



# Unità di controllo della pipeline

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

Università degli Studi di Milano

Riferimento al Patterson: 4.5, 4.6

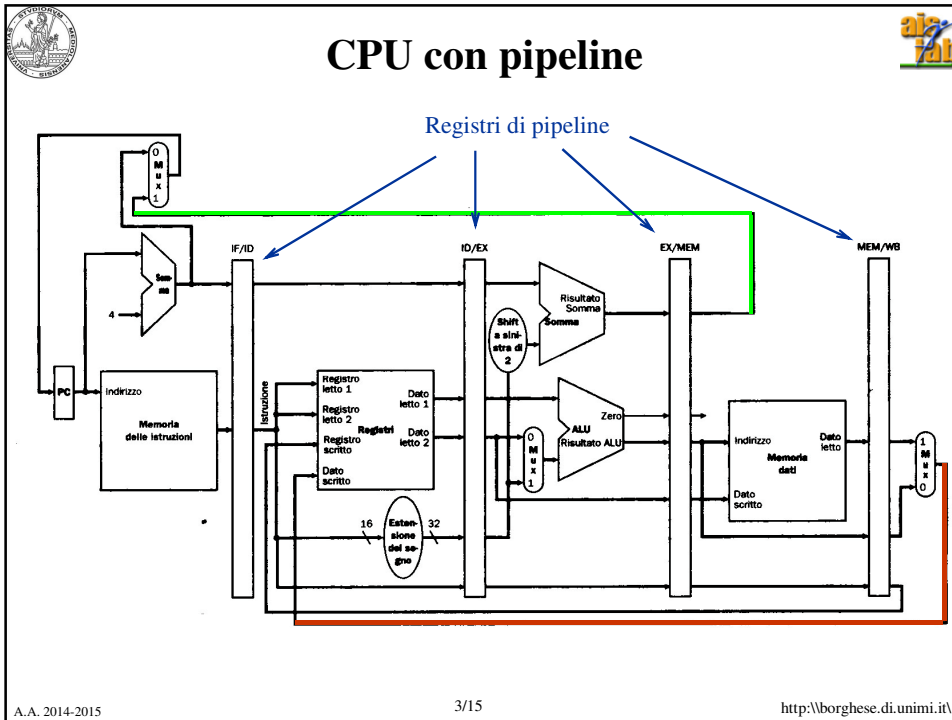


## Sommario

La CPU con pipeline

L'Unità di Controllo della pipeline

Le criticità



## Gli stadi di esecuzione

IF – Instruction Fetch  
 ID – Instruction Decode (e lettura register file)  
 EX – Esecuzione o calcolo dell'indirizzo di memoria.  
 MEM – Accesso alla memoria dati.  
 WB – Write Back (scrittura del risultato nel register file).

NB: I registri al termine di ogni fase prendono il nome dalle 2 fasi:  
 IF/ID      ID/EX      EX/MEM      MEM/WB

Perchè non c'è un registro WB/IF?

Il data-path procede da sx a dx.

A.A. 2014-2015 4/15 <http://borghese.di.unimi.it/>



## Il ruolo dei registri



Ciascuno stadio produce un risultato. La parte di risultato che serve agli stadi successivi deve essere memorizzata in un registro.

Il registro mantiene l'informazione anche se lo stadio in questione riutilizza l'unità funzionale.

Esempio: l'istruzione letta viene salvata nel registro IF/ID (cf. Instruction Register).



## Esempio di esecuzione



Cosa si trova nella pipeline durante l'esecuzione di questo segmento di codice (dati + controllo)?

0x4000	lw \$t1, 24(\$t2)	WB
0x4004	sw \$t2, 36(\$t1)	M
0x4008	beq \$t1, \$s2, 20	EX
0x400C	add \$s0, \$t1, \$s2	DEC
0x4010	sub \$t0, \$t1, \$t2	FF
0x4014	or \$s3, \$t4, \$t5	---

NB Occorre specificare il contenuto della parte master e slave dei registri di pipeline.

Data path che fluisce da sinistra a destra.

I registri letti dal register file sono specificati nella fase di ID, quello scritto nella fase di WB.



# Sommario



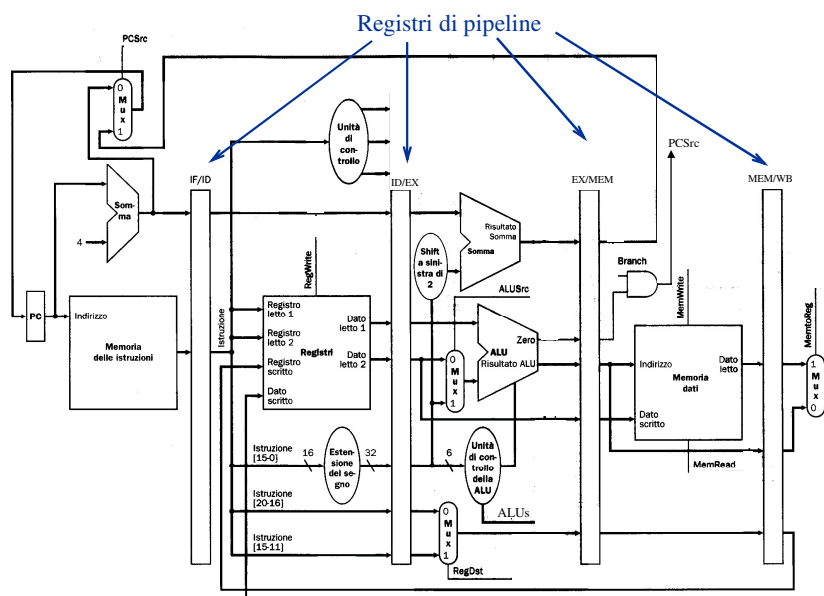
La CPU con pipeline

L'Unità di Controllo della pipeline

Le Criticità



# CPU con pipeline





## La UC della CPU con pipeline



Definizione dei segnali di controllo per ogni stadio, per ogni istruzione.

Definizione dell'UC in grado di generare correttamente questi segnali.



## Segnali di controllo su 1 bit



Nome 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

Scrittura PC e scrittura dei registri di pipeline ad ogni fronte di clock (ad ogni stadio).



## Osservazioni



Il contenuto di *rt* ed il numero di scrittura nel Register File (*rd*) vengono portati attraverso i vari stadi.

Nella fase di fetch e di decodifica non esistono segnali di controllo particolari.

I segnali di controllo particolari (legati alle diverse istruzioni) si possono così raggruppare:

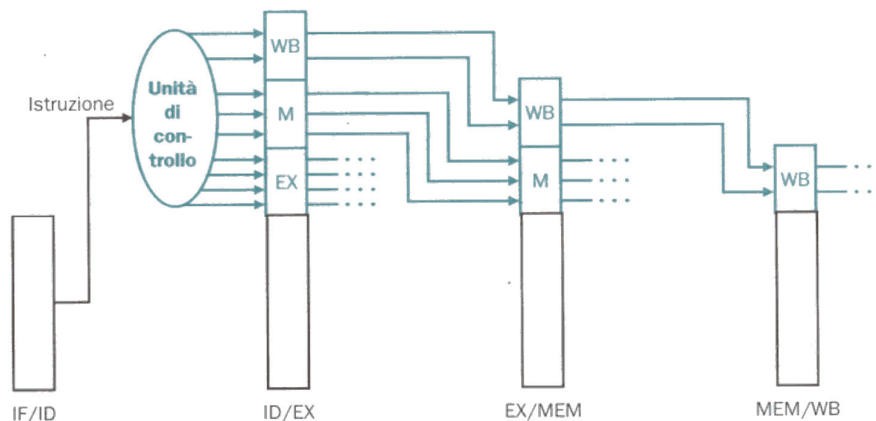
Istruzione	Exec				Memory			WB	
	Reg Dst	ALUs1	ALUs0	ALU Src	Branch	Mem Read	Mem Write	Reg Write	Mem2 Reg
Format-R	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X



## Generazione dei segnali di controllo



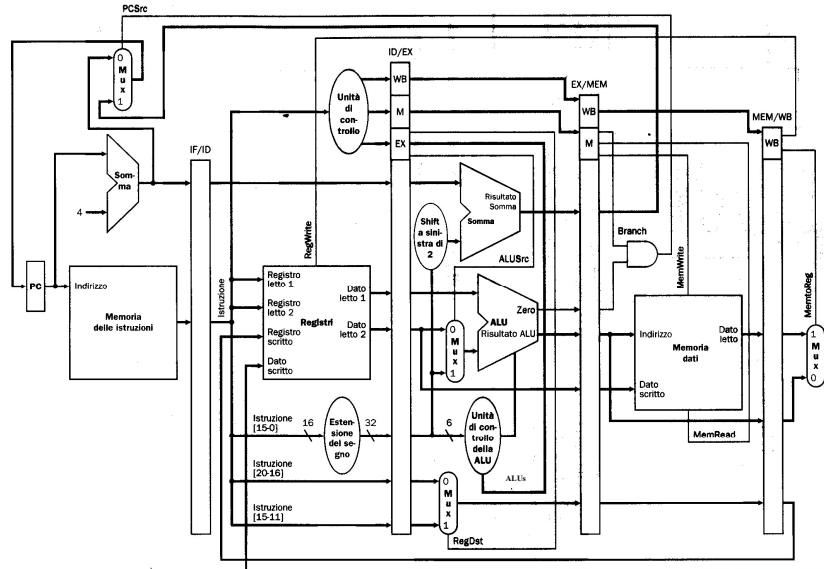
I segnali di controllo vengono generati nello stadio di decodifica e propagati.



Viene separata la fase di generazione dei segnali di controllo dalla fase di utilizzo.



## UC per pipeline



A.A. 2014-2015

13/15

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



## Esempio di esecuzione



Cosa si trova nella pipeline durante l'esecuzione di questo segmento di codice (dati + controllo)?

0x4000 lw \$t1, 24(\$t2)  
 0x4004 beq \$t1, \$s2, 20  
 0x4008 add \$s0, \$t1, \$s2  
 0x400C sw \$t2, 36(\$t1)  
 0x4010 sub \$t0, \$t1, \$t2  
 0x4014 or \$s3, \$t4, \$t5

NB Occorre specificare il contenuto della parte master e slave dei registri di pipeline.

I segnali di controllo fluiscono anch'essi da sinistra verso destra.

Il segnale di scrittura nel register file viene generato nella fase ID ma viene consumato nella fase WB.

Il segnale di scrittura nel register file proviene dalla fase di WB.

A.A. 2014-2015

14/15

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



# Sommario



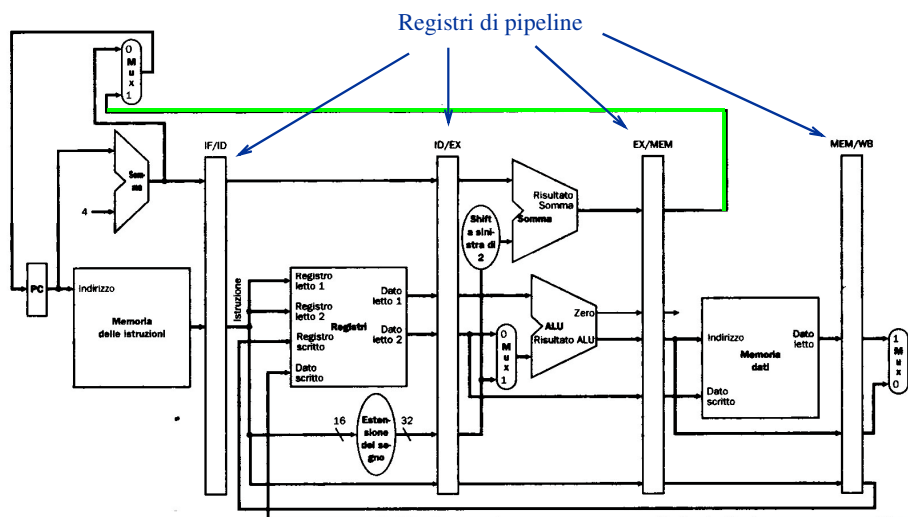
La CPU con pipeline

L'Unità di Controllo della pipeline

Le Criticità



# CPU con pipeline







## Criticità (hazard)



Un'istruzione non può essere eseguita nel ciclo di clock immediatamente successivo a quella precedente (mancano i dati necessari alla lavorazione di un qualche suo stadio).

### Strutturali:

- Dovrei utilizzare la stessa unità funzionale due volte nello stesso ciclo di clock (e.g. se non avessi duplicato la memoria)..

### Controllo:

- Dovrei prendere una decisione (sull'istruzione successiva) prima che l'esecuzione dell'istruzione corrente sia terminata (e.g. Istruzioni successive ad una branch).

### Dati:

- Dovrei eseguire un'istruzione in cui uno dei dati è il risultato dell'esecuzione di un'istruzione precedente.

*Esempio:*

```
add $s0, $t1, $t1
add $s2, $s0, $t3
```



## Soluzione delle criticità strutturali



Le criticità strutturali sono risolte con la duplicazione (suddivisione) delle unità funzionali.

**Triplicazione delle ALU**

**Duplicazione della Memoria (stessa memoria ma separazione della memoria dati dalla memoria istruzioni).**



## Sommario



La CPU con pipeline

L'Unità di Controllo della pipeline

Le Criticità