3.14 第4周 charpter two

1、RISC和CISC这两种架构各自的优缺点：

CISC的优点在于实现相同操作所需的指令数少，指令类型丰富，操作灵活，但高性能的硬件设计也会变得复杂。RISC架构由于指令类型简单，格式统一，硬件开发的周期更短，但在指令的灵活度上有一定的限制。

RISC 单个指令完成的任务量少且功能单一，指令长度相对固定，硬件设计相对简洁简单，适合用流水线来提升性能，但是对编译器设计要求较高，程序代码密度较低；而CISC则单个指令完成任务量大且功能复杂，指令长度灵活，适应于早期处理器主频低、运算速度慢、存储器容量小、对程序代码非常敏感，编译器技术不成熟以致于难以承担复杂的优化工作的特点，如CISC特具有复杂功能和结构的指令集加入到指令系统中，以增强硬件的运算能力，降低程序编译的难度和目标代码的容量，这样CISC对编译器和存储空间的要求大大降低，但也面临硬件设计复杂、测试难度高等问题，随着指令愈来愈多，为支持这些指令并保持兼容以前的指令，CISC处理器的硬件结构越来越复杂。基于微码控制方法实现复杂结构导致设计与调试难度持续走高。

2、RISC-V中的基本指令集有3个，即 RV32I：使用32位寄存器的基本32位整数指令；RV32E：只使用16个寄存器的基本32位指令，适用于低端的嵌入式应用；RV64I：使用64位寄存器的基本64位整数指令。

RISC-V的扩展指令集(3)(共5个)

M　扩展了整数乘法和除法指令

A　扩展了并发操作中的原子指令

F　扩展了IEEE标准单精度浮点数运算指令，增加了32个32位浮点寄存器

D　扩展了IEEE标准双精度浮点运算指令，增加了32个64位浮点寄存器

Q　扩展了四精度浮点运算指令

4. (1) RV32I中add指令和RV64I中的addw指令

均为32位整数指令，它们的操作码opcode分别为 0110011 和 0111011

RV32I中的add与RV64I中的add指令的opcode均为 0110011

原因：RV32I中的add与RV64I中 addw 功能不一样，addw的功能是对低32位有符号加法指令，addw rd, rs1, rs2 ： tmp[31:0] ← rs1[31:0] + rs2[31:0]

rd ← sign_extend(tmp[31:0])

另外 RV32I的add与 RV64I的add指令有着同样的操作码，是由于2个指令本质运算类似，只是操作对象位数不同，这为相同操作码的设计大大提高了ISA的兼容性好、复用性、系统整体性、可扩展性等.

(2) 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高位的符号位扩展才能用于后续64位计算？

不需要，因为addw与addiw指令已经进行了sign_extend操作了.

5. RISC-V的I标准指令集中存在的HINT指令空间，这是一组用于给处理器提供提示信息的指令. HINT指令并非是必须的指令，而是可以提高处理器性能和降低功耗的指令。HINT指令空间由一个16位的opcode和一个4位的funct3功能码空段组成。HINT向处理器发出提示信息，用一种灵活的方式告诉处理器如何更好地利用计算资源。

div rd, rs1, rs2

div rd ← rs1 / rs2

有符号除法指令

$a_2 = \frac{a_0}{a_1} = -3.2$

6. 指令序列

div a2, a0, a1

rem a3, a0, a1

其中 li a0, 16

li a1, -5

指令执行完后，a2中值为 -3.2

a3中值为 1

RISC-V的M标准指令集中那条除指令的符号规则

rem rd, rs1, rs2

rem rd ← rs1 % rs2 有符号取余指令

$a_3 = a_0 \% a_1 = 1$

被除数与除数均为有符号数，且符号相同？有符号取余指令则类似。

11、下列指令使用的寻址模式

(1) jal ra, 0x88　　PC相对地址寻址，0x88为偏移量。

(2) jalr x0, ra, 0　　偏移量寻址（基址ra + 偏移量0）。

(3) addi a0, a1, 4　　立即数寻址。

(4) mul a0, a1, a2　　寄存器寻址。

(5) lw a0, 16(sp)　　偏移量寻址（基址sp + 偏移量16）。