



Microcontroladores PIC

on-line **GRÁTIS!**

- ▶ [Índice](#)
- ▶ [Sistema de desenvolvimento](#)
- ▶ [Contacte-nos](#)



CAPÍTULO 3

Conjunto de Instruções

[Introdução](#)

[Conjunto de instruções da família PIC16Cxx de microcontroladores](#)

[Transferência de dados](#)

[Lógicas e aritméticas](#)

[Operações sobre bits](#)

[Direcção de execução do programa](#)

[Período de execução da instrução](#)

[Listagem das palavras](#)

Introdução

Já dissemos que um microcontrolador não é como qualquer outro circuito integrado. Quando saem da cadeia de produção, a maioria dos circuitos integrados, estão prontos para serem introduzidos nos dispositivos, o que não é o caso dos microcontroladores. Para que um microcontrolador cumpra a sua tarefa, nós temos que lhe dizer exactamente o que fazer, ou, por outras palavras, nós temos que escrever o programa que o microcontrolador vai executar. Neste capítulo iremos descrever as instruções que constituem o assembler, ou seja, a linguagem de baixo nível para os microcontroladores PIC.

Conjunto de Instruções da Família PIC16Cxx de Microcontroladores

O conjunto completo compreende 35 instruções e mostra-se na tabela que se segue. Uma razão para este pequeno número de instruções resulta principalmente do facto de estarmos a falar de um microcontrolador RISC cujas instruções foram optimizadas tendo em vista a rapidez de funcionamento, simplicidade de arquitectura e compacidade de código. O único inconveniente, é que o programador tem que dominar a técnica "desconfortável" de fazer o programa com apenas 35 instruções.

Transferência de dados

A transferência de dados num microcontrolador, ocorre entre o registo de trabalho (W) e um registo 'f' que representa um qualquer local de memória na RAM interna (quer se trate de um registo especial ou de um registo de uso genérico).

As primeiras três instruções (observe a tabela seguinte) referem-se à escrita de uma constante no registo W (MOVLW é uma abreviatura para MOVa Literal para W), à cópia de um dado do registo W na RAM e à cópia de um dado de um registo da RAM no registo W (ou nele próprio, caso em que apenas a flag do zero é afectada). A instrução CLRF escreve a constante 0 no registo 'f' e CLRW escreve a constante 0 no registo W. A instrução SWAPF troca o nibble (conjunto de 4 bits) mais significativo com o nibble menos significativo de um registo, passando o primeiro a ser o menos significativo e o outro o mais significativo do registo.

Lógicas e aritméticas

De todas as operações aritméticas possíveis, os microcontroladores PIC, tal como a grande maioria dos outros microcontroladores, apenas suportam a subtracção e a adição. Os bits ou flags C, DC e Z, são afectados conforme o resultado da adição ou da subtracção, com uma única excepção: uma vez que a subtracção é executada como uma adição com um número negativo, a flag C (Carry), comporta-se inversamente no que diz respeito à subtracção. Por outras palavras, é posta a '1' se a operação é possível e posta a '0' se um número maior tiver que ser subtraído de outro mais pequeno.

A lógica dentro do PIC tem a capacidade de executar as operações AND, OR, EX-OR, complemento (COMF) e rotações (RLF e RRF).

Estas últimas instruções, rodam o conteúdo do registo através desse registo e da flag C de uma casa para a esquerda (na direcção do bit 7), ou para a direita (na direcção do bit 0). O bit que sai do registo é escrito na flag C e o conteúdo anterior desta flag, é escrito no bit situado do lado oposto no registo.

Operações sobre bits

As instruções BCF e BSF põem a '0' ou a '1' qualquer bit de qualquer sítio da memória. Apesar de parecer uma operação simples, ela é executada do seguinte modo, o CPU primeiro lê o byte completo, altera o valor de um bit e, a seguir, escreve o byte completo no mesmo sítio.

Direcção de execução de um programa

As instruções GOTO, CALL e RETURN são executadas do mesmo modo que em todos os outros microcontroladores, a diferença é que a pilha é independente da RAM interna e é limitada a oito níveis. A instrução 'RETLW k' é idêntica à instrução RETURN, excepto que, ao regressar de um subprograma, é escrita no registo W uma constante definida pelo operando da instrução. Esta instrução, permite-nos implementar facilmente listagens (também chamadas tabelas de lookup). A maior parte das vezes, usamo-las determinando a posição do dado na nossa tabela adicionando-a ao endereço em que a tabela começa e, então, é lido o dado nesse local (que está situado normalmente na memória de programa).

A tabela pode apresentar-se como um subprograma que consiste numa série de instruções 'RETLW k' onde as constantes 'k', são membros da tabela.

```

Main    movlw 2
        call Lookup
Lookup  addwf PCL, f
        retlw k
        retlw k1
        retlw k2
        :
        :
        retlw kn

```

Nós escrevemos a posição de um membro da nossa tabela no registo W e, usando a instrução CALL, nós chamamos o subprograma que contém a tabela. A primeira linha do subprograma 'ADDWF PCL, f', adiciona a posição na tabela e que está escrita em W, ao endereço do início da tabela e que está no registo PCL, assim, nós obtemos o endereço real do dado da tabela na memória de programa. Quando regressamos do subprograma, nós vamos ter no registo W o conteúdo do membro da tabela endereçado. No exemplo anterior, a constante 'k2' estará no registo W, após o retorno do subprograma.

RETFIE (RETurn From Interrupt – Interrupt Enable ou regresso da rotina de interrupção com as interrupções habilitadas) é um regresso da rotina de interrupção e difere de RETURN apenas em que, automaticamente, põe a '1' o bit GIE (habilitação global das interrupções). Quando a interrupção começa, este bit é automaticamente reposto a '0'. Também quando a interrupção tem início, somente o valor do contador de programa é posto no cimo da pilha. Não é fornecida uma capacidade automática de armazenamento do registo de estado.

Os saltos condicionais estão sintetizados em duas instruções: BTFSC e BTFSS. Consoante o estado lógico do bit do registo 'f' que está a ser testado, a instrução seguinte no programa é ou não executada.

Período de execução da instrução

Todas as instruções são executadas num único ciclo, excepto as instruções de ramificação condicional se a condição for verdadeira, ou se o conteúdo do contador de programa for alterado pela instrução. Nestes casos, a execução requer dois ciclos de instrução e o segundo ciclo é executado como sendo um NOP (Nenhuma Operação). Quatro oscilações de clock perfazem um ciclo de instrução. Se estivermos a usar um oscilador com 4MHz de frequência, o tempo normal de execução de uma instrução será de 1µs e, no caso de uma ramificação condicional de 2µs.

Listagem das palavras

f qualquer local de memória num microcontrolador

W registo de trabalho

b posição de bit no registo 'f'

d registo de destino

label grupo de oito caracteres que marca o início de uma parte do programa (rótulo)

TOS cimo da pilha

[] opcional

<> grupo de bits num registo

Menemónica		Descrição		Flag	CLK	Notas
Transferência de dados						
MOVLW	k	Mova literal para W	$k \rightarrow W$		1	
MOVWF	f	Mova W para f	$W \rightarrow f$		1	
MOVF	f, d	Mova f	$f \rightarrow d$	Z	1	1, 2
CLRW	-	Clear W (limpar W)	$0 \rightarrow W$	Z	1	
CLRF	f	Clear f (limpar f)	$0 \rightarrow f$	Z	1	2
SWAPF	f, d	Swap nibbles in f (trocar)	$f(7:4), (3:0) \rightarrow f(3:0), (7:4)$		1	1, 2
Lógicas e Aritméticas						
ADDLW	k	Adicionar literal a W	$W+k \rightarrow W$	C, DC, Z	1	
ADDWF	f, d	Adicionar W a f	$W+f \rightarrow d$	C, DC, Z	1	1, 2
SUBLW	k	Subtrair W de literal	$k-W \rightarrow W$	C, DC, Z	1	
SUBWF	f, d	Subtrair W de f	$f-W \rightarrow d$	C, DC, Z	1	1, 2
ANDLW	k	AND literal com W	$W \text{ .AND. } k \rightarrow W$	Z	1	
ANDWF	f, d	AND W com f	$W \text{ .AND. } f \rightarrow d$	Z	1	1, 2
IORLW	k	Inclusivo OR de literal com W	$W \text{ .OR. } k \rightarrow W$	Z	1	
IORWF	f, d	Inclusivo OR de W com f	$W \text{ .OR. } f \rightarrow d$	Z	1	1, 2
XORWF	f, d	Exclusivo OR de W com f	$W \text{ .XOR. } f \rightarrow d$	Z	1	1, 2
XORLW	k	Exclusivo OR de literal com W	$W \text{ .XOR. } k \rightarrow W$	Z	1	
INCF	f, d	Incrementar f	$f+1 \rightarrow f$	Z	1	1, 2
DECF	f, d	Decrementar f	$f-1 \rightarrow f$	Z	1	1, 2
RLF	f, d	Rode f p/ esquerda com o carry	$\left[\begin{array}{cccccccc} C & 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ \leftarrow & & & & & & & & \end{array} \right]$	C	1	1, 2
RRF	f, d	Rode f p/ a direita com o carry	$\left[\begin{array}{cccccccc} 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\ & & & & & & & C \end{array} \right]$	C	1	1, 2
COMF	f, d	Complementar f	$f \rightarrow d$	Z	1	1, 2
Operações sobre bits						
BCF	f, b	Bit Clear f (bit de f a '0')	$0 \rightarrow f(b)$		1	1, 2
BSF	f, b	Bit Set f (bit de f a '1')	$1 \rightarrow f(b)$		1	1, 2
Direcção do programa						
BTFSZ	f, b	Bit Test f, Salte se Clear('0')	salte se $f(b)=0$		1(2)	3
BTSSZ	f, b	Bit Test f, Salte se Set('1')	salte se $f(b)=1$		1(2)	3
DECFSZ	f, d	Decrementa f, salte se der 0	$f-1 \rightarrow d$, salte se der 0		1(2)	1, 2, 3
INCFSZ	f, d	Incrementa f, salte se der 0	$f+1 \rightarrow d$, salte se der 0		1(2)	1, 2, 3
GOTO	k	Go to address(Ir p/ endereço)	$k \rightarrow PC$		2	
CALL	k	Chamar subrotina	$PC \rightarrow TOS, k \rightarrow PC$		2	
RETURN	-	Retorno de subrotina	$TOS \rightarrow PC$		2	
RETLW	k	Retorno com literal em W	$k \rightarrow W, TOS \rightarrow PC$		2	
RETFIE	-	Retorno de interrupção	$TOS \rightarrow PC, 1 \rightarrow GIE$		2	
Outras instruções						
NOP	-	Nenhuma operação			1	
CLRWDI	-	Temporizador do Watchdog=0	$0 \rightarrow WDT, 1 \rightarrow TO, 1 \rightarrow PD$	$\overline{TO}, \overline{PD}$	1	
SLEEP	-	Entrar no modo 'sleep'	$0 \rightarrow WDT, 1 \rightarrow TO, 0 \rightarrow PD$	$\overline{TO}, \overline{PD}$	1	

*1 Se o porto de entrada/saída for o operando origem, é lido o estado dos pinos do microcontrolador.

*2 Se esta instrução for executada no registo TMR0 e se $d=1$, o prescaler atribuído a esse temporizador é automaticamente limpo.

*3 Se o PC for modificado ou se resultado do teste for verdadeiro, a instrução é executada em dois ciclos.