Set istruzioni del PIC16F877A

Questo è il set delle istruzioni riconosciute dal PIC16F877A:

Sintassi	Descrizione Microchip	Operazione equivalente
ADDLW k	Add Literal and W	W = W + k
ADDWF f,d	Add W and f	d = W + f (dove d può essere W o f)
ANDLW k	AND Literal with W	W = W AND k
ANDWF f,d	AND W with f	d = W AND f (dove d può essere W o f)
BCF f,b	Bit Clear f	f(b) = 0
BSF f,b	Bit Set f	f(b) = 1
BTFSC f,b	Bit Test f, Skip if Clear	f(b) = 0 ? Si, salta una istruzione
BTFSS f,b	Bit Test f, skip if Set	f(b) = 1 ? Si, salta una istruzione
CALL k	Subroutine Call	Chiama la subroutine all'indirizzo k
CLRF f	Clear f	f = 0
CLRW	Clear W Register	W = 0
CLRWDT	Clear Watchdog Timer	Watchdog timer = 0
COMF f,d	Complement f	d = not f (dove d può essere W o f)
DECF f,d	Decrement f	d = f -1 (dove d può essere W o f)
DECFSZ f,d	Decrement f, Skip if 0	d = f -1 (dove d può essere W o f) se d = 0 salta
GOTO k	Go to address	Salta all'indirizzo k
INCF f,d	Increment f	d = f +1 (dove d può essere W o f)
INCFSZ f,d	Increment f, Skip if 0	d = f +1 (dove d può essere W o f) se d = 0 salta
IORLW k	Inclusive OR Literal with W	W = W OR k
IORWF f,d	Inclusive OR W with f	d = f OR W (dove d può essere W o f)
MOVLW k	Move literal to W	W = k
MOVF f,d	Move f	d = f (dove d può essere W o f)
MOVWF f	Move W to f	f = W
NOP	No Operation	Nessuna operazione
OPTION	Load Option Register	OPTION = W
RETFIE	Return from Interrupt	Ritorna da un interrupt handler
RETLW k	Return Literal to W	Ritorna da una subroutine con W = k
RETURN	Return from Subroutine	Ritorna da una subroutine
RLF f,d	Rotale Left f through Carry	d = f << 1 (dove d può essere W o f)
RRF f,d	Rotale Right f through Carry	d = f >> 1 (dove d può essere W o f)
SLEEP	Go into Standby Mode	Mette in standby il PIC
SUBLW k	Subtract W from Literal	W = k - W
SUBWF f,d	Subtract W from f	d = f - W (dove d può essere W o f)
SWAPF f	Swap f	f = Swap dei bit 0123 con 4567 di f
TRIS f	Load TRIS Register	TRIS di f = W
XORLW k	Exclusive OR Literal with W	W = W XOR k
XORWF f,d	Exclusive OR W with f	d = f XOR W (dove d può essere W o f)

ADDLW — ADD Literal and W

Somma la costante k a W

Sintassi:

```
addlw   k
```

Operazione equivalente:

W = W + k

Descrizione:

Somma la costante **k** al valore memorizzato nell'accumulatore **W** e mette il risultato nell'accumulatore.

Esempio:

```
        org     00H

start
        movlw   10
        addlw   12
        ...
```

Dopo aver eseguito questo programma l'accumulatore W varrà 22.

Note:

Questa istruzione influenza i bit **Z**, **DC** e **C** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.
- **DC** vale 1 se il risultato dell'operazione è un numero superiore a 15.
- **C** vale 1 se il risultato è positivo ovvero se il bit 7 del registro contenente il risultato vale 0 e 1 se il risultato è negativo ovvero se il bit 7 del registro contenente il risultato vale 1.

ADDWF — ADD W and F

Somma il valore contenuto in W con il valore contenuto nel registro F

Sintassi:

```
addwf   f,d
```

Operazione equivalente:

d = W + f (dove d può essere W o f)

Descrizione:

Questa istruzione somma il valore contenuto nell'accumulatore **W** con il valore contenuto nel registro indirizzato dal parametro **f**. Il parametro **d** è un flag che indica su quale registro deve essere memorizzato il risultato.

Per **d = W** il risultato viene memorizzato nel **registro W**
Per **d = F** il risultato viene memorizzato nel **registro f**

Esempio:

Vediamo un esempio di somma tra due registri:

```
add1    equ     0CH
add2    equ     0DH

        org     00H

        movlw   10      ;Primo addendo = 10
        movwf   add1

        movlw   15      ;Secondo addendo = 15
        movwf   add2

        movf    add1,W  ;W = add1
        addwf   add2,W  ;W = W + add2
```

Note:

Questa istruzione influenza i bit **Z**, **DC** e **C** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.
- **DC** vale 1 se il risultato dell'operazione è un numero superiore a 15.
- **C** vale 1 se il risultato è positivo ovvero se il bit 7 del registro contenente il risultato vale 0 e 1 se il risultato è negativo ovvero se il bit 7 del registro contenente il risultato vale 1.

ANDLW — AND Literal with W

Effettua l'AND tra W ed una costante k

Sintassi:

```
andlw   k
```

Operazione equivalente:

W = W AND k

Descrizione:

Effettua l'AND tra il valore contenuto nell'accumulatore **W** ed il valore costante **k**. Il risultato viene memorizzato nell'accumulatore.

Esempio:

```
        org     00H

start
        movlw   10101010B
        andlw   11110000B
        ...
```

Dopo aver eseguito questo programma l'accumulatore W varrà 10100000B.

Note:

Questa istruzione influenza il bit **Z** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.

ANDWF — AND W with F

Effettua l'AND logico tra il valore contenuto in W ed il valore contenuto nel registro F

Sintassi:

```
andwf   f,d
```

Operazione equivalente:

d = W AND f (dove d può essere W o f)

Descrizione:

Questa istruzione effettua l'AND logico tra il valore contenuto nell'accumulatore **W** ed il valore contenuto nel registro indirizzato dal parametro **f**. Il parametro **d** è un flag che indica su quale registro deve essere memorizzato il risultato.

Per **d = W** il risultato viene memorizzato nel **registro W**
Per **d = F** il risultato viene memorizzato nel **registro f**

Esempio:

Spesso l'AND logico viene utilizzato per mascherare il valore di alcuni bit all'interno di un registro. Se ad esempio volessimo estrarre dal numero binario 01010101B i quattro bit meno significativi al fine di ottenere il seguente valore 00000101B, basterà preparare una maschera del tipo 00001111B e farne l'AND con il nostro valore di partenza, vediamo come:.

```
movlw   01010101B  ;Memorizza nel registro                    ; all'indirizzo
movwf   0CH        ;0CH il valore iniziale da mascherare

movlw   00001111B  ;Prepara la maschera di bit
andwf   0CH,W      ;Effettua l'AND e memorizza il
                   ;risultato nell'accumulatore W
```

Il risultato in W sarà 00000101B come rischiesto.

```
W = 00001111 AND
f = 01010101 =
---------------
W = 0000010101
```

La **ANDWF** influenza il bit **Z** del registro **STATUS** che varrà 1 se il risultato dell'operazione è 0.

BCF — Bit Clear F

Azzera un bit nel registro F

Sintassi:

```
bcf    f,b
```

Operazione equivalente:

f(b) = 0

Descrizione:

Questa istruzione azzera il **bit b** del registro all'indirizzo **f**.

Esempio:

```
parm1   equ     0CH

        org     00H

        movlw   11111111B  ;Valore iniziale
        movwf   parm1

        bcf     parm1,0    ;D0=0
```

Al termine del programma il registro parm1 varrà **11111110B**.

Note:

Questa istruzione non influenza alcun bit di stato

BSF — Bit Set F

Mette a uno un bit nel registro F

Sintassi:

```
bsf    f,b
```

Operazione equivalente:

f(b) = 1

Descrizione:

Questa istruzione mette a uno il **bit b** del registro all'indirizzo **f**.

Esempio:

```
parm1   equ     0CH

        org     00H

        movlw   00000000B  ;Valore iniziale
        movwf   parm1

        bsf     parm1,0    ;D0=1
```

Al termine del programma il registro parm1 varrà **00000001B**.

Note:

Questa istruzione non influenza alcun bit di stato

BTFSC — Bit Test F, Skip if Clear

Salta l'istruzione successiva se un bit nel registro F vale 0

Sintassi:

```
btfsc   f,b
```

Operazione equivalente:

f(b) = 0 ? Si, salta una istruzione

Descrizione:

Testa il bit b contenuto nel registro all'indirizzo f e salta l'istruzione successiva se questo vale 0.

Esempio:

```
parm1   equ     0CH

        org     00H
```

```
        movlw   11111110B  ;Valore iniziale
        movwf       parm1
loop
        btfsc   parm1,0    ;D0 = 0 ? Si, esce
        goto    loop       ;No, esegue il loop
```

Questa programma esegue un loop infinito lo stesso programma non esegue il loop se sostituiamo l'istruzione:

```
        movlw  11111110B   ;Valore iniziale
```

con l'istruzione:

```
        movlw  11111111B   ;Valore iniziale
```

Note:

Questa istruzione non influenza alcun bit di stato

BTFSS — Bit Test F, Skip if Set

Salta l'istruzione successiva se un bit nel registro F vale 1

Sintassi:

```
btfss    f,b
```

Operazione equivalente:

f(b) = 1 ? Si, salta una istruzione

Descrizione:

Testa il bit b contenuto nel registro all'indirizzo f e salta l'istruzione successiva se questo vale 1.

Esempio:

```
parm1   equ     0CH

        org     00H

        movlw   11111111B  ;Valore iniziale
        movwf   parm1
loop
        btfss   parm1,0    ;D0 = 1 ? Si, esce
        goto    loop       ;No, esegue il loop
```

Questa programma esegue un loop infinito lo stesso programma non esegue il loop se sostituiamo l'istruzione:

```
        movlw   11111111B  ;Valore iniziale
```

con l'istruzione:

```
        movlw   11111110B  ;Valore iniziale
```

Note:

Questa istruzione non influenza alcun bit di stato

CALL — Subroutine CALL

Chiamata a subroutine

Sintassi:

```
call      k
```

Descrizione:

Richiama in esecuzione una subroutine memorizzata all'indirizzo **k**.Il parametro k può essere specificato utilizzando direttamente il valore numerico dell'indirizzo oppure la relativa label.

Esempio:

```
        org     00H

        call    ledOn
        ...
        ;Subroutine di accensione di un led
ledOn
        bsf     PORTB,LED1
        return
```

Quando la CPU del PIC incontra una istruzione CALL, memorizza nello STACK il valore del registro PC + 1 in modo da poter riprendere l'esecuzione dall'istruzione successiva alla CALL, quindi scrive nel PC l'indirizzo della subroutine saltando all'esecuzione di quest'ultima.

Il valore originale del PC viene ripristinato all'uscita della subroutine con l'esecuzione dell'istruzione di ritorno RETURN o RETLW.

Nel PIC16F84 sono disponibili 8 livelli di stack, per cui il numero massimo di CALL rientranti, ovvero di istruzioni CALL all'interno di subroutine che a loro volta contengono altre CALL, è limitato ad 8 livelli.

Note:

Questa istruzione non influenza nessun bit di stato.

CLRF — CLeaR F register

Azzera il registro F

Sintassi:

```
clrf    f
```

Operazione equivalente:

f = 0

Descrizione:

Questa istruzione azzera il valore contenuto nel registro indirizzato dal parametro **f**.

Esempio:

Ipotizziamo di voler azzerare il registro **TMR0** il cui indirizzo è 01H esadecimale, l'istruzione da eseguire sara:

```
clrf    01H
```

Oppure, se si include all'inizio del nostro source il file P16F84.INC, potremo utilizzare il nome simbolico del registro **TMR0**.

```
clrf    TMR0
```

Dopo l'esecuzione di questa istruzione il bit **Z** del registro **STATUS** viene messo a **1**.

CLRW — CLeaR W register

Azzera il registro W

Sintassi:

```
clrw
```

Operazione equivalente:

W = 0

Descrizione:

Azzera il valore contenuto nel registro **W**.

Note:

Dopo l''esecuzione di questa istruzione il bit **Z** del registro **STATUS** viene messo a **1**.

CLRWDT — CLeaR WatchDog Timer

Reset del timer del watchdog

Sintassi:

```
clrwdt
```

Descrizione:

Questa istruzione deve essere utilizzata quando programmiamo il PIC con l'opzione Watchdog abilitata (fusibile WDTE). In questa modalità il PIC abilita un timer che, una volta trascorso un determinato tempo, effettua il reset del PIC. Per evitare il reset il nostro programma dovrà eseguire ciclicamente l'istruzione CLRWDT per azzerare il timer prima di detto tempo. Se non azzeriamo il timer in tempo, la circuiteria di watchdog (dall'inglese cane da guardia) interpreterà questo come un blocco del programma in esecuzione ed effettuerà il reset per sbloccarlo.

Esempio:

```
        org     00H

loop
        clrwdt
        goto    loop
```

Note:

Questa istruzione non influenza nessun bit di stato.

COMF — COMplement F

Effettua il complemento del registro F

Sintassi:

```
comf    f,d
```

Operazione equivalente:

d = NOT f (dove d può essere W o f)

Descrizione:

Questa istruzione effettua il complemento del valore contenuto nel registro indirizzato dal parametro **f**. Il parametro **d** determina la destinazione del valore ottenuto.

Per **d = W** il valore viene memorizzato nel **registro W**
Per **d = F** il valore viene lasciato nel **registro f**.

Esempio:

```
parm1   equ     0CH

        org     00H
```

```
movlw   01010101B
movwf   parm1

comf    parm1,F
...
```

Al termine dell'esecuzione del programma il valore del registro **parm1** sarà **10101010B**.

Note:

Questa istruzione influenza il bit **Z** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.

DECF — DECrement F register

Azzera il contenuto del registro F

Sintassi:

```
decf    f,d
```

Operazione equivalente:

d = f -1 (dove d può essere W o f)

Descrizione:

Questa istruzione decrementa il contenuto del registro indirizzato dal parametro **f**. Il parametro **d** è un flag che indica su quale registro deve essere memorizzato il risultato.

Per **d = W** il risultato viene memorizzato nel **registro W**
Per **d = F** il risultato viene memorizzato nel **registro f**

Esempio:

Con il seguente programma scriviamo il valore 23H nel registro all'indirizzo 0CH e quindi lo decrementiamo di uno. Al termine dell'esecuzione il registro alla locazione 0CH conterrà il valore 22H.

```
movlw   23H   ;Scrive in W il valore 23H
movwf   0CH   ;Copia nel registro 0CH il valore di W
decf    OCH,F ;Decremente il valore contenuto nel
              ;registro 0CH
```

Esempio:

Questa istruzione influenza il bit **Z** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.

DECFSZ — DECrement F, Skip if Zero

Decrementa il valore del registro f e salta l'istruzione successiva se il risultato vale zero

Sintassi:

```
decfsz  f,b
```

Operazione equivalente:

d = f -1 (dove d può essere W o f) se d = 0 salta

Descrizione:

Decrementa il valore del registro all'indirizzo **f** e se il risultato vale zero salta l'istruzione successiva. Il risultato del decremento può essere memorizzato nello stesso registro **f** oppure nell'accumulatore **W** in base al valore del flag **d**.

Per **d = W** il risultato viene memorizzato nel **registro W**
Per **d = F** il risultato viene memorizzato nel **registro f**

Esempio:

```
counter equ     0CH

        org     00H

        movlw   10        ;counter = 10
        movwf   counter
loop
        decfsz  counter,F  ;counter = counter -1
                           ;counter = 0 ? Si esce
        goto    loop       ;No, continua
```

Questa programma esegue per 10 volte l'istruzione decfsz finchè esce per counter = 0.

Note:

Questa istruzione non influenza alcun bit di stato.

GOTO — GO TO address

Vai in esecuzione all'indirizzo k

Sintassi:

```
goto    k
```

Descrizione:

Determina un salto del programma in esecuzione all'indirizzo k. Il parametro k può essere

specificato utilizzando direttamente il valore numerico dell'indirizzo oppure la relativa label.

Esempio:

```
      org     00H

loop
      goto    loop
```

Questro programma esegue un ciclo (loop) infinito.

Note:

Questa istruzione non influenza nessun bit di stato.

INCF — INCrement F

Incrementa il valore del registro all'indirizzo F

Sintassi:
incf f,d

Operazione equivalente:

d = f +1 (dove d può essere W o f)

Descrizione:

Incrementa il contenuto del registro all'indirizzo **f** e memorizza il risultato nello stesso registro o nell'accumulatore **W** in base al valore del flag **d**:

Per **d = W** il risultato viene memorizzato nel **registro W**
Per **d = F** il risultato viene memorizzato nello stesso **registro F**

Note:

Questa istruzione influenza il bit **Z** del registro **STATUS**..

- **Z** vale 1 se il risultato dell'operazione vale 0.

INCFSZ — INCrement F, Skip if Zero

Incrementa il valore del registro f e salta l'istruzione successiva se il risultato vale zero

Sintassi:
incfsz f,b

Operazione equivalente:

d = f +1 (dove d può essere W o f) se d = 0 salta

Descrizione:

Incrementa il valore del registro all'indirizzo **f** e se il risultato vale zero salta l'istruzione successiva. Il risultato dell'incremento può essere memorizzato nello stesso registro **f** oppure nell'accumulatore **W** in base al valore del flag **d**.

Per **d = W** il risultato viene memorizzato nel **registro W**
Per **d = F** il risultato viene memorizzato nel **registro f**

Esempio:

```
counter equ     0CH

        org     00H

        movlw   250       ;counter = 250
        movwf   counter
loop
        incfsz  counter,F ;counter = counter + 1
                          ;counter = 0 ? Si esce
        goto    loop      ;No, continua
```

Questa programma esegue per 256 - 10 = 6 volte l'istruzione incfsz finchè esce per counter = 0. Essendo counter un registro a 8 bit quando viene incrementato dal valore 255 assume nuovamente valore 0 da qui la formula 256 - 10 = 6.

Note:

Questa istruzione non influenza alcun bit di stato.

IORLW — Inclusive OR Literal with W

Effettua l'OR inclusivo tra W ed una costante k

Sintassi:
iorlw k

Operazione equivalente:

W = W OR k

Descrizione:

Effettua l'OR inclusivo tra il valore contenuto nell'accumulatore **W** ed il valore costante **k**.

Esempio:

```
      org     00H
```

```
start
        movlw   00001111B
        iorlw   11110000B
        ...
```

Dopo aver eseguito questo programma l'accumulatore W varrà **11111111B**.

Note:

Questa istruzione influenza il bit **Z** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.

IORWF — Inclusive OR W with F

Effettua l'OR inclusivo tra il valore contenuto in W ed il valore contenuto nel registro F

Sintassi:

```
iorwf     f,d
```

Operazione equivalente:

d = f OR W (dove d può essere W o f)

Descrizione:

Questa istruzione effettua l'OR inclusivo tra il valore contenuto nell'accumulatore **W** ed il valore contenuto nel registro indirizzato dal parametro **f**. Il parametro **d** determina dove viene memorizzato il risultato dell'operazione:

Per **d = W** il risultato viene memorizzato nell'accumulatore **W**.
Per **d = F** il risultato viene memorizzato nel registro **f**.

Esempio:

```
parm1   equ     0CH

        org     00H

        movlw   00001111B
        movwf   parm1

        movlw   11111111B
        iorwf   parm1,F
```

Al termine dell'esecuzione il valore del registro parm1 sarà **11111111B**.

Note:

Questa istruzione influenza il bit **Z** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.

MOVLW — MOVe Literal to W

Assegna a W un valore costante

Sintassi:

```
movlw   k
```

Operazione equivalente:

W = k

Descrizione:

Assegna all'accumulatore **W** il valore costante **k**.

Esempio:

```
        org   00H

start
        movlw   20
        ...
```

Dopo aver eseguito questo programma l'accumulatore W varrà 20.

Note:

Questa istruzione non influenza nessun bit di stato.

MOVF — MOVe F

Muove il contenuto del registro F

Sintassi:

```
movf    f,d
```

Operazione equivalente:

d = f (dove d può essere W o f)

Descrizione:

Questa istruzione copia il contenuto del registro indirizzato dal parametro **f** o nell'accumulatore **W**

o nello stesso **registro F**. Il parametro **d** determina la destinazione.

Per **d = W** il valore viene memorizzato nel **registro W**
Per **d = F** il valore viene lasciato nel **registro f**. In questo caso l'utilità dell'istruzione sta nel fatto che viene alterato il bit Z del flag STATUS in base al valore contenuto nel **registro f**.

Esempio:

L'esempio seguente copia il valore contenuto nel registro all'indirizzo 0CH nell'accumulatore W:

```
movf    0CH,W
```

MOVWF — MOVe W to F

Muove il contenuto del registro W nel registro F

Sintassi:

```
movwf   f
```

Operazione equivalente:

f = W

Descrizione:

Questa istruzione copia il contenuto del **registro W** nel registro indirizzato dal parametro **f**.

Esempio:

Ipotizziamo di voler scrivere il valore **10H** (esadecimale) nel registro **TMR0**. Le istruzioni da eseguire sono le seguenti.

```
movlw   10H     ;Scrive nel registro W il valore 10H
movwf   01H     ;e lo memorizza nel registro TMR0
```

Per i registri utilizzati dal PIC per funzioni specifiche, solitamente non viene inserito direttamente l'indirizzo ma il relativo nome simbolico definito nel file P16F84.INC. Il codice diventa quindi il seguente:

```
movlw   10H     ;Scrive nel registro W il valore 10H
movwf   TMR0    ;e lo memorizza nel registro TMR0
```

Note:

L'esecuzione della **MOVWF** non influenza nessun bit di stato.

NOP — No OPeration

Nessuna operazione

Sintassi:

```
nop
```

Descrizione:

Questa istruzione non esegue nessuna operazione ma è utile per inserire ritardi pari ad un ciclo macchina .

Esempio:

Utilizzando un quarzo da **4MHz** potremo ottenere un ritardo pari ad **1µs** per ogni istruzione **NOP** inserita nel nostro source..

```
nop   ;Esegue un ritardo pari ad 1µs
```

Note:

La **NOP** non influenza nessun bit di stato.

OPTION — load OPTION register

Assegna il valore in W al registro OPTION

Sintassi:

```
option
```

Operazione equivalnte:

OPTION = W

Descrizione:

Questa istruzione memorizza nel registro speciale OPTION il valore contenuto nell'accumulatore W.

Esempio:

```
      org     00H

start
      movlw   01000100B
      option
      ...
```

Note:

Questa istruzione esiste per mantenere la compatibilità con i PIC prodotti finora, la Microchip ne sconsiglia l'uso. In alternativa è consigliabile usare le seguenti istruzioni.

```
      org           00H
```

```
start
        bsf     STATUS,RP0 ;Attiva il banco registri 1

        movlw   01000100B
        movwf   OPTION_REG
        ...
```

In pratica si consiglia di scrivere direttamente nel registro OPTION presente nel banco 1 dei registri del PIC utilizzando la MOVWF anziche l'istruzione OPTION che in futuro potrebbe non essere più implementata.

Questa istruzione non influenza nessun bit di stato.

RETFIE — RET From Interrupt

Ritorna da una subroutine

Sintassi:

```
retfie
```

Descrizione:

Questa istruzione deve essere inserita al termine di ogni subroutine di gestione degli interrupt per ridare il controllo al programma principale.

Esempio:

```
        org     00H
loop
        goto    loop  ;Loop infinito

        org     04H   ;Interrupt vector
intHandler

        retfi         ;Ritorna dall'interrupt
```

In questo source il programma principale esegue un loop infinito. Se abilitiamo uno degli interrupt del 16F84 non appena esso si verificherà il controllo verrà dato automaticamente al programma allocato dall'indirizzo 04H (nell'esempio intHandler), l'istruzione RETFI determinerà quindi il ritorno al loop principale.

Note:

Questa istruzione non influenza alcun bit di stato.

RETLW — RETurn Literal to W

Ritorna da una subroutine con una costante in W

Sintassi:

```
retlw     k
```

Descrizione:

Questa istruzione ritorna il controllo da una subroutine al programma principale. A differenza dell'istruzione RETURN essa consente di passare, tramite l'accumulatore **W**, il valore costante **k** al programma principale.

Esempio:

```
rtc     equ     0CH
        org     00H

        call    mySub1
        movwf   rtc

        ...

mySub1
        nop
        retlw   10
```

Una volta eseguito questo programma memorizza nel registro rtc il valore 10 passato dalla subroutine mySub1.

Note:

Questa istruzione non influenza alcun bit di stato

Vedi anche l'istruzione RETURN.

RETURN — RETURN from subroutine

Ritorna da una subroutine

Sintassi:

```
return
```

Descrizione:

Questa istruzione deve essere inserita al termine di ogni subroutine per riprendere l'esecuzione del programma principale.

Esempio:

```
        org     00H

        call    mySub1
        ....

mySub1
        nop
        return
```

Note:

Nel PIC16F84 possono essere annidate fino ad 8 chiamate a subroutine del tipo:

```
        org     00H

        call    mySub1
        ....

mySub1
        call    mySub2
        return

mySub2
        call    mySub3
        return

mySub3
        return
```

Note:

Questa istruzione non influenza alcun bit di stato.

Vedi anche l'istruzione RETLW.

RLF — Rotate Left F through carry

Ruota a sinistra il contenuto del registro f passando per il Carry

Sintassi:

```
rlf     f,b
```

Operazione equivalente:

d = f << 1 (dove d può essere W o f)

Descrizione:

Ruota i bit contenuti nel registro all'indirizzo **f** verso sinistra (ovvero dai bit meno significativi verso quelli più significativi) passando per il bit **CARRY** del registro **STATUS** come illustrato in figura:

Il contenuto del bit CARRY del registro status viene spostato nel bit D0 mentre il valore in uscita dal bit D7 viene spostato nel CARRY.
Il valore del parametro **d** determina la destinazione del risultato ottenuto al termine della rotazione: Per **d = W** il risultato viene memorizzato nel **registro W** lasciando il registro f invariato.Per **d = F** il risultato viene memorizzato nello stesso **registro f**

Esempio:

```
parm1   equ     0CH

        org     00H

        clrf    C,STATUS    ;Azzera il CARRY

        movlw   01010101B   ;Valore iniziale
        movwf   parm1

        rlf     parm1,F
```

Al termine del programma il registro **parm1** varrà **10101010B** mentre il CARRY varrà 0.

Note:

Questa istruzione non influenza nessun altro bit di stato oltre al CARRY.

RRF — Rotate Right F through carry

Ruota a destra il contenuto del registro f passando per il Carry

Sintassi:

```
rrf     f,b
```

Operazione equivalente:

d = f >> 1 (dove d può essere W o f)

Descrizione:

Ruota i bit contenuti nel registro all'indirizzo **f** verso destra (ovvero dai bit più significativi verso quelli meno significativi) passando per il bit **CARRY** del registro **STATUS** come illustrato in figura:

Il contenuto del bit CARRY del registro status viene spostato nel bit D7 mentre il valore in uscita dal bit D0 viene spostato nel CARRY.

Il valore del parametro **d** determina la destinazione del risultato ottenuto al termine della rotazione:

Per **d = W** il risultato viene memorizzato nel **registro W** lasciando il registro f invariato.

Per **d = F** il risultato viene memorizzato nello stesso **registro f**

Esempio:

```
parm1   equ     0CH

        org     00H

        clrf    C,STATUS    ;Azzera il CARRY

        movlw   01010101B   ;Valore iniziale
        movwf   parm1

        rrf     parm1,F
```

Al termine del programma il registro **parm1** varrà **00101010B** mentre il CARRY varrà 1.

Note:

Questa istruzione non influenza nessun altro bit di stato oltre al CARRY.

SLEEP	go into standby mode
Mette il PIC in standby	

Sintassi:

```
sleep
```

Descrizione:

Questa istruzione blocca l'esecuzione del programma in corso e mette il PIC in stato di standby (sleep dall'inglese to sleep, dormire).

Esempio:

```
        org     00H

start
        sleep
```

Note:

Questa istruzione non influenza nessun bit di stato.

SUBLW	SUBtract W from Literal
Sottrae a k il valore in W	

Sintassi:

```
sublw   k
```

Operazione equivalente:

W = k - W

Descrizione:

Sottra alla costante **k** il valore memorizzato nell'accumulatore **W**.

Esempio:

```
        org     00H

start
        movlw   10
        sublw   12
        ...
```

Dopo aver eseguito questo programma l'accumulatore W varrà 2.

Note:

Questa istruzione influenza i bit **Z**, **DC** e **C** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.
- **DC** vale 1 se il risultato dell'operazione è un numero superiore a 15.
- **C** vale 1 se il risultato è positivo ovvero se il bit 7 del registro contenente il risultato vale 0 e 1 se il risultato è negativo ovvero se il bit 7 del registro contenente il risultato vale 1.

SUBWF	SUBstract W from F
Sottrae il valore contenuto in W dal valore contenuto nel registro F	

Sintassi:

```
subwf   f,d
```

Operazione equivalente:

d = f - W (dove d può essere W o f)

Descrizione:

Questa istruzione sottrae il valore contenuto nel **registro W** dal valore contenuto nel registro indirizzato dal parametro **f**. Il parametro **d** è un flag che indica su quale registro deve essere memorizzato il risultato.

Per **d = W** il risultato viene memorizzato nel **registro W**
Per **d = F** il risultato viene memorizzato nel **registro f**

Esempio:

Analizziamo un esempio estratto dal datasheet della Microchip:

Se inseriamo l'istruzione:

```
subwf   REG1,F
```

Dove **reg1** è l'indirizzo di un qualsiasi registro specificato con la direttiva:

```
REG1    RES     1
```

Per valori iniziali di REG1=3 e W=2, dopo l'esecuzione avremo REG1=1 e C=1 in quanto il risultato è positivo.

Per valori iniziali di REG1=2 e W=2 dopo l'esecuzione avremo REG1=0 e C=1 perché il risultato è sempre positivo.

Per valori iniziali di REG1=1 e W=2, avremo REG1=FFH ovvero -1 quindi C=0 perchè il risultato è negativo.

Note:

Questa istruzione influenza i bit **Z**, **DC** e **C** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.
- **C** vale 1 se il risultato è positivo ovvero se il bit 7 del registro contenente il risultato vale 0 e 1 se il risultato è negativo ovvero se il bit 7 del registro contenente il risultato vale 1.

SWAPF — SWAP F

Scambia il valore dei quattro bit più significativi del registro all'indirizzo F con i quattro bit meno significativi

Sintassi:

```
swap    f,d
```

Operazione equivalente:

f = Swap dei bit 0123 con 4567 di f

Descrizione:

Scambia il valore del quattro bit più significativi (D7-D4) contenuti nel registro all'indirizzo f con i quattro bit meno significativi (D3-D0) dello stesso. Il risultato viene memorizzato nell'accumulatore **W** o nello stesso registro **f** in base al valore di **d**:

Per **d = W** il risultato viene memorizzato nel **registro W**
Per **d = F** il risultato viene memorizzato nello stesso **registro F**

Note:

Questa istruzione non influenza alcun bit di stato

TRIS — load TRIS register

Assegna il valore in W al registro TRIS

Sintassi:

```
tris    f
```

Operazione equivalente:

TRIS di f = W

Descrizione:

Questa istruzione memorizza in uno dei registri speciale TRIS il valore contenuto nell'accumulatore W. I registri TRIS determinano il funzionamento in ingresso e uscita delle linea di I/O del PIC. Esiste un registro TRIS per ogni porta di I/O denominato TRISA, TRISB, ecc.

Esempio:

```
        org     00H

start
        movlw   11111111B
        tris    PORTA
        ...
```

Note:

Questa istruzione esiste per mantenere la compatibilità con i PIC prodotti finora, la Microchip ne sconsiglia l'uso. In alternativa è consigliabile usare le seguenti istruzioni.

```
        org     00H

start
        bsf     STATUS,RP0      ;Attiva il banco registri 1

        movlw   11111111B
        movwf   TRISA
        ...
```

In pratica si consiglia di scrivere direttamente nei registri TRIS presenti nel banco 1 dei registri del PIC utilizzando la MOVWF anziche l'istruzione TRIS che in futuro potrebbe non essere più implementata.

Note:

Questa istruzione non influenza nessun bit di stato.

XORLW — Exclusive OR Literal with W

Effettua l'OR esclusivo tra W ed una costante k

Sintassi:

```
xorlw   k
```

Operazione equivalente:

W = W XOR k

Descrizione:

Effettua l'OR esclusivo tra il valore contenuto nell'accumulatore **W** ed il valore costante **k**.

Esempio:

```
        org     00H

start
        movlw   00000000B
        xorlw   11110000B
        ...
```

Dopo aver eseguito questo programma l'accumulatore W varrà 11110000B.

Note:

Questa istruzione influenza il bit **Z** del registro **STATUS**.

- **Z** vale 1 se il risultato dell'operazione vale 0.

XORWF — eXclusive OR W with F

Effettua l'OR esclusivo tra il valore contenuto in W ed il valore contenuto nel registro F

Sintassi:

```
xorwf   f,d
```

Operazione equivalente:

d = f XOR W (dove d può essere W o f)

Descrizione:

Questa istruzione effettua l'OR esclusivo (XOR) tra il valore contenuto nell'accumulatore **W** ed il valore contenuto nel registro indirizzato dal parametro **f**. Il parametro **d** è un flag che indica su quale registro deve essere memorizzato il risultato.

Per **d = W** il risultato viene memorizzato nel **registro W**

Per **d = F** il risultato viene memorizzato nel **registro f**

Questa istruzione influenza i bit **Z** del registro **STATUS** che varrà 1 se il risultato dell'operazione è 0.

Esempio:

Ipotizziamo di dover effettuare lo XOR tra il registro W ed il registro **REG1** da noi definito all'indirizzo **0CH** con la direttiva:

```
REG1    EQU     0CH
```

possiamo utilizzare l'istruzione **IORWF** in due forme a seconda di dove vogliamo mettere il risultato, ovvero:

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

oppure:

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

Note:

L'OR esclusivo (XOR) è un'operazione tra due bit in cui il bit risultante vale 0 se i due bit sono uguali.

Spesso lo XOR viene utilizzato nell'assembler del PIC per effettuare la comparazione tra due valori in mancanza di un'istruzione specifica.

Vediamo come:

ipotizziamo di avere un valore nel registro REG1 e di voler verificare se è uguale a 57H. Le istruzioni da eseguire sono le seguenti:

```
movlw   57H         ;W = Valore da comparare = 57H
                    ;Risultato. W = 57H
xorwf   REG1,W      ;W = W XOR REG1 Effettua lo XOR con
                    ;il valore in REG1
btfss   STATUS,Z    ;Salta l'istruzione seguente se il
                    ;risultato dello XOR vale 0, ovvero
                    ;se il valore di REG1 e' pari a 57H
goto    diverso     ;Salta se diverso da 57H
goto    uguale      ;Salta se uguale da 57H
```