SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Architettura degli Elaboratori
Gestione dei data-hazard nella pipeline

Prof. Andrea Sterbini – sterbini@di.uniroma1.it

Argomenti

- Soluzione dell'esercizio
- Come gestire gli hazard
  - Scoprire che un hazard è presente
  - Realizzare la «scorciatoia» nel datapath per recapitare il dato giusto
  - Realizzare lo stallo quando il dato non è ancora pronto

Argomenti

- Soluzione dell'esercizio
- Come gestire gli hazard
  - Scoprire che un hazard è presente
  - Realizzare la «scorciatoia» nel datapath per recapitare il dato giusto
  - Realizzare lo stallo quando il dato non è ancora pronto

- **Data-hazard:** una istruzione dà il valore ad una delle due istruzioni successive (prima di WB)
- ovvero: il registro destinazione della istruzione precedente è uguale ad uno dei due registri argomenti delle due istruzioni successive
- ed inoltre: la istruzione precedente ha RegWrite = true

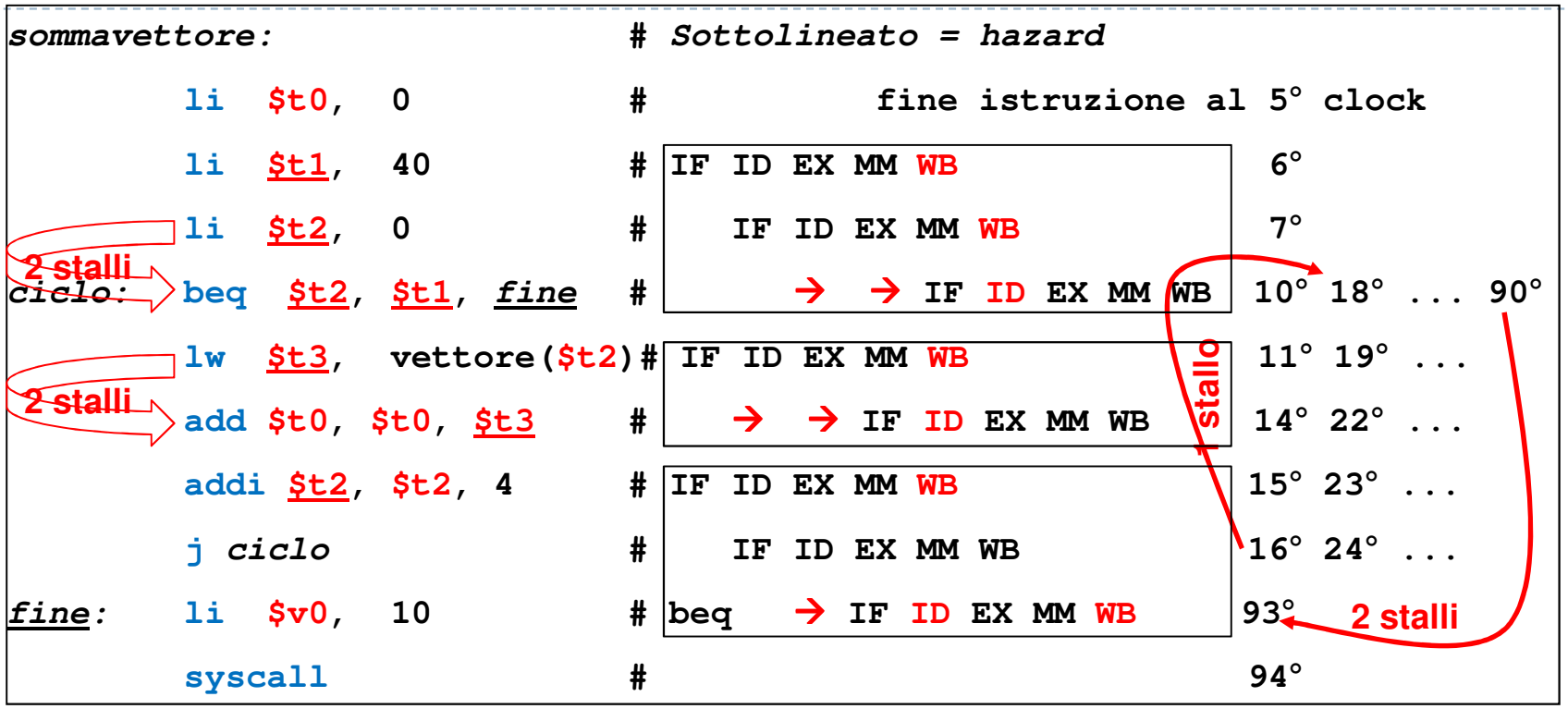
Argomenti

- Soluzione dell'esercizio
- Come gestire gli hazard
  - Scoprire che un hazard è presente
  - Realizzare la «scorciatoia» nel datapath per recapitare il dato giusto
  - Realizzare lo stallo quando il dato non è ancora pronto

- **Data-hazard:** una istruzione dà il valore ad una delle due istruzioni successive (prima di WB)
- ovvero: il registro destinazione della istruzione precedente è uguale ad uno dei due registri argomenti delle due istruzioni successive
- ed inoltre: la istruzione precedente ha RegWrite = true

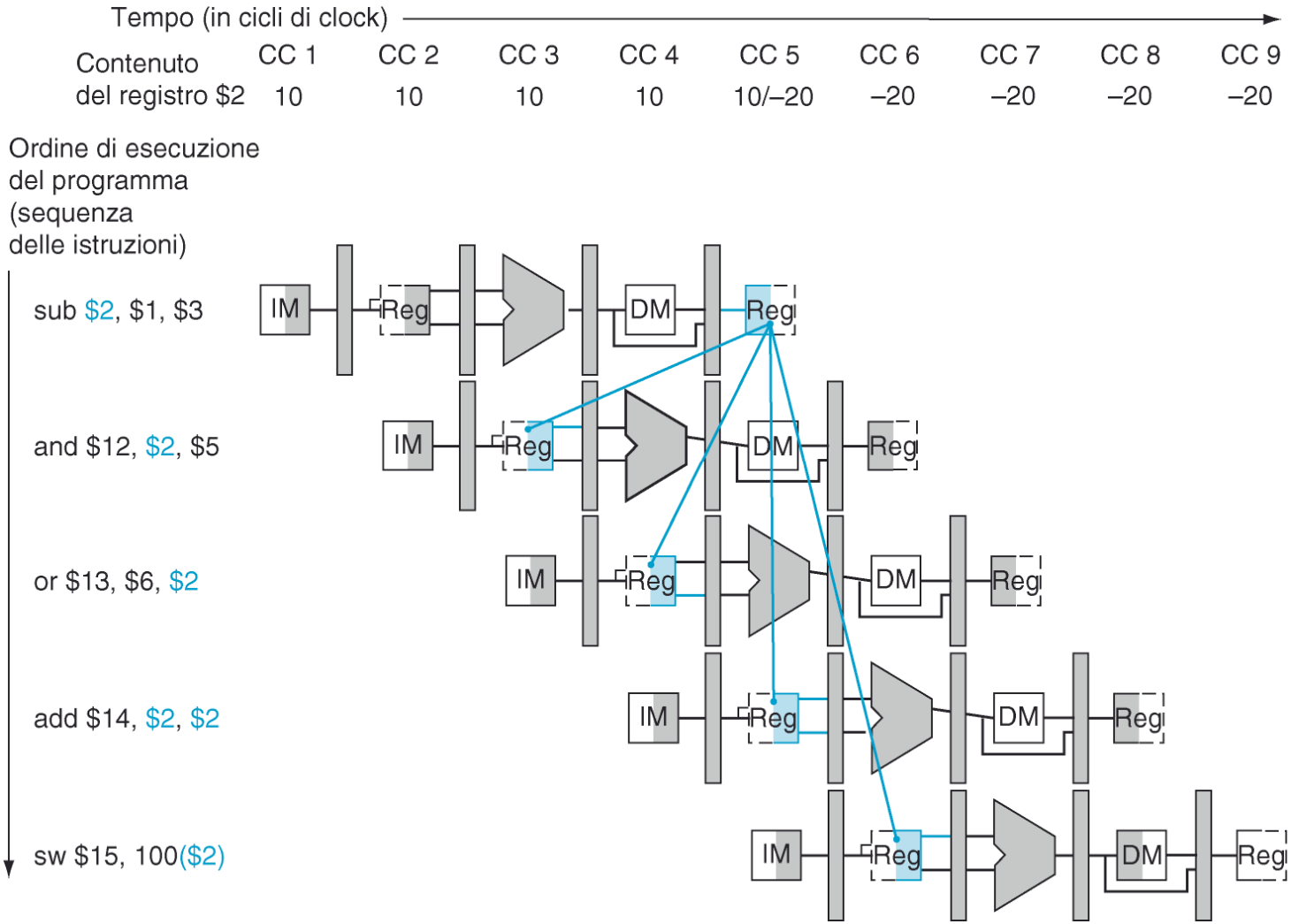
- Dove troviamo queste informazioni? nei registri della pipeline!

Soluzione senza forwarding



- Senza forwarding ad ogni data hazard bisogna allineare la fase WB con la ID (con 2 stalli).
- Per cui: nella parte iniziale ci sono 2 stalli tra li e beq.
- Il ciclo impiega 5 colpi di clock più 2 stalli tra lw e add ed 1 stallo tra addi e beq
- Il control-hazard sul salto beq alla fine del ciclo inserisce 2 stalli (agisce a fine EXE)
- Quindi in totale ci vogliono: 8 + **2** + 10*(5+**3**) + 2 + **2** **= 94 colpi di clock**

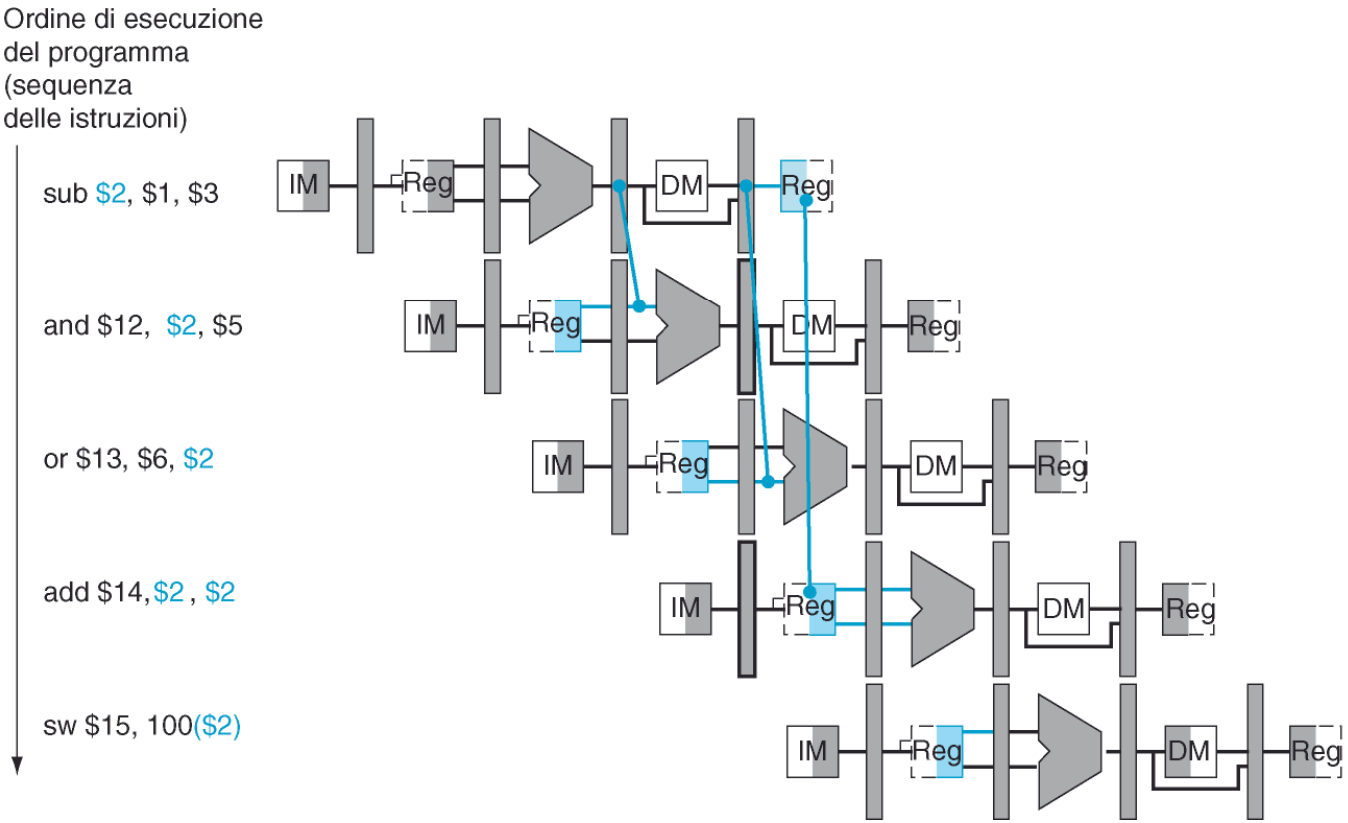
Data hazard senza forwarding



Data hazard con forwarding

Tempo (in cicli di clock) →

	CC 1	CC 2	CC 3	CC 4	CC 5	CC 6	CC 7	CC 8	CC 9
Contenuto del registro $2:	10	10	10	10	10/–20	–20	–20	–20	–20
Contenuto del registro EX/MEM:	X	X	X	–20	X	X	X	X	X
Contenuto del registro MEM/WB:	X	X	X	X	–20	X	X	X	X



Scoprire un data-hazard in EXE

- Il 1° argomento RS della istruzione da eseguire è lo stesso registro della destinazione RD della **precedente** (che è in EXE) … e l'istruzione precedente produce un valore,
  - **ID/EXE . RS = EXE/MEM . RD AND EXE/MEM.RegWrite = 1** (AND RS != 0)
- Oppure lo stesso per il 2° argomento RT ….
  - **ID/EXE . RT = EXE/MEM . RD AND EXE/MEM.RegWrite = 1** (AND RT != 0)

Scoprire un data-hazard in EXE

- Il 1° argomento RS della istruzione da eseguire è lo stesso registro della destinazione RD della **precedente** (che è in EXE) … e l'istruzione precedente produce un valore,
  - **ID/EXE . RS = EXE/MEM . RD AND EXE/MEM.RegWrite = 1** (AND RS != 0)
- Oppure lo stesso per il 2° argomento RT ….
  - **ID/EXE . RT = EXE/MEM . RD AND EXE/MEM.RegWrite = 1** (AND RT != 0)
- **OPPURE**
- Il 1° argomento RS della istruzione da eseguire è lo stesso registro della destinazione RD della **2° istr. prec.** (che è in MEM) … e la 2° istruzione precedente produce un valore
  - **ID/EXE . RS = MEM/WB . RD AND MEM/WB.RegWrite = 1** (AND RS != 0)
- Oppure lo stesso per il 2° argomento RT ….
  - **ID/EXE . RT = MEM/WB . RD AND MEM/WB.RegWrite = 1** (AND RT != 0)

Scoprire un data-hazard in EXE

- Il 1° argomento RS della istruzione da eseguire è lo stesso registro della destinazione RD della **precedente** (che è in EXE) … e l'istruzione precedente produce un valore,
  - **ID/EXE . RS = EXE/MEM . RD AND EXE/MEM.RegWrite = 1** (AND RS != 0)
- Oppure lo stesso per il 2° argomento RT ….
  - **ID/EXE . RT = EXE/MEM . RD AND EXE/MEM.RegWrite = 1** (AND RT != 0)
- **OPPURE**
- Il 1° argomento RS della istruzione da eseguire è lo stesso registro della destinazione RD della **2° istr. prec.** (che è in MEM) … e la 2° istruzione precedente produce un valore
  - **ID/EXE . RS = MEM/WB . RD AND MEM/WB.RegWrite = 1** (AND RS != 0)
- Oppure lo stesso per il 2° argomento RT ….
  - **ID/EXE . RT = MEM/WB . RD AND MEM/WB.RegWrite = 1** (AND RT != 0)

- Se è presente data-hazard su un argomento rispetto a entrambe le istruzioni precedenti **ha sempre la precedenza il valore prodotto per ultimo**, ovvero quello che sta nel registro **EXE/MEM** della pipeline (che è l'istruzione subito precedente a quella corrente in EXE)

Realizzare il forwarding in EXE

- **Forwarding** = sostituire il valore letto dal blocco registri con quello prodotto dalla istruzione precedente che sta in EXE (o dalla 2° istruzione precedente che sta in MEM)

Realizzare il forwarding in EXE

- **Forwarding** = sostituire il valore letto dal blocco registri con quello prodotto dalla istruzione precedente che sta in EXE (o dalla 2° istruzione precedente che sta in MEM)

- **Modifiche al datapath:** inserire un **MUX** prima della ALU per selezionare tra i 3 casi:

- **Non c'è forwarding:** il valore per la ALU viene dal registro **ID/EXE** della pipeline
- **Forwarding dalla istruz. prec.:** il valore per la ALU viene dal registro **EXE/MEM** della p.
- **Forwarding dalla 2° istr. prec.:** il valore per la ALU viene dal registro **MEM/WB** della p.

Realizzare il forwarding in EXE

- **Forwarding** = sostituire il valore letto dal blocco registri con quello prodotto dalla istruzione precedente che sta in EXE (o dalla 2° istruzione precedente che sta in MEM)

- **Modifiche al datapath:** inserire un **MUX** prima della ALU per selezionare tra i 3 casi:
  - **Non c'è forwarding:** il valore per la ALU viene dal registro **ID/EXE** della pipeline
  - **Forwarding dalla istruz. prec.:** il valore per la ALU viene dal registro **EXE/MEM** della p.
  - **Forwarding dalla 2° istr. prec.:** il valore per la ALU viene dal registro **MEM/WB** della p.

- Questo vale sia per il primo argomento che per il secondo argomento della ALU **(2 MUX)**

Realizzare il forwarding in EXE

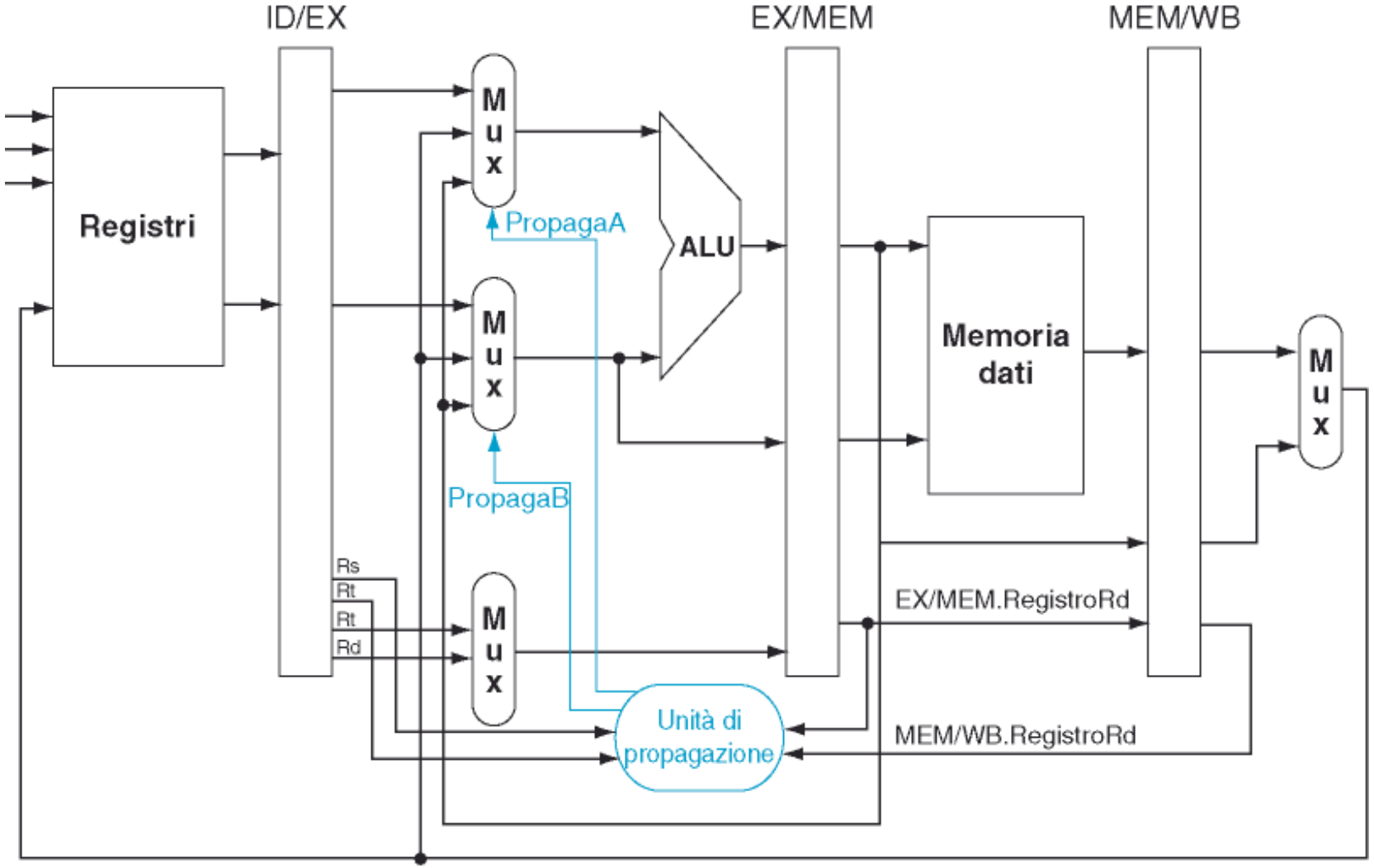
- **Forwarding** = sostituire il valore letto dal blocco registri con quello prodotto dalla istruzione precedente che sta in EXE (o dalla 2° istruzione precedente che sta in MEM)

- **Modifiche al datapath:** inserire un **MUX** prima della ALU per selezionare tra i 3 casi:
- **Non c'è forwarding:** il valore per la ALU viene dal registro **ID/EXE** della pipeline
- **Forwarding dalla istruz. prec.:** il valore per la ALU viene dal registro **EXE/MEM** della p.
- **Forwarding dalla 2° istr. prec.:** il valore per la ALU viene dal registro **MEM/WB** della p.

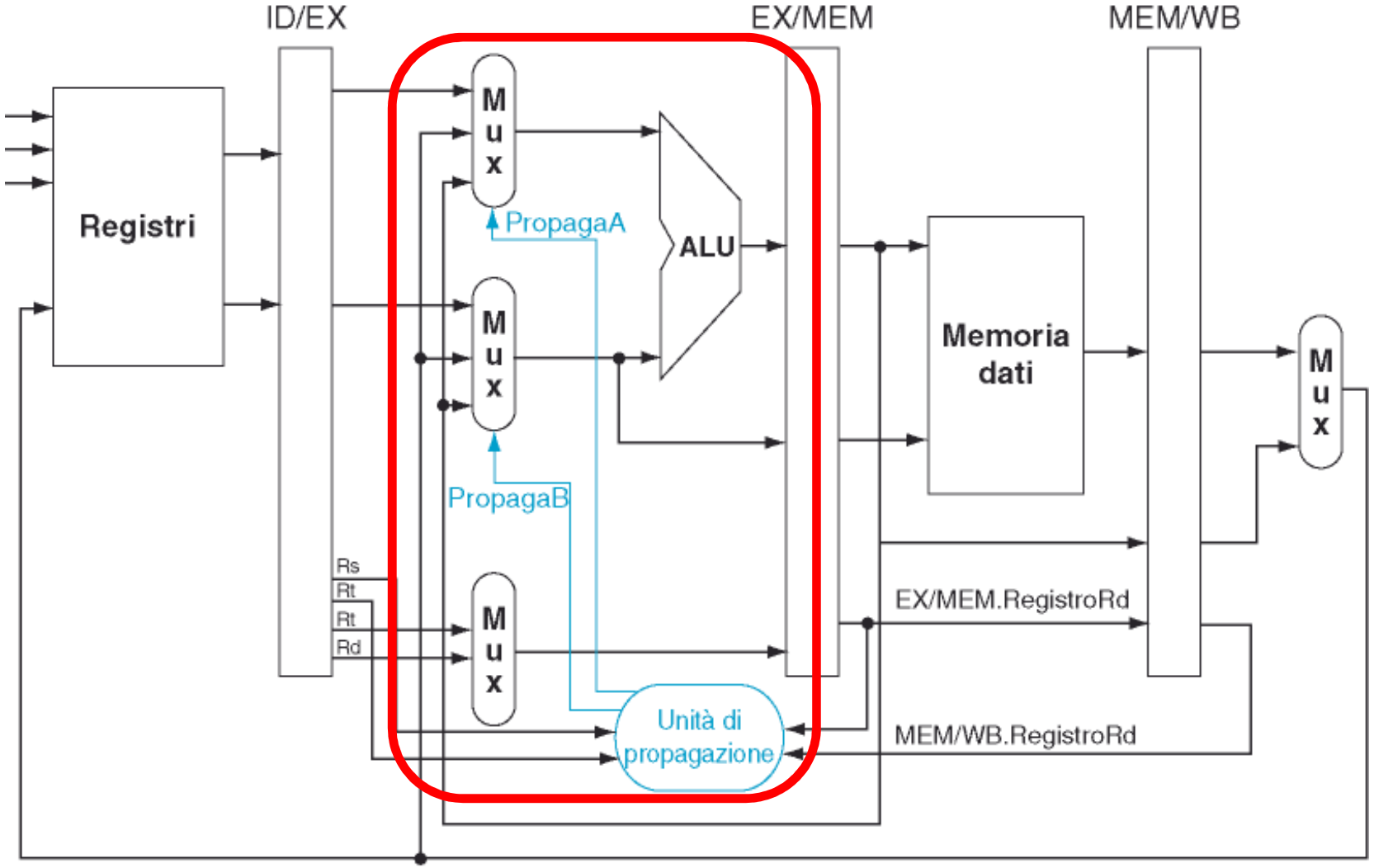
- Questo vale sia per il primo argomento che per il secondo argomento della ALU **(2 MUX)**

- **NOTA:** è possibile realizzare il forwarding anche:
- **nella fase ID** (necessario SOLO se la **beq** viene anticipata in ID)
- **nella fase MEM** (necessario SOLO se **lw $rt** ,… è subito seguita da una **sw $rt, …**)

Unità di forwarding in EXE



Unità di forwarding in EXE



Lo stallo della istruzione

- Talvolta **l'istruzione deve attendere che sia pronto il dato** perché possa avvenire il forwarding
- `lw $s1, vettore($s2) # IF ID EX MM WB`
- `addi $s1, $s1, 42 # IF ID EX MM WB stallo`
- `addi $s1, $s1, 42 # → IF ID EX MM WB ri-exe. + fw`

Lo stallo della istruzione

- Talvolta **l'istruzione deve attendere che sia pronto il dato** perché possa avvenire il forwarding

```
lw $s1, vettore($s2) # IF ID EX MM WB
addi $s1, $s1, 42    #    IF ID EX MM WB     stallo
addi $s1, $s1, 42    #    →  IF ID EX MM WB ri-exe. + fw
```

- Per **riconoscere di dover inserire uno stallo** dobbiamo esaminare le due istruzioni:
- la seconda ha bisogno del dato in EXE ma la prima lo produrrà solo nella fase MEM (**lw**)
  - **ID/EXE.MemRead = 1 AND (ID/EXE.RT = IF/ID.RT OR ID/EXE.RT = IF/ID.RS)**

Lo stallo della istruzione

- Talvolta **l'istruzione deve attendere che sia pronto il dato** perché possa avvenire il forwarding

```
lw $s1, vettore($s2) # IF ID EX MM WB
addi $s1, $s1, 42    #    IF ID EX MM WB     stallo
addi $s1, $s1, 42    #    →  IF ID EX MM WB ri-exe. + fw
```

- Per **riconoscere di dover inserire uno stallo** dobbiamo esaminare le due istruzioni:

- la seconda ha bisogno del dato in EXE ma la prima lo produrrà solo nella fase MEM (**lw**)
  - **ID/EXE.MemRead = 1 AND (ID/EXE.RT = IF/ID.RT OR ID/EXE.RT = IF/ID.RS)**
- la seconda richiede il dato in ID ma la prima lo produrrà solo dopo EXE (**beq** anticipato)
  - **ID/EXE.RegWrite = 1 AND (ID/EXE.RD = IF/ID.RT OR ID/EXE.RD = IF/ID.RS) AND op=beq**

Lo stallo della istruzione

- Talvolta **l'istruzione deve attendere che sia pronto il dato** perché possa avvenire il forwarding

```
lw $s1, vettore($s2) # IF ID EX MM WB
addi $s1, $s1, 42    #    IF ID EX MM WB     stallo
addi $s1, $s1, 42    #    →  IF ID EX MM WB ri-exe. + fw
```

- Per **riconoscere di dover inserire uno stallo** dobbiamo esaminare le due istruzioni:

- la seconda ha bisogno del dato in EXE ma la prima lo produrrà solo nella fase MEM (**lw**)
  - **ID/EXE.MemRead = 1 AND (ID/EXE.RT = IF/ID.RT OR ID/EXE.RT = IF/ID.RS)**
- la seconda richiede il dato in ID ma la prima lo produrrà solo dopo EXE (**beq** anticipato)
  - **ID/EXE.RegWrite = 1 AND (ID/EXE.RD = IF/ID.RT OR ID/EXE.RD = IF/ID.RS) AND op=beq**

- Per **fermare l'istruzione** con uno stallo, dobbiamo (nella fase ID):
  - Annullare l'istruzione che deve attendere (**una bolla che continuerà senza fare nulla**) ovvero basta **azzerare i segnali di controllo MemWrite e RegWrite e IF/ID.Istruzione**

Lo stallo della istruzione

- Talvolta **l'istruzione deve attendere che sia pronto il dato** perché possa avvenire il forwarding

```
lw $s1, vettore($s2) # IF ID EX MM WB
addi $s1, $s1, 42    #    IF ID EX MM WB     stallo
addi $s1, $s1, 42    #    →  IF ID EX MM WB ri-exe. + fw
```

- Per **riconoscere di dover inserire uno stallo** dobbiamo esaminare le due istruzioni:

- la seconda ha bisogno del dato in EXE ma la prima lo produrrà solo nella fase MEM (**lw**)
  - **ID/EXE.MemRead = 1 AND (ID/EXE.RT = IF/ID.RT OR ID/EXE.RT = IF/ID.RS)**
- la seconda richiede il dato in ID ma la prima lo produrrà solo dopo EXE (**beq** anticipato)
  - **ID/EXE.RegWrite = 1 AND (ID/EXE.RD = IF/ID.RT OR ID/EXE.RD = IF/ID.RS) AND op=beq**

- Per **fermare l'istruzione** con uno stallo, dobbiamo (nella fase ID):
  - Annullare l'istruzione che deve attendere (**una bolla che continuerà senza fare nulla**) ovvero basta **azzerare i segnali di controllo MemWrite e RegWrite e IF/ID.Istruzione**
  - **Ripetere l'istruzione che è stata letta e deve entrare nella fase ID** ovvero **impedire l'aggiornamento del registro IF/ID** della pipeline

Lo stallo della istruzione

- Talvolta **l'istruzione deve attendere che sia pronto il dato** perché possa avvenire il forwarding

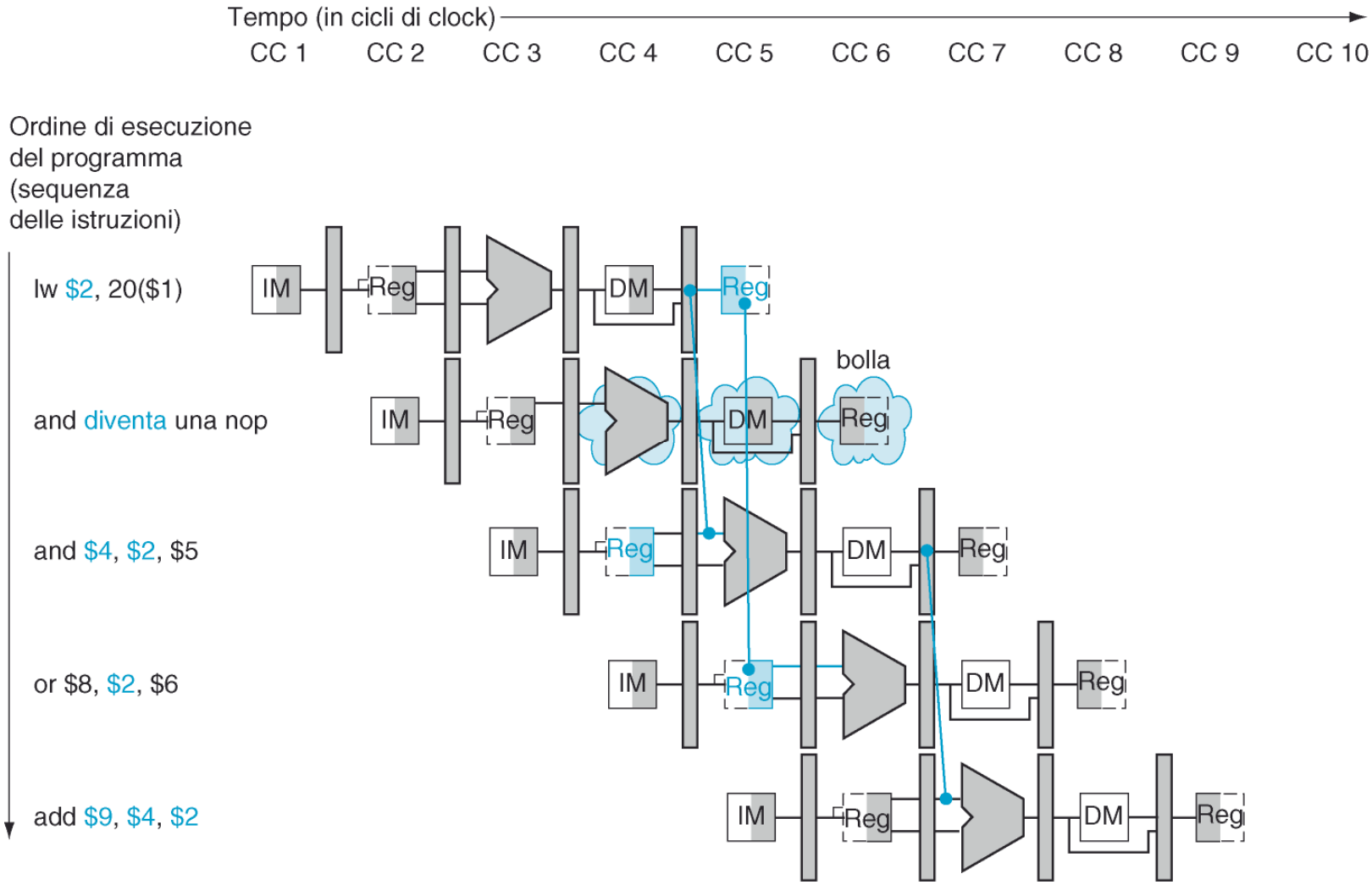
```
lw $s1, vettore($s2) # IF ID EX MM WB
addi $s1, $s1, 42    #    IF ID EX MM WB     stallo
addi $s1, $s1, 42    #    →  IF ID EX MM WB ri-exe. + fw
```

- Per **riconoscere di dover inserire uno stallo** dobbiamo esaminare le due istruzioni:

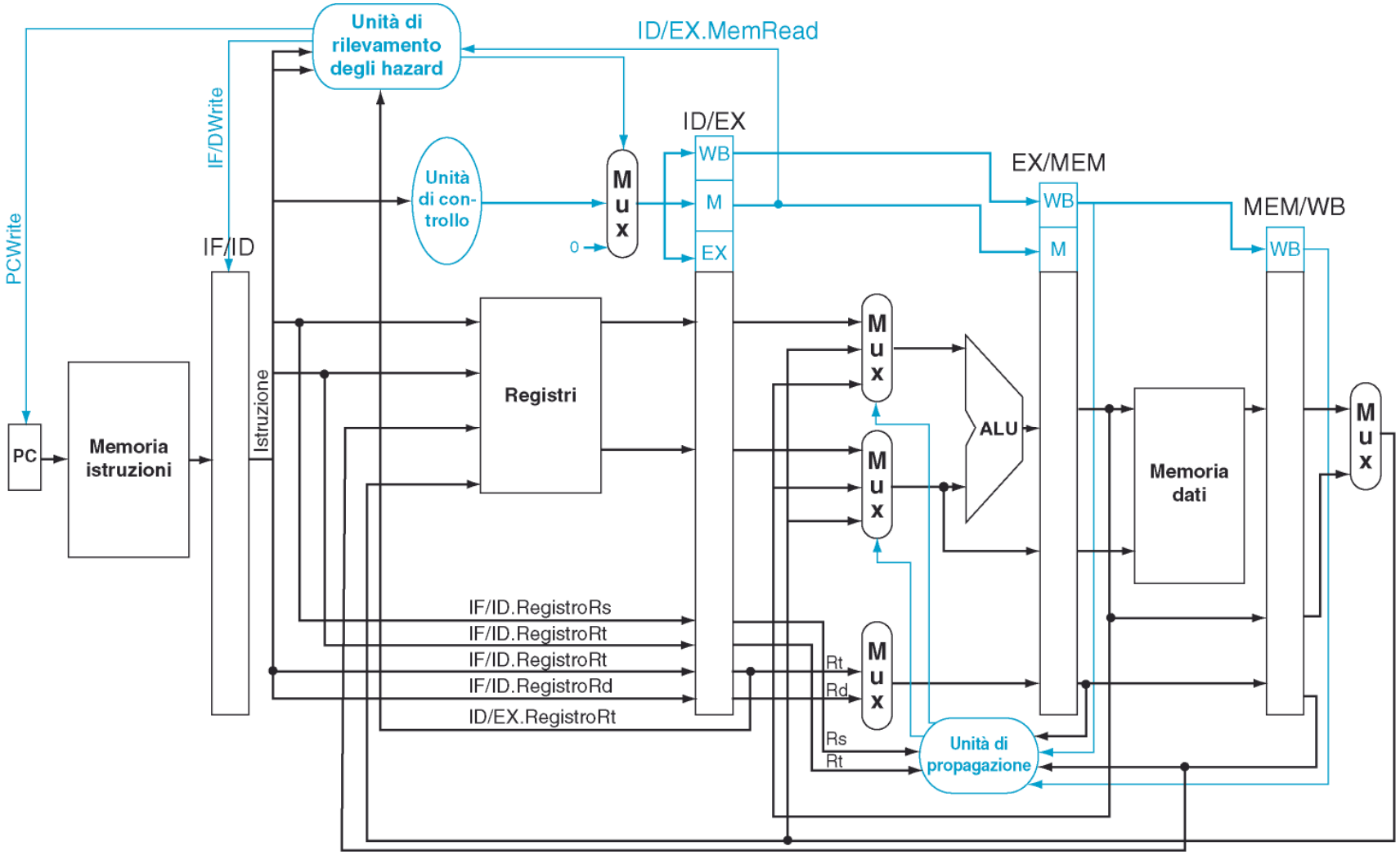
- la seconda ha bisogno del dato in EXE ma la prima lo produrrà solo nella fase MEM (**lw**)
  - **ID/EXE.MemRead = 1 AND (ID/EXE.RT = IF/ID.RT OR ID/EXE.RT = IF/ID.RS)**
- la seconda richiede il dato in ID ma la prima lo produrrà solo dopo EXE (**beq** anticipato)
  - **ID/EXE.RegWrite = 1 AND (ID/EXE.RD = IF/ID.RT OR ID/EXE.RD = IF/ID.RS) AND op=beq**

- Per **fermare l'istruzione** con uno stallo, dobbiamo (nella fase ID):
  - Annullare l'istruzione che deve attendere (**una bolla che continuerà senza fare nulla**) ovvero basta **azzerare i segnali di controllo MemWrite e RegWrite e IF/ID.Istruzione**
  - **Ripetere l'istruzione che è stata letta e deve entrare nella fase ID** ovvero **impedire l'aggiornamento del registro IF/ID** della pipeline
  - **Rileggere la stessa istruzione** di nuovo perchè possa essere rieseguita un clock dopo ovvero basta **impedire che il PC si aggiorni**

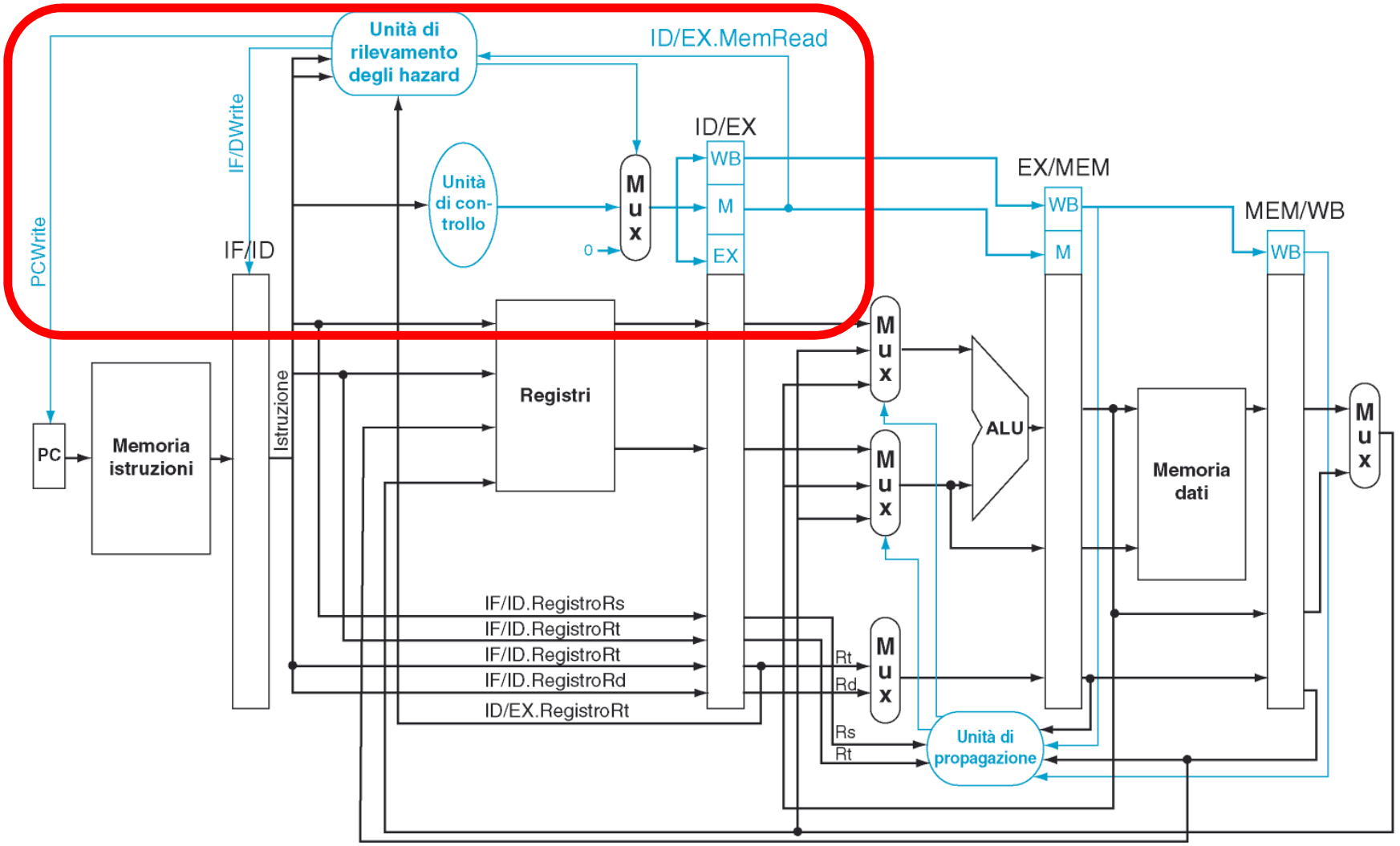
Stallo in azione



Rileviamo e realizziamo lo stallo



Rileviamo e realizziamo lo stallo



Esercizio

- Progettate la unità di forwarding per la fase MEM
  - è necessaria solo per la **sw $rt, ….**
    (quando è preceduta da una **lw $rt, …**)
  - non è possibile aver bisogno di stalli perché non ci sono istruzioni che generano il dato necessario 2 istruzioni prima (ovvero 2 fasi dopo)

- Indicate:
  - quali condizioni devono essere verificate e come farlo con i segnali della pipeline
  - come modificare il datapath per aggiungere il forwarding del valore