3/21 第二章

3. (1) addi x0,x0,0

(2) jalr x0, 0(x1)

(3) jal ra,offset

(4) add rd,rs,x0

(5) csrr rd,cycle

(6) addiw rd,rs,0

7. (1) slt t3,t0,t1

slt t4,t0,t2

(2) add t0,t1,t2

bltu t0,t1, overflow

(3) x86指令集通过标志寄存器中的进位标志CF(Carry Flag)和溢出标志OF(Overflow Flag)来检测加法溢出。加法指令执行时，CF标志将设置为1如果有进位，OF标志将设置为1如果加法结果无法用相应的数据类型表示。

ARM指令集也用类似标志来检测加法溢出，对于32位数，CPSR进位标志C和溢出标志V分别用于检测加法的进位和溢出。对64位数，这些标志存储在64位程序状态寄存器的对应位中

8. (1) Op=DIVU: rd= 0xffffffffffffffff

Op=REMU: rd= x

Op=DIV: rd= 0x ffffffffffffffff

Op=REM: rd=x

(2) NX-非精确异常：NX=1，产生非精确异常；UF-下溢异常：UF=1，产生下溢异常；

OF-上溢异常：OF=1，产生上溢异常；DZ-除0异常：DZ=1，产生除0异常；

NV-无效操作数异常：NV=1，产生无效操作数异常。

fflags被置位不会使处理器陷入异常。

(3) 在x86架构中，处理器会设置相应的异常标志位，即#DE，并且通过调用中断向量0来触发异常处理程序。

在ARM架构中，除以0异常会通过协处理器CP15来设置相应的异常标志位，并通过中断向量表跳转到异常处理程序中执行相应操作。

12. (1) S (2) M (3) M 与S (4) S (5) U

Bootloader在M模式下执行一些启动操作，加载操作系统内核时，要切换到S模式。

13.
```
vecMul:
    li a2,0
    j loop
Loop:
    blt a2,t0,99,a2, exit
    sll t0, a2,2
    sll t3, a2,2
    add t3,t3,t1
    lw t0,0(t0).
    lw t3, 0(t3)
    mul t0, t2, t1
    addi a2,a2,1
    j loop
exit:
    mv a0, 0(t0).
    ret
```

14.
```
    blt a1, a0, if
    sub a2, a0,a1
if:
    add a2, a0,a1
```

15.
```
sw t0,0(t0)
li t3,3
addi t0,t0,4
sw t1,0(t0)
addi t0,t0,8
sw t1, 0(t0)
```

16.
```
swap:
    mv t2,t0
    add t0,t1,x0
    add t1, t2,x0
```

17. 令a0=0, a1=1, a2=30.

如果a0与a2相等，跳转至done，执行exit命令.

如果a0与a2不相等，a1左移一位，即乘以2，并将a0每次加1，一直如此循环，直到a0增加至30与a2相等跳出循环，即循环30次.