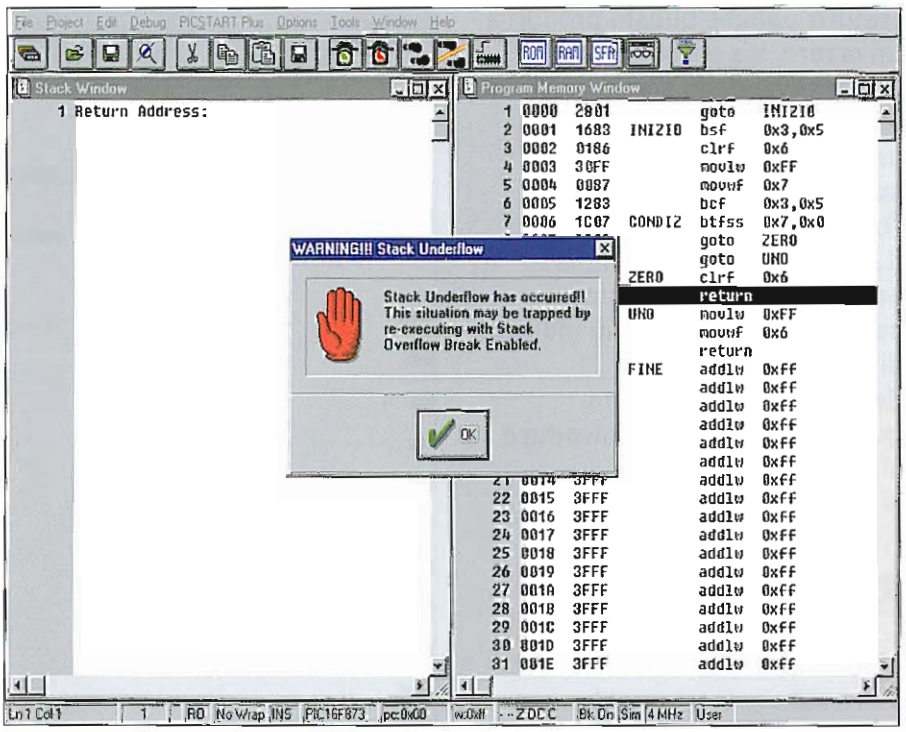
visualizzate lo stack da Window → Stack ed eseguite il programma passo a passo. Osservate che passando dalle istruzioni "goto" lo stack non scrive un indirizzo di ritorno, ma lo fa quando l'istruzione è "call". Visualizzate la finestra della memoria di programma in Window → Program → Memory, potrete in questo modo seguire l'esecuzione dei salti con i diversi indirizzi.

Cambiate ora il programma in modo che esegua un codice non corretto. Anche se compilandolo non si genera alcun errore, guardando le istruzioni vedremo che si passa da una parte all'altra del programma con istruzioni "goto" e si cerca di tornare con istruzioni "return", cosa impossibile. Eseguite questo nuovo programma passo a passo per vedere ciò che capita.

Quando l'istruzione "return" cerca di recuperare un indirizzo dallo stack che precedentemente non è stato inserito, si produce un errore e il simulatore ci avvisa. Se non utilizziamo un simulatore per provare il nostro programma, il sistema finale può funzionare male e non sapremo perché, poiché non avremo alcun tipo di segnalazione. Questo caso può risultare molto semplice da risolvere dopo aver visto che qualcosa non va. Il programma successivo, invece, produrrà un errore dello stesso tipo, ma è più complesso da trovare se non lo vediamo nella simulazione. Il caso contrario consiste nel porre più istruzioni "call" che



Il simulatore ci avvisa dei problemi con lo stack.

```
                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?
                call    ZERO    ; No
                call    UNO     ; Sì

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

FINE
                END
```

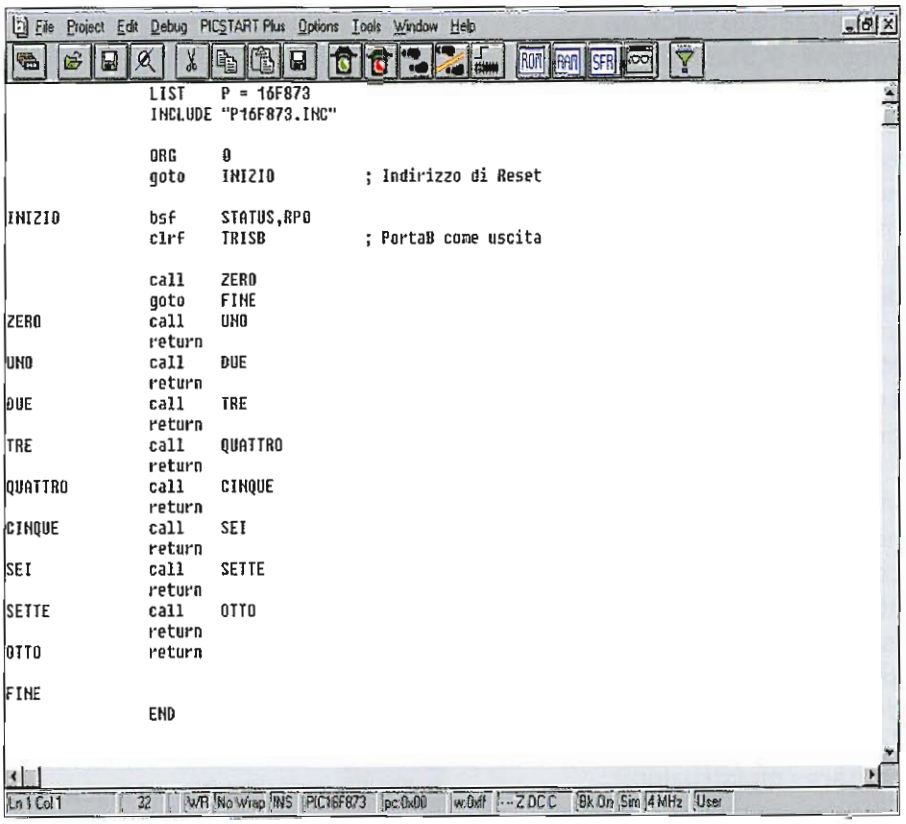
Programma con copie "call-return" utilizzate in modo scorretto.

"return", anche questo produrrà un errore, sia perché la dimensione dello stack viene superata sia perché un'istruzione "return" ritorna con un indirizzo di una "call" che non è la sua.
Il programma dell'esempio potrebbe essere corretto se non si inserissero più indirizzi di ritorno nello stack degli otto che lui accetta, quindi il primo che si introduce si perde e dopo non si potrà più tornare all'origine.

Ritorno da interrupt

Molte delle cose dette sino a ora sul comportamento dell'istruzione "return" nelle subroutine servono per l'istruzione "retfie" e gli interrupt. Questa istruzione ritorna da un interrupt per continuare il programma a partire dal punto dove questo si è prodotto (ricordate che nel caso di interrupt non c'è chiamata) e lascia il bit di abilitazione generale degli interrupt a 1 (si imposta a 0 automaticamente quando si produce l'interrupt).

Non necessita nemmeno di parametri. L'unica differenza fra "return" e " retfie" è in questa impostazione a 0 del bit GIE, quindi anche se ognuna di esse ha il proprio utilizzo, potremo sostituire l'una con l'altra nel caso ci tornasse utile.
Ad esempio, se al ritorno da un interrupt non vogliamo che se ne producano altri, potremo utilizzare "return" invece di "retfie".
Dato che per gli interrupt non c'è chiamata, non potremo confondere "goto" con "call", però sarebbe un errore accedere a un punto intermedio dell'interrupt tramite queste istruzioni.
Risulta anche difficile generare un overflow dello stack, dato che proprio l'impostazione automatica del bit GIE a 0 è fatta in modo da non generare un interrupt quando se ne sta già risolvendo un altro. Per provare tutti questi casi prendete un programma con interrupt ed eseguitelo passo a passo. Visualizzate lo stack e i registri specifici. Verificherete che è difficile fare degli errori "senza volerlo" nel caso degli interrupt.



Anche l'eccesso di istruzioni "call" può produrre errori.

MNEMONICO	PARAMETRI	SIGNIFICATO
retfie		Istruzione di ritorno da interrupt. Ritorna al punto dal quale e stata rotta la sequenza di programma per entrare nell'interrupt. Riporta il bit GIE a 1.

Utilizzo dell'istruzione "retfie".