

3.21 2/3/4, nop 等价于 $\text{addi } x0, x0, 0$

6/1 ret 等价于 $\text{jalu } x0, 0(\text{ra})$

(3) call offset 等价于 $\text{auipc } t0, \text{offset}[31:12]$ 和 $\text{jalu } t1, t0, \text{offset}[11:0]$

(4) mv rd, rs 等价于 $\text{addi } rd, rs, 0$

(5) rdcycle rd 等价于 $\text{csrrs } rd, \text{cycle}, x0$

16/1 sext.w rd, rs 等价于 $\text{slli } rd, rs, 0$ 和 $\text{srlw } rd, rd, 0x1f$

7/1/1 ~~slti~~ ~~t3~~, t2, 0

(2) bltu t0, t1, overflow

slti t4, t0, t1

(3) x86 架构用 jo 指令, 该指令检查 EFLAGS 寄存器中的 OF 位, 若为 1, 则溢出

ARM 架构可以用 SBC (subtract with carry) 指令, 它将两个操作数相减, 并将 Carry Flag 和 Overflow Flag 设为相应的值, 若 Carry Flag 为 1, 则为无符号数溢出; 若 Overflow Flag 为 1, 则为有符号数溢出

8/1/1 指令 rs1 rs2 Op=DIVU Op=REMU Op=DIV Op=REM

Op rd, rs1, rs2 X 0 $2^{\text{length}(x)} - 1$ X $2^{\text{length}(x)} - 1$ X

(2) NV: Invalid Operation, 无效操作 DZ: Divide by Zero: 除 0 异常

OF: Overflow 结果溢出 UF(Underflow): 结果下溢 NX: Inexact 结果不精确

Flags 被置为不会使处理器陷入系统调用

(3) x86中 整型除法为0会引起除0异常,可用异常处理程序来处理。浮点除法中除0会将FPU状态寄存器中标志位置位,不会引起异常。

ARM中,整型除法除数为0会引发除0异常,处理器会将异常的类型和地址信息保存在相关寄存器中并跳转到异常处理程序。处理异常程序中可以进行异常处理如输出错误信息等。

12/11 S (2) MM (3) MM (4) S (5) U

13 ~~add a2, zero, zero~~

add a3, zero, zero

addi a4, zero, 100

lw, a5, 0(t2)

Loop:

bge a3, a4, exit

sll a6, a3, 2

add a7, ~~a6~~, ~~a6~~, t0

add a6, a6, t1

~~lw a7, 0(a7)~~

lw a6, 0(a6)

mul ~~a6~~, a6, a5

sw a6, 0(a7)

addi, a3, a3, 1

j Loop

exit:

14 ~~beq~~ blt a0, a1, part1

beq a0, a1, part1

add a2, a1, a0

part1:

~~sub~~ a2, a0, a1

15 sw. ~~rp~~, 0(t0)

addi t1, zero, 3

sw t1, 4(t0)

sw t1, 12(t0)

17 功能是计算 2^{30}

16 lw. a1, ~~0(t0)~~ 0(t0)

~~lw a2, 0(t1)~~

~~addi a2, a1, 0~~

~~sw~~ a1, 0(t1)

sw a2, 0(t0)