1. 简要分析 CISC 和 RISC 架构各自的优势和劣势。

	CISC	RISC
优势	1. 设计灵活：可以设计出非常复杂的指令 2. 高效内存访问：CISC指令通常涵盖多个基本操作，可以减少内存访问的次数，降低延迟。	1. 指令集简单：RISC指令通常只涵盖基本操作，指令长度短且长度一定，易于实现和优化，执行速度快。 2. 容易并行处理：RISC指令的独立性和执行速度快的优势，使并行处理容易实现，可有效利用处理器的多核处理能力
劣势	1. 复杂度高：CISC指令通常很长，且具有很多不同的格式，会增加指令解码时间和电路复杂度。 2. 处理器速度不稳定：由于CISC长度不一，解码复杂，导致执行时间也可能不同，处理器运行速度不稳定。	由于RISC指令只能涵盖基本操作，不方便一些高级操作的实现。一些复杂的操作可能需要多个指令才能实现

2. RISC-V 中的基本指令集是什么？列举五个常见的 RISC-V 标准扩展指令集并简要说明它们的作用和应用范围。

RISC-V的基本指令集包含47条指令，分为I、U、J、S、B、R、F、D等不同类型。

五个常见指令集：

1. RV32I：RISC-V基本指令集，适用于32位机器
2. RV64I：RISC-V基本指令集，适用于64位机器
3. RV32M：RISC-V乘法和除法扩展指令集。
4. RV32F：RISC-V单精度浮点数扩展指令集。

（3、4）适用于32位机器

5. RV32C：RISC-V指令压缩扩展指令集。允许指令以16位的形式进行压缩。

4. 阅读 RISC-V 规范以回答以下问题：

1) RV32I 中的 add 指令和 RV64I 中的 addw 指令均为 32 位整型加法指令，它们是否具有相同的指令操作数（opcode）？此外，RV32I 中的 add 指令和 RV64I 中的 add 指令是否具有相同的指令操作数（opcode）？试分析为什么采取这样的设计。

1). 是. 否.

编译器和调用约定总是认为所有的32位数值总是以符号扩展的格式保存在64位寄存器中的. 即使是32位无符号整数,也会把它的31位扩展到63~32位. 因此,在无符号32位整数和有符号32位整数之间进行转换,并没有实质的操作,就像从有符号32位整数转成有符号64位整数一样. 现有的64位宽度SLTU指令和无符号分支比较指令,在这种不改变的约定下,仍然能够针对无符号32位整数正确工作. 相似的,在32位符号扩展整数上,执行现有的64位宽度逻辑操作,仍然保持了符号扩展的正确性. 需要一些新的指令(ADDIW/SUBW/SxxW),来执行针对32位数值的额外的加法、移位,以确保适当的性能.

2) 在 RV64I 中，addw 和 addiw 指令的目标寄存器中存放的 32 位计算结果是否需要进行额外的符号扩展才能用于后续 64 位计算？请说明理由。

需要.

addw和addiw指令的目标寄存器是一个32位寄存器,生成的结果也是一个32位的值. 该结果存储于rd中,若不进行符号扩展,则高32位将会被填充为0. 会导致在使用rd进行64位计算时得到的结果不符合预期.

6. 考虑如下指令序列：

```
div     a2,a0,a1
rem     a3,a0,a1
```

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。

a_2: -3. a_3: 1

div和rem: 有符号整数除法 divu和remu: 无符号整数除法

- 余数的符号与被除数相同.
- 除数与被除数同号则商为正,异号则商为负
- 除以0.结果的商的所有位被置1. 即对无符号除法来说,商是 $2^{XLEN}-1$,对于有符号除法来说,商是-1. 结果的余数等于被除数.
- 有符号除法溢出仅当用最小的负整数, -2^{XLEN-1},除以-1时才会出现.

此时商等于被除数,余数等于0. 无符号除法不会产生溢出.

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) 偏移量寻址.