計算機組織與 RISC-V CPU 設計 · 第 11 章

指令集: 定址模式與格式

七種定址模式 / 指令格式設計 / RV32I 的六種格式

摘要

運算元「在哪裡、怎麼找」是指令集設計的核心問題之一。本章系統性地介紹七種經典定址模式 (立即、直接、間接、暫存器、暫存器間接、位移、堆疊), 分析各自的優缺點與硬體代價; 接著討論指令格式設計: 指令長度、欄位分配、變長 vs. 定長; 最後剖析 RV32I 的六種指令格式 (R/I/S/B/U/J), 說明其欄位安排背後的硬體考量。本章對應 Stallings 第 11 章。

目錄

1 定址模式

設 A = 指令中的位址欄位、R = 暫存器編號、EA = 有效位址 (運算元實際所在)。

模式	算法	優點	缺點
立即	運算元 = A	無記憶體參考、最快	值域受欄位寬限制
直接	EA = A	簡單、一次記憶體參考	位址空間受欄位寬限制
間接	EA = (A)	位址空間大	兩次記憶體參考
暫存器	運算元 = (R)	無記憶體參考、指令短	暫存器數量有限
暫存器間接	EA = (R)	空間大、一次記憶體參考	需先載入指標
位移	EA = A + (R)	彈性大 (見下)	需一次加法
堆疊	EA = 堆疊頂	指令零位址	應用受限

1.1 位移定址的三種化身

位移 **(displacement)**= 基底 + 偏移, 依「誰當基底」有三種經典用法:

1. 相對定址: 基底 = PC。分支目標多在附近, 短偏移即可; 且程式位置無關。RV32I 所有分支與 `jal` 都是 PC 相對。
2. 基底暫存器定址:R 存結構/區段起點,A 為欄位偏移——存取 struct 成員、堆疊區域變數 (`lw t0, 8(sp)`)。
3. 索引定址:A 為陣列起點,R 為索引; 搭配自動增量可高效走訪陣列 (部分 ISA 提供 post-increment;RISC-V 基礎集不提供, 由編譯器強度削減達成)。

重點觀念

RV32I 刻意只保留兩種記憶體定址: 載入/儲存的「暫存器 + **12** 位元帶號位移」與分支/跳躍的「**PC** 相對」。沒有間接、沒有自動增量、沒有縮放索引——每條載入/儲存最多一次記憶體參考、一次加法, 這讓第 23 章的資料路徑天然簡單。複雜定址由多條簡單指令合成 (RISC 哲學, 第 13 章)。

合成陣列索引 A[i](4 位元組元素)

```
slli t1, t0, 2       # t1 = i * 4
add  t1, t1, s0      # t1 = &A + i*4  (s0 存 &A)
lw   t2, 0(t1)       # t2 = A[i]
```

x86 的 `mov eax,[rbx+rcx*4]` 一條做完;RV32I 三條——指令多, 但每條都能在單一週期的簡單硬體上完成。

2 指令格式設計

2.1 指令長度

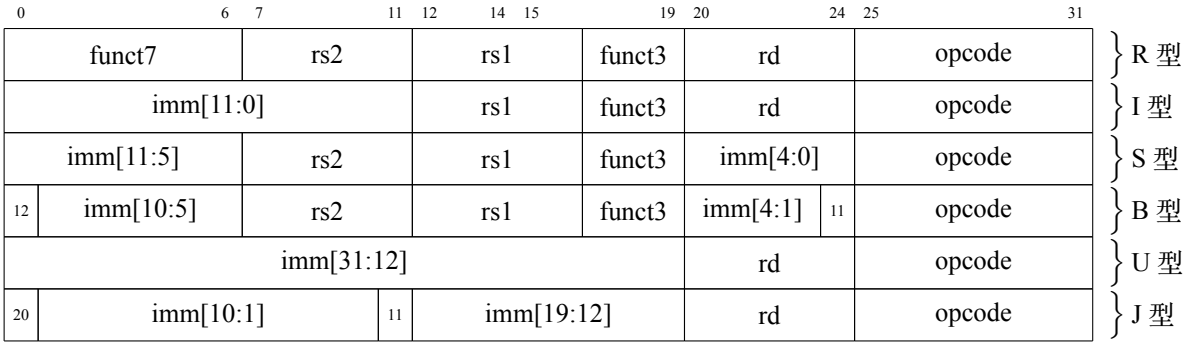
最根本的取捨:

- 變長指令 (x86:1–15 位元組): 編碼緊湊、劇目豐富; 但解碼複雜——不知道這條多長, 就不知道下一條在哪, 平行解碼困難。
- 定長指令 (RV32I: 固定 32 位元): 解碼簡單、提取規律、管線友善; 代價是密度較低。RISC-V 的 C 擴展 (16 位元壓縮指令) 在保住解碼簡單性的前提下拿回密度。

2.2 欄位分配

opcode 位元數 vs. 位址位元數的拉鋸。技巧包括: 擴展 **opcode**(部分編碼點展開為更長的 opcode)、依指令類型使用不同欄位佈局。原則: 愈常用的資訊, 位置愈固定——解碼器可以並行地「猜」。

3 RV32I 的六種指令格式



- **R** 型: 暫存器—暫存器運算 (`add, sub, sll, slt, xor, or, and...`);funct7+funct3+opcode 共同決定運算。
- **I** 型: 立即值運算 (`addi...`)、載入 (`lw`)、`jalr`;12 位元帶號立即值。
- **S** 型: 儲存 (`sw`); 立即值拆為兩段, 讓 **rs1/rs2** 欄位與 **R** 型完全對齊。
- **B** 型: 分支; 偏移單位為 2 位元組 (imm[0] 隱含為 0),±4 KiB 範圍。
- **U** 型:`lui/auipc`;20 位元高位立即值。
- **J** 型:`jal`;±1 MiB 範圍。

格式背後的硬體巧思

- **rs1**、**rs2**、**rd** 在所有格式中位置固定 (bits 19:15、24:20、11:7)——暫存器檔可以在解碼前就開始讀取, 不必等 opcode 判明。
- 立即值的符號位永遠在 **bit 31**——符號延伸電路只看一個位元, 關鍵路徑最短。
- S/B 型立即值的「奇怪」拆位, 是為了讓上述兩點成立: 欄位遷就硬體, 而非硬體遷就欄位。你將在第 23 章的立即值產生器 (ImmGen) 中親手實作這些拼接。

4 組譯器視角: 偽指令

RV32I 沒有「載入 32 位元常數」的單一指令, 以 `lui`(高 20 位)+`addi`(低 12 位) 合成; 組譯器以偽指令隱藏細節:`li t0, 0x12345678` 自動展開為兩條。常見偽指令:`mv, not, neg, nop, j, ret, call, beqz`(第 19 章詳述)。

5 本章重點回顧

本章要點

- 七種定址模式的核心差異: 記憶體參考次數 vs. 可達位址空間 vs. 硬體複雜度。
- 位移定址三化身:PC 相對 (分支)、基底 + 偏移 (結構/堆疊)、索引 (陣列)。
- 變長指令緊湊、定長指令好解碼;RV32I 定長 32 位元,C 擴展補密度。
- RV32I 六格式:R/I/S/B/U/J; 暫存器欄位固定、立即值符號位固定在 bit 31。

6 複習問題

1. 間接定址 EA=(A) 為何需要兩次記憶體參考? 暫存器間接如何改善?
2. PC 相對定址為何使程式「位置無關」? 這對動態連結有何意義?
3. 手工編碼 `addi x5, x6, -1` 為 32 位元機器碼 (十六進位)。
4. 為什麼 S 型要把立即值拆成兩段? 好處是什麼?
5. B 型立即值為何不含 bit 0? 分支範圍因此是多少?
6. 以 `lui`+`addi` 合成 `li t0, 0xDEADBEEF`; 注意 addi 立即值符號延伸造成的修正問題。

參考資料

[1] W. Stallings, *Computer Organization and Architecture*, 6th ed., Chapter 11.

[2] RISC-V International, *The RISC-V Instruction Set Manual, Volume I*, ver. 20260120.

[3] A. Waterman, *Design of the RISC-V Instruction Set Architecture*, PhD thesis, UC Berkeley, 2016.