Programmazione per Bioinformatica
Il Calcolatore e la Programmazione

Dr Damiano Macedonio
Università di Verona

Architettura del calcolatore

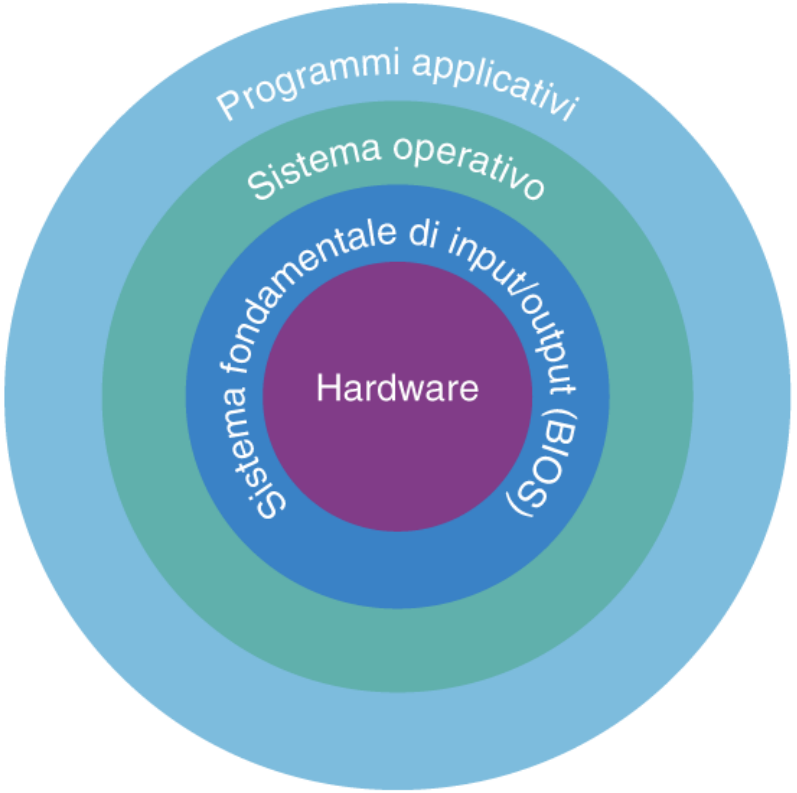
- La prima decomposizione di un calcolatore è relativa a due macro-componenti:
  - Hardware:

    componenti fisiche.
  - Software:

    tutti i possibili tipi di

    programmi utilizzati per

    fornire istruzioni al computer



Architettura del calcolatore

- L'architettura dell'**hardware** di un calcolatore reale è molto complessa
- La **macchina di von Neumann** è un modello semplificato dei calcolatori moderni, composta da
  - Unità di calcolo
  - Una memoria che contiene programmi e dati dei programmi

Le istruzioni da eseguire stanno in memoria,
vengono prelevate, decodificate ed eseguite.

- **John von Neumann,** matematico ungherese, progettò, verso il 1945, il primo calcolatore con programmi memorizzabili anziché codificati mediante cavi e interruttori

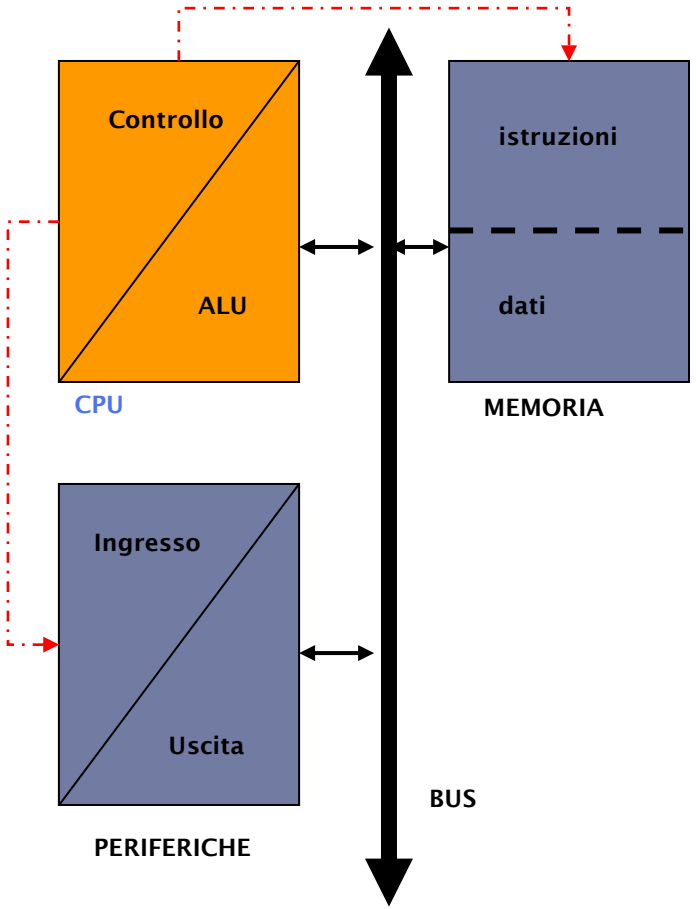
Macchina di Von Neumann

E' composta da 4 tipologie di componenti funzionali:

- **unità centrale di elaborazione** (**CPU**)
  - esegue istruzioni per l'elaborazione dei dati
  - svolge anche funzioni di controllo
- **memoria centrale**
  - memorizza e fornisce l'accesso a dati e programmi
- **interfacce di ingresso e uscita**
  - componenti di collegamento con le periferiche del calcolatore
- **bus**
  - svolge la funzionalità di trasferimento di dati e di informazioni di controllo tra le varie componenti funzionali

Modello di Von Neumann (1)

- La **CPU** (Central Processing Unit) spesso integra due distinte unità:
  - L'**Unità di controllo** si occupa di controllare tutte le operazioni del calcolatore, interpretare le istruzioni prelevate dalla memoria e inviare alle altre unità i segnali per l'esecuzione delle operazioni
  - L'**Unità aritmetico-logica**, detta **ALU** (Arithmetic & Logic Unit), fornisce la capacità di effettuare operazioni di tipo aritmetico/logico di base. Si occupa di eseguire le operazioni : somme, confronti… A volte e' affiancata da un co-processore matematico



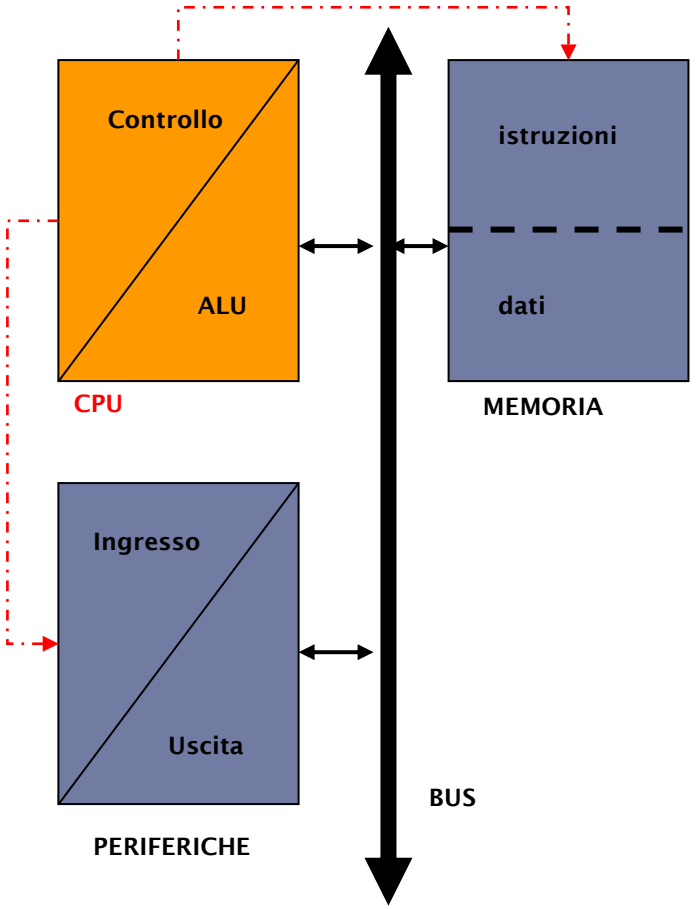
Le componenti fondamentali di un moderno calcolatore elettronico

Modello di Von Neumann (2)

- La **Memoria** centrale ha lo scopo di conservare le istruzioni e i dati da elaborare e i risultati ottenuti dalle elaborazioni;
- Le **Interfacce** collegano alle
  - **Unità di ingresso** (Input) che immette le informazioni nel calcolatore per farle elaborare;
  - **Unità di uscita** (Output) che riceve le informazioni dalla memoria del calcolatore per renderle pronte all'uso;

  le unità di ingresso e uscita

  sono anche dette **periferiche**
- Il **Bus**, vero e proprio canale di comunicazione che consente ai dati di transitare fra diversi componenti del calcolatore.



Le componenti fondamentali di un moderno calcolatore elettronico

Modello di Von Neumann (3)

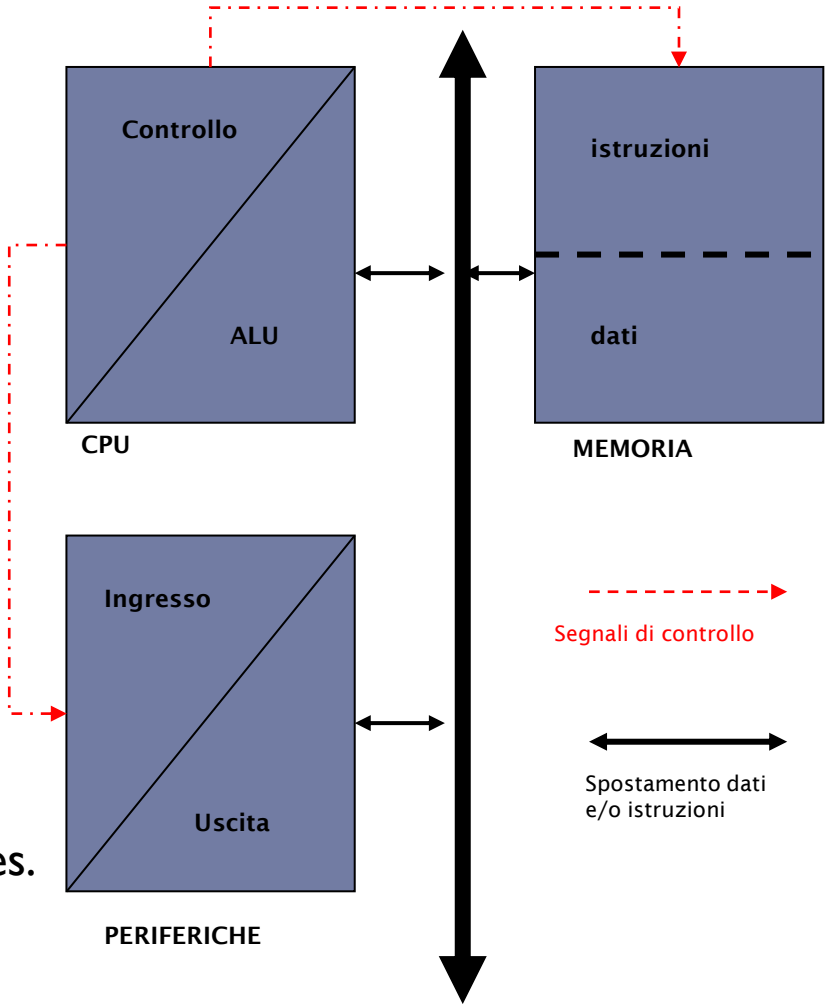
Per ogni istruzione del programma:

- la CPU, tramite la sua parte Controllo, ordina il **prelevamento** di una istruzione dalla Memoria;
- la **decodifica**, cioè la interpreta capendo quali azioni comporta;
- la **esegue** utilizzando le opportune unità coinvolte
  - durante l'esecuzione può:

    usare la **ALU**

    effettuare altri **accessi in memoria** per leggere o scrivere dati

    effettuare operazioni di ingresso (per es. leggi un dato dalla tastiera) **o di uscita** (per es. visualizza il risultato sul video).



Memorizzazione

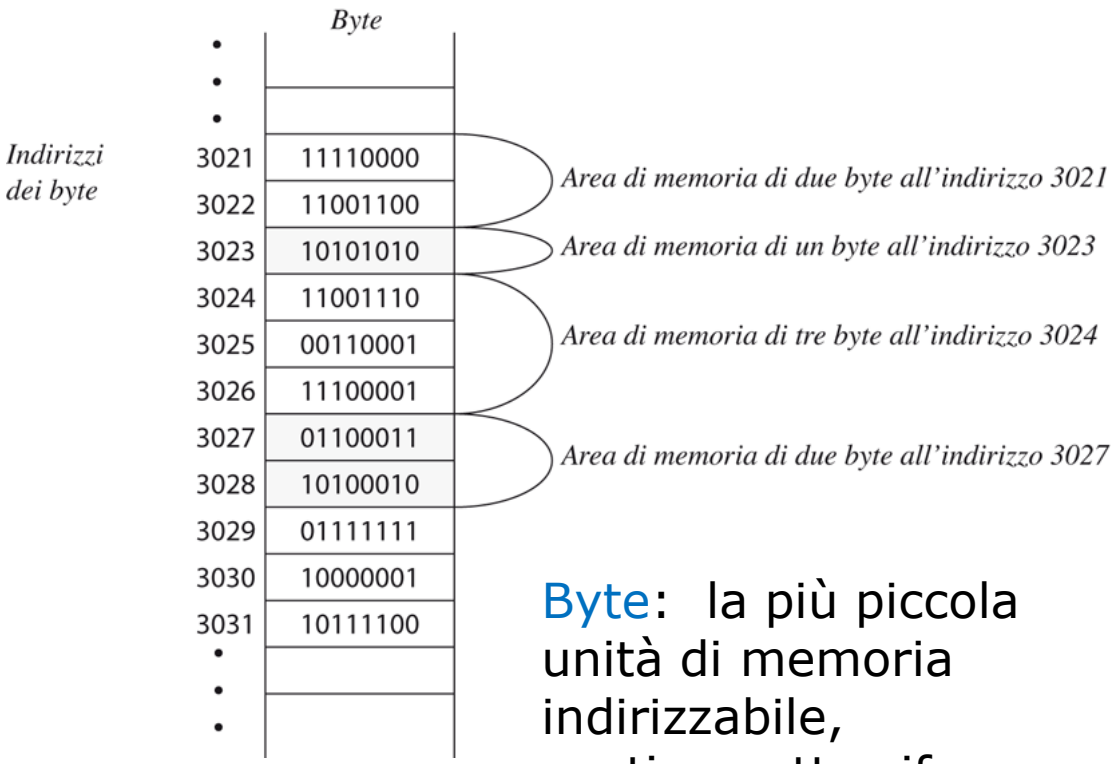
- Un calcolatore memorizza
  - i dati, che rappresentano informazioni di interesse
  - i programmi per l'elaborazione dei dati
- La memoria è l'unità responsabile della memorizzazione dei dati
- Una unità di memoria fornisce due sole operazioni
  - memorizzazione di un valore (scrittura)
  - accesso al valore memorizzato (lettura)

Memoria

Costituita da un lungo elenco di byte, ognuno con il proprio indirizzo

RAM: Volatile, contiene il programma in esecuzione e gran parte dei dati utilizzati

Memoria Ausiliaria: Conserva i dati in modo permanente. I byte sono raggruppati in unità più grandi detti file



Byte: la più piccola unità di memoria indirizzabile, contiene otto cifre binarie (bit).

Elaborazione

- Le istruzioni di un programma corrispondono ad operazioni elementari di elaborazione
  - operazioni aritmetiche
  - operazioni relazionali (confronto tra dati)
  - operazioni su caratteri e valori di verità
  - altre operazioni numeriche
- Un calcolatore sa svolgere poche tipologie di operazioni elementari ma in modo molto efficiente
  - un calcolatore può eseguire decine o centinaia di milioni di istruzioni al secondo

Linguaggio Macchina

- Ogni processore ha un proprio linguaggio macchina con un proprio formato delle istruzioni
- Le istruzioni sono sequenze di bit che codificano:
  - l'operazione da eseguire
  - I gli operandi su cui tale operazione deve essere eseguita (registri, locazioni di memoria, costanti ...)
- Il linguaggio assembler di un processore è la versione simbolica del linguaggio macchina

Linguaggio Assembler (un'idea...)

- Istruzioni di trasferimento:

```
LOAD R, x    STORE R, y    ...
```

- Istruzioni aritmetico/logiche

```
ADD R', R''     MUL R',R''    ...
```

- Istruzioni di controllo e di salto
  - salto incondizionato

```
JUMP alfa
   ...
alfa:...
```

  - salto condizionato

```
JZERO R1, alfa
   ...
alfa:...
```

L'algoritmo di Euclide in assembler (just for fun...)

```
        LOAD R1, 101
        LOAD R2, 102
alfa:   DIV R1, R2
        MUL R1, R2
        LOAD R2, 101
        SUB R2, R1
        JZERO R2, fine
        LOAD R1, 102
        STORE R1, 101
        STORE R2, 102
        JUMP alfa
fine:   LOAD R1, 102
        STORE R1, 103
```

Svantaggi del linguaggio macchina

- È necessario conoscere i dettagli dell'architettura del processore utilizzato e il relativo linguaggio
- Risulta impossibile trasportare i programmi da una macchina ad una differente
- Il programmatore si specializza nell'uso di "trucchi" legati alle caratteristiche specifiche della macchina
  - I programmi risultano difficili da comprendere e da modificare
- La struttura logica del programma é nascosta
  - Difficile comprendere il programma e correggerlo in presenza di errori

I linguaggi ad alto livello

Obiettivo: rendere la programmazione indipendente dalle caratteristiche peculiari della macchina utilizzata.

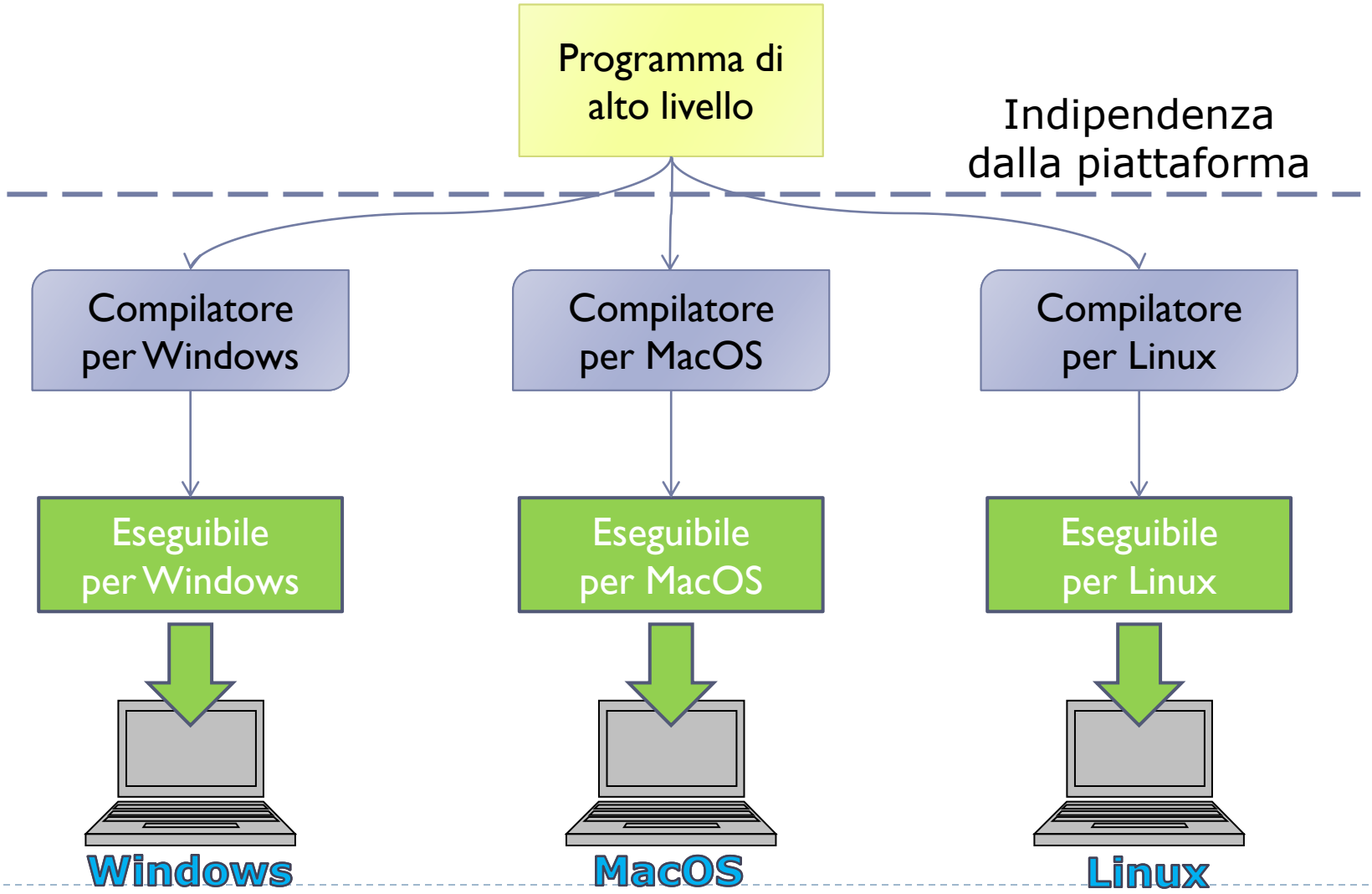
- Non sono pensati per essere compresi direttamente da macchine reali, ma da macchine "astratte" in grado di effettuare operazioni più ad alto livello rispetto alle operazioni elementari dei processori reali.

- L'attivita di programmazione viene svincolata dalla conoscenza dei dettagli architetturali della macchina utilizzata.

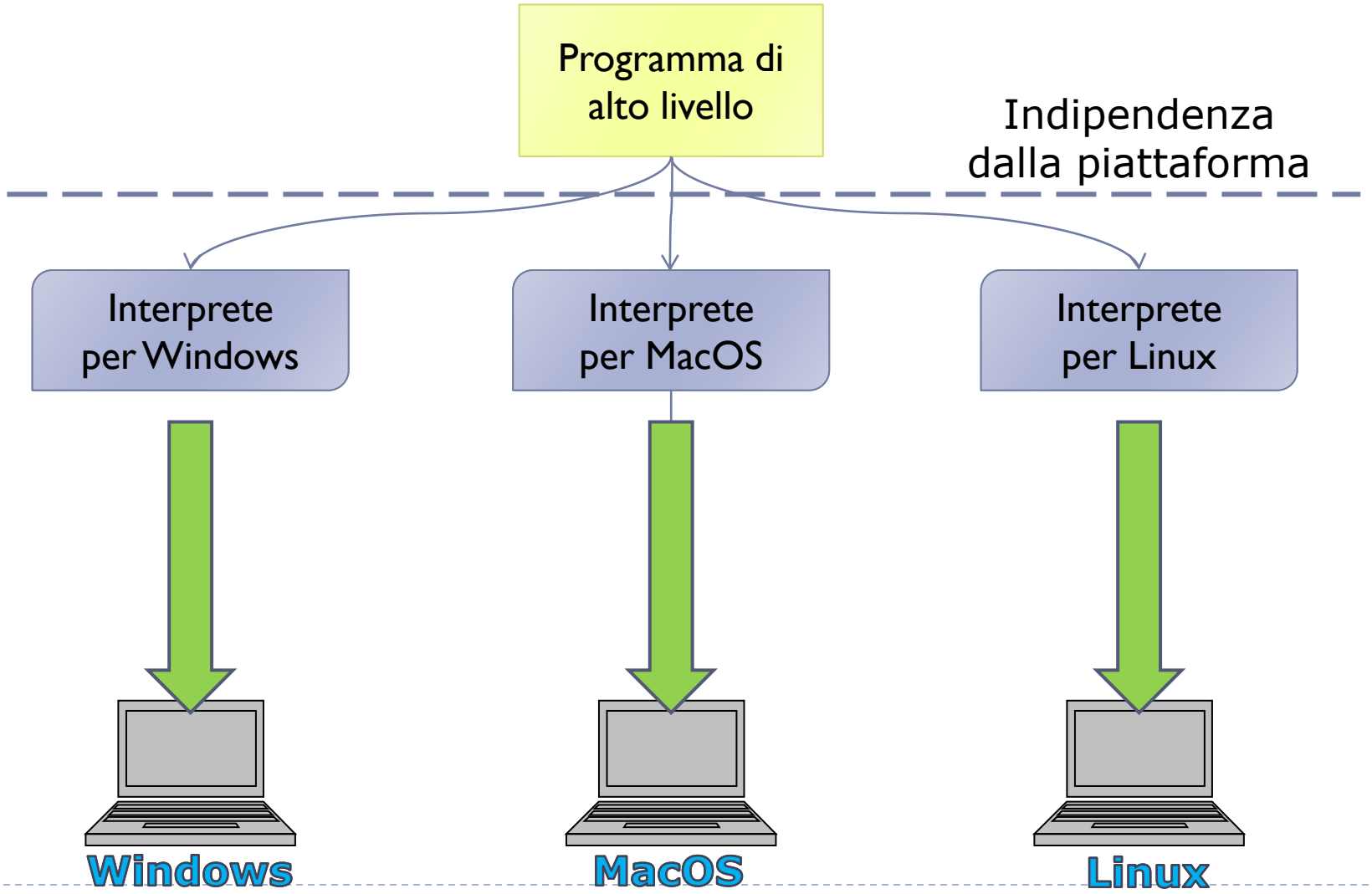
Compilatori e interpreti

- L'hardware del computer non è in grado di comprendere i linguaggi di alto livello.
- I programmi in linguaggio di altro livello devono essere tradotti in linguaggio macchina.
  - Compilatore: elabora il programma scritto in linguaggio di alto livello in modo che possa essere eseguito sulla macchina (compilazione). Una volata compilato, è possibile eseguire il programma risultante, senza doverlo ricompilare.
  - Interprete: traduce le istruzioni da un linguaggio ad alto livello a un linguaggio di basso livello. Esegue ogni singola porzione di codice subito dopo averla tradotta. La traduzione viene ripetuta a ogni esecuzione del programma.

Compilatori



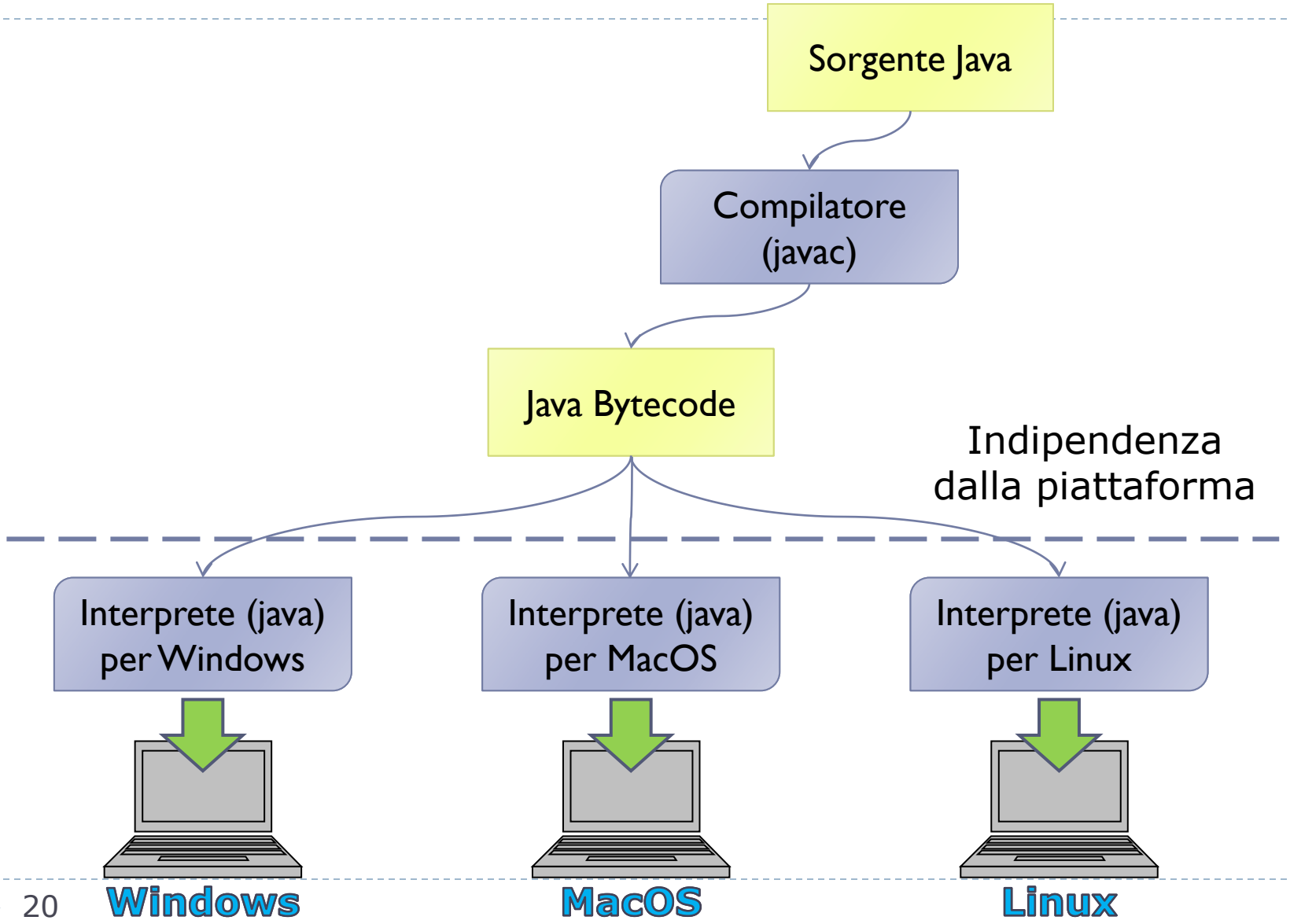
Interpreti


Programma di alto livello
Indipendenza dalla piattaforma
Interprete per Windows
Interprete per MacOS
Interprete per Linux
Windows
MacOS
Linux

Approccio di Java: il Bytecode

- Il compilatore Java traduce il programma in bytecode.
- Il bytecode è il liguaggio macchina della Java Virtual Machine (JVM).
- La JVM è un interprete che traduce il bytecode nel linguaggio macchina del computer effettivo.
- Tradurre il bytecode in linguaggio macchina è più semplice che tradurre il programma di alto livello.

Java Virtual Machine



Paradigmi di programmazione

Paradigma: un insieme di teorie, standard e metodi che rappresentano un modo di organizzare la conoscenza, cioè e un modo di guardare il mondo.

T. Kunh, La struttura delle rivoluzioni scientiche, 1970

- Un paradigma di programmazione fornisce un metodo per:
  - l concettualizzare il processo di computazione
  - l organizzare e strutturare i compiti che un calcolatore deve svolgere
- Paradigmi:
  - imperativo (Pascal, C, . . .)
  - funzionale (LISP, FP, . . .)
  - logico (Prolog)
  - ad oggetti (Java)

Java: Object-Oriented Language

Object-Oriented Programming: Si considera il programma come costituito da oggetti (o istanze) che possono agire da soli o interagire tra loro.

Gli oggetti sono caratterizzati da stato e comportamento.

Stato: Insieme delle proprietà (attributi) che caratterizzano l'oggetto in un determinato istante.

- cane: nome, colore, razza, età, . . .
- auto: colore, potenza, livello carburante, velocità, . . .

Comportamento: Insieme delle azioni che l'oggetto può eseguire.

- cane: abbaiare, scodinzolare, mangiare,. . .
- auto: accelerare, consumare, sterzare,. . .

Messaggi e Oggetti

Nella programmazione ad oggetti un'azione viene iniziata inviando un messaggio ad un agente (un oggetto) responsabile di svolgerla.

- Il messaggio codifica la richiesta di un'azione ed è corredato con l'informazione necessaria a soddisfarla (argomenti)
- Il ricevente, se accetta il messaggio, si assume la responsabilità di portare a compimento la relativa azione
- In risposta al messaggio il ricevente eseguirà un metodo per soddisfare la richiesta

Protocolli e contratti

Protocollo (interfaccia):

Definisce le regole di comunicazione con l'oggetto:

- l'insieme dei messaggi
- il formato dei messaggi

che l'oggetto può riconoscere.

Contratto

- Associato ad ogni messaggio
- Descrive il modo in cui l'oggetto garantisce di rispondere al messaggio

Classi in Java

Classe

Una classe è un modello che specifica lo stato e il comportamento di tutte le sue istanze (oggetti).

- Tutti gli oggetti sono istanze di una classe
- Il metodo utilizzato da un oggetto per rispondere a un messaggio è determinato dalla classe da cui è stato istanziato
- Tutti gli oggetti ottenuti istanziando una medesima classe rispondono ad un certo messaggio mediante il medesimo metodo

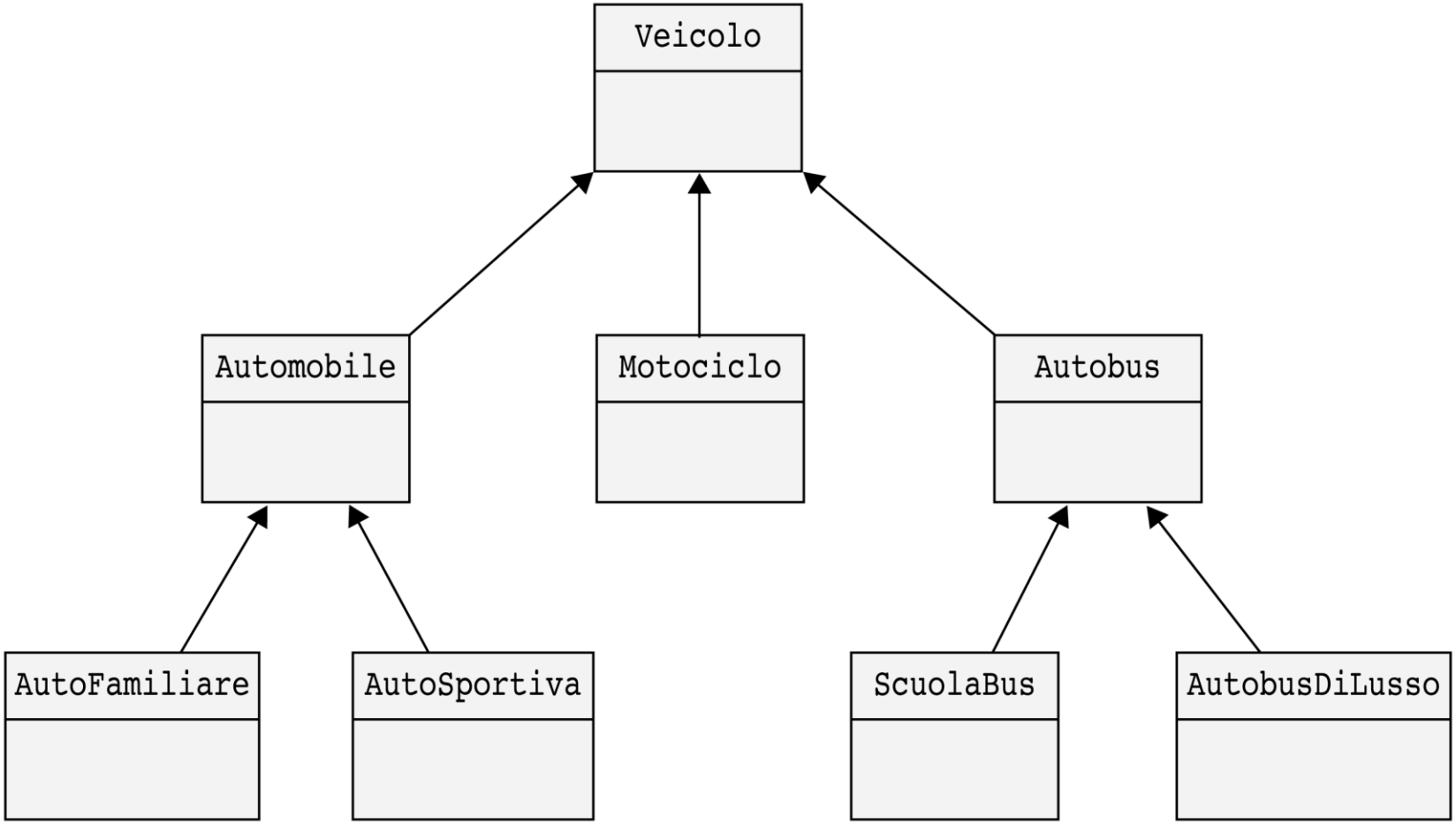
Principi Fondamentali

Incapsulamento: Si rende visibile all'utilizzatore solo una parte del programma, evitando alcuni dettagli.

Polimorfismo: Si permette ad una stessa istruzione di un programma di avere significati differenti in contesti differenti.

Ereditarietà: Ottimizza l'organizzazione delle classi. Si può definire una sola volta gli attributi e i comportamenti comuni e applicarli poi ad un intero insieme di classi.

Gerarchia di Classi



Errori (Bug)

Errori di sintassi: errori grammaticali del programma. Si violano le regole del linguaggio.

Errori a run-time: errori che avvengono durante l'esecuzione del programma (es. divisione per 0)

Errori logici: errori semantici. Si è scritto un programma corretto dal punto di vista della sintassi che non da errori run-time, ma… non fa quello che intendeva il programmatore!