

3. 伪指令

基础指令

hop

addi x0, x0, 0

ret

$\text{jalr } x0, 0(ra)$

call offset

5 auipc t0, offset[31:12]

7

mv rd, rs

addi rd, rs, 0

recycle

rdtime rd

CSRr

sext.w rd,rs

addiw rd, rs, 0

7. (1)

add t_0, t_1, t_2

$$\frac{5}{11} \text{ sr}$$
 $t_3, t_0, 31$

SH srl

 $t_4, t_{1.31}$

bne

t3, t4, overflow

(2)

~~add~~ to ~~t_0, t_1, t_2~~

~~add~~ ~~t_0, t_1, t_2~~

54

~~bgt~~

add

 t_0, t_1, t_2

stl

 t_0, t_1, t_2

bnez

to, over-

(3)

x86 架构中设置 EFLAGS 寄存器的 OF(进位) 和 OF(溢出) 位来检测

当有符号产生进位

10. 禁烟中设置

APSK 寄存器 $\sim C$ (相位) 和 $1/C$ (幅度) 估计检测 加误差

加江总

8.	$op = \text{divu}$	$op = \text{remu}$	$op = \text{div}$	$op = \text{rem}$
(1) $op \quad rd, rs1, rs2$	$0x\text{ffffffffffffffff}$	x	$0x\text{ffffffffffffffff}$	x

会引起 RISC-V 抛出异常，

这样设计 便于于检查异常，且不用区别有符号整型与无符号整型

(2) ~~0~~ 0: 除零标志位，当一个浮点数被零除时，该标志位被置位

1: 上溢标志位，当浮点运算结果太大，不能用浮点数表示时，该标志位被置位

2: 下溢标志位，当浮点运算结果太小，不能用浮点数表示时，该标志位被置位

3, 4, 5: 保留位，未使用

如果 $flags$ 被置位，不会导致处理陷入系统调用，它可以通知软件发生了浮点数运算异常

(3) 在 x86 中，当执行整数除法指令时，如果除数为 0，则会引发“除零异常”，该异常可由异常处理程序捕获和处理。

在 ARM 中，如果除数为 0，则会引发一个称为“被 0 除错误”异常，该异常可由异常处理程序捕获和处理。ARM 并不提供一个单独的异常状态寄存器，而是将异常状态信息保存在相关程序状态寄存器中。

3. rec~~Aut~~:

~~start:~~ ~~ti~~ ~~t3~~, 0

3. 13. vecMul:

```
start: li    t3, 0
        li    t4, 100
loop:   beq   t3, t4, end
        mv la t5, t0 t1
        mv la t6, t0 t0 t2
        mul   t5, t5, t6
        sw    t5, 0(t0)
        addi  t0, t0, 4
        addi  t1, t1, 4
        addi  t3, t3, 1
        j     loop
end:    lw     0(t0)
        ret
```

4. part1: bge a0, a1, ~~else~~ part2

negw a1, a1

add a2, a0, a1

j end

part2: add a2, a0, a1

end: ret

```

15.  mv    t2, t0
     sw    t2, 0(t0)    # p[0] = p
     li    t3, 3        # a = 3
     sw    t3, 4(t0t0)    # p[1] = a
     sw    t3, 12(t0)    # p[2] = a

```

16. swap:

```

     mv    t2, t1
     mv    t1, t0
     mv    t0, t2

```

17. 该代码实现的功能是计算 2^{30} 的值。

首先令 $a_0 = 0$, $a_1 = 1$, $a_2 = 31$

再使用循环。当 $a_0 \neq a_2$ 时, a_1 左移 1 位 (相当于乘 2), a_0 自增 1

这样当 $a = 31$ 时, 跳出循环。此时 a_1 左移 30 位, 相当于 1×2^{30}

最后, 循环结束, 程序返回