Pipeline

Architettura
degli Elaboratori e delle Reti,
Turno I

Alberto Borghese
Università degli Studi di Milano
Dipartimento di Scienze dell'Informazione
email: borghese@dsi.unimi.it

Rappresentazione grafica della pipeline

Esempio: add $s0, $t0, $t1

I rettangoli grigi a destra indicano lettura, a sinistra indicano scrittura. I componenti bianchi, indicano il loro non utilizzo.

Criticità

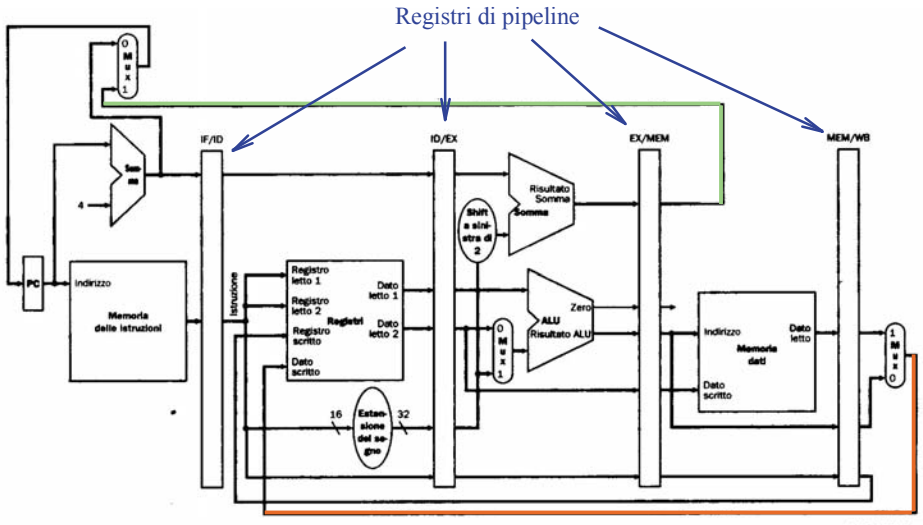
Strutturali:
•Dovrei utilizzare la stessa unità funzionale due volte nello stesso passo.
•Le unità funzionali non sono in grado di supportare le istruzioni (nelle diverse fasi) che devono essere eseguite in un determinato ciclo di clock.

Controllo:
• Dovrei prendere una decisione (sull'istruzione successiva) prima che l'esecuzione dell'istruzione sia terminata (e.g. branch).

Dati:
• Dovrei eseguire un'istruzione in cui uno dei dati è il risultato dell'esecuzione di un'istruzione precedente.

add $s0, $t1,$t1
add $s2, $s0, $t3

CPU con pipeline



Criticità nella CPU

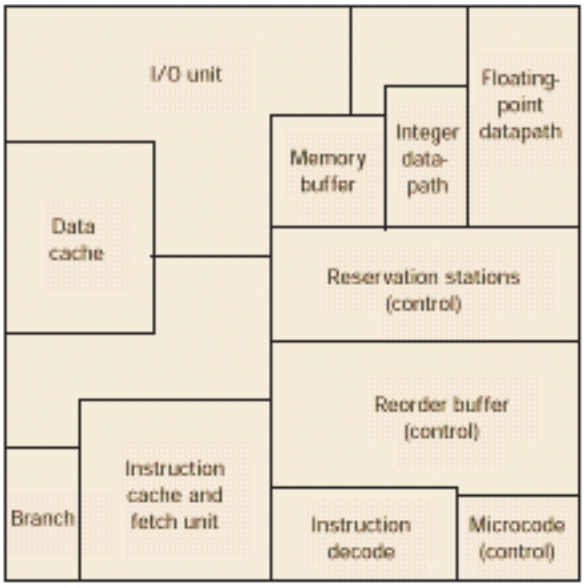
Le criticità strutturali sono risolte con la duplicazione (suddivisione) delle unità funzionali.

Prelievo nello stadio MEM dell'indirizzo di salto.

Prelievo nello stadio WB del risultato dell'operazione.

Il processore Pentium Pro

• 5,5 Milioni di transistor su 306mm^2 (2cm x 1.5cm) con cache esterna da 31 milioni.

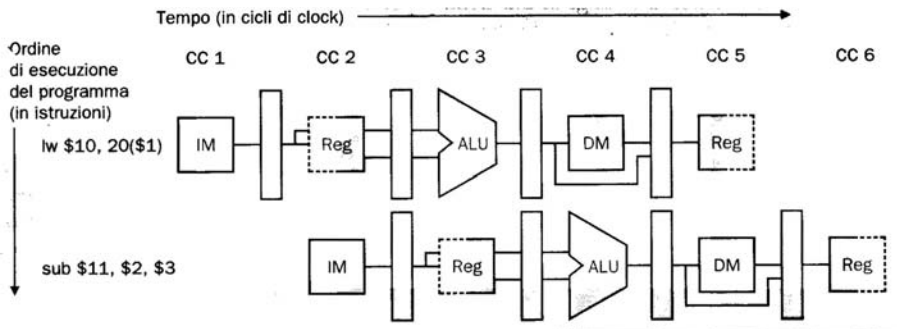


Pipeline per l'istruzione lw

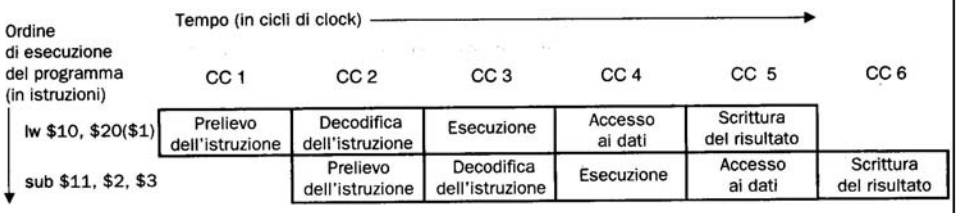
Passo esecuzione	ALU	ALU PC	ALU branch	Memoria Dati	Memoria Istruzioni	Register File
IF (Fase fetch)	N0	Yes	N0	N0	Yes	N0
ID (Decodifica)	N0	N0	N0	N0	N0	Yes
EX (Esecuzione)	Yes	N0	Yes	N0	N0	N0
MEM (Accesso memoria)	N0	N0	N0	Yes	N0	N0
WB (riscrittura)	N0	N0	N0	Yes	N0	Yes

Passi della 1a istruzione lw	IF	ID	EX	MEM	WB		
…..							
lw $t0, 8($t2)	ALU-PC Mem-Istr	RF	ALU	Mem	RF		
lw $t1, 12($t2)		ALU-PC Mem-Istr	RF	ALU	Mem	RF	
lw $t1, 16($t2)			ALU-PC Mem-Istr	RF	ALU	Mem	RF
…..							

Visualizzazione grafica di una pipeline



Visualizzazione grafica di una pipeline

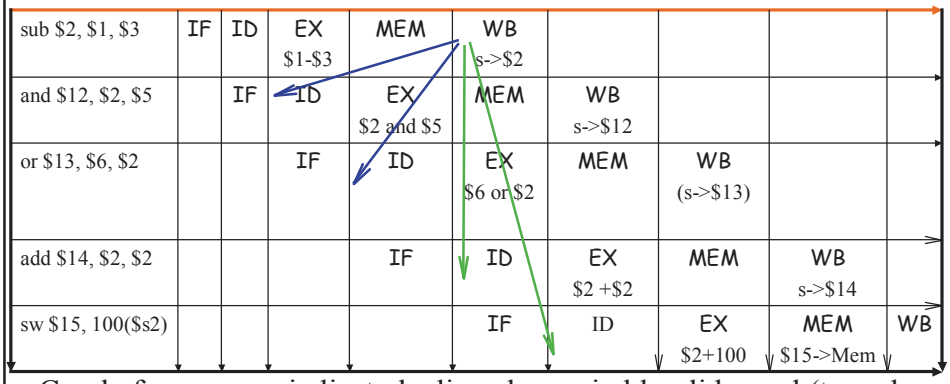


Criticità nei dati

Dovrei eseguire un'istruzione in cui uno dei dati è il risultato dell'esecuzione di un'istruzione precedente.

Soluzione mediante due tecniche:
•Riorganizzazione del codice (compilatore).
•Propagazione (forwarding) o scavalcamento (bypassing).

Esempio Hazard nei dati



Con le frecce sono indicate le dipendenze, in blu gli hazard (tra sub e and e tra sub e or).

Il dato in $2 viene pronto nel Register File nella fase di WB della sub. Non è ancora pronto quando viene effettuata la decodifica della and e della or successiva.

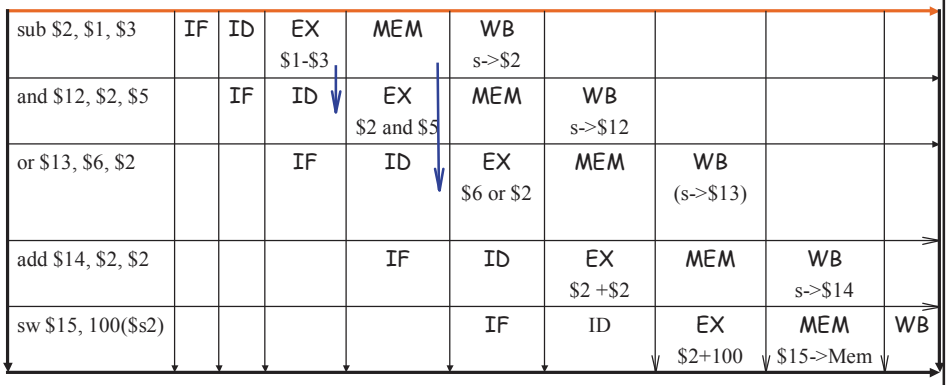
Hazard nei dati: soluzione tramite compilatore

```
sub $2, $1, $3
nop
nop
and $12, $2, $5
or $13, $6, $2
add $14, $2, $2
sw $15, 100($s2)
```

Spreco di 2 cicli di clock (in modo che la fase ID dell'istruzione and $12, $2, $5 vada a coincidere con la fase di WB della sub $2, $1, $3).

Situazione troppo frequente perché la soluzione sia accettabile.

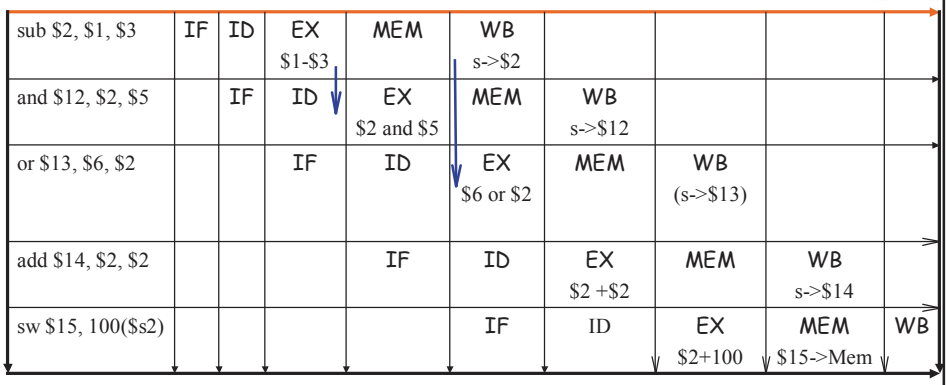
Hazard tra sub e and



• Prendo il risultato della sottrazione all'inizio dello stadio MEM (per le istruzioni R, lo stadio M è uno stadio idle): $1-$3 è già contenuto nel registro EX/MEM e si trova all'uscita del registro nella fase bassa del clock.

• Sovrascrivo il primo operando della *and* successiva ($2) con il risultato della sottrazione eseguita in precedenza (contenta all'uscita del registro EX/MEM), senza attendere la fase di WB.

Hazard tra sub e or

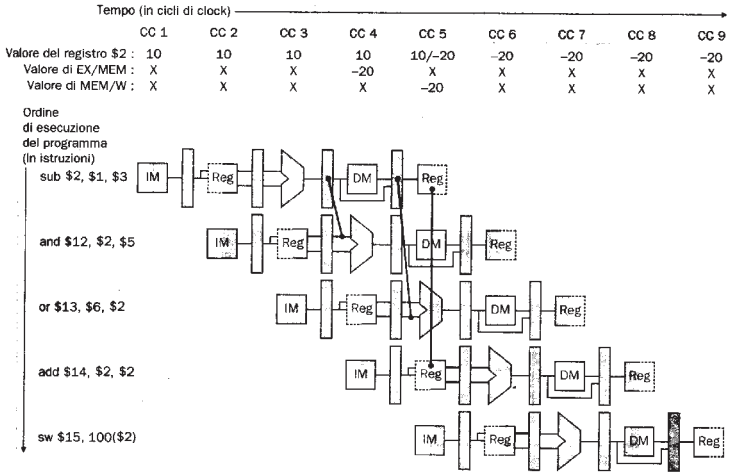


• Prendo il risultato della sottrazione all'inizio dello stadio WB, $1-$3, che è contenuto all'uscita del registro MEM/WB nella fase bassa del clock.

• Sovrascrivo il primo operando della *and* successiva ($2) con il risultato della sottrazione eseguita in precedenza (contenta all'uscita del registro MEM/WB), senza attendere la fase di WB.

Hazard nei dati: problema di disponibilità dei dati

Invia il risultato dell'istruzione precedente ad un passo intermedio dell'istruzione attuale (in pipe): *forwarding (corto-circuito la pipe-line).* Per questo motivo il contenuto dei registri EX/MEM e MEM/WB anticipa il contenuto del RegisterFile.



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.

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.

Soluzione architetturale per and

Occorre implementare la seguente funzione logica nella fase ID dell'esecuzione dell'istruzione *and $12, $2, $5:*

IF ($RD_{SUB} == RS_{AND}$)
then "cortocircuita RD_{t+1} con RS_{AND})

IF ($RD_{SUB} == RT_{AND}$)
then "cortocircuita RD_{t+1} con RT_{AND})

NB: $RD_{t+1} = RD_{SUB}$ = **$2**, ed è il contenuto del registro destinazione dell'istruzione precedente la *and* (*sub **$2**, $1, $3*) che è copiato nella fase MEM.

Soluzione architetturale per or

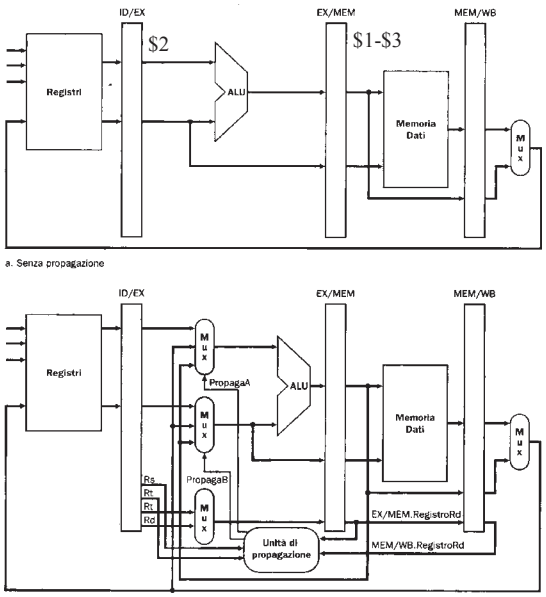
Occorre implementare la seguente funzione logica nella fase ID dell'esecuzione dell'istruzione *or $13, $6, $2*:

IF ($RD_{SUB} == RS_{AND}$)
then "cortocircuita RD_{t+2} con RS_{AND})

IF ($RD_{SUB} == RT_{AND}$)
then "cortocircuita RD_{t+2} con RT_{AND})

NB: $RD_{t+2} = RD_{SUB}$ = **$2**, ed è il contenuto del registro destinazione di 2 istruzioni precedenti la *or* (*sub **$2**, $1, $3*) che è copiato nella fase MEM.

Hazard nei dati: soluzione architetturale



Controllo Mux ingrsso alla ALU

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

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.

•Inserire una nop.

•Accettare uno stallo (non sempre si può evitare).