


# Unità di controllo della pipeline

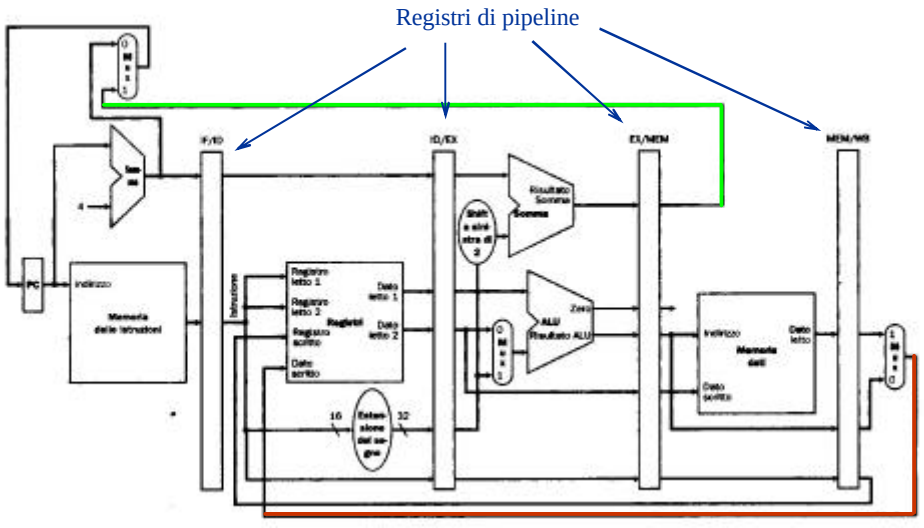
Prof. Alberto Borghese
Dipartimento di Scienze dell'Informazione
borghese@dsi.unimi.it

Università degli Studi di Milano




# Sommario

# 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).



# Sommario

La CPU con pipeline

L'Unità di Controllo della pipeline

Le criticità di una 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

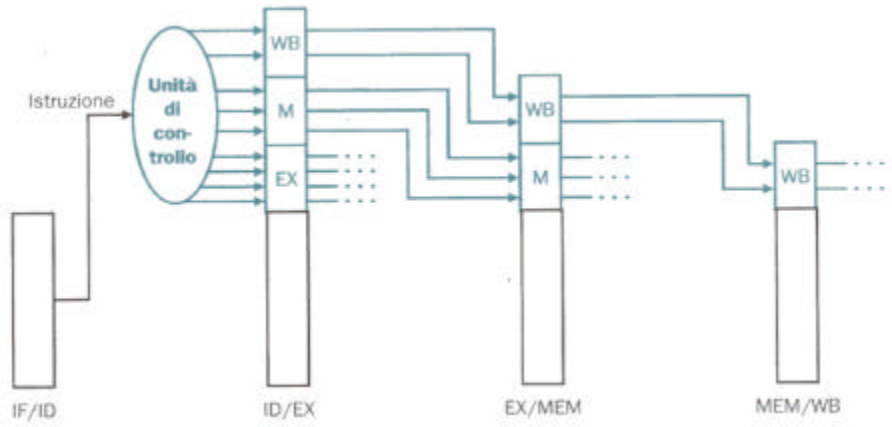
Il contenuto di RT ed il numero di RD vengono portati attraverso i vari stadi.
Nella fese di fetch e di decodifica non esistono segnali di controllo particolari.
I segnali di controllo particolari (legati alle diverse istruzioni) si possono così raggruppare:

| | Exec | | | | Memory | | | WB | |
|---|---|---|---|---|---|---|---|---|---|
| Istruzione | Reg Dst | ALU Op1 | ALU Op0 | 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 |

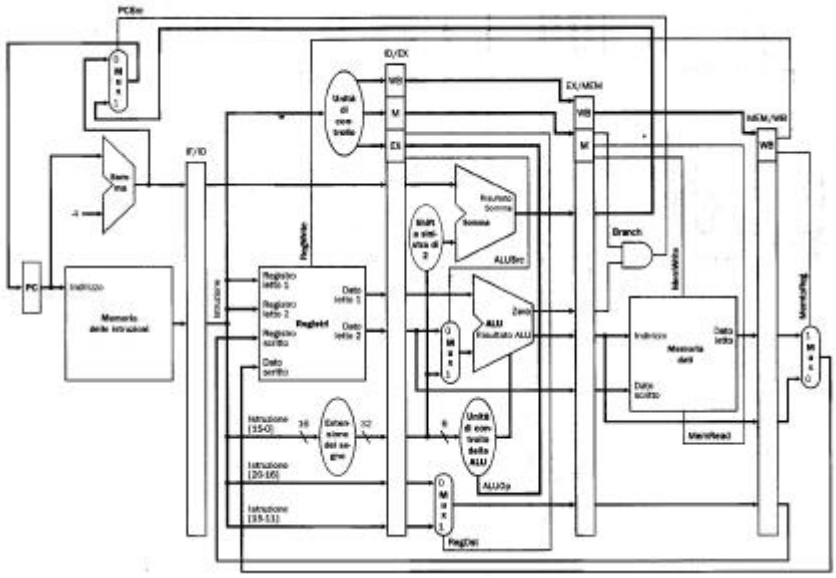


# Generazione dei segnali di controllo

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



# UC per pipeline





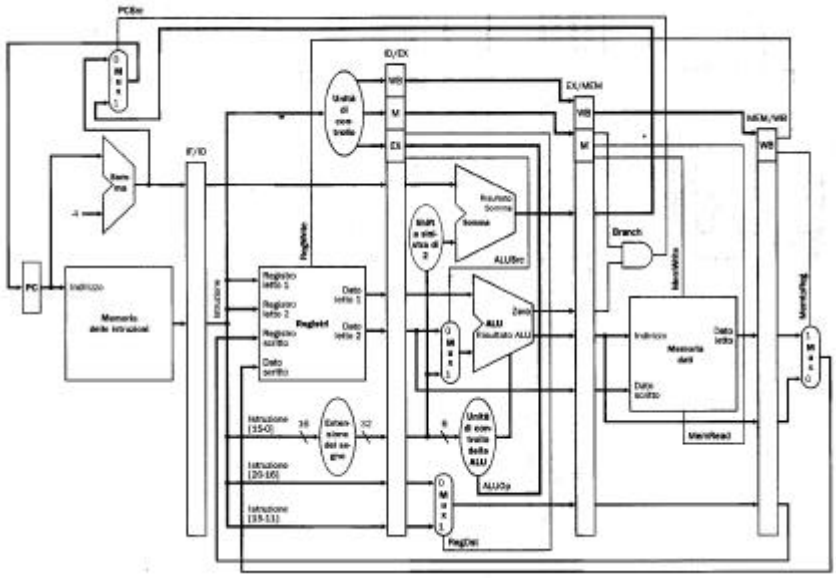

# Sommario

La CPU con pipeline

L'Unità di Controllo della pipeline.

La criticità nell'esecuzione della pipeline.

# 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
```

# Esempio di Hazard sui dati: lw / add

| | $t_0$ | $t_1$ | $t_2$ | $t_3$ | $t_4$ | $t_5$ | $t_6$ | $t_7$ |
|---|---|---|---|---|---|---|---|---|
| ….. | | | | | | | | |
| lw $t0, 8($s0) | FF (Mem, ALU) | DECOD (RF) | EXEC (ALU) | MEM (MEM) | WB (RF) | | | |
| add $t1, $t0, $s0 | | Buco (FF) | Buco (DEC) | Buco (EXEC) | Buco (MEM) | Buco (WB) | | |
| add $t1, $t0, $s0 | | | Buco | Buco | Buco | Buco | Buco | |
| add $t1, $t0, $s0 | | | | Buco | Buco | Buco | Buco | Buco |
| add $t1, $t0, $s0 | | | | | FF | DEC | EXEC | MEM |

I buchi (o bubble) inducano degli istanti di clock in cui non può essere eseguita l'istruzione successiva ➔ La pipeline va in stallo.



# Esempio di Hazard sui dati

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| sub $s2, $s1, $s3 | IF | ID | EX $1-$3 | MEM | WB s->$2 | | | | |
| and $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) | | |
| add $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 |

# Esempio di Hazard sul controllo

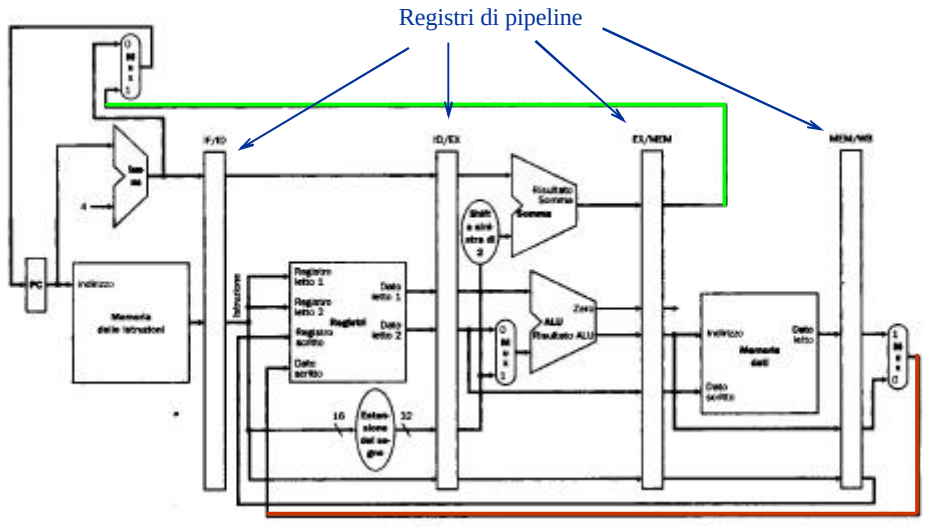
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| sub $s2, $s1, $s3 | IF | ID | EX<br>$1-$3 | MEM | WB<br>s->$2 | | | |
| beq $t2, $6, 24 | | IF | ID | EX<br>Zero if<br>($s2 == $s5) | MEM | WB | | |
| or $t7, $s6, $s7 | | | IF | ID | EX | MEM | WB | |
| add $t4, $s8, $s8 | | | | IF | ID | EX | MEM | WB |
| and $s5, $s6, $s7 | | | | | IF | ID | EX | MEM |
| add $t0, $t1, $t2 | | | | | | IF | ID | EX |

In caso di salto: dovrei avere disponibile all'istante in cui inizia l'esecuzione dell'istruzione or l'indirizzo dell'istruzione add e non eseguire la or, la add e la and.

NB L'indirizzo scritto nel PC corretto deve essere disponibile prima dell'inizio della fase di fetch. Ho 3 istruzioni sbagliate in pipeline.



# CPU con pipeline



# Soluzione delle criticità strutturali

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

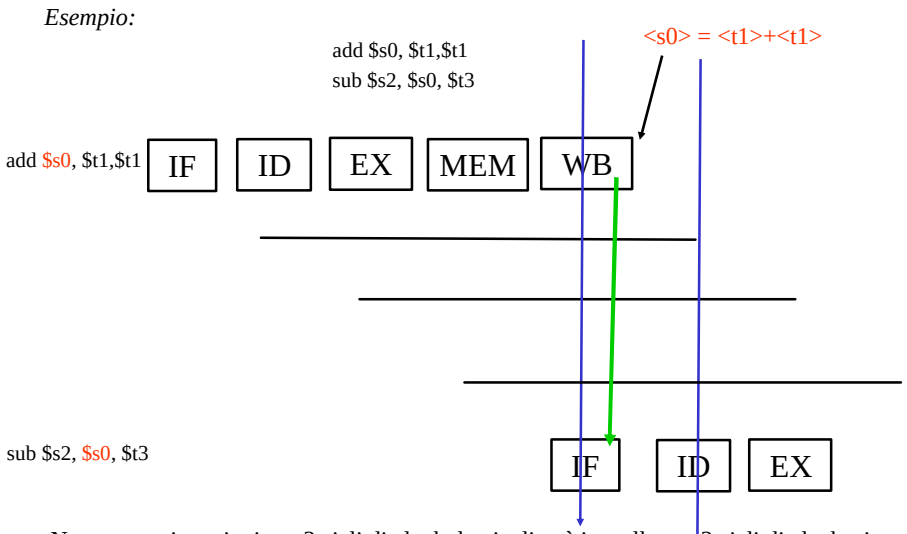
Triplicazione delle ALU

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



# 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.

# Sommario

La CPU con pipeline

L’Unità di Controllo della pipeline.

La criticità nell’esecuzione della pipeline.