Introduzione al software

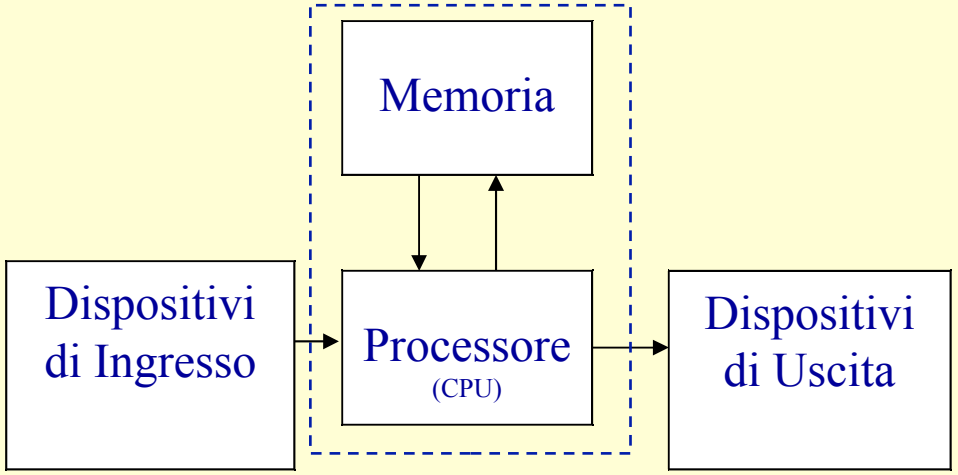
Maurizio Gabbrielli
Università di Bologna

Il calcolatore elettronico

- **Calcolatore** = hardware + software
- **Hardware**: i componenti fisici
- **Software**: le istruzioni che ``dicono'' all'hardware cosa fare (*i programmi*)

- Il calcolatore è un *esecutore di istruzioni molto efficiente ma ``poco creativo''* (fa solo ciò che gli si dice di fare)

Come è fatto un calcolatore



- **Processore (CPU)**
  - Central Processing Unit
  - Interpreta ed esegue le istruzioni
- **Memoria**
  - principale e ausiliaria
  - contiene dati ed istruzioni
- Dispositivi di Ingresso
  - tasiera, mouse ...
- Dispositivi di Uscita
  - monitor, stampante …

Two Kinds of Memory

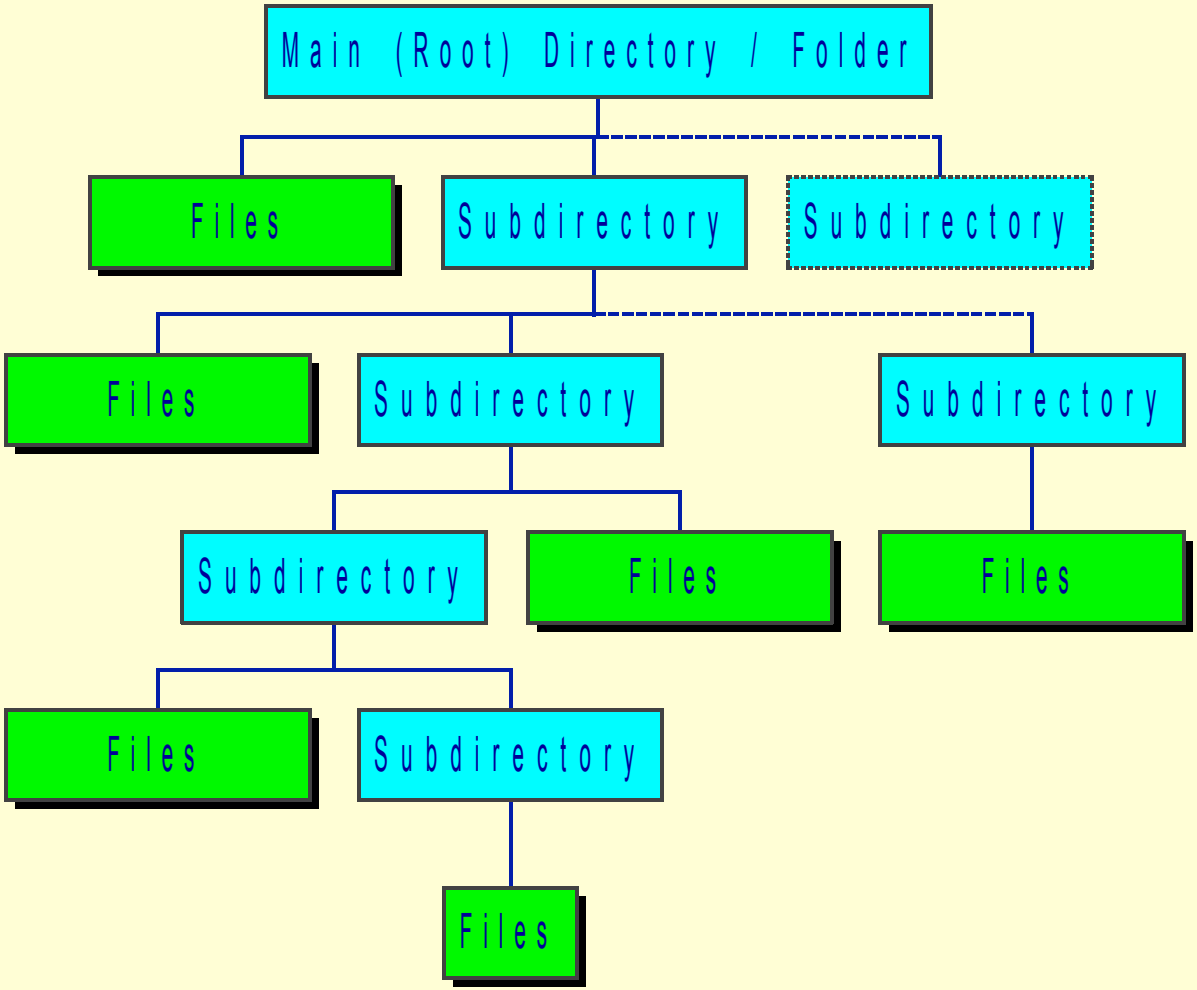
- Main
  - working area
  - temporarily stores program and data (while program is executing)
- Auxiliary
  - permanent (more or less)
  - saves program and results
  - includes floppy & hard disk drives, CDs, tape, etc.

Main Memory Organization

- Bit = one binary digit
  - Binary digit can have only one of two values, 0 or 1
- Byte = 8 bits
- "Byte Addressable"
  - Main memory is a list of numbered locations that contain one byte of data in each location
- Number of bytes per data item may vary

Address	Data Byte	
3021	1111 0000	Item 1: 2 bytes stored
3022	1100 1100	
3023	1010 1010	Item 2: 1 byte stored
3024	1100 1110	Item 3: 3 bytes stored
3025	0011 0001	
3026	1110 0001	
3027	0110 0011	Item 4: 2 bytes stored
3028	1010 0010	
3029	…	Next Item, etc.

Auxiliary Memory Organization



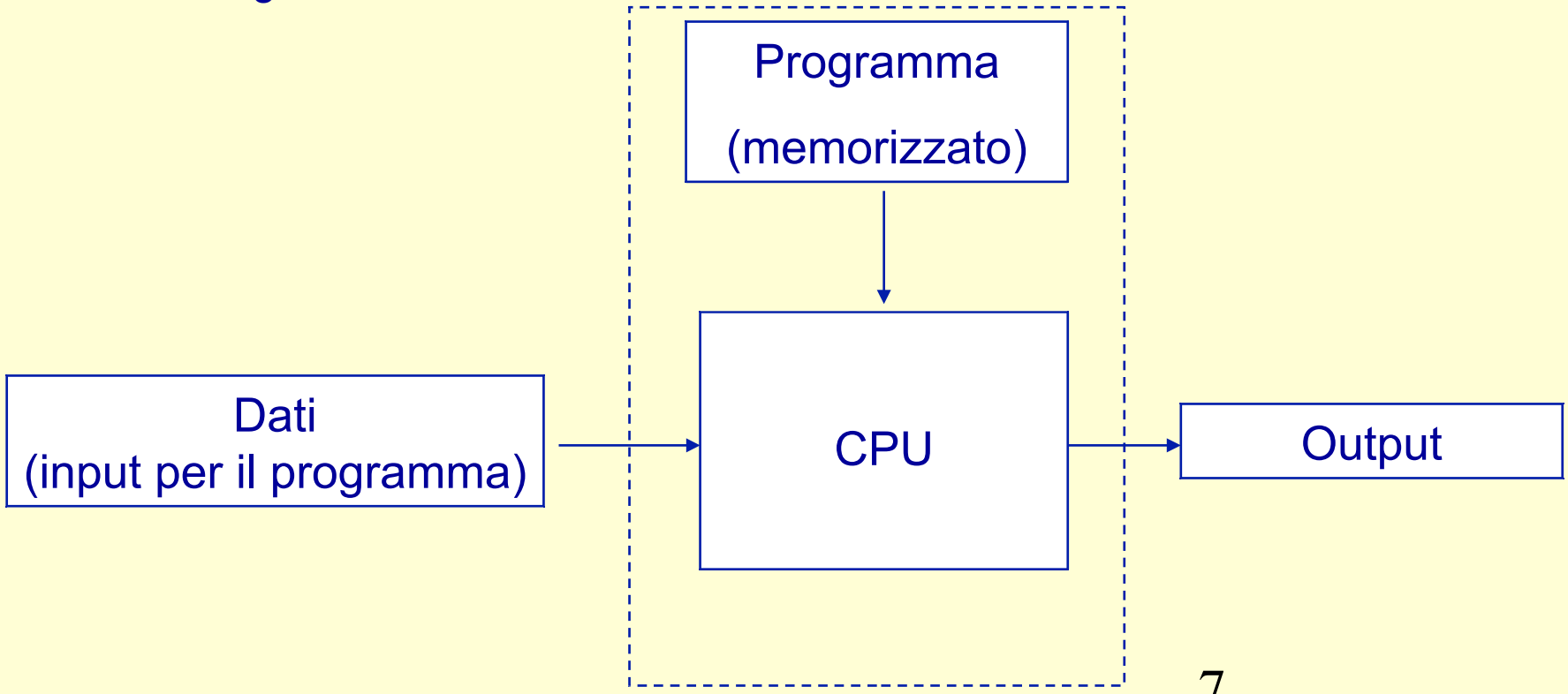
Cosa fa un calcolatore

La CPU ripete continuamente i seguenti due passi:

1) Preleva un'istruzione dalla memoria

2) Decodifica ed esegue l'istruzione, memorizza i risultato e torna al passo 1

Pogramma — in prima approssimazione un insieme di istruzioni che la CPU deve eseguire



Dall'algoritmo al programma

- **Algoritmo** = descrizione precisa dei passi da eseguire per giungere alla soluzione di un problema
  - algoritmo per la moltiplicazione
  - algoritmo per ordinare un insieme di numeri
  - ricetta di cucina ….
- Il calcolatore è un esecutore di algoritmi.
  - Perché un algoritmo possa essere eseguito esso deve essere *formalizzato in termini di un linguaggio comprensibile al calcolatore*.
- **Programma** = descrizione formale di un algoritmo in termini di un linguaggio di programmazione

Linguaggi di programmazione

- Linguaggio di programmazione: linguaggio artificiale, definito da una sintassi e da una semantica, che permette di esprimere istruzioni per un calcolatore (ossia programmi).

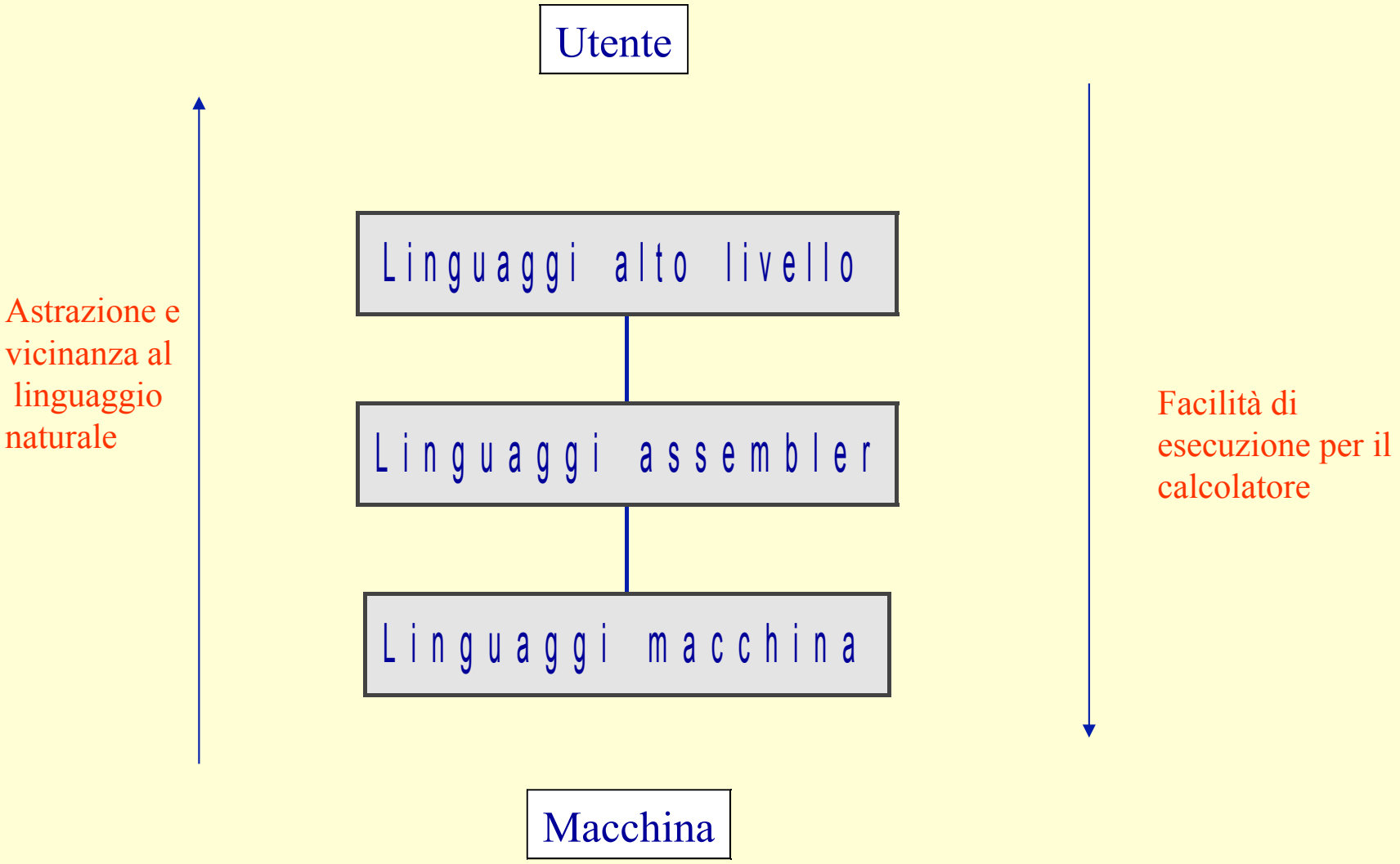
- Ogni linguaggio di programmazione consiste di :

  Operazioni di base:
  - operazioni aritmetiche
  - lettura da tastiera di un dato, scrittura su monitor di un dato ...

  Costrutti linguistici per combinare le operazioni di base:
  - eseguire una dopo l'altra una sequenza di operazioni,
  - eseguire una operazioni solo se una certa condizione è verificata
  - eseguire un'operazione per un certo numero di volte

- Ci sono varie tipologie di linguaggi di programmazione e molte centinaia di linguaggi diversi

La gerarchia dei linguaggi di programmazione



Linguaggi basso livello

- **Linguaggio macchina**
  - quello che una macchina esecutrice di algoritmi è in grado di comprendere
  - usualmente inteso come il linguaggio del calcolatore fisico (usa 0 e 1)
  - scarsamente comprensibile all'utente umano

- **Linguaggio assembly (o assembler)**
  - rappresentazione simbolica del linguaggio macchina
  - compreso dal programma assemblatore (*assembler*) che traduce *assembly* in linguaggio macchina
  - ogni ``famiglia'' di calcolatori ha il proprio linguaggio *assembly*

Linguaggi alto livello

- Linguaggi *più vicini all'utente che alla macchina*
  - vicini al linguaggio naturale
  - non direttamente comprensibili alla macchina
  - uso di notazione simbolica e meccanismi di astrazione

- Primi linguaggi alto livello definiti negli anni 50-60 (FORTRAN e ALGOL).
  - Varie centinaia di linguaggi definiti a oggi, con più di 100 linguaggi di ampia diffusione.
  - Sviluppo influenzato da vari fattori: *hardware disponibile, metodologie di programmazione, applicazioni …*

- I linguaggi moderni prediligono gli aspetti di *facilità di sviluppo e di correttezza* dei programmi

Come si esegue un programma alto livello ?

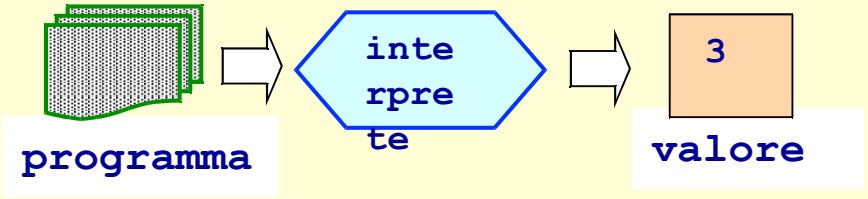
- I linguaggi alto livello sono utili per l'utente umano, ma lontani dal linguaggio ``compreso'' dalla macchina fisica

- Come rendere comprensibile al calcolatore fisico un programma alto livello ?

- Due soluzioni (spesso coesistenti)
  - Interprete
  - Compilatore

Cosa è un interprete?

- È un **valutatore** di programmi
  - segue il flusso di esecuzione del programma
  - esegue contemporaneamente la traduzione in linguaggio macchina dei comandi del programma e la loro esecuzione
  - l'output dell'interprete è il risultato della esecuzione del programma (ad esempio un numero, una stringa, …)



Cosa è un compilatore ?

- È un *traduttore* di programmi
  - traduce da linguaggio di programmazione *(linguaggio sorgente)* in linguaggio macchina *(linguaggio oggetto)*
  - l'output del compilatore è un programma in linguaggio macchina

Tipologie di linguaggi alto livello

- Tre tipi di linguaggi
  - Linguaggi imperativi
  - Linguaggi dichiarativi
    - Linguaggi Funzionali
    - Linguaggi Logici
  - Linguaggi orientati agli oggetti

- I linguaggi imperativi (Fortran, Algol, Cobol, C …) e quelli orientati agli oggetti (Java, C++, …) sono i linguaggi più diffusi

Programmazione orientata agli oggetti

- Ogni entità del sistema che si vuole descrivere è modellata mediante un oggetto.
- Un *programma è costituito da un insieme di oggetti che interagiscono*.

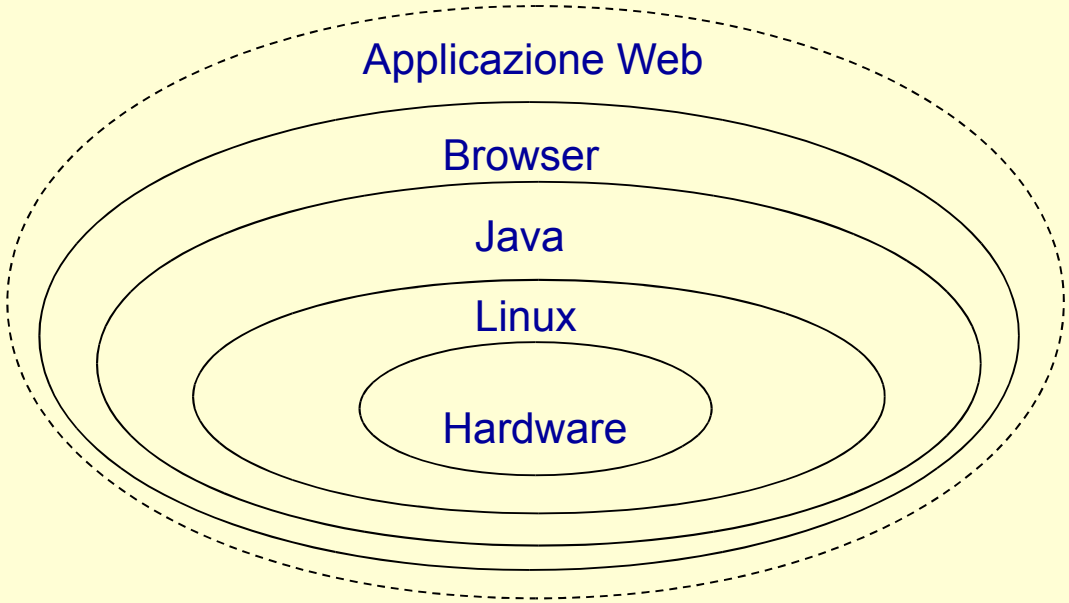
- Oggetto
  - rappresentazione di un elemento concreto del mondo reale o di un concetto
  - *costituito da uno stato e da un insieme di funzionalità*.

- Esempio: rettangolo
  - stato (campi): base, altezza, posizione
  - funzionalità (metodi): muovi, disegna, ottieni_Altezza, ottieni_Base, area

Architettura di un sistema software

- Ogni livello usa le funzionalità del livello sottostante e definisce nuove funzionalità
- Ogni livello definisce una sorta di ``computer virtuale''

Un esempio



Due tipi di software

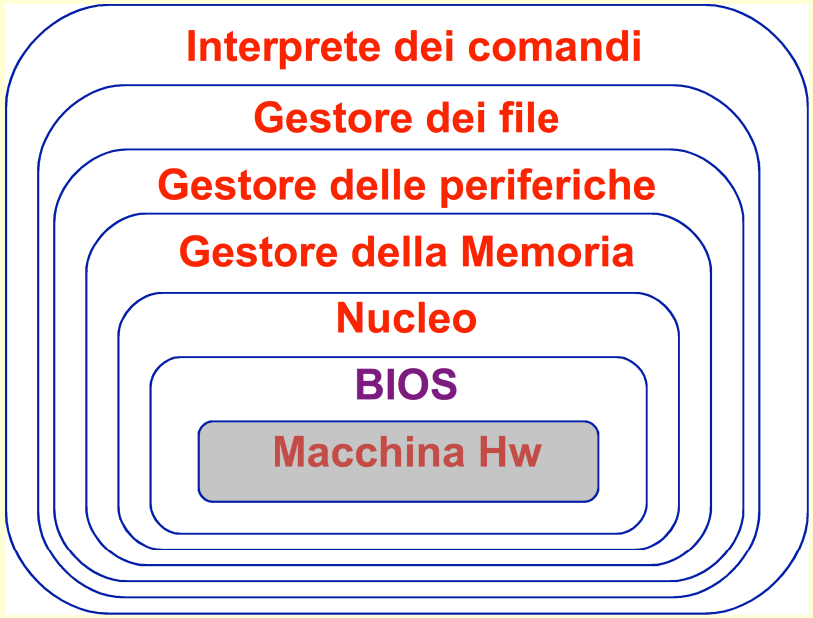
- ***Software di sistema***
  - Quello che comprende il *sistema operativo*
  - Di solito usa linguaggi che permettano anche operazioni di basso livello

- ***Software applicativo***
  - L'insieme dei programmi che realizzano le diverse applicazioni: videoscrittura, basi di dati, fogli elettronici, navigazione Internet ...
  - Vari linguaggi a seconda del tipo d'applicazione

Il sistema operativo

- Insieme di programmi che interagiscono e *controllano le funzionalità della macchina fisica*
- Estende le funzionalità della macchina fisica
- Risiede *nella memoria*
- ``Filtra'' l'interazione fra programma applicativo e hardware
- Offre un*'interfaccia amichevole* all'utente umano (menu e icone)

Componenti di un sistema operativo



Alcuni sistemi operativi

- **Unix**: sviluppato a partire dagli anni 70 presso AT&T.
  - Adatto principalmente per server di medie dimensioni.
- **DOS**: introdotto nel 1980 sul primo PC IBM,
  - interfaccia testuale.
- **MacOS**: sviluppato da Apple
  - Apple è stato il primo produttore ad offrire interfacce grafiche a finestre.
- **Windows** (95, 98, NT, 2000, …) Microsoft
  - tipico sistema operativo per PC, interfaccia grafica.
- **Linux** (anni 90): basato su Unix
  - sistema operativo per varie piattaforme basato sulla filosofia freeware, ossia è distribuito gratuitamente (si paga poi la manutenzione ed i servizi).

Software applicativo

- Insieme dei programmi che permettono di realizzare specifici compiti
  - *videoscrittura*
  - *navigazione in Internet*
  - *elaborazione di dati mediante fogli elettronici*
  - *gestione di basi di dati*
  - *elaborazioni multimediali*
  - *progettazione assistita dal computer (CAD)*
  - *commercio elettronico …….*

- Il software applicativo è un livello ulteriore costruito sul sistema operativo

Tipologie di software

- ***Software freeware***
  - Coperto da copyright, disponibile gratuitamente. Caso di Linux
- ***Software di pubblico dominio***
  - Non coperto da diritti, si può copiare liberamente
- ***Software proprietario***
  - Coperto da copyright, deve essere acquistato.
- ***Software shareware***
  - Coperto da copyright, disponibile gratuitamente ma per continuare a usarlo occorre acquistarlo.
- ***Software rentalware***
  - Coperto da copyright, viene noleggiato (a pagamento).

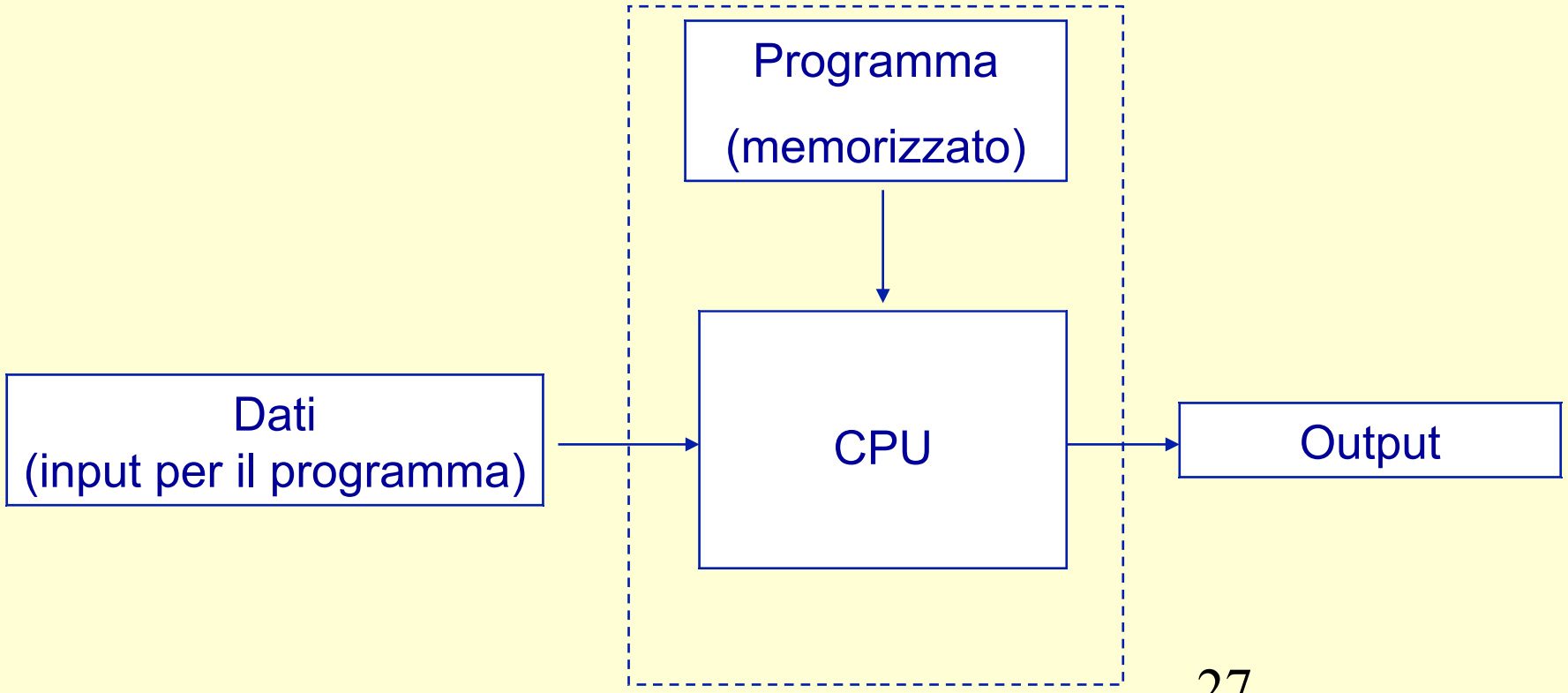
Alcune domande

- Cosa serve per poter esprimere tutti i possibili algoritmi ?
- Ci sono formalismi che permettono di esprimere più algoritmi di altri ?
- Ci sono problemi per i quali non esiste alcun algoritmo che li risolva ?

Cosa calcola un programma

- Dato lo schema visto è naturale pensare che un programma rappresenti una funzione

- Prog: Input ⟶ Output



Funzioni calcolabili: nozione intuitiva

- Una funzione f : A--> B e' calcolabile se esiste un qualsiasi algoritmo che la calcola, ossia se esiste un algoritmo P tale che: dato un qualsiasi x in A,

  se f(x) = y allora P con input X termina e produce y come output

Tesi di Church

- Church (~ 1930). Una funzione che possa essere ``calcolata da un qualsiasi algoritmo'' coincide con una funzione calcolabile da un modello matematico detto Macchina di Turing.
  - E' una tesi non dimostrata, dato che la nozione di algoritmo e' intuitiva. Pero' a tutt'oggi non si e' visto alcun controesempio.

- Le funzioni calcolabili dalla macchina di Turing coincidono con quelle calcolabili dai programmi scritti in un qualisiasi linguaggio di programmazione suficientemente generale (C, Java, Pascal ecc. ecc.).

Esistono funzioni non calcolabili ?

- Si, e sono la maggioaranza !

- Ad esempio:
  - La funzione che determina se un programma termina oppure no
  - La funzione che deteremina se due programmi “fanno la stessa cosa” (calcolano la stessa funzione)
  - ....