

3. 请写出以下伪指令等价的基本指令或指令组合

1) nop addi x<sub>0</sub>, x<sub>0</sub>, 0

2) ret jalr x<sub>0</sub>, ra, 0

3) call offset { auipc ra, offset[31:12]  
jalr ra, ra, offset[11:0]

其中 Len 表示寄存器位宽

4) mv rd, rs addi rd, rs, 0

5) rdcycle rd csrrs rd, cycle, x0

cycle 指周期计数器寄存器

6) sext.w. rd, rs slli rd, rs, Len-32  
srai rd, rd, Len-32

7. RISC-V 检查加法溢出

1) 考虑如下的指令序列

add t<sub>0</sub>, t<sub>1</sub>, t<sub>2</sub>

slti t<sub>3</sub>, t<sub>2</sub>, 0

slt t<sub>4</sub>, t<sub>0</sub>, t<sub>1</sub>

bne t<sub>3</sub>, t<sub>4</sub>, overflow

2) 当 t<sub>0</sub> 和 t<sub>2</sub> 均为无符号数时

有 add t<sub>0</sub>, t<sub>1</sub>, t<sub>2</sub>

bltu t<sub>0</sub>, t<sub>1</sub>, overflow 比较执行加法后和与其中一个加数的大小,

若和大于其中一个加数，则发生加法溢出，跳转至溢出处理函数进行

3) 在 x86 和 ARM 指令集中，加法溢出检测是通过处理器状态寄存器中的标志位来实现，进一步处理

标志位  
设置于FLAGS  
寄存器中

← 在 x86 中，加法指令会设置进位标志位(CP)、溢出标志位(OVF)和零标志位(ZF)。在 ARM 中，加法

指令会设置进位标志位(C)和溢出标志位(V)

加法结果产生溢出时，处理器状态寄存器中的标志位会被设置为1，否则为0。可以通过检查标志位判断  
加法是否溢出。

8. 阅读 RISC-V 规范以了解 RISC-V 对除数为 0 的除法指令的处理方法，回答以下问题。  
1) 对整型除法，填写下表。整型除法中除数为 0 是否会引起 RISC-V 抛出异常？试分析为什么采取这样的设计。

指令	rs1	rs2	Op=DIVU 时 rd 值	Op=REMU 时 rd 值	Op=DIV 时 rd 值	Op=REM 时 rd 值
Op rd, rs1, rs2	x	0	<u>XLEN - 1</u>	x	-1	x

2) 对浮点除法，除数为 0 将会引起 fcsr 控制寄存器中的相关标志位被置位。下图给出了 fcsr 的构成，请说明 fflags 的各位分别代表什么含义。fflags 被置位是否会使得处理器陷入系统调用？

31	8-7	5-4	3	2	1	0
Reserved		Rounding Mode (frm)	Accrued Exceptions (fflags)			
24	3		NV DZ OF UF NX	1	1	1

3) 调研其他指令集架构（如 x86、ARM 等）是如何处理除数为 0 的。

在 x86 架构中，除数为 0 时，会触发除法错误中断。

停止除法的进一步执行，抛出对应的异常。

根据 ARM 文档，ARMv7-M 中包括 SDIV 和 UDIV 指令。

在 ARMv7-R 文档中，有 SCTLR.D2 位启用除零故障检测：

SCTLR.C8 为 0 时，除以 0 得到的除法结果返回为 0。若 SCTLR.D2 为 1 时。

SDIV 和 UDIV 在除以零情况下会生成未定义的指令异常。SCTLR.D2 位在复位时被清零。

特权等级

12. 1) Linux Kernel Supervisor Mode  
2) BootROM Machine Mode  
3) Boot Loader Machine Mode  
4) USB Driver Supervisor Mode  
5) Vim User Mode