



UC della CPU a singolo ciclo

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

Università degli Studi di Milano

Riferimento sul Patterson: capitolo 4.2 , 4.4, D1, D2.



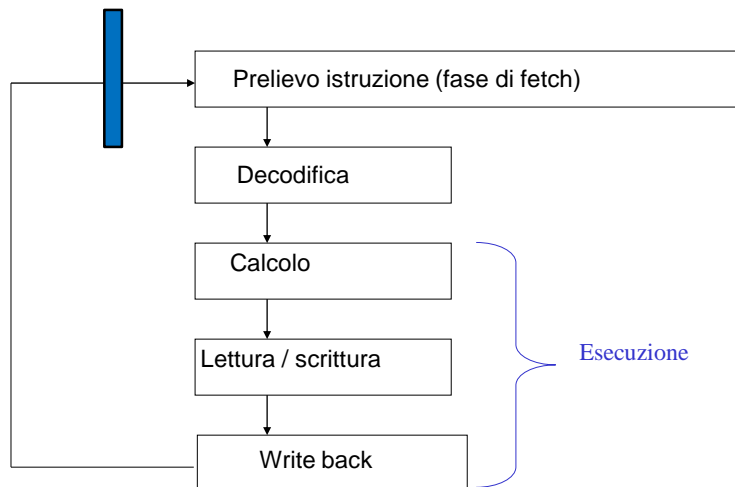
Sommario

UC della CPU

Control and Data path



Ciclo di esecuzione di un'istruzione MIPS

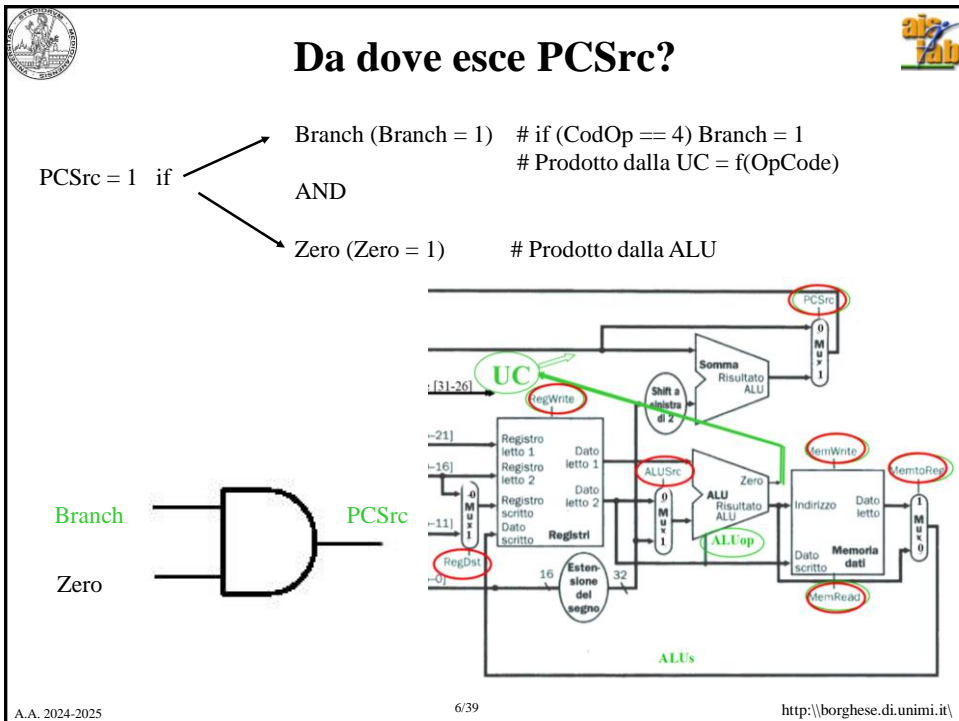
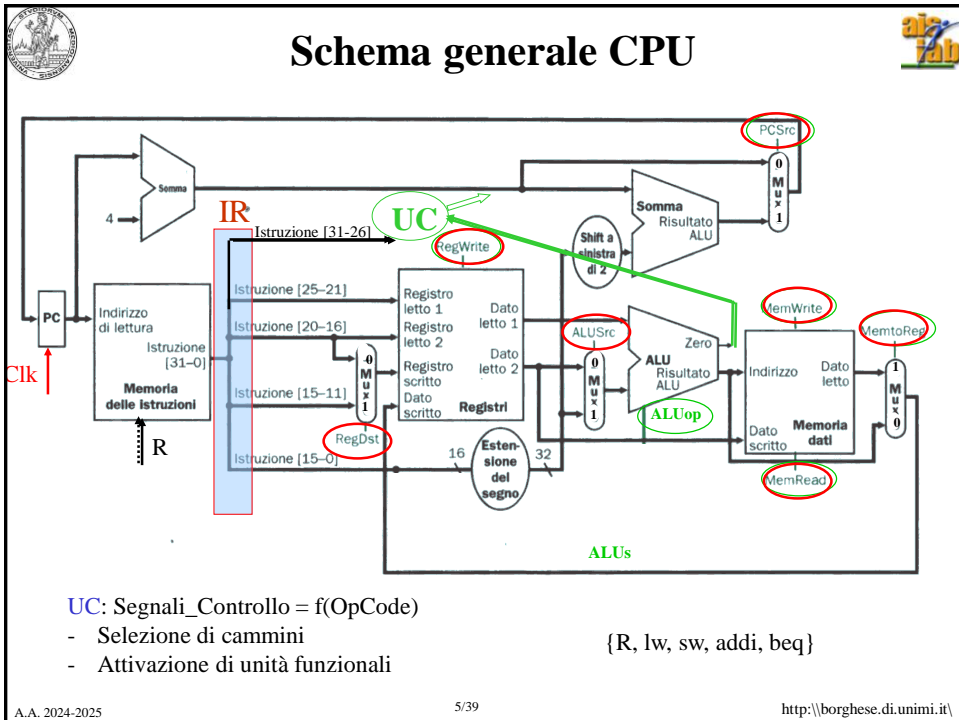


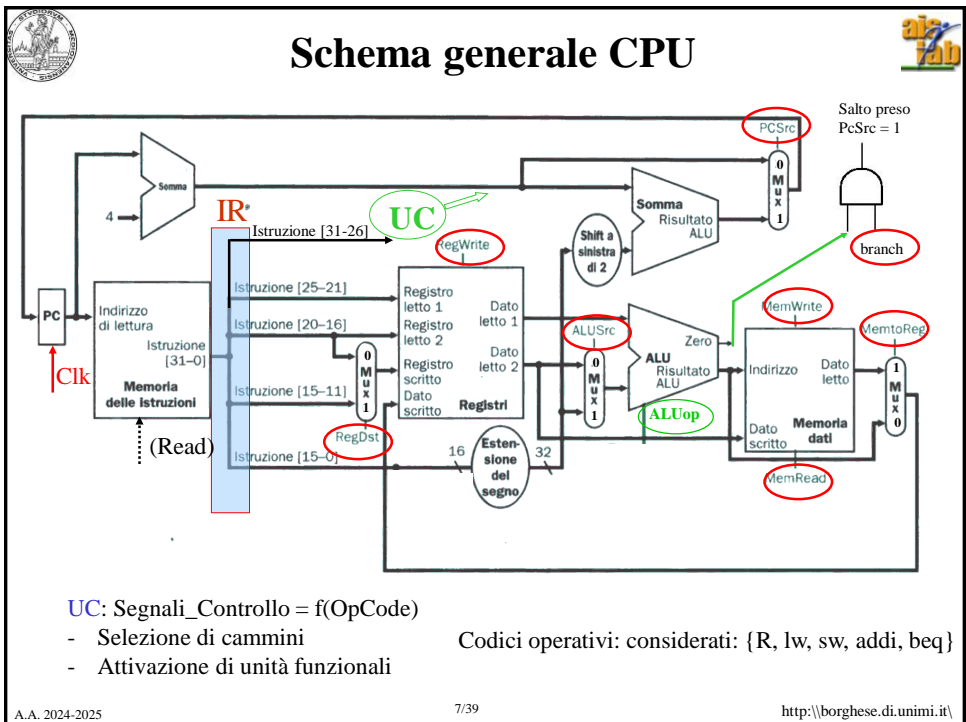
Codifica delle istruzioni



- 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 / costante		
J	op	indirizzo				







Segnali di controllo su 1 bit

Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst		
RegWrite		
ALUSrc		
Branch		
MemRead		
MemWrite		
MemtoReg		

A.A. 2024-2025 <http://borghese.di.unimi.it/>

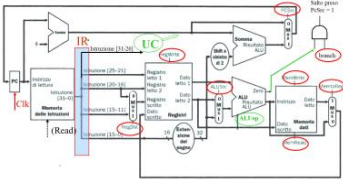


Segnali di controllo su 1 bit




Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite		
ALUSrc		
Branch		
MemRead		
MemWrite		
MemtoReg		


add \$t0, \$t1, \$t2
addi \$t0, \$t1, 24



A.A. 2024-2025<http://borghese.di.unimi.it/>

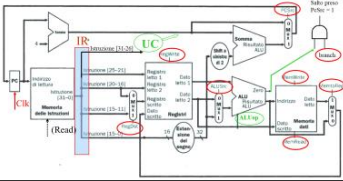


Segnali di controllo su 1 bit




Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite	Nessuno	Nel registro specificato all'ingresso registro scritto del Register File, viene scritto il valore presente all'ingresso Dato Scritto
ALUSrc		
Branch		
MemRead		
MemWrite		
MemtoReg		

add \$t0, \$t1, \$t2
addi \$t0, \$t1, 24
lw \$t0, 24(\$t1)



A.A. 2024-2025<http://borghese.di.unimi.it/>



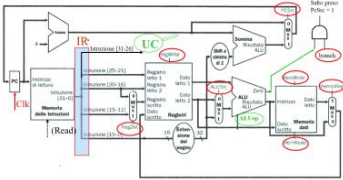
Segnali di controllo su 1 bit

Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite	Nessuno	Nel registro specificato all'ingresso registro scritto del Register File, viene scritto il valore presente all'ingresso Dato Scritto
ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita in lettura del Register File	Il secondo operando della ALU è la versione estesa (con segno) del campo offset
Branch	Il valore del PC viene preso dall'uscita del sommatore che calcola PC+4 (se il segnale di Zero della ALU = 0)	Il valore del PC viene preso dall'uscita del sommatore che calcola la destinazione del salto (se il segnale di Zero della ALU = 1)
MemRead		
MemWrite		
MemtoReg		

add \$t0, \$t1, \$t2


addi \$t0, \$t1, 24

lw \$t0, 24(\$t1)



A.A. 2024-2025

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



Segnali di controllo su 1 bit

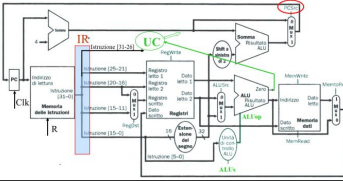
Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite	Nessuno	Nel registro specificato all'ingresso registro scritto del Register File, viene scritto il valore presente all'ingresso Dato Scritto
ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita in lettura del Register File	Il secondo operando della ALU è la versione estesa (con segno) del campo offset
Branch	Il valore del PC viene preso dall'uscita del sommatore che calcola PC+4 (se il segnale di Zero della ALU = 0)	Il valore del PC viene preso dall'uscita del sommatore che calcola la destinazione del salto (se il segnale di Zero della ALU = 1)
MemRead	Nessuno	Il contenuto della cella di memoria dati indirizzata dal MAR è posto nel MDR
MemWrite		
MemtoReg		

add \$t0, \$t1, \$t2

addi \$t0, \$t1, 24


lw \$t0, 24(\$t1)

sw \$t0, 24(\$t1)




A.A. 2024-2025

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

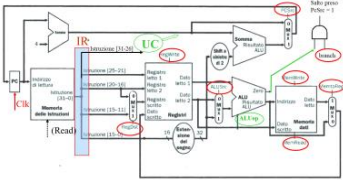


Segnali di controllo su 1 bit




Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite	Nessuno	Nel registro specificato all'ingresso registro scritto del Register File, viene scritto il valore presente all'ingresso Dato Scritto
ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita in lettura del Register File	Il secondo operando della ALU è la versione estesa (con segno) del campo offset
Branch	Il valore del PC viene preso dall'uscita del sommatore che calcola PC+4 (se il segnale di Zero della ALU = 0)	Il valore del PC viene preso dall'uscita del sommatore che calcola la destinazione del salto (se il segnale di Zero della ALU = 1)
MemRead	Nessuno	Il contenuto della cella di memoria dati indirizzata dal MAR è posto nel MDR
MemWrite	Nessuno	Il contenuto in ingresso al MDR, viene memorizzato nella cella il cui indirizzo è caricato nel MAR
MemtoReg		

add \$t0, \$t1, \$t2
addi \$t0, \$t1, 24
lw \$t0, 24(\$t1)
sw \$t0, 24(\$t1)




A.A. 2024-2025

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

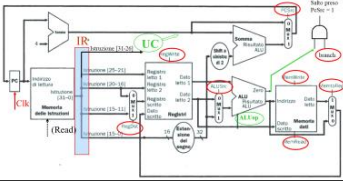


Segnali di controllo su 1 bit



Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite	Nessuno	Nel registro specificato all'ingresso registro scritto del Register File, viene scritto il valore presente all'ingresso Dato Scritto
ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita in lettura del Register File	Il secondo operando della ALU è la versione estesa (con segno) del campo offset
Branch	Il valore del PC viene preso dall'uscita del sommatore che calcola PC+4 (se il segnale di Zero della ALU = 0)	Il valore del PC viene preso dall'uscita del sommatore che calcola la destinazione del salto (se il segnale di Zero della ALU = 1)
MemRead	Nessuno	Il contenuto della cella di memoria dati indirizzata dal MAR è posto nel MDR
MemWrite	Nessuno	Il contenuto in ingresso al MDR, viene memorizzato nella cella il cui indirizzo è caricato nel MAR
MemtoReg	Il valore inviato all'ingresso Dato al Register File proviene dalla ALU	Il valore inviato all'ingresso Dato al Register File proviene dalla memoria

add \$t0, \$t1, \$t2
addi \$t0, \$t1, 24
lw \$t0, 24(\$t1)
sw \$t0, 24(\$t1)



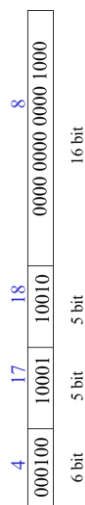
A.A. 2024-2025

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



Fase di decodifica

- 1) Leggo l'istruzione e genero i segnali di controllo opportuni.
Bus che connette i 6 bit del CodOp con la UC

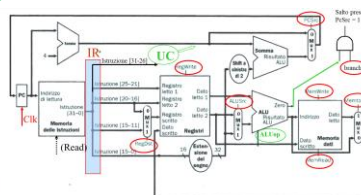


OpCode

Unità
Controllo

Segnali di controllo

0x8000 beq \$s1, \$s2, 8
0x8004 add \$s4, \$s1, \$t1
.....



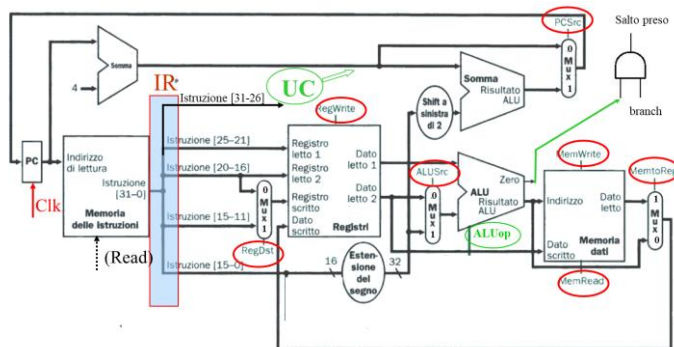
A.A. 2024-2025

15/39



Codifica dei segnali di controllo

Istruzione (OpCode)	RegDst	ALUSrc	MemtoReg	Reg Write	Mem Read	Mem Write	Branch	ALUs
R (000000)								
lw (100011)								
sw (101011)								
beq (000100)								
addi(001000)								



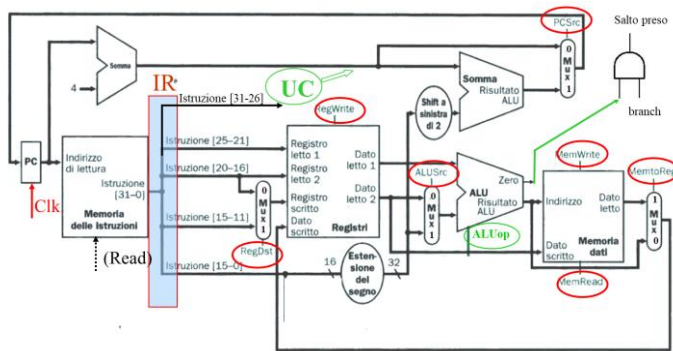
<http://borghese.di.unimi.it/>



Controllo del data-path



Istruzione (OpCode)	RegDst	ALUSrc	MemtoReg	Reg Write	Mem Read	Mem Write	Branch	ALUs
R (000000)	1	0	0	1	0	0	0	10 (R)
lw (100011)								
sw (101011)								
beq (000100)								
addi(001000)								



add \$t0, \$t1, \$t2

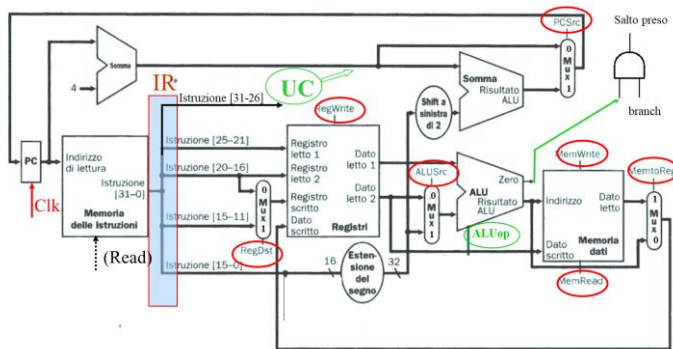
<http://borghese.di.unimi.it/>



Controllo del data-path



Istruzione (OpCode)	RegDst	ALUSrc	MemtoReg	Reg Write	Mem Read	Mem Write	Branch	ALUs
R (000000)	1	0	0	1	0	0	0	10 (R)
lw (100011)	0	1	1	1	1	0	0	00 (add)
sw (101011)								
beq (000100)								
addi(001000)								



lw \$t0, (24)\$t2

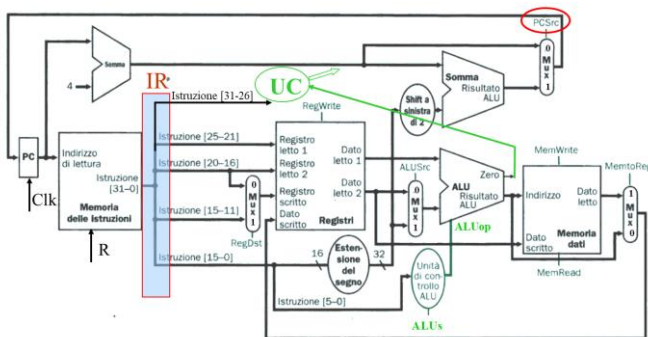
<http://borghese.di.unimi.it/>



Controllo del data-path



Istruzione (OpCode)	RegDst	ALUSrc	MemtoReg	Reg Write	Mem Read	Mem Write	Branch	ALUs
R (000000)	1	0	0	1	0	0	0	10 (R)
lw (100011)	0	1	1	1	1	0	0	00 (add)
sw (101011)	x	1	x	0	0	1	0	00 (add)
beq (000100)								
addi(001000)								



sw \$t0, (24)\$t2

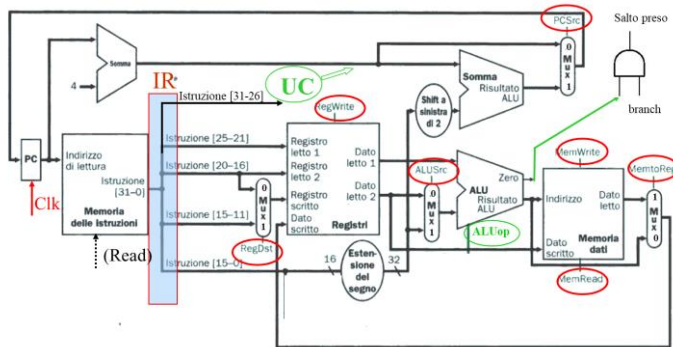
<http://borghese.di.unimi.it/>



Controllo del data-path



Istruzione (OpCode)	RegDst	ALUSrc	MemtoReg	Reg Write	Mem Read	Mem Write	Branch	ALUs
R (000000)	1	0	0	1	0	0	0	10 (R)
lw (100011)	0	1	1	1	1	0	0	00 (add)
sw (101011)	x	1	x	0	0	1	0	00 (add)
beq (000100)	x	0	x	0	0	0	1	01 (sub)
addi(001000)								



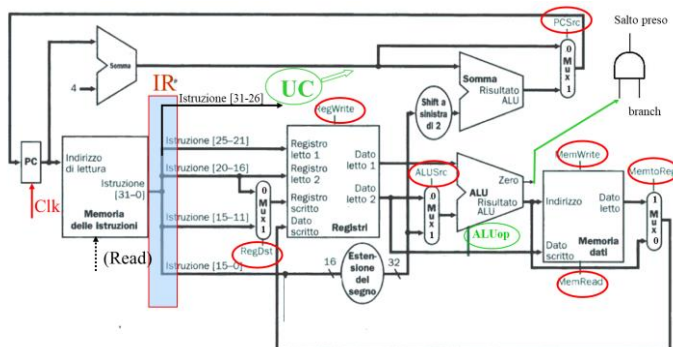
beq \$t0, \$t2, 24

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



Controllo del data-path

Istruzione (OpCode)	RegDst	ALUSrc	MemtoReg	Reg	Mem	Mem	Branch	ALUs
				Write	Read	Write		
R (000000)	1	0	0	1	0	0	0	10 (R)
lw (100011)	0	1	1	1	1	0	0	00 (add)
sw (101011)	x	1	x	0	0	1	0	00 (add)
beq (000100)	x	0	x	0	0	0	1	01 (sub)
addi(001000)	0	1	0	1	0	0	0	00 (add)



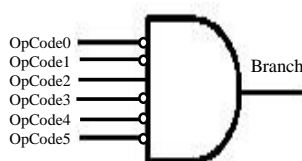
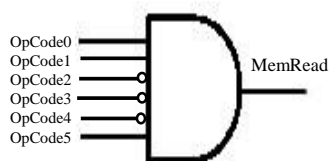
```
addi $t0, $t2, 24
```

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



Controllo del data-path

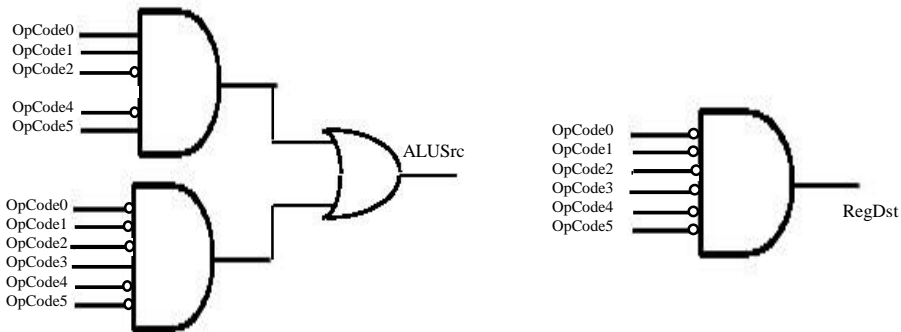
Istruzione (OpCode)	RegDst	ALUSrc	MemtoReg	Reg Write	Mem Read	Mem Write	Branch	ALUs
R (000000)	1	0	0	1	0	0	0	10
lw (100011)	0	1	1	1	1	0	0	00
sw (101011)	x	1	x	0	0	1	0	00
beq (000100)	x	0	x	0	0	0	1	01
addi(001000)	0	1	0	1	0	0	0	00



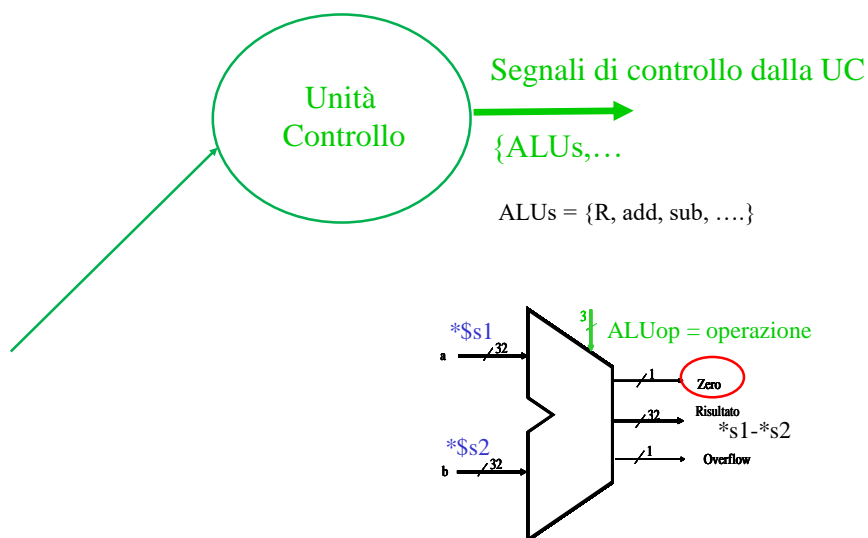


Controllo del data-path

Istruzione (OpCode)	RegDst	ALUSrc	MemtoReg	Reg Write	Mem Read	Mem Write	Branch	ALUs
R (000000)	1	0	0	1	0	0	0	10
lw (100011)	0	1	1	1	1	0	0	00
sw (101011)	x	1	x	0	0	1	0	00
beq (000100)	x	0	x	0	0	0	1	01
addi(001000)	0	1	0	1	0	0	0	00

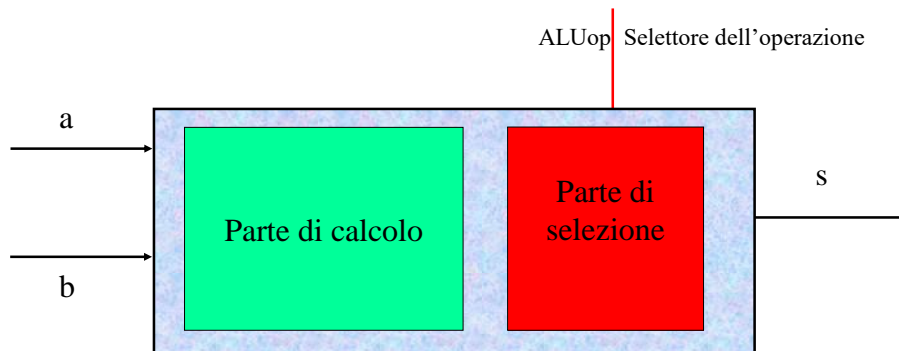


Controllo della ALU





Struttura a 2 livelli di una ALU



Eseguire un'operazione vuol dire scegliere di portare in uscita un certo risultato prodotto dalla Parte di calcolo oppure un altro



Segnali di UC e ALU

Visione della UC

	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
R I	op	rs	rt	rd	shamt	funct
	op	rs	rt	Indirizzo / costante		
Operazione? (ALUs = 10 = R)	sub \$s0, \$s1, \$s2	op = 0	rs = 18	rt = 19	rd = 17	Shamt=0, funct=34
Operazione (ALUs = 00 = somma)	addi \$0, \$1, 20	op = 8	rs = 17	rt = 18	cost = 20	

Visione della ALU

Le operazioni consentite dalla ALU (selezionate tramite ALUOp):



Tipo	Aluop
and	000
or	001
add	010
sub	110
slt	111



UC e ALU



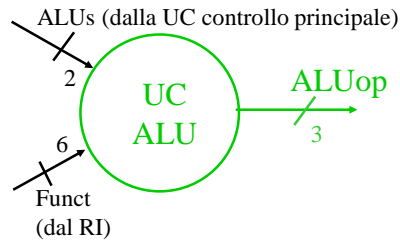
Data l'istruzione, l'UC deve inviare il comando opportuno alla ALU.

Campo Op Code

Campo Funct

Le operazioni consentite dalla ALU:

- and 000
- or 001
- add 010
- sub 110
- slt 111



Quali operazioni devono essere eseguite per le diverse istruzioni (ALUs):

- R -> Dipende dal campo funct
- lw -> Somma
- sw -> Somma
- beq -> Differenza

$ALUs = f(OpCode)$

$ALUop = f(ALUs, Funct)$

Output (ALUop): 3 bit

Funct: 6 bit

ALUs: 2 bit

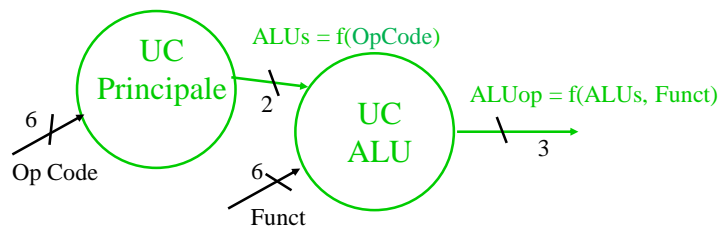


Controllo gerarchico





Le operazioni consentite dalla ALU:

- and 000
- or 001
- add 010
- sub 110
- slt 111



If (OpCode == R) then
Funct → ALUop

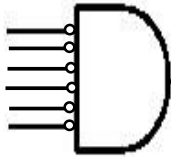
Else
OpCode → ALUop

Controllo della ALU

Istr	OpCode						ALUs	
lw	1	0	0	0	1	1	0	0
sw	1	0	1	0	1	1	0	0
beq	0	0	0	1	0	0	0	1
addi	0	0	1	0	0	0	0	0
add	0	0	0	0	0	0	1	0
sub	0	0	0	0	0	0	1	0
and	0	0	0	0	0	0	1	0
or	0	0	0	0	0	0	1	0
slt	0	0	0	0	0	0	1	0

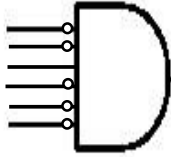
OpCode0
OpCode1
OpCode2



ALUs1

OpCode4
OpCode5

OpCode0
OpCode1
OpCode2
OpCode3
OpCode4
OpCode5



ALUs0



Sintetizzo i 2 bit come SOP

$ALUs = f(OpCode)$

A.A. 2024-2025

29/39

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

Controllo della ALU

Istr	OpCode						ALUs		Funct						ALUOp		
lw	1	0	0	0	1	1	0	0	x	x	x	x	x	x	0	1	0
sw	1	0	1	0	1	1	0	0	x	x	x	x	x	x	0	1	0
beq	0	0	0	1	0	0	0	1	x	x	x	x	x	x	1	1	0
addi	0	0	1	0	0	0	0	0	x	x	x	x	x	x	0	1	0
add	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	1	0
sub	0	0	0	0	0	0	1	0	1	0	0	0	1	0	1	1	0
and	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0	0
or	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0	1
slt	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1

UC principale

ALU control

$ALUOp = f(ALUs, Funct)$

A.A. 2024-2025

30/39

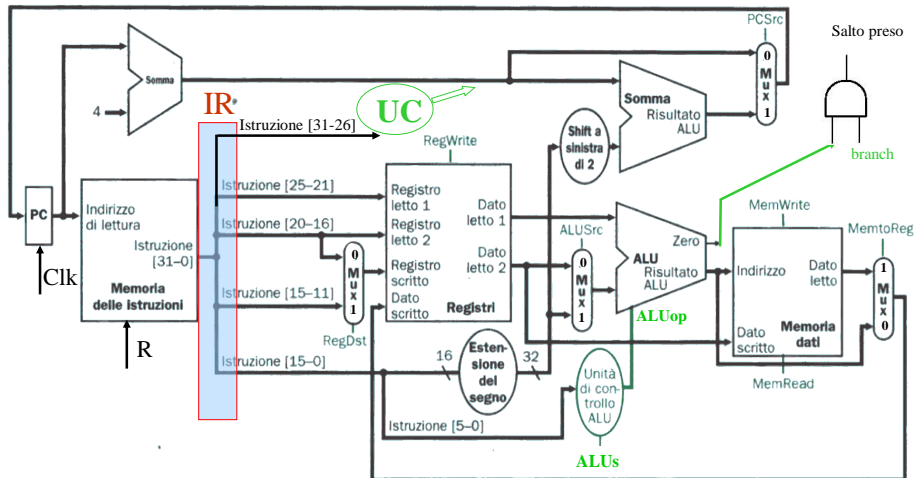
<http://borghese.di.unimi.it/>



Schema generale CPU



{R, lw, sw, addi, beq}



Sommario



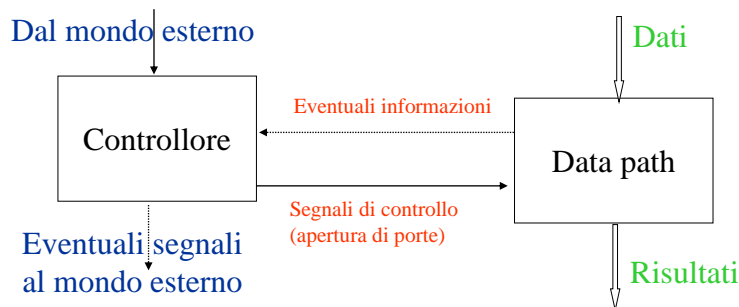
UC della CPU

Control and Data path



Rapporto UC - Dati

La CPU è un'architettura del tipo: Controllore - Data-path



Fase comune nel ciclo di esecuzione:

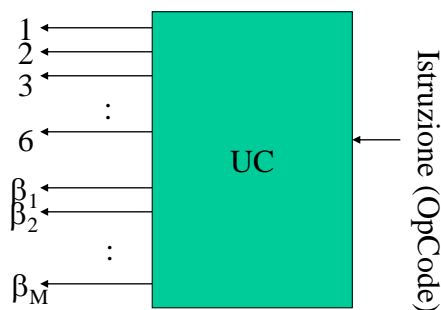
- Fase di fetch
- Decodifica (generazione dei segnali di controllo)

Fase diversa: Esecuzione (Calcolo, Memoria, WriteBack)



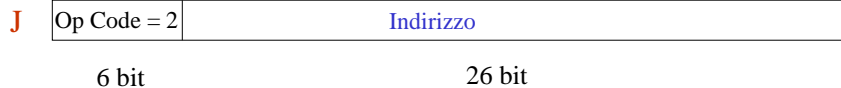
L'unità di controllo

- Unità di controllo coordina i flussi di informazione (è il "cervello" della CPU):
- 1) abilitando le vie di comunicazione opportune a seconda dell'istruzione in corso di esecuzione.
- 2) selezionando l'operazione opportuna delle ALU.
- 3) modificando un elemento di stato (memoria o register file)





L'istruzione jump



L'indirizzo di salto sarà determinato in due passi: 0x8000 j 80040

- A) Calcolo di Indirizzo = Indirizzo * 4.
B) Deteminazione dell'indirizzo di destinazione del salto come:

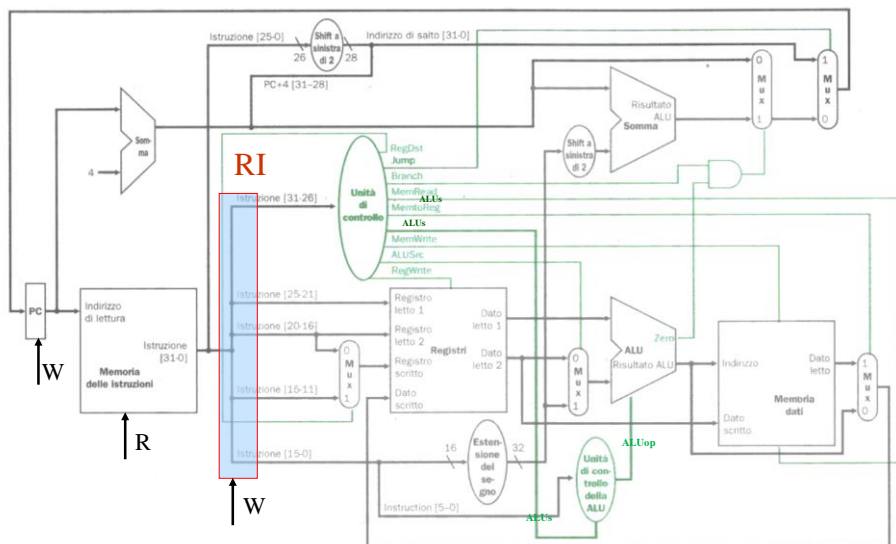
Base (PC) 0100 1000 0011 0001 1011 1011 1011 10 00
Nuovo indirizzo 0000 0110 0111 0000 0000 0001 00 00 =

Indirizzo salto 0100 0000 0110 0111 0000 0000 0001 00 00

L'indirizzo è un numero positivo (posizione in memoria assoluta).



CPU + UC completa (aggiunta di jump)





Controllo del data-path

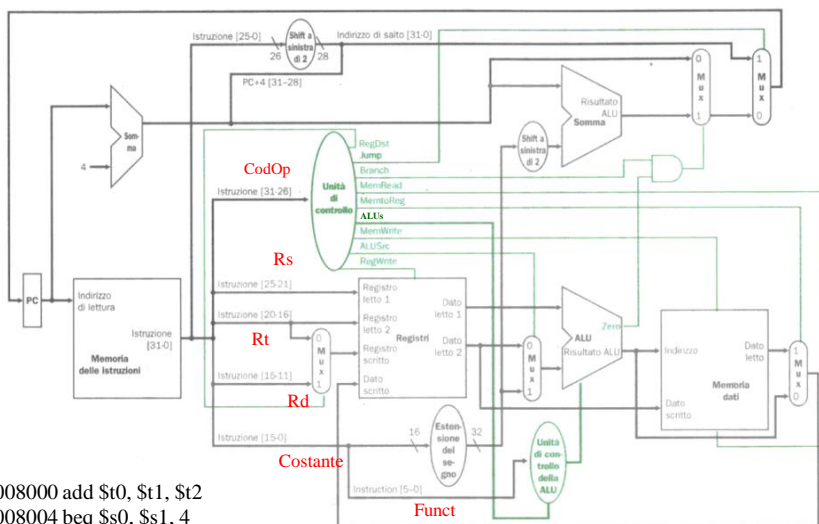


Istruzione (OpCode)	Reg Dst	ALU Src	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUs	Jump
R (000000)	1	0	0	1	0	0	0	10	0
lw (100011)	0	1	1	1	1	0	0	00	0
sw (101011)	x	1	x	0	0	1	0	00	0
addi (001000)	0	1	0	1	0	0	0	00	0
beq (000100)	x	0	x	0	0	0	1	01	0
j (000010)	x	x	x	0	0	0	0	xx	1

La lettura della memoria non è indolore soprattutto quando sono presenti dei livelli (di cache).



Contenuto della CPU per l'esecuzione di istruzioni diverse



0x00008000 add \$t0, \$t1, \$t2
0x00008004 beq \$s0, \$s1, 4
0x00008008 lw \$t1, 32(\$s0)
0x0000800C sw \$t1, 32(\$s0)
0x00008010 addi \$t1, \$t1, 16



Sommario



UC della CPU

Control and Data path