El repertorio de instrucciones RISC

Algunas normas para trabajar en Assembler

El uso de mayúsculas y minúsculas en el código de programación obedece a una serie de reglas o normas de estilo, comunes entre los programadores en Assembler, que aunque no son obligatorias, facilitan la lectura del código fuente y proporciona un "ambiente estandarizado".

Un resumen de estas reglas (como "normas de convivencia" entre programadores) empleadas son las siguientes:

- ✓ Para utilizar comentarios se antepone el símbolo ";"
- ✓ Primero se definen las directivas, luego las etiquetas y por último el programa principal
- ✓ Directivas del compilador en mayúsculas
- ✓ Nombres de variables en mayúsculas
- ✓ Nemónicos en minúsculas y
- ✓ Programa bien tabulado

Las instrucciones de los PICs de la gama media, tienen 14 bits de longitud; dicho formato se divide en diferentes campos de bits, cada uno de los cuales referencia a operandos o elementos que maneja la instrucción en la operación que realiza en el procesador. Se describen dichos campos:

- ✓ *0xhh*: se hace referencia a los números hexadecimales de dos dígitos HH
- ✓ *f*: representa la dirección de la memoria RAM de datos del registro fuente; tiene un tamaño de 7 bits, con un direccionamiento de 128 posiciones comprendidas entre la dirección *0x00* y la *0x7F*

- ✓ *b*: número de bit, codificado en 3 bits; el bit 0 siempre es el de menor peso
- ✓ *d*: es un bit que conforma el campo del formato de una instrucción que indica el registro destino. Si d=0 es W, y si d=1 es f
- ✓ *k*: campo que contiene un valor inmediato, que puede ser un operando (8 bits) o una dirección para el PC (11 bits)
- ✓ *x*: valor indeterminado de un bit. Puede ser 1 o 0, es indiferente
- ✓ *( )*: contenido
- ✓ *< >*: campo de un bit de un registro. Por ejemplo: STATUS <5>
- ✓ *∈*: en el conjunto de ejemplo: d ∈ [0,1]
- ✓ *[ ]*: opciones

· Manejo de la base del sistema de numérico: su representación

De acuerdo con el sistema de numeración con que se trabaje, se tendrá que indicar en el programa del lenguaje Assembler a que base numérica se está haciendo referencia.

Se representará mediante un simple ejemplo de aplicación, los diferentes casos que se pueden presentar:

a) base decimal: *d '21'*
b) base hexadecimal: *0xa1*
h 'a1'
a1
a1 h

c) base binaria: *b '1101'*

Nosotros trabajaremos generalmente en la edición del programa con base numérica hexadecimal y la binaria, haciendo las aclaraciones de la base en cada caso y cuando sea necesario.

Directivas e instrucciones del Assembler

Las sentencias que componen el código fuente del lenguaje de programación en Assembler tradicional, son de dos tipos:

a) *Instrucciones*: son representaciones simbólicas (nemotécnicas) del juego de instrucciones del microprocesador y

b) *Directivas*: indican al ensamblador qué hacer con las instrucciones y datos.
Las directivas del Assembler ancestral, es decir, aquel proveniente de los antiguos microprocesadores 80X86 todavía son utilizadas, y de hecho nosotros también las emplearemos y son las siguientes:

Directiva	Formato	Descripción
ORG	ORG expresión	ORiGin, pone al PC con el valor expresión. El ensamblador almacenará el código a partir de esa dirección.
RADIX	RADIX expresión	Cambia la base de numeración; por defecto la base es 10.
EQU	nombre EQU expresión	EQUivalence, asigna un nombre simbólico al valor de una expresión.
LIST		Genera el listado de salida del ensamblador.
END		Indica el final del programa fuente.

Directivas generales del lenguaje de programación Assembler

Veamos un ejemplo de aplicación donde se utilizan estas directivas, para una ayuda visual más efectiva se expresará el programa en Assembler con fuente de texto en *cursiva*:

```
;DIRECTIVAS DEL PROGRAMA
        LIST        P=16F84         ;Se emplea el PIC16F84
        RADIX       HEX             ;Sistema de numeracion hexadecimal

;ETIQUETAS DEL PROGRAMA
        W           EQU 0x00        ;Registro de destino W: d=0
        F           EQU 0x01        ;Registro de destino F: d=1
        STATUS      EQU 0x03        ;El registro STATUS ocupa la direccion 3 de los dos bancos
        PUERTOA     EQU 0x05        ;El PUERTO A ocupa la direccion 5 del banco 0
                                    ;y su registro de configuracion la direccion 5 del banco 1
        PUERTOB     EQU 0x06        ;El PUERTO B ocupa la direccion 6 del banco 0
```

```
                                    ;y su registro de configuracion la direccion 6 del banco 1
AUX             EQU 0x0C            ;Registro auxiliar

;PROGRAMA PRINCIPAL
                ORG 0               ;El programa comienza en la direccion 0 y
                goto    INICIO              ;salta a la direccion 5 para sobrepasar el
                ORG 5               ;vector de interrupcion (dirección 4)

INICIO          ...     ...                 ; ...
                ...     ...                 ; ...
```

Como convención adopta|emos como sistema numérico de trabajo el hexadecimal, por lo tanto, siempre tendremos que escribir la directiva *RADIX HEX* entre las directivas del programa; de ahí en adelante cualquier número que no se especifique el sistema numérico se supondrá que está en hexadecimal. En este curso se supone que existen los conocimientos previos necesarios para el manejo de las distintas bases numéricas (sobre todo la hexadecimal, decimal y binaria).

Hay que tener en cuenta que los PICs 16x84 admiten interrupciones, cuando se produce alguna, se guarda en la Pila el contenido actual del PC (dirección de retorno) y el PC se carga con la dirección 4 (vector de interrupción). considerando que es muy habitual el uso de interrupciones, conviene que el programa comience en la dirección 5, para iniciar el programa en la dirección del Reset 0 y al mismo tiempo situar la primera instrucción en la dirección 5, se coloca en la dirección 0 una instrucción de salto a la 5 (esto figura en las tres primeras líneas del programa principal).

Las 35 instrucciones del PIC 16X84

A continuación se describirá en forma detallada el juego de instrucciones en Assembler específico que tiene el PIC 16X84, con sus respectivas características y ejemplos de aplicación.
Para una mayor comodidad se dividió las instrucciones en grupos según la clase de funciones que realizan:

- ✓ Instrucciones que manejan registros,
- ✓ Instrucciones que manejan bits,
- ✓ Instrucciones de salto condicional,
- ✓ Instrucciones que manejan operandos inmediatos e
- ✓ Instrucciones especiales y de control

3.4.1.- Instrucciones que manejan registros

Responden a la sintaxis *nemónico f,d*; siendo f y d los dos operandos fuente y destino respectivamente, que se hallan implementados por registros de 8 bits de la memoria de datos.

El registro f viene referenciado por la dirección de 7 bits que ocupa, mientras que el destino sólo por 1 bit, que si vale 0 es el W y si vale 1 es el fuente.

Byte-oriented file register operations

d = 0 for destination W
d = 1 for destination f
f = 7-bit file register address

Nemónico Descripción Ciclos Código de op. Estatus Notas

BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d	Add W and f	1	00	0111	dfff	ffff	C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00	0101	dfff	ffff	Z	1,2
CLRF	f	Clear f	1	00	0001	lfff	ffff	Z	2
CLRW	-	Clear W	1	00	0001	0xxx	xxxx	Z	
COMF	f, d	Complement f	1	00	1001	dfff	ffff	Z	1,2
DECF	f, d	Decrement f	1	00	0011	dfff	ffff	Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1 (2)	00	1011	dfff	ffff		1,2,3
INCF	f, d	Increment f	1	00	1010	dfff	ffff	Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1 (2)	00	1111	dfff	ffff		1,2,3
IORWF	f, d	Inclusive OR W with f	1	00	0100	dfff	ffff	Z	1,2
MOVF	f, d	Move f	1	00	1000	dfff	ffff	Z	1,2
MOVWF	f	Move W to f	1	00	0000	lfff	ffff		
NOP	-	No Operation	1	00	0000	0xx0	0000		
RLF	f, d	Rotate Left f through Carry	1	00	1101	dfff	ffff	C	1,2
RRF	f, d	Rotate Right f through Carry	1	00	1100	dfff	ffff	C	1,2
SUBWF	f, d	Subtract W from f	1	00	0010	dfff	ffff	C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00	1110	dfff	ffff		1,2
XORWF	f, d	Exclusive OR W with f	1	00	0110	dfff	ffff	Z	1,2

.- Ejemplos de aplicación 1

Si se supone que inicialmente valen cero los registros W y los que ocupan las direcciones 0x01 y 0x02 de la memoria de datos ¿qué valor contendrán después de ejecutar el siguiente programa?

```
incf 0x01,0
incf 0x01,1
comf 0x02,1
iorwf 0x02,0
xorwf 0x01,1
```

Solución

incf 0x01,0

(W)= b '0000 0000' + b '0000 0001'

(W) = b '0000 0001'

incf 0x01,1

(0x01)= b '0000 0000' + b '0000 0001'

(0x01) = b '0000 0001'

comf 0x02,1

(0x02)= 1111 1111

iorwf 0x02,0

(W) = 0000 0001 or 1111 1111

(W)= b '1111 1111'

xorwf 0x01,1

(0x01) = 1111 1111 $\oplus$ 0000 0001

(0x01) = b '1111 1110'

- Ejemplo de aplicación 2

Existe una instrucción con la que se rota a la derecha a través del flag de acarreo C, el registro que ocupa la dirección 0x04 de la memoria de datos y deposita en W. Determinar:

a) El nemónico

b) si inicialmente C=1 y (0x04)=0x00 ¿qué valor se cargará en W?

Solución

a) *rrf 0x04,0*

b) W = b '1000 0000'

- Ejemplo de aplicación 3

Si el registro STATUS contiene el valor 0xF5 y se ejecuta la instrucción *swap STATUS,0* ¿qué valor contendrán STATUS y W?

Solución

La instrucción swap intercambia los nibbles (4 bits o medio byte) del contenido de cada registro, entonces:

swap STATUS,0

(STATUS) = b '1111 0101'

0xF5 h (no cambia)

(W) = b '0101 1111'

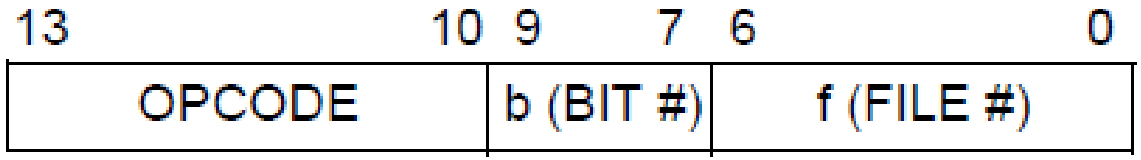
0x5F h

- Instrucciones que manejan bits

Sólo hay dos instrucciones en este grupo, pero son muy flexibles; una de ellas pone a 1 (BSF) cualquier bit de un registro, mientras que la otra lo pone a 0 (BCF).

En este tipo de instrucciones se representa con 3 bits la posición que ocupa en el registro el bit b. El registro se especifica mediante su dirección de 7 bits.

Bit-oriented file register operations



b = 3-bit bit address
f = 7-bit file register address

BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff		1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff		1,2

Ejemplo de aplicación

Si inicialmente, el registro W=0xFF y el registro OPTION=0x00 ¿con qué valor quedarán cargados tras ejecutar la siguiente parte del programa?

```
bsf OPTION,2
comf OPTION,0
swap OPTION,1
```

Solución

bsf OPTION,2

(OPTION) = b '0000 0100'

comf OPTION,0

(W)= b '1111 1011'

swap OPTION,1

(OPTION)=b '0100 0000'

Entonces, los valores finales en los registros son:

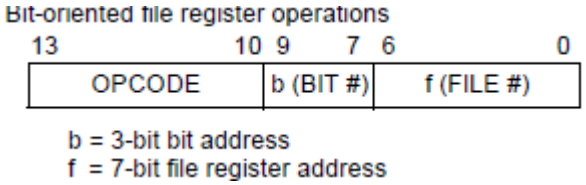
(W)= b '1111 1011'

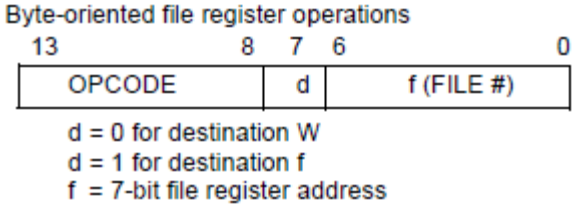
(OPTION)= b '0100 0000'

Instrucciones de salto condicional

Existen cuatro instrucciones de salto condicional, dos de ellas controlan un bit de un registro y según valga 1 (BTFSS) o 0 (BTFSC) saltan o no. Las otras dos instrucciones incrementan (INCFSZ) o decrementan (DECFSZ) un registro y la posibilidad del salto se efectúa si con esa operación el valor del registro ha llegado a cero.

Cuando estas instrucciones no saltan porque no se cumple la condición, tardan 1 ciclo de instrucción en ejecutarse, si salta demoran el doble.





Instrucción	Parámetros	Descripción	Ciclos	Operación	Registros afectados
BTFSC	f,b	Bit Test F, Skip if Clear	1 (2)	Testea bit (b) en file f, saltar si es 0	Ninguno
BTFSS	f,b	Bit Test F, Skip if Set	1 (2)	Testea bit (b) en file f, saltar si es 1	Ninguno
DECFSZ	f,d	DECrement F, Skip if Zero	1(2)	f-1 ⟶ d, saltar si es 0	Ninguno
INCFSZ	f,d	INCrement F, Skip if Zero	1(2)	f+1 ⟶ d, saltar si es 0	Ninguno

Tabla de instrucciones que manejan salto o bifurcaciones

(2) si esta instrucción es ejecutada desde el registro TMR0 (donde se aplica d=1) el prescaler será puesto a 0 si se asignó el módulo Timer0.

.- Ejemplo de aplicación 1

Escribir las instrucciones necesarias para averiguar el valor del bit 4 del registro STATUS. Si vale 1 se pone a cero el registro W y si dicho bit vale 0, se pone 1 ese bit.

Solución

```
btfss STATUS,4
bsf STATUS,4
clrw
```

- Ejemplo de aplicación 2

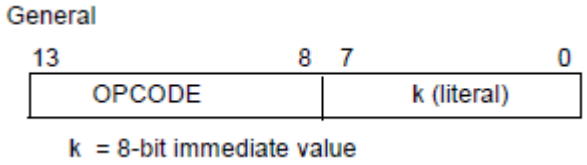
Expresar en Assembler la siguiente operación: decrementar el contenido del registro FSR hasta que valga cero y entonces borrar W.

Solución

```
bucle  decfsz FSR,1
       goto bucle
       clrw
```

- Instrucciones que manejan operandos inmediatos

Estas instrucciones realizan una operación con un valor inmediato o literal de 8 bits que se proporciona dentro del formato de la instrucción, el cual sólo tiene dos campos: el del Código OP (6 bits) y el del operando inmediato (8 bits).



Instrucción	Parámetros	Descripción	Ciclos	Operación	Registros afectados
ADDLW	k	ADD Literal and W	1	k+W → W	C, DC, Z
ANDLW	k	AND Literal with W	1	k and W → W	Z
IORLW	k	Inclusive OR Literal with W	1	k or W → W	Z
MOVLW	k	MOVe Literal to W	1	k → W	Ninguno
SUBLW	k	SUBtract W from Literal	1	k-W → W	C, DC, Z
XORLW	k	Exclusive OR Literal with W	1	k ⊕ W → W	Z

Tabla de instrucciones que manejan operandos inmediatos

.- Ejemplo de aplicación 1

Indicar el valor del registro W después de ejecutar el siguiente programa:

```
clrw
movwf OPTION
bsf OPTION,5
comf OPTION,1
btfsc OPTION,1
swap OPTION,0
```

Solución

clrw

(W)= b '0000 0000'

movwf OPTION

(OPTION) = (W) = b '0000 0000'

bsf OPTION,5

(OPTION) = b '0010 0000'

comf OPTION,1

(OPTION)= b '1101 1111'

btfsc OPTION,1

swap OPTION,0

(W)= b '1111 1101'

Por lo tanto, el registro W queda con el siguiente valor:

(W) = b'1111 1101'

- Ejemplo de aplicación 2

Averiguar el contenido de W después de ejecutar el siguiente programa:

```
clrw
addlw 0x55 h
andlw 0xFF
iorlw 0x0F
xorlw 0xAF
```

Solución

clrw

(W)= b'0000 0000'

addlw 0x55 h

(W)= 0000 0000 + 0101 0101

(W) = b '0101 0101'

andlw 0xFF

b '0101 0101' and b '1111 1111' = b'0101 0101'

(W)= b '0101 0101'

iorlw 0x0F

b '0101 0101' or b'0000 1111'= b'0101 1111'

xorlw 0xAF

b '0101 1111' ⊕ b'1010 1111' = b'1111 0000'

El valor del contenido de W es:

(W) = b '1111 0000'

0xF0 h

Instrucciones especiales y de control

En este grupo de instrucciones se incluyen aquellas que rompen la secuencia normal del programa porque alteran el contenido del PC y también las instrucciones especiales.

Las instrucciones de salto incondicional GOTO carga en el PC la dirección de la nueva instrucción.

La instrucción CALL de llamada a subrutina, antes de cargar en el PC con la dirección de la instrucción a saltar, salva la dirección de partida guardando en la cima de la Pila el valor actual del PC, de esta manera, al retornar de la subrutina se saca de la Pila el valor actual del PC, para poder regresar a la dirección del programa principal de donde se produjo el salto.

Para realizar un retorno de una subrutina se pueden emplear dos instrucciones, la más habitual es RETURN, que se limita a extraer de la cima de la Pila el valor que carga en el PC; otra más compleja es RETLW k, que además de hacer lo mismo que RETURN, carga en W el valor inmediato k que contiene, es decir, devuelve un parámetro desde la subrutina.

La instrucción RETFIE consiste en cargar en el PC el contenido de la cima de la Pila y poner el bit GIE=1, pues al comenzar la interrupción este bit se pone automáticamente a 0 para evitar que cuando se atiende una interrupción se produzca otra. GIE es el bit de permiso de todas las interrupciones.

En cuanto a las instrucciones especiales, CLRWDT pone a 0 el contenido del Perro Guardián, es decir, lo refresca o lo reinicializa; el Perro Guardián si se desborda (pasa de 0xFF a 0x00) provoca un reset. La instrucción CLRWDT hay que colocarla en forma estratégica en ciertos puntos del programa para evitar la reinicialización.

La instrucción SLEEP introduce al procesador en un modo de funcionamiento que se llama de hibernación o de bajo consumo, detiene el oscilador y el procesador queda congelado, no ejecutando instruc-

ciones y manteniendo el mismo valor los puertos de E/S; también pone los bits $\overline{PD}$=0 y $\overline{TO}$=1 y borra al Perro Guardián y al Divisor de Frecuencia.

LITERAL AND CONTROL OPERATIONS									
ADDLW *	k	Add literal and W	1	11	111x	kkkk	kkkk	C,DC,Z
ANDLW *	k	AND literal with W	1	11	1001	kkkk	kkkk	Z
CALL	k	Call subroutine	2	10	0kkk	kkkk	kkkk	
CLRWDT	-	Clear Watchdog Timer	1	00	0000	0110	0100	$\overline{TO}$,$\overline{PD}$
GOTO	k	Go to address	2	10	1kkk	kkkk	kkkk	
IORLW *	k	Inclusive OR literal with W	1	11	1000	kkkk	kkkk	Z
MOVLW *	k	Move literal to W	1	11	00xx	kkkk	kkkk	
RETFIE	-	Return from interrupt	2	00	0000	0000	1001	
RETLW	k	Return with literal in W	2	11	01xx	kkkk	kkkk	
RETURN	-	Return from Subroutine	2	00	0000	0000	1000	
SLEEP	-	Go into standby mode	1	00	0000	0110	0011	$\overline{TO}$,$\overline{PD}$
SUBLW *	k	Subtract W from literal	1	11	110x	kkkk	kkkk	C,DC,Z
XORLW *	k	Exclusive OR literal with W	1	11	1010	kkkk	kkkk	Z

* Estas instrucciones se repiten

Ejemplo de aplicación 1

Si un PIC 16F84 funciona con un oscilador de cristal de cuarzo de 4 MHz y el perro guardián está programado para que se desborde cada 18 ms (tiempo mínimo y considerado estándar), calcular cada cuantas instrucciones como máximo deberá refrescarse el perro guardián con la instrucción clrwdt.

Solución

$T_{osc}=1/4.10^6 = 250$ ns

Ciclo de instrucción = 4 . F_{osc} = 1000 ns = 1µs

18 ms = 18.000 µs

Por lo tanto, habrá que incluir una instrucción clrwdt cada 18000 instrucciones normales como máximo, las instrucciones de salto cuentan el doble.

- Ejemplo de aplicación 2

Explicar las diferencias entre las tres instrucciones de retorno que posee el Assembler del PIC 16x84.

Solución

- ✓ *return*: retorna de una subrutina al programa principal y carga el PC con el contenido de la cima de la Pila.

- ✓ *retlw k*: hace lo mismo que *return* y, además, carga en W el literal k.

- ✓ *retfie*: carga al PC con el contenido de la cima de la Pila y pone GIE=1.