4 - Conjunto de Instruções do PIC

O microcontrolador PIC16F628A apresenta em seu nível ISA 35 instruções, A maioria das instruções são executadas em apenas um ciclo de máquina, lembrando que o um ciclo de máquina, corresponde ao clock de entrada divido por 4. O Datasheet do PIC16F628A, divide o conjunto de instruções do PIC em três tipos:

- **Orientadas a bit (Bit-Oriented):** Operam com bits de um determinado registador;
- **Orientadas a byte (Byte-Oriented):** Operam com registradores completos (bytes);
- **Literal ou controle (Literal and control):** Instruções que usam literais (números ou endereços) como operando.

4.1 Formato do conjunto de instruções

Cada instrução do PIC16F628A tem o tamanho de 14 bits, onde alguns destes bits são usados para especificar o OPCODE (nome dado ao "comando" assembly), e de zero até dois operandos. Por conveniência, neste material a ultima categoria foi dividida em literal e controle.

O formato das instruções pode variar de acordo com o tipo, abaixo o formato da instrução do PIC para cada tipo. A figura abaixo mostra os formatos de instrução do PIC16F628A.

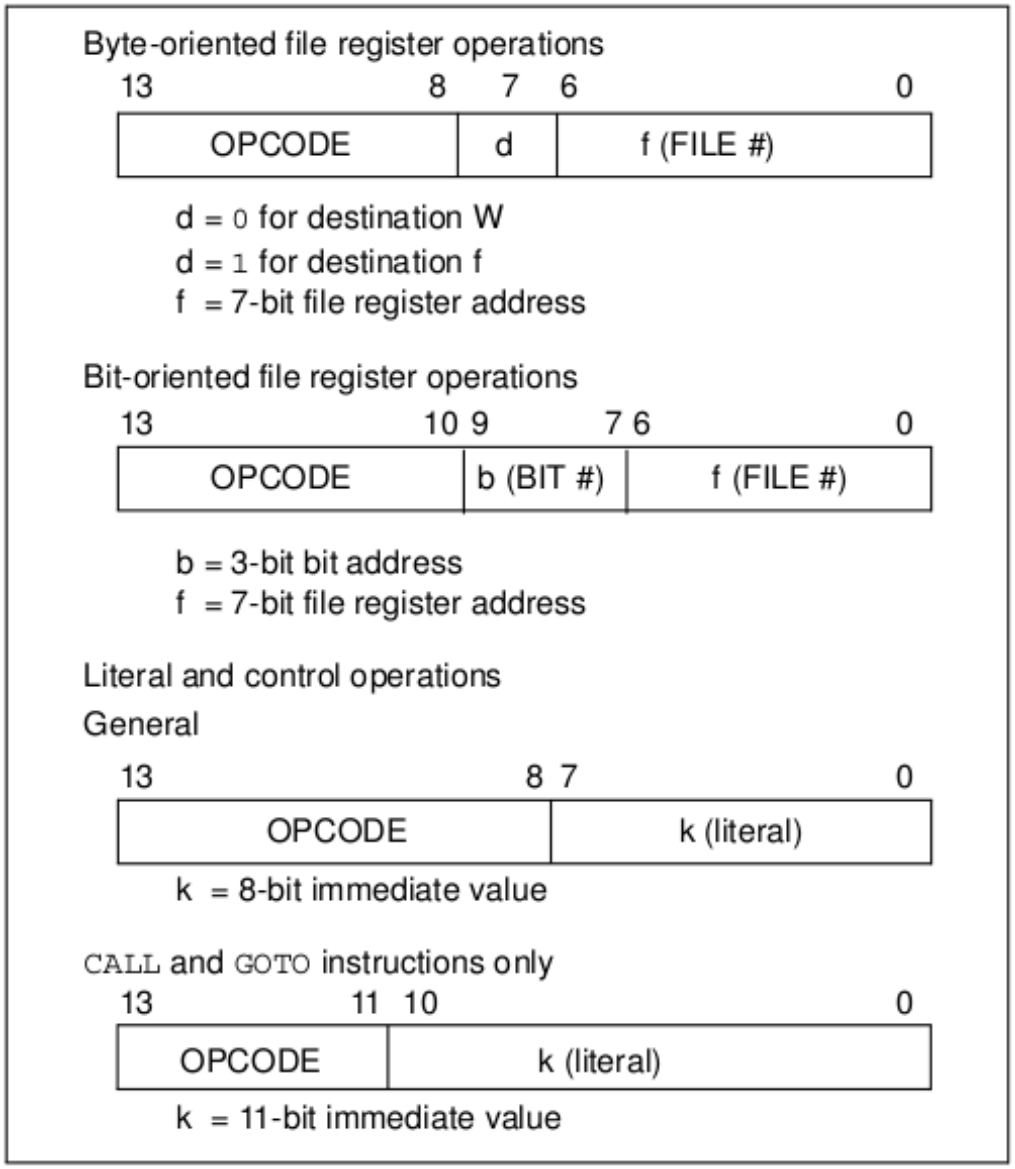


Figura 15-1 do Datasheet

As tabelas seguintes apresentam todas as instruções do PIC16F628A, com os seus respectivos operandos.

Orientadas a bit

			13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mnemônico	ciclos	Descrição	OPCODE				BIT (0-7)			REGISTRADOR (00h - 127h)						
BCF f,b	1	Limpa (zera) o bit b do registrador f	0	1	0	0	B	B	B	F	F	F	F	F	F	F
BSF f,b	1	Seta (iguala a 1) o bit b do registrador f	0	1	0	1	B	B	B	F	F	F	F	F	F	F
BTFSC f,b	1(2)	Testa o Bit b de f, pula se zero(clear)	0	1	1	0	B	B	B	F	F	F	F	F	F	F
BTFSS f,b	1(2)	Testa o Bit b de f, pula se um(set)	0	1	1	1	B	B	B	F	F	F	F	F	F	F

Orientadas a byte

			13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mnemônico	ciclos	Descrição	OPCODE						D	REGISTRADOR (00h - 127h)						
NOP	1	Gasta um ciclo	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CLRWDT	1	Zera o timer *watch dog*	0	0	0	0	0	0	0	1	1	0	0	1	0	0
MOVWF f	1	Copia o conteudo de w em f	0	0	0	0	0	0	1	F	F	F	F	F	F	F
CRLW	1	Zera o conteudo do resgistrador W	0	0	0	0	0	1	0	X	X	X	X	X	X	X
CLRF f	1	Zera o conteudo de f	0	0	0	0	0	1	1	F	F	F	F	F	F	F
SUBWF f,d	1	subtrai W de f e armazena em d(d <= f − W)	0	0	0	0	1	0	D	F	F	F	F	F	F	F
DECF f,d	1	decrementa f e armazena em d(d <= f − 1)	0	0	0	0	1	1	D	F	F	F	F	F	F	F
IORWF f,d	1	OU normal (Inclusivo) de W com F (d <= f OR W)	0	0	0	1	0	0	D	F	F	F	F	F	F	F
ANDWF f,d	1	E entre W e F (d <= f AND W)	0	0	0	1	0	1	D	F	F	F	F	F	F	F
XORWF f,d	1	OU exclusivo entre W e F (d <= f XOR W)	0	0	0	1	1	0	D	F	F	F	F	F	F	F
ADDWF f,d	1	Adiciona W com F (d <= f + W)	0	0	0	1	1	1	D	F	F	F	F	F	F	F
MOVF f,d	1	Move F (d <= f)	0	0	1	0	0	0	D	F	F	F	F	F	F	F
COMF f,d	1	Complemento de f (d <= NOT f)	0	0	1	0	0	1	D	F	F	F	F	F	F	F
INCF f,d	1	incrementa f e armazena em d(d <= f − 1)	0	0	1	0	1	0	D	F	F	F	F	F	F	F
DECFSZ f,d	1(2)	Decrementa f (d <= f − 1) e salta se zero	0	0	1	0	1	1	D	F	F	F	F	F	F	F
RRF f,d	1	Rotaciona F para direita com carry out	0	0	1	1	0	0	D	F	F	F	F	F	F	F
RLF f,d	1	Rotaciona F para esquerda com carry out	0	0	1	1	0	1	D	F	F	F	F	F	F	F
SWAPF f,d	1	Troca os nibbles mais e menos significativos de f	0	0	1	1	1	0	D	F	F	F	F	F	F	F
INCFSZ f,d	1(2)	incrementa f (d <= f − 1) e salta se zero	0	0	1	1	1	1	D	F	F	F	F	F	F	F

De controle (quando efetuam desvios gastam 2 ciclos)

			13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mnemônico	ciclos	Descrição	OPCODE						K literal							
RETURN	2	Retorna de uma subrotina chamada por CALL	0	0	0	0	0	0	0	0	0	0	1	0	0	0
RETFIE	2	Retorna de uma interrupção	0	0	0	0	0	0	0	0	0	0	1	0	0	1
SLEEP	0	Põe o controlador em stand-by	0	0	0	0	0	0	0	1	1	0	0	0	1	1
Mnemônico	ciclos	Descrição	OPCODE			K posição de memória										
CALL k	2	Salva PC+1 na pilha e faz PC=k (salta a execução para o endereço k)	1	1	0	K	K	K	K	K	K	K	K	K	K	K
GOTO k	2	Pula para o endereço k (11 bits) usa 2 ciclos	1	0	1	K	K	K	K	K	K	K	K	K	K	K

Operações com literais

			13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mnemônico	ciclos	Descrição	OPCODE						K literal							
MOVLW k	1	Move literal to W (W = k)	1	1	0	0	X	X	K	K	K	K	K	K	K	K
RETLW K	1	faz W=k e retorna de uma subrotina	1	1	0	1	X	X	K	K	K	K	K	K	K	K
IORLW k	1	W = k OR W	1	1	1	0	0	0	K	K	K	K	K	K	K	K
ANDLW k	1	W = k AND W	1	1	1	0	0	1	K	K	K	K	K	K	K	K
XORLW k	1	W = k AND W	1	1	1	0	1	0	K	K	K	K	K	K	K	K
SUBLW k	1	W = k - W	1	1	1	1	0	X	K	K	K	K	K	K	K	K
ADDLW k	1	W = k + W	1	1	1	1	1	X	K	K	K	K	K	K	K	K

```
Legenda:
X = não importa o valor do bit
k = o bit pertence a um literal
f = O bit pertence a um endereço de registrador
d = onde o resultado será armazenado W(d=0) ou F(d=1)
```

4.2 Mnemônicos

Uma instrução do PIC é normalmente da seguinte forma:

```
INST OP1,OP2
```

Onde INST é o OPCODE da instrução e OP1 e OP2 são os operandos. Uma instrução pode ter zero, um ou dois operandos. Um operando pode ser um Literal (k), um registrador (f), bit (b onde $0 \leq b \leq 7$) ou um destino (d), onde d=0 work e d=1 file;

O nome das instruções do PIC são construídos de acordo com certa lógica. Entender essa lógica pode facilitar muito o trabalho do programador, pois com esse entendimento é mais fácil decorar o set de instruções do PIC e os seus operandos, e diminuir eventuais consultas a documentação na hora de ler um código. Os termos abaixo são usados na construção dos nomes de instrução:

(W)ork – Refere-se ao registrador temporário W. Observando o esquema interno do PIC é possível perceber que esse registrador está localizado na saída da ULA. Quase todas as operações sobre bytes, de alguma forma, envolvem esse registrador. As instruções que manipulam esse registrador diretamente tem o W na sua formação.

(L)iteral – Refere-se a um valor numérico. Pode ser escrito na forma binária, decimal ou hexadecimal. As instruções que operam com literais apresentam o L na sua formação.

(F)ile – Refere-se a uma posição de memória (File Register). As instruções que operam diretamente com registrador apresentam um F na sua formação.

Ex.: Para fazer uma atribuição do tipo:

```
A=25
```

É necessário carregar W com o valor do literal 25 e depois carregar o conteúdo do registrador W na posição representada por A.

```
A equ H'007'
...
MOVLW d'25'             ; Move o (L)iteral (25 no caso) para (W)ork
MOVWF A                 ; move (W)ork para (F)ile (A no caso)
```

Observe que as referencias a W, são implícitas, ou seja, não aparecem nos operandos.

(B)it – Refere-se a um bit específico do registrador F, as instruções que operam diretamente com bit tem um B na sua formação.

(S)et/(C)lear – Refere-se a ação de por em nível lógico um (set) ou zero (clear) um determinado registrador (neste caso o pic só permite zerar um registrador específico) ou bit específico do registrador. As instruções que setam um determinado bit apresentam S na sua formação.

São exemplos:

```
BCF 0x5h,3      ; Bit Clear File - zera Bit 3 de 0x5h
BSF 0x5h,3      ; Bit Set File- Faz o bit 3 de 0x5h igual a 1
```

(T)est – Refere-se ao ato de testar uma determinada condição que está associada a um um bit específico do registrador F. As instruções que fazem testes tem o T na sua formação.

(S)kip – Refere-se ao ato de promover algum desvio no fluxo sequencia de instrução com base em alguma condição. A instruções que realizam saltos tem um S na sua formação.

OBS.: Para desvios incondicionais normais existem as instruções GOTO e CALL.

(Z)ero – Refere-se ao teste se o resultado de uma operação especifica é zero, normalmente verificando o registrador de STATUS (03h,2). As instruções que fazem esse tipo de teste apresentam Z na sua formação. Essas instruções normalmente trabalham com bytes.

Exemplos:

```
BTFSC 0x3h,2    ;Bit Test File Skip Clear - se bit 2 de 3h iguala a zero
                        ;salta a próxima instrução
DECFSZ  0x7h    ;Decrementa (DEC) File Skip Zero - se o resultado do
                        ;decremento for zero salta a próxima instrução
```

Combinando os termos acima com um dos termos seguir com os termos acima formam-se as instruções do PIC que trabalham com bytes:

- ADD – Adiciona dois operandos que podem ser W, F ou L. O resultado sempre será armazenado em W ou F
- AND – Faz o E lógico entre dois operandos que podem ser W, F ou L. O resultado sempre será armazenado em W ou F
- CLR – torna zero Work ou um determinado registrador F
- COM – Complemento de registrador F

- DEC – Decrementa
- INC - Incrementa
- IOR – OU inclusivo
- MOV – Move um dado ou literal
- RL – Rotaciona para a esquerda (x2)
- RR – Rotaciona para a direita (x2)
- SUB - Subtração
- SWAP – Troca interna entre o bits mais ou menos significativos
- XOR – OU exclusivo

4.3 Hello world em PIC

Agora que já conhecemos, ainda que superficialmente, o funcionamento do PIC e o seu conjunto de instruções, podemos implementar um programa. O programa hello world é usado em todas as literaturas sobre linguagens de programação para descrever de maneira simplória um programa numa linguagem qualquer. Aqui será feito um programa que faça um acender.

```
ORG 0x00                ;Vetor de reset
GOTO INICIO

INICIO

    BSF  3H,5           ;Liga o Bit 5 (RP0) do registrador 3h (STATUS); Seleciona banco
                        ;de memória 1. TRISA e TRISB estão no banco 1
    MOVLW b'0000000'    ;Move zero para o registrador W (o mesmo que CRLW)
    MOVWF 86H           ; Move W para o registrador 86H (TRISB) ou Seja TRISB=0
                        ;Todos pinos do PORTB estão configurados com output
    BCF  3H,5           ;Desliga o Bit 5 (RP0) do registrador 3h (STATUS)  seleciona o
                        ;banco de memória 0

MAIN

    BSF 6h,2            ;Liga o pino 0 do PORTB (RB2)
    GOTO MAIN

END
```

Listagem 1: Hello World

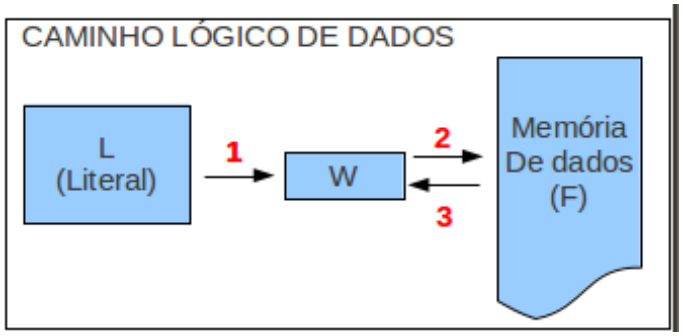
Laboratório: Para veficiar o funcionamento di código acima através de simulação, favor execute o laboratório 1

A seguir um breve comentário sobre as principais instruções exibidas na tabela acima

4.4 INSTRUÇÕES ORIENTADAS A BYTE

Movimentação de dados (MOVLW, MOVWF, MOVF)

O fluxo de movimentação dos dados pode ser mapeado conforme a figura abaixo:



O movimento 1 é executado pela instrução MOVLW *LITERAL*. Essa instrução preenche o registrador W com o valor informado em *Literal* requer apenas um operando que é justamente o literal que será movimentado, se observamos o formato da instrução MOVLW na tabela no inicio do capítulo veremos que 8 bits estão disponíveis para representar esse literal, desta forma o maior literal que pode ser armazenado em W é 255.

O movimento 2 é feito pela instrução MOVWF *END*, essa instrução copiará no endereço informado em END o valor armazendo em W. Observando a tabela no inicio do capítulo pode-se verificar que existe apenas 7 bits disponíveis para informar endereço, o que torna a instrução capaz de endereçar apenas 128 posições (0-127), entretando o intevalo de endereçamento de dados do pic vai até o endereço 511, como foi mostrado no capitulo anterior. Para resolver esse problema a memória do PIC foi dividida em bancos que podem ser selecionados usando os bits 6 e 5 do registrador status (endereço 3h)

O movimento 3 é feito pela instrução MOVF *END,DST*. Essa instrução recebe 2 operandos *END* e *DST*, onde END é o endereço do dado que será movimentado e DST é o destino da movimentação. Se DST=0 a movimentação será feita para registrador W. Se DST=1 a movimentação será para o mesmo endereço apontado por END, ou seja copia o valor para ele mesmo. A questão que pode surgir é para que serve fazer uma cópia para ele mesmo? E a resposta é para verificar se um valor armazenado em uma posição de memória é igual 0. Mais sobre isso na seção dicas de programação.

Resumindo é o seguinte: Todo valor literal para chegar a algum registrador (posição de memória) tem que necessariamente passar pelo acumulador W. Sendo assim para atribuir algum valor a algum registrador primeiro precisamos movê-lo para o acumulador W e em seguida mover o conteúdo de W para o registrador em questão. Ex:

```
MOVLW d'45' ; literal para W
MOVWF 85h ; W para (F) memória
```

Já para mover o conteúdo de um registrador (EX: 05h->06h) para outro existe a função MOVF:

```
MOVF 05h,0 ; F para W (indicado pelo 0 no segundo operando)
MOVWF 06h
```

Limpar registradores (CLRW, CLRF)

Uma questão que é sempre necessária em qualquer programa é a inicialização de variáveis com o valor 0. Desta forma para se zerar um determinado registador podemos usar as seguintes instruções:

```
MOVLW 0x00
MOVWF 05h
```

A instrução CLRF pode zerar um registrador em apenas um ciclo, economizando com isso posições, na escassa memória de programação.

```
CRLF 05h
```

A instrução CLRW, é usada para zerar o registrador W, neste caso não tem vantagem, em termos de ciclos, em relação a instrução MOVLW. No entanto as instruções CRLF e CRLW ajustam o bit Z no registrador STATUS, mantendo a consistência, diferentemente das instruções de movimentação.

Lógicas (ANDWF, ANDLW, IORWF, IORLW, XORWF, XORLW, COMF)

Aritméticas (ADDWF, ADDLW, SUBWF, SUBLW, INCF, DECF)

4.5 INSTRUÇÕES ORIENTADAS A BIT

As instruções orientadas da BIT permitem operar com bits dentro de um determinado registrador.

BSF, BCF

A instrução BSF (*Bit Set File*) permite setar (nível lógico 1) um determindo bit de um registrador. O exemplo a seguir seta o bit 3 do registrador 86H:

```
BSF 86h,3;
```

Já instrução BCF (*Bit Clear File*) permite limpar (nível lógico 0) um determindo bit de um registrador. O exemplo a seguir limpa o bit 7 do registrador 85H:

```
BCF 85h,7;
```

RRF, RLF

As instruções RRF e RLF permitem rotacionar (deslocar a cadeia de bits) os bits de um determinado registrador e escolher se o resultado será armazenado em W (d=0) ou no próprio registrador (d=1). Existem vária razões para se rotacionar um conjunto de bits, dentre elas podemos citar a multiplicação ou divisão por 2, ou ainda conseguir o efeito luminoso de sequencial na saída do PIC.

A instrução RRF (*Rotate Right File*) permite rotacionar o registrador um bit para DIREITA.

Por exemplo imagine o registrador 07H cujo o valor armazenado é 16 (em Hexa é 10H e em binário é 00010000) a instrução:

```
RRF 07H,1
```

Irá fazer o conteúdo de 07H igual á 32 (em Hexa é 20H e em binário é 00100000). Por outro lado a instrução instrução RLF (*Rotate Left File*) rotacionará o registrador para esquerda. Aplicando essa instrução ao mesmo registrador (com valor 16) do exemplo anterior:

```
RLF 07H,1
```

Irá fazer o conteúdo de 07H igual á 8 (em Hexa é 8H e em binário é 00001000).

SWAPF

A instrução SWAPF troca os nibbles(sequencia de 4 bits) alto e baixo de um registrador. Por exemplo imagine que o conteúdo do registrador 08H seja FEh (em decimal é 254 e em binário é 11111110):

```
SWAPF 08H,1
```

Após a execução da instrução acima o conteúdo do registrador 08H será EFh (em decimal é 239 e em binário é

11101111).

4.6 INSTRUÇÕES DE CONTROLE

O Modelo de execução de qualquer computador prevê que o fluxo de execução das instruções é sequencial. Em alguns casos é necessário desviar o fluxo normal de execução para algum outro ponto do programa. Existem três tipos de situações que podem levar a um desvio de fluxo: Desvios (condicional ou não), Chamada de Procedimento (CALL) e uma interrupção de hardware. O conjunto de instruções do PIC oferece algumas alternativas para programar esse desvio.

4.6.1 Desvios condicionais

Um desvio condicional muda o fluxo de execução do programa, se alguma condição for verdadeira. As instruções de desvio condicional do PIC, não permitem a especificação de um endereço especifico para o salto, o que ocorrem é um salto da próxima instrução caso a condição seja verdadeira. Instruções do PIC que se encaixam nesta categoria estão na tabela abaixo.

Mnemônico	operandos	Limites	Descrição
INCFSZ	f,d	$0 \leq f \leq 127$ $0 \leq d \leq 1$	Incrementa o conteúdo de F, armazena o resultado em W (d=0) e em F se d=1, salta se o resultado da operação for ZERO.
DECFSZ	f,d	$0 \leq f \leq 127$ $0 \leq d \leq 1$	Decrementa o conteúdo de F, armazena o resultado em W (d=0) e em F se d=1, salta se o resultado da operação for ZERO.
BTFSC	f,b	$0 \leq f \leq 127$ $0 \leq b \leq 7$	Salta se o bit b do registrador f estiver em nível lógico 1.
BTFSS	f,b	$0 \leq f \leq 127$ $0 \leq b \leq 7$	Salta a instrução se o bit b do registrador f estiver em nível lógico 1.

Essas instruções normalmente são usadas em conjunto com a instrução GOTO. Vejamos alguns exemplos.

Comparação

A comparação é o principal recurso usado pelo programador para tomar decisões sobre a execução de um determinado bloco de código. Qualquer linguagem deve fornecer a possibilidade de realizar comparações. Abaixo um exemplo de uma comparação:

```
SE A=B
        PORTA=8
ELSE
        PORTA=24
```

Observe o conjunto de instruções do PÌC não fornece diretamente uma instrução de comparação. Em geral as comparações são feitas usando subtrações (SUB), nestes casos é feita a subtração entre dois operandos e observado os bits Z e C do registrador STATUS. Por exemplo:

```
SUBLW D'15' ; W = 15-W
```

Se W=15 então Z (STATUS,2) = 1

Se W<15 então Z (STATUS,2) = 0 e C (STATUS,0)=0

Se W>15 então Z (STATUS,2) = 0 e C (STATUS,0)=1

Na implementação Abaixo estamos comparando o conteúdo dos registadores 0x07(A) e 0x08(B). Se o conteúdo destes registradores apresentarem valores iguais, liga-se o Bit RA3, caso contrário liga RA3 e RA4. Observe que nos exemplos , os conteúdos das posições A e B não foram definidos.

```
STATUS  EQU     H'003'
PORTA           EQU     H'005'
A               EQU     H'020'
B               EQU     H'021'
W               EQU     D'0'

MOVF A,W                                ; Move o valor de A para W
SUBWF B,W                               ; Subtrai B d W e armazena o resultado em W
BTFSS STATUS,2                  ; Verifica se Z(STATUS,2) igual 1 (A igual a B)
GOTO DIFERENTE                  ; Salta para o rotulo DIFERENTE
GOTO IGUAL                              ; Salta para o rotulo IGUAL

IGUAL
        MOWLW D'8'                      ; Move 8 para W
        MOVWF PORTA                     ; Move W(8) para PORTA
        GOTO FIM                        ; Pula para o rótulo FIM
DIFERENTE
        MOWLW D'24'                     ; Move 8 para W
        MOVWF PORTA                     ; Move W(24) para PORTA
FIM
        END                             ; Fim do programa
```

Listagem 2: Comparação

Laço

Os laços permitem a execução de um mesmo bloco de código por várias vezes seguidas, a seguir um exemplo em uma pseudolinguagem:

```
PARA I=10 ATÉ 0 FAÇA:
       VAR=VAR+1
```

Na maioria das arquiteturas a Implementação de um laço se dá com auxílio de saltos, no caso do PIC16F628A será usado a instrução goto, a listagem a seguir mostra uma possibilidade para implementar o laço acima:

```
VAR     EQU     H'007'
I       EQU     H'008'
F       EQU     D'1'

MOVLW D'10'
MOVWF VAR
LOOP
        INCF VAR
        DECFSZ I,F
        GOTO LOOP
END
```

Listagem 3: Laço

Desvios Incondicionais

O desvio incondicional é uma mudança no fluxo de execução independente de qualquer teste. As seguintes instruções se encaixam neste tipo:

Mnemônico	Operando	Limites	Descrição
GOTO	k	$0 \leq k \leq 255$	Desvia o Fluxo do programa para a posição k. No programa a k é substituído por um rótulo.
CALL	k	$0 \leq k \leq 2048$	Chama como procedimento a posição k, empilha PC+1. No programa a k é substituído por um rótulo.
RETURN	-	-	Faz o PC=topo da pilha. Deve ser chamado em conjunto com CALL.
RETLW	k	$0 \leq k \leq 255$	Armazena k em W e faz PC=topo da pilha. Deve ser chamado em conjunto com CALL.
RETFIE	-	-	Instrução de Retorno de uma interrupção.

4.7 ANATOMIA DE UM PROGRAMA ASM(PIC)

Aqui falaremos de: ORG, END, INCLUDE e DEFINE