

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.

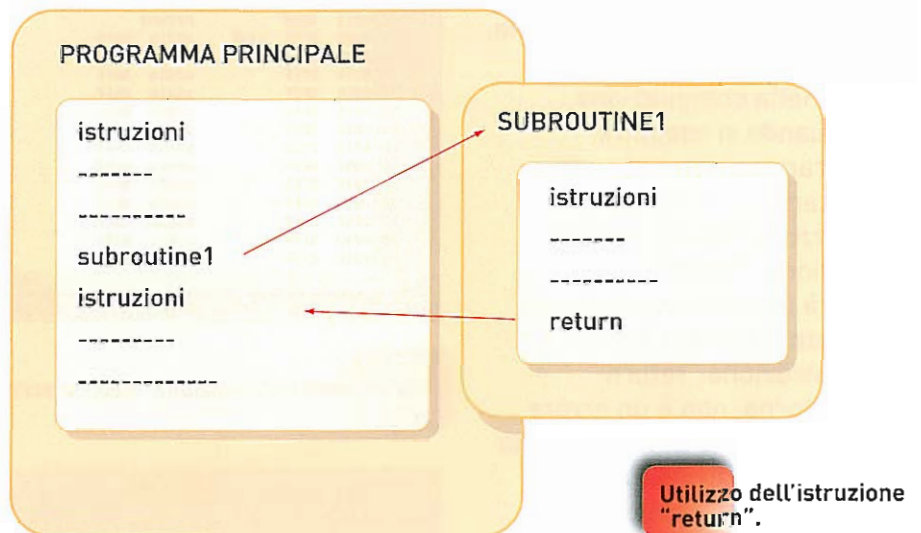
Ritorno da subroutine

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.

Errori comuni

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

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.



```

LIST      P = 16F873
INCLUDE  "P16F873.INC"

ORG      0
goto    INIZIO      ; Indirizzo di Reset

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

CONDIZIONE btfs  PORTC,0      ; RC0-1?
           goto SPEGNERE    ; No
           goto ACCENDERE   ; Sì

SPEGNERE call  ZERO
           goto FINE
ACCENDERE call  UNO
           goto FINE

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

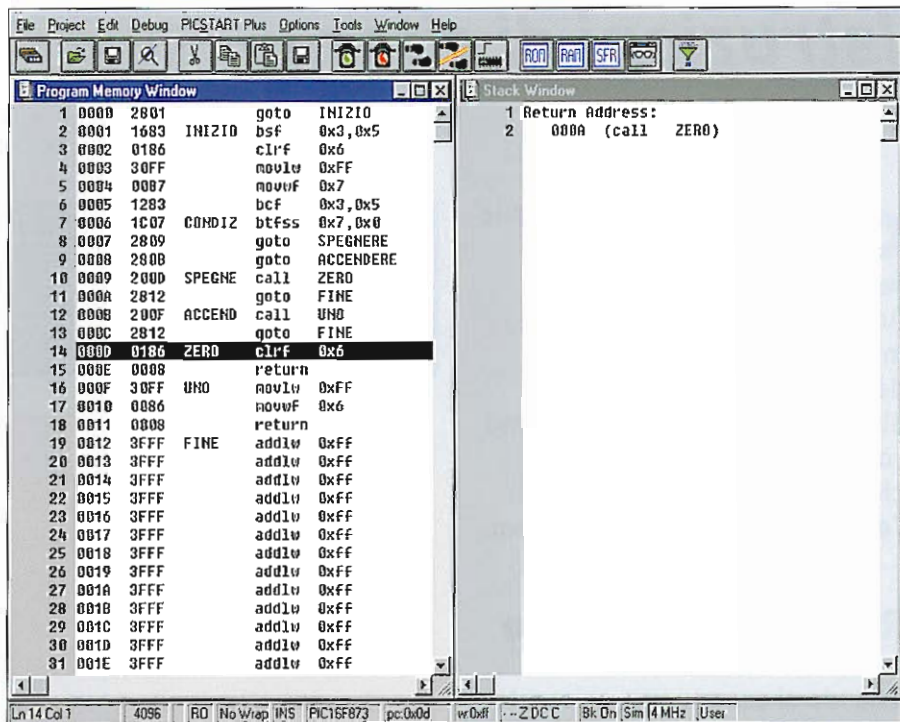
FINE
         END
    
```

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

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,

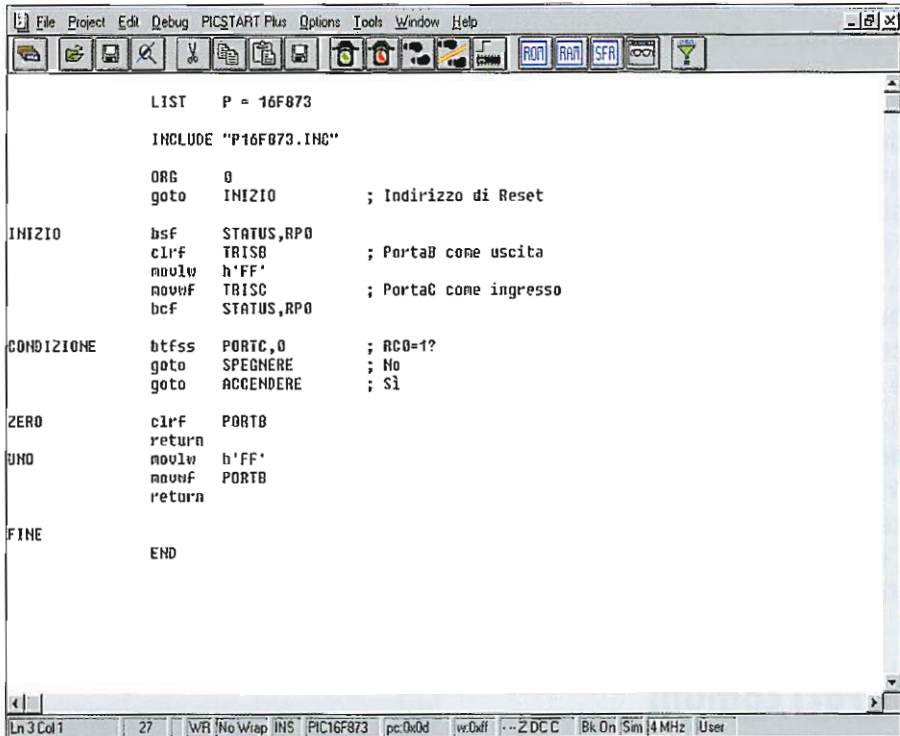


The screenshot shows the PICSTART Plus software interface. The 'Program Memory Window' displays assembly code with addresses and labels. The 'Stack Window' shows the return address stored on the stack.

Address	Label	Instruction
1 0000	2801	goto INIZIO
2 0001	1683	INIZIO bsf 0x3,0x5
3 0002	0186	clrf 0x6
4 0003	30FF	movlw 0xFF
5 0004	0087	movwf 0x7
6 0005	1283	bcf 0x3,0x5
7 0006	1C07	CONDIZ btfss 0x7,0x0
8 0007	2809	goto SPEGNERE
9 0008	280D	goto ACCENDERE
10 0009	200D	SPEGNE call ZERO
11 000A	2812	goto FINE
12 000B	200F	ACCEND call UNO
13 000C	2812	goto FINE
14 000D	0186	ZERO clrf 0x6
15 000E	0008	return
16 000F	30FF	UNO movlw 0xFF
17 0010	0086	movwf 0x6
18 0011	0008	return
19 0012	3FFF	FINE addlw 0xFF
20 0013	3FFF	addlw 0xFF
21 0014	3FFF	addlw 0xFF
22 0015	3FFF	addlw 0xFF
23 0016	3FFF	addlw 0xFF
24 0017	3FFF	addlw 0xFF
25 0018	3FFF	addlw 0xFF
26 0019	3FFF	addlw 0xFF
27 001A	3FFF	addlw 0xFF
28 001B	3FFF	addlw 0xFF
29 001C	3FFF	addlw 0xFF
30 001D	3FFF	addlw 0xFF
31 001E	3FFF	addlw 0xFF

Stack Window:
1 Return Address:
2 000A (call ZERO)

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



The screenshot shows the PICSTART Plus software interface displaying assembly code. The code includes labels for INIZIO, CONDIZIONE, ZERO, UNO, and FINE, with instructions for setting bits, moving data, and jumping.

```
LIST P = 16F873
INCLUDE "P16F873.INC"

ORG 0
goto INIZIO ; Indirizzo di Reset

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

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

ZERO clrf PORTB
      return

UNO movlw h'FF'
     movwf PORTB
     return

FINE END
```

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