Programando en *assembler* a los microcontroladores RISC. PIC de microchips

Tito Flórez C.*

RESUMEN

Programar en *assembler* a los PIC se hace relativamente sencillo, cuando se minimiza el número de instrucciones a unas pocas (14 para el PIC16C84). El funcionamiento de esas instrucciones se explica mediante ejemplos sencillos, y el funcionamiento del programa en conjunto se explica con un programa ejemplo. De igual forma se explica la forma como debe de ser *quemado* el PIC.

INTRODUCCIÓN

Comúnmente se cree que programar en lenguaje assembler es algo muy complicado y exclusivo de personal muy capacitado. Lo anterior tiene tal vez relación con el hecho de que empiezan a trabajar con máquinas CISC (computadores con serie de instrucciones complejas) y se pierden al instante en un mar de instrucciones diferentes con diversos formatos. Lo anterior puede solucionarse en buena medida cuando se empieza a trabajar con máquinas RISC (computadores con una serie reducida de instrucciones) y se posea una buena guía inicial, ya que por un lado la cantidad de instrucciones son relativamente pocas, y los formatos similares.

Debido a que los PIC son muy similares en sus instrucciones, programación y arquitectura, permite centrarnos en la programación de uno de ellos, con la seguridad de que los conocimientos adquiridos son fácilmente trasladados a los demás.

I. EL PIC 16C84

Se escoge el PIC 16C84 para trabajar, pues es de fácil consecución, costo reducido (cerca de US$ 4), tamaño pequeño (2,3 cm de largo por 0,8 cm de ancho), es un buen prototipo dentro de los PIC, y posee memoria EEPROM (memoria ROM borrable eléctricamente) que lo hace apto para programarlo directamente una y otra vez, sin tener que estar sometiéndolo a la acción del borrador de luz ultravioleta, lo cual exige tener el borrador y poseer la paciencia y el tiempo que esto conlleva.

En la actualidad salió al mercado el PIC 16F84, el cual posee más registros de propósito general (68) que el PIC 16C84, pero los dos se programan exactamente igual. La configuración de los pines del 16C84 aparece en la figura 1.

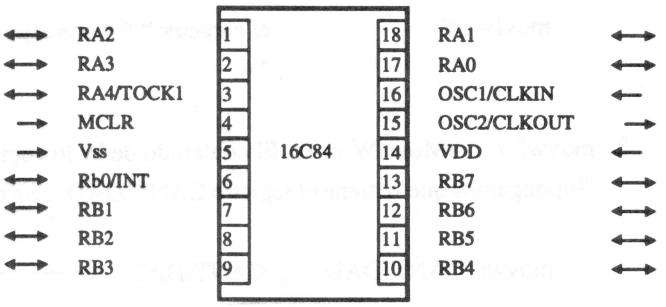

Figura 1. Configuración de los pines del 16C84.

II. INSTRUCCIONES CLAVES

El 16C84 posee 35 instrucciones muy sencillas (la gran mayoría compatibles con los demás PIC.), pero una gran mayoría de los programas pueden realizarse usando solamente las siguientes 14 instrucciones (cada instrucción se explica con un ejemplo), lo cual viene a facilitar enormemente la programación; en donde:

f : representa el nombre de un registro, ejemplo CANTIDAD.
d : si *d* es 0 , el resultado se coloca en W.
Si *d* es 1 , el resultado se coloca en el registro *f*.
b : representa el número del bit del registro *f* sobre el que queremos actuar.

*Departamento de Ingenería de Sistemas. Universidad Nacional, Santa Fe de Bogotá.

k : representa un literal o un rótulo (*label*).

1. movlw k (Move literal to W). Mueve el literal *k* a W.

   Si se desea guardar el número 42 (decimal) se puede proceder así:

```
                        ; un punto y coma indica que empieza ...
                        ;    un comentario.
movlw .42               ; en base 10  (ojo con el punto antes del 42).
movlw D'42'             ; en base 10.

movlw B'00101010'       ; En base 2.

movlw O'52'             ; en base Ocho ( la letra O está antes de la ...
                        ; comilla)

movlw H'2A'             ; en base 16.
movlw 0x2A              ; en base 16.
movlw 2Ah               ; en base 16.
movlw 2A                ; en base 16.

movlw '*'               ; el carácter " * " corresponde al hexadecimal
                        ; 2A.
```

2. movwf f (Move W to f) . El contenido de W lo copia en el registro *f*

   Supongamos que se tiene el registro CANTIDAD, y en él se desea cargar lo que hay en el W (en este caso el decimal 42).

```
movwf CANTIDAD    ; CANTIDAD <------- W.
```

3. movf f,d (Move f). Mueva *f* (el registro *f* conserva su valor).

```
movf CANTIDAD,0   ;     W          <------- CANTIDAD.
movf CANTIDAD,1   ;  CANTIDAD      <------- CANTIDAD.
```

4. decfsz f,d (Decrement f, skip if 0).

   Decrementa en 1 el contenido del registro *f*, y se salta la siguiente instrucción si el resultado en *f* es 0. Si *d* es 0, el resultado se coloca en W, y si *d* es 1, el resultado queda en el registro *f*. Esta instrucción es muy útil para realizar ciclos.

   Supongamos que se quiere realizar un ciclo 42 veces y para ello se desea usar el registro CANTIDAD (el cual contiene el decimal 42). Se procede a realizar el ciclo de la siguiente forma:

```
OTRA_VEZ            ...
                    ...
                    ...
        instrucciones dentro del ciclo que se realiza 42 veces
                    ...
                    ...
        decfsz CANTIDAD, 1  ; si llegó a 0 salta a SIGUE_DE_LARGO.
        goto    OTRA_VEZ    ; si no llegó a 0 regresa a OTRA_VEZ.

SIGUE_DE_LARGO
                    ...
                    ...
                    ...
```

5. addwf f,d (Add W and f). Suma W con el registro *f*.

```
addwf   CANTIDAD,0      ;     W       <------  CANTIDAD + W.
addwf   CANTIDAD,1      ; CANTIDAD    <------  CANTIDAD + W.
```

6. subwf f,d (Subtract W from f). Resta W de f.

```
subwf   CANTIDAD,0      ;     W       <------  CANTIDAD - W.
subwf   CANTIDAD,1      ; CANTIDAD    <------  CANTIDAD - W.
```

7. bcf f,b (Bit Clear f). El bit # *b* del registro *f* lo pone en 0.

```
bcf   CANTIDAD,5        ; el bit # 5 de CANTIDAD se pone en 0.
                        ; acordarse que el bit menos significativo,
                        ; se denomina el bit # 0.
```

8. bsf f,b (Bit Set f). El bit # *b* del registro *f* se pone en 1.

```
bsf   CANTIDAD,5        ; el bit # 5 de CANTIDAD se pone en 1.
```

9. goto k (Go to address). Va al rótulo *k.*

```
goto   SITIO_1          ; Vaya al SITIO_1 ( goto: es una sola palabra ).
```

10. btfss f,b (Bit Test f, Skip if Set). Salte si el bit # *b* del registro *f* es 1 .

Es muy importante para tomar decisiones cuando se comparan dos números. Para saber si dos números son iguales, pueden restarse, y si son iguales el bit # 2 (Z) del registro STATUS se pone en 1 . Luego se puede usar la instrucción btfss STATUS,2 para cambiar el curso normal del programa.

```
subwf   CANTIDAD,0      ; W  <-------  CANTIDAD - W.
btfss   STATUS,2        ; salte si el bit # 2 de STATUS es 1.
goto    SITIO_1         ; si CANTIDAD  no =  W, se va a SITIO_1.
goto    SITIO_2         ; si CANTIDAD     =  W, se va a  SITIO_2.
```

Para saber si el contenido del registro *f* es mayor que lo que hay en W, se puede restar W del registro *f,* y si el contenido del registro *f* es mayor, el bit # 0 (C) del registro STATUS se pone en " 1 ". Luego se puede usar la instrucción: btfss STATUS,0 para cambiar el curso normal del programa.

```
subwf   CANTIDAD,0      ; W  <-------  CANTIDAD - W.
btfss   STATUS,0        ; salte si el bit # 0 de STATUS es 1.
goto    SITIO_1         ; si CANTIDAD  <=  W  se va a SITIO_1.
goto    SITIO_2         ; si CANTIDAD   >  W  se va a  SITIO_2.
```

11. call k (Call subroutine). Va a ejecutar la subrutina *k.*

```
call    INICIALIZAR     ;  Va a la subrutina INICIALIZAR.
```

12. return (Return from subroutine). Regrese de la subrutina, a la instrucción siguiente en donde fue llamada.

```
return                  ; retorne.
```

13. andwf f,d (AND W and f). Realiza un AND entre el registro *f* y W.

```
andwf  CANTIDAD,0       ; W  <-------  CANTIDAD  AND  W.
```

14. rlf f,d (Rotate left through carry). Rote a la izquierda a través del *carry* .

Para esta instrucción se puede considerar que el acarreo (*carry*), (bit # 0 del registro STATUS) se añade a un extremo del registro *f* y se rota una posición a la izquierda. El valor del *carry* queda en el bit # 0 de *f* y el bit # 7 de *f* queda en el *carry* .

```
rlf   CANTIDAD,0        ; CANTIDAD se rota una posición a la
        ...             ; izquierda.
```

III. Ejemplo

Con la información anterior, aun cuando sencilla, se está en completa capacidad para empezar a programar los PIC (se debe usar un editor o procesador de palabra en solo TEXTO). Cuando se va a realizar por primera vez una programación en los PIC, se está desorientado, pues no se sabe cómo definir los registros, cómo y en qué lugar deben ir las instrucciones, los comentarios, las subrutina etcétera. Mas todo lo anterior se soluciona fácilmente observando el siguiente ejemplo:

```
;=================================================================

; este programa, suma  dos números binarios, cada uno de tres bits.
; El primer número (que se guardará en DATO1), entrará por los tres bits ...
;    menos significativos del puerto A, es decir,  por RA0, RA1 y RA2, ...
;    que corresponde a los pines 17, 18 y 1 respectivamente.
; El segundo número (que se guardará en DATO2), entrará por los tres bits...
;   menos significativos del puerto B, es decir,  por RB0, RB1 y RB2, que ...
;   corresponde a los pines 6, 7 y 8, respectivamente.
; La suma (que se guardará en SUMA), saldrá por los cuatro bits más ...
;   significativos del puerto B, es decir, por RB4, RB5, RB6 y RB7 respectivamente.
; Si colocamos en SUMA  el resultado de:  DATO1 + DATO2, siendo DATO1...
;    y DATO2 de tres bits cada uno, el resultado será máximo de cuatro bits,...
;    y ocupará solamente los cuatro bits menos significativos de SUMA. Como ...
;    dicho resultado se piensa sacar por los cuatro bits más significativos de PORTB,...
;    una forma de hacerlo es rotando el contenido de SUMA cuatro bits hacia la ...
;    izquierda, de tal forma que ahora quede la respuesta en los cuatro bits más ...
;    significativos de SUMA, y después trasladamos SUMA hacia PORTB, ...
;    haciendo coincidir de esta forma la respuesta con los cuatro bits más ...
     significativos de PORTB.
; En todos los casos anteriores, el bit menos significativo del número, se hace ...
;    corresponder con el bit menos significativo de los respectivos bits del puerto asignado.

;=================================================================

STATUS   equ   03  ; estas son las direcciones de estos registros, las cuales...
PORTA    equ   05  ;  se encuentran en una de las figuras del manual  ...
TRISA    equ   85  ;   MICROCHIP DATA BOOK, en el capítulo del...
PORTB    equ   06  ;   PIC16C84, bajo el nombre de ...
TRISB          equ   86  ;   REGISTERS FILE MAP.
INTCON   equ   0B

DATO1    equ   0C  ; en la dirección 0C empiezan los registros de ...
DATO2    equ   0D  ; propósito general.
SUMA     equ   0E
;=================================================================
```

```
;=================================================================
        org   0x00                  ; se colocan en la posición 00 hexadecimal.

        goto  INICIO
        org   0x04                  ; se colocan en la  posición 04 hexadecimal.

        goto  INTERRUPCIONES        ; vector de interrupción. Cuando se ...
                                    ;   manejan interrupciones y ocurre una, ...
                                    ;   el programa va a esta subrutina.
        org   0x10                  ; se colocan en la  posición 10 hexadecimal.

;=================================================================
;                   Aquí empieza el programa principal.

INICIO nop
        call  INICIALIZAR           ; se llama a la subrutina INICIALIZAR

CICLO nop
        movf  PORTA,0               ; W      <------   PORTA
        andlw B'00000111'           ; W      <------   W AND "00000111"
        movwf DATO1                 ; DATO1  <------   W

        movf  PORTB,0               ; W      <------   PORTB
        andlw B'00000111'           ; W      <------   W AND "00000111"
        movwf DATO2                 ; DATO2  <------   W

        movf  DATO1,0               ; W      <------   DATO1
        addwf DATO2,0               ; W      <------   W + DATO2
        movwf SUMA                  ; SUMA   <------   W

        rlf   SUMA,1                ; se rota SUMA hacia la izquierda 4 bits,...
        rlf   SUMA,1                ;  para que la suma, que abarca sus 4 bits ...
        rlf   SUMA,1                ;  menos, los podamos colocar en sus 4 ...
        rlf   SUMA,1                ;  bits más significativos, para hacerlo ...
                                    ;  coincidir con los 4 bits más significativos...
                                    ;  de PORTB, que es por donde se va a ...
                                    ;  sacar el resultado de la suma.

        movf  SUMA,0                ; W      <------   SUMA
        movwf PORTB                 ; PORTB  <------   DATO1. Se saca el ...

                                            ; resultado de la suma por RB4 - RB7

        goto  CICLO                 ; vaya de nuevo a CICLO
; Aquí termina el programa principal.
;=================================================================
```

```
; ============================================================

; en esta subrutina definimos los pines de entrada y de salida.

INICIALIZAR
      bsf   STATUS,5      ; selecciona pagina # 1, ya que en ésta se ...
                          ; encuentran los registros TRISA y TRISB, los ...
                          ; cuales sirven para definir los pines de ...
                          ; entrada y los pines de salida.
      movlw B'00000111'   ; RA0 - RA2 como entradas ( un 1 es entrada  y ...
      movwf TRISA         ;                un 0 es salida )

      movlw B'00000111'   ; RB0-RB2 se definen como entradas y ...
      movwf TRISB         ; RB3-RB7 se definen como salidas.

      bcf   STATUS,5      ; regresa a la página 0 en la cual funciona ...
                          ; el programa.
Fin_INICIALIZAR return    ; fin del subprograma INICIALIZAR ....
                          ; ponerle SIEMPRE rótulo al  return.

; ============================================================

INTERRUPCIONES
                          ; no  está trabajándose con interrupciones,
                          ; por eso está vacía la subrutina.
Fin_INTERRUPCIONES retfie ; fin de la subrutina de Interrupción ...
                          ; ponerle SIEMPRE rótulo al  retfie.

; ============================================================

      end                 ; FIN

; ============================================================
```

IV. Programar *(quemar)* el PIC

Una vez escrito en solo texto el programa, cuyo nombre debe terminar en .ASM (nombre.ASM), se pasa a lenguaje de máquina, usando para ello un programa ensamblador como el MPASM.EXE. Dicho programa puede producir varios archivos de salida, y el más importante es el terminado en .HEX (nombre.HEX). Otro programa de salida es el terminado en .LST(nombre.LST); es muy importante para encontrar errores en la programación (este nombre.LST puede ser analizado por el programador, cuando lo llame desde el editor o procesador de palabra). El cual produce un archivo llamado «nombre, HEX»

Antes de proceder a programar *(quemar)* el microcontrolador, verificar que el PIC esté correctamente situado en el programador, ya que si se usa el:

1. PICSTART-16B: debe colocarse justificado hacia el extremo derecho.
2. PICSTART PLUS: debe colocarse justificado hacia el extremo izquierdo.

Para programar el PIC, se corre un programa como el MPS16B.EXE, el cual inicialmente trata de hacer conexión con el programador a través del puerto serial COM1 del computador. Si el COM1 está ocupado, colocarlo en COM2 - COM4 y suministrar esta información, seleccionando del menú: *Options* y después *Comm Port Selection* (si no se posee sino el COM1, es necesario desconectar el *ratón*, y en su lugar conectar el programador).

Si no se posee el *ratón*, el movimiento a través de este menú, puede hacerse con las teclas de *Tabulación*, las *flechas* y el *espaciador*, este último sirve también para escoger la opción deseada.

Una vez dentro del menú:

1. "Device edit": para escoger el PIC se desea programar (en el ejemplo que se va a desarrollar se escoge el 16C84).

2. "fuse Edit" :

   a. "LP", "XT" o "HS" : si se va a usar un cristal como oscilador (dependiendo de su frecuencia de oscilación).

   b. "RC" : si se va a usar como oscilador una resistencia (Rext) y un condensador (Cext). Un buen valor para la resistencia puede estar entre 3 Kohm y 100 Kohm y para el condensador, valores mayores de 20 picofaradios.

   c. "Watchdog Timer On" : cuando se vaya a usar el Watchdog, la forma de seleccionarlo es colocando la X es su respectivo renglón (como en nuestro ejemplo no se usará el "Watchdog", anulamos la X).

3. "File" : en él se escoge el archivo deseado "nombre.HEX".

4. "Program" : en él se programa finalmente el PIC.

Si lo que se posee es el sistema de desarrollo PICSTART PLUS, el cual es diseñado para correr desde Windows, la información anterior se suministra de forma más cómoda, ya que no es nesesario decirle explicitamente que programas debe correr.

V. Montaje del PIC

Una vez montado en el "Protoboard" el PIC 16C84, verificar que:

1. El pin # 5 vaya a 0 voltios (Vss).
2. Los pines # 4 y #14 vayan al voltaje de alimentación Vdd.
3. Si se va a trabajar con RC, del pin # 16 debe salir la resistencia Rext hacia el voltaje Vdd. y el condensador Cext hacia 0 voltios. Si no se va a trabajar con RC, remitirse al manual a *Oscillator Configurations.*
4. Para este ejemplo en particular, si se van a usar LED para detectar los bits de salida, cerciorase primero que de los pines # 10, #11, #12 y # 13, que corresponden a RB4-RB7, salen sus respectivas resistencias de 1 Kohm hacia los LED.

Conclusiones

- Los PIC son excelentes para iniciarse en la programación del lenguaje *assembler.*

- Programar los PIC, es relativamente sencillo cuando se conocen cuáles instrucciones son las importantes y se posee una buena guía de ejecución.

- La programación de los microcontroladores no es exclusivo de personal especializado en lenguaje *assembler* con grandes conocimientos en sistemas y electrónica.

- Un individuo con algunos conocimientos en programación, puede perfectamente empezar a manejar los microcontroladores.

BIBLIOGRAFÍA

1. Emb*edded Control Handbook. 1993. Microchip Tech*nology Inc.,USA.

2. Mic*rochip Data Book. 1994. Microchi*p Technology Inc.,USA.

3. MPA*SM User's Guide. 1994. Microchi*p Technology Inc.,USA.

4. MPS*IM User's Guide. 1994. Microchi*p Technology Inc.,USA.

5. PIC*Start-16B1 User's Guide. 1994. Microchip Technol*ogy Inc.,USA.

6. PIC*Start Plus, Development System User's Guide. 1996. Microchip Technology Inc.,USA.*