La Programmazione

Cos'è la programmazione?

Concetti preliminari

Sommario

- La programmazione, questa sconosciuta
- Programmiamo Macchine Astratte
- Linguaggi di basso e alto livello e loro implementazione
- Esempi: C, Java

Programmare vs Usare un computer

- Esite una grossa differenza, ma in genere sconosciuta ai più
- “Ah! Tu sei un informatico! Senti, come faccio a scansire e modificare un’immagine e a metterla sul mio sito?”
- Sta parlando di **come usare** il computer, di come utilizzare
  - Un certo numero di programmi già scritti da qualcuno e installati sul pc
  - Dei servizi offerti da certi provider internet

Programmare vs Usare un computer

- Una delle competenze base di un informatico è quella, invece, di conoscere almeno un modo per **programmare**, cioè per scrivere programmi che fanno qualcosa, quello che vuole lui/lei/il suo datore di lavoro/una comunità
- La programmazione è un'attività interessante, entusiasmante, creativa: moltissime persone nel mondo lo fanno per puro divertimento (oltre che per guadagnarsi da vivere)

Programmi ognidove

- Esistono in circolazione moltissimi programmi, che fanno le cose più svariate (word-processor, browser internet, computer graphics, riproduzione suoni/immagini, ...)
- Il codice di diversi di loro è disponibile per essere letto e/o modificato: si tratta del codice **open-source**, scaturito per lo più dal tempo libero di programmatori di tutto il mondo: essi ricavano soddisfazione e gratificazione dallo scrivere un buon programma e dal fatto che poi questo venga anche usato da altri

Programmi ognidove

- Nonostante l'abbondanza esiste sempre l'esigenza di scrivere programmi nuovi o di migliorarne di già esistenti
- In ogni caso esistono dei limiti a quello che un computer può calcolare: ci sono dei problemi per cui non esiste nessun **algoritmo** risolutivo
- Un algoritmo è il cuore di un programma: è il procedimento di calcolo che il programma deve seguire

Programmare

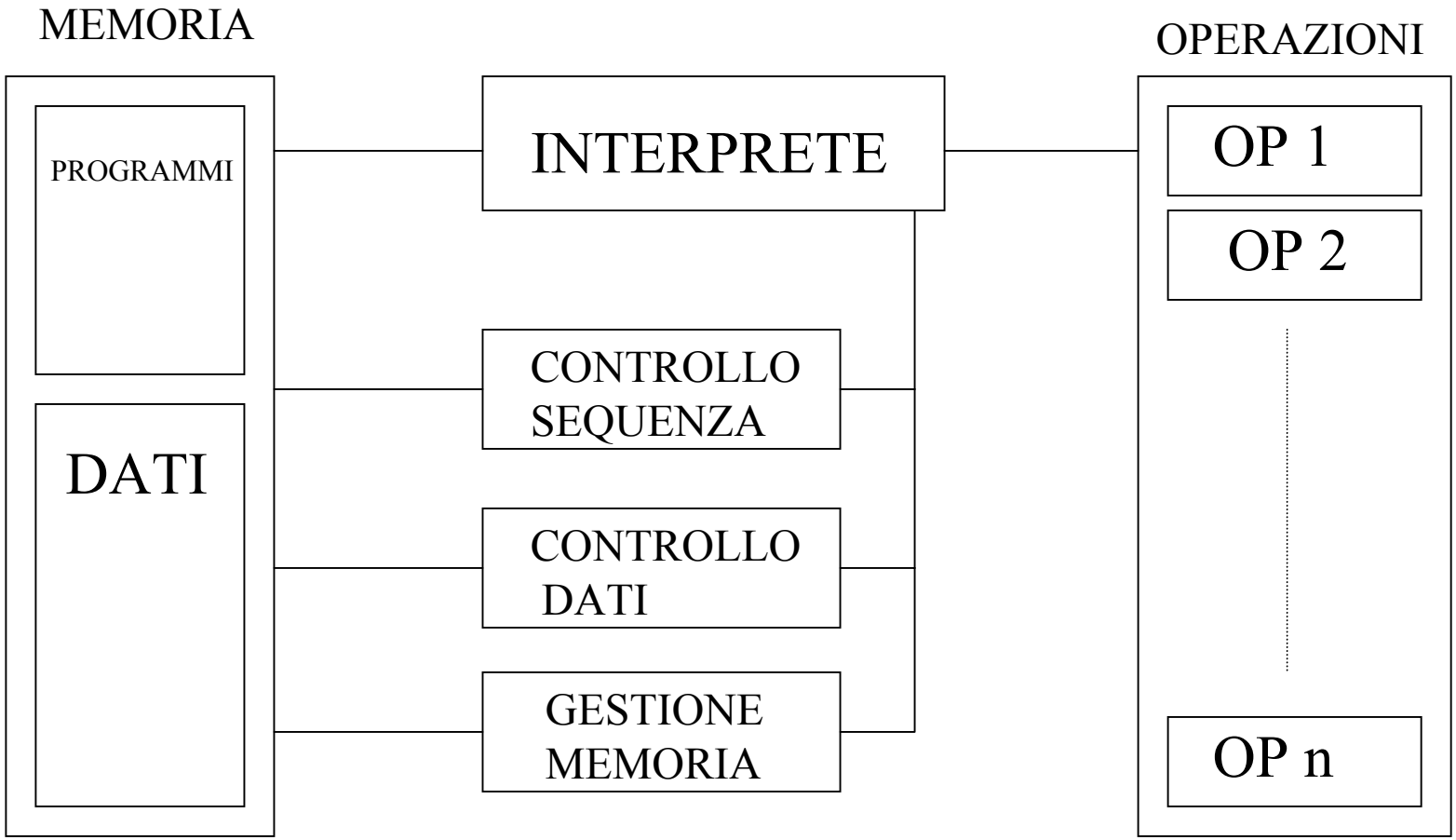
- Per poter programmare una certa macchina abbiamo bisogno di:
  - Un algoritmo che calcola ciò che il programma deve calcolare
  - Un linguaggio per specifacare l'algoritmo. La macchina su cui vogliamo far girare il programma deve saper capire questo linguaggio
- In genere il programma che si scrive è una serie di istruzioni

Macchine astratte

- Esiste un insieme di concetti che sono validi per una qualunque macchina e un qualunque linguaggio di programmazione
- Questo insieme di concetti si chiama **Macchina Astratta**
- È molto utile per fissare dei punti di riferimento generali nella programmazione e per definire anche precisamente come è implementato un certo linguaggio

Macchina Astratta

un insieme di **strutture dati** ed **algoritmi** in grado di **memorizzare** ed **eseguire** programmi



Es: Una macchina fisica

- Operazioni Primitive
  - Operazioni aritmetico-logiche
  - Operazioni di manipolazione di stringhe di bit
  - Lettura/Scrittura di celle di memoria e registri
  - Input/output
- Controllo di Sequenza (salti, condizionali, chiamate e ritorni dai sottoprogrammi)
  - Registro contatore istruzioni (PC)
  - Strutture dati che contengono i punti di ritorno dei sottoprogrammi

Es: Una macchina fisica

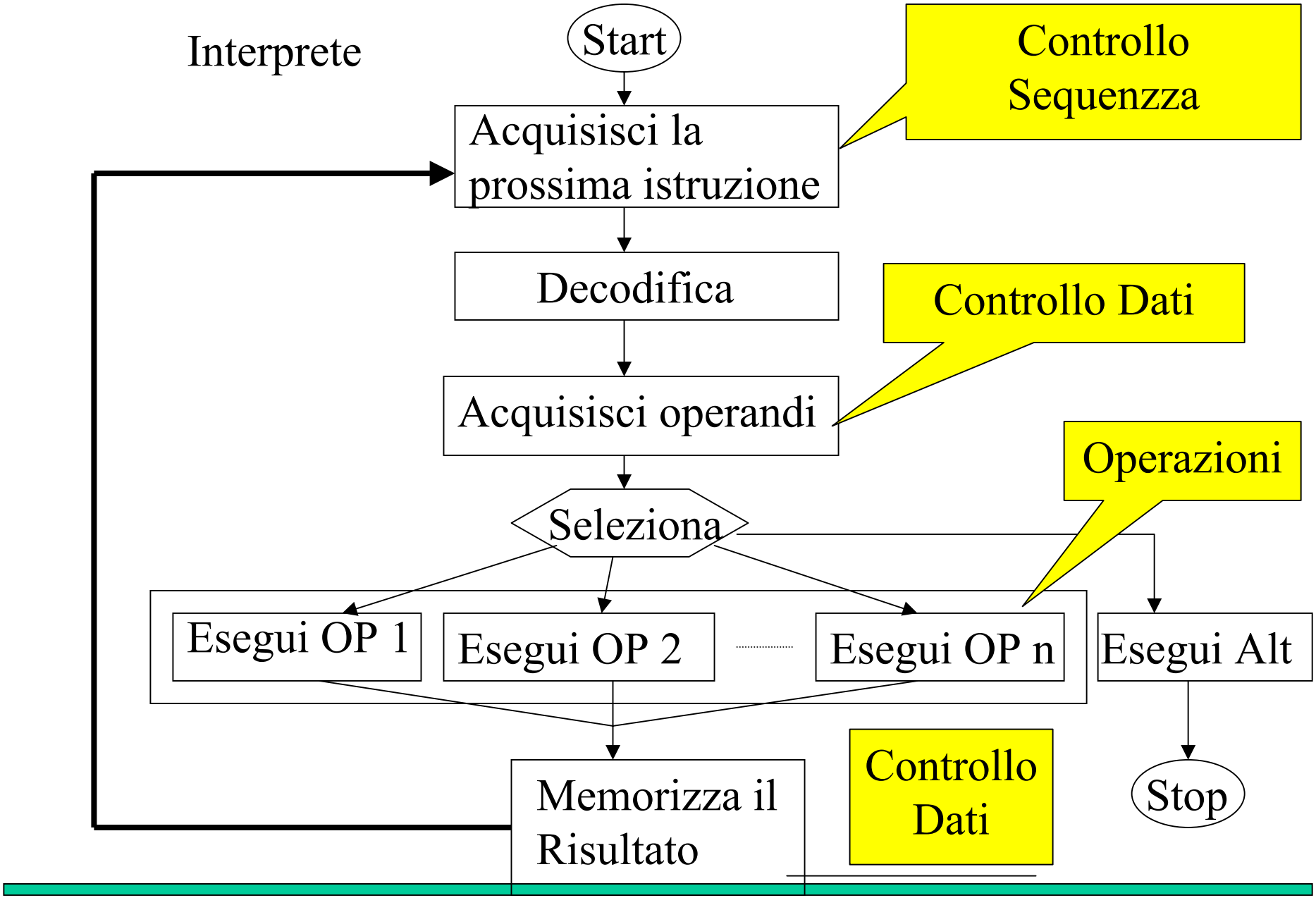
- Controllo dati
  - Acquisizione operandi
  - Memorizzazione risultato
  - Architettura a registri:
    - Registri indice
    - Indirizzamento indiretto
  - Architettura a Pila:
    - Gestione della Pila

Es: Una macchina fisica

- Gestione della memoria
  - Architettura a registri:
    - Nessuna poiché la memorizzazione è statica
  - Architettura a Pila:
    - Allocazione e recupero dei dati sulla Pila

L’interprete

- La struttura dell’interprete è sempre la stessa per una qualunque macchina astratta
- Quello che cambia sono le altre componenti


Interprete
Start
Controllo Sequenza
Acquisisci la prossima istruzione
Decodifica
Controllo Dati
Acquisisci operandi
Operazioni
Seleziona
Esegui OP 1
Esegui OP 2
Esegui OP n
Esegui Alt
Memorizza il Risultato
Controllo Dati
Stop

Il linguaggio di una macchina astratta

- M macchina astratta
- L_M linguaggio “macchina” di M: è il linguaggio in cui si esprimono tutti i programmi interpretati dall’interprete di M
- I programmi sono particolari dati primitivi su cui opera l’interprete

Macchine astratte

- Ai componenti di M corrispondono i componenti di L_M
  - Tipi di dato primitivi
  - Meccanismi per il controllo della sequenza
  - Meccanismi per il controllo del trasferimento dei dati
  - Meccanismi per la gestione della memoria

Realizzazione di Macchine Astratte

- Una macchina astratta è una collezione di strutture dati ed algoritmi
- Può essere realizzata combinando 3 tecniche
  1. Realizzazione in hardware
  2. Emulazione o simulazione via firmware
  3. Simulazione software

Dai linguaggi alle macchine astratte

- $M \rightarrow L_M$
- $L \rightarrow M_L$ è la macchina astratta che ha L come linguaggio macchina
- Se L è un linguaggio ad alto livello, M_L può essere molto complessa
- Implementare L vuol dire realizzare M_L
- Come?

Implementazione di M_L

- Generalmente mediante simulazione (software od eventualmente firmware) su una macchina (astratta) ospite M_0
- Se l'interprete di M_L è simulato, l'implementazione si chiama **interpretativa**
- Esiste un'alternativa basata su tecniche di traduzione (soluzione **compilativa**)

Programmare in un linguaggio

- Conoscere un linguaggio di programmazione corrisponde a conoscere tutti i suoi costrutti e come questi vengono eseguiti dalla relativa macchina astratta
- Avendo queste conoscenze si possono scrivere programmi nel linguaggio scelto e si può anche non conoscere per niente il tipo di implementazione della macchina astratta del linguaggio

Programmare in un linguaggio

- In buona parte di questo corso ci occuperemo di capire bene la macchina astratta Java, cioè studieremo le strutture dati, i costrutti del linguaggio Java e che effetti hanno quando vengono eseguiti
- Vedremo anche alcuni dettagli dell'implementazione del linguaggio

Linguaggi ad alto e basso livello

- I linguaggi di programmazione si possono classificare, in base alla distanza tra la loro macchina astratta e alla macchina ospite si cui poi verranno fatti girare:
  - Piccola distanza (poche differenze): linguaggi di basso livello. Es: codice macchina di un pc, assembly.
  - Grande distanza (la struttura della macchina astratta del linguaggio è molto diversa da quella della macchina ospita): linguaggi di alto livello.

Linguaggi di basso livello

- Una tipica macchina ospite, almeno per applicazioni di ufficio/domestiche, è un pc
- Un pc è può essere visto come una macchina astratta che risulta dalla composizione di diversi livelli:
  - L'hardware
  - Il firmware (sequenze di operazioni elementari che implementano istruzioni di linguaggio macchina)
  - Il sistema operativo (che aggiunge funzionalità di gestione delle risorse)

Linguaggi di basso livello

- Un esempio tipico di linguaggio di basso è il linguaggio macchina di un pc.
- Istruzioni molto semplici che operano su dati molto semplici
  - Sposta un valore intero dalla memoria ad un registro
  - Incrementa il valore di un registro
  - Salta ad una certa istruzione se il valore di un registro è maggiore di zero
  - ...

Linguaggi di basso livello

- Ogni istruzione elementare è identificata da un codice numerico
- I dati che servono all'istruzione per operare sono numeri
- Il programma viene caricato in memoria ad un certo indirizzo e ognuna delle istruzioni è individuabile dal punto di memoria in cui si trova tramite un indirizzo
- Il processore esegue le istruzioni in sequenza (eseguendo dei salti in base a determinate istruzioni)

Linguaggi di basso livello

- Risulta molto difficoltoso, ripetitivo e fonte di errori il programmare in linguaggio macchina
- Un linguaggio leggermente migliore è il linguaggio **assembly**:
  - Ogni istruzione è identificata da un codice mnemonico (mov, add, goto,...)
  - Le istruzioni e i dati possono essere etichettati con codici mnemonici per riferirli in maniera semplice
  - Un programma, detto assemblatore, si occupa di tradurre in codice macchina numerico

Linguaggi di basso livello

- Un esempio di programma assembly

```
MOVF id3, R2
MULF \#60.0, R2
MOVF R2, id1
ADDF R2, R1
MOVF R1, id1
```

- id1, id3 sono etichette che si riferiscono a dati (variabili intere); R1, R2 sono registri; MOVF, MULF, ADDF sono codici mnemonici di istruzioni macchina

Linguaggi di alto livello

- È facile immaginare la difficoltà di scrivere grandi applicazioni scrivendo programmi assembly
- Il numero di istruzioni da scrivere e la loro organizzazione richiederebbe moltissimo tempo e la probabilità di commettere errori sarebbe altissima, oltre al fatto che individuare gli errori sarebbe molto difficile
- Per questo ben presto sono emersi i linguaggi di alto livello

Linguaggi di alto livello

- Astraggono diverse componenti della macchina ospite fisica
- I programmi sono più leggibili e intelligibili dagli esseri umani
- La scrittura di programmi diventa anch'essa una espansione del linguaggio tramite la definizione di nuove astrazioni, tipi di dato, operazioni
- Il debugging è più semplice e gli errori sono limitati dalla strutturazione imposta

Linguaggi di alto livello

- Richiedono una implementazione per poter essere poi eseguiti in una macchina fisica ospite
- Esistono diversi modi per implementarli:
  - Compilazione o traduzione
  - Interpretazione
  - Varie combinazioni delle due

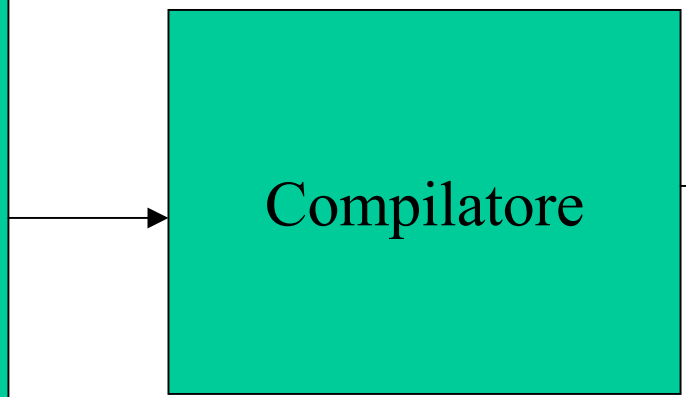
Compilazione

- Un compilatore è un programma che si occupa di tradurre un altro programma scritto in un linguaggio L in un programma **equivalente** scritto in L0, linguaggio di una macchina fisica ospite

Compilazione

Programma scritto in L

```
int main() {
double x;
int z = 10;
while (z >0)
.....
```



Programma equivalente in L0

```
MOV l1, R1
MOVF l3, R2
....
```

Compilazione

- Molti linguaggi sono compilati: C, C++, Pascal, FORTRAN, COBOL, Java (parzialmente)
- Scrivere un compilatore è un compito difficile, ma una volta fatto esso permette di far eseguire tutti i programmi del linguaggio L per cui è stato scritto da macchine che funzionano con il linguaggio L0

Interpretazione

- Il linguaggio L è implementato su una macchina che funziona su L0 tramite una simulazione software
- Un programma in L0 (generalmente chiamato interprete) si occupa di prelevare nel giusto ordine le istruzioni di un programma dato in L, di simulare la loro esecuzione e di restituirne i risultati.
- Esempi di linguaggi interpretati: PROLOG, LISP, Java bytecode

Interpretazione

Programma scritto in L

```
int main() {
double x;
int z = 10;
while (z >0)
.....
```

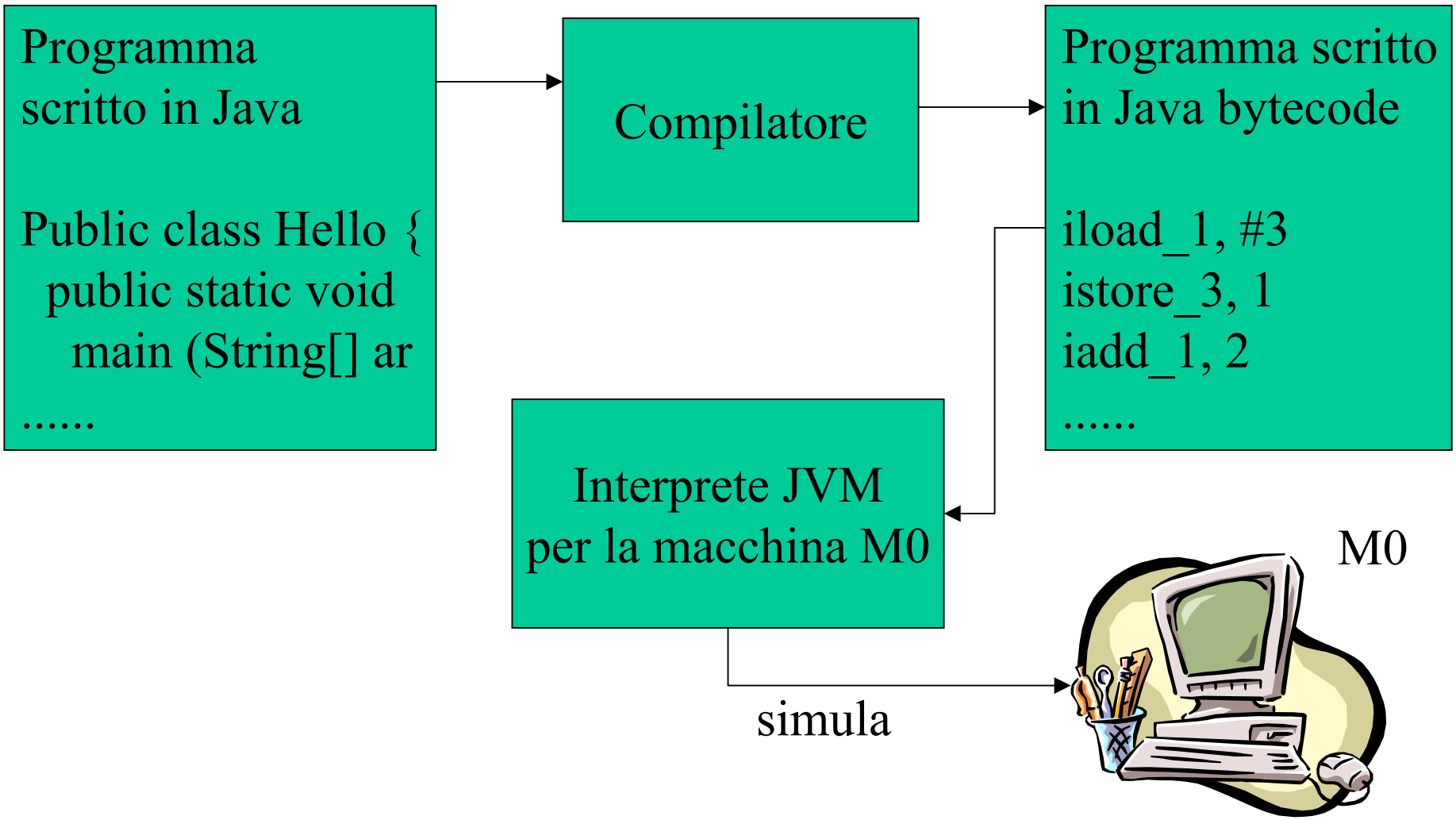
Esempio: linguaggio C

- Ad alto livello
- Compilato
- Il codice compilato può essere eseguito sulla macchina ospite con il supporto di alcune librerie che implementano diverse funzionalità (supporto a tempo di esecuzione)

Esempio: Java

- L'implementazione del linguaggio Java è mista
- I programmi in Java vengono compilati, ma il risultato della compilazione è eseguibile su una macchina astratta (la Java Virtual Machine - JVM) che di solito non è una macchina fisica
- Il linguaggio della JVM si chiama Java bytecode ed ha le caratteristiche di un linguaggio di basso livello con diverse funzionalità specifiche per gestire le particolarità del linguaggio Java (oggetti, classi, eccezioni etc.)
- La JVM viene simulata tramite interpretazione su diverse macchine fisiche

Esempio: Java



Esempio: Java

- La JVM è disponibile per molte architetture (macchine fisiche)
- In questo modo uno stesso programma Java compilato può essere eseguito su diverse architetture
- Portabilità
- Indipendenza dall'architettura