

PCLATH-PCL-PC Y TABLAS

HACER LLAMADO A TABLAS DE TABLAS

POR: BrunoF

www.todopic.com.ar

LOW y HIGH(y UPPER que no utilicé aquí) son tres directivas utilizadas para indicarle cuál byte utilizaremos cuando nos enfrentamos a un valor que supera el byte de tamaño.

Por ejemplo:

Si tengo como valor 0x8FA5, cuyo tamaño es de 2 bytes, si escribo:

movlw LOW 0x8FA5

el ensamblador tomará como valor el byte bajo, quedando: movlw 0xA5

Si ahora escribo:

movlw HIGH 0x8FA5

el ensamblador tomará como valor el byte alto, quedando: movlw 0x8F

El uso de estas directivas es necesario cuando usas valores que superan el byte de tamaño porque la instrucción movlw sólo puede mover un literal de 8 bits a W.

Si pones:

movlw 0x8FA5

el ensamblador va a descartar por defecto todos aquellos bits que no pueda copiar porque excede el tamaño esperado. Entonces, sólo se quedará con los 8 bits de menor peso del valor declarado. Entonces, se reducirá a: movlw 0xA5. En este caso el comportamiento es idéntico al del movlw LOW 0x8FA5, aunque lo adecuado es utilizar el LOW para no confundirse. Si no usas el LOW, el MPASM emitirá una advertencia indicandote que va a omitir todos los bits superiores y se va a quedar solo con los últimos 8.

Mi algoritmo está basado en el de Maunich. Lo que se hace, es saltar a cualquier parte de la memoria FLASH, usando el PC. No importa ni siquiera en qué banco está. Si cargas el PCLATH con los valores adecuados y luego modificas el valor del PCL, el programa saltará a la posición de memoria indicada por el PC(para saber cómo está conformado el PC, lee el tutorial de León que ahí lo explico).

Cada tabla está conformada por una serie de valores, que van leyéndose utilizando la instrucción RETLW valorX. Como lo que se desea era acceder en tiempo de ejecución a alguna de ellas, indicándolo mediante una variable(elemento en mi algoritmo), y dentro de ellas ir recorriendo cada valor utilizando otra variable(llamada subelemento en mi algoritmo) lo que hice fue implementar otra tabla. Esta tabla es un MAPA. Un MAPA que indica dónde se encuentra cada una de las tablas a las que se desea acceder. Esta tabla contiene las posiciones en las que se encuentra cada tabla ubicada dentro de la memoria FLASH del PIC.

El problema, es que como el PIC posee más de 256 posiciones FLASH, no me alcanza con una tabla para obtener el valor de la posición a la cual debo saltar. Por eso, he partido la tabla del MAPA en dos partes. Una tabla(DirBaja) con la parte baja de la dirección a la que debe ir y otra tabla(DirAlta) con la parte alta de la dirección a la que debe ir.

Entonces el programa va a hacer lo siguiente:

Primero:

El usuario elige el número de tabla del cual desea leerse un subelemento. Lo hace mediante la variable elemento. Para optimizar un poco, el número de tabla a leer se pasa como argumento en W, luego la subrutina(leer_elemento) lo almacena inmediatamente en la variable "elemento". Y también el usuario especifica el subelemento a leer de la tabla, mediante la variable "subelemento". Este es el que(comenzando desde 0 por lo general) y luego incrementándolo cada vez que se llama a la subrutina "leer_elemento" va devolviendo uno a uno cada valor de la tabla seleccionada.

Segundo:

Dentro de la subrutina "leer_elemento" lo que se procede a hacer es a obtener la dirección de la tabla a leer. Para obtener la dirección, se utiliza otra tabla, que a su vez se divide en 2 tablas(DirBaja y DirAlta) mencionadas anteriormente.

Entonces, primero se va a saltar a la posición de la FLASH indicada por: DirBaja+elemento. Llamamos a la subrutina que hará el salto: SaltaYLee.

Como DirBaja puede estar ubicada en cualquier lugar de la FLASH, es necesario "partir" su dirección en 2 partes: baja y alta. La alta se cargará en el PCLATH, y la baja sumada al número de tabla a leer("elemento") se moverá al registro PCL. al afectar el registro PCL, el programa automáticamente salta a la posición pedida. Si hemos saltado correctamente, entonces habremos saltado a una instrucción RETLW x, por lo que retornaremos a la subrutina "leer_memoria" con W cargado con la posición FLASH baja de la tabla a la que deseamos acceder. Almacenamos el valor obtenido en "dirL";

Luego, se va a saltar a la posición de la FLASH indicada por: DirAlta+elemento.

Llamamos a la subrutina que hará el salto: SaltaYLee.

Como DirAlta puede estar ubicada en cualquier lugar de la FLASH, es necesario "partir" su dirección en 2 partes: baja y alta. La alta se cargará en el PCLATH, y la baja sumada al número de tabla a leer("elemento") se moverá al registro PCL. al afectar el registro PCL, el programa automáticamente salta a la posición pedida. Si hemos saltado correctamente, entonces habremos saltado a una instrucción RETLW x, por lo que retornaremos a la subrutina "leer_memoria" con W cargado con la posición FLASH alta de la tabla a la que deseamos acceder. Almacenamos el valor obtenido en "dirH";

Entonces, tenemos finalmente la posición de la tabla a la que deseamos acceder. Las variables "dirH" y "dirL" contienen la posición FLASH de la tabla(que es la del primer elemento de la tabla,

es decir el primer RETLW X) a la que se desea acceder.

Tercero:

Ahora que sabemos en qué posición de la FLASH está ubicada la tabla, agregaremos el valor de "subelemento" a la posición encontrada, para poder acceder a un RETLW X específico dentro de la tabla elegida.

Acá viene una diferencia: Para saltar a la tabla, no voy a utilizarla como subrutina a la etiqueta "SaltaYLee" sino que voy a saltar a ella usando un "goto" en lugar de un "call".

Cuarto:

Cargo el valor de "dirH" en el PCLATH, y sumo a "dirL" el valor de "subelemento". Muevo esta suma al PCL. El PC saltará a la posición deseada. Si hicimos todo bien saltará al subelemento elegido dentro de la tabla elegido. Este debería ser un RETLW X, por lo cargará a W con el valor X y volverá. Como había usado un "goto" en lugar de un "call", el programa sale de la subrutina "leer_elemento", "aterrizando" finalmente ante la instrucción "nop" con W conteniendo el valor deseado.

Te hago un ejemplo, pero indicando también la posición de dónde se guarda cada instrucción en la FLASH del PIC para que lo puedas entender mejor:

```
list    P=16F876
include <p16F876.inc>
```

```
elemento      equ      0x20
subelemento    equ      0x21
```

```
tmpL          equ      0x22
tmpH          equ      0x23
tmpsub        equ      0x24
```

```
dirL          equ      0x25
dirH          equ      0x26
```

```
    movlw      0x00          ;subelemento del elemento a leer(guardado en 0x000)
    movwf      subelemento    ;(guardado en 0x001)
    movlw      0x01          ;elemento a leer(guardado en 0x002)
    pagesel    leer_elemento  ;(guardado en 0x003 y 0x004)
    call       leer_elemento  ;(guardado en 0x005)
```

nop ;(guardado en 0x006)
 (luego del nop se carga cualquier registro con el valor de W que retorna luego de el llamado de las tablas)

leer_elemento

movwf	elemento	;(guardado en 0x007)
movlw	LOW DirBaja	;(guardado en 0x008)
movwf	tmpL	;(guardado en 0x009)
movlw	HIGH DirBaja	;(guardado en 0x00A)
movwf	tmpH	;(guardado en 0x00B)
movf	elemento,w	;(guardado en 0x00C)
movwf	tmpsub	;(guardado en 0x00D)
movf	elemento,W	;(guardado en 0x00E)
pagesel	SaltaYLee	;(guardado en 0x00F Y 0X010)
call	SaltaYLee	;(guardado en 0x011)
movwf	dirL	;(guardado en 0x012)
movlw	LOW DirAlta	;(guardado en 0x013)
movwf	tmpL	;(guardado en 0x014)
movlw	HIGH DirAlta	;(guardado en 0x015)
movwf	tmpH	;(guardado en 0x016)
movf	elemento,w	;(guardado en 0x017)
movwf	tmpsub	;(guardado en 0x018)
movf	elemento,W	;(guardado en 0x019)
pagesel	SaltaYLee	;(guardado en 0x01A Y 0X01B)
call	SaltaYLee	;(guardado en 0x01C)
movwf	dirH	;(guardado en 0x01D)
movf	dirL,W	;(guardado en 0x01E)
movwf	tmpL	;(guardado en 0x01F)
movf	dirH,W	;(guardado en 0x020)
movwf	tmpH	;(guardado en 0x021)
movf	subelemento,w	;(guardado en 0x022)
movwf	tmpsub	;(guardado en 0x023)
pagesel	SaltaYLee	;(guardado en 0x024)
goto	SaltaYLee	;(guardado en 0x025)

SaltaYLee

movf	tmpH,W	;(guardado en 0x026)
movwf	PCLATH	;(guardado en 0x027)
movf	tmpL,W	;(guardado en 0x028)

```

addwf    tmpsub,W      ;(guardado en 0x029)
btfsc    STATUS,C      ;(guardado en 0x02A)
incf     PCLATH,F      ;(guardado en 0x02B)

```

```

movwf    PCL            ;(guardado en 0x02C)

```

org 0X1300 ;agrego esta linea para hacerlo un poco más real,
porque las tablas seguramente irán al final de todo el programa.

DirBaja

```

retlw    LOW  tabla0    ;(guardado en 0x1300)
retlw    LOW  tabla1    ;(guardado en 0x1301)

```

DirAlta

```

retlw    HIGH tabla0    ;(guardado en 0x1302)
retlw    HIGH tabla1    ;(guardado en 0x1303)

```

org 0X19FE ;lo mismo, más real. Puede que el código este
desordenado.

```

                                ;'H'   'O'   'L'   'A'   0X00
tabla0: dt  "HOLA", 0          ;(guardado en 0x19FE,0x19FF,0x1A00,0x1A01,0x1A02)
                                ;'B'   'R'   'U'   'N'   'O'   'F'
                                ''   'E'   'S'
tabla1: dt  "BRUNOF ESTUVO AQUI", 0 ;(guardado en
0x1A03,0x1A04,0x1A05,0x1A06,0x1A07,0x1A08,0x1A09,0x1A0A,0x1A0B
                                ;'T'   'U'   'V'   'O'   ''   'A'
'Q'   'U'   'I'   0X00
                                ;
0x1A0C,0x1A0D,0x1A0E,0x1A0F,0x1A10,0x1A11,0x1A12,0x1A13,0x1A14,0x1A15
END

```

Ahora, el mismo ejemplo, pero muestro cómo quedarían las dos tablas mas importantes cuando ensamblás el código:

```

list  P=16F876
include <p16F876.inc>

```

```

elemento    equ    0x20
subelemento equ    0x21

```

tmpL	equ	0x22
tmpH	equ	0x23
tmpsub	equ	0x24

dirL	equ	0x25
dirH	equ	0x26

movlw	0x00	;subelemento del elemento a leer(guardado en 0x000)
movwf	subelemento	; (guardado en 0x001)
movlw	0x01	;elemento a leer(guardado en 0x002)
pagesel	leer_elemento	; (guardado en 0x003 y 0x004)
call	leer_elemento	; (guardado en 0x005)
nop		; (guardado en 0x006)

leer_elemento

movwf	elemento	; (guardado en 0x007)
movlw	0x00	;movlw LOW DirBaja ; (guardado en 0x008)
movwf	tmpL	; (guardado en 0x009)
movlw	0x13	;movlw HIGH DirBaja ; (guardado en 0x00A)
movwf	tmpH	; (guardado en 0x00B)
movf	elemento,w	; (guardado en 0x00C)
movwf	tmpsub	; (guardado en 0x00D)
movf	elemento,W	; (guardado en 0x00E)
pagesel	SaltaYLee	; (guardado en 0x00F Y 0x010)
call	SaltaYLee	; (guardado en 0x011)
movwf	dirL	; (guardado en 0x012)

movlw	0x02	;movlw LOW DirAlta ; (guardado en 0x013)
movwf	tmpL	; (guardado en 0x014)
movlw	0x13	;movlw HIGH DirAlta ; (guardado en 0x015)
movwf	tmpH	; (guardado en 0x016)
movf	elemento,w	; (guardado en 0x017)
movwf	tmpsub	; (guardado en 0x018)
movf	elemento,W	; (guardado en 0x019)
pagesel	SaltaYLee	; (guardado en 0x01A Y 0x01B)
call	SaltaYLee	; (guardado en 0x01C)

movwf	dirH	; (guardado en 0x01D)
-------	------	-----------------------

movf	dirL,W	; (guardado en 0x01E)
movwf	tmpL	; (guardado en 0x01F)

```

movf      dirH,W           ;(guardado en 0x020)
movwf     tmpH             ;(guardado en 0x021)
movf      subelemento,w    ;(guardado en 0x022)
movwf     tmpsub           ;(guardado en 0x023)
pagesel   SaltaYLee        ;(guardado en 0x024)
goto      SaltaYLee        ;(guardado en 0x025)

```

SaltaYLee

```

movf      tmpH,W           ;(guardado en 0x026)
movwf     PCLATH           ;(guardado en 0x027)

```

```

movf      tmpL,W           ;(guardado en 0x028)
addwf     tmpsub,W         ;(guardado en 0x029)
btfsc     STATUS,C        ;(guardado en 0x02A)
incf      PCLATH,F        ;(guardado en 0x02B)

```

```

movwf     PCL              ;(guardado en 0x02C)

```

org 0X1300 ;agrego esta linea para hacerlo un poco más real,
porque las tablas seguramente irán al final de todo el programa.

DirBaja

```

retlw    0xFE           ;retlw LOW tabla0      ;(guardado en 0x1300)
retlw    0x03           ;retlw LOW tabla1      ;(guardado en 0x1301)

```

DirAlta

```

retlw     0x19             ;retlw HIGH tabla0    ;(guardado en 0x1302)
movlw     0x1A             ;retlw HIGH tabla1    ;(guardado en 0x1303)

```

org 0X19FE ;lo mismo, más real. Puede que el código este
desordenado.

```

                                ;'H'   'O'   'L'   'A'   0X00
tabla0: dt  "HOLA", 0          ;(guardado en 0x19FE,0x19FF,0x1A00,0x1A01,0x1A02)
                                ;'B'   'R'   'U'   'N'   'O'   'F'
                                ''     'E'   'S'
tabla1: dt  "BRUNOF ESTUVO AQUI", 0 ;(guardado en
0x1A03,0x1A04,0x1A05,0x1A06,0x1A07,0x1A08,0x1A09,0x1A0A,0x1A0B
                                ;'T'   'U'   'V'   'O'   ''   'A'
'Q'   'U'   'I'   0X00
                                ;
0x1A0C,0x1A0D,0x1A0E,0x1A0F,0x1A10,0x1A11,0x1A12,0x1A13,0x1A14,0x1A15
END

```

Fijate que las tablas DirBaja y DirAlta almacenan las posiciones de las tablas, parte baja y alta

Fijate que las tablas DirBaja y DirAlta almacenan las posiciones de las tablas, parte baja y alta respectivamente.

En cuanto al STACK: Te marco los niveles(del STACK) con el codigo resumido para que lo veas:

Inicialmente NIVEL =0

```
;...
call leer_elemento      ;NIVEL=1
;...
call SaltaYLee          ;NIVEL=2
retlw X                 ;NIVEL=1
;...
call SaltaYLee          ;NIVEL=2
retlw X                 ;NIVEL=1
;...
goto SaltaYLee          ;NIVEL=1. el goto no carga el STACK
retlw X                 ;NIVEL=0
nop
```

El <, 0> al final de las tablas significa que estás agregando un retlw 0x00 al final de cada tabla, para indicar el fin de la misma(lo que estabas usando en tu programa). La ',' concatena valores o cadenas de texto.

La limitación es de 256 WORDS POR TABLA. Esto quiere decir que CADA tabla no debería exceder los 256 WORDS de longitud. Si es necesario una tabla de mayor tamaño, debes utilizar una variable de 16 bits para realizar el offset(es decir, subelemento debe ser de 16 bits).Las tablas en su conjunto pueden usar toda la FLASH disponible si te da la gana.