


Pipeline – criticità

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

Università degli Studi di Milano

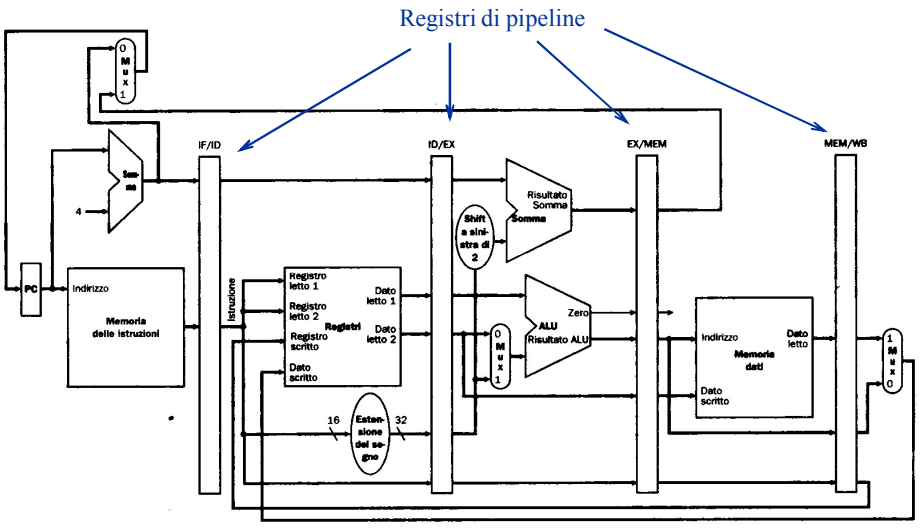
Riferimento al Patterson: 4.5, 4.6


Sommario

La CPU con pipeline

Le criticità

CPU con pipeline



Gli stadi di esecuzione

IF – Instruction Fetch
ID – Instruction Decode (e lettura register file)
EX – Esecuzione o calcolo dell'indirizzo di memoria.
MEM – Accesso alla memoria dati.
WB – Write Back (scrittura del risultato nel register file).

NB: I registri al termine di ogni fase prendono il nome dalle 2 fasi:
IF/ID ID/EX EX/MEM MEM/WB
Perchè non c'è un registro WB/IF?

Il data-path procede da sx a dx.

Da dx a sx si ha la scrittura del PC e la scrittura nel Register File che creano criticità (vanno contro-corrente).

Supponiamo che ciascuno stadio abbia la sua unità di controllo.

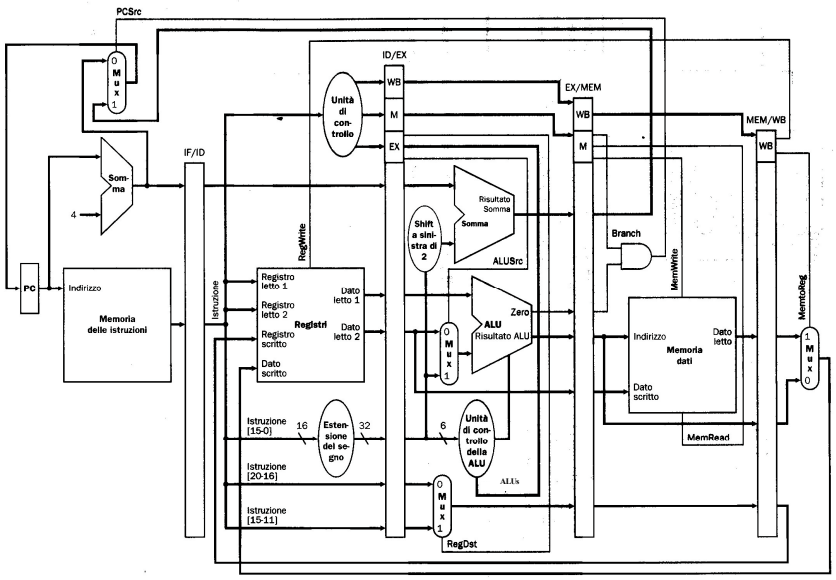
Il ruolo dei registri

Ciascuno stadio produce un risultato. La parte di risultato che serve agli stadi successivi deve essere memorizzata in un registro.

Il registro mantiene l'informazione anche se lo stadio in questione riutilizza l'unità funzionale.

Esempio: l'istruzione letta viene salvata nel registro IF/ID (cf. Instruction Register).

CPU con pipeline



Esempio di esecuzione

Cosa si trova nei diversi cicli di clock nella pipeline durante l'esecuzione di queste istruzioni?

lw $s0, 20($s1)	sub $t0, $t1, $t2	add $t1, $t2, $t3
beq $s1, $s2, 20	sw $t2, 36($t3)	or $t6, $s0, $s1

NB Occorre specificare il contenuto della parte master e slave dei registri di pipeline.

Esempio di esecuzione

Cosa si trova nella pipeline durante l'esecuzione di questo segmento di codice (dati + controllo)?

0x4000 lw $t1, 24($t2)
0x4004 beq $t1, $s2, 20
0x4008 add $s0, $t1, $s2
0x400C sw $t2, 36($t1)
0x4010 sub $t0, $t1, $t2
0x4014 or $s3, $t4, $t5

NB Occorre specificare il contenuto della parte master e slave dei registri di pipeline.

I segnali di controllo fluiscono anch'essi da sinistra verso destra.

Il segnale di scrittura nel register file viene generato nella fase ID ma viene consumato nella fase WB.
Il segnale di scrittura nel register file proviene dalla fase di WB.

Sommario

La CPU con pipeline

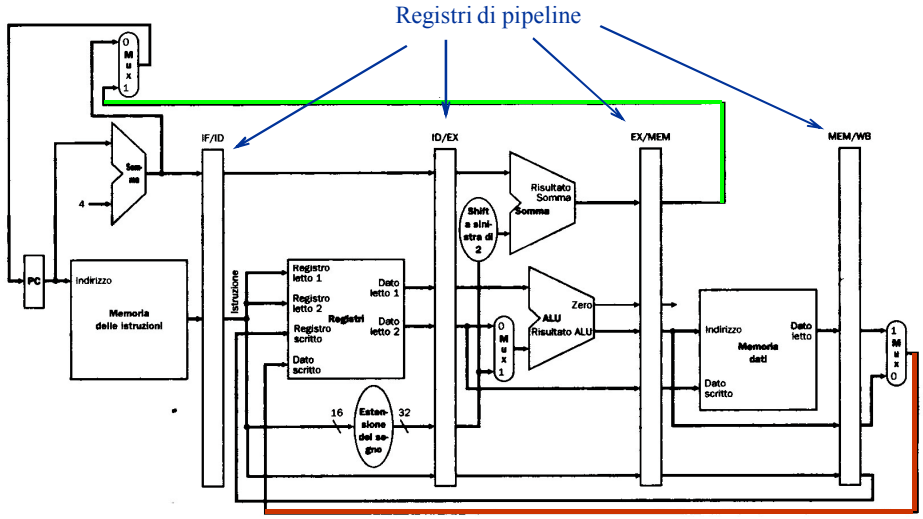
Le criticità

Sommario

La CPU con pipeline

Le Criticità

CPU con pipeline



Criticità (hazard)

Un'istruzione non può essere eseguita nel ciclo di clock immediatamente successivo a quella precedente (mancano i dati necessari alla lavorazione di un qualche suo stadio).

Strutturali:

- Dovrei utilizzare la stessa unità funzionale due volte nello stesso ciclo di clock (e.g. se non avessi duplicato la memoria)..

Controllo:

- Dovrei prendere una decisione (sull'istruzione successiva) prima che l'esecuzione dell'istruzione corrente sia terminata (e.g. Istruzioni successive ad una branch).

Dati:

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

Esempio:

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

Soluzione delle criticità strutturali

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

Triplicazione delle ALU

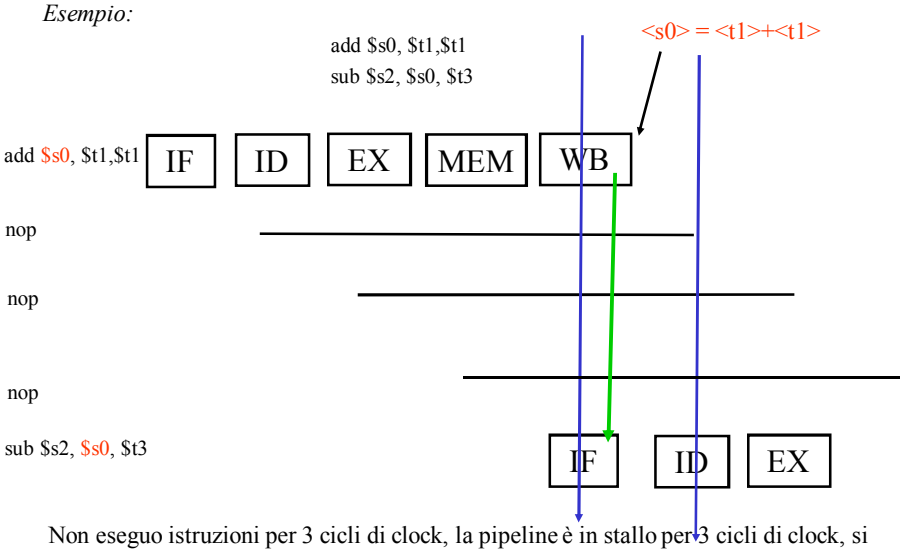
Duplicazione della Memoria (stessa memoria ma separazione della memoria dati dalla memoria istruzioni).

Esempio di Hazard sui dati

sub $s2, $s1, $s2	IF	ID	EX $1-$2	MEM	WB s->$2				
add $t2, $s2, $s5		IF	ID	EX $2 and $5	MEM	WB s->$t2			
or $t3, $s6, $s2			IF	ID	EX $6 or $2	MEM	WB (s->$t3)		
and $t4, $s2, $3				IF	ID	EX $2 & $3	MEM	WB s->$t4	
sw $t5, 100($s2)					IF	ID	EX $2+100	MEM $t5 ->Mem	WB

Soluzione mediante stallo

Esempio:



Non eseguo istruzioni per 3 cicli di clock, la pipeline è in stallo per 3 cicli di clock, si formano 3 bolle (bubbles) nel funzionamento della pipeline.

Hazard nei dati: soluzione tramite compilatore

add $s0, $t1,$t1
nop
nop
nop
sub $s1, $s0, $s0
and $t2, $t0 $t1
or $t5, $t3, $t4
add $s2, $s7, $t7
sw $3, 100($t0)

add $s0, $t1,$t1
nop
nop
nop
sub $s1, $s0, $s0
and $t2, $t0 $t1
or $t5, $t3, $t4
add $s2, $s7, $t7
sw $3, 100($t0)

add $s0, $t1,$t1
and $t2, $t0 $t1
or $t5, $t3, $t4
add $s2, $s7, $t7
sub $s2, $s0, $t3
sw $15, 100($s2)

Spreco di 3 cicli di clock (in modo che la fase IF dell'istruzione *sub $s2, $s0, $t3* vada a coincidere con la fase di WB della *add $s0, $t1,$t1*.

Situazione troppo frequente perché la soluzione sia accettabile.

Il codice viene riorganizzato in fase di compilazione. Non sempre è possibile o efficace.

Sommario

Criticità in una pipeline

Dipendenza tra i dati e propagazione

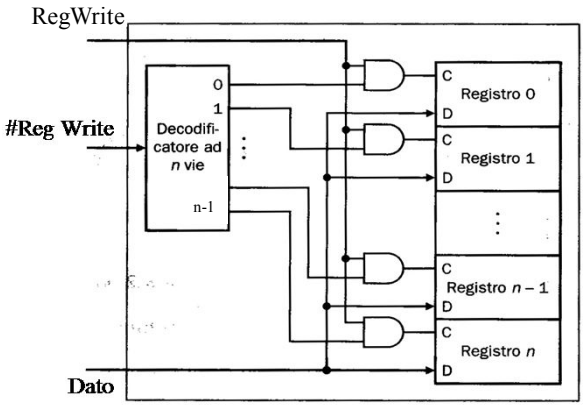
Propagazione a due passi

Hazard sui dati

sub $s2, $s1, $s2	IF	ID	EX $s1-$s2	MEM	WB s->$2				
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)		
and $t4, $s2, $s3				IF	ID	EX $s2 & $3	MEM	WB s->$t4	
sw $t5, 100($s2)					IF	ID	EX $s2+100	MEM $t5 ->Mem	WB

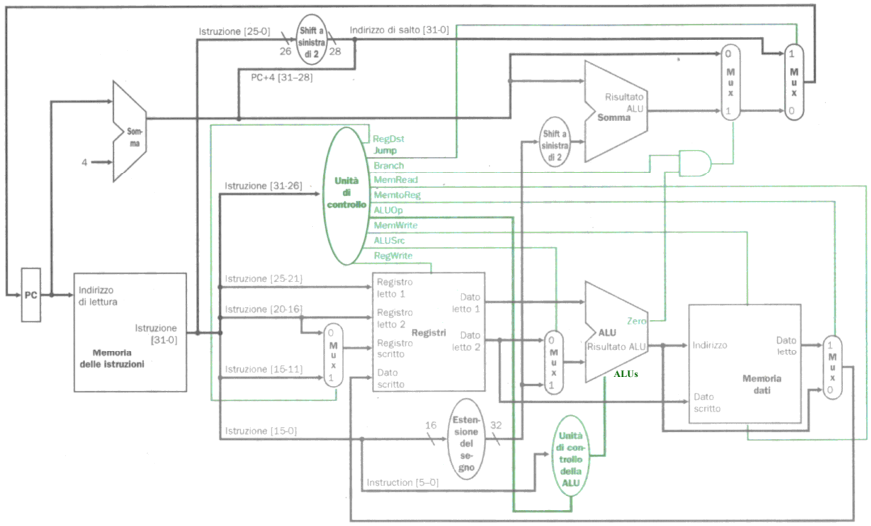
Con le frecce sono indicate le dipendenze, in blu gli hazard (tra sub e and, sub e add).
Il dato in $s2 viene scritto nel Register File nella fase di WB della sub, è pronto al clock successivo. Non è ancora pronto quando viene effettuata la decodifica della and, della or e della add successiva.

E’ una criticità il Register File?



No, a patto che utilizziamo Latch di tipo D e non flip-flop come nel caso della CPU singolo ciclo

Pipeline a singolo ciclo

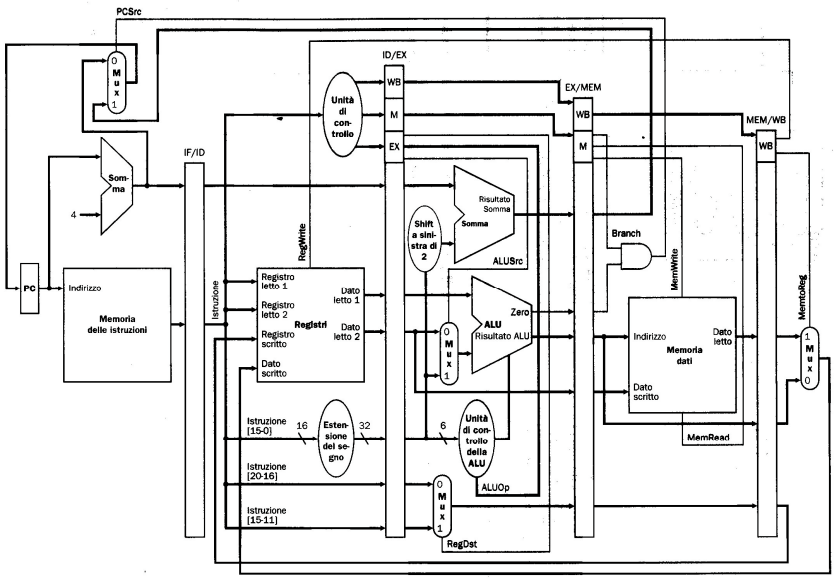


Hazard sui dati

sub $s2, $s1, $s2	IF	ID	EX $s1-$s2	MEM	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)		
and $t4, $s2, $s3				IF	ID	EX $s2 & $s3	MEM	WB s->$t4	
sw $t5, 100($s2)					IF	ID	EX $s2+100	MEM $t5 ->Mem	WB

Con le frecce sono indicate le dipendenze, in blu gli hazard (tra sub, e and e or), dopo la modifica del RegisterFile: il dato è disponibile in lettura, già nella prima parte del clock.
Il dato in $s2 viene scritto nel Register File nella fase di WB della sub, è pronto al clock successivo. Non è ancora pronto quando viene effettuata la decodifica della and e della or successiva.

CPU con pipeline



Sommario

La CPU con pipeline

Le Criticità