


Architettura degli elaboratori - II
Introduzione

Prof. Alberto Borghese
Dipartimento di Informatica
borghese@di.unimi.it

Università degli Studi di Milano

Riferimento sul Patterson 5th edition: capitolo 4.1-4.4.


Introduzione alla CPU

- Introduzione
- Administratives
- La CPU a ciclo singolo

Obbiettivo 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 al calcolatore di acquisire informazioni dall'ambiente esterno.
- 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.

Cosa fa un elaboratore?

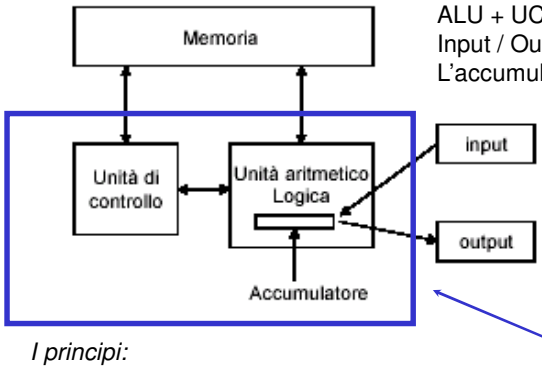
- Algoritmi (sequenza di **istruzioni**).
  Calcoli (calcolatore).
  Operazioni logiche (elaboratore).
- Programma (Ada Lovelace, 1830) = *Algoritmi in Software.*

Come lo fa? *Hardware: le istruzione vengono eseguite dall'hardware.*

Input ==> Elaborazione ==> Output

- Terza rivoluzione della nostra civiltà: la rivoluzione agricola, la rivoluzione industriale e la rivoluzione dell'informatica.

Architettura di Von Neumann



ALU + UC sono incorporate nella CPU.
Input / Output vengono in realtà trasferiti via bus
L'accumulatore è un possibile tipo di architettura

I principi:
• I dati e le istruzioni sono memorizzate in una memoria read/write.
• Il contenuto della memoria può essere recuperato in base alla sua posizione, e non è funzione del tipo di dato.
• L'esecuzione procede sequenzialmente da un'istruzione alla seguente.

•Gia' vista e modificata (evoluzione nel tempo).

Alcuni tipi di architetture

Accumulator (1 register = 1 indirizzo di memoria).

1 address	add A	acc ← acc + mem[A]
1+x address	addx A	acc ← acc + mem[A + x]

Stack (posso operare solo sui dati in cima allo stack):

0 address	add	tos ← tos + next

General Purpose Register (tanti diversi indirizzi di memoria quanti sono i registri, indirizzamento indiretto):

2 address	add A B	EA(A) ← EA(A) + EA(B)
3 address	add A B C	EA(A) ← EA(B) + EA(C)

Indirizzamento misto (registro, stack, ….)

Load/Store (posso operare solamente sui dati contenuti nei registri. Devo prima caricarli dalla memoria).

3 address	load Ra Rb	Ra ← mem[Rb]
	add Rd Ra Rc	Rd ← Ra + Rc
	store Rd Rb	mem[Rb] ← Rd

Architetture LOAD/STORE

- Il numero dei registri ad uso generale (ad esempio 32 registri da 32 bit ciascuno) non è sufficientemente grande da consentire di memorizzare tutte le variabili di un programma ⇒ ad ogni variabile viene assegnata una locazione di memoria nella quale trasferire il contenuto del registro quando questo deve essere utilizzato per contenere un'altra variabile.

- *Architetture LOAD/STORE:* gli operandi dell'ALU possono provenire soltanto dai registri ad uso generale contenuti nella CPU e **non** possono provenire dalla memoria. Sono necessarie apposite istruzioni di:
  - *caricamento (LOAD)* dei dati da memoria ai registri;
  - *memorizzazione (STORE)* dei dati dai registri alla memoria.

  Vedremo quando parleremo di memoria in che modo questa architettura può essere particolarmente efficiente.

CPU di tipo CISC (*Complex Instruction Set Computer*)

- 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 ⇒ complessità di gestione della fase di prelievo o *fetch* in quanto a priori non è nota la lunghezza dell'istruzione da caricare.

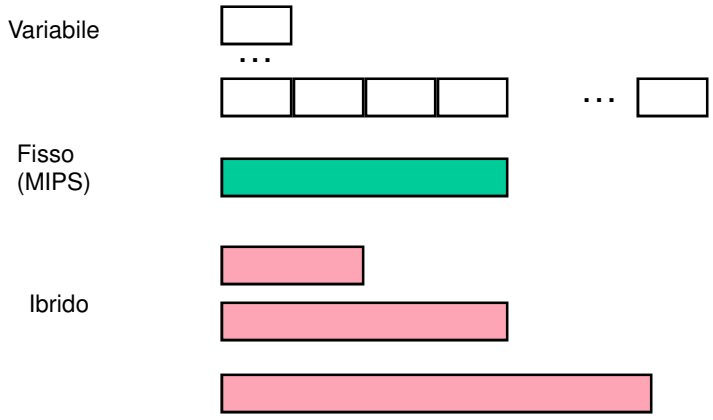
- Elevata complessità della *CPU* stessa (cioè dell'hardware relativo) in termini degli elementi che la compongono con la conseguenza di rallentare i tempi di esecuzione delle operazioni. Elevata profondità dell'albero delle porte logiche, utilizzato per la decodifica.

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%
	Total	96%

° Simple instructions dominate instruction frequency => RISC

I diversi formati di istruzioni



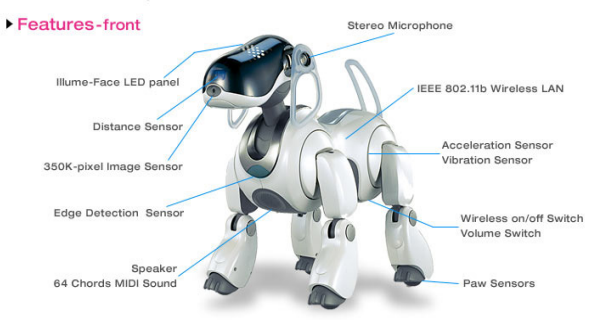
Il formato fisso consente di massimizzare la velocità, il formato ibrido consente di minimizzare la lunghezza del codice.

Architetture di tipo RISC (*Reduced Instruction Set Computer*)

- 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

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.
- Principali obiettivi delle architetture RISC:
  - Semplificare la progettazione dell'hardware e del compilatore
  - Massimizzare le prestazioni
  - Minimizzare i costi



Simulatore MIPS

- **SPIM: A MIPS R2000/R3000 Simulator : PCSPIM version 6.3**
- http://www.cs.wisc.edu/~larus/spim.html

Oppure da:

- http://borghese.di.unimi.it/Teaching/Architettura_II/_Arch_II.html

- Piattaforme:
  - Unix or Linux system
  - Microsoft Windows (Windows 95, 98, NT, 2000, XP)
  - Microsoft DOS

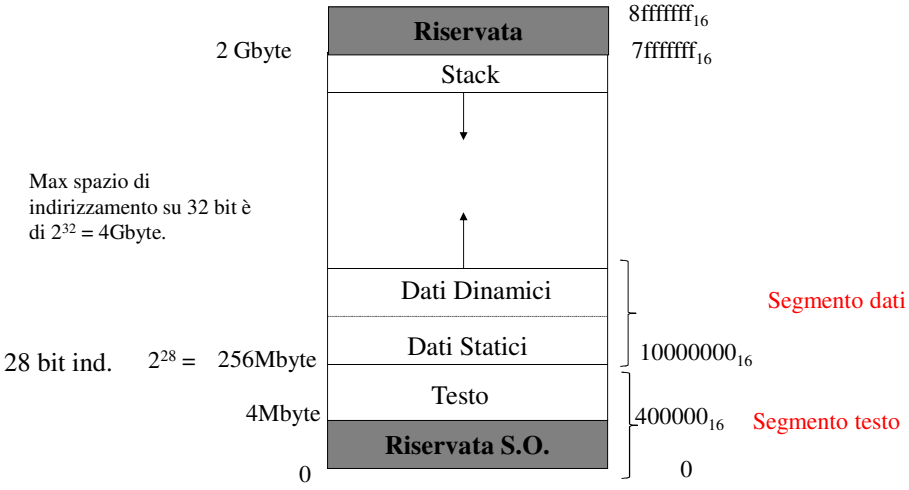
I componenti di un'architettura

CPU

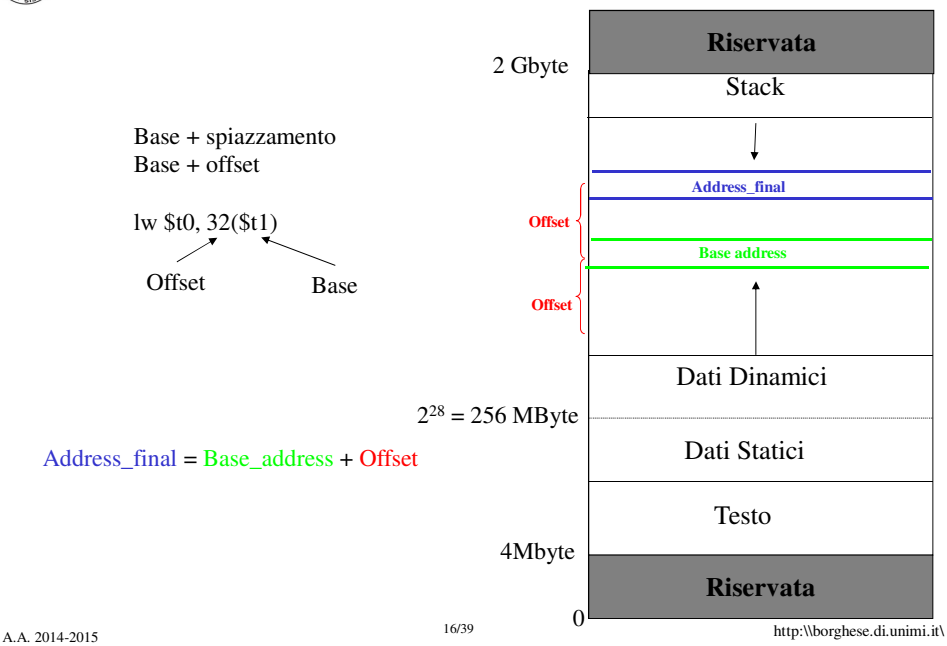
- Banco di registri *(Register File)* ad accesso rapido, in cui memorizzare i dati di utilizzo più frequente. Il tempo di accesso ai registri è circa 10 volte più veloce del tempo di accesso alla memoria principale.
- Registro *Program counter (PC).* Contiene l'indirizzo dell'istruzione corrente da aggiornare 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)*, sommatori ausiliari, ecc.;
- ***Unità di controllo***. Controlla il flusso e determina le operazioni di ciascun blocco.

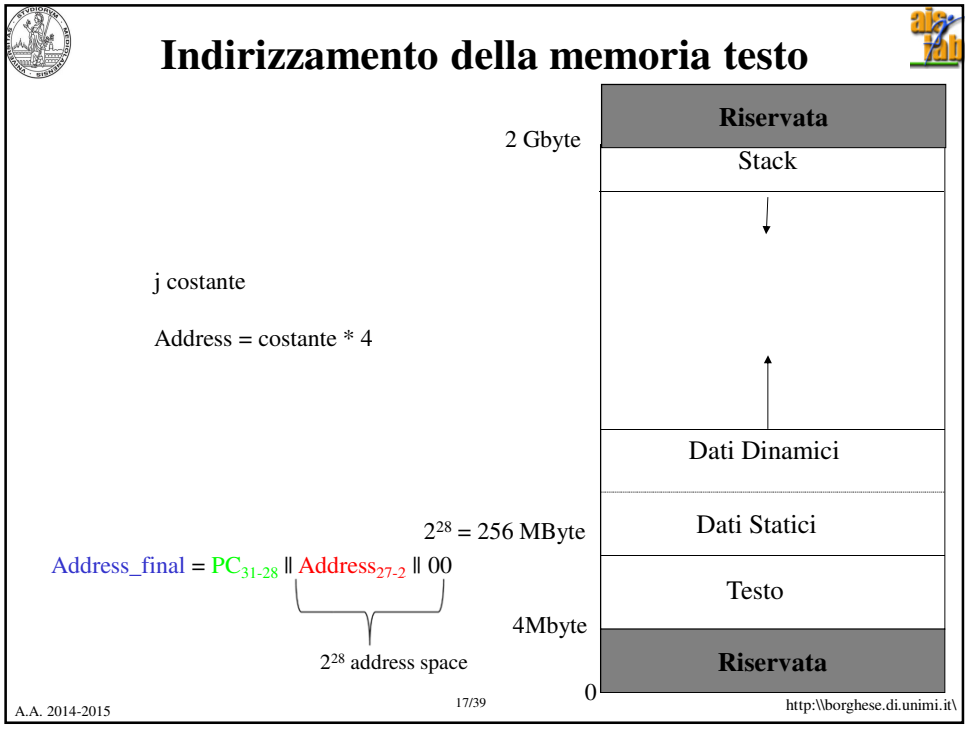
MEMORIA PRINCIPALE

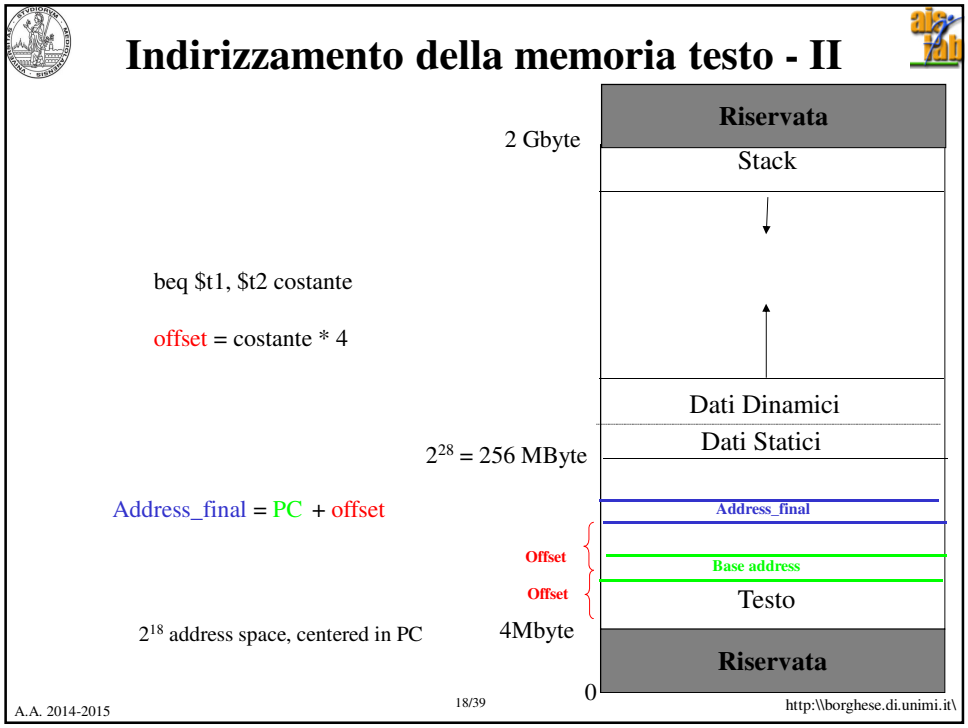
Organizzazione logica della memoria



Indirizzamento della memoria dati




Indirizzamento della memoria testo
2 Gbyte
Riservata
Stack
j costante
Address = costante * 4
Dati Dinamici
Dati Statici
2^{28} = 256 MByte
Address_final = PC_{31-28} || $Address_{27-2}$ || 00
Testo
4Mbyte
2^{28} address space
Riservata
0
A.A. 2014-2015
17/39
http:\\borghese.di.unimi.it\


Indirizzamento della memoria testo - II
2 Gbyte
Riservata
Stack
beq $t1, $t2 costante
offset = costante * 4
Dati Dinamici
Dati Statici
2^{28} = 256 MByte
Address_final = PC + offset
Address_final
Offset
Base address
Offset
Testo
2^{18} address space, centered in PC
4Mbyte
Riservata
0
A.A. 2014-2015
18/39
http:\\borghese.di.unimi.it\

Istruzioni di trasferimento dati

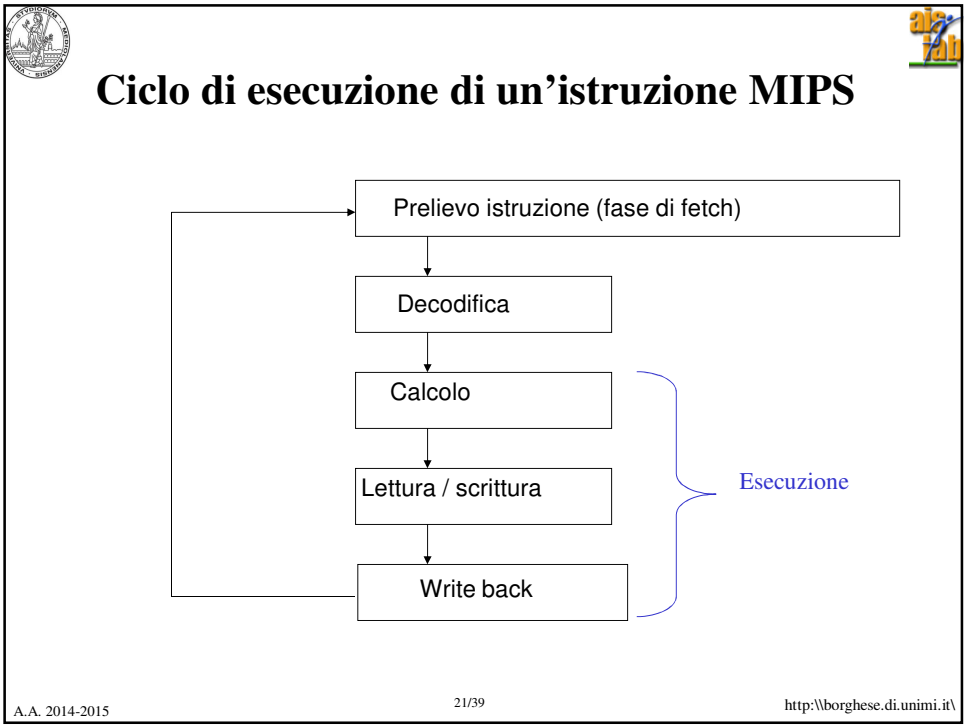
- Gli operandi di una istruzione aritmetica devono risiedere nei registri che sono in numero limitato (32 nel MIPS). I programmi in genere richiedono un numero maggiore di variabili.
- Cosa succede ai programmi i cui dati richiedono più di 32 registri (32 variabili)?

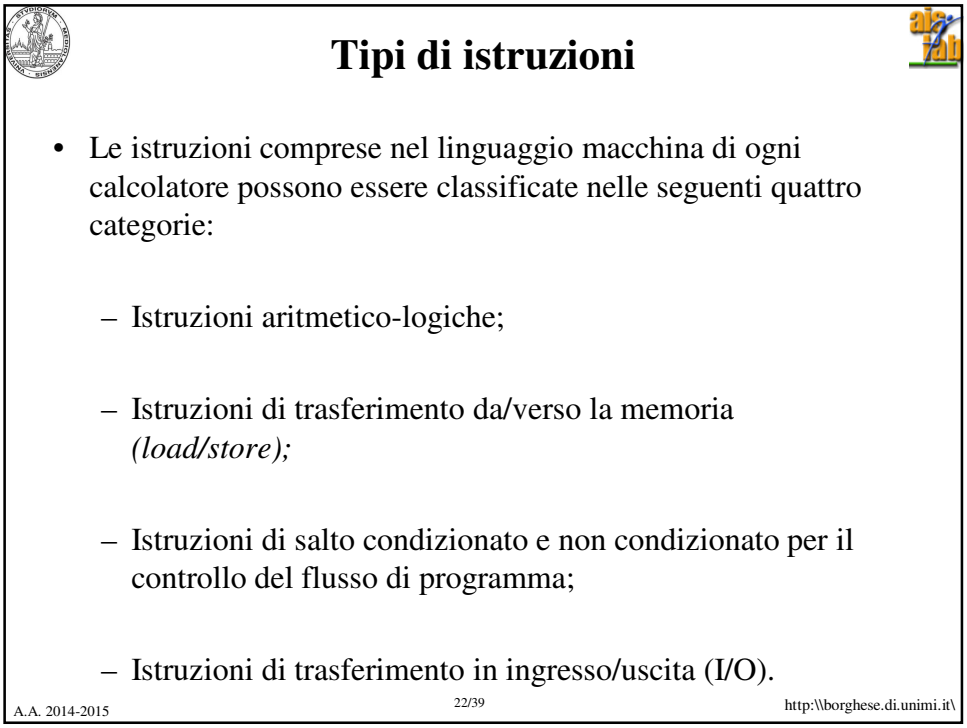
  Alcuni dati risiedono in memoria.
- La tecnica di mettere le variabili meno usate (o usate successivamente) in memoria viene chiamata ***Register Spilling***.

Servono istruzioni apposite per trasferire dati da memoria a registri e viceversa

MIPS: Software conventions for Registers

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


Ciclo di esecuzione di un'istruzione MIPS
Prelievo istruzione (fase di fetch)
Decodifica
Calcolo
Lettura / scrittura
Write back
Esecuzione
A.A. 2014-2015
21/39
http:\\borghese.di.unimi.it\


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).
A.A. 2014-2015
22/39
http:\\borghese.di.unimi.it\

Contenuto di un'istruzione

Tutte le istruzioni MIPS hanno la **stessa** dimensione **(32 bit) – Architettura RISC.**

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 delle istruzioni

- Tutte le istruzioni MIPS hanno la **stessa** dimensione **(32 bit) – Architettura RISC.**
- 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		
J	op	indirizzo				

Istruzioni

`add $s1, $s2, $s3`	000000	10010	10011	10001	00000	100000

`beq $s1, $s2, -100`	000100	10001	10010	1111	1111	1110	0111
`lw $t0, 32 ($s3)`	100011	10011	01000	0000	0000	0010	0000
`sw $t0, 32 ($s3)`	101011	10011	01000	0000	0000	0010	0000
`addi $t0, $s3, 64`	001000	10011	01000	0000	0000	0100	0000
`j 0x80000`	000010	00 0000 0100 0000 0000 0000 0000					

Definizione di un'ISA

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.

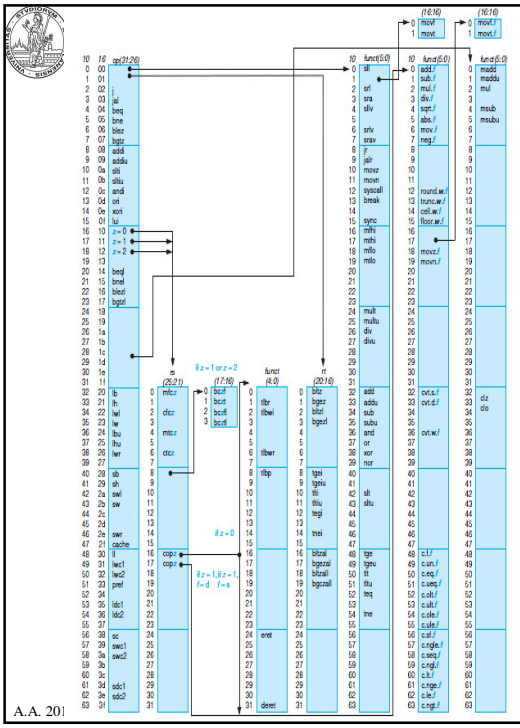
Le istruzioni di un'ISA

Devono contenere tutte le informazioni necessarie ad eseguire il ciclo di esecuzione dell'istruzione: registri, comandi, ...

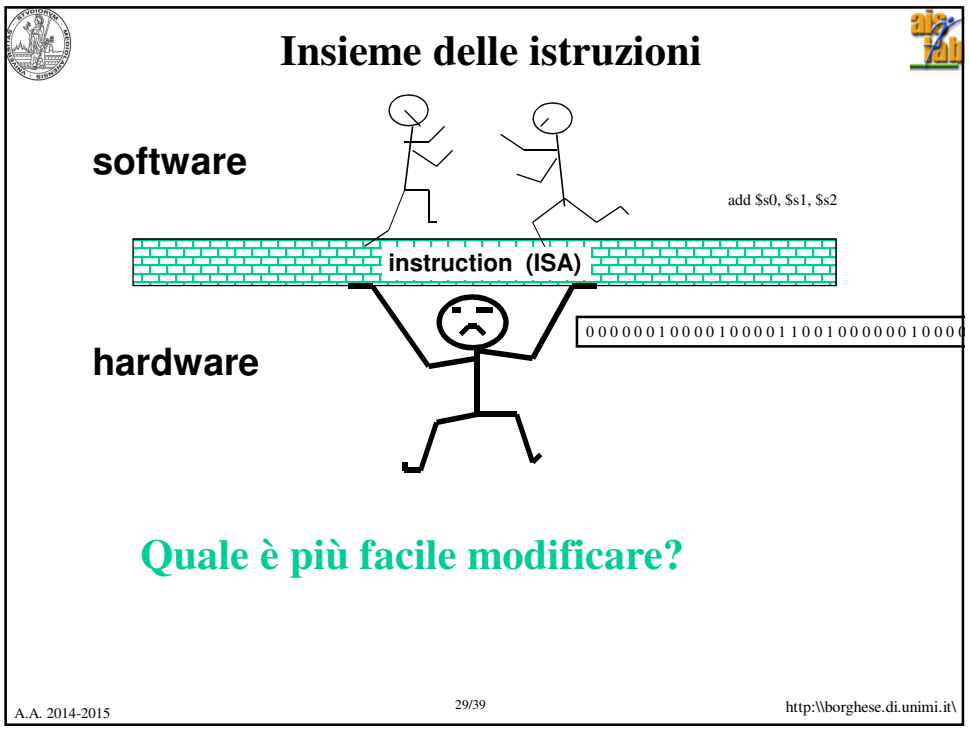
Ogni architettura di processore ha il suo linguaggio macchina

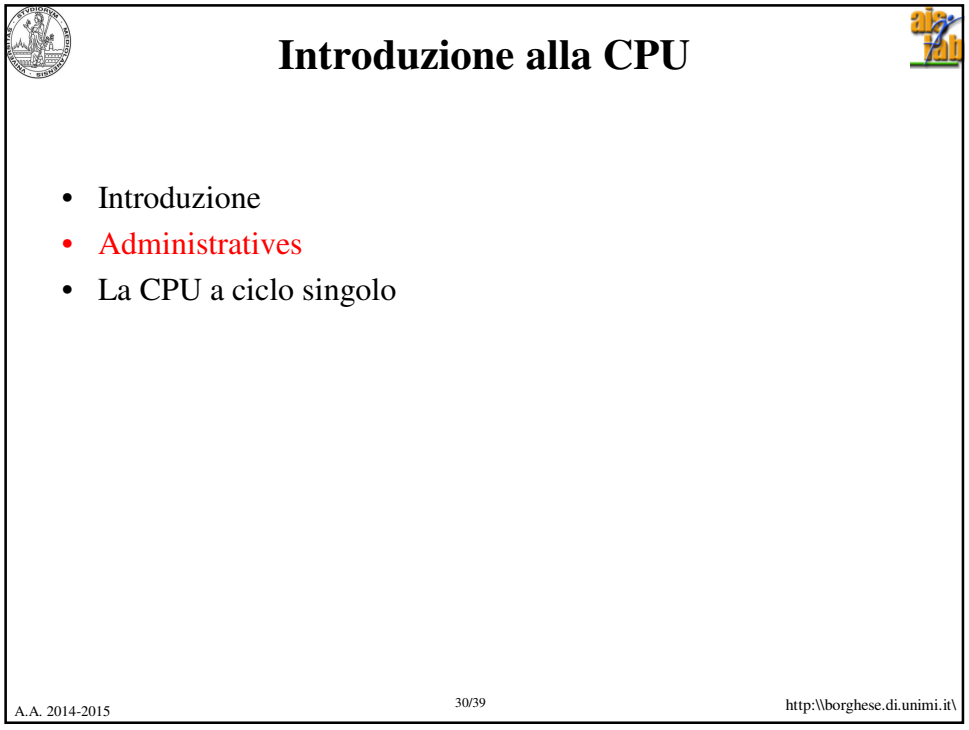
- Architettura dell'insieme delle istruzioni elementari messe a disposizione dalla macchina (in linguaggio macchina).
  - **ISA (Instruction Set Architecture)**
- Due processori con lo stesso linguaggio macchina hanno la stessa architettura delle istruzioni anche se le implementazioni hardware possono essere diverse.
- Consente al SW di accedere direttamente all'hardware di un calcolatore.

L'architettura delle istruzioni, specifica come vengono costruite le istruzioni in modo tale che siano comprensibili alla macchina (in linguaggio macchina).



I codici operativi


Insieme delle istruzioni
software
add $s0, $s1, $s2
instruction (ISA)
hardware
0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 0 1 0 0 0 0 0 0 1 0 0 0 0
Quale è più facile modificare?
A.A. 2014-2015
29/39
http:\\borghese.di.unimi.it\


Introduzione alla CPU
• Introduzione
• Administratives
• La CPU a ciclo singolo
A.A. 2014-2015
30/39
http:\\borghese.di.unimi.it\

Architetture II (6cfu)

Turno 1 - Cognomi A-F

Docente: Prof. N. Alberto Borghese: alberto.borghese@unimi.it
Laboratorio Assembler: Dott. Nicola Basilico: nicola.basilico@unimi.it

Orario e aule:

Mercoledì	Ore 8.30-10.30	Aula 405, Via Celoria 20
Giovedì	Ore 13.30-15.30	Aula 309, Via Celoria 20 (laboratorio)
Venerdì	**Ore 10.30-12.30**	**Aula V3, Via Venezian**

Orario di ricevimento: su appuntamento.

Strumento principale di contatto: email!

Programma

Sito principale:
http://borghese.dsi.unimi.it/Teaching/Architettura_II/_Arch_II.html

Programma:
http://borghese.dsi.unimi.it/Teaching/Architettura_II/Programma_2014-2015.html

Argomenti principali:
- CPU (avanzate)
- Gerarchie di memoria
- Interconnessioni

Esame

Parte teorica (2/3 del voto).

- Prova scritta + orale. Appelli ogni 1 / 2 / 3 mesi, al di fuori dal periodo delle lezioni.
- 2 compitini in itinere durante l'anno. I compitini sostituiscono interamente scritto e orale.
- Per superare la parte di teoria con i compitini occorre avere preso almeno 17 in tutti e due i compitini e che la media dei 2 compitini sia ≥ 18. I compitini sono consigliati solo a chi frequenta.
- L'orale con i compitini è facoltativo.
- Per sostenere l'esame occorre iscriversi sul SIFA all'**appello scritto**. Non è permesso portare il telefonino nell'aula dello scritto.

Progetto di laboratorio in Assembler (PC-Spim, 1/3 del voto).

Materiale didattico

See web page

http://borghese.dsi.unimi.it/Teaching/Architettura_II/References.rtf

Testo di base (è disponibile sia in inglese che in italiano):
Struttura e progetto dei calcolatori: l'interfaccia hardware-software, D.A. Patterson and J.L. Hennessy, Terza edizione, Zanichelli, estate 2010 (Nota: la terza edizione Zanichelli è la traduzione della quarta edizione inglese).

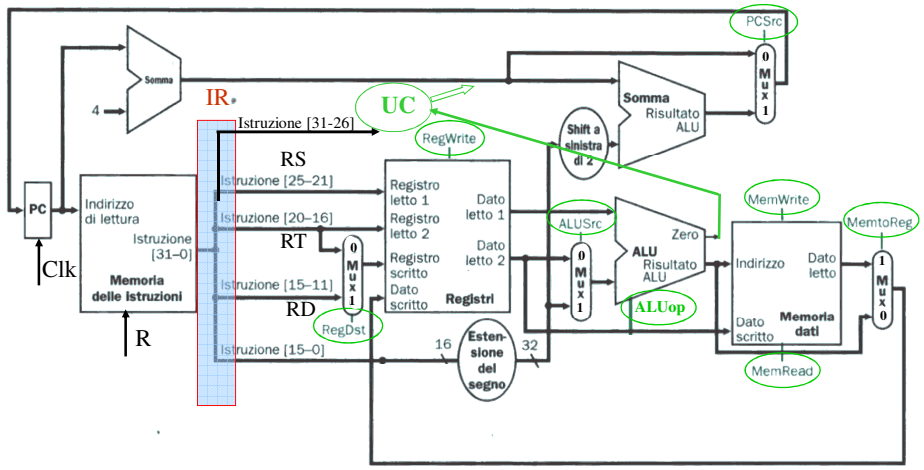
"Computer Organization & Design: The Hardware/Software Interface", D.A. Patterson and J.L. Hennessy, Morgan Kaufmann Publishers, Fifth Edition, 2013.

Potete trovare esercizi del testo svolti al seguente URL: http://books.elsevier.com/companions/1558606041/.

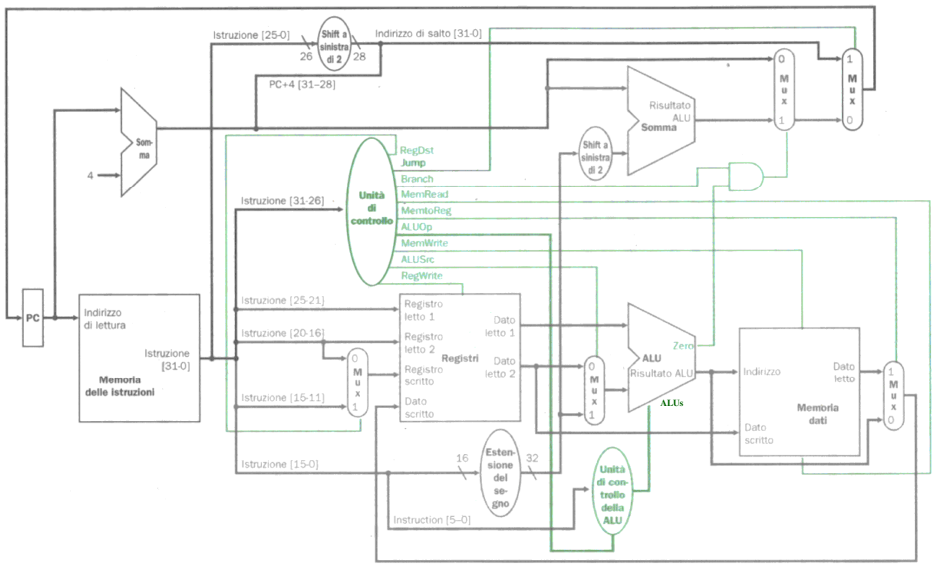
Introduzione alla CPU

- Introduzione
- Administratives
- La CPU a ciclo singolo

Schema generale (lw/sw + R + beq)



The control and data path



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.

Introduzione alla CPU

- Introduzione
- Administratives
- La CPU a ciclo singolo