

Programmazione e Algoritmi 1

Introduzione

prof. Gianluca Amato

Corso di Laurea in Economia e Informatica per l'Impresa
a.a. 2024/25

7 settembre 2024

Obiettivo di queste slide:

- Presentare i concetti base di hardware e software.
- Presentare i due termini che compaiono nel nome del corso
 - Programmazione
 - Algoritmi

Parti sul libro di testo corrispondenti:

- sezioni 1.1, 1.2, 1.3, 1.7
- argomenti avanzati 1.1

Introduzione

Gianluca
Amato

HW / SW

Programmi

Algoritmi

1 Hardware e Software

2 Programmazione

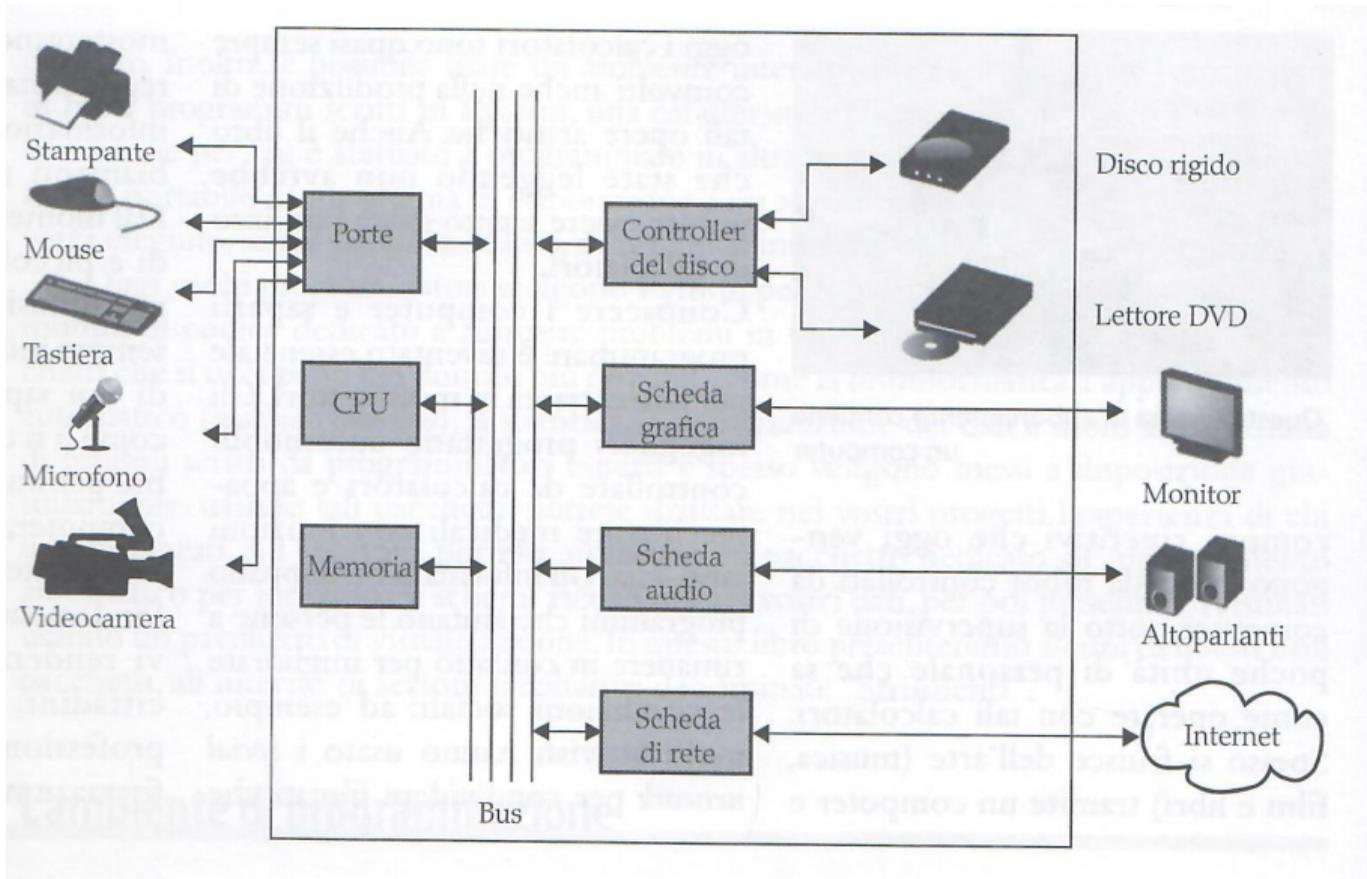
3 Algoritmi

Hardware elementi fisici che compongono un sistema di elaborazione

- tastiera
- monitor
- memoria centrale
- ...

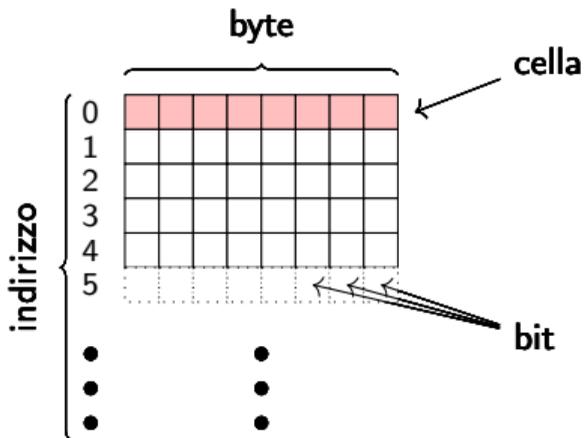
Software elementi intangibili che compongono un sistema di elaborazione

- programmi (e sinonimi, come applicazioni, app, ...)
- dati



Usata per memorizzare sia i programmi che i dati.

- Memoria centrale o principale
 - RAM (Random Access Memory)
 - lettura e scrittura
 - volatile (si cancella quando manca l'alimentazione)
 - contiene programmi e dati in corso di elaborazione
 - ROM (Read Only Memory) e varianti
 - sola lettura (quantomeno nell'uso ordinario)
 - non volatile (il contenuto permane anche in assenza di alimentazione)
 - contiene programmi necessari per l'avvio del computer
- Memoria di massa o secondaria
 - tipicamente lettura e scrittura
 - non volatile
 - più capiente ma molto più lenta della memoria centrale
 - contiene programmi e dati che non sono in uso attivo



- la memoria è divisa in celle
- ogni cella ha un indirizzo numerico
- ogni cella contiene una sequenza di 8 bit chiamata byte
- **bit** è l'acronimo di "binary digit" e indica un cifra binaria: 0 / 1
- un **byte** è un gruppo di 8 bit
 - rappresenta un numero tra 0 e 255;
 - è la quantità di memoria necessaria per memorizzare un carattere.
- se un byte non è sufficiente per memorizzare qualcosa, si possono usare più celle consecutive

Si usa “B” per indicare il singolo byte con tutti i consueti prefissi.

prefisso	nome	valore	valore alternativo
K	kilo	$10^3 = 1000$	$2^{10} = 1024$
M	mega	$10^6 = 1 \text{ milione}$	$2^{20} = 1024 \cdot 1K = 1.048.576$
G	giga	$10^9 = 1 \text{ miliardo}$	$2^{30} = 1024 \cdot 1M$
T	tera	$10^{12} = 1000 \text{ miliardi}$	$2^{40} = 1024 \cdot 1G$

Attenzione!

Di solito si usa la colonna “valore”, che è quella standard del sistema internazionale. Ma in certi contesti, ad esempio per misurare la dimensione della memoria centrale, si preferisce usare il valore alternativo, basato sulle potenze di 2.

Esempio

Una chiavetta USB da 64 GB può contenere 64 miliardi di byte.

Un PC con 64 GB di RAM ha $64 \cdot 2^{30}$ byte = 68.719.476.736 byte.

Software di sistema gestisce il computer e ne consente l'utilizzo

- sistema operativo
 - Windows / Linux / MacOS
 - Android / iOS
 - software di sistema della PlayStation
 - ...
- driver di una periferica
- ...

Software applicativo svolge una funzione specifica

- elaborazione testi (LibreOffice Writer, Microsoft Word, ...)
- videogioco
- programma di contabilità
- ...

Introduzione

Gianluca
Amato

HW / SW

Programmi

Algoritmi

1 Hardware e Software

2 Programmazione

3 Algoritmi

I “programmi” costituiscono la parte principale del software.

Definizione (Programma)

Una sequenza finita di istruzioni scritte in un linguaggio che la macchina comprende ed è in grado di eseguire.

Definizione (Programmazione)

L'attività di scrivere programmi per computer.

Per scrivere i programmi non si usa l'italiano, l'inglese o altri **linguaggi naturali**, ma linguaggi particolari chiamati **linguaggi di programmazione**.

Il computer capisce un solo linguaggio: il **linguaggio macchina** (LM)

Vantaggi

- è possibile sfruttare tutte le potenzialità dell'hardware.

Svantaggi

- estremamente complesso;
- ogni istruzione è codificata con un certo numero di byte (difficile da ricordare);
- cambia completamente da un famiglia di CPU all'altra
 - Intel/AMD a 64 bit
 - ARM (CPU usata per gli smartphone o i nuovi Mac)
 -
- il programma è strettamente legato al sistema operativo per cui è scritto: un programma in LM per Linux non funziona su Windows o Mac, neanche se la CPU è la stessa.

Esempio (Programma in linguaggio macchina)

Questo programma visualizza la scritta "Hello, world!" sullo schermo e poi termina. Funziona su PC con Linux e CPU Intel/AMD a 64 bit.

```

48 c7 c0 01 00 00 00 48 c7 c7 01 00 00 00 48 c7
c6 00 00 00 00 48 c7 c2 0f 00 00 00 0f 05 48 c7 ← byte scritto in base 16
c0 3c 00 00 00 48 c7 c7 00 00 00 00 0f 05 48 65
6c 6c 6f 2c 20 77 6f 72 6c 64 21 0a
    
```

```
.text
.global _start
_start:
    mov $1, %rax
    mov $1, %rdi
    mov $msg, %rsi
    mov $len, %rdx
    syscall
    mov $60, %rax
    mov $0, %rdi
    syscall
.data
    msg:
        .string "Hello, world!\n"
msgend:
.equ len, msgend - msg
```

Il linguaggio macchina non è fatto per gli esseri umani, e nessuno (oggi) lo usa direttamente.

In qualche occasione si usa il linguaggio **assembly**:

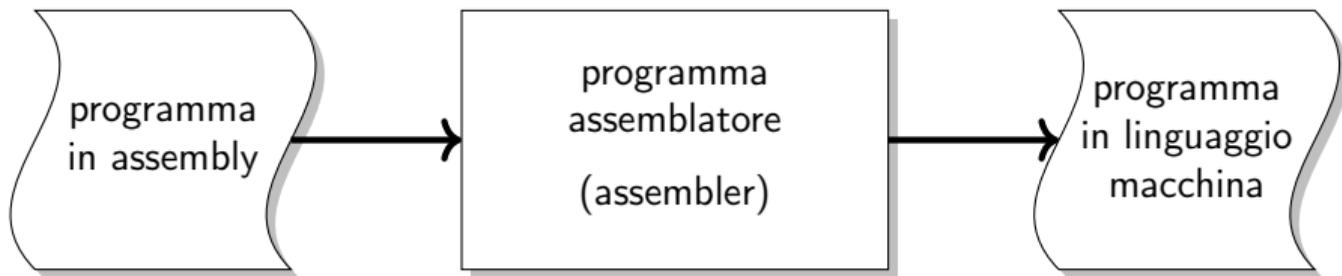
- ogni istruzione assembly corrisponde a una istruzione in linguaggio macchina;
- più facile da ricordare rispetto al linguaggio macchina.

```

1          .text
2          .global _start
3          _start:
4 0000 48C7C001  mov $1, %rax
4          000000
5 0007 48C7C701  mov $1, %rdi
5          000000
6 000e 48C7C600  mov $msg, %rsi
6          000000
7 0015 48C7C20F  mov $len, %rdx
7          000000
8 001c 0F05      syscall
8
9 001e 48C7C03C  mov $60, %rax
9          000000
10 0025 48C7C700  mov $0, %rdi
10         000000
11 002c 0F05      syscall
12
13         .data
13         msg:
14 0000 48656C6C      .string "Hello, world!\n"
14         6F2C2077
14         6F726C64
14         210A00
    
```


 istruzione in assembly


 istruzione in linguaggio macchina



Sia il linguaggio macchina che l'assembly si considerano **linguaggi a basso livello**. Normalmente si programma con **linguaggi ad alto livello**:

- più semplici da comprendere per un essere umano;
- non dipendono dalla CPU utilizzata;
- non dipendono (almeno per le cose semplici) dal sistema operativo;
- non sono eseguibili direttamente dalla CPU.

Esempio (Programma "Hello, world!" in Python)

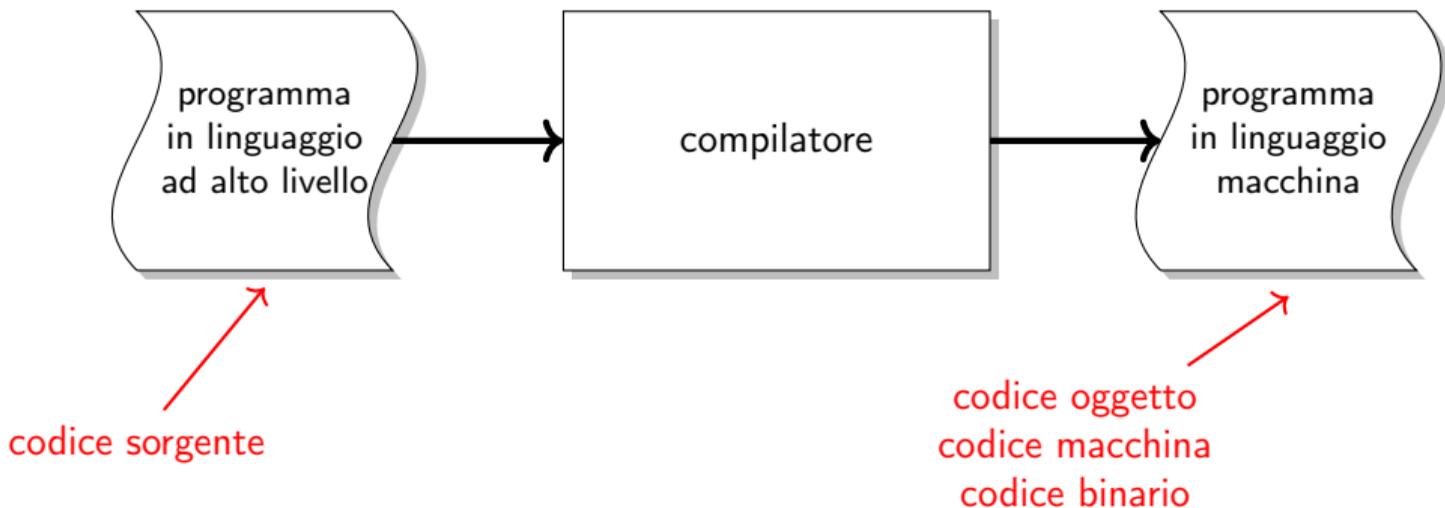
```
print("Hello, world!")
```

Come si fa a far eseguire un programma scritto in linguaggio ad alto livello?

- **compilatore**
- **interprete**
- soluzione ibrida

Legge il programma in linguaggio ad alto livello e lo traduce in linguaggio macchina tutto in una volta. Una volta tradotto il compilatore non serve più.

Esempi di linguaggi tipicamente compilati: C, C++, Rust



Legge il programma **una istruzione alla volta**, e la esegue immediatamente, senza tradurla esplicitamente in linguaggio macchina.

Esempi di linguaggi tipicamente interpretati: R, Python (?)

Vantaggi (rispetto al compilatore)

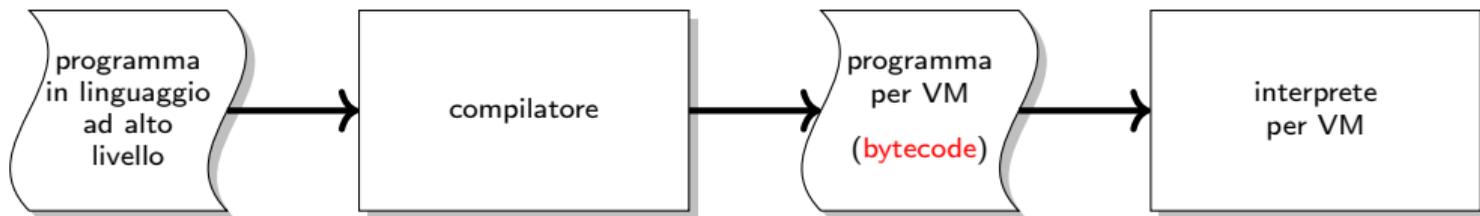
- utilizzo più immediato
 - non devo compilare un programma prima di doverlo eseguire;
 - posso eseguire anche una parte di programma o anche una singola istruzione;
- più facile da implementare.

Svantaggi (rispetto al compilatore)

- esecuzione molto più lenta.

Il programma viene compilato per una CPU virtuale, che non esiste in realtà, chiamata **macchina virtuale** (VM). Un interprete esegue la versione “compilata”.

Esempi di linguaggi che usano **esplicitamente** questa soluzione: Java, C#



La maggior parte dei linguaggi interpretati usa in realtà la soluzione ibrida anche se non lo esplicitano:

- esecuzione più efficiente di un interprete puro;
- non si perde di immediatezza: la compilazione in bytecode avviene quando necessario, ed è (quasi) completamente trasparente al programmatore.

- Python nasce negli anni '90 grazie a Guido van Rossum.
- Ci sono varie implementazioni di Python: la più comune, CPython, è un interprete che usa la metodologia ibrida.
- Nasce originariamente come **linguaggio di scripting**: un linguaggio per automatizzare procedure ripetitive nella gestione di un computer, quali creazione utenti, installazione programmi, backup, etc. . .
- Si è diffuso successivamente in altri ambiti:
 - didattica;
 - applicazioni web;
 - data science e intelligenza artificiale.
- Caratteristiche del linguaggio:
 - sintassi più semplice della maggior parte degli altri linguaggi di programmazione come C, Java, etc. . . ;
 - focalizzato sul ridurre i tempi di sviluppo del programma piuttosto che sulle prestazioni;
 - tipizzazione dinamica;
 - orientato agli oggetti.

Introduzione

Gianluca
Amato

HW / SW

Programmi

Algoritmi

1 Hardware e Software

2 Programmazione

3 Algoritmi

Definizione (Algoritmo)

Una sequenza finita di istruzioni che, se eseguite correttamente, risolvono un problema.
L'algoritmo deve essere:

- non ambiguo (esecutori diversi devono ottenere lo stesso risultato);
- eseguibile (può essere eseguito da un essere umano o da una macchina);
- terminante (l'esecuzione deve terminare).

Alcune di queste caratteristiche dipendono dal contesto in cui l'algoritmo viene eseguito: ovviamente un essere umano è in grado di svolgere delle operazioni che una macchina non può fare, e viceversa.

Problema

Vuoi bere del succo d'arancia. C'è un bicchiere pulito in credenza e una bottiglia di succo d'arancia nel frigorifero.

Algoritmo

- prendi un bicchiere pulito dalla credenza;
- apri il frigorifero;
- prendi la bottiglia di succo d'arancia;
- apri la bottiglia di succo d'arancia;
- versa il succo d'arancia nel bicchiere;
- chiudi la bottiglia di succo d'arancia;
- metti la bottiglia nel frigorifero;
- chiudi il frigorifero;
- bevi il succo d'arancia.

Problema

Avete € 10 000 in banca, ad un tasso di interesse del 5% annuo. Dopo quanti anni il vostro capitale sarà raddoppiato?

Algoritmo

- disegna una tabella con due colonne (anno e capitale);
- riempi la prima riga della tabella con anno=0 e capitale=10 000;

year	balance
0	10000

Problema

Avete € 10 000 in banca, ad un tasso di interesse del 5% annuo. Dopo quanti anni il vostro capitale sarà raddoppiato?

Algoritmo

- disegna una tabella con due colonne (anno e capitale);
- riempi la prima riga della tabella con anno=0 e capitale=10 000;
- finché il capitale non ha raggiunto il valore di 20 000, ripeti i seguenti passi:
 - aggiungi una nuova riga con il valore di anno incrementato di 1;
 - metti come valore per il capitale il valore precedente moltiplicato per 1.05;
- quando il capitale ha raggiunto il valore 20 000, il valore dell'anno è il risultato cercato.

year	balance
0	10000
1	10500
14	19799.32
15	20789.28

Nella slide di prima abbiamo descritto l'algoritmo in italiano. Alternative più comuni sono:

- un linguaggio di programmazione:
in tal caso si dice anche che il programma **implementa** l'algoritmo
- un linguaggio a metà tra il linguaggio naturale e un linguaggio di programmazione:
in tal caso si parla di **pseudocodice**

Esempio (Pseudocodice)

```

poni anno = 0
poni capitale = 10 000
finché il capitale è inferiore a 20 000
    aumenta di 1 il valore dell'anno
    moltiplica il capitale per 1.05
restituisce il valore dell'anno come risultato
    
```

Esempio (Python)

```

anno = 0
capitale = 10000
while capitale < 20000:
    anno = anno + 1
    capitale = capitale * 1.05
print(anno)
    
```

Ricordiamo le definizioni di programma e algoritmo.

Definizione (Programma)

Una sequenza finita di istruzioni scritte in un linguaggio che la macchina comprende ed è in grado di eseguire.

Definizione (Algoritmo)

Una sequenza finita di istruzioni che, se eseguite correttamente, risolvono un problema. L'algoritmo deve essere: ... *omissis* ...

Algoritmo e programma sembrano concetti molto simili, ma ci sono importanti differenze:

- un programma è scritto obbligatoriamente in un linguaggio di programmazione, un algoritmo invece può anche essere scritto in linguaggio naturale;
- un algoritmo è focalizzato alla risoluzione di un problema specifico.

In generale, un programma complesso può implementare molti algoritmi per risolvere problematiche diverse.

- Un word processor come LibreOffice Writer implementa algoritmi per
 - impaginare i documenti
 - visualizzatore il testo sullo schermo con la formattazione corretta
 - correttore ortografico
 - ...
- Un videogioco implementa algoritmi per
 - intelligenza artificiale dei personaggi non giocanti;
 - gestione della fisica del gioco;
 - generazione della grafica 3D;
 - ...