

3月14日

1. CISC 和 RISC 各自优势和劣势

(1). CISC.

优势: ① 单周期完成复杂功能, 代码量少, 对编译器和存储空间要求低

缺点: 指令不统一, 形式复杂, 译码困难; 硬件结构复杂, 测试验证难度高.

(2) RISC.

优势: 指令统一, 形式简单, 译码简单; 硬件结构简单, 适合流水线

劣势: 对编译器设计要求高, 代码密度低.

2. RISC-V 的基本指令集: RV32I, RV32E, RV64I, RV128I.

其他常见的扩展:

A (原子指令集) 作用: 原子操作, 加载保留, 条件存储.

应用: 多核体系, 并发操作; 与内存一致性模型相匹配.

M (乘除法指令集) 作用: 乘除运算

D 应用: 需要使用乘除法时

F (单精度浮点数) 作用: 浮点运算和转化

应用: 需要浮点运算的系统 (需要通用浮点寄存器).

C

作用: 定义了部分指令的 6 位版本

应用: 小内存嵌入式.

V (向量操作)

作用: 向量操作 (并行运算, 读取和写回, 依靠掩码的并行执行).

应用: 向量化程序.

4.

(1) 不相符. RV32I 的 add 指令 opcode = 0110011, RV64I 的 addw 指令 opcode = 0111011

而 RV32I 和 RV64I 的 add 指令的 opcode 都为 0110011. 这样设计是为了使



RV64I 能够支持 RV32I 指令。而 addw 指令对 32 位有符号数运算并将结果截断为 32 位。这样 RV64I 可支持不同位宽操作。

(2) ~~不需~~

(2). 问题: RV64I 中, addw 和 addiw 的目标寄存器中存放的 32 位计算结果是否需要额外的符号扩展才能用于后续 64 位计算? 为什么?

答: 不需要。因为 RV64I 寄存器是 64 位的, addw 的执行原理是: 将结果的高 32 位忽略, 将低 32 位进行 sign-extend 后存入目标寄存器。而 addiw 同理。

5. 什么是 HINT? 作用?

HINT 指令空间是 RISC-V 标准指令集中的保留指令编码之一。它表明当前的 opcode 被保留用于未来的标准微架构提示。所有以 HINT 标记的指令, 在实现时都当作空操作。

6.

$a_2 = -3$ $a_3 = 1$.

符号规定:

① div: 被除数和除数符号相同时为正, 否则为负

② divu: 结果为正

③ rem: 余数与被除数符号相同

④ remu: 余数为正数



11.

(1) jal ra, 0x88 \Rightarrow 偏移量寻址.

(2) jalr x0, ra, 0 \Rightarrow 内存直接寻址.

(3) addi a0, a1, 4 \Rightarrow 立即数寻址.

(4) mul a0, a1, a2 \Rightarrow 寄存器直接寻址.

(5) ld a4, 16(sp) \Rightarrow 寄存器间接寻址.

