2-1. CISC架构的机器，单个指令完成的任务较多，指令长度很灵活，因此执行效率比较高，同时对编译器和程序存储空间的要求较低；缺点是硬件的设计非常复杂，测试 验证的难度较高；

RISC架构的指令长度相对固定，单个指令完成的任务量少，这导致它硬件设计比较简单，适合利用流水线来提升性能；缺点是指令较为简单，导致一个任务可能需要多条指令完成，这导致指令数的增加，降低了程序密度，同时对编译器设计的要求比较高。

2-2. RISC-V的基本指令集是RV32I，其包含基本的算术与逻辑指令。

常见的扩展指令集：

RV32M：作用是添加了乘法与除法指令，适用于需要高效数学运算的机器。

RV32F：扩展了单精度浮点数运算的指令。

RV32D：扩展了双精度浮点数运算的指令。RV32F/D适用于需要高精度浮点数运算，现实模拟运算的情景。

RV32C：扩展了压缩指令，可以将一些常用指令压缩为16位，适用于存储空间小，或需要高效存储器访问的机器。

RV32A：扩展了原子指令，增加了对原子操作的支持，适用于需要多线程与处理器并发编程以及需要快速响应事件的情景。

2-4. 1) RV32I中add指令的opcode为：0110011，RV64I中addw：0111011

它们的opcode不相同。RV64I中add指令的opcode也为0110011，与RV32I相同。

(riscv-spec-20191213.pdf, P5)

原因：查阅了官方文档找到了对该问题的解释：RV64I的重点是在其中支持32位整数的指令，而不是提供32位ISA的兼容性。因此RV64I与RV32I中add指令的opcode相同。其动机是消除并非所有操作码都带有W后缀引起的不对称性。这是同时设计RV32I与RV64I两个独立的ISA的结果，而不是将RV64I作为RV32I的扩展。这可能是不合理的，但是我们仍能把带W后缀的RV64I的指令作为RV32I的扩展。这是RV64I中addW与add的opcode不同的原因。

2) 不需要。查阅资料后发现(riscv-spec-20191213.pdf, P37)

ADDW指令对计算得到的32位整数已经作了符号位扩展。

2-5. HINT指令空间是指RISCV为HINT指令预留出的编码空间。HINT指令用于给微处理器发送相关的提示，使得CPU可以执行优化每操作。这样就可以提高程序的性能。但这些提示不是必需的，也不会改变程序原本的语义。

2-6. 完成后：$a_2 = -3$，$a_3 = 1$。

RV32M对除法、取余的符号规定：对于DIV以及REM两个有符号数操作，商的符号与被除数相同，余数的符号与除数相同；而对于DIVU、REMU两个无符号操作，商与余数皆为正。

2-11 1) 偏移量寻址。 2) 寄存器间接寻址 3) 立即数寻址

4) 寄存器直接寻址 5) 偏移量寻址