計算機組織與 RISC-V CPU 設計 · 第 15 章

控制單元與微程式控制

微操作 / 控制訊號 / 硬佈線實現 / 微程式控制

摘要

控制單元是處理器的「指揮官」: 它讓資料路徑上的每個部件在正確的時間做正確的事。本章先把指令週期分解到最底層的微操作 **(micro-operations)**, 說明控制單元的輸入 (IR、旗標、時脈) 與輸出 (控制訊號); 然後介紹兩種實現方式: 硬佈線 **(hardwired)**——控制邏輯是組合電路, 以及微程式 **(microprogrammed)**——控制邏輯是存放在控制記憶體中的「微指令」程式。本章整合 Stallings 第 15、16 章, 是第 23 章控制單元 Verilog 實作的直接理論來源。

目錄

1 微操作

指令週期 (提取、間接、執行、中斷) 可分解為一系列微操作: 暫存器間的一次傳輸、一次 ALU 運算、一次記憶體讀寫。每個微操作在一個時脈週期 (或子週期) 內完成。

1.1 提取週期的微操作

時間	微操作
t_1	MAR $\leftarrow$ PC
t_2	MBR $\leftarrow$ M[MAR]; PC $\leftarrow$ PC + 4
t_3	IR $\leftarrow$ MBR

分組規則:(1) 事件順序不能亂 (位址要先進 MAR 才能讀);(2) 同一時間不能衝突 (不能同時讀寫同一暫存器);(3) 每個微操作只占一個簡單動作。

1.2 執行週期

每種 opcode 有自己的微操作序列。例如 `ADD R1, X`(R1 $\leftarrow$ R1 + M[X]):

t_1	MAR $\leftarrow$ IR(位址欄位)
t_2	MBR $\leftarrow$ M[MAR]
t_3	R1 $\leftarrow$ R1 + MBR

RV32I 的 `add rd,rs1,rs2` 在單週期實作中即「讀 rs1/rs2 → ALU 相加 → 寫 rd」一氣呵成; 多週期實作則類似上表分拍進行。

2 控制單元的功能

- 定序 **(sequencing)**: 決定下一步執行哪一組微操作 (提取? 執行哪條指令的第幾拍?);
- 執行 **(execution)**: 發出控制訊號讓該組微操作發生。

輸入: 時脈、IR(opcode 決定做什麼)、旗標 (條件分支結果)、外部控制匯流排訊號 (中斷請求等)。輸出:CPU 內部控制訊號 (暫存器間傳輸門、ALU 功能選擇)、對控制匯流排的訊號 (記憶體讀/寫、中斷確認)。

重點觀念

「控制訊號」在 Verilog 實作中就是多工器選擇線與寫致能:`RegWrite`、`ALUSrc`、`MemRead`、`MemWrite`、`Branch`、`ALUOp`……第 23 章的主控制器真值表, 就是本節抽象概念的具體化。

3 硬佈線實現

控制單元 = 組合邏輯: 輸入 (解碼後的 opcode、時序計數器、旗標) 直接以邏輯閘網路產生每個控制訊號。

$$C_i = f(\text{opcode 解碼線}, T_k, \text{旗標})$$

例如某訊號 $C_5 = T_2 \cdot (\text{LW} + \text{SW}) + T_3 \cdot \text{ADD} \cdots$。

優點: 快 (純組合延遲)、面積可最佳化。缺點: 設計複雜、改指令集要重新設計電路。RISC 指令集簡單規律, 幾乎全部採硬佈線——RV32I 的控制器不過幾十行 Verilog(第 23 章)。

4 微程式控制

4.1 核心思想

Wilkes(1951) 的洞見: 控制訊號的產生本身就像「執行程式」。把每個時拍要發出的控制訊號組合寫成一個控制字 **(**微指令**)**, 存入控制記憶體; 控制單元變成一部微型計算機:

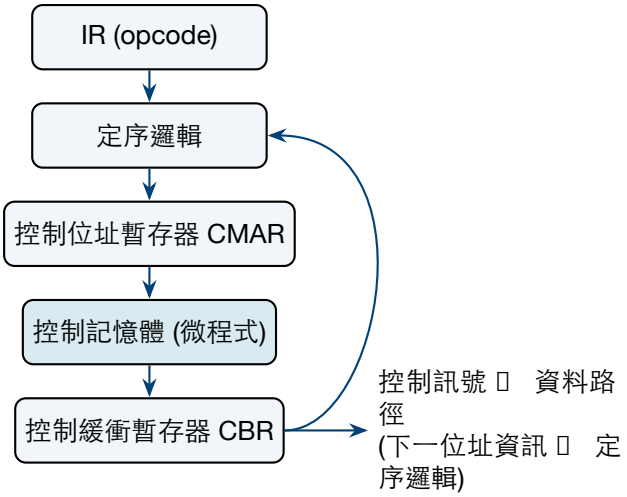

圖 1 微程式控制單元的組織

每個機器指令對應一段微常式: 提取微常式共用;opcode 經映射得到該指令微常式的入口; 微指令逐拍發出訊號, 結尾跳回提取。

4.2 水平 vs. 垂直微指令

- 水平微指令: 每個控制訊號占一位——寬 (可能上百位), 完全平行, 無解碼延遲;
- 垂直微指令: 訊號編碼壓縮——窄, 省控制記憶體, 但需解碼且犧牲部分平行性。

4.3 評價與歷史地位

優點: 設計系統化、易改易擴充 (改微碼即改指令集)、同一硬體可模擬多種指令集;CISC 時代 (IBM S/360 家族、VAX) 的基石——同一結構、不同價位的機型, 就是以不同速度的控制記憶體 + 微碼實現的。缺點: 比硬佈線慢 (每拍多一次控制記憶體讀取)。

現代處理器的折衷: 常用簡單指令走硬佈線快速路徑, 罕用複雜指令 (x86 的字串操作、微碼修補) 落入微碼 **ROM**——兩種技術共存至今。微碼還帶來可修補性:CPU 出廠後仍能以微碼更新修正錯誤。

5 本章重點回顧

本章要點

- 指令週期 → 微操作序列; 控制單元負責定序與發訊號。
- 輸入:IR、旗標、時脈、外部訊號; 輸出: 內部控制訊號 + 控制匯流排訊號。
- 硬佈線 = 組合邏輯, 快而不易改;RISC 的選擇。
- 微程式 = 控制記憶體中的微指令, 系統化易擴充;CISC 的基石; 現代兩者混用。

6 複習問題

1. 寫出提取週期的微操作序列, 並說明分組的三條規則。
2. 為 RV32I 的 `lw rd, imm(rs1)` 寫出多週期微操作序列 (IF/ID/EX/MEM/WB 五拍)。
3. 硬佈線與微程式控制的取捨為何? 為什麼 RISC 幾乎都用硬佈線?
4. 水平與垂直微指令的差異?
5. 微碼如何支撐「同一結構、多種機型」的商業模式? 現代 x86 為何仍保留微碼?

參考資料

[1] W. Stallings, *Computer Organization and Architecture*, 6th ed., Chapters 15–16.

[2] M. V. Wilkes, “The Best Way to Design an Automatic Calculating Machine,” Manchester, 1951.