

ASTRAZIONE

Esistono linguaggi a vari livelli di astrazione

Linguaggio Macchina:

- implica la conoscenza dei metodi utilizzati per la rappresentazione delle informazioni

Linguaggio Macchina e Assembler:

- implica la conoscenza dettagliata delle caratteristiche della macchina (registri, dimensioni dati, set di istruzioni)
- semplici algoritmi implicano la specifica di molte istruzioni

Linguaggi di Alto Livello:

- Il programmatore può astrarre dai dettagli legati all'architettura ed esprimere i propri algoritmi in modo simbolico



Sono indipendenti dalla macchina hardware sottostante
ASTRAZIONE

1

ASTRAZIONE

• Linguaggio Macchina:

```
0100 0000 0000 1000
0100 0000 0000 1001
0000 0000 0000 1000
```

Difficile leggere e capire un programma scritto in forma binaria

• Linguaggio Assembler:

```
... LOADA H
   LOADB Z
   ADD
...
```

Le istruzioni corrispondono univocamente a quelle macchina, ma vengono espresse tramite nomi simbolici (parole chiave)

• Linguaggi di Alto Livello:

```
main()
{ int A;
  scanf("%d",&A);
  if (A==0) {...}
...}
```

Sono indipendenti dalla macchina

2

ESECUZIONE

• Per eseguire sulla macchina hardware un programma scritto in un linguaggio di alto livello è necessario tradurre il programma in sequenze di istruzioni di basso livello, direttamente eseguite dal processore, attraverso:

- **interpretazione** (ad es. BASIC)
- **compilazione** (ad es. C, FORTRAN, Pascal)

3

COME SVILUPPARE UN PROGRAMMA

Qualunque sia il linguaggio di programmazione scelto occorre:

- Scrivere il **testo del programma** e memorizzarlo su supporti di memoria permanenti (*fase di editing*)
- Se il linguaggio è compilato:
 - *Compilare il programma, ossia utilizzare il compilatore che effettua una traduzione automatica del programma scritto in un linguaggio qualunque in un programma equivalente scritto in **linguaggio macchina***
 - *Eseguire il programma tradotto*
- Se il linguaggio è interpretato:
 - *Usare l'interprete per eseguire il programma*

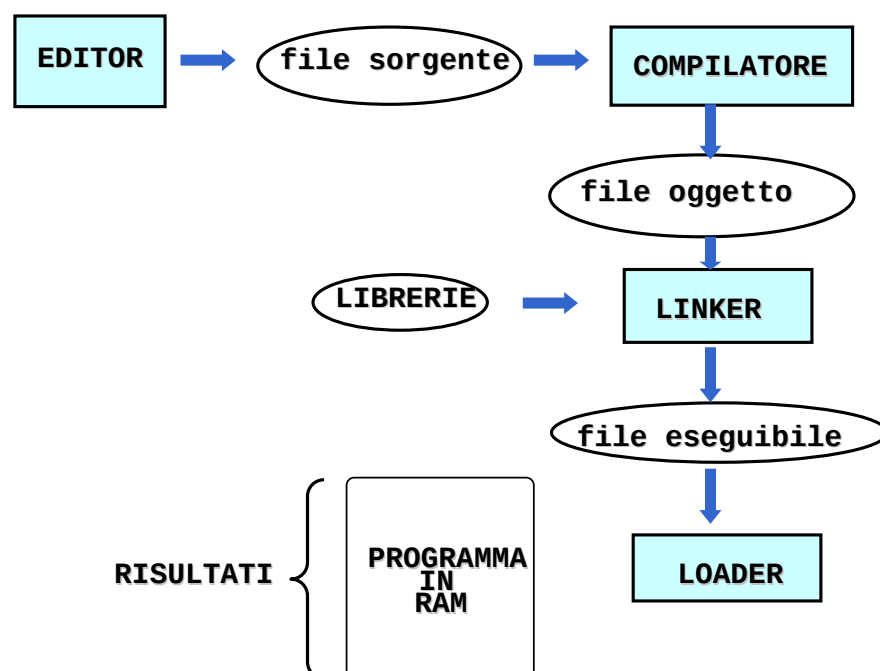
4

COMPILATORI E INTERPRETI

- I **compilatori** traducono automaticamente un programma dal linguaggio L a quello macchina (per un determinato elaboratore)
- Gli **interpreti** sono programmi capaci di eseguire direttamente un programma in linguaggio L istruzione per istruzione
- I programmi compilati sono in generale *più efficienti* di quelli interpretati

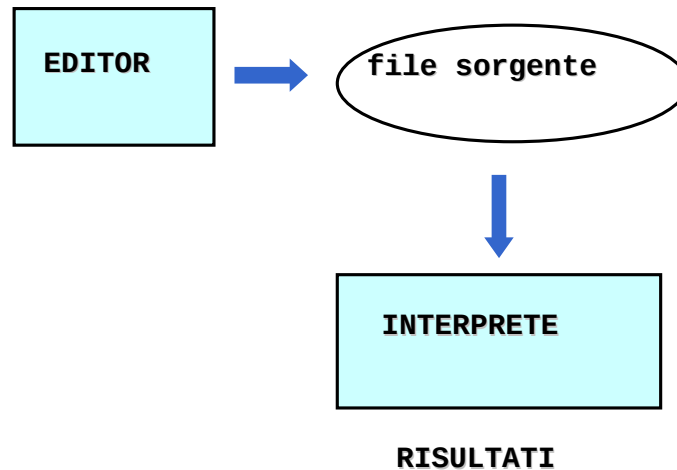
5

APPROCCIO COMPILATO: SCHEMA



6

APPROCCIO INTERPRETATO: SCHEMA



7

COMPILATORI: MODELLO

La costruzione di un compilatore per un particolare linguaggio di programmazione è complessa

- La complessità dipende dal linguaggio sorgente

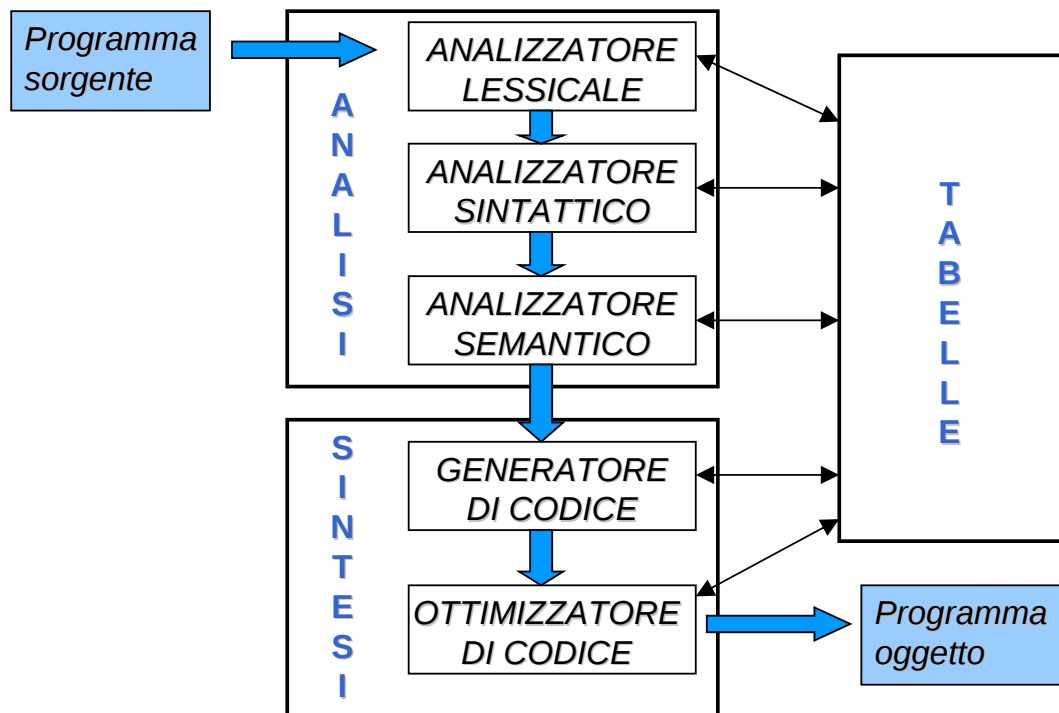
Compilatore: traduce il programma sorgente in programma oggetto

Due compiti:

- **ANALISI** del programma sorgente
- **SINTESI** del programma oggetto

8

COMPILATORI: MODELLO



9

ANALISI

Il compilatore nel corso dell'analisi del programma sorgente verifica la correttezza sintattica e semantica del programma:

- **ANALISI LESSICALE** verifica che i simboli utilizzati siano legali cioè appartengano all'alfabeto
- **ANALISI SINTATTICA** verifica che le regole grammaticali siano rispettate => albero sintattico
- **ANALISI SEMANTICA** verifica i vincoli imposti dal contesto

SINTESI

- **Generatore di codice:** trasla la forma intermedia in linguaggio assembler o macchina

Prima della generazione di codice:

- ALLOCAZIONE DELLA MEMORIA
- ALLOCAZIONE DEI REGISTRI

Eventuale passo ulteriore di **ottimizzazione del codice**