Fondamenti di Informatica

CAP. 2

Accademia di belle Arti di Verona

Università degli Studi di Verona

A.A. 2022-2023

Docente - Vincenzo Giannotti

CAPITOLO 2 – ALGORITMI E COMPUTER

Il concetto di Algoritmo

- Nella nostra carrellata storica abbiamo visto che molti hanno tentato per secoli di costruire delle macchine automatiche in grado di realizzare delle operazioni e di produrre dei risultati senza l'intervento dell'uomo: da Pascal e Poleni a Babbage e poi Turing, Zuse fino ai grandi scienziati e imprenditori moderni.
- Fare delle operazioni in maniera automatica significa: realizzare degli **algoritmi** senza che sia dovuto l'intervento dell'uomo.
- Poiché questo è l'unico scopo di una macchina automatica e poiché noi siamo interessati a capire come funziona una tale macchina, sarà utile e necessario, prima di proseguire, cercare di comprendere bene cosa sia un algoritmo.

Il concetto di Algoritmo

Il dizionario Treccani lo definisce più o meno così:

- *“Termine, derivato dall’appellativo al-Khuwārizmī del matematico Muḥammad ibn Mūsa del 9° sec., originario di quello che è l’attuale Uzbekistan, che designa qualunque schema o procedimento sistematico di calcolo. Con un algoritmo si tende a esprimere in termini matematicamente precisi il concetto di procedura generale, di metodo sistematico valido per la soluzione di una certa classe di problemi”.*

Se questa definizione non è abbastanza chiara, proviamo più semplicemente a dirlo così:

Il concetto di Algoritmo

“un algoritmo è una sequenza finita di operazioni da svolgere per risolvere un dato problema”.

- Perché vi ho presentato queste due definizioni, così diverse, di algoritmo?
- Perché ancora oggi non esiste una «definizione di algoritmo» che ne definisca pienamente il concetto in termini formali; ci si deve accontentare di definizioni più o meno intuitive come quella prima evidenziata.
- Per la verità un tentativo di formalizzazione è quello proposto da Turing con la sua «Macchina di Turing» che è ancora oggi lo strumento più potente utilizzato per decidere se un dato procedimento sia o meno calcolabile con una macchina automatica.

Per giustificare questa affermazione ricordiamo che una macchina di Turing è costituita da una serie di regole che specificano come leggere e scrivere su un nastro di memoria, in base ai simboli presenti sul nastro e allo stato corrente della macchina. L'idea fondamentale è che la macchina di Turing possa rappresentare qualsiasi algoritmo calcolabile, ovvero qualsiasi procedura eseguibile da un computer.

In termini formali, il funzionamento di una macchina di Turing è definito da una funzione di transizione che descrive come la macchina di Turing si comporta quando si trova in uno stato specifico e legge un simbolo specifico dal nastro di input. La funzione di transizione prende in ingresso due argomenti:

- lo stato corrente della macchina di Turing
- il simbolo letto dalla posizione corrente del nastro di input

E restituisce una tripla composta da:

- il simbolo da scrivere sul nastro di lavoro,
- la direzione in cui la testina della macchina di Turing deve muoversi (destra o sinistra o nessuno spostamento)
- lo stato successivo della macchina di Turing.

La funzione di transizione viene espressa come una tabella o come un grafo orientato in cui ogni nodo rappresenta uno stato della macchina di Turing e ogni arco rappresenta una transizione con l'etichetta che rappresenta il simbolo letto, il simbolo da scrivere, la direzione in cui spostare la testina e lo stato successivo.

La quintupla $\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, N, R\}$ definisce formalmente una macchina di Turing come modello matematico di un computer dove Q è l'insieme degli stati possibili e Γ l'alfabeto dei simboli del nastro. La funzione di transizione δ è l'elemento chiave che consente alla macchina di Turing di eseguire un algoritmo, specificando come la macchina deve leggere e scrivere simboli sul nastro in base allo stato corrente.

In sostanza, l'idea dietro alla definizione formale di un algoritmo è quella di fornire una descrizione precisa e univoca di una procedura computazionale. Una volta definita la macchina di Turing, è possibile utilizzarla come modello per descrivere qualsiasi algoritmo, o per dimostrare che un problema non è risolvibile da un algoritmo (ad esempio, dimostrando che non esiste una macchina di Turing che lo risolve).

Un problema di questo tipo è, per esempio, il **Problema del commesso viaggiatore**: dato un insieme di città e le distanze tra di esse, determinare il percorso più breve che visita tutte le città una volta sola e ritorna alla città di partenza. Per questo problema non esiste un algoritmo deterministico efficiente che lo risolva per tutte le istanze in quanto il tempo di computazione cresce esponenzialmente al crescere del numero di città. Esistono algoritmi di tipo euristico (non deterministico) che sono in grado di calcolare delle soluzioni non esatte ma ottimali e accettabili in pratica.

Il concetto di Algoritmo

Detto questo, quello di «**Algoritmo»** è forse il più importante concetto dell'Informatica e noi, per comodità, lo considereremo nella sua accezione più semplice che ribadiamo:

"**un algoritmo è una sequenza finita di operazioni da svolgere per risolvere un dato problema**"

- Se qualcuno vi insegna la ricetta della pasta frolla, vi sta fornendo un algoritmo:
  - In una terrina mettete la farina
  - formate una conca in mezzo ad essa e rompete al centro le uova
  - ora metteteci lo zucchero, la scorza di limone grattugiata, il sale e il burro
  - quindi iniziate a lavorare gli ingredienti con le mani
  - ad operazione conclusa formate una palla, avvolgetela con la pellicola per alimenti e adagiatela in frigorifero a riposare per 30 minuti.
- Se l'algoritmo è corretto e voi lo svolgete rigorosamente, certamente la vostra pasta frolla sarà ottima (processi industriali).

Il concetto di Algoritmo

- Anche senza bisogno che qualcuno ce lo spieghi, vediamo subito che per essere sicuri di fare un'ottima pasta frolla abbiamo bisogno che:
    - Ci sia dato il materiale con cui lavorare (gli ingredienti)
    - Ci sia spiegato quale procedimento applicare attraverso un numero finito di istruzioni elementari (la ricetta)
    - Tutto questo, alla fine, deve produrre il risultato volute (la pasta frolla).

- Utilizzando un linguaggio più affine all'informatica, abbiamo bisogno:
    - Di un **input**
    - Di una serie di **passi elementari**
    - Di un **output**

Esempi di Algoritmi

- Classici esempi di algoritmi in informatica sono:
  - L’algoritmo di Euclide per il calcolo del Massimo Comun Divisore (MCD) di due numeri interi
  - L’algoritmo di compressione *Jpeg* di una immagine
  - L'algoritmo di *Dijkstra* utilizzato per cercare i cammini minimi in un grafo (i.e. in un navigatore)
  - L’algoritmo *PageRank di Google*, che stabilisce quale sito far apparire prima nei risultati di una ricerca

- Ma anche i seguenti sono esempi di algoritmi:
  - Le indicazioni stradali per raggiungere una data località
  - Le istruzioni per assemblare un mobile dell’IKEA

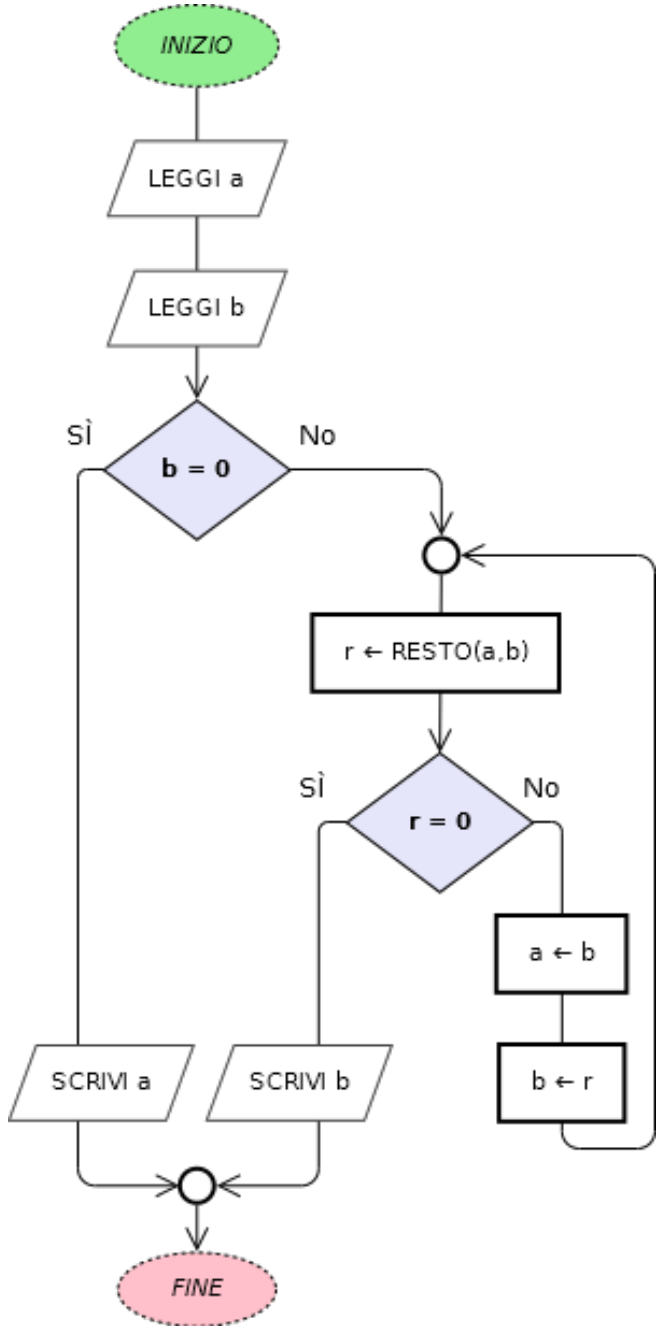
Tipi di Algoritmi

Alcuni algoritmi sono classificati in base alla loro funzione o alla tecnica di realizzazione, ad esempio:

- **Algoritmi di ordinamento**: utilizzati per elencare gli elementi di un insieme secondo una data regola (es. dal più piccolo al più grande)
- **Algoritmi di ricerca**: utilizzati per trovare un elemento avente determinate caratteristiche all'interno di un insieme di altri elementi (es. il più grande)
- **Algoritmi iterativi**: sono delle tipologie di algoritmi, molto utilizzate in informatica, nei quali una sequenza di azioni viene ripetuta finché è necessaria la ripetizione stessa ovvero finché non viene soddisfatta una data condizione (es. dividi per due finché il risultato è minore di una data soglia)
- **Algoritmi ricorsivi**: in informatica un algoritmo viene detto ricorsivo quando la sua esecuzione richiama l'esecuzione dello stesso algoritmo su un nuovo insieme di dati . È una tecnica molto elegante e tuttavia richiede una certa abilità per essere applicata in quanto l'esecuzione può facilmente andare fuori controllo (loop infinito) se non studiata correttamente (es. calcolo della successione di Fibonacci $F_n=F_{n-1}+F_{n-2}$ con $F_0=0$ e $F_1=1$).

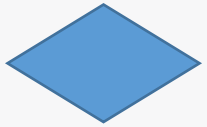
I diagrammi di flusso

- In informatica il **diagramma di flusso** (in inglese flow chart) è un linguaggio di modellazione grafico per rappresentare i diversi step di un processo o il flusso di controllo ed esecuzione di un algoritmo. Nel caso in figura, il calcolo del MCD; esso consente di descrivere in modo schematico attraverso dei simboli grafici:
  - le operazioni da compiere, rappresentate mediante sagome convenzionali (rettangoli, rombi, parallelogrammi, rettangoli smussati...), ciascuna con un preciso significato logico e all'interno delle quali un'indicazione testuale descrive l'attività da svolgere
  - la sequenza nella quale tali operazioni devono essere compiute, rappresentate con frecce di collegamento (flusso).
- Questi diagrammi vengono spesso chiamati col nome del flusso che rappresentano, come: diagramma di flusso dei processi, diagramma di flusso funzionale, diagramma di flusso di una procedura, mappatura dei processi di una organizzazione etc.



Come si costruisce un diagramma di flusso

- Come detto, un diagramma di flusso serve a descrivere un flusso di attività o una sequenza di operazioni. È molto utilizzato per documentare un lavoro, per pianificare un progetto, per comunicare un processo complesso e per sviluppare un algoritmo, per rappresentare il flusso di informazioni all'interno di un'organizzazione e così via.
- Per poterlo costruire correttamente è necessario aver chiaro l'obiettivo per cui lo si vuole utilizzare ed è necessario organizzare le attività da rappresentare secondo un ordine (cronologico, funzionale o altro) che si ricava sviluppando passo passo ciò che si sta analizzando, anche con la collaborazione di chi svolge abitualmente queste attività, nel caso per esempio di un processo lavorativo.
- Per completezza vediamo brevemente alcune delle grafiche principali utilizzate per la costruzione di un flow chart.

Simbolo	Descrizione
Rettangolo 	Rappresenta un'operazione o un'attività. Solitamente contiene il nome dell'operazione o dell'attività che deve essere eseguita. Ad esempio, potrebbe rappresentare un'istruzione di un programma, un'operazione di calcolo o una fase di un processo produttivo.
Rombo 	Rappresenta una decisione o un'alternativa. Viene utilizzato per indicare che il flusso del programma o del processo deve seguire un ramo o un altro in base a una condizione specifica. Ad esempio, potrebbe rappresentare una decisione basata su una condizione, come "se x è maggiore di y, allora esegui l'operazione A, altrimenti esegui l'operazione B".
Ovale 	Rappresenta l'inizio o la fine di un processo. Viene utilizzato per indicare il punto iniziale o finale del programma o del processo. Ad esempio, potrebbe rappresentare l'avvio di un programma o la fine di un processo produttivo.
Freccia	Rappresenta il flusso di controllo tra i vari simboli del diagramma di flusso. Le frecce possono essere di diversi tipi a seconda del tipo di flusso che rappresentano. Le frecce continue rappresentano il flusso diretto, mentre le frecce tratteggiate rappresentano il flusso indiretto.
Parallelo-gramma 	Rappresenta l'input o l'output di un processo o di un programma. Viene utilizzato per indicare l'ingresso o l'uscita dei dati dal processo o dal programma. Ad esempio, potrebbe rappresentare l'input di dati da parte dell'utente o l'output di un risultato da parte di un programma.

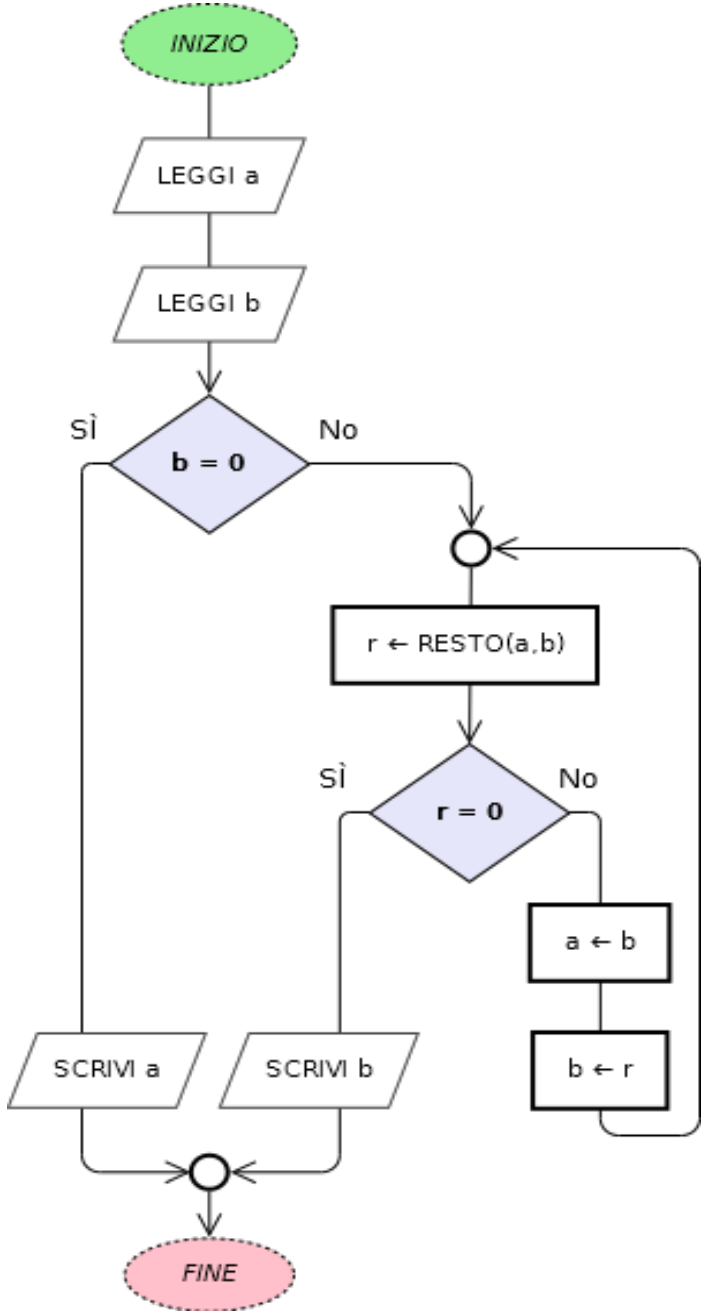
Esempio di costruzione di un Flow Chart

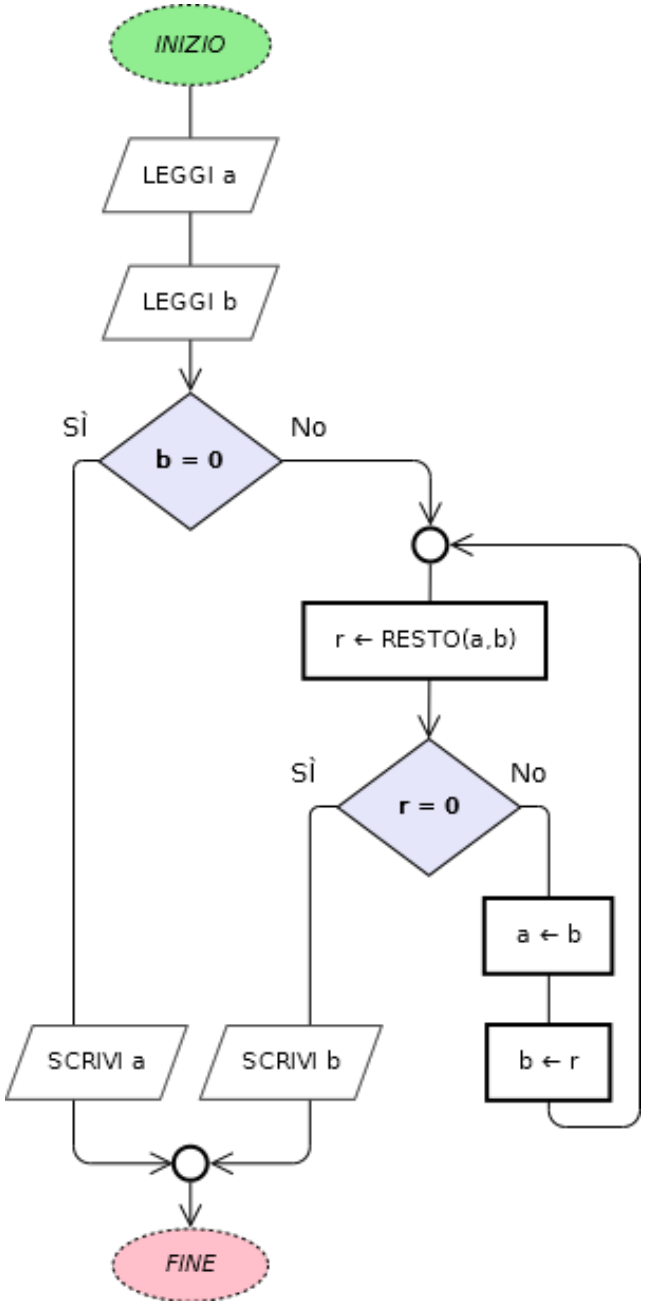
ALGORITMO PER IL CALCOLO DEL MCD

Il Massimo Comun Divisore di due numeri è il divisore più grande, comune ai due numeri considerati.

Dati due numeri naturali *a* e *b*, si controlla se *b* è zero. Se lo è, allora *a* è il MCD. Se non lo è, si divide *a / b* e si assegna ad *r* il resto della divisione. Se *r = 0* allora si può terminare affermando che *b* è il MCD cercato, altrimenti si assegna *a' = b* e *b' = r* ripetendo nuovamente la divisione.

(provare con a=21 e b=15)





Esempio di costruzione di un Flow Chart

Vogliamo calcolare il MCD di a=21 e b=15

- b=0 ? NO
- 21/15=1 r=6
- r=0 ? NO
- a=15; b=6
- Ritorno all'inizio del ciclo
- 15/6=2 r=3
- r=0 ? NO
- a=6; b=3
- Ritorno all'inizio del ciclo
- 6/3=2 r=0
- r=0 ? SI
- Scrivo b
- **Il MCD è 3**

Diagrammi di flusso, a blocchi e mappe concettuali

- Per la loro connotazione topologica i diagrammi di flusso possono essere ricondotti alla classe più ampia dei **diagrammi a blocchi**, che però sono utilizzati principalmente per descrivere in modo strutturato e logico un processo, un sistema o un'architettura, e sono utilizzati soprattutto in ingegneria e informatica. I blocchi rappresentano idee, attività, processi, informazioni o componenti di un sistema, e le relazioni tra di essi sono indicate da linee di connessione. I blocchi sono organizzati in modo gerarchico, cosicché i blocchi superiori rappresentino le informazioni di livello superiore e quelli inferiori le informazioni di livello inferiore.
- I diagrammi a blocchi a loro volta rientrano nell'ancora più vasta categoria delle **mappe concettuali**, utilizzate soprattutto per la descrizione e la rappresentazione delle informazioni e della conoscenza. Le mappe concettuali sono utilizzate per rappresentare concetti e informazioni in modo non strutturato, e sono utilizzate soprattutto nel campo educativo e didattico. Le informazioni sono organizzate in nodi o concetti, che rappresentano parole chiave o concetti importanti, e le relazioni tra di essi sono indicate da frecce che connettono i concetti. I concetti sono organizzati in modo che quelli più importanti siano posizionati al centro della mappa, e quelli meno importanti siano posizionati ai margini.

Proprietà degli Algoritmi

A questo punto possiamo trarre le **conclusioni** di quello che è il concetto di algoritmo che, come abbiamo visto non è strettamente legato all'informatica ma che nell'informatica trova la sua più ampia applicazione. In Informatica, per essere definito tale, un algoritmo deve rispettare alcune proprietà, definite negli anni '60 da Donald Knuth (Stanford University):

Finitezza: Un algoritmo deve terminare dopo un numero finito di passi
Definitezza: Ogni passo deve essere rigorosamente definito; le azioni da svolgere devono essere rigorose e non ambigue (operazioni elementari)
Input: deve avere un numero finito di dati/oggetti in entrata che rappresentano i dati del problema
Output: deve produrre un risultato univoco che rappresenta la soluzione del problema
Efficacia: tutte le operazioni debbono avvenire in un tempo finito ragionevole e l'algoritmo deve essere abbastanza efficiente da poter essere eseguito da una macchina di calcolo con risorse limitate.

L'obiettivo principale di queste proprietà è quello di garantire che gli algoritmi siano corretti, ovvero che risolvano il problema per cui sono stati progettati in modo preciso e affidabile e quindi che un computer sia in grado di portare a termine il suo compito.
Inoltre, queste proprietà aiutano anche a valutare la complessità e l'efficienza degli algoritmi, consentendo di confrontare e scegliere la migliore soluzione possibile per un determinato problema.

Dall’algoritmo al programma

Prima che un computer possa realizzare un dato compito, per esempio tramite un software commissionato da un cliente, è necessario passare attraverso alcune fasi dello sviluppo:

1. **Analisi dei requisiti**: in questa fase si identificano le esigenze degli utenti, i requisiti funzionali e non funzionali del software e gli obiettivi che si vogliono raggiungere con il software. Si definiscono i casi d'uso, i requisiti di sistema e gli scenari di utilizzazione del software.
2. **Progettazione**: in questa fase si definiscono l'architettura del software, il design dettagliato dei componenti del software e le interazioni tra di essi. Si creano i diagrammi di flusso, i diagrammi di sequenza e altri strumenti per rappresentare l'architettura e il design del software.
3. **Programmazione**: in questa fase si traduce il design del software in codice sorgente eseguibile. Si scrive il codice, si integrano i vari componenti e si realizzano le interfacce utente.

Dall’algoritmo al programma

4. **Testing**: in questa fase si verifica che il software funzioni correttamente e soddisfi tutti i requisiti identificati nella fase di analisi dei requisiti. Si effettuano vari tipi di test con diverse tipologie di utenti con lo scopo di verificare il buon funzionamento di tutte le parti*.
5. **Rilascio**: in questa fase il software viene distribuito e reso disponibile agli utenti finali. Si effettua il deployment del software (installazione e configurazione) sui dispositivi degli utenti finali, si installano le versioni finali dei componenti software e si configurano le interfacce utente.
6. **Manutenzione**: in questa fase si effettuano interventi per migliorare, correggere o estendere il software in base alle esigenze dell'utente. Si monitorano le prestazioni del software, si risolvono eventuali problemi e si aggiornano i componenti per mantenerli al passo con le nuove tecnologie o le nuove esigenze degli utenti.

La **fase beta è una fase di testing del software che segue solitamente la fase di sviluppo e precede il rilascio ufficiale del software. Durante la fase beta, gli sviluppatori rilasciano una versione preliminare del software ai beta tester per raccogliere feedback sulle funzionalità e sui problemi del software*

Dall’algoritmo al programma

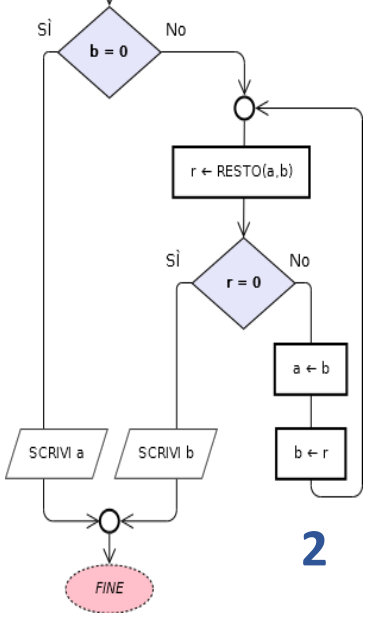
All’interno di queste fasi lo **studio degli algoritmi** interviene soprattutto in quelle di Progettazione e di Implementazione:

- Nella fase di Progettazione, lo studio degli algoritmi aiuta a definire l'algoritmo più adatto per risolvere problemi specifici all'interno del software e ad identificare eventuali limitazioni di prestazioni.
- Nella fase di Implementazione, lo studio degli algoritmi aiuta i programmatori a scrivere codice più efficiente e a migliorare le prestazioni del software.

Ciò non toglie che, in generale, lo studio degli algoritmi possa essere utilizzato in tutte le fasi dello sviluppo del software, poiché gli algoritmi rappresentano la base di molte funzionalità del software e possono avere un impatto significativo sulle prestazioni e sulla qualità del software stesso.

Dall’algoritmo al programma

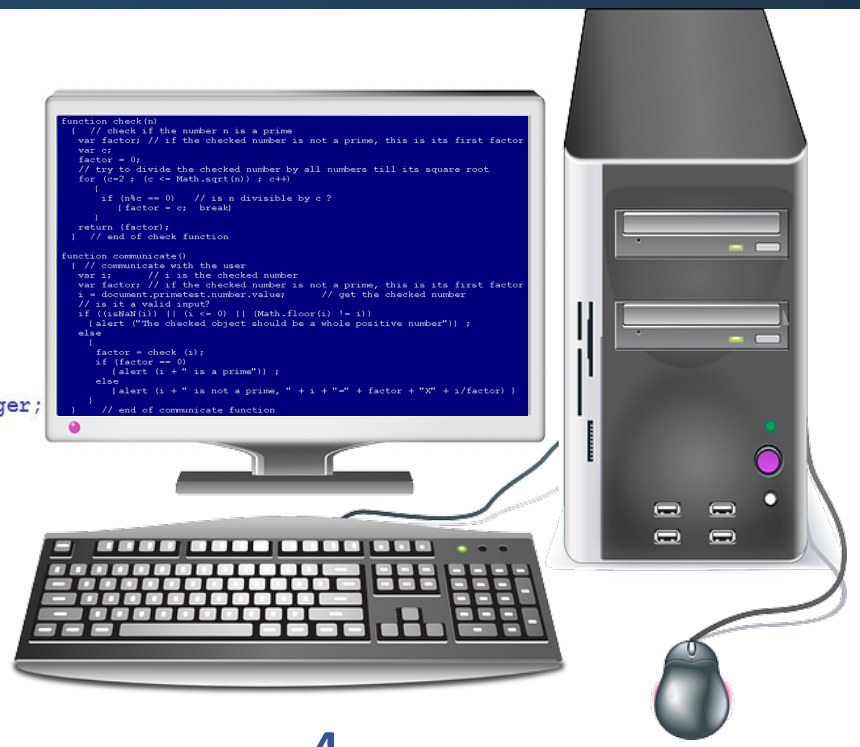
1



2

```
PROGRAM Euclide;
VAR
    M,N:integer;

FUNCTION MCD(M,N:integer):integer;
    VAR
    Resto:integer;

    BEGIN
    Resto:=M MOD N;
         WHILE Resto<>0 DO
         BEGIN
              M:=N;
              N:=Resto;
              Resto:=M MOD N

         END;
         MCD:=N
    END;

BEGIN
writeln('Assegna M e N ');
read(M);read(N);
writeln('Il MCD e'':= ',MCD(M,N));
readln
END.
```

3



4

Dall’algoritmo al programma

- In linea teorica, una volta che viene scoperto un nuovo algoritmo che realizza un dato compito (per esempio l’algoritmo di Euclide per trovare il Massimo Comun Divisore di due numeri interi o gli algoritmi utilizzati per comprimere musica in mp3 o immagini in jpg, l’algoritmo di Dijkstra per il calcolo dei percorsi minimi), questo viene dato per acquisito e non viene richiesto di comprenderne il funzionamento.
- In questo senso l’algoritmo rappresenta un «mattone fondamentale» che non viene ulteriormente messo in discussione.
- Soprattutto nell’informatica moderna, questo concetto è ampiamente utilizzato e un grande numero di questi «algoritmi standard» sono diventati parte integrante della formazione di base di ogni programmatore e sono utilizzati in molte applicazioni e in molti campi dell'informatica. Molti algoritmi (standard o meno) si trovano nel Web ad uso e consumo dei programmatori e degli appassionati di tutto il mondo.

Il Computer

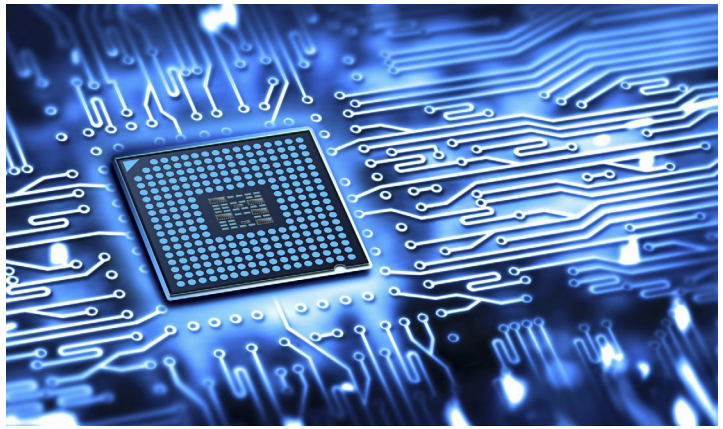
Ripartiamo dalla definizione di Computer:

«Macchina in grado di elaborare dati, automatica e programmabile»

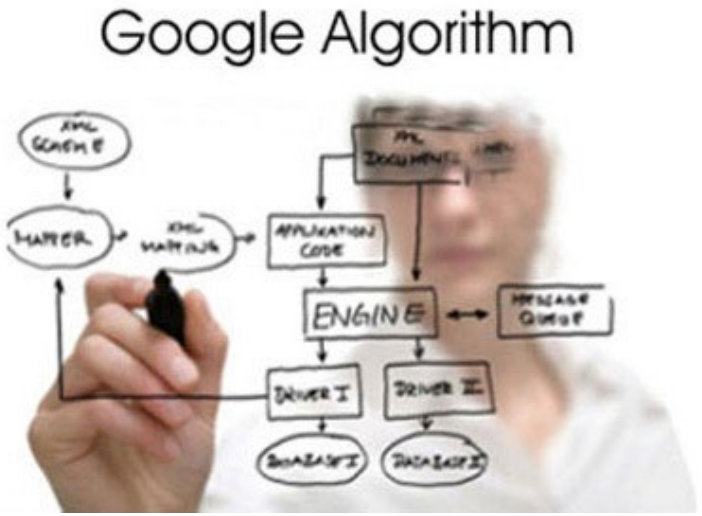
Ricordiamo le sue due componenti fondamentali:

- La componente fisica ovvero «**Hardware**»;
- La componente immateriale ovvero «**Software**»

Hardware



Software


Google Algorithm
ENGINE


Norton
AntiVirus
Windows
Professional
Office xp
Professional
Adobe
Photoshop 6.0
Encarta
Enciclopedia
Plus 2002
2002

Il Computer

La prima cosa da fare per cercare di spiegare il computer e il suo impatto sulla società moderna è quella di porsi la seguente domanda:

Perché il computer è diventato uno strumento così importante e pervasivo della nostra società?

Il Computer

Perché tutta l'informazione del pianeta può essere ricondotta a sequenze di impulsi elettrici.

- Immagini, testi, voce, dati, film, musica, vengono codificati e archiviati in sequenze di 0 e 1 (BIT) 0110011100110…
- Questo vale anche per l'informazione che non abbiamo a disposizione, come quella degli Ambienti Virtuali (Metaverso, rendering etc…)

Il Computer

- A differenza della mente umana che è in grado di elaborare informazioni in modo flessibile e adattivo e di affrontare e risolvere problemi utilizzando capacità cognitive, intuizione e ragionamento e anche informazioni di natura complessa, come le emozioni e i sentimenti, il computer, che è una macchina, è invece un semplice «**esecutore**» di istruzioni impartite dall'esterno.
- Il computer, in quanto mero esecutore di algoritmi, deve essere opportunamente istruito e ciò avviene ad opera di un programmatore capace di comunicare e interagire con la macchina attraverso dei **linguaggi di programmazione** che possono essere ad **alto, medio o basso livello di astrazione**.

I linguaggi di programmazione

- I **linguaggi di alto livello** sono progettati per essere facilmente comprensibili e utilizzabili dai programmatori, senza la necessità di conoscere i dettagli del funzionamento del computer. Questi linguaggi sono caratterizzati da una sintassi e una struttura semantica che rendono il codice più leggibile e più vicino al linguaggio naturale. Alcuni esempi di linguaggi di alto livello includono Python, Java, JavaScript ed i più datati Pascal, ADA, Lisp.
- I **linguaggi di medio livello**, o linguaggi di assemblaggio, sono un po' più vicini al funzionamento del computer rispetto ai linguaggi di alto livello, ma ancora più lontani di quelli di basso livello. Questi linguaggi consentono ai programmatori di scrivere codice che è in grado di interagire più direttamente con l'hardware del computer. Alcuni esempi di linguaggi di medio livello includono C e C++.
- I **linguaggi di basso livello**, o linguaggi macchina, sono i linguaggi di programmazione più vicini all'hardware del computer. Questi linguaggi sono scritti direttamente con un linguaggio elementare o addirittura in binario e sono generalmente difficili da leggere e da scrivere. Tuttavia, essi sono molto efficienti e possono essere utilizzati per creare software altamente ottimizzato. Esempi di linguaggi di basso livello includono il linguaggio Assembly e il codice macchina.

Calcolo del fattoriale di un numero usando una funzione ricorsiva in **linguaggio PYTHON**

```python
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

print(factorial(5)) # Output: 120
```

// Calcolo del fattoriale di un numero usando un ciclo for in **linguaggio C**

```c
#include <stdio.h>

int main() {
    int n = 5;
    int fact = 1;

    for (int i = 1; i <= n; i++) {
        fact *= i;
    }

    printf("%d", fact); // Output: 120
    return 0;
}
```

; Calcolo del fattoriale di un numero usando un ciclo while in **linguaggio ASSEMBLY**

```
section .text
global _start

_start:
    mov ecx, 5    ; Numero di cui calcolare il fattoriale
    mov eax, 1    ; Fattoriale iniziale
    mov ebx, 1    ; Contatore iniziale

loop:
    mul ebx       ; Moltiplica EAX per EBX
    inc ebx       ; Incrementa EBX
    cmp ebx, ecx  ; Confronta EBX con ECX
    jle loop      ; Ripeti finché EBX <= ECX

    ; Stampa il risultato
    mov eax, 1    ; Syscall per scrivere su stdout
    mov ebx, 1    ; fd = 1 (stdout)
    mov ecx, eax  ; Buffer di output
    mov edx, 1    ; Lunghezza del buffer
    int 0x80      ; Chiamata di sistema

    ; Termina il programma
    mov eax, 0    ; Syscall per uscire
    xor ebx, ebx  ; Return code = 0
    int 0x80      ; Chiamata di sistema
```

Il Computer

A questo punto, prima di iniziare a parlare del computer ed a cercare di descriverne il funzionamento, ci vogliamo porre delle domande alle quali proveremo a rispondere nel corso delle prossime lezioni:

- Cosa sono i BIT e cosa è la codifica binaria?
- Cos'è un programma e come lo si rappresenta?
- Come fa a lavorare il computer?
- Che cosa ne governa il funzionamento?
- Come possono i computer dialogare fra di loro?
- Quali rischi corriamo quando utilizziamo un computer?

BIT e BYTE

Abbiamo già visto che i computer hanno bisogno di un proprio linguaggio per poter comunicare tra loro e con gli esseri umani.

Così come gli uomini utilizzano dei simboli elementari nel proprio linguaggio (fonemi per linguaggio verbale; caratteri per linguaggio scritto), anche il computer utilizza delle proprie entità elementari che noi codifichiamo in:

0 e 1

BIT (Binary digIT)

Tutta l'informatica è costruita su questi due simboli.

BIT e BYTE

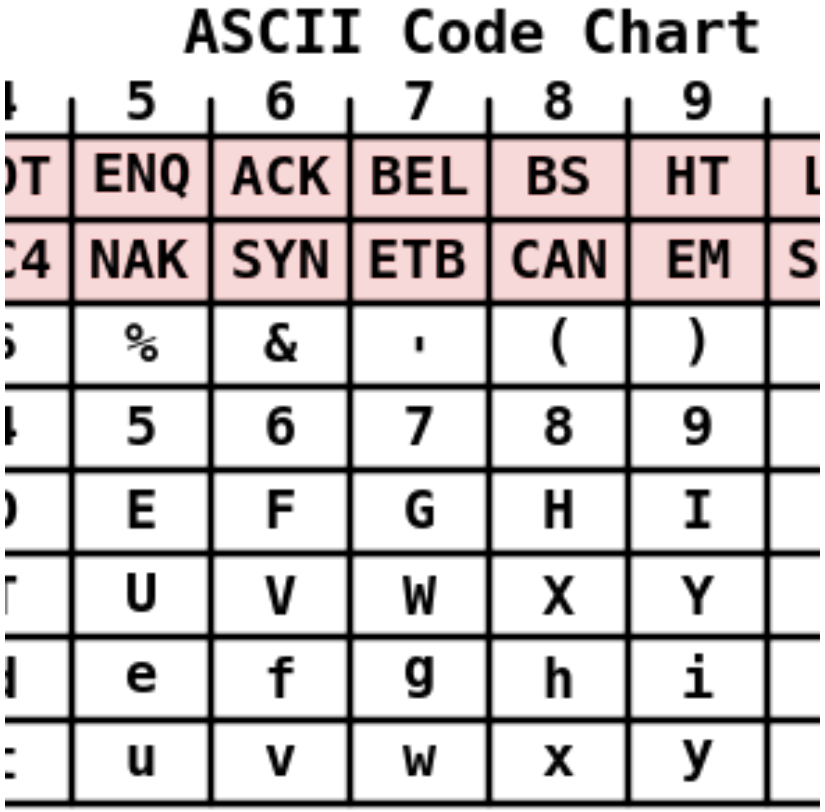
Perché vengono utilizzati solo i due simboli 0 e 1 ?

La ragione è già detta e facilmente intuibile: dal punto di vista tecnologico è molto facile rilevare il passaggio o meno di una corrente all'interno di un conduttore. Sarebbe altresì molto difficile e dispendioso costruire dei circuiti in grado di distinguere tra dieci segnali diversi (0..9).

Quella binaria invece è una «informazione di stato» molto semplice che può assumere soltanto due valori: vero/falso; acceso/spento; si/no. In pratica:

- La corrente c'è => 1
- La corrente non c'è => 0

Attraverso diverse combinazioni di 1 e 0 può essere rappresentato, in un computer, l'intero «**Universo delle Informazioni**».



ASCII Code Chart

4	5	6	7	8	9	
DT	ENQ	ACK	BEL	BS	HT	L
C4	NAK	SYN	ETB	CAN	EM	S
5	%	&	'	(	)	
4	5	6	7	8	9	
D	E	F	G	H	I	
T	U	V	W	X	Y	
d	e	f	g	h	i	
t	u	v	w	x	y	

BIT e BYTE

Il BIT è l'elemento fondamentale di informazione ma normalmente viene aggregato in un insieme che costituisce esso stesso un elemento importante in informatica. Questo è il **BYTE**

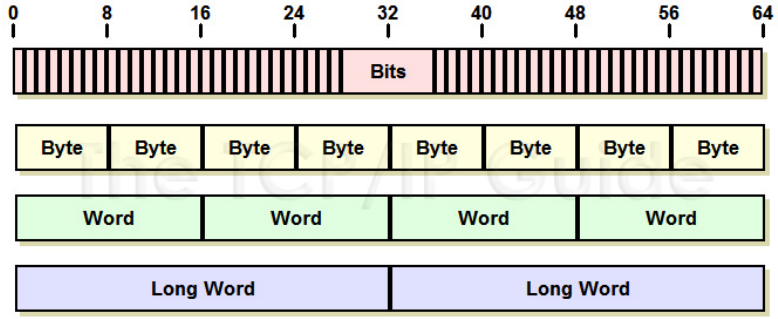
Il BYTE è costituito da una sequenza di 8 BIT. Poiché ad un BYTE può essere associato uno dei nostri caratteri, esso talvolta viene anche detto «carattere».

Il BYTE è anche l'entità elementare utilizzata nella memorizzazione.

BIT e BYTE

Un altro termine importante da conoscere e che spesso viene utilizzato come riferimento è «WORD» che è dato da 2 BYTE e LONGWORD (4 BYTE).

Con WORD si rappresentano per esempio i piccoli interi, con LONGWORD gli interi e i numeri reali, e così via.



BIT e BYTE
i multipli

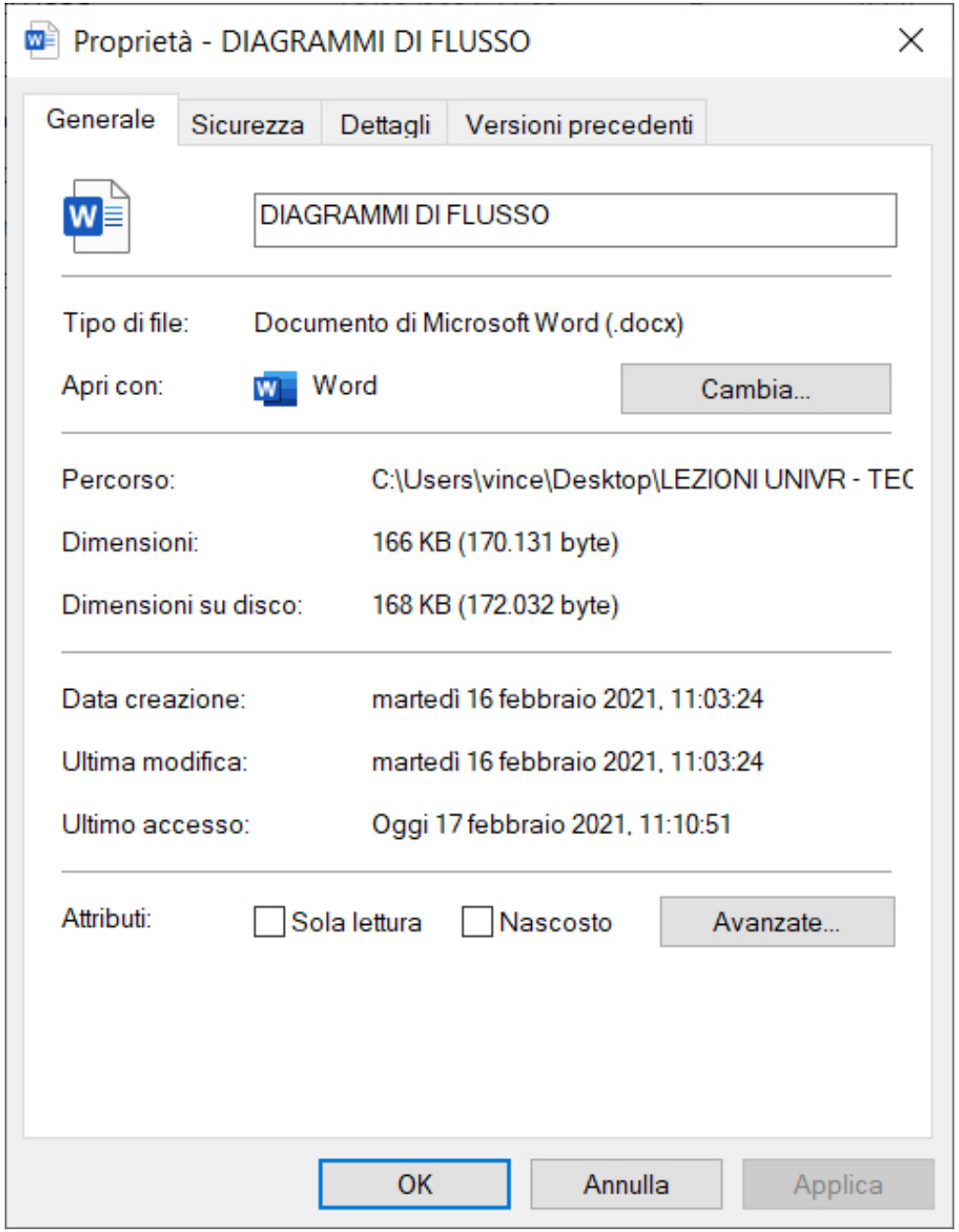
È importante a questo punto fare una breve panoramica su quelli che sono i «multipli» del BIT/BYTE che ci servono per rappresentare, per esempio, la capacità di memoria o la velocità di un dispositivo di archiviazione.

- KILO **k** ≈ un migliaio
  (Kilobit Kb) (Kilobyte kB)
- MEGA **M** ≈ un milione
  (Megabit Mb) (Megabyte MB)
- GIGA **G** ≈ un miliardo
  (Gigabit Gb) (Gigabyte GB)
- TERA **T** ≈ mille miliardi
  (Terabit Tb) (Terabyte TB)

BIT e BYTE
i multipli

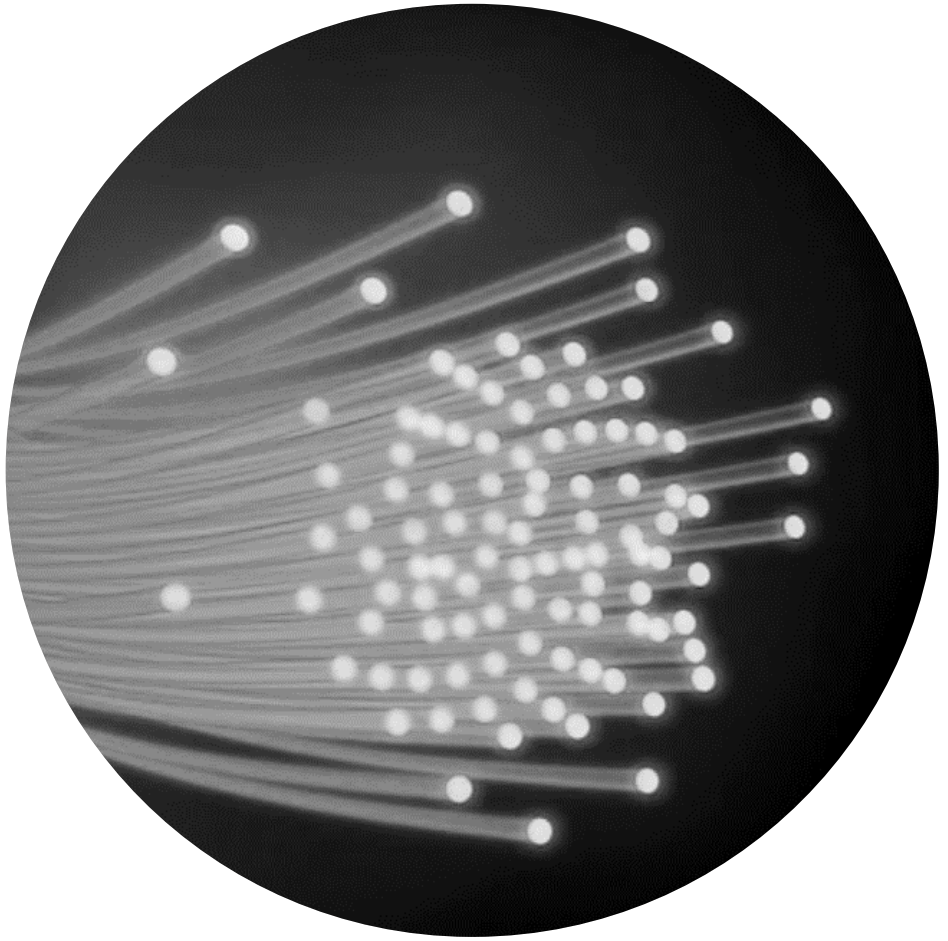
- 1,024 (e.g. one Kilobyte = 1,024 bytes).

- Bit = 1 bit
- Byte = 8 bits
- Kilobyte = 1024 bytes
- Megabyte = 1024 kilobytes
- Gigabyte = 1024 megabytes
- Terabyte = 1024 gigabytes
- Petabyte = 1,048,576 gigabytes
- Exabyte = 1,073,741,824 gigabytes
- Zettabyte = 1,099,511,627,776 gigabytes

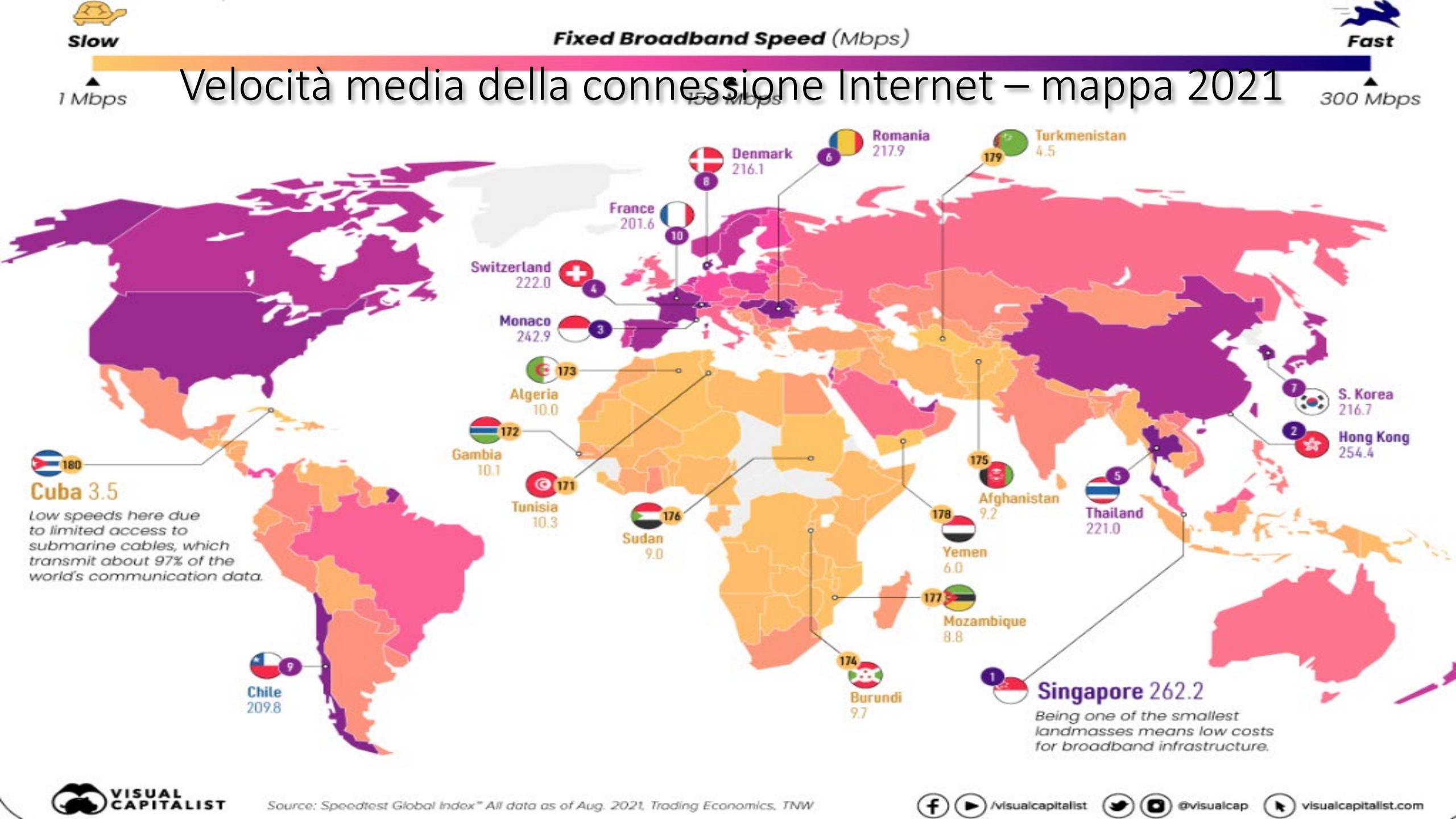


BIT e BYTE
i multipli

- È importante sottolineare che i valori dati (come potenze di 2) valgono quando si parla di supporti di memorizzazione, non nel caso ci si riferisca a questi valori per misurare la velocità di trasmissione.
- Per esempio, quando vi interconnettete a Internet e vi viene segnalato che state scaricando un file alla velocità di 300Kb/s, sappiate che in questo caso 300K vale 300.000 e non 307.200 (300X1024).



Velocità media della connessione Internet – mappa 2021



Global Speedtest 2022

Mobile (sx)-Fixed (dx)

Italia al 47° (mob.) e 63° (fixed)

#		Country	Mbps
1	-	United Arab Emirates	179.61
2	-	Qatar	160.33
3	+1	South Korea	138.46
4	-1	Norway	131.23
5	-	Denmark	123.66
6	+1	Kuwait	119.79
7	+1	China	116.70
8	-2	Netherlands	114.28
9	-	Saudi Arabia	101.88
10	+11	Bulgaria	97.60
11	-1	Bahrain	96.53
12	+3	Macau (SAR)	92.41

#		Country	Mbps
1	-	Singapore	237.15
2	+1	China	226.77
3	+4	Monaco	226.03
4	-2	Chile	224.44
5	-1	United Arab Emirates	219.47
6	-	Thailand	203.28
7	+2	Denmark	200.62
8	-3	Hong Kong (SAR)	198.53
9	-1	United States	198.17
10	-	Spain	180.30
11	-	Romania	175.08
12	+1	France	163.84

powered by Speedtest intelligence

Architettura del Computer

Vediamo ora quali sono i «blocchi» fondamentali che costituiscono il computer, sia esso un «Personal Computer» un «Main Frame» oppure un «Super Computer».

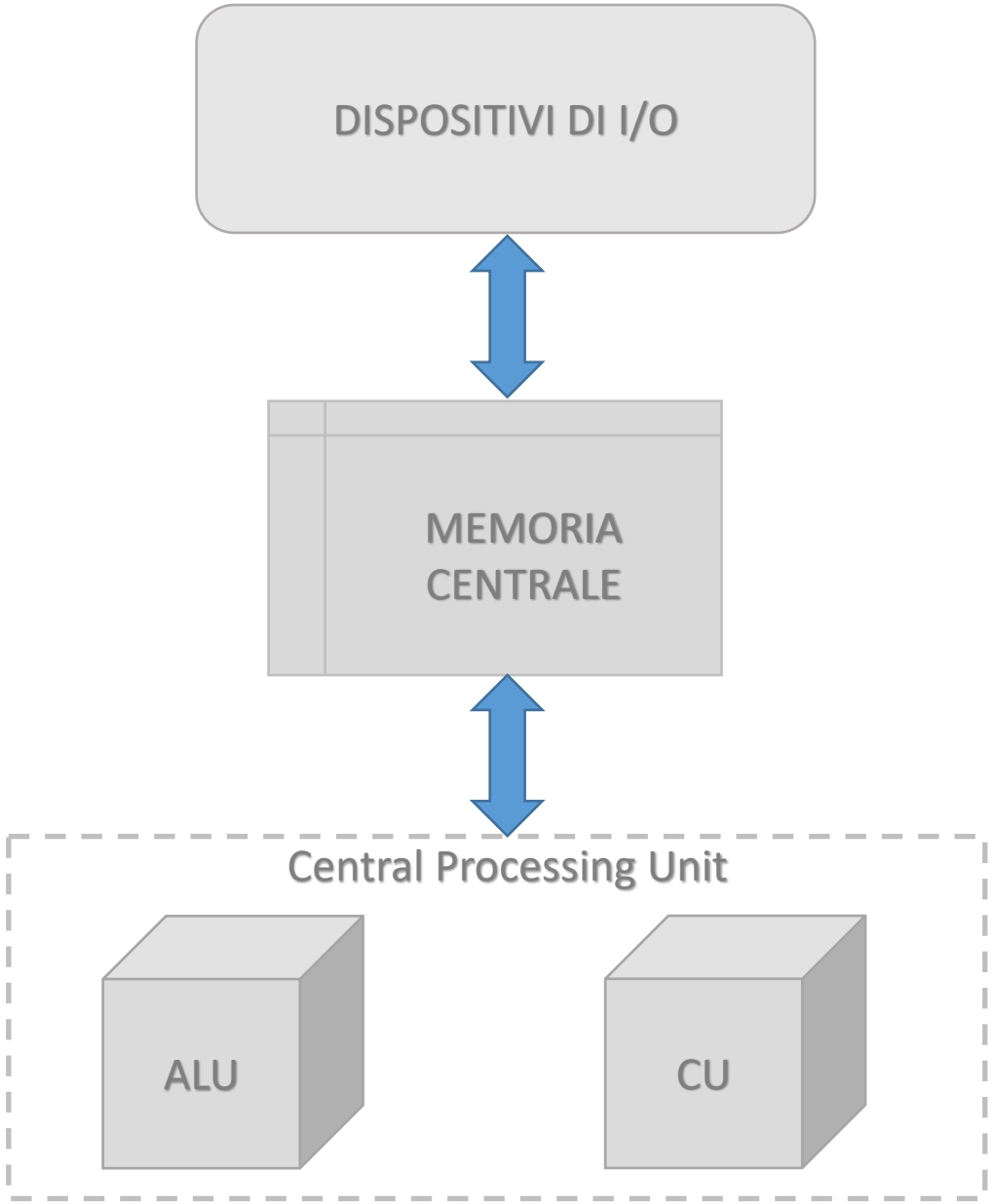
Ogni computer del mondo, indipendentemente dalle sue prestazioni o dal suo costo, può essere ricondotto ad un modello unitario: uno schema di «blocchi funzionali» che lo descrive in termini generici.

ARCHITETTURA DI VON NEWMANN

Architettura del Computer

Il modello di von Neumann è di notevole importanza in quanto descrive l'architettura hardware su cui è basata la maggior parte dei moderni computer programmabili. Fu sviluppato negli anni ‘40 dal matematico ungherese John von Neumann nell’ambito del progetto EDVAC.

In questa architettura i dati e i programmi risiedono entrambi nella memoria centrale.



Architettura del Computer – I/O

Da una parte abbiamo l'uomo, dall'altra abbiamo il computer. Due linguaggi completamente diversi. Serve qualcosa che consenta di trasferire l'informazione da una parte all'altra. Questa prima componente è una

«unità di ingresso al mondo dei BIT»

È un mediatore linguistico capace di acquisire informazioni dall'Utente ovvero dal mondo esterno e di trasferirle al Computer.

Architettura del Computer – I/O

In modo del tutto simmetrico abbiamo necessità di trasferire informazione dal mondo del Computer al mondo Esterno: al mondo degli Esseri Umani. Dati che sono codificati in forma di BIT all'interno della memoria del computer, debbono essere trasferiti a noi in una forma che sia per noi comprensibile.

«unità di uscita dal mondo dei BIT»

Architettura del Computer – I/O

Architettura del Computer – CPU

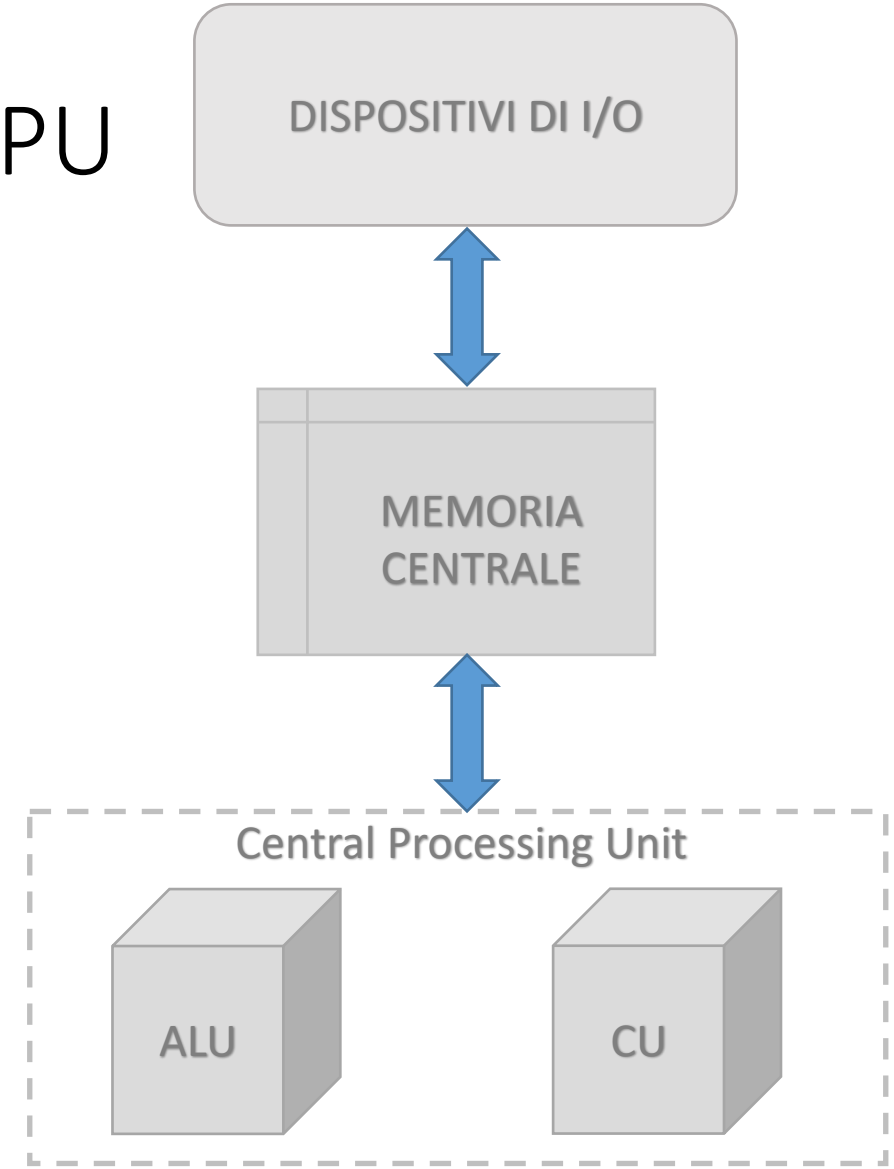
La **Central Processing Unit** (CPU) ovvero Unità di Elaborazione Centrale in italiano, o meglio ancora, **Microprocessore**, risiede all'interno del computer. È il suo cuore e ha la funzione di:

«elaborare i dati ricevuti in ingresso, secondo gli ordini impartiti dal programma»

Ma questo può essere fatto solo con l'ausilio di un'altra unità, ossia la

«Memoria Centrale»

nella quale risiedono sia i dati da elaborare, sia i programmi da eseguire.



Architettura del Computer - Memoria

- La memoria centrale del computer è comunemente nota come RAM (Random Access Memory) ed è uno dei principali componenti hardware di un computer.
- La RAM funziona come un'area di lavoro temporanea per il processore del computer. Quando si avvia un programma, questo viene caricato nella RAM da un supporto di archiviazione di massa. Il processore quindi accede ai dati nella RAM per eseguire le istruzioni del programma. In generale, maggiore è la quantità di RAM presente nel computer, maggiore sarà la quantità di dati che il processore può accedere e gestire simultaneamente, migliorando le prestazioni complessive del sistema.
- La memoria RAM è organizzata in piccole celle di memoria che possono essere lette o scritte in modo indipendente. Questo rende la RAM molto veloce e adatta per l'elaborazione dei dati in tempo reale.
- La capacità della memoria RAM oggigiorno è dell’ordine dei gigabyte (GB) e la velocità di trasferimento dei dati viene misurata in megabyte al secondo (MB/s). La velocità della RAM è influenzata da vari fattori, come la frequenza di clock e il tipo di memoria.

Architettura del Computer - Memoria

La memoria che interviene nella comunicazione con la CPU è sempre e solo la Memoria Centrale con cui la CPU può dialogare direttamente e velocemente.

In prima approssimazione possiamo tuttavia distinguere due tipi fondamentali di memoria:

- La Memoria centrale **RAM** (Random Access Memory)
- La **Memoria di Massa** (Hard Disk, SSD…)

La prima è una memoria «volatile» che ha la funzione di supportare le attività di elaborazione (alta velocità, bassa capacità) della CPU;

La seconda è una memoria «permanente» e serve a memorizzare i dati e i programmi che debbono poter essere utilizzati anche in più sessioni di utilizzazione del computer e fa parte dei dispositivi di I/O.

Prossimo Capitolo – COME FUNZIONA IL COMPUTER

- Nel prossimo capitolo vedremo con maggior dettaglio come è fatto un computer sia dal punto di vista dell'Hardware, sia dal punto di vista del Software:
  - Come funziona la CPU (microprocessore)
  - Cosa è un Sistema Operativo
  - Cosa è un Programma Applicativo