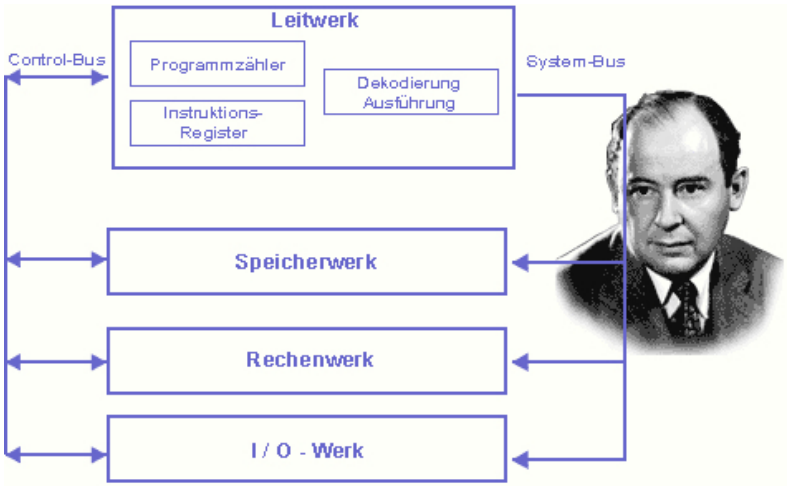
Cenni sull'Architettura degli Elaboratori

Eugenio G. Omodeo

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

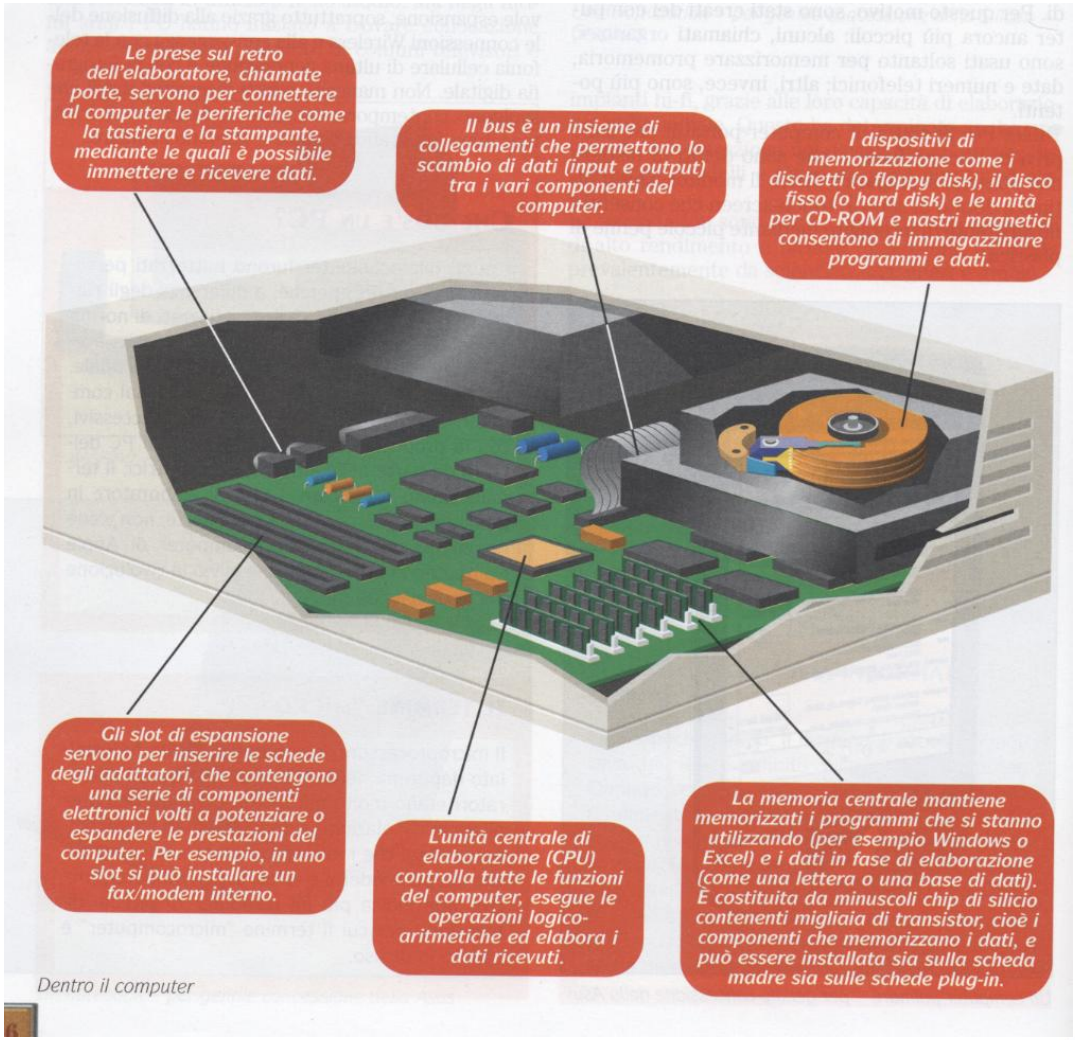
Dip. Matematica e Geoscienze — DMI



Trieste, 24/11/2021

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

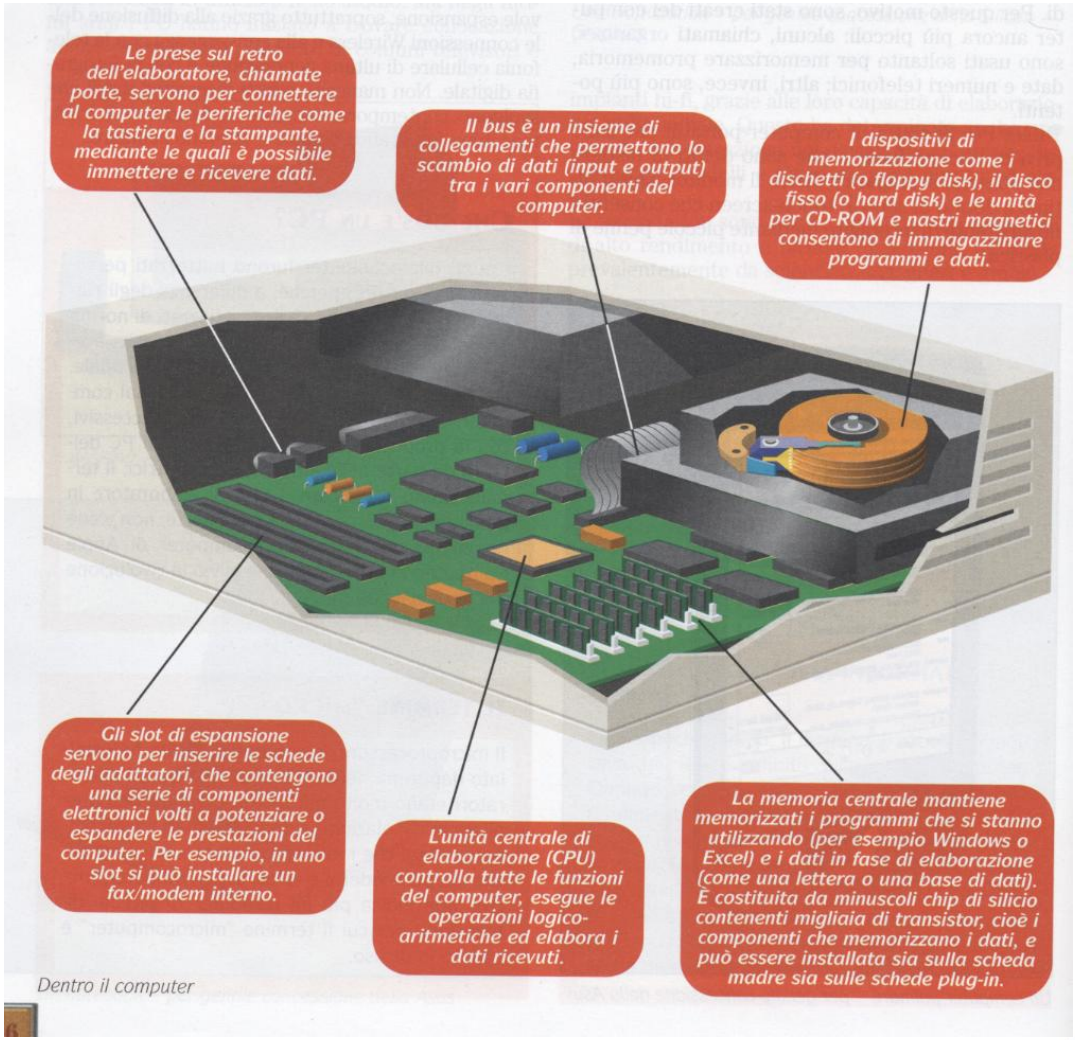
Interno di un calcolatore *desktop*



Dentro il computer

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

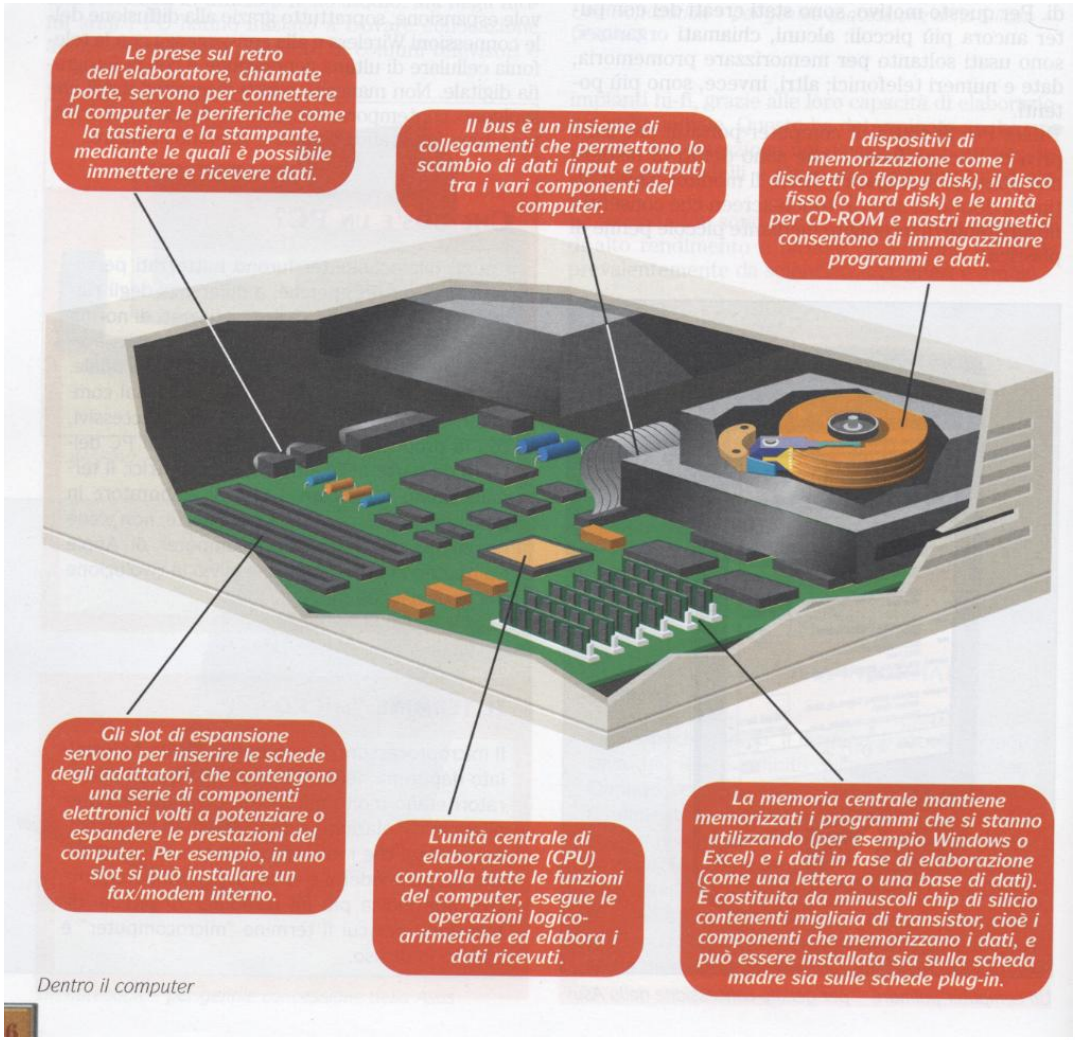
Interno di un calcolatore *desktop*



(In che differisce, questo, da un *laptop* ?)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Interno di un calcolatore *desktop*



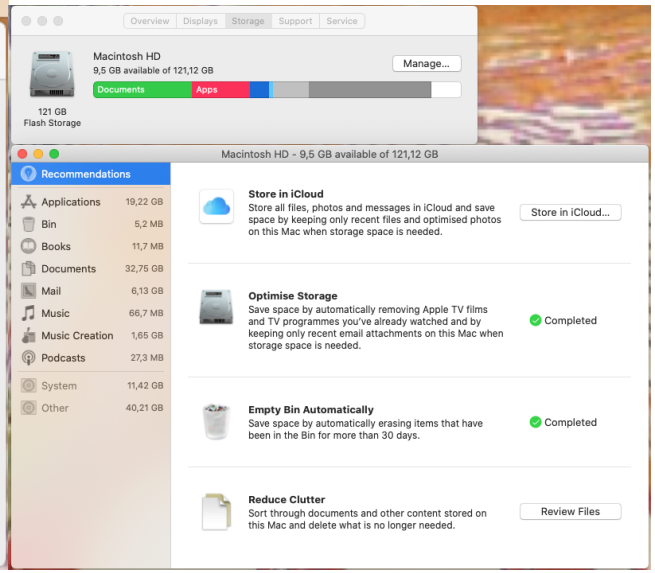
(Il disco è un HD o un SSD ?)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Un esempio specifico: Introspezione di un Mac

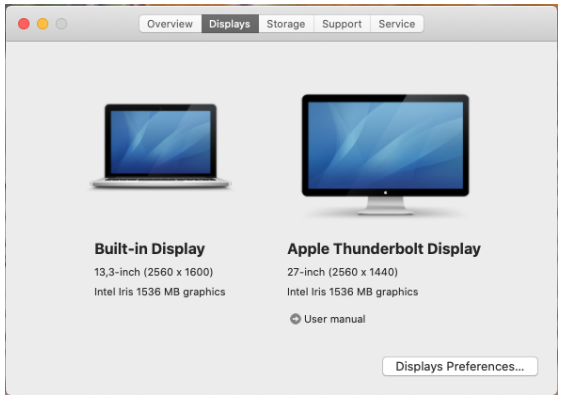
UNIVERSITÀ DEGLI STUDI DI TRIESTE

Un esempio specifico: Introspezione di un Mac



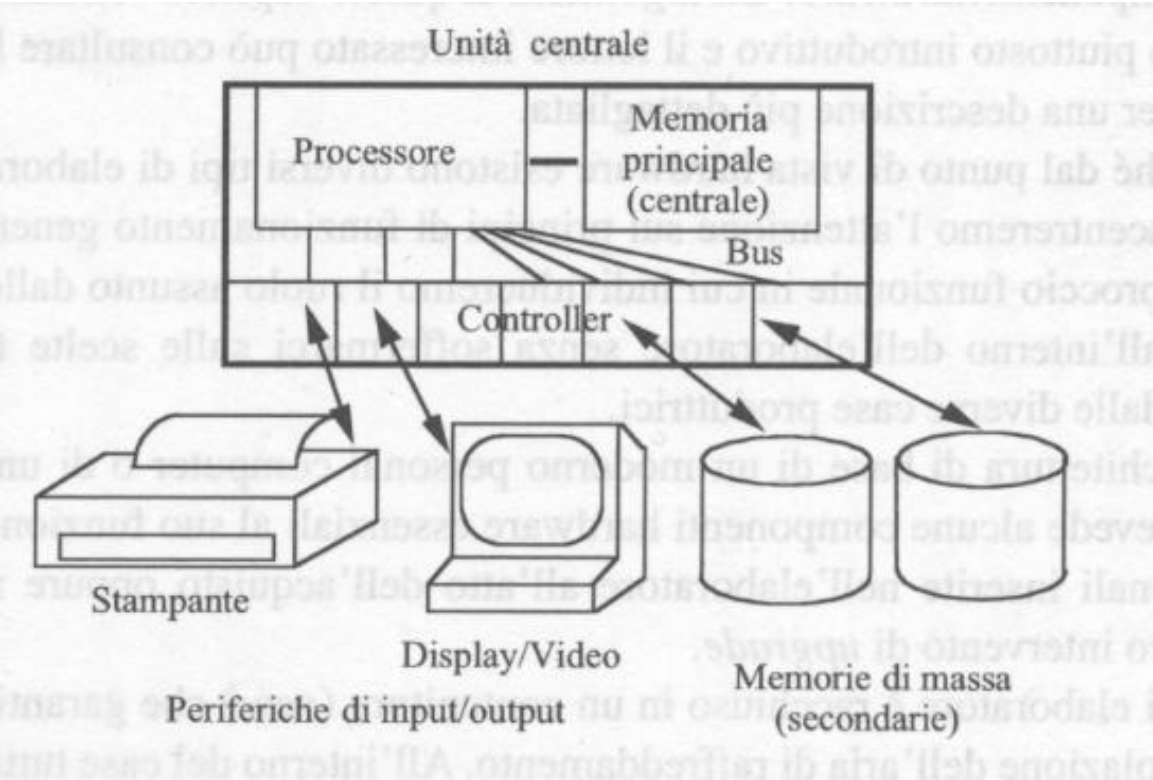
UNIVERSITÀ DEGLI STUDI DI TRIESTE

Un esempio specifico: Introspezione di un Mac

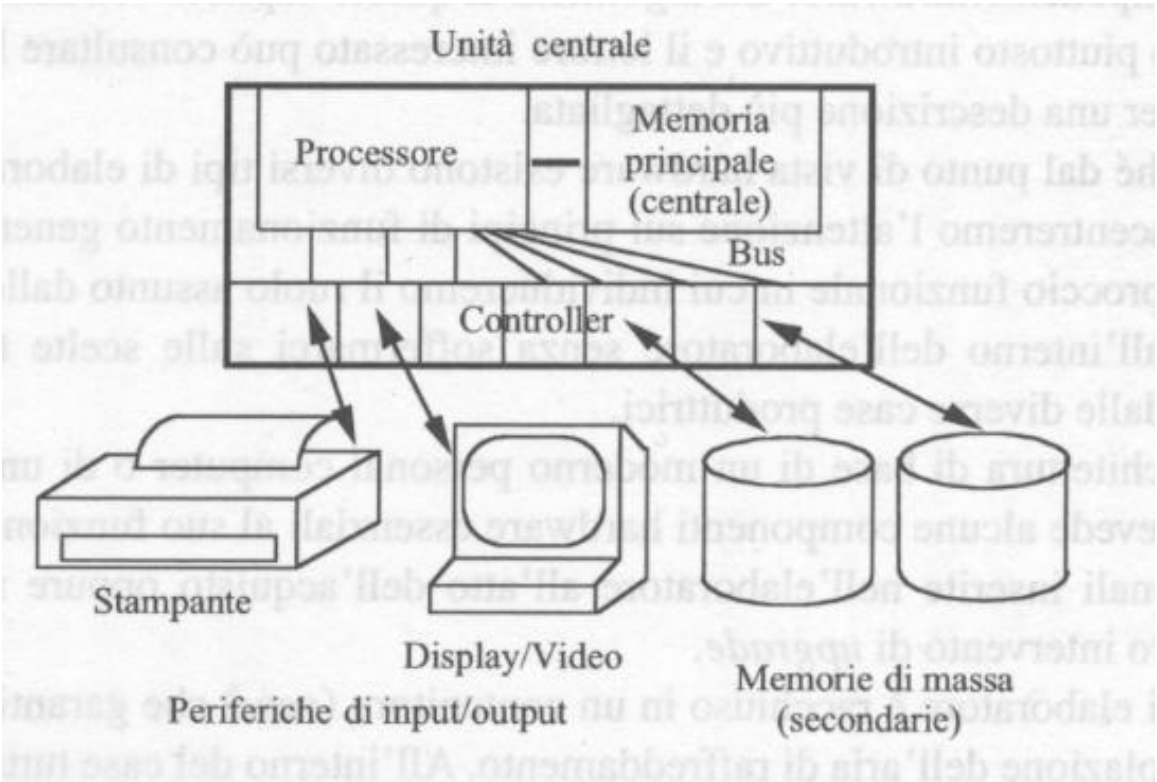


UNIVERSITÀ DEGLI STUDI DI TRIESTE

Altro ritratto di un *computer*



Altro ritratto di un *computer*



(Come individuare ciò che davvero caratterizza un computer ?)

Una definizione rigorosa

Poiché ha poca importanza se il computer è connesso a una stampante o a un accelerometro [· · ·] chiameremo i computer con il loro nome piú tecnico, processore. Un processore comprende la CPU e una piccola area di memoria chiamata cache, ed è connesso a dispositivi di input e output. La differenza principale la fanno i dispositivi connessi al computer e, naturalmente, il software.

Lawrence Snyder, Alessandro Amoroso (2015)

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Una definizione rigorosa ???

Poiché ha poca importanza se il computer è connesso a una stampante o a un accelerometro [· · ·] chiameremo i computer con il loro nome piú tecnico, processore. Un processore comprende la CPU e una piccola area di memoria chiamata cache, ed è connesso a dispositivi di input e output. La differenza principale la fanno i dispositivi connessi al computer e, naturalmente, il software.

Lawrence Snyder, Alessandro Amoroso (2015)

Dubbio: Non è una concezione troppo minimalista ?

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Una definizione rigorosa

Calcolatore $=_{\text{Def}}$ dispositivo elettronico veloce che accetta in ingresso informazione digitalizzata, la elabora in base a una lista (detta *programma*) di istruzioni memorizzate al suo interno e fornisce in uscita l'informazione risultante.

V. C. Hamacher, Z. G. Vrasenic, S. G. Zaky (2001)

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Una definizione rigorosa ???

Calcolatore $=_{\text{Def}}$ dispositivo elettronico veloce che accetta in ingresso informazione digitalizzata, la elabora in base a una lista (detta *programma*) di istruzioni memorizzate al suo interno e fornisce in uscita l'informazione risultante.

V. C. Hamacher, Z. G. Vrasenic, S. G. Zaky (2001)

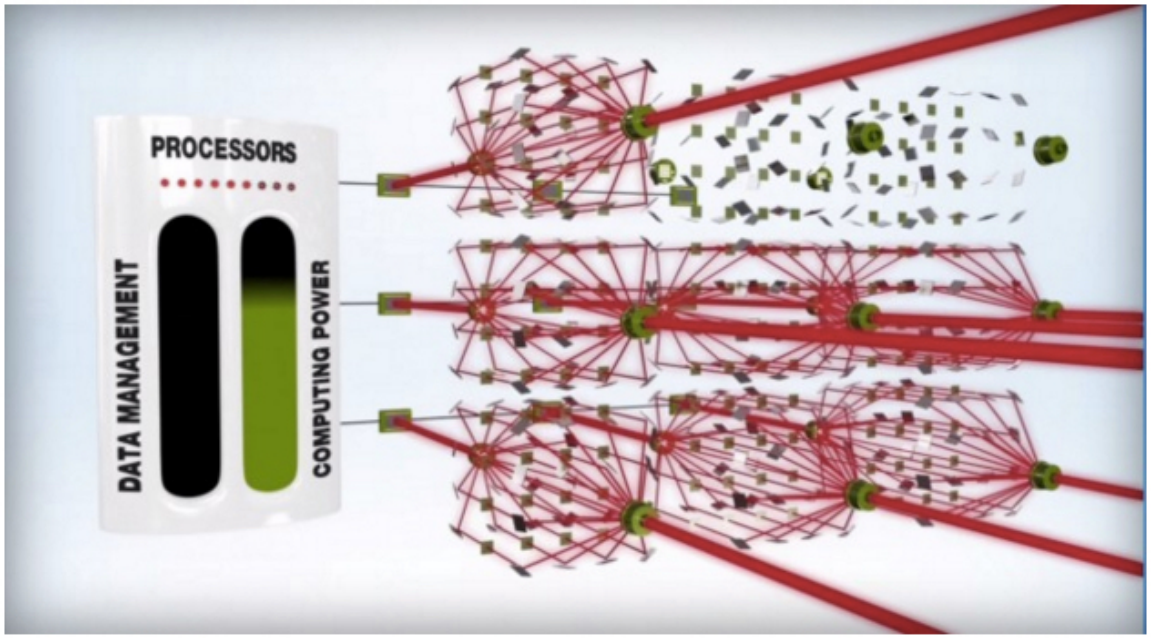
Dubbio: Quanto è cruciale l'elettronica ?

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Breve digressione

By 2020, you could have an exascale speed-of-light optical computer on your desk

By Sebastian Anthony on August 8, 2014 at 1:29 pm | 62 Comments



Breve digressione

Optical or **photonic computing** uses photons produced by lasers or diodes for computation. For decades, photons have promised to allow a higher bandwidth than the electrons used in conventional computers.


UNIVERSITÀ
DEGLI STUDI DI TRIESTE

Breve digressione

OpenMind BBVA THEMES DISCOVER Search Login En | Es

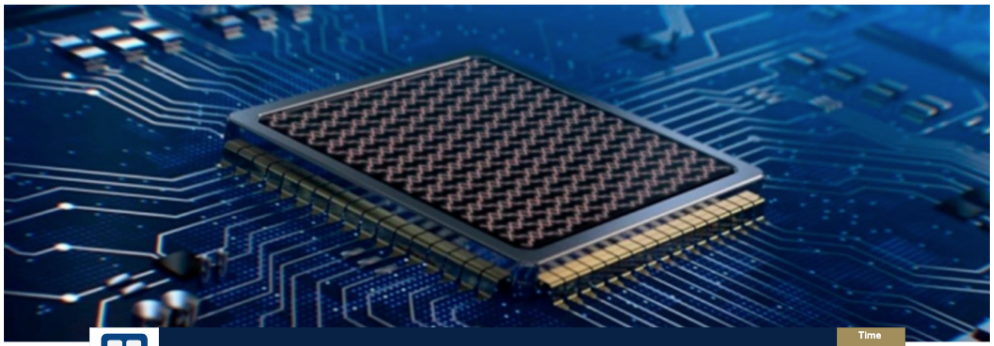
Start > Optical Computing: Solving Problems at the Speed of Light

TECHNOLOGY · FUTURE

07 February 2020

Optical Computing: Solving Problems at the Speed of Light

Computing | Innovation | Technology



Ventana al Conocimiento (Knowledge Window)
Scientific journalism

Time 4 ′ to read

According to Moore's law —actually more like a forecast, formulated in 1965 by Intel co-founder Gordon Moore— the number of transistors in a microprocessor doubles about every two years, **boosting the power of the chips without increasing their energy consumption**. For half a century, Moore's prescient vision has presided over the spectacular progress made in the world of computing. However, by 2015, the engineer himself predicted that we are reaching a saturation point in current technology. Today, quantum computing holds out hope for a new technological leap, but there is another option on which many are pinning their hopes: optical computing, which **replaces electronics (electrons) with light (photons)**.

The end of Moore's law is a natural consequence of physics: to pack more transistors into the same space they have to be shrunk down, which increases their speed while simultaneously reducing their energy consumption. The miniaturisation of silicon transistors has succeeded in breaking the 7-nanometre barrier, which used to be considered the limit, but this reduction cannot continue indefinitely. And although more powerful systems can always be obtained by increasing the number of transistors, in doing so the processing speed will decrease and the heat of the chips will rise.

Related publications

Chemical Computing, the Future of Artificial Intelligence

What is Affective Computing?

Quantum Computing

Do you want to stay up to date with our new publications?

Receive the OpenMind newsletter with all the latest contents published on our website

Find out more here

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Breve digressione

Calcolatori a fluido

di O. Lew Wood e Harold L. Fox

In breve: *L'idea di un calcolatore che operi per mezzo di getti d'acqua o di aria sembra a prima vista troppo stupida per essere presa in considerazione in questi tempi in cui l'elettronica è così altamente perfezionata. Tuttavia si è recentemente dimostrato che un flusso di elementi fluidi è capace di attuare le stesse funzioni logiche del « flusso » di elettricità e che risente in minor grado di fattori di disturbo come le radiazioni, il calore e le vibrazioni. L'interazione di getti di fluido l'uno con l'altro e con le pareti delle apparecchiature che li contengono rende attuabile un sistema stabile per il calcolo numerico. Inoltre criteri standard per la selezione dei materiali per le apparecchiature assicurano la loro stabilità, come anche la possibilità di avere sistemi logici a basso prezzo e fabbricati in serie.*

Uno dei maggiori impedimenti, tuttavia, per una diffusa applicazione dei calcolatori con logica basata sui fluidi, sarà la bassa velocità intrinseca. Questa limitazione naturalmente non li escluderà da quei campi d'impiego dove l'uomo fa parte dell'« anello » di controllo o dove i requisiti nel tempo di risposta non sono così essenziali come un'alta sicurezza e uno stabile funzionamento in differenti condizioni. Perciò questo nuovo metodo di attuare le funzioni logiche richiede un attento esame prima di decidere che non sia applicabile.

Moderni orientamenti della scienza e della tecnica, Etas-Kompass, 1966

L'HARDWARE

- L'aspetto *hardware* del calcolatore è rappresentato dai circuiti elettronici ed elettromeccanici che lo compongono;

V. C. Hamacher, Z. G. Vrasenic, S. G. Zaky (2001)

1

L'HARDWARE

- L'aspetto *hardware* del calcolatore è rappresentato dai circuiti elettronici ed elettromeccanici che lo compongono;
- l'*architettura* del calcolatore, invece, è definita come la combinazione delle funzionalità operative delle singole unità hardware che costituiscono il sistema di calcolo, il flusso di informazioni tra queste unità e il relativo controllo.

V. C. Hamacher, Z. G. Vrasenic, S. G. Zaky (2001)

1

L'HARDWARE

- L'aspetto *hardware* del calcolatore è rappresentato dai circuiti elettronici ed elettromeccanici che lo compongono;
- l'*architettura* del calcolatore, invece, è definita come la combinazione delle funzionalità operative delle singole unità hardware che costituiscono il sistema di calcolo, il flusso di informazioni tra queste unità e il relativo controllo.

V. C. Hamacher, Z. G. Vrasenic, S. G. Zaky (2001)

E il software[1] ?

[1] Ossia i programmi, le *app*...

HW / SW

"Al tempo dei primi calcolatori la distinzione fra hardware e software era chiarissima. Nel tempo si è piuttosto confusa ···

Hardware e software sono logicamente equivalenti.

si prevede che la funzione cambi."

A. S. Tenenbaum (2000)

HW / SW

"Al tempo dei primi calcolatori la distinzione fra hardware e software era chiarissima. Nel tempo si è piuttosto confusa ···

Hardware e software sono logicamente equivalenti.

Qualsiasi operazione venga effettuata dal software può essere direttamente inglobata nell'hardware:

L'hardware è software pietrificato.

si prevede che la funzione cambi."

A. S. Tenenbaum (2000)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

HW / SW

"Al tempo dei primi calcolatori la distinzione fra hardware e software era chiarissima. Nel tempo si è piuttosto confusa ···

Hardware e software sono logicamente equivalenti.

Qualsiasi operazione venga effettuata dal software può essere direttamente inglobata nell'hardware:

L'hardware è software pietrificato.

Naturalmente è vero anche il contrario: qualsiasi istruzione eseguita dall'hardware può venir simulata dal software.

si prevede che la funzione cambi."

A. S. Tenenbaum (2000)

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

HW / SW

"Al tempo dei primi calcolatori la distinzione fra hardware e software era chiarissima. Nel tempo si è piuttosto confusa ···

Hardware e software sono logicamente equivalenti.

Qualsiasi operazione venga effettuata dal software può essere direttamente inglobata nell'hardware:

L'hardware è software pietrificato.

Naturalmente è vero anche il contrario: qualsiasi istruzione eseguita dall'hardware può venir simulata dal software. La decisione di inglobare certe funzioni nell'hardware e altre nel software si basa su fattori come costo, velocità, affidabilità, e frequenza con cui si prevede che la funzione cambi."

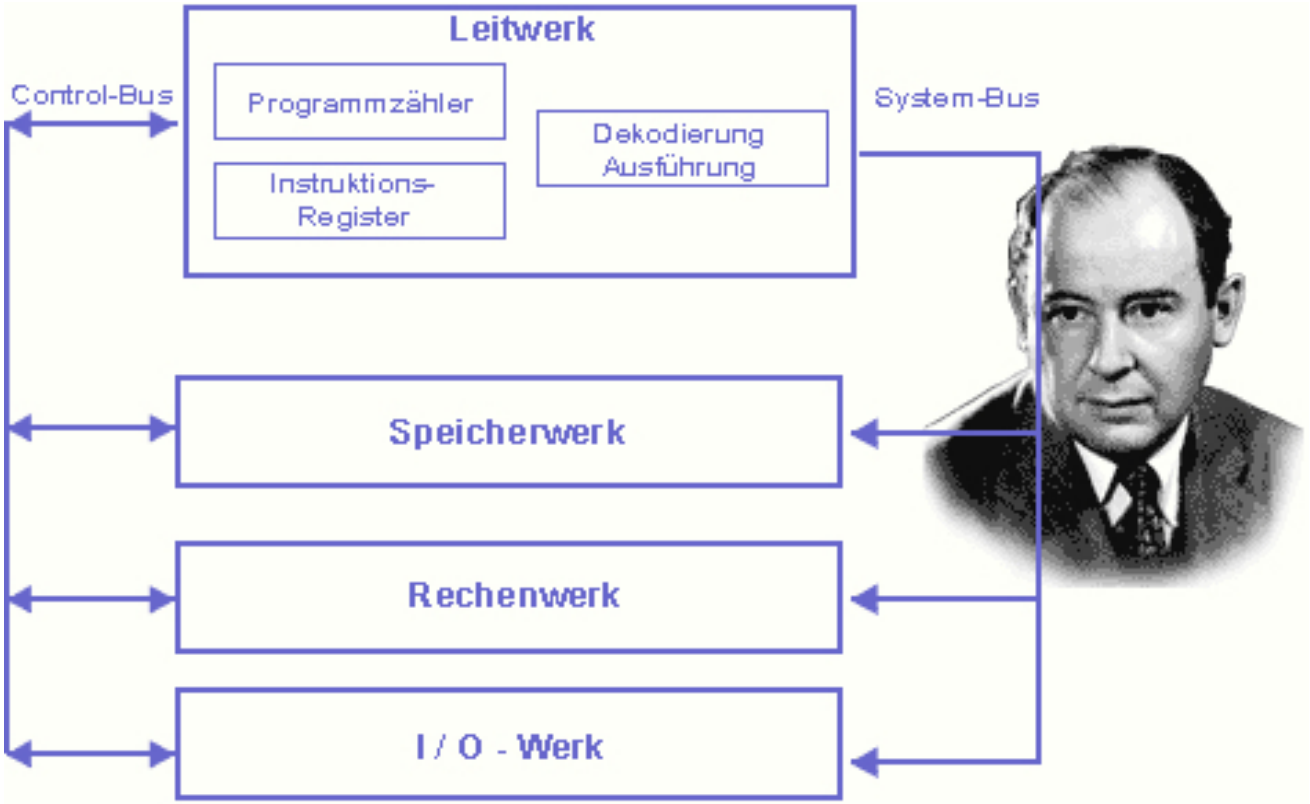
A. S. Tenenbaum (2000)

UNIVERSITÀ
DEGLI STUDI DI TRIESTE

A PROPOSITO DI *architettura* DEL *calcolatore*

Considerando quanto è sfumata la distinzione fra `HW` e `SW`, piuttosto che di calcolatore (o '*computer*') qui sarebbe proprio parlare di sistema di elaborazione.

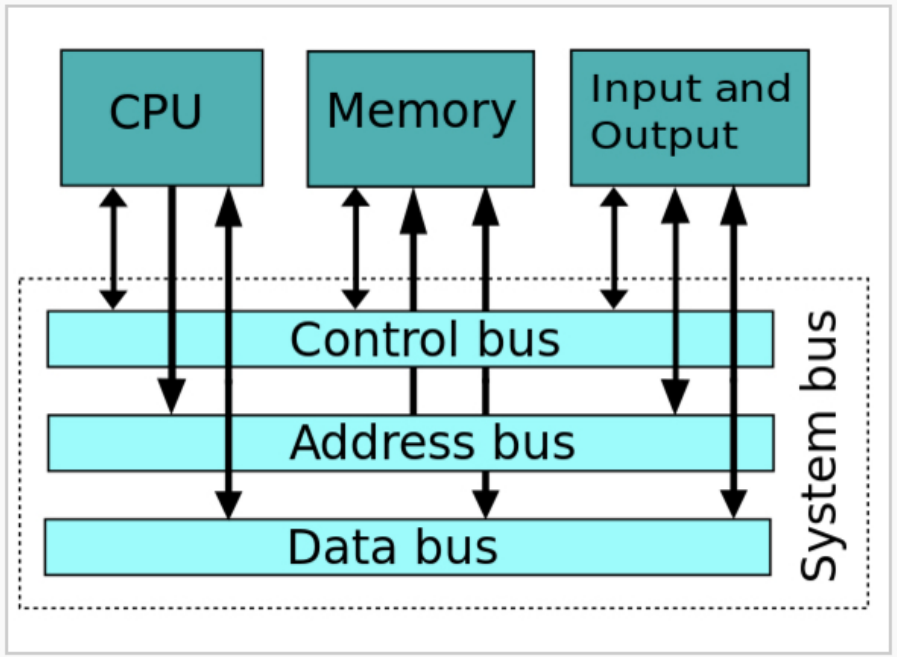
Un'intramontabile organizzaz. dell'hardware



John Louis von Neumann, nato János Lajos Neumann (Budapest, 1903 – Washington, 1957), poliedrico "matematico" ungherese naturalizzato statunitense.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

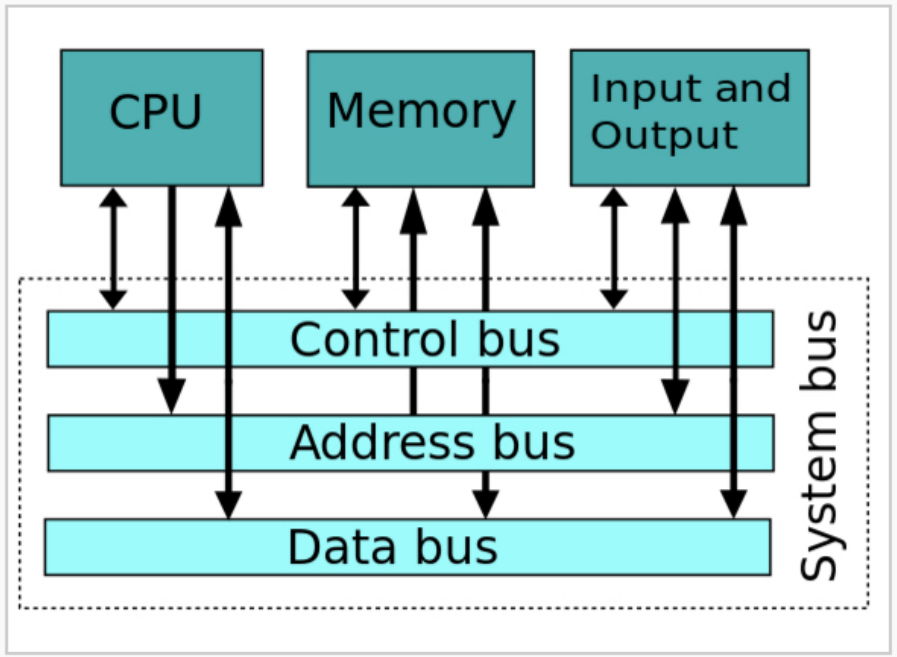
Schema dell'architettura di von Neumann



In senso orario:

1. Microprocessore

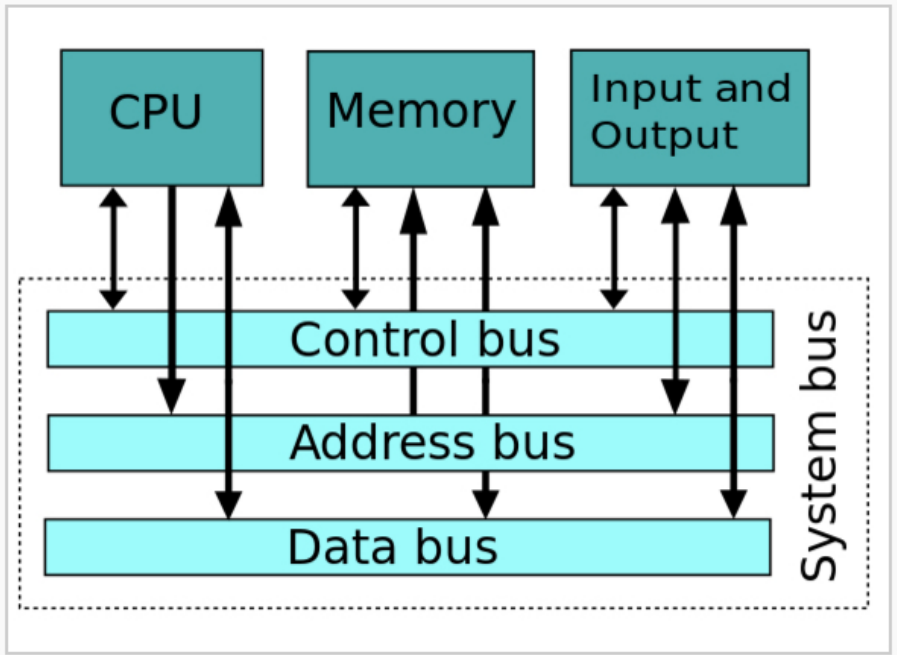
Schema dell'architettura di von Neumann



In senso orario:

1. Microprocessore
2. Memoria centrale

Schema dell'architettura di von Neumann

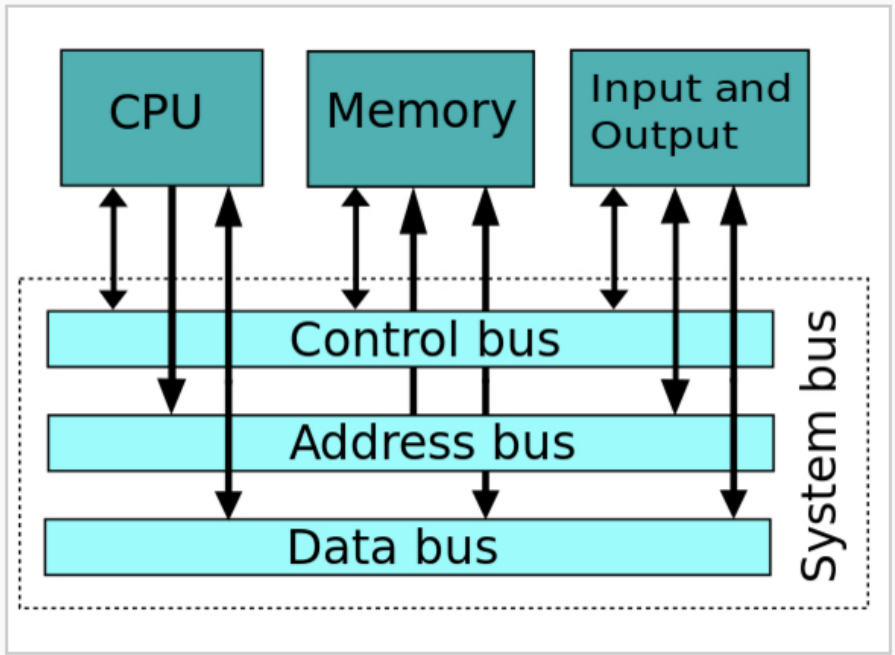


In senso orario:

1. Microprocessore
2. Memoria centrale
3. Interfacce con le periferiche

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Schema dell'architettura di von Neumann

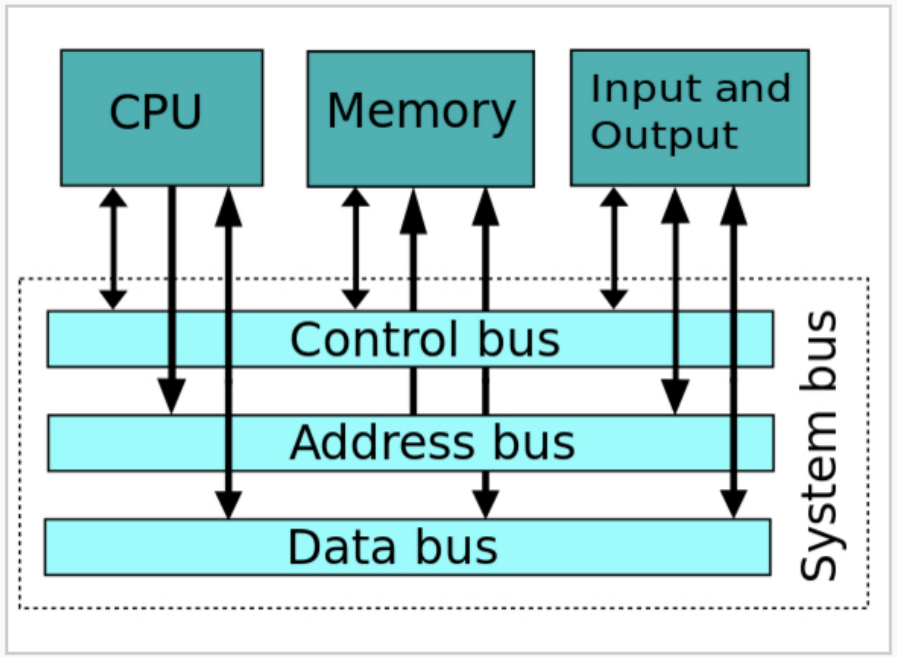


In senso orario:

1. Microprocessore
2. Memoria centrale
3. Interfacce con le periferiche
4. Bus di sistema (tripartito)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Schema dell'architettura di von Neumann



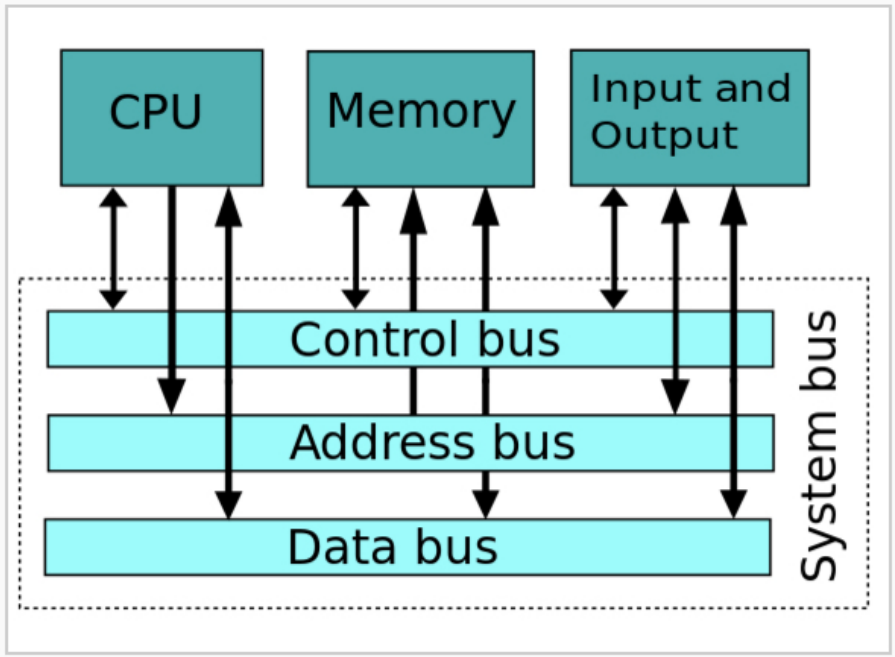
EDVAC

In senso orario:

1. Microprocessore
2. Memoria centrale
3. Interfacce con le periferiche
4. Bus di sistema (tripartito)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Schema dell'architettura di von Neumann



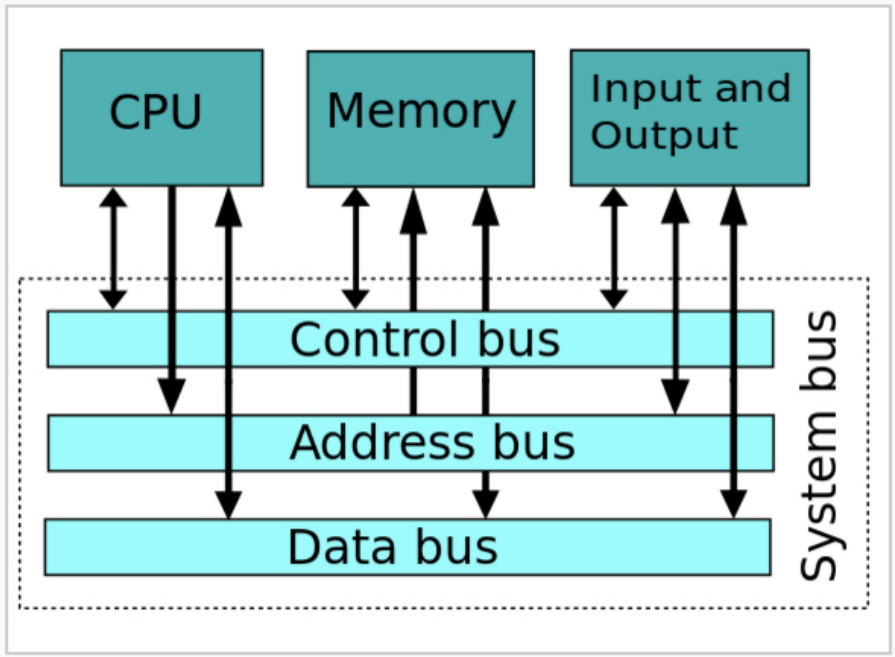
EDVAC IAS Machine

In senso orario:

1. Microprocessore
2. Memoria centrale
3. Interfacce con le periferiche
4. Bus di sistema (tripartito)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Schema dell'architettura di von Neumann



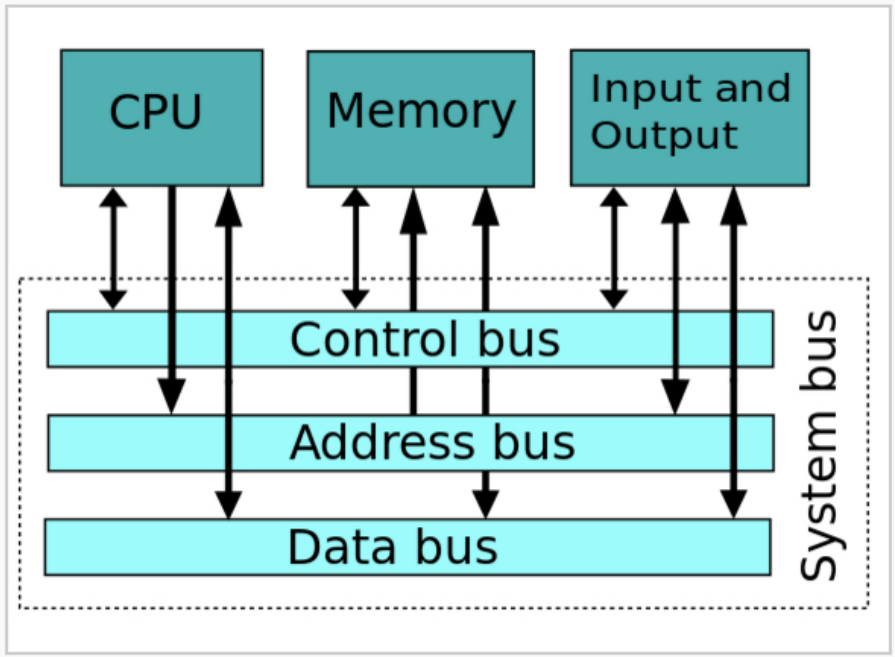
In senso orario:

1. Microprocessore
2. Memoria centrale
3. Interfacce con le periferiche
4. Bus di sistema (tripartito)

`EDVAC` `IAS Machine` `ACE` (?)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Schema dell'architettura di von Neumann



In senso orario:

1. Microprocessore
2. Memoria centrale
3. Interfacce con le periferiche
4. Bus di sistema (tripartito)

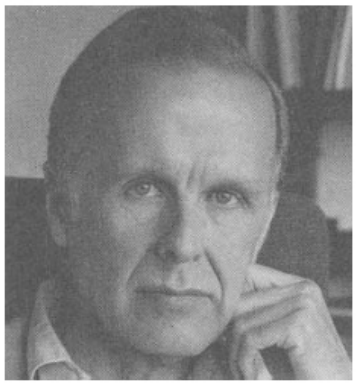
EDVAC IAS Machine ~~ACE~~ (?)

CISC vs RISC

L'architettura di von Neumann è innovabile ?

Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs

John Backus
IBM Research Laboratory, San Jose



Author's address: 91 Saint Germain Ave., San Francisco, CA 94114.

Conventional programming languages are growing ever more enormous, but not stronger. Inherent defects at the most basic level cause them to be both fat and weak: their primitive word-at-a-time style of programming inherited from their common ancestor—the von Neumann computer, their close coupling of semantics to state transitions, their division of programming into a world of expressions and a world of statements, their inability to effectively use powerful combining forms for building new programs from existing ones, and their lack of useful mathematical properties for reasoning about programs.

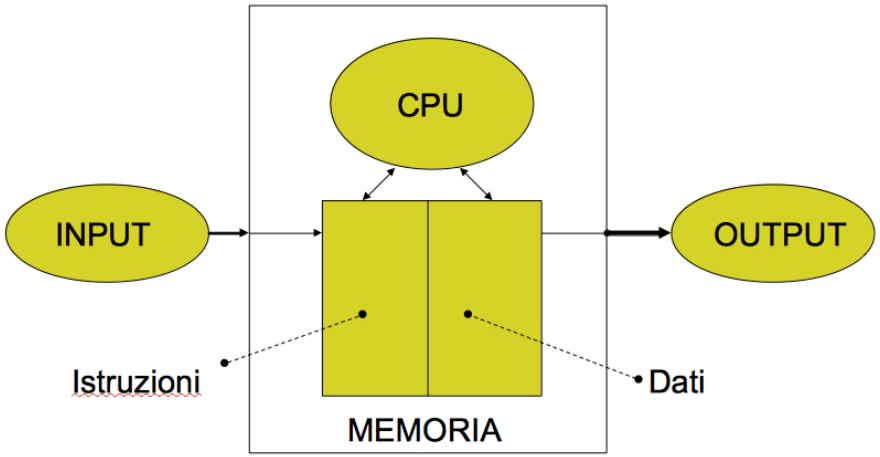
An alternative functional style of programming is founded on the use of combining forms for creating programs. Functional programs deal with structured data, are often nonrepetitive and nonrecursive, are hierarchically constructed, do not name their arguments, and do not require the complex machinery of procedure declarations to become generally applicable. Combining forms can use high level programs to build still higher level ones in a style not possible in conventional languages.

613

Communications of the ACM

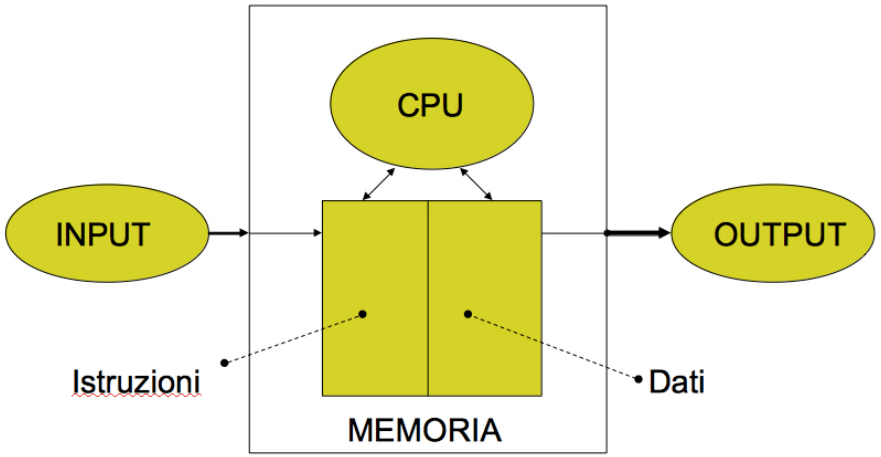
August 1978
Volume 21
Number 8

Computer a programma memorizzato



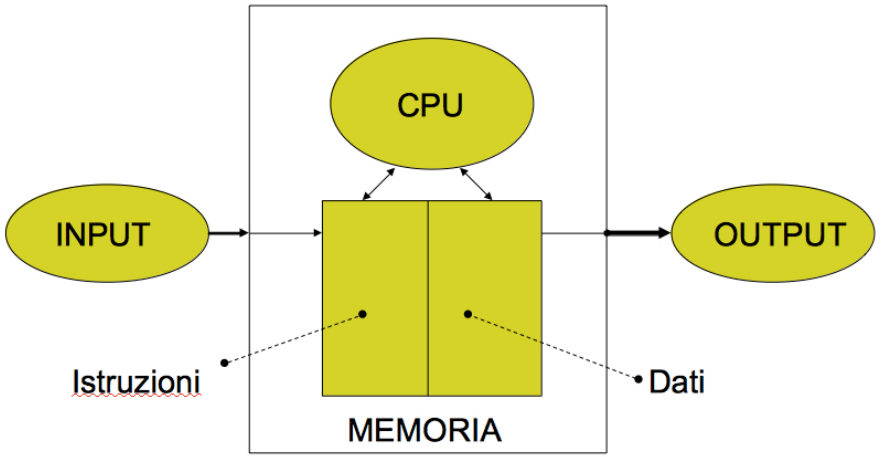
L'espressione "*stored-program computer*" ("computer a programma memorizzato") viene utilizata in riferimento alla memoria centrale: fu von Neumann a introdurla in *First draft of a report on the EDVAC*, datato 30 giugno 1945, con il significato particolare che gli attribuiamo oggi.

Computer a programma memorizzato



Pregio: Rapidità di accesso alle istruzioni.

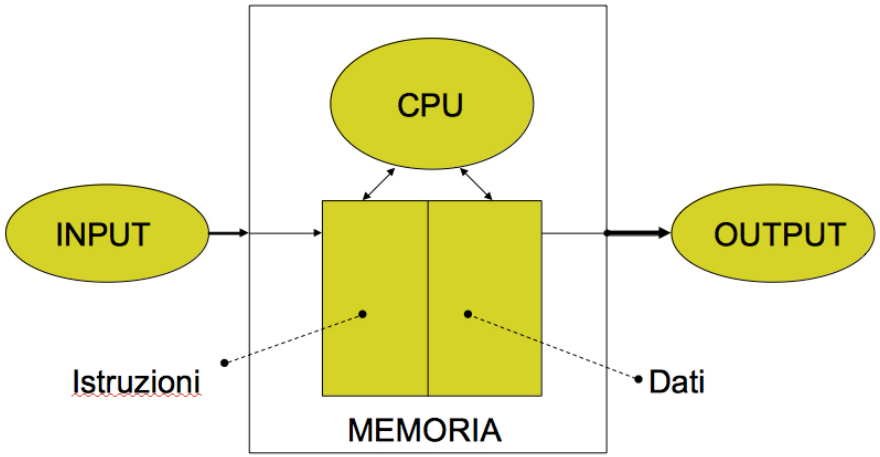
Computer a programma memorizzato



Pregio: Rapidità di accesso alle istruzioni.

Vulnerabilità: Possibilità che un programma, alterando le proprie istruzioni, pregiudichi il suo stesso funzionamento.

Computer a programma memorizzato

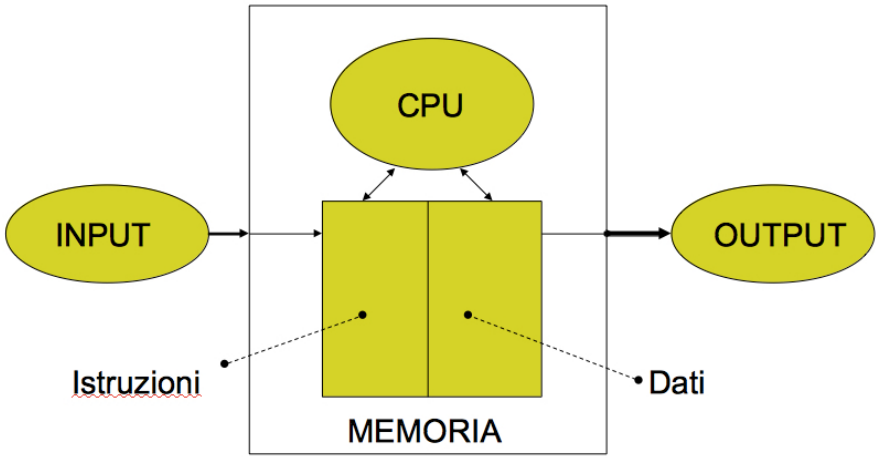


Pregio: Rapidità di accesso alle istruzioni.

Vulnerabilità: Possibilità che un programma, alterando le proprie istruzioni, pregiudichi il suo stesso funzionamento.

Pregio: Possibilità che un programma modifichi le proprie istruzioni, com'è giusto avvenga in un processo di apprendimento.

Computer a programma memorizzato



Dopo il *Colossus Mark I* del 1943, il *Colossus Mark II* del 1944 e l'ENIAC del 1946, viene realizzato nel 1948 a Manchester, UK, lo *Small-Scale Experimental Machine*, primo computer elettronico a programma memorizzato della storia. A partire dal 1948 il computer a programma memorizzato si diffonde, diventando in breve tempo la norma per il computer programmabile.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Lo sviluppo di *software* a livelli

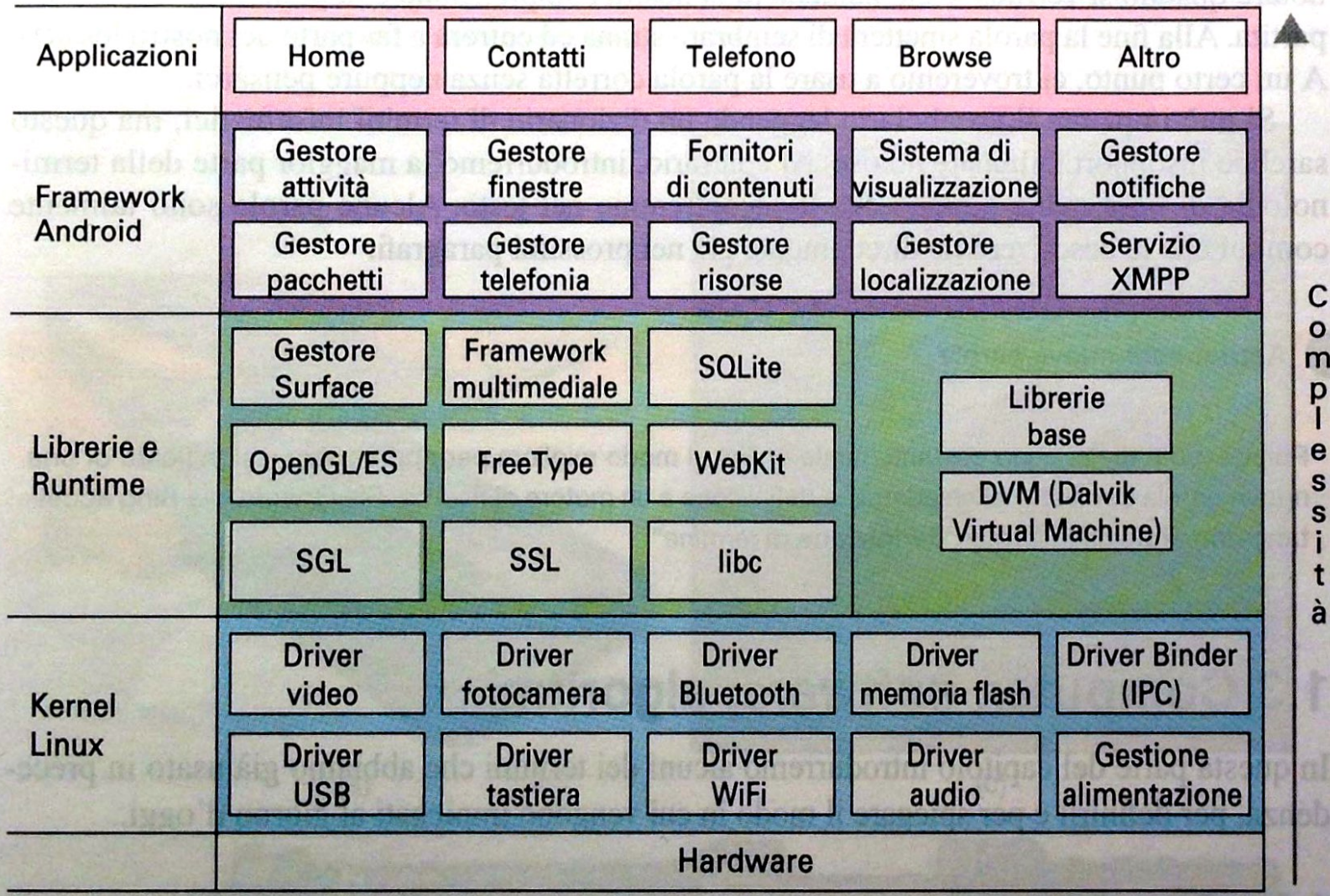
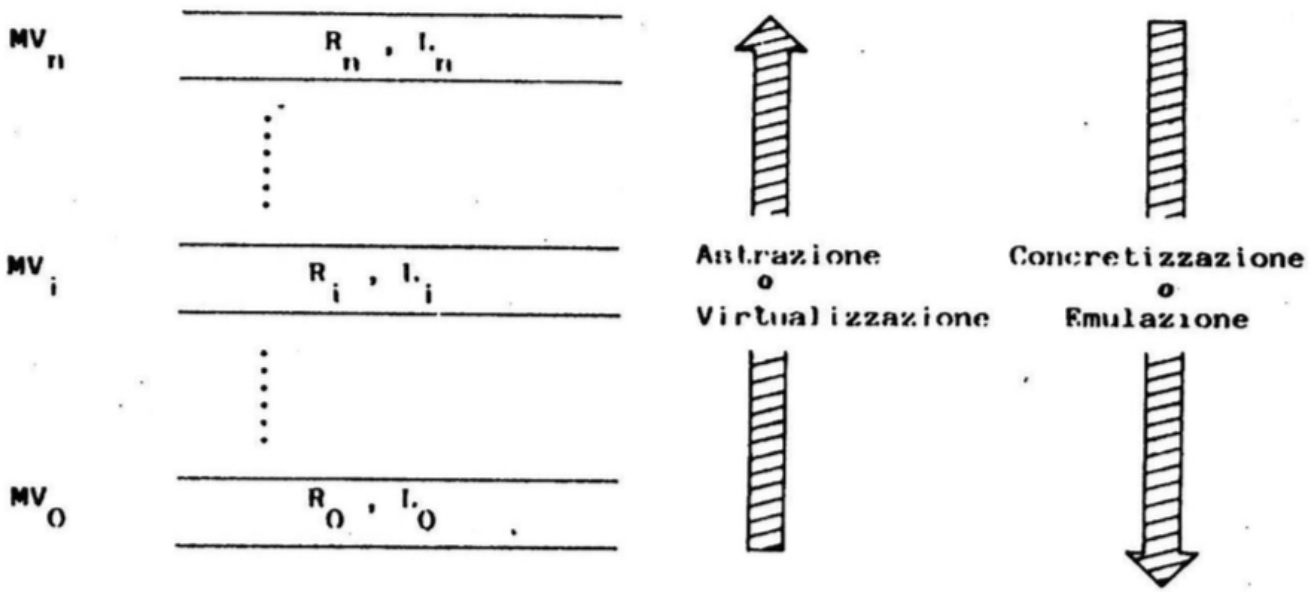


Figura 1.8
La pila del softwa
di uno smartphon
Android; in basso
si trova l'hardwar
in alto le app.

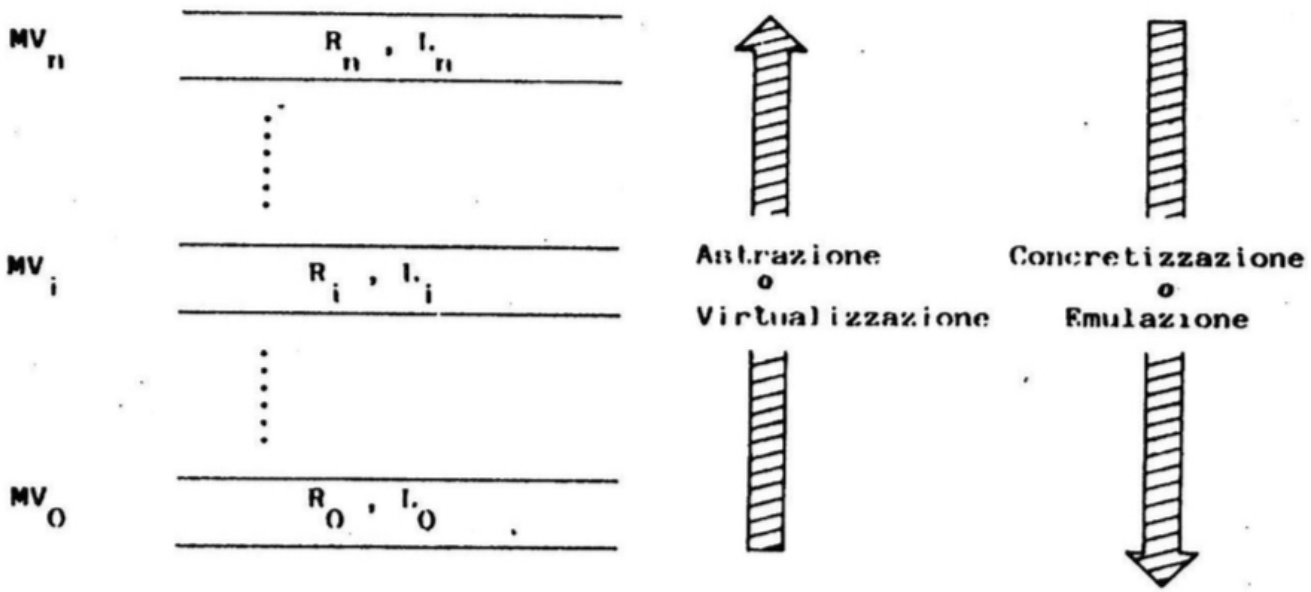
Dalle componenti a un'architettura

Le funzionalità di un sistema di elaborazione nel suo complesso possono essere viste come ripartite, secondo un certo numero di livelli, o **macchine virtuali** $\{MV_0, MV_1, \ldots, MV_n\}$, come schematizzato in figura:



Dalle componenti a un'architettura

Le funzionalità di un sistema di elaborazione nel suo complesso possono essere viste come ripartite, secondo un certo numero di livelli, o **macchine virtuali** $\{MV_0, MV_1, \ldots, MV_n\}$, come schematizzato in figura:

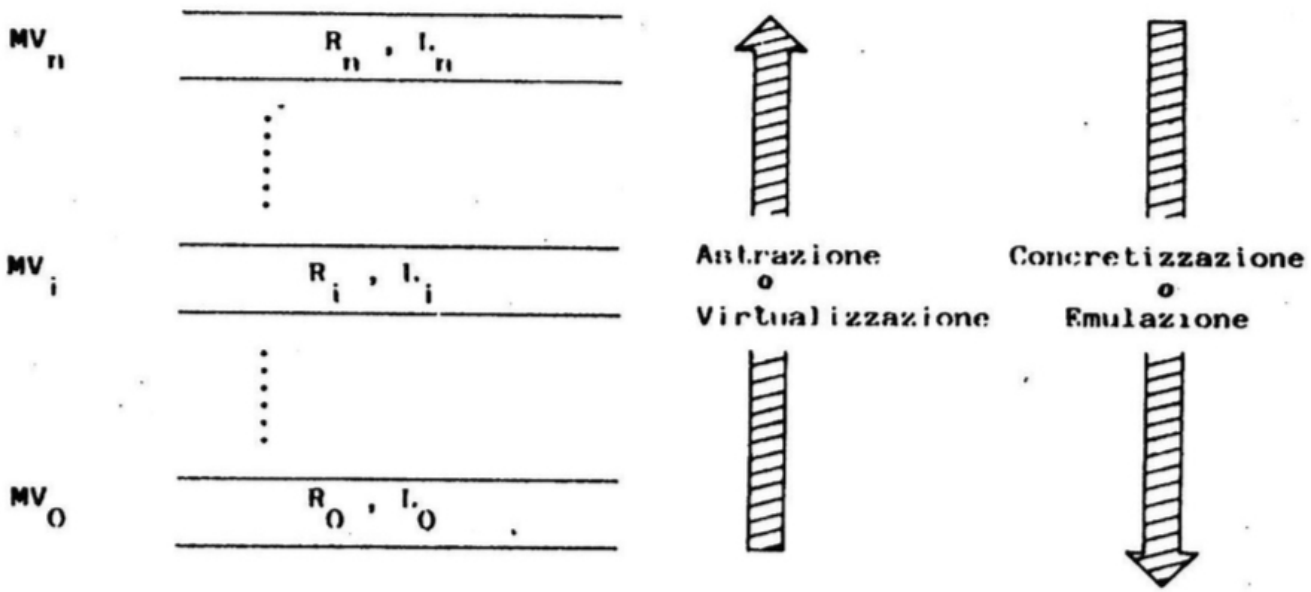


Il livello MV_0 è tipicamente quello dei componenti fisici (elettronici) a partire dai quali viene costruita una **astrazione**, o **virtualizzazione**, sempre maggiore delle funzionalità di interesse per il sistema e degli oggetti su cui esso è destinato ad operare. Il livello MV_n è quello che possiamo chiamare delle "applicazioni" in quanto fornisce la visione, di volta in volta ritenuta più opportuna degli oggetti e degli e degli strumenti mediante i quali il sistema viene utilizzato dall'esterno. Questo livello necessita normalmente, per potere essere effettivamente implementato, di un procedimento di **concretizzazione**, o **emulazione** ottenuto mediante uno o più livelli intermedi $\{MV_j \mid j = 1, \ldots, n\text{-}1\}$.

ESTE

Dalle componenti a un'architettura

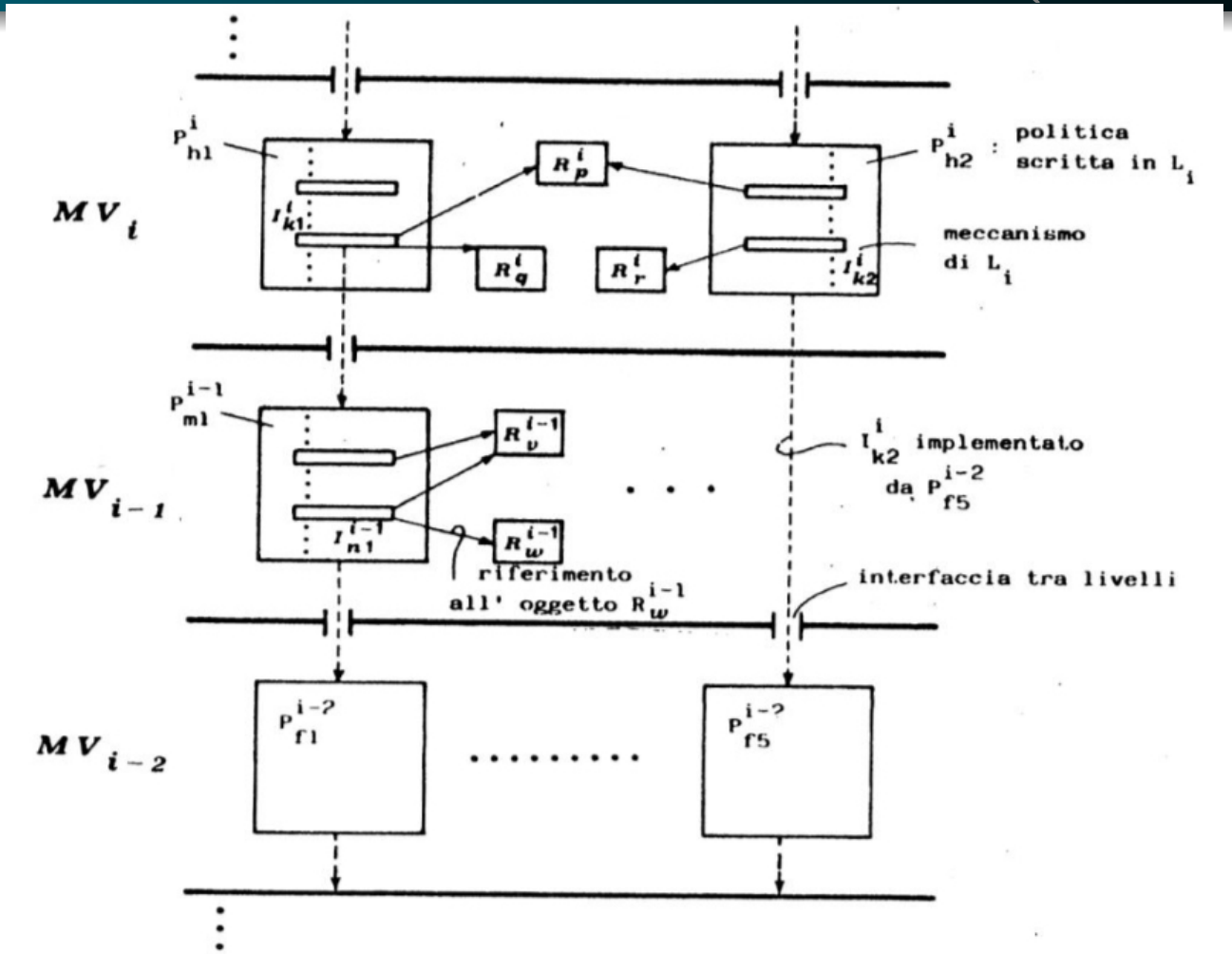
Le funzionalità di un sistema di elaborazione nel suo complesso possono essere viste come ripartite, secondo un certo numero di livelli, o **macchine virtuali** $\{MV_0, MV_1, \ldots, MV_n\}$, come schematizzato in figura:



Il livello MV_0 è tipicamente quello dei componenti fisici (elettronici) a partire dai quali viene costruita una **astrazione**, o **virtualizzazione**, sempre maggiore delle funzionalità di interesse per il sistema e degli oggetti su cui esso è destinato ad operare. Il livello MV_n è quello che possiamo chiamare delle "applicazioni" in quanto fornisce la visione, di volta in volta ritenuta più opportuna degli oggetti e degli e degli strumenti mediante i quali il sistema viene utilizzato dall'esterno. Questo livello necessita normalmente, per potere essere effettivamente implementato, di un procedimento di **concretizzazione**, o **emulazione** ottenuto mediante uno o più livelli intermedi $\{MV_j \mid j = 1, \ldots, n-1\}$.

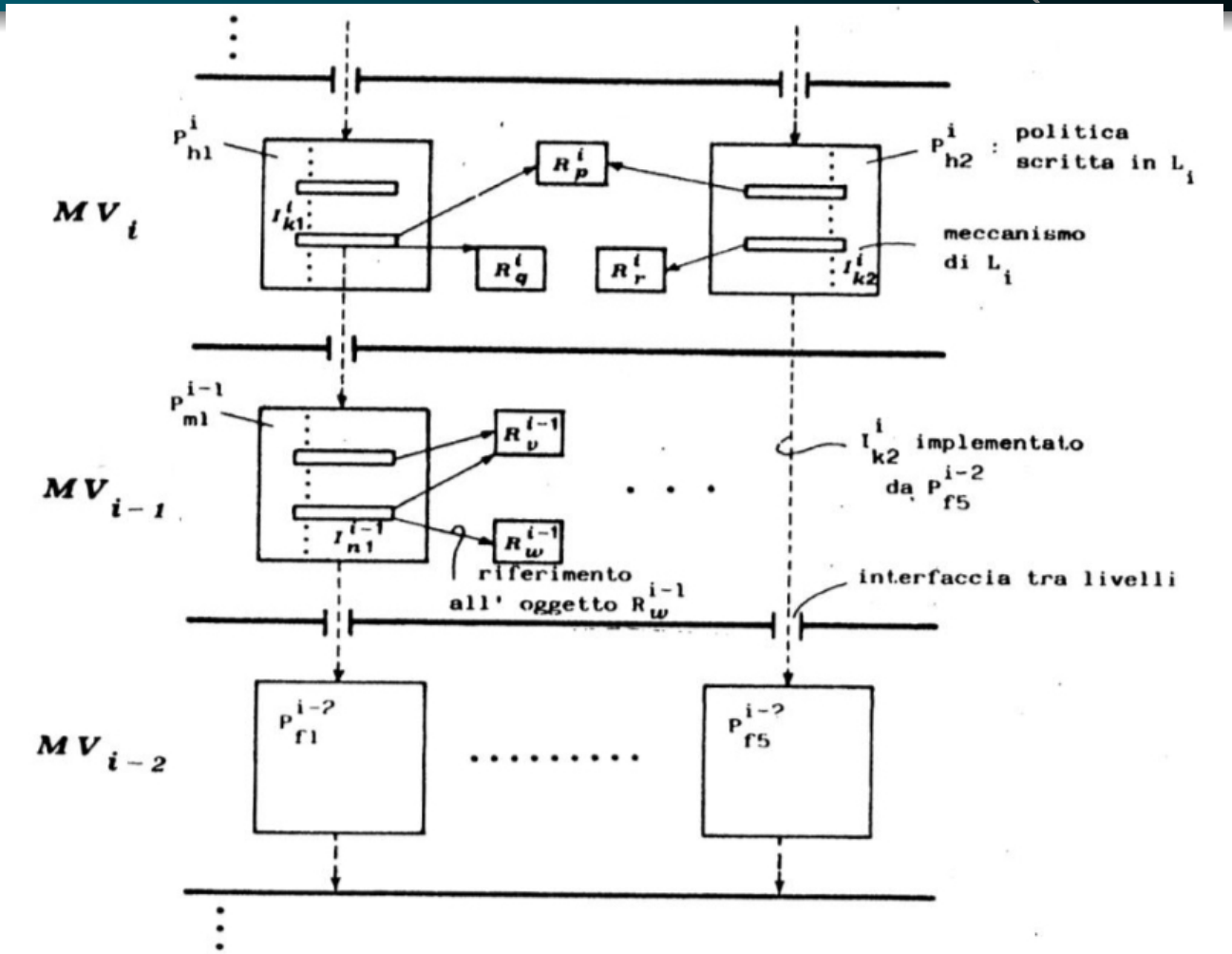
(Marco Vanneschi) ESTE

Gerarchia di macchine virtuali (Vanneschi)



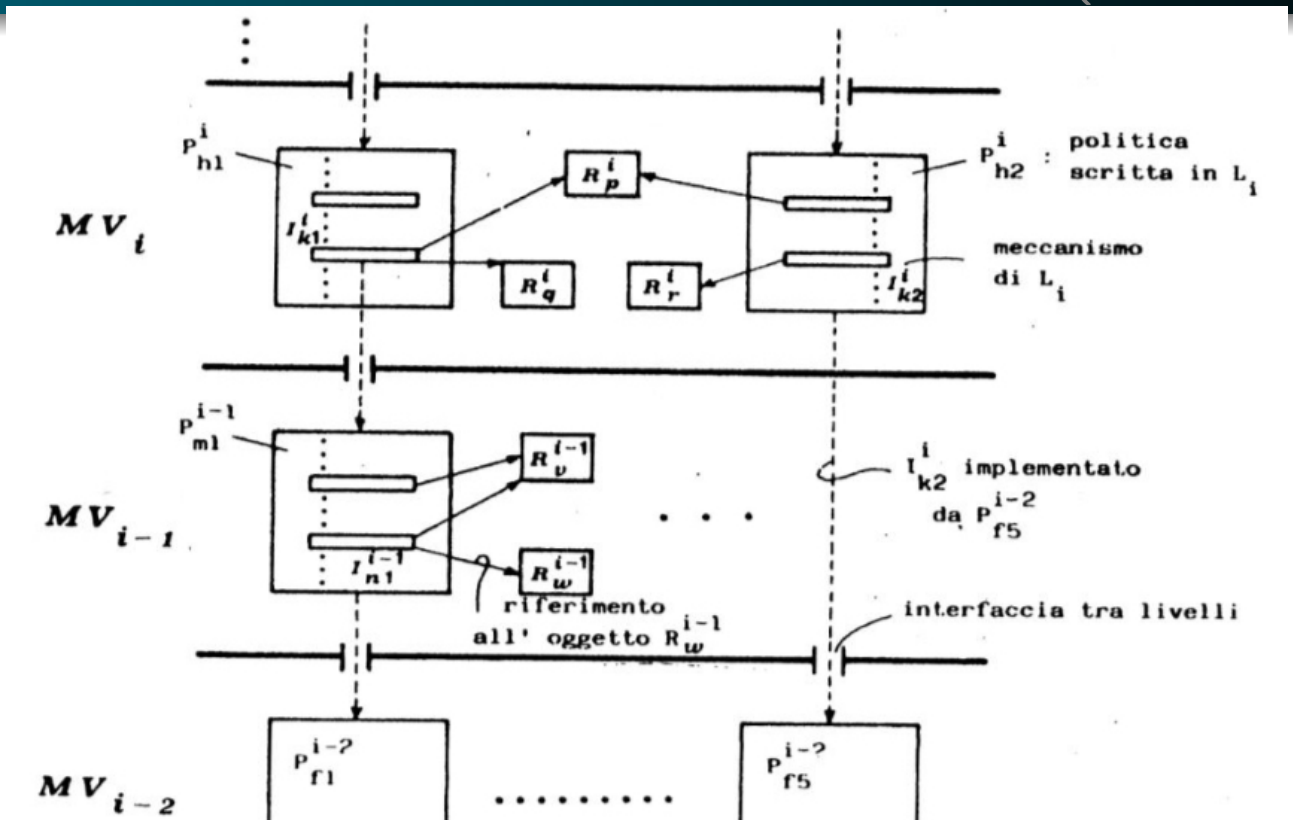
UNIVERSITÀ DEGLI STUDI DI TRIESTE

Gerarchia di macchine virtuali (Vanneschi)



I livelli { MV_0, …, MV_n} sono ordinati secondo una **relazione gerarchica**. Ciò significa che ogni istruzione primitiva I_k^i, *o meccanismo* del linguaggio L_i ($i > 0$) è implementata da un programma P_k, o **politica**, scritto nel linguaggio L_j, come mostrato in figura:

Gerarchia di macchine virtuali (Vanneschi)



dove:

a) $0 \leq j < i$: anche se il caso più frequente è quello in cui $j = i - 1$, talvolta possono esistere varie ragioni, legate alle prestazioni del sistema e/o al costo dell'implementazione, per le quali conviene che sia $j < i - 1$;

b) se I_{k1}^{i} e I_{k2}^{i} sono meccanismi distinti di L_i, essi possono essere implementati da politiche scritte rispettivamente con L_{j1} e L_{j2}, dove $j1 \neq j2$, anche il caso più frequente è quello in cui $j1 = j2$: valgono considerazioni analoghe al caso precedente.

IESTE

Esempio: Un programma Java '*multi-threaded*'...

Posso scrivere un programma Java (si tratta di un testo) che

Esempio: Un programma Java '*multi-threaded*'...

Posso scrivere un programma Java (si tratta di un testo) che

- una volta tradotto in codice JVM
  (ossia codice per la '*Java Virtual Machine*')...

Esempio: Un programma Java '*multi-threaded*'...

Posso scrivere un programma Java (si tratta di un testo) che

- una volta tradotto in codice JVM
  (ossia codice per la '*Java Virtual Machine*')...
- ...può essere mandato in esecuzione su qualsiasi(?) Sistema di Elaborazione...

Esempio: Un programma Java '*multi-threaded*'...

Posso scrivere un programma Java (si tratta di un testo) che

- una volta tradotto in codice JVM
  (ossia codice per la '*Java Virtual Machine*')...
- ...può essere mandato in esecuzione su qualsiasi(?) Sistema di Elaborazione...
- ...creando l'illusione che vi siano piú micro processori ...

Esempio: Un programma Java '*multi-threaded*'...

Posso scrivere un programma Java (si tratta di un testo) che

- una volta tradotto in codice JVM
  (ossia codice per la '*Java Virtual Machine*')...
- ...può essere mandato in esecuzione su qualsiasi(?) Sistema di Elaborazione...
- ...creando l'illusione che vi siano piú micro processori ...
- ...e che la memoria abbia una capienza maggiore della sua capacità fisica

Esempio: Un programma Java '*multi-threaded*'...

Posso scrivere un programma Java (si tratta di un testo) che

- una volta tradotto in codice JVM
  (ossia codice per la '*Java Virtual Machine*')...
- ...può essere mandato in esecuzione su qualsiasi(?) Sistema di Elaborazione...
- ...creando l'illusione che vi siano piú micro processori ...
- ...e che la memoria abbia una capienza maggiore della sua capacità fisica

(CPU virtuali, Memoria virtuale...)

UNIVERSITÀ
DEGLI STUDI DI TRIESTE