作业04

3. 1) nop ⇒ addi x0, x0, 0

2) ret ⇒ jalr x0, x1, 0

3) call offset ⇒ auipc x1, offset[31:12]

jalr x1, x1, 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

7. 1) sub t3, t0, t1

~~sub t4, t0, t2~~ mv t4, t2

2) add t0, t1, t2

bltu t0, t1, overflow

3) ARM中通过CPSR的状态寄存器反映当前指令的溢出

x86中执行算术指令时，条件代码将设置在CPU状态字中，使用条件分支指令JO可在溢出时跳转

8. 1) $2^{64}-1$ x -1 x

除数为0不会引起异常，可能是为了提高程序的效率和性能，认出现除数为0的可能性很小，如果存在这样的可能，自己定义一个异常即可

2) 4: NV invalid operation fflags被置位是不会使处理器陷入系统调用，而会一直保持累积.

3: DZ divide by zero

2: OF overflow

1: UF underflow

0: NX inexact，不精确

3) ARM中先将除数与0进行比较，除数不为0时才进行除法运算，若除数为0抛出异常

x86中除数为0也会抛出异常

12. 1) Linux Kernel S

2) BootROM M

3) BootLoader M

4) USB Driver U

5) vim U

13. vecmul:

```
addi  sp,sp,-32
sw    ra,28(sp)
sw    s0, 24(sp)
addi  s0,sp,32
sw    t0, 20(sp)
sw    t1, 16(sp)
sw    t2, 12(sp)
lw    a2, 20(sp)  #A
```

```
    lw   a3, 16(sp)   # *B
    lw   a4, 12(sp)   # C
    add  a5, x0, x0   # i=0
    addi a6, x0, 100  # a6=100
loop:
    bge  a5, a6, end  # lw t1, 0(a2)  # t1=A[i]
    mul  a2, a3, a4   # lw t2, 0(a3)  # t2=B[i]
                      # mul t1, t2, a4
    addi a2, a2, 4    # *A+=4
    addi a3, a3, 4    # *B+=4
    addi a5, a5, 1    # i=i+1
    j    loop
end: lw   a1, 20(sp)  # a1=*A
    lw   a0, 0(a1)    # a0=A[0]
    lw   ra, 28(sp)
    lw   s0, 24(sp)
    addi sp, sp, 32
    ret
```

14.

```
    bge  a1, a0, elf  # if(a>b)
    add  a2, a0, a1   # c=a+b
elf:
    sub, a2, a0, a1   # c=a-b
```

15.
```
sw   t0, 0(t0)   # p[0] = p
addi t1, x0, 3   # a=3
sw   t1, 4(t0)   # p[1] = a
slli t2, t1, 2   # t2 = a*4
add  t2, t2, t1  # t2 = p+a*4
sw   t1, 0(t2)   # p[a] = a
```

16.
```
swap:
    addi  sp, sp, -32
    sw    ra, 28(sp)
    sw    s0, 24(sp)
    addi  s0, sp, 32
    lw    t2, 0(t0)   # t2 = *a
    lw    t3, 0(t1)   # t3 = *b
    sw    t3, 0(t0)   # *a = *b
    sw    t2, 0(t1)   # *b = *a
    lw    ra, 28(sp)
    lw    s0, 24(sp)
    addi  sp, sp, 32
    ret
```

17. 得到 $a1 = 2^{30}$