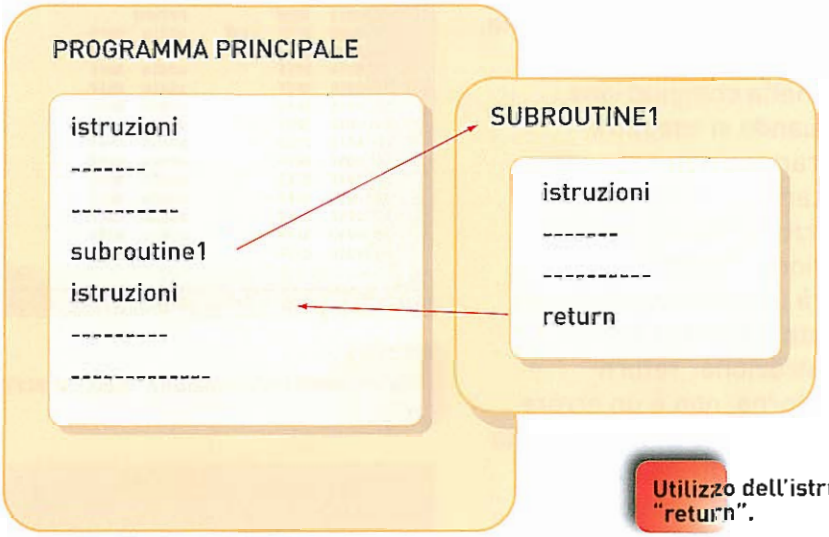
Istruzioni di ritorno

Siamo arrivati alle ultime due istruzioni di Assembler, entrambe di ritorno. Anche se sono state nominate in diverse occasioni a causa della loro relazione con altri argomenti, ora mostreremo con il simulatore ciò che succede durante l'esecuzione di queste istruzioni.

MNEMONICO	PARAMETRI	SIGNIFICATO
return		Istruzioni di ritorno da subroutine. Ritorno dal punto dal quale è stata rotta la sequenza del programma per seguire il codice della subroutine.



Utilizzo dell'istruzione "return".

Ritorno da subroutine

Quando si esegue una chiamata da subroutine, e dopo che è stato eseguito il suo codice, ci sono due istruzioni che ci permettono di tornare al punto del programma da cui è partita la chiamata.
Una è "retlw", che si utilizza quando si lavora con tabelle e della quale abbiamo già parlato; questa istruzione, quando si esegue lascia un valore sul registro di lavoro W.
L'altra è "return", che non ha parametri e il cui comportamento è mostrato nella figura.
Il suo compito è prendere dallo stack del microcontroller l'ultimo indirizzo caricato durante l'esecuzione dell'istruzione di chiamata a subroutine.

```
                LIST    P = 16F873

                INCLUDE "P16F873.INC"

                ORG     0
                goto    INIZIO          ; Indirizzo di Reset

INIZIO          bsf     STATUS,RP0
                clrf    TRISB           ; PortaB come uscita
                movlw   h'FF'
                movwf   TRISC           ; PortaC come ingresso
                bcf     STATUS,RP0

CONDIZIONE      btfss   PORTC,0         ; RC0=1?
                goto    SPEGNERE        ; No
                goto    ACCENDERE       ; Sì

SPEGNERE        call    ZERO
                goto    FINE
ACCENDERE       call    UNO
                goto    FINE

ZERO            clrf    PORTB
                return
UNO             movlw   h'FF'
                movwf   PORTB
                return

FINE
                END
```

Programma esempio di un utilizzo corretto dell'istruzione "return".

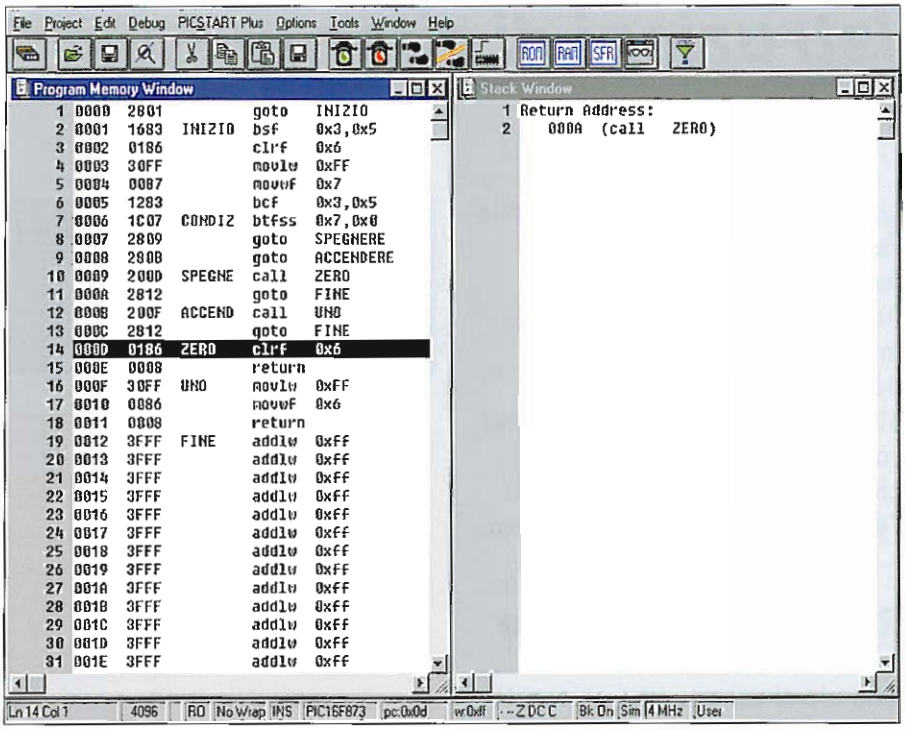
Errori comuni

Lo schema che accompagna la definizione dell'istruzione mostra la "programmazione

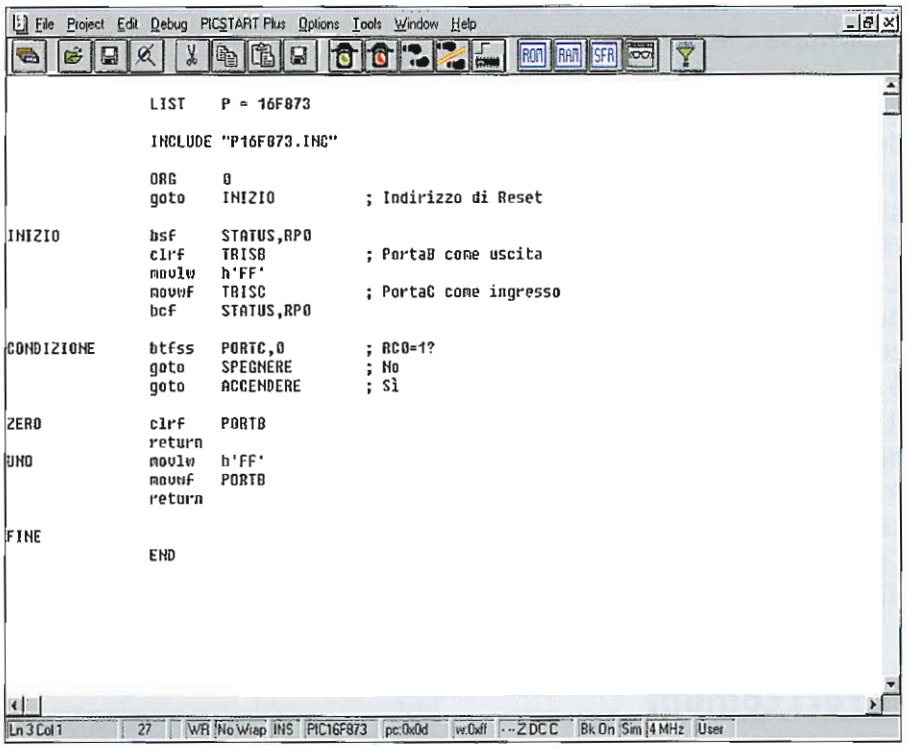
ideale", una subroutine con un unico punto d'ingresso e un unico punto di uscita, in altre parole, per tornare si chiama la subroutine con una istruzione "call" e per tornare c'è solamente una istruzione "return".

Se invece di "call" si utilizza "goto", sia per chiamare la subroutine al completo sia per accedere a una parte di essa partendo da un'etichetta, non si produrrà alcun errore nella compilazione, però quando si eseguirà il programma non verrà caricato sullo stack l'indirizzo di ritorno, quindi l'istruzione "return" riceverà un indirizzo di ritorno sbagliato. L'utilizzo di più di un'istruzione "return" per il ritorno, non è un errore in sé, e il programma risultante non dovrebbe funzionare male, ma questo contraddice le regole della programmazione strutturata, e ciò potrebbe portare a errori, soprattutto semantici, se non viene trattato con attenzione. Occorre quindi ricordare che ogni istruzione ha un suo utilizzo, e non è nemmeno corretto evitare a priori l'utilizzo dell'istruzione "goto".

Nel programma della figura quindi, alla verifica del valore della linea di una porta, le istruzioni che vengono inserite di seguito sono "goto", e queste ci portano a una parte del programma da cui non si ritorna. Quando si utilizza "call" invece, si va a una subroutine da cui si torna con un "return". Per verificare cosa succede internamente al microcontroller,



Eseguendo un'istruzione "call" si scrive un indirizzo nello stack.



Programma che utilizza un'istruzione "return" per tornare da una "goto".