計算機組織與 RISC-V CPU 設計 · 第 12 章

處理器結構與功能: 管線

暫存器組織 / 指令週期細部 / 指令管線 / 管線危障 / 分支處理

摘要

本章進入處理器內部: 先檢視處理器的暫存器組織 (使用者可見暫存器與控制/狀態暫存器), 再把指令週期細化為子週期, 最後聚焦本章的重頭戲——指令管線 **(instruction pipelining)**: 像工廠裝配線一樣讓多條指令的不同階段重疊執行。我們將建立管線加速的定量模型, 並詳細分析限制管線效能的三類危障 **(hazard)**: 結構、資料、控制, 以及分支處理的各種對策。本章對應 Stallings 第 12 章, 是第 25 章管線 CPU 實作的理論基礎。

目錄

1 處理器組織 **2**

1.1 暫存器組織 2

2 指令週期細化 **2**

3 指令管線 **2**

3.1 裝配線思想 2

3.2 定量模型 3

4 管線危障 **3**

4.1 結構危障 (資源衝突) 3

4.2 資料危障 3

4.3 控制危障 (分支) 3

4.3.1 對策綜覽 3

5 本章重點回顧 **4**

6 複習問題 **4**

1 處理器組織

處理器必須: 提取指令、解譯指令、提取資料、處理資料、寫回資料。為此需要:ALU、控制單元、暫存器, 以及內部匯流排把它們連起來。

1.1 暫存器組織

- 使用者可見暫存器:
  - 通用暫存器 (RV32I:x0–x31, 其中 x0 恆為 0);
  - 資料/位址暫存器 (部分 ISA 分開;RISC-V 不分);
  - 條件碼 (旗標)——RISC-V 無。
- 控制與狀態暫存器:PC、IR、MAR、MBR; 程式狀態字 **PSW**(旗標、特權模式、中斷致能……)。RISC-V 對應物是 `mstatus` 等 CSR(第 21 章)。

設計議題: 暫存器要幾個?(RISC 的答案:32 個——太少則頻繁溢出到記憶體, 太多則編碼位元與脈絡切換成本上升); 要多寬?(至少能裝一個位址)。

2 指令週期細化

指令週期可分解為更細的微操作序列 (第 15 章), 此處先建立五步觀點——這正是經典五級管線的由來:

階段	名稱	工作 (以 **RV32I** 為例)
IF	指令提取	從指令記憶體讀 M[PC];PC←PC+4
ID	解碼/讀暫存器	解析欄位; 讀 rs1、rs2; 產生立即值
EX	執行	ALU 運算 / 位址計算 / 分支條件判定
MEM	記憶體存取	載入讀資料記憶體; 儲存寫資料記憶體
WB	寫回	結果寫入 rd

3 指令管線

3.1 裝配線思想

若每階段需 1 個時脈, 循序執行每條指令需 5 週期。管線讓五個階段的硬體同時服務五條不同的指令:

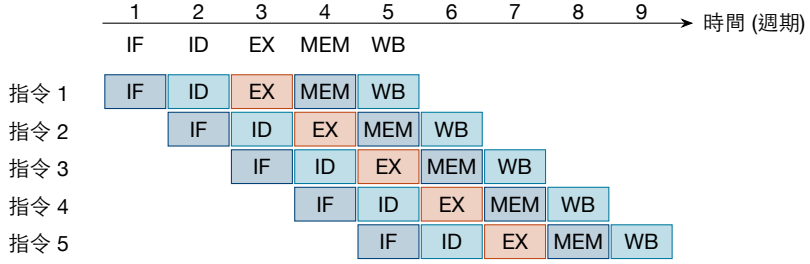

圖 1 五級管線: 第 5 週期起, 每週期完成一條指令

3.2 定量模型

k 級管線、每級 τ(含級間鎖存器延遲), 執行 n 條指令:

$$T_{pipe} = [k + (n-1)]\,\tau, \qquad \text{加速比} S = \frac{nk\tau}{[k+(n-1)]\tau} \xrightarrow{n\to\infty} k$$

理想情況下加速比趨近級數 k。但實際上:(1) 各級工作量不均,τ 由最慢級決定;(2) 級間鎖存器有固定開銷;(3) 危障造成停頓。級數也非愈多愈好——控制邏輯與鎖存開銷隨級數成長, 且分支代價變大。

4 管線危障

4.1 結構危障 (資源衝突)

兩條指令同週期爭用同一硬體。經典例:IF 要讀指令記憶體、MEM 要讀資料記憶體——若指令與資料共用單一記憶體埠就衝突。對策: 分離的指令/資料快取 (哈佛結構, 第 4 章)、多埠暫存器檔 (2 讀 1 寫)。

4.2 資料危障

後指令依賴前指令尚未寫回的結果 (RAW, 讀在寫後——管線中最主要者):

```
add x5, x6, x7    # x5 在第 5 週期 (WB) 才寫回
sub x8, x5, x9    # 但第 3 週期 (ID) 就要讀 x5!
```

程式 1 RAW 相依

對策 (細節與實作見第 25 章):

1. 停頓 **(stall)**: 插入氣泡等待——簡單但慢;
2. 前遞 **(forwarding/bypassing)**:EX 的輸出直接繞回 EX 的輸入, 多數 RAW 零代價;
3. 載入—使用危障:`lw` 的資料 MEM 級才有, 緊接著用的指令必須停 1 拍 (前遞也救不了), 或由編譯器排程填充。

4.3 控制危障 (分支)

分支結果在 EX 級才知道, 但下一拍就要提取下一條指令——提取誰? 猜錯就得沖銷 **(flush)** 已進入管線的指令。分支在程式中約占 15–25%, 是管線效能的頭號敵人。

4.3.1 對策綜覽

- 多重指令流: 兩路都抓 (硬體加倍, 合流再選);
- 提前判定: 把分支比較移到 ID 級, 縮短代價;
- 延遲分支: 分支後固定執行 1 條 (delay slot;MIPS 採用,**RISC-V** 拒絕——它把微結構細節漏進 ISA);
- 靜態預測: 一律猜不跳/一律猜跳/向後跳猜跳 (迴圈友善);

- 動態預測: 分支歷史表 (BHT) 以 2 位元飽和計數器記錄近期行為; 分支目標緩衝器 (BTB) 記住目標位址,IF 級即可改向。現代預測器 (TAGE、感知器) 正確率 >97%。

2 位元飽和計數器

狀態: 強不跳 00 ↔ 弱不跳 01 ↔ 弱跳 10 ↔ 強跳 11; 每次實際結果向對應方向推一格, 預測看最高位。比 1 位元預測器好在: 迴圈最後一次的「例外」不會立刻推翻整體傾向, 典型迴圈每輪只誤預測一次。

重點觀念

效能公式:CPI = 1 + 每指令平均停頓週期。管線不減少單條指令的延遲 (反而略增), 它提升的是吞吐量。危障處理的所有技巧, 都是在把「平均停頓」壓向零。

5 本章重點回顧

本章要點

- 五級管線 IF-ID-EX-MEM-WB; 理想加速比 = 級數, 實際受不均勻分級、鎖存開銷與危障限制。
- 結構危障: 資源不夠 → 分離 I/D 快取、多埠暫存器檔。
- 資料危障 (RAW): 前遞解決大多數; 載入—使用必須停 1 拍。
- 控制危障: 靜態/動態分支預測;BHT(方向)+BTB(目標);RISC-V 無延遲槽。
- CPI = 1 + 平均停頓; 管線提升吞吐量而非單指令延遲。

6 複習問題

1. 4 級管線、每級 1 ns、執行 1000 條指令: 求管線與非管線時間及加速比。
2. 為何管線級數不是愈多愈好? 列出兩個原因。
3. 指出下列程式的危障類型與位置, 並說明前遞能否消除:`lw x5,0(x6); add x7,x5,x8; sub x9,x7,x5`。
4. 分支占 20%、猜錯率 10%、猜錯代價 2 週期: 求 CPI。
5. 2 位元預測器對「跳 9 次、不跳 1 次」循環的長期誤預測率是多少?1 位元預測器呢?
6. RISC-V 為何不採延遲分支? 從 ISA 長壽性論述。

參考資料

[1] W. Stallings, *Computer Organization and Architecture*, 6th ed., Chapter 12.

[2] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design, RISC-V Edition*, 2nd ed., Morgan Kaufmann, 2021, Ch. 4.