

一，CISC和RISC架构各自的优势和劣势

CISC:

- 优势
 1. 更少的指令数：CISC指令包含多项操作，因此需要更少的指令数来完成同样的任务，这降低了程序的长度和复杂度。
 2. 更灵活的编程：CISC指令的多功能性使得编写程序变得更加简单和容易。相比之下，RISC指令更加简单和直接，需要更多的代码才能完成同样的任务。
 3. 更少的存储访问：由于每个指令包含多个操作，因此CISC架构需要更少的存储访问来完成同样的任务，这提高了程序的效率。
- 劣势
 1. 更复杂的硬件：由于CISC指令包含多个操作，因此需要更多的硬件资源来处理它们。这使得CISC处理器更加昂贵，同时也增加了系统的复杂度。
 2. 更难以优化：由于CISC指令的复杂性，编译器难以对其进行有效的优化，这导致了程序的执行速度较慢。
 3. 更难以设计：由于CISC指令的复杂性，处理器的设计变得更加困难，需要更多的时间和精力来完成。

RISC:

- 优势
 1. 更快的执行速度：由于RISC指令非常简单，处理器可以更快地执行它们，这提高了程序的执行速度。
 2. 更容易优化：由于RISC指令非常简单，编译器可以更容易地对其进行优化，这提高了程序的执行速度。
 3. 更容易设计：由于RISC指令非常简单，处理器的设计变得更加容易，需要更少的时间和精力来完成。
 4. 更支持流水线：由于RISC指令短小精悍，一个指令只执行一个简单操作，因此非常方便流水线切割。
- 劣势
 1. 更多的指令数：由于每个指令只能完成一项操作，因此需要更多的指令来完成同样的任务，这使得程序变得更长和更复杂。
 2. 更多的存储访问：由于RISC指令只能完成一项操作，因此需要更多的存储访问来完成同样的任务，这降低了程序的效率。
 3. 更少的编程灵活性：由于RISC指令非常简单，编写程序变得更加困难和繁琐

二，RISC-V中的基本指令集及五个扩展指令集

1. 基本指令集——整数指令集R32I/R64I

- 用于寄存器-寄存器操作的 **R 类型指令**
- 用于短立即数和访存 **load** 操作的 **I 型指令**
- 用于访存 **store** 操作的 **S 型指令**
- 用于条件跳转操作的 **B 类型指令**

- 用于**长立即数**的 **U 型指令**
- 用于**无条件跳转**的 **J 型指令**

2. 扩展指令集

- 原子操作指令 (A)

提供了一些原子性的内存读/写操作。旨在支持并发访问内存，防止由多个并发线程或处理器同时读写同一内存位置而导致的数据竞争问题。主要应用于多线程并发控制、同步、锁定等场景。例

- 位操作指令 (B)

分为无条件分支和有条件分支。用于控制程序流程，例如实现循环和条件分支等。

- 乘除法指令 (M)

用于进行高效的乘除法操作

- 双精度浮点指令 (D)

支持基本的双精度浮点数运算，如加、减、乘、除、平方根、转换等，以及一些高级运算，如三角函数、指数函数、对数函数等。主要用于需要高精度运算的应用领域，如科学计算、数据处理、图形处理等。

- 单精度浮点指令 (F)

支持单精度浮点数的运算，如浮点加、减、乘、除、取负、取绝对值、比较、类型转换等等。可用于加速许多需要使用浮点数的应用程序，例如图形处理、科学计算、信号处理等。

三， RISC-V规范

1.

RV32I中add指令的opcode为0110011，而RV64I中addw指令的opcode为0111011。

RV64I的add指令的opcode也是0110011，和RV32I一样。

这样设计的原因是提高指令集覆盖率。在RV64I中，由于操作数位数的不同，指令的格式稍有变化，但是其功能码仍然和RV32I保持一致。这样设计的好处是可以提高指令的可移植性和代码的重用性，同时也能够降低指令集的复杂度和设计成本。

2.

不需要，因为addw和addiw指令都会对结果进行64位符号位扩展。

四， RISC-V的I标准指令集的HINT指令空间及作用

1. HINT指令空间

HINT 指令空间是指令头部的 6 位为 0b000001 的指令。这些指令通常用于向处理器发出一些提示信息，比如调试信息、控制指令执行的优先级等。由于这些指令不会修改寄存器或内存，也不会产生异常，因此它们的执行时间是很短的，通常只需要一个时钟周期即可完成。

2. 作用

HINT 指令的作用是向处理器发出提示信息，比如：

- ECALL 和 EBREAK 指令用于向处理器发出调试信息，包括异常的触发原因、当前指令地址等信息。
- FENCE 指令用于向处理器发出指令序列中的内存顺序提示，以确保指令的执行顺序与内存操作的顺序一致。
- NOP 指令用于向处理器发出空指令提示，通常用于填充指令序列或者延迟执行。

五,

6/ 考虑如下指令序列：
div a2,a0,a1 $a2 = a0 \div a1 = -3.2 \rightarrow -3$
rem a3,a0,a1 $a3 = a0 \% a1 = 1$
假设寄存器 a0 和 a1 的初始值分别为 16 和 -5，则上述指令序列执行完成后 a2 和 a3 寄存器中的值分别是多少？简要说明 RISC-V 的 M 标准指令集中对除法和余数指令的符号规定。
在执行运算前先对 rs1 和 rs2 进行有符号数扩展

7 RISC-V 标准指令集并未为加法指令引入未用的标志位，因此通常需额外的指令

六,

- 11/ 写出以下指令使用的寻址模式。
- 1) jal ra,0x88 *偏移量寻址*
 - 2) jalr x0,ra,0 *内存直接寻址*
 - 3) addi a0,a1,4 *立即数寻址*
 - 4) mul a0,a1,a2 *寄存器直接寻址*
 - 5) ld a4,16(sp) *寄存器间接寻址*