Il pipelining: criticità sui dati

Architetture Avanzate dei Calcolatori

Valeria Cardellini

Tipi di criticità sui dati

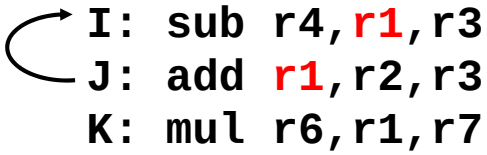
- Consideriamo due istruzioni **`I`** e **`J`**, con **`I`** che precede **`J`** nell'esecuzione
- In generale, le criticità sui dati sono di tre tipi:
  - RAW (**Read After Write**)
  - WAR (**Write After Read**)
  - WAW (**Write After Write**)

- Criticità RAW
  - L'istruzione **`J`** prova a leggere un dato sorgente prima che l'istruzione **`I`** lo abbia scritto
    - L'istruzione **`J`** legge un dato sbagliato

  - Presente nella pipeline MIPS

Tipi di criticità sui dati (2)

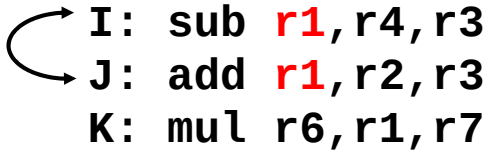
- Criticità WAR
  - L'istruzione *j* prova a scrivere un dato destinazione prima che l'istruzione *i* lo abbia letto
    - L'istruzione *i* legge un dato sbagliato



  - Assente nella pipeline MIPS
    - Tutte le istruzioni richiedono 5 stadi
    - Le operazioni di lettura dei registri avvengono sempre nello stadio ID
    - Le operazioni di scrittura dei registri avvengono sempre nello stadio WB

Tipi di criticità sui dati (2)

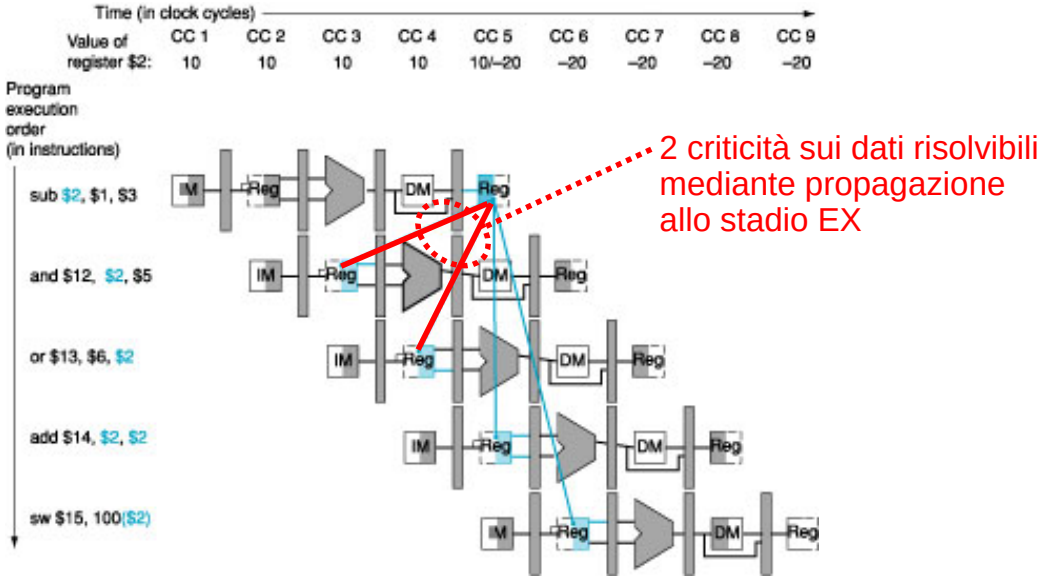
- Criticità WAW
  - L'istruzione *j* prova a scrivere un dato destinazione prima che l'istruzione *i* lo abbia scritto
    - Le operazioni di scrittura vengono eseguite nell'ordine sbagliato



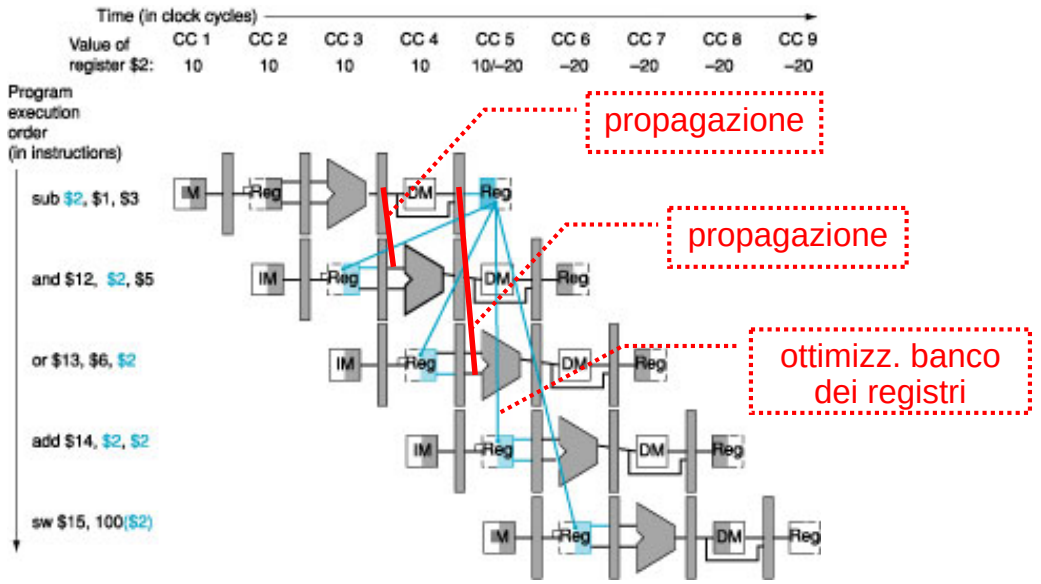
  - Assente nella pipeline MIPS
    - Tutte le istruzioni richiedono 5 stadi
    - Le operazioni di scrittura dei registri avvengono sempre nello stadio finale WB

Criticità sui dati

- Consideriamo una sequenza di 5 istruzioni



Soluzione con propagazione



Soluzione con propagazione (2)

- Consideriamo la prima criticità (EX):

  sub $2, $1, $3 e and $12, $2, $5

  – Il dato prodotto dall'istruzione sub è disponibile alla fine dello stadio EX (CC 3)
  – Il dato è richiesto dall'istruzione and all'inizio dello stadio EX (CC 4)
  – La criticità può essere rilevata quando l'istruzione and si trova nello stadio EX e l'istruzione sub si trova nello stadio MEM

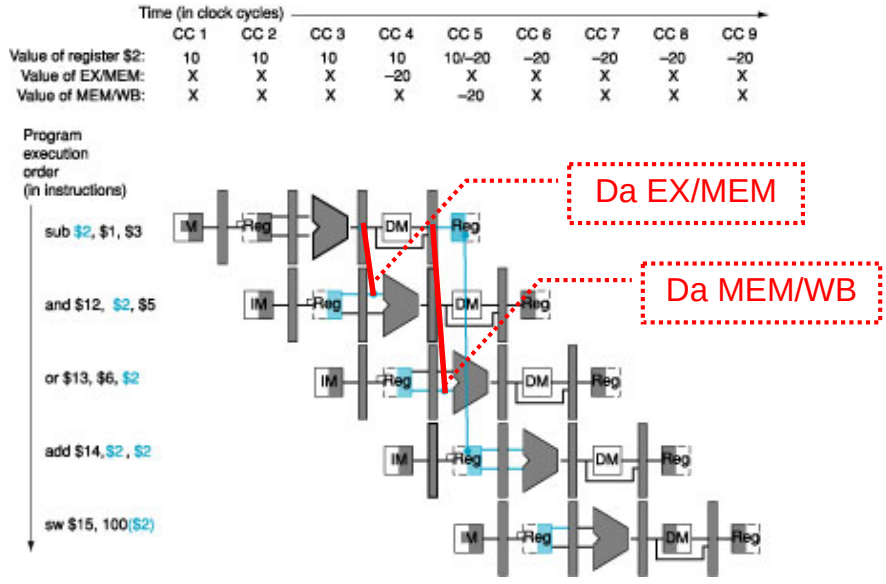
- Consideriamo la seconda criticità (MEM):

  sub $2, $1, $3 e or $13, $6, $2

  – Il dato prodotto dall'istruzione sub è disponibile alla fine dello stadio EX (CC 3)
  – Il dato è richiesto dall'istruzione or all'inizio dello stadio EX (CC 5)
  – La criticità può essere rilevata quando l'istruzione or si trova nello stadio EX e l'istruzione sub si trova nello stadio WB

Propagazione dai registri di pipeline

- Gli ingressi alla ALU sono forniti dai registri di pipeline anziché dal banco dei registri
  – In questo modo le dipendenze sono in avanti nel tempo



Riconoscimento della criticità sui dati

- Usiamo la notazione:
  - *NomeRegistroPipeline.CampoRegistro*
- Condizioni che generano la criticità sui dati
  1a. EX/MEM.RegisterRd = ID/EX.RegisterRs
  1b. EX/MEM.RegisterRd = ID/EX.RegisterRt
  2a. MEM/WB.RegisterRd = ID/EX.RegisterRs
  2b. MEM/WB.RegisterRd = ID/EX.RegisterRt

- Consideriamo la prima criticità (EX):
  sub $2, $1, $3 e and $12, $2, $5
  - E' verificata la condizione 1a
    EX/MEM.RegisterRd = ID/EX.RegisterRs = $2
- Consideriamo la seconda criticità (MEM):
  sub $2, $1, $3 e or $13, $6, $2
  - E' verificata la condizione 2b
    MEM/WB.RegisterRd = ID/EX.RegisterRt = $2

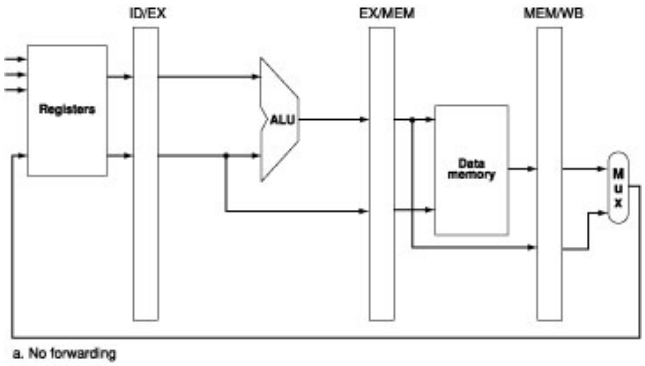
Riconoscimento della criticità sui dati (2)

- Per evitare propagazioni inutili raffiniamo le condizioni
  - Non tutte le istruzioni scrivono un registro
    - Controlliamo se RegWrite è asserito
  - Il registro $0 come destinazione non richiede propagazione
    - Aggiungiamo EX/MEM.RegisterRd≠0 e MEM/WB.RegisterRd≠0

- Quindi le condizioni divengono:
  1a. EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRs)
  1b. EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd = ID/EX.RegisterRt)
  2a. MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRs)
  2b. MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRt)

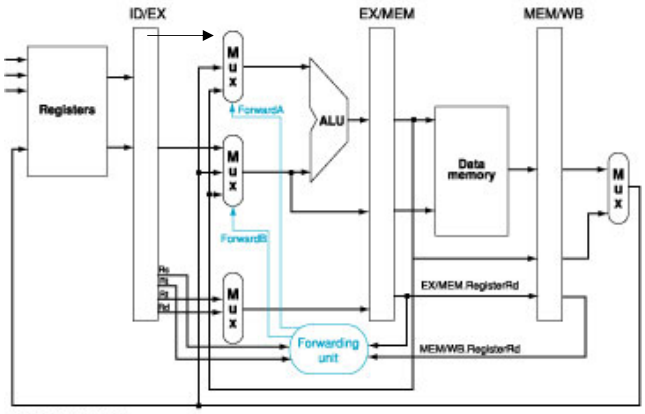
Hardware per la propagazione

- ALU e registri di pipeline senza propagazione



a. No forwarding

- ALU e registri di pipeline con propagazione
  - ***Unità di propagazione*** *(forwarding unit)*: assegna un valore ai segnali di controllo ForwardA e ForwardB per i due mux davanti alla ALU
  - In ID/EX salvato anche Instruction[25-21] (numero del registro sorgente rs)



b. With forwarding

Segnali di controllo per la propagazione

Controllo MUX	Sorgente	Significato
ForwardA = 00	ID/EX	Primo operando della ALU dal banco dei registri
ForwardA = 10	EX/MEM	Primo operando della ALU propagato dal precedente risultato della ALU
ForwardA = 01	MEM/WB	Primo operando della ALU propagato dalla memoria dati o da un precedente risultato della ALU
ForwardB = 00	ID/EX	Secondo operando della ALU dal banco dei registri
ForwardB = 10	EX/MEM	Secondo operando della ALU propagato dal precedente risultato della ALU
ForwardB = 01	MEM/WB	Secondo operando della ALU propagato dalla memoria dati o da un precedente risultato della ALU

Condizioni e segnali di controllo

- Criticità EX

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd ≠ 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRs))
ForwardA = 10

if (EX/MEM.RegWrite
and (EX/MEM.RegisterRd ≠ 0)
and (EX/MEM.RegisterRd = ID/EX.RegisterRt))
ForwardB = 10

Condizioni e segnali di controllo (2)

- Criticità MEM

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠ 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs))
ForwardA = 01

if (MEM/WB.RegWrite
and (MEM/WB.RegisterRd ≠ 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRt))
ForwardB = 01

Condizioni e segnali di controllo (3)

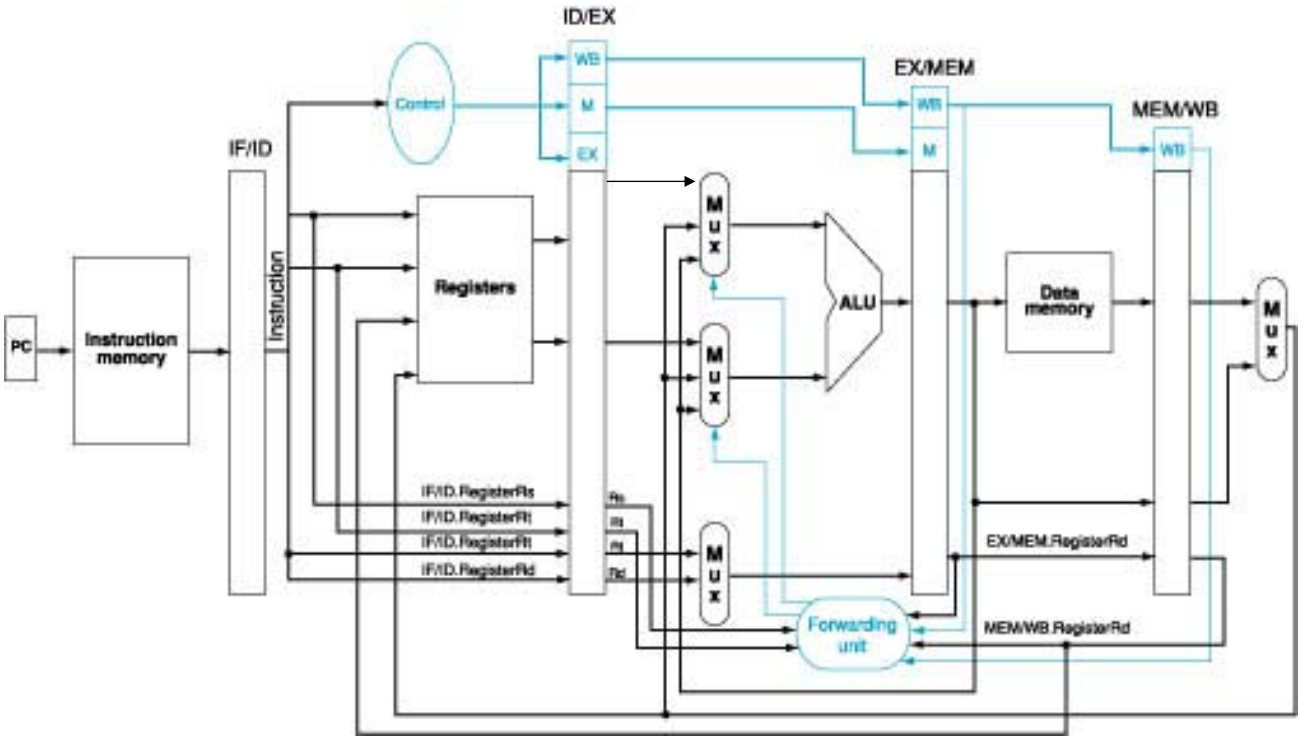
- Potenziale criticità tra risultato dell'istruzione nello stadio WB, risultato dell'istruzione nello stadio MEM e operando sorgente dell'istruzione nello stadio EX
  - Esempio

    add $1, $1, $2
    add $1, $1, $3
    add $1, $1, $4
- Quindi, le condizioni per la criticità MEM diventano:

  if (MEM/WB.RegWrite
  and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRs)
  and (MEM/WB.RegisterRd ≠ 0)
  and (MEM/WB.RegisterRd = ID/EX.RegisterRs)) ForwardA = 01

  if (MEM/WB.RegWrite
  and (EX/MEM.RegisterRd ≠ ID/EX.RegisterRt)
  and (MEM/WB.RegisterRd ≠ 0)
  and (MEM/WB.RegisterRd = ID/EX.RegisterRt)) ForwardB = 01

Unità di elaborazione dati con forwarding



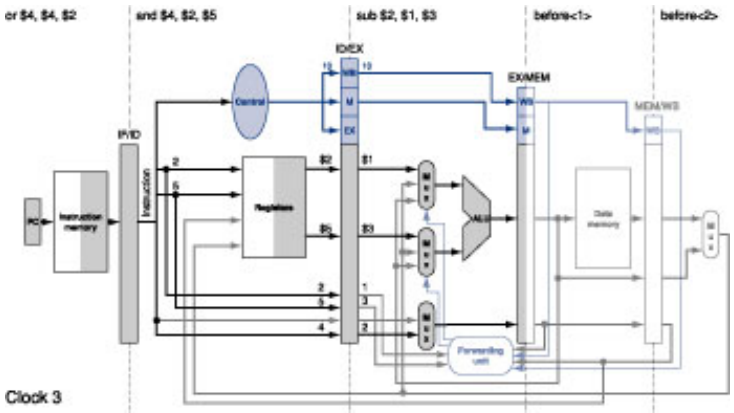
Esempio

- Consideriamo la sequenza di istruzioni MIPS

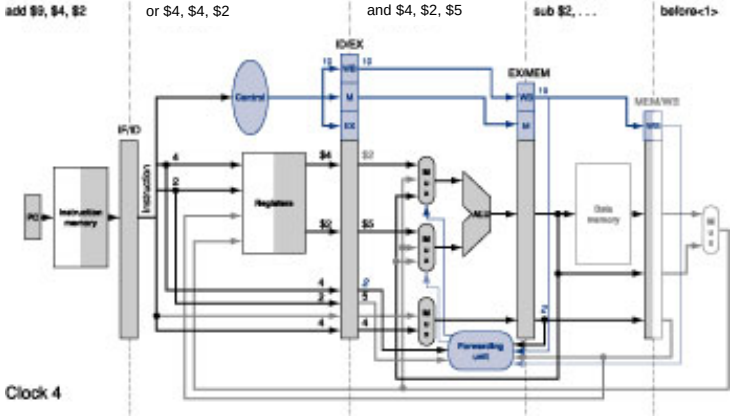
  sub $2, $1, $3
  and $4, $2, $5
  or $4, $4, $2
  add $9, $4, $2
- Analizziamo l'esecuzione della sequenza nei cicli di clock da 3 a 6
  - Ciclo 3: sub in EX
  - Ciclo 4: and in EX
  - Ciclo 5: or in EX
  - Ciclo 6: add in EX
- *Nota*: l'istruzione or ha una doppia criticità sui dati

Esempio: cicli di clock 3 e 4

- or: entra nella pipeline
- Nessuna criticità
- and: in ID/EX vengono scritti $2, $5, 2, 5, e 4 (numeri dei registri)



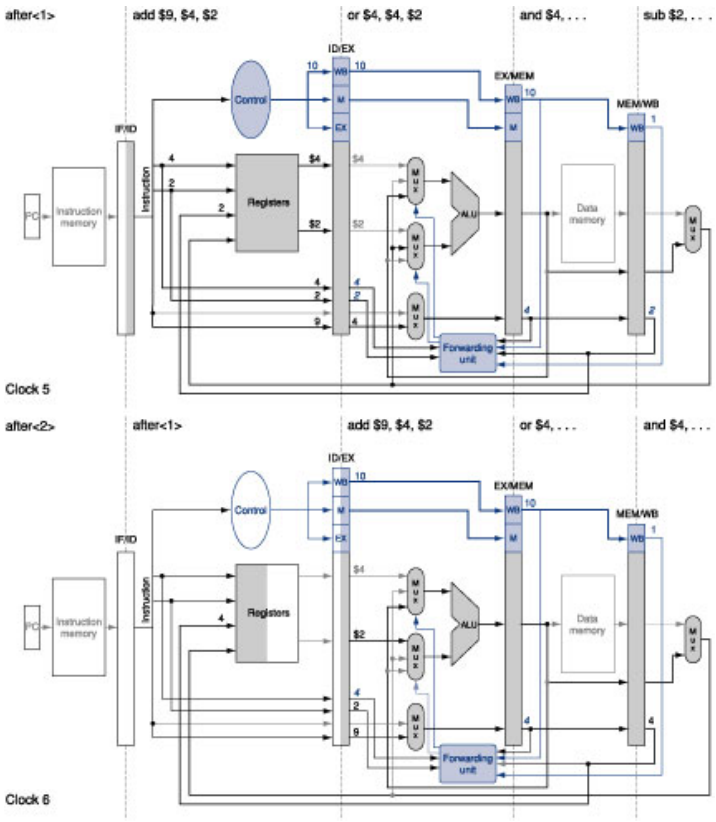
- add: entra nella pipeline
- Criticità tra and e sub su $2
- and: $2 da EX/MEM, $5 da ID/EX



Esempio: cicli di clock 5 e 6

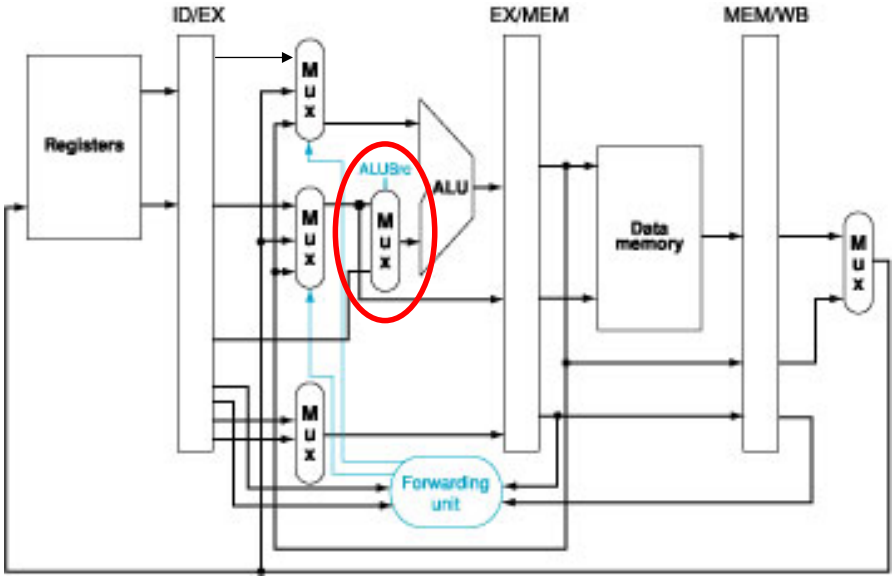
- sub: termina l'esecuzione scrivendo nella prima metà del ciclo di clock $2 nel banco dei registri (no criticità tra add e sub su $2)
- Criticità tra or e and su $4 e tra or e sub su $2
- or: $4 da EX/MEM, $2 da MEM/WB

- and: termina l'esecuzione
- Criticità tra add e or su $4
- add: $4 da EX/MEM, $2 da ID/EX



ALU e registri di pipeline con forwarding

- Aggiungiamo un MUX per scegliere come secondo operando sorgente della ALU anche il valore immediato (esteso in segno a 32 bit)



Esercizio

- Considerare la sequenza di istruzioni MIPS

  add $1, $1, $3

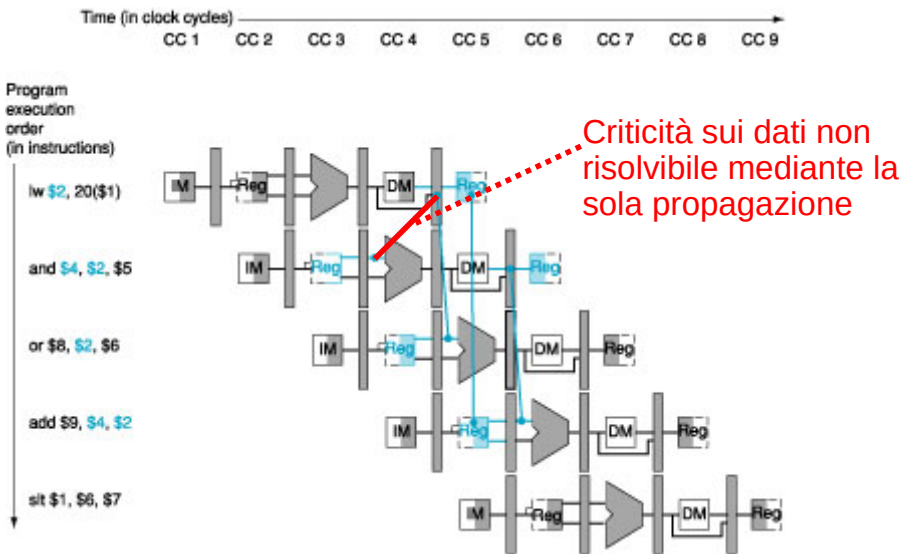
  add $4, $2, $1

  and $5, $4, $1

- Analizzare l'esecuzione della sequenza nei cicli di clock da 3 a 5

Criticità sui dati e stalli

- La propagazione non basta per risolvere una criticità sui dati determinata da un'istruzione che legge il registro scritto dalla precedente istruzione di lw (criticità *load/use*)
  - Occorre individuare la criticità ed inserire uno stallo della pipeline



Condizione per la criticità load/use

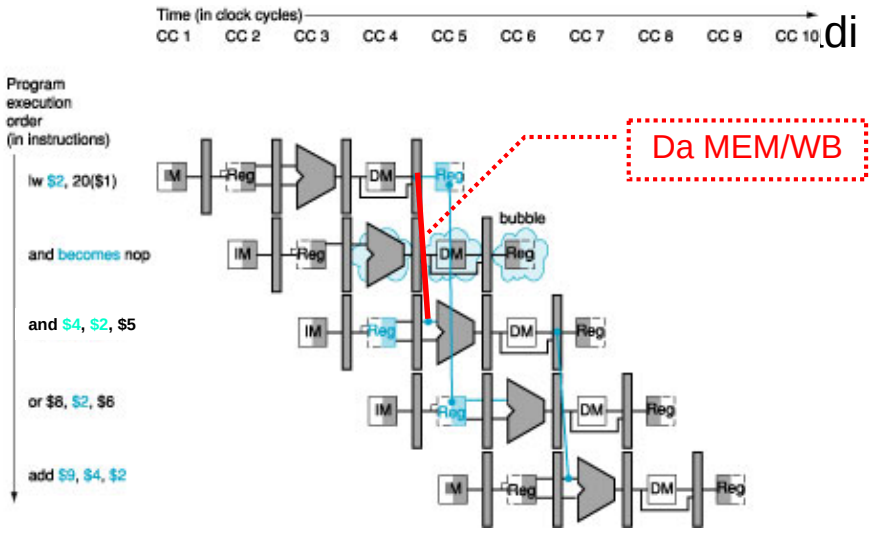
- Condizione per individuare la criticità sui dati di tipo load/use
  - Controllare se istruzione lw nello stadio EX
  - Controllare se il registro da caricare con lw è usato come operando dall'istruzione corrente nello stadio ID
  - In caso affermativo, bloccare la pipeline per un ciclo di clock

    if (ID/EX.MemRead and
    and ((ID/EX.RegisterRt = IF/ID.RegisterRs) or
    (ID/EX.RegisterRt = IF/ID.RegisterRt)))
    stall the pipeline

- Condizione implementata dall'*unità di rilevamento di criticità (hazard detection unit)*

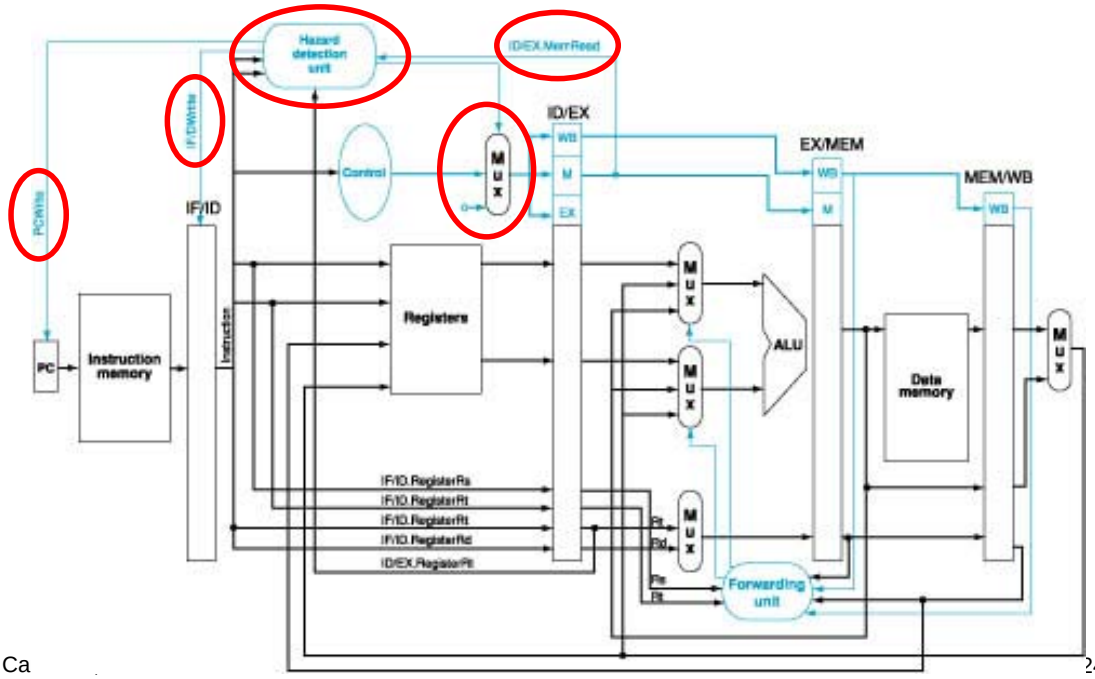
Implementazione di uno stallo

- Per un ciclo di clock
  - Non aggiornare il PC (PCWrite = 0)
  - Mantenere il contenuto del registro IF/ID (IF/IDWrite = 0)
  - di



Unità di elaborazione

- Pipelining
- Propagazione
- Rilevamento di criticità e stallo



Esempio

- Consideriamo la sequenza di istruzioni MIPS

  lw $2, 20($1)

  and $4, $2, $5

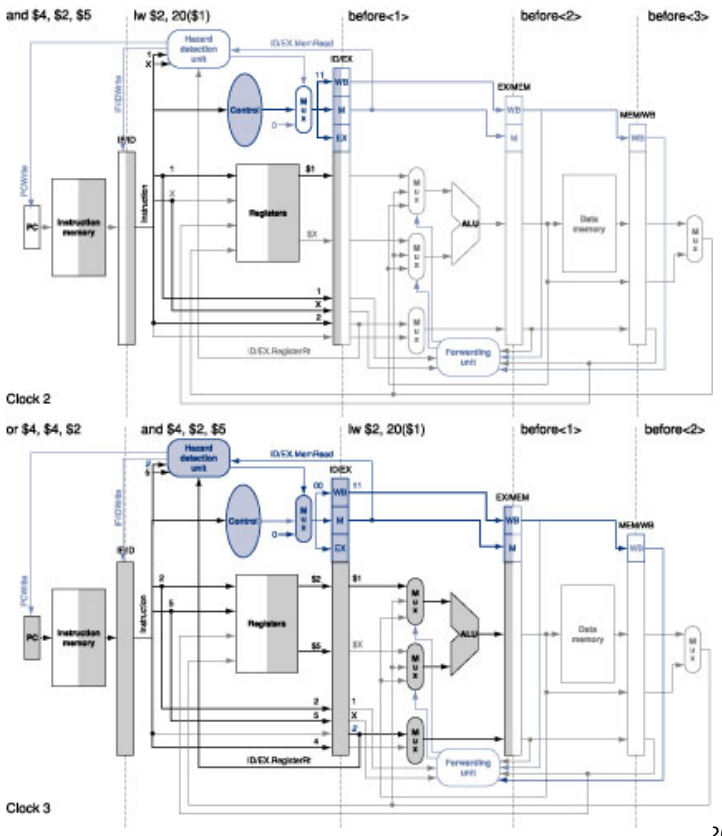
  or $4, $4, $2

  add $9, $4, $2

- Simile all'esempio del lucido 16 ad eccezione della prima istruzione
- Analizziamo l'esecuzione della sequenza nei cicli di clock da 2 a 7

Esempio: cicli di clock 2 e 3

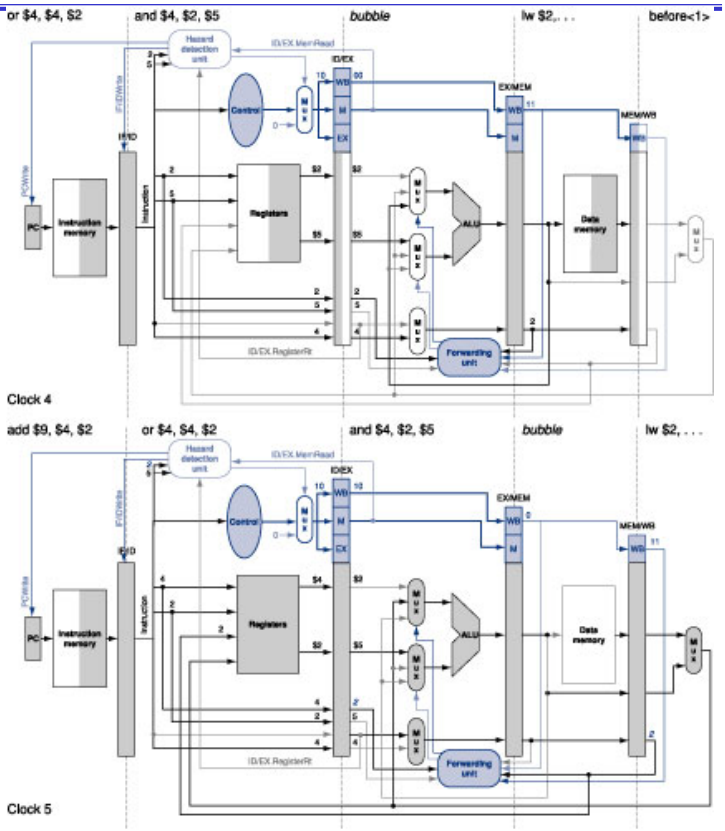
- and: entra nella pipeline
- Nessuna criticità
- lw: In ID/EX vengono scritti $1, 1 e 2 (numeri dei registri)



- or: entra nella pipeline
- and: dovrebbe leggere il valore scritto in $2 da lw. L'unità di rilevamento di criticità blocca l'avanzamento delle istruzioni and e or: PCWrite=0 e IF/ID.Write=0

Esempio: cicli di clock 4 e 5

- Viene inserito lo stallo

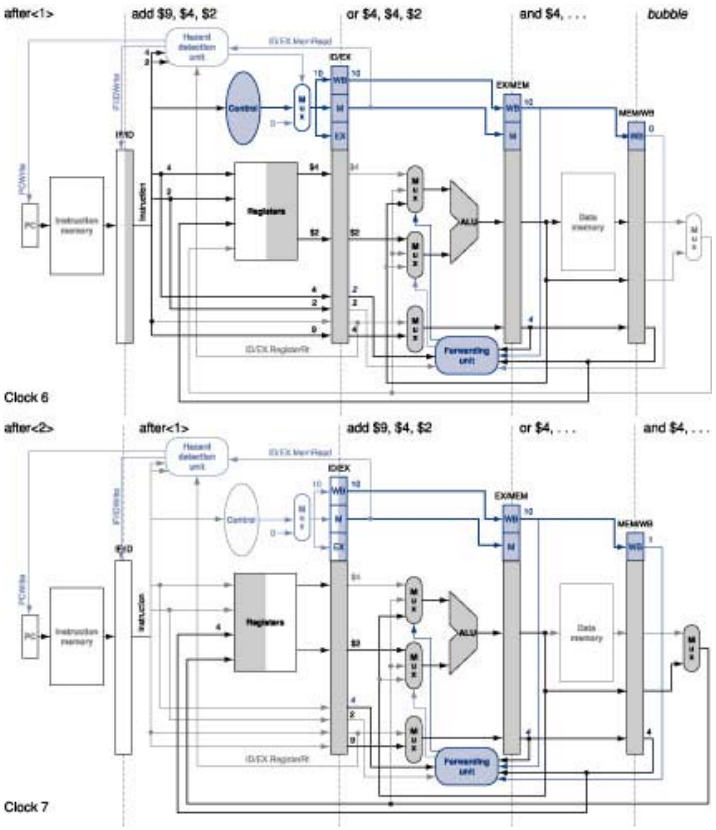


- and e or possono riprendere l'esecuzione
- and: $2 da MEM/WB, $4 da ID/EX

Esempio: cicli di clock 6 e 7

- Lo stallo ha consentito a lw di scrivere nel quinto ciclo: nel sesto ciclo non occorre la propagazione da MEM/WB per or
- or: $4 da EX/MEM e $2 da ID/EX

- Criticità tra add e or su $4
- add: $4 da EX/MEM e $2 da ID/EX



Esercizio

- Considerare la sequenza di istruzioni MIPS

  lw $2, 20($1)

  add $4, $5, $2

  sub $4, $4, $2
- Analizzare l'esecuzione della sequenza nei cicli di clock da 3 a 6