3. 1) nop: addi x_0, x_0, 0.

2) ret: jr ra

3) call offset:

auipc t1, offset[31:12]

addi t1, t1, offset[11:0]

jalr ra, t1, 0

4) mv rd, rs: addi rd, rs, 0

5) rdcycle rd: csrrs rd, cycle, x_0

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

7. 1) srai $t_3, t_1, 32$
srai $t_4, t_0, 32$
2) add t_0, t_1, t_2
~~slt t_3, t_0, t_1~~ blt t_0, t_1, overflow
~~bne t_3, x_0, overflow~~
3) 在x86架构中，加法指令会设置CF（进位标志）和OF（溢出标志）标志位，可以通过检查这两个标志位来确定是否发生溢出。
在ARM架构中，有专门的ADC（带进位加法）和SBC（带进法减法）指令，它们会设置CF和VF（溢出标志）标志法，可以通过检查这两个标志位来确定是否发生溢出。

~~8. DIVU指令、REMU指令在除数为零时不会抛出异常，而是将结果设置为最大无符号数~~

8. 1)

指令	rs1	rs2	rd值 OP=DIVU	OP=REMU	OP=DIV	OP=REM
Op, rd, rs1, rs2	x	0	2^n-1	2^n-1	异常	异常

对于无符号数指令DIVU与REMU，将结果设置为2^n-1，即最大的无符号数而不是0，是因为0可能被认为是合法的结果，而2^n-1能清楚地表明结果无效。

而对于有符号数指令DIV和REM，若和无符号数指令一样设为最大或最小的无符号数，则当最小的数-2^n除以-1时，将溢出异常。因此采用这样的设计。

2) ① NV：当执行无效的浮点操作时，该标志位将被置位。

② DZ：当执行浮点除法且除法的数为0时，该标志位将被置位。

③ OF：当执行浮点运算时，除数为0时，该标志将被置位。

④ UF：当执行浮点运算时结果的精度超出了浮点数能够表示的范围时，该标志位将被置位。

⑤ NX：当浮点运算的结果无法完全精确地表示为浮点数时，该标志位将被置位。

当fflags被置位时，处理器不会陷入系统调用，但是程序员可以通过使用浮点异常处理机制来处理这些异常

3) x86、ARM等会在除数为0时抛出一个异常，称为"除数为0异常"/"浮点除0异常"，并跳转到异常处理程序，并进行相应处理。

12、1) ~~监管理员模式或~~机器模式

2) 机器模式

~~3) 用户模式~~ 管理员模式

4) 用户模式

5) 用户模式

13、
```
li t3, 0
li t4, 100

loop:
bge t3, t4, end
lw t6, 0(t1)
lw t7, 0(t2)
mul t5, t6, t7
sw t5, 0(t0)
addi t3, t3, 1

addi ~~t4, t4,~~ t0, t0, 4
addi t1, t1, 4

j loop
end: ~~jr~~ lw a0, 0(t0)
jr ra
```

14、
```
compare:
blt a0, a1, ab
j ba
ab:
add a2, a0, a1
ba:
sub ~~add~~ a2, a0, a1
```

~~15、li t1, 3~~

15、
```
lw t0, 0(t0)
li t1, 3
addi t0, t0, 4
lw t1, 0(t0)
addi t0, t1, t0, 8
lw t1, 0(t0)
```

16. start: addi sp, sp, -16
sw ra, 0(sp)
sw t_0, 4(sp)
sw t_1, 8(sp)

lw t_2, 0(t_0)

lw t_0, 0(t_1)

lw t_1, 0(t_2)

end: lw ra, 0(sp)
~~lw s0~~
addi sp, sp, 16
ret

17. 求 2^{30} 功能：