Architettura degli Elaboratori I

Corso di Laurea Triennale in Informatica

Università degli Studi di Milano

Dipartimento di Informatica “Giovanni Degli Antoni”

Edizione 2 (Cognomi H-Z), A.A. 2024-2025, Nicola.Basilico@unimi.it

Introduzione

Di cosa si occupa questo insegnamento?

Cerchiamo il significato di «architettura» nel dizionario:

- *arte e **tecnica del progettare**, disegnare, **realizzare** edifici o altre **grandi opere**, basandosi su principi estetici e **ingegneristici***

La «grande opera» che ci proponiamo di studiare si chiama **elaboratore**

Sempre dal dizionario:

- *(in informatica) **struttura** dei componenti di un sistema di elaborazione*

Obiettivi

- Comprendere i principi fondamentali alla base della realizzazione di un elaboratore digitale

- Studieremo come l'informazione viene rappresentata dentro un elaboratore e secondo quali principi essa può essere manipolata
- Analizzeremo i componenti fondamentali che permettono di far eseguire allo hardware operazioni elementari
- Combineremo tali componenti per realizzare elaborazioni sempre più complesse fino a ottenere …
- … il componente centrale di un elaboratore: la CPU

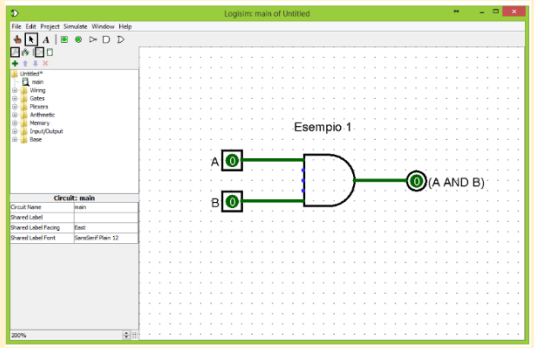
Il programma che seguiremo

Teoria:

- introduzione
- rappresentazione ed elaborazione logica dell'informazione: codifiche binarie per numeri naturali, interi e reali, funzioni logiche e algebra di Boole;
- rappresentazione ed elaborazione fisica dell'informazione: porte logiche e tabelle di verità;
- sintesi di funzioni logiche con circuiti combinatori, realizzazione dell'unità aritmetico-logica (ALU);
- memorizzazione dell'informazione: elementi di logica sequenziale (bistabili, latch, sincronizzazione tramite clock);
- sintesi di circuiti sequenziali, macchine a stati finiti;
- progetto di una CPU singolo-ciclo e cenni al caso multi-ciclo;

Laboratorio

- codifica binaria;
- logica combinatoria (forme canoniche, cammino critico);
- logica combinatoria avanzata (moltiplicazione, ALU);
- logica sequenziale (memorie, macchine a stati finiti);



logisim

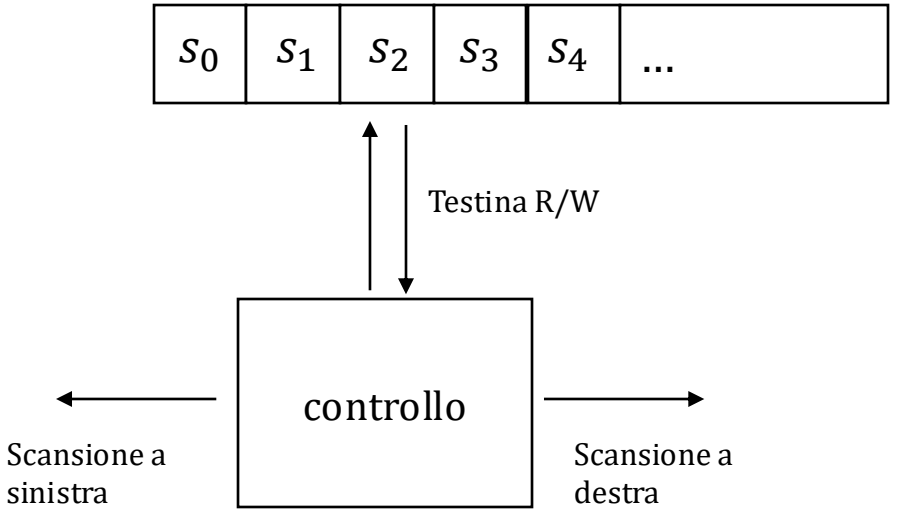
L’elaboratore

Ci sono molte definizioni possibili, proviamo a darne una:
un elaboratore è una macchina capace di rappresentare, memorizzare e manipolare delle informazioni date in input per produrne delle altre in output

Esempio: una macchina che riceve in input 10 numeri interi, li ordina e salva il risultato in memoria

Il concetto stesso di elaborazione è intimamente connesso a quello di macchina che la esegue

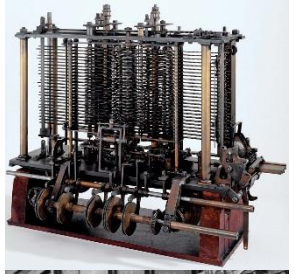
Il modello matematico più noto dell’elaborazione si chiama: **Macchina di Turing** (1936), è un elaboratore teorico in grado di eseguire qualsiasi algoritmo



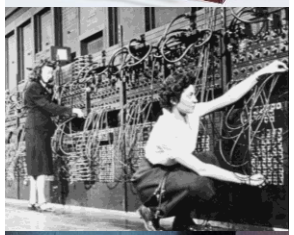
Breve storia dell'elaboratore

- Era pre-elettronica: calcolatori meccanici e elettromeccanici (1623-1945)

- **Calcolatori elettronici** (salti generazionali associati ad importanti innovazioni tecnologiche):
  - Prima generazione (1946-1955): utilizzo di valvole, diodi e triodi, prestazioni migliorate di 1000 volte. ENIAC, programmi realizzati cambiando il cablaggio della macchina manualmente
  - Seconda generazione (1952-1963): introduzione dell'elettronica allo stato solido (semiconduttori) e delle memorie ferromagnetiche, primo super-calcolatore CDC, invenzione del Fortran, il primo linguaggio di alto livello
  - Terza generazione (1964-1971): avvento dei circuiti integrati, IBM 360 prima famiglia di calcolatori
  - Quarta generazione (1971-1977), miniaturizzazione in larga scala (VLSI), introduzione del microprocessore, memorie a semiconduttori, Apple II (1977)
  - Quinta generazione (1978-2003): avvento dei Personal Computer, PC come workstation
  - Futuro: quantum computers, molecular computers, …



L' Analytical Engine progettato da Charles Babbage (1837)



ENIAC (1946), primo computer general purpose, Università della Pennsylvania

IBM 7090 (1959) usa transistor anziché valvole

IBM 360 (1964), la prima famiglia di calcolatori

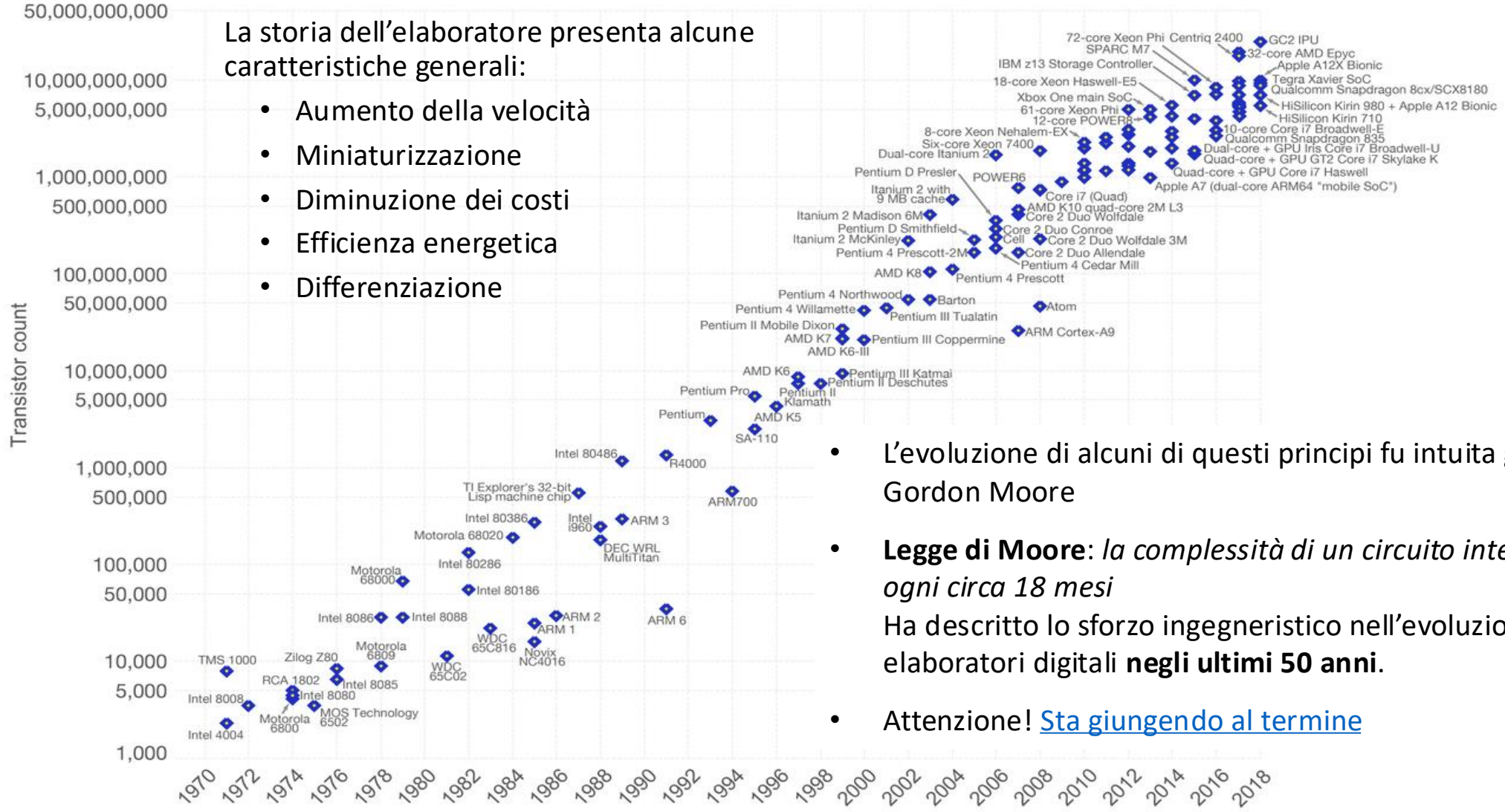
Apple II (1977), primo home computer su larga scala

MS DOS (1982)

Moore’s law

La storia dell’elaboratore presenta alcune caratteristiche generali:

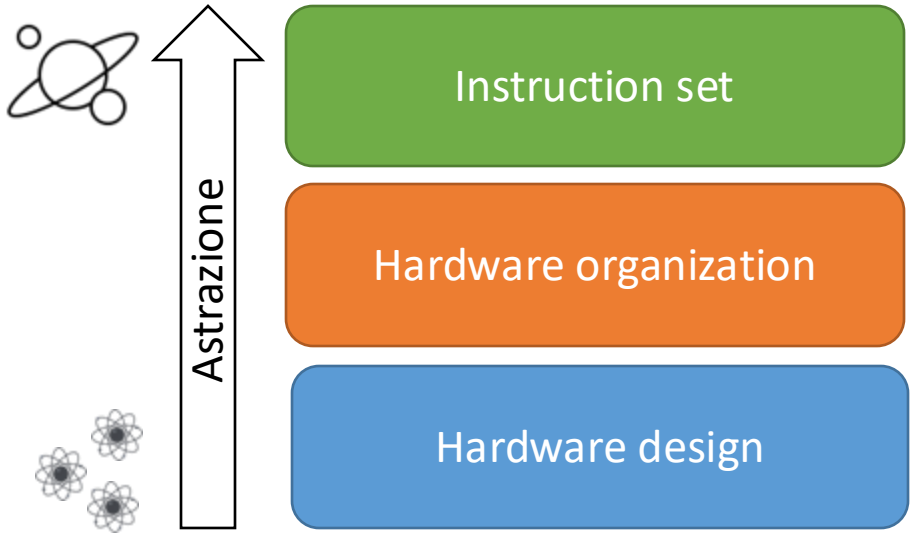
- Aumento della velocità
- Miniaturizzazione
- Diminuzione dei costi
- Efficienza energetica
- Differenziazione



- L’evoluzione di alcuni di questi principi fu intuita già nel 1965 da Gordon Moore
- **Legge di Moore**: *la complessità di un circuito integrato raddoppia ogni circa 18 mesi*
  Ha descritto lo sforzo ingegneristico nell’evoluzione degli elaboratori digitali **negli ultimi 50 anni**.
- Attenzione! Sta giungendo al termine

Architettura di un elaboratore

- L'architettura di un elaboratore è una descrizione di come è fatta una macchina in grado di svolgere elaborazione automatica dell'informazione



Quali sono le operazioni elementari (istruzioni) che la macchina può eseguire

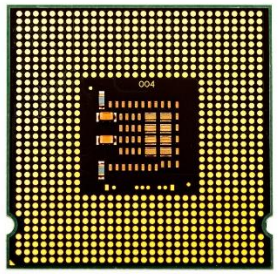
Come i componenti sono connessi e operano tra loro

Design logico dei circuiti che realizzano i vari componenti dell'elaboratore: ALU, control unit, bus, …

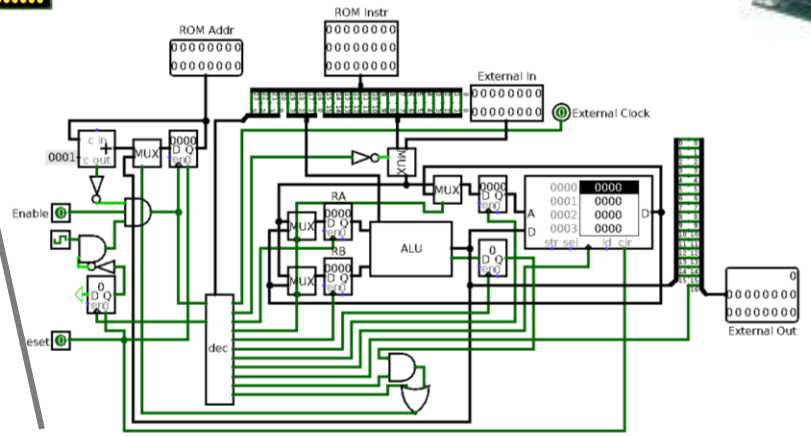
- Il vantaggio dell'astrazione: ogni livello si avvale degli elementi definiti nel livello sottostante trascurando come questi sono fatti all'interno
- In questo corso seguiamo un approccio bottom-up

Hardware

- È l’insieme dei componenti fisici che compongono un elaboratore (la «ferramenta»), ciascuno adibito ad una particolare funzione

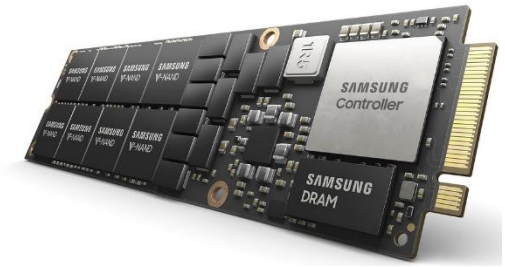


CPU



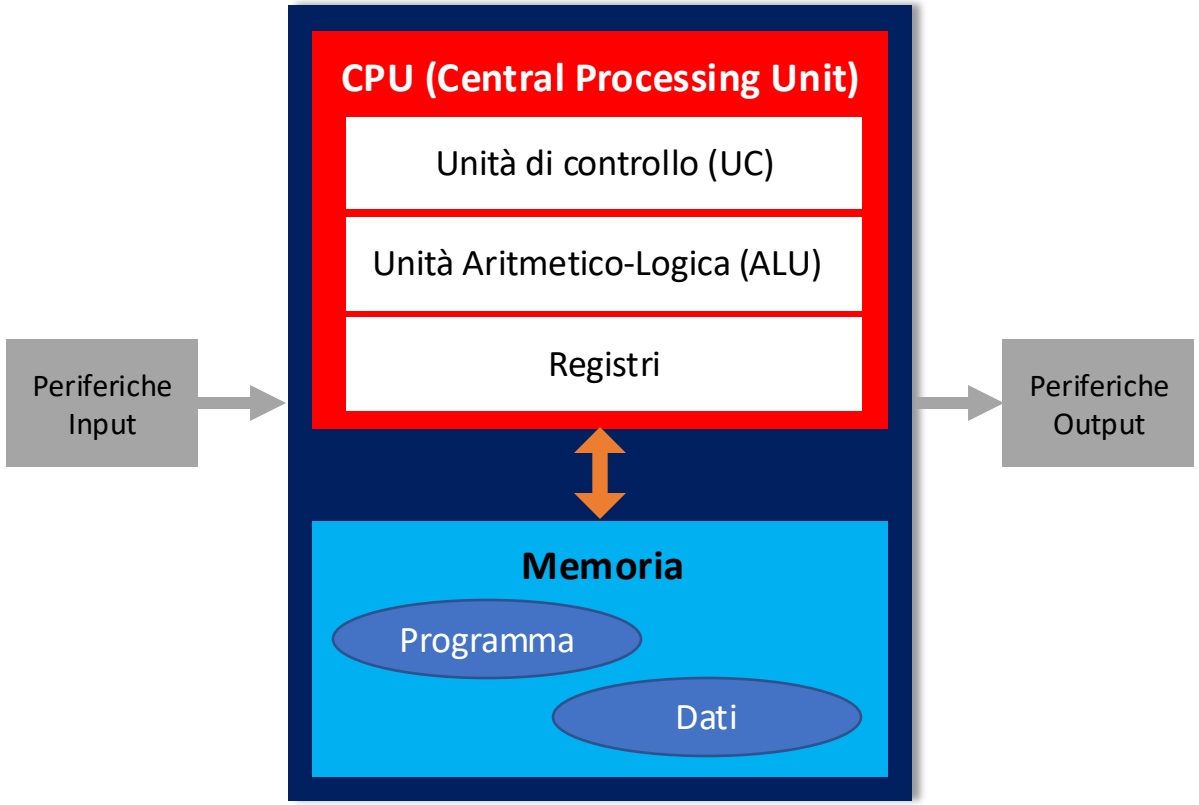
Motherboard

Memoria



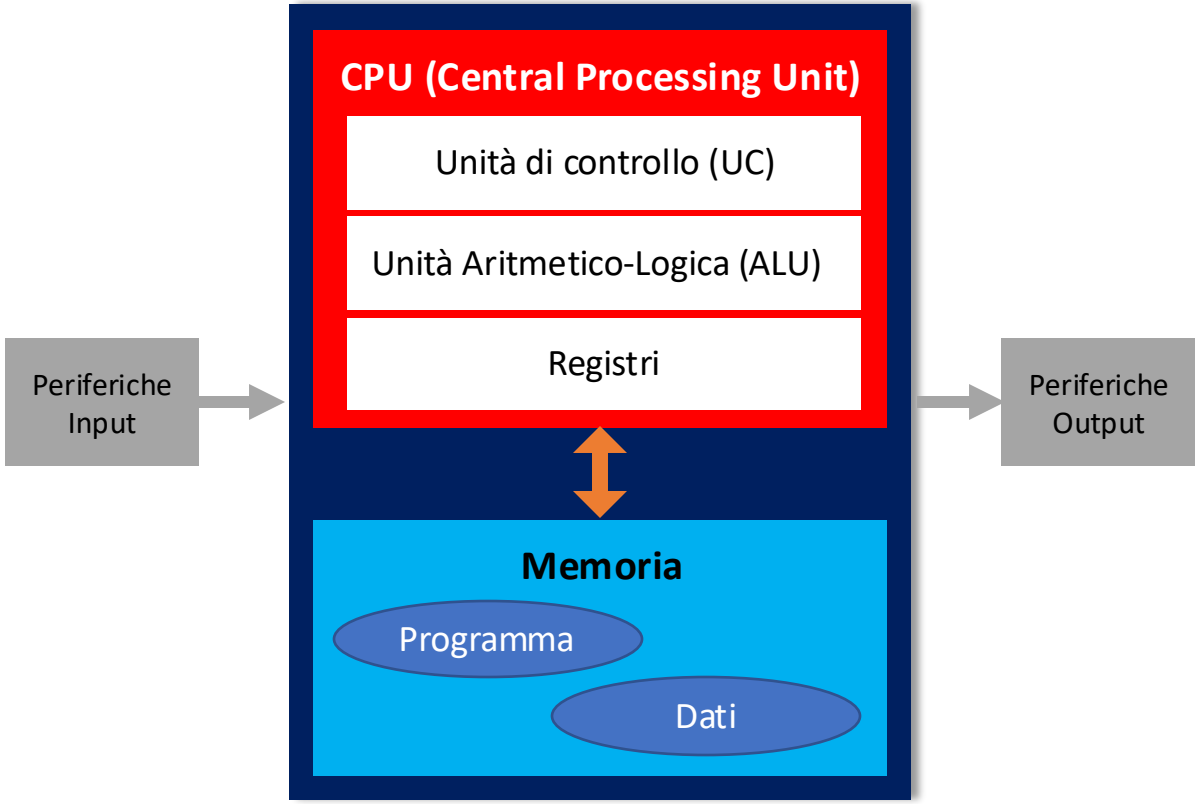
Organizzazione

- L'organizzazione della maggior parte degli elaboratori oggi segue un modello chiamato **Architettura di Von Neumann** o anche Macchina di Von Neumann



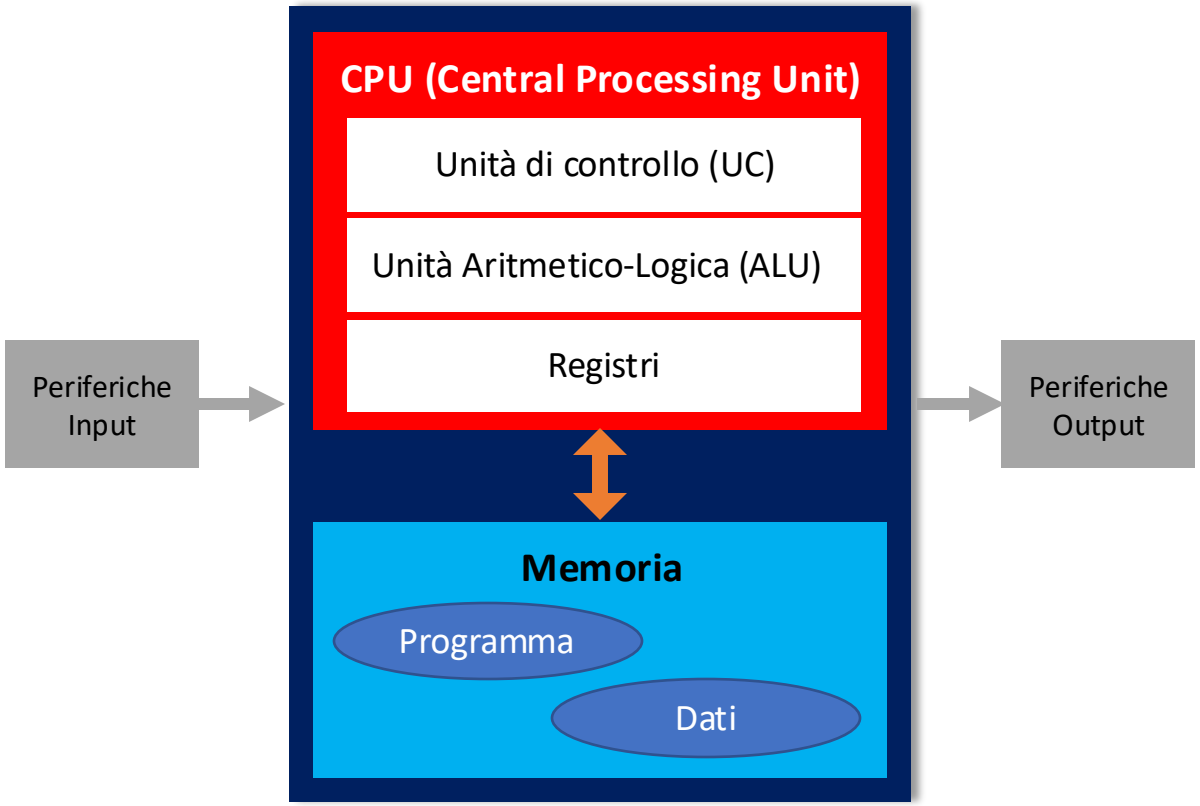
- Vediamo una panoramica generale che approfondiremo poi in questo e nel corso di Architettura II

Von Neumann Architecture (1)



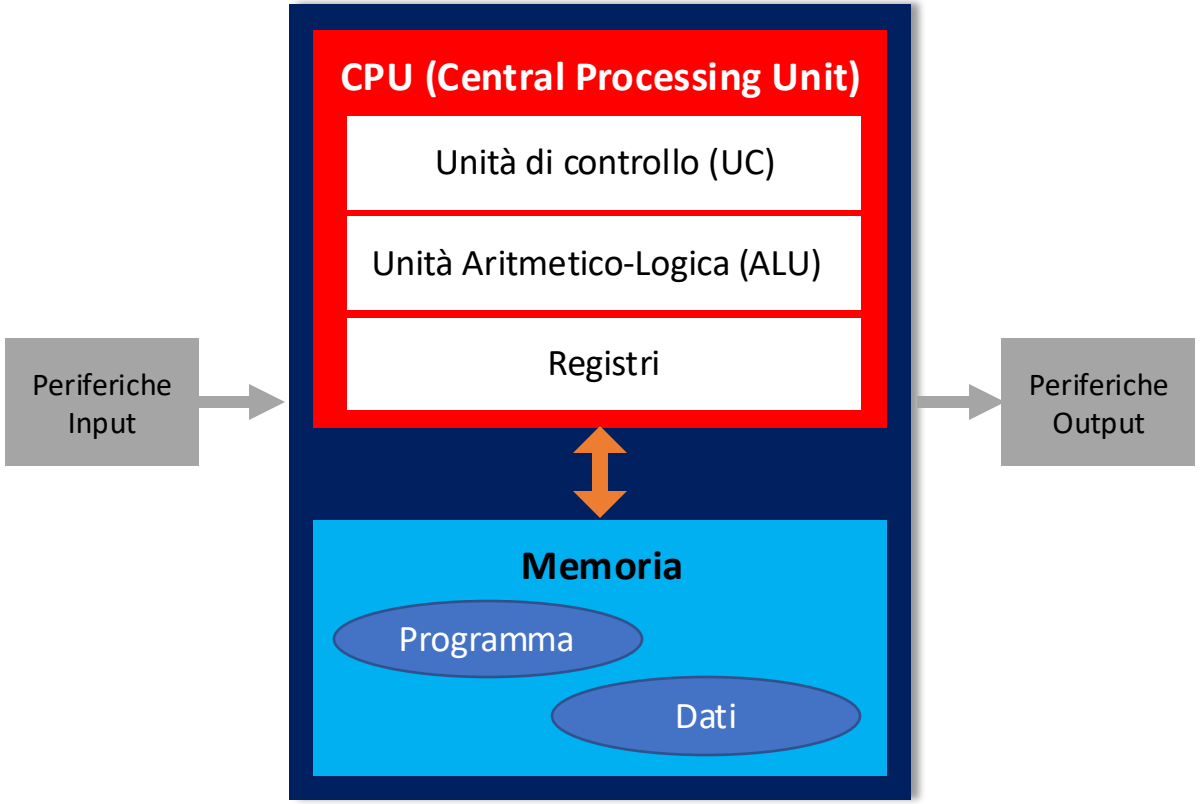
- La **CPU** ha il compito di eseguire delle istruzioni in sequenza (ad esempio somme, moltiplicazioni, test di disuguaglianza, accesso a memoria, …)
- Per fare questo si avvale di
  - **UC**: supervisore dell'esecuzione
  - **ALU**: esecutore di calcoli logici o aritmetici
  - **Registri**: memoria interna, il banco di lavoro

Von Neumann Architecture (2)



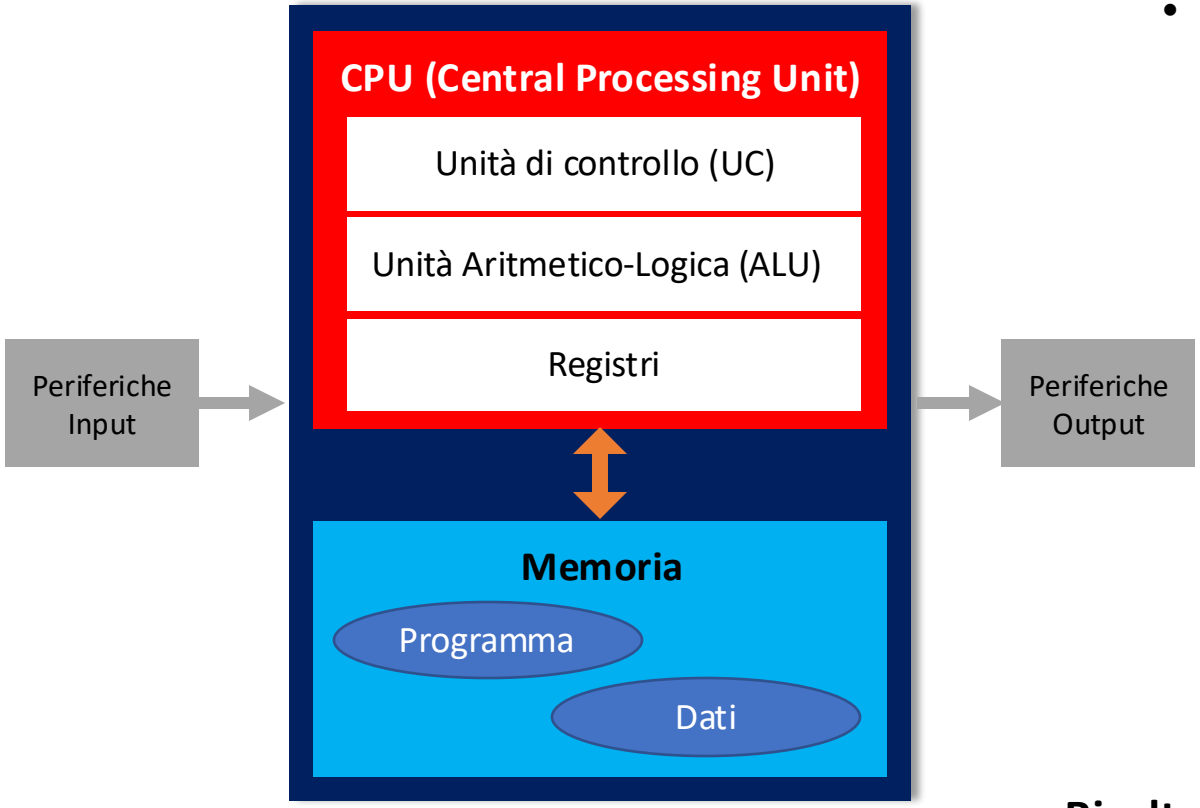
- La **Memoria** ha il compito immagazzinare
  - **Il programma** (detto anche «testo»): la sequenza di istruzioni che deve eseguire la CPU
  - **I dati**: valori su cui le istruzioni lavorano o che rappresentano i loro risultati
- La CPU può accedervi in lettura o in scrittura attraverso un canale di comunicazione detto **bus**
- È strutturata in parole, ogni parola ha un indirizzo. Per leggere/scrivere un dato da/in una parola è necessario specificare l'indirizzo di tale parola

Von Neumann Architecture (3)



- Un elaboratore deve poter comunicare con il mondo esterno, ad esempio per chiedere un dato all'utente o visualizzare il risultato di un'operazione su di un terminale:
  - **Periferiche di input**: acquisizione un dato (tastiera, mouse, dischi, …)
  - **Periferiche di output**: visualizzare il dato (display, dischi, …)

Von Neumann Architecture (4)



- Limiti di questo modello: **Bottleneck** (collo di bottiglia)

L'evoluzione delle CPU (cfr. legge di Moore) ha portato ad un drammatico aumento delle performance (velocità di esecuzione)

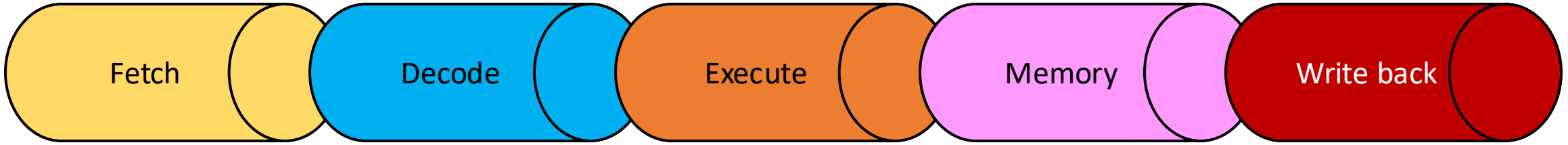
Le memorie, d'altro canto, si sono evolute principalmente in densità e affidabilità (anche in velocità ma non quanto le CPU!)

Risultato: la CPU non fa altro che aspettare!

Soluzione? Pensare ad una architettura più «complicata» (vedremo più avanti)

Ciclo di esecuzione

- I componenti della Architettura di Von Neumann che abbiamo introdotto svolgono ciascuno, in modo complementare, una particolare funzione
- Interagendo e coordinandosi tra di loro, questi componenti collaborano all'esecuzione di un programma, istruzione dopo istruzione
- L'esecuzione di ciascuna istruzione attraversa **5 fasi** che, ripetute per ogni istruzione, costituiscono il **ciclo di esecuzione**



L'istruzione viene prelevata da memoria e caricata in un registro speciale (Instruction Register)

La UC riconosce di che istruzione si tratta e predispone la ALU per la sua esecuzione

La ALU svolge le operazioni aritmetico/logiche necessarie all'esecuzione dell'istruzione

Viene effettuato un accesso a memoria se l'istruzione lo prevede

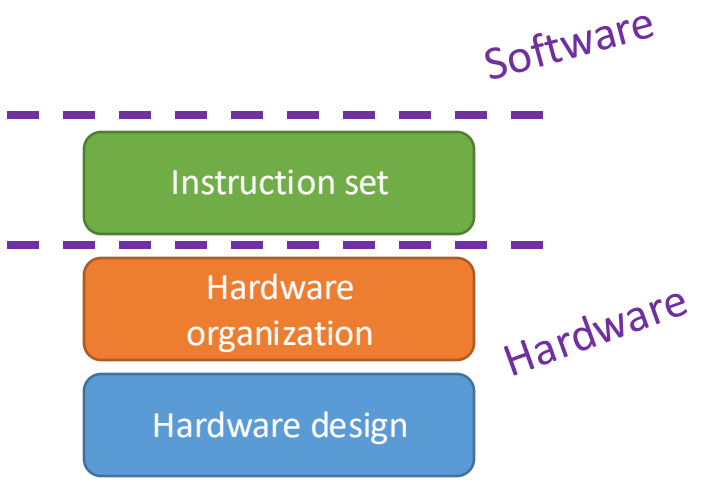
Vengono aggiornati i registri se l'istruzione lo prevede

Instruction Set

ADD Eb Gb 00	ADD Ev Gv 01	ADD Gb Eb 02	ADD Gv Ev 03	ADD AL Ib 04	ADD eAX Iv 05	PUSH ES 06	POP ES 07	OR Eb Gb 08	OR Ev Gv 09	OR Gb Eb 0A	OR Gv Ev 0B	OR AL Ib 0C
ADC Eb Gb 10	ADC Ev Gv 11	ADC Gb Eb 12	ADC Gv Ev 13	ADC AL Ib 14	ADC eAX Iv 15	PUSH SS 16	POP SS 17	SBB Eb Gb 18	SBB Ev Gv 19	SBB Gb Eb 1A	SBB Gv Ev 1B	SBB AL Ib 1C
AND Eb Gb 20	AND Ev Gv 21	AND Gb Eb 22	AND Gv Ev 23	AND AL Ib 24	AND eAX Iv 25	ES: 26	DAA 27	SUB Eb Gb 28	SUB Ev Gv 29	SUB Gb Eb 2A	SUB Gv Ev 2B	SUB AL Ib 2C
XOR Eb Gb 30	XOR Ev Gv 31	XOR Gb Eb 32	XOR Gv Ev 33	XOR AL Ib 34	XOR eAX Iv 35	SS: 36	AAA 37	CMP Eb Gb 38	CMP Ev Gv 39	CMP Gb Eb 3A	CMP Gv Ev 3B	CMP AL Ib 3C
INC eAX 40	INC eCX 41	INC eDX 42	INC eBX 43	INC eSP 44	INC eBP 45	INC eSI 46	INC eDI 47	DEC eAX 48	DEC eCX 49	DEC eDX 4A	DEC eBX 4B	DEC eSP 4C
PUSH eAX 50	PUSH eCX 51	PUSH eDX 52	PUSH eBX 53	PUSH eSP 54	PUSH eBP 55	PUSH eSI 56	PUSH eDI 57	POP eAX 58	POP eCX 59	POP eDX 5A	POP eBX 5B	POP eSP 5C
PUSHA 60	POPA 61	BOUND Gv Ma 62	ARPL Ew Gw 63	FS: 64	GS: 65	OPSIZE: 66	ADSIZE: 67	PUSH Iv 68	IMUL Gv Ev Iv 69	PUSH Ib 6A	IMUL Gv Ev Ib 6B	INSB Yb DX 6C
JO Jb 70	JNO Jb 71	JB Jb 72	JNB Jb 73	JZ Jb 74	JNZ Jb 75	JBE Jb 76	JA Jb 77	JS Jb 78	JNS Jb 79	JP Jb 7A	JNP Jb 7B	JL Jb 7C
ADD Eb Ib	ADD Ev Iv	SUB Eb Ib	SUB Ev Ib	TEST Eb Gb	TEST Ev Gv	XCHG Eb Gb	XCHG Ev Gv	MOV Eb Gb	MOV Ev Gv	MOV Gb Eb	MOV Gv Ev	MOV Ew Sw

- L'instruction set è l'insieme delle istruzioni che la macchina è in grado di eseguire (esempio, add, $load$, and, …)
- Sono espresse in linguaggio macchina: sequenze di bit (per noi saranno 32) che la UC è in grado di riconoscere come istruzione (decodifica)
- Conoscere l'instruction set di una macchina ci permette di scrivere programmi per quella macchina senza dover considerare i dettagli hardware (astrazione). Ad esempio potremmo scrivere un compilatore!

- È il primo livello di astrazione sull'hardware: **l'inizio del software**
- Così importante da essere chiamato **«Instruction Set Architecture» (ISA)** e a volte anche semplicemente «Architecture»
- Macchine diverse possono implementare la stessa ISA (eseguono gli stessi programmi) ma con hardware diverso (e quindi diverse prestazioni e costi)



- **ISA RISC** (Reduced Instruction Set Computer):
  istruzioni semplici e regolari, ciascuna richiede all'hardware poco lavoro per essere eseguita
- **ISA CISC** (Complex Instruction Set Computer):
  istruzioni complesse ciascuna può svolgere diversi task e richiede più lavoro all'hardware

RISC e CISC (1)

RISC

- **Vantaggi:**
  - **Semplicità**: istruzioni semplici e regolari, più facili da progettare, ma anche decodificare ed eseguire.
  - **Prestazioni**: istruzioni più semplici possono eseguite più rapidamente, si possono ottenere alte prestazioni in task di calcolo che richiedono molte operazioni aritmetico/logiche.
  - **Efficienza**: istruzioni più semplici comportano un consumo di energia più basso e una minore emissione di calore, sono quindi adatte per dispositivi mobili e sistemi embedded.
  - **Facilità per il compilatore**: più facili da compilare.
- **Svantaggi**
  - **Aumento della quantità del codice**: per fare cose complesse servono tantissime istruzioni
  - **Accessi più frequenti alla memoria**: quindi tempi di attesa

CISC

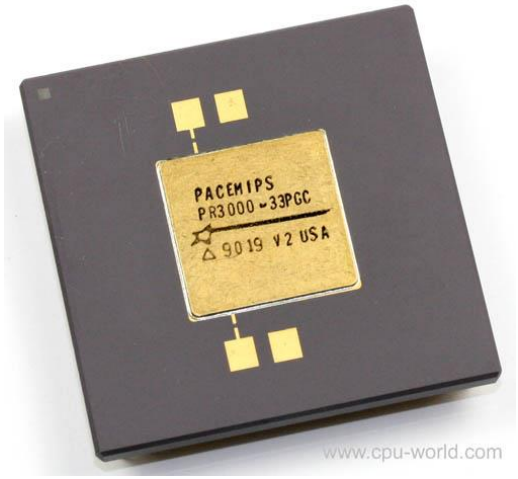
- **Vantaggi:**
  - **Codice compatto**: una singola istruzione può corrispondere a più operazioni
  - **Set di istruzioni molto ricco**: maggiore flessibilità
  - **Meno accessi a memoria**: meno attese
- **Svantaggi**
  - **Complessità**: è richiesta una architettura più complessa, più difficile da progettare e con consumi energetici più elevati
  - **Difficoltà per il compilatore**: generare codice efficiente è un task molto più difficile
  - **Obsolescenza**: la specificità di un'istruzione può portare ad un suo abbandono nel tempo

RISC e CISC (2)

- Le architetture RISC sono usate nei dispositivi embedded (NAS, stampanti, router, TV, …), device mobili (come gli smartphone) e nei componenti IoT
  - PowerPC: usata in alcuni PC (in passato anche da Apple), ora per lo più in processori integrati (e.g., Wii U)
  - ARM: smartphone, raspberry pi, …
  - RISC V: Open source, percepita come concorrente di ARM
  - **MIPS:** quella che studieremo noi!
  - Una istruzione «molto RISC» (MIPS): `lw $9 8($6)` #*copia la parola di memoria all'indirizzo $6+8 nel registro $9*

- Le architetture CISC sono usate spesso nelle macchine general purpose (come i PC Desktop) e i server
  - Intel X86 (e sua estensione X64)
  - AMD 64
  - Una istruzione «molto CISC» (x86): `rep movsb` *#copia il byte all'indirizzo SI nel byte di indirizzo DI, incrementa SI e DI, decrementa CX, se CX non è 0 ripeti*

MIPS



- L'architettura di riferimento per questo corso è il **MIPS** (Multiprocessor without Interlocked Pipeline Stages) che implementa un'ISA di tipo **RISC**
- Nasce a metà anni '80 come architettura *general purpose*

- Inizialmente è un progetto accademico (Stanford), poco dopo diventa commerciale
- Oggi è impiegata prevalentemente nell'ambito dei *sistemi embedded*
- È adottata nella maggior parte dei corsi accademici per la sua semplicità (nel corso di Architettura II approfondiremo questa sua caratteristica)

MIPS timeline