2.1 CISC 在处理器发展早期时受存储器容量及编译器优化水平的限制，在早期广泛使用但其由于兼容性问题其结构越来越复杂，此时 RISC 应运而生。

RISC 单个指令完成任务量少且功能单一，长度固定，对右更件设计而言更容易进行设计但其代码密度较低。

2.2 基本为 RV32I 指令，其包含寄存器类型（R-TYPE），立即数型（I-TYPE），访存指令（LOAD 为 I-TYPE，Store 为 S-TYPE），跳转指令，等操作，能完成除原子操作外所有操作。

常见的拓展有：M（乘法）、A（原子操作）、F/D（单、双精度操作）和 C（压缩指令）

其中 RV32IMAFD 写作 RV32G，通常编译器会支持 16bit 的压缩指令，则称为 RV32GC，这也是常见编译器所支持的 RISCV 指令集。

2.4 (1) add 为 0110011，addw 为 0111011，故其 opcode 不同

但 add 均为 0110011（参见 spec v20191213，P131 页），这表明 RV64I 对 RV32I 有兼容的特性。但是，RV32 中 add 为 32 位加法，RV64 中为 64 位加法，其实际意义不同。

(2) 由 spec，RV 的寄存器存储 64 位数，addiw 操作为：其将立即数又带符号拓展到 32 位与目标寄存器值相加，并忽略高于 32 位的溢出，再将结果符号拓展 64 位并存储。即是运算后的 32 位结果也是经符号拓展到 64 位的。因为右更件寄存器一定是 64 位的（addiw 指令相似）

2.5 RV32I 保留了大量指令空间，用于提示指令，通常用于对架构作出一定的提示，且不会产生任何效果（类似于 NOP）其主要通过 X0 寄存器来使用。

spec 认为 hint 可用于内存系统时间/空间局域性提示、分支预测提示、线程调度提示、安全性标签以及模拟/仿真的仪表标志。

2.6 由 spec，我们可知除法的余数符号与被除数一致

(与阵数数)

则 ⇒ $a_2=-3$

$a_3=1$

但如果在python中结果为(-4,-4)说明两语言在除法上的差异. 但C语言中这一块表现就一致, 即不同架构结果不同. 本题反映了RISC-V架构的情况

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