Conjunto de Instrucciones PIC16F877

Sergio Fco. Hernández Machuca

ADDLW **ADD Literal and W**

Sintaxis: [Etiqueta] ADDLW **k**

Operandos: $0 \leq \mathbf{k} \leq 255$

Codificación: 11 111x kkkk kkkk

Palabras, ciclos: 1, 1

Operación: **(W)** + **k** → **W**

Bit de estado: C, DC, Z

Descripción: *Añade el contenido de **W** a la literal **k**, y almacena el resultado en **W**.*

ADDWF **ADD W and F**

Sintaxis: [Etiqueta] ADDWF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 0111 dfff ffff

Palabras, ciclos: 1, 1

Operación: (**W**) + **f** $\rightarrow$ **f**, si **d** = 1 ó
(**W**) + **f** $\rightarrow$ **W**, si **d** = 0

Bit de estado: C, DC, Z

Descripción: *Añade el contenido de **W** al contenido de **f**, almacena el resultado en **W** si **d** = 0, y en **f** si **d** = 1.*

ANDLW **AND Literal with W**

Sintaxis: [Etiqueta] ANDLW **K**

Operandos: $0 \leq \mathbf{k} \leq 255$

Codificación: 11 1001 kkkk kkkk

Palabras, ciclos: 1, 1

Operación: (**W**) AND **k** → (**W**)

Bit de estado: Z

Descripción: *Efectúa un AND lógico entre el contenido de **W** y la literal **k**, y almacena el resultado en **W**.*

ANDWF **AND W with F**

Sintaxis: [Etiqueta] ANDWF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 0101 dfff ffff

Palabras, ciclos: 1, 1

Operación: (**W**) AND **f** → **f** , si **d** = 1 ó
(**W**) AND **f** → (**W**), si **d** = 0

Bit de estado: Z

Descripción: *Efectúa un AND lógico entre el contenido de **W** y el contenido de **f** y coloca el resultado en **W** si **d** = 0, y en **f** si **d** = 1.*

BCF	**Bit Clear F**
Sintaxis:	[Etiqueta] BCF **f**, **b**
Operandos:	$0 \leq \mathbf{f} \leq 127$, $0 \leq \mathbf{b} \leq 7$
Codificación:	01 00bb bfff ffff
Palabras, ciclos:	1, 1
Operación:	$0 \rightarrow (\mathbf{f}\langle\mathbf{b}\rangle)$
Bit de estado:	Ninguno
Descripción:	*Pone a cero el bit número "**b**" del registro **f**.*

BSF	**Bit Set F**
Sintaxis:	[Etiqueta] BCF **f**, **b**
Operandos:	$0 \leq \mathbf{f} \leq 127$, $0 \leq \mathbf{b} \leq 7$
Codificación:	01 01bb bfff ffff
Palabras, ciclos:	1, 1
Operación:	$1 \rightarrow (\mathbf{f}\langle\mathbf{b}\rangle)$
Bit de estado:	Ninguno
Descripción:	*Pone a uno el bit número "**b**" del registro **f**.*

BTFSC **Bit Test, Skip if Clear**

Sintaxis: [Etiqueta] BTFSC **f**, **b**

Operandos: $0 \leq \mathbf{f} \leq 127$, $0 \leq \mathbf{b} \leq 7$

Codificación: 01 10bb bfff ffff

Palabras, ciclos: 1, 1 o 2 (ver descripción)

Operación: Salta la siguiente instrucción (no la ejecuta) si (**f**<**b**>) = 0

Bit de estado: Ninguno

Descripción: *Si el bit número "**b**" de **f** está en cero, la instrucción que sigue a ésta se ignora y se trata como una instrucción NOP. En este caso, y sólo en este caso, la instrucción BTFSC precisa dos ciclos para ejecutarse.*

CLRF	**Clear F**
Sintaxis:	[Etiqueta] CLRF **f**
Operandos:	$0 \leq \mathbf{f} \leq 127$
Codificación:	00 0001 1fff ffff
Palabras, ciclos:	1, 1
Operación:	$00 \rightarrow \mathbf{f}$, $1 \rightarrow Z$
Bit de estado:	Z
Descripción	*Pone el contenido de **f** a cero y activa el bit Z.*

BTFSS **Bit Test, Skip if Set**

Sintaxis: [Etiqueta] BTFSS **f**, **b**

Operandos: $0 \leq \mathbf{f} \leq 127$, $0 \leq \mathbf{b} \leq 7$

Codificación: 01 11bb bfff ffff

Palabras, ciclos: 1, 1 o 2 (ver descripción)

Operación: Salta la siguiente instrucción (no la ejecuta) si (**f**<**b**>) = 1

Bit de estado: Ninguno

Descripción: *Si el bit número "**b**" de **f** está a 1, la instrucción que sigue a ésta se ignora y se trata como un NOP. En este caso, y sólo en este caso, la instrucción BTFSS precisa dos ciclos para ejecutarse.*

CALL	**Subroutine call**
Sintaxis:	[Etiqueta] CALL **k**
Operandos:	$0 \leq \mathbf{k} \leq 2047$
Codificación:	10 0kkk kkkk kkkk
Palabras, ciclos:	1, 2
Operación:	(PC) + 1 → TOS, **K** → PC<10:0>, (PCLATH<4:3>) → PC<12:11>
Bit de estado:	Ninguno

Descripción: *Salvaguarda la dirección de retorno en la pila y después llama a la subrutina situada en la dirección cargada en el PC. Primero es guardada la dirección de retorno (PC+1) en la pila (stack).*

CLRW	**Clear W register**
Sintaxis:	[Etiqueta] CLRW
Operandos:	Ninguno
Codificación:	00 0001 0xxx xxxx
Palabras, ciclos:	1, 1
Operación:	00 → **W**, 1 → Z
Bit de estado:	Z
Descripción:	*Pone el registro **W** a cero y activa el bit Z.*

CLRWDT **Clear Watchdog Timer**

Sintaxis: [Etiqueta] CLRWDT

Operandos: Ninguno

Codificación: 00 0000 0110 0100

Palabras, ciclos: 1, 1

Operación: 00 → WDT, 1 → TO, 1 → PD

Bit de estado: TO, PD

Descripción: *Pone a cero el registro contador del temporizador watchdog, así como el pre-escalador del WDT (Watch Dog Timer). Los bits TO y PD se ponen a 1.*

COMF **Complement F**

Sintaxis: [Etiqueta] COMF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 1001 dfff ffff

Palabras, ciclos: 1, 1

Operación: (/**f**) → **f** si **d** = 1, ó (/**f**) → **W** si **d** = 0

Bit de estado: Z

Descripción: *Hace el complemento de **f**, bit a bit. El resultado se almacena de nuevo en **f**, si **d** = 1, y en **W** si **d** = 0 (en este caso, **f** no varía).*

DECF **Decrement F**

Sintaxis: [Etiqueta] DECF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 0011 dfff ffff

Palabras, ciclos: 1, 1

Operación: (**f**) – 1 → **f** si **d** = 1, ó
(**f**) – 1 → **W** si **d** = 0

Bit de estado: Z

Descripción: *Decrementa el contenido de **f** en una unidad. El resultado se almacena de nuevo en **f** si **d** = 1, y en **W** si **d** = 0 (en este caso, **f** no varía).*

DECFSZ **Decrement F, Skip if Zero**

Sintaxis: [Etiqueta] DECFSZ **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 1011 dfff ffff

Palabras, ciclos: 1, 1 (2)

Operación: (**f**) – 1 → **f** , si **d** = 1, ó (**f**) – 1 → **W** si **d** = 0 y salta la siguiente instrucción (omite ejecutarla) si **f** – 1 = 0.

Bit de estado: Ninguno

Descripción: *Decrementa el contenido de **f** en una unidad. El resultado se almacena de nuevo en **f** si **d** = 1, y en **W** si **d** = 0 (en este caso, **f** no varía). Si el resultado es nulo, se ignora la siguiente instrucción y, en ese caso, esta instrucción dura dos ciclos.*

GOTO **Go To**

Sintaxis: [Etiqueta] GOTO **k**

Operandos: $0 \leq \mathbf{k} \leq 2047$

Codificación: 10 1kkk kkkk kkkk

Palabras, ciclos: 1, 2

Operación: **k** → PC <10:0>,
PCLATH<4:3>) → PC<12:11>

Bit de estado: Ninguno

Descripción *GOTO es un salto incondicional, el onceavo bit del valor dado como argumento es cargado en el bit PC <10:0>, los bits más significativos de PC son cargados de PCLATH<4:3>, ésta es una instrucción de 2 ciclos. Llama a la subrutina en la dirección cargada en el PC.*

INCF **Increment F**

Sintaxis: [Etiqueta] INCF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 1010 dfff ffff

Palabras, ciclos: 1, 1

Operación: $(\mathbf{f}) + 1 \rightarrow \mathbf{f}$, si $\mathbf{d} = 1$ ó
$(\mathbf{f}) + 1 \rightarrow \mathbf{W}$, si $\mathbf{d} = 0$

Bit de estado: Z

Descripción: *Incrementa el contenido de **f** en una unidad. El resultado se almacena de nuevo en **f** si **d** = 1, y en **W** si **d** = 0 (en este caso, **f** no varía).*

INCFSZ **Increment F Skip if Zero**

Sintaxis: [Etiqueta] INCFSZ **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 1111 dfff ffff

Palabras, ciclos: 1, 1 (2)

Operación: (**f**) + 1 → **f**, si **d** = 1 ó
(**f**) + 1 → **W** si **d** = 0 y salta si **f** + 1 = 0

Bit de estado: Ninguno

Descripción: *Incrementa el contenido de **f** en una unidad. El resultado se almacena de nuevo en **f** si **d** = 1, y en **W** si **d** = 0 (en este caso, **f** no varía), Si el resultado es nulo, se ignora la siguiente instrucción y, en ese caso, esta instrucción dura dos ciclos.*

IORLW	**Inclusive OR Literal with W**
Sintaxis:	[Etiqueta] IORLW **k**
Operandos:	$0 \leq \mathbf{k} \leq 255$
Codificación:	11 1000 kkkk kkkk
Palabras, ciclos:	1,1
Operación:	(**W**) OR **k** → (**W**)
Bit de estado:	Z
Descripción:	*Efectúa un OR lógico inclusivo entre el contenido de **W** y el literal **k**, y almacena el resultado en **W**.*

IORWF **Inclusive OR W with F**

Sintaxis: [Etiqueta] IORWF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 11 0100 dfff ffff

Palabras, ciclos: 1, 1

Operación: (**W**) OR $\rightarrow$ **f**, si **d** = 1 ó
(**W**) OR **f** $\rightarrow$ **W**, si **d** = 0

Bit de estado: Z

Descripción *Efectúa un OR lógico inclusivo entre el contenido de **W** y el contenido de **f**, y almacena el resultado en **f** sí **d** = 1 y en **W** si **d** = 0.*

MOVF **Move F**

Sintaxis: [Etiqueta] MOVF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 1000 dfff ffff

Palabras, ciclos: 1, 1

Operación: (**f**) → **f**, si **d** = 1, ó (**f**) → **W**, si **d** = 0

Bit de estado: Z

Descripción: *Desplaza el contenido de **f** a **f** si **d** = 1 ó a **W** si **d** = 0. ¡Atención!: El desplazamiento de **f** a **f**, que a priori parece inútil, permite comprobar el contenido de **f** con respecto a cero, ya que esta instrucción actúa sobre el bit Z.*

MOVLW	**Move Literal to W**
Sintaxis:	[Etiqueta] MOVLW **k**
Operandos:	$0 \leq \mathbf{k} \leq 255$
Codificación:	11 00xx kkkk kkkk
Palabras, ciclos:	1, 1
Operación:	**k** → (**W**)
Bit de estado:	Ninguno
Descripción:	*Carga **W** con el literal **k**.*

MOVWF	**Move W to F**
Sintaxis:	[Etiqueta] MOVWF **f**
Operandos:	$0 \leq \mathbf{k} \leq 255$
Codificación:	00 0000 1fff ffff
Palabras, ciclos:	1, 1
Operación:	**(W)** → **f**
Bit de estado:	Ninguno
Descripción:	*Carga **f** con el contenido de **W**.*

NOP **No Operation**

Sintaxis: [Etiqueta] NOP

Operandos: Ninguno

Codificación: 00 0000 0xx0 0000

Palabras, ciclos: 1, 1

Operación: Ninguna

Bit de estado: Ninguno

Descripción: *Sólo consume tiempo de máquina (en este caso, un ciclo) como cualquier otro NOP.*

RETFIE **Return From Interrupt**

Sintaxis: [Etiqueta] RETFIE

Operandos: Ninguno

Codificación: 00 0000 0000 1001

Palabras, ciclos: 1, 2

Operación: TOS → PC, 1 → GIE

Bit de estado: Ninguno

Descripción: *Carga el PC con el valor que se encuentra en la parte superior de la pila, asegurando así la vuelta de la interrupción. Pone a 1 el bit GIE, con el fin de autorizar de nuevo que se tengan en cuenta las interrupciones.*

RETLW **Return Literal to W**

Sintaxis: [Etiqueta] RETLW **k**

Operandos: $0 \leq \mathbf{k} \leq 255$

Codificación: 11 01xx kkkk kkkk

Palabras, ciclos: 1, 2

Operación: **k** → **W**, TOS → PC

Bit de estado: Ninguno

Descripción: *Carga **W** con el literal **k**, y después carga el PC con el valor que se encuentra en la parte superior de la pila, efectuando así un retorno de subrutina. Esta instrucción dura dos ciclos.*

RETURN **Return from Subroutine**

Sintaxis: [Etiqueta] RETURN

Operandos: Ninguno

Codificación: 00 0000 0000 0000

Palabras, ciclos: 1, 2

Operación: TOS → PC

Bit de estado: Ninguno

Descripción: *Carga el PC con el valor que se encuentra en la parte superior de la pila, efectuando así una vuelta de subrutina. Se trata de la instrucción RETLW simplificada. Esta instrucción dura dos ciclos.*

RLF **Rotate Left F through Carry**

Sintaxis: [Etiqueta] RLF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 1101 dfff ffff

Palabras, ciclos: 1,1

Operación: Ver descripción

Bit de estado: C

Descripción: *Rotación de un bit a la izquierda del contenido de **f**, pasando por el bit de acarreo "C". Si **d** = 1 el resultado se almacena en **f**, si **d** = 0 el resultado se almacena en **W**.*

C ← 7 ← 6 ← 5 ← 4 ← 3 ← 2 ← 1 ← 0 ← C

RRF **Rotate Right F through Carry**

Sintaxis: [Etiqueta] RRF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 1100 dfff ffff

Palabras, ciclos: 1, 1

Operación: Ver descripción

Bit de estado: C

Descripción: *Rotación de un bit a la derecha del contenido de **f**, pasando por el bit de acarreo "C". Si **d** = 1 el resultado se introduce en **f**, si **d** = 0 el resultado se almacena en **W**.*

C → 7 → 6 → 5 → 4 → 3 → 2 → 1 → 0 → C

SLEEP	**Sleep**
Sintaxis:	[Etiqueta] SLEEP
Operandos:	Ninguno
Codificación:	00 0000 0110 0011
Palabras, ciclos:	1, 1
Operación:	0 → PD, 1 → TO, 00 → WDT, 0 → Preescalador del WDT
Bits de estado:	TO, PD
Descripción:	*Pone al circuito en modo sleep con parada del oscilador. ¡Atención!: Consulte los capítulos dedicados a cada tipo de circuito para ver las posibilidades exactas y las consecuencias de la entrada del circuito en modo SLEEP.*

SUBLW **Substract W from Literal**

Sintaxis: [Etiqueta] SUBLW **k**

Operandos: $0 \leq \mathbf{k} \leq 255$

Codificación: 11 110x kkkk kkkk

Palabras, ciclos: 1, 1

Operación: **k** – (**W**) → **W**

Bit de estado: C, DC, Z

Descripción: *Sustrae el contenido de **W** del literal **k**, y almacena el resultado en **W**. La sustracción se realiza en complemento a dos.*

SUBWF **Substract W from F**

Sintaxis: [Etiqueta] SUBWF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 0010 dfff ffff

Palabras, ciclos: 1, 1

Operación: $(\mathbf{f}) - (\mathbf{W}) \rightarrow \mathbf{f}$, si $\mathbf{d} = 1$, ó
$(\mathbf{f}) - (\mathbf{W}) \rightarrow \mathbf{W}$, si $\mathbf{d} = 0$.

Bit de estado: C, DC, Z

Descripción: *Sustrae el contenido de **W** del contenido de **f**, y almacena el resultado en **W** si **d** = 0, y en **f** si **d** = 1. La sustracción se realiza en complemento a dos.*

SWAPF **Swap Nibbles in F**

Sintaxis: [Etiqueta] SWAPF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 1110 dfff ffff

Palabras, ciclos: 1, 1

Operación: (**f**<3:0>) → (destino <7:4>) ,
(**f**<7:4>) → (destino <3:0>)
Depende el destino del valor **d**.
Si **d** = 1, se deposita en **f**, si **d** = 0 en **W**.

Bit de estado: Ninguno.

Descripción: *Intercambia los cuatro bits de mayor peso con los cuatro bits de menor peso de **f**, y almacena el resultado en **f** si **d** = 1, o en **W** si **d** = 0.*

XORLW	**Exclusive OR Literal with W**
Sintaxis:	[Etiqueta] XORLW **k**
Operandos:	$0 \leq \mathbf{k} \leq 255$
Codificación:	11 1010 kkkk kkkk
Palabras, ciclos:	1, 1
Operación:	(**W**) OR EXCLUSIVO **k** → (**W**)
Bit de estado:	Z
Descripción:	*Efectúa un OR lógico exclusivo entre el contenido de **W** y el literal **k**, y almacena el resultado en **W**.*

XORWF **Exclusive OR W with F**

Sintaxis: [Etiqueta] XORWF **f**, **d**

Operandos: $0 \leq \mathbf{f} \leq 127$, $\mathbf{d} \in (0,1)$

Codificación: 00 0110 dfff ffff

Palabras, ciclos: 1, 1

Operación: (**W**) OR EXCLUSIVO (**f**) → **f**, si **d** = 1 ó
(**W**) OR EXCLUSIVO (**f**) → **W**, si **d** = 0

Bit de estado: Z

Descripción: *Efectúa un OR lógico exclusivo entre el contenido de **W** y el contenido de **f**, y almacena el resultado en **f** si **d** = 1 y en **W** si **d** = 0.*

Ejemplos de uso del Conjunto de Instrucciones

Manipulación de un bit:

```
Etiqueta:       bcf     PORTB, 0        ; Limpia el bit “0” de PORTB.

                bsf     STATUS, C       ; Haz “verdadera” la bandera de Acarreo.
```

Limpieza y movimiento:

```
Base:           clrw                    ; Limpia el registro “W”.

                clrf    TEMP1           ; Limpia la variable temporal TEMP1.

                movlw   5               ; Carga el valor “5” en el registro “W”.

                movlw   10              ; Carga D’10’, H’0A’, B’00001010’,
                                        ; ó 0x0A en “W”.

                movwf   TEMP1           ; Mueve “W” en la variable TEMP1.

                movf    TEMP1, W        ; Mueve el valor de la variable a “W”.

                swapf   TEMP1, F        ; Intercambiar nibbles de TEMP1.

                swapf   TEMP1, W        ; Copiar TEMP1 en “W”, intercambiar
                                        ; nibbles, TEMP1 no se altera.
```

Ejemplos de uso del Conjunto de Instrucciones (continuación)

Incremento, Decremento y Complemento:

```
Aqui:           incf    TEMP1, F        ; Incrementa el valor de variable TEMP1.

                incf    TEMP1, W        ; Incremente TEMP1, guarda el resultado
                                        ; en "W". El valor de TEMP1 o se altera.

                decf    TEMP1, F        ; Decrementa el valor de la variable.

                comf    TEMP1, F        ; Intercambia los "ceros" a "unos" y los
                                        ; "unos" a "ceros".
```

Manipulación de Múltiples bits:

```
Entrada:        andlw   B'00000111'     ; Forzar los cinco bits más
                                        ; significativos de "W" a "cero".

                andwf   TEMP1, F        ; Realiza la operación "AND" entre el
                                        ; valor de la variable y "W",
                                        ; el resultado queda en la variable.

                andwf   TEMP1, W        ; Igual que lo anterior, pero ahora
                                        ; el resultado queda en "W".

                iorlw   B'00000011'     ; Hace los dos bits menos significativos
                                        ; de "W" de un valor de "uno".
```

Ejemplos de uso del Conjunto de Instrucciones (continuación)

Manipulación de Múltiples bits (continuación):

```
                iorwf   TEMP1, F        ; Realiza la operación “OR” entre la
                                        ; variable y “W”, deposita el resultado
                                        ; en la variable.

                xorlw   B’00000111’     ; Complementa los tres bits menos
                                        ; significativos de “W”.
```

Suma y Resta:

```
Proceso:        addlw   D’5’            ; Agrega 5 a “W”.

                addwf   TEMP1, F        ; Agrega a TEMP1 el valor de “W”,
                                        ; se actualiza TEMP1.

                sublw   D’5’            ; Calcula: 5 – “W”, dejando en “W” el
                                        ; resultado.

                subwf   TEMP1, F        ; Calcula TEMP1 – “W”, actualizando
                                        ; TEMP1.
```

Rotación:

```
Salida:         rlf     TEMP1, F        ; Rotación a la izquierda, a través de
                                        ; la bandera de “acarreo”.
```

Ejemplos de uso del Conjunto de Instrucciones (continuación)

Rotación (continuación):

```
Salida:         rrf     TEMP1, W        ; Copia TEMP1 en "W", rota "W" a través
                                        ; de la bandera de "acarreo", el valor
                                        ; de TEMP1 no se altera.
```

Salto condicional:

```
                btfsc   TEMP1, 0        ; "Salta" la siguiente instrucción si
                                        ; si el bit "0" de TEMP1 es "cero".

                btfss   STATUS, C       ; No ejecuta la siguiente instrucción
                                        ; si la bandera de "acarreo" está
                                        ; puesta.

                decfsz  TEMP1, F        ; Decrementa la variable, actualízala,
                                        ; evita la siguiente instrucción si
                                        ; la variable vale "cero".

                incfsz  TEMP1, W        ; Si el valor de TEMP1 fuera 0xFF (antes
                                        ; de esta línea), no ejecutes la
                                        ; siguiente instrucción. No altera
                                        ; el valor de TEMP1.
```

Ejemplos de uso del Conjunto de Instrucciones (continuación)

Saltos, Subrutinas, Retornos:

```
Lazo:
                movwf    TEMP1, W          ; Copia TEMP1 en “W”.
                ...
                ...
                goto     Lazo              ; Salta a la línea en donde está la
                                           ; etiqueta “Lazo”.

                call     Sumatoria         ; Guarda en el “stack” el valor actual
                                           ; del contador de programa (PC), la
                                           ; próxima instrucción a ejecutarse está
                                           ; adonde se encuentra la etiqueta
                                           ; “Sumatoria”.

                return                     ; Recupera el último valor depositado
                                           ; en el “stack” y cópialo en el “PC”.

                retlw    D’9’              ; Recupera el último valor depositado
                                           ; en el “stack” y cópialo en el “PC”,
                                           ; antes de ejecutar la siguiente
                                           ; instrucción, carga “W” con 9.

                retfie                     ; Recupera el último valor depositado
                                           ; en el “stack” y cópialo en el “PC”,
                                           ; habilita la atención a la ocurrencia
                                           ; de interrupciones.
-------------------------------------------------------------------------------
```

Ejemplos de uso del Conjunto de Instrucciones (continuación)

Otras instrucciones:

```
Lazo:
                clrwdt                  ; Si el “WatchDog Timer” está habilitado,
                                        ; reinicia el contador WDT, antes de que
                                        ; se reinicie en microcontrolador.

                sleep                   ; Para el circuito interno de oscilación,
                                        ; reduce la potencia, espera por un
                                        ; agotamiento del WDT (si estuviera
                                        ; habilitado) o una señal externa, para
                                        ; continuar con la ejecución del
                                        ; programa.

                nop                     ; No hace nada, espera por un ciclo de
                                        ; reloj.
-------------------------------------------------------------------------------
```

Problemas a resolver:

* Implementar un contador de 16 bits. Para esta estructura, resolver:

- Decremento del contador de 16 bits.
- Incremento del contador de 16 bits.
- Prueba para verificar que el contador es igual a cero.
- Prueba de que el contador tiene un valor en particular.

* Implementar un “stack” en software. Considere las localidades de memoria que son comunes en todos los bancos. Reserve 10 localidades de memoria para uso del “stack”. Debe existir un Registro el cual apunte a la localidad del “stack” en donde se operará. Implementar lo siguiente para esta estructura:

- Guardado de un dato (operación “Push”).
- Recuperación de un dato (operación “Pop”).

Propuesta de soluciones:

```
Decremento:
                movf    ContadorL, F        ; Ajusta la bandera de "cero" si la parte
                                            ; Baja del contador es igual a "0".
                btfsc   STATUS, Z           ; Si así fuera, ...
                decf    ContadorH, F        ; ... decrementa la parte Alta del contador.
                decf    ContadorL, F        ; En cualquier caso, decrementa la parte Baja.

-------------------------------------------------------------------------------

Prueba:
                movf    ContadorL, F        ; Ajusta la bandera de "cero" si la parte
                                            ; Baja del contador es igual a "0".
                btfsc   STATUS, Z           ; Si no fuera así, se concluyó la prueba.
                movf    ContadorH           ; Ajusta la bandera de "cero" si la parte
                                            ; Alta del contador es igual a "0".
                btfsc   STATUS, Z           ; Si no fuera así, se terminó ...
                                            ; (El contador NO fue igual a "0")
                goto    ContadorIgualCero   ; En caso contrario, ...
                                            ; ... (El contador SI fue igual a "0")

Continua:                                   ; Aquí va el código para seguir ...
```