



# Hazard e forwarding

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

Università degli Studi di Milano

Riferimento al Patterson, 5th edition: 4.7



## Sommario

Criticità in una pipeline

Dipendenza tra i dati e propagazione

Propagazione a due passi



## 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

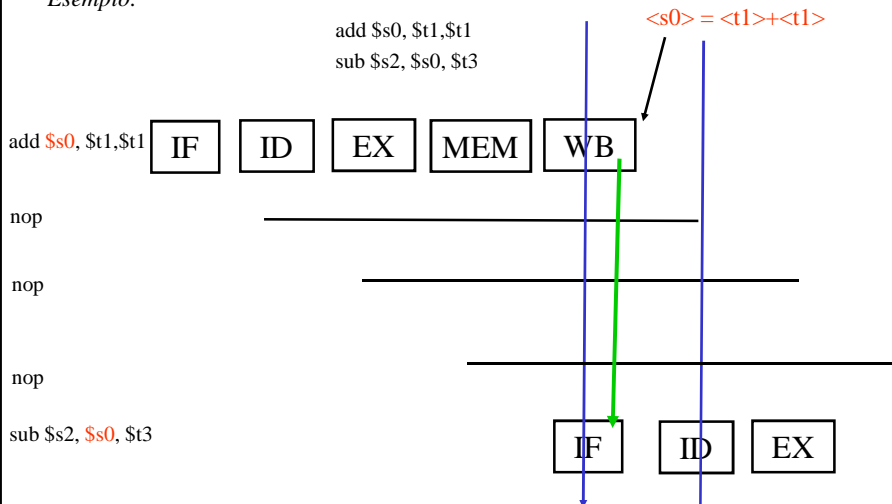


## Soluzione mediante stallo



Esempio:

add \$s0, \$t1, \$t1  
 sub \$s2, \$s0, \$t3



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 \$t5, 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

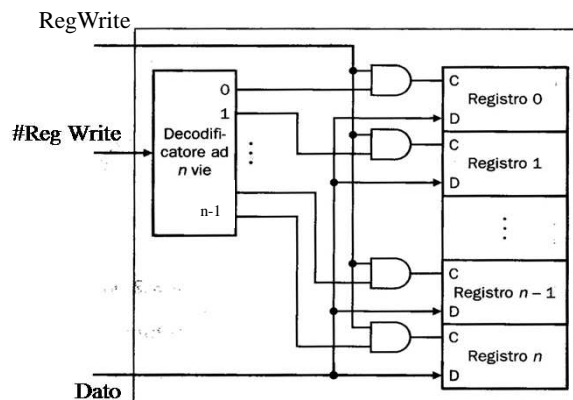
sub \$s2, \$s1, \$s2	IF	ID	EX \$s1-\$s2	MEM	WB s->\$2				
add \$t2, \$s2, \$s5		IF	ID	EX \$s2 and \$s5	MEM	WB s->\$t2			
or \$t3, \$s6, \$s2			IF	ID	EX \$s6 or \$s2	MEM (s->\$t3)			
and \$t4, \$s2, \$s3				IF	ID	EX \$s2 & \$s3	MEM	WB s->\$t4	
sw \$t5, 100(\$s2)					IF	ID	EX \$s2+100	MEM \$t5 ->Mem	WB

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.

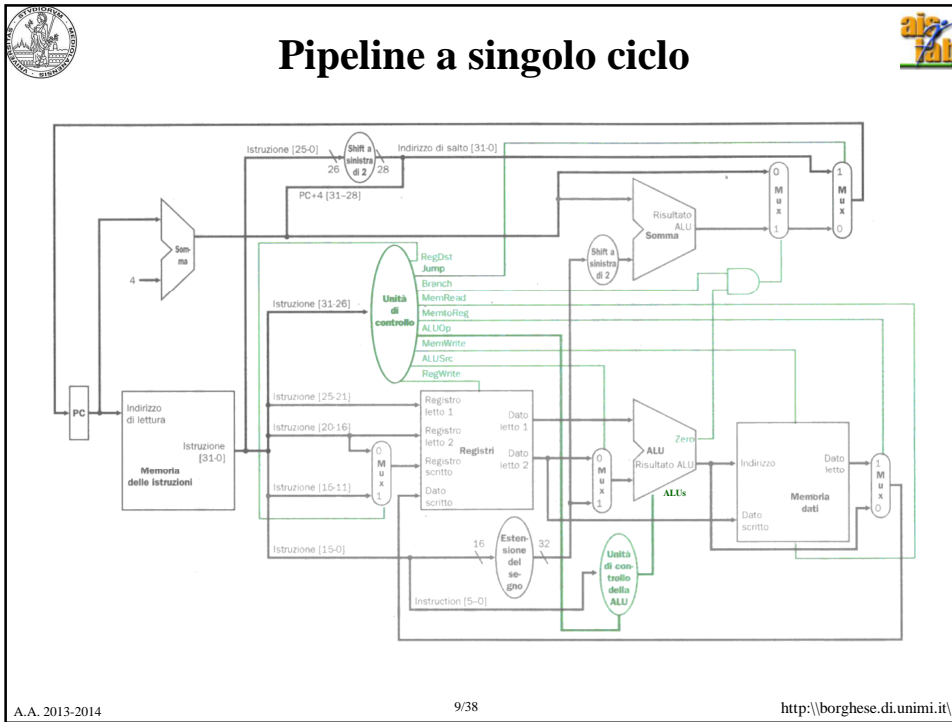
imi.it\



## 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



### Hazard sui dati

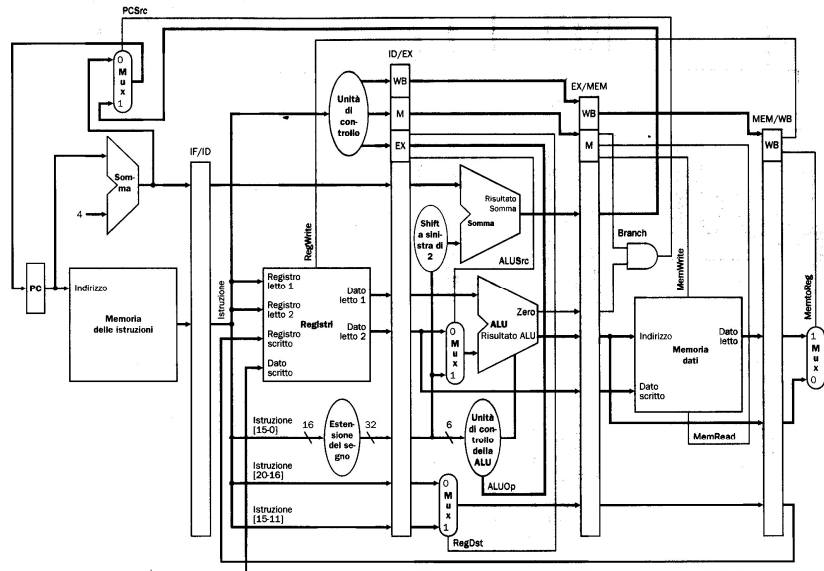
sub \$s2, \$s1, \$s2	IF	ID	EX \$s1-\$s2	MEM	WB ->\$s2				
add \$t2, \$s2, \$s5		IF	ID \$s2 and \$s5	EX	MEM	WB s->\$t2			
or \$t3, \$s6, \$s2			IF	ID \$s6 or \$s2	EX	MEM (s->\$t3)	WB		
and \$t4, \$s2, \$s3				IF	ID \$s2 & \$s3	EX	MEM	WB s->\$t4	
sw \$t5, 100(\$s2)					IF	ID	EX \$s2+100	MEM \$t5	WB ->Mem

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.



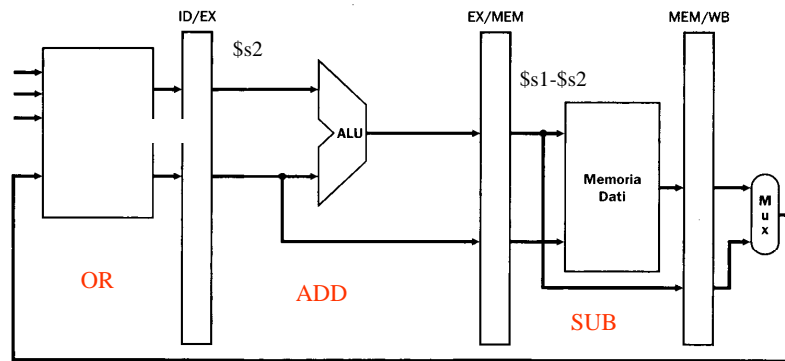
## Hazard nei dati: forwarding



```

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

```



A.A. 2013-

it\



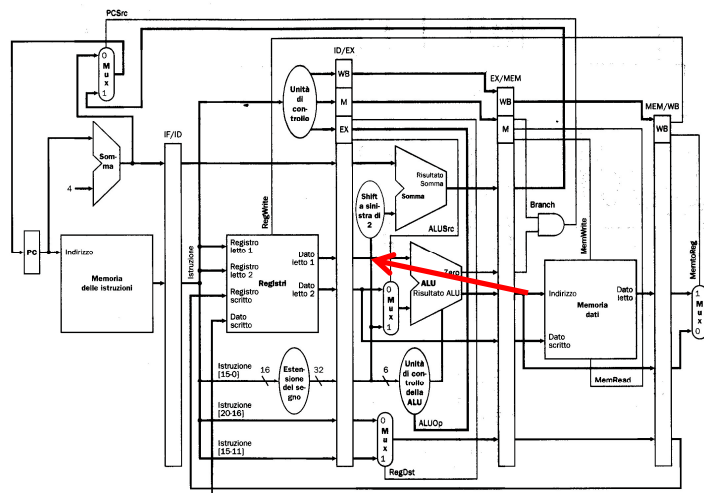
## Soluzione della criticità



```

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

```



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

A.A. 2013-2014

16/38

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



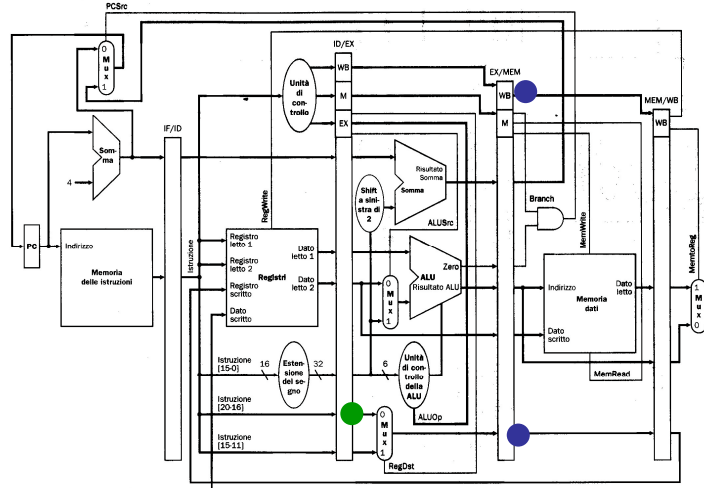


# Identificazione delle criticità – EX/MEM



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

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



A.A. 2013-2014

mi.it

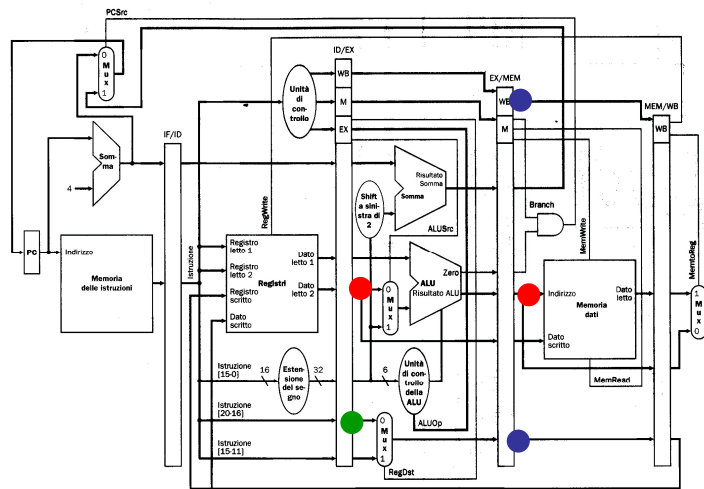


# Abbiamo identificato il problema, dobbiamo ora risolverlo



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

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



A.A. 2013-2014

mi.it

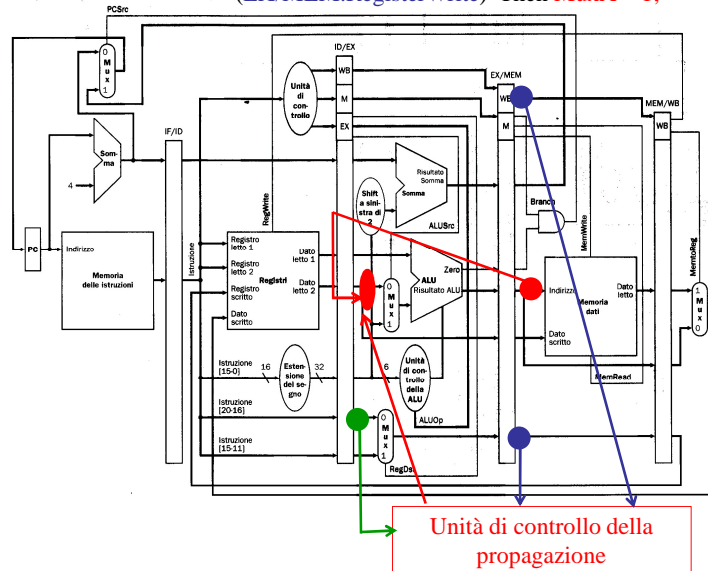


## CPU con propagazione



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

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



## 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?

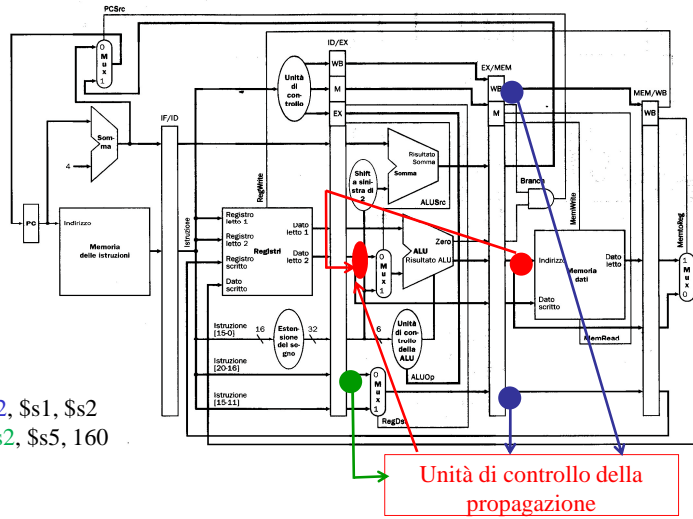


CodOp = 8
rs = 21
rt = 18
Cost = 160

## Criticità su addi

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

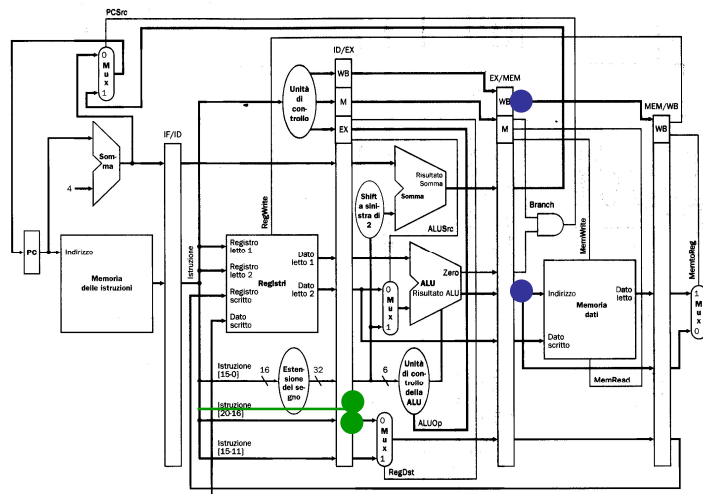
sub \$s2, \$s1, \$s2  
 addi \$s2, \$s5, 160



## 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





# Soluzione della criticità



```

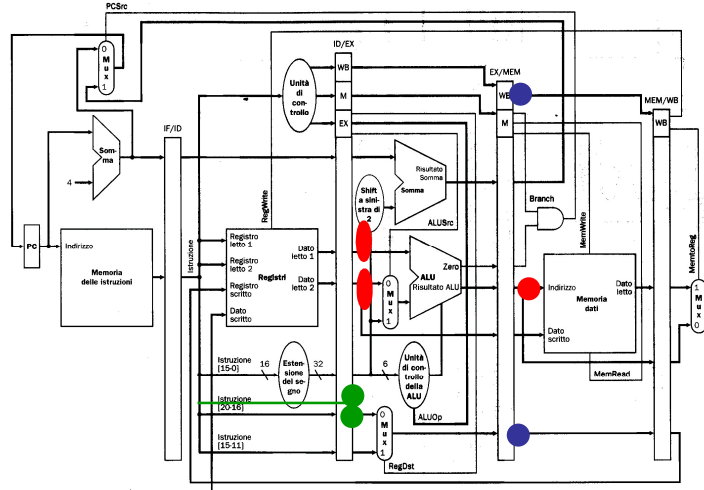
sub $s2, $s1, $s2
add $t2, $s2, $s2

```

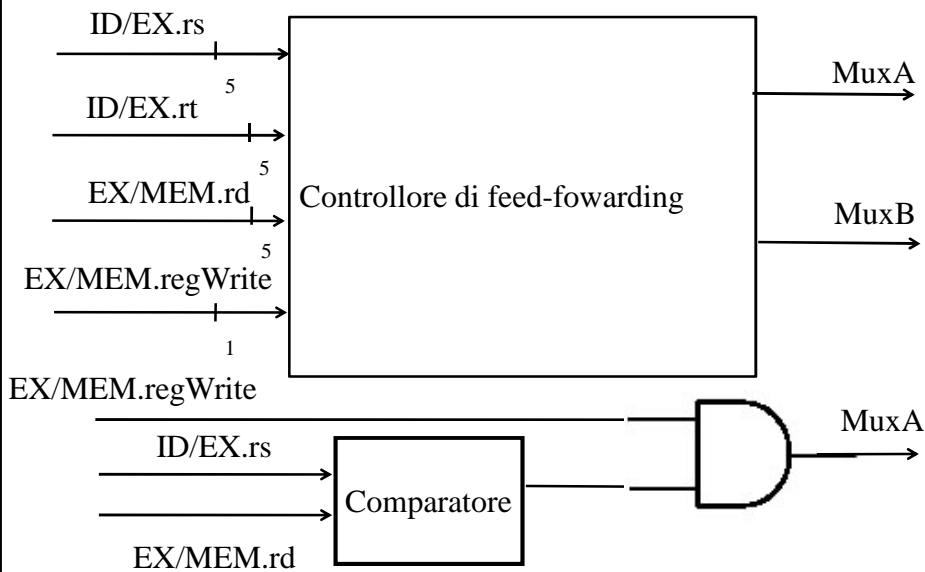
```

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;

```



# 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.

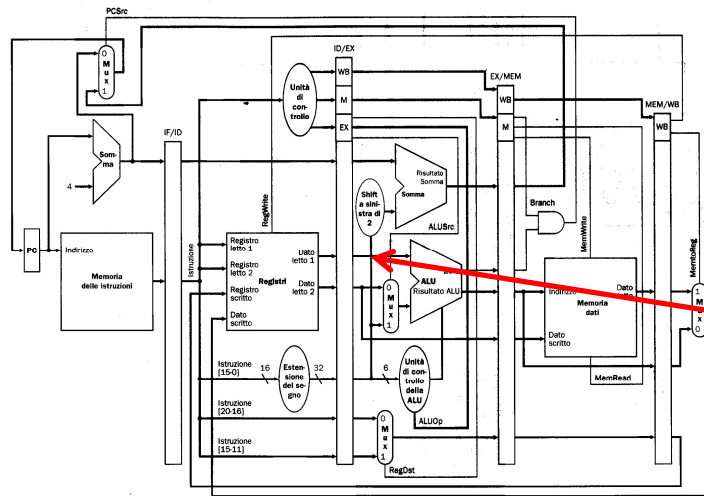
Dobbiamo risolvere la criticità sulla or



# 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.

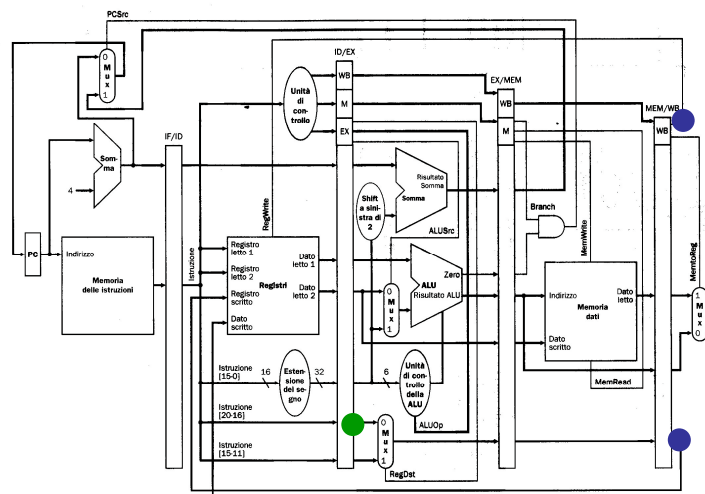


# Identificazione delle criticità – MEM/WB



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



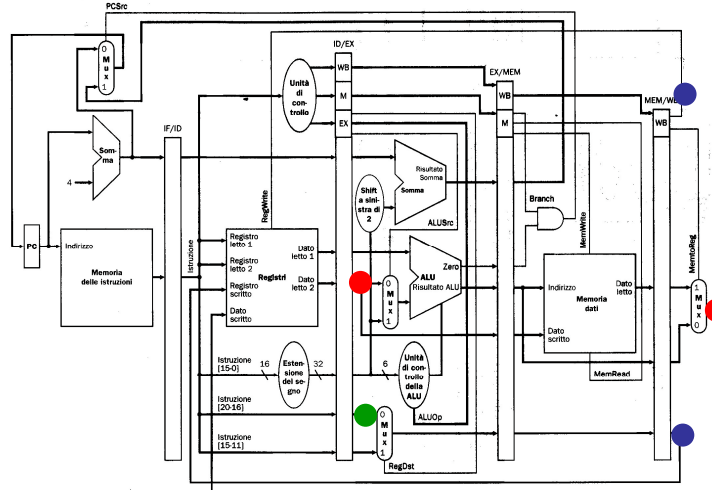


# Soluzione della criticità



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

IF ( (EX/MEM.RegistroRd == EX/MEM.RegistroRs) &&  
(EX/MEM.RegistroWrite) ) Then HazardRt = 1;



A.A. 2013-2014

mi.it

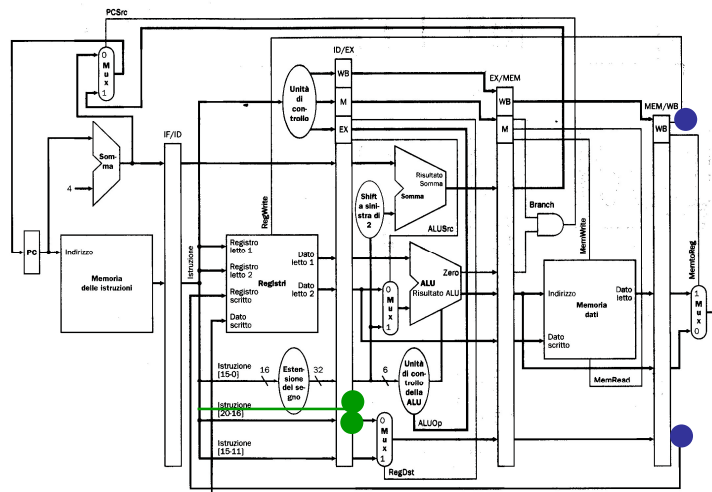


# Identificazione delle criticità – MEM/WB



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. 2013-2014

mi.it



## Soluzione della criticità



```

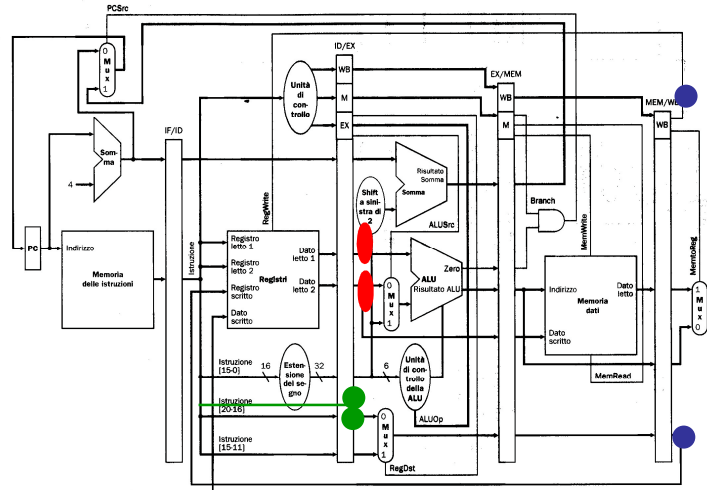
sub $s2, $s1, $s2
add $t2, $s2, $s5
or $t3, $s6, $s2

```

```

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. 2013-2014

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.

A.A. 2013-2014

32/38

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



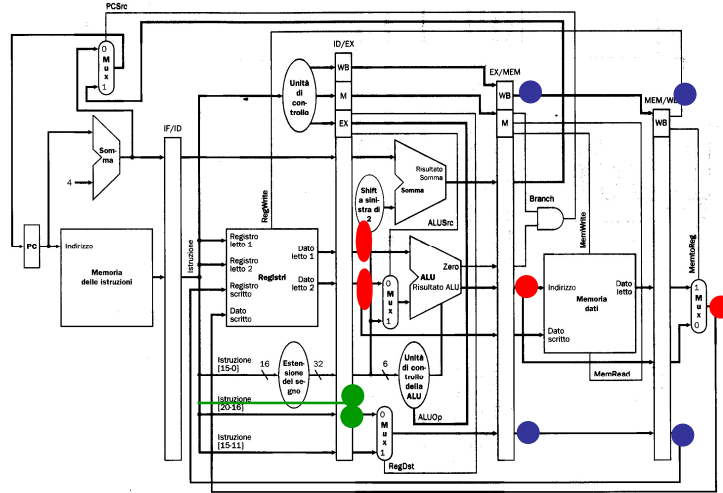


# Un'unica CPU per la soluzione della criticità



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

I multiplexer possono propagare sia da WB che da MEM il dato prodotto dalla ALU.

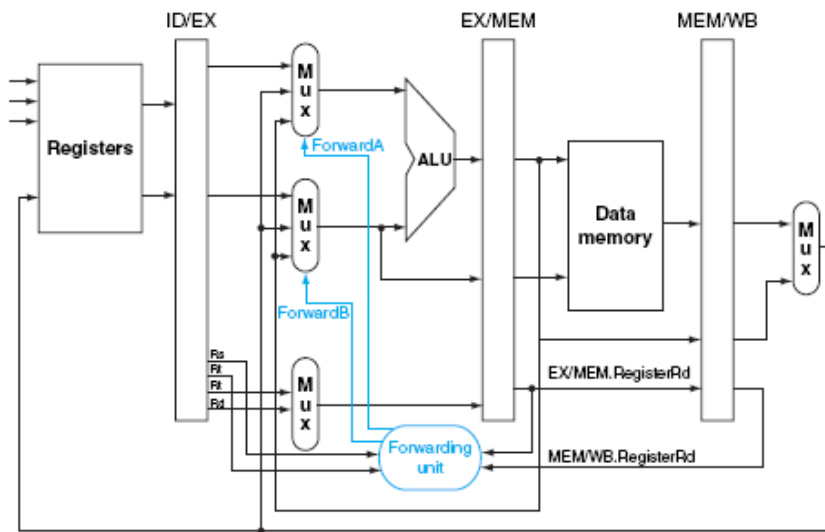


A.A. 2013-2014

mi.it\



## Circuito di propagazione



A.A. 2013-2014

34/38

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



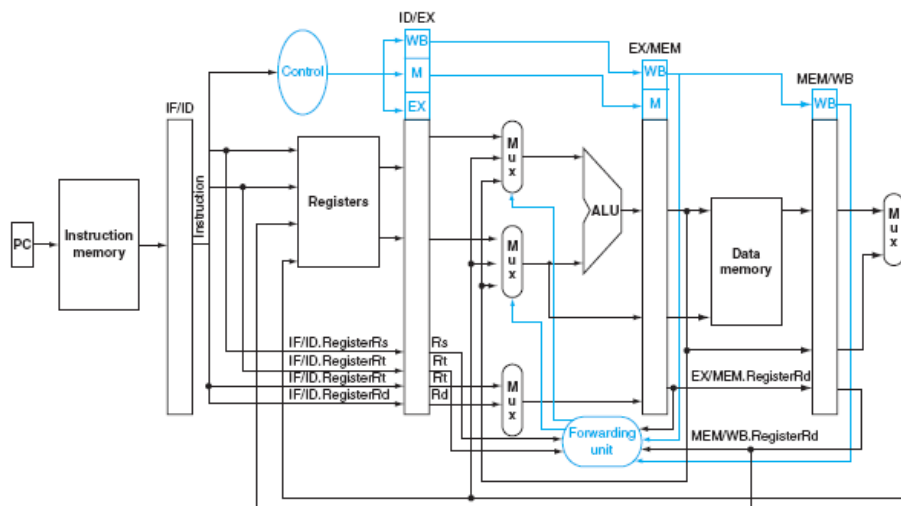
## Controllo Mux ingresso alla ALU



Controllo Multiplexer	Registro Sorgente	Funzione
PropagaA = 00	ID/EX	Il primo operando della ALU proviene dal Register File
PropagaA = 01	EX/MEM	Il primo operando della ALU è propagato dal risultato della ALU per l'istruzione precedente.
PropagaA = 10	MEM/WB	Il primo operando della ALU è propagato dalla memoria o da un'altra istruzione precedente.
PropagaB = 00	ID/EX	Il secondo operando della ALU proviene dal Register File
PropagaB = 01	EX/MEM	Il secondo operando della ALU è propagato dal risultato della ALU per l'istruzione precedente.
PropagaB = 10	MEM/WB	Il secondo operando della ALU è propagato dalla memoria o da un'altra istruzione precedente.



## CPU con unità 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?



## Sommario



Criticità in una pipeline

Dipendenza tra i dati e propagazione

Propagazione a due passi