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) slti t3, t2, 0

slt t4, t0, t1

2) bltu t0, t1, overflow

3) 在x86架构中 add指令的结果会更新进EFLAGS寄存器，其中可储存的标志位发生了溢出。可以使用jno或者jo指令来进行溢出判断。在ARM架构中，add指令会根据条件更新CPSR寄存器，其中V位表示加法是否发生了溢出。可以使用bvs或bvc指令来进行判断。

8. 1) ① 异常 ② × ③ 异常 ④ ×

为了避免除数为0导致的错误计算，同时也符合异常处理的规范

2) NV：浮点非法操作异常，发生无效操作

DZ：浮点除0异常

OF：浮点溢出异常

UF：浮点下溢异常：结果过小，无法表示为非0数

NX：浮点无穷大异常：发生了舍入错误，结果无法表示为有限小数

3) x86中，如果除数为0，会触发除0异常，处理器会跳转到异常处理程序。在ARM

架构中，当执行除法指令时，会触发相应的异常，如Divide by Zero Exception.

12. 1)M 2)M 3)M 4)U 5)U

13.
```
    li t3, 0
LoopStrat:
    beq t3, 100, LoopEnd
    lw a0, 0(t1)
    mul a0, a0, t2
    sw a0, 0(t0)
    addi t0, t0, 4
    addi t1, t1, 4
    addi t3, t3, 1
    j LoopStart
LoopEnd:
    lw a0, 0(t0)
    jr ra
```

14.
```
    bgt a0, a1, Label1
    sub a0, a0, a1
    j Exit
Label1:
    add a2, a0, a1
Exit:
```

15. sw t0, 0(t0)
    sw t1, 4(t0) <li t1, 3
    slli t1, t1, 2
    addi t1, t1, 4

16 swap:
    addi sp, sp, -8
    sw ra, 4(sp)
    sw s0, 0(sp)
    lw s0, 0(t0)
    lw t2, 0(t1)
    sw t2, 0(t0)
    sw s0, 0(t1)
    lw ra, 4(sp)
    lw s0, 0(sp)
    addi sp, sp, 8
    ret

17. 这段代码是将a1中的值不断右移，每次右移一次，直到a0中的值第30为止，每次循环，a0会加1，a1会右移一位，最终到done处结束。