

5.21

3. u, nop : addi ~~no~~ x0, x0, 0

(1) ret : jalr x0, x1, 0

(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, cycle, x0

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

7. (1) slti t3, t2, 0

slti t4, t0, t1

(2) add t0, t1, t2

bltu t0, t1, overflow

(若  $t_0 = t_1 + t_2$  比  $t_1/t_2$  还小, 则跳转 overflow)

(3) x86 架构通过 CF 标志位找到加法进位位检测;

ARM 架构通过 CPSR 状态寄存器反映当前指令溢出状态

8. (1) 指令	rs1	rs2	divu rd	remu rd	div	rem
op rd, rs1, rs2	x	0	$2^{xlen}-1$	x	1	x

(2) NV: 非法操作标志 ; DZ: 除以0标志 ; OF: 上溢标志

UF: 下溢标志 ; NX: 不精确标志

若lags被置位, 则后续操作浮点数的指令会引起一个非法指令陷阱。

(3) x86在除数为0时会出现除法0中断

ARM在除数为0时会禁止IRQ中断和FIQ中断。复位后CPSR中最后状态为0011, 并进入管理模式, 执行操作系统程序, 初始化后切换到正常用户模式。

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

13. 解:

~~VERA~~

addi t3, x0, 0 # i=0

addi t4, x0, 100 # t4=100

Loop:

beg t3, t4, end

sll t5, t3, 2

add t5, t5, t1

lw t5, 0(t5) # \*(B+i)

~~add~~ t6, t5, t0

lw t6, 0(t6) # \*(A+i)

~~mul~~ t6, t5, t2

addi t3, t3, 1 # i++

end: j Loop

14. ~~beq~~ blt a0, a1, 2f      # a < b 跳到 2f  
       beq a0, a1, 2f      # a = b 跳到 2f  
       j 1f

1f:

add a2, a1, a0

2f: j end

2: ~~sub~~

sub a2, a1, a0

j end

end:

15. 解: ~~addi~~ ~~to~~

addi t1, x0, 3

mv t2, t0

add 1(t2), x0, t1

add 3(t2), x0, t1

16. 解:

~~addi~~ add t2, x0, t0      # temp = \*a

add t0, x0, t1      # \*a = b

add t1, x0, t2      # \*b = temp

end:

```
17.  int A[30]
    for (int i=0; i<30; i++) {
        A[i] = 1;
    }
```