



Stall on load e Hazard sul controllo

Prof. N. Alberto Borghese
Dipartimento di Informatica
alberto.borghese@unimi.it

Università degli Studi di Milano

Riferimento al Patterson: 4.7, 4.8



Sommario

Identificazione delle criticità che richiedono stallo

Soluzione della criticità mediante stallo

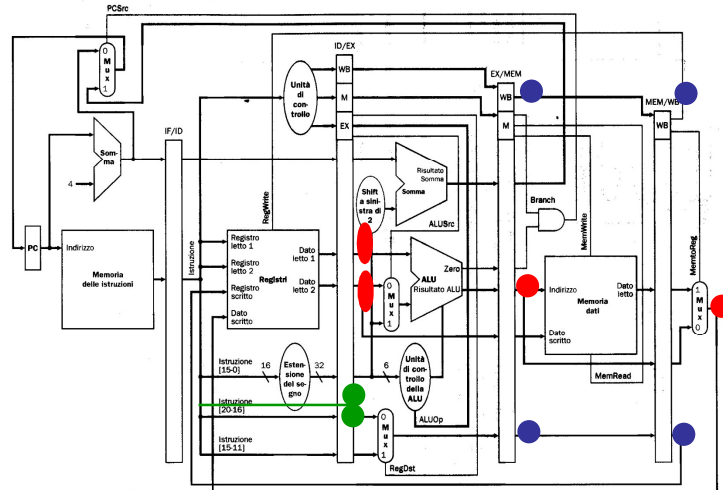
Introduzione agli hazard sul controllo



Un'unica CPU per la soluzione della criticità



sub \$s2, \$s1, \$s3
 add \$t2, \$s2, \$s5
 or \$t3, \$s6, \$s2



A.A. 2012-2013

mi.it\



Data Path e criticità



Nella CPU a singolo ciclo, generazione, produzione e restituzione del dato avvengono nello stesso ciclo.

Nella CPU multi-ciclo, generazione, produzione e restituzione del dato avvengono in cicli diversi. Ad esempio: add \$t0, \$t1, \$t2.

Generazione del dato: fase di decodifica.

Produzione del dato: fase di calcolo.

Restituzione del dato: fase di WB

Questo provoca gli hazard sui dati nelle pipeline.

A.A. 2012-2013

4/40

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



Soluzione mediante propagazione

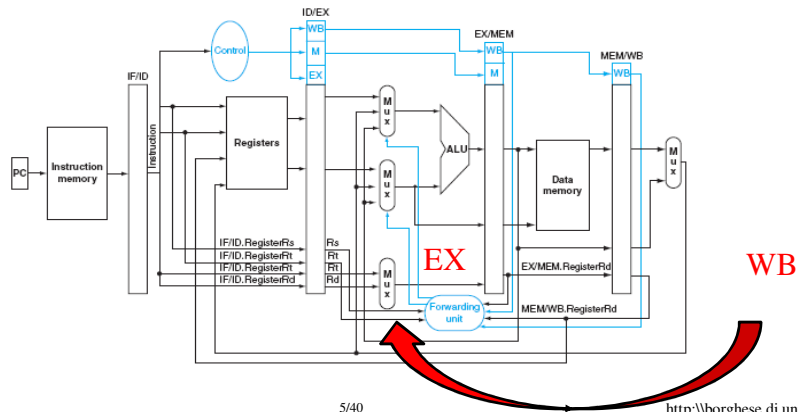


Si attiva quando c'è una criticità sui **dati** originata da una dipendenza.

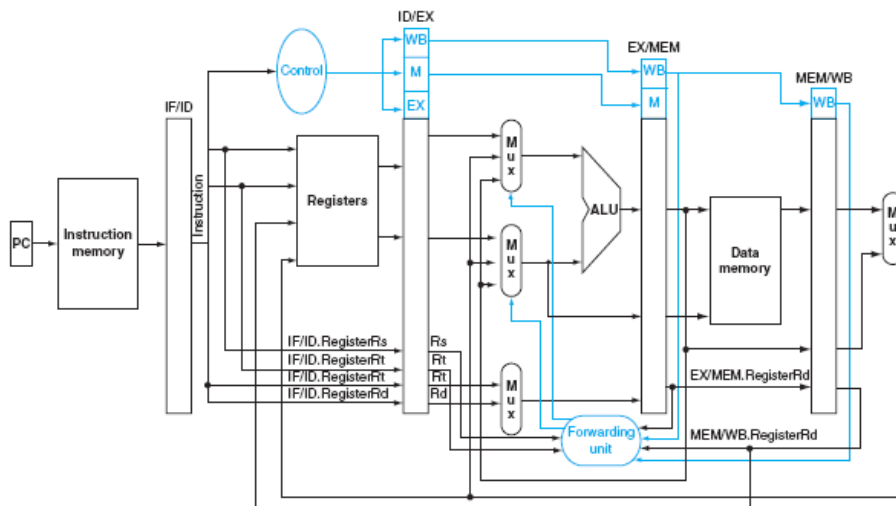
Se il dato critico è già disponibile all'interno della pipe-line si preleva e si propaga (mediante tunnelling) all'indietro all'istruzione che ne ha bisogno.

Il dato è disponibile quando l'istruzione che genera il dato, ha come fase utile la fase di EX.

Si attiva un ammino a ritroso all'interno del data-path dalla fase di WB (o di MEM) alla fase di EX).



CPU con unità di propagazione



Come si deve modificare questa CPU per eseguire correttamente le seguenti 2 istruzioni?
add \$s2, \$s1, \$s0
addi \$s2, \$s1, 32



Hazard sui dati: lw



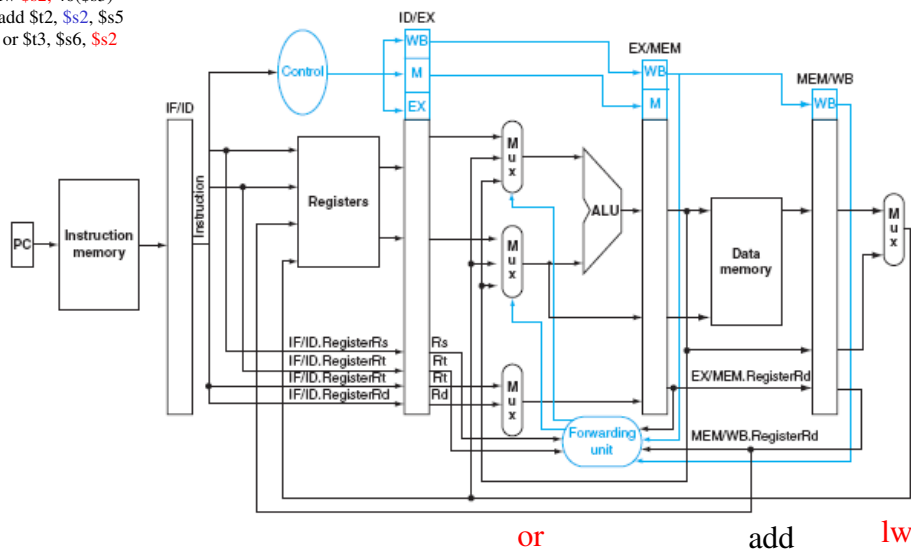
lw \$s2, 40(\$s3)	IF	ID	EX \$s3+40	MEM <\$s3+40>	WB ->\$s2				
add \$t2, \$s2, \$s5		IF	ID	EX \$2 and \$5	MEM	WB s->\$t2			
or \$t3, \$s6, \$s2			IF	ID	EX \$s6 or \$s2	MEM (s->\$t3)			
add \$t4, \$s2, \$s2				IF	ID	EX \$s2 + \$s2	MEM	WB s->\$t4	
sw \$t5, 100(\$s2)					IF	ID	EX \$s2+100	MEM \$t5	WB ->Mem



Criticità a 2 passi è risolta?



lw \$s2, 40(\$s3)
add \$t2, \$s2, \$s5
or \$t3, \$s6, \$s2





Hazard sui dati: lw, rilevamento della criticità

lw \$s2, 40(\$s3)	IF	ID	EX \$s3+ 40	MEM <\$s3+40>	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)		

Il dato corretto per \$s2 è pronto nella lw solamente alla fine della fase **MEM**, ed è perciò utilizzabile solamente a partire dall'inizio della fase di **WB**.

Rilevo la criticità (dato non corretto) su **or** quando **or** inizia la fase di **EX**. In questo caso il dato corretto si trova all'inizio della fase **WB** della **lw** e può essere propagato.

La criticità (dato non corretto) si manifesta su **and** quando **and** inizia la fase di **EX**. In questo caso il dato corretto non è ancora stato prodotto dalla **lw**. **Non posso risolvere questo hazard.**



Soluzione mediante stallo

	t ₀	t ₁	t ₂	t ₃	t ₄	t ₅	t ₆	t ₇
....								
lw \$s2, 40(\$s3)	FF (Mem, ALU)	DECOD (RF)	EXEC (ALU)	MEM (MEM)	WB (RF)			
nop		Buco (FF)	Buco (DEC)	Buco (EXEC)	Buco (MEM)	Buco (WB)		
add \$t2, \$s2, \$s5			FF	DEC	EXEC	MEM		

I buchi (o bubble) inducano degli istanti di clock in cui non può essere eseguita l'istruzione successiva → **La pipeline va messa in stallo.**

Devo bloccare l'esecuzione della **and** e ripeterla un ciclo dopo, quando è possibile utilizzare il valore corretto del registro \$s2

Il cammino in blu invece indica il cammino di propagazione interna dei dati che consente l'esecuzione corretta della **add** dopo 1 ciclo di stallo.

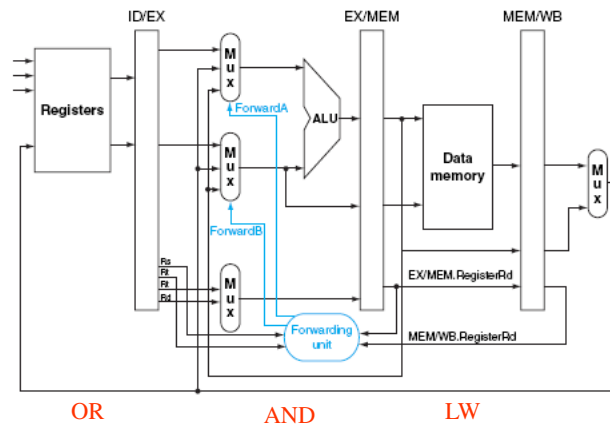


Rilevamento della criticità della lw



IF [(ID/EX.CodOp = 'lw')] \longrightarrow Read in fase di EX
 AND
 {[(IF/ID.RegistroRt) == ID/EX.RegistroRw] OR
 [(IF/ID.RegistroRs) == IF/EX.RegistroRw]} EX - lw \$s2, 40(\$s3)
 THEN "Metti in stallo la pipeline" DEC - add \$t2, \$s2, \$s5

Richiede di portare avanti il codice operativo.
 Prendo il numero di registro destinazione a valle del Mux.

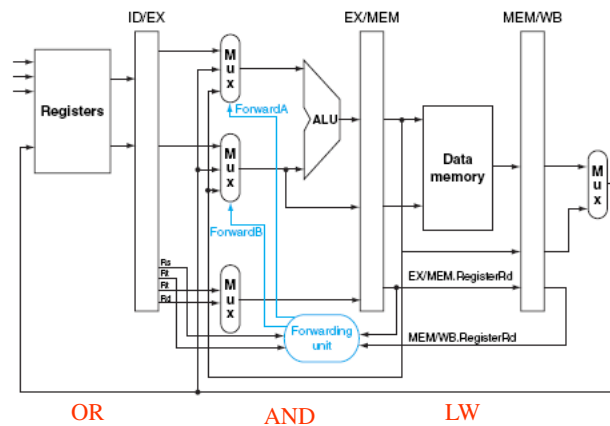


Rilevamento della criticità della lw



IF [(ID/EX.MemRead)] \longrightarrow Read in fase di EX
 AND
 {[(IF/ID.RegistroRt) == ID/EX.RegistroRt] OR
 [(IF/ID.RegistroRs) == IF/EX.RegistroRt]} EX - lw \$s2, 40(\$s3)
 THEN "Metti in stallo la pipeline" DEC - add \$t2, \$s2, \$s5

Cerco di capire il prima possibile se c'è una criticità per la quale occorre mettere in stallo la pipeline.





Sommario



Identificazione delle criticità che richiedono stallo

Soluzione della criticità mediante stallo

Introduzione agli hazard sul controllo

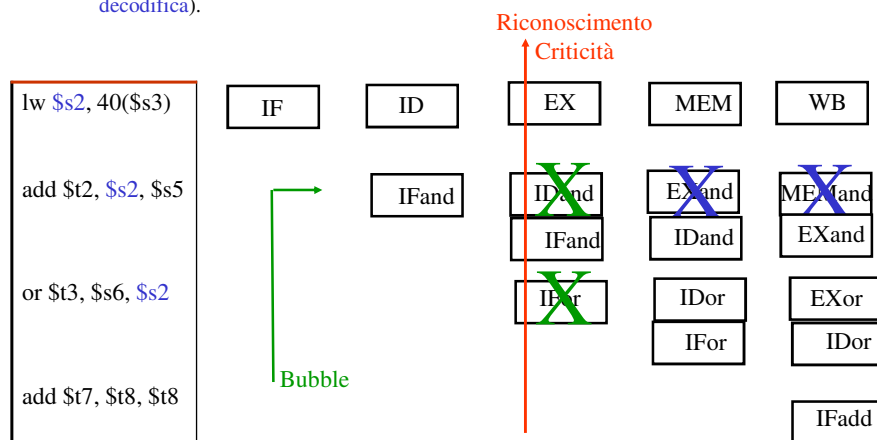


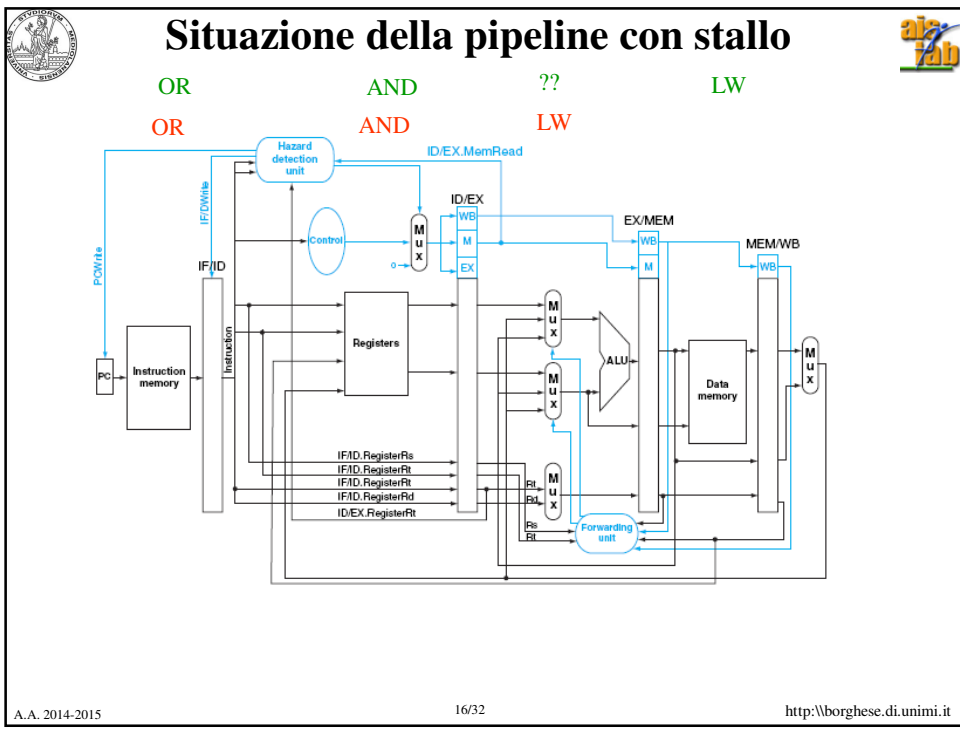
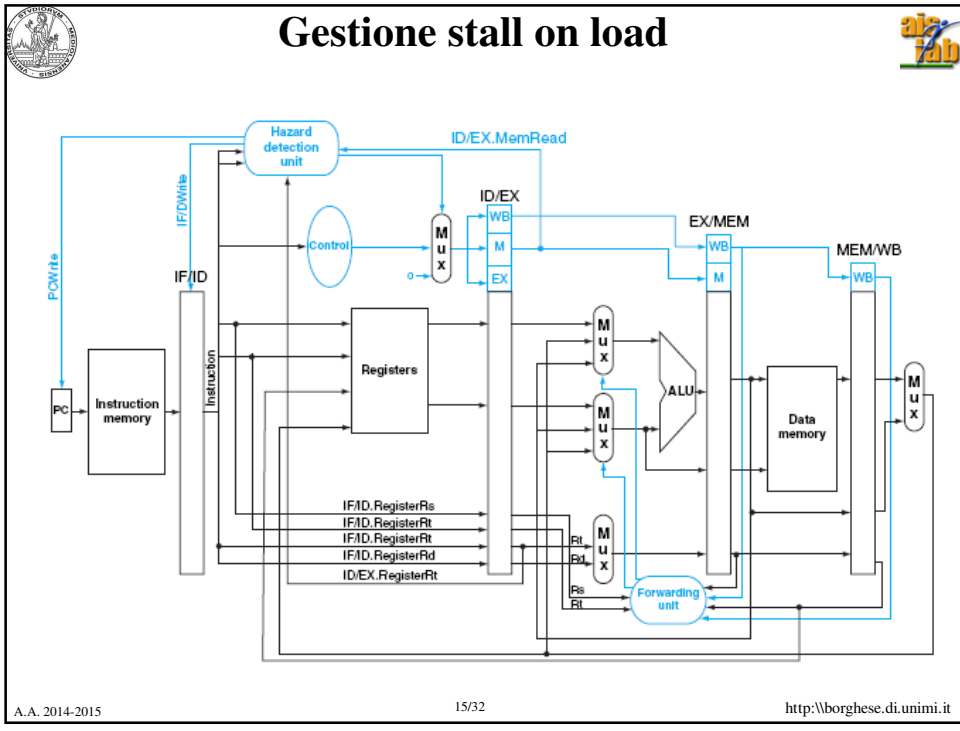
Stallo della pipeline



Azioni volte a non modificare elementi di stato:

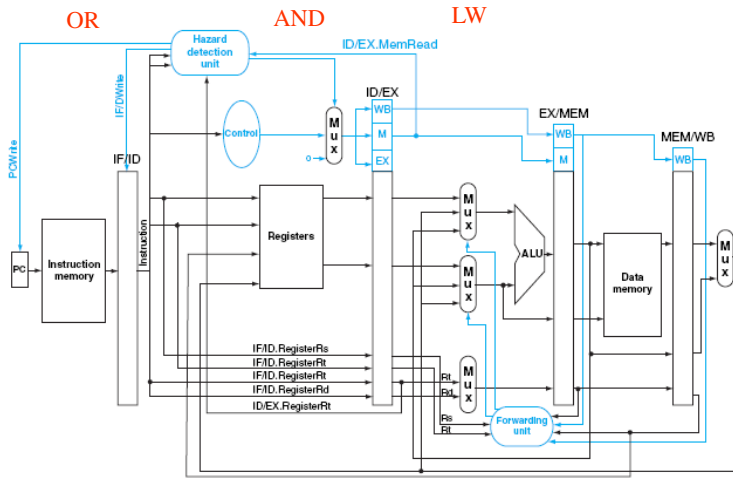
- Annullare i segnali di controllo generati nella fase ID per l'esecuzione dell'istruzione (successiva alla lw).
- Ripetere la lettura e la decodifica delle 2 istruzioni successive (ripetere la fase di fetch e decodifica).







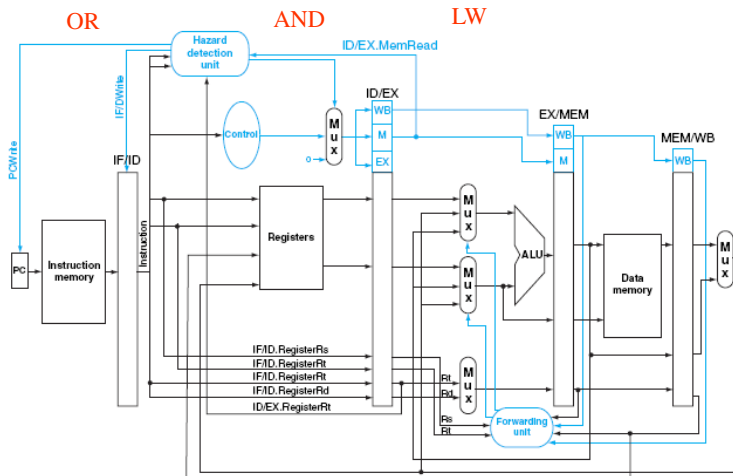
Annullamento dell'istruzione



Annullamento dei segnali di controllo associati. Perché invece non annullo la scrittura dei registri ID/EX, EX/MEM e MEM/WB?



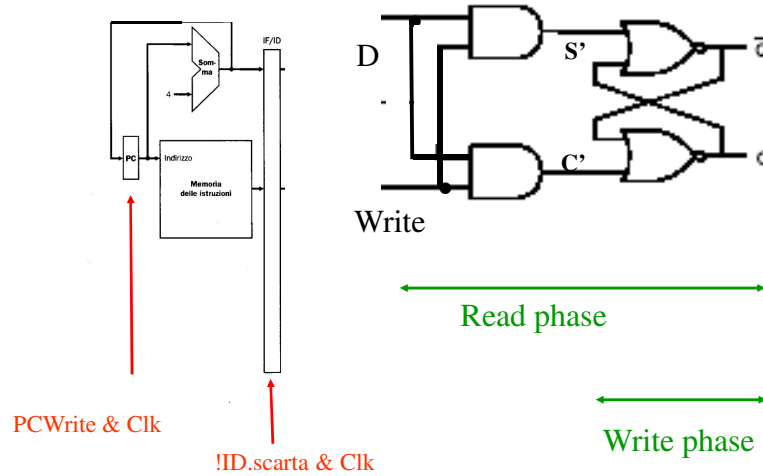
Ripetizione delle fasi ID e IF delle due istruzioni successive



Disabilitazione della scrittura del PC e del registro IF/ID nella fase di Exec della lw.



Disabilitazione della scrittura dei registri



Ipotesi: L'unità di controllo della criticità è in grado di prendere una decisione in tempo utile (prima dell'inizio della fase di Write).



Hazard sui dati della lw



1) Rilevamento della criticità

$$\text{IF} [(\text{ID/EX.MemRead})] \text{ AND } \{[(\text{IF/ID.RegistroRt}) == \text{ID/EX.RegistroRt}] \text{ OR } [(\text{IF/ID.RegistroRs}) == \text{IF/EX.RegistroRt}]\}$$

2) Correzione del problema -> stallo

- 2a) faccio eseguire l'istruzione in ID con segnali di controllo a 0: esecuzione fasulla.
- 2b) inibisco la scrittura dei registri ID e PC.

Inserisco una frattura tra le istruzioni che continuano l'esecuzione (quelle in EX, MEM e WB) e quelle che vengono fermate per un ciclo di clock (quelle in ID e IF)



Hazard nei dati: soluzioni



- Buona scrittura del codice (il programmatore deve conoscere la macchina per scrivere un buon codice!).
- Compilatore efficiente (che riordini il codice).
- Architettura che renda disponibile i dati appena pronti alla fase di esecuzione.
- Accettare uno stallo (non sempre si può evitare).



Sommario



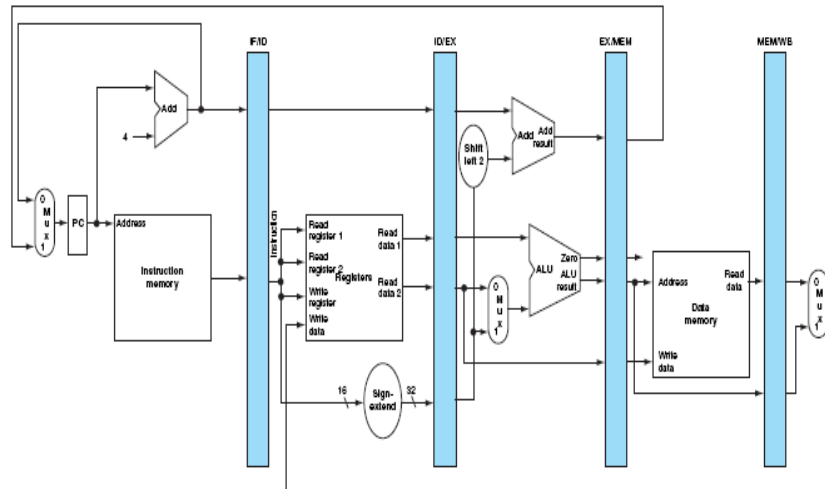
Identificazione delle criticità che richiedono stallo

Soluzione della criticità mediante stallo

Introduzione agli hazard sul controllo



CPU con pipeline



Esempio di Hazard sul controllo



sub \$s2, \$s1, \$s3	IF	ID	EX \$1-\$3	MEM	WB s->\$2			
beq \$t2, \$s6, 24		IF	ID	EX Zero if (\$s2 == \$s5)	MEM	WB		
or \$t7, \$s6, \$s7			IF	ID	EX	MEM	WB	
add \$t4, \$s8, \$s8				IF	ID	EX	MEM	WB
and \$s5, \$s6, \$s7					IF	ID	EX	MEM
add \$t0, \$t1, \$t2						IF	ID	EX

In caso di salto: dovrei avere disponibile all'istante in cui inizia l'esecuzione dell'istruzione or l'indirizzo dell'istruzione add e non eseguire la or, la add e la and.

NB L'indirizzo scritto nel PC corretto deve essere disponibile prima dell'inizio della fase di fetch. Ho 3 istruzioni sbagliate in pipeline.

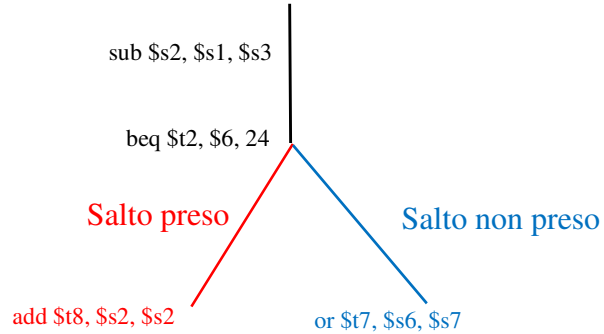
NB Il PC è master/slave per cui occorre che l'indirizzo sia pronto prima dell'inizio della fase di fetch.



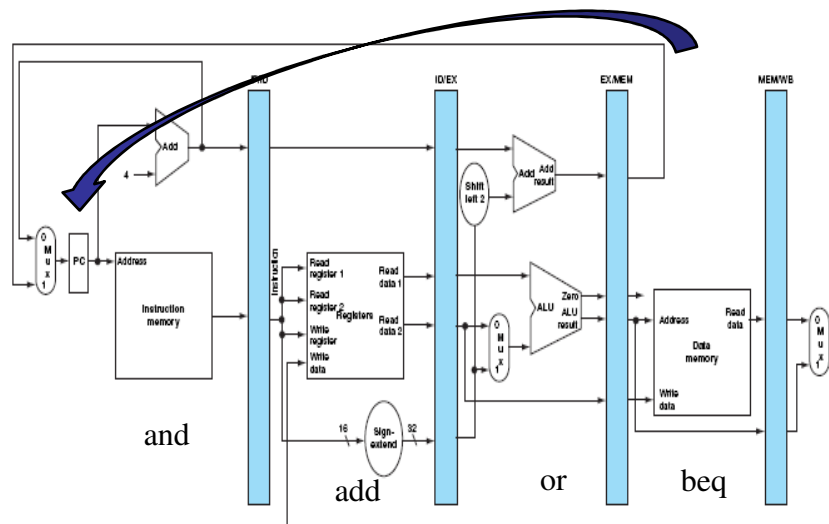
Hazard sul controllo



sub \$s2, \$s1, \$s3
beq \$t2, \$6, 24
or \$t7, \$s6, \$s7
add \$t4, \$s8, \$s8
and \$s5, \$s6, \$s7
add \$t0, \$t1, \$t2
sw \$s3, 24(\$t1)
addi \$t7, \$s6, 10
add \$t8, \$s2, \$s2
and \$s5, \$s6, \$s7
add \$t0, \$t1, \$t2



Criticità sul controllo



Il valore del PC può essere aggiornato nella fase di Memoria
 Le istruzioni in fase di IF, ID, EX potrebbero non dovere essere eseguite.



Soluzioni alla criticità nel controllo



Modifiche strutturali per l'anticipazione dei salti.
&
Riordinamento del codice (delayed branch).



Esempio di riorganizzazione del codice per le istruzioni di branch



<pre>if (a == b) { s2 = s0 + s1; }</pre>	<pre>if (a == b) { s2 = s0 + s1; s3 = s4 + s5; } else { s3 = s4 + s5; }</pre>
<pre>s3 = s4 + s5; salta: s6 = 2;</pre>	<pre>s6 = 2;</pre>



Esempio di riorganizzazione del codice - II



```

if (a == b)
{
    s2 = s0 + s1;
}
else
{
    t2 = t0 + t1;
}
s5 = s4 + s3;
t5 = 2;

```

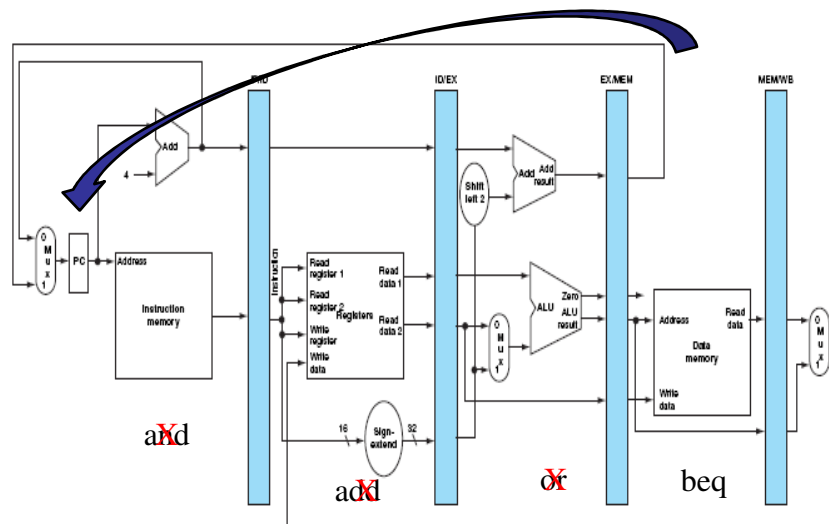
```

if (a == b)
{
    s2 = s0 + s1;
    s5 = s4 + s3;
}
else
{
    t2 = t0 + t1;
    s5 = s4 + s3;
}
t5 = 2;

```



Criticità sul controllo



beq è un salto ritardato, rispetto al codice, il salto viene generato 3 cicli dopo.
 Questi 3 cicli di ritardo vengono chiamati **delay slot**.



Esempio di delayed branch (1 solo delay slot)



Originale

sub \$t5, \$t8, \$s8
add \$s4, \$t0, \$t1
beq \$s5, \$s6, salto
and \$s0, \$s0, \$s1

salto:

add \$t5, \$t4, \$t3
add \$t6, \$t7, \$t7

From target

sub \$t5, \$t8, \$s8
add \$s4, \$t0, \$t1
beq \$s5, \$s6, salto
add \$t5, \$t4, \$t3
and \$s0, \$s0, \$s1

salto:

add \$t6, \$t7, \$t7

From before

add \$s4, \$t0, \$t1
beq \$s5, \$s6, salto
sub \$t5, \$t8, \$s8
and \$s0, \$s0, \$s1

salto:

add \$t5, \$t4, \$t3
add \$t6, \$t7, \$t7

L'istruzione *add \$t5, \$t4, \$t3* o *sub \$t5, \$t8, \$s8* viene comunque eseguita, il salto (se richiesto) avviene all'istante successivo.

Riempio quindi con queste istruzioni, istruzioni da eseguire comunque, lo slot dopo la branch, denominato branch delay slot.

Controllo di non inserire Hazard sui dati



Sommario



Identificazione delle criticità che richiedono stallo

Soluzione della criticità mediante stallo

Introduzione agli hazard sul controllo