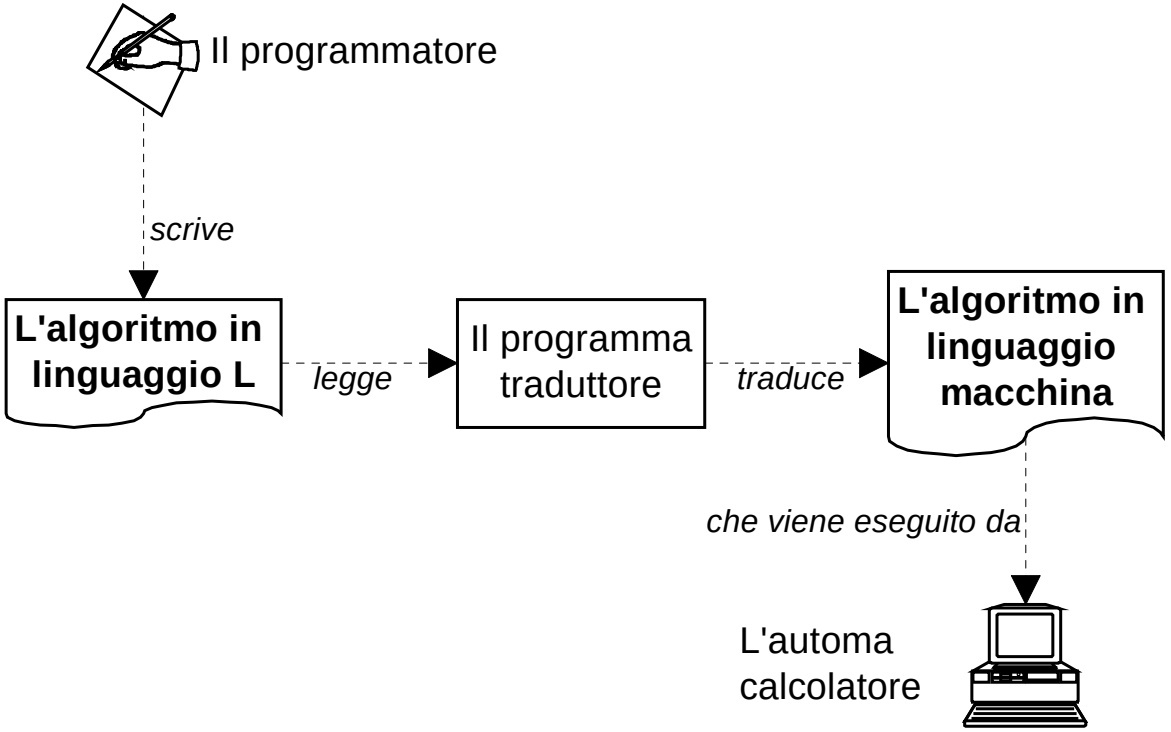
LINGUAGGI E TRADUTTORI



Il programma traduttore può:

- tradurre ed eseguire frase per frase (***interprete***)
- tradurre tutte le frasi e solo successivamente eseguire (***assemblatore o compilatore***)

Compilazione

- Un ***Compilatore*** è un programma che legge il programma sorgente e lo **traduce interamente** in un programma scritto in linguaggio macchina (programma oggetto)
  - Verifica la correttezza sintattica di ciascuna istruzione
  - Il programma oggetto è generato solo se non ci sono errori sintattici
  - La correttezza semantica è effettuata solo in fase di esecuzione

Interpretazione

- Un ***Interprete*** è un programma che legge il programma sorgente e, per ogni istruzione
  1. Verifica la correttezza sintattica
  2. Effettua la traduzione nella corrispondente sequenza di istruzioni in linguaggio macchina
  3. Esegue direttamente la sequenza di istruzioni in linguaggio macchina

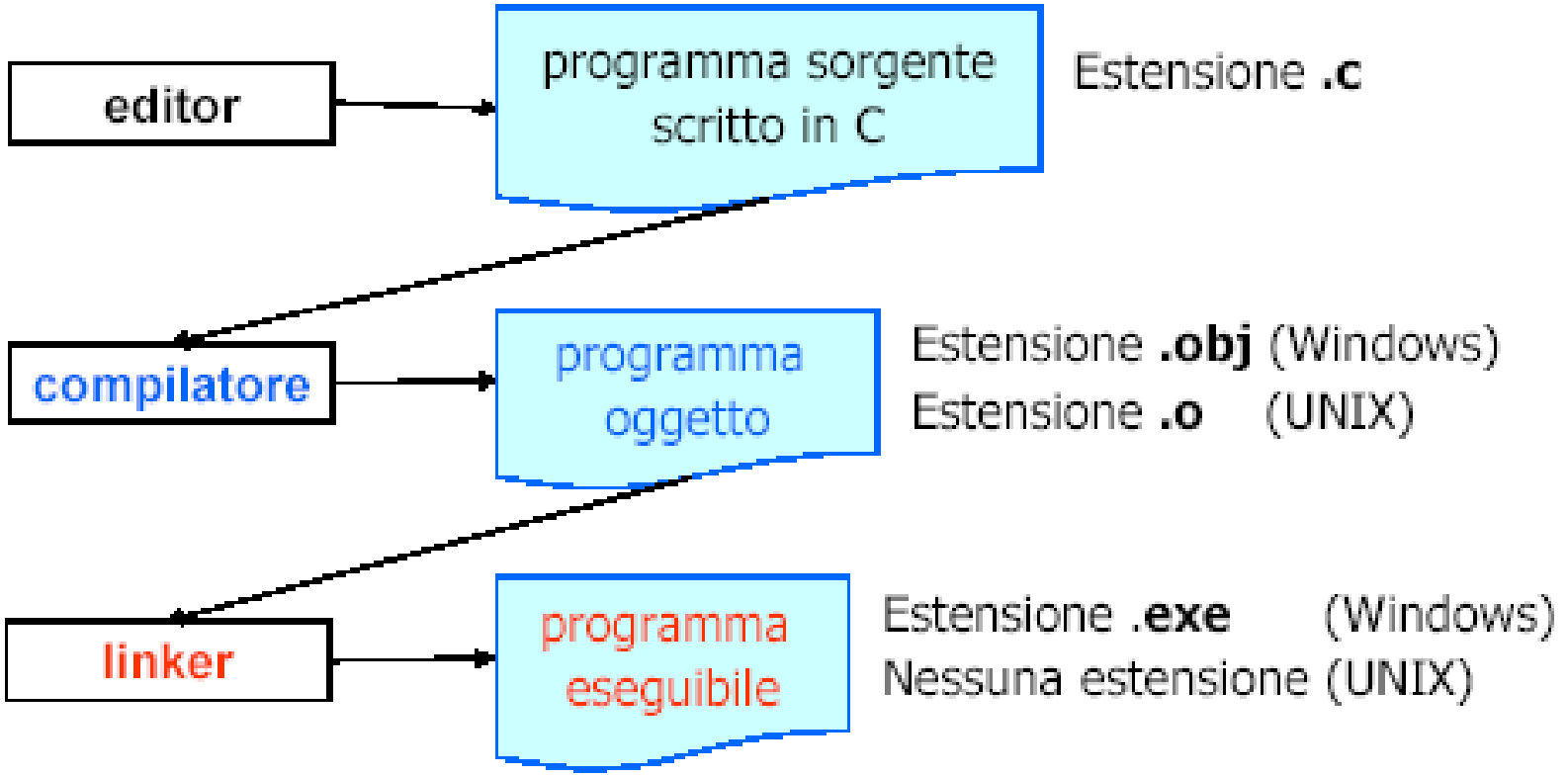
- SVANTAGGIO: Istruzioni eseguite più volte (es. ciclo), vengono verificate e tradotte più volte
- VANTAGGIO: Facile sviluppo e correzione dei programmi

Interpretazione vs Compilazione

- Velocità di esecuzione:
  - Bassa per i linguaggi interpretati
  - Alta per i linguaggi compilati

- Facilità di messa a punto dei programmi:
  - Alta per i linguaggi interpretati
  - Bassa per i linguaggi compilati

Creazione di programmi eseguibili



Moduli oggetto

I compilatori traducono ogni procedura sorgente come un modulo oggetto a se stante

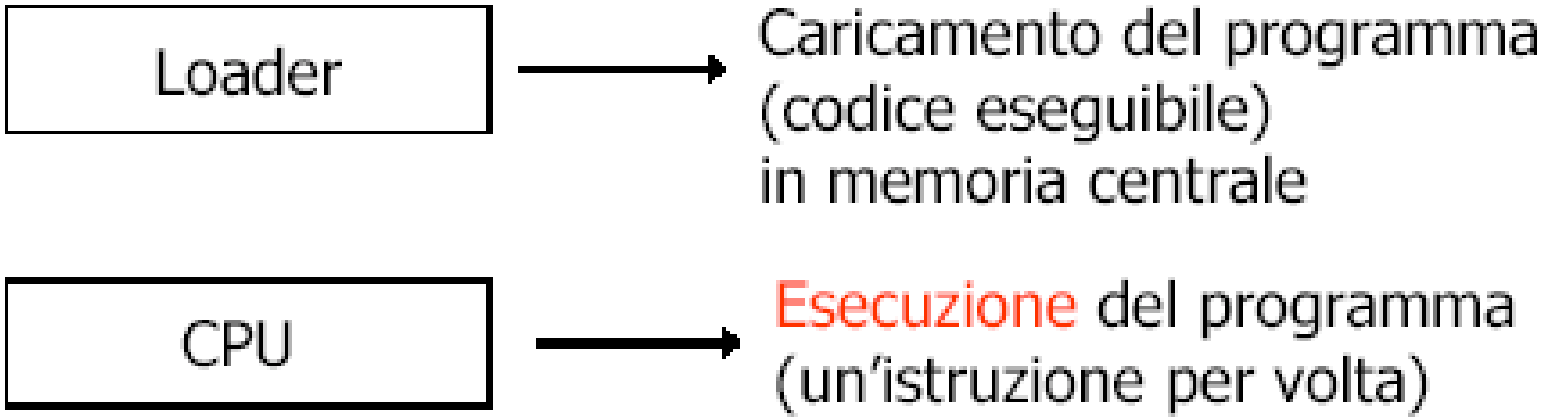
- una modifica in una procedura sorgente comporta solo una nuova traduzione della procedura modificata
- si evita di ritradurre le procedure sorgente non modificate (anche se ovviamente è necessario ricollegare tutti i moduli)

Tipicamente, un modulo oggetto contiene:

- il codice binario corrispondente ad una procedura
- Riferimenti esterni a funzioni e variabili dichiarati in altri moduli
- Funzioni e variabili utilizzabili da altri moduli

Ogni modulo oggetto ha il suo spazio di indirizzamento separato

Esecuzione di un programma



Esempio: Compilatore C Borland

Editor: genera il codice sorgente *prova.c*

```
#include <stdio.h>
int main()
{
printf("Questa è una prova!");
return 0;
}
```

Compilazione: genera il codice oggetto *prova.obj*

```
tcc -c prova.c
```

Linking: genera il codice eseguibile *prova.exe*

```
cc prova.obj [<files.obj>]
```

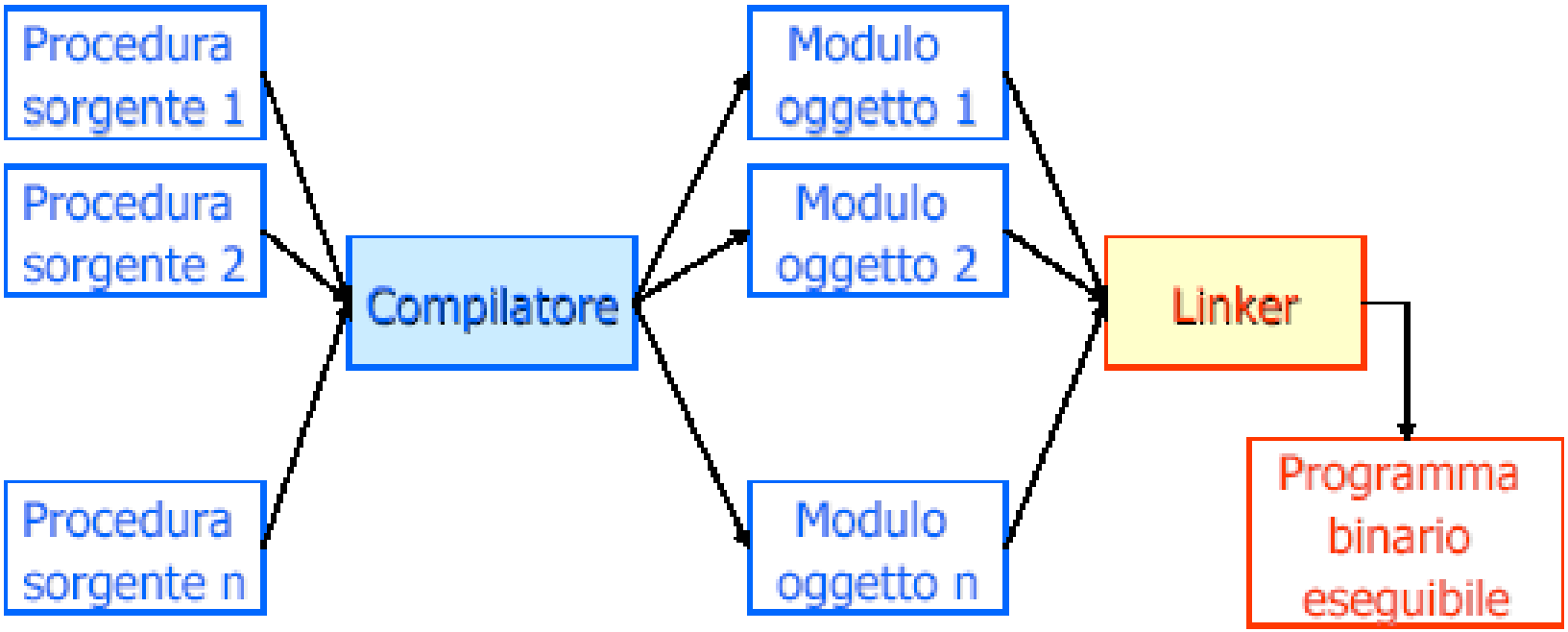
Caricamento ed **Esecuzione**

```
prova
Questa è una prova!
```

Compilazione e linking

- Tipicamente un programma è composto da più procedure o da più file sorgente
- La traduzione completa del programma richiede due fasi:
  1. Compilazione: ogni procedura sorgente (istruzioni in linguaggio di alto livello) è tradotta in un modulo oggetto (istruzioni scritte in linguaggio macchina)
     - Cambiamento di livello
  2. Collegamento dei moduli oggetto (linker)
     - Nessun cambiamento di livello: sia l'input che l'output del linker sono programmi scritti nello stesso linguaggio

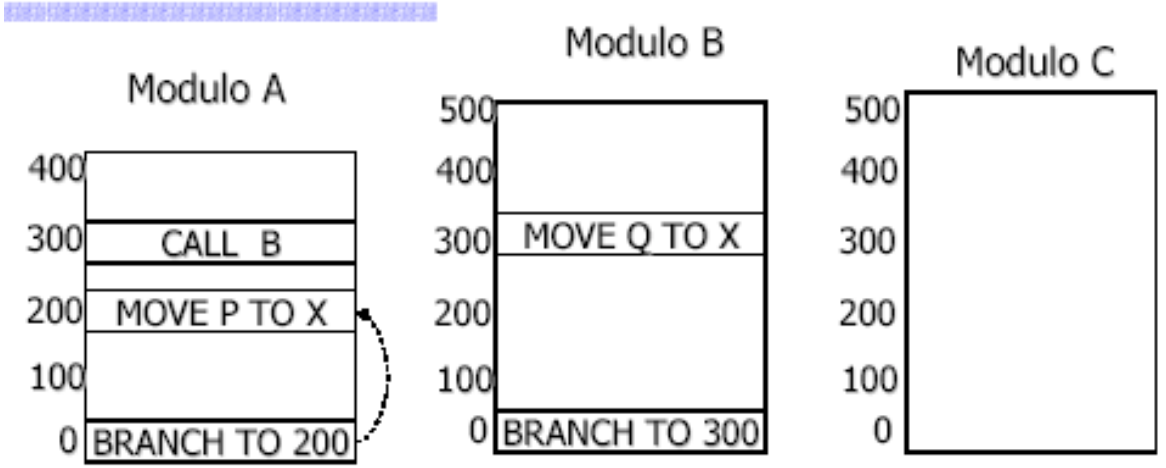
Collegamento di moduli oggetto



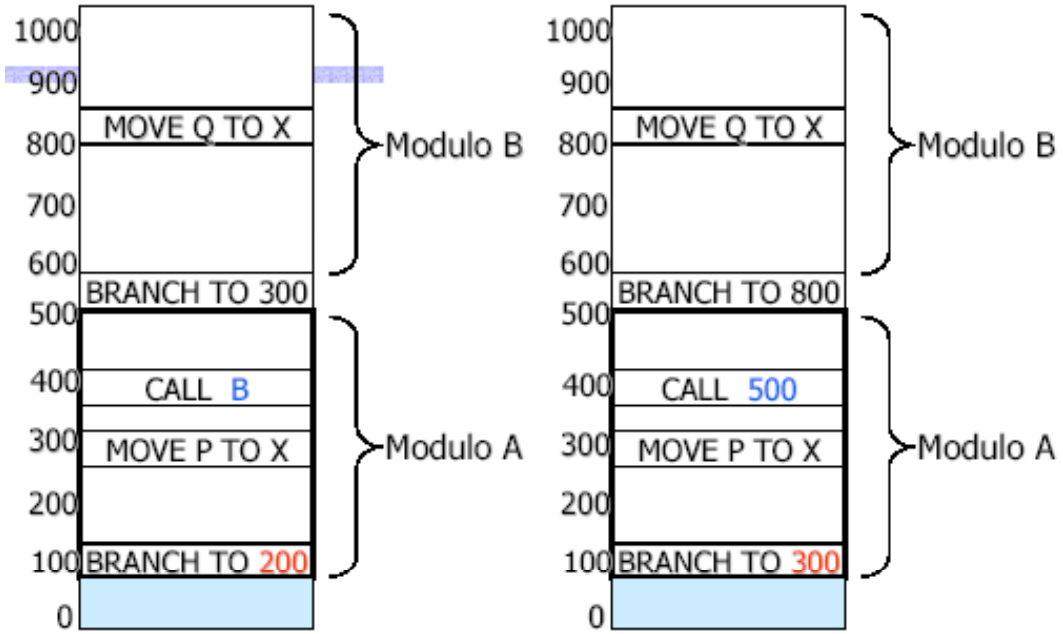
Linker

- Il linker ha la funzione di raccogliere le procedure tradotte separatamente (moduli oggetto) e collegarle (linking) tra loro, dando origine al programma binario eseguibile
- Il linker porta i moduli oggetto in memoria principale, per fornire un unico ***modulo oggetto rilocabile***
  - Viene creata una "immagine" dello spazio di indirizzamento virtuale del programma eseguibile

Moduli oggetto e indirizzi rilocabili



Spazio di indirizzi prima e dopo la rilocazione



Linker

- Il linker fonde gli spazi di indirizzamento separati dei vari moduli oggetto in un unico spazio di indirizzamento lineare:
  - Crea una tabella con tutti i moduli oggetto e la loro lunghezza
  - Basandosi su questa tabella, assegna un indirizzo di inizio ad ogni modulo oggetto
  - Riloca gli spazi di indirizzamento dei vari moduli (Rilocazione), ossia in tutte le istruzioni che contengono un indirizzo di memoria somma a ciascun indirizzo una "costante di rilocazione" uguale all'indirizzo di inizio del modulo in cui la istruzione è contenuta
  - Trova tutte le istruzioni di chiamata ad una procedura (External Reference) e le aggiorna con l'indirizzo di partenza della procedura stessa

Stadi di un programma

- Ogni programma, dalla creazione alla esecuzione, attraversa quattro stadi
  1. Codice sorgente
  2. Codice oggetto
  3. Codice eseguibile
  4. Codice in esecuzione

- In ciascuno di questi stadi il riferimento agli indirizzi di memoria è fatto in modo diverso