

# Architettura degli Elaboratori

## Introduzione



SAPIENZA  
UNIVERSITÀ DI ROMA

Alessandro Checco

[alessandro.checco@uniroma1.it](mailto:alessandro.checco@uniroma1.it)

Special thanks and credits:

Claudio Di Ciccio, Iacopo Masi, e

Andrea Sterbini

# Alessandro Checco



Assistant professor

Ph.D. in Mathematics, Hamilton Institute, Ireland

Main research interests:

Data privacy

Crowdsourcing

Information Retrieval

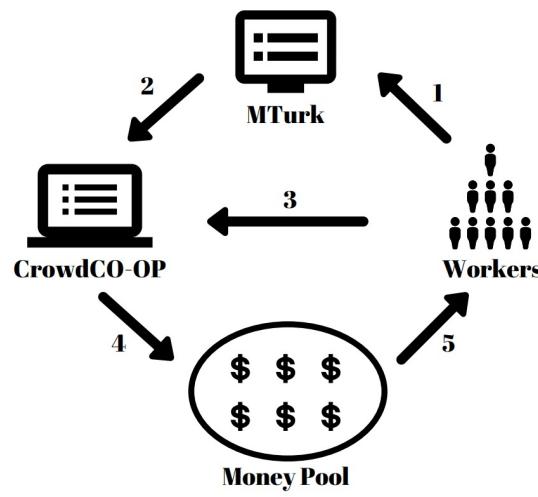
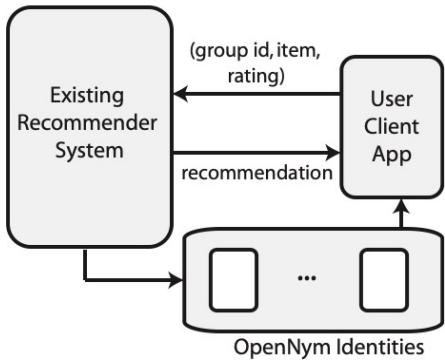
Bayesian Modeling

Decentralized Machine Learning

Web Science



User accesses system via an account shared with other users having similar interests.



# Logistica e Comunicazioni

Accesso al materiale e comunicazioni: Google Classroom

Architettura degli Elaboratori (a.a. 2023-24)  
Canale 2 (M-Z)

Stream Classwork People

tori...  
23-...

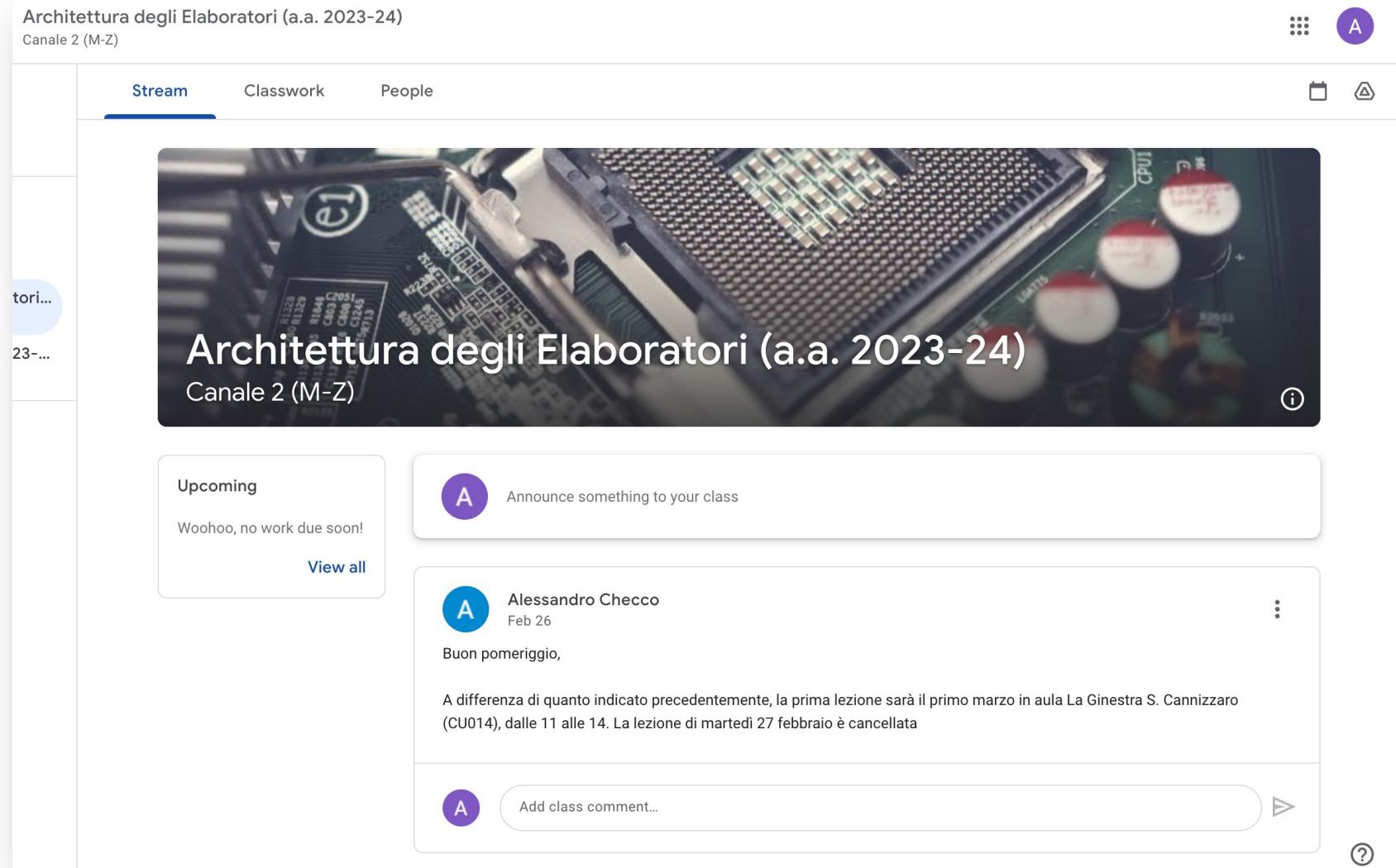
**Architettura degli Elaboratori (a.a. 2023-24)**  
Canale 2 (M-Z)

Upcoming  
Woohoo, no work due soon!  
[View all](#)

A Announce something to your class

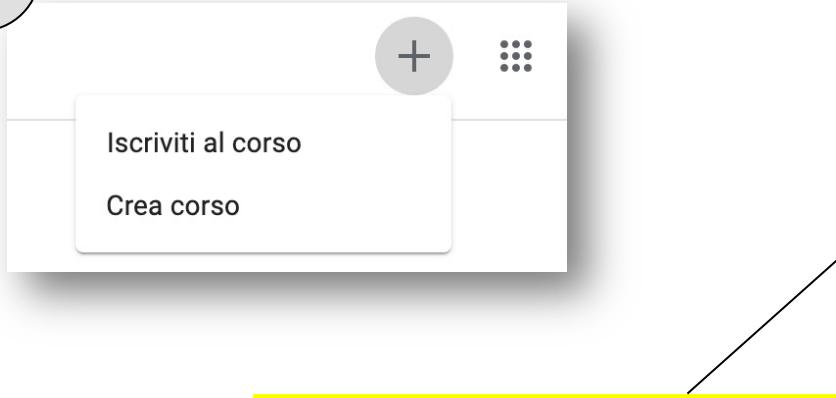
Alessandro Checco  
Feb 26  
Buon pomeriggio,  
  
A differenza di quanto indicato precedentemente, la prima lezione sarà il primo marzo in aula La Ginestra S. Cannizzaro (CU014), dalle 11 alle 14. La lezione di martedì 27 febbraio è cancellata

Add class comment... ▶ ?



# Google Classroom

1



Codice corso  
Chiedi il codice del corso all'insegnante e inseriscilo qui.

Per accedere con un codice di corso

- Utilizza un account autorizzato
- Utilizza un codice corso con 5-7 lettere o numeri, senza spazi né simboli

Se hai problemi a iscriverti al corso, consulta l'[articolo del Centro assistenza](#)

2

Il codice (o link di registrazione) è anche in bacheca:

xz6gckm

# Calendario delle lezioni

Controllare il calendario per potenziali cambi di programma. Se siete iscritti a Google Classroom del corso riceverete una notifica.

22-23) Stream Classwork People

View your work Google Calendar Class Drive folder

Calendario delle lezioni Posted 10:48 AM

Materiale didattico

Today March 2024 Week

MON	TUE	WED	THU	FRI
4	5	6	7	8
7 AM				
8 AM				
9 AM				
10 AM				
11 AM				
12 PM	[Teaching] [AE] Architettura degli Elaboratori 12 – 2pm			
1 PM				
2 PM				
3 PM				
4 PM				

[AE] Architettura degli Elaboratori  
11am – 2pm  
Tullio Levi Civita edificio Castelnuovo

- Tutte le lezioni saranno a Tullio Levi Civita edificio Castelnuovo (aula 3)
- Break a metà lezione?

# Regole di ingaggio

---

- Per questioni tecniche su **corso, esami, domande tecniche** si può chiedere: a lezione, ricevimento ed email
  - Non ho capito salto condizionato
  - Che differenza c'è fra CISC e RISC?
  - Accedere cartella esercizi
  - Non riesco a registrarmi a un appello
  - Varie ed eventuali
- Email as last resort!
  - Subject esplicativo
  - Chi siete, corso + matricola
  - Cosa avete già provato per risolvere il problema
- Spazio aperto per imparare e discutere
  - Ognuno di noi impara diversamente e ha un background diverso
  - Ogni domanda è ben accetta, supportiamoci e aiutiamoci
  - Bullismo e discriminazione non sono tollerati

# Introduzione al corso

## Libro di testo (preferenziale)



## Libro di testo (alternativo)



non il risc V

# Introduzione al corso

## Libro di testo



## Argomenti del corso

Introduzione storica e struttura di una CPU

L'assembler MIPS

Progetto della CPU MIPS ad un colpo di clock

Introduzione alla pipeline

Progetto della CPU MIPS con pipeline

Gestione degli hazard ed eccezioni

Gerarchia di memoria e cache

Parallelismo con cache

Memoria virtuale e protezione

### Esame:

Prova pratica su programmazione assembly

Prova scritta

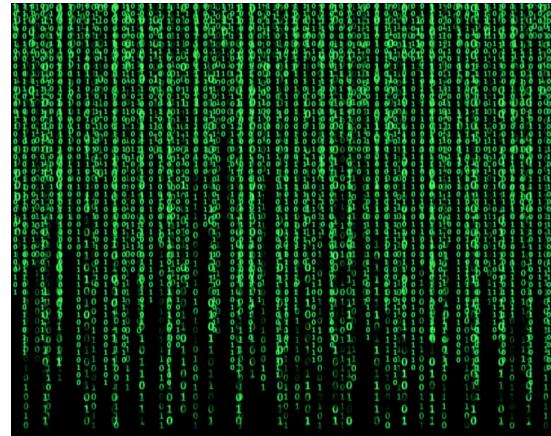
Il superamento della prova pratica abilita la registrazione della votazione determinata dalla prova scritta.

# Presentazione introduttiva



SAPIENZA  
UNIVERSITÀ DI ROMA

# Motivazione sul corso



Sapete programmare con linguaggi ad alto livello come **Python** o **C**?

Questo corso vi permette di capire cosa succede «sotto il cofano» (linguaggio macchina). Ad esempio capire come nella macchina viene implementato un **singolo ciclo for**

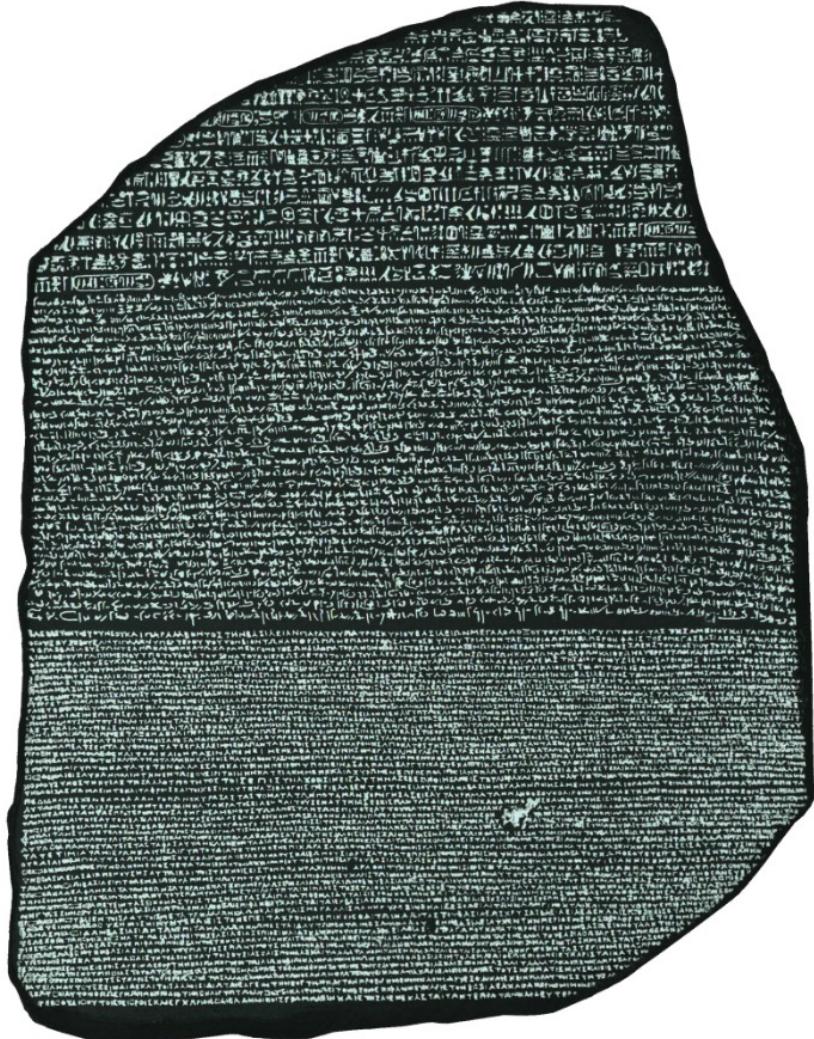


Infarinatura di **base ma solida** su come è costituita una CPU moderna (basata su architettura **RISC**). Molto comune in smartphone, tablet, ...

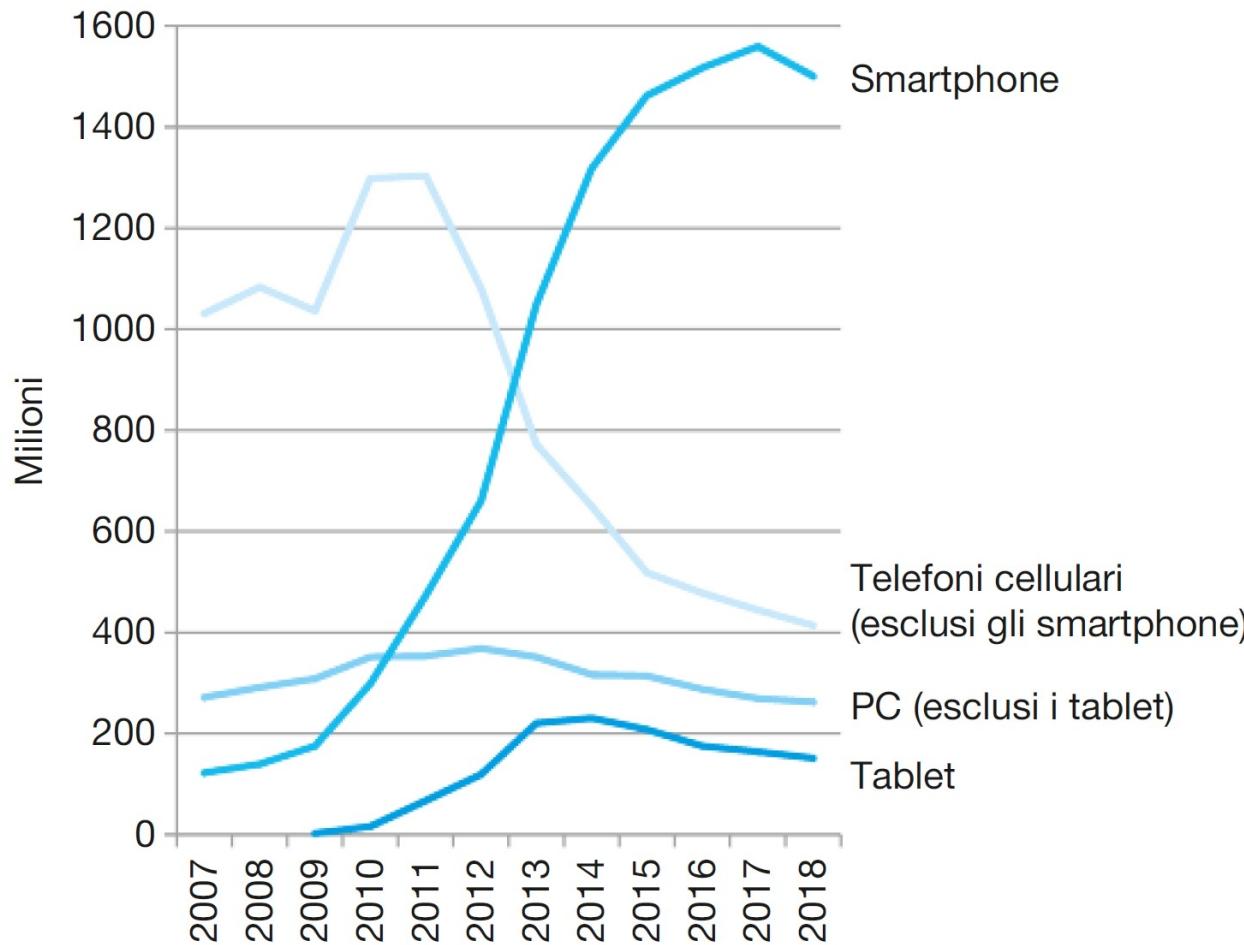
# Obiettivi

---

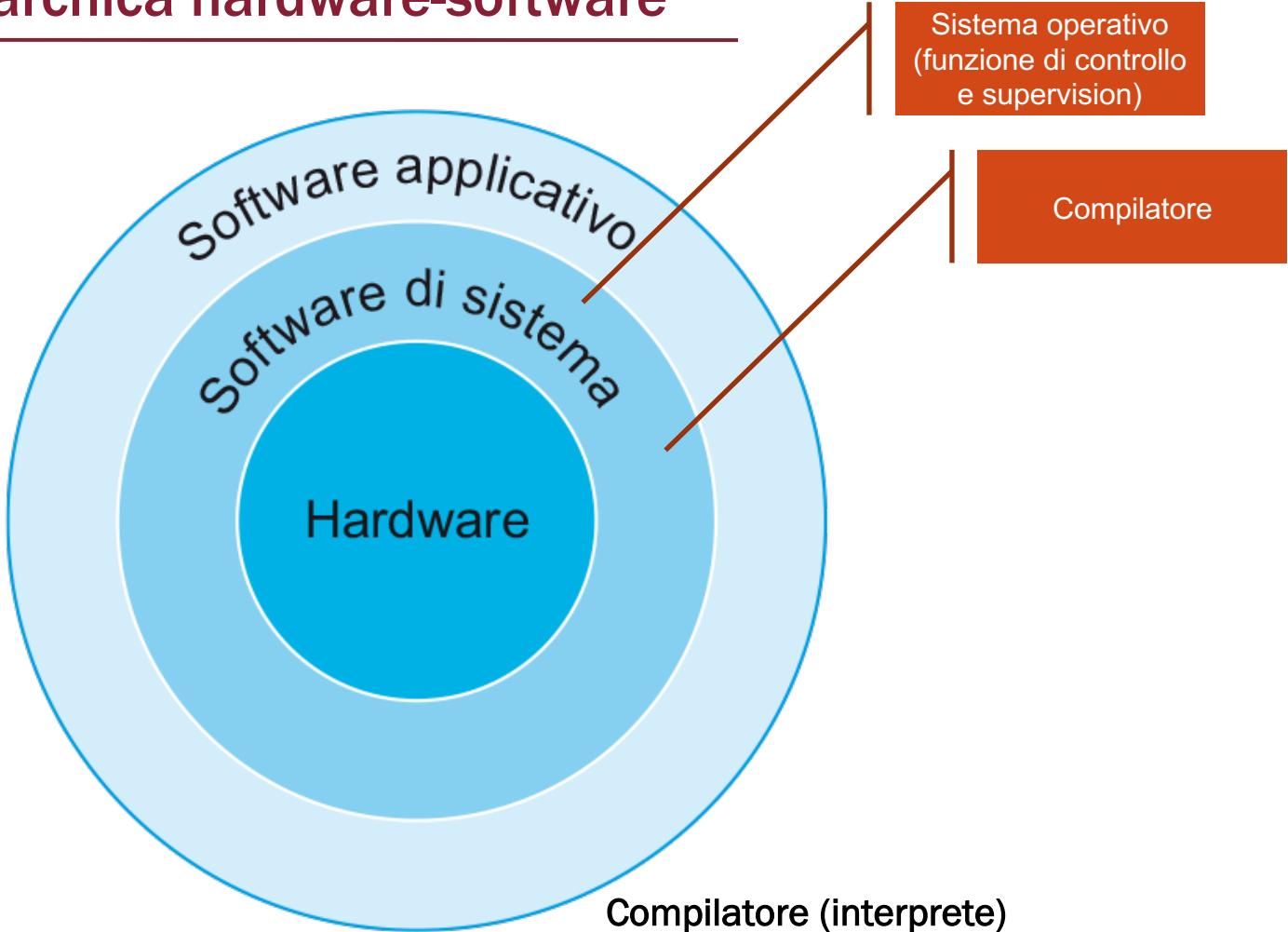
- Come tradurre da linguaggio ad alto livello a linguaggio del calcolatore?
- Qual è l'interfaccia tra hardware e software?
- Cosa determina le prestazioni di un programma e come migliorarle?
- Come passare dall'elaborazione sequenziale a quella parallela?
- Come migliorare l'efficienza energetica?
- Quali sono state le migliori idee dei progettisti?



## Era post-PC



# Relazione gerarchica hardware-software



## Sistema Operativo

- Gestire operazioni di input/output
- Allocare spazio nella memoria principale e nei dispositivi di memoria di massa
- Consentire a più applicazioni di usare hardware simultaneamente

## Compilatore (interprete)

- Traduce un programma da linguaggio di alto livello (C, python) in linguaggio macchina

# Compilazione ed assemblaggio

“Swap the element of array v at k-th position with the subsequent one”



Idea



Binary machine language program (for MIPS)	00000000101000100000000100011000 00000000100000100001000000100001 10001101111000100000000000000000000 100011100001001000000000000000000000 1010111000010010000000000000000000000 1010110111100010000000000000000000000 00000011111000000000000000000000000000
---	---

binary digits

# Compilazione ed assemblaggio

“Swap the element of array v at k-th position with the subsequent one”



Idea



Binary machine language program (for MIPS)	00000000101000100000000100011000 00000000100000100001000000100001 1000110111100010000000000000000000 10001110000100100000000000000000000 101011100001001000000000000000000000 101011011110001000000000000000000000 0000001111100000000000000000000000000
---	--

binary digits  
bits

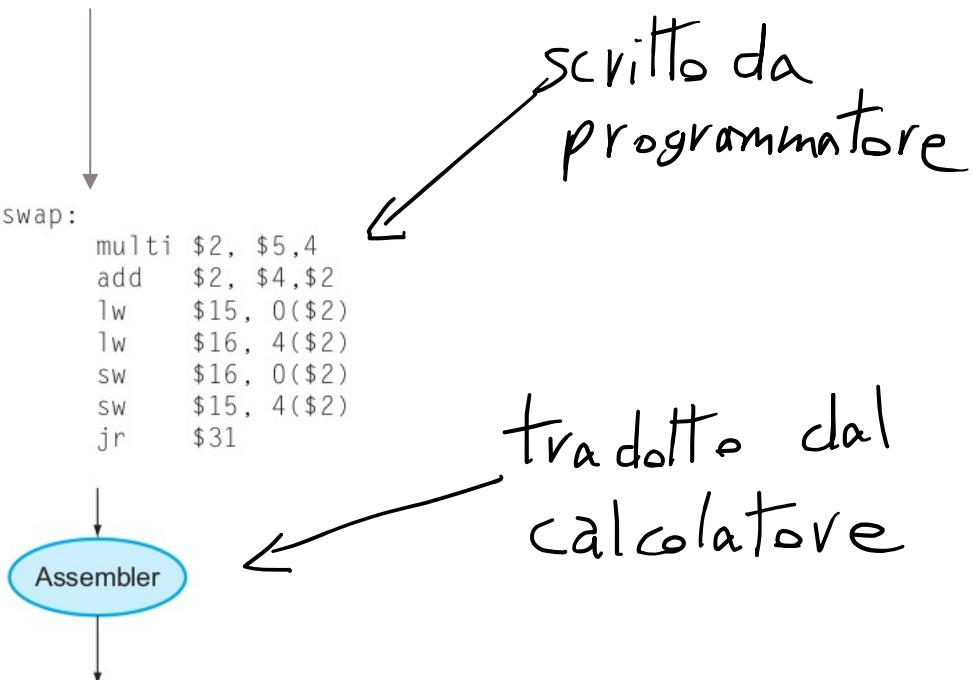
# Compilazione ed assemblaggio

“Swap the element of array v at k-th position with the subsequent one”

Idea: usare il  
calcolatore  
stesso per  
generare linguaggio  
macchina → assembler!

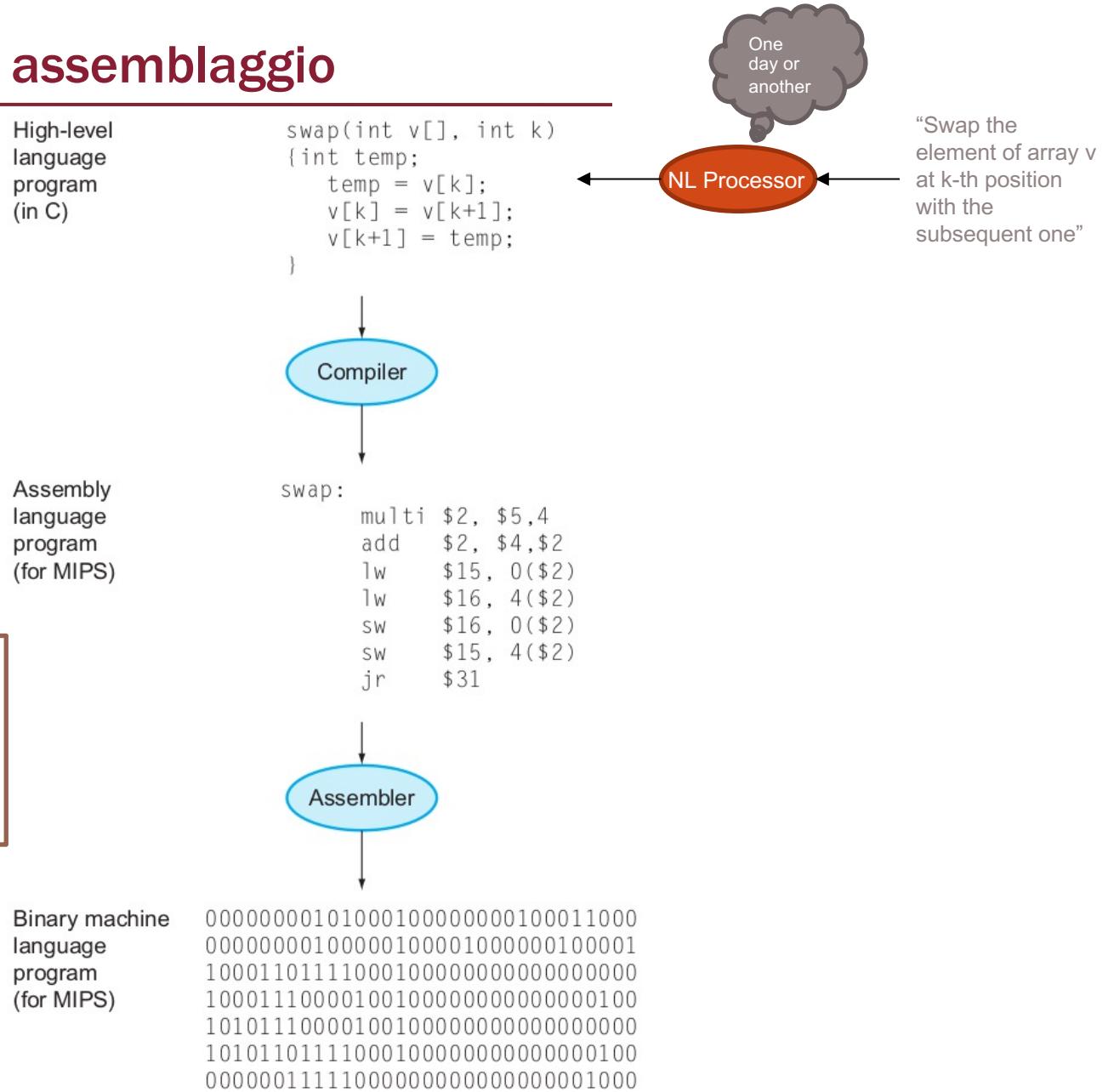
## Assembly language program (for MIPS)

```
swap:  
    multi $2, $5,4  
    add   $2, $4,$2  
    lw    $15, 0($2)  
    lw    $16, 4($2)  
    sw    $16, 0($2)  
    sw    $15, 4($2)  
    jr    $31
```



Binary machine language program (for MIPS)

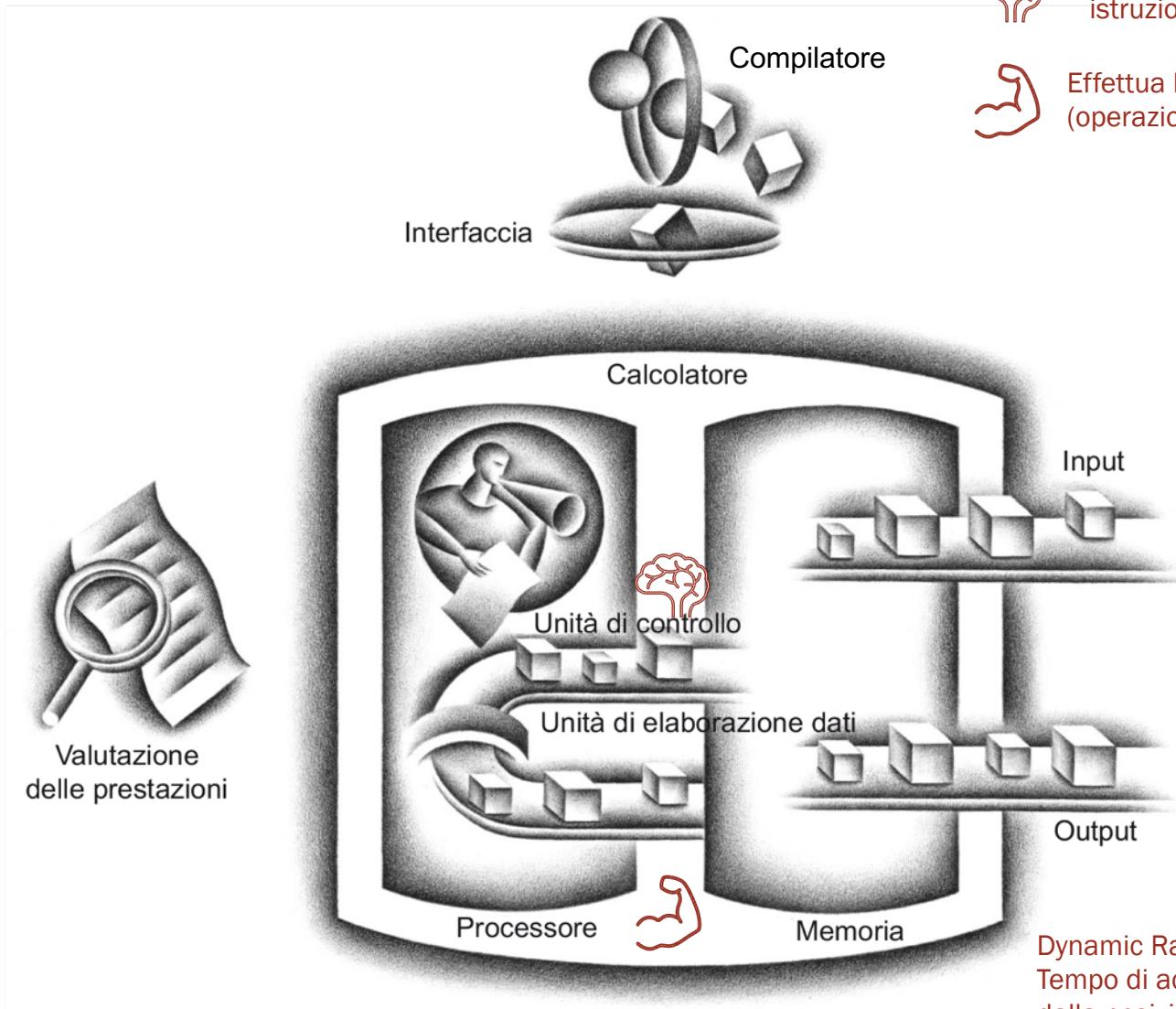
# Compilazione ed assemblaggio



## Vantaggi:

- Ragionamento più naturale
  - Maggiore produttività
  - Indipendenza dalla piattaforma

# L'architettura del calcolatore

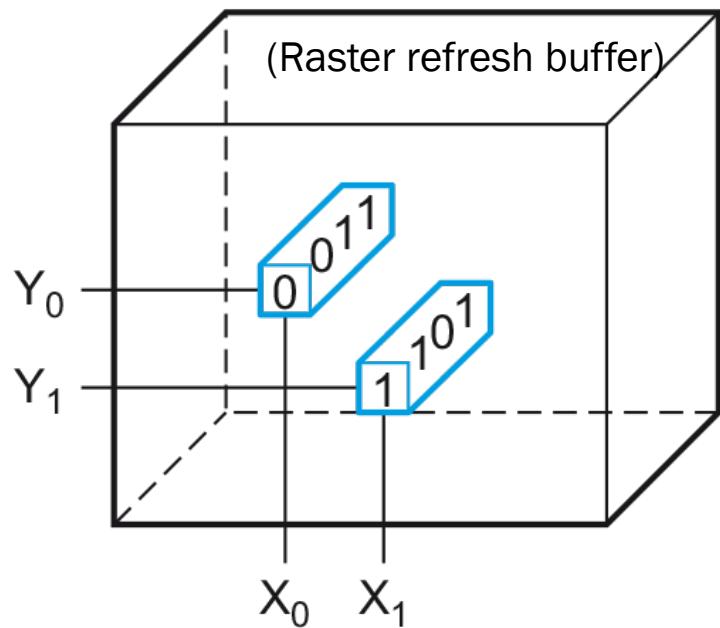


Dynamic Random Access Memory  
Tempo di accesso è indipendente  
dalla posizione

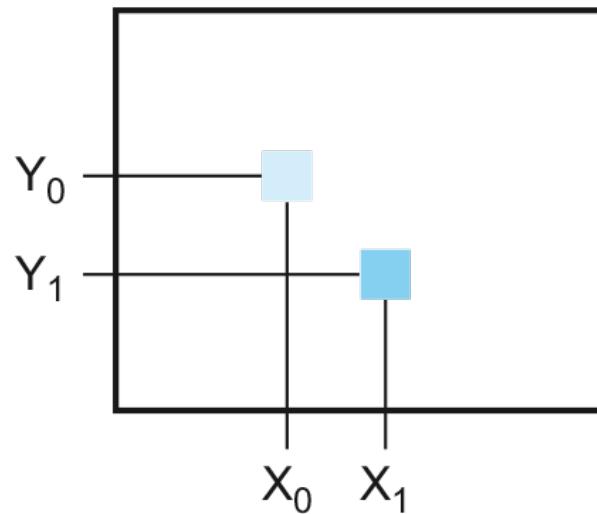


## Il mondo dei pixel

Frame buffer



Terminale video a tubo catodico



```
h2 {  
padding: 1rem 1.5rem;  
background-color: #060405;  
color: #DEE1FC;  
background: linear-gradient(to right, rgba(22, 22, 22, 0.0) 0%, rgba(22, 22, 22, 0.5) 75%, #060405 100%) #060405;  
margin-bottom: 2rem;  
margin-top: 2rem;  
}
```

# Un po' sulle immagini

---



Le immagini possono essere **Raster**

Griglia che quantizza i colori

Formati: jpeg, png, tiff

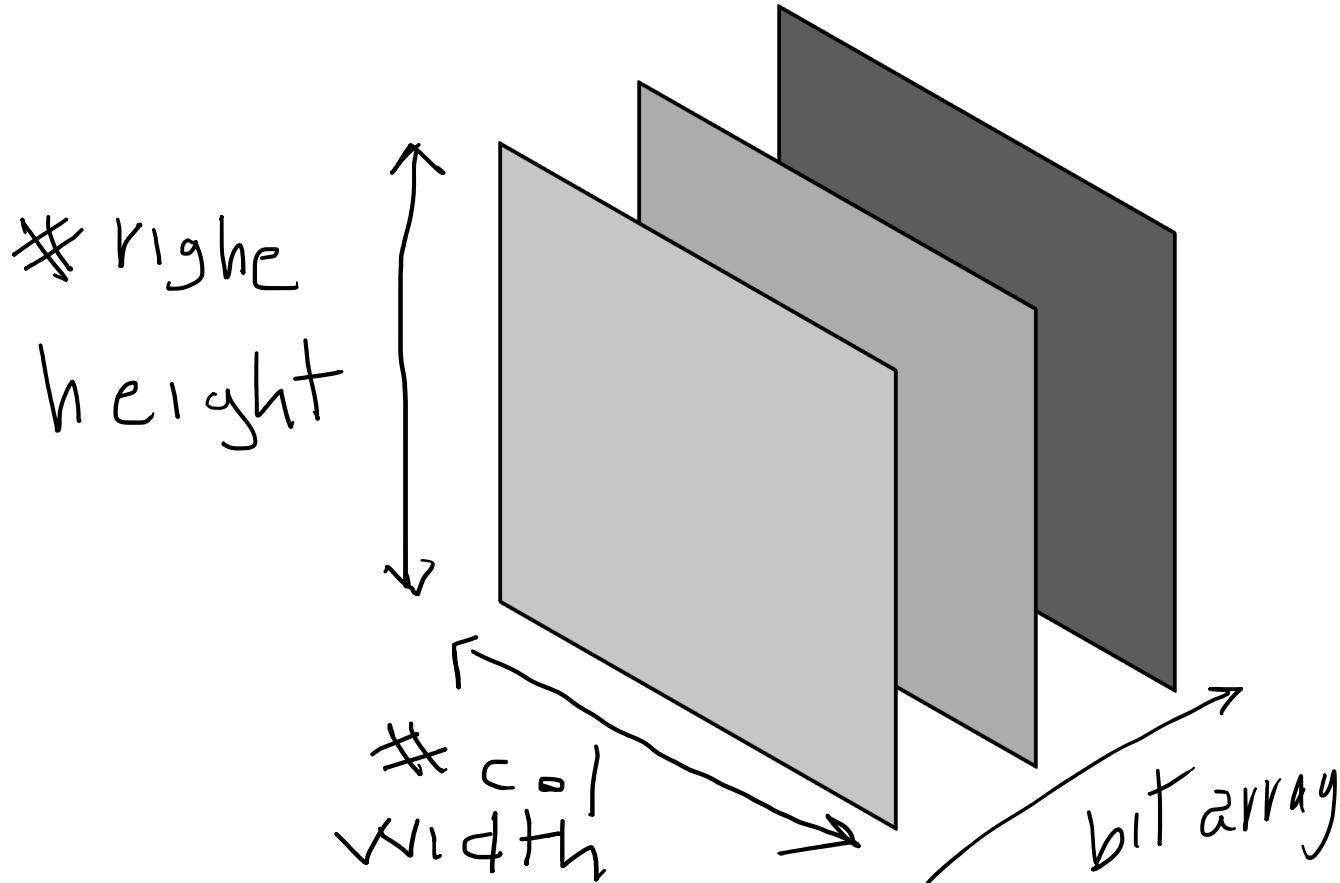
...oppure **vettoriali**

Curve e tracciati matematici che descrivono l'immagine

Formati: svg, eps, pdf

## Un po' sulle immagini (bitmap)

---



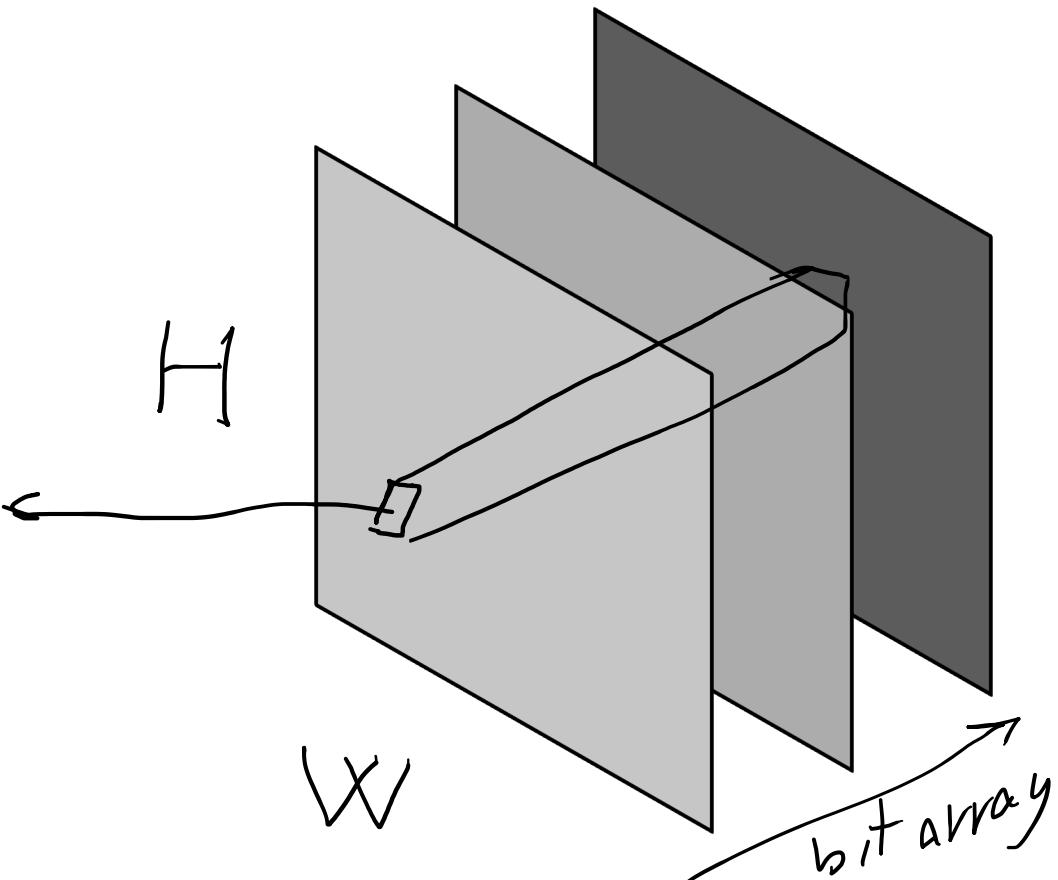
## Un po' sulle immagini (bitmap)

bit array  
2 1 0  
100

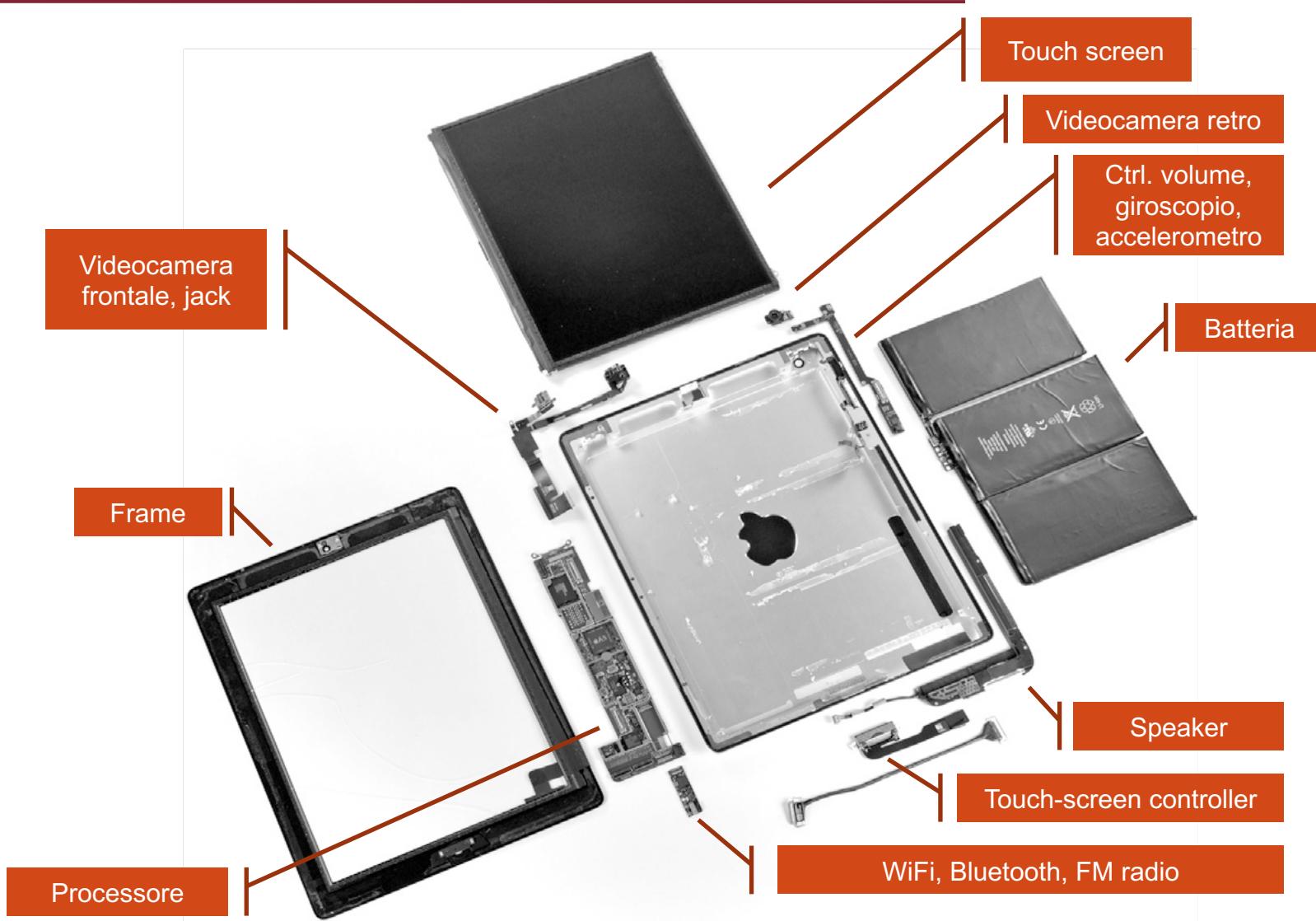
↓

color intensity "4"

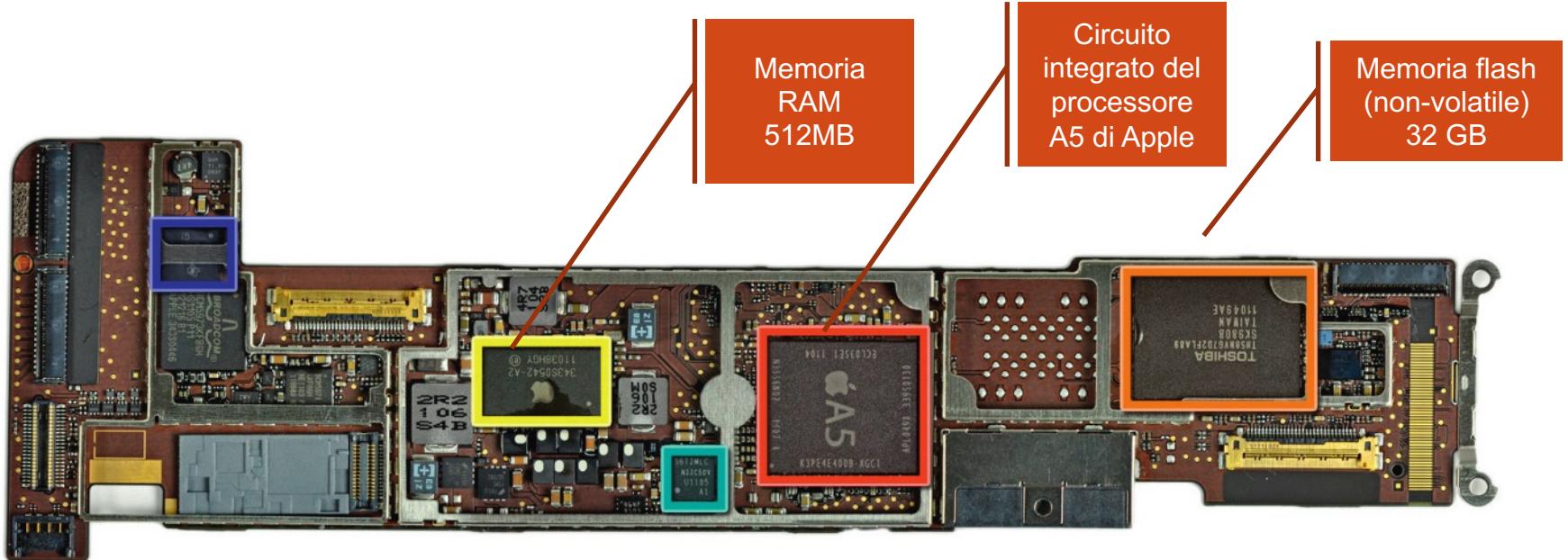
3 bit codifica  $2^3 = 8$  colori



# Dissezionando un iPad 2



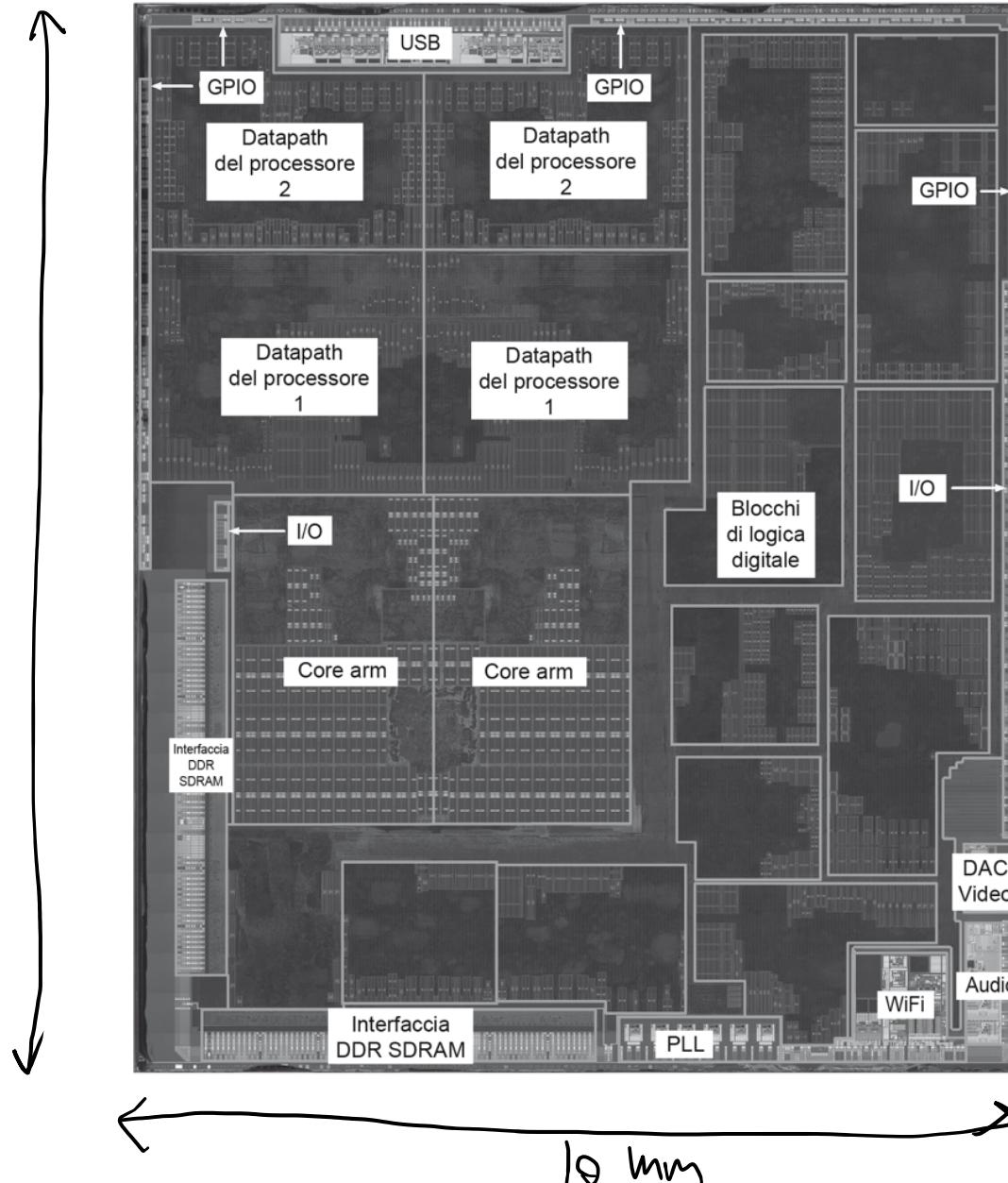
# La scheda del processore dell'iPad 2 (2011)



Apple A5: processore ARM (RISC)  
2 core, freq. di clock 1 GHz

# Circuito integrato del processore dell'iPad2

12 mm



Technology a

45 nm



nanometri

$10^{-9}$

# Transistor e legge [empirica] di Moore

Transistor: interruttore accesso/spento (0-1) controllato da un segnale elettrico

Anno	Tecnologia utilizzata nei calcolatori	Prestazioni relative per unità di costo
1951	Tubo a vuoto (valvola)	1
1965	Transitor	35
1975	Circuito integrato	900
1995	Circuito integrato a grandissima scala di integrazione	2 400 000
2020	Circuito integrato a scala di integrazione ultra-grande	500 000 000 000

**Figura 1.10 Prestazioni relative per unità di costo delle tecnologie via via utilizzate dai calcolatori nel tempo.** Fonte: Computer Museum, Boston. Il valore del 2020 è stato estrapolato dagli autori (vedi il paragrafo 1.13 ).

**Legge di Moore:** Le risorse messe a disposizione dai circuiti integrati vengono duplicate ogni 18-24 mesi

# Crescita delle capacità di chip DRAM e prezzo

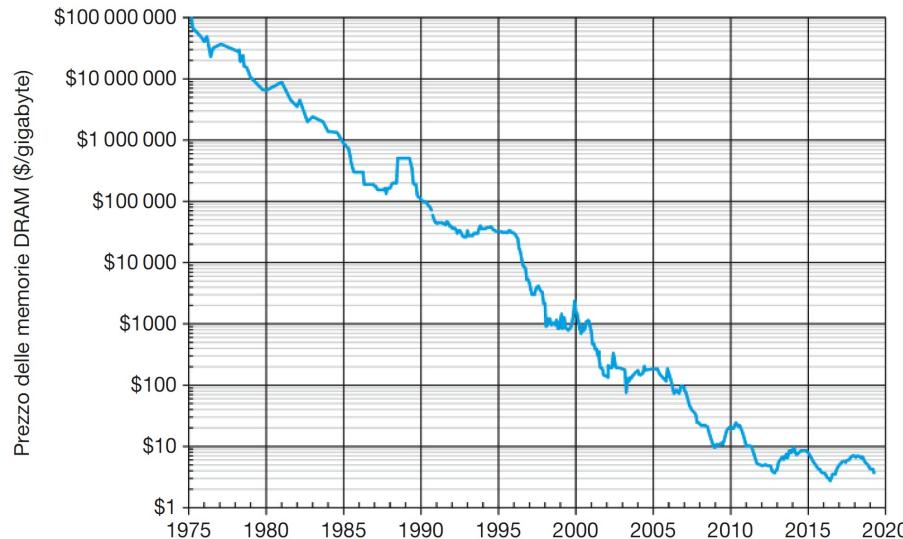
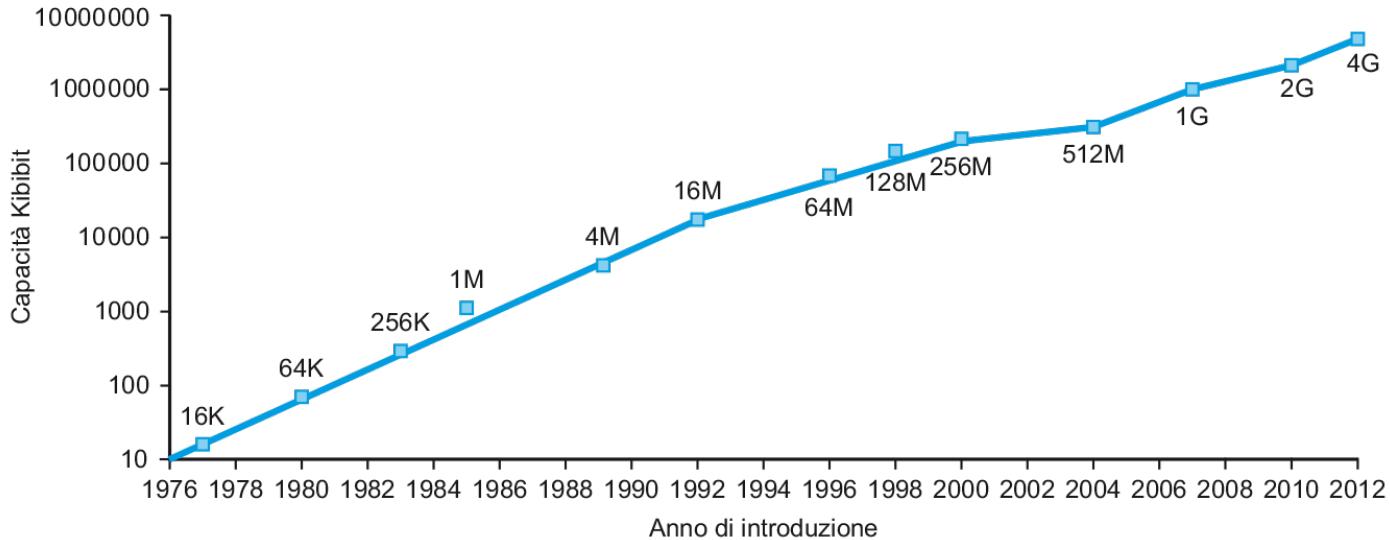


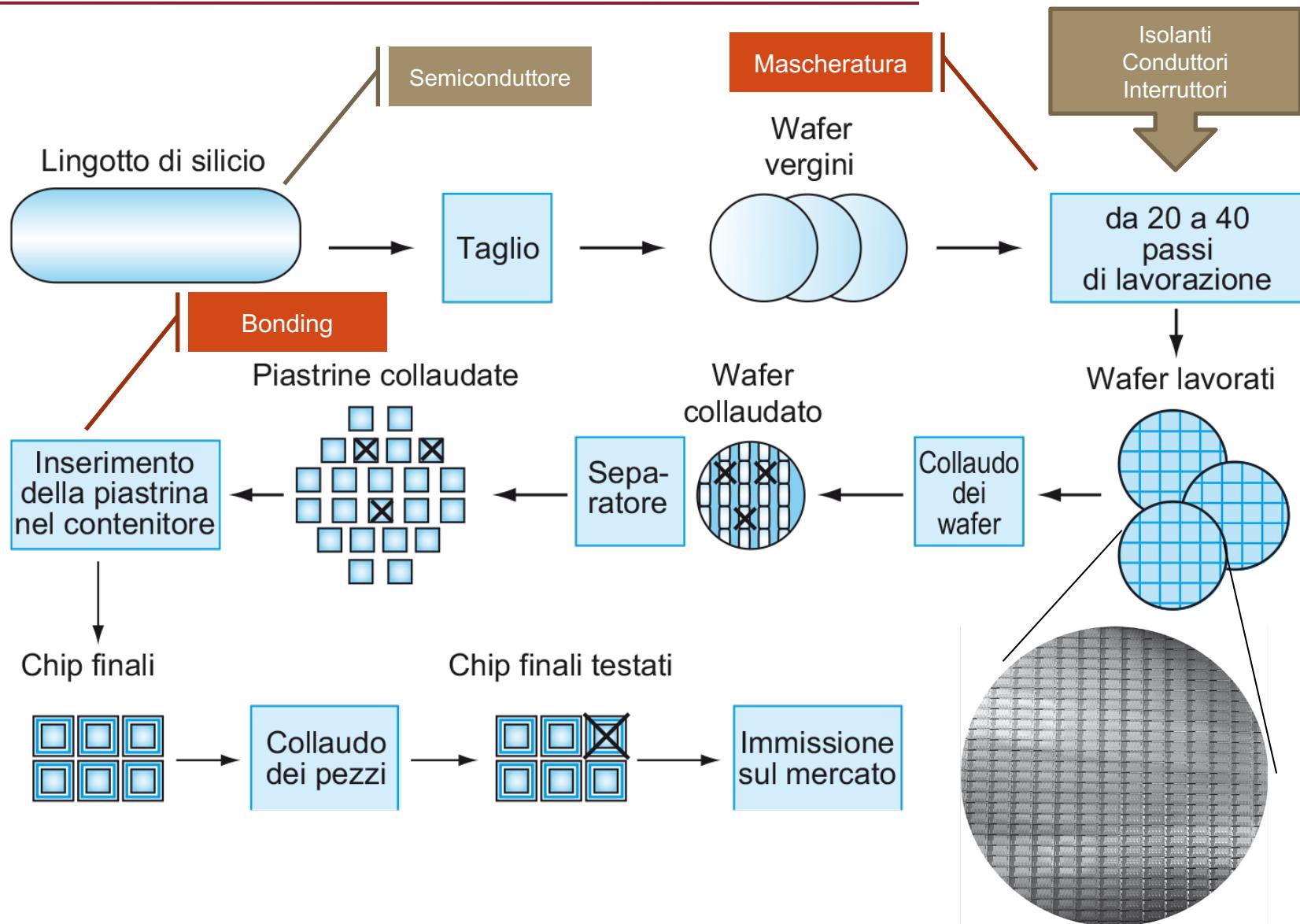
Figura 1.21 Prezzo delle memorie per gigabyte tra il 1975 e il 2020. (Fonte: <https://jcmi.net/memoryprice.htm>).

# Unità di misura

Termine decimale	Abbreviazione	Valore	Termine binario	Abbreviazione	Valore	% maggiore
Kilobyte	KB	$10^3$	Kibibyte	KiB	$2^{10}$	2%
Megabyte	MB	$10^6$	Mebibyte	MiB	$2^{20}$	5%
Gigabyte	GB	$10^9$	Gibibyte	GiB	$2^{30}$	7%
Terabyte	TB	$10^{12}$	Tebibyte	TiB	$2^{40}$	10%
Petabyte	PB	$10^{15}$	Pebibyte	PiB	$2^{50}$	13%
Exabyte	EB	$10^{18}$	Exbibyte	EiB	$2^{60}$	15%
Zettabyte	ZB	$10^{21}$	Zebibyte	ZiB	$2^{70}$	18%
Yottabyte	YB	$10^{24}$	Yobibyte	YiB	$2^{80}$	21%
Ronnabyte	RB	$1000^9$	Robibyte	RiB	$2^{90}$	24%
Queccabyte	QB	$1000^{10}$	Quebibyte	QiB	$2^{100}$	27%

**Figura 1.1** L'ambiguità nell'utilizzo della notazione  $2^x$  o  $10^x$  è stata risolta affiancando un'unità di misura binaria all'unità di misura decimale corrispondente per tutti i termini di uso comune. Nell'ultima colonna viene riportato, in percentuale, quanto l'unità di misura binaria è più grande dell'unità di misura decimale corrispondente. La differenza percentuale aumenta procedendo verso il basso. I prefissi valgono sia per i bit che per i byte, per cui 1 Gigabit rappresenta  $10^9$  bit, mentre 1 Gibibit rappresenta  $2^{30}$  bit. La società che gestisce il sistema metrico decimale ha creato i prefissi per le potenze di dieci. Gli ultimi due prefissi sono stati proposti nel 2019, in previsione della capacità globale dei sistemi di memoria per i dati. Tutti i nomi sono derivati dai termini che in latino indicano le potenze di 1000 che essi rappresentano.

# Genesi di una piastrina (die, comunemente detta chip)



# Prestazioni

---

Componente delle prestazioni	Unità di misura
Tempo di esecuzione della CPU per un dato programma	Secondi per programma
Numero di istruzioni	Istruzioni eseguite per singolo programma
Cicli di clock per istruzione (CPI)	Numero medio di cicli di clock per istruzione
Durata del ciclo di clock	Secondi per ciclo di clock

$$\text{Prestazioni}_X = \frac{1}{\text{Tempo di esecuzione}_X}$$

# Prestazioni

## Esercizio

$$\text{Tempo di CPU relativo a un programma} = \frac{\text{Cicli di clock della CPU relativi al programma}}{\text{Periodo del clock}}$$

$$\text{Tempo di CPU relativo a un programma} = \frac{\text{Cicli di clock della CPU relativi al programma}}{\text{Frequenza di clock}}$$

$$T = \frac{1}{\nu}$$

Il nostro programma preferito viene eseguito in 10 secondi dal calcolatore A, che è dotato di un clock a 2 GHz. Stiamo cercando di aiutare un progettista a costruire un calcolatore B in grado di eseguire lo stesso programma in 6 secondi. Il progettista ha concluso che è possibile aumentare in modo significativo la frequenza di clock; questa modifica avrà però un'influenza su tutto il progetto della CPU, facendo sì che il calcolatore B richieda un numero di cicli di clock maggiore di un fattore 1,2 rispetto al calcolatore A per eseguire il programma. Dovendo dare un consiglio al progettista, quale sarà la frequenza di clock che permette di raggiungere l'obiettivo?

# Prestazioni

$$\text{Tempo di CPU relativo a un programma} = \frac{\text{Cicli di clock della CPU relativi al programma}}{\text{Periodo del clock}}$$

$$\text{Tempo di CPU relativo a un programma} = \frac{\text{Cicli di clock della CPU relativi al programma}}{\text{Frequenza di clock}}$$

$$\text{Cicli di clock della CPU} = \frac{\text{Numero di istruzioni del programma}}{\text{Frequenza di clock}} \times \frac{\text{Numero medio di cicli clock per istruzione}}{\text{CPI}}$$

$$\text{Tempo di CPU} = \frac{\text{Numero di istruzioni} \times \text{CPI}}{\text{Frequenza di clock}}$$

$$T = \frac{1}{v}$$

CPI



# Prestazioni

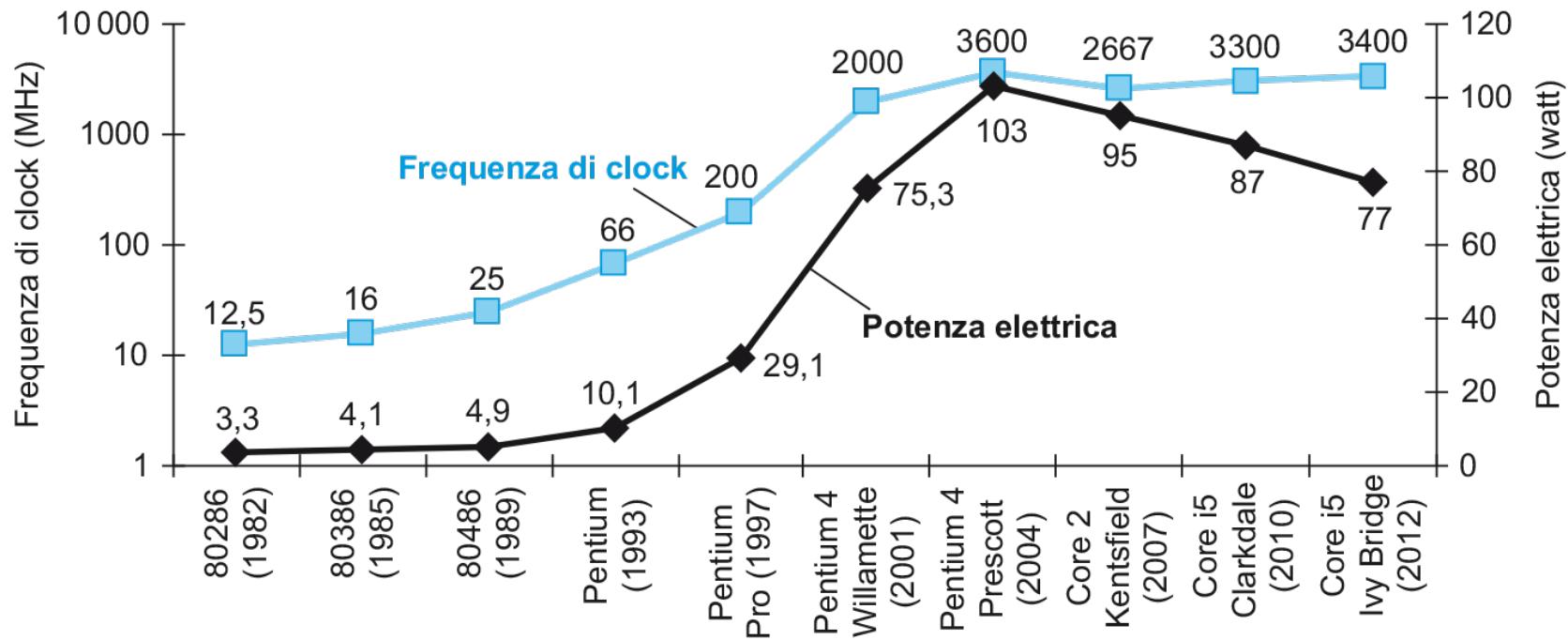
## Esercizio

	CPI per ciascun tipo di istruzione		
	A	B	C
CPI	1	2	3

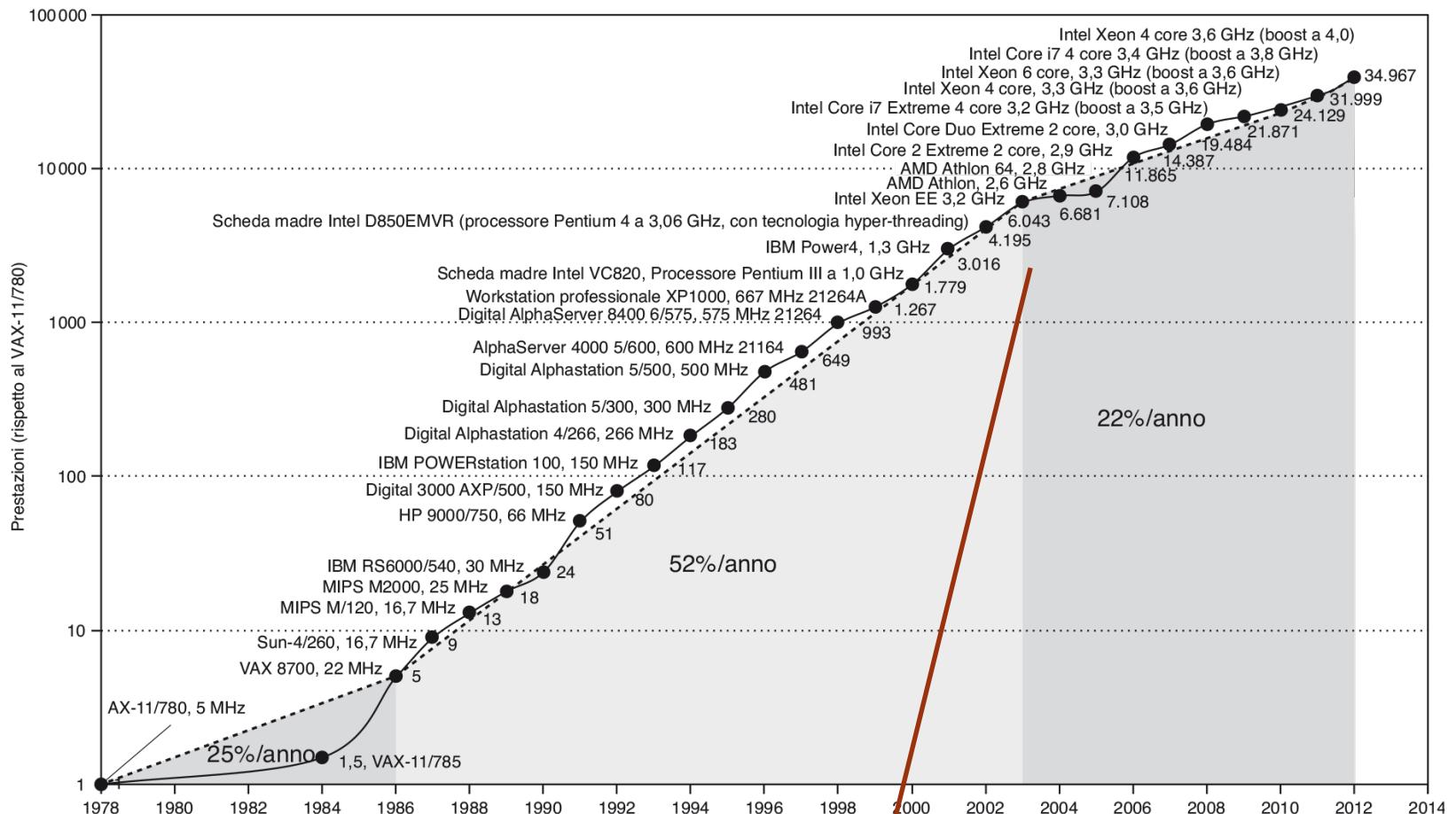
Sequenza di istruzioni	Numero di istruzioni in linguaggio macchina per ciascun tipo di istruzione		
	A	B	C
Sequenza 1	2	1	2
Sequenza 2	4	1	1

Quale sequenza richiede l'esecuzione di un maggior numero di istruzioni?  
Quale verrà eseguita più velocemente? Qual è il CPI delle due sequenze?

# La barriera dell'energia



# Trend delle prestazioni nel tempo



Limite sulla potenza assorbita, parallelismo implicito delle istruzioni, latenza memoria

# Benchmark (System Performance Evaluation Cooperative)

Descrizione	Nome	Numero di istruzioni $\times 10^9$	CPI	Periodo di clock (secondi $\times 10^{-9}$ )	Tempo di esecuzione (secondi)	Tempo di riferimento (secondi)	SPECratio
Interprete Perl	perlbench	2684	0,42	0,556	627	1774	2,83
Compilatore GNU C	gcc	2322	0,67	0,556	863	3976	4,61
Ottimizzazione combinatoria	mcf	1786	1,22	0,556	1215	4721	3,89
Libreria di simulazione di eventi discreti	omnetpp	1107	0,82	0,556	507	1630	3,21
Conversione da XML a HTML via XSLT	xalancbmk	1314	0,75	0,556	549	1417	3,21
Compressione video	x264	4488	0,32	0,556	813	1763	2,17
Intelligenza artificiale: ricerca su albero alpha-beta (scacchi)	deepsjeng	2216	0,57	0,556	698	1432	2,05
Intelligenza artificiale: ricerca ad albero Monte Carlo (Go)	leela	2236	0,79	0,556	987	1703	1,73
Intelligenza artificiale: generatore di soluzioni ricorsive (Sudoku)	exchange2	6683	0,46	0,556	1718	2939	1,71
Compressione generale di dati	xz	8533	1,32	0,556	6290	6182	0,98
Media geometrica							2,36

$$\sqrt[n]{\prod_{i=1}^n \text{Rapporto del tempo di esecuzione}_i}$$


## Benchmark: assorbimento medio di potenza

% Carico di lavoro	Prestazioni (ssj_ops)	Potenza media (watt)
100%	4 864 136	347
90%	4 389 196	312
80%	3 905 724	278
70%	3 418 737	241
60%	2 925 811	212
50%	2 439 017	183
40%	1 951 394	160
30%	1 461 411	141
20%	974 045	128
10%	485 973	115
0%	0	48
Somma totale	26 815 444	2165
$\Sigma ssj\_ops / \Sigma potenza =$		12 385

**Figura 1.19** Il programma ssj2008 del banchmark SPECpower eseguito su uno Xeon Platinum 8276L con due processori di Intel, 2,2 GHz, doppio connettore, con 192 GB di DRAM e un disco SSD da 80 GB.

# Errori di progettazione e Trabocchetti

---

*Trabocchetto: ci si aspetta che il miglioramento di uno dei componenti di un calcolatore produca un aumento delle prestazioni proporzionale alla dimensione del miglioramento.*

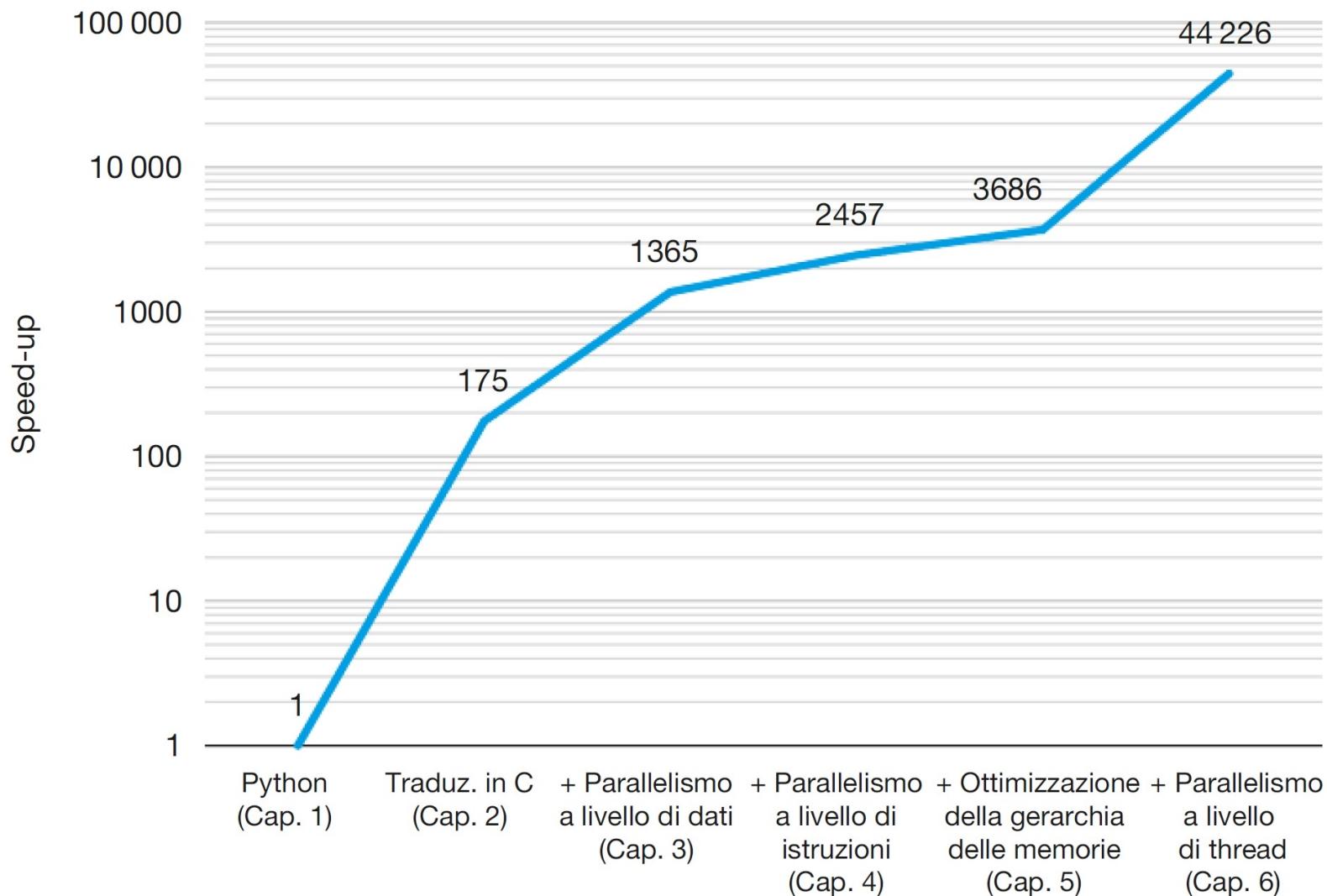


**Falso!** Aumento la velocità della istruzione di moltiplicazione del 10%, allora otterrò un miglioramento dell'hardware complessivo del 10% **Falso!**



La grande idea di **rendere veloce la situazione più comune**, ha un corollario che tormenta i progettisti dell'hardware e del software e che ci ricorda che l'impatto del miglioramento di una funzionalità dipende dal tempo per il quale quella funzionalità verrà utilizzata.

# Ottimizzazione del codice e dell'infrastruttura



## Legge di Amdahl

---

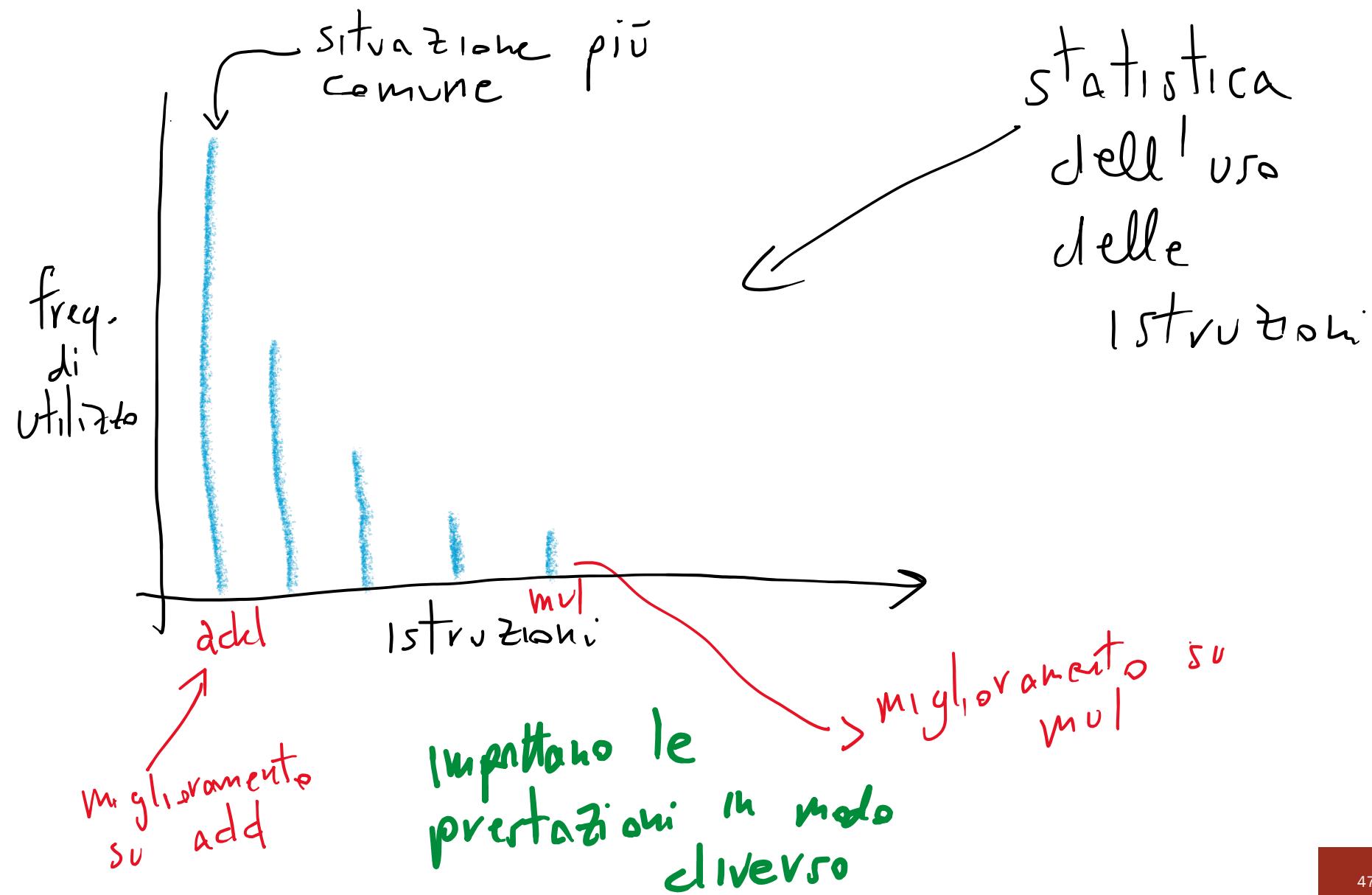
Tempo di esecuzione dopo il miglioramento =

$$= \frac{\text{Tempo di esecuzione influenzato dal miglioramento}}{\text{Miglioramento}} + \text{Tempo di esecuzione non influenzato dal miglioramento}$$

Possiamo utilizzare la Legge di Amdahl per stimare il miglioramento delle prestazioni quando conosciamo il tempo in cui viene utilizzata una certa funzionalità e il suo potenziale incremento di velocità. La Legge di Amdahl, insieme all'equazione delle prestazioni della CPU, è uno strumento facile per valutare potenziali miglioramenti; essa verrà analizzata più nel dettaglio negli esercizi.

Una delle linee guida nella progettazione dell'hardware è descritta da un corollario della Legge di Amdahl: *rendere veloce la situazione più comune*. Essa

## Rendere veloci le situazioni più comuni



## Cenni storici



SAPIENZA  
UNIVERSITÀ DI ROMA

Special thanks and credits:  
Andrea Sterbini

# Generazione “-100”

---

I primi esempi di macchina **meccanica**

150-100 AC:

macchina di **Antikythera** per predire le eclissi



1200 DC: Automi di Al-Jazari

Meccanismi “programmabili” a canne e pistoni  
(p.es. barca con banda musicale)



Medio Evo: Automi umanoidi

il “Moro” giocatore di scacchi (in realtà conteneva una persona  
di piccola statura)

Lo scrivano, il disegnatore ed il musicista di Drosz

Il cavaliere meccanico in armatura (Leonardo da Vinci)

... molti altri



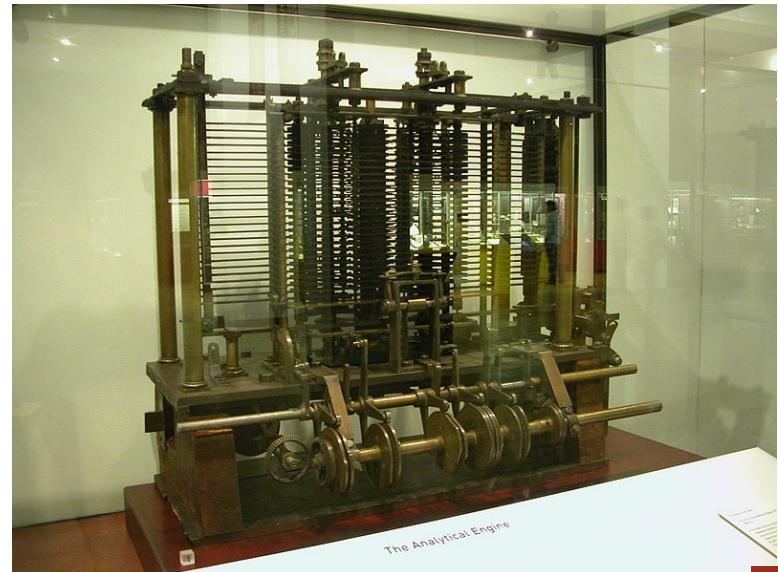
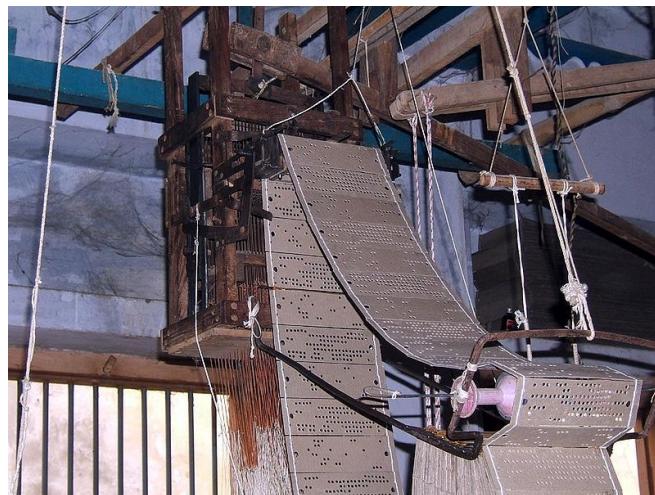
# GENERAZIONE “0”

1642: Blaise Pascal: la Pascalina  
calcolatrice meccanica



1833: Charles Babbage  
Analytical Engine (Ada Lovelace)

1839: Telaio automatico Jacquard  
(«programmabile» a schede perforate!)



# Generazione 1

I primi computer digitali

**1937 ABC (US)**

Atanasoff-Berry Computer

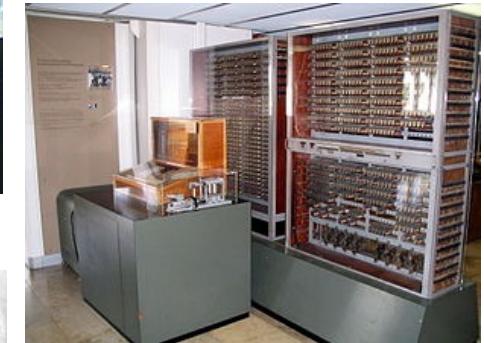
Risolve sistemi di equazioni lineari



**1941 Z3 (Konrad Zuse)**

2200 relè (elettromeccanico)

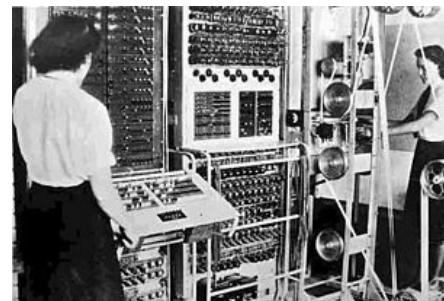
Programmato con nastro perforato



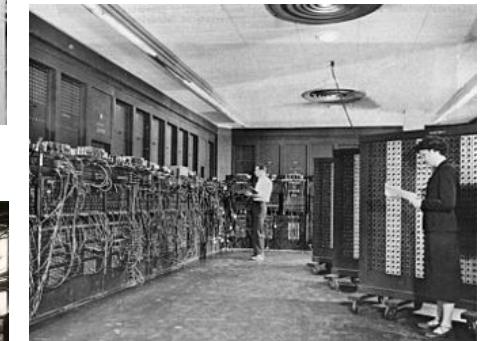
**1944 Colossus (UK)**

2400 Valvole

Cablato (programmato collegando tra loro le parti a seconda del compito da svolgere)

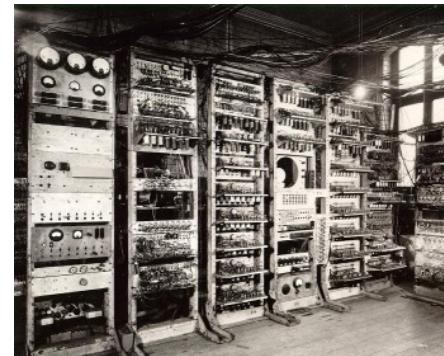


Per decodificare i messaggi nazisti, leggenda narra (ma Turing usò il Bombe, non Colossus)



**1946 ENIAC (US)**

18000 Valvole, Cablato, General purpose



**1948 Mark 1 (UK)**

Il primo computer a programma memorizzato

Programmato in Autocode

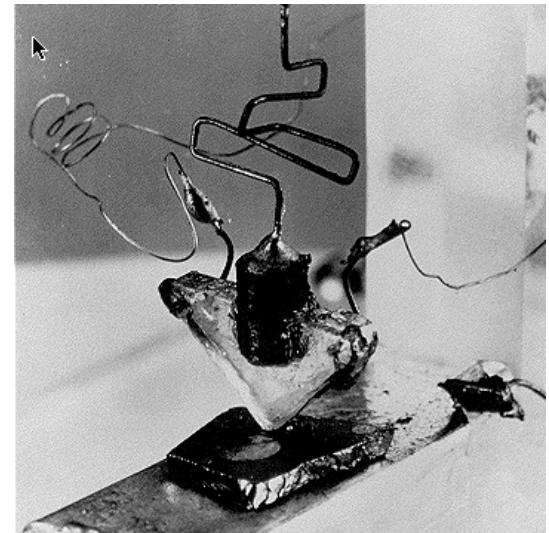
# Generazione 2

---

1947: Introduzione dei transistor

Piccoli, robusti, veloci, consumano poco  
(le valvole si rompevano o bruciavano)

ENIAC bruciava un paio di valvole al giorno  
(meno se lasciato sempre acceso)



NCR, RCA, IBM: i primi computer commerciali

ALU complesse

Trasferimento Diretto in Memoria (DMA)

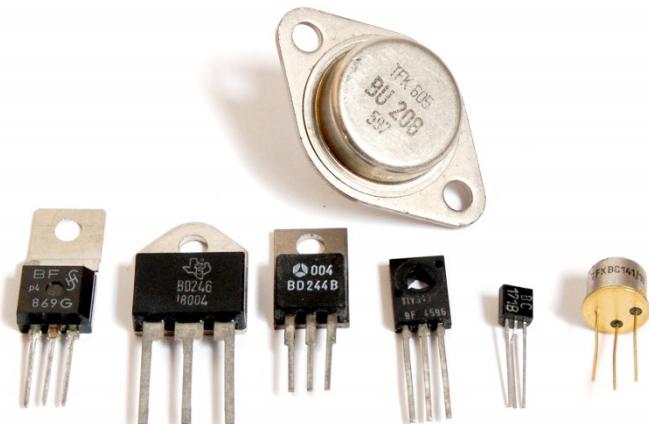
Linguaggi di alto livello

FORTRAN: calcoli numerici

ALGOL: simulazioni (a oggetti! Ispirazione per il futuro)

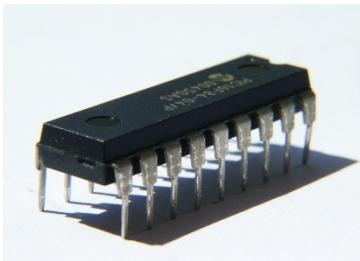
COBOL: calcoli finanziari

DEC sviluppa i primi Minicomputer



# Generazione 3

1958: Introduzione dei circuiti integrati



1964:

IBM serie 360: compatibili “verso l'alto”

- Stesse istruzioni (+ istr. avanzate)
- Stesso Sistema Op. (+ funz. avanzate)
- Velocità crescente
- Memoria crescente
- Parallelismo crescente
- Molto costoso (100K\$)

	da →	→ a
RAM	64 Kbyte	512 Kbyte
CLOCK	1 MHz	5 MHz
Banda CPU/MEM	0,5 MB/s	16 MB/s

DEC PDP-8: poco costoso (16000\$ → 2500\$)

Bus di interconnessione **standardizzato**

Facilità di espansione (produttori OEM)



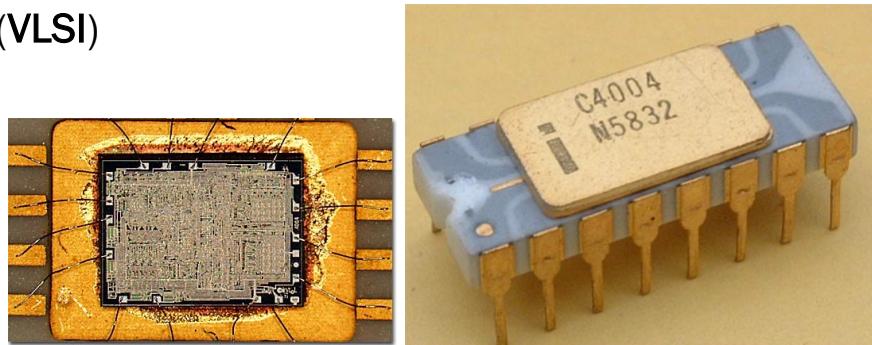
# Generazione 4

Circuiti integrati su larga e larghissima scala (VLSI)

1971:

Intel 4004: 1° processore integrato (a 4 bit)

Memorie integrate

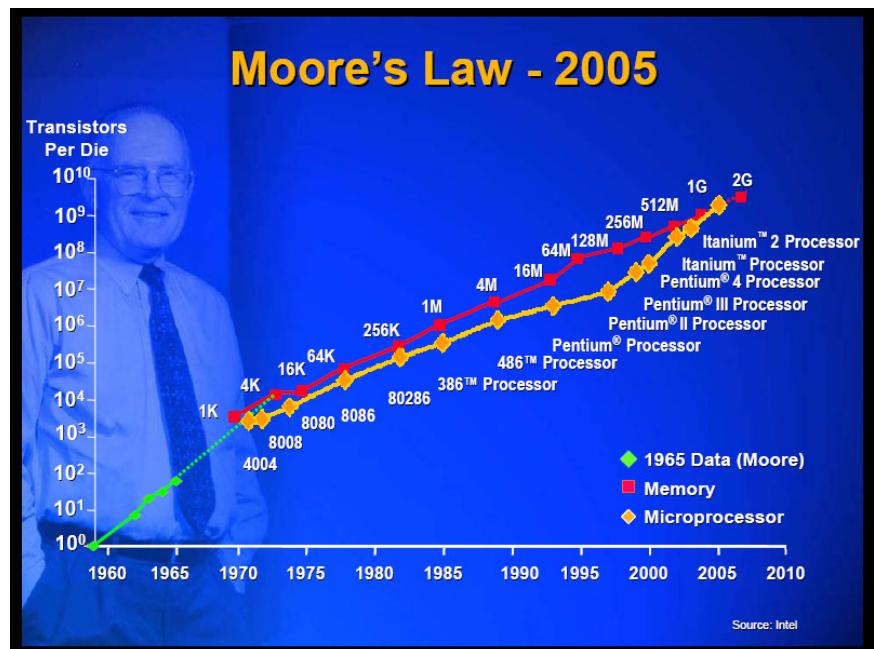


Legge (osservazione) di Moore:

“la densità di transistor raddoppia  
ogni 12-18 mesi”

Ne segue che:

- Il costo di produzione resta uguale
- Il prezzo per componente decresce
- La distanza tra componenti diminuisce
- La velocità aumenta
- L'energia ed il calore diminuiscono
- L'affidabilità aumenta



Source: Intel

# Introduzione alle Architetture

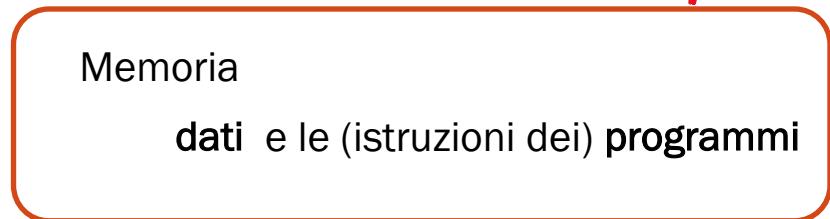


SAPIENZA  
UNIVERSITÀ DI ROMA

Special thanks and credits:  
Andrea Sterbini

# Architettura di Von Neumann a programma memorizzato

Un computer è diviso in:



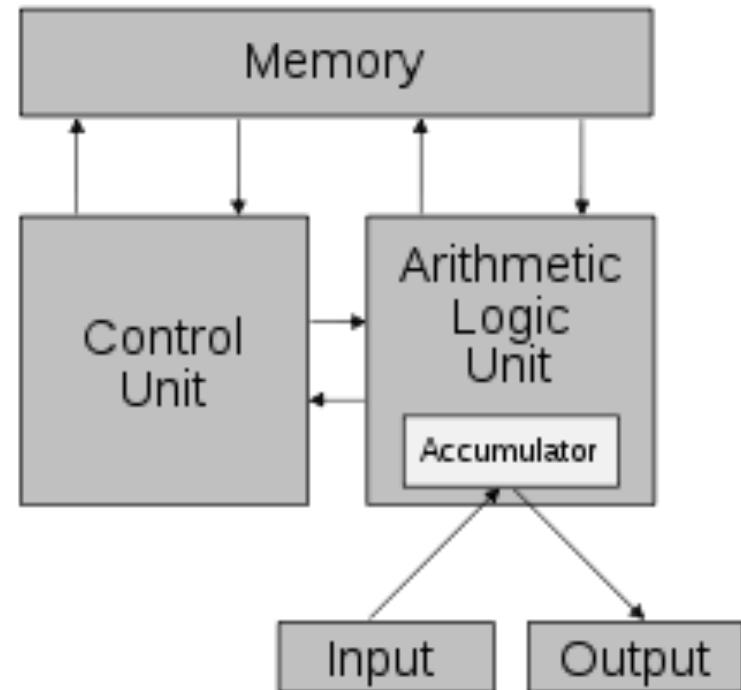
CPU (CU + ALU + registri)

Bus di comunicazione tra le diverse parti

Periferiche di I/O

(tastiera, schermo, stampante, scheda di rete eccetera)

concepto fondamentale



# La CPU

La Central Processing Unit è formata da:

La Unità di elaborazione Aritmetico/Logica (ALU)

fa solo calcoli

NON ha uno stato interno

I registri

A uso generale e speciale,

per manipolare istruzioni, indirizzi, dati, risultati

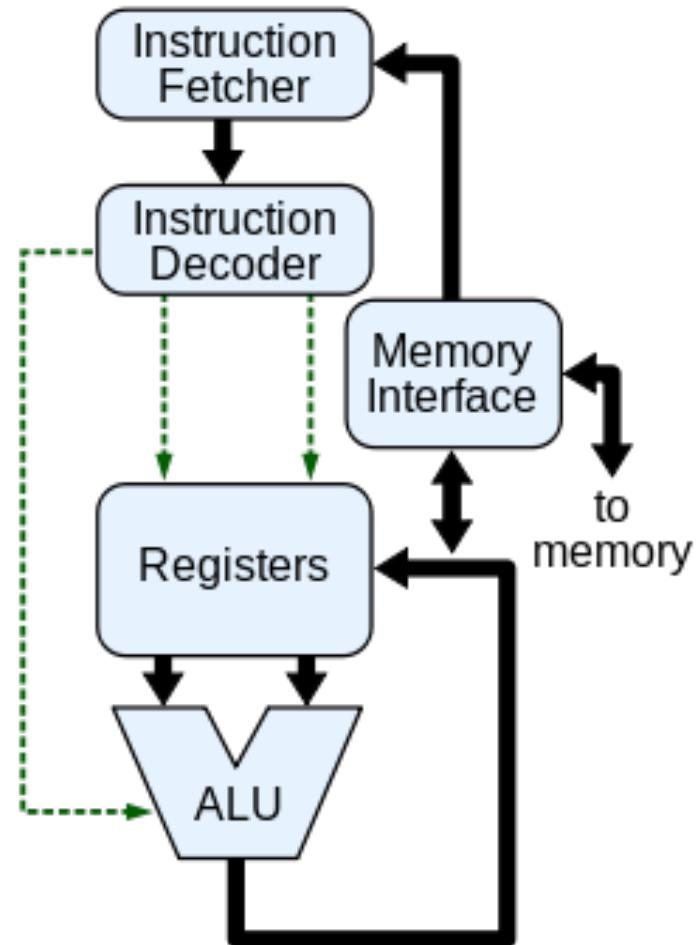
Il **bus** di comunicazione con la Memoria

Il **bus** di comunicazione con le periferiche

(non necessario col memory mapping)

La Unità di Controllo che coordina il tutto

(Instruction Decoder nella figura)



## Esempio: la IAS machine

1951 : IAS machine – Institute for Advanced Studies – Princeton



J. Robert Oppenheimer and John von Neumann in front of the IAS machine



Fonte wikipedia

# Esempio: la IAS machine

1951 : IAS machine – Institute for Advanced Studies – Princeton

Aveva una memoria di 1024 word da 40 bit ciascuna (ovvero 5 Kbyte totali!!!)

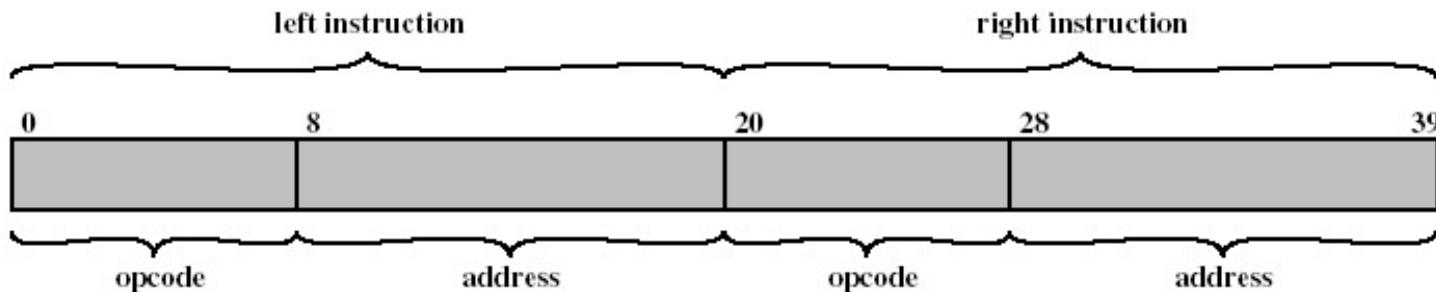
**dati:** solo interi nella rappresentazione in Complemento a 2

**istruzioni:** 2 istruzioni per ciascuna word

**indirizzi:** 12 bit per gli indirizzi (sarebbero bastati 10)



(a) Number word



(b) Instruction word

# La CPU della IAS machine

**MBR:** Memory Buffer Register

riceve/manda il dato dalla/alla memoria

**MAR:** Memory Address Register

indica l'indirizzo in memoria

**PC:** Program Counter

indica l'indirizzo dell'istr. da eseguire

**IR:** Instruction Register

riceve l'istruzione da eseguire

**IBR:** Instruction Buffer Register

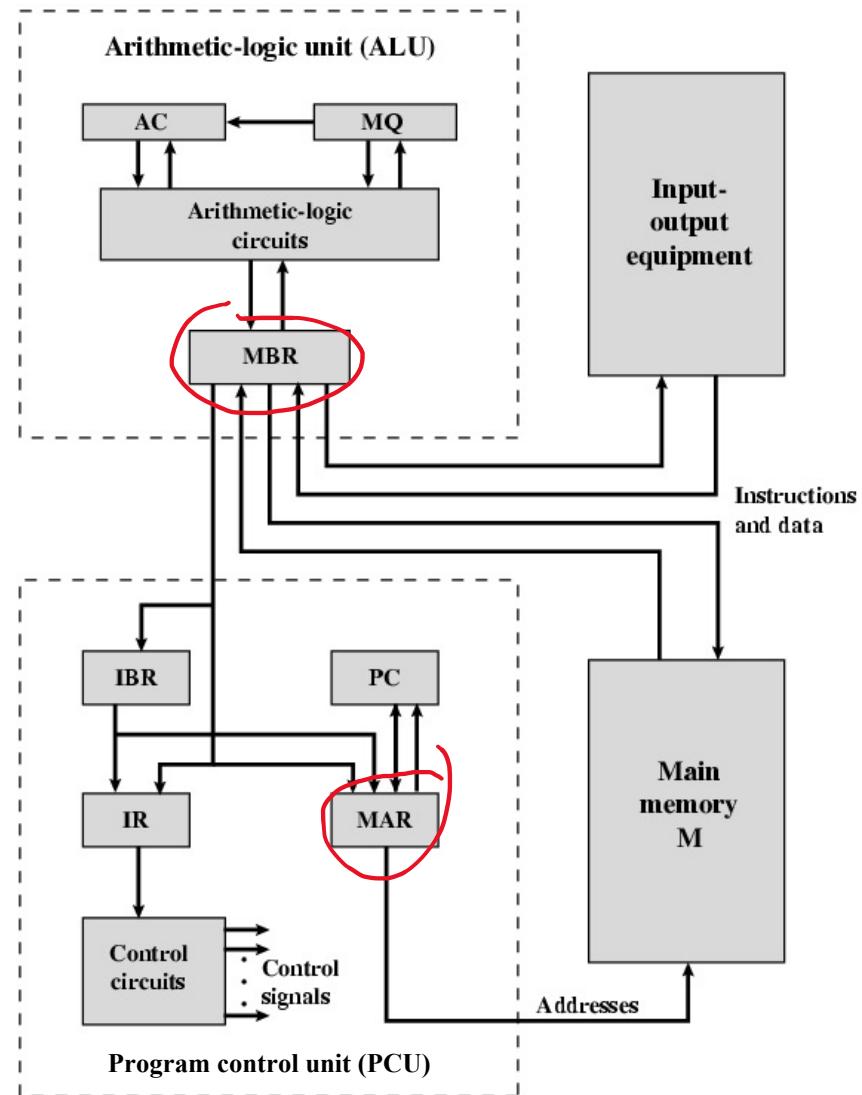
contiene la 2° istruzione della word

**AC:** Accumulatore

per i risultati parziali dei calcoli

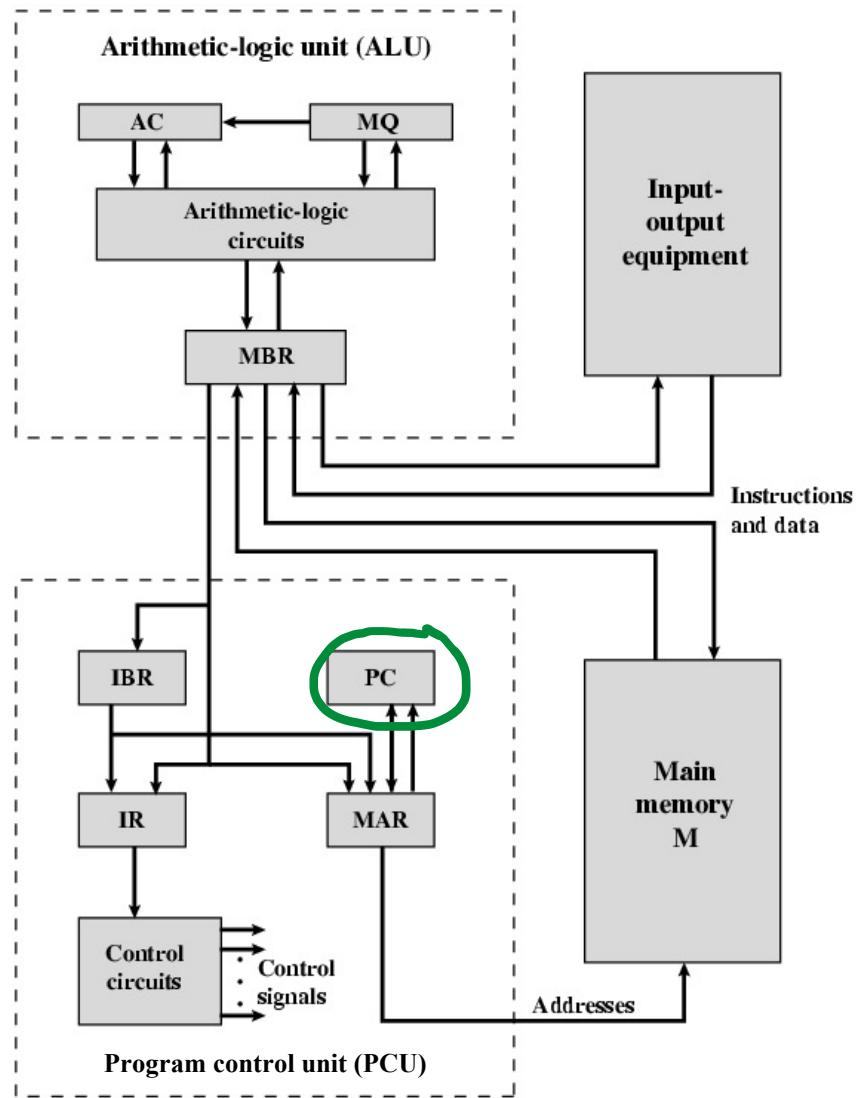
**MQ:** Multiplier Quotient

per i risultati parziali dei calcoli



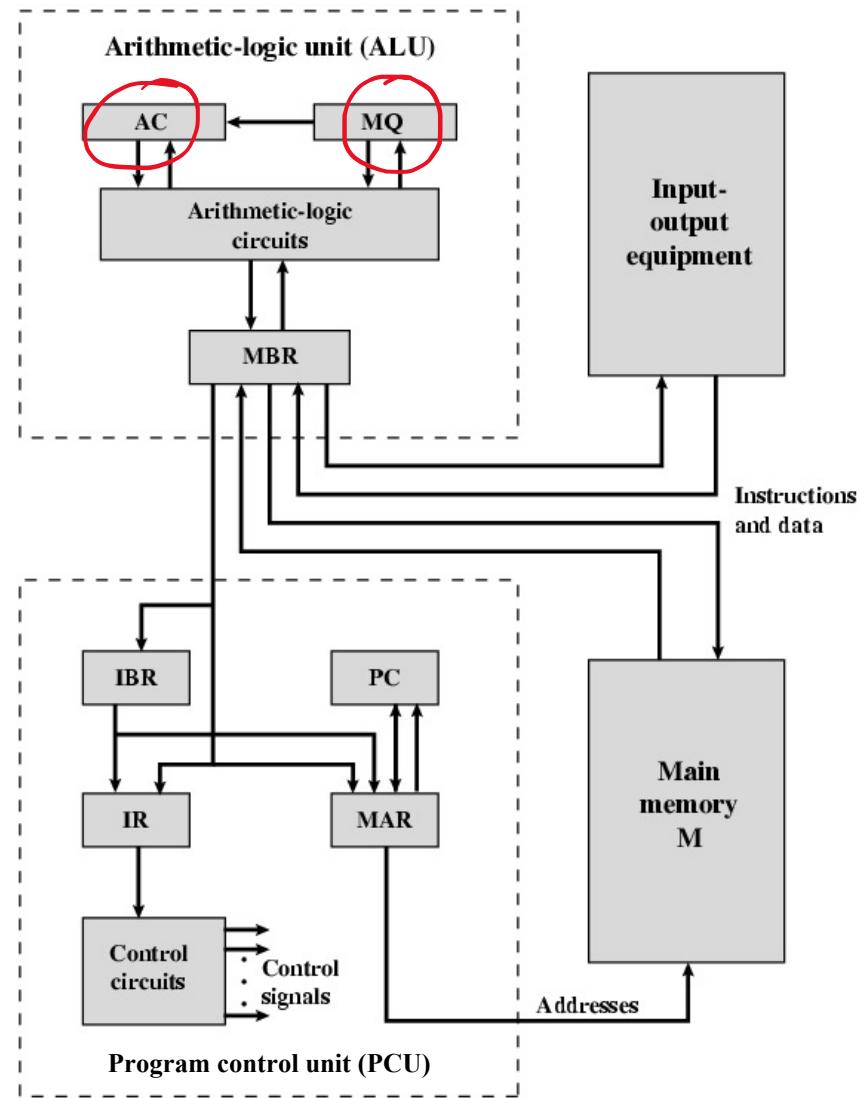
# La CPU della IAS machine

- MBR: Memory Buffer Register  
riceve/manda il dato dalla/alla memoria
- MAR: Memory Address Register  
indica l'indirizzo in memoria
- PC:** Program Counter  
indica l'indirizzo dell'istr. da eseguire
- IR: Instruction Register  
riceve l'istruzione da eseguire
- IBR: Instruction Buffer Register  
contiene la 2° istruzione della word
- AC: Accumulatore  
per i risultati parziali dei calcoli
- MQ: Multiplier Quotient  
per i risultati parziali dei calcoli



# La CPU della IAS machine

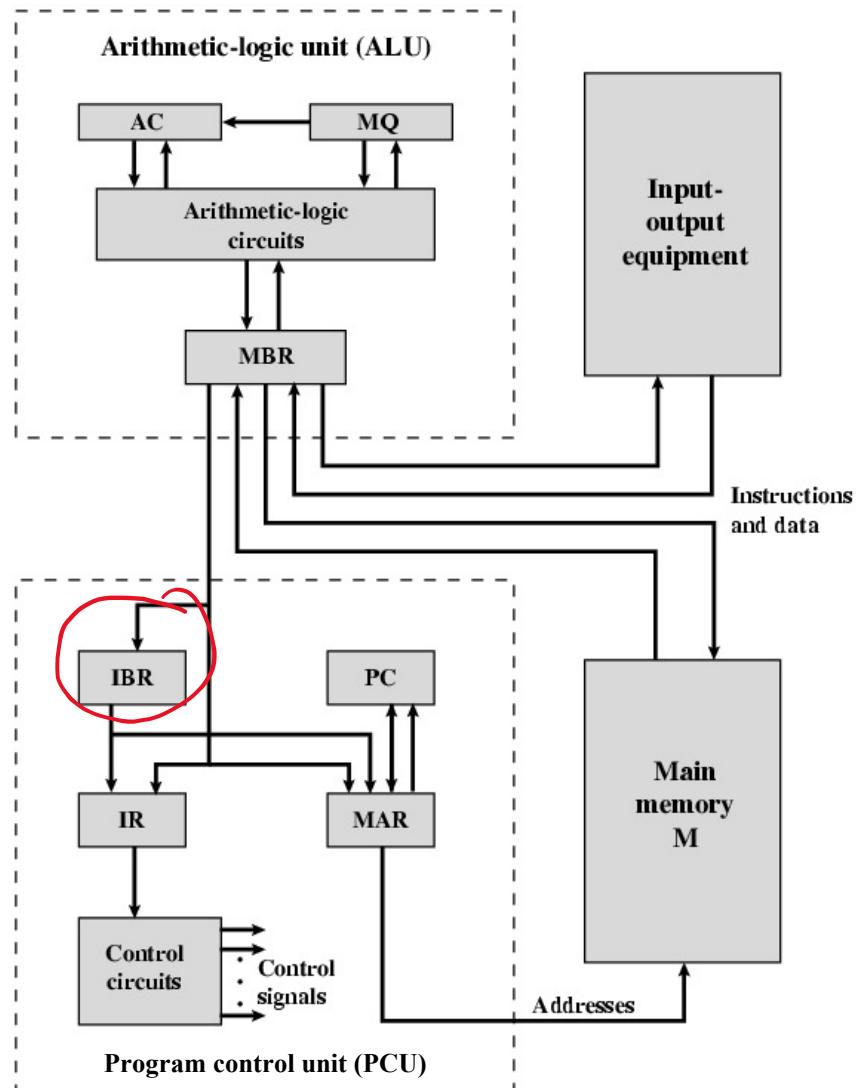
- MBR:** Memory Buffer Register  
riceve/manda il dato dalla/alla memoria
- MAR:** Memory Address Register  
indica l'indirizzo in memoria
- PC:** Program Counter  
indica l'indirizzo dell'istr. da eseguire
- IR:** Instruction Register  
riceve l'istruzione da eseguire
- IBR:** Instruction Buffer Register  
contiene la 2° istruzione della word
- AC:** Accumulatore  
per i risultati parziali dei calcoli
- MQ:** Multiplier Quotient  
per i risultati parziali dei calcoli



# La CPU della IAS machine

- MBR: Memory Buffer Register  
riceve/manda il dato dalla/alla memoria
- MAR: Memory Address Register  
indica l'indirizzo in memoria
- PC: Program Counter  
indica l'indirizzo dell'istr. da eseguire
- IR: Instruction Register  
riceve l'istruzione da eseguire
- IBR: Instruction Buffer Register  
contiene la 2° istruzione della word
- AC: Accumulatore  
per i risultati parziali dei calcoli
- MQ: Multiplier Quotient  
per i risultati parziali dei calcoli

1h IAS  
Una Word  $\Rightarrow$  2 istruzioni



# Istruzioni di trasferimento

---

LOAD caricamento dalla memoria con operazioni semplici (ABS, NEG, ADD, SUB)  
con oppure senza il valore precedente di AC

**$AC \leftarrow AC <\text{operazione}> \text{Memory(Address)}$**

**$AC \leftarrow <\text{operazione}> \text{Memory(Address)}$**

LDMQ caricamento nel registro MQ

**$MQ \leftarrow \text{Memory(Address)}$**

ST memorizzazione di AC come dato

**$\text{Memory(Address)} \leftarrow AC$**

AMODL memorizzazione di AC come indirizzo in una istruzione (parte bassa della word)

**$\text{Memory(Address)}[\text{bits } 0:11] \leftarrow AC[\text{bits } 0:11]$**

AMODH memorizzazione di AC come indirizzo in una istruzione (parte alta della word)

**$\text{Memory(Address)} [\text{bits } 20:31] \leftarrow AC[\text{bits } 0:11]$**

# Istruzioni di salto

---

## Salti NON condizionati

UBL      salto incondizionato all'istruzione (parte bassa della parola)  
                **PC ← Address;    offsetPC ← 0**

UBH      salto incondizionato all'istruzione (parte alta della parola)  
                **PC ← Address;    offsetPC ← 1**

## Salti condizionati

CBL      salto condizionato all'istruzione (parte alta della parola)  
                **if AC >= 0 then PC ← Address;    offsetPC ← 0**

CBH      salto condizionato all'istruzione (parte alta della parola)  
                **if AC >= 0 then PC ← Address;    offsetPC ← 1**

# Operazioni aritmetiche (e I/O)

---

MUL prodotto

$$AC, MQ \leftarrow AC * \text{Mem(Address)}$$

DIV divisione e resto

$$AC \leftarrow AC / \text{Mem(Address)}; \quad MQ \leftarrow AC \% \text{Mem(Address)}$$

LSHIFT shift a sinistra (ovvero prodotto per  $2^X$ )

$$AC, MQ \leftarrow AC, MQ << X$$

RSHIFT shift a destra con estensione del segno (ovvero divisione per  $2^X$ )

$$AC, MQ \leftarrow AC, MQ >> X$$

MOVE spostamenti da MQ ad AC con le (8) operazioni semplici (ADD, SUB, ABS, NEG ...)

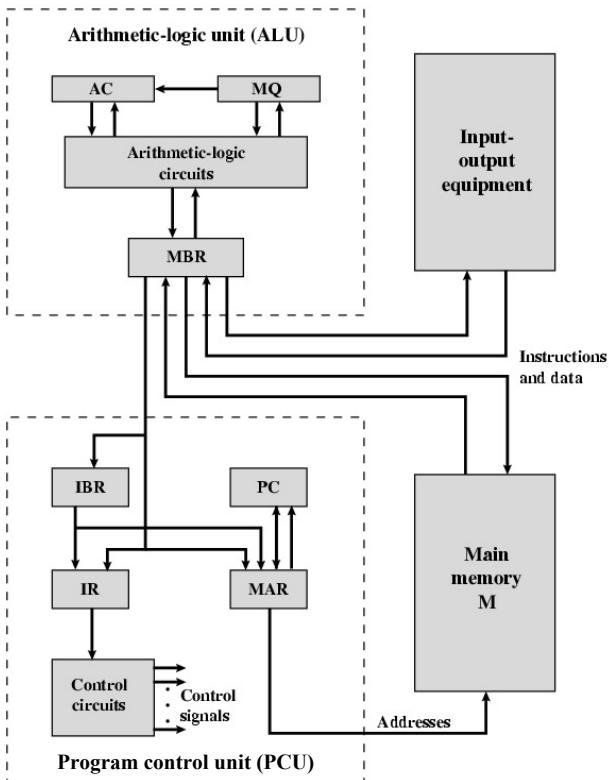
$$AC \leftarrow AC <\text{operazione}> MQ$$

IO trasferimento da e verso le periferiche

# Esempio: C=A+B

Se A, B e C sono le locazioni 101, 102, 103 in memoria, il codice per calcolare C=A+B è

LD	101	$AC \leftarrow Mem[101]$
ADD	102	$AC \leftarrow AC + Mem[102]$
SD	103	$Mem[103] \leftarrow AC$



## Esecuzione delle istruzioni

Fetch della istruzione dalla memoria

$$MAR \leftarrow PC \text{ (e } PCOffset=0\text{)}$$

$$IR, IBR \leftarrow MBR \leftarrow Mem[MAR]$$

Decodifica della istruzione (da IR)

$$MAR \leftarrow IR.Address; CU \leftarrow IR.Opcode$$

Sua esecuzione

$$AC \leftarrow MBR \leftarrow Mem[101]$$

$$PCOffset \leftarrow 1$$

Fetch dell'istr. succ. (da IBR)

$$MAR \leftarrow IBR.Address; CU \leftarrow IBR.Op$$

sua esecuzione

$$AC \leftarrow AC + MBR \leftarrow Mem[102]$$

Aggiornamento del PC

$$PC \leftarrow PC + 1$$

Fetch della istruzione successiva

$$MAR \leftarrow PC$$

$$IR, IBR \leftarrow MBR \leftarrow Mem[MAR]$$

Decodifica della istruzione (da IR)

$$MAR \leftarrow IR.Address; CU \leftarrow IR.Opcode$$

Sua esecuzione

$$Mem[103] \leftarrow MBR \leftarrow AC$$