3. (1) addi x0, x0, 0

(2) jalr x0, x1, 0

(3) auipc x6, offset[31:12]

jalr x1, x6, offset[11:0]

(4) addi rd, rs, 0

(5) csrrs rd, cycles, x0

(6) addiw rd, rs, 0

7. (1) slt t3, t1, x0

slt t4, t0, t2

(2) add t0, t1, t2

blt t0, t1, overflow

(3) 在x86指令集架构中，可以用指令"jo"(jump if overflow)或"jno"(jump if not overflow)来检测加法溢出；在ARM指令，可以使用adds(add and set flags)指令来完成相加并设置标志位。如果加法运算造成了溢出，那么状态寄存器中的V标志位会被设置为1，表示发生了溢出。

8. (1) 2^{XLEN-1} X -1 X -1

这样设置返回值可以使得完成除法后易于跳转其它的分支指令，或简化硬件设计等

(2) NV: Invalid Operation：无效操作

DZ: Divide by zero：除以0

OF: Overflow：溢出

UF: Underflow: 下溢

NX: Inexact：不精确

不会陷入系统调用

(3) 在x86指令集架构中，会抛出异常类型为"除法错误"；在ARM指令集架构中会抛出"除以零"异常。

12: (1) S (2) M (3) S (4) S (5) U

13:
```
vecMul:
        addi    sp, sp, -16
        sw      ra, 0(sp)
        sw      s0, 4(sp)
        sw      t0, 8(sp)
        li      s0, 0
    Loop:
        lw      t3, 0(t1)
        mul     t0, t3, t2
        sw      t0, 0(t0)
        addi    t0, t0, 4
        addi    t1, t1, 4
        addi    s0, s0, 1
        blt     s0, 100, Loop
        lw      t0, -4(t0)
        lw      ra, 0(sp)
        lw      s0, 4(sp)
        lw      t0, 8(sp)
        add     sp, sp, 16
        ret
```

14:
```
        bgt     a0, a1, funct1
        sub     a2, a0, a1
        j       next
funct1:
        add     a2, a0, a1
    next:
```

15: li t1, 3
sw t1, 4(t0)
add t0, t0, t1
sw t1, 0(t0)

16. swap:
add t2, t0, x0
mv t0, t1
mv t1, t2

17 实现了计算 2^{30} 的功能