3/21 3.(1) nop：空指令，等价指令：addi x_0, x_0, 0.

(2) ret：返回指令 等价指令：jalr x_0, 0(x_1).

(3) call offset：调用函数指令 等价指令：jalr x_1, offset[11:0](x_1) / auipc x_1, offset[31:12]+offset[11].

(4) mv rd, rs：令rd=rs的指令 等价指令：addi rd, rs, 0.

(5) rdcycle rd：读取时钟周期数指令 等价指令：csrrs rd, cycle, x_0.

(6) sext.w rd, rs：符号扩展指令 等价指令：addiw rd, rs, 0.

7.(1) slti t_3, t_2, 0 、 slt t_4, t_0, t_1 "分别比较加数t_2与0的关系，和结果与被加数t_1的关系".

(2) t_1和t_2均为无符号时，只需比较结果t_0与其一同的大小关系即可，若$t_0 < t_1$，则说明溢出，否则无溢出。

指令序列为：add t_0, t_1, t_2 、bltu t_0, t_1, overflow.

(3) ①x86架构：(a)采用一位符号位，用与或非门进行逻辑运算，判断是否溢出

(b)采用1位符号位，根据符号位的进位与最高数值位的进位是否相同判断是否溢出

(c)采用双符号位，正数符号位为00，负数为11，加法后若两位符号位不同，则溢出，否则不溢出。

②ARM架构，通过CPSR(当前程序状态寄存器)中的第28位V(共32位，从低至高)反映其溢出状态。

③MIPS架构，类似于RISC-V，通过指令触发跳转的方式产生溢出信号。

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	(XLEN代表数X的位数长度，即指令为该位数中最大数)
op rd, rs1, rs2	X	0	$2^{XLEN}-1$	X	-1	X	

(2) NV位：代表是否有无效操作 DZ：代表是否除数为0.

OF位：代是是否有向上溢出 UF位：代表是否有向下溢出 NX：代表是否有不精确产生.

ff0标志位置位不会使处理器陷入系统调用，但可由程序员或操作系统中的监控程序检查并处理

(3) ①x86架构：整型除法中除数为0会引起"除0异常"，可通过设置异常处理处理程序来解决这种特殊情况的产生，在浮点数除法，除数为0会将FPU状态寄存器标志位置位，从而提醒异常产生。

②ARM架构：会产生报错并进入异常处理过程分析，最终终止程序运行并显示报告

补(1)设计原因：这样的设计可以避免因为除数为0而产生异常结果，在无符号数除法中 $\frac{x}{0} = +\infty$ (即$2^{XLEN}-1$)最大数是一种自然的操作，全部置1也可降低硬件复杂度，同时将控制权转移给异常处理，符合系统通常。

12. (1) Linux kernel：管理员模式(S) (2) BootROM：机器模式(M) (3) BootLoader：机器模式(M)

(4) USB Driver：管理员模式(S) (5) vim：用户模式(U)

13. vecMul: addi $a_3, x_0, 0$

addi $a_4, x_0, 100$

Loop: bge a_3, a_4, exit

slli $a_5, a_3, 2$

add a_6, a_5, t_1.

add a_7, a_5, t_0

lw $a_1, 0(a_6)$

mul a_1, a_1, t_2

sw $a_1, 0(a_7)$

addi $a_3, a_3, 1$

j Loop

exit: lw $t_3, 0(t_0)$

mv a_0, t_3

ret.

14. blt a_0, a_1, else

beq a_0, a_1, else

add, a_2, a_0, a_1

else: sub a_2, a_0, a_1.

15. sw $t_0, 0(t_0)$

addi $t_1, x_0, 3$

addi t_2, t_0, 4.

sw $t_1, 0(t_2)$

slli $t_1, t_1, 2$

add t_2, t_0, t_1

sw $t_1, 0(t_2)$.

16. swap: lw $t_2, 0(t_0)$

lw $t_3, 0(t_1)$

sw $t_3, 0(t_0)$

sw $t_2, 0(t_1)$

ret.

17. C语言：int a=0, b=1, c=30;

while(a!=c) {

b=b*2;

a=a+1;

}

∴该代码功能为：计算2^{30}的值.