



# L'unità di controllo di CPU a singolo ciclo

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

Università degli Studi di Milano

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



## Sommario

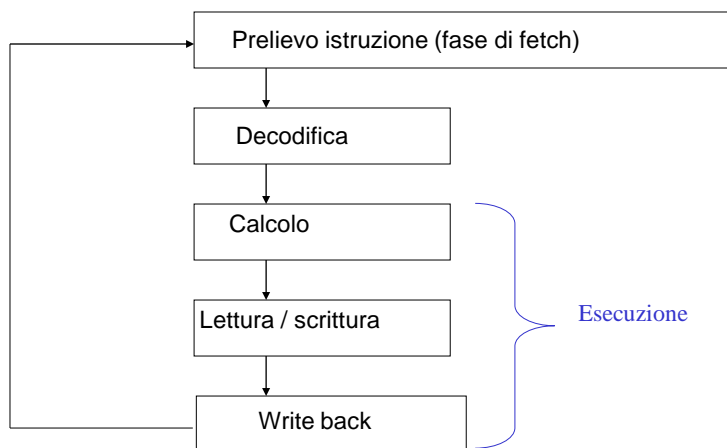
**Costruzione di una CPU per le istruzioni di tipo I (branch).**

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.

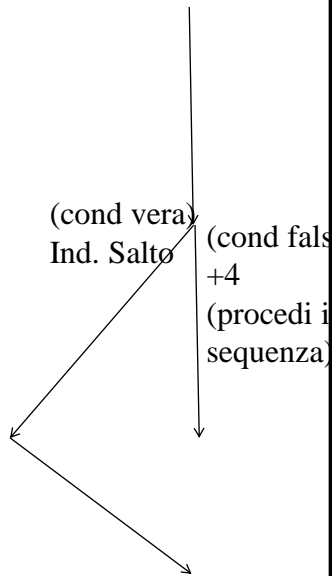
	6-bit	5-bit	5-bit	5-bit	5-bit	6-bit
R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	indirizzo		
J	op	indirizzo				



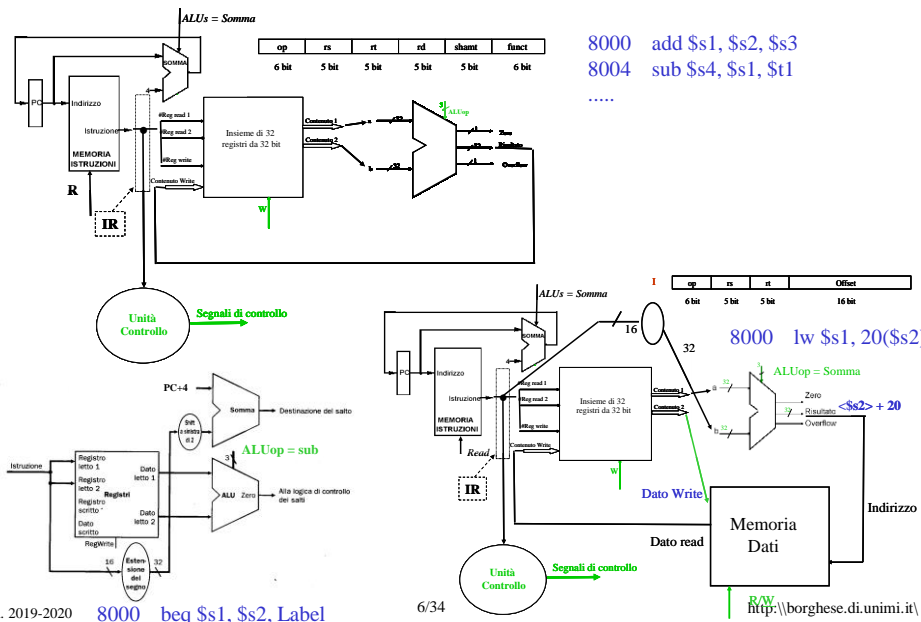
# Istruzioni di salto condizionato

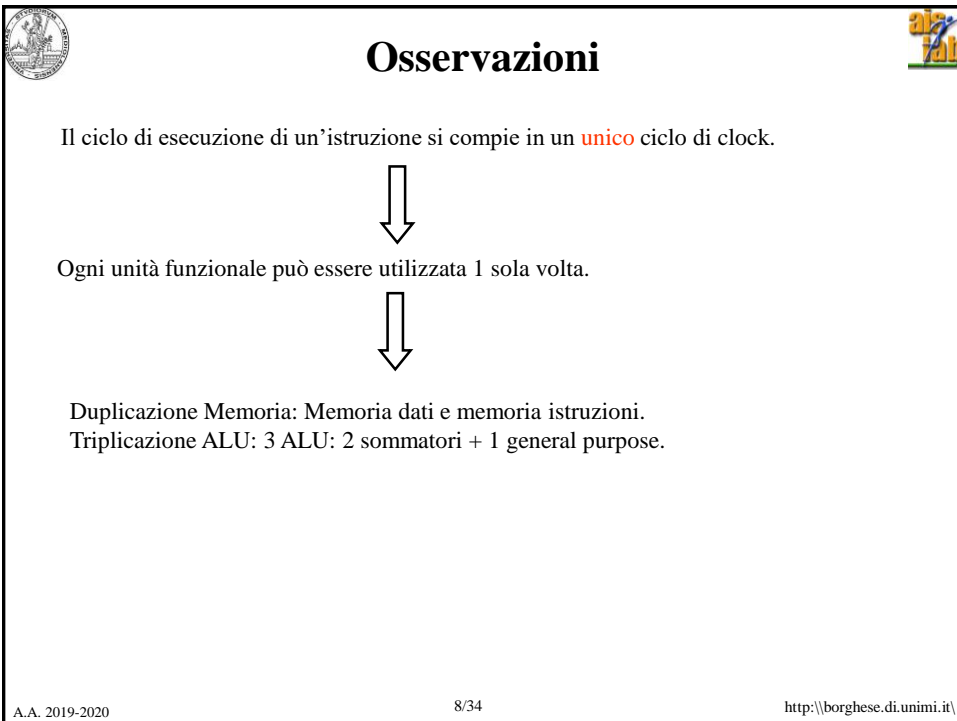
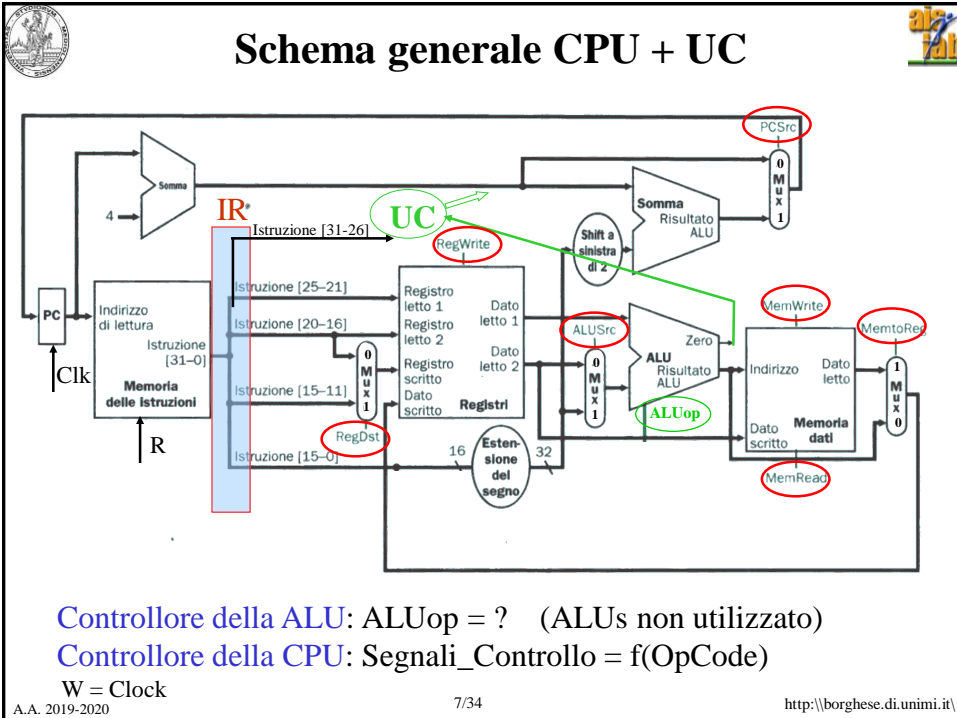


- Salti condizionati relativi:
  - `beq r1, r2, L1` (branch on equal)
  - `bne r1, r2, L1` (branch on not equal)
- Salti condizionati relativi:
  - Il flusso sequenziale di controllo cambia solo se la condizione è vera (`beq`)
  - Il calcolo del valore dell'etichetta **L1 (indirizzo di destinazione del salto)** avviene a partire dal Program Counter (PC).
  - Indirizzamento del tipo Base (PC) + Spiazzamento.



# CPU per l'esecuzione di istruzioni di tipo R/lw/beq







## Sommario



Costruzione di una CPU per le istruzioni di tipo I (branch).

UC della CPU

Control and Data path



## Segnali di controllo su 1 bit



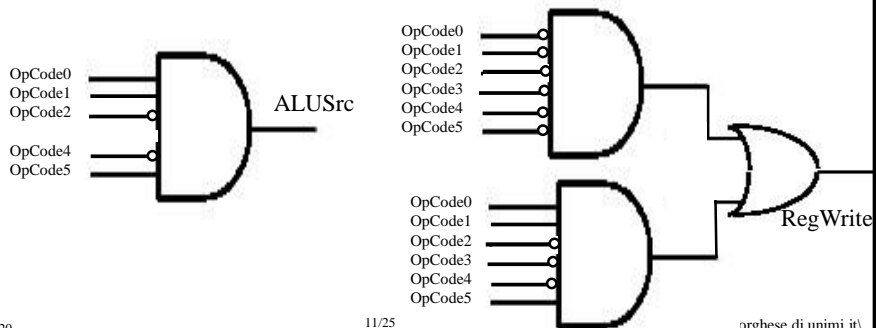
Nome del segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite	Nessuno	Nel registro specificato all'ingresso registro scritto del Register 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



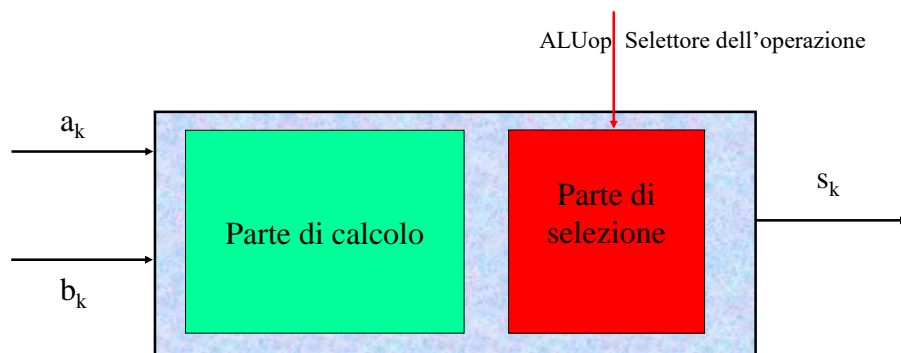
## Controllo del data-path



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)	1	1	0	1	0	0	0	00



## Struttura a 2 livelli di una ALU



Le operazioni consentite dalla ALU (selezionate tramite ALUOp):

and	000
or	001
add	010
sub	110
slt	111



## UC e ALU

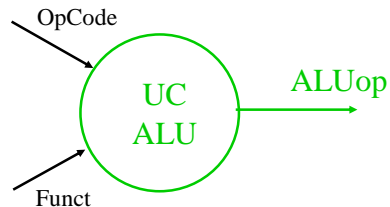


Data l'istruzione, l'UC deve inviare il comando opportuno alla ALU.

Campo Op Code  
Campo Funct

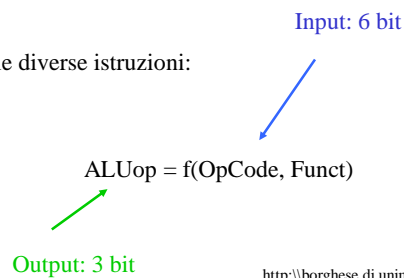
Le operazioni consentite dalla ALU:

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



Quali operazioni devono essere eseguite per le diverse istruzioni:

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

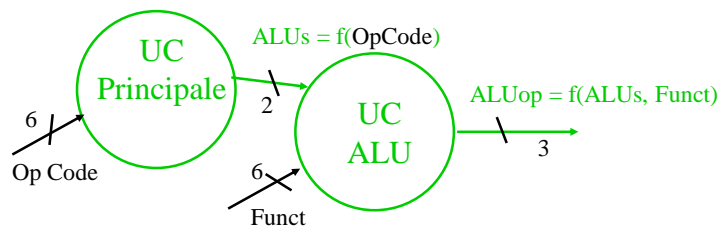


## Controllo gerarchico



Le operazioni consentite dalla ALU:

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

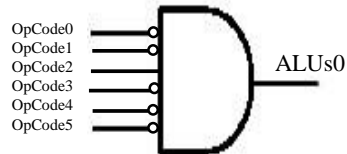
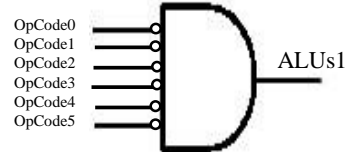


If (OpCode == R) then  
 Funct → ALUOp  
 Else  
 OpCode → ALUOp



## 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
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	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
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	0	1	0	0	0	0	0
or	0	0	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1
slt	0	0	0	0	0	0	1	0	1	0	1	0	1	0	1	1	1

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

SOP





# Sommario



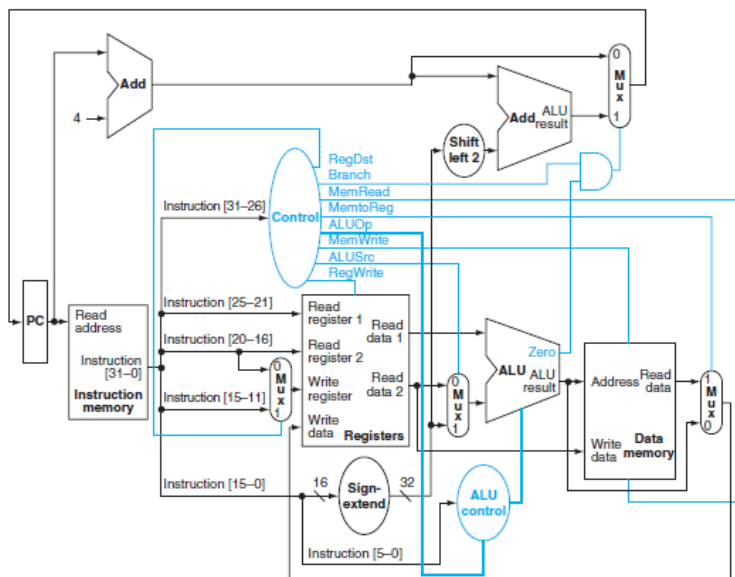
Costruzione di una CPU per le istruzioni di tipo I (branch).

UC della CPU

Control and Data path



# The control and data path

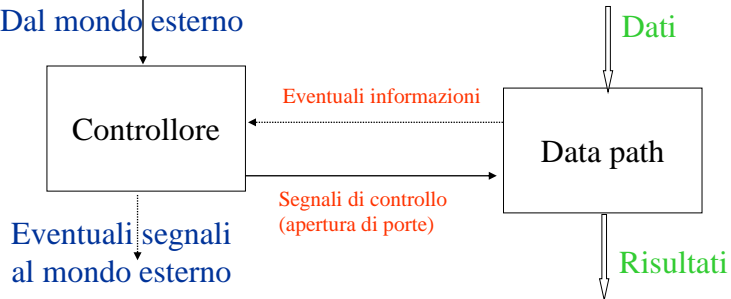




## Rapporto UC - Dati

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

Dal mondo esterno



*Fase comune nel ciclo di esecuzione:*

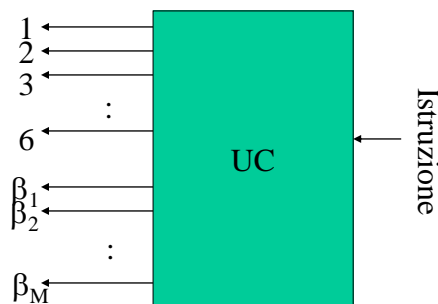
- Fase di fetch
- Decodifica (generazione dei segnali di controllo)

*Fase diversa:* Esecuzione (Calcolo, Accesso memoria, WriteBack)



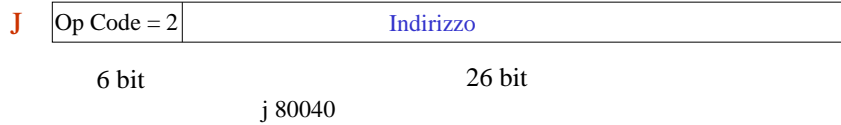
## L'unità di controllo

- 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.





# L'istruzione jump



L'indirizzo di salto sarà determinato in due passi:

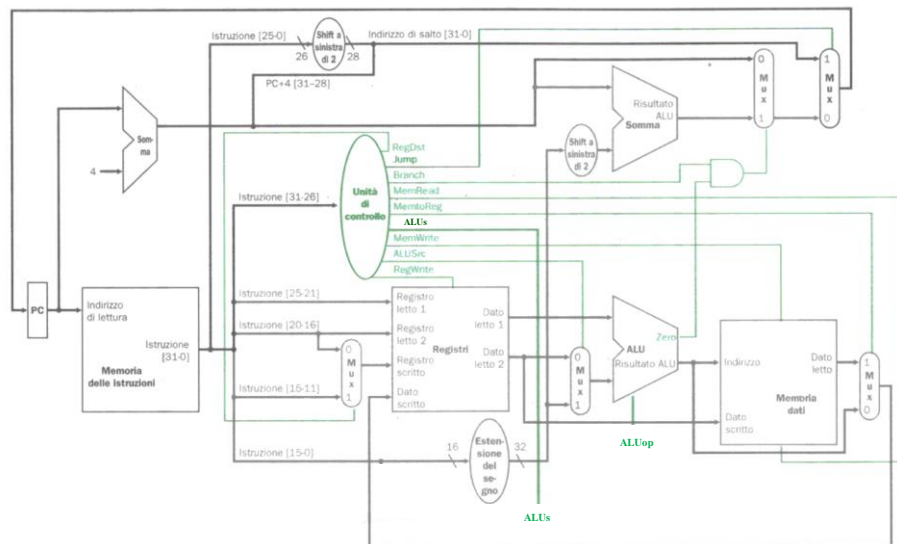
- A) Calcolo di Indirizzo = Indirizzo \* 4.
- B) Determinazione dell'indirizzo di salto come:

Base (PC)	0100	1000 0011 0001	1011 1011 1011 10 11 +
Nuovo indirizzo		1000 0110 0111	0000 0000 0001 00 00 =
Indirizzo salto	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

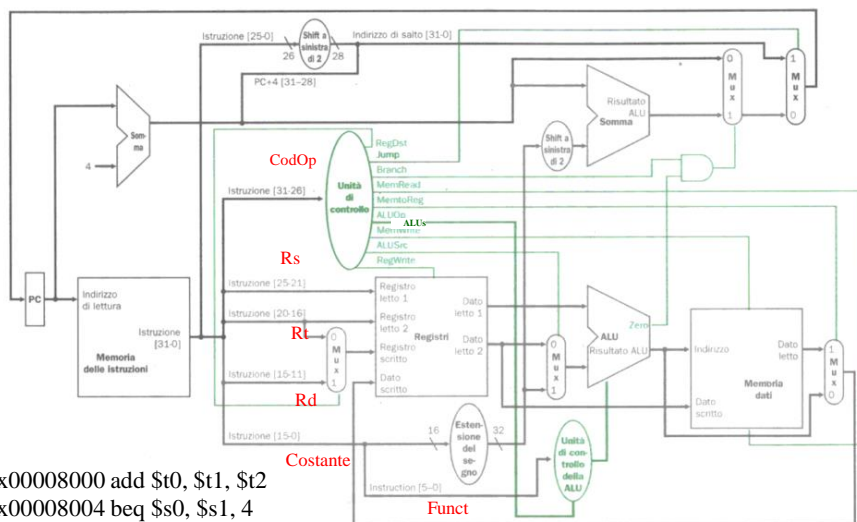


Istruzione (OpCode)	Reg Dst	ALU Src	Memto Reg	Reg Write	Mem Read	Mem Write	Branch	ALUs	Jump
R (00000)	1	0	0	1	0	0	0	10	0
lw (100011)	0	1	1	1	1	0	0	00	0
sw (101011)	x	1	x	0	0	1	0	00	0
beq (000100)	x	0	x	0	0	0	1	01	0
J (000010)	x	x	x	0	0	0	0	xx	1

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, \$t2



## Sommario



Costruzione di una CPU per le istruzioni di tipo I (branch).

UC della CPU

Control and Data path