3. (1) nop: addi x0, x0, 0　　(2) ret: jalr, x0, x1, 0

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

jalr x1, x6 offset[11:0]

(4) mv rd, rs: addi rd, rs, 0　(5) rdcycle rd: rdtime rd.

(6) sext.w rd, rs: addiw rd, rs, 0

7. (1) slti t3, t2, 0

slt t4, t0, t1

(2) bltu t0, t1, overflow

(3) X86体系下，CPU状态寄存OF位，若两正数相加为负数或两负数相加为正数则OF位为"1"表示溢出，否则为"0"表示不溢出。

ARM体系下，设置V值，与OF值相同。

8. (1) OP: DIVU　REMU　DIV　REM

rd: $2^{XLEN}-1$　x　-1　x

(2)

NV: Invalid Operation

DZ: Divide by Zero　　被置位不会使处理器陷入异常调用

OF: Overflow

UF: Underflow

NX: Inexact.

(3) X86: 触发#DE异常，并将控制转移到异常处理程序(由操作系统提供)

ARM: { SDIV: 返回特殊值 0x80000000 最小负数
　　　UDIV 返回 0.

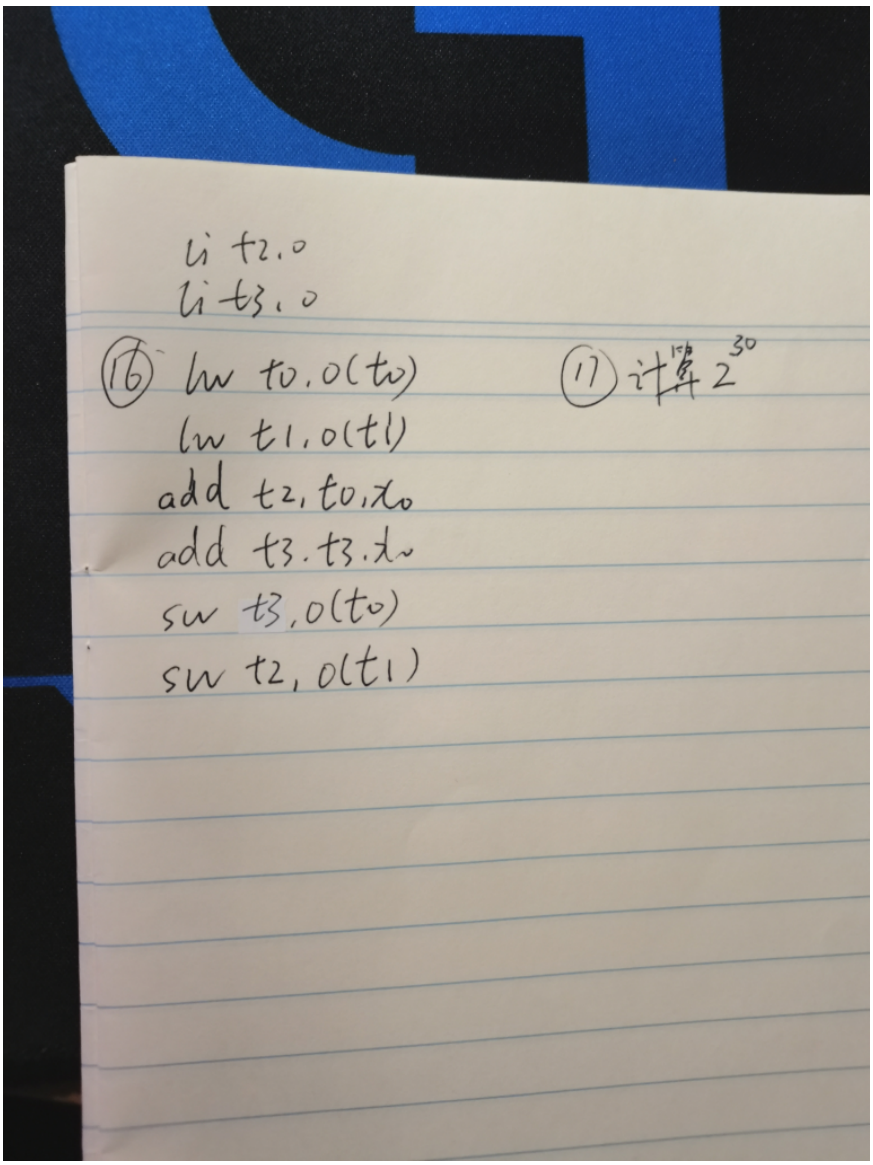
12 :Linux kernel: 管理员或机器模式
BootROM: 机器模式
Boot Loader: 管理员或机器模式
USB Driver: 管理员或机器模式
vim: 用户模式

(13)
```
    li t3,0
    li t4,0
loop: beq t3,100,done
    lw t4,0(t1) / lw t5,0(t2)
    mul t4,t4,t5
    sw t4,0(t0)
    addi t0,t0,4
    addi t1,t1,4
    addi t3,t3,1
    j loop
done: addi t0,t0,-400
    lw a0,0(t0)
    jr ra
```

(14)
```
    blt a1,a0,else
    sub a2,a0,a1
else: add a2,a1,a0
```

(15)
```
    addi t0,t0,4
    sw t0,0(t1)
    addi t0,t0,8
    sw t0,0(t1)
```

(13)
```
    li t3,0
    li t4,0
loop: beq t3,100,done
    slli t4,t3,2
    slli t6,t3,2 / lw t5,0(t2)
    add t4,t4,t1
    add t6,t6,t0
    mul t4,t4,t5
    sw t4,0(t6)
    addi t3,t3,1
    j loop
done: lw a0,
    jr ra.
```


li t2,0
li t3,0
⑯ lw t0,0(t0)
lw t1,0(t1)
add t2,t0,t0
add t3,t3,t3
sw t3,0(t0)
sw t2,0(t1)
⑰ 计算 2^{30}