Conjunto de Instruções do PIC16F628

Observações quanto aos termos utilizados na construção dos nomes das instruções e seus argumentos:

- **Work**: Trata-se de um registrador temporário para as operações da ULA. No Assembler do PIC, ele é conhecido como W. Também é comum chamá-lo de acumulador.
- **File**: Referência a um registrador (posição de memória) propriamente dito. Utilizaremos a letra F para sua representação nos nomes de instruções e f nos argumentos delas.
- **Literal**: Um número qualquer que pode ser escrito na forma decimal, hexadecimal ou binária. Utilizaremos a letra L para sua representação nos nomes de instruções e k nos argumentos delas.
- **Destino**: O local onde deve ser armazenado o resultado da operação. Os destinos podem ser 0 (W) ou 1 (F). A letra d será usada para indicar o destino de uma instrução, o destino pode ser o acumulador (d=0) ou o registrador (d=1).
- **Bit**: Refere-se a um bit específico dentro de um byte. Utilizaremos a letra B para sua representação nos nomes das instruções e b nos argumentos delas.

Para facilitar as operações de seus registradores especiais na RAM (que como recordamos estava incluído no código com a diretiva INCLUDE), a Microchip inseriu uma lista de nomes que identificam univocamente qualquer registrador especial e a qual está associado o endereço correspondente na área da memória RAM.

Se, por exemplo, quisermos definir toda a linha do PORTB do PIC como saída, devemos agir sobre o TRISB. Podemos escolher e referenciar diretamente o registrador com o seu endereço:

```
movlw B'00000000'
movwf 86H ; Endereço de TRISB
```

ou então, referenciar o mesmo registrador com o seu nome simbólico, neste caso tendo que ter a certeza de ter inserido a diretiva INCLUDE "P16F628.INC" (mostrado no Apêndice A desta apostila):

```
movlw B'00000000'
movwf TRISB ; Nome simbólico do endereço de TRISB
```

Para facilitar o estudo das instruções do PIC, organizamos a seguir duas tabelas. Uma com as instruções em ordem alfabética e a outra com as instruções divididas em quatro grupos, conforme as suas aplicações:

- Operações com registradores;
- Operações com literais;
- Operações com bits;
- Controles.

Conjunto de instruções do PIC16F628		
Operações em ordem alfabética		
Instrução	**Argumentos**	**Descrição**
ADDLW	K	Soma k com W, guardando o resultado em W (W = W + k).
ADDWF	f,d	Soma W e f, guardando o resultado em d (d = W + f).
ANDLW	K	Lógica “E” entre k e W, guardando o resultado em W (W = W AND k).
ANDWF	f,d	Lógica “E” entre W e f, guardando o resultado em d (d = W AND f).
BCF	f,b	Zera o bit b do registrador f.
BSF	f,b	Seta o bit b do registrador f.
BTFSC	f,b	Testa o bit b do registrador f, e pula a próxima linha se ele for 0 (zero).
BTFSS	f,b	Testa o bit b do registrador f, e pula a próxima linha se ele for 1 (um).
CALL	Label	Chamada a uma subrotina no endereço Label.
CLRF	F	Limpa o registrador f (f = 0).
CLRW		Limpa o acumulador (W = 0).
CLRWDT		Limpa o registrador WDT para evitar o reset (Watchdog timer = 0).
COMF	f,d	Pega o complemento de f, guardando o resultado em d (d = not f).
DECF	f,d	Decrementa f, guardando o resultado em d (d = f -1).
DECFSZ	f,d	Decrementa f, guardando o resultado em d, e pula a próxima linha se oresultado for zero (d = f –1, skip se zero).
GOTO	Label	Desvia para o endereço Label.
INCF	f,d	Incrementa f, guardando o resultado em d (d = f +1).
INCFSZ	f,d	Incrementa f, guardando o resultado em d, e pula a próxima linha se oresultado for zero (d = f+1, skip se zero).
IORLW	K	Lógica OU entre k e W, guardando o resultado em W (W = W OR k).
IORWF	f,d	Lógica OU entre W e f, guardando o resultado em d (d = f OR W).
MOVLW	K	Move (copia) valor literal k para o acumulador W (W = k).
MOVF	F,d	Move (copia) valor de registrador f para destino d (d = f).
MOVWF	F	Move (copia) valor do acumulador W para o registrador f (f = W).
NOP		Nenhuma operação, gasta um ciclo de máquina sem fazer nada.
RETFIE		Retorno de uma interrupção.
RETLW	K	Retorno de uma rotina, com k em W.
RETURN		Retorna de uma rotina.
RLF	f,d	Rotaciona f um bit a esquerda, guardando o resultado em d (d = f << 1).
RRF	f,d	Rotaciona f um bit a direita, guardando o resultado em d (d = f >> 1).
SLEEP		Coloca o PIC em modo sleep (dormindo) para economia de energia.
SUBLW	K	Subtrai W de k, guardando o resultado em W (W = k – W).
SUBWF	f,d	Subtrai W de f, guardando o resultado em d (d = f - W).

SWAPF	f,d	Executa uma inversão entre o nibble da parte alta e o nibble da partebaixa de f, guardando o resultado em d.
XORLW	W	Lógica ou-exclusivo entre k e W, guardando o resultado em W (W=WXOR k).
XORWF	f,d	Lógica ou-exclusivo entre W e f, guardando o resultado em d (d=WXOR f).

Conjunto de instruções do PIC16F628		
Instrução	**Argumentos**	**Descrição**
Grupo 1: Operações com registradores		
ADDWF	f,d	Soma W e f, guardando o resultado em d (d = W + f).
ANDWF	f,d	Lógica “E” entre W e f, guardando o resultado em d (d = W AND f).
CLRF	F	Limpa o registrador f (f = 0).
COMF	f,d	Pega o complemento de f, guardando o resultado em d (d = not f).
DECF	f,d	Decrementa f, guardando o resultado em d (d = f -1).
DECFSZ	f,d	Decrementa f, guardando o resultado em d, e pula a próxima linha se o resultadofor zero (d = f –1, skip se zero).
INCF	f,d	Incrementa f, guardando o resultado em d (d = f +1).
INCFSZ	f,d	Incrementa f, guardando o resultado em d, e pula a próxima linha se o resultadofor zero (d = f+1, skip se zero).
IORWF	f,d	Lógica OU entre W e f, guardando o resultado em d (d = f OR W).
MOVF	F,d	Move (copia) valor de registrador f para destino d (d = f).
MOVWF	F	Move (copia) valor do acumulador W para o registrador f (f = W).
RLF	f,d	Rotaciona f um bit a esquerda, guardando o resultado em d (d = f << 1).
RRF	f,d	Rotaciona f um bit a direita, guardando o resultado em d (d = f >> 1).
SUBWF	f,d	Subtrai W de f, guardando o resultado em d (d = f - W).
SWAPF	f,d	Executa uma inversão entre o nibble da parte alta e o nibble da parte baixa de f,guardando o resultado em d.
XORWF	f,d	Lógica ou-exclusivo entre W e f, guardando o resultado em d (d=W XOR f).
Grupo 2: Operações com literais		
ADDLW	K	Soma k com W, guardando o resultado em W (W = W + k).
ANDLW	K	Lógica “E” entre k e W, guardando o resultado em W (W = W AND k).
IORLW	K	Lógica OU entre k e W, guardando o resultado em W (W = W OR k).
MOVLW	K	Move (copia) valor literal k para o acumulador W (W = k).
SUBLW	K	Subtrai W de k, guardando o resultado em W (W = k – W).
XORLW	W	Lógica ou-exclusivo entre k e W, guardando o resultado em W (W=W XOR k).
Grupo 3: Operações com bits		
BCF	f,b	Zera o bit b do registrador f.
BSF	f,b	Seta o bit b do registrador f.
BTFSC	f,b	Testa o bit b do registrador f, e pula a próxima linha se ele for 0 (zero).
BTFSS	f,b	Testa o bit b do registrador f, e pula a próxima linha se ele for 1 (um).
Grupo 4: Controles		
CALL	Label	Chamada a uma subrotina no endereço Label.
CLRW		Limpa o acumulador (W = 0).
CLRWDT		Limpa o registrador WDT para evitar o reset (Watchdog timer = 0).
GOTO	Label	Desvia para o endereço Label.
NOP		Nenhuma operação, gasta um ciclo de máquina sem fazer nada.
RETFIE		Retorno de uma interrupção.
RETLW	K	Retorno de uma rotina, com k em W.
RETURN		Retorna de uma rotina.
SLEEP		Coloca o PIC em modo sleep (dormindo) para economia de energia.

1) ADDLW k ; Soma a constante k a W
Descrição: Soma a constante k ao valor memorizado no acumulador W e coloca o resultado no acumulador.
Exemplo:

```
movlw 10
addlw 12 ; após o trecho de programa, o acumulador W terá o valor 22
```

2) ADDWF f,d ; Soma o valor contido em W com o valor contido no registrador F
Descrição: Esta instrução soma o valor contido no acumulador W com o valor contido no registrador endereçado pelo parâmetro f.
Exemplo: Vejamos um exemplo de soma entre dois registradores:

```
add1 equ 0CH
add2 equ 0DH
    org 00H
    movlw 10 ;Primeiro somador = 10
    movwf add1
    movlw 15 ;Segundo somador = 15
    movwf add2
    movf add1,W ;W = add1
    addwf add2,W ;W = add1 + add2
```

3) ANDLW k ; Efetua o AND bit a bit entre W e uma constante k
Descrição: Efetua o AND bit a bit entre o valor contido no acumulador W e o valor constante k. O resultado será memorizado no acumulador.
Exemplo:

```
movlw ‘10101010’B
andlw ‘11110000’B
...
```

Depois de haver executado este trecho de programa o acumulador W irá valer 10100000B.

4) ANDWF f,d ; Efetua o AND bit a bit entre o valor contido em W e o valor contido no registrador F.
Descrição: Esta instrução efetua o AND bit a bit entre o valor contido no acumulador W e o valor contido no registrador endereçado pelo parâmetro f.
Exemplo: Freqüentemente o AND será utilizado para mascarar o valor de algum bit dentro de um registrador. Se por exemplo quiséssemos extrair do número binário 01010101B os quatro bits menos significativo a fim de obter o seguinte valor 00000101B, bastará preparar uma máscara do tipo 00001111B e fazer o AND com o nosso valor, vejamos como:

```
        movlw 01010101B ; Armazena em W o valor binário
        movwf 0CH ; Usa o endereço 0CH para armazenar o valor inicial da

        máscara
        movlw 00001111B ; Prepara a máscara do bit
        andwf 0CH,W ; Efetua o AND e memoriza o resultado no acumulador W
```

O resultado em W será 00000101B como descrito.

```
    W = 00001111 AND
    f = 01010101 =
    ----------
    W = 00000101
```

5) BCF f,b ; Zera o bit b do registrador F
Descrição: Esta instrução zera o bit b do registrador no endereço f .
Exemplo:

```
parm1 equ 0CH
    movlw '11111111'B ;Valor inicial
    movwf parm1
    bcf parm1,0
```

Ao término do programa o registrador parm1 será 11111110B.

6) BSF f,b ;Coloca em nível alto o bit b no registrador F.
Descrição: Esta instrução coloca em "um" no bit b do registrador que está no endereço f.
Exemplo:

```
parm1 equ 0CH
    movlw 00000000B ;Valore inicial
    movwf parm1
    bsf parm1,0 ;D0=1
```

Ao terminar o programa o registrador parm1 será 00000001B.

7) BTFSC f,b ; Pula a próxima instrução se o bit b do registrador F for 0
Descrição: Testa o bit b contido no registrador no endereço f e pula a próxima instrução se este valer 0.
Exemplo:

```
parm1 equ 0CH
    org 00H
    movlw 11111110B ;Valor inicial
    movwf parm1
loop
    btfsc parm1,0 ;bit0 = 0? Se for, pular próxima instrução.
    goto loop ;Se não, ficar no loop
```

Este programa executa um loop infinito. Entretanto, o mesmo programa não executará o loop se substituirmos a instrução: movlw 11111110B pela instrução: movlw 11111111B.

8) BTFSS f,b ; Pula a próxima instrução se o bit b do registrador F for 1
Descrição: Testa o bit b contido no registrador do endereço f e pula a instrução seguinte se

este for 1.
Exemplo:

```
    parm1 equ 0CH
    org 00H
    movlw 11111111B ;Valor inicial
    movwf parm1
    loop
    btfss parm1,0 ;bit0 = 1 ? Se for, pular próxima instrução.
    goto loop ;Se não, ficar no loop
```

Este programa executa um loop infinito. O mesmo programa não executará o loop se substituirmos a instrução:
movlw 11111111B pela instrução: movlw 11111110B.

9) CALL k ; Chamada a uma subrotina
Descrição: Chama uma subrotina memorizada no endereço k. O parâmetro k pode ser especificado utilizando-se diretamente o valor numérico do endereço ou então o relativo label.
Exemplo:

```
#define LED1 1
        org 00H
    call ledOn ; Chama a rotina ledOn
    ledOn
    btfsc PORTB,LED1 ; testa o bit 1 da porta B
    return
```

Quando a CPU do PIC encontra uma instrução CALL, memoriza no STACK o valor do registrador PC+1 de modo a poder retornar para instrução após o CALL, em seguida escreve no PC o endereço da subrotina pulando a execução desta ultima. O valor original do PC será recuperado pela subrotina com a execução da instrução de retorno RETURN ou RETLW.
No PIC16F628 estão disponíveis 8 níveis de stack (pilha), ou seja a instrução CALL dentro de uma subrotina pode ter no máximo 8 chamadas ou 8 níveis. As demais chamadas serão sobrepostas às primeiras.

10) CLRF f ; Zera o registrador F
Descrição: Esta instrução zera o valor contido no registrador endereçado pelo parâmetro f.
Exemplo: Se quisermos zerar o registrador TMR0 no qual o endereço é 01H hexadecimal, a instrução a se executar será:

```
        clrf 01H
```

Ou, se no endereço do nosso código incluirmos o arquivo P16F628.INC, poderemos utilizar o nome simbólico do registrador TMR0, ou seja, clrf TMR0.

11) CLRW ; Zera o registrador W
Descrição: Zera o valor contido no registrador W.
Exemplo:

```
clrw
```

12) CLRWDT ; Limpa o registrador WDT para não acontecer o Reset
Descrição: Esta instrução deve ser utilizada quando programarmos o PIC com a opção Watchdog (fusível WDTE). Nesta modalidade o PIC habilita um timer que, uma vez transcorrido um determinado tempo, efetua o reset do mesmo. Para evitar o reset do nosso programa deveremos executar ciclicamente a instrução CLRWDT para zerar o timer antes deste tempo. Se não zerarmos o WDT neste tempo, o circuito de watchdog (do inglês cão de guarda) interpretará este como um bloco de programa em execução e efetuará o reset para bloquea-lo.
Exemplo:

```
    org 00H
    loop
    clrwdt
    goto loop
```

13) COMF f,d ; Efetua o complemento do registrador F
Descrição: Esta instrução efetua o complemento do valor contido no registrador endereçado pelo parâmetro f.
Exemplo:

```
        parm1 equ 0CH
        org 00H
```

```
movlw 01010101B
movwf parm1
comf parm1,F
```
Ao término da execução do programa o valor do registrador parm1 será 10101010B.

14) DECF f,d ; Decrementa o conteúdo do registrador F
Descrição: Esta instrução decrementa o conteúdo do registrador endereçado pelo parâmetro f.
Exemplo:

```
movlw 23H ;Escreve em W o valor 23H
movwf 0CH ;Copia no registrador 0CH o valor de W
decf 0CH,F ;Decremente o valor contido no registrador 0CH
```

15) DECFSZ f,b ; Decrementa o valor do registrador f e pula a próxima instrução se o resultado for zero.
Descrição: Decrementa o valor de registrador do endereço f e se o resultado for zero pula a próxima instrução. **Exemplo**:

```
counter equ 0CH
    org 00H
    movlw 10 ;counter = 10
    movwf counter
    loop

    decfsz counter,F ;counter = counter –1, se counter = 0, pula próxima instrução
    goto loop ;se não, continua no loop
```
Este programa executa 10 vezes a instrução decfsz até que counter seja = 0.

16) GOTO k ; Desvia a execução do programa para o endereço especificado k.
Descrição: Determina o desvio incondicional do programa em execução para o endereço k. O parâmetro k pode ser especificado utilizando-se diretamente um valor numérico do endereço ou então o relativo label.
Exemplo:

```
org 00H
loop
goto loop
```
Este programa executa um cilclo (loop) infinito.

17) INCF f,d ; Incrementa o valor do registrador no endereço F.
Descrição: Incrementa o conteúdo do registrador no endereço f.
Exemplo:

```
movlw 23H ;Escreve em W o valor 23H
movwf 0CH ;Copia no registrador 0CH o valor de W
incf 0CH,F ;Incrementa de 1 valor contido no registrador 0CH
```

18) INCFSZ f,b ; Incrementa o valor do registrador f e pula a próxima instrução se o resultado for zero.
Descrição: Incrementa o valor do registrador f e se o resultado for zero pula a próxima instrução.
Exemplo:

```
counter equ 0CH
    org 00H
    movlw 250 ;counter = 250
    movwf counter
```

loop
incfsz counter,F ;counter = counter + 1, se counter = 0 ? pular próxima instrução
goto loop ; se não, continuar no loop
Este programa executa para 256-10 = 6 vezes a instrução incfsz até que counter seja 0. Sendo counter um registrador de 8 bit's quando for incrementado do valor 255 assume novamente o valor 0 e dai a formula 256 - 10 = 6.

19) IORLW k ; Efetua o OU inclusive entre W e uma constante k
Descrição: Efetua o OR inclusive entre o valor contido no acumulador W e o valor da constante k.
Exemplo:
org 00H

start
movlw 00001111B
iorlw 11110000B
Após ser executado esse programa o acumulador W será 11111111B.

20) IORWF f,d ; Efetua o OR inclusive entre o valor contido em W e o valor contido no registrador F
Descrição: Esta instrução efetua o OR inclusive entre o valor contido no acumulador W e o valor contido no registrador endereçado pelo parâmetro f.
Exemplo:
parm1 equ 0CH
org 00H
movlw 00001111B
movwf parm1
movlw 11111111B
iorwf parm1,F
Ao término do programa o valor do registrador parm1 será 11111111B.

21) MOVLW k ; Copia para W o valor constante k
Descrição: Passa ao acumulador W um valor constante k.
Exemplo:
org 00H
movlw 20
Após ter executado este programa o acumulador W irá a 20.

22) MOVF f,d ;Copia o conteúdo do registrador f para o destino d
Descrição: Esta instrução copia o conteúdo do registrador endereçado pelo parâmetro f para o parâmetro de destino d. **Exemplo**: O exemplo a seguir copia o valor contido no registrador do endereço 0CH no acumulador W:
movf 0CH,W

23) MOVWF f ; Copia o conteúdo do registrador W para o registrador F
Descrição: Esta instrução copia o conteúdo do registrador W no registrador de parâmetro f.
Exemplo: Para copia o valor 10H no registrador TMR0. A instrução a se executar será a seguinte:
movlw 10H ;Escreve no registrador W o valor 10H
movwf TMR0 ;e o memoriza no registrador TMR0

24) NOP ; Nenhuma operação

Descrição: Esta instrução não executa nenhuma operação mas é útil para inserir atrasos de um ciclo de maquina ou mais.
Exemplo:

```
nop
nop
```

Os dois nops acima vão provocar um atraso de 2 uS se utilizarmos um cristal de 4MHz no nosso hardware.

25) RETFIE ; Retorna de uma rotina de interrupção
Descrição: Esta instrução deve ser colocada no término de cada subrotina de controle de interrupções para retornar o controle ao programa principal.
Exemplo:

```
org 00H
loop
goto loop ;Loop infinito
org 04H ;Interrupt vector
intHandler
retfie ;Retorna da interrupção
```

Neste código o programa principal executa um loop infinito. Se habilitarmos uma das interrupções do 16F628 ele não apenas verificará o controle como irá automaticamente ao programa alocado no endereço 04H (no exemplo intHandler), a instrução RETFIE determinará então o retorno ao loop principal.

26) RETLW k ; Retorna de uma rotina com uma constante k em W
Descrição: Esta instrução retorna o controle de uma rotina ao programa principal. A diferença desta em relação à instrução RETURN é que retflw permite retornar, através do acumulador W, o valor k ao programa principal.
Exemplo:

```
rtc equ 0CH
org 00H
call mySub1
movwf rtc
...
mySub1
nop
retlw 10
```

Uma vez executado esse programa ele memorizará no registrador rtc o valor 10 passado pela subrotina mySub1.

27) RETURN ; Retorna de uma rotina
Descrição: Esta instrução deve ser inserida no termino de cada subrotina para retornar a execução ao programa principal.
Exemplo:

```
org 00H
call mySub1
....
mySub1
nop
return
```

Nota: No PIC16F628 podemos fazer apenas 8 chamadas a subrotinas, do tipo:

```
org 00H
call mySub1
```

```
....
mySub1
call mySub2
return
mySub2
call mySub3
return
mySub3
return
```

28) RLF f,b ;Rotaciona a esquerda o conteúdo do registrador f passando pelo Carry
Descrição: Rotaciona o bit contido no registrador do endereço f para a esquerda (ou seja do bit menos significativo para o mais significativo) passando pelo CARRY do registrador STATUS como ilustrado na figura a seguir:

Figura 7 – Rotação de bit a esquerda BIT7BIT6BIT5BIT4BIT3BIT2BIT1BIT0CARRY

O conteúdo do bit CARRY do registrador STATUS será deslocado para o bit0 enquanto que o valor do bit7 será deslocado para o CARRY.
Exemplo:

```
parm1 equ 0CH
org 00H
bcf STATUS,C ;Zera o CARRY
movlw 01010101B ;Valor inicial
movwf parm1
rlf parm1,F
```

Ao término do programa o registrador parm1 será 10101010B enquanto o CARRY será 0.

29) RRF f,b ; Rotaciona para a direita o conteúdo do registrador f passando pelo Carry
Descrição: Rotaciona o bit contido no registrador do endereço f para direita (ou seja do bit mais significativo para o menos significativo) passando pelo bit CARRY do registrador STATUS como ilustrado na figura a seguir:

Figura 8 – Rotação de bit a direita BIT7BIT6BIT5BIT4BIT3BIT2BIT1BIT0CARRY
O conteúdo do bit CARRY do registrador STATUS será deslocado para o bit7 enquanto o valor do bit0 será deslocado para o CARRY.
Exemplo:

```
parm1 equ 0CH
org 00H
bcf STATUS,C ;Zera o CARRY
movlw 01010101B ;Valor inicial
movwf parm1
rrf parm1,F
```

Ao término do programa o registrador parm1 será 00101010B enquanto o CARRY será 1.

30) SLEEP ; Coloque o PIC (para dormir) em standby
Descrição: Esta instrução bloqueia a execução do programa em andamento e coloca o PIC em standby (sleep do inglês = dormir).
Exemplo: org 00H

```
start
sleep
```

31) SUBLW k ; Subtraia de k o valor em W
Descrição: Subtrai a constante k do valor memorizado no acumulador W.
Exemplo:

```
org 00H
start
movlw 10 ; W = 10
sublw 12 ; W = 12 - 10
...
```

Depois de haver executado este programa o acumulador W será 2.

32) SUBWF f,d ;Subtraia o valor contido em W do valor contido no registrador F.
Descrição: Esta instrução subtrai o valor contido no registrador W do valor contido no registrador endereçado pelo parâmetro f.
Exemplo: Analisando um exemplo extraído do datasheet da Microchip:
Se inserirmos a instrução:

```
subwf REG1,F
```

Onde reg1 é o endereço de um registrador qualquer especificado com a diretiva:

```
REG1 RES 1
```

Para o valor inicial de REG1=3 e W=2, teremos REG1=1 e C=1 porque o resultado é positivo.
Para o valor inicial de REG1=2 e W=2, teremos REG1=0 e C=1 porque o resultado

é positivo.
Para o valor inicial de REG1=1 e W=2, teremos REG1=FFH ou seja -1 e C=0 porque o resultado é negativo.

33) SWAPF f,d ; Troca de nibbles.
Descrição: Troca o valor dos quatro bits mais significativo (D7-D4) contido no registrador do endereço f com os quatro bits menos significativo(D3-D0) do mesmo.
Exemplo:

```
movlw ‘11110000’B ;Valor inicial
swapf parm1,F
```

Ao término do programa o registrador parm1 será 00001111B.

34) XORLW k ; Efetua o OR exclusivo entre W e uma constante k
Descrição: Efetua o OR exclusivo entre o valor contido no acumulador W e o valor constante k.
Exemplo:

```
org 00H
start
movlw 00000000B
xorlw 11110000B
...
```

Após haver executado este programa o acumulador W será 11110000B.

35) XORWF f,d ; Efetua o OR exclusivo entre o valor contido em W e o valor contido no registrador
Descrição: Esta instrução efetua o OR exclusivo(XOR) entre o valor contido no acumulador W e o valor contido no registrador endereçado pelo parâmetro f.
Exemplo: Efetuar um XOR entre o registrador W e o registrador REG1 por nós definido no endereço 0CH com a diretiva:

```
REG1 EQU 0CH
```

podemos utilizar a instrução IORWF de duas formas, dependendo onde queremos colocar o resultado, ou seja:

```
xorwf COUNTER,F ;COUNTER = COUNTER XOR W
```

ou então:

```
xorwf COUNTER,W ;W = COUNTER XOR W
```