1.

CISC 架构的优势：

1) 复杂指令可以执行多个操作，可以完成复杂的任务，简化了程序员的编程工作；
2) 在相同的时钟速率下，CISC 架构可以处理更多的指令，因为它们能够在一个时钟周期内执行多个操作；
3) 内存访问操作相对较少，因此可以减少访问内存的时间，提高了效率；
4) CISC 架构具有高度的兼容性，可以运行大量的软件和应用程序。

CISC 架构的劣势：

1) 指令集过于复杂，导致处理器设计变得复杂，增加了处理器的制造成本；
2) 复杂指令执行需要更多的硬件资源，包括更多的电路和存储器，使处理器变得更加庞大，加大了功耗；
3) 在处理器的执行过程中，指令的执行需要消耗大量的时间，影响了处理器的性能表现。

RISC 架构的优势：

1) 精简指令集能够减少指令执行的时间，提高处理器的执行效率；
2) 精简指令集能够简化处理器设计，减少处理器的复杂度，降低制造成本；
3) 处理器中的寄存器比较多，因此可以减少对内存的访问，提高了处理器的效率；
4) 精简指令集的设计更加简洁、清晰，易于编译器的实现和优化。

RISC 架构的劣势：

1) 精简指令集的设计需要更多的代码，因此程序的大小比较大；
2) RISC 架构需要更多的指令来完成复杂的任务，程序的编写会更加困难；
3) 由于没有复杂指令的支持，处理器需要执行更多的指令才能完成同样的任务，因此，可能会需要更高的时钟频率才能达到与 CISC 相同的性能表现。

2.

RISC-V 的基本指令集： RISC-V 基本指令集包括 32 位和 64 位两个版本，其中 32 位的指令集共有 37 个指令，而 64 位的指令集共有 47 个指令。这些指令可以执行简单的算术、逻辑、移位、比较和控制操作，这些基本操作可以组合成更复杂的操作。

RV32I Base Instruction Set

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20|10:1|11|19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12|10:5]	rs2	rs1	000	imm[4:1|11]	1100011	BEQ
imm[12|10:5]	rs2	rs1	001	imm[4:1|11]	1100011	BNE
imm[12|10:5]	rs2	rs1	100	imm[4:1|11]	1100011	BLT
imm[12|10:5]	rs2	rs1	101	imm[4:1|11]	1100011	BGE
imm[12|10:5]	rs2	rs1	110	imm[4:1|11]	1100011	BLTU
imm[12|10:5]	rs2	rs1	111	imm[4:1|11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND
0000	pred succ	00000	000	00000	0001111	FENCE
0000	0000 0000	00000	001	00000	0001111	FENCE.I
000000000000		00000	000	00000	1110011	ECALL
000000000001		00000	000	00000	1110011	EBREAK
csr		rs1	001	rd	1110011	CSRRW
csr		rs1	010	rd	1110011	CSRRS
csr		rs1	011	rd	1110011	CSRRC
csr		zimm	101	rd	1110011	CSRRWI
csr		zimm	110	rd	1110011	CSRRSI
csr		zimm	111	rd	1110011	CSRRCI

RV64I Base Instruction Set (in addition to RV32I)

imm[11:0]		rs1	110	rd	0000011	LWU
imm[11:0]		rs1	011	rd	0000011	LD
imm[11:5]	rs2	rs1	011	imm[4:0]	0100011	SD
000000	shamt	rs1	001	rd	0010011	SLLI
000000	shamt	rs1	101	rd	0010011	SRLI
010000	shamt	rs1	101	rd	0010011	SRAI
imm[11:0]		rs1	000	rd	0011011	ADDIW
0000000	shamt	rs1	001	rd	0011011	SLLIW
0000000	shamt	rs1	101	rd	0011011	SRLIW
0100000	shamt	rs1	101	rd	0011011	SRAIW
0000000	rs2	rs1	000	rd	0111011	ADDW
0100000	rs2	rs1	000	rd	0111011	SUBW
0000000	rs2	rs1	001	rd	0111011	SLLW
0000000	rs2	rs1	101	rd	0111011	SRLW
0100000	rs2	rs1	101	rd	0111011	SRAW

常见的 RISC-V 标准扩展指令集：

1) M 扩展指令集：M 扩展指令集增加了乘法和除法指令，包括乘法、除法、取模等运算。这个扩展指令集适用于需要执行乘法和除法的应用，例如图形处理和信号处理等。

2) C 扩展指令集：C 扩展指令集增加了压缩指令，用于减小程序的大小，提高指令缓存的命中率。C 扩展指令集的指令长度为 16 位，可以有效地缩小程序的大小，节约内存空间，适用于需要节省存储空间的应用。
3) F 扩展指令集：F 扩展指令集增加了浮点运算指令，包括单精度和双精度浮点数的加、减、乘、除等操作。这个扩展指令集适用于需要高精度浮点运算的应用，例如科学计算和图形处理等。
4) D 扩展指令集：D 扩展指令集增加了双精度浮点运算指令，扩展了 F 扩展指令集的功能，适用于需要高精度双精度浮点运算的应用。
5) A 扩展指令集：A 扩展指令集增加了原子操作指令，包括读、写、比较和交换等操作。这个扩展指令集适用于需要进行原子操作的应用，例如多线程程序和操作系统等。

4.

不具有相同的操作数。具有相同操作数。两者的 add 指令都是用的是 32 位寄存器，所以使用同一操作数的指令，因其操作过程是一样的，但是 addw 是 64 位寄存器的加法操作，与 32 位的操作过程是不相同的。这种做法一方面节省的代码资源，一方面使得使用 32 与 64 位寄存器的加法操作都被实现。

需要，因为其使用的是 64 位寄存器，所以要进行符号扩展，运算后再截断。

5.

RISC-V 的 I 标准指令集中存在的 HINT 指令空间包括了一些特殊的指令，这些指令用于提供性能调试和其他系统级功能，它们不是必需的指令，但可以在需要时使用。HINT 指令空间中包含了 nop 指令，它用于在不执行任何操作的情况下延迟指令流水线，以便等待某些结果。此外，还有一些用于调试和性能监视的特殊指令，如 ebreak 和 fence。这些指令可以帮助开发人员更好地了解系统的性能瓶颈，并进行优化。

6.

假设寄存器 a0 和 a1 的初始值分别为 16 和-5，执行 div a2,a0,a1 指令后，a2 寄存器中存放的值为-3，因为 16÷(-5)的结果是-3 余 1。执行 rem a3,a0,a1 指令后，a3 寄存器中存放的值为 1，表示 16÷(-5)的余数为 1。在 RISC-V 的 M 标准指令集中，div 和 rem 指令对于除数和被除数都采用有符号数规定，即结果的符号由除数和被除数的符号决定。

11.

偏移量寻址。

内存直接寻址。

立即数寻址。

寄存器直接寻址。

寄存器间接寻址。