



# Stallo, hazard sul controllo

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

Università degli Studi di Milano

Riferimento al Patterson: 6.5 e 6.6



## Sommario

Modifiche alla CPU per la gestione di criticità sui dati, istruzioni di lw.

Hazard sul controllo

### Hazard sui dati: lw

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

A.A. 2006-2007 3/33 http://homes.dsi.unimi.it/~borgnese

### Hazard sui dati: lw, rilevamento della criticità

|                      |    |    |               |                     |                    |               |                 |  |  |
|----------------------|----|----|---------------|---------------------|--------------------|---------------|-----------------|--|--|
| lw \$s2, 40(\$s3)    | IF | ID | EX<br>\$s3+40 | MEM<br><\$s3+40>    | WB<br>s->\$s2      |               |                 |  |  |
| and \$t2, \$s2, \$s5 |    | IF | ID            | EX<br>\$s2 and \$s5 | MEM                | WB<br>s->\$t2 |                 |  |  |
| or \$t3, \$s6, \$s2  |    |    | IF            | ID                  | EX<br>\$s6 or \$s2 | MEM           | WB<br>(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 **and** quando and inizia la fase di EX. In questo caso il dato corretto non è ancora stato prodotto dalla lw.

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.

A.A. 2006-2007 4/33 http://homes.dsi.unimi.it/~borgnese



# Soluzione mediante stallo



|                      | t <sub>0</sub>      | t <sub>1</sub> | t <sub>2</sub> | t <sub>3</sub> | t <sub>4</sub> | t <sub>5</sub> | t <sub>6</sub> | t <sub>7</sub> |
|----------------------|---------------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| ....                 |                     |                |                |                |                |                |                |                |
| lw \$s2, 40(\$s3)    | FF<br>(Mem,<br>ALU) | DECOD<br>(RF)  | EXEC<br>(ALU)  | MEM<br>(MEM)   | WB<br>(RF)     |                |                |                |
| nop                  |                     | Buco<br>(FF)   | Buco<br>(DEC)  | Buco<br>(EXEC) | Buco<br>(MEM)  | Buco<br>(WB)   |                |                |
| nop                  |                     |                | Buco           | Buco           | Buco           | Buco           | Buco           |                |
| and \$t2, \$s2, \$s5 |                     |                |                | FF             | DEC            | EXEC           | MEM            |                |

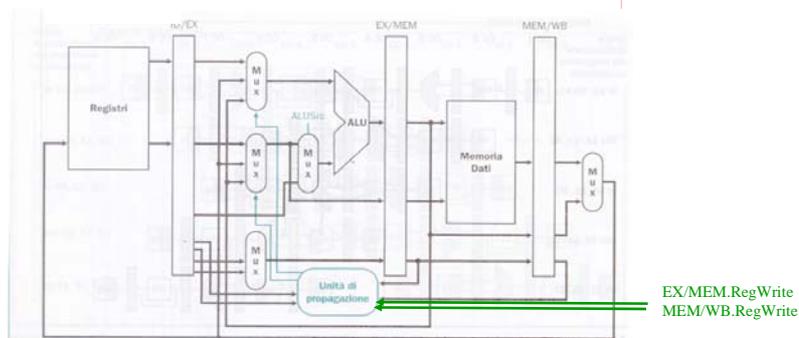
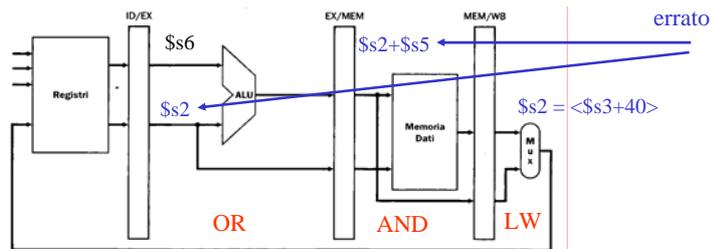
I buchi (o bubble) inducano degli istanti di clock in cui non può essere eseguita l'istruzione successiva → La pipeline va in stallo.



# Hazard nei dati: lw, forwarding



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





## Hazard nei dati: lw, unità di propagazione



Dato preso dalla fase WB:

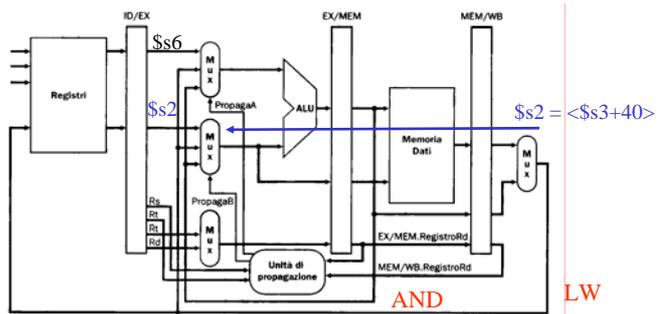
IF (ID/EX.RegistroRs == MEM/WB.RegistroRd) AND (MEM/WB.RegWrite)

ID/EX.RegistroRs = MEM/WB.RegistroRd

IF (ID/EX.RegistroRt == MEM/WB.RegistroRd) AND (MEM/WB.RegWrite)

ID/EX.RegistroRt = MEM/WB.RegistroRd

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



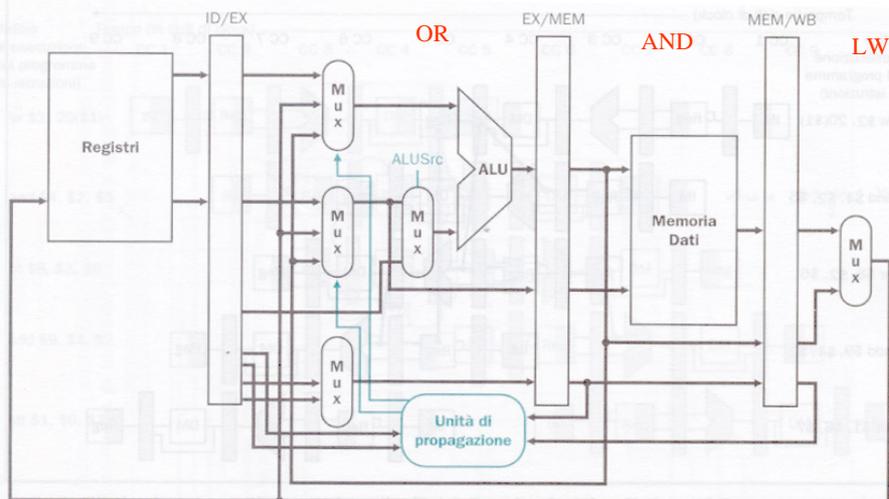
A.A. 2006-2007

7/33 OR

<http://homes.dsi.unimi.it/~borgnese>



## Il forwarding non è sufficiente



Nessuna modifica ma risolve solamente uno dei due problemi della lw.

A.A. 2006-2007

8/33

<http://homes.dsi.unimi.it/~borgnese>



## Hazard sui dati: lw, stallo



|                      |    |    |                     |                  |                     |     |                 |  |  |
|----------------------|----|----|---------------------|------------------|---------------------|-----|-----------------|--|--|
| lw \$s2, 40(\$s3)    | IF | ID | EX<br>\$s3+40       | MEM<br><\$s3+40> | WB<br>s->\$s2       |     |                 |  |  |
| and \$t2, \$s2, \$s5 | IF | ID | EX<br>\$s2 and \$s5 | MEM              | WB<br>s->\$t2       |     |                 |  |  |
| and \$t2, \$s2, \$s5 |    |    | IF                  | ID               | EX<br>\$s2 and \$s5 | MEM | WB<br>(s->\$t2) |  |  |

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 **EX**.

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



**Stallo della pipeline**



## Rilevamento della criticità sulla lw



|                      |    |    |                     |                |                    |     |                 |  |  |
|----------------------|----|----|---------------------|----------------|--------------------|-----|-----------------|--|--|
| lw \$s2, 40(\$s3)    | IF | ID | EX                  | MEM<br>\$s3+40 | WB<br>s->\$s2      |     |                 |  |  |
| and \$t2, \$s2, \$s5 | IF | ID | EX<br>\$s2 and \$s5 | MEM            | WB<br>s->\$t2      |     |                 |  |  |
| or \$t3, \$s6, \$s2  |    |    | IF                  | ID             | EX<br>\$s6 or \$s2 | MEM | WB<br>(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**.

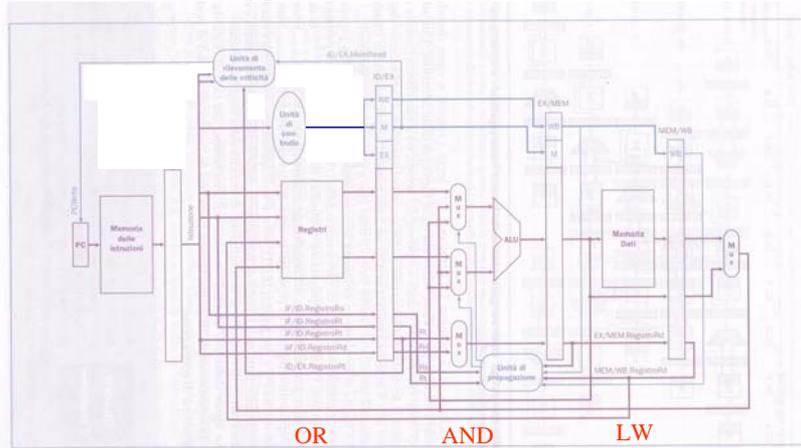
Rilevo questa criticità il prima possibile in modo da mettere in stallo prima possibile la pipeline: nello stadio di decodifica dell'istruzione **AND**.



## Rilevamento della criticità della lw



IF [(ID/EX (MemRead)) → Read in fase di EX  
 AND  
 {[ (IF/ID.RegistroRt) == ID/EX.RegistroRt] OR  
 [(IF/ID.RegistroRs) == IF/EX.RegistroRt]}  
 THEN  
 “Metti in stallo la pipeline”



A.A. 2006-2007

borghese

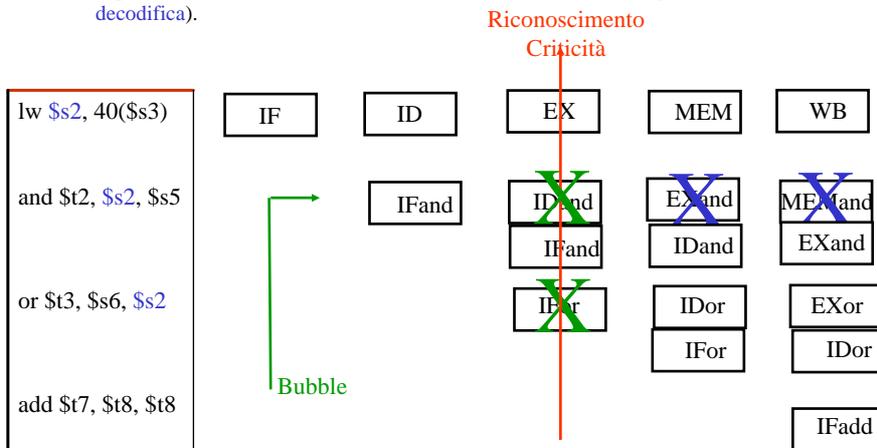


## Stallo della pipeline



### Azioni:

- 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).



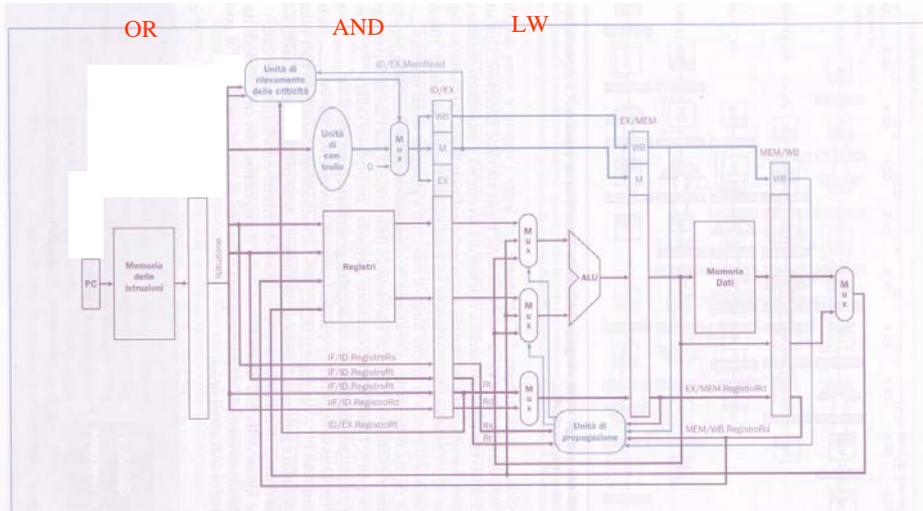
A.A. 2006-2007

12/33

http://homes.dsi.unimi.it/~borghese



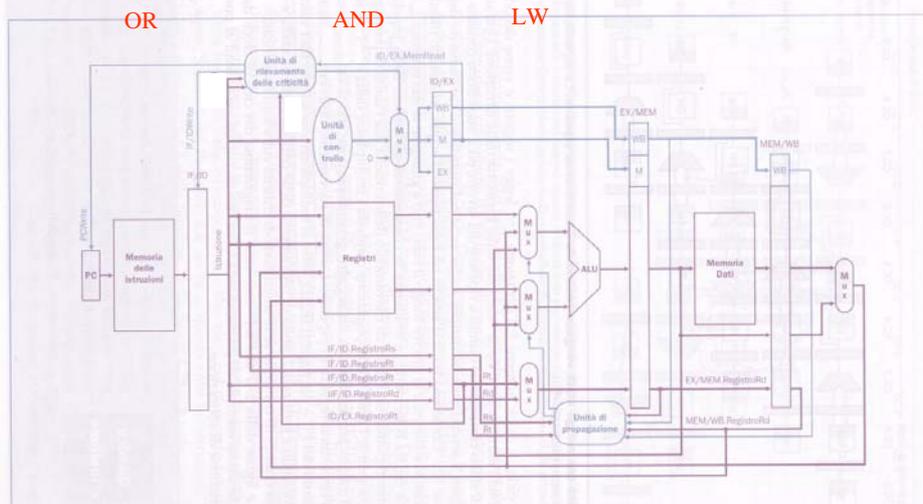
## Annulamento dell'istruzione in fase ID (and)



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



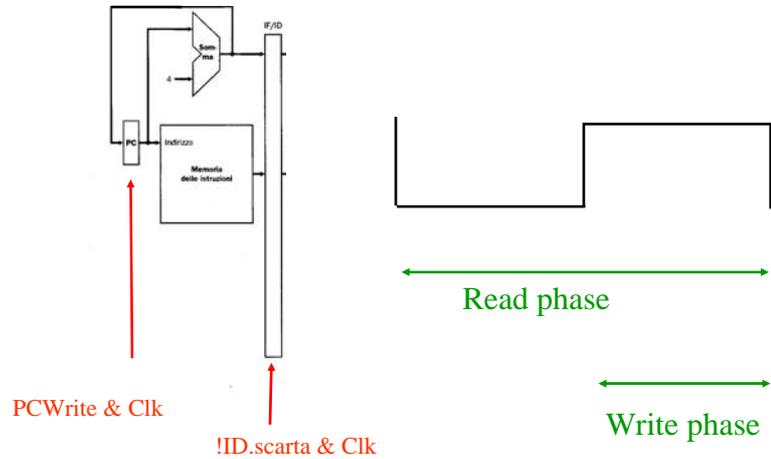
## Ripetizione delle fasi ID e IF delle due istruzioni successive



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



## Disabilitazione della scrittura dei registri



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



## Rilevamento della criticità sulla lw



|                      |    |    |    |     |     |     |    |  |  |
|----------------------|----|----|----|-----|-----|-----|----|--|--|
| lw \$s2, 40(\$s3)    | IF | ID | EX | MEM | WB  |     |    |  |  |
| and \$t2, \$s2, \$s5 |    | IF | ID | EX  | MEM | WB  |    |  |  |
| or \$t3, \$s6, \$s2  |    |    | IF | ID  | EX  | MEM | WB |  |  |

Annotations: A blue arrow points from the ID stage of the 'lw' instruction to the ID stage of the 'and' instruction. A red dotted arrow points from the MEM stage of the 'lw' instruction to the EX stage of the 'and' instruction. A red arrow points from the WB stage of the 'and' instruction to the WB stage of the 'or' instruction.

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 questa criticità il prima possibile in modo da mettere in stallo prima possibile la pipeline: nello stadio di decodifica dell'istruzione AND.

Potrei rilevare la criticità anche nello stadio EX dell'istruzione AND. Quale svantaggio avrei?



## Hazard sui dati della lw



### 1) Rilevamento della criticità

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



## Sommario



Modifiche alla CPU per la gestione di criticità sui dati, istruzioni di lw.

**Hazard sul controllo (branch hazard)**



## Salto incondizionato



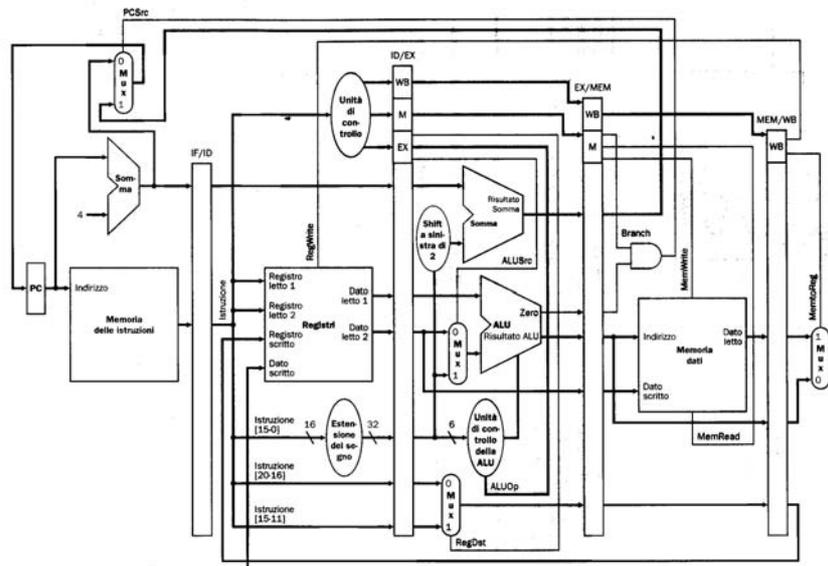
Utilizzato all'interno dei cicli for / while. Non pone problemi. Si risolve con la riorganizzazione del codice

|       |        |                      |        |                      |
|-------|--------|----------------------|--------|----------------------|
|       | 400:   | add\$s0, \$s1, \$s2  | 400:   | add\$s0, \$s1, \$s2  |
|       | 404:   | j 80000              | 404:   | j 80004              |
| Label | 408:   | and \$s1, \$s2, \$s3 | 408:   | or \$t0, \$t1, \$t2  |
|       |        |                      | 412:   | and \$s2, \$s2, \$s3 |
|       | 80000: | or \$t0, \$t1, \$t2  | 80004: | sub \$t3, \$t4, \$t5 |
|       | 80004: | sub \$t3, \$t4, \$t5 |        |                      |

j "lavora" nella fase di decodifica. Viene eseguita un'istruzione prima del salto: delayed jump.



## CPU con pipeline





## Esempio di Hazard sul controllo



|                      |    |    |               |                                 |              |     |     |     |
|----------------------|----|----|---------------|---------------------------------|--------------|-----|-----|-----|
| sub \$s2, \$s1, \$s3 | IF | ID | EX<br>\$1-\$3 | MEM                             | WB<br>s->\$2 |     |     |     |
| beq \$t2, \$s6, 24   |    | IF | ID            | EX<br>Zero if<br>(\$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.



## Soluzioni alla criticità nel controllo



Modifiche strutturali per l'anticipazione dei salti.

Riordinamento del codice (delayed branch).



## Modifica della CPU



Obbiettivi:

- Identificare l'hazard durante la fase ID di esecuzione della branch.
- Scartare una sola istruzione.

|                 |                                |    |    |                     |                                 |               |                |               |     |
|-----------------|--------------------------------|----|----|---------------------|---------------------------------|---------------|----------------|---------------|-----|
| 800:            | sub \$s2, \$s1, \$s3           | IF | ID | EX<br>\$s1-<br>\$s3 | MEM                             | WB<br>s->\$2  |                |               |     |
| 804:            | beq \$t2, \$s6, tag            |    | IF | ID                  | EX<br>Zero if<br>(\$s2 == \$s5) | MEM           | WB             |               |     |
| <del>808:</del> | <del>or \$t7, \$s6, \$s7</del> |    |    | <del>IF</del>       | <del>ID</del>                   | <del>EX</del> | <del>MEM</del> | <del>WB</del> |     |
| .....           |                                |    |    |                     |                                 |               |                |               |     |
| tag:            | add \$t4, \$s8, \$s8           |    |    |                     | IF                              | ID            | EX             | MEM           | WB  |
| tag<br>+4:      | and \$s5, \$s6, \$s7           |    |    |                     |                                 | IF            | ID             | EX            | MEM |
| Tag<br>+8       | add \$t0, \$t1, \$t2           |    |    |                     |                                 |               | IF             | ID            | EX  |

A.A. 2006-2007

23/33

<http://homes.dsi.unimi.it/~borgnese>



## Anticipazione del salto



|       |                      |    |    |                     |                                 |              |     |     |     |
|-------|----------------------|----|----|---------------------|---------------------------------|--------------|-----|-----|-----|
| 800:  | sub \$s2, \$s1, \$s3 | IF | ID | EX<br>\$s1-<br>\$s3 | MEM                             | WB<br>s->\$2 |     |     |     |
| 804:  | beq \$t2, \$s6, Tag  |    | IF | ID                  | EX<br>Zero if<br>(\$s2 == \$s5) | MEM          | WB  |     |     |
| 808:  | or \$t7, \$s6, \$s7  |    |    | IF                  | ID                              | EX           | MEM | WB  |     |
| 812:  | add \$t4, \$s8, \$s8 |    |    |                     | IF                              | ID           | EX  | MEM | WB  |
| 816:  | and \$s5, \$s6, \$s7 |    |    |                     |                                 | IF           | ID  | EX  | MEM |
| ..... |                      |    |    |                     |                                 |              |     |     |     |
| Tag   | add \$t0, \$t1, \$t2 |    |    |                     |                                 |              | IF  | ID  | EX  |

**In caso di salto:** dovrei avere disponibile all'istante in cui inizia l'esecuzione dell'istruzione o 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.

L'indirizzo è già pronto al termine della fase di EX, posso quindi risparmiare un ciclo di clock.  
Ho 2 istruzioni da eliminare.

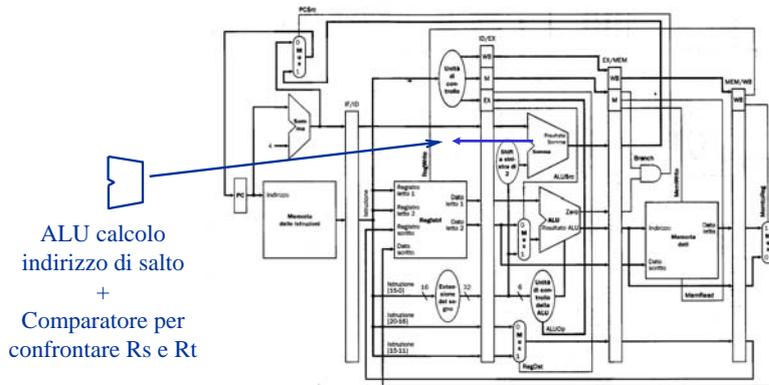
A.A. 2006-2007

24/33

<http://homes.dsi.unimi.it/~borgnese>



## Come identificare l'Hazard nella fase ID



ALU calcolo  
indirizzo di salto  
+  
Comparatore per  
confrontare Rs e Rt

**Anticipazione della valutazione della branch:** Modifica della CPU nella gestione dei salti: anticipazione del calcolo dell'indirizzo di salto.

- HW aggiuntivo: un comparatore all'uscita del Register File.
- Anticipazione del sommatore .



## Soluzione dell'Hazard sul controllo



Stallo della pipeline.

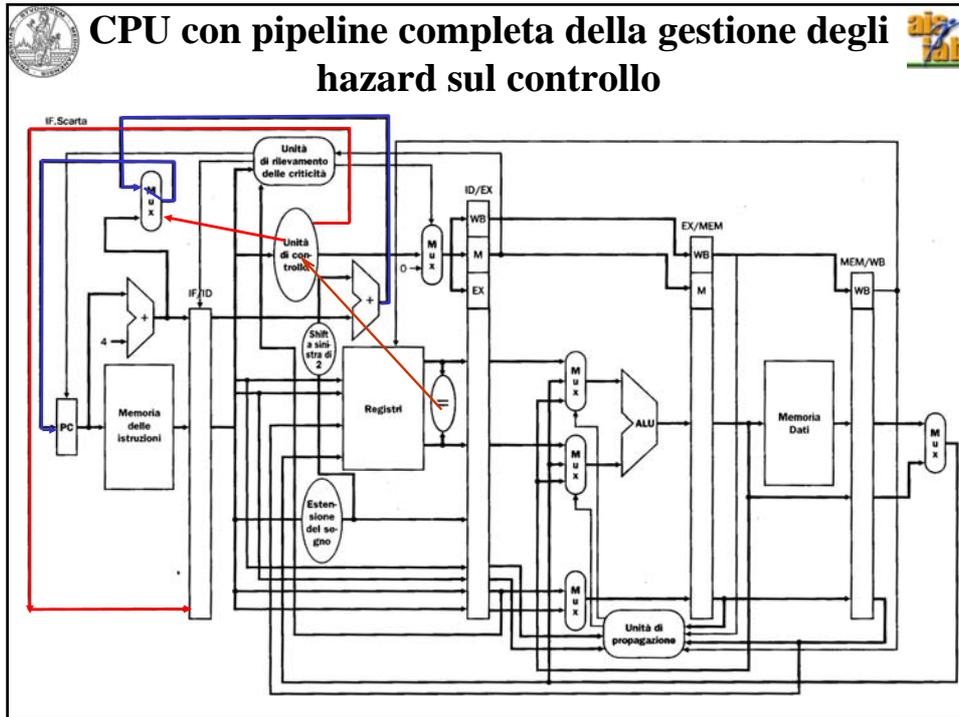
Dalla fase ID alla WB la beq non fa nulla.

Nella fase di IF è l'istruzione successiva che viene trasferita nell'IR (IF/ID), mentre in caso di salto dovrebbe essere trasferita l'istruzione all'indirizzo di salto.

Quindi:



Occorre annullare l'istruzione nel registro IF/ID.



## Come scartare un'istruzione

Si carica nel registro IF/ID un'istruzione nulla.

| Nome campo                         | op     | rs    | rt    | rd    | shamt        | funct  |
|------------------------------------|--------|-------|-------|-------|--------------|--------|
| Dimensione                         | 6-bit  | 5-bit | 5-bit | 5-bit | 5-bit        | 6-bit  |
| <code>sll \$s1, \$s2, 7</code>     | 000000 | X     | 10010 | 10001 | 00111<br>(7) | 000000 |
| $\$s1 = \$s2 = \$zero, shamt = 0$  |        |       |       |       |              |        |
| Nome campo                         | op     | rs    | rt    | rd    | shamt        | funct  |
| Dimensione                         | 6-bit  | 5-bit | 5-bit | 5-bit | 5-bit        | 6-bit  |
| <code>sll \$zero, \$zero, 0</code> | 000000 | 00000 | 00000 | 00000 | 00000<br>(0) | 000000 |

A.A. 2006-2007 28/33 http://homes.dsi.unimi.it/~borgnese



## Riordinamento del codice



Decisione ritardata (ci si affida al compilatore).

Aggiunta di un “branch delay slot”

- l’istruzione successiva ad un salto condizionato viene sempre eseguita
- contiamo sul compilatore/assemblatore per mettere dopo l’istruzione di salto una istruzione che andrebbe comunque eseguita indipendentemente dal salto (ad esempio posticipo un’istruzione precedente la branch).



## Esempio di delayed branch



|        | Originale                    | From target                  | From before                  |
|--------|------------------------------|------------------------------|------------------------------|
|        | <i>sub \$t5, \$t8, \$s8</i>  | <i>sub \$t5, \$t8, \$s8</i>  | <i>add \$s4, \$t0, \$t1</i>  |
|        | <i>add \$s4, \$t0, \$t1</i>  | <i>add \$s4, \$t0, \$t1</i>  | <i>beq \$s5, \$s6, salto</i> |
|        | <i>beq \$s5, \$s6, salto</i> | <i>beq \$s5, \$s6, salto</i> | <i>sub \$t5, \$t8, \$s8</i>  |
|        | <i>add \$s0, \$s0, \$s1</i>  | <i>add \$t5, \$t4, \$t3</i>  | <i>add \$s0, \$s0, \$s1</i>  |
| salto: |                              | <i>add \$s0, \$s0, \$s1</i>  | salto:                       |
|        | <i>add \$t5, \$t4, \$t3</i>  | salto:                       | <i>add \$t5, \$t4, \$t3</i>  |
|        | <i>add \$t6, \$t7, \$t7</i>  | <i>add \$t6, \$t7, \$t7</i>  | <i>add \$t6, \$t7, \$t7</i>  |

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

Controllo di non inserire Hazard sui dati

L’istruzione target può essere posizionata prima o dopo a seconda che la *beq* salti prima o dopo.



## Esempio di riorganizzazione del codice



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



## Esempio di riorganizzazione del codice - II



|  |  |
|--|--|
| <pre>if (a == b) {     s2 = s0 + s1; } else {     t2 = t0 + t1; }  s5 = s4 + s3; t5 = 2;</pre> | <pre>if (a == b) {     s2 = s0 + s1;     s5 = s4 + s3; } else {     t2 = t0 + t1;     s5 = s4 + s3; }  t5 = 2;</pre> |
|--|--|



## Sommario



Modifiche alla CPU per la gestione di criticità sui dati, istruzioni di lw.

Hazard sul controllo