2.3 (1) nop addi x0, x0, 0

(2) ret jalr x0, 0(x1)

(3) call offset { auipc x1, offset[31:12]+offset[11];
jalr x1, offset[11:0](x1).

(4) mv rd, rs addi rd, rs, 0.

(5) rdcycle rd csrrs rd, cycleh, x0.

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

2.7 (1) slti t2, t2, 0 (2) add t0, t1, t2

slt t4, t0, t1. bltu t0, t1, overflow.

(3) ARM: ① 进位与借位(无符号数): 状态寄存器的C位表示进位或非借位标志, 进位C=1

② 溢出: 当操作数与运算结果为二进制补码表示的带符号数, 用CPSR的V=1表示.

2.8 (1) 2^L-1 (L为操作指令的位宽) ; x ; -1 ; x.

(2) NV: 无效操作 DZ: 除零异常 OF: 溢出异常 UF: 下溢溢出 NX: 不精确

不知道, 没查到 ....

(3) ARM: 将除数进行除零判断后再进行运算.

2.12 (1) S (2) M (3) M (10) S (5) U

2.13
```
        add   t3, x0, x0   #Assign t3=i, t4=100
        add   t4, x0, 100
Loop:   bge   t3, t4, exit
        sll   t5, t3, 2
        add   t0, t0, t5
        add   t1, t1, t5
        lw    t1, 0(t1)
        mul   t0, t1, t2
        addi  t3, t3, 1
        j     Loop
exit:   ret
```

2.14
```
        blt   a0, a1, If   # a0<a1 or a>b.
        sub   a2, a1, a0
        j     end
If:     add   a2, a0, a1
        j     end
end:    ret.
```

2.15.
```
add  t0, t0, x0
addi t1, x0, 3
add  t2, x0, x0
sll  t3, t2, 2
add  t0, t0, x0
add  t0, t0, t3
add  t0, t1, x0
addx mul t3, t3, t1
add  t0, t0, t3
add  t0, t1, x0
```

2.16.
```
swap: lw  t0, 0(t0)
      add t2, x0, t0
      lw  t1, 0(t1)
      add t0, x0, t1
      sw  t0, 0(t0)
      add t1, t2, x0
      sw  t1, 0(t1)
      ret
```

2.17:
```
a0=0, a1=1  a2=30
loop: a0 = a2 直接结束循环.
      a1*2 覆盖
      a0+1 覆盖.
```

⇒

```
int a0=0, a1=1, a2=30;
for (a0=0, a0<a2, a0++){
    a1 = a1*2;
}
printf("%d", a1)
```

⇒ 计算 $a_1 = 2^{30} = ?$