



# L'unità di controllo di CPU a singolo ciclo

Prof. Alberto Borghese  
Dipartimento di Scienze dell'Informazione  
[borgnese@dsi.unimi.it](mailto:borgnese@dsi.unimi.it)

Università degli Studi di Milano



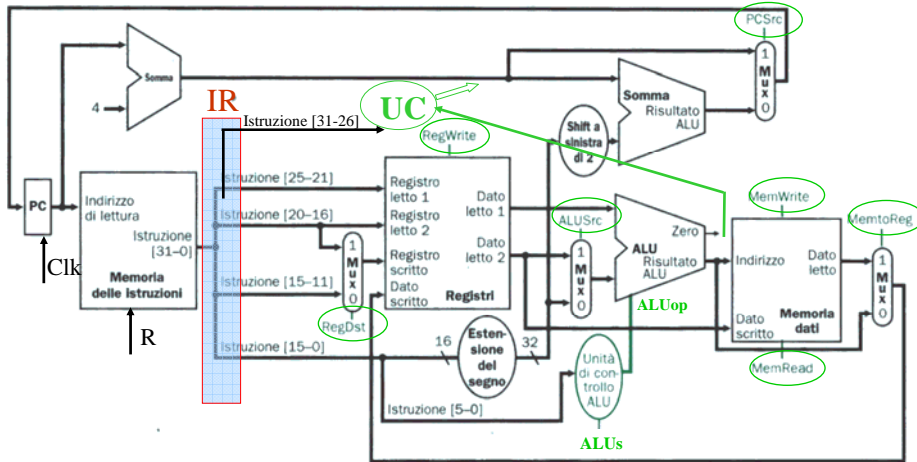
## Sommario

Controllore della ALU

Unità di Controllo Principale



## Schema generale (lw/sw + R + beq)



Controllore della ALU:  $ALUs = f(..)$

Controllore della CPU:  $Segnali\_Controllo = f(OpCode)$

W = Clock



## Tipi di istruzioni dell'ISA



Consideriamo istruzioni di tipo R, di tipo lw/sw, salto condizionato:

	31-26	25-21	20-16	15-11	10-6	5-0
R	op	rs	rt	rd	shamt	funct
lw/sw	35/43	rs	rt	offset		
beq	4	rs	rt	indirizzo		

### Architettura RISC:

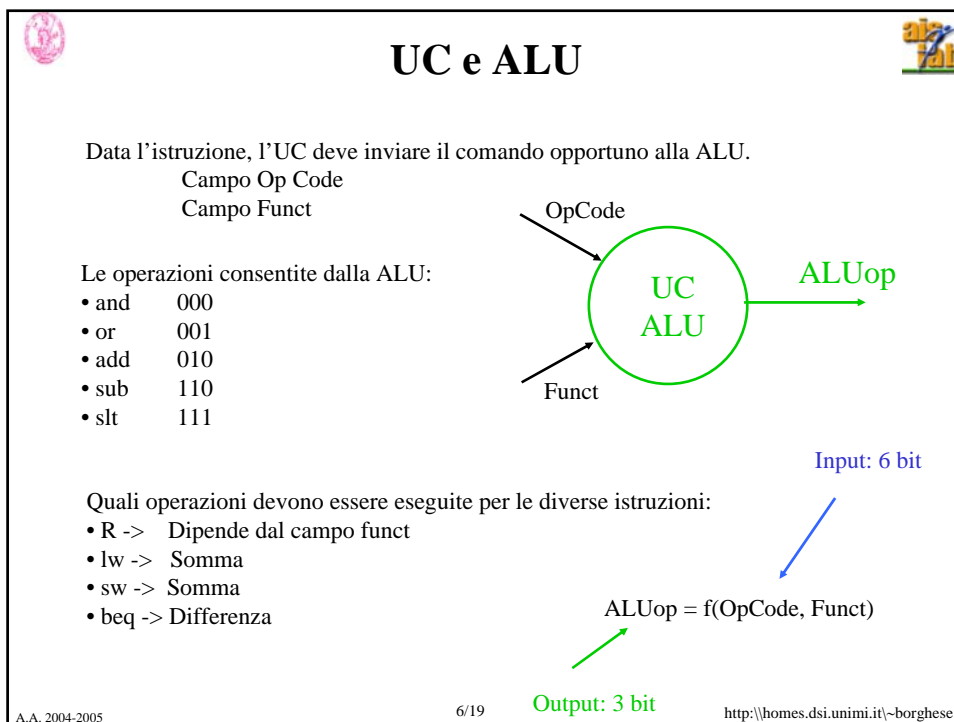
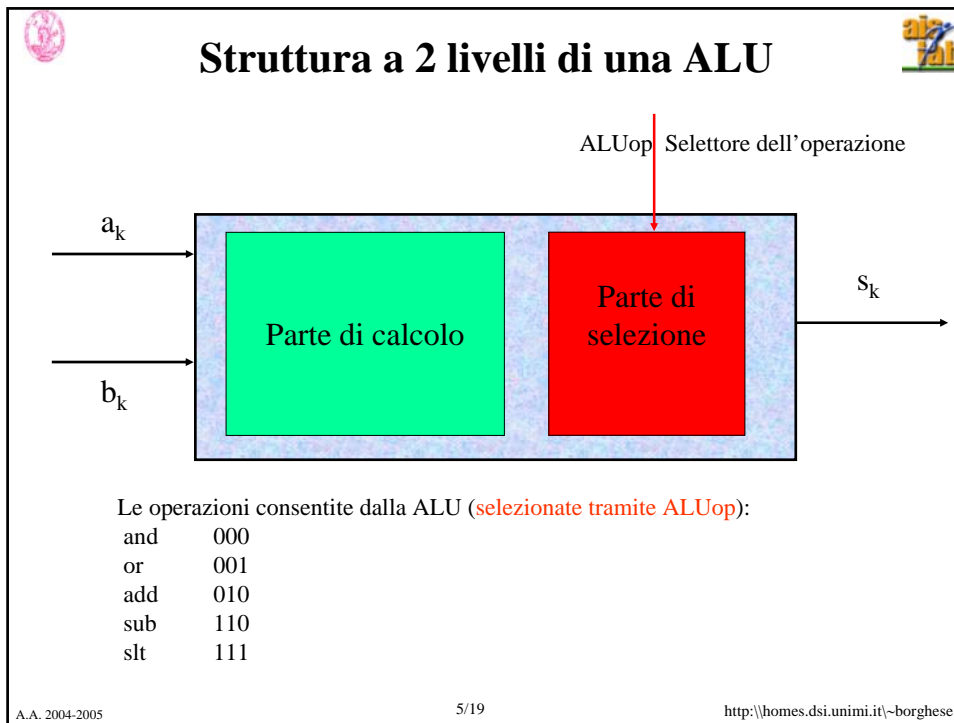
Campo Op sempre contenuto nei primi 6 bit, inseguito Op[5-0].

Registri da leggere, rs ed rt, in posizione 25-21 e 20-16. Vengono sempre letti dal Register File.

Registro base per lettura / scrittura, rs, sempre in posizione 25-21.

Offset sempre in posizione 15-0.

Registro destinazione, rd (15-11) per le istruzioni di tipo R, rt (20-16) per le istruzioni lw.



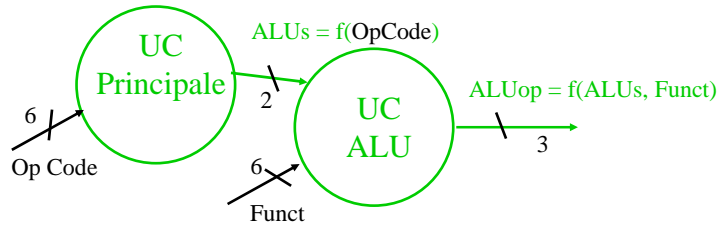


## Controllo gerarchico



Le operazioni consentite dalla ALU:

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



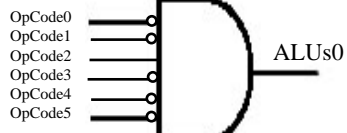
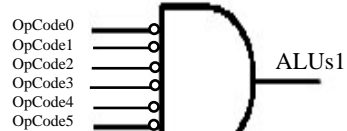
If (OpCode == R) then  
 Funct → ALUop  
 Else  
 OpCode → ALUop



## Controllo della ALU



Istr	OpCode						ALUs	
	1	0	0	0	1	1	0	0
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
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



Sintetizzo i 2 bit come SOP

$$ALUs = f(OpCode)$$



## Controllo della ALU



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

$$ALUop = f(ALUs, Funct)$$

SOP



## Sommario

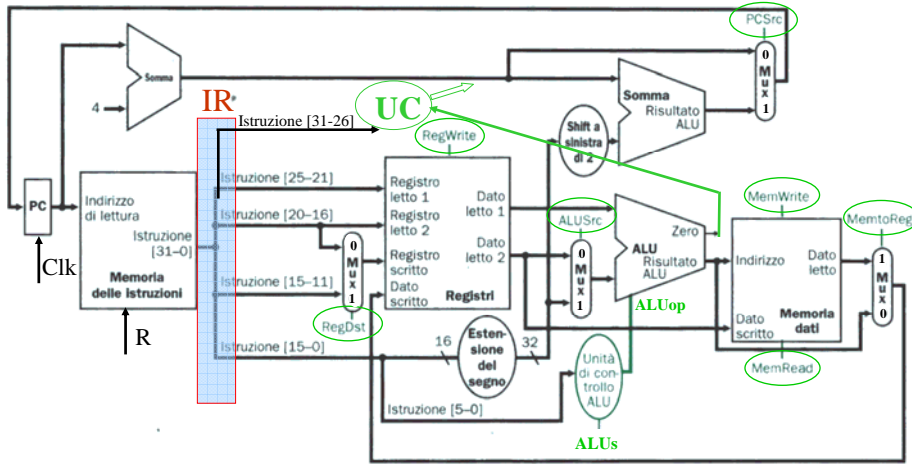


Controllore della ALU

Unità di Controllo Principale



## Schema generale (lw/sw + R + beq)



Controllore della ALU:  $ALUop = f(\dots)$

Controllore della CPU:  $Segnali\_Controllo = f(OpCode)$

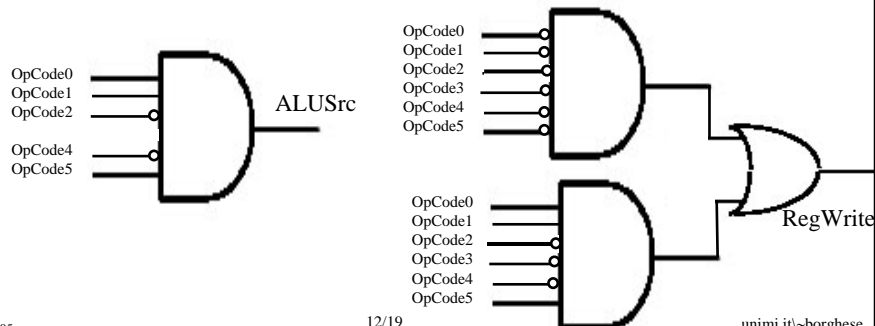
W = Clock

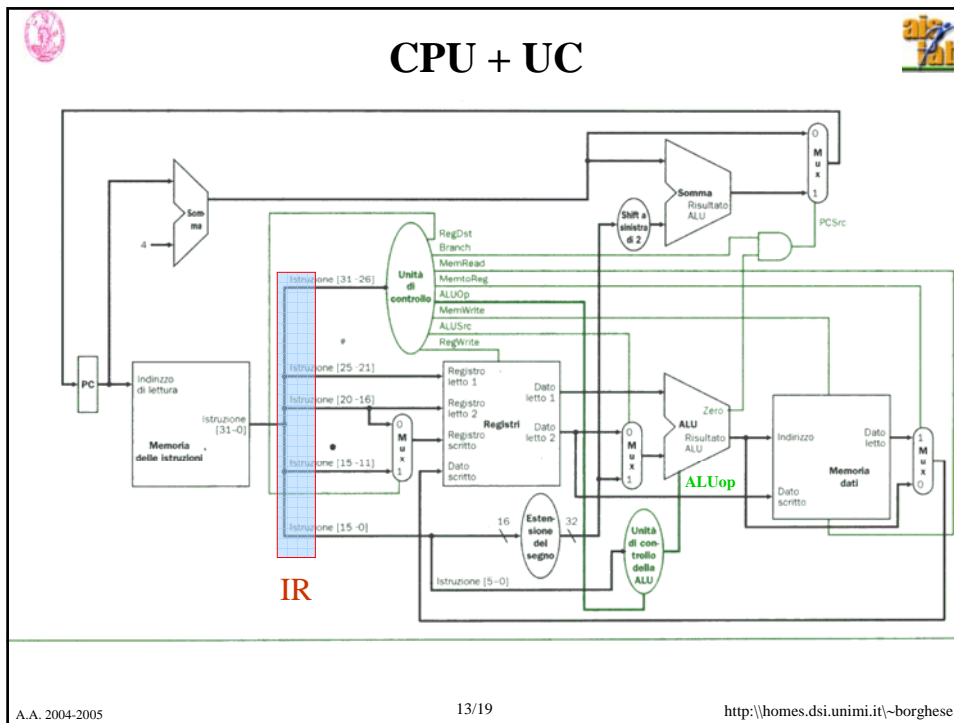


## Controllo del data-path



Istruzione (OpCode)	RegDst	ALUSrc	MemtoReg	Reg Write	Mem Read	Mem Write	Branch	ALUop
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





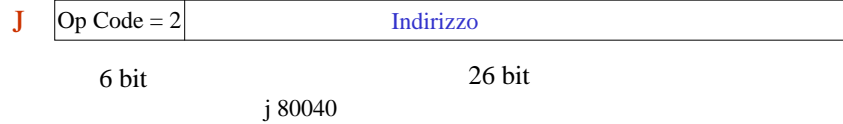
## Segnali di controllo su 1 bit

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 sostituito dall'uscita del sommatore che calcola PC+4 (condizionato all'uscita di ALU)	Il valore del PC viene sostituito dall'uscita del sommatore che calcola la destinazione del salto (condizionato all'uscita di ALU)
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

A.A. 2004-2005 14/19 http://homes.dsi.unimi.it/~borgnese



# L'istruzione jump



L'indirizzo di salto sarà determinato in due passi:

- A) Calcolo di Indirizzo = Indirizzo \* 4.
- B) Determinazione dell'indirizzo di salto come:

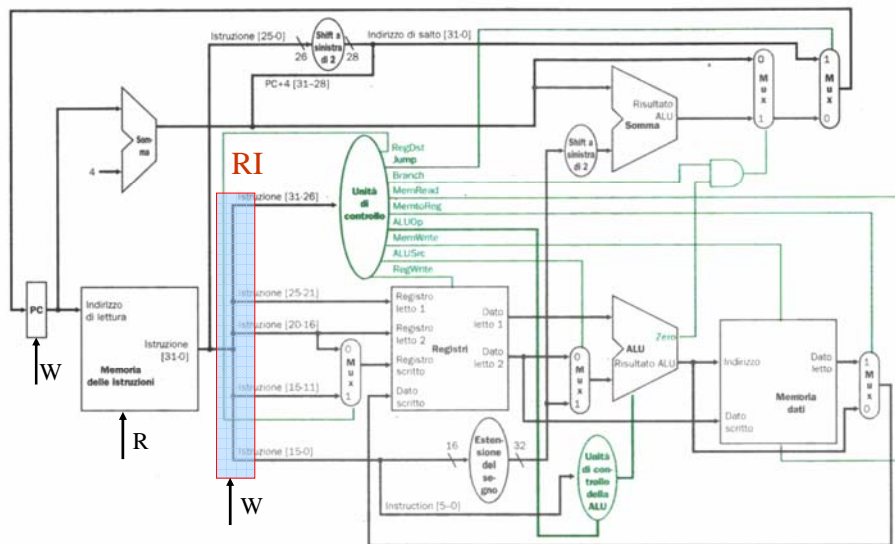
$$\begin{array}{r} \text{Base (PC)} \quad 0100\ 1000\ 0011\ 0001\ 1011\ 1011\ 1011\ 10\ 11 + \\ \text{Nuovo indirizzo} \quad 1000\ 0110\ 0111\ 0000\ 0000\ 0001\ 00\ 00 = \end{array}$$

$$\text{Indirizzo salto} \quad 0100\ 1000\ 0110\ 0111\ 0000\ 0000\ 0001\ 00\ 00$$

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	ALU op	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
beq (000100)	x	0	x	0	0	0	1	01	0
J (000010)	x	x	x	0	0	0	0	xx	1



## 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 sostituito dall'uscita del sommatore che calcola PC+4 (condizionato all'uscita di ALU)	Il valore del PC viene sostituito dall'uscita del sommatore che calcola la destinazione del salto (condizionato all'uscita di ALU)
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
Jump	Il valore del PC viene preso il PC è quello della branch oppure PC+4	Il valore del PC viene impostato al valore ottenuto dal campo dato della jump



## Sommario



Controllore della ALU

Unità di Controllo Principale