T3

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

2) ret: 从子程序中返回。 jarl $x_0, x_1, 0$

3) call offset: 跳转4KB-4GiB空间的函数, auipc x_6, offset[31:12] ~~jmp(subroutine~~
jalr x_1, x_6, offset[11:0]

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

5) rdcycle rd: csrrs rd, cycle[h], x_0 周期数读取

6) sext.w rd, rs: addiw rd, rs, 0

T7

1) slti $t_3, t_1, 0$
slt t_4, t_0, t_2

2) add t_0, t_1, t_2
blt t_0, t_1, overflow
~~sub t_3, t_0, t_2~~
~~mv t_4, t_1~~
~~bne t_3, t_4, overflow~~

3) X86采用 Overflow Flag (OF)标志位来检测加法溢出。OF位为0则不溢出，OF位为1则溢出(①正数+正数=负数 ②负数+负数=正数)。ARM同理。
对于无符号数：使用 Carry Flag (CF)标志检测加法溢出。若两个无符号数相加有进位则溢出，CF=1，反之CF则为0。

T8.

OP=	DIVU	DIV	REM	REMU
rd	rd=0x~~ffff ffff ffff ffff~~	rd=0x~~ffff ffff ffff ffff~~	被除数	被除数

会引起RISC-V抛出异常，这种设计是为了保证系统的可靠性和安全性，帮助编译器和程序员发现潜在的编程错误。

2) NV: Invalid operation 标志位，表示发生了无效操作

DZ: Division by zero，表示发生了除以零的情况.

OF: Overflow，表示发生了数值溢出

UF: Underflow，表示发生了数值下溢.

NX: Inexact，表示浮点运算结果无法被精确表示

处理器通常会在遇到浮点异常时抛出一个异常，并停止程序执行，然后将控制权交给异常处理程序，这可能会导致程序进入系统调用

3) 在x86架构中，当执行DIV或IDIV指令时，若除数为0，会引发#DE异常，此时处理器会将异常向量号传给操作系统，并跳转到异常处理程序.

在ARM架构中，执行SDIV或UDIV指令时，若除数为0，会引发Data Abort异常，处理器将异常向量号传递给操作系统，并跳转到异常处理程序。

T12.

1) Linux kernel 处于M等级. 对硬件资源管理操作.

2) BootROM 处于M等级. 对硬件资源初始化.

3) BootLoader 处于S等级. 进行系统管理操作.

4) USB Driver 处于S等级. 访问一些特定的硬件资源

5) Vim 处于U等级，只访问用户文件和目录.

T13.

```
mymul:
Start:
  addi  sp, sp, -32
  sd    ra, 24(sp)
```

```
sd s0, 16(sp)
addi s0, sp, 32
li t3, 100
li t4, 0
j loop < li t5, 0
lw a3, 0(t2)
loop:
    beq t4, t3, end
    slli t5, t4, 2
    lw a2, 0(t1 + t5)
    mul a4, a2, a3
    sw a4 0(t0 + t5)
    j loop
end:
    ld ra, 24(sp)
    ld s0, 16(sp)
    addi sp, sp, 32
    ret.
```

T14

```
part1:
    bge a0 a1, part2
    sub a2, a0, a1
    j end.
part2:
    add a2, a0, a1
```

T15.

```
sw t0, 0(t0)
li t1, 3
sw t1, 0(t0+4)
li a2, t0, 4
mul a3, a2, t1
sw t1, 0(t0 + a3)
```

T16.

```
lw a2, 0(t0)
lw a3, 0(t1)
sw a3, 0(t0)
sw a2, 0(t1)
```

T17.

左移 a_1 30次，即将 a_1 乘 2^{30}.