Concetti generali per la programmazione in assembler

Sinora abbiamo lavorato con il linguaggio basic, che comprendeva diverse norme comuni a tutti i linguaggi e altre specifiche per detto linguaggio. Ora inizieremo a lavorare in assembler, e anche qui succede la stessa cosa; vediamo queste norme, o concetti, prima di passare alle istruzioni.

Parti di un programma

Così come per il basic, in un programma assembler troveremo delle subroutines che contengono istruzioni, variabili e costanti, tuttavia non disporremo di strutture di controllo come tali, ma le dovremo costruire partendo dalle istruzioni.

Inoltre avremo a disposizione un tipo particolare di istruzioni, chiamate direttive, che sono "ordini" per il compilatore. Continueremo ad aver bisogno degli organigrammi come per il linguaggio precedente, anche se saranno leggermente diversi per adattarsi alle istruzioni assembler.

Direttive di inizio

La prima direttiva che troviamo in un programma assembler è sempre la direttiva con la quale comunichiamo al compilatore il tipo di PIC che vogliamo utilizzare. Nella figura abbiamo scelto il PIC16F84,

```
LIST    P=16F84        ; Direttive
RADIX   HEX
```

Prime direttive del programma.

ponendo come parametro "P=16F84"; questa è un'informazione legata al tipo di PIC, così ad esempio per un PIC16F870 bisognerà scrivere "P=16F870". Le direttive sono ordini per il compilatore, in modo che vengano eseguite prima dell'esecuzione del programma. Il compilatore ha bisogno di conoscere il tipo di PIC per verificare se le risorse utilizzate nel programma, come il numero dei registri e gli indirizzi delle istruzioni, sono impiegate in modo adeguato, e in caso contrario genera un errore corrispondente al momento della compilazione.

La direttiva successiva riportata nell'esempio, si utilizza per gli assembler di MS-DOS, per determinare il sistema di numerazione utilizzato — in questo caso esadecimale, grazie — al parametro HEX. Comunque, dato che utilizzeremo un ambiente di sviluppo inserito in Windows (MPLAB), disporremo di menù per cambiare comodamente questa caratteristica senza la necessità di inserirlo nel programma.

Etichette: variabili e costanti

Continuando troveremo le "Etichette" che seguono il formato "nome EQU valore". EQU è una direttiva che indica al compilatore di sostituire la parola "nome" nel suo

```
STATO   EQU   03    ; Etichette
Z       EQU   02
PORTAA  EQU   05
TRISA   EQU   05
AUX     EQU   0C
```

Definizione delle etichette di un programma.

corrispondente "valore" all'interno del programma. Il vantaggio che forniscono è quello di poter utilizzare dei nomi (etichette) durante la programmazione al posto dei numeri, ciò faciliterà la leggibilità del codice e la scrittura di quest'ultimo. Comunque non possiamo stabilire se queste etichette sono variabili o costanti, tutto dipende da come vengono utilizzate.

Ad esempio, se si utilizza un'etichetta che fa riferimento a un registro, la si può considerare una variabile, però se ciò che si utilizza è un valore letterale o il numero di un bit, l'etichetta sta funzionando come una costante. Nell'assembler si lavora con due dimensioni di dati: il bit e il byte (8 bit), quindi le etichette rappresentano numeri di una di queste due dimensioni. Non essendo necessario specificarlo, una stessa definizione si può utilizzare sia per un bit che per un byte, anche se questo può creare confusione. "DATO EQU 03" può essere il bit 3 di un registro o la posizione della memoria, cioè il registro 3. Una cosa da tenere presente è il codice specificato in MPLAB, poiché nel caso di codice decimale il registro "10" non sarà la stessa cosa del registro "10" riferito a codice esadecimale. In questo caso si lavora normalmente in esadecimale, e quando si ha bisogno di un altro tipo di codice lo dovremo specificare in modo esplicito. Se le etichette sono utilizzate per denominare dei registri, conviene che questi siano inizializzati prima del loro uso, dato che nel caso di un Reset di sistema, secondo il tipo di Reset, i registri possono subire delle inizializzazioni non gradite, comunque nel caso non vengano inizializzati non si genera per questo un errore.

Posizione nella memoria di programma

Ora ci occuperemo delle direttive che ci permettono di posizionare le istruzioni. L'utilizzo di questo tipo di direttiva è obbligatorio in assembler, e lungo il corso del programma è possibile utilizzare più direttive di questo tipo se è necessario. ORG indica al compilatore che l'istruzione successiva viene posizionata all'indirizzo di memoria specificato come parametro. Ci sono due istruzioni la cui collocazione all'interno della memoria è necessaria: l'istruzione di inizio del programma e la prima istruzione della routine di interrupt. Entrambe debbono occupare sempre le stesse posizioni, che sono la 0 e la 4 rispettivamente.

Come abbiamo già detto, questo avviene al momento della compilazione, in modo che la memoria di programma con le definizioni dell'esempio

```
ORG     0       ; Posizionamento primario
goto    inizio  ; istruzione

ORG     4
goto    inter   ; Vector di interrupt

ORG     5       ; Posizione di inizio
```

Utilizzo delle direttive per posizionare le istruzioni.

```
0   goto inizio
4   goto inter
5   inizio _____
        -----
    inter _____
        -----
    routine _____
        -----
```

Aspetto della memoria delle istruzioni dopo aver compilato il programma.