Appunti Architettura

Brendon Mendicino

October 12, 2022

Contents

1 Introduzione

2 Pipeline

Per misurare le prestazioni di una pipeline si usa il **throughput**. Il throughput è il numero di sitruzioni che escono dalla pipeline in un intervallo di tempo.

Il datapath è composto da:

- instruction fetch: si prende dalla memoria la prossima istruzione putnatata dal PC e si incrementa quest'ultimo di 4;
- decode: si decodifica l'istruzione, attivando il datapath in modo adeguato, a prescindere dal tipo di operazione carico i due registri rs ed rt, ed il campo immediato, anche nel caso l'istruzione non fosse immediata, risparmio del tempo aumentando leggermente il comsumo di potenza;
  - A ¡- Reg[rs];
  - B ¡- Reg[rt];
  - Imm ¡- ($IR_{16...31}$);
- execution/effective address cycle:
  - memory reference: ALUoutput ¡- A + Imm;
  - register-register: ALUoutput ¡- A op B;
  - register-immediate: ALUoutput ¡- A op Imm;
  - brach: ALUoutput ¡- NPC + Imm; Cond ¡- (A op 0);
- memory access/branch completion cycle:
  - LMD ¡- Mem[ALUoutput]; or Mem[ALUoutput] ¡- B;
  - branch: if (cond) PC ¡- ALUoutput else PC ¡- NPC;
- write-back cycle:
  - register-register: Reg[$IR_{16...20}$] ¡- ALUoutput;
  - ...

L'assunzione molto forte sarà che tutti i dati e le istruzioni saranno sempre nelle memoria cache, quindi si avrà un delay di un solo colpo di clock. Il registre file potrà sia essere letto che scritto, ci sarà dunque bisogno di soddisfare queste richeste in un solo colpo di clock, la scrittura avviene nella prima parte del colpo di clock mentre la lettura avviene nelle seconda parte del colpo di clock.

Si aggiungono dei registri (detti pipeline register),

Aggiungere i registri della pipeline aggiunge un overhead, inoltre il clock del processore comporta un rallentamento, causato dallo skew.

3 Pipeline Hazards

Sono dei casi in cui l'istruzione non viene esguita in modo corretto:

- structural hazards: dipende dalla memoria,
- data hazards: dipende da come i dati vengono scritti e letti;
- control hazards: dependi dai branch;

Stall Un modo di gestire gli hazards è di manadre la CPU in stallo.

Risolvere gli hazard strutturali comporta un costo, in termini di nuovo hardware e di migliorare quello esistente. Un processore con hazard strutturali avrà un clock più veloce ma problemi di accesso alla memoria, un processore senza hazard strutturali avrà un clock più lento ma nessuna limitizione di accesso alla memoria.

Data hazards Generati dalle dipendenze dei dati generati all'interno della pipeline. Esempio:

```
1 add r1, r2, r3
2 sub r4, r1, r5
3 and r6, r1, r7
4 or  r8, r1, r9
5 xor r10, r1, r11
```

Il registro r1 viene inizializzato nella prima istruzione e poi utilizzato nel resto del codice, ma l'operazione di scrittura in si trova alla fine, infatti l'istruzione successiva (sub) dovrebbe aspettare che r1 sia scritto prima che possa essere letto, se tuttavia proviamo a leggere r1 il risultato sarà non deterministico.

Per risolvere questo problema si hanno due soluzioni:

- mandare in stallo il processore;
- implementiamo un forwarding che permette di non attendere la scrittura del registro, ma di leggere direttamente il valore dei registri della pipeline;