

3. 请写出与以下伪指令等价的基本指令或指令组合

1) nop addi X0, X0, 0

2) ret jalr X0, ra, 0

3) call offset $\begin{cases} \text{auipc ra, offset}[31:12] \\ \text{jalr ra, ra, offset}[11:0] \end{cases}$ 其中 len 表示寄存器位数

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

5) rdcycle rd csrwr 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 t0, t1, t2

slli t3, t2, 0

sllt t4, t0, t1

bne t3, t4, overflow

2) 当 t0 和 t2 均为无符号数时

有 add t0, t1, t2

bltu t0, t1, 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, 跳至溢出处理函数处进行

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

设置于 EFLAGS 指令会设置进位标志位 (C) 和溢出标志位 (V)

寄存器中 加法指令结果产生溢出时, 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	$2^{XLEN}-1$	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
				1	1	1
				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, 抛出时应有的异常。

根据 ARM 文档, ARMv7-M 中包括 SDIV 和 UDIV 指令。

在 ARMv7-R 文档中, 有 SCTLR.DZ 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.DZ 为 1 时,

SDIV 和 UDIV 在除以零情况下会生成定义的指令异常, SCTLR.DZ 位在复位时

被清零

特权等级

12. 1) Linux Kernel

2) Boot ROM

3) Boot Loader

4) USB Driver

5) Vm

Supervisor Mode

Machine Mode

Machine Mode

Supervisor Mode

User Mode

13.

通过对除零结果返回值进行了设置与规定, 使得电路进行简化。

无符号除法返回的 $|11\dots1|$ 是无符号除法返回的自然值, 表示最大的无符号数, 也是简单无符号除法实现的自然结果。

· 有符号除法经常使用无符号除法电路实现, 指定相同的溢出结果可以简化硬件。

fflags 的各位: NV \rightarrow Invalid Operation 无效操作

DZ \rightarrow Divide by zero 除以 0

OF \rightarrow Overflow 溢出。

UF \rightarrow Underflow 下溢

NX \rightarrow Inexact 精度异常

fflags 被置位不会使处理器陷入系统调用。