



**Fondamenti di Informatica**  
**Ingegneria Clinica**  
**Lezione 25/11/2010**



**Prof. Raffaele Nicolussi**  
**FUB - Fondazione Ugo Bordoni**  
**Via del Policlinico, 147 - 00161 Roma**

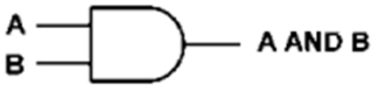
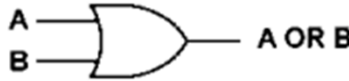
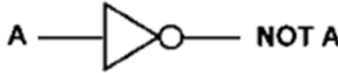


<b>Docente</b>	<b>Raffaele Nicolussi</b>	<i><a href="mailto:rnicolussi@fub.it">rnicolussi@fub.it</a></i> <i>0654803323</i>
<b>Lezioni</b> Aula 54 (ex aula 4) Via del Castro Laurenziano, 7	<b>Lunedì, Giovedì, Venerdì</b>	<b>12:00 – 13:30</b>
<b>Esercitazioni</b> Aula 15 Via Tiburtina, 205	<b>Lunedì</b>	<b>14:00 – 17:30</b>
<b>Ricevimento:</b>	<b>Per appuntamento</b>	<b>in FUB, per email, per telefono</b>
<b>Sito web:</b>	<b><a href="http://w3.uniroma1.it/IngClinFondinf">http://w3.uniroma1.it/IngClinFondinf</a></b>	

# Operatori logici

- ❑ **NOT**: restituisce 1 quando l'input vale 0 e viceversa
  - unico operatore unario (unario: lavora su un singolo operando)
- ❑ **AND**: restituisce 1 solo se tutti gli ingressi valgono 1
- ❑ **OR**: restituisce 1 quando esiste almeno un ingresso che vale 1

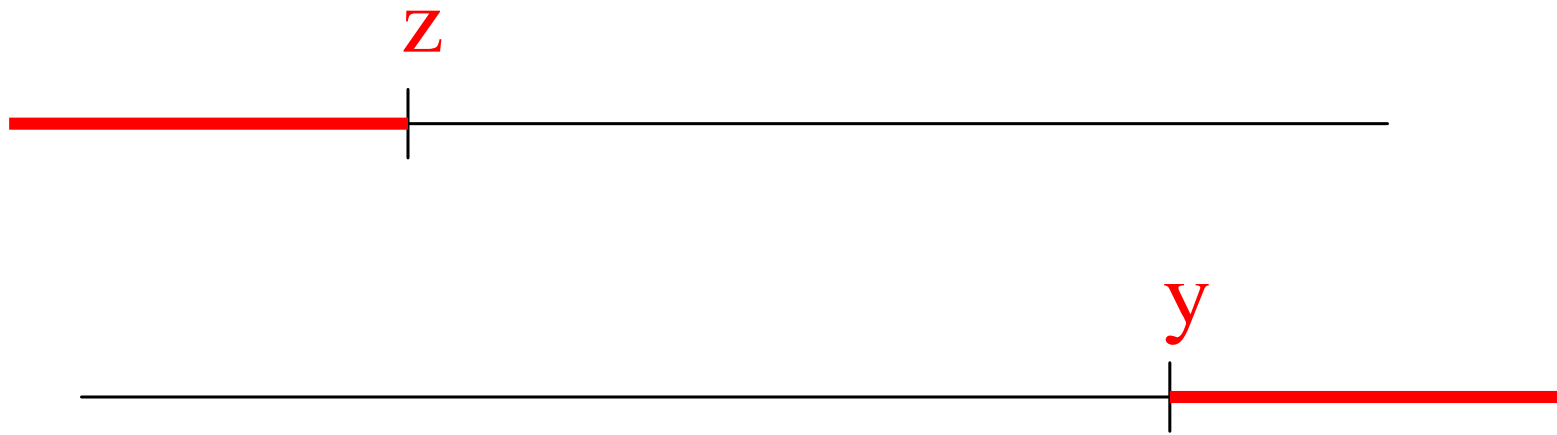
- ❑ **Priorità: NOT-> AND --> OR**

	<table border="1"><thead><tr><th>A</th><th>B</th><th>A AND B</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></tbody></table>	A	B	A AND B	0	0	0	0	1	0	1	0	0	1	1	1
A	B	A AND B														
0	0	0														
0	1	0														
1	0	0														
1	1	1														
	<table border="1"><thead><tr><th>A</th><th>B</th><th>A OR B</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></tbody></table>	A	B	A OR B	0	0	0	0	1	1	1	0	1	1	1	1
A	B	A OR B														
0	0	0														
0	1	1														
1	0	1														
1	1	1														
	<table border="1"><thead><tr><th>A</th><th>NOT A</th></tr></thead><tbody><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></tbody></table>	A	NOT A	0	1	1	0									
A	NOT A															
0	1															
1	0															

Operatore		Risultato		
AND	<b>&amp;&amp;</b>	<b><i>P</i></b>	<b><i>Q</i></b>	<b><i>P&amp;&amp;Q</i></b>
		<i>0</i>	<i>0</i>	<i>0</i>
		<i>0</i>	<i>1</i>	<i>0</i>
		<i>1</i>	<i>0</i>	<i>0</i>
		<i>1</i>	<i>1</i>	<i>1</i>
OR	<b>  </b>	<b><i>P</i></b>	<b><i>Q</i></b>	<b><i>P    Q</i></b>
		<i>0</i>	<i>0</i>	<i>0</i>
		<i>0</i>	<i>1</i>	<i>1</i>
		<i>1</i>	<i>0</i>	<i>1</i>
		<i>1</i>	<i>1</i>	<i>1</i>
NOT	<b>!</b>	<b><i>P</i></b>	<b><i>!</i></b>	
		<i>0</i>	<i>1</i>	
		<i>1</i>	<i>0</i>	



Condizione	Espressione logica
x e y maggiori di z	$(x > z) \ \&\& \ (y > z)$
x uguale a 1.0 o a 3.0	$(x == 1.0) \    \ (x == 3.0)$
x è compreso tra z e y, compresi gli estremi	$(z \leq x) \ \&\& \ (x \leq y)$
x è fuori dal range da z a y	$! (z \leq x \ \&\& \ x \leq y)$ $z > x \    \ x > y$



## Priorità

Operatore	Associatività
()	da sinistra a destra
!	da destra a sinistra
* / %	da sinistra a destra
- +	da sinistra a destra
< <= > >=	da sinistra a destra
== !=	da sinistra a destra
&&	da sinistra a destra
	da sinistra a destra
=	da destra a sinistra

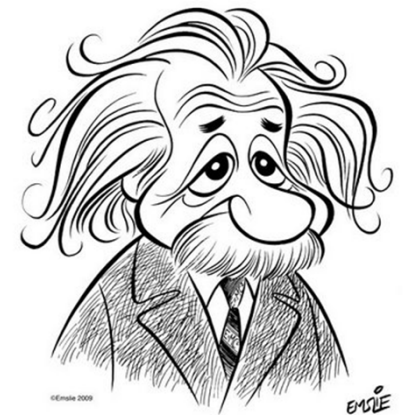
# Quiz



Carta e  
penna



cervello



Amico  
geniale



Con i seguenti valori

`int j=0, m=1, n= -1;`

`float x=2.5, y=0.0;`

Riempire la seguente tabella

<b><math>j &lt; m \ \&amp;\&amp; \ n &lt; m</math></b>	
<b><math>m + n \ \parallel \ !j</math></b>	
<b><math>x * 5 \ \&amp;\&amp; \ 5 \ \parallel \ m/n</math></b>	
<b><math>!x \ \parallel \ !n \ \parallel \ m+n</math></b>	
<b><math>(j \ \parallel \ m) + (x \ \parallel \ n+1)</math></b>	



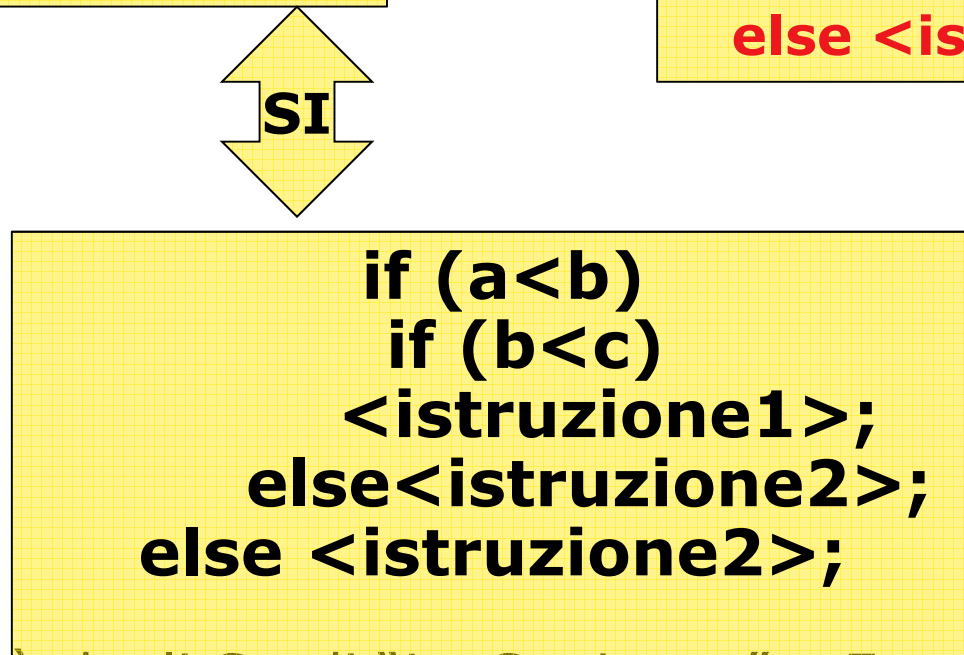
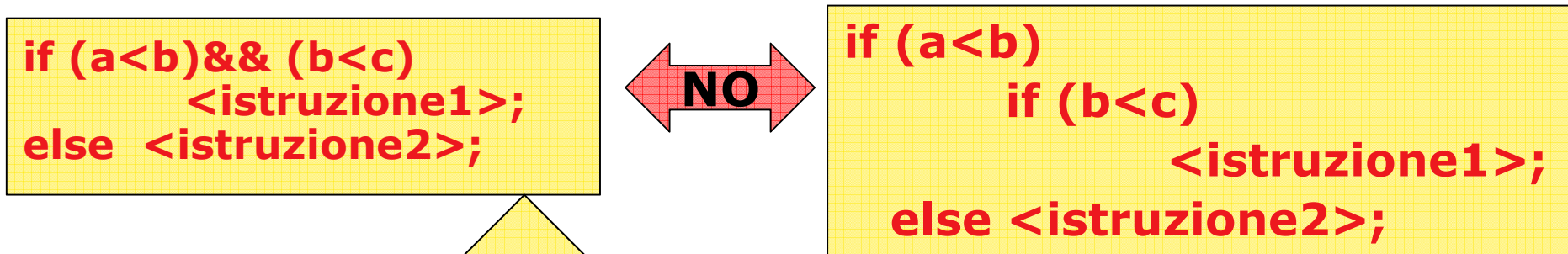
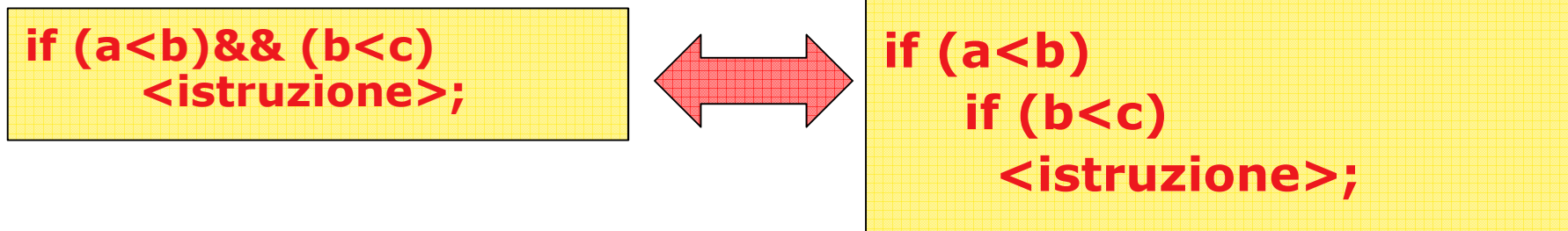


`int j=0, m=1, n= -1;`

`float x=2.5, y=0.0;`

<b><code>j&lt;m &amp;&amp; n&lt;m</code></b>	<b>1</b>
<code>(j&lt;m) &amp;&amp; (n&lt;m)</code>	
<b><code>m +n    !j</code></b>	<b>1</b>
<code>(m +n)    (!j)</code>	
<b><code>x*5 &amp;&amp; 5    m/n</code></b>	<b>1</b>
<code>((x*5) &amp;&amp; 5)    (m/n)</code>	
<b><code>!x    !n    m+n</code></b>	<b>0</b>
<code>((!x)    (!n))    (m+n)</code>	
<b><code>(j    m) + (x    n+1)</code></b>	<b>2</b>
<code>(j    m) + (x    (n+1))</code>	

# If equivalenti





```
if (a<b)&& (b<c)
    <istruzione>;
```

==

```
if (a<b)
    if (b<c)
        <istruzione>;
```

```
if (a<b)&& (b<c)
    <istruzione1>;
else <istruzione2>;
```

~~==~~

```
if (a<b)
    if (b<c)
        <istruzione1>;
else <istruzione2>;
```

==

```
if (a<b)
    if (b<c)
        <istruzione1>;
    else <istruzione2>;
else <istruzione2>;
```



**Problema:** Inserire una sequenza di 12 voti (tra 0 e 30) riportati da uno studente. Stampare il massimo e il minimo dei voti riportati (senza memorizzarli)

**Analisi dei dati:** tre variabili di tipo **int**

- minimo (*min*)
- massimo (*max*)
- voto corrente (*voto*)


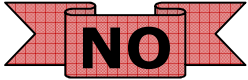

**Algoritmo:**

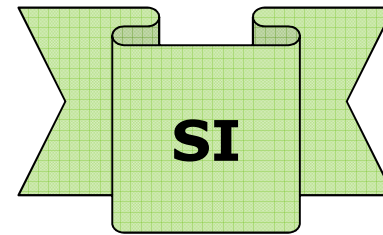
- inizializzo **min** e **max**
- per ogni voto letto*
  - se **min** è maggiore di **voto**, allora  $\text{min} = \text{voto}$
  - se **max** è minore di **voto**, allora  $\text{max} = \text{voto}$

**Problema:**

Come inizializzo min e max?

## Come inizializzo min e max ?

- ❑  $\text{min}=0, \text{max}=0???$  
- ❑  $\text{min}=0, \text{max}=30 ??$  
- ❑  $\text{min}=30, \text{max}=0 ??$  
- ❑  $\text{min}=\text{max}=\text{primo voto letto}??$



- ❑ Con  $\text{min} = 30$  e  $\text{max} = 0$  realizzo un algoritmo che va bene solo per i voti universitari
- ❑ Con  $\text{min} = \text{max} = \text{primo voto letto}$  risparmiamo due confronti e un'assegnazione



# Quiz :: provate ad implementare la soluzione





```
#include<stdio.h>
int main (void)
{
    int min, max, voto, i;

    printf("inserisci il primo numero voto\n");
    scanf("%d", &max);

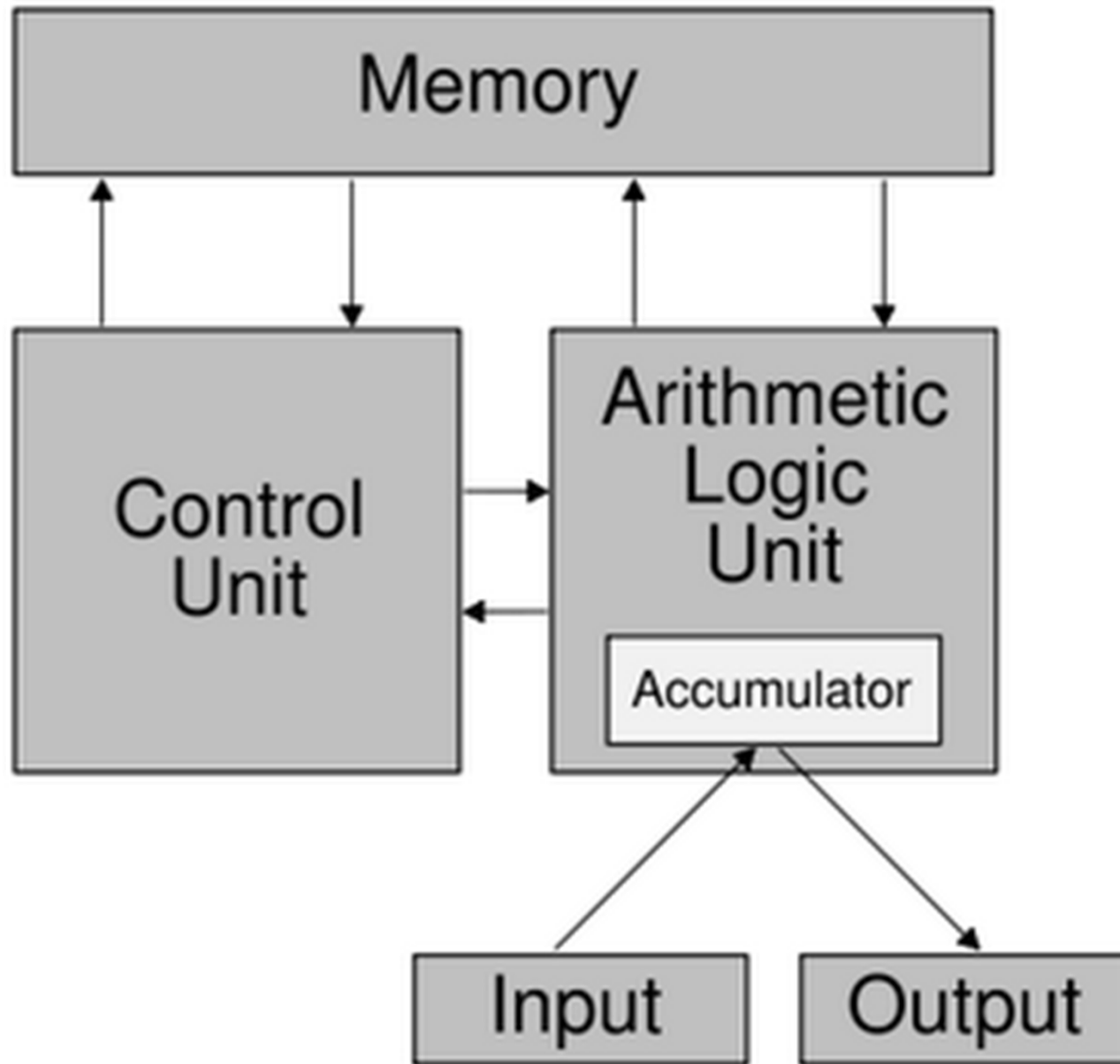
    min=max;

    for (i=0; i<=10; i=i+1)
    {
        printf("inserisci un altro voto\n");
        scanf("%d", &voto);
        if (min>voto)
            min=voto;
        if (max<voto)
            max= voto;
    }

    printf("Voto minimo: %d\n", min);
    printf("Voto max: %d\n", max);

    return 0;
}
```

# Struttura del Calcolatore







# Struttura di un calcolatore

## INDICE

- Architettura di **Von Neumann**
  - Unità I/O
  - Memoria
  - CPU
- Linguaggio assembler
- Struttura della memoria
- Dispense di riferimento (sul sito)
  - Architettura del calcolatore e rappresentazione dell'informazione (Daniela D'Aloisi)

# Il calcolatore elettronico



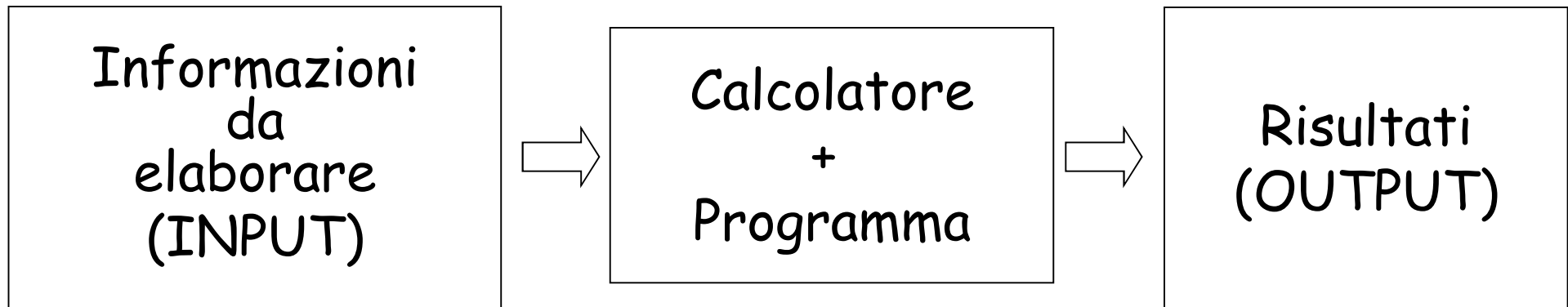
- Un calcolatore elettronico è un *sistema elettronico digitale programmabile*
  - **Sistema**: composto da parti interagenti
  - **Elettronico**: le funzioni fondamentali sono realizzate mediante componenti elettronici
  - **Digitale**: lavora su grandezze discrete
  - **Programmabile**: le operazioni svolte dal sistema sono regolate da un programma (che può essere cambiato)

# Elaborazione al calcolatore



Un calcolatore elettronico:

- Acquisisce i dati di input
- Elabora i dati di input in base ad un programma
- Restituisce i dati di output



# Hardware e Software



- **Hardware:** componenti fisiche del computer (processore, memoria, monitor, dischi, ecc.)
- **Software:** vari tipi di informazione presenti nel computer (programmi, dati da elaborare, dati memorizzati)

# Rappresentazione dell'informazione



- Per costruzione, il funzionamento *logico* di un calcolatore elettronico è regolato da grandezze **binarie** (possono assumere solo due stati significativi)
  
- Un **bit** (**binary digit**), i cui valori possibili sono **0** e **1** (per convenzione), può rappresentare due valori distinti di una qualsiasi grandezza
  
- Un insieme di **N** bit può rappresentare  **$2^N$**  valori distinti di una qualsiasi grandezza
  
- La rappresentazione di un'informazione mediante insiemi di bit si basa sulle seguenti tecniche:
  - **discretizzazione** di un insieme di valori **continui**
  - **riduzione** della dimensione di un insieme di valori **discreti**
  - **codifica** di un insieme **finito** di valori



## Bit, byte & co.

- In informatica ed in teoria dell'informazione, la parola **bit** ha due significati molto diversi, a seconda del contesto in cui rispettivamente la si usa:
  - un bit è l'unità di misura dell'informazione (dall'inglese "**binary unit**"), definita come la quantità minima di informazione che serve a discernere tra due possibili alternative equiprobabili.
  - un bit è una cifra binaria, (in inglese "**binary digit**") ovvero uno dei due simboli del sistema numerico binario, classicamente chiamati *zero* (0) e *uno* (1);
- <http://it.wikipedia.org/wiki/Bit>



# Bit, byte & co.

- Con 1 bit posso rappresentare **solo 2 situazioni**

0 spento	1 acceso
0 falso	1 vero

- Con 2 bit posso rappresentare **4 situazioni**

00 fermo	01 lento	10 veloce	11 massi ma
00 piano terra	01 primo piano	10 second o piano	11 tetto

- Con 3 bit posso rappresentare **8 situazioni**

000 N	001 N-E	010 E	011 S-E	100 S	101 S-O	110 O	111 N-O
----------	------------	----------	------------	----------	------------	----------	------------

# Informazioni rappresentabili



- E' necessario assumere che le grandezze da rappresentare formino un insieme **finito** di valori (il numero di bit a disposizione è finito)
  - **Discretizzazione**: un insieme di valori *continui* (p. es., i numeri reali nell'intervallo  $[-3, 2]$ ) viene trasformato in un insieme di valori discreti tramite la suddivisione in intervalli (rappresentazione a **precisione finita**)
  - **Riduzione**: un insieme di valori *discreti* (p. es., i numeri interi) viene **ridotto** ad un suo sottoinsieme finito
  - **Codifica**: ad ogni elemento di un insieme finito di valori viene univocamente associata una configurazione dell'insieme di bit usato



# Esempi di rappresentazioni



- Caratteri
  - Codice ASCII (128 caratteri distinti)
  
- Numeri Interi
  - 1 bit *di segno* e 15 bit *di modulo* (coperto intervallo  $[-32767, 32767]$ )
  
- Immagini
  - gif, jpeg, tiff, bmp
  
- Audio
  - mp3
  
- Video
  - avi

# Il calcolatore elettronico: sottosistemi funzionali



- La suddivisione funzionale è basata su un'architettura detta di Von Neumann (1947)
  
- Sono individuabili i seguenti sottosistemi
  - **Memoria principale** (primaria, centrale)
  - **Unità centrale di elaborazione** (CPU)
  - **Dispositivi di ingresso/uscita** (I/O)
  - **Canali di comunicazione** (bus)

# Struttura calcolatore di von Neumann

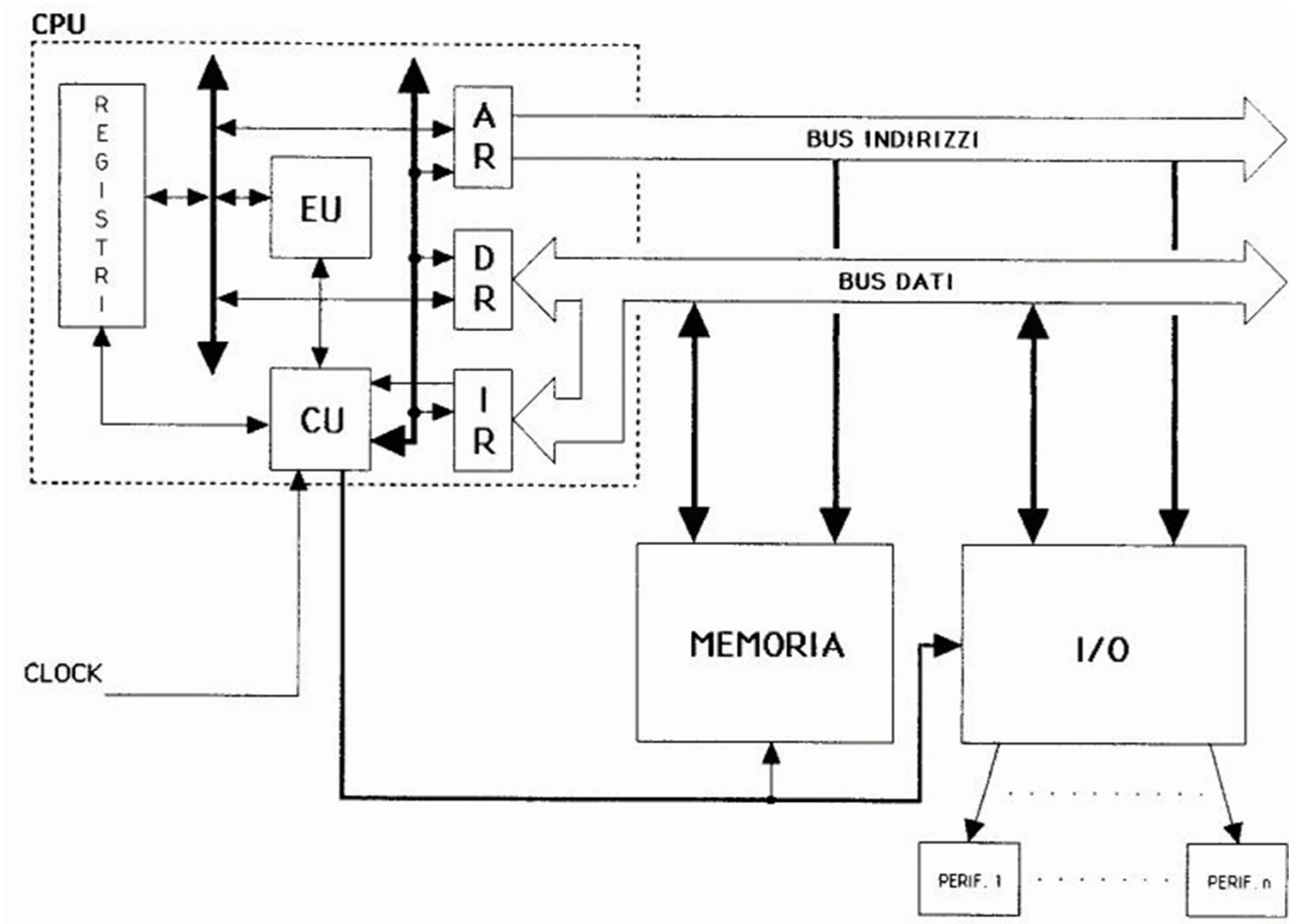


fig. 1.1 - Sistema basato sul modello Von Neumann



# John von Neumann

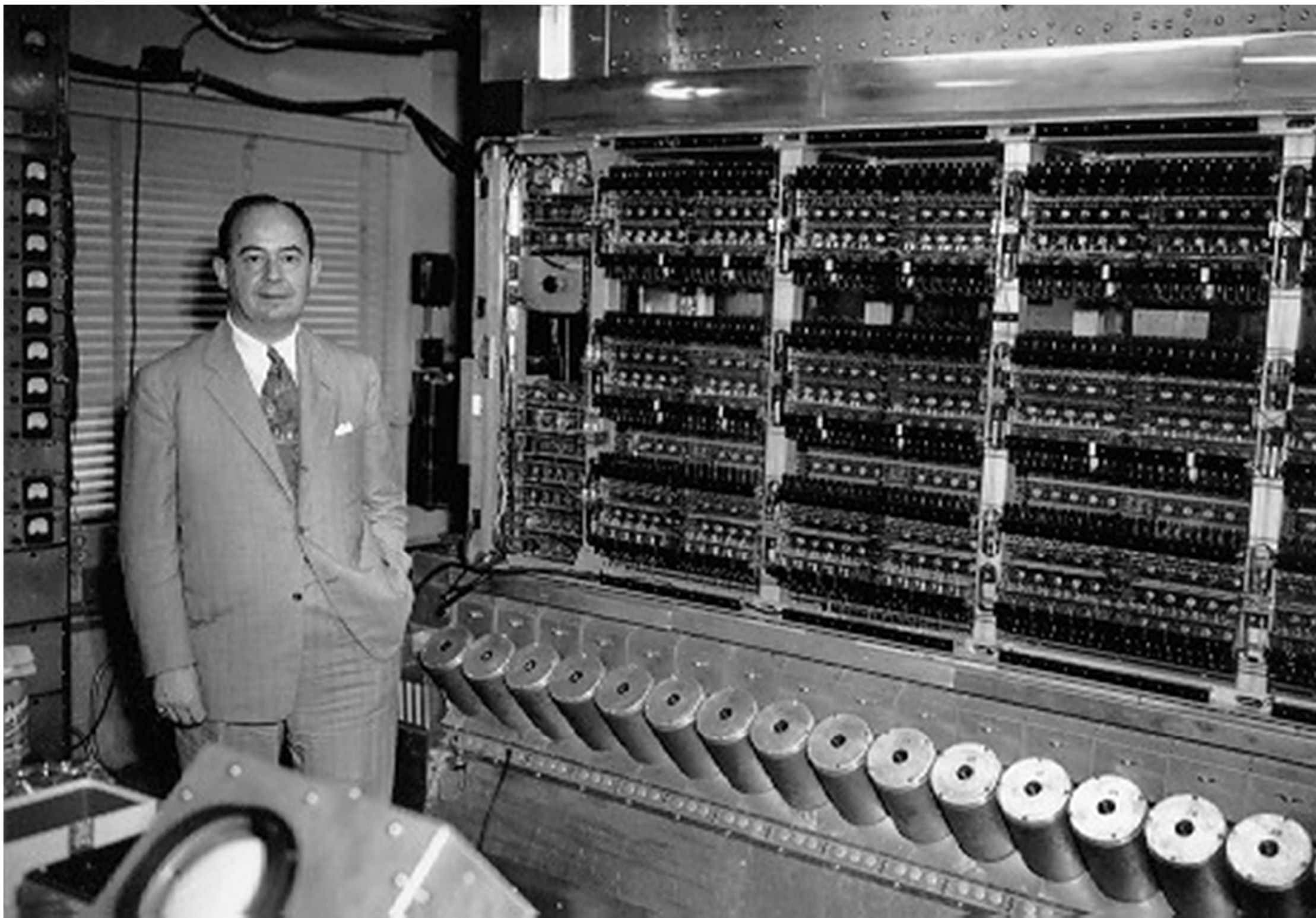


## John von Neumann

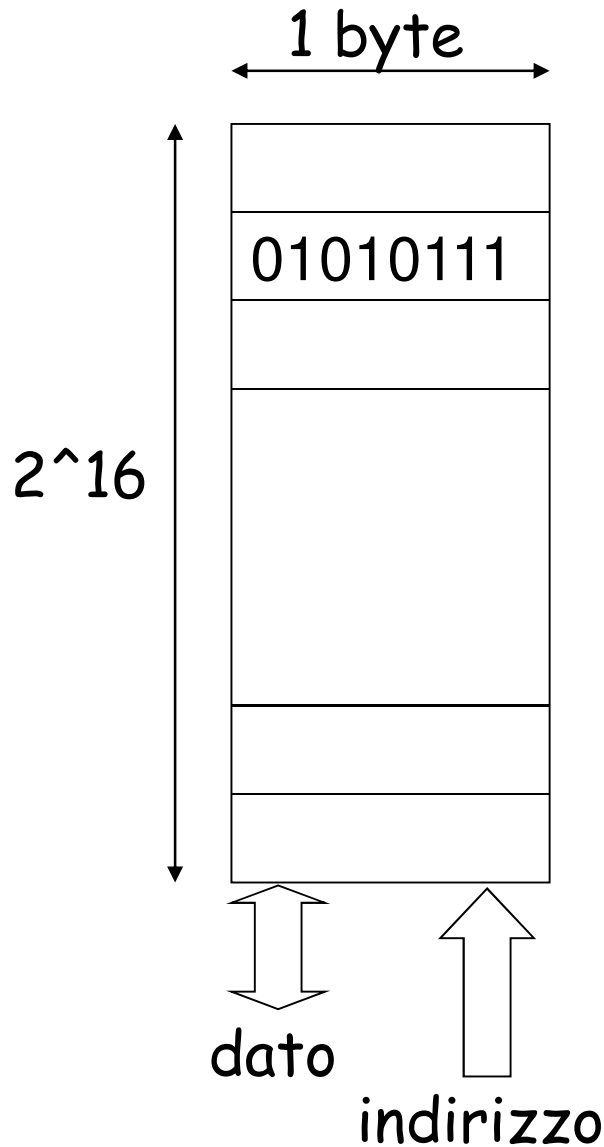
(Budapest, 28 .12.1903 – Washington, 8.02.1957)

è stato un matematico e informatico statunitense di origine **ungherese**.

Fu una delle personalità scientifiche preminenti del XX secolo cui si devono fondamentali contributi in campi come teoria degli insiemi, analisi funzionale, topologia, fisica quantistica, economia, informatica, teoria dei giochi, fluidodinamica e in molti altri settori della matematica.

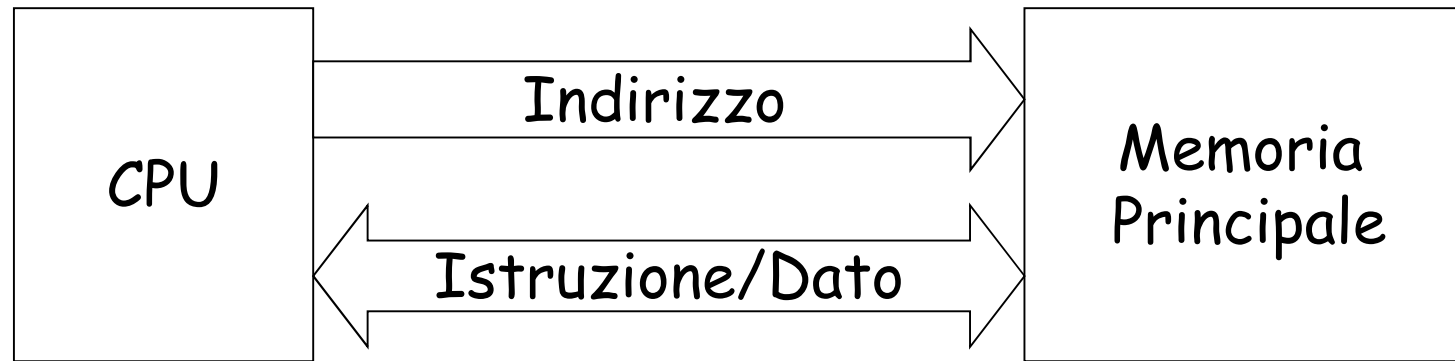


# Memoria Principale



- Insieme di celle per memorizzazione dati (**data**)
- In ogni cella si può memorizzare una stringa binaria di dimensione fissa (p.es. 1 byte = 8 bit)
- Una cella è individuata da uno specifico **indirizzo (address)**, consistente in una stringa binaria di dimensione fissa (p.es. 2 byte,  $2^{16}$  celle)
- Operazione di lettura (**read**): il dato contenuto nella cella specificata viene prelevato (*indolore*)
- Operazione di scrittura (**write**): il dato contenuto nella cella specificata assume valore specificato
- Tipicamente realizzata con memoria RAM (**Random Access Memory**):
  - tempo di accesso *indipendente* dall'indirizzo
  - *volatile*: dati persi in assenza di *alimentazione*

# Interazione CPU-Memoria Principale



- Nella memoria principale si possono distinguere due tipi di informazioni: **istruzioni** che la CPU (**Central Processing Unit**) può eseguire e **dati** di vario tipo
- Le istruzioni eseguibili da una data CPU sono rappresentate in uno specifico codice binario

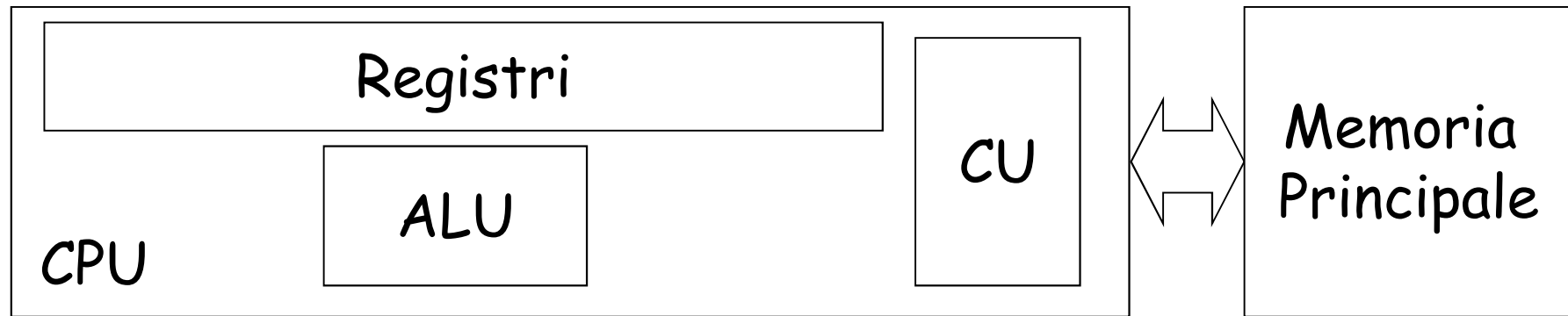
# Struttura della CPU



- Essenzialmente, la CPU contiene
  - **Unità di controllo CU (Control Unit)**: Coordina le attività interne alla CPU e le connessioni tra la CPU e le unità esterne (Memoria Principale e I/O)
  - **Unità Logico-Aritmetica ALU (Arithmetic Logic Unit)**: Esegue operazioni logico-aritmetiche
  - **Registri**: Celle di memoria interne alla CPU usate per dati di vario tipo (p. es. operandi per la ALU)



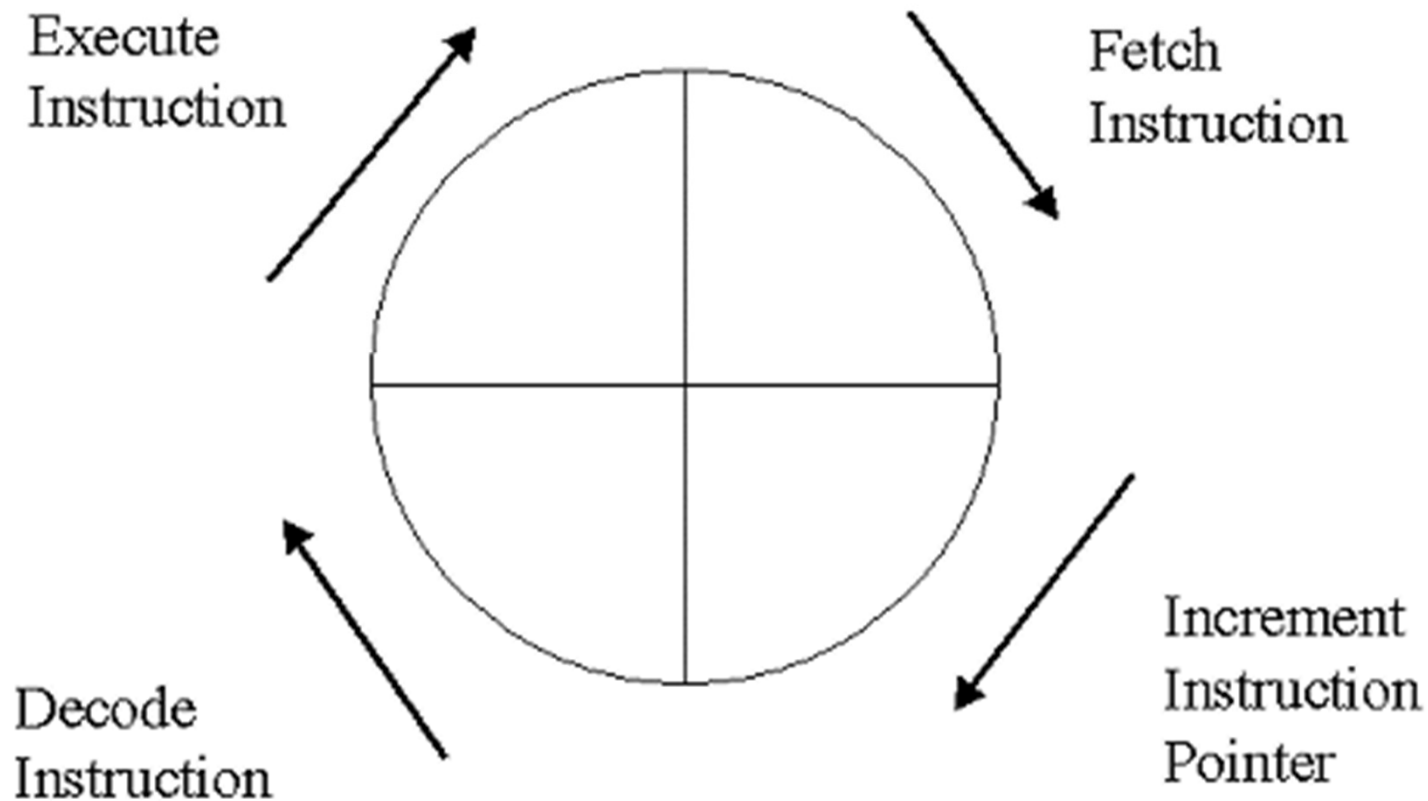
# Esecuzione Istruzioni



- ❑ **Caricamento (fetch)**: CU preleva dalla memoria principale l'istruzione da eseguire
- ❑ **Decodifica (decode)**: CU genera i comandi per l'esecuzione dell'istruzione e, se necessario, trasferisce i dati richiesti dalla memoria principale ai registri dell'unità logico-aritmetica
- ❑ **Esecuzione (execute)**: ALU produce i risultati richiesti e questi vengono memorizzati nei registri interni (o nella memoria principale)

# Fetch – Execute Cycle

## The Machine Cycle / The Fetch Execute Cycle



# Linguaggi Macchina



- Un'istruzione eseguibile da una CPU è detta **istruzione macchina** (*machine instruction*)
- Ogni CPU è in grado di eseguire soltanto uno specifico **insieme di istruzioni macchina** (*instruction set*), ognuna rappresentata secondo uno specifico codice binario detto **linguaggio macchina** (*machine language*)
- Esempi di azioni prodotte da istruzioni macchina:
  - **Esecuzione** di semplici **operazioni logico-aritmetiche**
  - **Trasferimenti** di dati tra la CPU e le unità esterne
  - **Modifica** del flusso di esecuzione delle istruzioni (normalmente sequenziale)
- Un **programma in linguaggio macchina** è costituito da un insieme **ordinato** di istruzioni macchina che verranno eseguite nell'ordine specificato

# Dispositivi I/O (periferiche)



Usati per lo scambio dati con l'esterno

- Esempi di periferiche di input
  - Tastiera, mouse, scanner, microfono, telecamera
- Esempi di periferiche di output
  - Monitor, stampante, casse acustiche
- Memorie di massa
  - Disco rigido (hard disk), dischetti, CD-ROM, pen-drive

# Memoria secondaria (di massa)



- Usata per la memorizzazione **non volatile** di informazioni (dati e/o programmi) (periferica di ingresso/uscita)
- Tipicamente, molto più lenta della memoria principale
  - 1 B = 1 byte = 8 bit
  - 1 KB = 1 kilobyte =  $2^{10}$  B
  - 1 MB = 1 megabyte =  $2^{10}$  KB =  $2^{20}$  B
  - 1 GB = 1 gigabyte =  $2^{10}$  MB =  $2^{30}$  B
  - 1 TB = 1 terabyte =  $2^{10}$  GB
  - Floppy disk (1.44 MB, 720 KB obsoleto)
  - Hard disk (10 - 2TB)
  - CD-ROM (670 MB)
  - DVD (5 GB)
  - Pen-drive USB (64MB - 64GB)

# KByte, MByte e GByte



## □ 1 KB (KiloByte)

- = 1000 byte? No!

- 1 KB =  $2^{10}$  Byte = 1024 Byte

[ $2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8, 2^4 = 16, 2^5 = 32,$

$2^6 = 64, 2^7 = 128, 2^8 = 256, 2^9 = 512, 2^{10} = 1024$ ]

## □ 1 MB (MegaByte)

- solitamente approssimato a 1 milione di Byte

- precisamente = 1024 KB =  $1024 \times 1024$  Byte =  
=  $2^{10} \times 2^{10}$  byte =  $2^{20}$  Byte = 1.048.576 Byte

## □ 1 GB (GigaByte) = circa 1 miliardo di Byte

- precisamente = 1024 MB =  $1024 \times 1024$  KB =  $1024 \times 1024 \times 1024$  Byte =  
=  $2^{10} \times 2^{10} \times 2^{10}$  Byte =  $2^{30}$  Byte = 1073741824 Byte