

INFORMATICA

Intelligenza Artificiale & Data Analytics

Precorsi a.a. 2024/25

Docente: Gloria Pietropolli

PRESENTAZIONI

PIETROPOLLI GLORIA

PostDoc in Computer Science - dipartimento di Matematica,
Informatica e Geoscienze

- ★ gloria.pietropolli@units.it
- ★ github.com/gpietrop/PreCorso-INF-unITS
- ★ ufficio: Via Economo (ma per queste settimane, C3)

OUTLINE DEL CORSO

20 ore di corso

- inizio corso: 09 settembre 2023
- fine corso: 21 (forse 20?) settembre 2024
- lezione tutti i giorni dalle 14:00 alle 16:00
- inizio delle lezioni **14:15**

NON ESISTONO DOMANDE STUPIDE, L'UNICA COSA STUPIDA E' NON FARE DOMANDE!

PROGRAMMA DEL CORSO

- | | |
|--|----|
| 1. Computer: che cos'è e come parla | 2h |
| 2. Il sistema operativo, file e filesystem | 3h |
| 3. Approfondimento sui file | 3h |
| 4. Linux 101 | 4h |
| 5. Version Control Systems e git | 4h |
| 6. Basi di programmazione | 4h |

TEAM DEL CORSO



Nessun Teams perché siamo a un precorso di programmazione quindi utilizzeremo GitHub

Cosa è GitHub?
Vediamolo insieme

1. COMPUTER: CHE COS'È E COME PARLA

Definizione e nostro primo modello di
computer

CHE COS'È UN COMPUTER

«A computer is a machine that can be **programmed** to carry out sequences of **arithmetic** or **logical operations** **automatically**»

Calcoli
aritmetici

Operazioni
logiche

Automaticamente
= senza
supervisione
umana

CHE COS'È UN COMPUTER?

Hardware

Parte materiale del sistema computer



Software

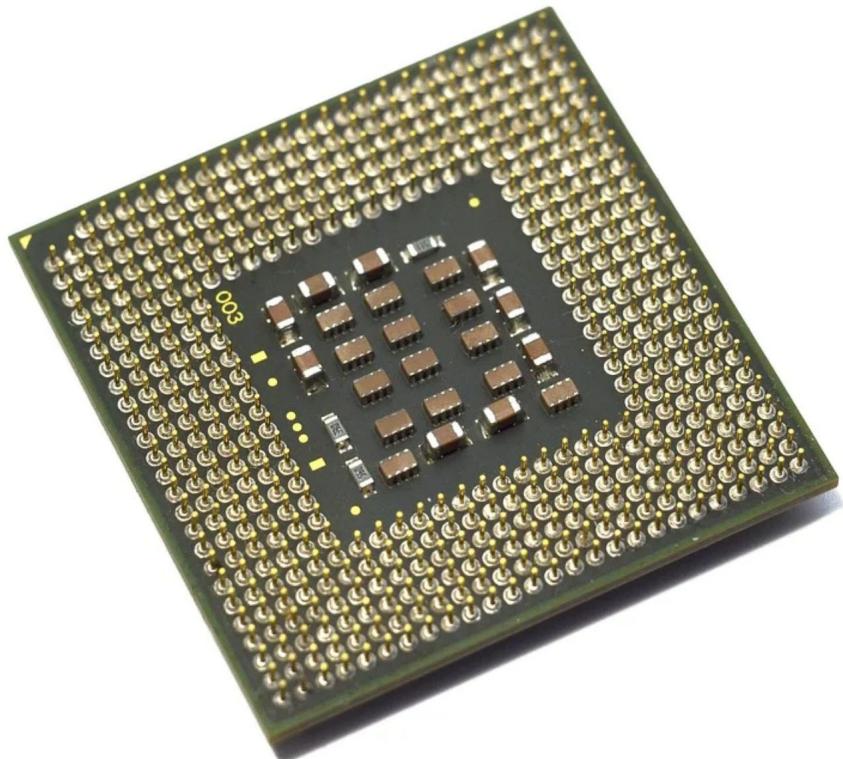
Parte immateriale/logica del sistema computer

- Dati
- Programmi (istruzioni che il computer deve svolgere)

HARDWARE – IL NOSTRO SCHEMA DI COMPUTER



HARDWARE – IL NOSTRO SCHEMA DI COMPUTER



La **CPU (Central Processing Unit)**, o **Unità di Elaborazione Centrale**, è il "cervello" del computer.

È responsabile dell'esecuzione delle istruzioni dei programmi, svolgendo calcoli e decisioni logiche.

La velocità della CPU, misurata in **GHz (Gigahertz)**, indica quante operazioni può eseguire al secondo.

Le CPU moderne contengono **più core**, che consentono di eseguire più operazioni in parallelo, migliorando la velocità di elaborazione.

HARDWARE – IL NOSTRO SCHEMA DI COMPUTER



Disco fisso (HDD/SSD): Il disco fisso è la memoria **permanente** del computer, che conserva i dati anche quando il computer è spento.

Esistono due tipi principali di dischi:

1. **HDD (Hard Disk Drive)**
2. **SSD (Solid State Drive)**

Gli SSD sono molto più veloci degli HDD perché non hanno parti meccaniche, ma sono anche più costosi.

HARDWARE – IL NOSTRO SCHEMA DI COMPUTER



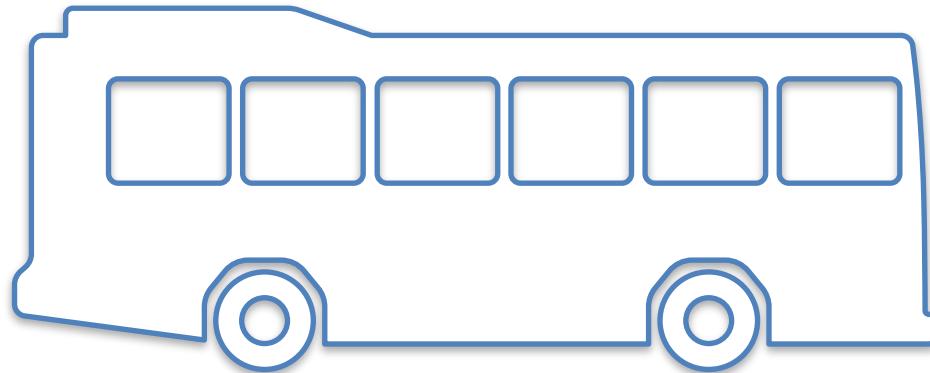
Memoria (RAM): La RAM è una memoria **volatile**, ovvero viene svuotata ogni volta che il computer viene spento.

Essa memorizza temporaneamente i dati che il processore deve elaborare.

Più RAM ha un computer, più dati può gestire contemporaneamente, migliorando la velocità e l'efficienza del sistema.

È comune trovare computer con 8GB di RAM per un uso quotidiano, mentre per attività come il gaming o l'editing video, 16GB o più sono consigliati.

HARDWARE – IL NOSTRO SCHEMA DI COMPUTER



Bus: Il **bus** è il sistema di collegamento che permette la comunicazione tra le varie componenti del computer, come la CPU, la RAM e il disco fisso.

La sua velocità di trasferimento dati, o **larghezza di banda**, è fondamentale per il corretto funzionamento del sistema, poiché influisce sulla velocità con cui le informazioni vengono trasmesse.

HARDWARE – IL NOSTRO SCHEMA DI COMPUTER



Periferiche di input/output (I/O): Le **periferiche di input** (come tastiera, mouse) e di **output** (come monitor, stampanti) sono dispositivi che permettono di interagire con il computer.

Le moderne interfacce come **USB**, **HDMI** e **Thunderbolt** consentono trasferimenti di dati ad alta velocità, facilitando la connessione con dispositivi esterni.

1. COMPUTER: CHE COS'È E COME PARLA

Come parlare ad un computer?

LINGUAGGI NATURALI



$$y = 5x^2 + \sin(x) + 10$$



IL LINGUAGGIO MACCHINA

Le informazioni sono codificate in **linguaggio binario**.

Qualsiasi dato (un numero, una lettera, un'immagine o un suono) viene rappresentato tramite **sequenze arbitrariamente lunghe di 0 e 1**. Chiamate BIT.

Anche se i dati possono sembrare complessi, sono sempre sequenze di 0 e 1.

01001001 00100000 01101100 01101111 01110110 01100101 00100000 01000001 01001001



I love AI

CODIFICA BINARIA (I)

Nella **codifica binaria**, l'alfabeto è composto dai simboli **0** e **1**.

- Bisogna quindi trovare un modo per **convertire** ogni tipo di dato in sequenze di 0 e 1.

L'esempio più intuitivo è quello della **rappresentazione dei numeri naturali**, quindi partiamo da qua!

Il nostro sistema prevede di **rappresentare i numeri naturali in base 10**.

- Qual è **l'alfabeto** del sistema decimale? (**ma poi, cosa si intende per alfabeto?**)

GLI ALFABETI

“La serie ordinata di tutti i segni (o lettere) di cui una determinata lingua dispone per indicare il sistema di scrittura relativo ai suoni vocalici o consonantici (scrittura alfabetica).”

- ❖ Alfabeto della lingua Italiana: **A, B, C, D...X, Y, Z**
- ❖ Alfabeto dei numeri in base 10: **0, 1, 2, 3, 4, 5, 6, 7, 8, 9**
- ❖ Alfabeto Musicale: **La-Si-Do-Re-Mi-Fa-Sol**
- ❖ **(e finalmente, quello che useremo noi)** Alfabeto dei numeri in base 2:
 - Abbiamo solo due simboli: **0 & 1**

CODIFICA BINARIA (II)

Possiamo rappresentare un **numero naturale** nella **codifica binaria**

- ★ $0 \rightarrow 0$
- ★ $1 \rightarrow 1$ **HO FINITO I SIMBOLI DEL MIO ALFABETO BINARIO!**

CODIFICA BINARIA (II)

Possiamo rappresentare un **numero naturale** nella **codifica binaria**

- ★ $0 \rightarrow 0$
- ★ $1 \rightarrow 1$ **HO FINITO I SIMBOLI DEL MIO ALFABETO BINARIO!**

Ma posso usare più simboli dell'alfabeto insieme, no?

- ★ $2 \rightarrow 10$
- ★ $3 \rightarrow 11$
- ★ $4 \rightarrow 100$

Ma se ho un numero alto, devo elencare tutti i numeri precedenti per trovare il modo di scriverlo in alfabeto binario?

REGOLA GENERALE PER LA CONVERSIONE

METODO DELLE DIVISIONI SUCCESSIVE

- Divido il numero per 2
- Tengo traccia del **quoziente** e del **resto**.

IL NUMERO 17

Riporto il
quoziente

$$17/2 = 8, \text{ resto } 1$$

Tengo nota del
resto

$$8/2 = 4, \text{ resto } 0$$

$$4/2 = 2, \text{ resto } 0$$

$$2/2 = 1, \text{ resto } 0$$

$$1/2 = 0, \text{ resto } 1$$

10001

Interrompo la catena quando il quoziente è 0

ESERCIZIO

Rappresentare il numero 53 in codifica binaria.

CONVERSIIONE BINARIO → DECIMALE

Ma ora devo anche capire come tornare indietro: se ho un numero scritto in base binaria come lo traduco in base decimale?

La regola è semplice: per ogni cifra del numero binario, devi moltiplicare il valore della cifra (0 o 1) per 2 elevato alla posizione della cifra, partendo da destra. Alla fine, sommi tutti i risultati.

Per fortuna è più facile a farsi che a dirsi!

CONVERSIONE BINARIO → DECIMALE

IL NUMERO 11001

$$\begin{array}{r} 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = \\ 16 \quad +8 \quad +0 \quad +0 \quad +1 \quad = \\ 25 \end{array}$$

ESERCIZIO

Rappresentare il numero 110101 in codifica decimale

CODIFICA ESADECIMALE

In informatica viene spesso utilizzata la **base esadecimale**. Questa è particolarmente utile, ad esempio, per l'indicizzazione delle posizioni di memoria.

La base esadecimale utilizza 16 simboli anziché i 10 del sistema decimale. Dato che abbiamo solo 10 cifre (0-9)

Che alfabeto utilizziamo per la base esadecimale?

Aggiungiamo 6 simboli presi dalle prime lettere dell'alfabeto: **A, B, C, D, E, F.**

CODIFICA ESADECIMALE (II)

Rappresentare un **numero naturale** nella **codifica esadecimale**

- ★ 0 → 0
- ★ 1 → 1
- ★ 10 → 10 **HO FINITO I SIMBOLI DEL MIO ALFABETO DECIMALE!**
- ★ 11 → A
- ★ 12 → B
- ★ 16 → F **HO FINITO I SIMBOLI DEL MIO ALFABETO ESADECIMALE!**
- ★ 17 → 10
- ★ 18 → 11

CODIFICA ESADECIMALE (II)

Take Home Message:
posso creare un alfabeto di lunghezza arbitraria e con i simboli che voglio!

PERCHÉ LA CODIFICA BINARIA?

Come funzionano i circuiti elettrici nei computer:

- I computer funzionano con **due stati di tensione**:
 - **0** = assenza di tensione (spento)
 - **1** = presenza di tensione (acceso)
- Questo sistema di **accensione/spento** permette al computer di:
 - Memorizzare e visualizzare informazioni come numeri, lettere e immagini.
 - Operare in modo **affidabile** grazie alla semplicità del sistema binario.
- Perché 0 e 1?
 - Sistemi con due stati sono più semplici da progettare e **meno soggetti a errori**.
 - Utilizzare più stati aumenterebbe la complessità e il rischio di errori dovuti a interferenze elettriche.

IL BIT

"Il bit (binary digit) è l'unità di misura standard dell'entropia, meglio nota come quantità di informazione"



IL BIT

Se utilizziamo più fili di corrente, possiamo rappresentare più **bit**, aumentando la complessità delle informazioni che possiamo codificare:

- **1 filo di corrente** = 1 bit = 2 stati (spento/acceso) = **0, 1**
- **2 fili di corrente** = 2 bit = 4 combinazioni = **00, 01, 10, 11**
- **3 fili di corrente** = 3 bit = 8 combinazioni = **000, 001, 010, 011, 100, 101, 110, 111**
- **4 fili di corrente** = 4 bit = 16 combinazioni = **0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111**

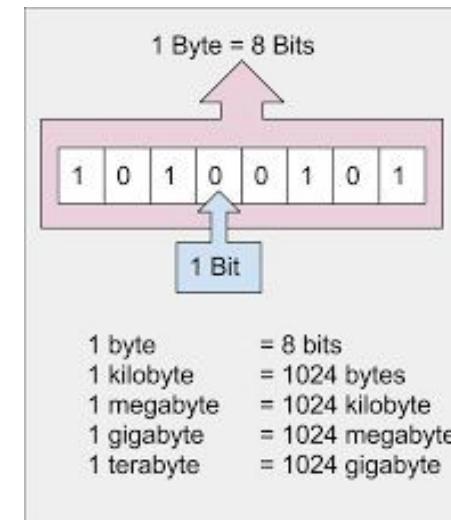
IL BYTE

- In informatica, raggruppiamo i bit in **insiemi di 8**
- Un insieme di 8 bit è chiamato **byte**
 - a. La sigla del bit nel SI è "**b**"
 - b. La sigla del byte è "**B**".
- Il byte è l'unità di misura fondamentale per la memoria



Hobbit

Hobbyte



RIASSUMENDO

- Il computer parla il linguaggio binario: 0 e 1
- Il **bit** è l'elemento base, può essere 0 o 1
- Un **byte** è un gruppo di 8 bit
- La memoria del computer utilizza elementi fisici (fori, carica elettrica) per rappresentare 0 e 1
- Memorie recenti usano semiconduttori, con la carica elettrica come indicatore del bit

1. COMPUTER: CHE COS'È E COME PARLA

La codifica dei numeri naturali e interi

MA NON MANCA QUALCOSA?

La **codifica binaria dei numeri naturali** ha un limite dovuto alla finitezza della memoria fisica. Non possiamo rappresentare un numero infinito di valori, ma dobbiamo stabilire un **limite massimo**.

Qual è il numero più grande che possiamo rappresentare con 1 byte?

- 1 byte = 8 bit
- Ogni bit può essere 0 o 1
- Le possibili combinazioni sono $2^8=256$
- Possiamo rappresentare i numeri da 0 a 255

Ma poi, c'è qualcosa che manca...

Con questa codifica non stiamo ancora rappresentando:

- Numeri negativi (Z)
- Numeri razionali (Q)
- Numeri irrazionali (R)

CODIFICA PER \mathbb{Z}

Bit del segno

Nel metodo del **bit del segno**, il bit più significativo (il primo a sinistra) rappresenta il segno del numero:

- **0** = numero positivo
- **1** = numero negativo

I restanti bit rappresentano il valore assoluto del numero. Ad esempio, per rappresentare il numero -6 con 4 bit:

- **Positivo**: 6 in binario è **0110**.
- **Negativo**: -6 diventa **1110** (dove il primo bit indica che il numero è negativo).

Limite di questo metodo: Con il bit del segno ci sono due rappresentazioni per lo zero ($+0 = 0000$, $-0 = 1000$), il che introduce ambiguità e può complicare i calcoli.

CODIFICA PER \mathbb{Z}

Complemento a due

Nel metodo del **complemento a due**, il bit più significativo (il primo a sinistra) è **anche parte del valore**. Per rappresentare un numero negativo:

1. Trova il valore assoluto del numero in binario.
2. **Inverti tutti i bit** (0 diventa 1 e viceversa).
3. **Aggiungi 1** al risultato.

Esempio per -6:

- 6 in binario è **0110**.
- Inverti i bit: **1001**.
- Aggiungi 1: **1010**.

Vantaggio del complemento a due: Non ci sono ambiguità per lo zero (esiste solo lo **0 positivo**: 0000). Inoltre, il complemento a due semplifica le operazioni aritmetiche, poiché il processore può trattare numeri negativi e positivi allo stesso modo.

Q, R?

In informatica, dobbiamo rappresentare diversi tipi di **numeri reali**, tra cui:

- Numeri razionali (es. frazioni come $1/2$)
- Numeri irrazionali (es. $\sqrt{2}$)

La rappresentazione dei **numeri reali** può richiedere **approssimazioni**, poiché molti numeri irrazionali hanno un numero infinito di cifre decimali.

Per gestire questi numeri, si utilizza la **rappresentazione in virgola mobile**, un metodo che vedrete in dettaglio nei corsi di **architettura degli elaboratori**.

CODIFICA DELLE STRINGHE [DI TESTO]

Oltre ai numeri, i computer devono anche **gestire il linguaggio naturale umano**.

Questo avviene attraverso la mappatura di caratteri in numeri.

Uno standard comune per questa mappatura è il **codice ASCII**:

- Associa a ogni carattere un **numero univoco**.
- Ogni carattere occupa **1 byte** (8 bit).
- Il codice ASCII originale utilizza **128 combinazioni** (0-127), sufficienti per lettere, numeri, segni di punteggiatura e comandi di controllo.
- Ad esempio, la parola "HELLO" viene rappresentata in ASCII come: **[72, 69, 76, 76, 79]**.

Quanti byte o bit servono per rappresentare la frase "Hello Joe"?

9 bytes (1 byte per ogni carattere, inclusi gli spazi).

CODIFICA DELLE STRINGHE [DI TESTO]

Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char	Byte	Cod.	Char
00000000	0	Null	00100000	32	Spc	01000000	64	@	01100000	96	`
00000001	1	Start of heading	00100001	33	!	01000001	65	A	01100001	97	a
00000010	2	Start of text	00100010	34	"	01000010	66	B	01100010	98	b
00000011	3	End of text	00100011	35	#	01000011	67	C	01100011	99	c
00000100	4	End of transmit	00100100	36	\$	01000100	68	D	01100100	100	d
00000101	5	Enquiry	00100101	37	%	01000101	69	E	01100101	101	e
00000110	6	Acknowledge	00100110	38	&	01000110	70	F	01100110	102	f
00000111	7	Audible bell	00100111	39	,	01000111	71	G	01100111	103	g
00001000	8	Backspace	00101000	40	(01001000	72	H	01101000	104	h
00001001	9	Horizontal tab	00101001	41)	01001001	73	I	01101001	105	i
00001010	10	Line feed	00101010	42	*	01001010	74	J	01101010	106	j
00001011	11	Vertical tab	00101011	43	+	01001011	75	K	01101011	107	k
00001100	12	Form Feed	00101100	44	,	01001100	76	L	01101100	108	l
00001101	13	Carriage return	00101101	45	-	01001101	77	M	01101101	109	m
00001110	14	Shift out	00101110	46	.	01001110	78	N	01101110	110	n
00001111	15	Shift in	00101111	47	/	01001111	79	O	01101111	111	o
00010000	16	Data link escape	00110000	48	0	01010000	80	P	01110000	112	p
00010001	17	Device control 1	00110001	49	1	01010001	81	Q	01110001	113	q
00010010	18	Device control 2	00110010	50	2	01010010	82	R	01110010	114	r
00010011	19	Device control 3	00110011	51	3	01010011	83	S	01110011	115	s
00010100	20	Device control 4	00110100	52	4	01010100	84	T	01110100	116	t
00010101	21	Neg. acknowledge	00110101	53	5	01010101	85	U	01110101	117	u
00010110	22	Synchronous idle	00110110	54	6	01010110	86	V	01110110	118	v
00010111	23	End trans. block	00110111	55	7	01010111	87	W	01110111	119	w
00011000	24	Cancel	00111000	56	8	01011000	88	X	01111000	120	x
00011001	25	End of medium	00111001	57	9	01011001	89	Y	01111001	121	y
00011010	26	Substitution	00111010	58	:	01011010	90	Z	01111010	122	z
00011011	27	Escape	00111011	59	;	01011011	91	[01111011	123	{
00011100	28	File separator	00111100	60	<	01011100	92	\	01111100	124	
00011101	29	Group separator	00111101	61	=	01011101	93]	01111101	125	}
00011110	30	Record Separator	00111110	62	>	01011110	94	^	01111110	126	~
00011111	31	Unit separator	00111111	63	?	01011111	95	_	01111111	127	Del

RIASSUMENDO

- I numeri naturali e interi possono essere convertiti in rappresentazioni binarie nei computer.
- Gli interi usano n bit (16 o 32), dove il primo bit indica il segno (+ o -) e i restanti (n-1) rappresentano il valore assoluto.
- I caratteri di testo sono associati a numeri tramite una mappa, come il codice ASCII.
- Questa mappatura permette di rappresentare simboli e caratteri utilizzando numeri naturali univoci.

1. COMPUTER: CHE COS'È E COME PARLA

Comunicare con un computer

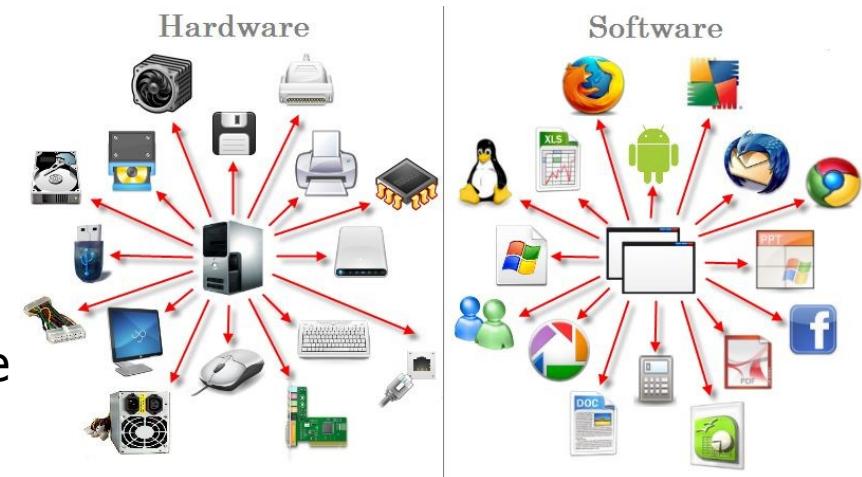
VERSO LA PARTE SOFTWARE...

Hardware: La Parte Fisica

- L'hardware è la **parte fisica** del computer, costituita da componenti materiali.
- Esempi: **processori, schede grafiche, periferiche di input/output** (tastiera, monitor, mouse, ecc.).

Software: La Parte Immateriale

- Il software è la **parte immateriale** del computer.
- Include: sistemi operativi, applicazioni, script e dati memorizzati.
- Guida il comportamento dell'hardware e permette al computer di eseguire attività.



Interazione tra Hardware e Software

- L'hardware esegue le istruzioni fornite dal software.
- **Il software sfrutta l'hardware** per completare compiti specifici.

PROGRAMMAZIONE

Che cos'è un computer?

"Un computer è una macchina che può essere programmata per eseguire automaticamente sequenze di operazioni aritmetiche o logiche."

Programmazione:

- **La programmazione** consente di **istruire il computer** a eseguire una serie di operazioni chiamate **istruzioni o programmi**.
- Le istruzioni sono scritte in un linguaggio comprensibile al computer e vengono eseguite per risolvere problemi o svolgere compiti specifici.

COME COMUNICARE?

Come il computer rappresenta i dati:

- Utilizzando il **linguaggio binario**, il computer può rappresentare informazioni come numeri o stringhe di testo.
- Ogni dato viene tradotto in sequenze di **0 e 1** (bit).

Comunicare con il computer:

- Come trasmettiamo dati al computer?
 - Tramite **input** (tastiera, mouse, ecc.), inseriamo dati da processare.
- Come istruiamo il computer a manipolare i dati?
 - Scriviamo programmi, che indicano al computer **come elaborare e trasformare** questi dati.
 - Il computer esegue **sequenze di istruzioni** per manipolare e restituire i dati.

Esempio: voglio istruire un computer a restituire il risultato della somma di due numeri a , b e dividere questa somma per un terzo numero c

Devo imparare il linguaggio binario?



Come comunico i valori di a , b , c ?

Come dico al computer di fare la somma di a , b e di dividere per c ?

FORTUNATAMENTE, NO

Comunicazione con i computer: dagli anni '50 a oggi

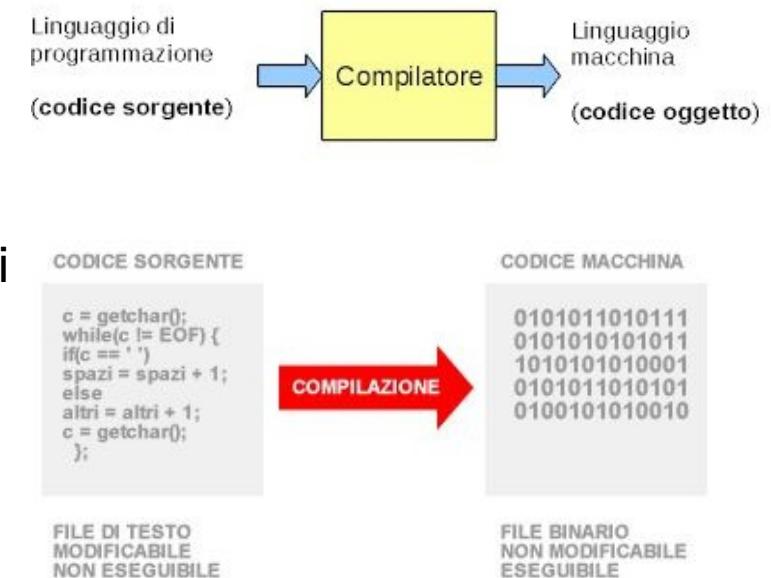
- Dagli anni '50, l'obiettivo è stato quello di **semplificare la comunicazione** tra esseri umani e computer, evitando l'uso diretto del **linguaggio binario**.
- Gli esseri umani scrivono **sequenze di istruzioni** in un formato comprensibile, chiamato **codice**.

Linguaggi di programmazione:

- Il codice è scritto in linguaggi di programmazione, progettati per essere vicini alla comprensione umana (es. Python, Java).

Compilatore o interprete:

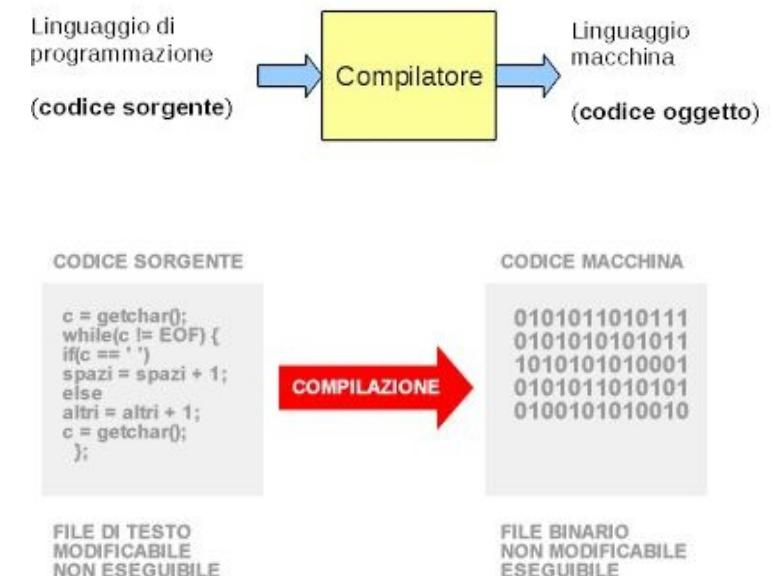
- Un programma speciale, chiamato **compilatore** o **interprete**, traduce il codice umano in **linguaggio binario** comprensibile al computer.
- Questo permette ai computer di **eseguire le istruzioni** scritte dagli esseri umani.



FORTUNATAMENTE, NO

Ora non rimane che capire come scrivere questi programmi per poter comunicare con il compilatore

Ma, cosa è un programma?
Cosa è un codice?
Quanti modi esistono di comunicare con il Computer?????



MA, CHE COS'È UN PROGRAMMA?

*“Un **programma**, in informatica, è un procedimento algoritmico applicato ad un problema dato da automatizzare”*

ALGORITMO

*"Un **algoritmo** è una strategia che serve per risolvere un problema ed è costituito da una sequenza finita di operazioni (dette anche istruzioni)..."*

ALGORITMO

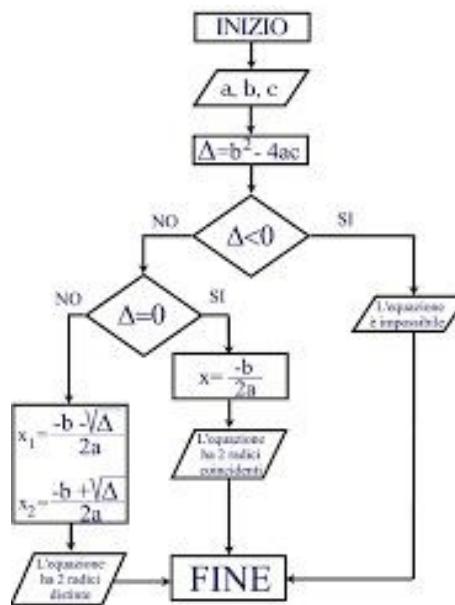
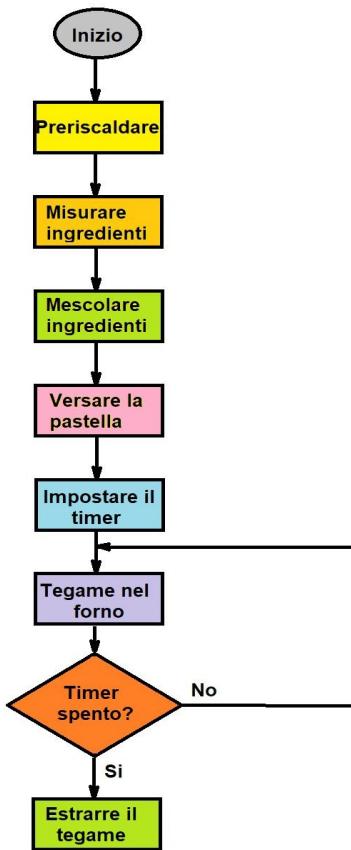
Caratteristiche degli algoritmi:

- **Ben definiti:** ogni passaggio è chiaro e preciso.
- **Finiti:** si completano in un numero finito di passaggi.
- **Efficaci:** risolvono il problema in modo efficiente.

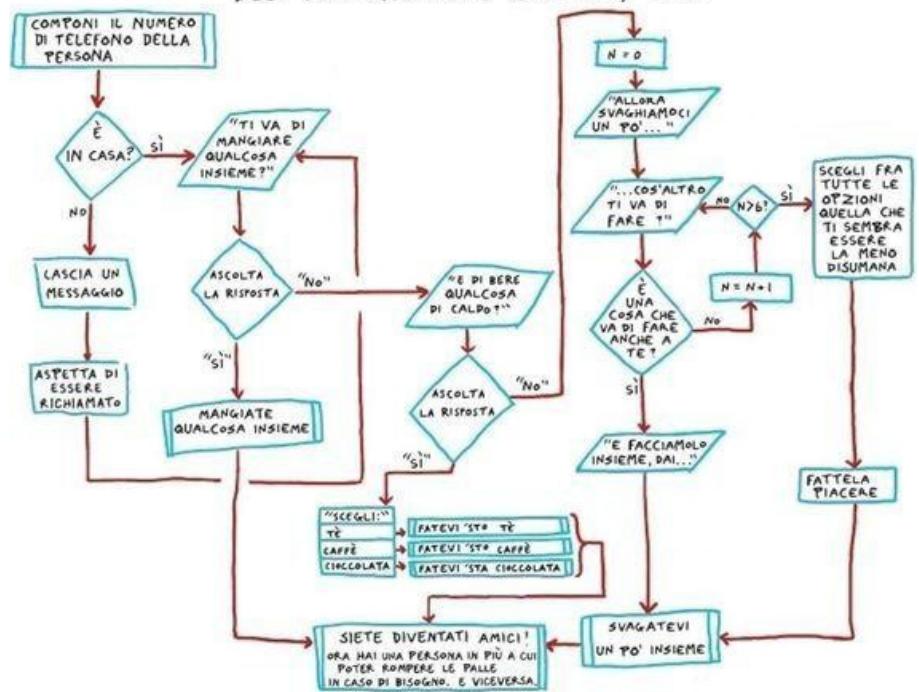
Ruolo degli algoritmi:

- Gli **algoritmi** sono fondamentali nella programmazione e nell'informatica.
- **Guidano il comportamento** del computer, fornendo le istruzioni necessarie per risolvere problemi.

ALGORITMO



L'ALGORITMO DELL'AMICIZIA DEL DR. SHELDON COOPER, Ph.D



DI NUOVO AL PROGRAMMA

Cos'è un programma?

- Un **programma** è un insieme di **istruzioni**, organizzate secondo un **algoritmo**.
- Queste istruzioni sono scritte come **linee di testo**, chiamate **codice**.

Cos'è il codice?

- Il **codice** è una sequenza ordinata di **comandi**.
- Questi comandi vengono **convertiti in linguaggio macchina**, così il computer può eseguirli.

LINGUAGGI COMPILATI E INTERPRETATI

Linguaggi Compilati:

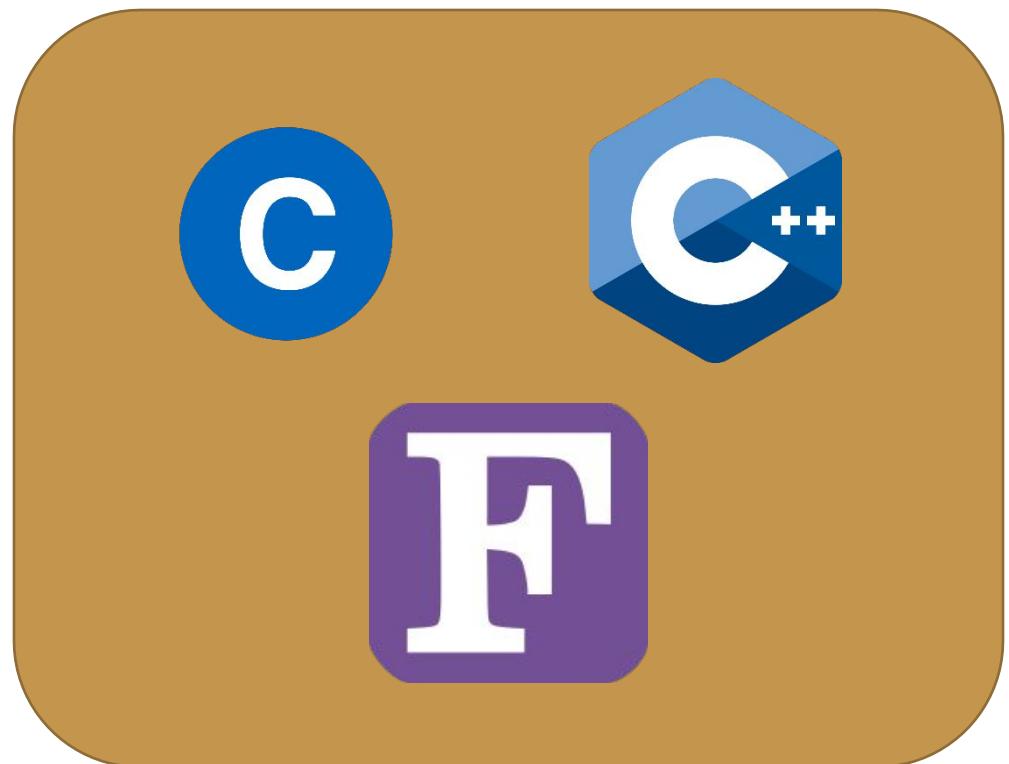
- **Esempi:** C, C++, Java.
- Il codice sorgente viene convertito in linguaggio macchina tutto in una volta tramite un compilatore.
- Vantaggi:
 - Esecuzione più **veloce** (il codice è già tradotto in linguaggio macchina).
 - Maggiore **efficienza**.
- Svantaggi:
 - Più difficile da **debuggare** (gli errori si trovano solo dopo la compilazione).
 - Deve essere **ricompilato** per ogni modifica.

Linguaggi Interpretati:

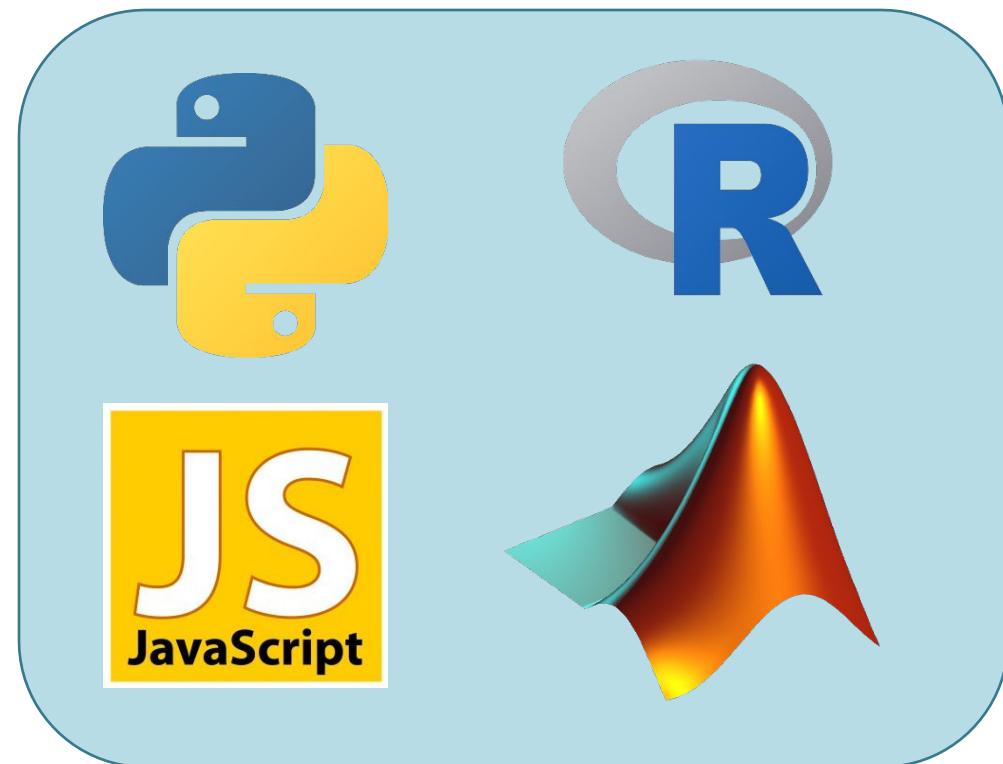
- **Esempi:** Python, JavaScript, Ruby.
- Il codice viene eseguito riga per riga da un interprete.
- Vantaggi:
 - Più facile da **debuggare** (gli errori si trovano durante l'esecuzione).
 - Nessuna necessità di **compilazione** prima dell'esecuzione.
- Svantaggi:
 - **Più lento** rispetto ai linguaggi compilati (deve tradurre il codice al momento).
 - Richiede un **interprete** installato.

LINGUAGGI COMPILATI E INTERPRETATI

Compilati



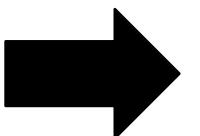
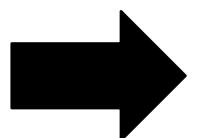
Interpretati



COMPILAZIONE (SEMPLIFICATA)

Programma scritto in
linguaggio di
programmazione

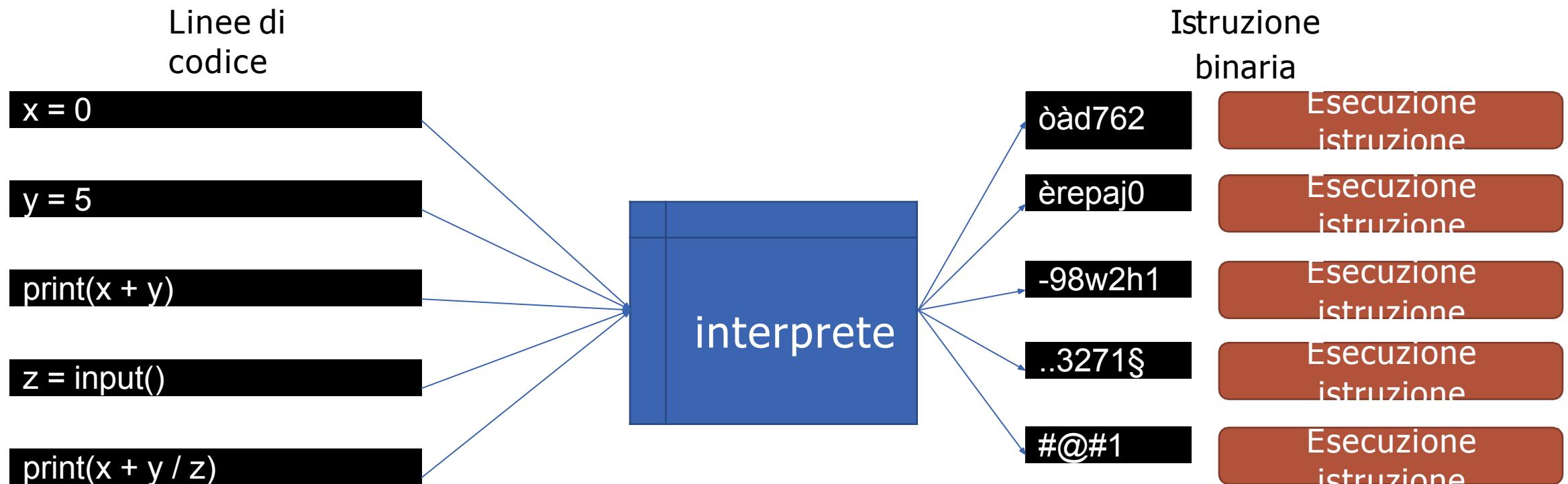
```
#include<stdio.h>
#include<math.h>
void main()
{
int oct,dec=0,i=0;
clrscr();
printf("Enter any Octal number : ");
scanf("%d",&oct);
while(oct!=0)
{
    dec=dec+(oct%10)*pow(8,i++);
    oct=oct/10;
}
printf("Decimal = %d\n", dec);
getch();
}
```



Istruzioni in linguaggio macchina

«FILE
ESSEGUITI FILE»

INTERPRETAZIONE (SEMPLIFICATA)



ISTRUZIONI E OPERATORI LOGICI

- Istruzioni matematiche:
 - Eseguono calcoli come **somma**, **sottrazione**, **moltiplicazione** o **radici**.
 - Esempio: somma di due numeri.
- Istruzioni logiche:
 - Verificano condizioni **logiche** come:
 - **E**: si verificano entrambe le condizioni.
 - **NON**: non si verifica una condizione.
 - Esempio: se **questo** E **quello** sono veri.
- Istruzioni imperative:
 - Comandi **condizionali** che eseguono azioni solo se una condizione è vera.
 - Esempio: **SE** accade qualcosa, **ALLORA** fai questo.

ESECUZIONE DEL PROGRAMMA

Si dice che il computer **esegue** un programma quando ne svolge le istruzioni contenute nel codice (compilato o interpretato).

La fase di esecuzione può essere così riassunta:

Istruzioni in linguaggio macchina contenute su disco

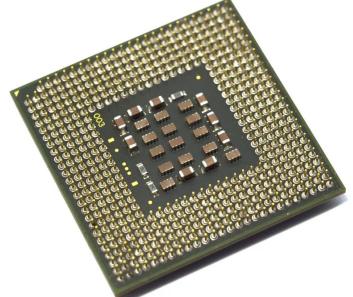


Caricamento in memoria

«Isolamento» di istruzioni «atomiche»



Passaggio
dell’istruzione
all’elaboratore che
effettua l’esecuzione
vera e propria
dell’istruzione



OPEN-SOURCE E CLOSED-SOURCE

Closed-source: il codice del programma è nascosto

Viene rilasciato solo l'eseguibile

Non si sa effettivamente che cosa faccia il programma una volta in esecuzione

Open-source: il codice è accessibile

Open source ≠ gratis

In alcuni casi è possibile apportare modifiche al programma in maniera collaborativa

RIASSUMENDO

- ★ Un **algoritmo** è un metodo o un procedimento per risolvere un problema che coinvolge una serie finita di operazioni
- ★ Un **programma** è un insieme finito di istruzioni espresse in un linguaggio di programmazione, generalmente sotto forma di codice sorgente
- ★ Le **istruzioni in un programma** possono essere di varia natura, inclusa matematica, logica, gestione dei dati e altro ancora.
- ★ L'**'esecuzione di un programma** coinvolge il caricamento del programma in memoria, l'isolamento delle operazioni in istruzioni atomiche e l'esecuzione di queste operazioni da parte della CPU.

LA PROSSIMA VOLTA...

Inizieremo a introdurre:

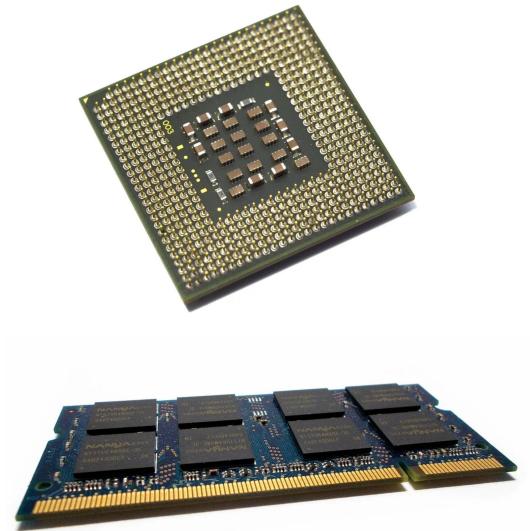
1. Il sistema operativo
2. La nozione di file
3. L'organizzazione dei file in cartelle gerarchiche e la nozione di *filesystem*

Portate un PC se ne avete uno a disposizione, ci saranno dei momenti interattivi.

HARDWARE: UNITÀ CENTRALE

Il «computer» vero è proprio (la parte che «elabora i dati») è la cosiddetta **unità centrale**, che è composta da:

- Un processore o CPU (Central Processing Unit) che è la componente che fisicamente effettua i calcoli
- Una memoria volatile o RAM (Random Access Memory) che permette il salvataggio temporaneo dei dati elaborati dalla CPU

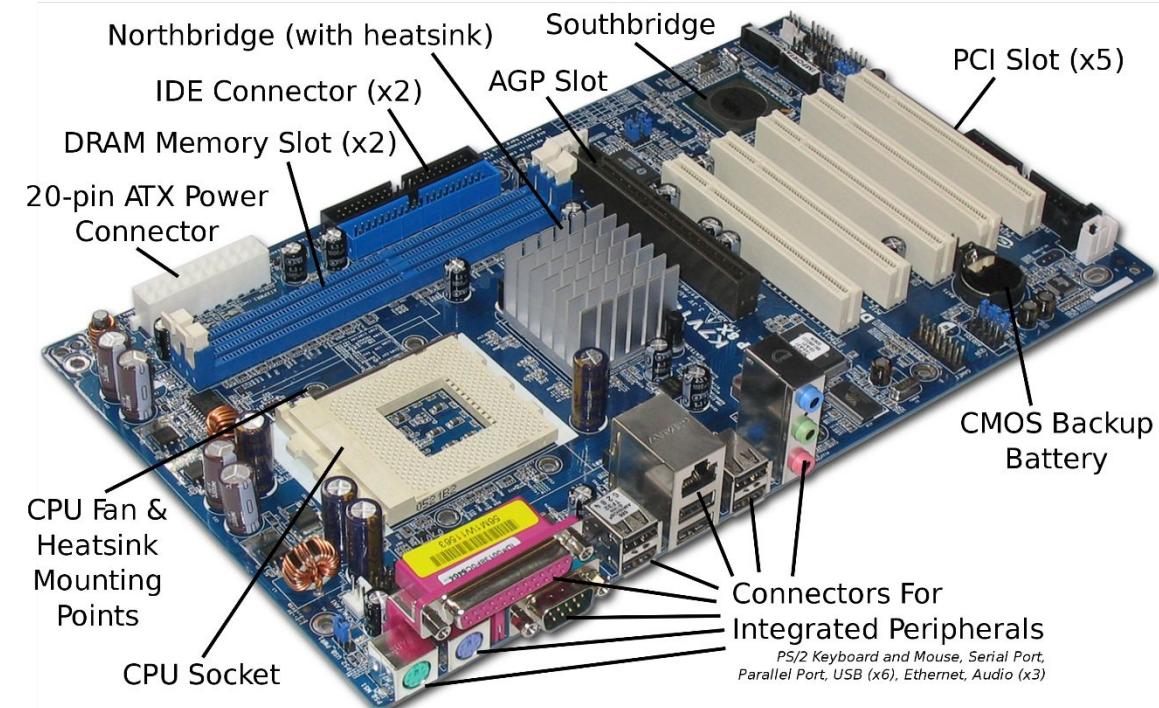


LA SCHEDA MADRE

La scheda madre o MoBo (MotherBoard) fa anch'essa parte dell'unità centrale e permette il collegamento e l'interfaccia dei componenti finora descritti.

La MoBo permette inoltre il collegamento della scheda con **l'alimentazione elettrica**.

Manca ancora qualcosa?



By user:Moxfyre - based on :Image:ASRock_K7VT4A_Pro_Mainboard.jpg by user:Darkone, CC BY-SA 2.5, <https://commons.wikimedia.org/w/index.php?curid=6544933>

PERIFERICHE DI INPUT E OUTPUT (I)

Teoricamente, un computer con le componenti finora descritte potrebbe funzionare potenzialmente all'infinito.

Il problema è che:

- Non abbiamo modo di comunicare con il computer (input)
- Il computer non ha modo di comunicare i suoi risultati a noi (output)

La quasi totalità delle componenti finora non descritte ricade nella categoria delle **periferiche di input o output (I/O)**.

PERIFERICHE DI INPUT E OUTPUT (II)

Sapete nominare 3 periferiche di input e 3 di output?

INPUT

OUTPUT

PERIFERICHE DI INPUT E OUTPUT (III)

INPUT



OUTPUT



EXCURSUS TASTIERE



Di solito possiamo classificare una tastiera in base ai primi 6 caratteri alfabetici della seconda riga
Le tastiere utilizzate in US e Italia sono di tipo QWERTY.

- Nonostante le lettere siano al medesimo posto, le tastiere IT e US differiscono per posizione e tipologia di simboli e altri caratteri

In Europa generalmente si utilizzano altri layout:

•DE (e gran parte dell'Europa continentale)→QWERTZ; FR→AZERTY... Vi è

un dibattito tuttora in corso sull'effettivo vantaggio di QWERTY & co.

- Tuttavia, sostituire il formato richiederebbe un cambio notevole nelle abitudini degli utenti
- Per questo motivo, un paradigma non efficiente (ma non terribile) ma enormemente radicato nelle abitudini di specifici utenti, si può anche denominare «il QWERTY di [qualcosa]»

PERIFERICHE DI INPUT/OUTPUT MEMORIA DI MASSA

Vi sono ancora ulteriori periferiche che possono svolgere entrambe le funzioni (chiamiamole *ibride*)

La memoria di massa è una memoria persistente che permette

- Al computer di salvare i dati temporanei presenti in RAM
- All'utente di fornire un input al computer tramite le informazioni contenute nel disco



PERIFERICHE DI INPUT/OUTPUT MODEM

Il modem è anch'esso una periferica ibrida in quanto permette un flusso, alternativo rispetto alle memorie di massa, sia in entrata che in uscita



ALTRÉ PERIFERICHE

Vi sono altre periferiche che sono di supporto a quelle finora indicate, ma che non sono necessariamente da classificare come input o output.



La scheda di rete permette al computer di interfacciarsi con il modem affinché possa inviare a e ricevere dati da quest'ultimo

La scheda video (GPU) offre supporto alla CPU permettendo di svolgere in maniera più rapida i calcoli necessari alla riproduzione delle immagini sui monitor. Si può quindi pensare alla GPU come una CPU con capacità ridotte.



PERIFERICHE: NOTE CONCLUSIVE

Ci sono ancora altre periferiche non coperte in queste slide (es. schede audio).

Molte periferiche «di supporto» nei computer recenti si trovano integrate all'interno della scheda madre (audio, rete...) o CPU (scheda grafica).

RIASSUNTO

Possiamo visualizzare un computer come una composizione delle sue componenti materiali (*hardware*) e immateriali (*software*).

La *base* del computer è l'unità centrale, composta da CPU (calcolatore in senso stretto), memoria [RAM] e scheda madre.

Le altre componenti sono usualmente

- Input e/o output (I/O)
- Periferiche di supporto alle componenti I/O



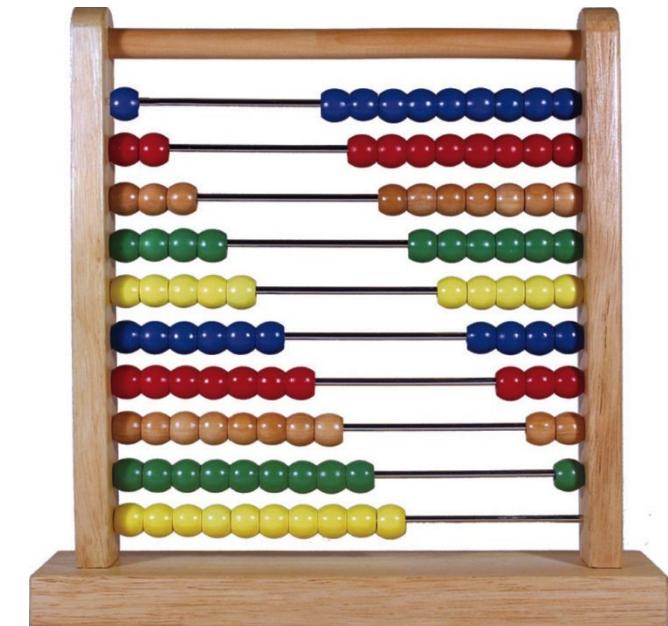
STORIA DEL COMPUTER

Excursus storico

CHE COSA PUÒ ESSERE EFFETTIVAMENTE DEFINITO «COMPUTER»?

Nel corso della storia, gli umani hanno spesso necessitato di macchinari per l'ausilio di calcoli basilari.

Se accettiamo qualsiasi supporto fisico per il calcolo come «computer», allora probabilmente il primo computer della storia è l'abaco, utilizzato da più di 4000 anni in Cina e Mesopotamia.



IL PRIMO «VERO» COMPUTER

L'abaco però è solamente un *supporto*, che facilità l'umano nel compito di tenere i conti, ma non li effettua in autonomia.

Il meccanismo di Antikythera era un calcolatore meccanico per effettuare previsioni accurate sulla posizione di stelle, pianeti ed eclissi con anni di anticipo. Si stima sia stato costruito fra il II e il I secolo a.C.

I resti sono stati trovati nel 1902 all'interno del relitto di una nave romana scoperto anni prima a largo dell'isola ellenica di Antikythera.

Possiamo definirlo un computer vero e proprio perché, ad un input dell'utente, produceva un output (le posizioni astronomiche) conducendo autonomamente le computazioni necessarie.



IL PRIMO COMPUTER MODERNO

Nel 1833, il matematico britannico Charles Babbage progettò (su carta) la macchina analitica, il primo progetto di **calcolatore programmabile** moderno.

Dati e istruzioni venivano passati sotto forma di schede forate. Anche l'output veniva prodotto in forma di schede forate, dacché la macchina era provvista di una «stampante» per la perforazione del cartone.

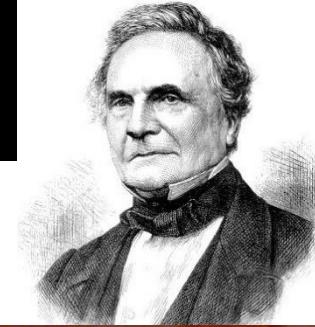
La RAM era composta da una serie di ingranaggi in grado di tenere 1000 numeri con 40 cifre decimali (ca. 17 kB di memoria).

La CPU era anch'essa composta da una serie di ingranaggi e le sue procedure erano *programmabili* tramite l'inserimento di pioli ad opera del programmatore. Poteva effettuare le operazioni $+- \times \sqrt{<>} = \neq$.

Nonostante gli impegni di Babbage, il computer non fu mai costruito in quanto il progetto era troppo avanzato rispetto alle tecnologie ingegneristiche dell'epoca.

Alcune varianti sono state costruite negli anni, l'ultima delle quali nel 1991.

La matematica britannica Ada Lovelace creò, nel 1843, dei programmi per la macchina analitica (utilizzando il linguaggio di programmazione a schede forate) ed è per questo definita il primo programmatore della storia.
(NB: Ada Lovelace day → 12 ottobre '21)

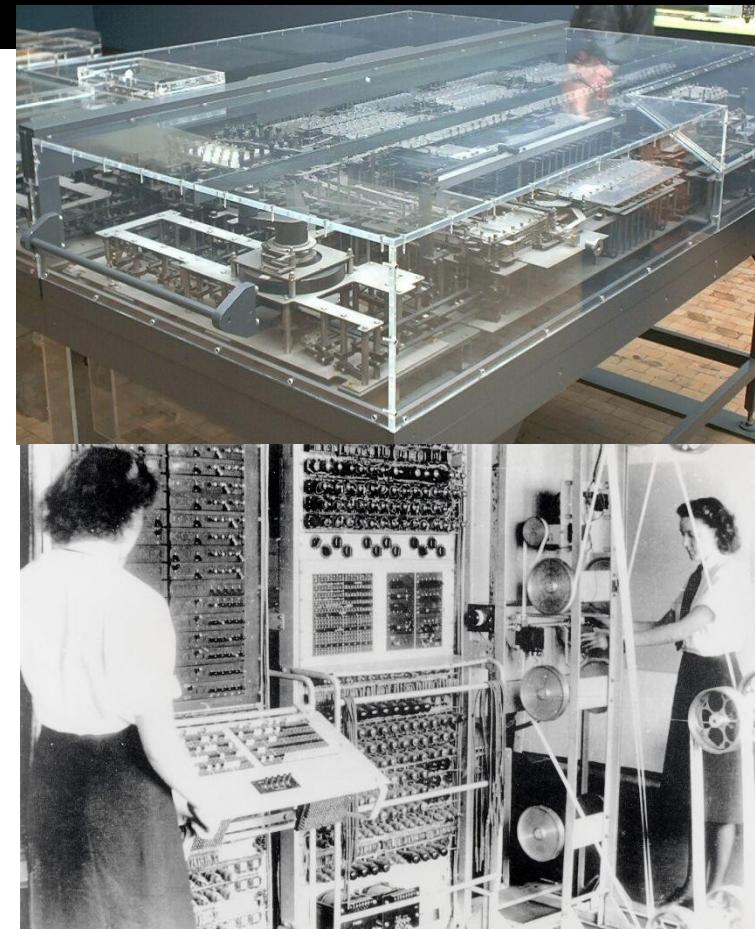


I PRIMI COMPUTER FUNZIONANTI

Dagli anni '30 del '900, si iniziano a vedere investimenti notevoli per la costruzione del primo computer programmabile *funzionante*.

Il titolo va a Konrad Zuse, ingegnere tedesco, che nel '39 costruì lo Z1.

Nel '44, inoltre, il Colossus, costruito in Regno Unito da parte di un team guidato dal «padre dell'informatica» Alan Turing, portò alla decifrazione del codice Enigma, utilizzato dall'Asse per criptare i messaggi segreti. Questo evento contribuì notevolmente alla vittoria alleata nella II guerra.



EVOLUZIONE DELL'I/O

Questi primi computer utilizzavano ancora l'idea «antica», già teorizzata da Babbage, di utilizzare schede forate per l'input.

Negli anni '50 si inizia ad utilizzare un monitor come periferica di output (anziché stampe o altre schede forate). La tastiera invece arrivò appena negli anni '70, mentre il mouse negli anni '80.



By Photographed by Andreas Franzkowiak (User:Bullenwächter) - Own work, CC BY-SA 4.0, <https://commons.wikimedia.org/w/index.php?curid=61373639>

IL WHIRLWIND, OVVERO L'ANTENATO DEI «PERSONAL COMPUTER»

Il Whirlwind è famoso per due motivi:

1. È il primo computer ad essere definito come «real-time», ovvero i cui tempi di elaborazione delle informazioni erano talmente rapidi da sembrare che una risposta ad un'interazione utente avvenisse «istantaneamente»
2. Era noleggiabile da chiunque per 15 minuti. Questo fece sì che anche i non-ricercatori potessero utilizzarlo e ciò contribuì alla nascita di un nuovo *bisogno*, ovvero quello di avere un computer personale per sé

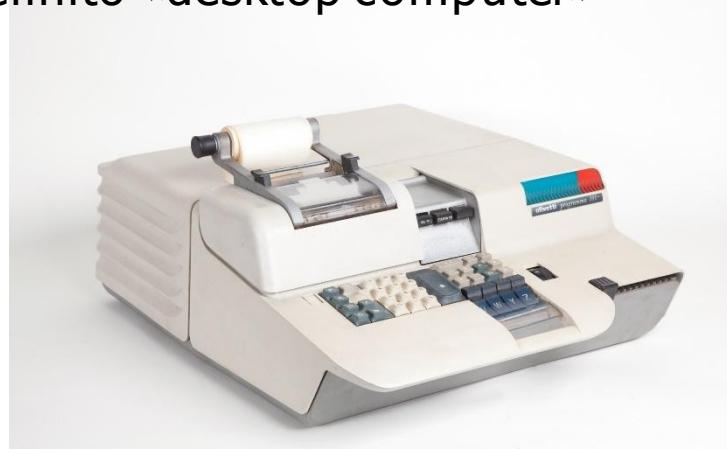
By Daderot - Own work, Public Domain, <https://commons.wikimedia.org/w/index.php?curid=8894406>



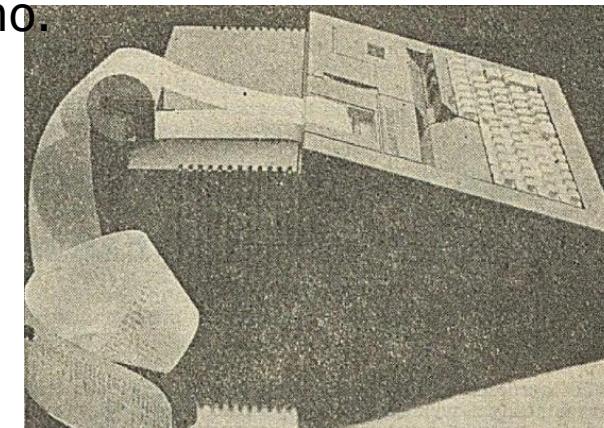
I COMPUTER OLIVETTI

L'azienda italiana Olivetti occupa un posto importante nello sviluppo dei personal computer dagli anni '60.

Olivetti «Programma 101» è un calcolatore programmabile, uno dei primi al mondo ad essere definito «desktop computer»

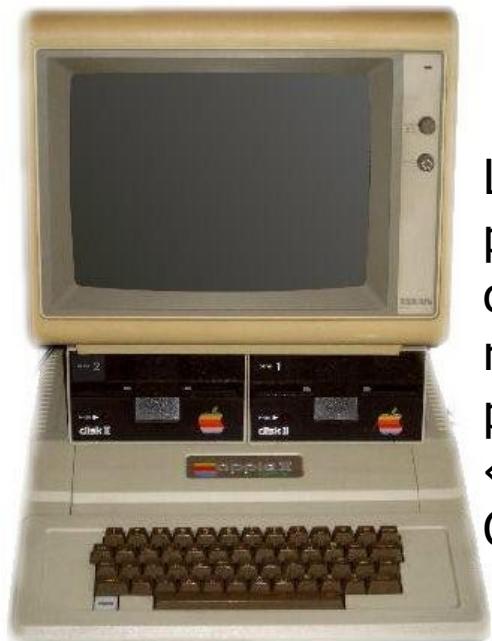
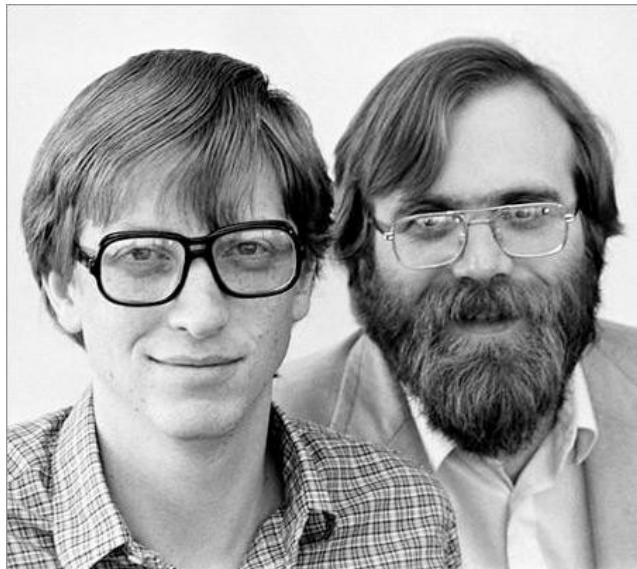


Olivetti «P6040» è un vero e proprio computer, il primo al mondo ad essere dotato di lettore floppy disk integrato, e dal peso e volume ridottissimo.



MICROSOFT E APPLE

Nel 1975 Bill Gates e Paul Allen inventano un nuovo linguaggio di programmazione, BASIC, e decidono di commercializzarlo fondando una nuova azienda, Microsoft.



La Apple è la prima azienda a commercializzare i propri prodotti come «Personal Computer»

Nel 1976, Steve Jobs e Steve Wozniak, convinti che il computer sia prossimo a diventare un oggetto di consumo, fondano la Apple, con lo scopo di renderlo «al pari di un elettrodomestico».



E POI?

Qui si conclude il nostro primo excursus nella storia dei computer.

D'ora in poi sarà necessario conoscere alcune nozioni software più importanti.

Continueremo con la narrazione dopo che avremo introdotto i **sistemi operativi**.

LA PROSSIMA VOLTA...

Inizieremo a spostarci verso il *software*, andando a vedere

1. Come il computer *rappresenta* le informazioni (numeri, testo...)
2. Che cos'è un *programma*
3. Come il computer *esegue* le istruzioni contenute in un programma