High Performance and Quantum Computing

Anno Accademico
2025/26

Rocco Lo Russo
Agostino D'Amora

Contents

Introduzione

In questo documento vengono raccolti appunti del corso *High Performance and Quantum Computing* tenuto nell'anno 2025-26 dal professor Cilardo presso l'università di Napoli Federico II.

Chapter 1

Architetture superscalari

1.1 Richiami Pipelining

Il pipelining è una tecnica di implementazione di processori dove diverse istruzioni si sovrappongono durante l'esecuzione. Questa tecnica, chiave nella progettazione dei moderni processori, trae vantaggio dal parallelismo che esiste intrinsecamente tra le azione necessarie per l'esecuzione di un'istruzione. Nella pipeline, ogni passo necessario all'esecuzione di un'istruzione è svolto da un'unità funzionale autonoma, denominata *pipe segment* o *pipe stage*. Ogni unità è collegata alla successiva, e il throughput di una pipeline è determinato dal *tasso di attraversamento*, ovvero dalla frequenza con la quale un'istruzione esce dalla pipeline. Il tempo richiesto da un'istruzione per procedere allo stage successivo è denominato *processor cycle*, e dato che tutti gli stage devono necessariamente essere pronti a procedere allo stesso tempo, la lunghezza di un processor cycle è determinato dal tempo richiesto dallo stage più lento. In un computer il processor cycle è quasi sempre un colpo di clock.
Se il tempo di ogni stage è perfettamente bilanciato, chiamando T_i il tempo per ogni istruzione sul processore pipelined, T_{np} il tempo per ogni istruzione su un processore non pipelined ed n il numero di stages della pipe, allora vale la relazione: $T_i = \frac{T_{np}}{n}$.
Sotto queste condizioni ideali, lo speedup vale proprio n. Tuttavia, il tempo di ogni stage non è mai perfettamente bilanciato, e in più c'è da considerare il tempo di overhead introdotto dalla complessità del sistema. Una semplice pipline a cinque stadi presenta i seguenti blocchi funzionali:

- **Instruction fetch**: Viene letto il PC per prelevare l'indirizzo della prossima istruzione; Si accede al registro Instruction Memory e si carica l'istruzione; Si calcola il prossimo valore del PC (+4).
- **Instruction Decode**: L'istruzione viene decodificata, ovvero l'unità di controllo capisce di che tipo di istruzione si tratta; si leggono i valori dei registri sorgente; si verifica l'estensione del segno.
- **Execute**: In questa fase avviene la vera elaborazione.
  - *istruzione aritmetico-logica* $\rightarrow$ l'ALU esegue l'operazione;
  - *istruzione load/store* $\rightarrow$ si calcola l'indirizzo effettivo (base + offset);
  - *istruzione branch* $\rightarrow$ si calcola la condizione e il nuovo PC;

- **Memory Access**: Operazioni di lettura/scrittura in memoria;
- **Write Back**: Scrittura nei registri del processore. Non tutte le istruzioni scrivono un registro.

Da questa presentazione emergono tre osservazioni: innanzitutto, è necessario separare memorie di istruzione e memorie dati, attraverso l'implementazione di diverse caches. Questa soluzione permette di evitare conflitti durante le fasi di IF e MEM; Il register file è usato in due stages, ovvero in ID e WB. Quindi, è necessario in un solo colpo di clock performare due letture e una scrittura, e questo si ottiene effettuando la scrittura nella prima parte di ciclo di clock e le lettura nella seconda parte; infine, è necessario considerare un circuito hardware che incrementi automaticamente il PC nella fase di IF, e un circuito che calcoli un eventuale indirizzo di branch durante lo stadio ID. La condizione del salto viene valutata in fase EXE, con una parte della ALU che confronta due registri.
Per fare in modo che l'output di uno stage non interferisca con lo stage successivo, vengono introdotti dei registri tra uno stage e l'altro, in modo che alla fine di un ciclo di clock, tutti i risultati di uno stage sono conservati in un registro che viene usato con input allo stage successivo al prossimo ciclo di clock. I registri svolgono un ruolo chiave anche nel trasporto di eventuali risultati intermedi dove stage di origine e destinazione non sono immediatamente adiacenti.

Il pipelining incrementa il throughput di istruzioni del processore, ovvero il numero di istruzioni completate per unità di tempo, ma non riduce il tempo di esecuzione di una singola istruzione. Al più infatti il tempo di esecuzione di ogni singola istruzione aumenta a causa dell'overhead introdotto nel controllo della pipeline. Questo fatto pone un limite importante sulla profondità pratica di una pipeline. In particolare, lo sbilanciamento tra i tempi di attraversamento di ciascuno stage implica una riduzione di perfomance in quanto il clock può essere soltanto veloce quanto il tempo richiesto dallo stage più lento; inoltre, l'overhead introdotto dalla pipeline è causato dalla combinazione di fattori quali il ritardo dei registri della pipeline e il *clock skew*. Il tempo di setup dei registri della pipeline è il tempo necessario affinchè l'input che viene dato ad un registro si stabilizzi prima che il clock attivi una scrittura. Il clock skew è il massimo ritardo che segna la ricezione del colpo di clock tra qualsiasi coppia di registri.