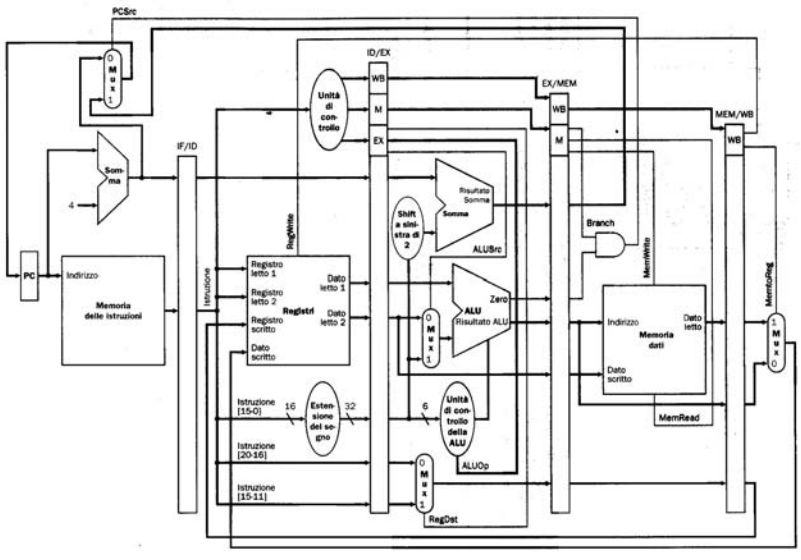
Pipeline

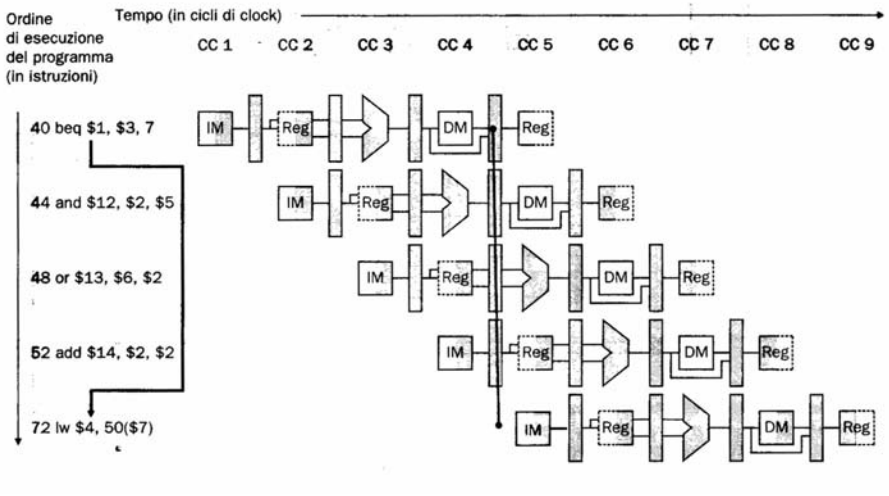
Architettura
degli Elaboratori e delle Reti,
Turno I

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

CPU con pipeline



L'istruzione lw dopo la branch è un hazard



Soluzioni alla criticità nel controllo

Riordinamento del codice (delayed branch).

Modifiche strutturali per l'anticipazione dei salti.

Ottimizzazioni.

Delayed branch

Decisione ritardata (ci si affida al compilatore)

• Aggiungere un "branch delay slot"

– l'istruzione successiva ad un salto condizionato viene sempre eseguita

– contiamo sul compilatore/assemblatore per mettere dopo l'istruzione di salto una istruzione che andrebbe comunque eseguita indipendentemente dal salto.

Esempio di delayed branch

```
        add $s4, $t0, $t1
        beq $s5, $s6, salto
        add $s0, $s0, $s1
salto:
        add $t5, $t4, $t3
        add $t6, $t7, $t7
```

```
        add $s4, $t0, $t1
        beq $s5, $s6, salto
        add $t5, $t4, $t3
        add $s0, $s0, $s1
salto:
        add $t6, $t7, $t7
```

L'istruzione add $t5, $t4, $t3 viene comunque eseguita, il salto (se richiesto) avviene dopo.

Modifica della CPU

Guessing (il salto non viene eseguito).

Modifiche architetturali per:

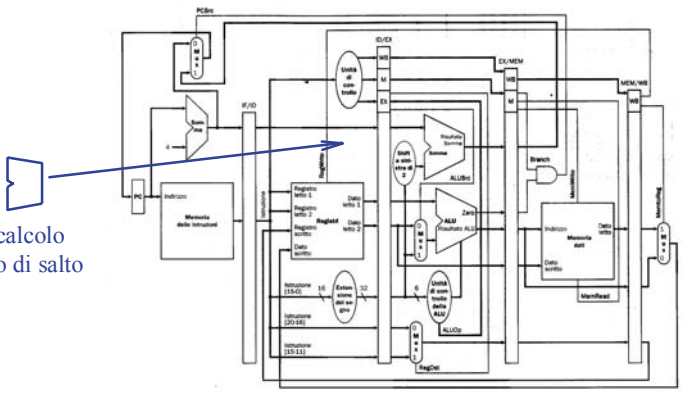
- possibilità di scartare delle istruzioni già in lavorazione.
- diminuire il numero di passi necessari per valutare la branch da 3 (fase MEM) a 1 (fase DEC).

Come scartare le istruzioni

Istruzione in fase di fetch: sovrascrivo IF / ID con l'istruzione nop (sll in MIPS).

Istruzione in esecuzione in un altro stadio: metto a 0 i segnali di controllo nel registro di pipeline di quello stadio.

Modifiche da apportare alla CPU



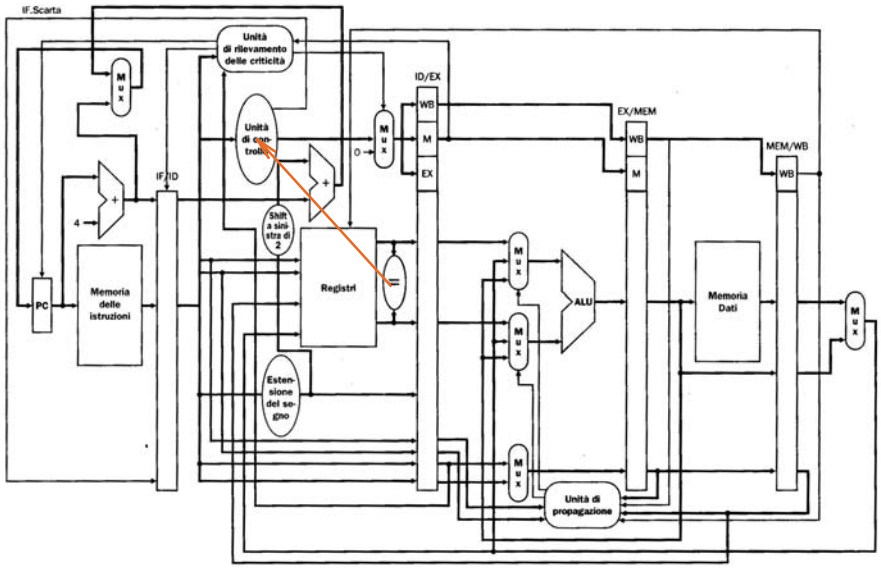
ALU calcolo indirizzo di salto

Anticipazione della valutazione della branch: Modifica della CPU nella gestione dei salti: anticipazione del calcolo dell'indirizzo di salto.

•HW addizionale: un comparatore all'uscita del Register File.

•Modifica dell'Unità di Controllo.

CPU con pipeline completa della gestione degli hazard.



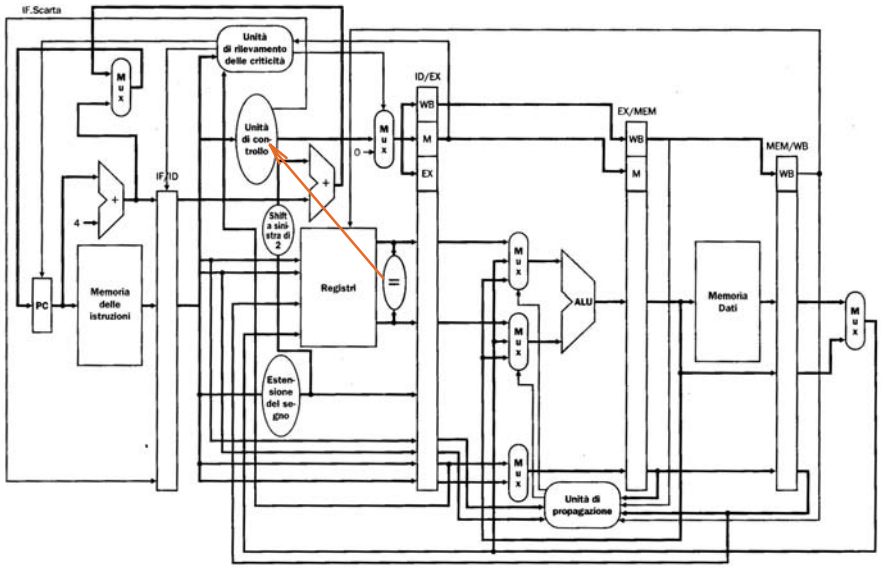
Come viene gestito il buco di uno stadio

Nello stadio ID noi modifichiamo il nostro indirizzo istruzione, ma l'istruzione successiva è già in fase di lettura.

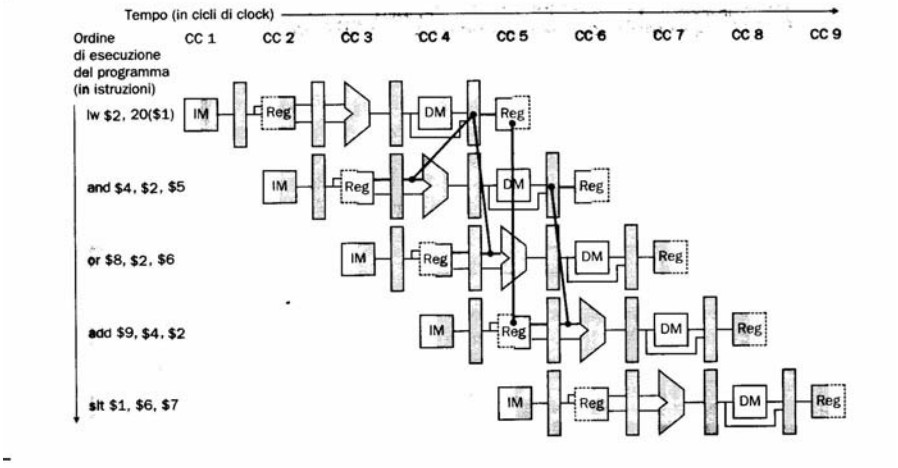
Ci sono due possibilità:

- Forziamo il nuovo indirizzo nel MAR della Instruction Cache (soluzione costosa in termini di tempo, si allunga il tempo di esecuzione dello stadio).
- Forziamo il nuovo indirizzo nello stadio master del PC (sarà disponibile per l'istruzione successiva). In questo caso, perdiamo un'istruzione.

CPU con pipeline completa della gestione degli hazard.

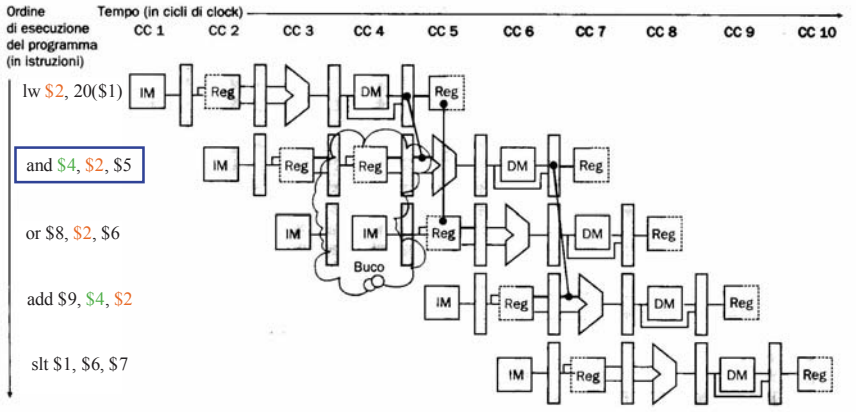


Hazard sui dati (istruzione lw)



- **Non può essere risolto con la tecnica precedente**
- **Serve una *unità di rilevamento degli azzardi***
- **Deve ritardare oppure entrare in stallo**

Stallo della pipeline

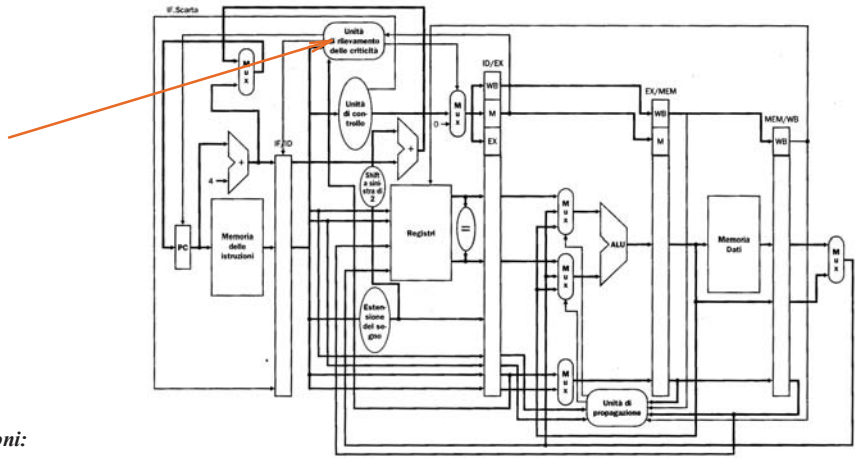


NB. L'hardware di *lettura / scrittura* del register file è congegnato in modo tale che la scrittura avvenga nel primo periodo del ciclo di clock e la lettura nel secondo periodo. Ad quando il clock è basso, e l'informazione viene trasferita agli slave della pipeline, può essere scritta in Register File (nello stadio WB). In questo stadio, la fase di decodifica non è attiva.

CPU con gestione dello stallo

Condizioni:

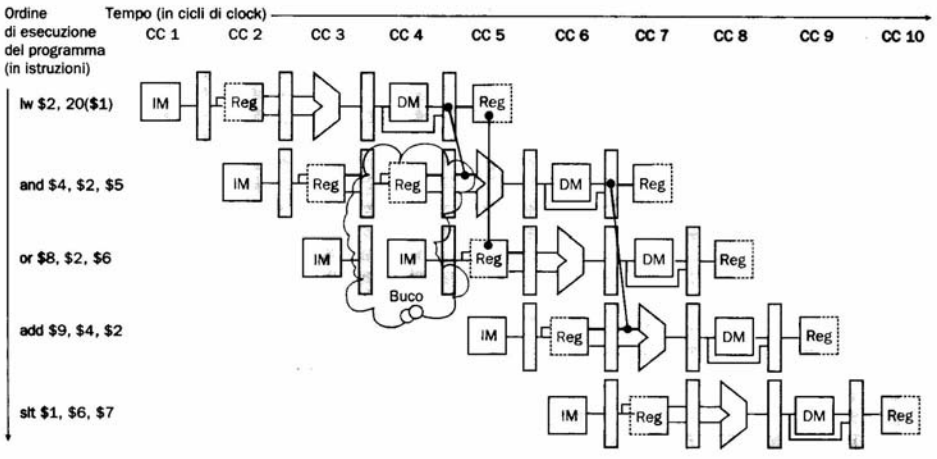
- OpCode = lw ➔ il segnale di controllo MemRead è attivo.
- Il registro target è uguale ad uno dei due registri operando dell'istruzione successiva.



Azioni:

- Annullare i segnali di controllo generati nella fase ID per l'esecuzione dell'istruzione (successiva alla lw).
- Ripetere la lettura e la decodifiche delle 2 istruzioni successive (ripetere la fase di fetch e decodifica).

Inserimento di un buco nella pipeline




TREND DI SVILUPPO DELLE PIPELINE

Ottimizzazione sulle criticità del controllo

Branch prediction buffer (4 kbyte nel Pentium 4).

Algoritmi di ottimizzazione dello scheduling per predizione ottima del salto.

Buffer di memoria con associato all'indirizzo di salto, l'informazione se debba essere eseguito o meno.

Pipeline più spinte

• Superpipelining: pipeline più lunghe (e.g. Digital DecAlpha, 5-7 stadi).

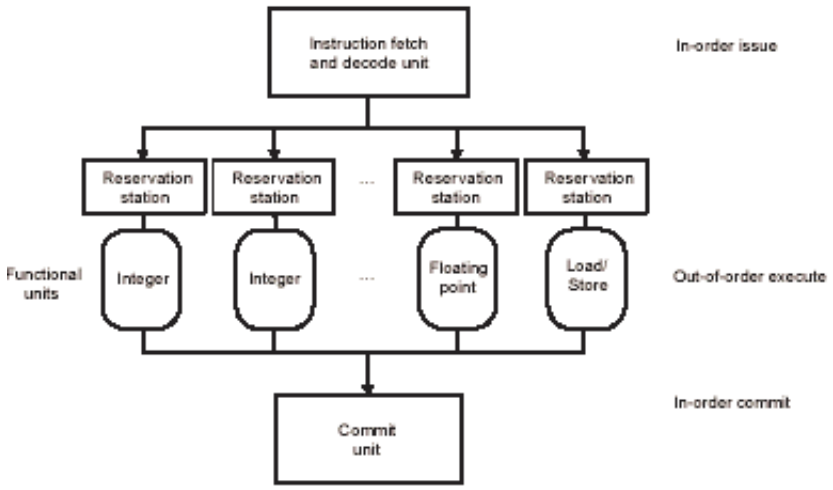
• Superscalar: più di una istruzione viene iniziata nello stesso ciclo (tutti i processori moderni).

E.g. i tempi di esecuzione della ALU del Pentium 4 sono la metà del periodo di clock e possono essere contemporaneamente in lavorazione (comprese le fasi di prenotazione) fino a 126 istruzioni contemporaneamente.

• Scheduling dinamico (solitamente combinato con architettura superscalare):

– l'hardware cerca di individuare le istruzioni pronte per essere eseguite
– è possibile l'esecuzione in ordine diverso da quello dato.
– è possibile avere un'unica istruzione per più dati (architettura SIMD: Single Instruction for Multiple Data).

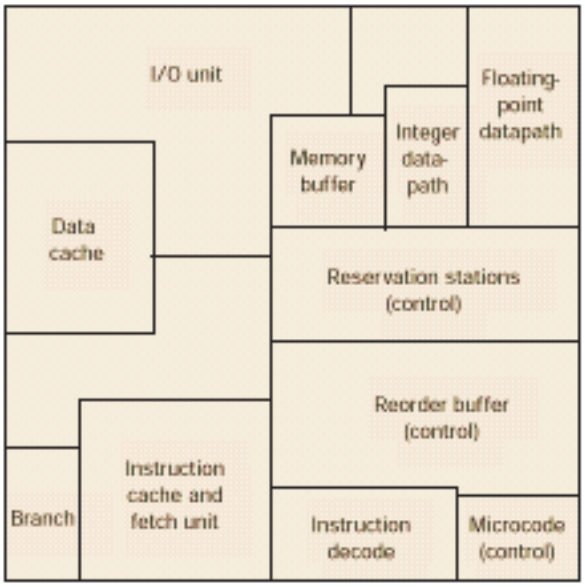
Pipeline con schedulazione dinamica



Esistono diversi cammini paralleli (per la fase di esecuzione e memoria) dell'istruzione.

Il processore Pentium Pro

Esecuzione "speculativa": scheduling dinamico + predizione dei salti (e.g. Intel 80x86 dal Pentium).



Pipeline ottimizzata

1) Viene nascosta la latenza della memoria (caricamento in cache di quanto servirà).

2) Evitare le situazioni di stallo che non possono essere risolte dal compilatore.

3) Riempire i buchi della pipeline.