TECNOLOGÍA ELECTRÓNICA DE COMPUTADORES

2º Curso – GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

Tema 9: Circuitos integrados: microcontroladores

Lección 19. Microcontroladores PIC 16xxx: Programación y juego de instrucciones

Lección 19. Microcontroladores PIC 16xxx: programación y juego de instrucciones

19.1. Programación en lenguaje ensamblador

19.2. Estructuración del código fuente: campos

19.3. Juego de instrucciones

19.4. Ciclo de instrucción

19.5. Directivas y macros

19.6. Información adicional

Programación de microcontroladores PIC

Para poder programar una aplicación con el microcontrolador, es necesario definir y conocer:

1. Hardware a utilizar en la aplicación desarrollada:
   - Organización de la memoria interna (de programa y datos)
   - Dispositivos de E/S a utilizar y cómo gestionarlos
2.- Juego de Instrucciones

3.- Directivas y Macros

19.1. Programación en lenguaje ensamblador

• El único lenguaje que entienden los microcontroladores es el código máquina formado por ceros y unos del sistema binario, en el que está codificada la secuencia de instrucciones a ejecutar.

• El lenguaje ensamblador permite:

  • Expresar las instrucciones de una forma más natural al ser humano

  • Trabajar de forma similar a cómo lo hace el microcontrolador, ya que cada una de esas instrucciones se corresponde con otra en código máquina.

• El lenguaje ensamblador trabaja con nemónicos, que son grupos de caracteres alfanuméricos que simbolizan las órdenes o tareas a realizar.

•La traducción de los nemónicos a código máquina entendible por el microcontrolador la lleva a cabo un programa ensamblador.

• El programa escrito en lenguaje ensamblador se denomina código fuente (*.asm). El programa ensamblador proporciona a partir de este fichero el correspondiente código máquina, que suele tener la extensión *.hex.

Lenguaje ensamblador
y
Programa ensamblador

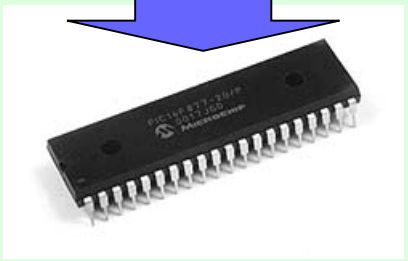
Código Fuente
.asm

Lenguaje
Ensamblador

Programa
Ensamblador
(MPASM)

110100001….1100

Código Máquina
.hex



19.2. Estructuración del código fuente: campos

• **El código fuente** está compuesto por una sucesión de líneas de texto.

• Cada línea puede estructurarse en hasta cuatro campos o columnas separados por uno o más espacios o tabulaciones entre sí.

- Campo de etiquetas. Expresiones alfanuméricas escogidas por el usuario para identificar una determinada línea. Todas las etiquetas tienen asignado el valor de la posición de memoria en la que se encuentra el código al que acompañan.
- Campo de código. Corresponde al nemónico de una instrucción, de una directiva o de una llamada a macro.
- Campo de operandos y datos. Contiene los operandos que precisa el nemónico utilizado. Según el código, puede haber dos, uno o ningún operando. Los operandos pueden ser etiquetas que definen variables
- Campo de comentarios. Dentro de una línea, todo lo que se encuentre a continuación de un punto y coma (;) será ignorado por el programa ensamblador y considerado como comentario.

Campo de código

Puede corresponder ese código a:

- Instrucciones: son aquellos nemónicos que son convertidos por el ensamblador en código máquina que puede ejecutar el núcleo del microcontrolador. En la gama media (PIC16xxx) cada nemónico se convierte en una palabra en la memoria de programa

- Directivas. Pseudoinstrucciones que controlan el proceso de ensamblado del programa, pero no son parte del código. Son indicaciones al programa ensamblador de cómo tiene que generar el código máquina

- Macros: Secuencia de nemónicos que pueden insertarse en el código fuente del ensamblador de una manera abreviada mediante una simple llamada.

Ejemplo de código fuente

```
; Fichero CUENTA.ASM
;
; Programa de Prueba para la placa PICDEM-2 plus
; Por el Puerto B se saca en binario, el numero de veces
; que se pulsó el pulsador conectado a la entrada RA4
; si el pulsador está pulsado la entrada está a 0 y si no está pulsado está a 1
;
                    LIST        P=16F877        ; Directiva para definir listado y microcontrolador
                    INCLUDE     P16F877.INC     ; Inclusión de fichero de etiquetas
                    ORG         0
                    BSF         STATUS,RP0      ; Paso al banco 1 de la memoria de datos
                    CLRF        TRISB           ; para definir el PORTB como salida
                    BCF         STATUS,RP0      ; Volvemos al banco 0
                    CLRF        PORTB           ; Ponemos a cero el PORTB para que aparezca ese
                                                ; valor cuando se defina como salida
ESPERA              BTFSS       PORTA,4         ; Esperamos a que se pulse la tecla
                    CALL        INCREMENTO      ; en cuyo caso RA4 pasa a 0 y vamos a
                    GOTO        ESPERA          ; subprograma de INCREMENTO
;Subprograma de INCREMENTO

INCREMENTO          INCF        PORTB,F         ; Si se pulsó incrementamos PORTB
SOLTAR
                    BTFSS       PORTA,4         ; no salimos hasta que se haya soltado
                    GOTO        SOLTAR          ; la tecla, en ese caso RA4 pasaría a 1
                    RETURN                      ; y volvemos al programa principal
                    END
```

Campo de Operandos y Datos

• El ensamblador MPASM (distribuido por Microchip) soporta los sistemas de numeración decimal, hexadecimal, octal, binario y ASCII.

• Los nemónicos que tengan una constante como operando deberán incluirla respetando la sintaxis que se indica a continuación.

TIPO	SINTAXIS		
Decimal	**D'**<valor>'	**d'**<valor>'	.<valor>
Hexadecimal	**H'**<valor>'	**h'**<valor>'	**0x**<valor>
	<valor>**H**	<valor>**h**	
Octal	**O'**<valor>'	**o'**<valor>'	
Binario	**B'**<valor>'	**b'**<valor>'	
ASCII	**A'**<carácter>'	**a'**<carácter>'	'<carácter>'
Cadena	"<cadena>"		

Las constantes hexadecimales que empiecen por una letra deben ir precedidas de un cero para no confundirlas con una etiqueta. Ejemplo: ***movlw 0F7h***

Programación de microcontroladores PIC

Para poder programar una aplicación con el microcontrolador, es necesario definir y conocer:

1. Hardware a utilizar en la aplicación desarrollada:
   - Organización de la memoria interna (de programa y datos)
   - Dispositivos de E/S a utilizar y cómo gestionarlos
2. Juego de Instrucciones
3. Directivas y Macros

19.3. Juego de instrucciones

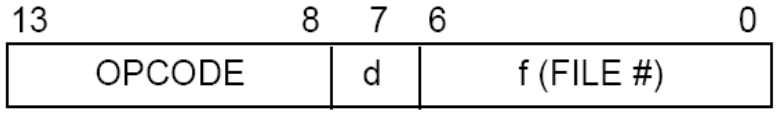
- Es un juego reducido de 35 instrucciones muy simples (Procesador RISC = Reduced Instruction Set Computer)
- La mayoría de las instrucciones se ejecuta en 4 ciclos de reloj; los saltos y llamadas a subprogramas se ejecutan en 8 (no se aprovecha *pipeline*)
- Todas las instrucciones tienen la misma longitud en la gama media: 14 bits.
- Por lo tanto el cálculo del tiempo de ejecución y de lo que ocupa un programa resulta simple
- Las instrucciones se pueden clasificar atendiendo a dos criterios:

  1. Formato
  2. Funcionalidad

El juego de instrucciones: SEGÚN SU FORMATO

1.- Orientadas al byte

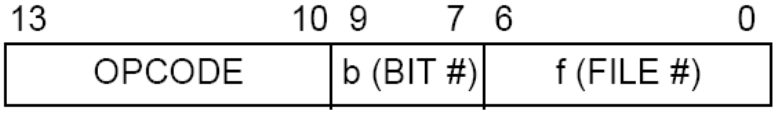
Byte-oriented file register operations



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

2.- Orientadas al bit

Bit-oriented file register operations

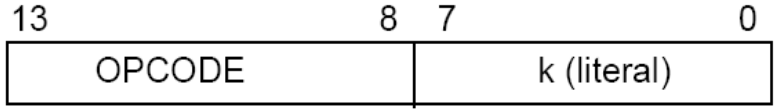


b = 3-bit bit address
f = 7-bit file register address

3.- Literales y de control

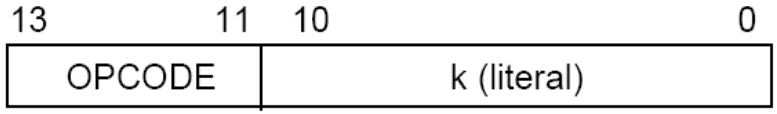
Literal and control operations

General



k = 8-bit immediate value

`CALL` and `GOTO` instructions only



k = 11-bit immediate value

El juego de instrucciones: SEGÚN SU FORMATO

1.- Instrucciones orientadas al byte

¿Qué hacen? W Opera con Operando en dir fuente

¿Cómo se escriben en ensamblador?

7 bits o algo que pueda sustituir a 7 bits, si es mayor se trunca

1 bit o algo que pueda sustituir a 1 bit, si es mayor se trunca

Operación fuente, destino

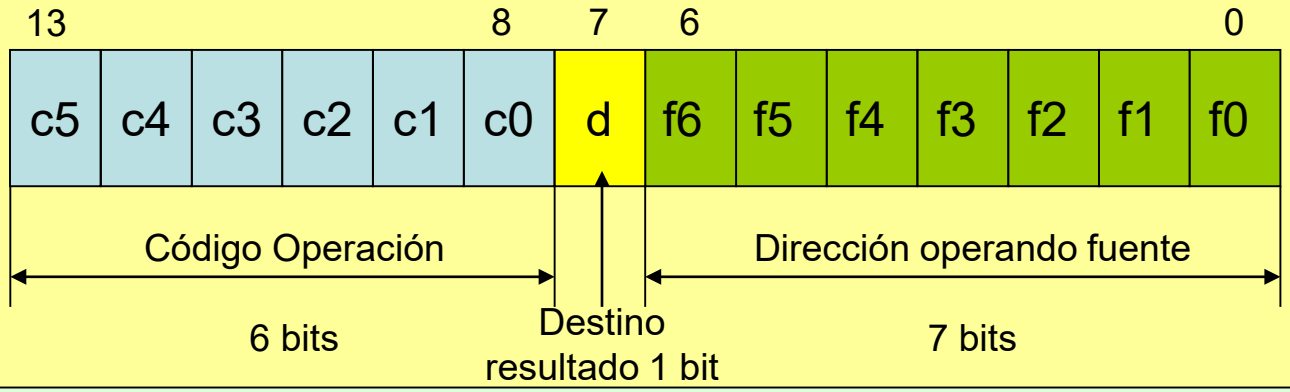
Nemónico reservado

d = 0 ó d = W → W

¿Dónde va a parar el resultado?

d = 1 ó d = f → Dirección fuente

¿Cómo se codifican?



El juego de instrucciones: SEGÚN SU FORMATO

2.- Instrucciones orientadas al bit

¿Qué hacen? Opera o explora el bit de la posición #bit del operando fuente

¿Cómo se escriben en ensamblador?

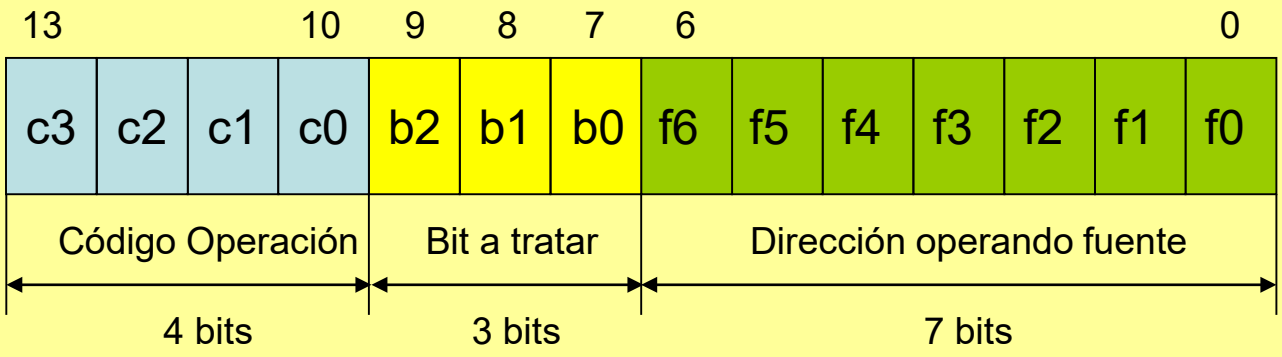
7 bits o algo que pueda sustituir a 7 bits, si es mayor se trunca

Operación fuente, bit

3 bits o algo que pueda sustituir a 3 bits, si es mayor se trunca

Nemónico reservado

¿Cómo se codifican?



El juego de instrucciones: SEGÚN SU FORMATO

3.- Instrucciones literales o de control

¿Qué hacen? Operan con el valor literal directamente

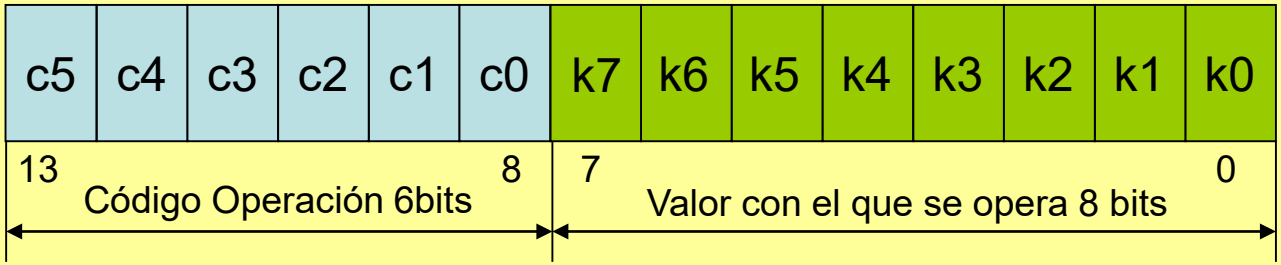
¿Cómo se escriben en ensamblador?

Operación valor literal

Nemónico reservado

8 bits o algo que pueda sustituir a 8 bits, si es mayor se trunca excepto en instrucciones del tipo GOTO ó CALL en que el valor es de 11 bits

¿Cómo se codifican?



En el caso GOTO y CALL

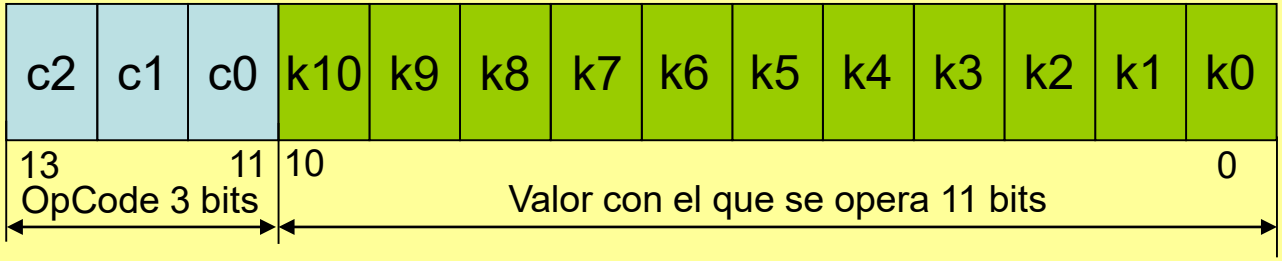


Tabla-resumen de instrucciones según su formato

Orientadas al byte (18)

Orientadas al bit (4)

Literales y de Control (13)

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb LSb	Status Affected	Notes
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
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
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff		3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff		3
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	

El juego de instrucciones: SEGÚN SU FUNCIONALIDAD

Desde el punto de vista de la funcionalidad tenemos:

- Instrucciones de carga de registros
- Instrucciones de ajuste de bits
- Instrucciones aritméticas
- Instrucciones lógicas
- Instrucciones de salto
- Instrucciones de manejo de subrutinas
- Instrucciones especiales

El juego de instrucciones: SEGÚN SU FUNCIONALIDAD

Instrucciones de CARGA (5)		
NEMÓNICO	DESCRIPCIÓN	FLAGS AFECTADOS
clrf f	00 → (f)	Z
clrw	00 → (W)	Z
movf f,d	(f) → (destino)	Z
movwf f	(W) → (f)	Ninguno
movlw k	k → (W)	Ninguno

El juego de instrucciones: SEGÚN SU FUNCIONALIDAD

Instrucciones de BIT (2)			
NEMÓNICO		DESCRIPCIÓN	FLAGS AFECTADOS
bcf	f,b	Pone a 0 el bit ‘b’ del registro ‘f’	Ninguno
bsf	f,b	Pone a 1 el bit ‘b’ del registro ‘f’	Ninguno

Instrucciones ARITMÉTICAS (6)			
NEMÓNICO		DESCRIPCIÓN	FLAGS AFECTADOS
addlw	k	(W) + k → (W)	C – DC - Z
addwf	f,d	(W) + (f) → (destino)	C – DC - Z
sublw	k	K - (W) → (W)	C – DC - Z
subwf	f,d	(f) - (W) → (destino)	C – DC - Z
decf	f,d	(f) - 1 → (destino)	Z
incf	f,d	(f) + 1 → (destino)	Z

El juego de instrucciones: SEGÚN SU FUNCIONALIDAD

Instrucciones LÓGICAS (10)			
NEMÓNICO		DESCRIPCIÓN	FLAGS AFECTADOS
andlw	k	(W) AND k → (W)	Z
andwf	f,d	(W) AND (f) → (destino)	Z
iorlw	k	(W) OR k → (W)	Z
iorwf	f,d	(W) OR (f) → (destino)	Z
xorlw	k	(W) XOR k → (W)	Z
xorwf	f,d	(W) XOR (f) → (destino)	Z
rlf	f,d	Rota (f) a izquierda → (destino)	C
rrf	f,d	Rota (f) a derecha → (destino)	C
comf	f,d	(/f) → (destino)	Z
swap	f,d	Intercambia nibbles (f) → (destino)	Ninguno

El juego de instrucciones: SEGÚN SU FUNCIONALIDAD

Instrucciones de SALTO (5)			
NEMÓNICO		DESCRIPCIÓN	FLAGS AFECTADOS
btfsc	f,b	Salta si el bit ‘b’ de ‘f’ es 0	Ninguno
btfss	f,b	Salta si el bit ‘b’ de ‘f’ es 1	Ninguno
decfsz	f,d	(f) - 1 → (destino) y salta si es 0	Ninguno
incfsz	f,d	(f) + 1 → (destino) y salta si es 0	Ninguno
goto	k	Salta a la dirección ‘k’	Ninguno

Instrucciones de manejo de SUBRUTINAS (4)			
NEMÓNICO		DESCRIPCIÓN	FLAGS AFECTADOS
call	k	Llamada a subrutina	Ninguno
return		Retorno de una subrutina	Ninguno
retfie		Retorno de una interrupción	Ninguno
retlw	k	Retorno con un literal en (W)	Ninguno

El juego de instrucciones: SEGÚN SU FUNCIONALIDAD

Instrucciones ESPECIALES (3)		
NEMÓNICO	DESCRIPCIÓN	FLAGS AFECTADOS
nop	No operación	Ninguno
sleep	Entra en modo de bajo consumo	/TO - /PD
clrwdt	Borra Timer del Watchdog	/TO - /PD

- Modo “sleep”. El consumo se reduce en varios órdenes de magnitud: sólo permanecen activas algunas partes del microcontrolador que permitirán “despertarlo”.

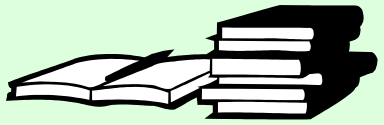
- Watchdog = “Perro guardián”. Si está activo, se genera una interrupción o RESET por hardware si pasa cierto número de ciclos sin borrarse el Timer del Watchdog

Campo	Descripción
f	Posición de memoria de datos (Register file address, desde 0x00 to 0x7F)
W	Registro de trabajo (acumulador)
b	nº Bit dentro de una posición de memoria (0-7)
k	Valor literal, constante o etiqueta (puede de 8 o 11 bits, según la instrucción)
x	No importa el valor (0 ó 1) El ensamblador genera código con x=0.
d	Selección de destino: d = 0: almacena el resultado en W, d = 1: almacena el resultado en una posición de la memoria de datos f.
dest	Indica si el destino es el registro W o la posición de memoria de datos especificada
label	nombre de etiqueta
TOS	Cima de la pila
PC	Contador de programa
PCLATH	Latch de la parte alta del contador de programa
GIE	Bit de habilitación de interrupción global
WDT	Temporizador Watchdog
TO	Time-out bit
PD	Power-down bit
[ ]	Optional
( )	Contenido
→	Asignado a
< >	Register bit field
∈	En el conjunto de
italics	termino definido por el usuario (courier)

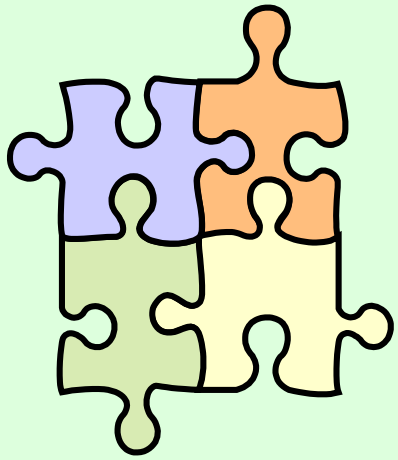
Las claves del juego de instrucciones

Más información

Fichero con el detalle de todas las instrucciones del juego de instrucciones.

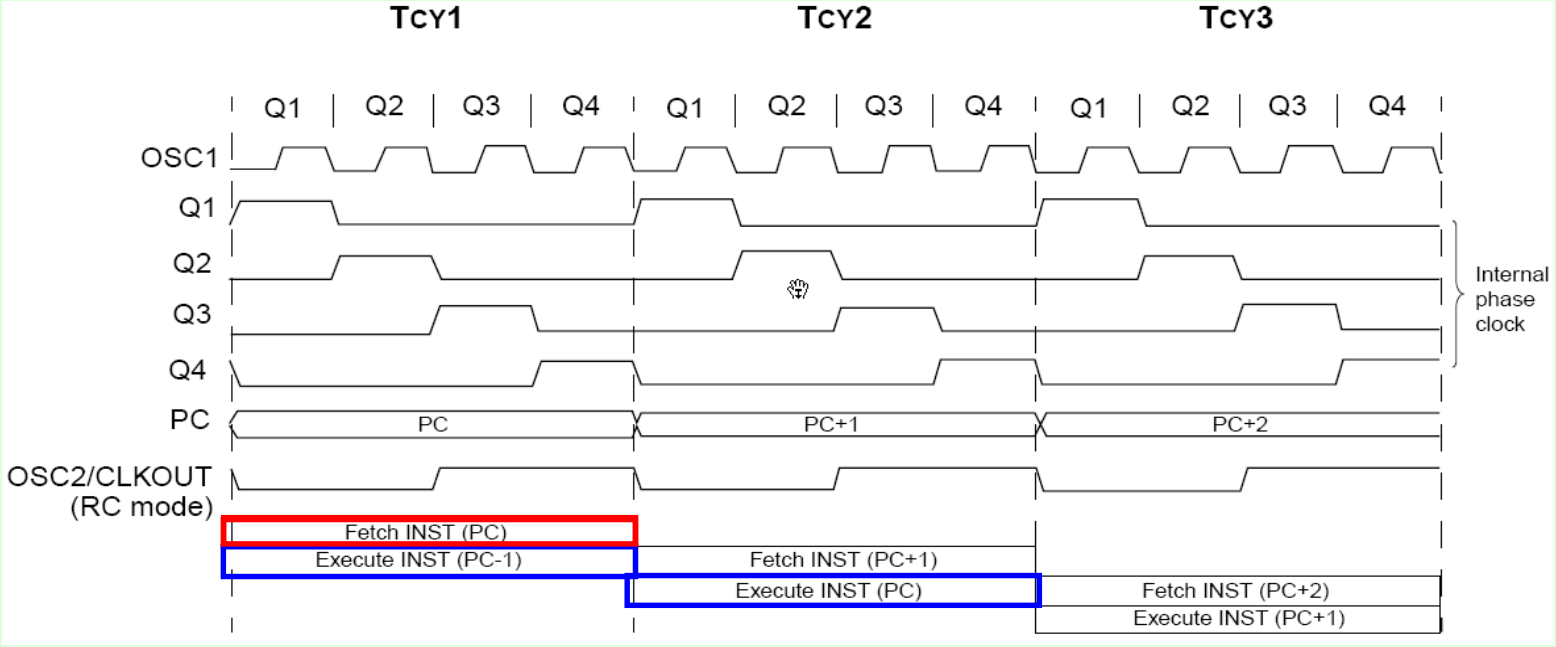


Juego_instrucciones_PIC.pdf



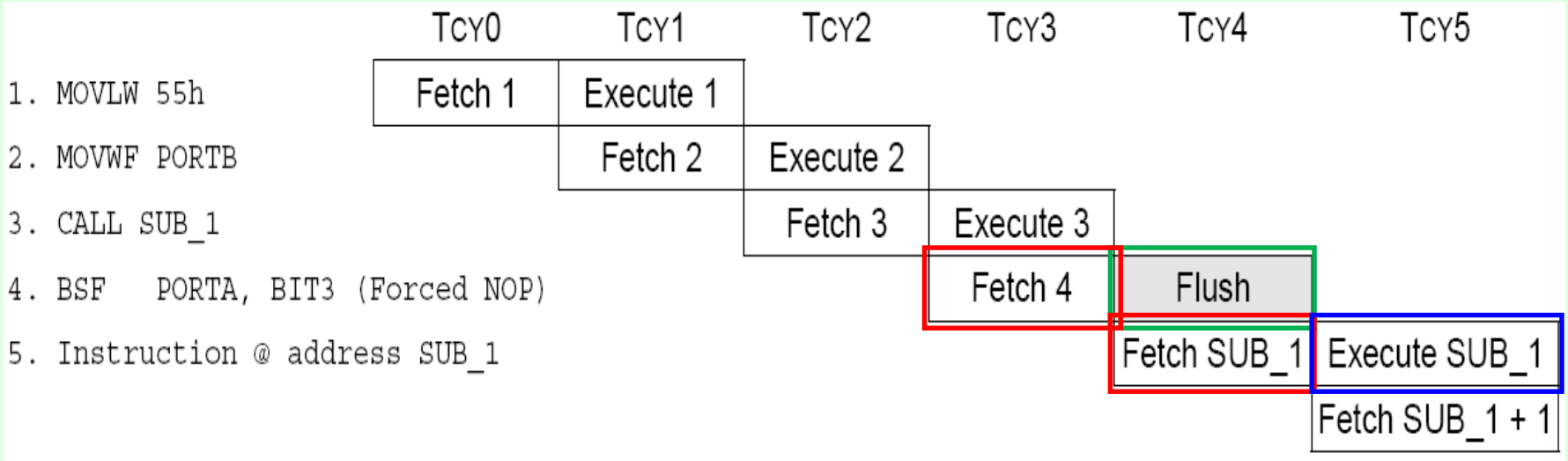
19.4. Ciclo de instrucción

- Ciclo de instrucción: es el tiempo que se tarda en ejecutar una instrucción, y es igual para todas las instrucciones del microcontrolador (salvo instrucciones de salto)
- En los PIC16xxx, un ciclo de instrucción dura 4 ciclos de reloj.
- Primera etapa: la instrucción se trae a la CPU: dura un ciclo de instrucción $T_{CY.}$
- Segunda etapa: se ejecuta la instrucción. Esto lleva otro ciclo $T_{CY.}$.
- No obstante, debido al solapamiento (*pipelining* ó entubado) de los procesos de traer la instrucción actual y ejecutar la instrucción previa, en cada T_{CY} se trae una instrucción y se ejecuta otra.



Ciclo de instrucción en instrucciones de salto

Si el resultado de ejecutar la instrucción anterior modifica el contenido del Contador de Programa (Ej: GOTO ó CALL) hay un ciclo de instrucción de retardo .
Se suspende el entubado (pipelining) de las instrucciones durante un ciclo para que la instrucción guardada en el punto de salto se traiga a la CPU.



Programación de microcontroladores PIC

Para poder programar una aplicación con el microcontrolador, es necesario definir y conocer:

1. Hardware a utilizar en la aplicación desarrollada:
   - Organización de la memoria interna (de programa y datos)
   - Dispositivos de E/S a utilizar y cómo gestionarlos

☑ 2.- Juego de Instrucciones

➡ 3.- Directivas y Macros

19.5. Directivas y macros

• Controlan el proceso de ensamblado del programa, pero no son parte del mismo (también se conocen como pseudoinstrucciones).

• Hay más de 50 directivas reconocidas por MPASM.

• Las más usadas :

END

Es la única directiva obligatoria. Indica al ensamblador dónde debe detener el proceso de ensamblado. Debe colocarse en la última línea del programa.

<etiqueta> **EQU** <expresión>

El valor <expresión> es asignado a <etiqueta>. Estas directivas se suelen colocar al principio del programa y habitualmente se usan para definir constantes y direcciones de memoria.

[<etiqueta>] **ORG** <expresión>

Las instrucciones del código fuente que siguen a esta directiva se ensamblan a partir de la posición indicada por <expresión>.

__CONFIG <expresión> [& <expresión> & ... & <expresión>]

Permite indicar la configuración elegida para la grabación del PIC

Ejemplo: __CONFIG _CP_OFF & _WDT_OFF & _XT_OSC

LIST P=16F877

Indica el tipo de microcontrolador utilizado.

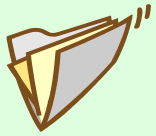
INCLUDE <p16F877.inc> o **INCLUDE** "p16F877.inc"

- Incluye en el programa un fichero donde se definen las etiquetas con las que se nombra a los diferentes registros y sus bits.
- Este fichero se encuentra en el directorio principal del programa ensamblador.
- Puede usarse esta directiva para incluir cualquier otro fichero. (¡Ojo! El fichero de inclusión no puede terminar con una directiva END).

DT <expr1> [, <expr2>, ... , <exprN>]

Genera una instrucción *retlw* por cada expresión que incluya ladirectiva. Si la expresión es del tipo cadena, se generará una instrucción *retlw* por cada carácter de la misma.

19.6. Información adicional



Ficheros adicionales:

Juego de instrucciones: Juego_instrucciones_PIC.pdf

Todas las directivas de MPASM: Directivas MPASM.pdf

Guía de uso del MPASM, MPLINK y MPLIB:

MPASM_Users_Guide_con_MPLINK_MPLIB.pdf