



# Architettura degli elaboratori – II

## Turno 1 (cognomi A-G)

### Introduzione

Prof. Alberto Borghese  
Dipartimento di Informatica  
[alberto.borghese@unimi.it](mailto:alberto.borghese@unimi.it)

Università degli Studi di Milano

Riferimento sul Patterson 6th edition: capitolo 4.1-4.4.



## Sommario

- **Introduzione**
- Administratives
- La CPU a ciclo singolo



## Obiettivo di un'architettura



Elabora in modo adeguato un input per produrre l'output.

- Le unità di *ingresso* (tastiera, mouse, **rete**, **interfacce con dispositivi di acquisizione**, ecc.) permettono all'elaboratore di acquisire informazioni e richieste dall'ambiente esterno (non solo controllo diretto dall'utente).
- L'architettura di elaborazione.
- Le unità di *uscita* (terminale grafico, stampanti, rete, ecc.) consentono al calcolatore di comunicare i risultati ottenuti dall'elaborazione all'ambiente esterno.



Input ==> Elaborazione ==> Output



## Cosa fa un elaboratore?





- Algoritmi (sequenza di operazioni elementari → **istruzioni elementari**).
  - Calcoli (calcolatore).
  - **Operazioni logiche per controllo del flusso dell'elaborazione**
  - Trasferimento con la memoria
  - I/O



- **Programma** (Ada (Byron) Lovelace, 1830) = *Algoritmi in Software (calcolo della serie di Bernulli, linguaggio «Ada»)*

**Come lo fa?** *Hardware: calcoli e operazioni logiche vengono implementare da un'architettura di elaborazione.* Circuiti costituiti da **porte logiche**. Sono architetture **sequenziali** (una istruzione alla volta)

Terza rivoluzione della nostra civiltà: la rivoluzione agricola, la rivoluzione industriale e la rivoluzione dell'informatica (e delle biotecnologie).

## Architetture LOAD/STORE

- CPU + registri + memoria locale (dentro la CPU - cache)

↑

↑

↓

↓

load Ra Rb	Ra ← mem[Rb]	lw \$r2, cost(\$r1)
add Rd Ra Rc	Rd ← Ra + Rc	add \$r4,\$r2,\$r3
store Rd Rb	mem[Rb] ← Rd	sw \$r4, cost(\$r1)



Architetture *LOAD/STORE*: gli operandi dell'ALU possono provenire **soltanto** dai registri ad uso generale contenuti nella CPU e **non** possono provenire dalla memoria principale. Sono necessarie apposite istruzioni di:

- *caricamento (LOAD)* dei dati da memoria ai registri;
- *memorizzazione (STORE)* dei dati dai registri alla memoria.

- Memoria

Uno degli obiettivi del corso è capire come sia stato reso efficiente il sistema di memoria.

A.A. 2022-2023 5/57 <http://borghese.di.unimi.it/>

## Famiglie di architetture

- CISC (Complex Instruction Set Computer)
- RISC (Reduced Instruction Set Computer)

“Reduced” è un elemento di forza! (cf. rasoio di Occam: “the simplest the best” – “Entities should not be multiplied without necessity” - legge della parsomonia)

A.A. 2022-2023 6/57 <http://borghese.di.unimi.it/>



## CPU di tipo CISC



- Caratterizzate da elevata complessità delle istruzioni eseguibili ed elevato numero di istruzioni che costituiscono l'insieme delle istruzioni.
- Numerose modalità di indirizzamento per gli **operandi** dell'*ALU* che possono provenire da registri oppure da memoria, nel qual caso l'indirizzamento può essere diretto, indiretto, con registro base, ecc.
- Dimensione *variabile* delle istruzioni a seconda della modalità di indirizzamento di ogni operando  $\Rightarrow$  complessità di gestione della fase di prelievo o *fetch* in quanto a priori non è nota la lunghezza dell'istruzione da caricare.
- Elevata complessità della *CPU* stessa (cioè dell'hardware relativo) in termini degli elementi che la compongono con la conseguenza di rallentare i tempi di esecuzione delle operazioni. Elevata profondità dell'albero delle porte logiche, utilizzato per la decodifica.




## Utilizzo architettura Intel 80x86: le 10 istruzioni più frequenti




Rank	instruction	Integer Average Percent total executed
1	load	22%
2	conditional branch	20%
3	compare	16%
4	store	12%
5	add	8%
6	and	6%
7	sub	5%
8	move register-register	4%
9	call	1%
10	return	1%
	<b>Total</b>	<b>96%</b>

◦ **Simple instructions dominate instruction frequency**

$\Rightarrow$  RISC



## I diversi formati di istruzioni



Variable

...  





...

Fisso (MIPS)


Ibrido

Il formato fisso consente di massimizzare la velocità, il formato ibrido consente di minimizzare la lunghezza del codice.

A.A. 2022-2023 9/57 http://borghese.di.unimi.it/





## Architetture di tipo RISC



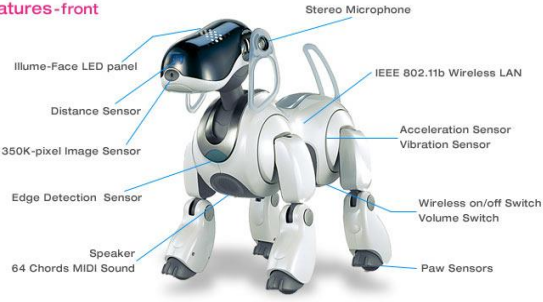
- Ispirate al principio di eseguire soltanto istruzioni semplici: le operazioni complesse vengono scomposte in una serie di istruzioni più semplici da eseguire in un ciclo base ridotto, con l'obiettivo di migliorare le prestazioni ottenibili dalle *CPU CISC*.
- Caratterizzate da istruzioni molto semplificate.
- Gli operandi dell'*ALU* possono provenire dai registri ma *non* dalla memoria. Per il trasferimento dei dati da memoria ai registri e viceversa si utilizzano delle apposite operazioni di caricamento (*load*) e di memorizzazione (*store*)  
 ⇒ **architetture load/store.**
- *CPU* relativamente semplice ⇒ si riducono i tempi di esecuzione delle singole istruzioni, che sono però meno potenti delle istruzioni *CISC*.
- Dimensione *fissa* delle istruzioni ⇒ più semplice la gestione della fase di prelievo (*fetch*) e della codifica delle istruzioni da eseguire
- **Architetture RISC sono architetture load/store native**

A.A. 2022-2023 10/57 http://borghese.di.unimi.it/

## Architettura MIPS



- Architettura MIPS appartiene alla famiglia delle architetture **RISC (Reduced Instruction Set Computer)** sviluppate dal 1980 in poi
  - Esempi: Sun Sparc, HP PA-RISC, IBM Power PC, DEC Alpha, Silicon Graphics, AIBO-Sony, **ARM** (e.g. Exynos 2100: tre core Cortex-A78 ad alte prestazioni e quattro Cortex-A55 a basso consumo energetico: Samsung S21), **Risc V** (recently sponsored by Google).
- Principali obiettivi delle architetture RISC:
  - Semplificare la progettazione dell'hardware e del compilatore
  - Massimizzare le prest ► Features-front
  - Minimizzare i costi



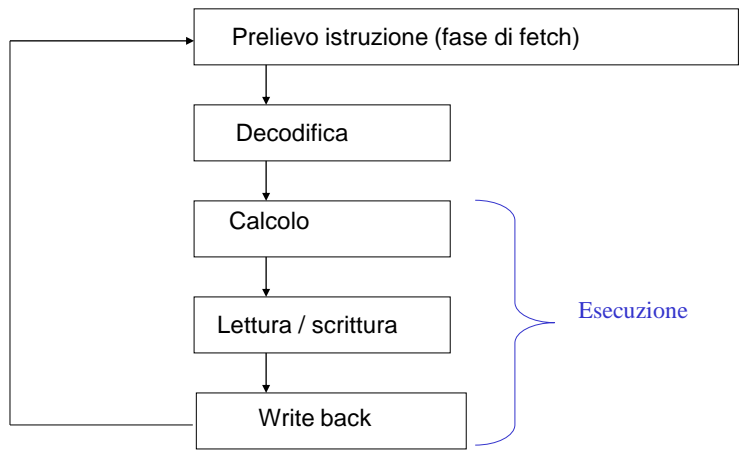
A.A. 2022-2023

11/57

<http://borghese.di.unimi.it/>

## Ciclo di esecuzione di un'istruzione



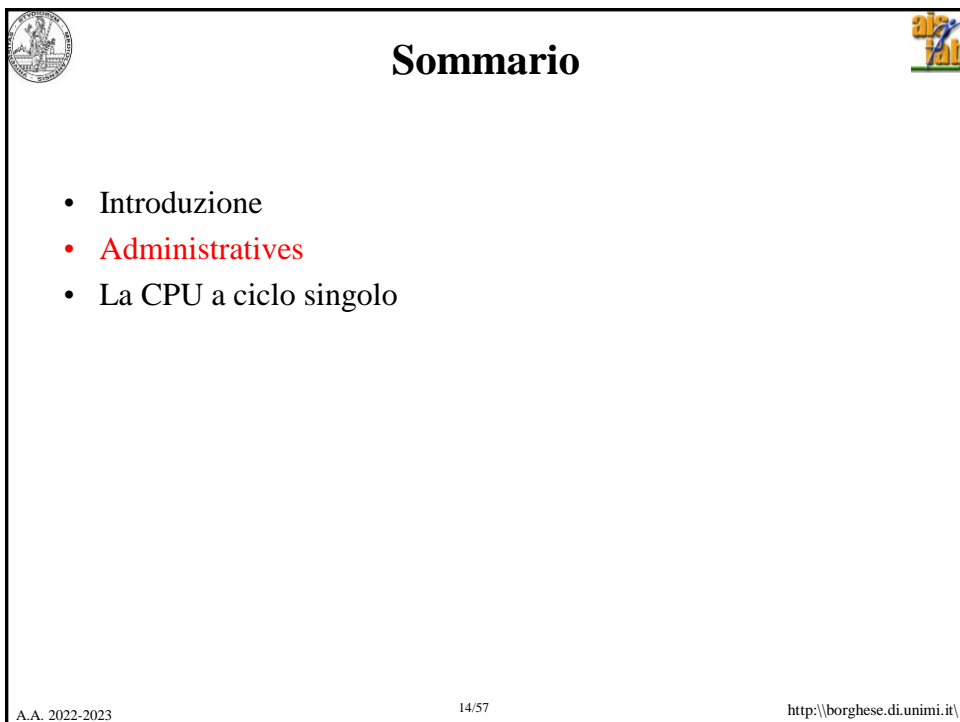
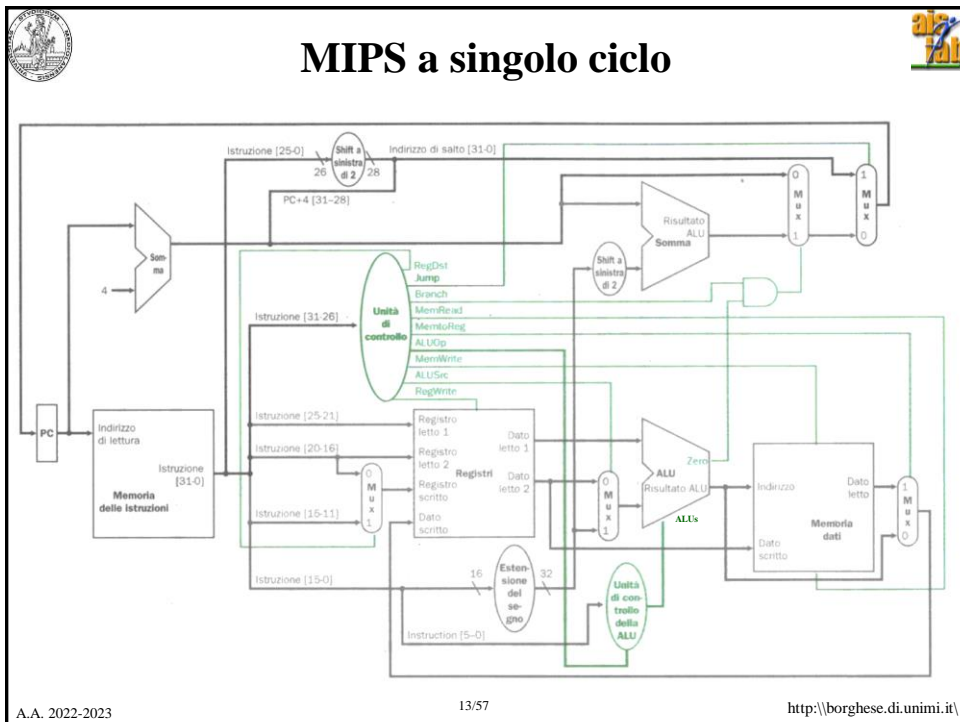
```

graph TD
    A[Prelievo istruzione (fase di fetch)] --> B[Decodifica]
    B --> C[Calcolo]
    C --> D[Lettura / scrittura]
    D --> E[Write back]
    E --> A
    subgraph Esecuzione
    C
    D
    E
    end
  
```

A.A. 2022-2023

12/57

<http://borghese.di.unimi.it/>





## Architetture II (6cfu)



**Turno 1 - Cognomi A-G** (turno 2 – Cognomi H-Z prof. Nicola Basilico)

**Docente: Prof. N. Alberto Borghese:** [alberto.borghese@unimi.it](mailto:alberto.borghese@unimi.it)

**Laboratorio Assembler (configurazione attuale):**

**Prof. Marco Tarini (Turno 1)**

Dott. Matteo Re (Turno 2)

**Orario e aule per la parte di teoria (se riaprirà la didattica in presenza):**

Lunedì                      Ore 10.30-12.30      Aula Magna, via Celoria 18

Mercoledì                  Ore 08.30-10.30      Aula Magna, via Celoria 18

Orario di ricevimento: su appuntamento.

Materiale sensibile delle lezioni caricato su Ariel.

**Strumento principale di contatto: email!**



## Programma



**Sito principale:**

[http://borghese.di.unimi.it/Teaching/Architettura\\_II/\\_Arch\\_II.html](http://borghese.di.unimi.it/Teaching/Architettura_II/_Arch_II.html)

**Programma (syllabus + slide):**

[http://borghese.di.unimi.it/Teaching/Architettura\\_II/Programma\\_2022-2023.html](http://borghese.di.unimi.it/Teaching/Architettura_II/Programma_2022-2023.html)

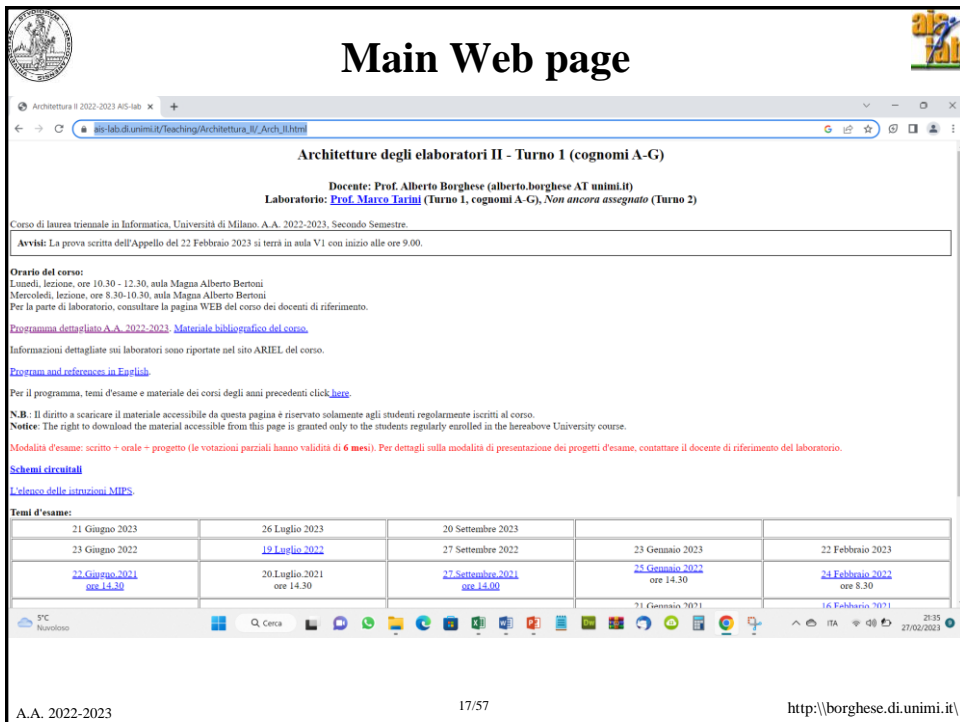
**Argomenti principali:**

CPU (avanzate)

Gerarchie di memoria e memoria virtuale

Interconnessioni





# Main Web page

Architetture degli elaboratori II - Turno I (cognomi A-G)

Docente: Prof. Alberto Borghese (alberto.borghese AT unimi.it)  
Laboratorio: Prof. Marco Tarini (Turno 1, cognomi A-G), Non ancora assegnato (Turno 2)

Corso di laurea triennale in Informatica, Università di Milano, A.A. 2022-2023, Secondo Semestre.

**Avvisi:** La prova scritta dell'Appello del 22 Febbraio 2023 si terrà in aula V1 con inizio alle ore 9.00.

**Orario del corso:**  
Lunedì, lezione, ore 10.30 - 12.30, aula Magna Alberto Bertoni  
Mercoledì, lezione, ore 8.30-10.30, aula Magna Alberto Bertoni  
Per la parte di laboratorio, consultare la pagina WEB del corso dei docenti di riferimento.

[Programma dettagliato A.A. 2022-2023](#) [Materiale bibliografico del corso](#).

Informazioni dettagliate sui laboratori sono riportate nel sito ARIEL del corso.

[Program and references in English](#).

Per il programma, temi d'esame e materiale dei corsi degli anni precedenti click [here](#).

**N.B.:** Il diritto a scaricare il materiale accessibile da questa pagina è riservato solamente agli studenti regolarmente iscritti al corso.  
**Notice:** The right to download the material accessible from this page is granted only to the students regularly enrolled in the hereabove University course.

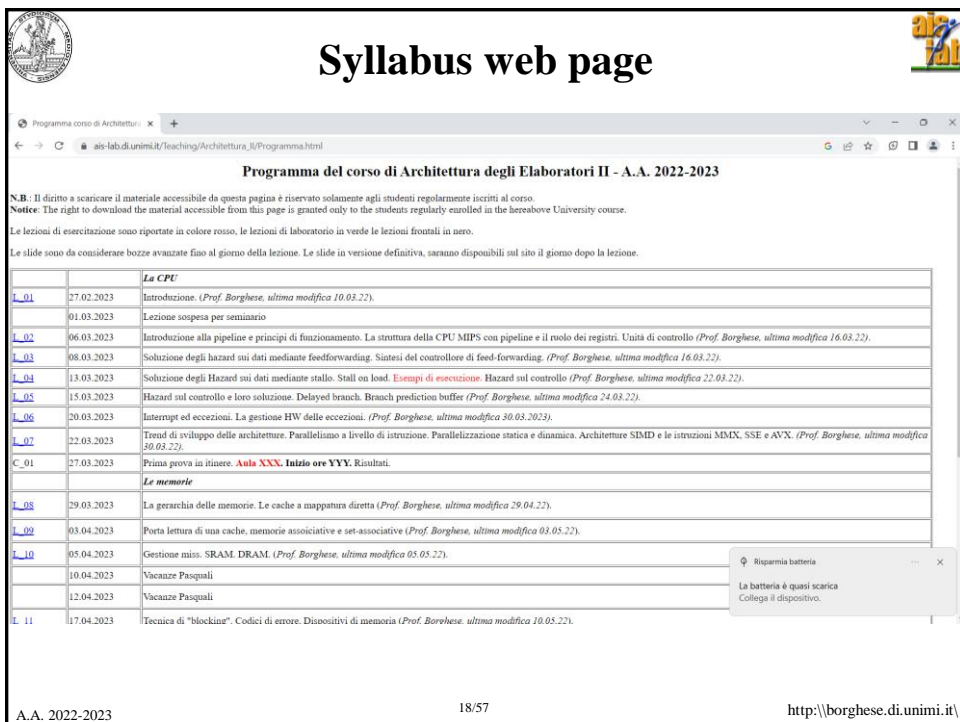
**Modalità d'esame:** scritto + orale + progetto (le votazioni parziali hanno validità di 6 mesi). Per dettagli sulla modalità di presentazione dei progetti d'esame, contattare il docente di riferimento del laboratorio.

[Schemi circuitali](#)  
[L'elenco delle istruzioni MIPS](#)

**Temi d'esame:**

21 Giugno 2023	26 Luglio 2023	20 Settembre 2023	23 Gennaio 2023	22 Febbraio 2023
23 Giugno 2022	<a href="#">19 Luglio 2022</a>	27 Settembre 2022	25 Gennaio 2022	
<a href="#">22 Giugno 2021</a> ore 14.30	20 Luglio 2021 ore 14.30	<a href="#">27 Settembre 2021</a> ore 14.00	ore 14.30	<a href="#">24 Febbraio 2022</a> ore 8.30

A.A. 2022-2023 17/57 http:\\borghese.di.unimi.it\



# Syllabus web page

Programma del corso di Architettura degli Elaboratori II - A.A. 2022-2023



**N.B.:** Il diritto a scaricare il materiale accessibile da questa pagina è riservato solamente agli studenti regolarmente iscritti al corso.  
**Notice:** The right to download the material accessible from this page is granted only to the students regularly enrolled in the hereabove University course.

Le lezioni di esercitazione sono riportate in colore rosso, le lezioni di laboratorio in verde le lezioni frontali in nero.

Le slide sono da considerare bozze avanzate fino al giorno della lezione. Le slide in versione definitiva, saranno disponibili sul sito il giorno dopo la lezione.

		<b>La CPU</b>
<a href="#">L_01</a>	27.02.2023	Introduzione. (Prof. Borghese, ultima modifica 10.03.22).
	01.03.2023	Lezione sospesa per seminario
<a href="#">L_02</a>	06.03.2023	Introduzione alla pipeline e principi di funzionamento. La struttura della CPU MIPS con pipeline e il ruolo dei registri. Unità di controllo (Prof. Borghese, ultima modifica 16.03.22).
<a href="#">L_03</a>	08.03.2023	Soluzione degli hazard sui dati mediante forwarding. Sintesi del controllore di feed-forwarding. (Prof. Borghese, ultima modifica 16.03.22).
<a href="#">L_04</a>	13.03.2023	Soluzione degli Hazard sui dati mediante stallo. Stall on load. <b>Esempi di esecuzione:</b> Hazard sul controllo (Prof. Borghese, ultima modifica 22.03.22).
<a href="#">L_05</a>	15.03.2023	Hazard sul controllo e loro soluzione. Delayed branch. Branch prediction buffer (Prof. Borghese, ultima modifica 24.03.22).
<a href="#">L_06</a>	20.03.2023	Interrupt ed eccezioni. La gestione HW delle eccezioni. (Prof. Borghese, ultima modifica 30.03.2023).
<a href="#">L_07</a>	22.03.2023	Temi di sviluppo delle architetture. Parallelismo a livello di istruzione. Parallelizzazione statica e dinamica. Architetture SIMD e le istruzioni MMX, SSE e AVX. (Prof. Borghese, ultima modifica 30.03.22).
<b>C_01</b>	27.03.2023	Prima prova in itinere. <b>Aula XXX. Inizio ore YYY.</b> Risultati.
		<b>Le memorie</b>
<a href="#">L_08</a>	29.03.2023	La gerarchia delle memorie. Le cache a mappatura diretta (Prof. Borghese, ultima modifica 29.04.22).
<a href="#">L_09</a>	03.04.2023	Porta lettura di una cache, memorie associative e set-associative (Prof. Borghese, ultima modifica 03.05.22).
<a href="#">L_10</a>	05.04.2023	Gestione main. SRAM. DRAM. (Prof. Borghese, ultima modifica 05.05.22).
	10.04.2023	Vacanze Pasquali
	12.04.2023	Vacanze Pasquali
<a href="#">L_11</a>	17.04.2023	Tecnica di "blockin". Codici di errore. Dispositivi di memoria (Prof. Borghese, ultima modifica 10.05.22).

A.A. 2022-2023 18/57 http:\\borghese.di.unimi.it\



## Esame

**Parte teorica (2/3 del voto).**

- Prova scritta + orale. Appelli ogni 1 / 2 / 3 mesi, al di fuori dal periodo delle lezioni.
- L'orale sarà in presenza e inizierà con la correzione della prova scritta.
- I compitini sostituiranno integralmente l'esame -> l'orale è facoltativo.
- Per sostenere l'esame occorre iscriversi sul SIFA all'**appello scritto**. Non è permesso portare il telefonino nell'aula dello scritto.

**Parte di laboratorio (1/3 del voto).**

A.A. 2022-2023 19/57 <http://borghese.di.unimi.it/>

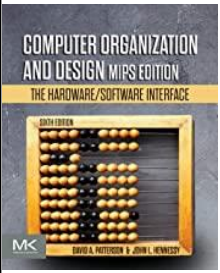
## Materiale didattico


[http://borghese.di.unimi.it/Teaching/Architettura\\_II/References.rtf](http://borghese.di.unimi.it/Teaching/Architettura_II/References.rtf)

**Testo di base (è disponibile sia in inglese che in italiano):**  
 “Computer Organization & Design: The Hardware/Software Interface (**MIPS edition**)”, D.A. Patterson and J.L. Hennessy, Morgan Kaufmann Publishers, **Sixth Edition**, 2018. Sono disponibili anche una versione RISC V e una versione ARM, che **non** sono state adottate. La sesta edizione presenta cambiamenti minimi rispetto alla quinta edizione.



Struttura e progetto dei calcolatori: l'interfaccia hardware-software, D.A. Patterson and J.L. Hennessy, Quinta edizione, Zanichelli, 2022 (Nota: la quinta edizione Zanichelli è la traduzione della sesta edizione inglese).

Per il laboratorio viene utilizzato il SW MARS di emulazione dell'assembler MIPS:  
<https://courses.missouristate.edu/KenVollmar/MARS/>





A.A. 2022-2023 20/57 INFORMATICA ZANICHELLI

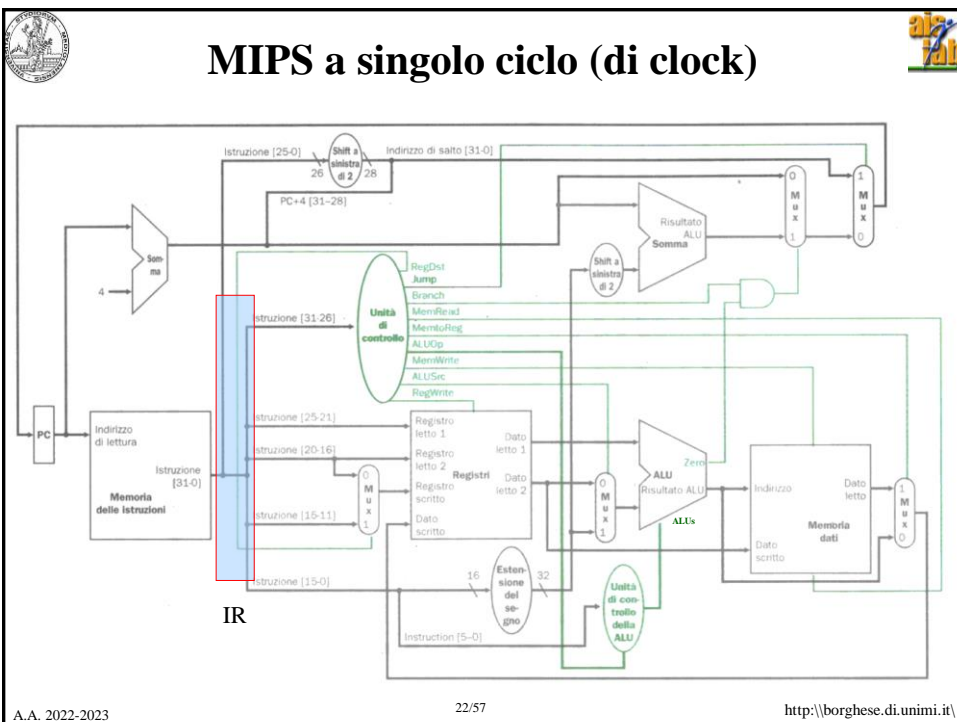
## Sommaro

- Introduzione
- Administrative
- **La CPU a ciclo singolo**

A.A. 2022-2023

21/57

<http://borghese.di.unimi.it/>





## I componenti di un'architettura



### CPU

- Banco di registri (*Register File*) ad accesso rapido, in cui memorizzare i dati di utilizzo più frequente. Il tempo di accesso ai registri è circa 10 volte più veloce del tempo di accesso alla memoria principale.
- Registro *Program counter (PC)*. Contiene l'indirizzo dell'istruzione corrente da aggiornare durante l'evoluzione del programma, in modo da prelevare dalla memoria la corretta sequenza di istruzione;
- Registro *Instruction Register (IR)*. Contiene l'istruzione in corso di esecuzione. Questo registro verrà utilizzato più avanti nelle architetture multi-ciclo.
- Unità per l'esecuzione delle operazioni aritmetico-logiche (*Arithmetic Logic Unit - ALU*). I dati forniti all'*ALU* possono provenire da registri oppure direttamente dalla memoria, a seconda delle modalità di indirizzamento previste;
- Unità aggiuntive per elaborazioni particolari come unità aritmetiche per dati in virgola mobile (*Floating Point Unit - FPU*), sommatore ausiliari, ecc.;
- **Unità di controllo**. Controlla il flusso e determina le operazioni di ciascun blocco.

### MEMORIA PRINCIPALE



## MIPS: Software conventions for Registers



0	zero	constant 0	16	s0	callee saves
1	at	reserved for assembler	...		(caller can clobber)
2	v0	expression evaluation &	23	s7	
3	v1	function results	24	t8	temporary (cont'd)
4	a0	arguments	25	t9	
5	a1		26	k0	reserved for OS kernel
6	a2		27	k1	
7	a3		28	gp	Pointer to global area
8	t0	temporary: caller saves	29	sp	Stack pointer
...		(callee can clobber)	30	fp	frame pointer (s8)
15	t7		31	ra	Return Address (HW)



## Tipi di istruzioni



- Le istruzioni comprese nel linguaggio macchina di ogni calcolatore possono essere classificate nelle seguenti quattro categorie:
  - *Istruzioni aritmetico-logiche;*
  - *Istruzioni di trasferimento da/verso la memoria (load/store);*
  - *Istruzioni di salto condizionato e non condizionato per il controllo del flusso di programma;*
  - Istruzioni di trasferimento in ingresso/uscita (I/O).



## Definizione di un'ISA (Instruction Set **Architecture**)



*Definizione del funzionamento: insieme delle istruzioni (interfaccia verso i linguaggi ad alto livello).*


- Tipologia di istruzioni.
- Meccanismo di funzionamento.

*Definizione del formato: codifica delle istruzioni (interfaccia verso l'HW).*


- Formato delle istruzioni.
- Suddivisione in gruppi omogenei dei bit che costituiscono l'istruzione.

Alcune domande:

- Come e dove si specifica il tipo di istruzione?
- Come e dove si specifica da dove vengono letti i dati?
- Come e dove si specifica dove si scrivono i dati prodotti?
- Come viene gestita la memoria in lettura e scrittura?
- Come vengono gestiti i salti?




## Codifica (modulare) delle istruzioni architettura




- Tutte le istruzioni MIPS hanno la **stessa dimensione (32 bit)** – Architettura RISC.
- Le istruzioni vengono suddivise in gruppi di bit funzionali (campi).
- I 32 bit hanno un significato diverso a seconda del formato (o tipo) di istruzione
  - il tipo di istruzione è riconosciuto in base al valore di alcuni bit (**6 bit**) più significativi (**codice operativo - OPCODE**)
- Le istruzioni MIPS sono di **3 tipi** (formati):
  - **Tipo R (register)** – Lavorano su **3 registri**.
    - Istruzioni aritmetico-logiche.
  - **Tipo I (immediate)** – Lavorano su **2 registri**. L'istruzione è suddivisa in un **gruppo di 16 bit contenenti informazioni + 16 bit riservati ad una costante**.
    - Istruzioni di accesso alla memoria o operazioni contenenti delle costanti.
  - **Tipo J (jump)** – Lavora **senza registri: codice operativo + indirizzo di salto**.
    - Istruzioni di salto incondizionato.

	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	Indirizzo / costante		
J	op	indirizzo				

A.A. 2022-2023
27/57
<http://borghese.di.unimi.it/>



## Istruzioni

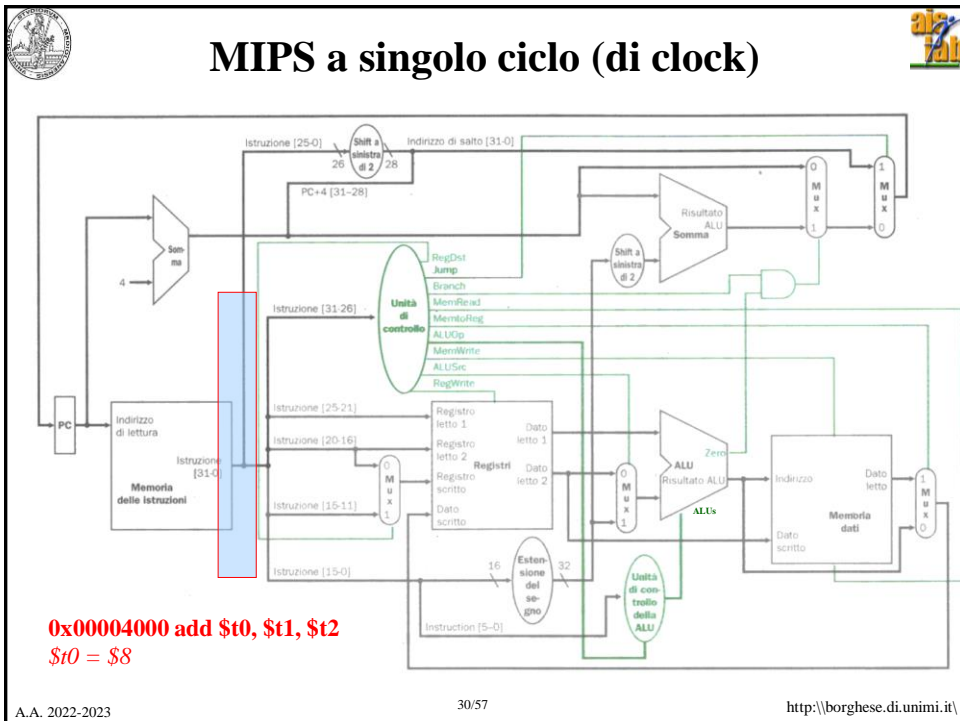


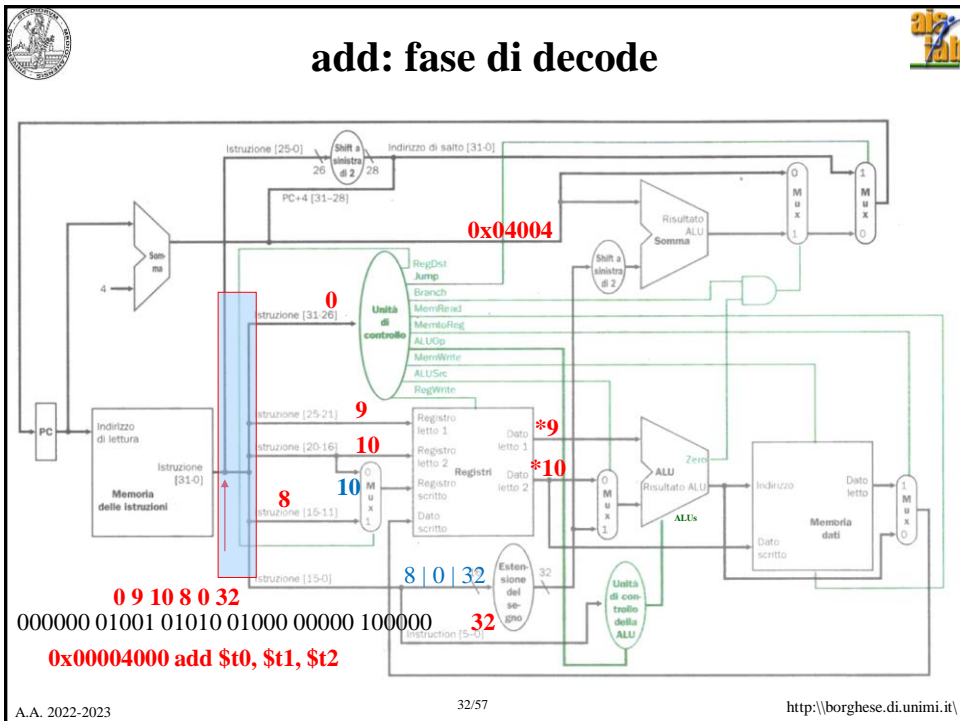
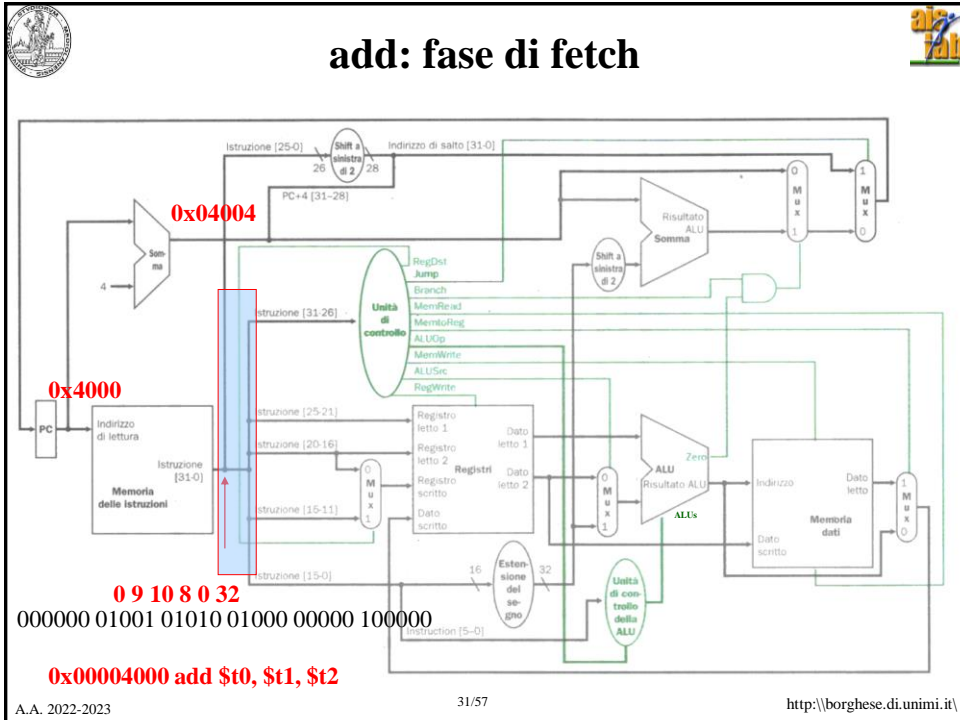
<code>add \$s1, \$s3, \$t0</code>	000000	10011	0100	10001	00000	100000
<code>beq \$s3, \$t0, -100</code>	000100	10011	01000	1111	1111	1110 0111
<code>lw \$t0, 32 (\$s3)</code>	100011	10011	01000	0000	0000	0010 0000
<code>sw \$t0, 32 (\$s3)</code>	101011	10011	01000	0000	0000	0010 0000
<code>addi \$t0, \$s3, 64</code>	001000	10011	01000	0000	0000	0100 0000
<code>j 0x80000</code>	000010	00 0000 0100 0000 0000 0000 0000				

↑

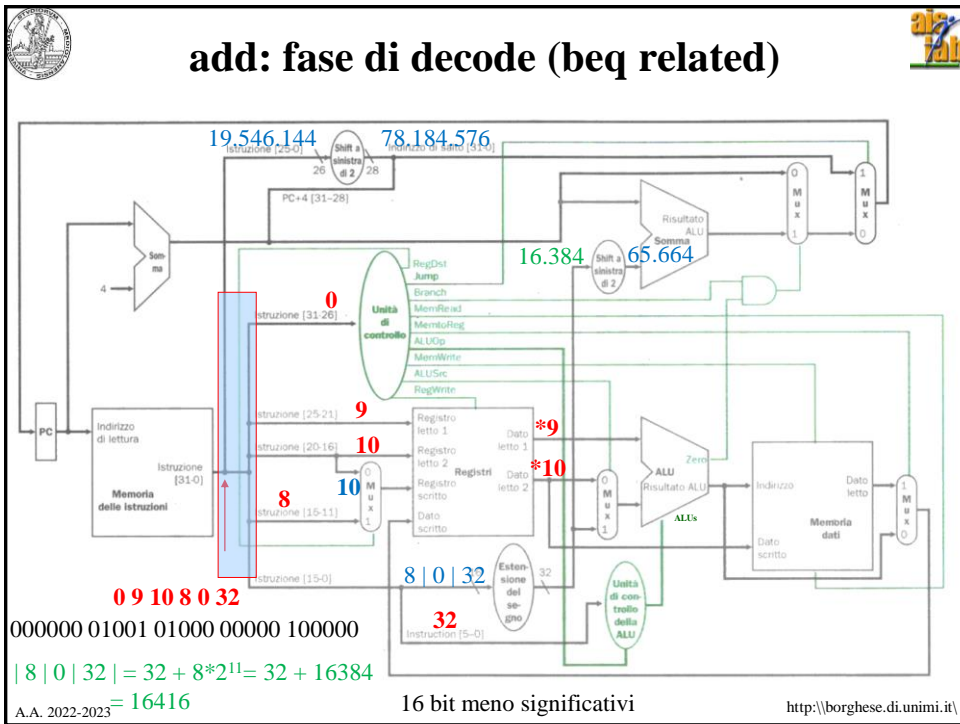
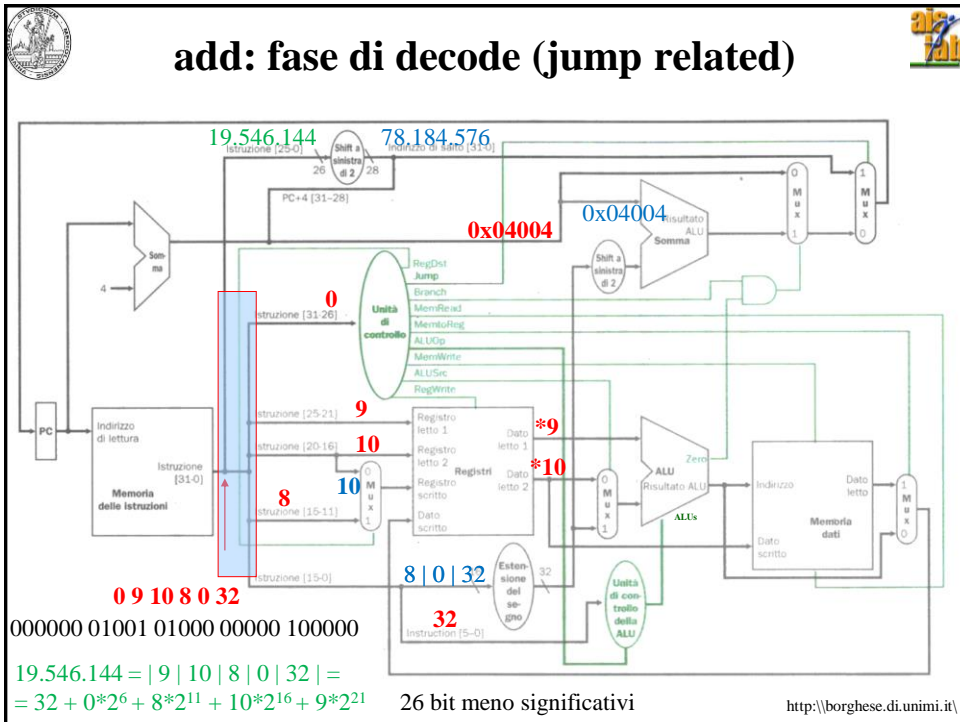
CodOp

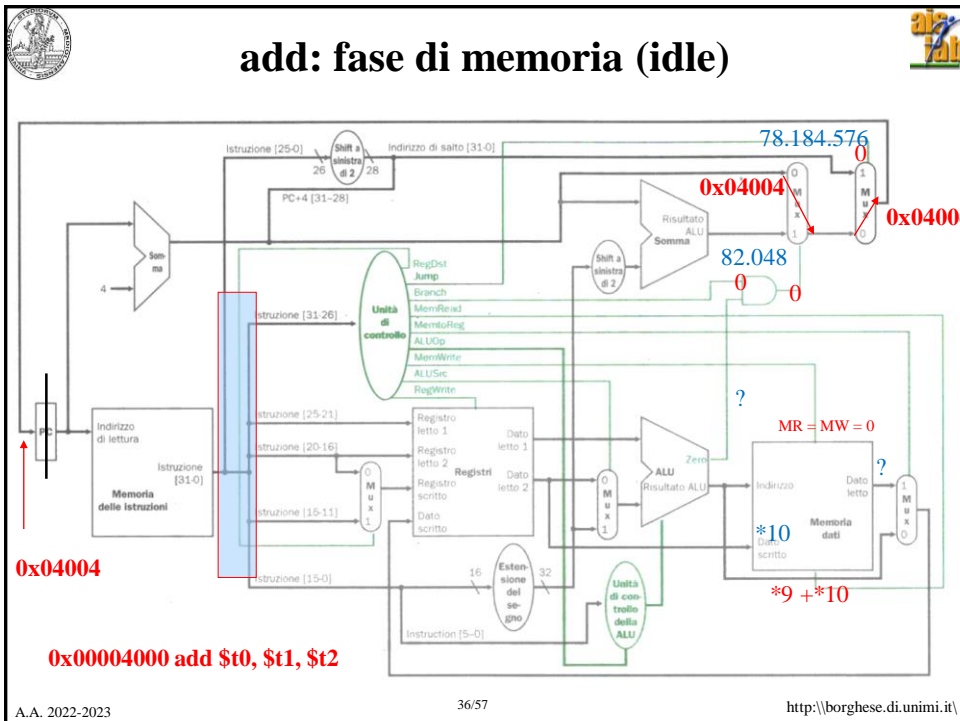
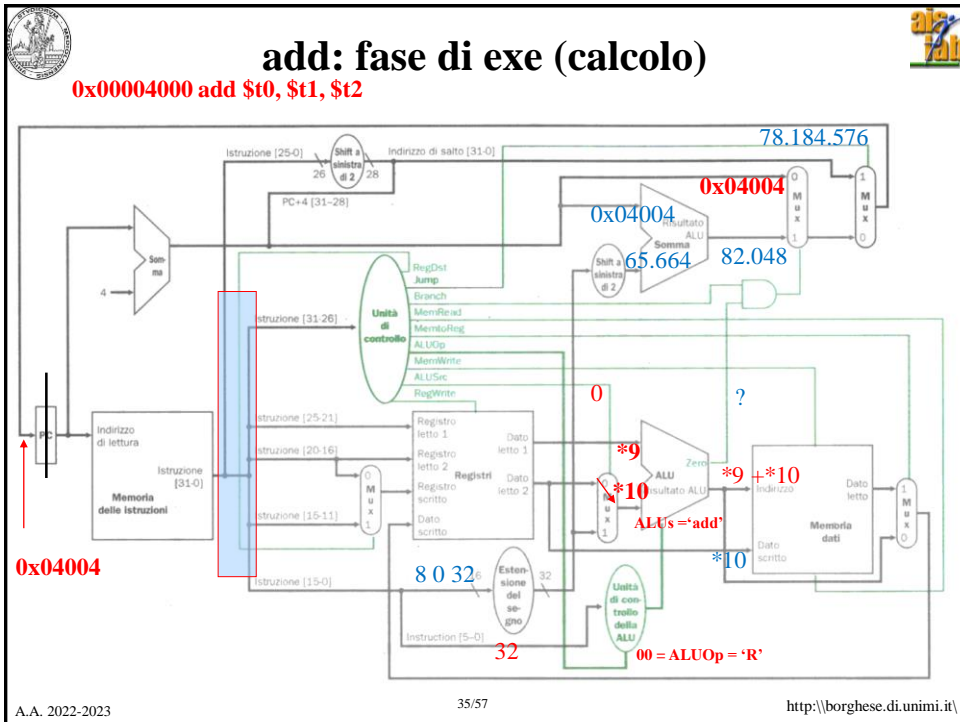
A.A. 2022-2023
28/57
<http://borghese.di.unimi.it/>

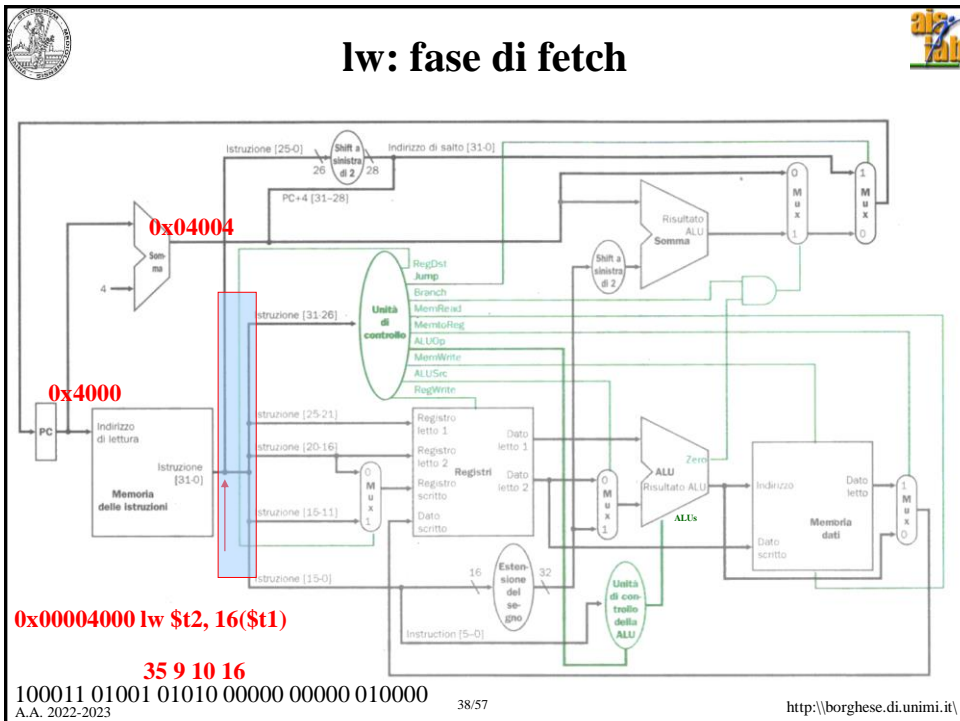
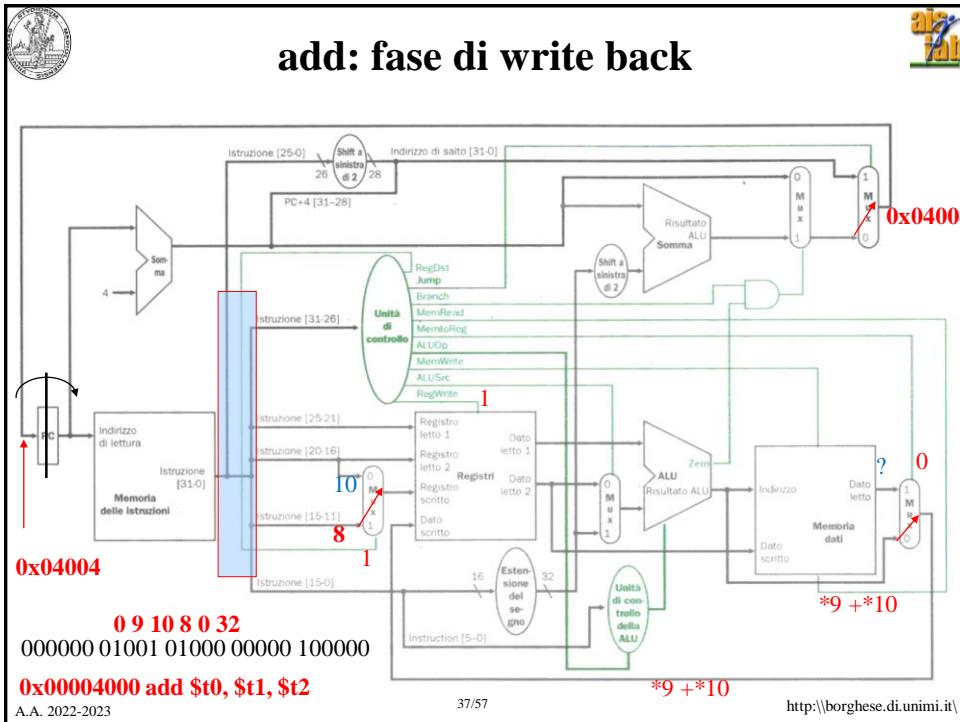


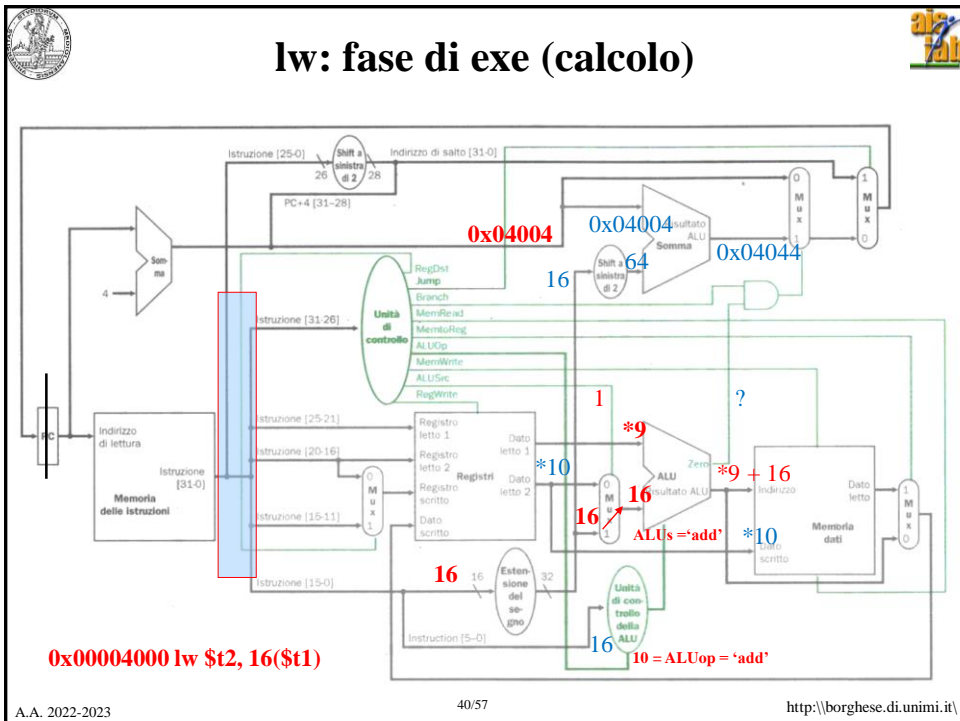
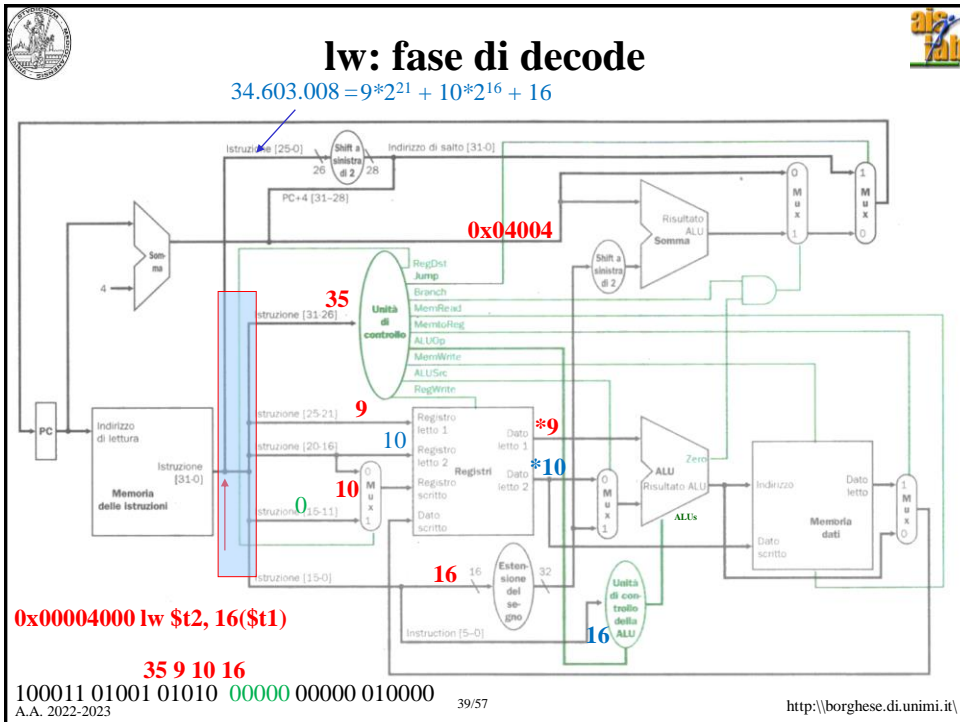


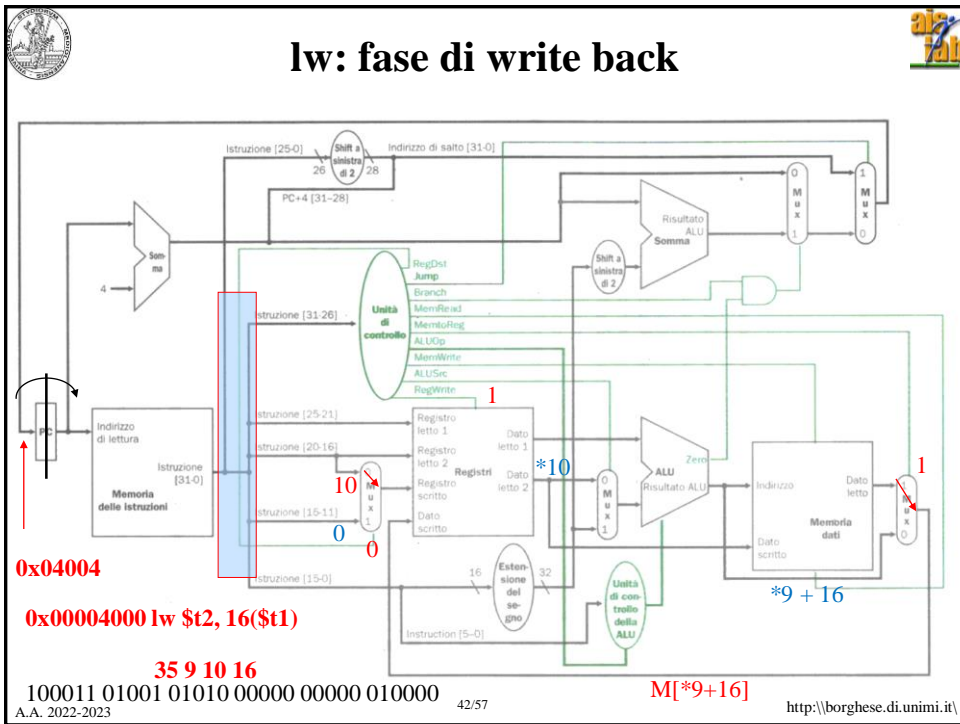
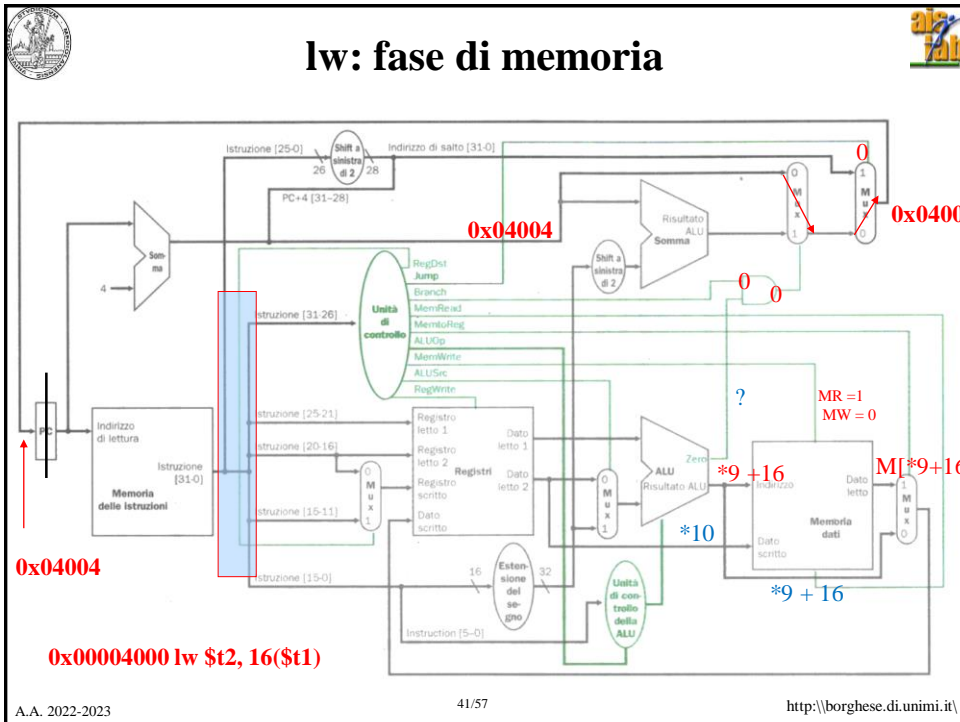


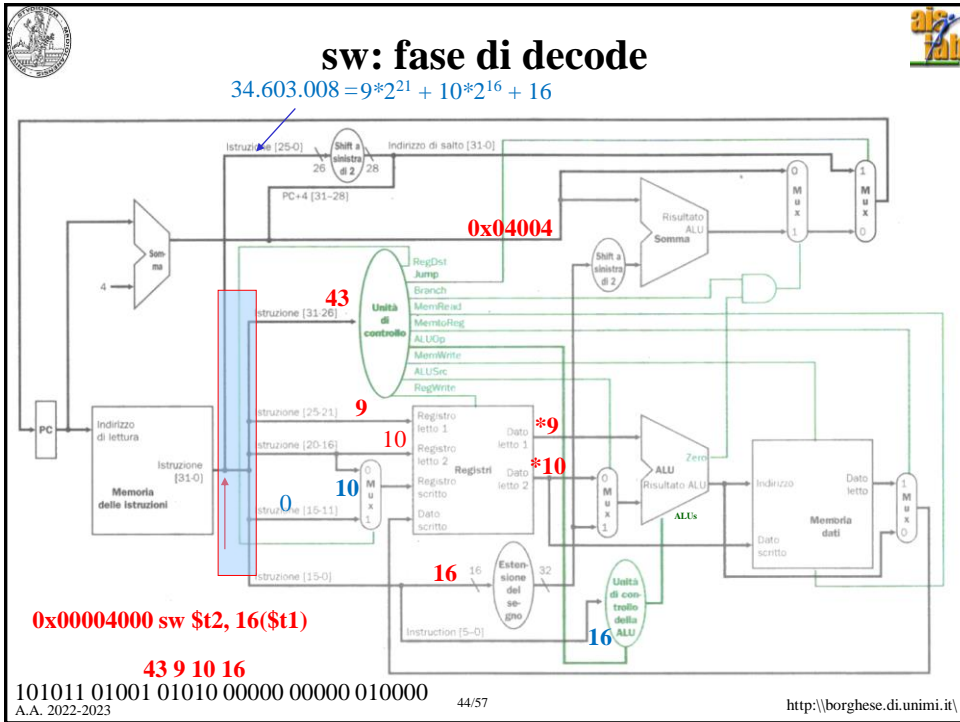
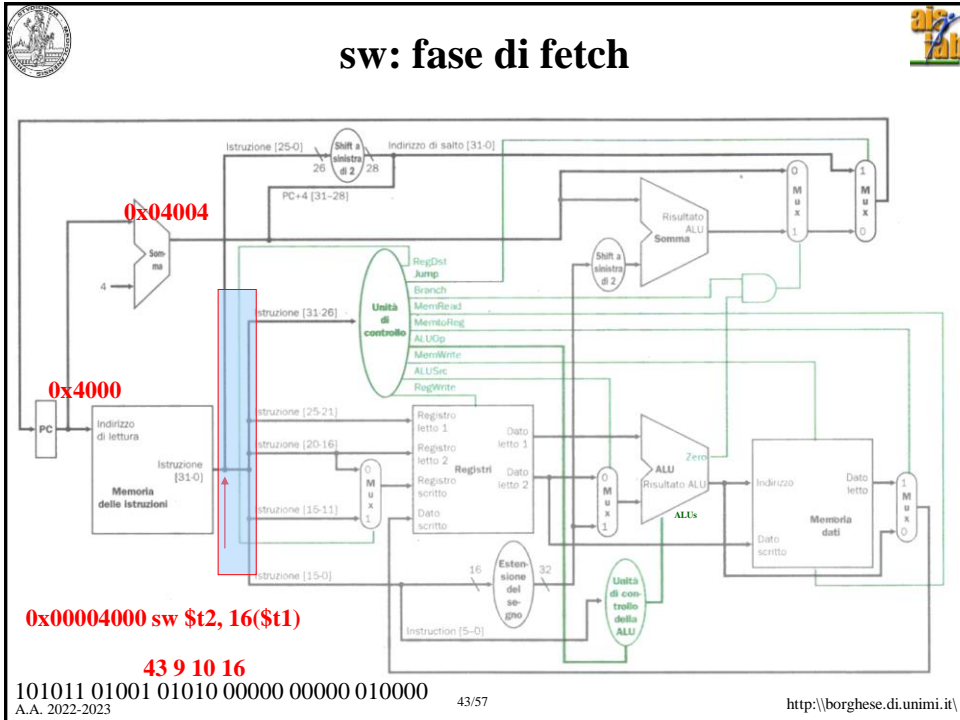




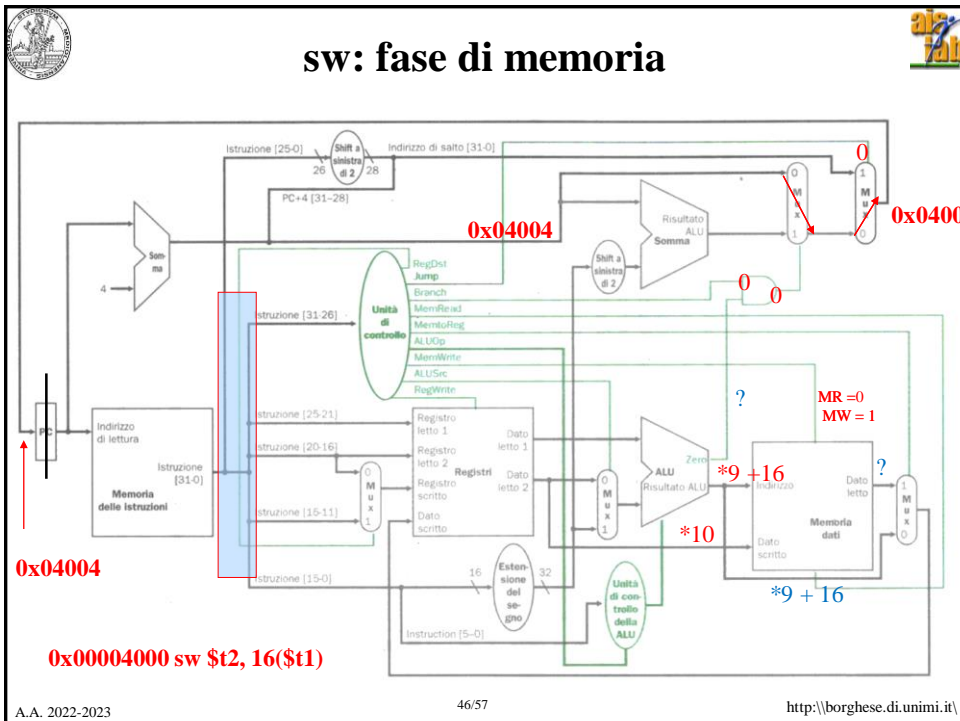
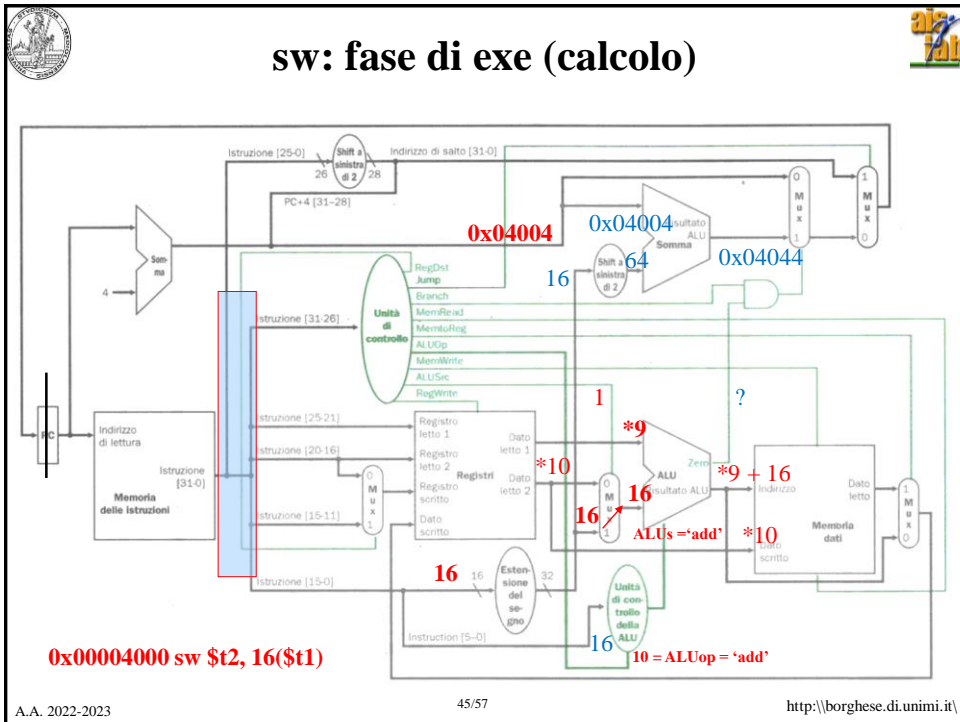


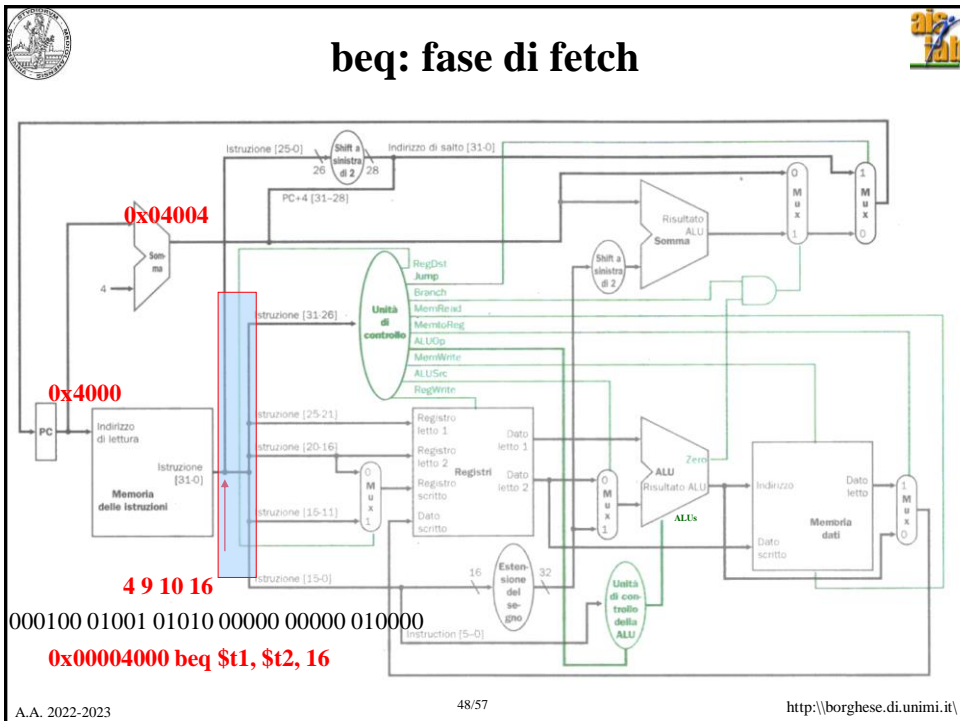
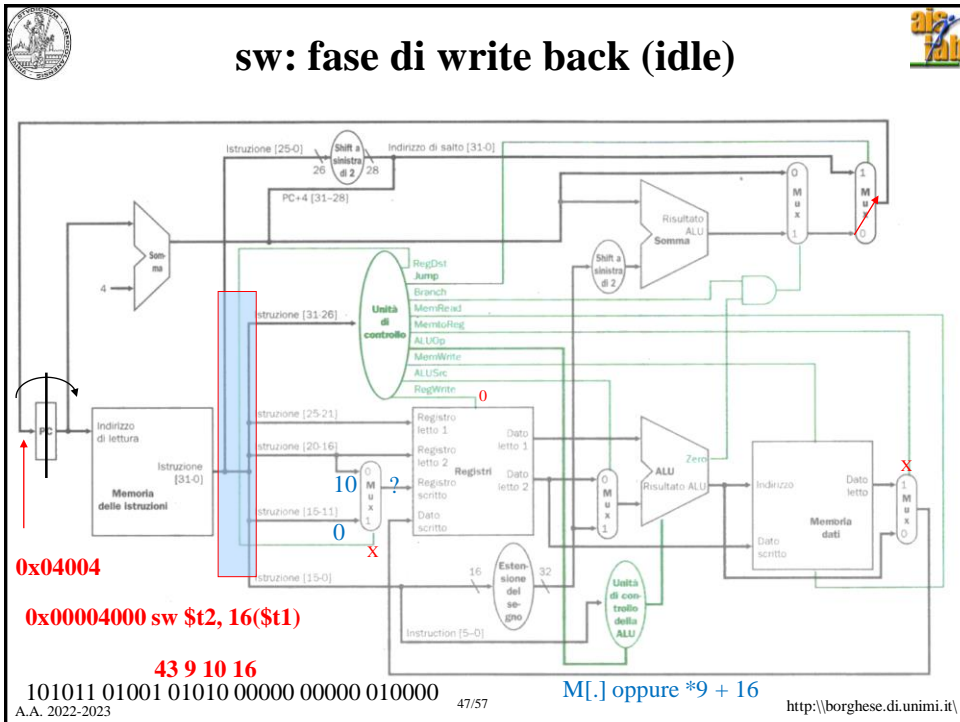




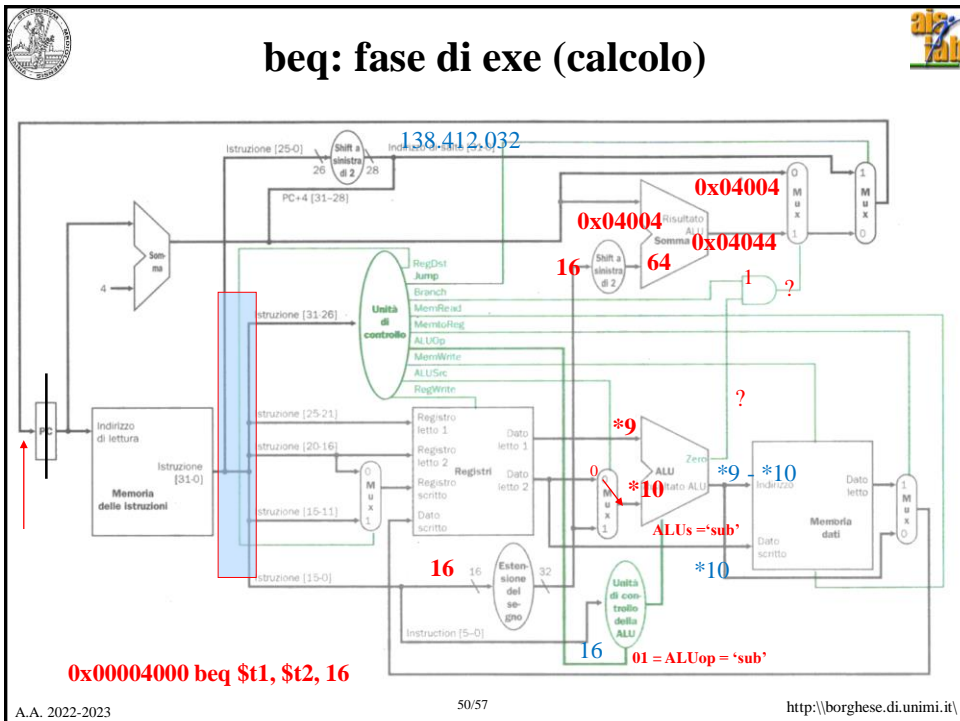
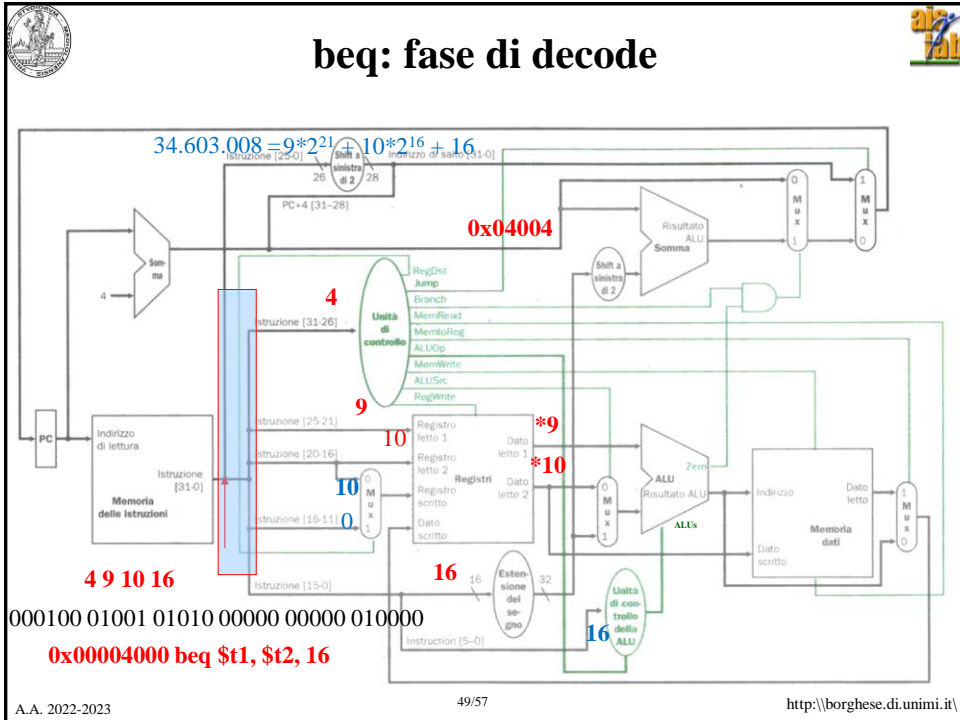


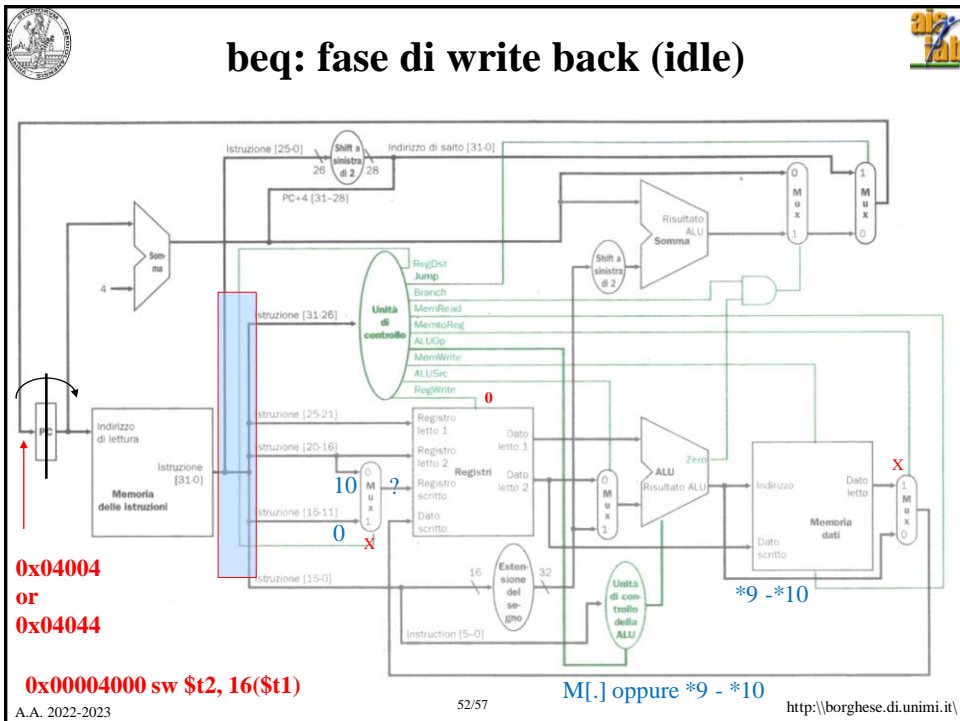
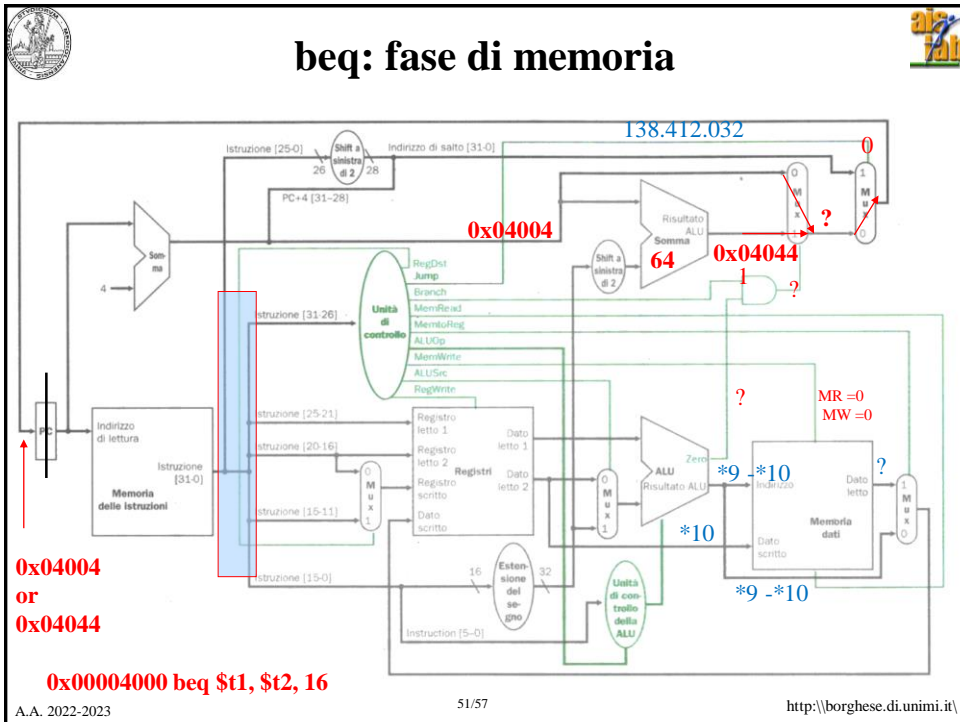


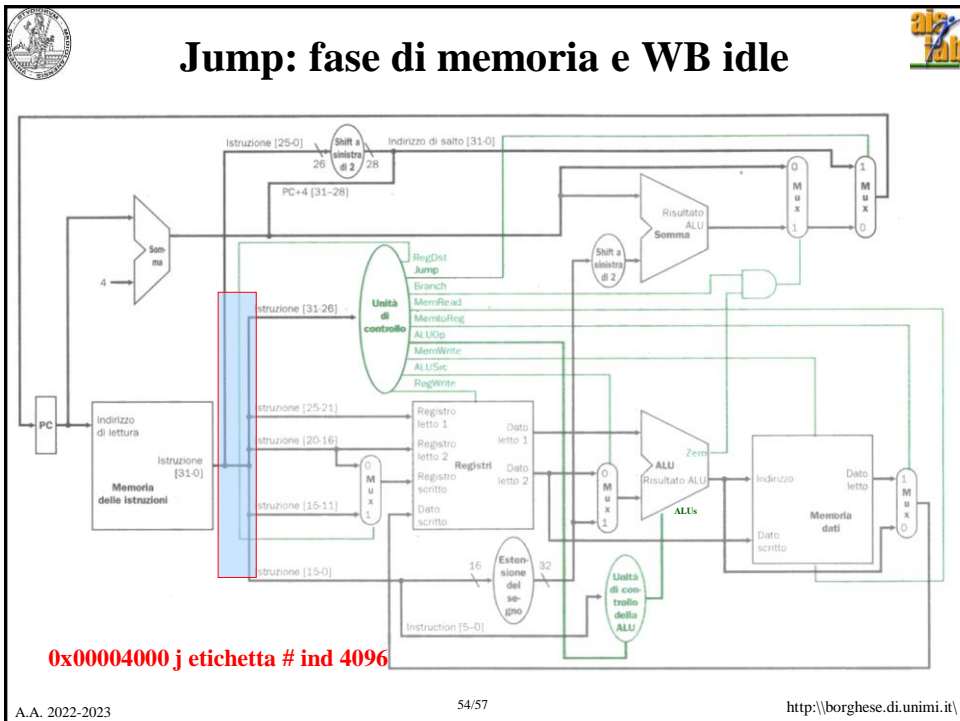
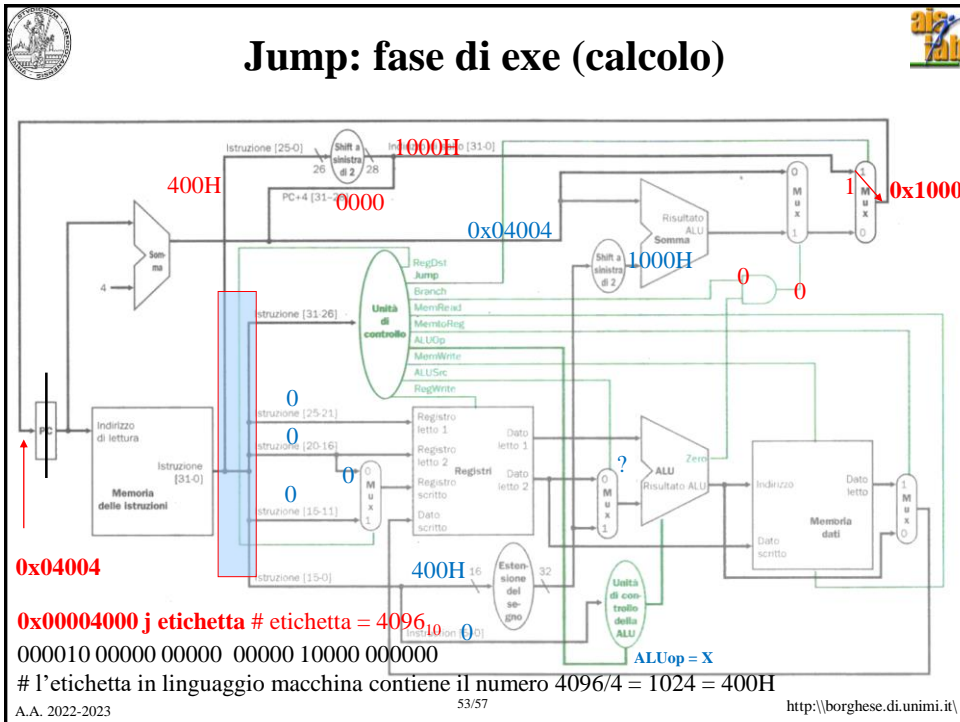














## Osservazioni

Il ciclo di esecuzione di un'istruzione si compie in un **unico** ciclo di clock.

↓

Ogni unità funzionale può essere utilizzata 1 sola volta.



↓

Duplicazione Memoria: Memoria dati e memoria istruzioni.  
 Triplicazione ALU: 3 ALU: 2 sommatori + 1 general purpose.

↓

Il periodo del clock è determinato dal cammino critico dell'istruzione più lunga: lw.  
 Utilizzare un clock per ogni fase di esecuzione porterebbe ad un tempo di esecuzione variabile per le diverse istruzioni: beq = 3 cicli di clock, aritmetiche e sw, 4 cicli di clock, lw 5 cicli di clock. Si può fare di meglio.

A.A. 2022-2023 55/57 http://borghese.di.unimi.it/

## Alcuni esercizi

Specificare il contenuto di tutti i bus durante l'esecuzione delle seguenti istruzioni:

```

0x00000400 or $s5, $t2, $t1
0x00000400 sw $s1, 1024($s0)
0x00010000 add $t4, $s5, $s1
0x00010000 addi $t1, $t4, 100
0x00000400 beq $s2, $s1, label

```

Si supponga label l'istruzione all'indirizzo 0x000410 e che il salto venga preso.

A.A. 2022-2023 56/57 http://borghese.di.unimi.it/



## Sommario



- Introduzione
- Administrative
- La CPU a ciclo singolo