計算機組織與 RISC-V CPU 設計 · 第 20 章

標準擴展:M、C 與其他

RV32M 乘除法 / C 壓縮指令 / A 原子操作概覽 / 擴展的取捨

摘要

基礎集之外, 真實的 RISC-V 處理器總會搭配若干標準擴展。本章詳述與本教材實作最相關的 **M** 擴展 (乘法與除法): 八條指令的語意、mulh 家族的用途、除零與溢位的定義行為; 介紹 **C** 擴展 (16 位元壓縮指令) 的編碼思想與硬體代價; 概覽 **A** 擴展 (原子操作) 如何支撐多核同步; 最後討論「該選哪些擴展」的工程取捨。

目錄

1 M 擴展: 乘法與除法

八條指令, 全部 R 型 (opcode=0110011,funct7=**0000001**):

指令	**funct3**	語意
mul	000	rd = (rs1 × rs2)[31:0](低 32 位; 帶號/無號同)
mulh	001	rd = (帶號 × 帶號)[63:32]
mulhsu	010	rd = (帶號 × 無號)[63:32]
mulhu	011	rd = (無號 × 無號)[63:32]
div	100	rd = rs1 ÷ rs2(帶號, 向零截斷)
divu	101	rd = rs1 ÷ rs2(無號)
rem	110	rd = 帶號餘數 (符號同被除數)
remu	111	rd = 無號餘數

1.1 定義行為 (不設陷阱)

情況	商	餘數
除以零	全 1(-1 / $2^{32}-1$)	被除數
帶號溢位 ($-2^{31} \div -1$)	-2^{31}	0

RISC-V 不因除零觸發例外——語言運行期 (如 C 的 UB、Java 的 ArithmeticException) 自行檢查。硬體簡單、語意可移植。

取得完整 64 位元乘積

```
mulh t1, a0, a1     # 高 32 位 (必須先發!)
mul  t0, a0, a1     # 低 32 位
# 微結構提示: 規格建議把 mulh/mul 相鄰成對,
# 硬體可融合為一次乘法 (macro-op fusion)。
```

1.2 硬體實作光譜

- 迭代式: 移位—加法 (第 9 章 Booth),32 週期, 面積極小——微控制器;
- 陣列/華萊士樹: 單週期或 2–4 級管線化乘法器——應用處理器;
- 除法幾乎都是迭代 (每週期 1–2 位), 或以牛頓法用乘法器逼近。

2 C 擴展: 壓縮指令

C 擴展提供 16 位元編碼的常用指令別名:`c.addi`、`c.lw`、`c.mv`、`c.j`……。設計要點:

- 每條 c.* 都能一對一展開為 32 位元指令——組譯器/解碼器前端做展開, 後端資料路徑完全不變;
- 常用的 8 個暫存器 (x8–x15) 用 3 位元欄位編碼; 立即值/位移縮短;

- 指令流成為 16/32 位元混合: 程式大小平均縮小 25–30%,I-cache 命中率上升;
- 代價: 提取對齊複雜化 (32 位元指令可能跨字邊界)、解碼器前端多一級展開;jalr 目標因此只強制 2 位元組對齊 (第 18 章)。

嵌入式市場幾乎必選 C(Arm 的 Thumb-2 證明過這條路); 高效能核也普遍支援。

3 A 擴展: 原子操作概覽

多 hart 同步需要「讀—改—寫」不可分割:

- `lr.w rd,(rs1)` / `sc.w rd,rs2,(rs1)`: 載入保留/條件儲存——sc 僅在保留未被打破時成功 (回傳 0), 否則失敗 (非 0)。以重試迴圈實作任意原子操作 (CAS、自旋鎖)。
- `amoadd.w`、`amoswap.w`、`amoand/or/xor/min/max`: 單指令原子讀改寫, 適合計數器、佇列指標。
- 記憶體序修飾位 aq/rl(acquire/release) 配合 RVWMO(第 16 章)。

```
lock:
    li   t0, 1
retry:
    lr.w t1, (a0)         # 讀鎖
    bnez t1, retry        # 已被持有 -> 忙等
    sc.w t1, t0, (a0)     # 嘗試寫 1
    bnez t1, retry        # sc 失敗 -> 重試
    ret
unlock:
    sw   x0, 0(a0)        # 寫 0 釋放 (配合 fence)
    ret
```

程式 1 自旋鎖 (教學版)

4 工程取捨: 選哪些擴展?

應用	典型組合
極簡控制核 (FSM 替代品)	RV32E 或 RV32I
微控制器 (本教材實作級)	RV32I 或 RV32IM(C 視程式記憶體壓力)
RTOS 級嵌入式	RV32IMAC + Zicsr
Linux 應用處理器	RV64GC(+H 虛擬化、V 向量視需求)

模組化的威力: 驗證與面積成本只花在用得到的功能上。我們的 Verilog 實作選擇純 RV32I——最小可教學集; 讀者可把 M 擴展當作第一個自我挑戰 (第 26 章習題)。

5 本章重點回顧

本章要點

- RV32M 八條:mul/mulh 家族取低/高 32 位;div/rem 除零與溢位有定義結果、不陷阱。
- C 擴展 = 常用指令的 16 位元別名, 前端展開、後端不變; 省 25–30% 程式空間。
- A 擴展:lr/sc 與 amo* 支撐鎖與無鎖結構。
- 擴展選配是成本工程:RV32I → RV32IMC → RV64GC 隨應用升級。

6 複習問題

1. 為什麼 mul 不分帶號/無號,mulh 卻要分三種?
2. 計算 `div`/`rem` 對 $(-7) \div 2$ 的結果;RISC-V 的截斷方向是什麼?
3. C 擴展為何能「後端不變」? 這對驗證成本意味著什麼?
4. 以 lr/sc 寫出原子的 fetch-and-add。
5. 你要設計一顆電池供電感測器 SoC 的控制核, 選哪些擴展? 說明理由。

參考資料

[1] RISC-V International, *ISA Manual Vol. I*, ver. 20260120: “M”、“C”、“A” chapters.

[2] A. Waterman, *Design of the RISC-V ISA*, UCB PhD thesis, 2016(C 擴展的定量依據).