

ARCHITETTURA DI VON NEUMANN

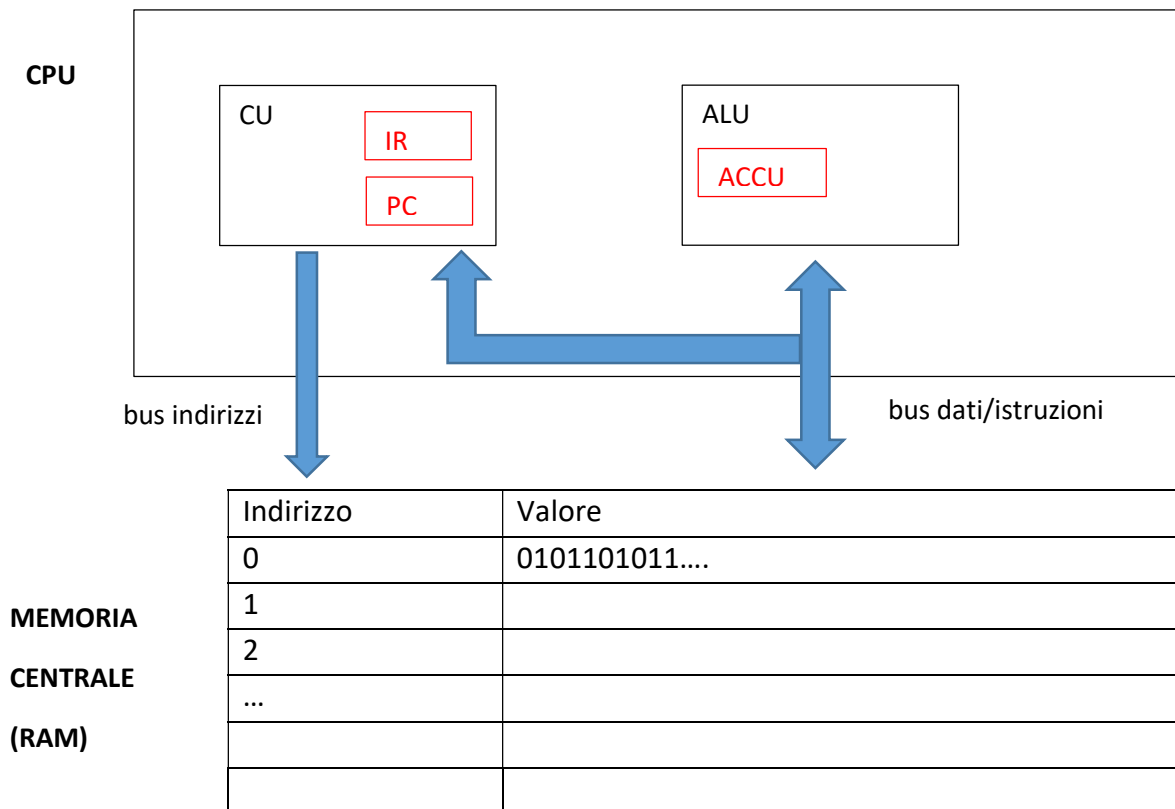
John Von Neumann è un matematico ungherese naturalizzato americano. Fra le sue molteplici attività, partecipa nel 1945 al progetto EDVAC (Electronic Discrete Variable Automatic Calculator), un progetto dell'università della Pennsylvania a Filadelfia per la realizzazione di un calcolatore automatico.

Per questo progetto Von Neumann idea una particolare architettura che si imporrà come l'architettura utilizzata in tutti i calcolatori fino ai giorni nostri. L'EDVAC è importante proprio perché è il primo computer ad implementare tale architettura, nota con il nome di **architettura di Von Neumann** (chiamiamolo VN).

Qui c'è un video con la biografia di Von Neumann (10 minuti):
https://www.youtube.com/watch?v=RF_CZpmVGzw&ab_channel=scuolanetwork

L'idea principale dell'architettura di VN è quella di utilizzare una memoria in cui memorizzare (in binario) sia i dati sia le istruzioni, a differenza del calcolatore che lo aveva preceduto, chiamato ENIAC (Electronic Numerical integrator and computer), in cui le istruzioni (programmi) non venivano memorizzati in una memoria ma venivano realizzati modificando fisicamente i collegamenti (cablaggi) dei circuiti elettrici che li implementavano. L'inserimento dei programmi e dei dati avveniva mediante nastri perforati o schede perforate.

Dal punto di vista logico l'architettura di VN è costituita da due componenti fondamentali la CPU (Central Process Unit) e la Memoria Centrale.



La CPU è “il cervello” del calcolatore e contiene due componenti: la CU e la ALU.

La CU (Contro Unit) sovrintende al funzionamento della macchina (dice cosa fare). Controlla la sequenza di istruzioni da eseguire.

La ALU (Arithmetic Logic Unit) svolge i calcoli aritmetici e logici.

L’ UC contiene due memorie chiamate registri:

- L’IR (Instruction Register) contiene il codice (codice operativo) che rappresenta l’istruzione da eseguire
- Il PC (Program Counter) contiene l’indirizzo della memoria centrale in cui si trova la prossima istruzione da eseguire.

La ALU contiene un registro chiamato ACCUMULATORE (ACCU) in cui vengono memorizzati i risultati delle operazioni svolte.

La MEMORIA CENTRALE è una tabella che contiene delle celle di memoria. Ciascuna cella di memoria è identificata da un indirizzo e contiene un valore. Il valore contenuto in una cella può essere un dato (di input, di output o di calcolo) oppure il codice operativo di un’istruzione da eseguire.

Cosa può fare la CPU:

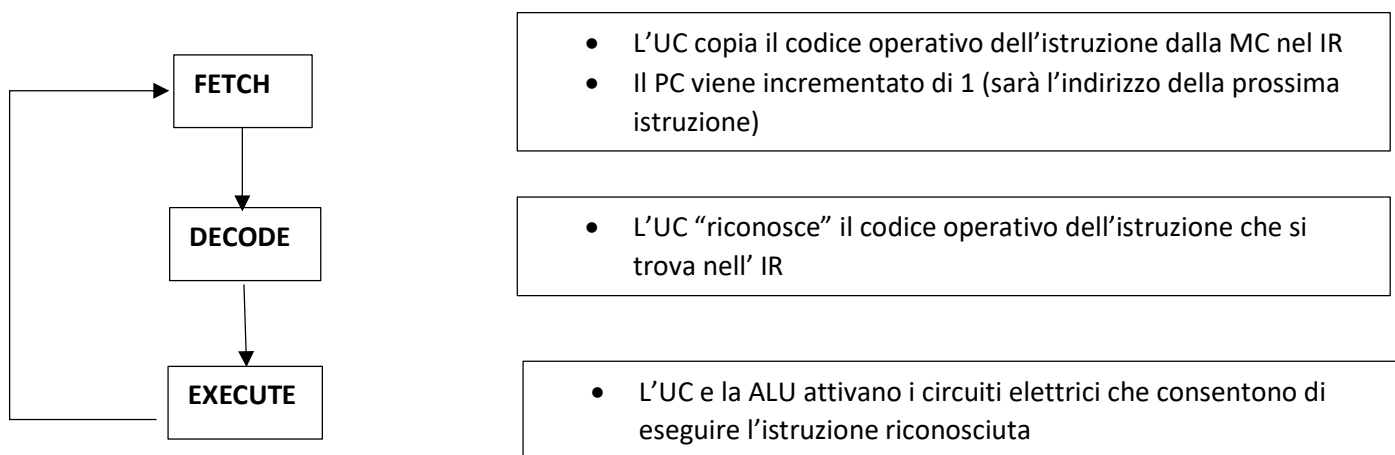
- Leggere delle istruzioni dalla MC (Memoria Centrale) e porle nell’ IR
- Leggere dei dati dalla MC e porli nell’ ACCU
- Scrivere nella MC un valore che si trova nell’ ACCU

L'indirizzo della MC in cui scrivere i dati o da cui leggere i dati/istruzioni viene scritto dalla CPU sul bus indirizzi. Il valore letto da / scritto nella memoria centrale viene trasmesso fra MC e CPU sul bus dati/istruzioni. I BUS sono costituiti da un insieme di conduttori elettrici, su ciascuno dei quali viene trasmesso un segnale di tensione elettrica che corrisponde ad un 1 (5 volt) o a uno 0 (0 volt).

Come viene eseguito un programma.

Un programma è una serie di istruzioni binarie scritte una di seguito all'altra e memorizzate nella MC. Nel PC viene posto l'indirizzo della prima istruzione del programma da eseguire.

La CPU per eseguire il programma ripete all'infinito queste tre fasi: fetch – decode – execute. Ogni ripetizione di queste fasi porta all'esecuzione di una istruzione.



Queste tre fasi vanno sincronizzate, nel senso che "la fase successiva deve iniziare solamente dopo che la fase precedente è stata conclusa". Per questa sincronizzazione si utilizza un particolare dispositivo elettronico chiamato **clock** che emette un segnale elettrico ogni intervallo di tempo (chiamato periodo di clock). L'inverso del periodo di clock è la frequenza del processore e si misura in Hertz. Ad esempio:

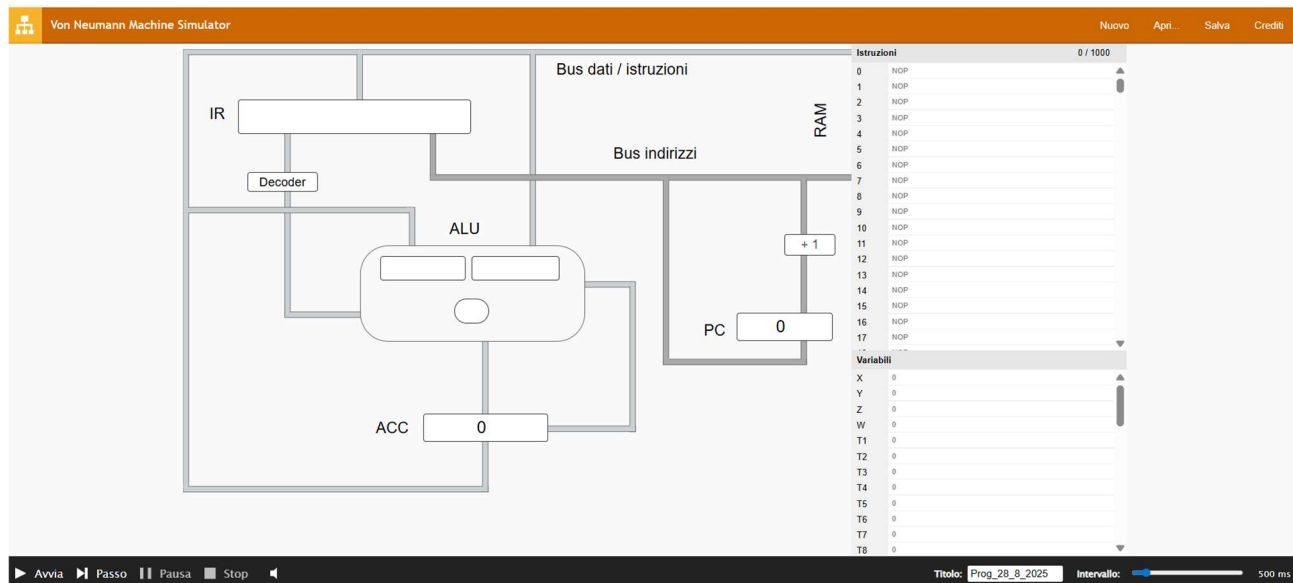
$$T \text{ di clock} = 1 \text{ ns} \quad \rightarrow \quad \text{freq di clock} = 1/(1 \cdot 10^{-9}) = 1 \cdot 10^9 \text{ Hz} = 1 \text{ GHz}$$

La CPU è progettata in modo tale che il segnale di clock non possa essere generato prima che una "fase" sia conclusa.

Nei processori moderni ogni fase è scomposta a sua volta in micro-operazioni, il clock sincronizza l'esecuzione di queste micro operazioni.

Elenco delle istruzioni

Per capire il funzionamento dell'architettura di Von Neumann che è alla base dei processori moderni, utilizzeremo un simulatore della macchina di Von Neumann. Questo simulatore è una CPU semplificata e ci permetterà di realizzare semplici programmi utilizzando un linguaggio assembly semplificato. Di seguito viene mostrato il simulatore della macchina di VN e l'elenco delle istruzioni che essa è in grado di eseguire. Nel simulatore gli indirizzi che contengono le istruzioni del programma sono numerici (0,1,2...) mentre quelli che contengono i dati sono identificati da lettere maiuscole (X,Y,Z,...)



Istruzione	Descrizione sintattica	Operazione calcolata
LOD ind	Carica il valore contenuto nella locazione di memoria di indirizzo <i>ind</i> nel registro accumulatore	$A \leftarrow RAM[ind]$
LOD #num	Carica direttamente il numero specificato nel registro accumulatore	$A \leftarrow num$
STO ind	Copia il valore contenuto nel registro accumulatore nella locazione di memoria di indirizzo <i>ind</i>	$RAM[ind] \leftarrow A$
ADD ind	Addiziona il valore contenuto nel registro accumulatore e il valore contenuto nella locazione di memoria di indirizzo <i>ind</i> calcolando la somma nel registro accumulatore	$A \leftarrow A + RAM[ind]$
ADD #num	Addiziona il valore contenuto nel registro accumulatore e il numero specificato calcolando la somma nel registro accumulatore	$A \leftarrow A + num$
SUB ind	Sottrae il valore contenuto nella locazione di memoria di indirizzo <i>ind</i> dal valore contenuto nel registro accumulatore calcolando la differenza nel registro accumulatore	$A \leftarrow A - RAM[ind]$
SUB #num	Sottrae il numero specificato dal valore contenuto nel registro accumulatore calcolando la differenza nel registro accumulatore	$A \leftarrow A - num$
MUL ind	Moltiplica il valore contenuto nel registro accumulatore con il valore contenuto nella locazione di memoria di indirizzo <i>ind</i> calcolando il prodotto nel registro accumulatore	$A \leftarrow A \times RAM[ind]$
MUL #num	Moltiplica il valore contenuto nel registro accumulatore con il numero specificato calcolando il prodotto nel registro accumulatore	$A \leftarrow A \times num$
DIV ind	Divide il valore contenuto nel registro accumulatore per il valore contenuto nella locazione di memoria di indirizzo <i>ind</i> calcolando il quoziente nel registro accumulatore (il quoziente è un numero intero anche se il valore contenuto in A non è divisibile per il valore contenuto nella memoria: in questo caso viene trascurato il «resto»)	$A \leftarrow A : RAM[ind]$

Istruzione	Descrizione analitica	Descrizione sintattica
DIV #num	Divide il valore contenuto nel registro accumulatore per il numero specificato calcolando il quoziente nel registro accumulatore (il quoziente è un numero intero anche se il valore contenuto in A non è divisibile per il numero specificato: in questo caso viene trascurato il «resto»)	$A \leftarrow A : \text{num}$
JMP ind	Inserisce il valore <i>ind</i> nel registro PC (la prossima istruzione eseguita sarà l'istruzione contenuta nella locazione di memoria avente indirizzo <i>ind</i> , anziché l'istruzione contenuta nella posizione di memoria successiva a quella contenente questa istruzione)	$PC \leftarrow \text{ind}$
JMZ ind	Inserisce il valore <i>ind</i> nel registro PC se il contenuto del registro accumulatore è zero (in questo caso la prossima istruzione eseguita sarà l'istruzione contenuta nella locazione di memoria avente indirizzo <i>ind</i> , altrimenti sarà l'istruzione contenuta nella locazione di memoria successiva a quella contenente questa istruzione)	Se A è zero allora $PC \leftarrow \text{ind}$
NOP	Operazione nulla	
HLT	Interrompe il ciclo di funzionamento della macchina	

Si osservi che le istruzioni che comportano operazioni aritmetiche sono sempre svolte fra un dato contenuto nell'ACCU e un dato contenuto in un certo indirizzo della MC indicato con RAM[ind]. Il risultato viene sempre memorizzato nell' ACCU. Le operazioni aritmetiche sono dunque del seguente tipo:

$\text{ACCU} \leftarrow \text{ACCU} \blacklozenge \text{RAM}[\text{Ind}]$

Significa: prendi il valore di ACCU, esegui la generica operazione (+,-,x,/) con il valore all'indirizzo "Ind" della MC e poni il risultato in ACCU

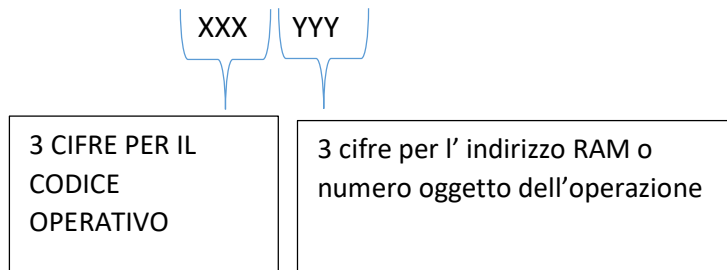
Nei processori moderni le operazioni che la CPU è in grado di svolgere sono molto più numerose. I processori si classificano in base alla dimensione del loro set di istruzioni in due famiglie:

- **CISC (Complex Instruction Set Computer):** hanno un set di istruzioni costituito da numerose istruzioni (migliaia) che consentono di eseguire operazioni complesse.
- **RISC (Reduced Instruction Set Computer):** hanno un set di istruzioni costituito da meno istruzioni (centinaia) che consentono di eseguire operazioni meno complesse. Per realizzare un'operazione con un processore RISC è necessario dunque utilizzare più istruzioni, che però, essendo più semplici, vengono generalmente eseguite più velocemente delle istruzioni CISC.

Elenco dei codici operativi.

Come detto, la RAM contiene nelle celle di memoria le istruzioni del programma. Per semplicità ogni operazione viene rappresentata da un codice mnemonico (LOD, STO, ADD, ecc.) ma in realtà nella MC saranno memorizzate con un codice numerico che le identifica (il codice operativo).

Ogni istruzione è formata da due parti:



ELENCO CODICI OPERATIVI SIMULATORE MVN

TABELLA 6

Istruzione	Codice operativo	Esempio
LOD ind	4.ind	LOD 100 → 4100
LOD #num	20.num	LOD #10 → 20 010
STO ind	5.ind	STO 101 → 5101
ADD ind	0.ind	ADD 100 → 0.100
ADD #num	16.num	ADD #1 → 16 001
SUB ind	1.ind	SUB 100 → 1100
SUB #num	17.num	SUB #1 → 17 001
MUL ind	2.ind	MUL 100 → 2100
MUL #num	18.num	MUL #2 → 18 002
DIV ind	3.ind	DIV 100 → 3100
DIV #num	19.num	DIV #2 → 19 002
JMP ind	12.ind	JMP 10 → 12 010
JMZ ind	13.ind	JMZ 10 → 13 010
NOP	14.000	
HLT	15.000	

Quindi, ad esempio:

4100 è il codice per l'operazione LOD 100

20010 è il codice per l'operazione LOD #10

17001 è il codice per l'operazione SUB #1

Quando un'istruzione (ad esempio 17001) viene caricata nell' IR, la UC la decodifica ed attiva i circuiti elettrici che consentono di eseguire l'istruzione.

Questo per sottolineare che sia le istruzioni sia i valori sono rappresentati da numeri e che quindi possono essere convertiti in binario e rappresentati nei dispositivi elettronici come valori 1 e 0.

ESEMPI ED ESERCIZI CON SIMULATORE

Vedi cartella Esempi Esercizi 2025 2026