APÉNDICE A

SET DE INSTRUCCIONES DEL PIC18F252

Los PIC 18FXX está compuesto por una CPU de tipo RISC con un juego de 75 instrucciones o de 83 si se trabaja en el modo extendido frente a las 35 instrucciones de los PIC16FXX. Como en el resto de microcontroladores PIC la mayoría de las instrucciones se ejecutan en un ciclo de máquina y en el propio código de la instrucción se incluye los campos necesarios para su instrucción. Algunas diferencias son:

- Son instrucciones de 16 bits de ancho, 1 word. Algunas instrucciones ocupan 2 words (32 bits) y pueden necesitar hasta tres ciclos de ejecución.
- Aparecen 40 nuevas instrucciones respecto a las que emplean los PIC16.
- Aparece un nuevo modo de direccionamiento llamado “Access Bank” (banco de acceso).
- Se dispone de un modo extendido que añade otras 8 instrucciones y un nuevo modo de direccionamiento indexado que se aplica a todas las instrucciones orientadas a bytes y a bits. Este modo extendido lo puede habilitar o no el usuario mediante la palabra de configuración **CONFIG4L**.

DESCRIPCION DE LOS CAMPOS DE LAS INSTRUCCIONES

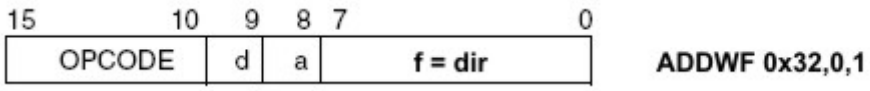
La mayor parte de las instrucciones se componen del código de operación (OPCODE) que las define y de una serie de campos u operandos necesarios para determinar su funcionamiento en un momento dado. Estos campos se resumen en la siguiente tabla

CAMPO	DESCRIPCION
a	Bit para la selección del Banco de acceso a=0 Banco de acceso, se ignora el registro BSR a=1 El banco deseado se indica en BSR
bbb	Selecciona a uno de los 8 bits de cualquier registro al que se desea acceder
BSR	Registro selector de banco. Se usa para seleccionar el banco de RAM que se va a utilizar
C,DC,Z,OV,N	Representan a los bits del registro de estado (STATUS) de la ALU (Carry, Digit Carry, Zero, Overflow, Negative)
d	Bit para la selección de destino d=0 Almacena el resultado en WREG d=1 Almacena el resultado en el registro f empleado
f	Expresa una dirección de 8 bits (0x00-0xFF) del área de datos u un valor de 2 bits para selección del registro FSR.
fs	Valor de 12 bits (0x000-0xFFF) para representar una dirección fuente de RAM
fd	Valor de 12 bits (0x000-0xFFF) para representar una dirección destino en RAM
k	Campo para representar un valor inmediato o constante de 8, 12 o 20 bits
mm * *+ *- +*	Representa el modo en que el registro TBLPTR debe actuar en las instrucciones de lectura/escritura de tablas. El registro no se actualiza al realizar la operación de lectura/escritura de tablas Post incremento del registro al realizar la operación de lectura/escritura de tablas Post decremento del registro al realizar la operación de lectura/escritura de tablas Pre incremento del registro al realizar la operación de lectura/escritura de tablas
n	Representa la dirección de destino en las instrucciones de salto relativo o la dirección de destino en las instrucciones de salto directo
s	Bit de selección para los modos rápidos de CALL y RETURN s=0 No se salvan/recuperan de forma automática los registros WREG, STATUS y BSR s=1 Esos registros se salvan/recuperan de forma transparente al usuario
TBLPTR	Registro de 21 bits que actúa como puntero de una posición del área de programa
TABLAT	Registro de 8 bits donde se retiene el valor actual de la posición indicada por TBLPTR
u	Bit no usado
WDT	Representa al temporizador WDT
WREG	Representa al registro de trabajo o Acumulador
x	El estado de estos bits no es relevante. El ensamblador los pone siempre a “0”
Zs	Valor de 7 bits para el direccionamiento indirecto del registro fuente
Zd	Valor de 7 bits para el direccionamiento indirecto del registro de destino
{ }	Argumentos u operandos opcionales
[ ]	Representa direccionamientos indexados
( )	Contenido
→	Sentido de la transferencia
< >	Campo del bit de un registro
∈	entre los valores

FORMATO GENERAL DE LAS INSTRUCCIONES

Aunque hay excepciones, la mayor parte de las instrucciones ocupan un único Word de 16 bits y se alojan en correlativas direcciones de la memoria de programa organizada también en posiciones de 16 bits.

- **Instrucciones orientadas a byte:** Tienen el siguiente formato



ADDWF 0x32,0,1

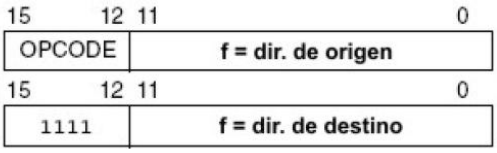
El código de operación (OPCODE) de la instrucción lo establece los 6 bits de más peso <15:10>.

El campo “d” determina si el resultado va a parar al registro W (d=0) o a la dirección del registro emplead (d=1). El campo “a” determina el tipo de acceso. Si a=0 el campo “f” expresa una dirección entre 0x00 y 0x5F del banco 0 o de 0x60 a 0xFF para los registros especiales SFR.

Si a=1 el campo “f” expresa una dirección entre 0x00 y 0xFF de cualquier banco y el registro BSR selecciona el banco (por defecto). Si a=0 y está activado el modo extendido de instrucciones, se emplea el direccionamiento indexado y “f” representa el desplazamiento u offset a añadir a FSR2.

En el ejemplo, el registro W se suma con el contenido de la posición 0x32. El resultado se deposita en el acumulador (d=0). Como a=1 se supone que el registro BSR indica el banco de RAM que se va a emplear.

- **Instrucciones Byte a Byte:** Tienen el siguiente formato formato

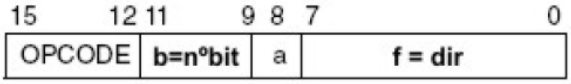


MOVFF 0x120,0x245

Son instrucciones que ocupan dos words (32 bits) en la memoria de programa. Los bits <15:12> del primer Word representan el propio código de la instrucción. Los 12 bits "f" de menos peso <11:0> de ambos words representan la dirección del registro RAM origen y de destino respectivamente. Como ambas direcciones son de 12 bits, se puede acceder a cualquiera de las 4096 posiciones de memoria RAM posibles.

En el ejemplo el contenido de la posición 0x120 se copia sobre la posición 0x245.

- **Instrucciones orientadas al bit:** Permiten actuar individualmente sobre cualquier bit particular de cualquier registro. Su formato es el siguiente.



BSF 0x25,4,0

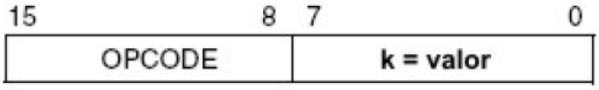
Los 4 bits de más peso <15:12> expresan el código de la instrucción. Los bits "b" <11:9> indican uno de los 8 bits posibles (000-111).

El campo "a" determina si se utiliza el BSR para seleccionar cualquier banco (a=1) o se emplea el acceso a banco 0(a=0).

El campo "f" de 8 bits expresa la dirección del registro (0x00-0xFF).

En el ejemplo se activa el bit 4 de la posición 0x25. Como a=0 no se emplea el BSR sino que se hace referencia al banco 0 directamente.

Instrucciones inmediatas o literales: Tienen el siguiente formato:

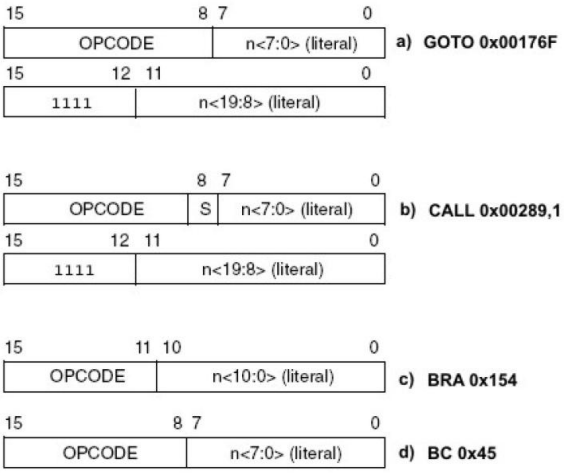


MOVLW 0x84

El código de la instrucción se expresa en los 8 bits de más peso <15:8>. Los 8 bits de menos peso <7:0> corresponden al campo "k" y representan el dato propiamente dicho.

En el ejemplo se trata de cargar de forma inmediata el registro W con el valor 0x84.

Instrucciones de control: Este tipo de instrucciones se corresponden, en general, con todas aquellas que implican un salto a lo largo del área de programa, modificando el valor actual del PC. La figura muestra 4 posibles formatos para este tipo de instrucciones. En cualquiera de ellos, tras el código de operación, se proporciona un valor de 20, 11 u 8 bits que representa la dirección a la que se desea saltar. Las instrucciones siempre ocupan 2 palabras de 16 bits (words).



a) En este ejemplo los 8 bits de menos peso de del 1er. Word n<7:0> junto con los 12 de menos peso del 2º Word n<19:8>, forma una dirección de 20 bits que permite establecer cualquier dirección de salto a lo largo de la totalidad del área de memoria de programa. Se salta a la dirección 0x00116F.

b) De manera similar al formato anterior, se forma una dirección de 20 bits que permite acceder a cualquier posición del área de programa. Cuando el campo s=0 se produce el salto a la subrutina de la dirección indicada y en la memoria de stack se guarda el valor actual del PC+2 para el posterior retorno. Si s=1, como en el ejemplo, se guardan también los registros W, STATUS y BSR de forma espontánea sobre unos registros internos y transparentes para el usuario. Esto posibilita recuperarlos durante el retorno.

c) Se corresponde con un salto relativo incondicional. La instrucción va acompañada de un valor con signo de 11 bits n<10:0>. Este valor se le añade al valor actual del PC, lo que da lugar a la dirección de destino dentro del área de programa. Es posible hacer desplazamientos de +/- 1024 posiciones mediante una instrucción que ocupa un único Word.

d) Se trata de un salto relativo condicional. La instrucción va acompañada de un valor de 8 bits con signo n<7:0>. Si la condición de salto se cumple, el valor de 8 bits se le añade al valor actual del PC, lo que da lugar a la dirección de destino dentro del área de programa. Es posible realizar desplazamientos de +/- 128 posiciones con una instrucción que ocupa un único Word.

SET DE INTRUCCIONES DEL PIC18FXX2

NEMÓNICO	DESCRIPCIÓN	CICLOS	CÓDIGO DE OPERACIÓN	FLAGS AFECTADOS	Nota
Instrucciones de registros orientadas a byte					
addwf f,d,a	W+f → destino	1	0010 01da ffff ffff	C,DC,Z,OV,N	1,2
addwfc f,d,a	W+f+C →destino	1	0010 00da ffff ffff	C,DC,Z,OV,N	1,2
andwf f,d,a	W AND f →destino	1	0001 01da ffff ffff	Z,N	1,2
clrf f,a	0 → f	1	0110 101a ffff ffff	Z	2
comf f,d,a	$\overline{f}$ → destino	1	0001 11da ffff ffff	Z,N	1,2
cpfseq f,a	f-W, if f=W,PC+4→PC else PC+2 →PC	1(2 ó 3)	0110 001a ffff ffff	Ninguno	4
cpfsgt f,a	f-w, if f>W,PC+4→PC else PC+2 →PC	1(2 ó 3)	0110 010a ffff ffff	Ninguno	4
cpfslt f,a	f-w, if f<W, PC+4 →PC else PC+2 →PC	1(2 ó 3)	0110 000a ffff ffff	Ninguno	1,2
decf f,d,a	f-1 → f	1	0000 01da ffff ffff	C,DC,Z,OV,N	1,2,3,4
decfsz f,d,a	f-1 → destino, if destino=0 PC+4 →PC else, PC+2→PC	1(2 ó 3)	0000 01da ffff ffff	Ninguno	1,2,3,4
decfsnz f,d,a	f-1 → destino, if destino≠0 PC+4 →PC else, PC+2→PC	1(2 ó 3)	0100 11da ffff ffff	Ninguno	1,2
incf f,d,a	f+1 → destino	1	0010 10da ffff ffff	C,DC,Z,OV,N	1,2,3,4
incfsz f,d,a	f-1 → destino, if destino=0 PC+4 →PC else, PC+2→PC	1(2 ó 3)	0011 11da ffff ffff	Ninguno	4
infsnz f,d,a	f+1 → destino, if destino≠0 PC+4 →PC else, PC+2→PC	1(2 ó 3)	0100 10da ffff ffff	Ninguno	1,2
iorwf f,d,a	W OR f → destino	1	0001 00da ffff ffff	Z,N	1,2
movf f,d,a	f → destino	1	0101 00da ffff ffff	Z,N	1
movff f_s,f_d	f_{fuente} → $f_{destino}$	2	1100 ffff ffff ffff 1111 ffff ffff ffff	Ninguno	
movwf f,a	W → f	1	0110 111a ffff ffff	Ninguno	
mulwf f,a	W*f →PRODH:PRODL	1	0000 001a ffff ffff	Ninguno	
negf f,a	f+1→ PRODH:PRODL	1	0110 110a ffff ffff	C,DC,Z,OV,N	1,2
rlcf f,d,a	Rota f a izquierda. a través del Carry →(destino)	1	0011 01da ffff ffff	C,Z,N	
rlncf f,d,a	Rota a la izquierda (sin carry) → destino	1	0100 01da ffff ffff	Z,N	1,2
rrcf f,d,a	Rota a la derecha a través del Carry → destino	1	0011 00da ffff ffff	C,Z,N	
rrncf f,d,a	Rota a la derecha (sin carry) → destino	1	0100 00da ffff ffff	Z,N	
setf f,a	0xFF → f	1	0110 100a ffff ffff	Ninguno	
subfwb f,d,a	W-f → destino	1	0101 11da ffff ffff	C,DC,Z,OV,N	
subwfb f,d,a	W-f-C → destino	1	0101 01da ffff ffff	C,DC,Z,OV,N	1,2
swapf f,d,a	f<3:0> → destino<7:4> f<7:4> → destino<3:0>	1	0011 10da ffff ffff	Ninguno	4
tstfsz f,a	PC+4 →PC si f=0, else PC+2→PC	1 (2 ó 3)	0110 011a ffff ffff	Ninguno	1,2
xorwf f,d,a	W XOR f → destino	1	0001 10da ffff ffff	Z, N	
Instrucciones de registros orientadas a bits					
bcf f,b,a	0 →f(b)	1	1001 bbba ffff ffff	Ninguno	1,2
bsf f,b,a	1 →f(b)	1	1000 bbba ffff ffff	Ninguno	1,2
btfsc f,b,a	Salta s f(b) = 0	1(2 ó 3)	1011 bbba ffff ffff	Ninguno	3,4
btfss f,b,a	Salta si f(b) = 1	1(2 ó 3)	1010 bbba ffff ffff	Ninguno	3,4
btg f,b,a	NOT f(b) → f(b)	1	0111 bbba ffff ffff	Ninguno	1,2

Operaciones de Control						
bc	n	if C=1, PC+2+2*n →PC else PC+2 →PC	1 (2)	1110 0010 nnnn nnnn	Ninguno	
bn	n	if N=1, PC+2+2*n →PC else PC+2 →PC	1 (2)	1110 0110 nnnn nnnn	Ninguno	
bnc	n	if C=0, PC+2+2*n →PC else PC+2 →PC	1 (2)	1110 0011 nnnn nnnn	Ninguno	
bnn	n	if N=0, PC+2+2*n →PC else PC+2 →PC	1 (2)	1110 0111 nnnn nnnn	Ninguno	
bnov	n	if OV=0, PC+2+2*n →PC else PC+2 →PC	1 (2)	1110 0101 nnnn nnnn	Ninguno	
bnz	n	if Z=0, PC+2+2*n →PC else PC+2 →PC	2	1110 0001 nnnn nnnn	Ninguno	
bov	n	if OV=1, PC+2+2*n →PC else PC+2 →PC	1 (2)	1110 0100 nnnn nnnn	Ninguno	
bra		PC+2+2*n →PC	1 (2)	1101 0nnn nnnn nnnn	Ninguno	
bz	n	if Z=1, PC+2+2*n →PC else PC+2 →PC	1 (2)	1110 0000 nnnn nnnn	Ninguno	
call	n,s	PC+4 → TOS, n→PC<20:1>, Si s=1, W →WS, STATUS → STATUSS, BSR →BSRS	2	1110 110s kkkk kkkk 1111 kkkk kkkk kkkk		
clrwdt		00 → WDT	1	0000 0000 0000 0100	$\overline{TO}$, $\overline{PD}$	
daw		if W<3:0> >9 or DC=1, W<3:0>+6 → W<3:0>, else W<3:0> →W<3:0>; if W<7:4> >9 0r C=1, W<7:4>+6 → W<7:4>; else W<7:4>→ W<7:4>;	1	0000 0000 0000 0111	C	
goto	n	N → PC<20:1>	2	1110 1111 kkkk kkkk 1111 kkkk kkkk kkkk	Ninguno	
nop		No operación	1	0000 0000 0000 0000	Ninguno	
pop		TOS-1 → TOS	1	0000 0000 0000 0110	Ninguno	
push		PC+2 → TOS	1	0000 0000 0000 0101	Ninguno	
rcall	n	PC+2 → PC PC+2+2*nd→ PC	2	1101 1nnnn nnnn nnnn	Ninguno	
reset		Reset por software	1	0000 0000 1111 1111	Todos	
retfie	s	TOS→PC,1→GIE/GIEH or PEIE/GIEL, if s=1, WS→W, STATUSS→STATUS, BSRS→ BSR, PCLATU/PCLATH es inalterado	2	0000 0000 0001 000s	GIE/GIEH PEIE/GIEL	
retlw		Kk → W	2	0000 1100 kkkk kkkk	Ninguno	
sleep		0 → WDT, 0→ WDT postescaler, 1→TO, 0→PD	1	0000 0000 0000 0011	$\overline{TO}$, $\overline{PD}$	
Operaciones con literales						
addlw	k	W + k → W	1	0000 1111 kkkk kkkk	C,DC,Z,OV,N	
andlw	k	W AND k → W	1	0000 1011 kkkk kkkk	Z,N	
iorlw	k	W OR k → W	1	0000 1001 kkkk kkkk	Z,N	
lfsr	f,k	Carga un litera del 12 bits en el FSR (segunda palabra)	2	1111 0000 kkkk kkkk 0000 0001 0000 kkkk	Ninguno	
movlb	k	k → BSR	1	0000 0001 0000 kkkk	Ninguno	
movlw	k	k → W	1	0000 1101 kkkk kkkk	Ninguno	
mullw		(W)x k →PRODH	1	0000 1101 kkkk kkkk	Ninguno	
retlw	k	Retorno de subrutina con un literal en W	2	0000 1100 kkkk kkkk	Ninguno	
sublw	k	K – W → W	1	0000 1000 kkkk kkkk	C,DC,Z,OV,N	
xorlw	k	W OR esclusiva k → W	1	0000 1010 kkkk kkkk	Z,N	
Memoria de datos ←→Operaciones en memoria de programa						
tblrd*		Memoria de programa	2	0000 0000 0000 1000	Ninguno	

	(TBLPTR) → TABLAT				
tblrd*+	Memoria de programa (TBLPTR) → TABLAT TBLPTR+1→TBLPTR		0000 0000 0000 1001	Ninguno	
tblrd*-	Memoria de programa (TBLPTR) → TABLAT TBLPTR-1→TBLPTR		0000 0000 0000 1010	Ninguno	
tblrd+*	TBLPTR+1 → TBLPTR Memoria de programa (TBLPTR)→TABLAT		0000 0000 0000 1011	Ninguno	
tblwt*	TABLAT →Memoria de programa (TBLPTR)	2(5)	0000 0000 0000 1100	Ninguno	
tblwt*+	TABLAT →Memoria de programa (TBLPTR) TBLPTR + 1→ TBLPTR		0000 0000 0000 1101	Ninguno	
tblwt*-	TABLAT →Memoria de programa (TBLPTR) TBLPTR-1 →TBLPTR		0000 0000 0000 1110	Ninguno	
tblwt+*	TABLAT +1→ TBLPTR TABLAT →Memoria de programa (TBLPTR)		0000 0000 0000 1111	Ninguno	

Notas 1: Cuando el registro de un PUERTO se modifica como una función de sí mismo (por ejemplo, MOVF PORTB, 1, 0), el valor usado será el presente en los pines. Por ejemplo, si el latch de los datos es '1 ' para un pin configurado como entrada y se pone a nivel bajo por un dispositivo externo el pin correspondiente, el dato que se escribirá será un "0".

2: Si esta instrucción se ejecuta en el registro de TMR0 (poniendo d = 1), los prescaler se inicializan si están asignados.

3: Si el Contador de Programa (PC) se modifica o un test condicional es verdad, la instrucción requiere dos ciclos. El segundo ciclo se ejecuta como un NOP.

4: Algunas instrucciones son instrucciones de 2-palabras. La segunda palabra de estas instrucciones se ejecutará como un NOP, a menos que la primera palabra de la instrucción recupere la información estos 16-bits. Esto asegura que todas las posiciones de memoria de programa tienen una instrucción válida.

5: Si la Escritura de Tabla comienza el ciclo de escritura interna de memoria, la escriba continuará hasta terminó.

Tabla 1 Repertorio de instrucciones del PIC18Fxx2

addlw — Suma el literal k con W

Sintaxis: [*Etiqueta*] addlw k
Operandos: $0 \le k \le 255$
Operación: (W) + k → (W)
Flags afectados: C, OV,C,DC, Z
Código de OP :

0000	1111	kkkk	kkkk

Descripción: Suma el contenido del registro W al literal k y almacena el resultado en W. Si se produce acarreo, el flag C se pone a "1"
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal K	Procesa datos	Escribe en W

Ejemplo 1: addlw 0x15 ; (W) + 0x15 → (W)
Antes Instrucción: (W) = 0x10, y C = ¿?.
Después Instrucción: (W) = 0x10 + 0x15 = 0x25 y C = 0.
(W) = b'0001 0000' + b'0001 0101' = b'0010 0101'

addwf — Suma W con el registro f

Sintaxis: [*Etiqueta*] addwf f[,d[a]
Operandos: $0 \le k \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (W) + (f) → (destino)
Flags afectados: N,OV,C, DC, Z
Código de OP :

0010	01da	dfff	ffff

Descripción: Suma el contenido del registro W al contenido del registro f, y almacena el resultado en W si d = 0, y en el registro f si d = 1. Si se produce acarreo el flag C se pone a "1". Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe W en destino

Ejemplo: *addwf Registro,0,0* ; (Registro) + (W) → (W)
Antes Instrucción: (W) = 0x17, (Registro) = 0xC2, y C = ¿?.
Después Instrucción: (W) = 0xD9, (Registro) = 0xC2, y C = 0.

addwfc — Suma W y el bit de Carry con el registro f

Sintaxis: [*Etiqueta*] ADDWFC f[,d[a]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (W) + (f) + (C) → (destino)
Flags afectados: N,OV,C, DC, Z
Código de OP :

0010	00da	ffff	ffff

Descripción: Suma el contenido del registro W, el flag de Carry y el contenido del registro f, y almacena el resultado en W si d = 0, y en el registro f si d = 1. Si se produce acarreo el flag C se pone a "1". Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.

Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe W en destino

Ejemplo: *addwfc Registro,0,1* ; (Registro) + (W) → (W)
Antes Instrucción: (W) = 0x4D, (Registro) = 0x02, y C = 1.
Después Instrucción: (W) = 0x50, (Registro) = 0x02, y C = 0.

andlw — W AND Literal k

Sintaxis: [*Etiqueta*] andlw k
Operandos: $0 \le k \le 255$
Operación: (W) AND (k) → (W)
Flags afectados: N,Z
Código de OP :

0000	1011	kkkk	kkkk

Descripción: Efectúa la operación AND lógica entre el contenido del registro W y el literal k, el resultado se almacena en W.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "K"	Procesa datos	Escribe en W

Ejemplo: andlw b'01011111' ; (W).AND. b'01011111' → (W).
Antes Instrucción: (W) = b'10100011' y Z = ¿?.
Después Instrucción: (W) = b'00000011' y Z = 0.

andwf — W AND f

Sintaxis: [*Etiqueta*] andwf f[,d[,a]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (W) AND (f) → (destino)
Flags afectados: N,Z
Código de OP :

0001	01da	dfff	ffff

Descripción: Efectúa la operación AND lógica entre el contenido del registro W y el contenido del registro f, y almacena el resultado en W si d = 0, y en f si d = 1. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el Registro "f"	Procesa datos	Escribe en destino

Ejemplo1: andwf Registro,0,0 ; (W).AND.(Registro) → (Registro)
Antes Instrucción: (W) = b'0010111', (Registro) = b'11000010' y Z = ¿?.
Después Instrucción: (W) = b'0000010', (Registro) = b'11000010' y Z = 0.

bc — Salta si hay acarreo

Sintaxis: [*Etiqueta*] bc n
Operandos: $-128 \le a \le 127$
Operación: Si el bit de carry es "1"
(PC) + 2 + 2n → (PC)
Flags afectados: ninguno
Código de OP :

1110	0010	nnnn	nnnn

Descripción: Si el bit de Carry es "1", entonces el programa hace un salto relativo.
El complemento a dos del número 2n se le suma al PC. Como al ejecutar la instrucción se trae la siguiente instrucción que se debería ejecutar y que luego no se ejecuta, el contador de programa toma el valor PC+2+2n. Por lo tanto esta instrucción tiene dos ciclos de instrucción.

Palabras: 1
Ciclos: 1(2)
Ciclos de actividad Q
Si salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	Escribe en el PC
No operación	No operación	No operación	No operación

Si no salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal “n”	Procesa datos	No operación

```
Ejemplo1:          AQUI   bc   5          ;si C=1 (PC) → (PC+2+2n)
                                          ;si C=0 (PC) → (PC+2)
Antes  Instrucción:    PC = dirección (AQUÍ)

Después Instrucción:   si Carry =1
                              PC = dirección (AQUÍ +12)
                       si Carry = 0
                              PC = dirección (AQUÍ + 2)
```

bcf — Borra un bit de f

Sintaxis: [*Etiqueta*] bcf f,b[,a]
Operandos: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$
Operación: 0 → (f <b>)
Flags afectados: Ninguno
Código de OP :

1001	bbba	ffff	ffff

Descripción: Pone a cero el bit número b del registro f. Si “a” es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si “a”=1 se usa el BSR para indicar el banco de registro seleccionado (por defecto).
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro “f”	Procesa datos	Escribe en el registro “f”

```
Ejemplo:               bcf   FlagReg,7,0        ; 0 → (FlagReg,7)
Antes Instrucción:     (FlagReg) = b’11000111’.
Después Instrucción:   (FlagReg) = b’01000111’.
```

bn — Salta si es negativo

Sintaxis: [*Etiqueta*] bn n
Operandos: $-128 \le a \le 127$
Operación: Si el bit de Negativo es “1”
(PC) + 2 + 2n → (PC)
Flags afectados: ninguno
Código de OP :

1110	0010	nnnn	nnnn

Descripción: Si el bit de Negativo es “1”, entonces el programa hace un salto relativo.
El complemento a dos del número 2n se le suma al PC. Como al ejecutar la instrucción se trae la siguiente instrucción que se debería ejecutar y que luego no

se ejecuta, el contador de programa toma el valor PC+2+2n. Por lo tanto esta instrucción tiene dos ciclos de instrucción.

Palabras: 1
Ciclos: 1(2)
Ciclos de actividad Q
Si salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	Escribe en el PC
No operación	No operación	No operación	No operación

Si no salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	No operación

Ejemplo1: AQUI bn SALTO ;si N=1 (PC) → (SALTO)
;si N=0 (PC) → (PC+2)
Antes Instrucción: PC = dirección (AQUÍ)

Después Instrucción: si Negativo =1
PC = dirección (SALTO)
Si Negativo = 0
PC = dirección (AQUÍ + 2)

bnc — Salta si no hay acarreo

Sintaxis: [*Etiqueta*] bnc n
Operandos: $-128 \le a \le 127$
Operación: Si el bit de Carry es "0"
(PC) + 2 + 2n → (PC)
Flags afectados: ninguno
Código de OP : 1110 | 0011 | nnnn | nnnn
Descripción: Si el bit de Carry es "0", entonces el programa hace un salto relativo.
El complemento a dos del número 2n se le suma al PC. Como al ejecutar la instrucción se trae la siguiente instrucción que se debería ejecutar y que luego no se ejecuta, el contador de programa toma el valor PC+2+2n. Por lo tanto esta instrucción tiene dos ciclos de instrucción.

Palabras: 1
Ciclos: 1(2)
Ciclos de actividad Q
Si salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	Escribe en el PC
No operación	No operación	No operación	No operación

Si no salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	No operación

```
Ejemplo1:              AQUI   bnc   SALTO    ;si C=0 (PC) → (SALTO)
                                              ;si C=1 (PC) → (PC+2)
Antes  Instrucción:    PC = dirección (AQUÍ)

Después Instrucción:   si Carry = 0
                              PC = dirección (SALTO)
                       Si Carry = 1
                              PC = dirección (AQUÍ + 2)
```

bnn — Salta si no es negativo

Sintaxis: [*Etiqueta*] bnn n
Operandos: $-128 \leq a \leq 127$
Operación: Si el bit de Negativo es "0"
(PC) + 2 + 2n → (PC)
Flags afectados: ninguno
Código de OP :

1110	0111	nnnn	nnnn

Descripción: Si el bit de Negativo es "0", entonces el programa hace un salto relativo. El complemento a dos del número 2n se le suma al PC. Como al ejecutar la instrucción se trae la siguiente instrucción que se debería ejecutar y que luego no se ejecuta, el contador de programa toma el valor PC+2+2n. Por lo tanto esta instrucción tiene dos ciclos de instrucción.

Palabras: 1
Ciclos: 1(2)
Ciclos de actividad Q
Si salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	Escribe en el PC
No operación	No operación	No operación	No operación

Si no salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	No operación

```
Ejemplo1:              AQUI   bn   SALTO     ;si N=0 (PC) → (SALTO)
                                              ;si N=1 (PC) → (PC+1)
Antes  Instrucción:    PC = dirección (AQUÍ)

Después Instrucción:   si Negativo =0
                              PC = dirección (SALTO)
                       Si Negativo = 1
                              PC = dirección (AQUÍ + 2)
```

bnov — Salta si no overflow

Sintaxis: [*Etiqueta*] bnov n
Operandos: -128 ≤ a ≤ 127
Operación: Si el bit de Overflow es "0"
(PC) + 2 + 2n → (PC)
Flags afectados: ninguno
Código de OP :

1110	0101	nnnn	nnnn

Descripción: Si el bit de Overflow es "0", entonces el programa hace un salto relativo.
El complemento a dos del número 2n se le suma al PC. Como al ejecutar la instrucción se trae la siguiente instrucción que se debería ejecutar y que luego no se ejecuta, el contador de programa toma el valor PC+2+2n. Por lo tanto esta instrucción tiene dos ciclos de instrucción.

Palabras: 1
Ciclos: 1(2)
Ciclos de actividad Q
Si salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	Escribe en el PC
No operación	No operación	No operación	No operación

Si no salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	No operación

```
Ejemplo1:              AQUI   bnov   SALTO   ;si OV=0 (PC) → (SALTO)
                                              ;si OV=1 (PC) → (PC+1)
Antes  Instrucción:    PC = dirección (AQUÍ)

Después Instrucción:   si Overflow =0
                              PC = dirección (SALTO)
                       Si Overflow = 1
                              PC = dirección (AQUÍ + 2)
```

bnz — Salta si no es cero

Sintaxis: [*Etiqueta*] bnz n
Operandos: -128 ≤ a ≤ 127
Operación: Si el bit de Zero es "0"
(PC) + 2 + 2n → (PC)
Flags afectados: ninguno
Código de OP :

1110	0001	nnnn	nnnn

Descripción: Si el bit de Zero es "0", entonces el programa hace un salto relativo.
El complemento a dos del número 2n se le suma al PC. Como al ejecutar la instrucción se trae la siguiente instrucción que se debería ejecutar y que luego no se ejecuta, el contador de programa toma el valor PC+2+2n. Por lo tanto esta instrucción tiene dos ciclos de instrucción.

Palabras: 1
Ciclos: 1(2)
Ciclos de actividad Q
Si salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	Escribe en el PC
No operación	No operación	No operación	No operación

Si no salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	No operación

```
Ejemplo1:              AQUI   bnz   SALTO    ;si Z=0 (PC) → (SALTO)
                                             ;si Z=1 (PC) → (PC+1)
Antes  Instrucción:    PC = dirección (AQUÍ)

Después Instrucción:   si Zero =0
                              PC = dirección (SALTO)
                       si Zero = 1
                              PC = dirección (AQUÍ + 2)
```

bra — Salta incondicional

Sintaxis: [*Etiqueta*] bra n
Operandos: $-1024 \le n \le 1023$
Operación: (PC) + 2 + 2n → (PC)
Flags afectados: ninguno
Código de OP : 1101 | 0nnn | nnnn | nnnn
Descripción: El programa hace un salto relativo. El complemento a dos del número 2n se le suma al PC. Como al ejecutar la instrucción se trae la siguiente instrucción que se debería ejecutar y que luego no se ejecuta, el contador de programa toma el valor PC+2+2n. Por lo tanto esta instrucción tiene dos ciclos de instrucción.

Palabras: 1
Ciclos: 1(2)
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	Escribe en el PC
No operación	No operación	No operación	No operación

```
Ejemplo1:              AQUI   bra   SALTO    ; (PC) → (SALTO)

Antes  Instrucción:    PC = dirección (AQUÍ)

Después Instrucción:   PC = dirección (SALTO)
```

bsf — Activa un bit de f

Sintaxis: [*Etiqueta*] bsf f,b[,a]
Operandos: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$
Operación: 1 → (f<b>)
Flags afectados: Ninguno
Código de OP:

1000	bbba	ffff	ffff

Descripción: Pone a uno el bit número b del registro f. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el Registro "f"	Procesa datos	Escribe en el registro "f"

Ejemplo: bsf FlagReg, 7, 1 ; 1 → (FlagReg,7)
Antes Instrucción: (FlagReg) = b'00001010'.
Después Instrucción: (FlagReg) = b'10001010'.

btfsc — Test de bit de f y salta si es cero

Sintaxis: [*Etiqueta*] btfsc f,b[,a]
Operandos: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$
Operación: Salta si (f<b>) = 0
Flags afectados: Ninguno
Código de OP :

1011	bbba	ffff	ffff

Descripción: Si el bit número b del registro f es cero, la instrucción que sigue a esta se ignora y se trata como un *"nop"*. En este caso y solo en este caso, la instrucción *btfsc* precisa dos ciclos para ejecutarse. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1(2)
Nota: 3 ciclos si el salto y lleva a cavo una instrucción de la segunda palabra.
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el Registro "f"	Procesa datos	No operación

Si salta

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación

Si salta y lleva a cavo una instrucción de la segunda palabra

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación
No operación	No operación	No operación	No operación

Ejemplo:

```
        Aqui    btfsc   Flag, 1, 0      ; Si el bit 1 del registro "Flag" es cero salta.
        Falso   goto    ProcesoX        ; Ha sido uno.
        Verdad          ...             ; Ha sido cero.
                                        ...
```

Antes Instrucción: (PC) = Dirección de "Aquí".
Después Instrucción: Si el bit Flag <1> = 0, (PC) = Dirección de "Verdad".
Si el bit Flag <1> = 1, (PC) = Dirección de "Falso".

btfss — Test de bit de f y salta si es uno

Sintaxis: [*Etiqueta*] BTFSS f,b[,a]
Operandos: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$
Operación: Salta si (f<b>) = 1
Flags afectados: Ninguno
Código de OP :

1010	bbba	ffff	ffff

Descripción: Si el bit número b del registro f es uno, la instrucción que sigue a ésta se ignora y se trata como un *"nop"*. En este caso y solo en este caso, la instrucción *btfss* precisa dos ciclos para ejecutarse. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1(2)
Nota: 3 ciclos si el salto y lleva a cavo una instrucción de la segunda palabra.
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el Registro "f"	Procesa datos	No operación

Si salta

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación

Si salta y lleva a cavo una instrucción de la segunda palabra

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación
No operación	No operación	No operación	No operación

Ejemplo:

```
        Aqui    btfss   Flag, 1, 0      ; Si el bit 1 del registro "Flag" es uno salta.
        Falso   goto    ProcesoX        ; Ha sido cero.
        Verdad          ...             ; Ha sido uno.
                                        ...
```

Antes Instrucción: (PC) = Dirección de "Aquí".
Después Instrucción: Si el bit Flag <1> = 0, (PC) = Dirección de "Falso".
Si el bit Flag <1> = 1, (PC) = Dirección de "Verdad".

btg — bit toggle de f

Sintaxis: [*Etiqueta*] BTG f,b[,a]

Operandos: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operación: Salta si (f<$\bar{b}$>) → f<b>

Flags afectados: Ninguno

Código de OP :

0111	bbba	ffff	ffff

Descripción: El bit "b" de la posición de memoria "f" se invierte. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.

Palabras: 1

Ciclos: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el Registro "f"	Procesa datos	Escribe en el registro "f"

Ejemplo:

```
        btg     PORTC, 4, 0     ; complenta el valor del bit 4 del PORTC.
                                ...
```

Antes Instrucción: PORTC = 0111 0101 [0x75]

Después Instrucción: PORTC = 0110 0101 [0x65]

bov — Salta si overflow

Sintaxis: [*Etiqueta*] bov n

Operandos: $-128 \le a \le 127$

Operación: Si el bit de Overflow es "1"
(PC) + 2 + 2n → (PC)

Flags afectados: ninguno

Código de OP :

1110	0100	nnnn	nnnn

Descripción: Si el bit de Overflow es "1", entonces el programa hace un salto relativo.
El complemento a dos del número 2n se le suma al PC. Como al ejecutar la instrucción se trae la siguiente instrucción que se debería ejecutar y que luego no se ejecuta, el contador de programa toma el valor PC+2+2n. Por lo tanto esta instrucción tiene dos ciclos de instrucción.

Palabras: 1

Ciclos: 1(2)

Ciclos de actividad Q

Si salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	Escribe en el PC
No operación	No operación	No operación	No operación

Si no salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	No operación

```
Ejemplo1:              AQUI   bov   SALTO    ;si OV=1 (PC) → (SALTO)
                                              ;si OV=0 (PC) → (PC+1)
Antes  Instrucción:    PC = dirección (AQUÍ)

Después Instrucción:   si OV =1
                              PC = dirección (SALTO)
                       si OV = 0
                              PC = dirección (AQUÍ + 2)
```

bz — Salta si es cero

Sintaxis: [*Etiqueta*] bz n
Operandos: $-128 \leq a \leq 127$
Operación: Si el bit de Zero es "1"
(PC) + 2 + 2n → (PC)
Flags afectados: ninguno
Código de OP :

1110	0000	nnnn	nnnn

Descripción: Si el bit de Zero es "1", entonces el programa hace un salto relativo.
El complemento a dos del número 2n se le suma al PC. Como al ejecutar la instrucción se trae la siguiente instrucción que se debería ejecutar y que luego no se ejecuta, el contador de programa toma el valor PC+2+2n. Por lo tanto esta instrucción tiene dos ciclos de instrucción.

Palabras: 1
Ciclos: 1(2)
Ciclos de actividad Q
Si salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	Escribe en el PC
No operación	No operación	No operación	No operación

Si no salta

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n"	Procesa datos	No operación

```
Ejemplo1:              AQUI   bz   SALTO     ;si Z=1 (PC) → (SALTO)
                                              ;si Z=0 (PC) → (PC+1)
Antes  Instrucción:    PC = dirección (AQUÍ)

Después Instrucción:   si Zero =1
                              PC = dirección (SALTO)
                       si Zero = 0
                              PC = dirección (AQUÍ + 2)
```

call — Llamada a Subrutina

Sintaxis: [*Etiqueta*] call k [,s]
Operandos: $0 \le K \le 1048575$
$s \in [0,1]$
Operación: (PC)+4 → TOS
k → (PC<20:1>),
si s=1
(W) → WS
(STATUS) → STATUSS
(BSR) → BSRS
Flags afectados: Ninguno
Código de OP :

1110	110s	$k_7$kkk	kkkk$_0$
1111	k_{19}kkk	kkkk	kkkk$_8$

Descripción: Llamada a subrutina dentro de un rango de 2 Mb ytes. En primer lugar se Salvaguarda la dirección de vuelta en la Pila (PC+4) si "s"=1, además W, el registro de STATUS y BSR se guardan en sus registros de guarda WS, STATUSS y BSRS respectivamente. Si "s"=0 no se guardan estos registros y los 20 bits del valor "k" se cargan en el PC<20:1>. Tarda dos ciclos máquina en ejecutarse.

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "k" <7:0>	Push PC en el STACK	Lee literal K<19:8> y escribe en el PC
No operación	No operación	No operación	No operación

Ejemplo: Aqui call Alli,1
Antes Instrucción: (PC) = Dirección de "Aqui".
Después Instrucción: (PC) = Dirección de "Alli".
(TOS) = Dirección de ("Aqui" + 4)
WS = W
BSRS = BSR
STATUSS = STATUS

clrf — Borra f

Sintaxis: [Etiqueta] clrf f
Operandos: $0 < f < 255$
$a \in [0,1]$
Operación: 00h ((f)
1 → Z
Flags afectados: Z
Código de OP :

0110	101a	ffff	ffff

Descripción: Se borra el contenido del registro f y el flag Z se activa poniéndose a "1". Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee registro f	Procesa datos	Escribe en el reg f
No operación	No operación	No operación	No operación

Ejemplo: clrf FlagReg,1 ; 0 ((FlagReg).
Antes Instrucción: (FlagReg) = 0x5A y Z = ¿?
Después Instrucción: (FlagReg) = 0x00 y Z = 1.

clrwdt — Borra el Timer del Watchdog

Sintaxis: [Etiqueta] clrwdt
Operandos: Ninguno
Operación: 00h → (WDT)
000h → WDT postscaler,
1 → TO,
1 → PD
Flags afectados: TO y PD
Código de OP : | 0000 | 0000 | 0000 | 0100 |
Descripción: Se borra tanto el Timer del WDT (Watchdog) como su postscaler. Los bits TO y PD se ponen a 1

Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	No operación	Procesa datos	No operación

Ejemplo: clrwdt ; 0 ((FlagReg).
Antes Instrucción: (Timer WDT) = ¿?
Después Instrucción: (Timer WDT) = 0x00
WDT Postscaler = 0
TO = 1
PD = 1

comf — Cpmplementa f

Sintaxis: Etiqueta] comf f
Operandos: $0 \le f \le 255$
$d \in 0,1]$
$a \in [0,1]$

Operación: $(\bar{f})$ → destino

Flags afectados: N,Z

Código de OP : | 0001 | 11da | ffff | ffff |
Descripción: Hace el complemento del contenido del registro f bit a bit. El resultado se almacena en el registro f si d=1 y en el registro W si d=0, en este caso f no varía.

Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.

Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee registro f	Procesa datos	Escribe en destino

Ejemplo: comf Reg1,0,0 ; (/Reg1) ((W)
Antes Instrucción: (Reg1) = b'00010011', (W) = ¿? y Z = ¿?. N= ¿?.
Después Instrucción: (Reg1) = b'00010011', (W) = b'11101100' (invertido unos y ceros) , Z =0 y N=1.

cpfseq — Compara f con W, salta si f = W

Sintaxis: [Etiqueta] cpfseq f[,a]

Operandos: 0 < f < 255
a ∈ [0,1]
Operación: (f)-(W),
Salta si (f) = (W)
(comparación sin resultado)
Flags afectados: Ninguno

Código de OP :

0110	001a	ffff	ffff

Descripción: Compara el contenido del registro "f" con el contenido de W realizando una resta sin resultado. Si "f" = W la instrucción siguiente se desecha y se ejecuta una instrucción NOP en su lugar, cuando esto ocurre la instrucción tarda dos ciclos máquina en ejecutarse. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1 (2) Nota: 3 ciclos si después de la condición del salto se encuentra una instrucción de dos palabras
Ciclos de actividad Q

Decodificación	Lee Registro 2f"	Procesa datos	No operación

Si salta

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación

Si después del salto hay una instrucción de 2 palabras

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación
No operación	No operación	No operación	No operación

```
Ejemplo:        AQUÍ        cpfseq  Reg, 0      ; Salta si (REG) =W
                NO_IGUAL    :
                IGUAL       :
```

Antes Instrucción: Dirección PC = AQUI
(W) = ¿?
Reg = ¿?
Después Instrucción:
Si Reg = W
PC = (IGUAL)
Si Reg ≠ W
PC = (NO_IGUAL)

cpfsgt — Compara f con W, salta si f > W

Sintaxis: [Etiqueta] cpfsgt f[,a]

Operandos: $0 \leq f \leq 255$

$a \in [0,1]$

Operación: (f) - (W),

Salta si (f) > (W)

(comparación sin resultado)

Flags afectados: Ninguno

Código de OP :

0110	010a	ffff	ffff

Descripción: Compara el contenido del registro "f" con el contenido de W realizando una resta sin resultado. Si el contenido de "f" es mayor que el de W la instrucción siguiente se desecha y se ejecuta una instrucción NOP en su lugar, cuando esto ocurre la instrucción tarda dos ciclos máquina en ejecutarse. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee Registro 2f"	Procesa datos	No operación

Si salta

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación

Si después del salto hay una instrucción de 2 palabras

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación
No operación	No operación	No operación	No operación

```
Ejemplo:        AQUÍ        cpfsgt  Reg, 0      ; Salta si (REG) =W
                NO_IGUAL    :
                IGUAL       :

Antes Instrucción:      Dirección PC = AQUI
```

```
            (W) = ¿?
            Reg = ¿?
      Después Instrucción:
            Si  Reg > W
                  PC = (IGUAL)
            Si Reg < W
                  PC = (NO_IGUAL)
```

cpfslt — Compara f con W, salta si f < W

Sintaxis: [*Etiqueta*] cpfslt f[,a]
Operandos: $0 \le f \le 255$
$a \in [0,1]$
Operación: (f)-(W),
Salta si (f) < (W)
(comparación sin resultado)
Flags afectados: Ninguno
Código de OP :

0110	000a	ffff	ffff

Descripción: Compara el contenido del registro "f" con el contenido de W realizando una resta sin resultado. Si el contenido de "f" es menor que el de W la instrucción siguiente se desecha y se ejecuta una instrucción NOP en su lugar, cuando esto ocurre la instrucción tarda dos ciclos máquina en ejecutarse. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1(2)
Nota: 3 ciclos si después de la condición del salto se encuentra una instrucción de dos palabras.
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	No operación

Si salta

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación

Si después del salto hay una instrucción de 2 palabras

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación
No operación	No operación	No operación	No operación

```
Ejemplo:          AQUÍ        cpfslt   Reg, 1        ; Salta si (REG) <W
                  NO_MENOR    :
                  MENOR       :

Antes Instrucción:      Dirección PC = AQUI
            (W) = ¿?
            Reg = ¿?
      Después Instrucción:
            Si  Reg =>W
                  PC = (MENOR)
            Si Reg  ≥ W
```

PC = (NO_MENOR)

daw — Ajuste decimal del registro W

Sintaxis: [*Etiqueta*] daw
Operandos: Ninguno

Operación: si [W<3:0> >9 o [DC=1] entonces
(W<3:0>) + 6 → W<3:0>;
sino
(W<3:0>) → W<3:0>;
si [W<7:4> >9 o [C=1] entonces
(W<7:4>) + 6 → W<7:4>;
sino
(W<7:4>) → W<7:4>;

Flags afectados: C
Código de OP : 0000 | 0000 | 0000 | 0111
Descripción: DAW ajusta el valor de los ocho bits del registro W, resultado de la suma de dos variables (cada una empaquetada en BCD) y produce un resultado correcto empaquetado en BCD.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro W	Procesa datos	Escribe en W

Ejemplo 1: daw ; Ajusta a BCD

Antes Instrucción:
(W) = 10100101
C = 0
DC = 0
Después Instrucción:
(W) = 00000101
C = 1
DC = 0

Ejemplo 2:
Antes Instrucción:
(W) = 11001110
C = 0
DC = 0
Después Instrucción:
(W) = 00110100
C = 1
DC = 0

dec — Decrementa f

Sintaxis: [*Etiqueta*] decf f[,d[a]]
Operandos: $0 \le f \le 255$
$d \in [0,1]$

a ∈ [0,1]
Operación: (f)-1 → (destino)
Flags afectados: C,DC,N,OV,Z
Código de OP :

0000	01da	Ffff	Ffff

Descripción: Se decrementa el contenido del registro f en una unidad. El resultado se almacena en f si d=1 y en W si d=0, en este caso f no varía. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	Escribe en destino

Ejemplo1: decf Contador, 1,0 ; (Contador) – 1 → (Contador).
Antes Instrucción: (Contador) = 0x01 y Z = ¿?.
Después Instrucción: (Contador) = 0x00 y Z =1.

decfsz — Decrementa f y salta si el resultado es 0

Sintaxis: [*Etiqueta*] decfsz f [,d[,a]]
Operandos: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]
Operación: (f) - 1 → (destino); Salta si el resultado es 0
Flags afectados: Ninguno
Código de OP :

0010	11da	ffff	ffff

Descripción: Decrementa el contenido del registro f en una unidad, el resultado se almacena en f si d=1 y en W si d=0, en este caso, f no varía. Si el resultado del decremento es cero, se ignora la siguiente instrucción y se ejecuta un NOP y en ese caso la instrucción tiene una duración de dos ciclos. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1(2)
Nota: 3 ciclos si después de la condición del salto se encuentra una instrucción de dos palabras.
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	Escribe en destino

Si salta

Q1	Q2	Q3	Q4
No Operación	No Operación	No Operación	No Operación

Si salta una instrucción de dos palabras

Q1	Q2	Q3	Q4
No Operación	No Operación	No Operación	No Operación
No Operación	No Operación	No Operación	No Operación

Ejemplo:

```
Aqui     decfsz   Contador, 1, 1
         goto     NoEsCero
EsCero            ....
                  ....
```

....

Antes Instrucción: (PC) = Dirección de "Aquí".
Después Instrucción: (Contador) = (Contador) – 1 y además:
- Si (Contador) = 0, (PC) = Dirección de "EsCero".
- Si (Contador) ≠ 0, (PC) = Dirección de "Aquí" + 2.

decfnz — Decrementa f y salta si el resultado no es 0

Sintaxis: [*Etiqueta*] decfnz f [,d[,a]]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (f) - 1 → (destino); Salta si el resultado es ≠ 0
Flags afectados: Ninguno
Código de OP :

0100	11da	ffff	ffff

Descripción: Decrementa el contenido del registro f en una unidad, el resultado se almacena en f si d=1 y en W si d=0, en este caso, f no varía. Si el resultado del decremento es distinto de cero, se ignora la siguiente instrucción y se ejecuta un NOP y en ese caso la instrucción tiene una duración de dos ciclos. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1(2)
Nota: 3 ciclos si después de la condición del salto se encuentra una instrucción de dos palabras.
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	Escribe en destino

Si salta

Q1	Q2	Q3	Q4
No Operación	No Operación	No Operación	No Operación

Si salta una instrucción de dos palabras

Q1	Q2	Q3	Q4
No Operación	No Operación	No Operación	No Operación
No Operación	No Operación	No Operación	No Operación

Ejemplo:

```
Aqui      decfnz   Contador, 1, 0
Es_Cero   :
No_Cero   :
```

Antes Instrucción: (PC) = Dirección de "Aqui".
Después Instrucción: (Contador) = (Contador) – 1 y además:
- Si (Contador) = 0, (PC) = Dirección de "Es_Cero".
- Si (Contador) ≠ 0, (PC) = Dirección de "No_Cero".

goto — Salto incondicional

Sintaxis: [*Etiqueta*] goto k
Operandos: $0 \le k \le 1048575$
Operación: k → (PC<20:1>
Flags afectados: Ninguno
Código de OP :

1110	1111	$k_7$kkk	Kkkk$_0$
1111	k_{19}kkk	kkkk	Kkkk$_8$

Descripción: Realiza un salto incondicional dentro del rango de 2 Mbytes de memoria. Los 20 bits del valor "k" se cargan en el PC<20:1>. Esta instrucción se ejecuta en dos ciclos máquina.
Palabras: 2
Ciclos: 2
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "k"<7:0>	No operación	Lee el literal "k"<19:8>
No operación	No operación	No operación	No operación

Ejemplo: goto Principal ; Principal → (PC)
Antes Instrucción: (PC) = ¿?.
Después Instrucción: (PC) = Dirección apuntada por la etiqueta "Principal".

incf — Incrementa f

Sintaxis: [*Etiqueta*] incf f [,d[,a]]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (f) +1 → (destino)
Flags afectados: C,DC,N,OV,Z
Código de OP :

0010	10da	ffff	ffff

Descripción: El contenido del registro f se incrementa en una unidad, si d=1 el resultado se almacena en f, si d=0 el resultado se almacena en W, en este caso el resultado de f no varía. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	Escribe en destino

Ejemplo1: incf Contador, 1, 0 ; (Contador) + 1 → (Contador).
Antes Instrucción: (Contador) = 0xFF y Z = ¿?, C = ¿?, DC = ¿?.
Después Instrucción: (Contador) = 0x00 y Z = 1, C = 1, DC = 1

incfsz — Incrementa f y salta si el resultado es 0

Sintaxis: [*Etiqueta*] incfsz f [,d[a,]]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (f) + 1 → (destino); Salta si el resultado es 0
Flags afectados: Ninguno
Código de OP :

0011	11da	ffff	ffff

Descripción: Incrementa el contenido del registro f en una unidad. El resultado se almacena en f si d=1 y en W si d=0, en este caso, f no varía. Si el resultado del incremento es cero, se ignora la siguiente instrucción y, en ese caso la instrucción tiene una duración de dos ciclos máquina. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1(2)
Nota: 3 ciclos si después de la condición del salto se encuentra una instrucción de dos palabras.
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	Escribe en destino

Si salta

Q1	Q2	Q3	Q4
No Operación	No Operación	No Operación	No Operación

Si salta una instrucción de dos palabras

Q1	Q2	Q3	Q4
No Operación	No Operación	No Operación	No Operación
No Operación	No Operación	No Operación	No Operación

Ejemplo:

```
Aqui      incfsz  Contador, 1, 0
N0_Cero   :
Cero      :
```

Antes Instrucción: (PC) = Dirección de "Aquí".
Después Instrucción: (Contador) = (Contador) + 1 y además:
- Si Contador = 0 (256), (PC) = Dirección de "Cero".
- Si Contador ≠ 0, (PC) = Dirección de "No_Cero

infsnz — Incrementa f y salta si el resultado no es 0

Sintaxis: [*Etiqueta*] infsnz f [,d[a,]]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (f) + 1 → (destino); Salta si el resultado no es 0
Flags afectados: Ninguno
Código de OP :

0100	10da	ffff	ffff

Descripción: Incrementa el contenido del registro f en una unidad. El resultado se almacena en f si d=1 y en W si d=0, en este caso, f no varía. Si el resultado del incremento no es cero, se ignora la siguiente instrucción y, en ese caso la instrucción tiene una

duración de dos ciclos máquina. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.

Palabras: 1

Ciclos: 1(2)

Nota: 3 ciclos si después de la condición del salto se encuentra una instrucción de dos palabras.

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	Escribe en destino

Si salta

Q1	Q2	Q3	Q4
No Operación	No Operación	No Operación	No Operación

Si salta una instrucción de dos palabras

Q1	Q2	Q3	Q4
No Operación	No Operación	No Operación	No Operación
No Operación	No Operación	No Operación	No Operación

Ejemplo:

```
Aqui      incfsz   Contador, 1, 0
Cero      :
No_Cero   :
```

Antes Instrucción: (PC) = Dirección de "Aquí".

Después Instrucción: (Contador) = (Contador) + 1 y además:

- Si Contador ≠ 0, (PC) = Dirección de "No_Cero"
- Si Contador = 0 (256), (PC) = Dirección de "Cero".

iorlw — OR entre W y el lilteral k

Sintaxis: [*Etiqueta*] iorlw k

Operandos: $0 \le f \le 255$

Operación: (W) OR (k) → W

Flags afectados: N,Z

Código de OP :

0000	1001	kkkk	kkkk

Descripción: Efectúa la operación lógica OR entre el contenido del registro W y el literal k. El resultado se almacena en el registro W.

Palabras: 1

Ciclos: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "k"	Procesa datos	Escribe en W

```
Ejemplo:            iorlw   b'00110101'.   ; (W) OR b'00110101' → (W)
Antes Instrucción:  (W) =   b'10011010'  y  Z = ¿?.
Después Instrucción: (W) =  b'10111111'  y  Z = 0.
```

iorwf — OR entre W y f

Sintaxis: [*Etiqueta*] iorwf f [,d[,a]]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (W) OR (f) → (destino)
Flags afectados: N,Z
Código de OP :

0001	00da	ffff	ffff

Descripción: Efectúa la operación lógica OR entre el contenido del registro W y el contenido del registro f. Almacena el resultado en f si d=1 y en W si d=0. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	Escribe en destino

Ejemplo: iorwf Resultado, 0,1 ; (W) OR (Resultado) → (W)
Antes Instrucción: (Resultado) = b'00010011' , (W) = b'10010001' y Z = ¿?.
Después Instrucción: (Resultado) = b'00010011' , (W) = b'10010011' y Z = 0.

lfsr — Carga el FSR

Sintaxis: [*Etiqueta*] lfsr f ,k
Operandos: $0 \le f \le 2$
$0 \le k \le 4095$
Operación: k → FSRf
Flags afectados: Ninguno
Código de OP :

1110	1110	00ff	k_{11}kkk
1111	0000	$k_7$kkk	kkkk

Descripción: Los 12 bits del literal "k" se cargan en el registro apuntado por el registro "f", que puede valer 0,1 'o 2.
Palabras: 2
Ciclos: 2
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "k" MSB	Procesa datos	Escribe en literal los MSB "k" en FSRfH
Decodificación	Lee el literal "k" LSB	Procesa datos	Escribe en literal los LSB "k" en FSRfL

Ejemplo: lfsr 2, 0x3AB ; (W) OR (Resultado) → (W)
Después Instrucción: FSR2H = 0x03
FSR2L = 0xAB

movff — Mueve f a f

Sintaxis: [*Etiqueta*] movff fs ,fd
Operandos: $0 \leq f_s \leq (4095$
$0 \leq f_d \leq (4095$
Operación: $(f_s) \rightarrow (f_d)$
Flags afectados: Ninguno

Código de OP :

1100	ffff	ffff	$Ffff_s$
1111	ffff	ffff	$ffff_d$

Descripción: El contenido de registro fuente 'fs ' se mueven al registro destino 'fd '. El registro 'fs ' puede estar en cualquier parte en los 4096 bytes FSR (desde 000h a FFFh), y el registro destino 'fd ' también puede ser en cualquier registro del 000h a FFFh.
Tanto el registro fuente como el destino pueden ser W (una útil situación especial).
MOVFF es particularmente útil para transferir una posición de memoria de datos a un registro periférico.
La instrucción de MOVFF no puede usar el PCL, TOSU, TOSH o TOSL como el registro del destino.
La nota: La instrucción de MOVFF no debe usarse para modificar la configuración de las interrupciones mientras cualquier interrupción esté habilita.

Palabras: 2

Ciclos: 2 (3)

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f (scr)	Procesa datos	No Operación
Decodificación	No Operación	No Operación	Escribe en registro “f” (destino)

Ejemplo: movff Reg1, Reg2 ; (Reg1) (Reg2
Antes de la instrucción: Reg1 = 0x33
Reg2 = 0x11
Después Instrucción: Reg1 = 0x33
Reg2 = 0x33

movlb — mueve literal a nibble bajo BSR

Sintaxis: [Etiqueta] movlb k
Operandos: $0 \leq f \leq 255$
Operación: k →BSR
Flags afectados: Ninguno

Código de OP :

0001	0001	kkkk	kkkk

Descripción: El valor de 8 bits del literal k se carga en el Registro de Selección de Banco (BSR)
Palabras: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "k"	Procesa datos	Escribe el literal K en BSR

Ejemplo: movlb 0x05 ; 05h (BSR.
Antes Instrucción: Registro BSR = 0x02
Después Instrucción: Registro BSR = 0x05

movlw k — mueve literal W

Sintaxis: [Etiqueta] movlw k
Operandos: 0 ≤f ≤255
Operación: k →W
Flags afectados: Ninguno

Código de OP :

0000	1110	kkkk	kkkk

Descripción: El registro W se carga con el valor de 8 bits del literal k
Palabras: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "k"	Procesa datos	Escribe el literal W

Ejemplo: movlw 0x5A ; 5Ah → (W).
Antes Instrucción: (W) = ¿?
Después Instrucción: (W) = 0x5A

movwf — mueve W a f

Sintaxis: [Etiqueta] movwf f{,a}
Operandos: 0 ≤f ≤255
a∈[0,1]
Operación: (W) →f
Flags afectados: Ninguno

Código de OP :

0110	111a	ffff	ffff

Descripción: Mueve el contenido del registro W al registro f . El registro "f" puede ser cualquiera de los 256 bytes del banco. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro W	Procesa datos	Escribe en el registro

Ejemplo: movwf Reg, 0 ; (W) ((Reg)

Antes Instrucción: (Reg) = 0xFF y (W) = 0x4F.

Después Instrucción: (Reg) = 0x4F y (W) = 0x4F.

mullw multiplica literal por W

Sintaxis: [Etiqueta] mullw k
Operandos: $0 \leq k \leq 255$
$a \in [0,1]$
Operación: (W) x k →PRODH:PRODL
Flags afectados: Ninguno

Código de OP :

0000	1101	kkkk	kkkk

Descripción: Realiza la multiplicación sin acarreo del contenido del registro W con los 8 bits del literal "k".El resultado de 16-bits se guarda en la pareja de registros PRODH:PRODL .PRODH guarda el byte alto. W permanece inalterado. Ninguno de los flags del registro de STATUS se ve afectado
Nota: No es posible que se desborde el resultado de la operación. Un resultado cero es posible pero no es detectado.
Palabras: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "k"	Procesa datos	Escribe en los registros PRODH:PRODL

Ejemplo: mullw b'11000100' ; (W) x k (PRODH:PRODL

Antes Instrucción: (W) = 11100010 PRODH = ¿? Y PRODL = ¿?.

Después Instrucción: (W) = 11100010 PRODH = 10101101 Y PRODL= 00001000

mulwf multiplica W por f

Sintaxis: [Etiqueta] mulwf f[,a]
Operandos: $0 \leq f \leq 255$
$a \in [0,1]$
Operación: (W) x (f) →PRODH:PRODL
Flags afectados: Ninguno

Código de OP :

0000	001a	ffff	ffff

Descripción: Realiza la multiplicación sin acarreo del contenido del registro W con los 8 bits del registro "f". El resultado de 16-bits se guarda en la pareja de registros PRODH:PRODL .PRODH guarda el byte alto. W permanece inalterado. Ninguno de los flags del registro de STATUS se ve afectado.
Nota: No es posible que se desborde el resultado de la operación. Un resultado cero es posible pero no es detectado. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el banco de registro seleccionado.
Palabras: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	Escribe en los registros PRODH:PRODL

Ejemplo: mulwf Reg, 1 ; (W) x Reg (PRODH:PRODL
Antes Instrucción: (W) = 0xC4 Reg = 0xb5
PRODH = ¿? Y PRODL:¿?.
Después Instrucción: (W) = 0xC4 Reg = 0xB5
PRODH = 0x8A PRODL= 0x94

negf — f negativo

Sintaxis: [*Etiqueta*] negf f[,a]
Operandos: $0 \le f \le 255$
$A \in [0,1]$
Operación: $(\overline{f}) + 1 \rightarrow f$
Flags afectados: N,=V,C,DC,Z
Código de OP :

0110	110a	ffff	ffff

Descripción: Realiza el complemento a dos del registro "f" y deja el resultado en el registro "f". Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro "f"	Procesa datos	Escribe en el registro "f"

Ejemplo: negf Reg, 1 ; $(\overline{Reg}) + 1 \rightarrow f$
Antes Instrucción: Reg = 0011 1010 [0x3A]
Después Instrucción: Reg = 1100 0110 [0xC6]

nop — No Operación

Sintaxis: [*Etiqueta*] nop
Operandos: Ninguno
Operación: No operar
Flags afectados: Ninguno
Código de OP :

0000	0000	0000	0000
1111	xxxx	xxxx	xxxx

Descripción: No realiza operación alguna. Consume un ciclo de instrucción sin hacer nada.
Palabras: 1
Ciclos: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	No operación	No operación	No operación

Ejemplo: nop

pop — Pop Top of Return Stack

Sintaxis: [*Etiqueta*] pop
Operandos: Ninguno
Operación: (TOS) → bit bucket
Flags afectados: Ninguno
Código de OP : 0000 0000 0000 0110
Descripción: El valor de TOS se saca de la PILA y se descarta. El TOS pasa a ser el valor guardado anteriormente en la PILA. Esta instrucción permite al usuario controlar la PILA por software.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	No operación	Valor POS TOS	No operación

Ejemplo: pop
goto Nuevo
Antes Instrucción: TOS = 0031A2h
Stack (Nivel 1) = 014332h
Después Instrucción: TOS = 014332h
Snack (1 level sown) = NEW

push — Push Top of Return Stack

Sintaxis: [*Etiqueta*] push
Operandos: Ninguno
Operación: (PC+2) → TOS
Flags afectados: Ninguno
Código de OP : 0000 0000 0000 0101
Descripción: El valor PC+2 se coloca en la parte alta de la pila del retorno. El valor de TOS anterior se empuja abajo en la pila.
Esta instrucción permite llevar a cabo una pila del software modificando TOS, al tiempo que se empuja sobre la pila del retorno.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	PUSH PC+2 parte alta de la pila de retorno	No operación	No operación

Ejemplo: push
Antes Instrucción: TOS = 00345Ah

	PC	= 000124h
Después Instrucción:	PC	= 000124h
	TOS	= 000126h
	Stack (nivel 1) = 00345Ah	

rcall — Salto relativo a Subrutina

Sintaxis: [*Etiqueta*] rcall n
Operandos: $-1024 \leq n \leq 1023$
Operación: (PC+2) → TOS
(PC) + 2 + 2n → PC
Flags afectados: Ninguno
Código de OP :

1101	1nnn	nnnn	nnnn

Descripción: Llamada a subrutina con un salto relativo máximo de 1K respecto a la dirección actual. En primer lugar, la dirección del retorno (PC+2) se empuja hacia la pila. Entonces, se suma el complemento a 2 del número "2n" al PC.
El PC habrá incrementado para ejecutar la próxima instrucción, cuya nueva dirección será PC+2+2n.
Esta instrucción es una instrucción del dos-ciclo.
Palabras: 1
Ciclos: 2
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "n" Guarda PC en la Pila	Proceso de datos	Escribe en el PC
No operación	No operación	No operación	No operación

Ejemplo:	Aqui	rcall Salto
Antes Instrucción:	PC	= dirección Aqui
Después Instrucción:	PC	= dirección (Salto)
	TOS	= dirección (Aquí + 2)

reset — Reset

Sintaxis: [*Etiqueta*] reset
Operandos: ninguno
Operación: Restablece todos los registros y flags que son afectador por un MCLR Reset.
Flags afectados: Todos
Código de OP :

0000	0000	1111	1111

Descripción: Esta instrucción realiza un reset por software
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Inicia el reset	No operación	No operación

Ejemplo:	reset	
Después Instrucción:	Registros	= Valores de Reset
	Flags	= Valores de Reset

retfie — Retorno de Interrupción

Sintaxis: [*Etiqueta*] retfie [s]
Operandos: s ∈ [0,1]
Operación: TOS → (PC)
1 → GIE/GIEH o PEIE/GIEL
si s = 1
(WS) → W
(STATUSS) → STATUS
(BSRS) → BSR
PCLATU, PCLATH está inalterado
Flags afectados: GIE/GIEH, PEIE/GIEL.
Código de OP :

0000	0000	0001	000s

Descripción: Carga el PC con el valor que se encuentra en la parte alta 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. Tarda dos ciclos máquina.
Si 's ' = 1, el contenido de los registros de guarda WS, STATUSS y BSRS se cargan sus correspondientes registros W, ESTADO y BSR. Si 's ' = 0, no se actualizan estos registros (es el valor por defecto).
Palabras: 1
Ciclos: 2
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	No operación	No operación	Pop PC al snack Pone a "1" GIEH o GIHL
No operación	No operación	No operación	No operación

Ejemplo: retfie 1 ; Retorna de la interrupción.
Después Instrucción: PC = TOS
W = WS
BSR = BSRS
STATUS = STATUSS
GIE/GIEH = 1

retlw — Retorno con un literal en W

Sintaxis: [*Etiqueta*] retlw k
Operandos: 0 ≤ k ≤ 255
Operación: k → (W)
TOS → (PC)
PCLATU, PCLATH sin cambios
Flags afectados: Ninguno
Código de OP :

0000	1100	kkkk	kkkk

Descripción: Carga el registro 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. Tarda dos ciclos máquina. El match de la parte alta del PC (PCLATH) permanece inalterado.
Palabras: 1
Ciclos: 2

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal "k"	No operación	Pop PC al snack Escribe en W
No operación	No operación	No operación	No operación

```
Ejemplo:            call    Tabla   ; W contiene el valor de offset de tabla
                    .... ....
                Tabla               ; Comienza la tabla
                    addwf   PCL     ; (PC) = (PC) + (W)
                    retlw   k0
                    retlw   k1
                    retlw   k2                  .
                    .
                    retlw   kn      ;Fin de tabla

Antes Instrucción:  (W) = 0x02
Después Instrucción: (W) = Toma el valor de k2
```

return — Retorno de subrutina

Sintaxis: [*Etiqueta*] return [s]
Operandos: s ∈ [0,1]
Operación: TOS → (PC)
Si s=1
(WS) → W
(STATUSS) → STATUS
(BSRS) → BSR
PCLATU, PCLATH permanecen inalterados
Flags afectados: Ninguno
Código de OP :

0000	0000	0001	001s

Descripción: Carga el PC con el valor que se encuentra en la parte superior de la pila, efectuando así un retorno de subrutina. Si 's ' = 1, el contenido de los registros de guarda WS, STATUSS y BSRS se cargan en su registro correspondiente, W, ESTADO y BSR. Si 's ' = 0, no se modifica ninguno de estos registros (valor por defecto).
Palabras: 1
Ciclos: 2
Ciclos de actividad Q

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación

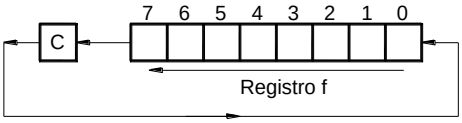
```
Ejemplo:            return      ; Retorna de la subrutina.
Antes Instrucción:  (PC) = ¿?
Después Instrucción: (PC) = (TOS)
```

rlcf — Rota a la izquierda con el Carry

Sintaxis: [*Etiqueta*] rlf f[,d[a]
Operandos: 0 ≤ f ≤ 255
d ∈ [0,1]

a ∈ [0,1]

Operación:



Flags afectados: C,N,Z

Código de OP :

0011	01da	ffff	ffff

Descripción: Rotación de un bit a la izquierda del contenido del registro 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. Si "a" = 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a" =1 se usa el BSR para indicar el banco seleccionado.

Palabras: 1

Ciclos: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe en destino

Ejemplo: rlcf Reg, 0, 0
Antes Instrucción: (Reg) = b'1110 0110', W = ¿? y C = 0.
Después Instrucción: (Reg1) = b'1110 0110', (W) = b'1100 1100' y C = 1.

rlncf — Rota a la izquierda (sin el Carry)

Sintaxis: [*Etiqueta*] rlncf f[,d[a]

Operandos: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operación:

Flags afectados: N,Z

Código de OP :

0100	01da	ffff	ffff

Descripción: Rotación de un bit a la izquierda del contenido del registro f, e introduce un "1" en el bit menos significativo. Si d=1 el resultado se almacena en f, si d=0 el resultado se almacena en W. Si "a" = 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a" =1 se usa el BSR para indicar el banco seleccionado.

Palabras: 1

Ciclos: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe en destino

Ejemplo: rlncf Reg, 1, 0
Antes Instrucción: (Reg) = b'1010 1011'
Después Instrucción: (Reg) = b' 01010111'

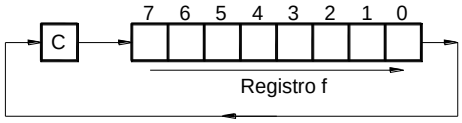
rrcf — Rota a la derecha con el Carry

Sintaxis: [*Etiqueta*] rrcf f[,d[a]

Operandos: 0 ≤ f ≤ 255

d ∈ [0,1]
a ∈ [0,1]

Operación:



Flags afectados: C,N,Z

Código de OP :

0011	00da	ffff	ffff

Descripción: Rotación de un bit a la derecha del contenido del registro 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. Si "a" = 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a" =1 se usa el BSR para indicar el banco seleccionado.

Palabras: 1

Ciclos: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe en destino

Ejemplo: rrcf Reg, 0, 0
Antes Instrucción: (Reg) = b'1110 0110', W = ¿? y C = 0.
Después Instrucción: (Reg1) = b'1110 0110', (W) = b'0111 0011' y C = 0.

rrncf — Rota a la derecha (sin el Carry)

Sintaxis: [*Etiqueta*] rrncf f[,d[a]
Operandos: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operación:

Flags afectados: N,Z

Código de OP :

0100	00da	ffff	ffff

Descripción: Rotación de un bit a la derecha del contenido del registro f, e introduce un "1" en el bit más significativo. Si d=1 el resultado se almacena en f, si d=0 el resultado se almacena en W. Si "a" = 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a" =1 se usa el BSR para indicar el banco seleccionado.

Palabras: 1

Ciclos: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe en destino

Ejemplo 1: rrncf Reg, 1, 0
Antes Instrucción: (Reg) = b'1101 0111'
Después Instrucción: (Reg) = b' 1110 1011'

Ejemplo 2: rrncf Reg, 0, 0
Antes Instrucción: (Reg) = b'1101 0111' W= ¿?
Después Instrucción: (Reg) = b'1101 0111' W = 1110 1011

setf — Puesta a Set del registro f

Sintaxis: [*Etiqueta*] setf f [,a]
Operandos: $0 \le f \le 255$
$a \in [0,1]$
Operación: ffh → f
Flags afectados: Ninguno
Código de OP :

0110	100a	ffff	ffff

Descripción: Carga el registro "f" don el valor ffh. Si "a" = 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a" =1 se usa el BSR para indicar el banco seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe en el registro f

Ejemplo : setf Reg, 1
Antes Instrucción: (Reg) = b'1101 0111'
Después Instrucción: (Reg) = b' 1111 1111'

sleep — Pasa a "Standby" o modo de bajo consumo

Sintaxis: [*Etiqueta*] sleep
Operandos: Ninguno
Operación: 00h → WDT
0 →WDT postscaler
1 → $\overline{TO}$
0 → $\overline{PD}$
Flags afectados: $\overline{TO}$, $\overline{PD}$
Código de OP :

0000	0000	0000	0011

Descripción: Pone al circuito en modo *Sleep* (bajo consumo) con parada del oscilador y Timer 0. Se puede salir de este estado por:

- Activación del pin $\overline{MCLR}$ para provocar un reset
- Desbordamiento del Watchdog si quedó operativo en el modo reposo.
- Generación de una interrupción que no sea TMR0 ya que ésta se desactiva con la instrucción *sleep*.

Ejemplo: sleep ; Pasa a "standby" o modo de bajo consumo.
Antes de la instrucción: $\overline{TO}$ = ¿?
$\overline{PD}$ = ¿?
Después de la instrucción: $\overline{TO}$ = 1 (*)
$\overline{PD}$ = 0
(*) Este bit se pone a "0" si el WDT se desborda

subfw — resta f a W con el debe

Sintaxis: [*Etiqueta*] subfwb f[,d[a]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (W) - (f) + ($\overline{C}$) → (destino)
Flags afectados: N,OV,C, DC, Z
Código de OP :

0101	01da	ffff	ffff

Descripción: Resta registro "f" y el flan de carry (borrow) al contenido del registro W (por el método de complemento a dos) el resultado se guarda en W si d = 0, y en el registro f si d = 1.Si el resultado es positivo el flag C=1. Si el resultado es cero el flan Z=1 y C=1. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe en destino

Ejemplo 1: *subfwb Registro, 1, 0* ; (W) - (Registro) - $\overline{C}$ → f
Antes Instrucción: (W) = 0x02, (Registro) = 0x03, y C = 1.
Después Instrucción: (W) = 0x02, (Registro) = 0xff, y C = 0, Z = 0, N = 1

Ejemplo 2: *subfwb Registro, 0, 0* ; (W) - (Registro) - $\overline{C}$ → (W)
Antes Instrucción: (W) = 0x05, (Registro) = 0x02, y C = 1.
Después Instrucción: (W) = 0x03, (Registro) = 0x02, y C = 1, Z = 0, N = 0

Ejemplo 3: *subfwb Registro, 1, 0* ; (W) - (Registro) - $\overline{C}$ → f
Antes Instrucción: (W) = 0x02, (Registro) = 0x01, y C = 0.
Después Instrucción: (W) = 0x02, (Registro) = 0x00, y C = 1, Z = 1, N = 0

sublw — Resta el literal k a W

Sintaxis: [*Etiqueta*] sublw k
Operandos: $0 \le k \le 255$
Operación: k - (W) → (W)
Flags afectados: N,OV,C, DC, Z
Código de OP :

0000	1000	kkkk	kkkk

Descripción: Resta por el método de complemento a dos al literal k el contenido del registro W. Almacena el resultado en W.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal k	Procesa datos	Escribe en el W

Ejemplo 1: sublw 0x02 ; 02h – (W) → (W)
Antes Instrucción: (W) = 0x01, C = ¿?
Después Instrucción: (W) = 0x01, C = 1 (el resultado es positivo) , Z = 0 y N = 0.

Ejemplo 2: sublw 0x02 ; 03h – (W) → (W)
Antes Instrucción: (W) = 0x02, C = ¿?
Después Instrucción: (W) = 0x00, C = 1, Z = 1 (el resultado es cero) y N = 0.

Ejemplo 3: sublw 0x02 ; 03h – (W) → (W)
Antes Instrucción: (W) = 0x03, (+3 en decimal), C = ¿? y Z = ¿?.
Después Instrucción: (W) = 0xFF,(-1 en decimal)
C = 0 (resultado negativo), Z = 0 y N = 0.

subwf — Resta el registro f a W

Sintaxis: [*Etiqueta*] subwf f[,d[,a]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (f) - (W) → (destino)
Flags afectados: N, OV, C, DC, Z
Código de OP :

0101	11da	ffff	ffff

Descripción: Resta por el método de complemento a dos el contenido del registro f menos el contenido del registro W. Almacena el resultado en W si d=0 y en f si d=1. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe en destino

Ejemplo 1: subwf Reg1, 1, 0 ; (Reg1) – (W) → (Reg1)
Antes Instrucción: (Reg1) = 0x03, (W) = 0x02, C = ¿?) y Z = ¿?.
Después Instrucción: (Reg1) = 0x01, (W) = 0x02
C = 1 (positivo) y Z = 0.

Ejemplo 2: subwf Reg1, 0, 0 ; (Reg1) – (W) → (W)
Antes Instrucción: (Reg1) = 0x02, (W) = 0x02, C = ¿? y Z = ¿?.
Después Instrucción: (Reg1) = 0x02, (W) = 0x00
C = 1, Z = 1 (resultado cero) N =0.

Ejemplo 3: subwf Reg1, 1, 0 ; (Reg1) – (W) → (Reg1)
Antes Instrucción: (Reg1) = 0x01, (+1 decimal), (W) = 2, C = ¿?) y Z = ¿?.
Después Instrucción: (Reg1) = 0xFF, (-1 decimal), (W) = 2
C = 0 (negativo), Z = 0 y N = 0.

subwfb — resta al registro f W y el bit de Borrow

Sintaxis: [*Etiqueta*] subwfb f[,d[a]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operación: (f) - (W) + ($\overline{C}$) → (destino)
Flags afectados: N,OV,C, DC, Z

Código de OP :	0101	10da	ffff	ffff

Descripción: Resta al contenido del registro f (en complemento a 2) el contenido del registro W y del flag de Carry, y almacena el resultado en W si d = 0, y en el registro f si d = 1. Si el resultado es positivo el flag C se pone a "1". Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.

Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe W en destino

Ejemplo 1:	*subwfb Registro, 1, 0*	; (Registro) + (W) → (f)
Antes Instrucción:	(Registro) = 0x19	(0001 1001)
	(W) = 0x0d	(0000 1101)
	C = 1.	
Después Instrucción:	(Registro) = 0x0C	(0000 1011)
	(W) = 0x0d	(0000 1101)
	C = 1, Z = 0 y N = 0	
Ejemplo 2:	*subwfb Registro, 0, 0*	; (Registro) + (W) → (W)
Antes Instrucción:	(Registro) = 0x1B	(0001 1011)
	(W) = 0x1A	(0001 1010)
	C = 0.	
Después Instrucción:	(Registro) = 0x1B	(0001 1011)
	(W) = 0x00	(0000 0000)
	C = 1, Z = 1 (resultado = 0) y N = 0	
Ejemplo 3:	*subwfb Registro, 1, 0*	; (Registro) + (W) → (f)
Antes Instrucción:	(Registro) = 0x03	(0000 0011)
	(W) = 0x0E	(0000 1101)
	C = 1.	
Después Instrucción:	(Registro) = 0xF5	(1111 0101) (complemento a 2)
	(W) = 0x0E	(0000 1101)
	C = 0, Z = 0 y N = 0	

swapf — Intercambia nibbles de f

Sintaxis: [*Etiqueta*] swapf f [,d[,a]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (f<3:0>) → (d <7:4>)
(f<7:4>) → (d <3:0>)
Flags afectados: Ninguno

Código de OP :	0011	10da	ffff	ffff

Descripción: Los cuatro bits de más peso del registro f se intercambian con los 4 bits de menos peso del mismo registro. Si d=0 el resultado se almacena en W, si d=1 el resultado se almacena en f.

Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.

Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa datos	Escribe W en destino

Ejemplo: swapf Reg, 1, 0
Antes Instrucción: (Reg) = 0x53
Después Instrucción: (Reg) = 0x35

tblrd — lee tabla

Sintaxis: [*Etiqueta*] tblrd (*; *+; *-; +*)
Operandos: Ninguno
Operación: Si tblrd *,
(memoria de Programa (TBLPTR)) → TABLAT;
tblptr – No cambia;
Si tblrd *+;
(memoria de Programa (TBLPTR)) → TABLAT;
(TBLPTR) + 1 → TBLPTR;
Si tblrd *-,
(memoria de Programa (TBLPTR)) → TABLAT;
(TBLPTR) - 1 → TBLPTR;
Si tblrd +*,
(TBLPTR) + 1) → TABPTR;
(memoria de Programa (TBLPTR)) → TABLAT;
Flags afectados: Ninguno
Código de OP :

0000	0000	0000	10 nn nn=0 * = 1 *+ =2 * - =3 +*

Descripción: Esta instrucción se usa para leer el contenido de la Memoria de Programa (P.M). Para direccionar la memoria de programa, se utiliza un puntero de tabla llamado (TBLPTR).
El TBLPTR (un puntero de 21-bits) apunta a cada byte en la memoria del programa. TBLPTR tiene un rango de 2 Mbyte de direccionamiento.
TBLPTR[0] = 0: byte menos significativo de la palabra de memoria.
TBLPTR[0] = 1: Byte más significativo de la palabra de memoria
La instrucción de TBLRD puede modificar el valor de TBLPTR como sigue:

- ningún cambio
- Post-incremento
- Post-decremento
- Pre-incremento de

Palabras: 1
Ciclos: 2
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	No operación	No operación	No opercación

No operación	No operación (lectura memoria de programa)	No operación	No operación (Escribe TABLAT)

```
Ejemplo1:           tblrd   *+      ;
Antes Instrucción:  TABLAT          = 0x55
                    TBLPTR          = 0x00A356
                    Memoria (0x00A356)  = 0x34
Después Instrucción: TABLAT         = 0x34
                    TBLPTR          = 0x00A356

Ejemplo2:           tblrd   +*      ;
Antes Instrucción:  TABLAT          = 0xAA
                    TBLPTR          = 0x01A357
                    Memoria (0x01A357)  = 0x12
                    Memoria (0x01A358)  = 0x34
Después Instrucción: TABLAT         = 0x34
                    TBLPTR          = 0x01A358
```

tblwt — escribe tabla

Sintaxis: [*Etiqueta*] tblwt (*; *+; *-; +*)
Operandos: Ninguno
Operación: Si tblwt *,
(TABALT) → Registro base
TBLPTR no cambia
Si tblwt*+
(TABALT) → Registro base
(TBLPTR) + 1 → TBLPTR;
Si tblwt*-,
(TBLPTR)-1 → TBLPTR;
Si tblwt+*,
(TBLPTR) + 1 → TBLPTR ;
(TABLAT) → Registro base

Flags afectados: Ninguno

Código de OP :

0000	0000	0000	11 nn nn=0 * = 1 *+ =2 * - =3 +*

Descripción: Esta instrucción usa los 3 bits menos significativos del TBLPTR para determinar cual de los 8 registros respecto al registro base TABLAT se escribirá. Los 8 registros respecto al registro base se usan para modificar la memoria de programa.
El TBLPTR (un puntero de 21-bits) apunta a cada byte en la memoria del programa. TBLPTR tiene un rango de direccionamiento de 2 El LSb del TBLPTR selecciona la posición para acceder al byte de la memoria de programa.
TBLPTR[0] = 0: byte menos significativo de la palabra de memoria.
TBLPTR[0] = 1: Byte más significativo de la palabra de memoria
La instrucción de TBLRD puede modificar el valor de TBLPTR como sigue:

- ningún cambio
- Post-incremento
- Post-decremento
- Pre-incremento de

Palabras: 1
Ciclos: 2

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	No operación	No operación	No opercación
No operación	No operación (leeTABLAT)	No operación	No operación (Escribe en Registro base + Memoria)

```
Ejemplo1:           tblwt   *+      ;
Antes Instrucción:  TABLAT                  = 0x55
                    TBLPTR                  = 0x00A356
                    Registro Base (0x00A356)= 0xFF
Después Instrucción:TABLAT                  = 0x55
                    TBLPTR                  = 0x00A357
                    Registro Base (0x00A356)= 0x55

Ejemplo2:           tblwt   +*      ;
Antes Instrucción:  TABLAT                  = 0x34
                    TBLPTR                  = 0x01389A
                    Registro Base (0x01389A)= 0xFF
                    Registro Base (0x01389B)= 0xFF
Después Instrucción:TABLAT                  = 0x34
                    TBLPTR                  = 0x01389B
                    Registro Base (0x01389A)= 0xFF
                    Registro Base (0x01389B)= 0x34
```

tstfsz — Test de f y salta si es cero

Sintaxis: [*Etiqueta*] tstfsz f[,a]
Operandos: $0 \le f \le 255$
$a \in [0,1]$
Operación: Salta si f = 0
Flags afectados: Ninguno
Código de OP :

0110	011a	ffff	ffff

Descripción: Si el registro f es cero, la instrucción que sigue a esta se ignora y se trata como un *"nop"*. En este caso y solo en este caso, la instrucción *btfsc* precisa dos ciclos para ejecutarse. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.
Palabras: 1
Ciclos: 1(2)
Nota: 3 ciclos si el salto y lleva a cavo una instrucción de la segunda palabra.
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el Registro "f"	Procesa datos	No operación

Si salta

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación

Si salta y lleva a cavo una instrucción de la segunda palabra

Q1	Q2	Q3	Q4
No operación	No operación	No operación	No operación
No operación	No operación	No operación	No operación

Ejemplo:

```
        Aqui    tstfsz  CNT, 1          ; Si el registro "CNT" es cero salta.
        Falso   goto    ProcesoX        ; No ha sido uno.
        Verdad          ...             ; Ha sido cero.
                                        ...
```

Antes Instrucción: (PC) = Dirección de "Aquí".
Después Instrucción: Si registro CNT = 0, (PC) = Dirección de "Verdad".
Si registro CNT ≠ 0, (PC) = Dirección de "Falso".

xorlw — OR-Exclusiva del literal k con W

Sintaxis: [*Etiqueta*] xorlw k
Operandos: $0 \le k \le 255$
Operación: (W) XOR k → (W)
Flags afectados: N, Z
Código de OP.:

0000	1010	kkkk	kkkk

Descripción: Realiza la función OR-Exclusiva entre el contenido del registro W y la constante k de 8 bits. El resultado se almacena en W
Palabras: 1
Ciclos: 1
Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el literal k	Procesa Datos	Escribe en W

```
Ejemplo:            xorlw   b'10101111'     ; (W) XOR b'10101111' → (W)
Antes Instrucción:  (W) =   b'10110101' y   Z = ¿?.
Después Instrucción:(W) =   b'00011010' y   Z = 0
```

xorwf — OR-Exclusiva de W con el registro f

Sintaxis: [*Etiqueta*] xorwf f [,d[,a]
Operandos: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$
Operación: (W) XOR (f) → (destino)
Flags afectados: N,Z

Código de OP :

0001	10da	ffff	ffff

Descripción: Realiza la función OR-Exclusiva entre el contenido del registro W y el contenido del registro f. Almacena el resultado en f si d=1 y en W si f=0. Si "a" es 0 el BSR es ignorado y se utiliza el modo ACCESS. Si "a"=1 se usa el BSR para indicar el bando de registro seleccionado.

Palabras: 1

Ciclos: 1

Ciclos de actividad Q

Q1	Q2	Q3	Q4
Decodificación	Lee el registro f	Procesa Datos	Escribe en destino

Ejemplo1:	xorwf Reg, 1, 0	; (W) XOR (Reg) → (Reg)	
Antes Instrucción:	(Reg) = b'10101111',	(W) = b'10110101'	y Z = ¿?.
Después Instrucción:	(Reg) = b'00011010',	(W) = b'10110101'	y Z = 0.