibility with the PIC16C5X products. Since TRIS registers are readable and writable, the user can directly address them.

Words: 1

Cycles: 1

Example

To maintain upward compatibility with future PIC16CXX products, do not use this instruction.

XORLW **Exclusive OR Literal with W**

Syntax: [label] XORLW k

Operands: 0 ≤ k ≤ 255

Operation: (W) .XOR. k → (W)

Status Affected: Z

Encoding:

11	1010	kkkk	kkkk

Description: The contents of the W register are XOR'ed with the eight bit literal 'k'. The result is placed in the W register.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `XORLW 0xAF`

Before Instruction
W = 0xB5

After Instruction
W = 0x1A

XORWF **Exclusive OR W with f**

Syntax: [label] XORWF f,d

Operands: 0 ≤ f ≤ 127
d ∈ [0,1]

Operation: (W) .XOR. (f) → (destination)

Status Affected: Z

Encoding:

00	0110	dfff	ffff

Description: Exclusive OR the contents of the W register with contents of register 'f'. If 'd' is 0 the result is stored in the W register. If 'd' is 1 the result is stored back in register 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example `XORWF REG 1`

Before Instruction
REG = 0xAF
W = 0xB5

After Instruction
REG = 0x1A
W = 0xB5