第五次作业

3. 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

7. 1) ☆ add t0, t1, t2
slti t3, t2, 0
slt t4, t0, t1
bne t3, t4, overflow

用结果的大小比较去实现

2) addu t0, t1, t2
bltu t0, t1, overflow

3) X86处理器中CF标志(对应于ARM架构C标志)指示当前计算结果是否为无符号整数溢出
X86处理器中OF标志)(对应于ARM架构V标志)指示当前计算结果是否为带符号整数溢出

8. 1) Op=DIVU时 rd值 0xffffffffffffffff
Op=REMU时 rd值 x
Op=DIV时 rd值 0xffffffffffffffff
Op=REM时 rd值 x
不会引起异常
避免因一个小错误导致整个程序停止,且通过近似无穷大和余数可作估计值

2) NX=1 产生非精确异常
UF=1 产生下溢异常
OF=1 产生上溢异常
DZ=1 产生除0异常
NV=1 产生无效操作数异常
} 等于0为正常

会使处理器陷入系统调用

3) arm架构有一标志位DZ，置为0时，则除0无异常；置为1时，除0抛出异常
X86架构也有对相应exception抛出错误的设置

12. 1) Linux Kernel 机器模式
2) BootROM 机器模式
3) BootLoader 机器模式
4) USB Driver 管理员模式
5) vim 用户模式

13
```
start:  addi  sp, sp, -32
        sd    ra, 24(sp)
        sd    s0, 16(sp)
        addi  s0, sp, 32
        addi  a0, x0, 0      # i=0
        addi  a1, x0, 100    # a1=100

Loop:   bge   a0, a1, return
        slli  a3, a0, 2
```

```
        add  a3, a3, t0
        add  a4, a2, t1
        lw   a5, 0(a4)
        lw   a6, 0(t2)
        mul  a5, a5, a6
        sw   a5, 0(a3)
        addi a0, a0, 1
        j Loop

return: lw   a0, 0(t0)
        ld   ra, 24(sp)
        ld   s0, 16(sp)
        addi sp, sp, 32
        ret
```

14.

```
        bge  a1, a0, else
        add  a2, a0, a1
        j end
else: sub a2, a0, a1
end:
```

15.
```
add  a0, t0, x0
sw   t0, 0(a0)
addi t1, x0, 3
addi a0, a0, 4
sw   t1, 0(a0)
slli a1, t1, 2
add  a2, t0, a1
sw   t1, 0(a2)
```

16
```
start: addi sp,sp,-32
       sd   ra,24(sp)
       sd   s0,16(sp)
       addi s0,sp,32
       lw   a0,0(t0)
       lw   a1,0(t1)
       sw   a1,0(t0)
       sw   a0,0(t1)
end:   ld   ra,24(sp)
       ld   s0,16(sp)
       addi sp,sp,32
       ret
```

17

```
addi  a0, x0, 0        a0=0
addi  a1, x0, 1        a1=1
addi  a2, x0, 30       a2=30
loop: beq  a0, a2, done    判断, 若 a0=a2 跳出
slli  a1, a1, 1        a1=a1×2
addi  a0, a0, 1        a0=a0+1
j loop                 跳回 loop
done: # exit code
```

```
for(a0=0; a0!=30; a0++)
{    a1=a1·2;}
```

功能：计算 2^{30} 的值