UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

Instruction Set Architecture (ISA)

Prof.ssa Giulia Cisotto

giulia.cisotto@units.it

Trieste, 3 marzo 2025

AGENDA DI OGGI

- *Quiz autovalutazione su rappresentazione dell'informazione*
- Introduzione al concetto di **Instruction Set Architecture (ISA)**
- Filosofie di **design per architetture** di elaborazione
- Dettagli su architetture di elaborazione
- **Gerarchie di memoria**
- Memoria centrale in MIPS32
- **Flusso di elaborazione**

$$0111111100011010_2 = 7F1A_{16}$$

$$AB2_{16} = 101010110010_2$$

$-4_{10} = \mathbf{1}0000100_2$

$-7_{10} = 01001_2$

$$3.625_{10} = 11.101_2$$

1 10000010 01001100000000000000000 **= -10.375**

01101001 01101110 01100110 01101111 01110010 01101101 01100001 01110100 01101001 01100011 01100001

i	n	f	o	r	m	a	t	i	c	a

X + Y = Z

00000001000010010101000000100000

00000001000010010101000001000000

Sotto assunzione di star condividendo la stessa architettura programmativa e quindi standard di codifica:

questa sequenza di 32 bit è l'istruzione di una SOMMA.

add $10, $8, $9 *Assembly*

X + Y = Z oppure Z = X + Y

Supponendo che X ed Y siano contenuti nei registri (celle di memoria) numero 8 e numero 9 e il risultato vada memorizzato (temporaneamente) nel registro 10, si faccia la somma dei due operandi

QtSpim

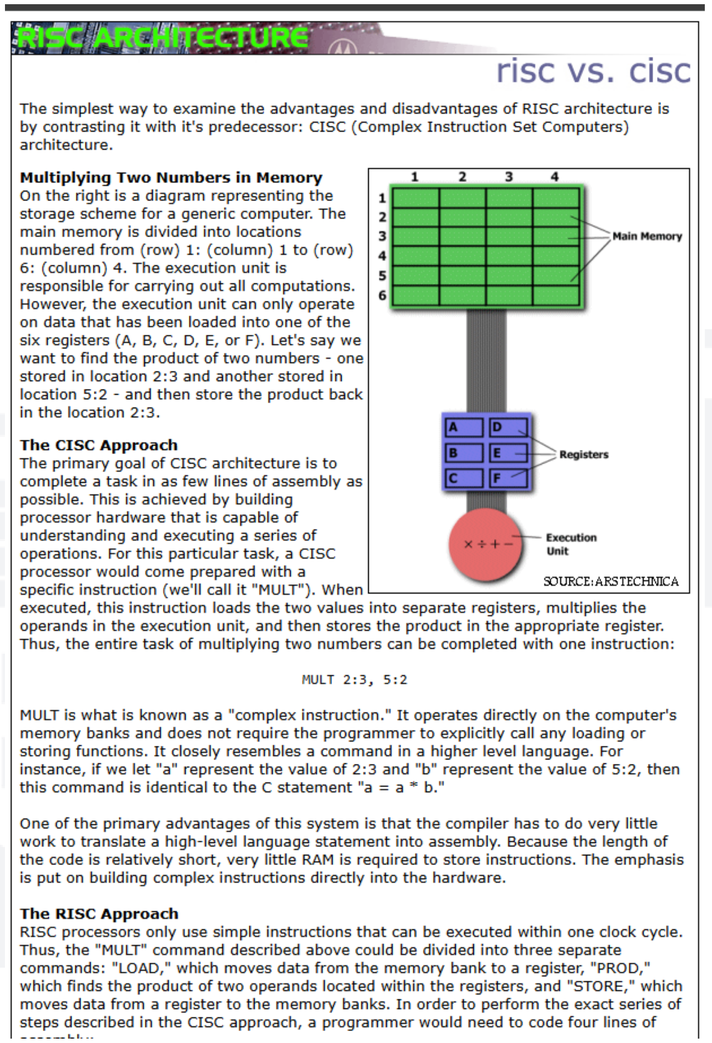
FILOSOFIE DI COSTRUZIONE DELL'ARCHITETTURA

RISC (Reduced Instruction Set Computing)

- Poche istruzioni semplici
- Struttura circuitalmente semplice
- Esecuzione veloce di una singola istruzione
- *Occorrono più istruzioni per fare cose anche semplici*
- Esempio: MIPS, ARM, PowerPC…

CISC (Complex Instruction Set Computing)

- Istruzioni complesse
- Struttura circuitalmente complicata
- Esecuzione più lenta di una singola istruzione
- *Occorrono meno istruzioni*
- Esempio: Intel x86 e tutti i derivati (Pentium ecc.)



RISC ARCHITECTURE

risc vs. cisc

The simplest way to examine the advantages and disadvantages of RISC architecture is by contrasting it with it's predecessor: CISC (Complex Instruction Set Computers) architecture.

Multiplying Two Numbers in Memory
On the right is a diagram representing the storage scheme for a generic computer. The main memory is divided into locations numbered from (row) 1: (column) 1 to (row) 6: (column) 4. The execution unit is responsible for carrying out all computations. However, the execution unit can only operate on data that has been loaded into one of the six registers (A, B, C, D, E, or F). Let's say we want to find the product of two numbers - one stored in location 2:3 and another stored in location 5:2 - and then store the product back in the location 2:3.

The CISC Approach
The primary goal of CISC architecture is to complete a task in as few lines of assembly as possible. This is achieved by building processor hardware that is capable of understanding and executing a series of operations. For this particular task, a CISC processor would come prepared with a specific instruction (we'll call it "MULT"). When executed, this instruction loads the two values into separate registers, multiplies the operands in the execution unit, and then stores the product in the appropriate register. Thus, the entire task of multiplying two numbers can be completed with one instruction:

```
MULT 2:3, 5:2
```

MULT is what is known as a "complex instruction." It operates directly on the computer's memory banks and does not require the programmer to explicitly call any loading or storing functions. It closely resembles a command in a higher level language. For instance, if we let "a" represent the value of 2:3 and "b" represent the value of 5:2, then this command is identical to the C statement "a = a * b."

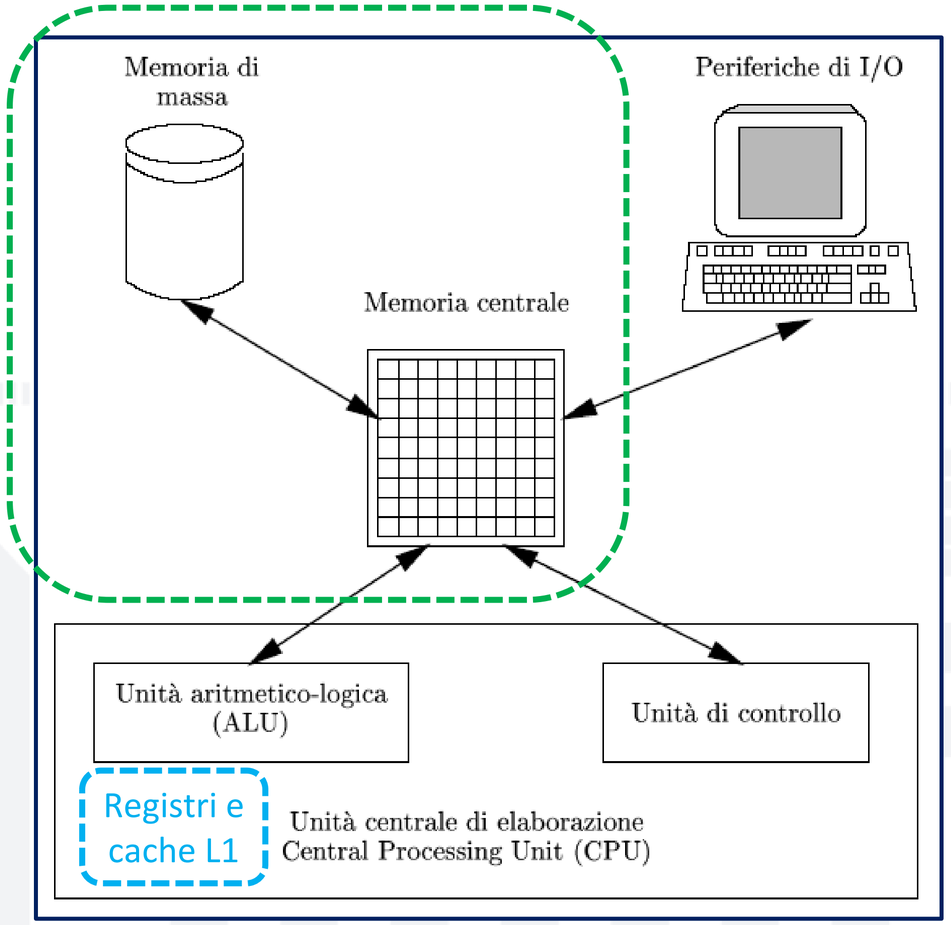
One of the primary advantages of this system is that the compiler has to do very little work to translate a high-level language statement into assembly. Because the length of the code is relatively short, very little RAM is required to store instructions. The emphasis is put on building complex instructions directly into the hardware.

The RISC Approach
RISC processors only use simple instructions that can be executed within one clock cycle. Thus, the "MULT" command described above could be divided into three separate commands: "LOAD," which moves data from the memory bank to a register, "PROD," which finds the product of two operands located within the registers, and "STORE," which moves data from a register to the memory banks. In order to perform the exact series of steps described in the CISC approach, a programmer would need to code four lines of

[Architetture RISC vs CISC (articolo Stanford Univ)]

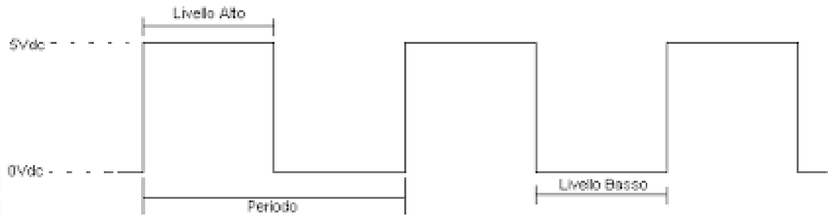
MODULO 1: Architettura degli elaboratori

John Von Neumann
Matematico ed informatico di origine ungherese che viveva e lavorava negli Stati Uniti negli anni ‘40



Architettura di un elaboratore di Von Neumann

Le varie parti dell’architettura devono sincronizzare le proprie attività: serve un **«CLOCK»**



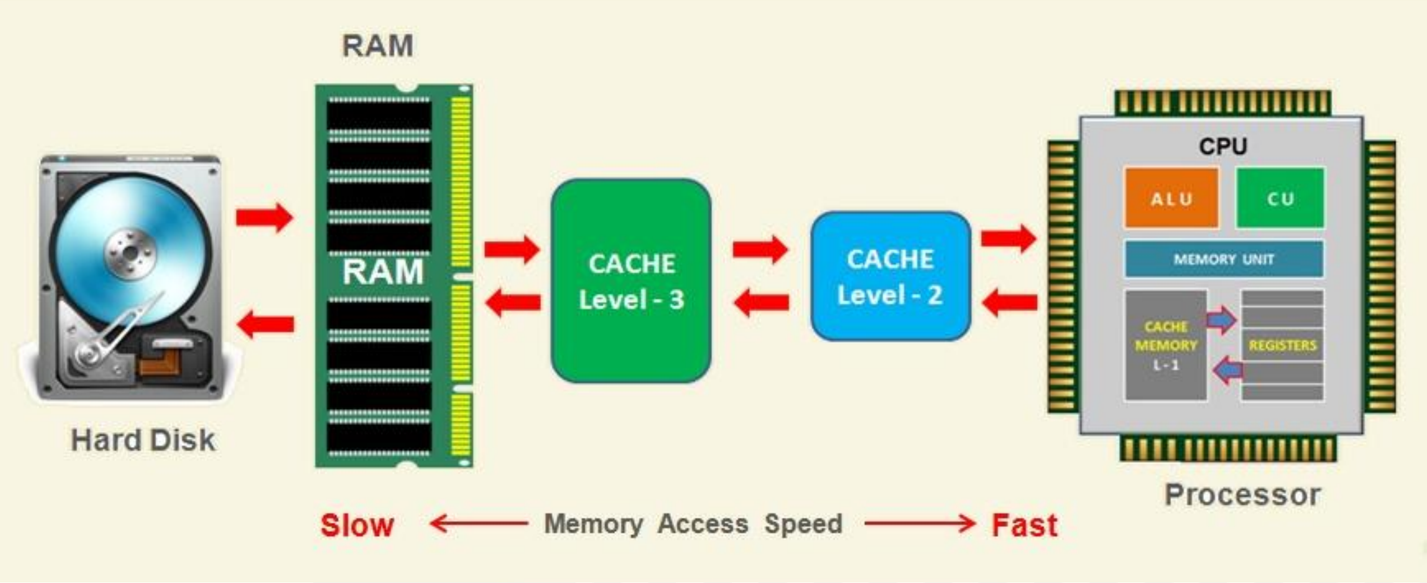
E’ un segnale periodico.

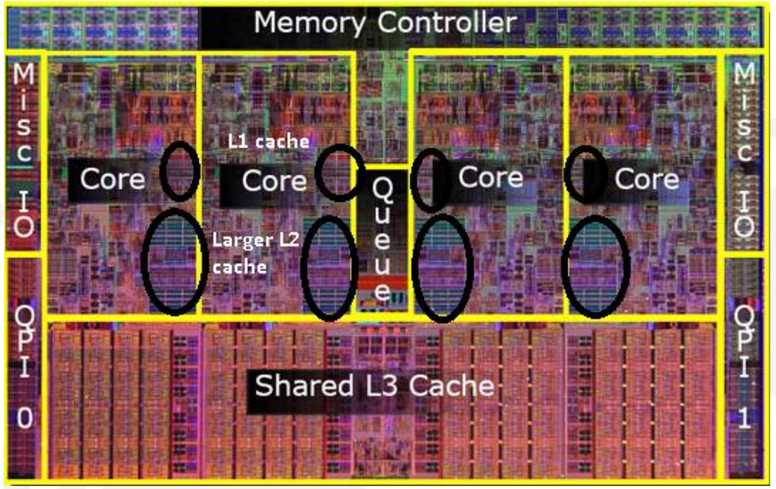
Più è veloce, più attività si possono fare nell’unità di tempo!

1GHz = 10^9 Hz

MEMORIE E LORO GERARCHIA (1/2)

In un'architettura di elaborazione esistono **diversi tipi di memorie** organizzate secondo una precisa **gerarchia** che bilancia la **velocità** di recupero/scrittura dati con la **durata e spazio** di memorizzazione.





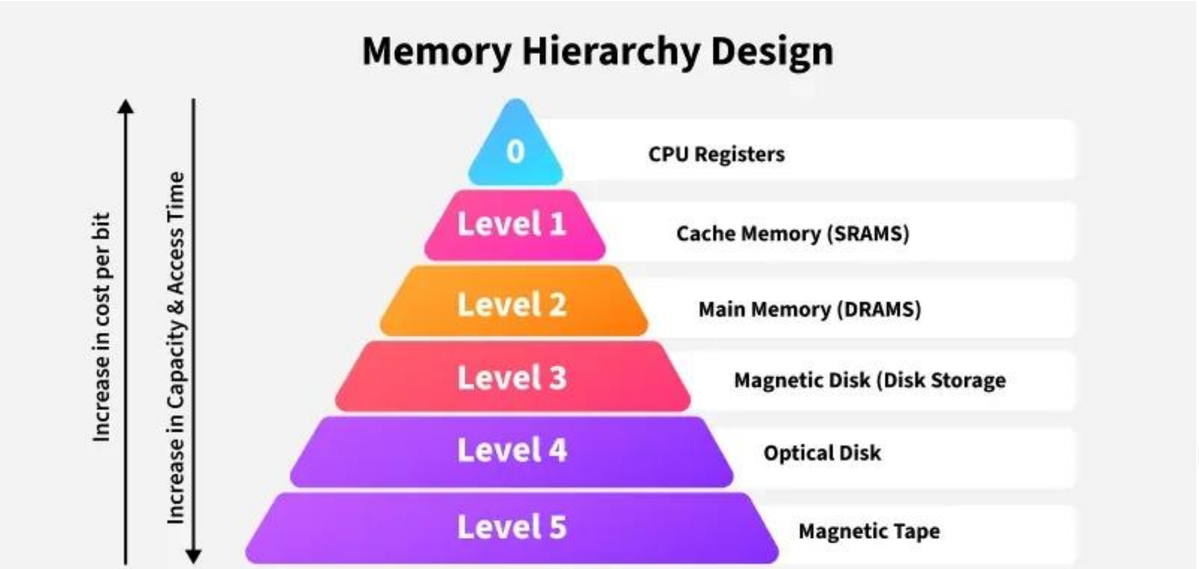
https://softwareg.com.au/it-it/blogs/hardware-per-computer/cose-una-cache-della-cpu?srsltid=AfmBOoqm4WpdBQJW_vu7BI0cpz6ve9tLMgwhsHOjJg3MQs8KpM1ulEl3

Esempio di memorie in AMD Zen 4 [Wiki]

Memoria primaria: RAM, cache e registri, ROM

Memoria secondaria: memorie di massa esterne

MEMORIE E LORO GERARCHIA (2/2)

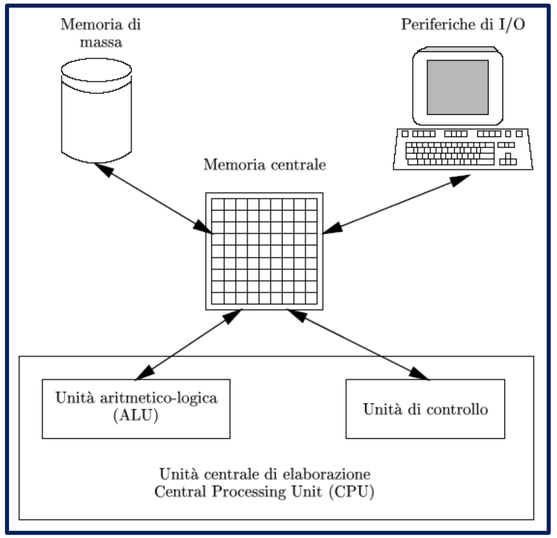


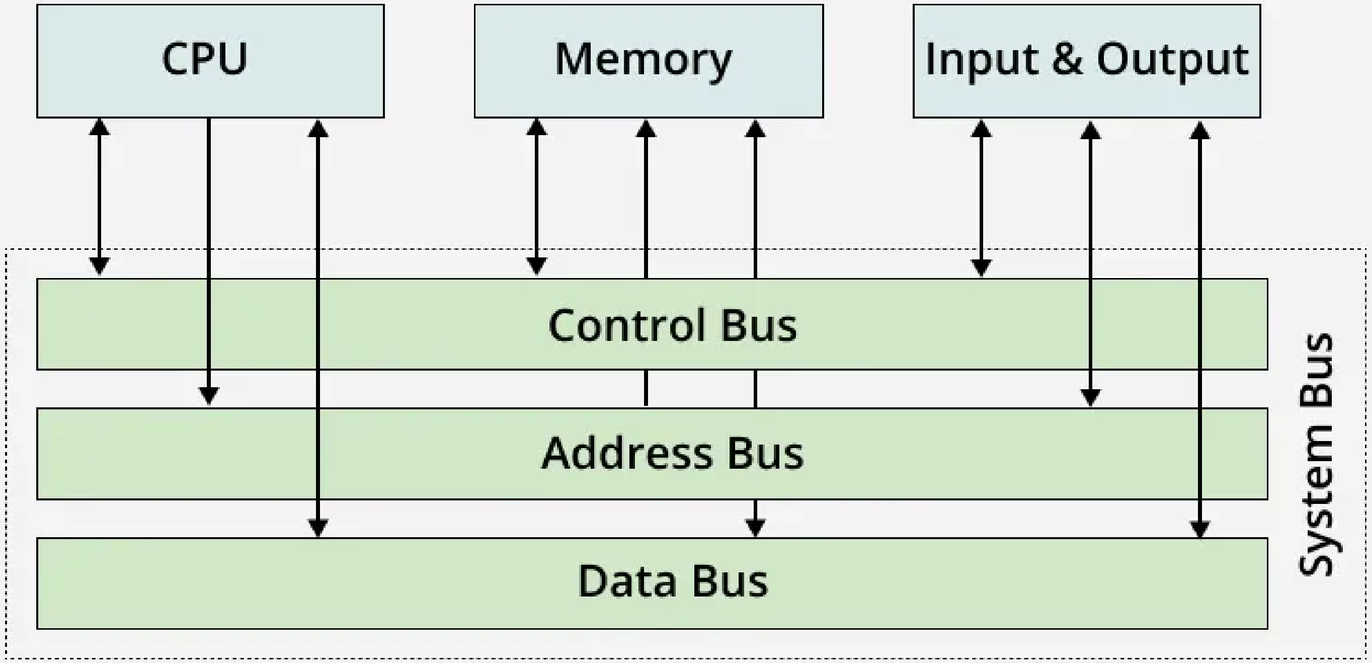
Patterson & Hennessy, Cap.5

Livello	Dove sta	Chi lo gestisce	Velocità	Dimensione
Registri	Dentro il core	Il compilatore / CPU	<<1ns (1 ciclo di clock)	pochissimi KB
L1 Cache	Dentro il core	Hardware automatico	<1ns	~32–128 KB
L2 Cache	Vicino al core	Hardware	2-10 ns	~256 KB–2 MB
L3 Cache	Condivisa dai core	Hardware	10-50 ns	MB–decine MB
RAM	Fuori dalla CPU	Memory controller	50-100 ns	GB

SYSTEM BUS (1/2)

Il system bus è un insieme di linee elettroniche che permette il trasferimento di informazioni tra la CPU, la memoria (RAM) e le periferiche I/O.

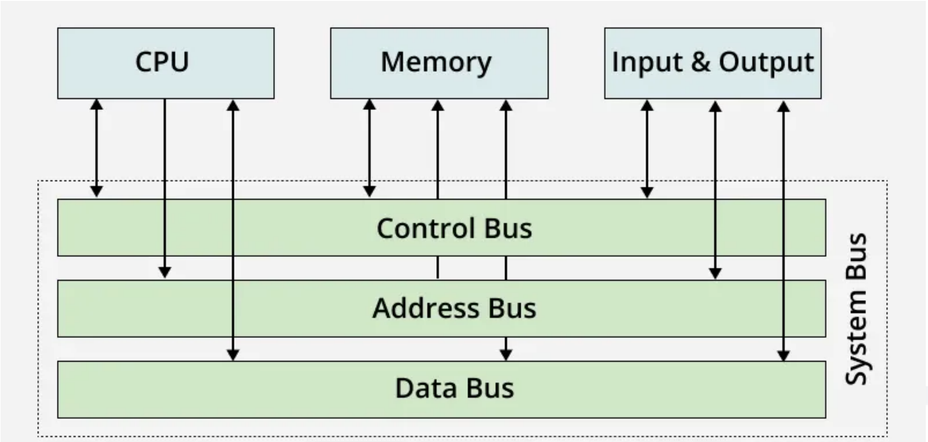




UNIVERSITÀ DEGLI STUDI DI TRIESTE

SYSTEM BUS (2/2)

Control Bus: porta i segnali tra la **CPU e memoria o I/O** (bidirezionale).
Non porta informazioni (indirizzi o dati), ma coordina il funzionamento della macchina tramite trasporto di *segnali di controllo e di stato* (Read/Write, clock, seleziona ingressi dei multiplexer, attiva la ALU).



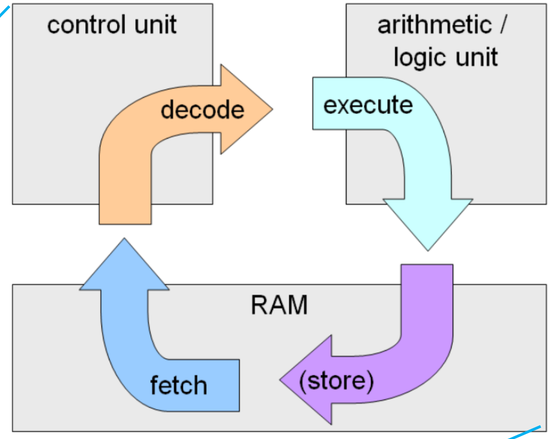
Address Bus: va da CPU a memoria o alle periferiche I/O (di solito unidirezionale). Indica quale indirizzo leggere/scrivere. Per farlo usa n bit, quindi può indirizzare fino a 2^n locazioni.

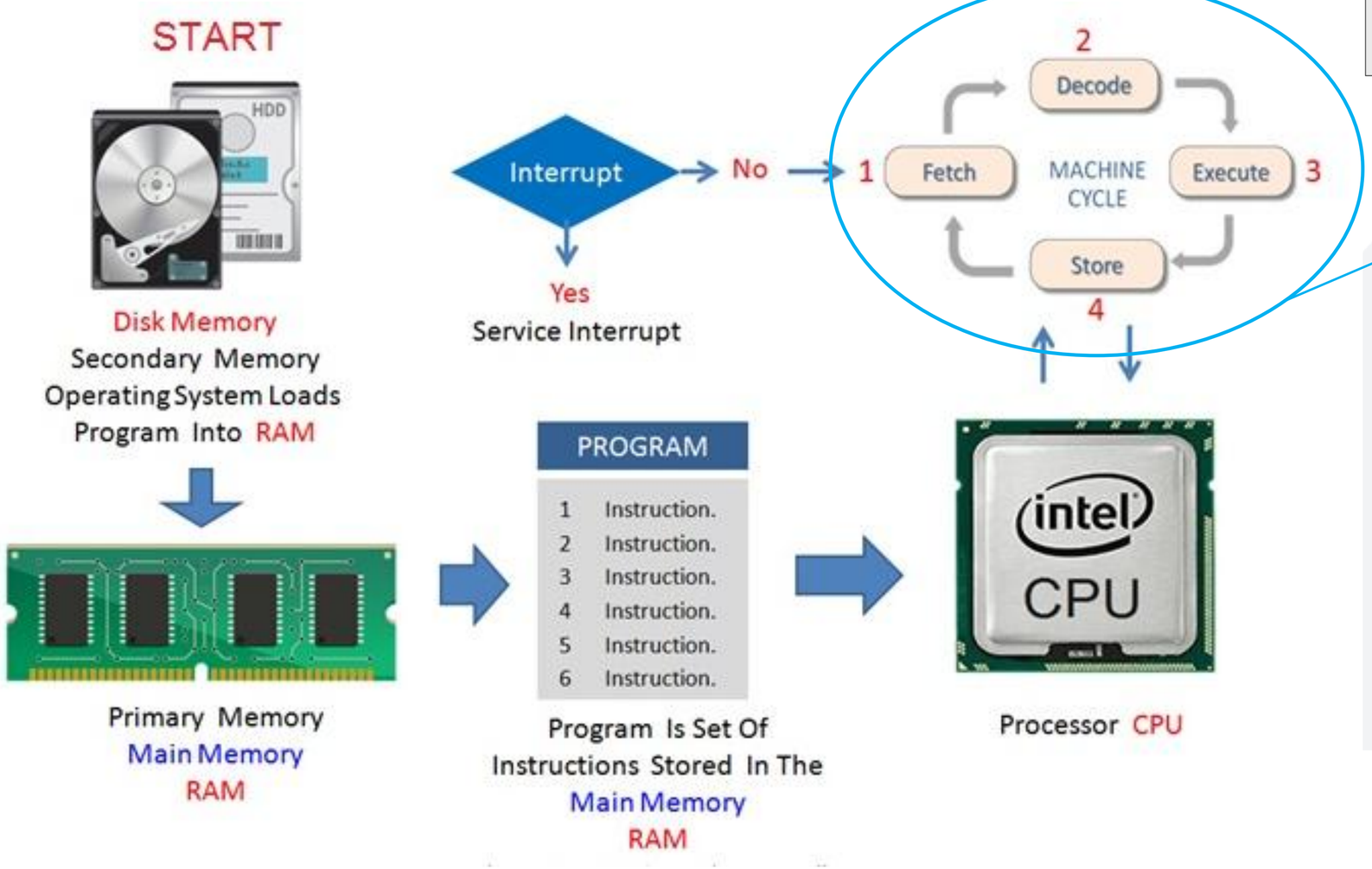
Può diventare bidirezionale solo in presenza di bus mastering / DMA, quando una periferica prende il controllo del bus e genera lei l'indirizzo per accedere alla memoria senza passare dalla CPU.

Data Bus: va dalla CPU alla memoria o alle periferiche I/O in modo bidirezionale. Trasporta *i dati effettivi*. Se porta m linee in parallelo (m bit), allora può trasferire fino a m bit in contemporanea.
Ad esempio, se la linea è di 8 bit, allora una word (32 bit) richiede 4 trasferimenti da 8 bit.

L'**address bus** dice *dove* andare, il **data bus** trasporta *i dati*, il **control bus** dice *cosa fare e quando*.

DATAPATH E
FASI DI ESECUZIONE DI UN'ISTRUZIONE

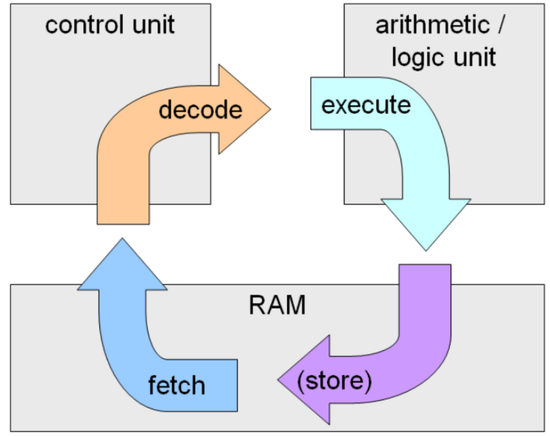




NOTA: nella memoria RAM si memorizzano **sia le istruzioni del programma che i dati dell'utente** durante l'esecuzione

UNIVERSITÀ DEGLI STUDI DI TRIESTE

MEMORIA E
FASI DI ESECUZIONE DI UN'ISTRUZIONE



Fetch

- La CPU legge l'istruzione dalla memoria (segmento .text)
- Usa il **Program Counter (PC)** come indirizzo
- vengono chiamati in cause PC + bus + memoria
- *si aggiorna il PC (per leggere la prossima istruzione)*

Decode

- Decodifica dettagli dell'istruzione (tipologia e *campi* - - > *si veda prossima lezione*)
- Lettura dei registri sorgente dal *Register file* (accesso ai registri vicini al processore)
- *Control Unit* genera segnali utili all'esecuzione dell'istruzione
- *Nessun accesso alla RAM*

Execute

- Operazione logico/aritmetiche: operazioni tra registri, nessun accesso alla RAM
- Lettura/scrittura RAM, accesso alla *Data memory* (segmento utente sulla RAM)
- Scrittura del risultato in un registro indicato dal Register file

ARCHITETTURA MIPS32

MIPS = Microprocessor without **I**nterlocked **P**ipeline **S**tages

- E’ architettura RISC
- 32 registri vicini alla CPU da 32 bit
- Memoria RAM indirizzabile con 32 bit
- Istruzioni semplici (32 bit)
  - ➢ Operazioni logico-aritmetiche
  - ➢ Alterazione del flusso di controllo (“salti”)
  - ➢ Manipolazioni di dati solo sui registri
  - ➢ Trasferimento di dati tra memoria e registri

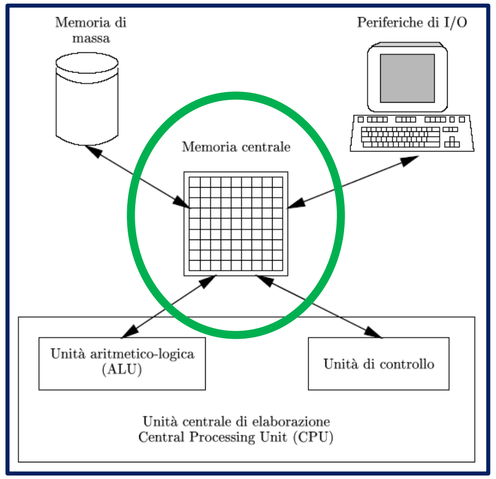
MIPS R3000A-compatible (R3051)
32bit a 33,8688 MHz della PlayStation

QtSpim

QtSpim è SIMULATORE di MIPS32
- SPIM = simulatore sw di MIPS
- versioni per diversi sistemi (Windows, Mac, Linux)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

MEMORIA RAM: CAPIENZA



MIPS32 è un'architettura a 32 bit, dunque può indirizzare al massimo:

2^{32} bytes = 4 294 967 296 bytes = **4 GB**

Quindi, teoricamente, un processore MIPS32 potrebbe accedere a **fino a 4 GB** di RAM.

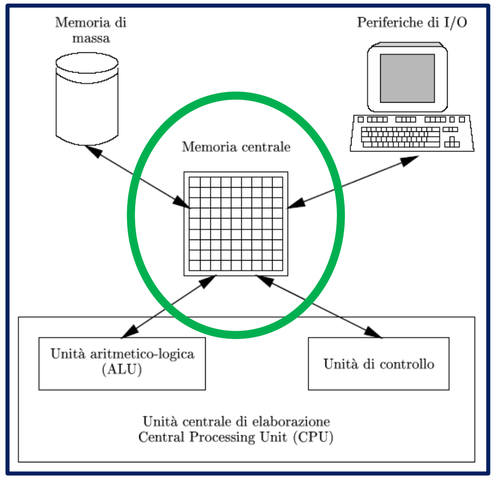
Nella pratica:

- Raramente vengono raggiunti i 4 GB massimi teorici per ragioni di costo e design.
- Sistemi più avanzati basati su MIPS32 possono avere decine o centinaia di megabyte di RAM.
- Esiste anche MIPS64, quindi può supportare memoria ben superiore a 4 GB (ARM64 o Intel x86-64).
- Se un dispositivo MIPS deve gestire grandi quantità di memoria (>4 GB), utilizzerà MIPS64, non MIPS32.
- Oggi MIPS (32 bit) non è tipicamente usata per computer desktop o smartphone consumer, ma è comune in ambito *embedded*, dove i requisiti di RAM sono bassi e la semplicità dell'architettura è un grande vantaggio.

Alcuni esempi reali:

- Sistemi embedded industriali
- Dispositivi IoT (Internet of Things)
- Router e dispositivi di rete basati su MIPS spesso hanno memoria nell'ordine di decine o centinaia di MB.
- Dispositivi didattici (come emulatori o simulatori MIPS usati per apprendimento) possono avere pochi megabyte.
- Set-top box (ricevitori satellitari, decoder TV)

MEMORIA RAM



Con MIPS32 possiamo «**indirizzare al byte**»

Ogni byte ha un suo indirizzo individuale

Potremmo inserire un byte di informazione alla volta, *però…*

bit											
	0	0	1	0	1	1	0	0	**0**	← byte	
	1	0	0	1	1	1	1	0	1		
	0	0	0	1	1	1	1	0	2		
	1	0	0	1	0	0	1	0	3		
	1	0	0	1	1	1	0	1	**4**	0x00400000	Casa* Alex: civico 0
	0	0	1	1	1	0	0	1	5	0x00400001	Casa Barbara: civico 1
	1	1	1	1	0	0	1	0	6	0x00400002	Casa Leo: civico 2
	0	0	1	1	0	0	1	1	7	0x00400003	Casa Penelope: civico 3
	1	0	0	0	1	1	1	1	**8**		:
	0	1	1	0	0	1	0	0	9		

*Tutte le case sono della stessa grandezza (8 bit)

I bit che ho inserito sono del tutto ***casuali***!

MEMORIA RAM

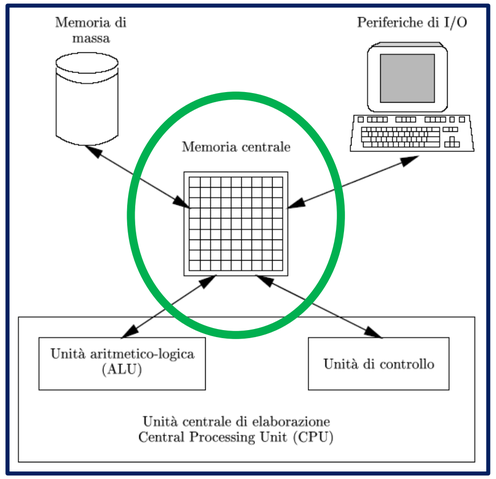
Però in realtà PER CONVENZIONE e per semplificare l'architettura, si segue un **ALLINEAMENTO A WORD**

Si scrivono le info (dati e istruzioni) **a gruppi di 4 byte** (=1word)

0	0	1	0	1	1	0	0	**0**
1	0	0	1	1	1	1	0	1
0	0	0	1	1	1	1	0	2
1	0	0	1	0	0	1	0	3
1	0	0	1	1	1	0	1	**4**
0	0	1	1	1	0	0	1	5
1	1	1	1	0	0	1	0	6
0	0	1	1	0	0	1	1	7
1	0	0	0	1	1	1	1	**8**
0	1	1	0	0	1	0	0	9

WORD

WORD



I bit che ho inserito sono del tutto ***casuali***!

Materiale per la lezione