

1. CISC 单个指令完成的任务量大且指令长度灵活

优点: 对编译器和程序存储空间的要求较低

缺点: 硬件设计复杂, 测试验证难度较高

RISC: 单个指令完成的任务量少且功能单一, 指令长度相对固定

优点: 硬件设计较为简单, 适合利用流水线提升性能

缺点: 对编译器设计的要求较高, 程序的代码密度较低

2. RISC-V 指令集是目前较为流行的一种 RISC 架构指令集, 同时它也是寄存器-寄存器型指令集, 其由整数的基本指令集和一系列扩展指令集组成。这种划分的组织方式使 RISC-V 指令集能够适应多种应用场景, 小到简易嵌入式设备, 大到高性能多核向量处理器, 都适用。

M: 扩展整数乘法和除法指令, 作用范围: 整数

F: 扩展了 IEEE 标准单精度浮点数运算指令, 增加了 32 个 32 位浮点寄存器

用于浮点运算和转码, 算术和逻辑运算, 乘除运算, 类型转换

D

~ 32 个 64 位浮点寄存器

范: 适用需要浮点数的系统, 需要 32 个通用寄存器, 需要保存 CSR。

V: 向量操作, 向量预处理, 向量读取和写入, 依靠代码和硬件执行

范围: 需要 32 个通用向量寄存器, 适用于向量化程序, 节省指令存储, 并行化执行

A: 内存原子操作, 加载保留, 条件存储,

范围: 与内存一致性模型相匹配的多核体系, 便于信息交流。

4. RV32I add opcode: 0110011

RV64I addw opcode: 0111011

]不同

RV64I add opcode: 0110011 相同

原因: ① 应用范围不同: addw 为 64 位寄存器指令, 将结果低 32 位保留, 高位进行符号扩展存入寄存器中, 而 add 指令在 32 位中为两 32 位相加, 高位溢出存入寄存器中, 64 位亦如此, 因此 add 与 32 位

64 位 addw 操作数不同



兼容性, RV-32I add 和 RV-64I add 具有相同的操作模式, 使用相同的操作码有利于提升指令的兼容性与活性

2) 要

在RV中, addw 和 addiw 分别用于双位有符号数相加, 和立即数加法, 并将结果截断为32位有符号整数, 因此, 如果需要将其用于后续的64位计算, 需要进行符号位扩展。

5. HINT空间是一组预留给实现者使用, 用于定义特定的指令集序列, 以提高处理器的性能和能效的空间

作用: 为处理器实现者提供一组可供自定义的指令空间, 以用于处理器的性能和能效优化。

eg: HINT指令可实现一些特殊操作: 预取数据, 跳转预测, 分支预测等。

6: $u_2 = -3$

$a_3 = 1$

div: 有符号数除法 为0报错

divu 无 ~ ~

rem: 有符号数取余 余数符号和被除数一致

remu: 无 ~ ~ 取正

11) ~~立即数寻址~~ ~~寄存器寻址~~ 内存直接寻址

12) 寄存器寻址

13) 立即数寻址

14) 寄存器直接寻址

15) 偏移量寻址

