3题解：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　csrrs rd, cycle, x0

6) sext.w rd,rs　addiw rd, rs, 0

4题解：1) slti t3, t2, 0

slt t4, t0, t1

2) add t0, t1, t2

bltu t0, t1, overflow

3) arm架构利用CPSR的状态寄存器反映当前指令的溢出状态

对于无符号数，C表示进位或非借位标志，加法时C=1或减法时C=0溢出

对于有符号数，V表示溢出标志，V=1表示溢出

x86架构利用 cf, of 寄存器反映溢出状态

对于无符号数，cf表示进位或非借位标志，加法时cf=1或减法时cf=0溢出

对于有符号数，of表示溢出，of=1溢出

5题解：1)

DIVU	REMU	DIV	REM	
$2^{xLEN}-1$	X	-1	X	不会产生异常

避免在处理器中引发异常，同时简化硬件设计。因为产生异常时，会增加处理器设计难度。

2) NV 无效操作　DZ 除以0，　OF 结果溢出

UF 结果下溢　NX 不精确

不会使处理器陷入系统调用，会产生涉点异常，引发异常处理例程，根据异常类型进行处理，如跳到异常处理程序，恢复寄存器状态等；

3) x86中 会触发异常"divide error"，处理器产生异常处理程序，中止当前程序，保存断点信息，执行异常处理程序，恢复现场，继续执行

ARM 产生"divide by zero"异常，处理器产生异常处理程序，程序地址由异常向量表中对应指令获得

12. 解: 1) Linux Kernel　　Kernel
2) BootRom　　Kernel
3) BootLoader　　Kernel
4) USBDriver　　Device Driver
5) vim　　Application

13解: vecMul:

start:

```
addi  sp,sp,-32
sd    s0,24(sp)
sd    s0,16(sp)
addi  s0,sp,32
sw    t0,-16(s0)
sw    t1,-20(s0)
sw    t2,-24(s0)
addi  a5,x0,0
addi  a6,x0,100
```

part1:

```
lw  t0, -16(s0)   # &A[0]
lw  t1, -20(s0)   # &B[0]
lw  a4, (t2)      # C
bge a5, a6, end
sll a7, a5, 2
add t1, t1, a7    # &B[i]
lw  a3, 0(t1)
mul a3, a3, a4
add t0, t0, a7    # &A[i]
sw  a3, t0
addi a5, a5, 1    # ++i
j   part1
```

end:

```
mv  a0, t0
ld  ra, 24(sp)
ld  s0, 16(sp)
addi sp, sp, 32
ret
```

part1:

```
    bge b, a part2
    add a2, a0, a1
part2:
```

```
sub a2, a0, a1
```

15题：

```
sw t0, -20(s0)
sw t0, (t0)
addi t1, x0, 3
addi t0, t0, 4
sw t1, (t0)
addi t0, t0, 8
sw t1, (t0)
```

16题：swap：

```
start:
    addi sp, sp, -32
    sd ra, 24(sp)
    sd s0, 16(sp)
    addi s0, sp, 32
    sw t0, -20(s0)
    sw t1, -24(s0)

part1:
    lw t0, -20(s0)
    lw t1, -24(s0)
    lw a4, 0(t0)   # *a
    lw a5, 0(t1)   # *b
```

```
    add   a6, x0, a4
    add   a4, x0, a5
    add   a5, x0, a6
    sw    a4, t0
    sw    a5, t1

end:
    ld    ra, 24(sp)
    ld    s0, 16(sp)
    addi  sp, sp, 32
    ret
```

门级：将a0置为0，a1置为1，a2置为30

当a0<a2，a1乘2，a0自增1

当a0=a2时，退出循环

用来计算 2^{30}