3. 1) nop : addi $x_0$, $x_0$, 0

2) ret : jalr $x_0$, $x_1$, 0

3) call offset : auipc $x_1$, offset[31:12]
jalr $x_1$, $x_1$, offset[11:0]

4) mv rd, rs : addi rd, rs, 0

5) rdcycle rd : csrrs rd, cycle, $x_0$

6) sext.w rd, rs : slli rd, rs, 32
srli rd, rd, 32

7. 1) srai $t_2$, $t_0$, 31
srai $t_4$, $t_1$, 31

2) add $t_0$, $t_1$, $t_2$
slt $t_3$, $t_0$, $t_1$
bne $t_3$, $x_0$, overflow

3) x86 和 ARM 架构中通常设置进位标志

9. 1)

| 指令 | $rs_1$ | $rs_2$ | DIVU | REMU | DIV | REM |
|---|---|---|---|---|---|---|
| op rd, $rs_1$, $rs_2$ | x | 0 | 0x ffffffffffffffff | x | 0x ffffffffffffffff | x |

2) fflags[0] : Invalid Operation NV
fflags[1] : Dvided by Zero DZ
fflags[2] : Overflow OF
fflags[3] : Underflow UF
fflags[4] : Inexact NX

12. 1) Supervisor
2) Machine
3) Machine
4) Supervisor
5) User

13.

```
vecMul:
    li  t3, 0
loop:
    bge  t3, 100, end
    lw   t4, 0(t1)
    lw   t5, 0(t2)
    mul  t6, t4, t5
    sw   t6, 0(t0)
    addi t3, t3, 1
    addi t0, t0, 4
    addi t1, t1, 4
    j    loop

end: lw  a0, 0(t0)
     jr  ra
```

14.

```
lw   a0, 0(sp)
lw   a1, 4(sp)
lw   a2, 8(sp)
blt  a0, a1, ELSE
add  a2, a0, a1
```

```
    j     END
ELSE: sub  a2, a0, a1

END:
```

15.
```
sw   t0 , 0(t0)
li   t1 , 3
sw   t1 , 4(t0)
sw   t1 , 12(t0)
```

16.
```
lw   a5 , 0(t0)
lw   a4 , 0(t1)
sw   a4 , 0(t0)
sw   a5 , 0(t1)
```

17.
```
      addi  a0 , x0 , 0          # a0 = 0
      addi  a1 , x0 , 1          # a1 = 1
      addi  a2 , x0 , 30         # a3 = 30
loop: beq   a0 , a2 , done       /* if a0 == a2, 跳转至loop (即 a0 < 30)
      slli  a1 , a1 , 1            else  a1 = a1 << 1
      addi  a0 , a0 , 1            a0++  */
      j     loop                 # 返回循环进行
done        # exit code
```

转为C代码：

```
int i = 0 , j = 1, k = 30;
for (i = 0, i < 30, i++)
  { a1 = a1 << 1 ; }
return 0.
```