Miscela di assembler e BASIC

Come avete avuto modo di verificare, il LetPicBasic ci permette di realizzare programmi in modo rapido e comodo, senza preoccuparci della struttura che il processore ha all'interno. Come contropartita a questa semplicità, man mano che avanziamo nella programmazione, ci accorgeremo che il programma sta limitando la potenza che potrebbe fornirci il microcontroller.

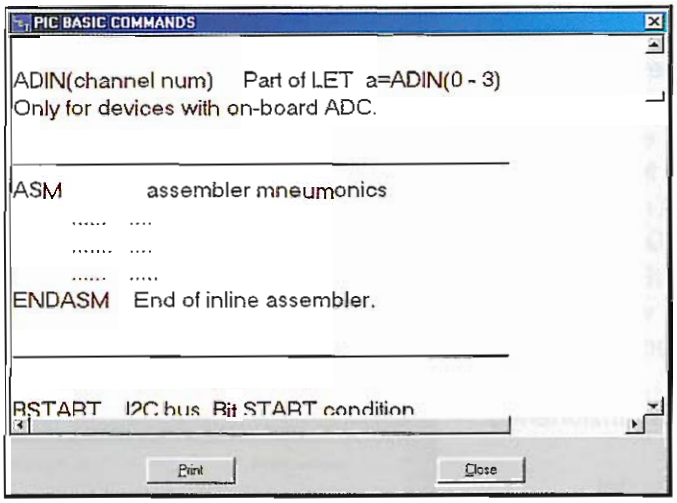
La soluzione la possiamo trovare nel linguaggio assembler.

L'assembler dei PIC

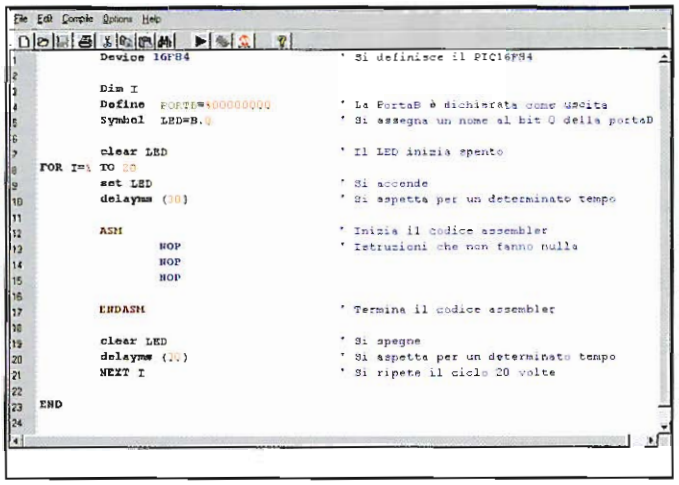
L'assembler è il linguaggio più vicino al codice macchina.
A volte è considerato più difficile, però in alcuni casi permette maggior libertà. Nei PIC l'insieme delle istruzioni è progettato per gestirne al meglio la struttura interna, che viene chiamata architettura RISC, si tratta quindi di un insieme di istruzioni ridotto e semplice. Nello specifico, per il modello di PIC che utilizzeremo, avremo a disposizione 35 istruzioni. Poco a poco faremo conoscenza di ognuna di esse e le applicheremo a nuovi programmi.

Istruzioni assembler all'interno di programmi BASIC

Una soluzione intermedia può essere introdurre istruzioni

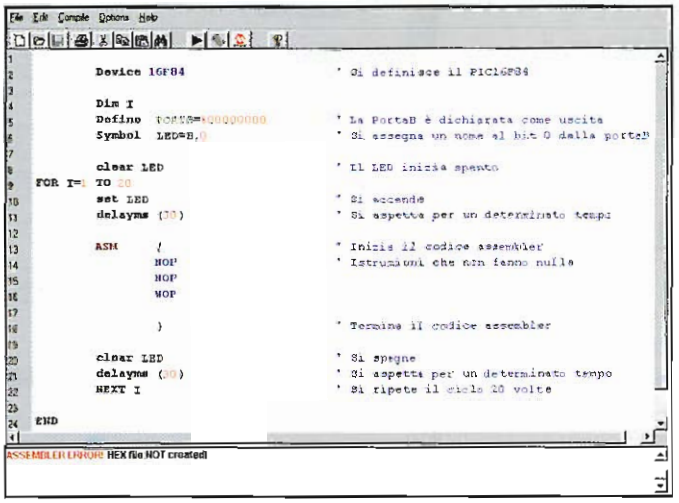


Modo generale di inserire istruzioni assembler secondo l'help in linea del programma.



```
Device 16F84                    ' Si definisce il PIC16F84

Dim I
Define  PORTB=%00000000         ' La PortaB è dichiarata come uscita
Symbol  LED=B.0                 ' Si assegna un nome al bit 0 della portaB

clear LED                       ' Il LED inizia spento
FOR I=1 TO 20
set LED                         ' Si accende
delayms (50)                    ' Si aspetta per un determinato tempo

ASM                             ' Inizia il codice assembler
        NOP                     ' Istruzioni che non fanno nulla
        NOP
        NOP

ENDASM                          ' Termina il codice assembler

clear LED                       ' Si spegne
delayms (50)                    ' Si aspetta per un determinato tempo
NEXT I                          ' Si ripete il ciclo 20 volte

END
```

Programma esempio di inserimento di istruzioni assembler.



```
Device 16F84                    ' Si definisce il PIC16F84

Dim I
Define  PORTB=%00000000         ' La PortaB è dichiarata come uscita
Symbol  LED=B.0                 ' Si assegna un nome al bit 0 della portaB

clear LED                       ' Il LED inizia spento
FOR I=1 TO 20
set LED                         ' Si accende
delayms (50)                    ' Si aspetta per un determinato tempo

ASM     {                       ' Inizia il codice assembler
        NOP                     ' Istruzioni che non fanno nulla
        NOP
        NOP

        }                       ' Termina il codice assembler

clear LED                       ' Si spegne
delayms (50)                    ' Si aspetta per un determinato tempo
NEXT I                          ' Si ripete il ciclo 20 volte

END
ASSEMBLER ERROR! HEX file NOT created!
```

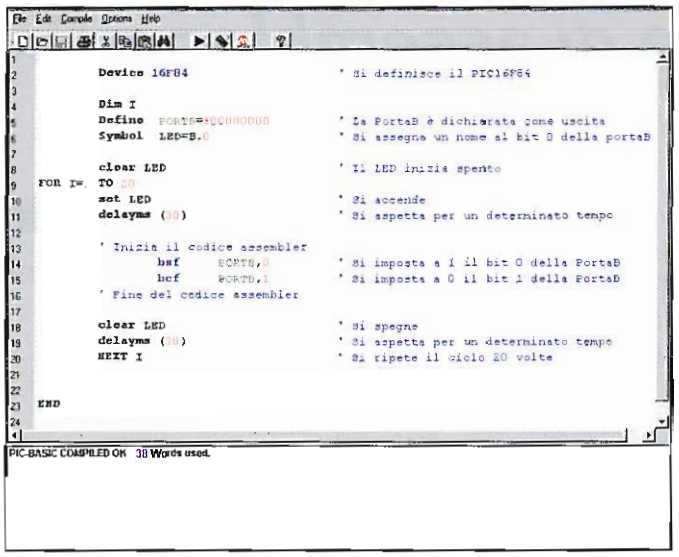
L'utilizzo di parentesi graffe non è accettato in questa versione.

assembler all'interno dei programmi BASIC che già conosciamo. In questo modo uniamo la facilità del Basic alla potenza dell'assembler. Il modo di farlo è semplice così come spiega l'help in linea del vostro programma. Scegliete uno qualsiasi dei programmi che avete già realizzato e inserite qualche istruzione. Nell'esempio abbiamo utilizzato l'istruzione NOP diverse volte. Questa istruzione non fa nulla, di solito è utilizzata quando si vuole perdere qualche ciclo, e il tempo di questo ciclo dipenderà dal clock con cui sta funzionando il microcontroller.

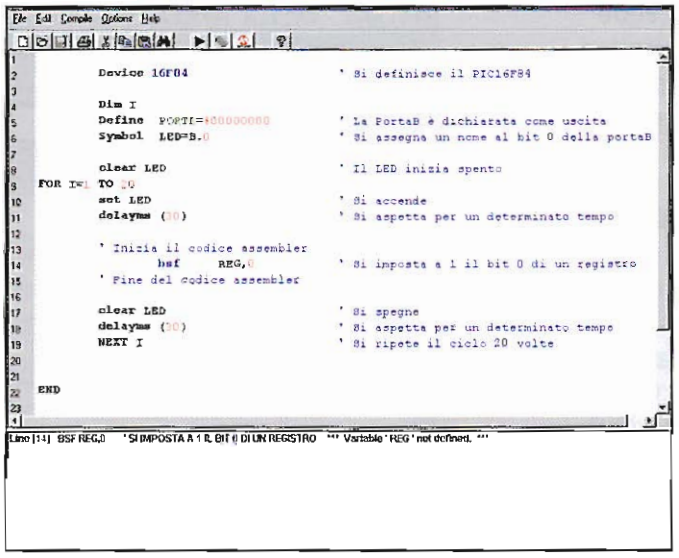
Nell'aiuto esterno del LetPicBasic è spiegato anche un altro modo di inserire codice, però se provate con la vostra versione del programma vedrete che non sarà accettato. Il terzo modo di inserire codice, anch'esso corretto, consiste nell'utilizzare le istruzioni tali e quali.
La differenza dalla prima opzione è che nel primo caso le istruzioni sono passate direttamente all'assembler senza prima essere compilate, invece il codice introdotto direttamente all'interno del BASIC viene compilato e deve seguire le stesse norme che regolano il BASIC. Quindi se non definiamo un registro prima di utilizzarlo causeremo un errore.

Risultato dell'unione

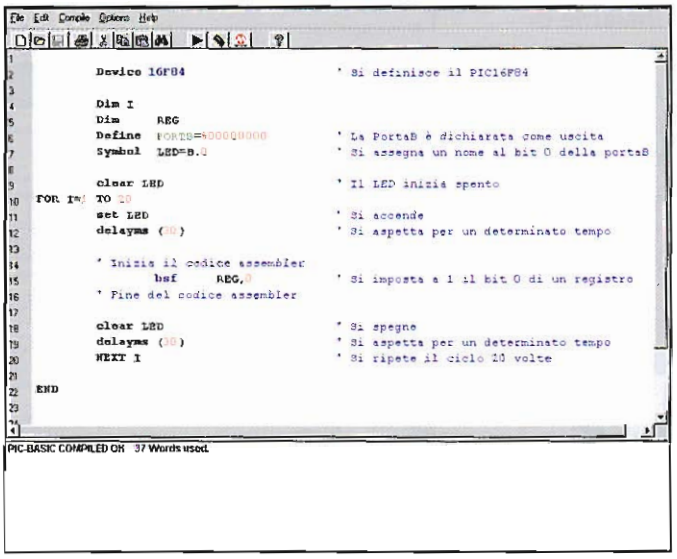
Nell'uno o nell'altro modo, dopo aver introdotto le istruzioni assembler all'interno del codice BASIC, il compilatore si farà carico di unificarle tutte passandole in assembler. Il file generato avrà



Un altro modo di inserire codici assembler.



Se non si definiscono i registri si genera un errore.



Modo corretto di utilizzare le istruzioni.