取余符号：被除数为正数 则余数为正数，反之为负数

2.11 (1) jal ra, 0x88　直接寻址　(2) jalr x0, ra, 0　寄存器间接寻址

(3) addi a0, a1, 4　立即数寻址　(4) mul a0, a1, a2　寄存器寻址

(5) ld a4, 16(sp)　偏移量寻址

2.3 nop：addi x0, x0, 0　ret：jalr x0, x1, 0

call offset：auipc x6, offset[31:12]　mv rd, rs：addi rd, rs, 0

jalr x1, x6, offset[11:0]

rdcycle rd：csrrs rd, cycle, x0　sext.w rd, rs：addiw rd, rs, 0

2.7 1) sub t3, t0, t1　mv t4, t2

2) bltu t0, t1, overflow

3) x86和ARM指令集规定了检测溢出的标志寄存器：无符号数加法发生进位时，其中的进位标志会被置1，有符号数加法发生溢出时溢出标志会被置1

2.8 1) OP = DIV和DIVU时 rd = 0xffffffffffffffff

OP = REM和REMU时 rd = x

2) NV：非法操作，表示发生了不合法的浮点运算，如零除以零

DZ：除以零　OF：上溢出，表示发生了超过最大正数或最小负数的结果

UF：下溢出，表示发生了小于最小正数或最大负数的运算结果

NX：不精确，表示发生了舍入误差

不会，系统调用是由用户/应用主动发出的，除数为0仅仅是一种异常，操作系统会调用异常处理程序进行处理

3) 除数为0时程序有可能中断运行并进行异常处理，也有可能根据指令集定义的特殊情况，返回default值，或特殊值（如无穷大或NaN）

2.12 1) Linux Kernel = M　2) BootROM = M　3) BootLoader = M

4) USB Driver = S　5) vim = U

2.13

```
addi sp, sp, -32
sd   ra, 24(sp)
sd   s0, 16(sp)
addi s0, sp, 32
mv   a2, zero
addi a3, zero, 100
Loop:
bge  a2, a3, end
slli a4, a2, 2
add  a4, a4, t0
slli a5, a2, 2
add  a5, a5, t1
lw   a5, 0(a5)
mul  a6, a5, t1
sw   a6, 0(a4)
addi a2, a2, 1
j    Loop
end:
```

```
lw   a0, 0(t0)
ld   ra, 24(sp)
ld   s0, 16(sp)
addi sp, sp, 32
ret
```

2.14

```
bge  a1, a0, 1f
add  a2, a0, a1
j    2f
1:
sub  a2, a0, a1
2:
……
```

2.15

```
sw   t0, 0(t0)
addi t1, zero, 3
sw   t1, 4(t0)
mv   t2, t1
slli t2, t2, 2
add  a2, t0, t2
sw   t1, 0(a2)
```

2.16

```
addi sp, sp, -32
sd   ra, 24(sp)
sd   s0, 16(sp)
addi s0, sp(32)
lw   t2, 0(t0)
lw   a2, 0(t1)
sw   a2, 0(t0)
sw   t2, 0(t1)
ld   ra, 24(sp)
ld   s0, 16(sp)
addi sp, sp, 32
ret
```

2.17

```
addi a0, x0, 0    # a0 = 0
addi a1, x0, 1    # a1 = 1
addi a2, x0, 30   # a2 = 30
loop:
beq  a0, a2, done
slli a1, a1, 1    # a1 *= 2
addi a0, a0, 1    # a0 ++
j    loop
done: # exit code
```

实现了 a1 左移 30 位，得到 $a_1 = 2^{30}$