Prof. Giuseppe Ascia

# Linguaggi, compilatori e interpreti

# Il codice macchina

Prof. Giuseppe Ascia

- Ciascun calcolatore ha un ampio insieme di istruzioni che è in grado di eseguire.
- Le istruzioni vengono rappresentate mediante sequenze di bit

| 001000100011 | 0101010111 | 1100101001 |
|---|---|---|
| Codice operativo | Operando 1 | Operando 2 |

- Poiché il numero di istruzioni in un programma possono essere anche migliaia è assai facile commettere errori
- Scrivere un programma in linguaggio macchina richiede molto tempo.

***Quale soluzione adottare per semplificare***

***la produzione del software ?***

# Il linguaggio assembly

Prof. Giuseppe Ascia

- Costituisce la prima soluzione adottata.
- Ancora oggi è utilizzato per realizzare in modo ottimizzato alcune parti di programmi.
- Le istruzioni, gli indirizzi e i dati vengono rappresentati in forma simbolica.

| Codice operativo | Oper. 1 | Oper. 2 |
|---|---|---|
| 001000100011 | 0001 | 1001 0000 0000 1111 |
| ↓ | ↓ | ↓ |
| LOAD | R1 | 900Fh |

- LOAD R1 , 900Fh  Carica nel registro R1 il contenuto della locazione di indirizzo 900Fh

# Il linguaggio assembly

Prof. Giuseppe Ascia

- Ogni processore ha un proprio insieme di istruzioni.
- Non esiste un unico ***Linguaggio Assembly*** per tutti i possibili processori.
- Ogni processore ha il suo *Linguaggio Assembly*.

# Esempio di Linguaggio Assembly

Prof. Giuseppe Ascia

- 
- LOAD $R_i$, IND; Lettura dall' indirizzo IND a reg. $R_i$
- STORE IND, $R_i$; Scrittura dal registro $R_i$ all'indirizzo IND
- ADD $R_K$, $R_i$, $R_j$ Somma il contenuto dei reg. $R_i$ e $R_j$ in $R_k$

**Esempio di programma**

Si vuole realizzare la somma di due grandezze contenute in memoria agli indirizzi IND1e IND2.

Il risultato deve essere conservato all'indirizzo IND3:

# Esempio di Linguaggio Assembler

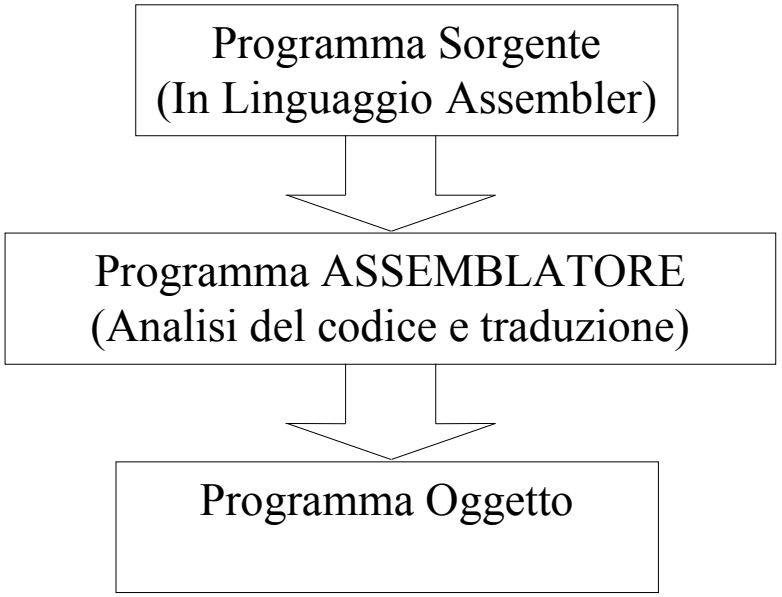


```
PROGRAMMA
 .data

IND1:   .word 10/H
IND2:   .word 05/H
IND3:   .space 4

.text
LOAD R1,IND1;      Copia in R1 da IND1
LOAD R2,IND2;      Copia in R1 da IND1
ADD R3, R1, R2;    Somma R1, R2 in R3
STORE IND3, R3;    Copia da R3 in IND3;
```

# ASSEMBLER



- Un programma scritto in un linguaggio assembly non è eseguibile dalla macchina.
- È necessario un traduttore dal linguaggio assembly al linguaggio della macchina.
- Il traduttore è un programma chiamato ASSEMBLER.



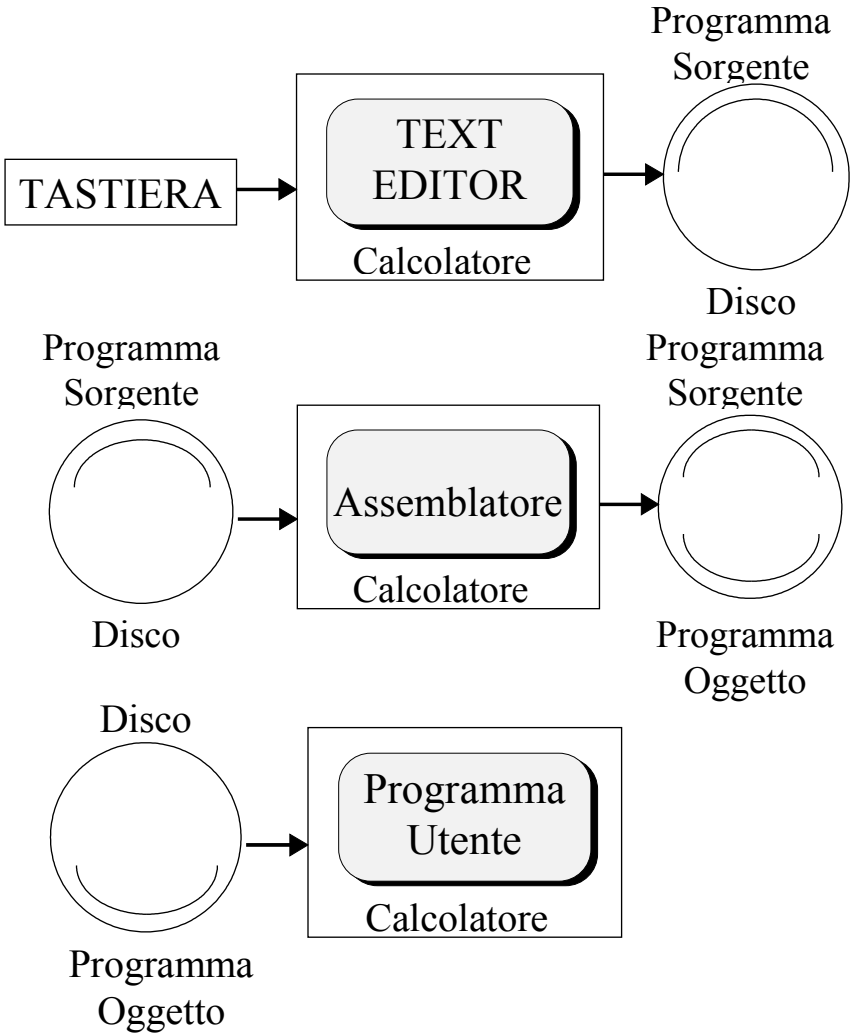
# Fasi di sviluppo di un programma in Assembly

Prof. Giuseppe Ascia

1) Creazione del sorgente e salvataggio su disco

2) Traduzione del sorgente nel programma oggetto

3) Caricamento in memoria ed esecuzione



# I linguaggi di alto livello



**Svantaggi dell'Assembly**:

- è legato al tipo di calcolatore, non *trasportabilità*;
- scomodità nella scrittura dei programmi;
- facilità nel commettere errori;
- scomodità nella gestione dei dispositivi di I/O;

Per superare queste ed altre difficoltà sono stati definiti dei **Linguaggi ad alto livello**.

Essi permettono

- di gestire in modo semplice le operazioni di I/O;
- di usare un linguaggio più vicino a quello naturale;
- di produrre programmi in modo più veloce e con minori errori;
- di ottenere programmi più leggibili.

# Traduzione dei linguaggi ad alto livello



- Essi, come per il linguaggio Assembler, richiedono una traduzione in linguaggio macchina per essere eseguiti.
- Tale attività può essere realizzata tramite:
  - ✓ compilazione
  - ✓ interpretazione;

**Compilatore**

- Il programma compilatore analizza e traduce nella sua interezza il programma sorgente.
- Il risultato della compilazione è un programma oggetto.
- Solo dopo la traduzione dell'intero programma è possibile eseguirlo

**Interprete**

- Il programma interprete analizza e traduce istruzione per istruzione in linguaggio macchina.
- Non appena una istruzione è interpretata può essere eseguita.

# Confronto tra Compilatore e Interprete

Prof. Giuseppe Ascia

**Vantaggi del compilatore:**

- *migliori prestazioni* nell'esecuzioni;
- il codice ottenibile può essere *ottimizzato*.

**Vantaggi dell'interprete:**

- consente una *minore occupazione* di memoria;
- *minore costo* nella modifica dei programmi.

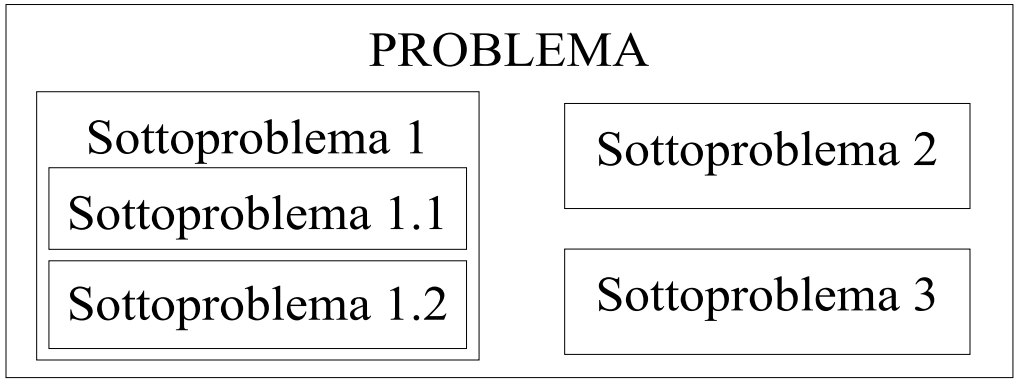
- La compilazione porta ad applicativi più veloci rispetto all'interpretazione a discapito del tempo di sviluppo.

- L'interprete consente tempi di sviluppo minori a discapito dell'efficienza nell'esecuzione.

# Il linker



- La soluzione di grossi problemi può essere risolta scomponendoli in sottoproblemi.
- Un sottoproblema può essere ulteriormente scomponibile.




- A ciascuna soluzione parziale può corrispondere un programma parziale, detto modulo.
- Ciascun modulo può essere compilato separatamente.

# Il linker



- Ciascuno dei moduli compilati ha un indirizzo logico iniziale pari a zero.
- E' necessaria una fase di collegamento dei diversi moduli per produrre un unico programma.
- Il collegamento dei diversi moduli è realizzato dal programma LINKER (collegatore).