



# Pipeline – criticità e forwarding

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

Università degli Studi di Milano

Riferimento al Patterson: 4.5, 4.6



## Sommario

La CPU con 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.

Da dx a sx si ha la scrittura del PC e la scrittura nel Register File che creano criticità (vanno contro-corrente).

Supponiamo che ciascuno stadio abbia la sua unità di controllo.



## Il ruolo dei registri



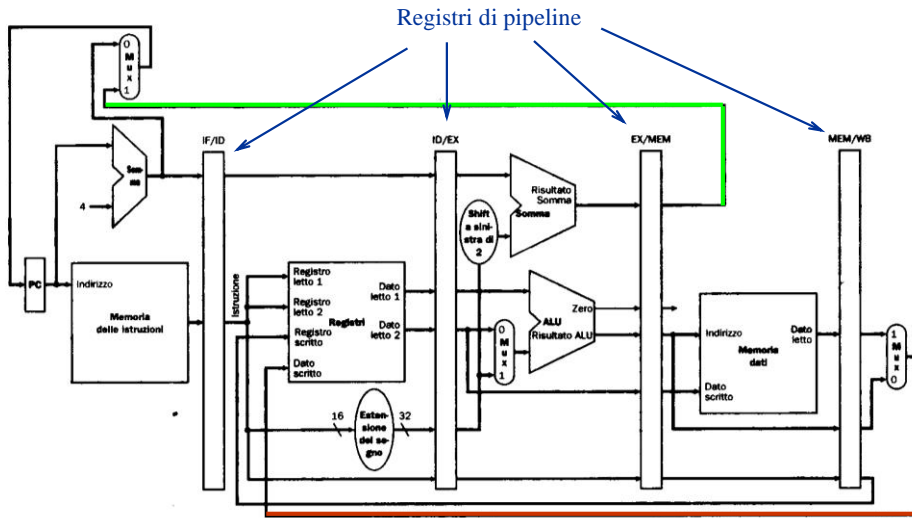
Ciascuno stadio produce un risultato. La parte di risultato che serve agli stadi successivi deve essere memorizzata in un registro.

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).



## 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).



## Esempio di Hazard sui dati



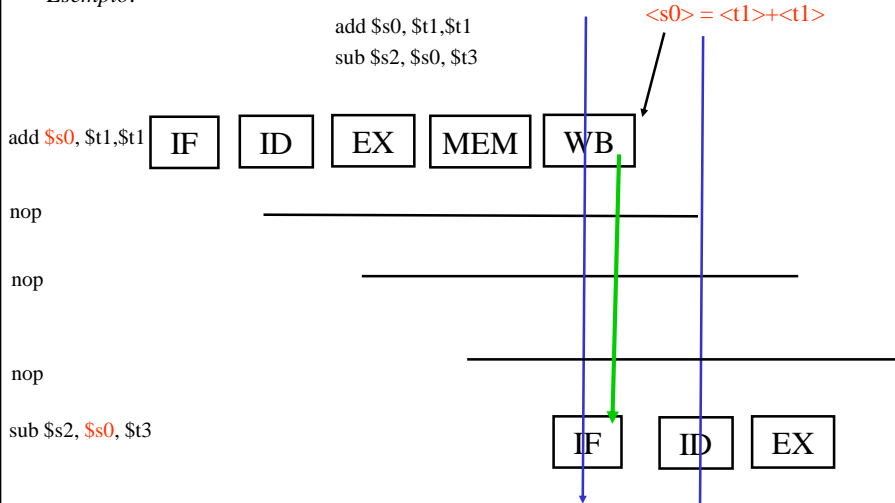
sub \$s2, \$s1, \$s2	IF	ID	EX \$1-\$2	MEM	WB s->\$2				
add \$t2, \$s2, \$s5		IF	ID	EX \$2 and \$5	MEM	WB s->\$t2			
or \$t3, \$s6, \$s2			IF	ID	EX \$6 or \$2	MEM	WB (s->\$t3)		
and \$t4, \$s2, \$3				IF	ID	EX \$2 & \$3	MEM	WB s->\$t4	
sw \$t5, 100(\$s2)					IF	ID	EX \$2+100	MEM \$t5	WB ->Mem



## Soluzione mediante stallo



Esempio:



Non eseguo istruzioni per 3 cicli di clock, la pipeline è in stallo per 3 cicli di clock, si formano 3 bolle (bubbles) nel funzionamento della pipeline.



## Hazard nei dati: soluzione tramite compilatore



<i>add \$s0, \$t1, \$t1</i>	<i>add \$s0, \$t1, \$t1</i>	<i>add \$s0, \$t1, \$t1</i>
<i>nop</i>	<i>nop</i>	<i>and \$t2, \$t0, \$t1</i>
<i>nop</i>	<i>nop</i>	<i>or \$t5, \$t3, \$t4</i>
<i>nop</i>	<i>nop</i>	<i>add \$s2, \$s7, \$t7</i>
<i>sub \$s1, \$s0, \$s0</i>	<i>sub \$s1, \$s0, \$s0</i>	<i>sub \$s2, \$s0, \$t3</i>
<i>and \$t2, \$t0, \$t1</i>	<i>and \$t2, \$t0, \$t1</i>	<i>sw \$15, 100(\$s2)</i>
<i>or \$t5, \$t3, \$t4</i>	<i>or \$t5, \$t3, \$t4</i>	
<i>add \$s2, \$s7, \$t7</i>	<i>add \$s2, \$s7, \$t7</i>	
<i>sw \$3, 100(\$t0)</i>	<i>sw \$3, 100(\$t0)</i>	

Spreco di 3 cicli di clock (in modo che la fase IF dell'istruzione *sub \$s2, \$s0, \$t3* vada a coincidere con la fase di WB della *add \$s0, \$t1, \$t1*).

Situazione troppo frequente perché la soluzione sia accettabile.

Il codice viene riorganizzato in fase di compilazione. Non sempre è possibile o efficace.



# Sommario



Criticità in una pipeline

Dipendenza tra i dati e propagazione

Propagazione a due passi



# Hazard sui dati



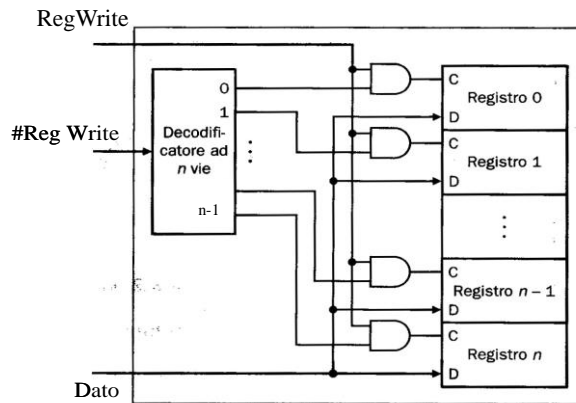
sub \$s2, \$s1, \$s2	IF	ID	EX \$s1-\$s2	MEM	WB s->\$2					
add \$t2, \$s2, \$s5		IF	ID	EX \$s2 and \$s5	MEM	WB s->\$t2				
or \$t3, \$s6, \$s2			IF	ID	EX \$s6 or \$s2	MEM	WB (s->\$t3)			
and \$t4, \$s2, \$s3				IF	ID	EX \$s2 & \$3	MEM	WB s->\$t4		
sw \$t5, 100(\$s2)					IF	ID	EX \$s2+100	MEM \$t5	WB ->Mem	

Con le frecce sono indicate le dipendenze, in blu gli hazard (tra sub e and, sub e add).

Il dato in \$s2 viene scritto nel Register File nella fase di WB della sub, è pronto al clock successivo. Non è ancora pronto quando viene effettuata la decodifica della and, della or e della add successiva.



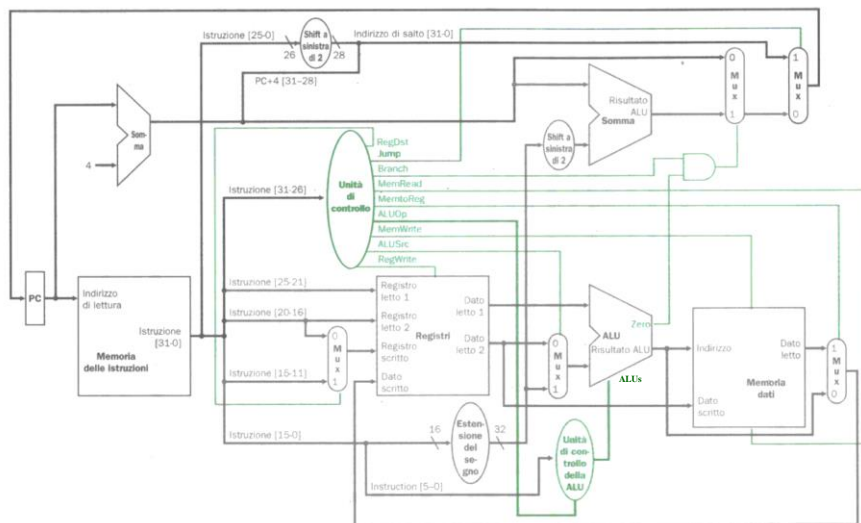
## E' una criticità il Register File?



No, a patto che utilizziamo Latch di tipo D e non flip-flop come nel caso della CPU singolo ciclo



## Pipeline a singolo ciclo





## Hazard sui dati



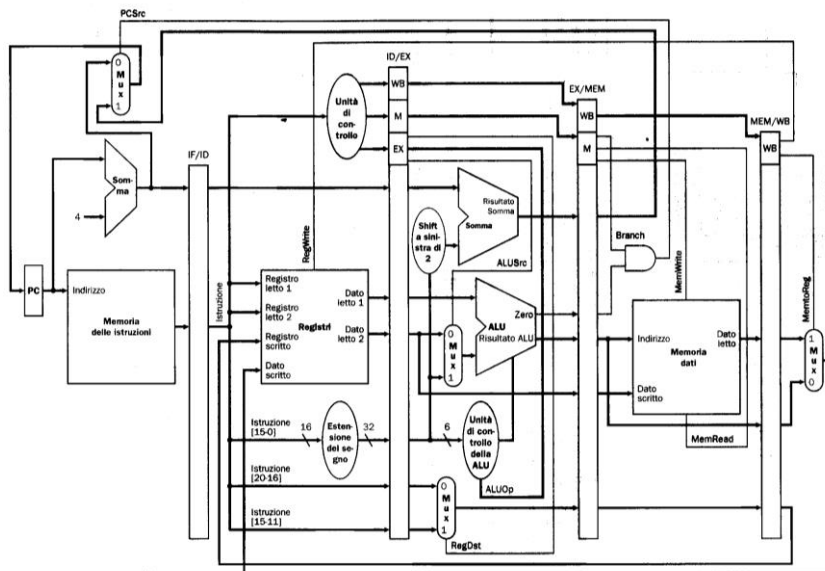
sub \$s2, \$s1, \$s2	IF	ID	EX \$s1-\$s2	MEM	WB s->\$s2				
add \$t2, \$s2, \$s5		IF	ID	EX \$s2 and \$s5	MEM	WB s->\$t2			
or \$t3, \$s6, \$s2			IF	ID	EX \$s6 or \$s2	MEM	WB (s->\$t3)		
and \$t4, \$s2, \$s3				IF	ID	EX \$s2 & \$s3	MEM	WB s->\$t4	
sw \$t5, 100(\$s2)					IF	ID	EX \$s2+100	MEM \$t5	WB ->Mem

Con le frecce sono indicate le dipendenze, in blu gli hazard (tra sub, e and e or), dopo la modifica del RegisterFile: il dato è disponibile in lettura, già nella prima parte del clock. Il dato in \$s2 viene scritto nel Register File nella fase di WB della sub, è pronto al clock successivo. Non è ancora pronto quando viene effettuata la decodifica della and e della or successiva.

mi.it\



## CPU con pipeline







## Soluzione architetturale della criticità sui dati



La criticità nei dati ha a che fare essenzialmente con la disponibilità di dati corretti.

Identificazione della criticità (funzione del tipo di istruzione e dei registri coinvolti).

Correzione della situazione: propagazione a ritroso (negli stadi della pipeline = in avanti nel tempo) su data-path alternativi dei dati richiesti.



## Hazard sui dati: rilevamento della criticità



sub \$s2, \$s1, \$s2	IF	ID	EX \$s1- \$s2	MEM	WB s->\$s2				
add \$t2, \$s5, \$s2		IF	ID	EX \$s5 + \$s2	MEM	WB Ris -> \$t2			
or \$t3, \$s2, \$s6			IF	ID	EX \$s2 or \$s6	MEM	WB Ris -> \$t3)		
and \$t8, \$s2, \$s3				IF	ID	EX	MEM	WB	

La criticità riguarda le due istruzioni immediatamente seguenti alla sub.



## Hazard sui dati: rilevamento della criticità



sub \$s2, \$s1, \$s2	IF	ID	EX \$s1- \$s2	MEM Risultato pronto	WB s->\$s2				
add \$t2, \$s5, \$s2		IF	ID	EX \$s5 + \$s2	MEM	WB Ris -> \$t2			
or \$t3, \$s2, \$s6			IF	ID	EX \$s2 or \$s6	MEM	WB Ris -> \$t3)		

Nella pipeline il contenuto corretto di s2 è già disponibile al termine della fase di EX (inizio della fase di MEM) della sub.

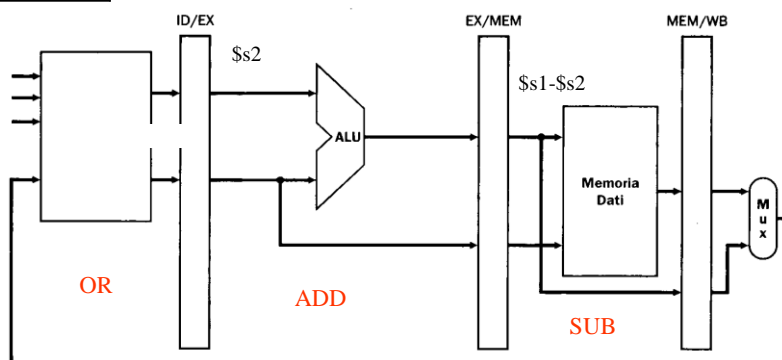
Il contenuto di s2 viene utilizzato nella fase di EX della add. In questa fase il contenuto corretto di s2 è già stato calcolato anche se non ancora scritto nel register file.



## Hazard nei dati: forwarding



```
sub $s2, $s1, $s2
add $t2, $s2, $s5
or $t3, $s6, $s2
```

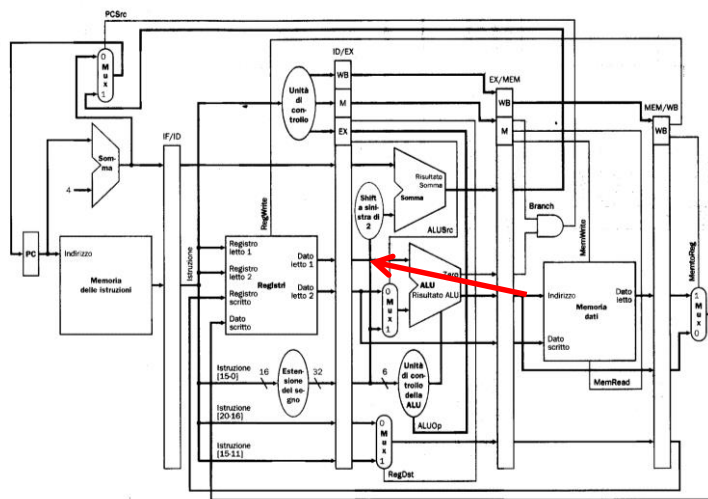




## Soluzione della criticità



```
sub $s2, $s1, $s2
add $t2, $s5, $s2
```



Cammino di retro-propagazione (feed-forward) del risultato della ALU.  
Il cammino non deve essere utilizzato sempre.

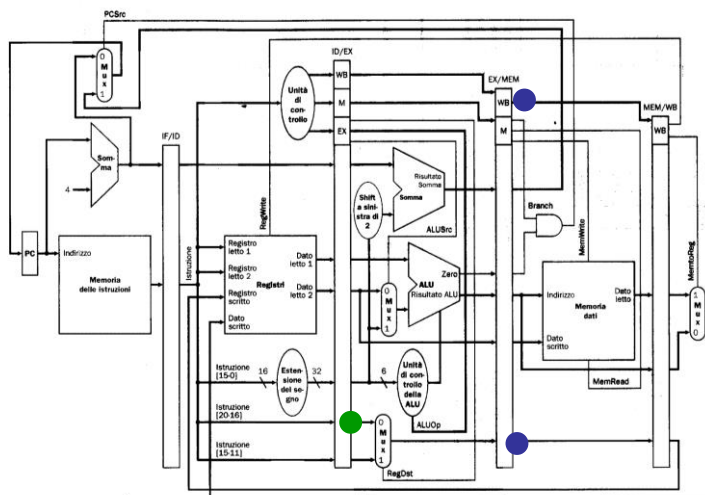


## Identificazione delle criticità – EX/MEM



```
sub $s2, $s1, $s2
add $t2, $s5, $s2
```

- 1a. EX/MEM.RegistroRd = ID/EX.RegistroRt
- 1b. EX/MEM.RegWrite



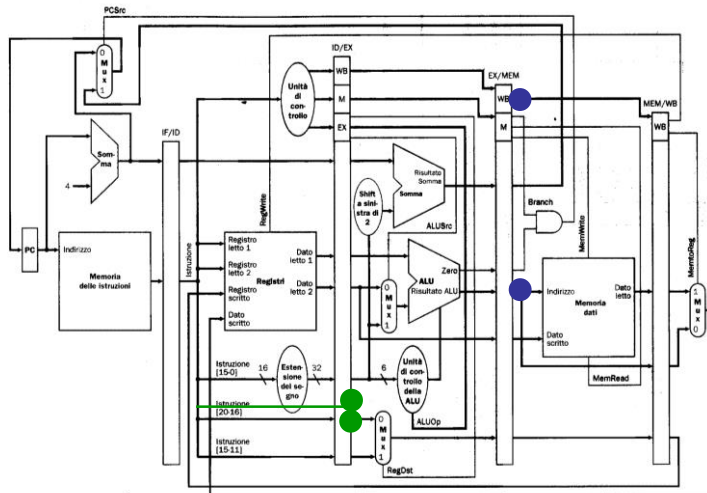


# Identificazione delle criticità – EX/MEM



sub \$s2, \$s1, \$s2  
add \$t2, \$s2, \$s2

- 1a. EX/MEM.RegistroRd = ID/EX.RegistroRs
- 1b. EX/MEM.RegistroRd = ID/EX.RegistroRt
- 1c. EX/MEM.RegWrite



A.A. 2018-2019

mi.it



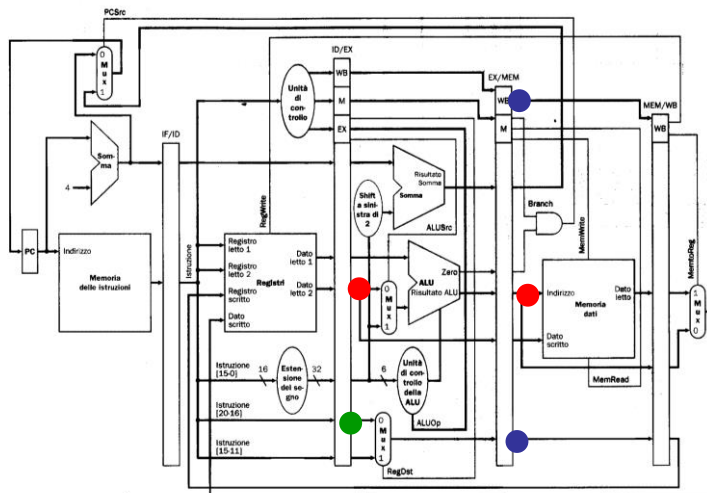
# Abbiamo identificato il problema, dobbiamo ora risolverlo



sub \$s2, \$s1, \$s2  
add \$t2, \$s5, \$s2

IF ( (EX/MEM.RegistroRd == ID/EX.RegistroRs) &&  
(EX/MEM.RegWrite) ) Then

HazardRt = 1;



A.A. 2018-2019

mi.it

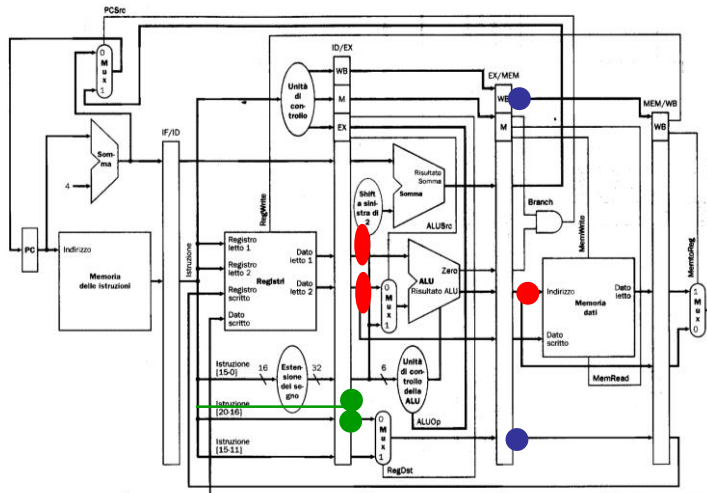


## Soluzione della criticità



sub \$s2, \$s1, \$s2  
 add \$t2, \$s2, \$s2

IF ( (EX/MEM.RegistroRd == ID/EX.RegistroRs) &&  
 (EX/MEM.RegisterWrite) ) Then MuxA = 1;  
 IF ( (EX/MEM.RegistroRd == ID/EX.RegistroRt) &&  
 (EX/MEM.RegisterWrite) ) Then MuxB = 1;



A.A. 2018-2019

mi.it\



## Posizionamento dei Mux



Il mux che seleziona tra il dato che proviene dal registro ID/EX ed i dati retropropagati dai registri EX/MEM oppure MEM/WB deve essere inserito prima o dopo il mux che seleziona tra contenuto del registro target e l'estensione a 32 bit del campo costante e perché?

Il campo RT viene preso prima del mux nello stadio EX perchè?

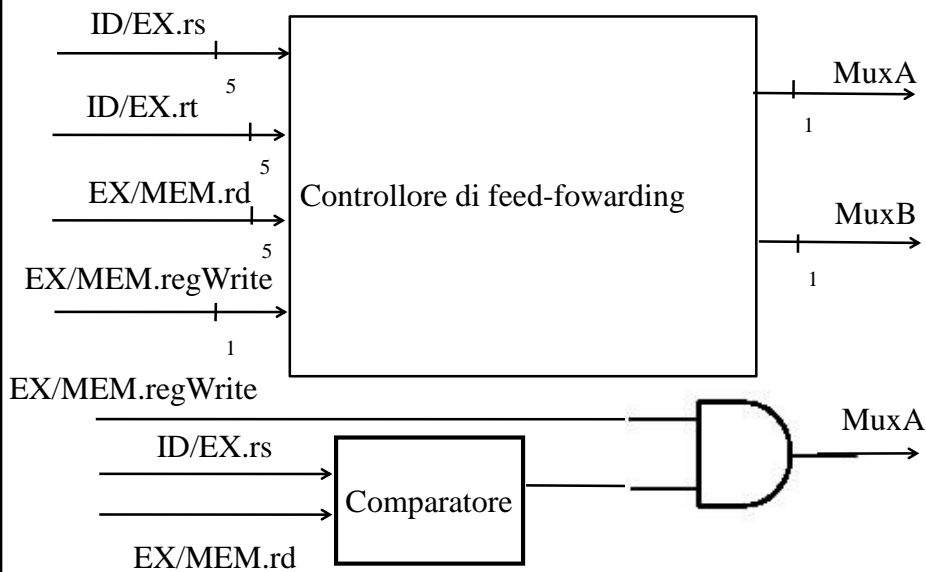
A.A. 2018-2019

26/42

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



## Unità di controllo della propagazione



## Sommario



Criticità in una pipeline

Dipendenza tra i dati e propagazione

**Propagazione a due passi**



## Hazard sui dati: rilevamento della criticità



sub \$s2, \$s1, \$s2	IF	ID	EX \$s1- \$s2	MEM Risultato pronto	WB s->\$s2				
add \$t2, \$s5, \$s2		IF	ID	EX \$s5 ≠ \$s2	MEM	WB Ris -> \$t2			
or \$t3, \$s2, \$s6			IF	ID	EX \$s2 or \$s6	MEM	WB Ris -> \$t3)		
and \$t8, \$s2, \$s4				IF	ID	EX	MEM	WB	

La criticità riguarda le due istruzioni immediatamente seguenti alla sub.

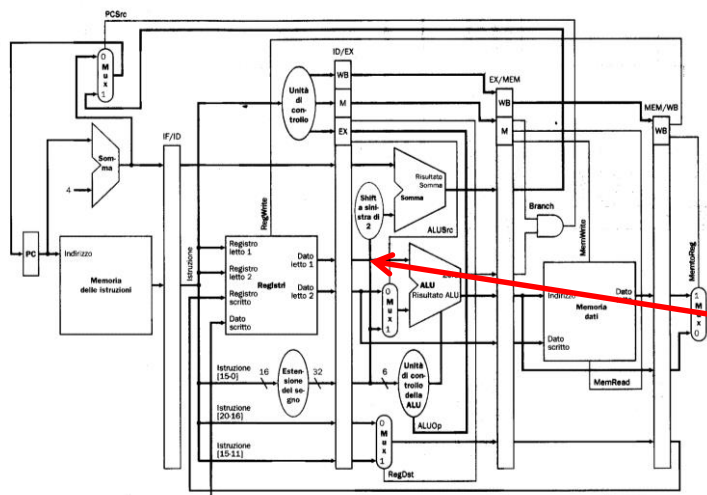
Dobbiamo risolvere la criticità sulla or



## Soluzione della criticità



sub \$s2, \$s1, \$s2  
add \$t2, \$s5, \$s2



Cammino di retro-propagazione (feed-forward) del risultato della ALU.  
Il cammino non deve essere utilizzato sempre.

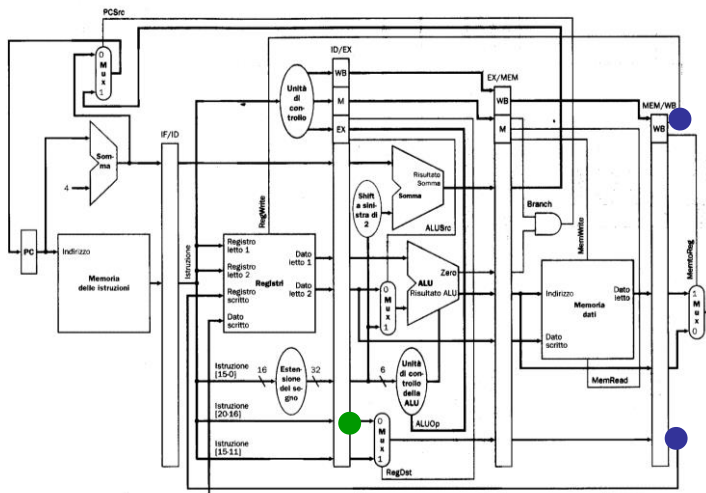


# Identificazione delle criticità – MEM/WB



sw \$s1, 100(\$t1)  
 sub \$s2, \$s1, \$s2  
 add \$t2, \$s2, \$s5  
 or \$t3, \$s6, \$s2

2a. MEM/WB.RegistroRd = ID/EX.RegistroRt  
 2b. MEM/WB.RegisterWrite



A.A. 2018-2019

mi.it

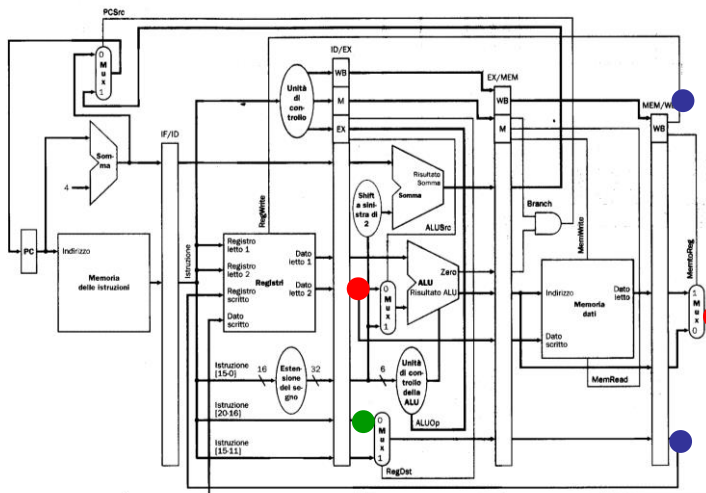


# Soluzione della criticità



sub \$s2, \$s1, \$s2  
 add \$t2, \$s5, \$s2

IF ( (EX/MEM.RegistroRd == EX/MEM.RegistroRt) &&  
 (EX/MEM.RegisterWrite) Then HazardRt = 1;



A.A. 2018-2019

mi.it



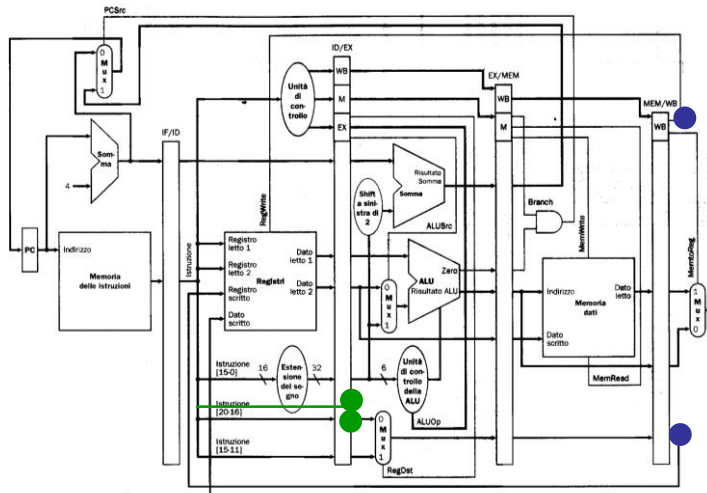


# Identificazione delle criticità – MEM/WB



sub \$s2, \$s1, \$s2  
 add \$t2, \$s2, \$s5  
 or \$t3, \$s6, \$s2

- 1a. MEM/WB.RegistroRd = ID/EX.RegistroRs
- 1b. MEM/WB.RegistroRd = ID/EX.RegistroRt
- 1c. MEM/WB.RegWrite



A.A. 2018-2019

mi.it

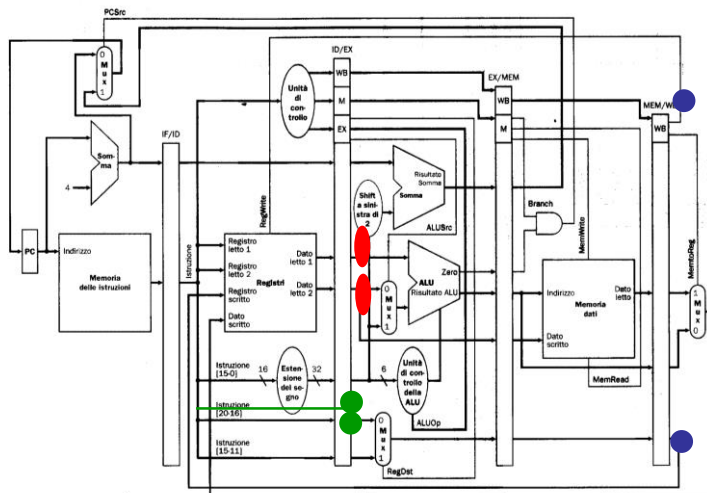


# Soluzione della criticità



sub \$s2, \$s1, \$s2  
 add \$t2, \$s2, \$s5  
 or \$t3, \$s6, \$s2

- IF ( (MEM/WB.RegistroRd == ID/EX.RegistroRs) &&  
 (MEM/WB.RegisterWrite) ) Then MuxA = 1;
- IF ( (MEM/WB.RegistroRd == ID/EX.RegistroRt) &&  
 (MEM/WB.RegisterWrite) ) Then MuxB = 1;



A.A. 2018-2019

mi.it



## Relazione tra forwarding e contenuto del registro ID/EX



Nel normale funzionamento, il registro ID/EX contiene quanto letto dal Register File.

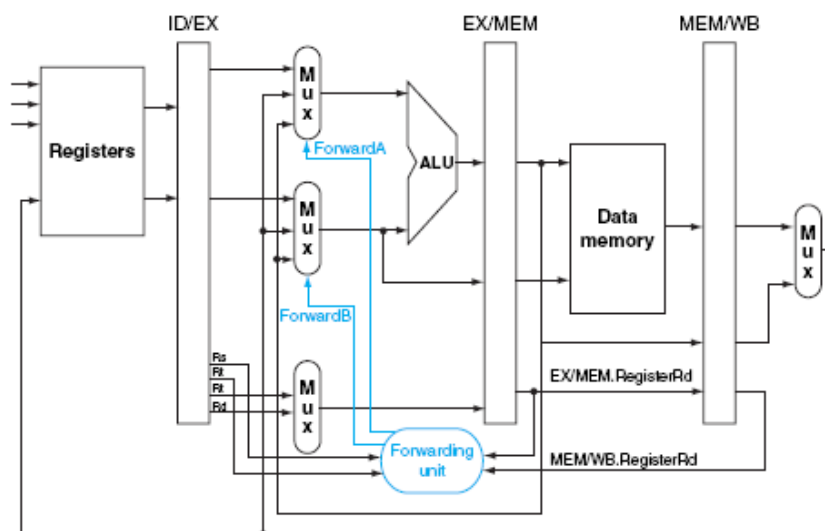
Quando abbiamo forwarding, quello che viene letto dal registro ID/EX nella fase di esecuzione viene sovrascritto da quanto letto dal registro EX/MEM o MEM/WB.

Nel registro EX/MEM è contenuto il risultato dell'operazione eseguita all'istante precedente.

Nel registro MEM/WB è contenuto il risultato dell'operazione eseguita 2 istanti precedenti.



## Circuito di propagazione







## Unità di controllo del forwarding



Deve controllare che la criticità sia effettiva (che l'istruzione precedente scriva il RegisterFile).

E' attiva nella fase di esecuzione (EX) ed implementa le seguenti funzioni:

*Dato preso dalla fase MEM:*

IF (ID/EX.RegistroRs == EX/MEM.RegistroRd) AND (EX/MEM.RegWrite)  
ALUSrcA = <EX/MEM.Data>

IF (ID/EX.RegistroRt == EX/MEM.RegistroRd) AND (EX/MEM.RegWrite)  
ALUSrcB = <EX/MEM.Data>

*Dato preso dalla fase WB:*

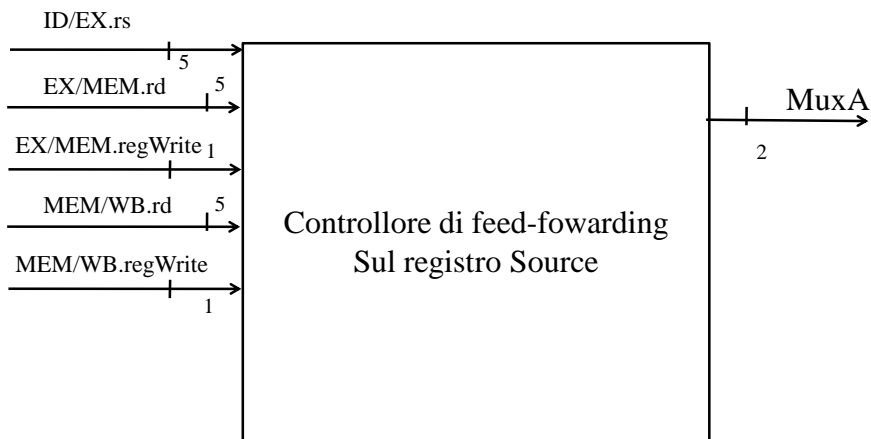
IF (ID/EX.RegistroRs == MEM/WB.RegistroRd) AND (MEM/WB.RegWrite)  
ALUSrcA = <MEM/WB.Data>

IF (ID/EX.RegistroRt == MEM/WB.RegistroRd) AND (MEM/WB.RegWrite)  
ALUSrcB = <MEM/WB.Data>

Cosa succede se è rilevata una criticità su RS (RT) sia con la fase di MEM che di WB?



## Unità di controllo della propagazione



sub \$s2, \$s1, \$s0  
add \$s2, \$s2, \$s6  
or \$t2, \$s2, \$s5

Circuito che decide la precedenza se il dato deve essere propagato sia da M che da WB



CodOp = 8

rs = 21

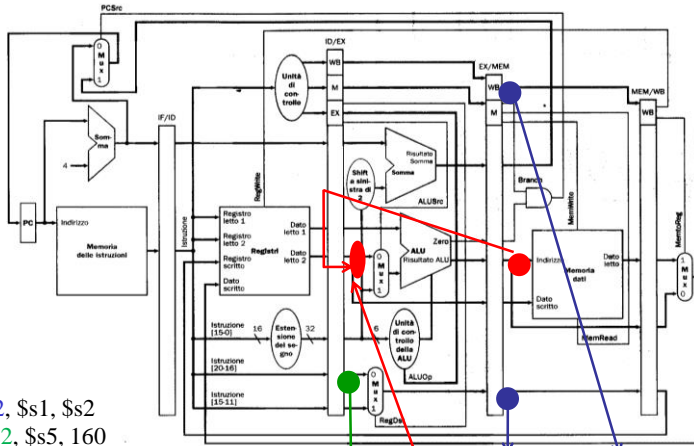
rt = 18

COST = 160

# Criticità su addi



EX/MEM.RegWrite = 1 → Propago s1-s3. Commetto un errore?  
EX/MEM.rd = ID/EX.rt = 18



```
sub $s2, $s1, $s2
addi $s2, $s5, 160
```

Unità di controllo della propagazione



# Sommario



Criticità in una pipeline

Dipendenza tra i dati e propagazione

Propagazione a due passi