Lezione 1 – Linguaggio Assembly LC-2

Architettura degli elaboratori

Modulo 2 – Linguaggio macchina

Unità didattica 3 – Supporti allo sviluppo di programmi per la CPU LC-2

Nello Scarabottolo

Università degli Studi di Milano - Ssri - CDL ONLINE

Motivazioni del linguaggio Assembly

- La programmazione in linguaggio macchina è estremamente scomoda e di difficile manutenzione.
- Il calcolo degli indirizzi fisici di memoria è a sua volta macchinoso e difficile.
- Ogni istruzione macchina ha un codice mnemonico definito in fase di progettazione dell'ISA.
- Anche i registri GPR hanno codici simbolici.

Definiamo un linguaggio con corrispondenza biunivoca fra istruzione mnemonica e istruzione macchina: il linguaggio Assembly.

Struttura del linguaggio Assembly

Label Opcode Operands ;comments

Label riferimento simbolico scelto dal programmatore per indicare l'indirizzo di memoria dell'istruzione.

Opcode codice mnemonico dell'istruzione (**ADD**, **JSR**, ...).

Operands riferimenti simbolici a registri o indirizzi di memoria.

comments testo libero di spiegazione del significato dell'istruzione.

Esempio:

```
loop:   LDR     R1, R0, #0   ;legge prossimo numero
```

Pseudo-istruzioni: cosa sono

La traduzione da linguaggio Assembly a codice macchina viene effettuata automaticamente da un opportuno programma (Assembler).

Può essere utile fornire all'Assembler opportune direttive, che guidino tale traduzione automatica.

Le direttive si presentano come istruzioni macchina (da cui il nome pseudo-istruzioni) ma con l'Opcode preceduto dal carattere 'punto'.

Pseudo-istruzioni: quali sono

.orig	consente di segnalare all'Assembler da quale indirizzo di memoria caricare il programma.
.fill	consente di inizializzare con un valore costante il contenuto di una cella di memoria.
.blkw	consente di riservare un certo numero di celle di memoria (per es. per contenere variabili).
.stringz	consente di inizializzare una sequenza di celle di memoria con la codifica ASCII di una frase (racchiusa fra "doppi apici"). Dopo l'ultimo carattere, viene inserito il terminatore 0.
.end	segnala all'Assembler il termine del programma da tradurre.

Costanti

Per inserire valori costanti, si possono utilizzare le seguenti notazioni:

Binaria **bBBBBBBBBBBBBBBB**

16 bit codificati come **0** o **1**, preceduti dal carattere 'b'.

Esadecimale **xEEEE**

4 cifre esadecimali (**0**÷**9**, **a**, **b**, **c**, **d**, **e**, **f** minuscole o maiuscole) precedute dal simbolo 'x'.

Decimale **#DDDDD**

fino a 5 cifre decimali, precedute dal carattere '#', eventualmente seguito dal segno '-'.

Un esempio

Ritorniamo al secondo programma presentato nella Lez.3, U.D.2 di questo modulo (l'esempio sulle istruzioni di controllo):

```
        .orig   x3000
        LEA     R0, table       ;carica inizio tabella in punt.
        AND     R2, R2, #0      ;azzera totalizzatore
loop:   LDR     R1, R0, #0      ;legge prossimo numero
        BRZ     finish          ;se nullo ha finito
        ADD     R2, R2, R1      ;somma a totalizzatore
        ADD     R0, R0, #1      ;incrementa puntatore
        BRNZP   loop            ;prossimo numero
finish  ST      R2, result      ;scrive risultato in memoria
table   .blkw   #4              ;vettore di numeri
        .fill   #0
result  .blkw   #1              ;risultato
        .end
```

In sintesi...

- Il linguaggio Assembly: un supporto alla codifica di programmi in linguaggio macchina.
- L'Assembler: un programma di traduzione automatica da linguaggio Assembly a codice macchina.
- Le pseudo-istruzioni per guidare la fase di traduzione.

Vediamo i supporti effettivamente disponibili...

Chiusura

Fine della lezione