

T3

1) add x0, x0, 0

2) jalr x0, ra, 0

3) auipc x6, offset[31:12]
jalr ra, x6, offset[11:0]

4) ADDI rd, rs, 0

5)

6) slli rd, rs, 32
srai rd, rd, 32

T7

1) slti t3, t2, 0
slt t4, t3, t1

2) add t0, t1, t2
sltu t0, t0, t1
bnez t0, overflow

3) x86与ARM分别使用标志寄存器中的OF与条件码寄存器中的V来标志溢出

T8

1) $2^{64}-1, x, -1, x$

不会抛出异常, 因为除0后, 结果为全1, 则
若为有符号数则为-1, 若无符号数则为 $2^{64}-1$, 且
余数就是被除数, 这增强了 RISC-V 的处理
特殊情况能力

2) $\sim x$: 非精确异常

UF: 下溢异常

UF: 上溢异常

DZ: 除0异常

INV: 无效操作数异常

不会

3) x86下, 除法指令执行时会检查除数是否为0, 若是会
触发除数错误异常并中断程序, ARM下, 在进行 UOIV 与
 SDIV 等操作时, 若除数为0, 会抛出除以0错误异常

T12 1) 最高特权等级 Ring(0)

2) Ring(0)

3) Ring(1) 或 Ring(2)

4) Ring(1) 或 Ring(2)

5) Ring(3)

T13

rec/mymul:

start: addi sp, sp, -32

sd ra, 24(sp)

sd s0, 16(sp)

addi s0, sp, 32

addi a3, x0, 0 # a3 = 2

addi a4, x0, 99

mv a0, t0 # a0 = A

mv a1, t1 # a1 = B

ld a5, 0(t2) # a5 = C (1A32)

loop: beq a3, a4, end

slli a6, a3, 2

add a7, a0, a6 # a7 = &A[i]

add t3, a1, a6 # t3 = &B[i]

ld t4, 0(t3) # t4 = B[i] (1A32)

mulw t4, t4, a5 # t4 = B[i] * C (1A32)

sd t4, 0(a7) # A[i] = B[i] * C (1A32)

addi a3, a3, 1 # i++

end: ld ra, 24(sp) ←

ld s0, 16(sp)

addi sp, sp, 32

ret

(插入一句) ld a0, 0(a0)

a0 = A[0]

T14 blt a1, a0, if
 subw a2, a0, a1
 j exit

if: addw a2, a0, a1

exit: # 返回

T15 sd t0, 0(t0) # p[0] = P
 li t1, 3 # a = 3
 sd t1, 4(t0) # p[1] = a
 slli t2, t1, 2
 addi t2, t0, t2
 sd t1, 0(t2) } # p[a] = a.

T16 addi sp, sp, -32
 sd ra, 24(sp)
 sd s0, 16(sp)
 addi s0, sp, 32
 ld t2, 0(t0) # t2 = *a (1个 32位)
 ld t3, 0(t1) # t3 = *b (1个 32位)
 sd t3, 0(t0) # 交换
 sd t2, 0(t1)
 ld ra, 24(sp)

ld \$0, 16(\$p)

addi \$p, \$p, 32

ret

17 将a中的值变为 2^{30}

T17 将 2^{30} 放入 a_1 中