

3.解: 1) nop : addi zero, zero, 0

2) ret : jalr x0, x1, 0

3) call offset : jal label

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

5) rdcycle rd : retime to
csrrwi cycle, to
addi x1, zero, 0

6) sext.w rd, rs : slli rd, rs, 0
slli rd, rd, 0

7.解: 1) sub t3, t0, t1
mv t4, t2

2) add t0, t1, t2
bltu t0, t1, overflow

3) X86 架构:

X86 架构中, 用于检测加法溢出的指令是 jno/jnc 指令, 分别代表“跳转不溢出”和“跳转不溢出”

有符号数的情况下溢出时, CPU 将根据 OF 标志位的值设置 jno 指令的跳转码, 该指令将跳转到指定的目标地址。
对于无符号数, 检测加法是否溢出的指令为 jnc/jnc, CPU 会根据 CF 标志位的值设置 jnc 指令的跳转码, 该指令将跳转到指定的目标地址。

ARM 架构:

在 ARM 架构中, 用于检测加法溢出的指令是四种“条件码移位指令”(CMP, CMN, TST 和 TEQ), 它们中的信息一般都可以用来比较两个操作数。

当使用 CMP 或 CMN 指令进行有符号数的比较时, V 标志位被设置为 1 表示运算结果溢出, 否则它将被清除为 0。
当使用 TST 或 TEQ 指令进行无符号数比较时, C 标志位被设置为 1 表示运算结果溢出, 否则它将被清除为 0。

8.解: 1)

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

除数为 0 会引起 RISC-V 异常, 这样设计能保证系统的可靠性和安全性, 防止因除数为 0 引起的崩溃或错误破坏系统的稳定运行。

- 2) NV: 操作数错误, 表示某个操作数不存在或不能代表一个实际数值。
DZ: 破坏性, 表示浮点运算结果非法, 因为结果的指数太大而不能表示。
OF: 上溢, 表示某个操作的结果太大, 超出了浮点数的范围。
UF: 下溢, 表示某个操作的结果太小, 超出了浮点数的范围。
NX: 未对齐, 表示浮点运算产生的结果是近似的, 即数值上可能存在一些误差。
fflags 被置位不会导致处理器陷入系统调用。

3) X86 架构:

X86 指令集的除法指令在执行时会判断除数是否为 0, 如果除数为 0, 则会触发一个称为 Divide Error 的异常。

Divide Error 异常用于表示在执行除法指令时除数为 0 的情况, 处理器将生成异常类型为 #DE 的异常。

当 Divide Error 异常发生时, 程序可以通过异常处理程序来恢复错误状态。

ARM 架构:

无符号除法中除数为 0 会发生 Undefined Instruction 异常, 该异常的异常代码为 0x0000000b, 表示未定义操作。

对于浮点除法, 当除数有时会引起浮点异常, 如下所示:

浮点除以 0: “正无穷大 (rm)”或“负无穷大 (rm)”是一个无穷数, 超出了浮点表示范围。

在 ARM 架构中, 发生异常时程序可以通过异常处理程序来恢复错误状态。

12.解: 1) M 等级

2) M 等级

3) S 等级

4) U 等级

5) U 等级

13. 解:

```

addi t0, A, 0
addi t1, B, 0
addi t2, C, 0
addi t3, zero, 0
lw t4, 0(t1)
mult t4, t2
mflo t5
sw t5, 0(t0)
Loop:
addi t3, t3, 1
bge t3, 100, Exit
addi t1, t1, 4
lw t4, 0(t1)
mult t4, t2
mflo t5
addi t0, t0, 4
sw t5, 0(t0)
j Loop
Exit:
lw t6, 0(t0)
jr ra

```

14. 解:

```

addi t0, zero, a
addi t1, zero, b
addi t2, zero, c
slt t3, t1, t0
beqz t3, Else
add t4, t0, t1
addi t5, t4, 0
sw t5, 0(t2)
j End
Else:
sub t4, t0, t1
addi t5, t4, 0
sw t5, 0(t2)
j End
End:

```

15. 解:
addi t2, t0, 0
sw t2, 0(t0)
addi t3, t1, 0
addi t0, t0, 4
sw t3, 0(t0)
slli t4, t3, 2
add t5, t0, t4
sw t3, 0(t5)

16. 解:
lw t2, 0(t0)
lw t3, 0(t1)
sw t3, 0(t0)
sw t2, 0(t1)
jr ra

17. 解: 计算 2^{30}