計算機組織與 RISC-V CPU 設計 · 第 3 章

計算機功能與匯流排互連

指令週期 / 中斷 / 互連結構 / 匯流排設計

摘要

本章從頂層視角檢視計算機的功能與互連。首先詳述指令週期: 提取 (fetch) 與執行 (execute) 兩大階段, 以及中斷機制如何讓處理器與緩慢的 I/O 裝置有效並行。接著討論三大模組 (CPU、記憶體、I/O) 之間必須傳遞的資訊類型, 以及承載這些傳輸的匯流排互連: 資料線、位址線、控制線的功能, 匯流排階層的必要性, 以及仲裁、時序等關鍵設計議題。本章對應 Stallings 第 3 章; 所介紹的指令週期概念是第 22 章起 CPU 實作的直接基礎。

目錄

1 計算機的部件與 von Neumann 設計

von Neumann 結構基於三個關鍵概念:

1. 資料與指令儲存於單一讀寫記憶體;
2. 記憶體內容依位置定址, 不管其中儲存的資料型態;
3. 執行以循序方式 (除非明確更改) 從一條指令進行到下一條。

為什麼要「儲存程式」? 若用硬體直接實現某個計算, 接線本身就是程式 (hardwired program), 改功能就得重新接線。反之, 若提供一組通用的算術與邏輯功能部件, 再以控制訊號指揮它們, 那麼「程式」就只是一串控制碼; 新的計算只需提供新的控制碼序列, 硬體完全不動。指令解譯器負責把每條指令轉成該指令對應的控制訊號。

2 指令週期

2.1 基本兩步驟: 提取與執行

程式執行的最小單位是指令週期 **(instruction cycle)**, 最簡形式含兩步:

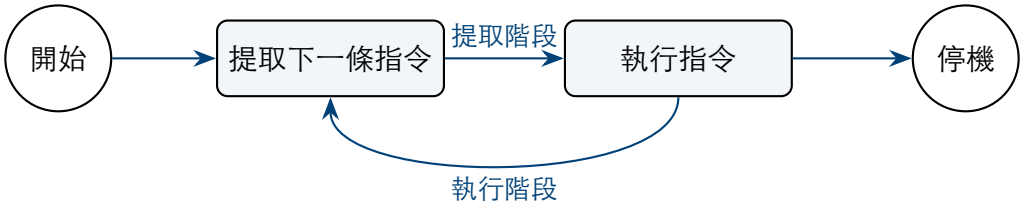

圖 1 基本指令週期

- 提取: 處理器把程式計數器 **(PC)** 所指的指令從記憶體讀入指令暫存器 **(IR)**, 然後 PC 遞增 (在 RV32I 中為 PC+4), 使下次提取下一條指令。
- 執行: 處理器解譯 IR 中的指令並執行所要求的動作。動作可分四類:
  1. 處理器–記憶體: 資料在兩者間傳輸 (載入/儲存);
  2. 處理器–**I/O**: 資料在處理器與 I/O 模組之間傳輸;
  3. 資料處理: 對資料執行算術或邏輯運算;
  4. 控制: 更改執行順序 (跳躍/分支)。

一部假想機器的加法程式

考慮 Stallings 書中的假想機 (16 位元指令:4 位元 opcode + 12 位元位址;opcode 1= 載入 AC、2= 儲存 AC、5=AC 加上記憶體)。程式片段「把位址 940 的內容加到位址 941 的內容, 結果存回 941」需要三個指令週期:

1. PC=300: 提取指令 `1940`(載入), 執行後 AC ← M[940] = 3;
2. PC=301: 提取指令 `5941`(相加), 執行後 AC ← 3+2 = 5;
3. PC=302: 提取指令 `2941`(儲存), 執行後 M[941] ← 5。

同樣的工作在 RV32I 上是 `lw`/`lw`/`add`/`sw` 四條指令——因為 RISC 把「從記憶體取數並相加」拆成獨立的載入與運算 (第 13、18 章)。

3 中斷

3.1 為什麼需要中斷

大多數 I/O 裝置遠慢於處理器。若處理器以程式化 **I/O** 方式寫一筆資料到印表機後空轉等待, 將浪費數以百萬計的指令週期。中斷 **(interrupt)** 機制讓 I/O 模組在準備好接受或提供資料時主動通知處理器; 處理器發出 I/O 命令後即可繼續執行其他指令, 收到中斷再回頭處理。

中斷的類型包括: 程式中斷 (除以零、非法指令、記憶體越界等例外)、計時器中斷 (作業系統排程的基礎)、**I/O** 中斷 (完成或錯誤通知)、硬體失效中斷 (電源、同位錯誤)。

3.2 中斷與指令週期

加入中斷後, 指令週期多了中斷階段: 每條指令執行完畢後, 處理器檢查是否有未處理的中斷; 若有, 則:

1. 暫停目前程式的執行, 保存其脈絡 (至少包括 PC 與狀態暫存器);
2. 將 PC 設為中斷處理常式 **(interrupt handler)** 的起始位址;
3. 執行處理常式; 結束時恢復被中斷程式的脈絡, 從中斷點繼續。

重點觀念

中斷檢查發生在指令邊界, 因此單條指令的執行是不可分割的; 這保證了脈絡保存與恢復的一致性。RISC-V 的機器模式以 `mepc`、`mcause`、`mtvec` 等 CSR 實現同樣的機制 (第 21 章)。

3.3 多重中斷

處理多重中斷有兩種策略:

- 停用中斷 **(sequential)**: 處理中斷時停用其他中斷, 新中斷保持懸置 (pending), 處理完再依序處理。簡單, 但無法反映緊急程度。
- 優先權巢狀 **(nested)**: 高優先權中斷可以打斷低優先權中斷的處理常式, 形成巢狀。例如通訊線路 (資料稍縱即逝) 優先於印表機。

4 互連結構

計算機由 CPU、記憶體、I/O 三類模組組成, 互連結構必須支援以下傳輸:

傳輸類型	說明
記憶體 → 處理器	讀取指令或資料
處理器 → 記憶體	寫入資料
I/O → 處理器	讀取 I/O 裝置資料/狀態
處理器 → I/O	送出資料/命令給 I/O 裝置
I/O ↔ 記憶體	DMA:I/O 模組直接與記憶體交換資料, 不經過處理器

5 匯流排互連

5.1 匯流排結構

匯流排 **(bus)** 是連接兩個以上裝置的共享通訊路徑, 關鍵特徵是廣播: 任一裝置送出的訊號可被所有連接裝置看到; 但同一時間只能有一個裝置成功發送。系統匯流排通常由 50–100 條以上的線組成, 功能上分三群:

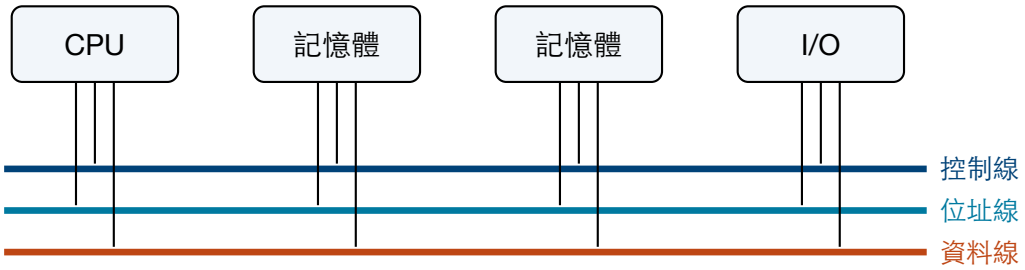

圖 2 匯流排互連架構

- 資料線 **(data bus)**: 搬移資料的路徑, 寬度 (32、64、128 位元) 是決定系統效能的關鍵因素——它決定每次能傳輸多少位元。
- 位址線 **(address bus)**: 指定資料的來源或目的地; 寬度決定系統最大記憶體容量 (RV32 為 32 位元位址, 可定址 4 GiB 空間)。高位位址也常用來選擇模組、低位選擇模組內位置。
- 控制線 **(control bus)**: 控制對資料線與位址線的存取與使用。典型控制訊號: 記憶體讀/寫、I/O 讀/寫、傳輸確認 ACK、匯流排請求/允許 (bus request/grant)、中斷請求/確認、時脈、重置。

5.2 多匯流排階層

若把過多裝置掛在單一匯流排上:(1) 匯流排愈長, 傳播延遲愈大, 協調控制的時間愈長;(2) 總資料需求逼近匯流排容量時形成瓶頸。因此現代系統採階層式多匯流排:

- 本地匯流排: 處理器 ↔ 快取;
- 系統匯流排: 快取 ↔ 主記憶體;
- 高速匯流排: 掛載高速 I/O(圖形、網路、SCSI/NVMe), 經橋接器 (bridge) 連上系統匯流排;
- 擴充匯流排: 掛載較慢的傳統 I/O。

快取結構把記憶體流量與 I/O 流量隔離; 高速匯流排讓高需求裝置更貼近處理器, 又與處理器時脈解耦。

5.3 匯流排設計要素

設計面向	選項與說明
型式	專用 (功能專用/實體專用)vs. 多工 (位址與資料分時共用同一組線, 省接腳但較慢)
仲裁	集中式 (單一匯流排控制器/仲裁器)vs. 分散式 (各模組共同參與決定)
時序	同步 (以時脈定界, 簡單但遷就最慢裝置)vs. 非同步 (以事件互鎖, 彈性大)
寬度	位址寬度決定定址範圍; 資料寬度決定單次傳輸量
傳輸類型	讀、寫、讀–修改–寫、區塊傳輸 (一個位址 + 連續多筆資料)

同步讀取時序

同步匯流排的讀取:T1 週期, 主控端把位址放上位址線並發出讀取訊號;T2, 受控端 (記憶體) 解碼位址、取出資料放上資料線;T3, 主控端讀入資料並撤銷訊號。若受控端來不及, 可插入等待週期 **(wait state)**。你在第 26 章的 testbench 中觀察 CPU 與記憶體的互動時, 將看到一模一樣的節奏。

5.4 從 PCI 到現代互連

Stallings 書中詳述的 **PCI**(Peripheral Component Interconnect,1992) 是當年的主流:32/64 位元多工匯流排、33/66 MHz、集中仲裁、支援區塊傳輸。今日 PCI 已演進為點對點串列的 PCI Express(PCIe): 不再是共享平行匯流排, 而是以多條 lane 組成的交換式網路, 第 5 代單 lane 可達 32 GT/s。晶片內部的互連也走向 AMBA AXI、TileLink(RISC-V 生態常用) 等匯流排協定——但位址/資料/控制三分、仲裁、交握等基本概念與本章所述一脈相承。

6 本章重點回顧

本章要點

- 指令週期 = 提取 + 執行 (+ 中斷檢查);PC/IR 是週期運作的核心暫存器。
- 中斷讓處理器與慢速 I/O 並行, 大幅提升效率; 中斷檢查發生於指令邊界。
- 匯流排 = 資料線 + 位址線 + 控制線; 資料線寬度決定吞吐、位址線寬度決定定址範圍。
- 單一匯流排會成為瓶頸, 現代系統採用多層匯流排階層與橋接器。
- 匯流排設計要素: 專用/多工、集中/分散仲裁、同步/非同步時序、傳輸類型。

7 複習問題

1. 指令週期的兩大階段是什麼? 各自完成哪些工作?
2. 指令執行的動作可分哪四類? 各舉一條 RV32I 指令為例。
3. 為什麼中斷能提升處理器使用效率? 畫出含中斷階段的指令週期狀態圖。
4. 兩種多重中斷處理策略各有何優缺點?

5. 匯流排的三類訊號線各承擔什麼功能? 其寬度各影響系統的哪個面向?
6. 為什麼需要多匯流排階層? 橋接器扮演什麼角色?
7. 同步與非同步時序各適合什麼場景?
8. 一同步匯流排時脈 100 MHz、資料線 64 位元, 區塊傳輸每週期一筆, 求峰值頻寬 (MB/s)。

參考資料

[1] W. Stallings, *Computer Organization and Architecture*, 6th ed., Chapter 3.

[2] PCI-SIG, *PCI Express Base Specification*. https://pcisig.com

[3] SiFive, *TileLink Specification*. https://www.sifive.com