RISC-V 汇编中存在许多伪指令，它们一般是具有特殊操作数的基本指令或指令组合。请写出与以下伪指令等价的基本指令或指令组合。

1) nop
2) ret
3) call offset
4) mv rd,rs
5) rdcycle rd
6) sext.w rd,rs

1) addi x0, x0, 0
2) jalr x0, x1, 0
3) auipc x6, offset[31:12]
   jalr x1, x6, offest[11:0]
4) addi rd, rs, 0
5) csrrs rd, cycle, x0
6) addiw rd, rs, 0

7. RISC-V 标准指令集并未为加法指令的溢出引入专用的标志位，因此通常需要额外的指令以检查加法溢出。

1) 考虑如下的指令序列：

add t0,t1,t2

_____

_____

bne t3,t4,overflow

若 t0 和 t1 都是有符号数，请在横线处填入正确的指令，使得当 t0 和 t1 的加法发生溢出时，控制流可以正确跳转到 overflow 位置。（请勿使用除 t0~t4 以外的任何寄存器）

2) 当 t0 和 t1 都是无符号数时，请给出尽量简单的检测 add t0,t1,t2 指令加法是否溢出的指令序列。

3) 调研其他指令集架构（如 x86、ARM 等）是如何检测加法溢出的。

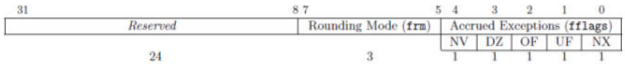
1) slti t3, t2, 0
   slt t4, t0, t1
2) addu t0, t1, t2
   bltu t0, t1, overflow
3) ARM体系结构中，通过CPSR的状态寄存器反应当前指令的溢出状态
   MIPS 通过指令触发中断的方式产生溢出信号，通知处理器
   对于x86体系有各种标志位，标志位的结果由硬件直接给出，以溢出由硬件判断

8. 阅读 RISC-V 规范以了解 RISC-V 对除数为 0 的除法指令的处理方法，回答以下问题。

1) 对整型除法，填写下表。整型除法中除数为 0 是否会引起 RISC-V 抛出异常？试分析为什么采取这样的设计。

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				

2) 对浮点除法，除数为 0 将会引起 fcsr 控制寄存器中的相关标志位被置位。下图给出了 fcsr 的构成，请说明 fflags 的各位分别代表什么含义。fflags 被置位是否会使得处理器陷入系统调用？



3) 调研其他指令集架构（如 x86、ARM 等）是如何处理除数为 0 的。

1) 2^{XLEN-1} . X , -1 , X

这些异常会在为数执行环境中引发异常，但这是ISA中唯一的算术陷阱。

如果语言标准要求除以0异常必然立即引发控制流的变化，则只需要向每个除0操作添加一个分支指令，并且这个分支指令可以在除0之后预测，从而减少增加运行时开销

2) NV：无效的操作　　DZ：除数为0.　　OF：溢出

UF：下溢 underflow　NX：不精确的

浮点异常会设置标志并写入默认值，但不会引发陷阱.

3) x86 除数为0. 认为是除溢出，引起0号中断

在ARM中强制进入管理模式，到ARM状态，跳转到绝对地址 PC=0x00000000 出执行

禁止IRQ和FIQ中断，复位后 CPSR中最后4位 0011，进入管理模式，执行操作系统程序.

之后切换到用户模式，开始执行正常的用户程序.

12. 写出以下程序在 RISC-V 中应当处于的特权等级。

1) Linux Kernel
2) BootROM
3) BootLoader
4) USB Driver
5) vim

1) 机器模式 3
2) 机器模式 3
3) 机器模式 3
4) 管理员模式 1
5) 用户模式 0

13. 写出实现以下 C 程序的 32 位 RISC-V 汇编代码。假设 A 和 B 的起始地址存放于寄存器 t0 和 t1，C 的地址存放于寄存器 t2。

```
int vecMul(int *A, int *B, C){
    for(int i = 0;i < 100;++i){
    A[i] = B[i] * C;
    }
    return A[0];
}
```

```
        addi  sp, sp, -32
        sd    ra, 24(sp)
        sd    s0, 16(sp)
        addi  s0, sp, 32
        add   x11, x0, x0
        addi  x12, x0, 100
Loop:   bge   x11, x12, exit
        sll   x13, x11, 2
        mv    x14, x13
        add   x13, x13, t0
        add   x14, x14, t1
        lw    x14, 0(x14)
        mul   x13, x14, t2
        sw    x13, 0(x13)
        addi  x11, x11, 1
        j     Loop
exit:   add   x13, x0, t0
        lw    x13, 0(x13)
        mv    t0, x13
        ld    ra, 24(sp)
        ld    s0, 16(sp)
        addi  sp, sp, 32
        ret
```

14. 写出实现以下 C 程序的 32 位 RISC-V 汇编代码。假设 a、b 和 c 分别对应寄存器 a0、a1 和 a2。

```
int a, b, c;
if(a > b){
    c = a + b;
}
else{
    c = a – b;
}
```

```
bge  a1, a0, lf
add  a2, a0, a1
2f
1: sub  a2, a0, a1
2: # next codes
```

15. 写出实现以下 C 程序的 32 位 RISC-V 汇编代码。假设指针 p 已经通过程序 int *p = (int *) malloc(4*sizeof(int))得到，且 p 存放于 t0 中，a 存放于 t1 中。

```
p[0] = p;
int a = 3;
p[1] = a;
p[a] = a;
```

```
sw   t0, 0(t0)
addi t1, x0, 3
sw   t1, 4(t0)
sw   t1, 12(t0)
```

16. 写出实现以下 C 程序的 32 位 RISC-V 汇编代码。假设指针 a 和 b 分别存放于 t0 和 t1 中。

```
void swap(int *a, int *b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
    return;
}
```

```
addi sp, sp, -32
sd   ra, 24(sp)
sd   s0, 16(sp)
addi s0, sp, 32
lw   t2, 0(t0)
add  t2, t2, x0   #t2 -> tmp
lw   t3, 0(t1)
sw   t3, 0(t0)
sw   t2, 0(t1)
```

17. 解释以下 RISC-V 汇编代码实现的功能。

```
        addi a0,x0,0
        addi a1,x0,1
        addi a2,x0,30
loop:   beq a0,a2,done
        slli a1,a1,1
        addi a0,a0,1
        j loop
done:   # exit code
```

a0从0开始. a1初始值为1. 每次循环a1左移1位. a0加1, 直到a0为30. 相当于a1乘上2^{30}.