

3. ^{nop:} 1) `addi x0, x0, 0` 2) `ret: jalr x0, x1, 0`
 3) `call offset: suipc x6, offset[31:12]`
 `jalr x1, x6, offset[11:0]`
 4) `mv rd, rs: addi rd, rs, 0`
 5) `rdcycle rd: csrrs rd, cycle, x0`
 6) `sxt.w rd, rs: addiw rd, rs, 0`

7. 1) `slti t3, t2, 0`
 `slti t4, t0, t1`
 2) `add t0, t1, t2`
 `bltu t0, t1, overflow`

3) 首先①负数+正数一定不会产生溢出
 对于有符号数而言: ②两正数相加得到负数或两负数相加得到正数, 则溢出
 ③最高有效位产生进位, 若结果为正数则溢出, 若为负数则不溢出

无符号数: 当结果小于两个加数时溢出。

8. (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	0xffffffff = $2^{63} - 1$	x	0xffffffff = -1	x

所以不会引起异常

原因: 若会引发异常, 则该异常会在大部分执行环境中会导致一个自陷,
 而为了改进这一情况, 会在原法操作后加一条分支指令。

(2) NV: 非法操作; DZ: 除以0; OF: 上溢; UF: 下溢; NX: 不精确
 fflags被置位不会使处理器陷入系统调用, 而是会设置异常标志, 需
 要软件明确对该标志进行检查。

(3) 会产生一个内中断, 暂停当下程序, 跳转到对应的中断处理
 程序, 执行完后再返回之前的程序执行。

12. (1) M级, 也可选 ~~S级~~, 并在M级下运行

(2) M级

(3) M级 ~~或 S级~~

(4) M级, 要直接访问硬件资源

(5) U级

13. li t3, 0

li t4, 100

lw a5, 0(t2)

j part1

part1: blt t3, t4, part2

j end

part2: addi t3, t3, 1 sll t4, t3, 2

add t5, t1, t4

lw a6, 0(t5)

add t5, t0, t4

mul a7, a6, a5

sw a7, 0(t5)

j part2

end lw a0, 0(t0)

ret

14. part1: blt a1, a0, part2

add a2, a1, a0

j end

part2: sub a2, a0, a1

j end

end:

15. sw t0, 0(t0)

li a1, 3

sw t1, 4(t0)

sll ~~t2~~, t1, 2

add t2, t0, t2

sw t1, 0(t2)

16. lw a1, 0(t0)

lw ~~a2~~, 0(t1)

sw a2, 0(t0)

sw a1, 0(t1)

ret

17. int a;

int b = 1

int c = 30

for (a=0; a<30; a++)

b = b * 2

return

实现的是计算 2^{30} 。