


Unità di controllo della pipeline

Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

Università degli Studi di Milano

Riferimento al Patterson: 4.5, 4.6

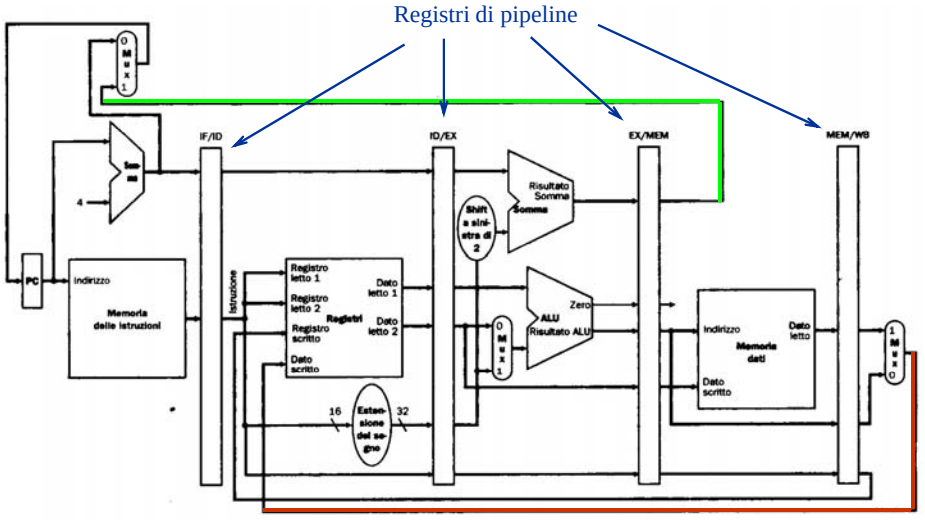

Sommario

La CPU con pipeline

L'Unità di Controllo della pipeline

Criticità in una pipeline

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.

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).

Esempio di esecuzione

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

```
lw  $t1, 24($t2)
add $s0, $t1, $s2
beq $t1, $s2, 20
sw $t2, 36($t1)
sub $t0, $t1, $t2
or $s3, $t4, $t5
```

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

Data path che fluisce da sinistra a destra.

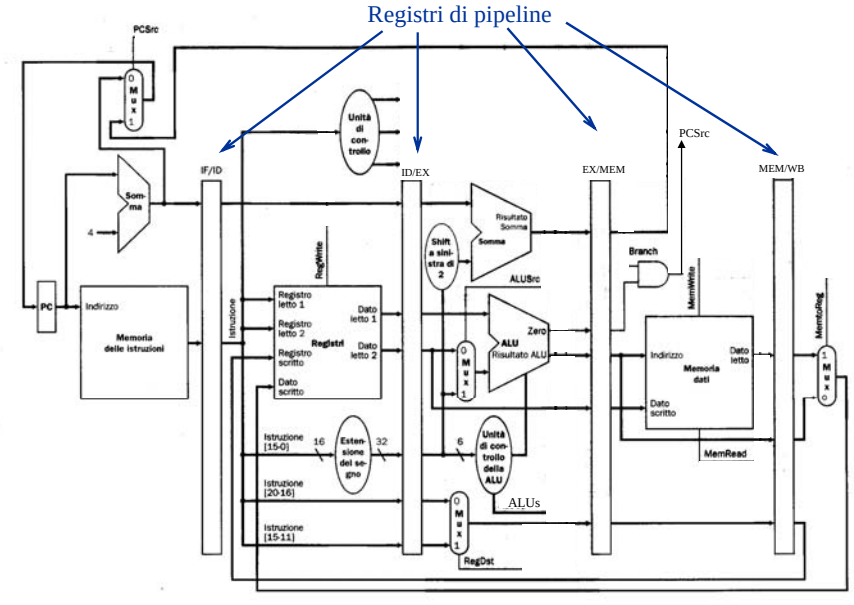
Sommario

La CPU con pipeline

L'Unità di Controllo della pipeline

Criticità in una pipeline

CPU con pipeline



La UC della CPU con pipeline

Definizione dei segnali di controllo per ogni stadio, per ogni istruzione.

Definizione dell'UC in grado di generare correttamente questi segnali.

Segnali di controllo su 1 bit

Nome segnale	Effetto quando è negato	Effetto quando è affermato
RegDst	Il numero del registro destinazione proviene dal campo rt (R2, bit 20-16)	Il numero del registro destinazione proviene dal campo rd (bit 15-11)
RegWrite	Nessuno	Nel registro specificato all'ingresso registro scritto del Register File, viene scritto il valore presente all'ingresso Dato Scritto
ALUSrc	Il secondo operando della ALU proviene dalla seconda uscita in lettura del Register File	Il secondo operando della ALU è la versione estesa (con segno) del campo offset
Branch	Il valore del PC viene sostituito dall'uscita del sommatore che calcola PC+4 (condizionato all'uscita di ALU)	Il valore del PC viene sostituito dall'uscita del sommatore che calcola la destinazione del salto (condizionato all'uscita di ALU)
MemRead	Nessuno	Il contenuto della cella di memoria dati indirizzata dal MAR è posto nel MDR
MemWrite	Nessuno	Il contenuto in ingresso al MDR, viene memorizzato nella cella il cui indirizzo è caricato nel MAR
MemtoReg	Il valore inviato all'ingresso Dato al Register File proviene dalla ALU	Il valore inviato all'ingresso Dato al Register File proviene dalla memoria

Scrittura PC e scrittura dei registri di pipeline ad ogni fronte di clock (ad ogni stadio).

Osservazioni

Il contenuto di *rt* ed il numero di scrittura nel Register File (*rd*) vengono portati attraverso i vari stadi.

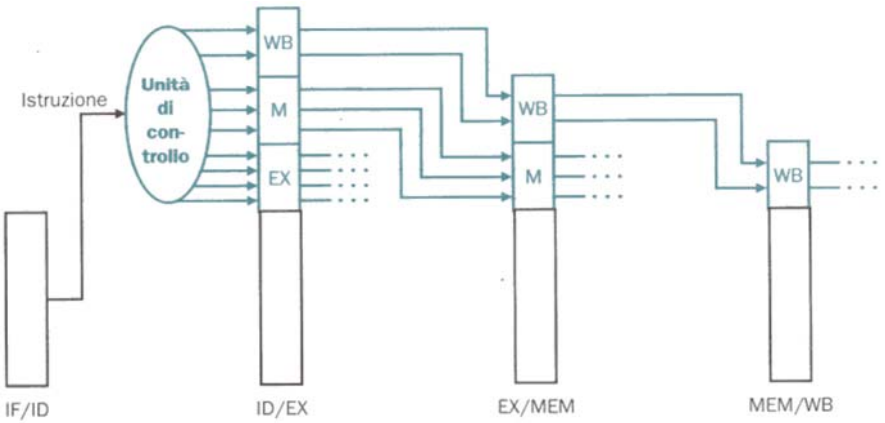
Nella fese di fetch e di decodifica non esistono segnali di controllo particolari.

I segnali di controllo particolari (legati alle diverse istruzioni) si possono così raggruppare:

	Exec				Memory			WB	
Istruzione	Reg Dst	ALUs1	ALUs0	ALU Src	Branch	Mem Read	Mem Write	Reg Write	Mem2 Reg
Format-R	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

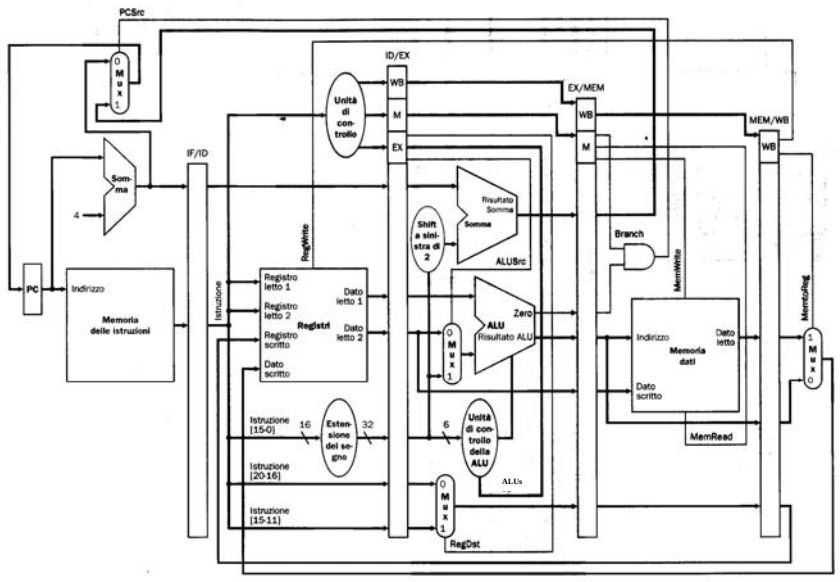
Generazione dei segnali di controllo

I segnali di controllo vengono generati nello stadio di decodifica e propagati.



Viene separata la fase di generazione dei segnali di controllo dalla fase di utilizzo.

UC per pipeline



Esempio di esecuzione

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

```
lw  $t1, 24($t2)
add $s0, $t1, $s2
beq $t1, $s2, 20
sw $t2, 36($t1)
sub $t0, $t1, $t2
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.

Sommario

La CPU con pipeline

L'Unità di Controllo della pipeline

Criticità in una 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.

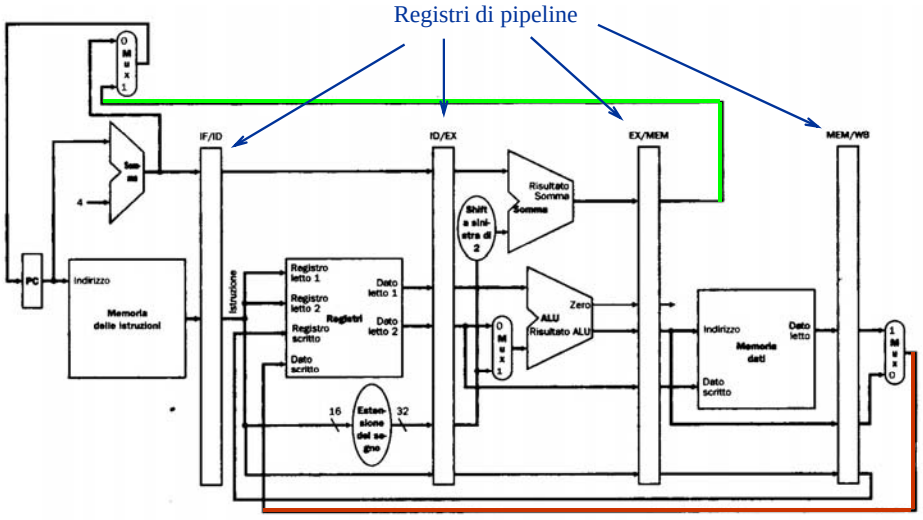
Triplicazione delle ALU

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

Esempio di Hazard sui dati

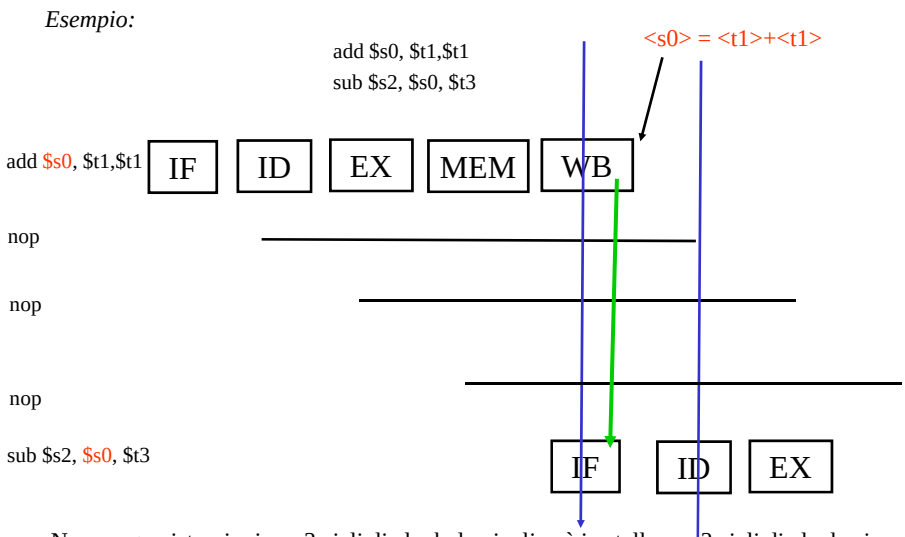
sub $s2, $s1, $s3	IF	ID	EX $1-$3	MEM	WB s->$2				
add $t2, $s2, $s5		IF	ID	EX $2 and $5	MEM	WB s->$t2			
or $t3, $s6, $s2			IF	ID	EX $6 or $2	MEM	WB (s->$t3)		
and $t4, $s2, $s2				IF	ID	EX $2 +$2	MEM	WB s->$t4	
sw $t5, 100($s2)					IF	ID	EX $2+100	MEM $t5 ->Mem	WB

CPU con pipeline



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
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.

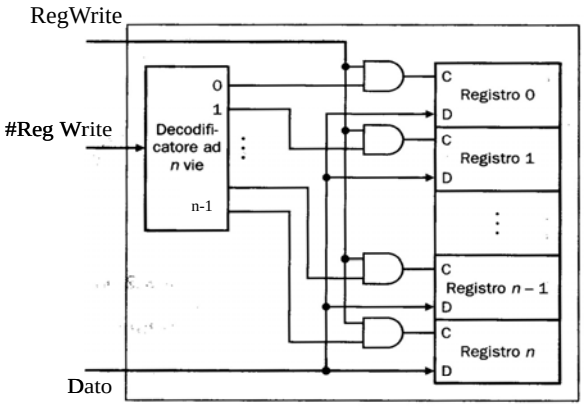
Hazard sui dati

sub $s2, $s1, $s3	IF	ID	EX $s1-$s3	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, $s2				IF	ID	EX $s2 +$2	MEM	WB s->$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

Hazard sui dati

sub $s2, $s1, $s3	IF	ID	EX $s1-$s3	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, $s2				IF	ID	EX $s2 +$s2	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.

Sommario

La CPU con pipeline

L'Unità di Controllo della pipeline

Criticità in una pipeline