Istruzioni di decremento/ incremento

Per ridurre il valore della variabile TEMPO1 possiamo utilizzare un'istruzione di sottrazione e verificare, testando il flag Z, se il risultato è 0, per saltare in un'altra zona del programma o continuare all'interno del ciclo. L'altra opzione simile, dato che dobbiamo sottrarre un'unità, consiste nell'utilizzare la nuova istruzione di decremento riportata nella figura. Il risultato dell'operazione verrà portato su W o sullo stesso registro di origine, a seconda del parametro D. L'operazione di incremento funziona come quella di decremento, incrementando di un'unità il valore del registro.

Questa istruzione non è necessaria nel nostro esempio.

MNEMONICO	PARAMETRI	SIGNIFICATO
decf	F, D	Decrementa di 1 il valore del registro F e porta il risultato su W (se D=0) o sullo stesso registro (se D=1).
incf	F, D	Incrementa di 1 il valore del registro F e porta il risultato su W (se D=0) o sullo stesso registro (se D=1).

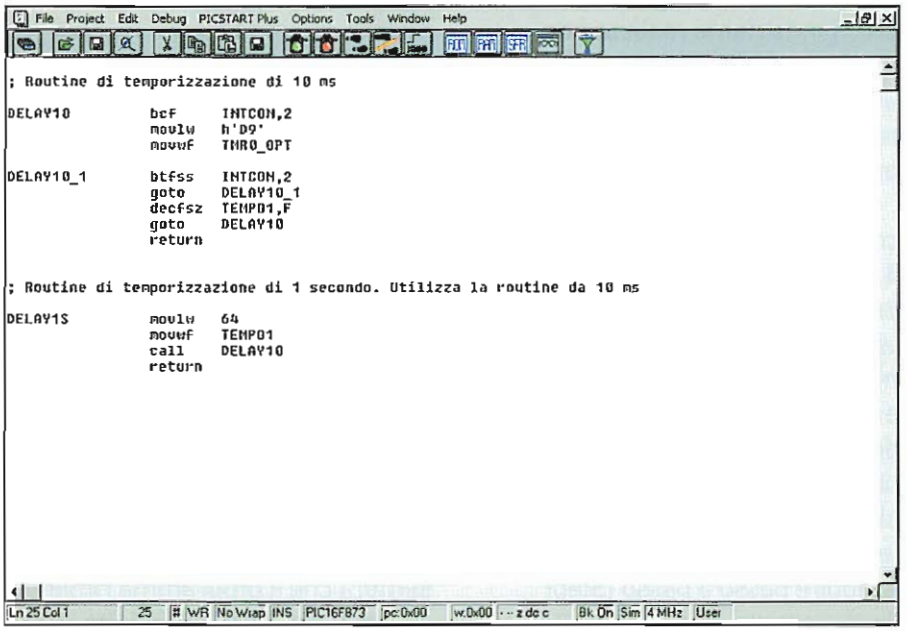
Operazioni di decremento e incremento.

MNEMONICO	PARAMETRI	SIGNIFICATO
decfsz	F, D	Decrementa di 1 il valore del registro F portando il risultato su W (se D=0) o sullo stesso registro F (se D=1). Inoltre salta un'istruzione se il risultato dell'operazione è 0.
incfsz	F, D	Incrementa di 1 il valore del registro F portando il risultato su W (se D=0) o sullo stesso registro F (se D=1). Inoltre salta un'istruzione se il risultato dell'operazione è 0.

Operazioni di decremento e incremento con salto.

Istruzione di decremento/ incremento con salto

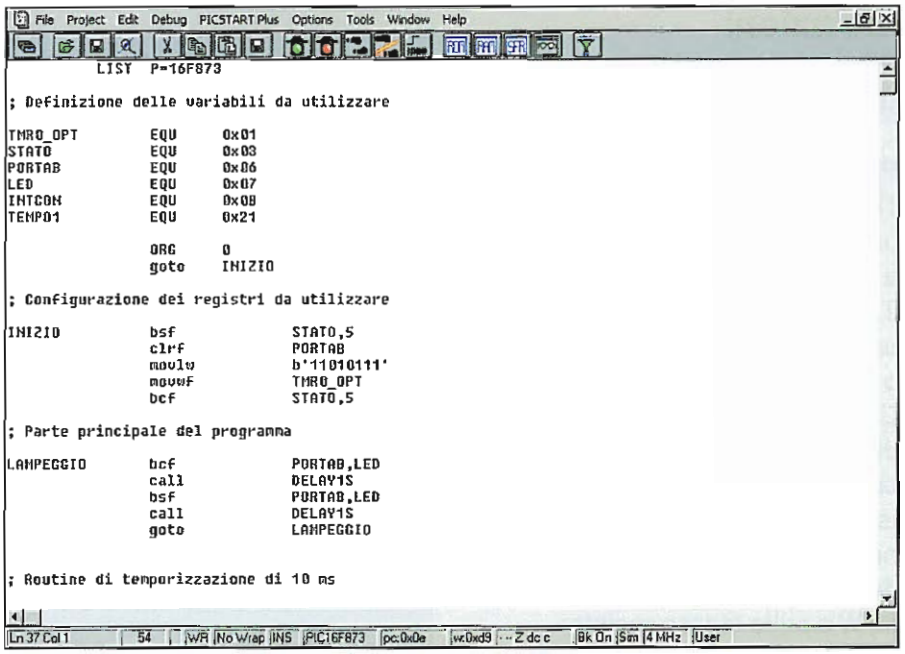
Anche se l'istruzione di decremento, che abbiamo visto in precedenza, potrebbe essere utilizzata nel nostro programma, ne esiste un'altra che facilita il lavoro, dato che oltre a decrementare il registro di una unità realizza un salto se il risultato dell'operazione è zero. Assomiglia all'istruzione che esegue il testo di un bit, infatti anche se funziona su di un dato differente, il salto è realizzato allo stesso modo: con un'unica istruzione. L'operazione di incremento con salto è come quella di decremento, però non è utile in questo caso.



Routine di temporizzazione di 1 secondo.

Programma della routine di temporizzazione di 1 secondo

Ora tradurremo gli organigrammi visti nel secondo caso in istruzioni Assembler. Dal programma principale verrà chiamata una routine di temporizzazione di 1 secondo, questa routine dovrà eseguire la chiamata a un'altra routine ausiliaria di 10 ms. Prima di poterla utilizzare dovremo definire i registri, sia quelli specifici che quelli di utilizzo generale, inserendo queste definizioni nel punto giusto del programma e, dato che stiamo utilizzando il TMR0, anche nel registro OPTION, come abbiamo già visto nella routine precedente.

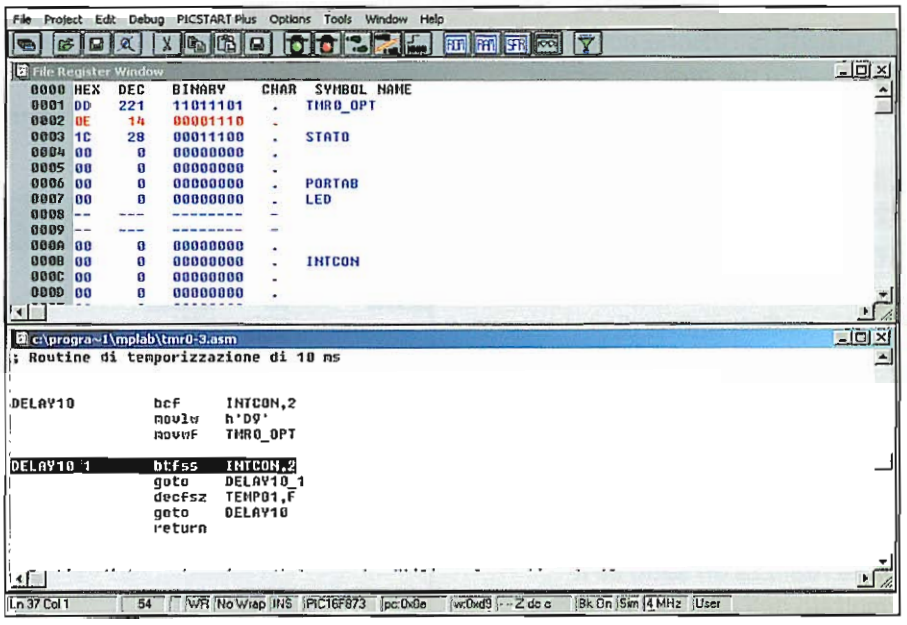


Programma di prova per le routines di temporizzazione.

Prova delle routines

Per fare le prove di funzionamento delle routines, dovremo inserirle all'interno di un programma. Quello che vi mostriamo realizza una funzione semplice: fa lampeggiare un diodo LED con una temporizzazione di un secondo fra ogni accensione e spegnimento.

Vi mostriamo solo la parte corrispondente all'inizializzazione e il programma principale. In seguito vengono le routines che abbiamo visto. Il programma le eseguirà in modo continuo, accendendo e spegnendo il bit 7 della porta B, dove si suppone sia collegato un LED. Per provare l'esempio sul simulatore è sufficiente aprire la finestra dei registri generale, ed eseguire passo a passo (Step) il programma. Vedrete che, arrivando alla routine di temporizzazione, il TMR0 sembra non incrementarsi, quindi



Videate necessarie per la simulazione.

sembra che il programma rimanga all'interno del ciclo all'infinito; questo succede perché abbiamo impostato il divisore di frequenza al suo massimo, e sono necessarie 256 istruzioni perché il TMR0 si incrementi di 1. La scelta migliore è fare la prova abbassando il valore del divisore, del TMR0 e della variabile del ciclo esterno.