第五周

3. 1) addi x0, x0, 0

2) jalr x0, x1, 0

3) auipc x6, offset[31:12]

jalr x1, x6, offset[11:0]

4) addi rd, rs, 0

5) csrrs rd, cycle, x1

6) addiw rd, rs, x0

7. 1) slti t3, t2, 0

slt t0, t0, t1

2) bltu x5, x6, overflow

3) ARM中，通过CPSR的状态寄存器反映当前指令的溢出状态

MIPS中，通过指令触发中断的方式产生溢出信号，通知处理器

如：add rd, rs, rt

如果运算溢出，则会产生溢出异常，同时不保存结果

x86中，通过 OF 标志位检测溢出

8. 1) Op = DIVU时，rd值为 $2^{XLEN}-1$ （XLEN为除数和被除数位数）

Op = REMU时，rd值为 x

Op = DIV时，rd值为 -1

Op = REM时，rd值为 x

除数为0不会引起RISC-V执的出异常，因为触发异常在绝大多数执行环境里，会导致一个自陷。这将成为标准ISA中唯一的算术自陷，并且此时需要语言实现者与执行函数的自陷处理函数交互。但如果不触发异常，只需要在每个除法操作时增加

一条分支指令即可，且该分支指令通常不跳转，因此对性能影响很小。

2) NV — 非法操作　DZ — 除以0　OF — 上溢　UF — 下溢

NX — 不精确

fflags被置位不会陷入系统调用.

3) X86中，通过标志位判断中断，再执行相应的中断服务程序，处理完后返回，继续执行原程序。

12. 1) Machine

2) Machine

3) Machine

4) Supervisor

5) User

13. VecMul:

```
# assign x10=i
add     x10, x0, x0     # i=0
addi    x11, x0, 100    # x11=100
lw      x12, 0(t2)      # x12=C

Loop:
bge     x10, x11, exit
sll     x13, x10, 2     # x13=i*4
add     x14, x13, t0    # x14=&A[i]
add     x15, x13, t0    # x15=&B[i]
lw      x15, 0(x15)     # x15=B[i]
mul     x13, x15, x12   # x13=B[i]*C
sw      x13, 0(x14)     # A[i]=B[i]*C
addi    x10, x10, 1     # i++
j       Loop

exit:
lw      a0, 0(t0)       # a0=A[0]
ret
```

14. Part1:

```
        blt  a1, a0, Part2
        sub  a2, a0, a1
Part2:
        add  a2, a0, a1
```

15.

```
sw   t0, 0(t0)
li   t1, 3
sw   t1, 4(t0)
sll  a5, t1, 2
add  a5, a5, t0
sw   t1, 0(a5)
```

16. swap:

```
        lw   a4, 0(t0)   # a4 = *a
        lw   a5, 0(t1)   # a5 = *b
        sw   a5, 0(t0)   # *a = *b
        sw   a4, 0(t1)   # *b = *a
        ret
```

17. 计算 2^{30}