計算機組織與 RISC-V CPU 設計 · 第 17 章

RISC-V 架構總覽與設計哲學

開放 ISA / 模組化設計 / 暫存器模型 / 生態系

摘要

本章開啟教材第二部:RISC-V 指令集架構。我們先回答「RISC-V 是什麼、為什麼重要」: 一個由 UC Berkeley 發源、由 RISC-V International 治理的開放、免授權費指令集標準; 剖析它的模組化設計——小而穩定的基礎整數指令集 (RV32I/RV64I) 加上可選擴展 (M/A/F/D/C/V/Zicsr...); 建立 RV32I 的程式設計師模型:32 個通用暫存器、PC、無旗標; 並說明命名規則 (RV32IMAC 之類的字串怎麼讀)、hart 概念與生態系現況。本章是第 18–21 章的導覽。

目錄

1 RISC-V 的緣起與治理

- **2010**:UC Berkeley Par Lab 需要一個可自由使用、修改、發表的 ISA 供研究與教學,Krste Asanović、Andrew Waterman、Yunsup Lee 與 David Patterson 啟動第五代 Berkeley RISC 計畫——RISC-V(“five”)。
- **2011**: 公開第一版使用者層 ISA;**2014**: 凍結 RV32I/RV64I 基礎集——基礎集永不再變, 軟體投資永遠有效。
- **2015**: 成立 RISC-V Foundation(2020 年遷冊瑞士、更名 RISC-V International), 以會員制治理規格的批准 (ratification)。
- 任何公司、學校、個人都可以免費實作、販售 RISC-V 處理器, 無需授權合約——這與 x86(封閉)、Arm(授權費 + 權利金) 形成根本差異。

重點觀念

「指令集應該像 TCP/IP 一樣是開放標準」——RISC-V 的核心主張。開放的是規格; 具體的處理器實作 (微結構) 可以開源 (如 Rocket、BOOM、PicoRV32、VexRiscv) 也可以商業閉源 (SiFive、平頭哥、Andes 的核)。結構與組織的分離 (第 1 章) 在此獲得制度性的體現。

2 模組化: 小基礎 + 選配擴展

x86/Arm 的指令集隨年代不斷膨脹, 所有實作都得背負全部歷史。RISC-V 反其道:

2.1 基礎整數指令集

基礎	XLEN	說明
RV32I	32	32 位元基礎整數指令集,**40** 條指令 (本教材主角)
RV32E	32	嵌入式精簡版: 僅 16 個暫存器
RV64I	64	64 位元基礎 (暫存器 64 位元, 增加 W 尾碼字運算)
RV128I	128	為未來保留 (草案)

RV32I 小到什麼程度?**40** 條指令、六種格式、無旗標、無條件碼——一名學生可以在一學期內以 Verilog 完整實作 (第三部將證明這一點), 卻足以支撐完整的編譯器目標與作業系統。

2.2 標準擴展

字母	名稱	內容
M	整數乘除	mul/mulh*/div/rem(第 20 章)
A	原子操作	lr/sc、amoswap/amoadd...(多核同步)
F / D	單/雙精度浮點	32 個 f 暫存器、IEEE 754(第 9 章)
C	壓縮指令	常用指令的 16 位元編碼, 程式縮小約 25–30%
V	向量	可變長度向量運算 (SIMD 的優雅解)
Zicsr	CSR 指令	csrrw/csrrs...(第 21 章)
Zifencei	指令流同步	fence.i
B/Zb*	位元操作	旋轉、計數、抽取等

命名法:RV32IMAC = RV32I + M + A + C;**G** = IMAFD + Zicsr + Zifencei(“General”), 故 RV64GC 是 Linux 級應用處理器的代名詞。設定檔 (Profiles, 如 RVA23) 進一步規定應用處理器必備的擴展組合, 保證軟體相容性。

3 程式設計師模型 (RV32I)

暫存器	**ABI** 名	保存責任	用途
x0	zero	—	恆為 0; 寫入被丟棄
x1	ra	呼叫者	返回位址
x2	sp	被呼叫者	堆疊指標 (16 位元組對齊)
x3	gp	—	全域指標
x4	tp	—	執行緒指標
x5–x7	t0–t2	呼叫者	臨時
x8	s0/fp	被呼叫者	保存暫存器/框架指標
x9	s1	被呼叫者	保存暫存器
x10–x11	a0–a1	呼叫者	參數/回傳值
x12–x17	a2–a7	呼叫者	參數
x18–x27	s2–s11	被呼叫者	保存暫存器
x28–x31	t3–t6	呼叫者	臨時

- **x0** 恆零是 RISC 經典技巧: 許多操作免專用指令——`mv=addi rd,rs,0`、`nop=addi x0,x0,0`、`neg rd,rs=sub rd,x0,rs`、丟棄結果 = 寫入 x0。
- **PC** 不是通用暫存器 (對比 Arm32 的 r15): 讀取 PC 用 `auipc`, 控制轉移只經分支/跳躍指令——微結構因此不必監控「任何指令都可能改 PC」。
- 沒有旗標暫存器: 比較與分支合一; 多字加法的進位以 `sltu` 求得。所有指令的相依關係都顯式地經由暫存器——對管線、超純量、編譯器都是好消息 (第 12、14 章)。
- **hart**(hardware thread): 規格用語, 一個硬體執行緒 = 一套暫存器 +PC。多核 = 多 hart。

4 記憶體模型

- 位址空間 2^{32} 位元組, 小端序 (第 10 章);
- 載入/儲存粒度: 位元組 (b)、半字 (h,2B)、字 (w,4B);
- 未對齊存取: 規格允許 (可由硬體或陷阱模擬), 但效能不保證;
- I/O 採記憶體映射 (第 7 章);
- 多 hart 的記憶體序為弱序 RVWMO, 同步靠 `fence` 與 A 擴展 (第 16 章)。

5 生態系現況

- 軟體:GCC/LLVM、Linux、glibc/musl、QEMU、OpenSBI/U-Boot 均已上游支援;
- 開源硬體:Rocket/BOOM(Chisel)、PicoRV32、VexRiscv、CVA6、Ibex——閱讀它們是絕佳的進階教材;

- 商用:SiFive、Andes(晶心)、平頭哥玄鐵、Ventana、Tenstorrent 等; 出貨量最大宗是嵌入式控制核心 (硬碟、GPU、基頻內的管理核), 應用處理器與 AI 加速器快速成長;
- 工具:Spike(黃金參考模擬器)、riscv-tests(相容性測試, 第 26 章將借用其思想)、riscof。

6 本章重點回顧

本章要點

- RISC-V= 開放標準 ISA: 規格免費、實作自由; 基礎集凍結永不變。
- 模組化:RV32I(40 條指令)+M/A/F/D/C/V/Zicsr...;RV64GC=Linux 級標配。
- 程式模型:32 個 x 暫存器 (x0 恆零)、獨立 PC、無旗標、小端序。
- ABI 名稱 (ra/sp/a0/s0/t0...) 與保存責任是第 19 章呼叫慣例的基礎。

7 複習問題

1. 「開放 ISA」開放的是什麼、不是什麼? 與 Arm 授權模式的差異?
2. RV32IMAC、RV64GC 各包含哪些模組?
3. x0 恆零帶來哪些好處? 舉出三個利用 x0 合成的偽指令。
4. RISC-V 為何不設旗標暫存器? 這對管線設計有何影響?
5. RV32E 與 RV32I 的差異? 各適合什麼場景?
6. 什麼是 hart? 單核雙執行緒 (SMT) 的處理器有幾個 hart?

參考資料

[1] RISC-V International, *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture*, ver. 20260120.

[2] K. Asanović and D. Patterson, “Instruction Sets Should Be Free,” UCB/EECS-2014-146.

[3] D. Patterson and A. Waterman, *The RISC-V Reader*, Strawberry Canyon, 2017.