Titolo unità didattica: Introduzione al linguaggio C [03]

Titolo modulo : Linguaggi di programmazione [01-T]

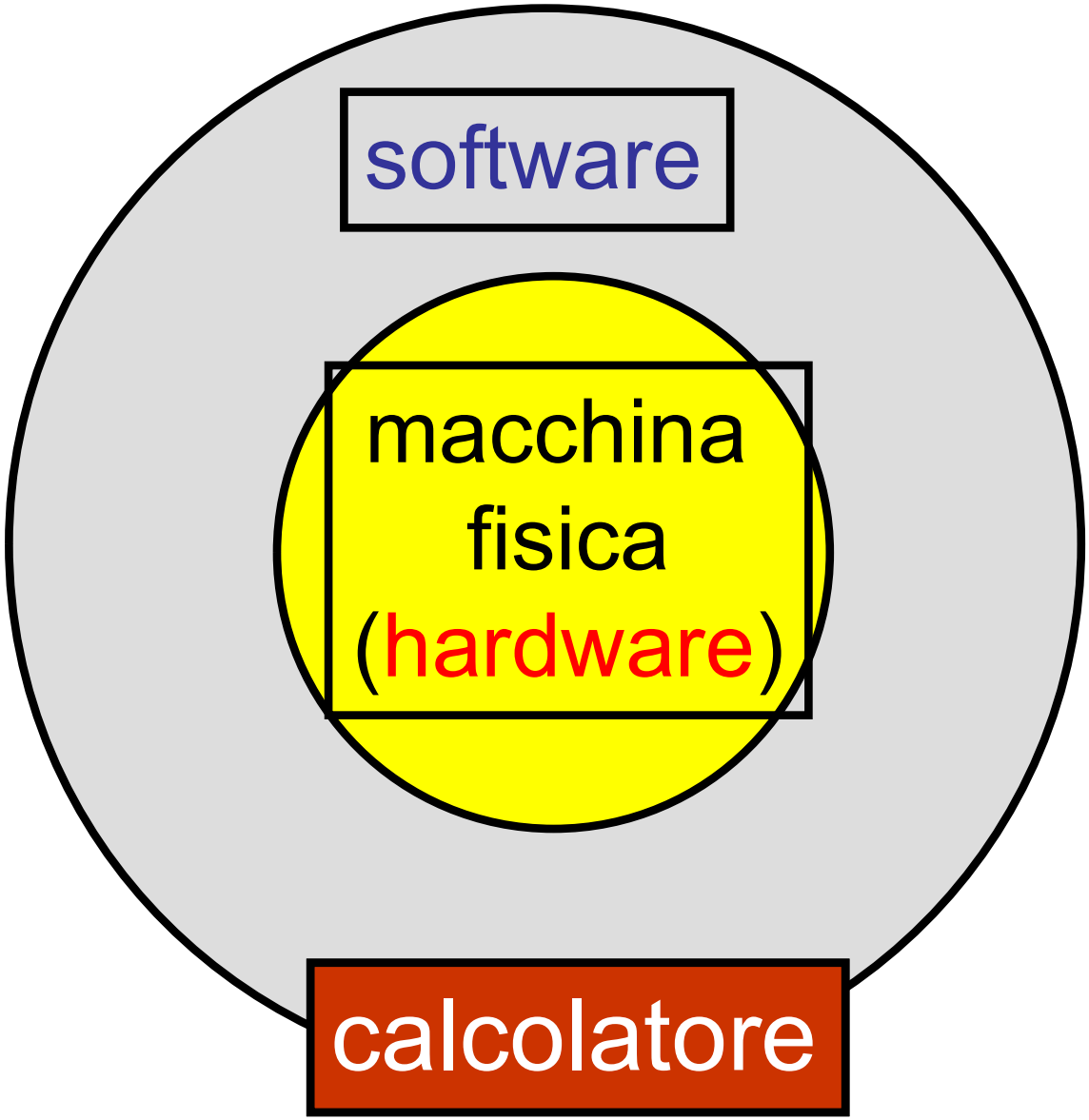
Linguaggio macchina, linguaggi di programmazione di alto livello e programmi traduttori

Argomenti trattati:

- ✓ linguaggi di programmazione
- ✓ traduttori di linguaggi di programmazione
- ✓ compilatori
- ✓ interpreti

Prerequisiti richiesti: nessuno

livello **software** + livello **hardware** = **calcolatore**



HARDWARE

esegue programmi scritti nel (proprio)
linguaggio macchina

il **linguaggio macchina** è

- estremamente elementare
- usa l'alfabeto binario
- di difficile gestione da parte del programmatore

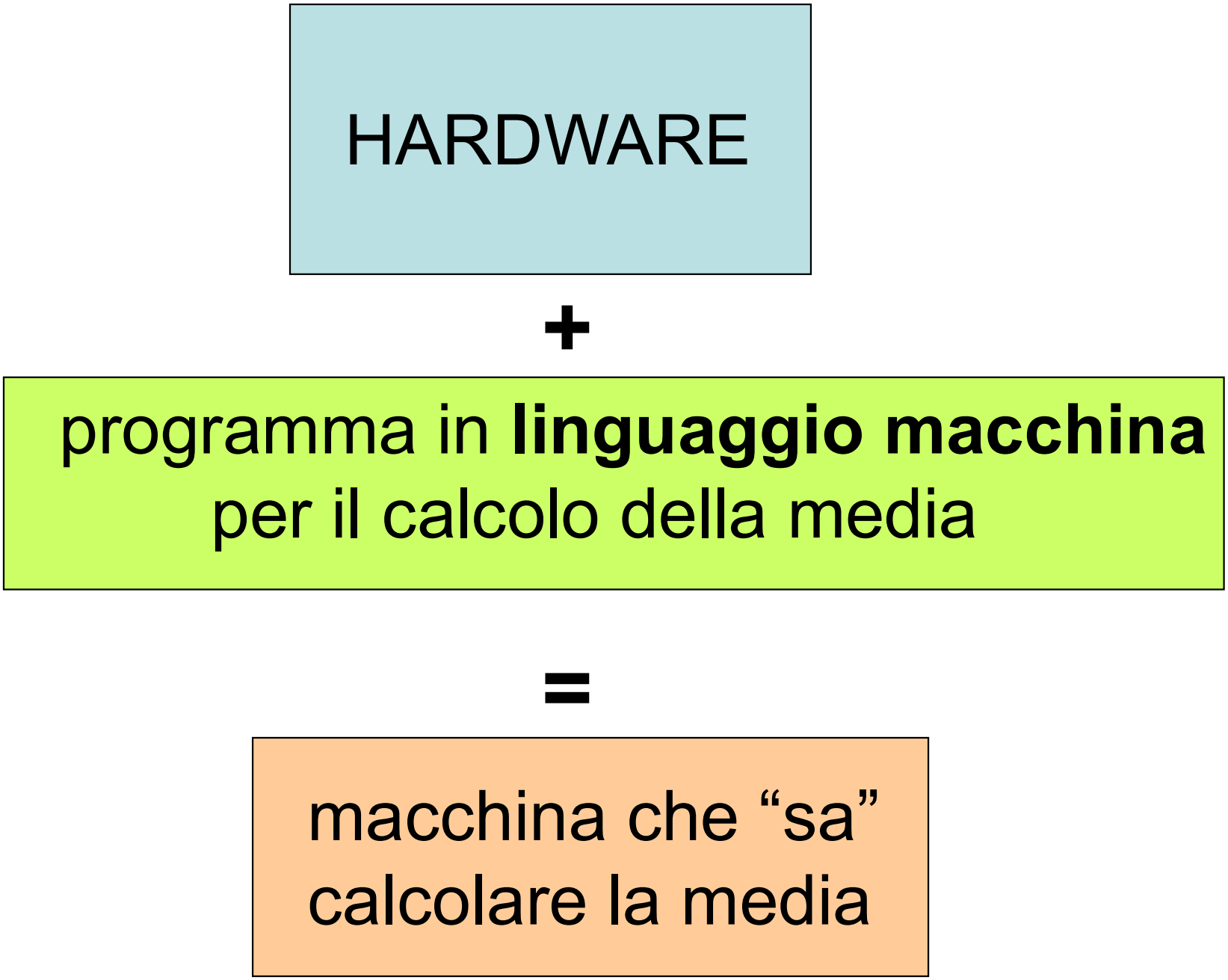
HARDWARE

+

programma in **linguaggio macchina**
per il calcolo della somma di n numeri

=

macchina che “sa”
calcolare la somma di n numeri


HARDWARE
+
programma in **linguaggio macchina**
per il calcolo della media
=
macchina che “sa”
calcolare la media

linguaggio di programmazione

scopo:
descrizione semplice e sintetica di un algoritmo, cioè dei dati e delle sequenze di operazioni su di essi

programma = algoritmo scritto in un linguaggio di programmazione

un programma è un testo che descrive una computazione

un **linguaggio di programmazione** è un insieme di convenzioni (standard) formali per descrivere un **programma**

linguaggio di programmazione

il più semplice dei programmi C

```
#include <stdio.h>
void main()
{
printf("questo e' il mio primo programma C\n");
}
```

linguaggio di programmazione

- ✓ **lessico**
- ✓ **sintassi**

l'insieme delle **parole corrette** del linguaggio (il *dizionario*)

le **regole** per mettere insieme le parole per **costruire una frase**

linguaggio formale

linguaggio formale

≠

linguaggio naturale

- ✓ un linguaggio formale è progettato per essere completamente non ambiguo
- ✓ un linguaggio formale è conciso e non ridondante
- ✓ un linguaggio formale può essere facilmente tradotto in un altro linguaggio formale

HARDWARE

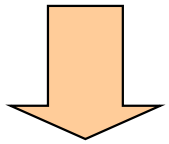
+

programma (in **l.m.)** che traduce un programma scritto in **C** in un programma scritto nel **linguaggio macchina**

=

macchina che “sa” eseguire **programmi C**

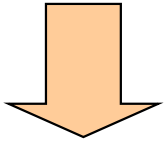
linguaggio di programmazione



programma traduttore

un **programma traduttore** del **linguaggio L** per **la macchina M** è un programma che trasforma (traduce) un programma (o una porzione di programma) scritto nel linguaggio **L** (programma **sorgente**, **source code**) in un programma equivalente (programma **oggetto**, **object code**) scritto nel linguaggio macchina della macchina **M**

programmi traduttori



compilatori

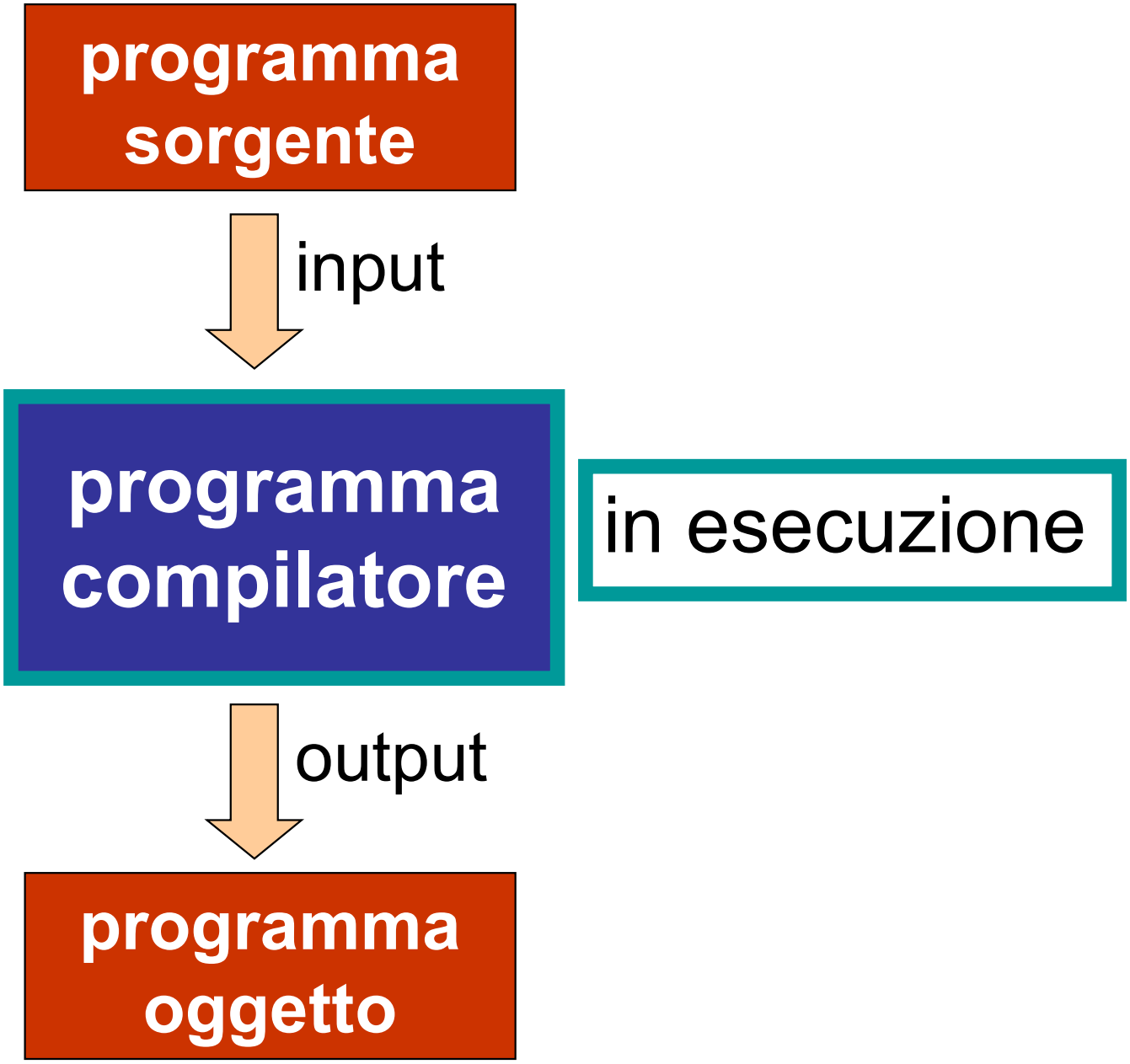
traduzione globale

esecuzione globale

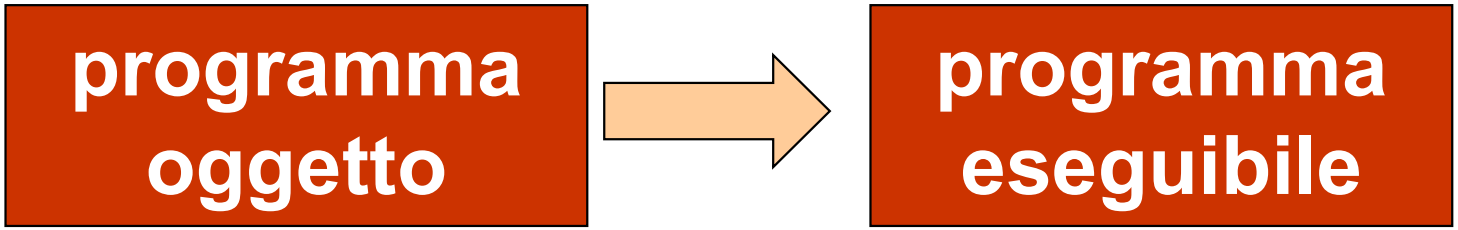
interpreti

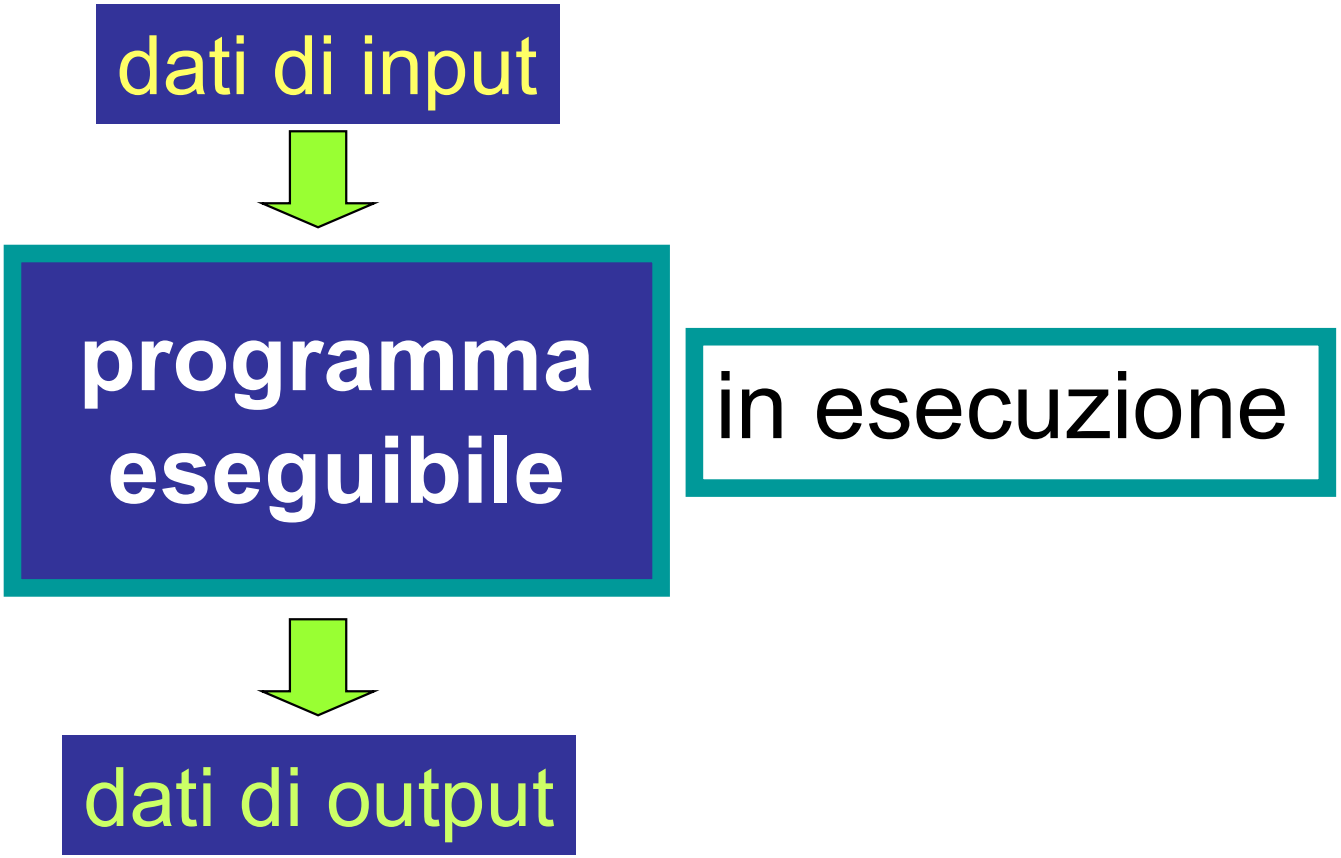
traduzione ed esecuzione di una istruzione alla volta

processo di compilazione



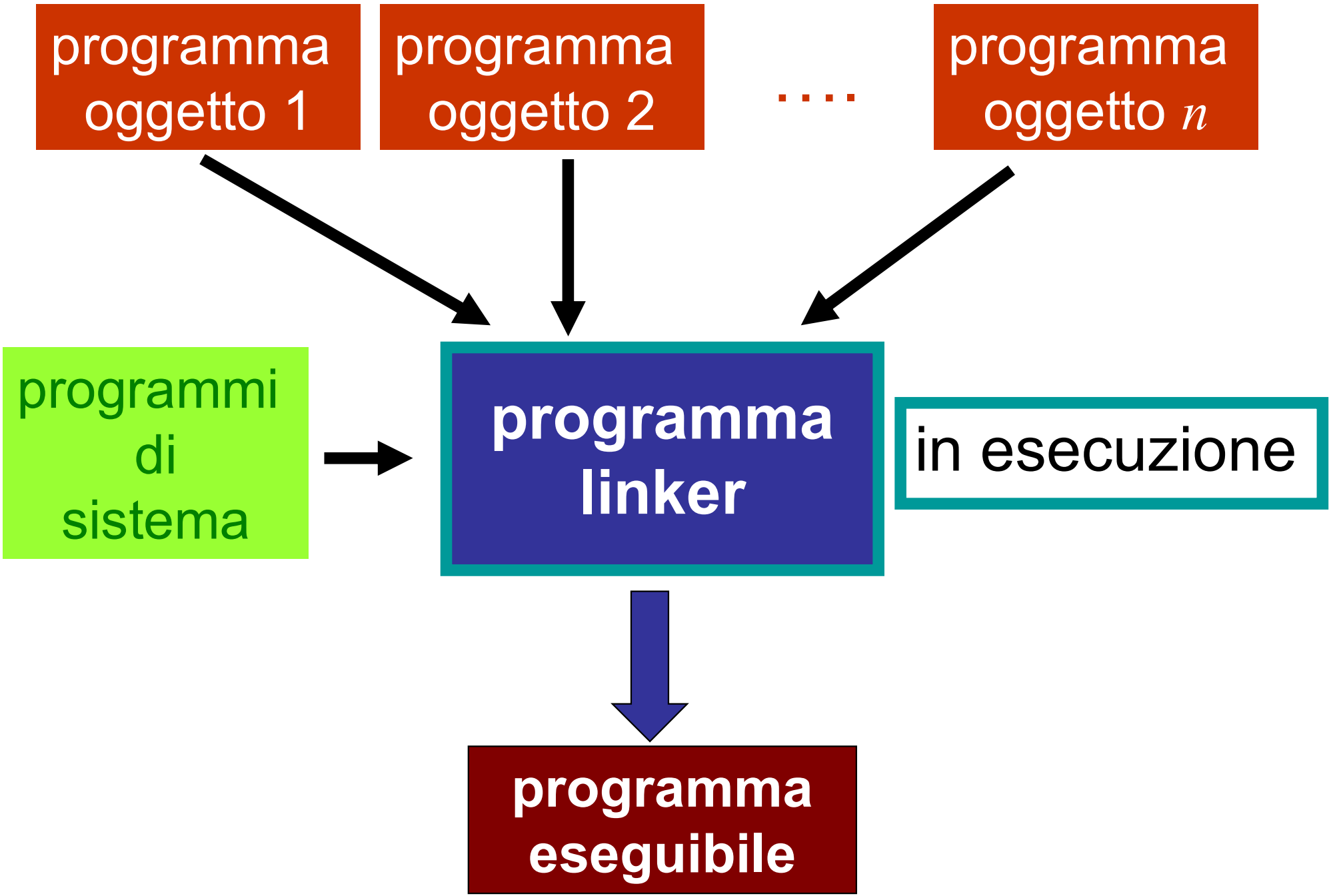
esecuzione


programma oggetto
programma eseguibile


dati di input
programma eseguibile
in esecuzione
dati di output

eseguire un programma (*source code*) in linguaggio compilato:

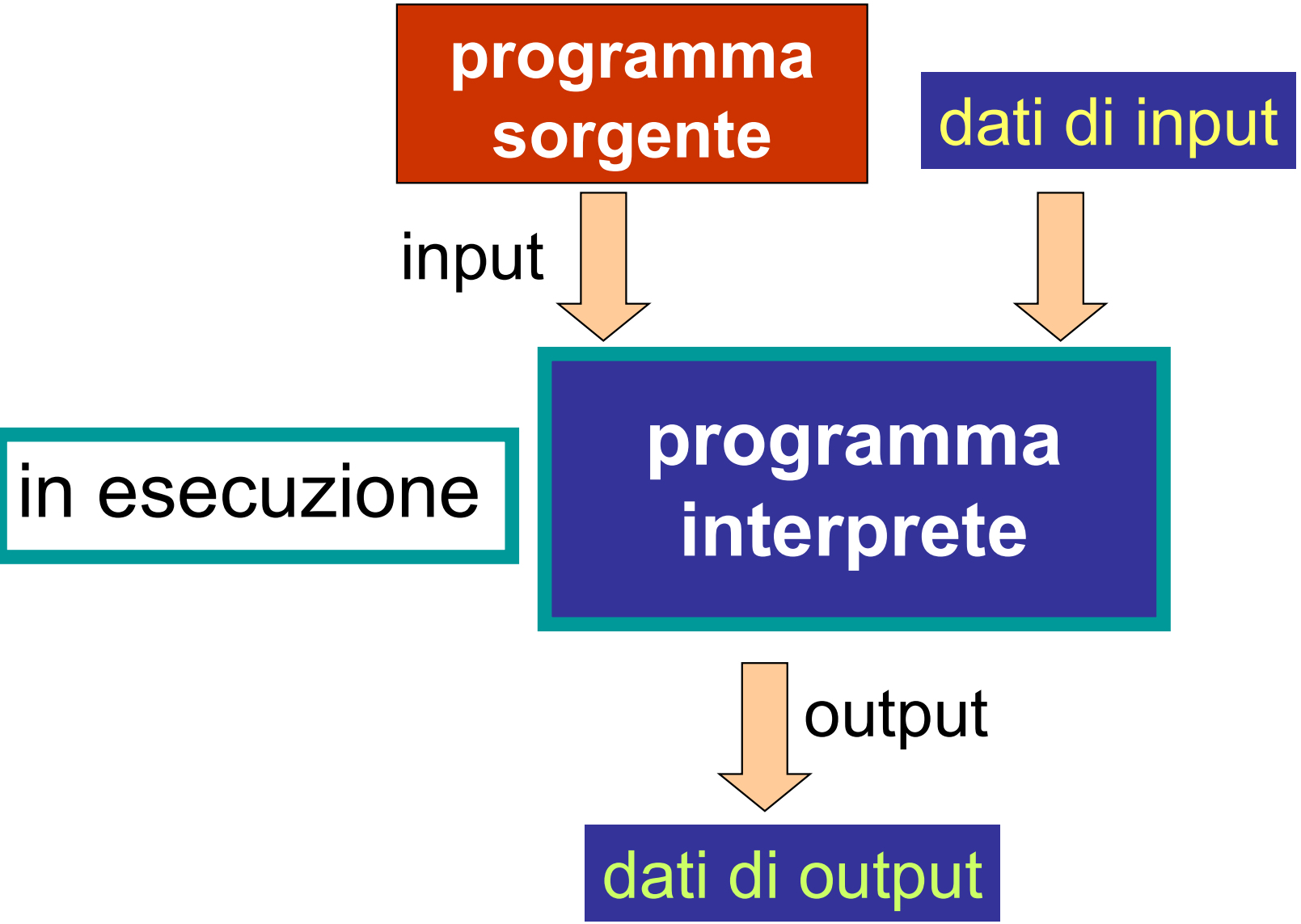
- ✓ eseguire il **programma compilatore**, che produce il programma **oggetto** (*object code*)
- ✓ eseguire il programma (**linker**) che trasforma il programma oggetto in programma **eseguibile** (*executable code*)
- ✓ **eseguire** il programma **eseguibile**


programma oggetto 1
programma oggetto 2
....
programma oggetto n
programmi di sistema
programma linker
in esecuzione
programma eseguibile

fasi di compilazione:

- ✓ **analisi lessicale**, identificazione e classificazione dei simboli
- ✓ **analisi sintattica**, correttezza della combinazione dei simboli in strutture sintattiche
- ✓ **generazione del codice** in linguaggio macchina
- ✓ **ottimizzazione** del programma oggetto

processo di interpretazione



eseguire un programma in linguaggio interpretato:

- ✓ eseguire il **programma interprete,** che ripete le seguenti azioni
  - considera una direttiva del programma sorgente
  - **traduce** la direttiva in una sequenza di istruzioni in linguaggio macchina
  - **esegue** tale sequenza di istruzioni
  - conserva i dati di input e di output generati durante l'esecuzione

compilatore

PRO: una successiva esecuzione del programma non richiede una successiva compilazione

CONTRO: non c'è interazione tra programmatore e programma in esecuzione

interprete

PRO: elevata interattività tra programmatore e programma in esecuzione

CONTRO: ogni successiva esecuzione del programma comporta ogni volta un processo di interpretazione