W5

3. RISC-V 汇编中存在许多伪指令，它们一般是具有特殊操作数的基本指令或指令组合。请写出与以下伪指令等价的基本指令或指令组合。

1) nop
2) ret
3) call offset
4) mv rd,rs
5) rdcycle rd
6) sext.w rd,rs

1) addi x0, x0, 0
2) jalr x0, x1, 0
3) auipc x6, offset[31:12]
jalr x1, x6, offset[11:0]
4) addi rd, rs, 0
5) csrrs rd, instret[h], x0
6) addiw rd, rs, 0

7. RISC-V 标准指令集并未为加法指令的溢出引入专用的标志位，因此通常需要额外的指令以检查加法溢出。

1) 考虑如下的指令序列：

add t0,t1,t2
slti t3, t2, 0
slt t4, t0, t1
bne t3,t4,overflow

若 t0 和 t1 都是有符号数，请在横线处填入正确的指令，使得当 t0 和 t1 的加法发生溢出时，控制流可以正确跳转到 overflow 位置。（请勿使用除 t0~t4 以外的任何寄存器）

2) 当 t0 和 t1 都是无符号数时，请给出尽量简单的检测 add t0,t1,t2 指令加法是否溢出的指令序列。

3) 调研其他指令集架构（如 x86、ARM 等）是如何检测加法溢出的。

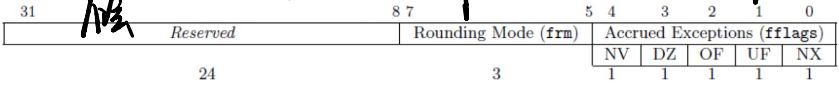
2) add t0, t1, t2
bltu t0, t1, overflow
3) ARM：通过 CPSR 的状态寄存器反映运算状态

8. 阅读 RISC-V 规范以了解 RISC-V 对除数为 0 的除法指令的处理方法，回答以下问题。

1) 对整型除法，填写下表。整型除法中除数为 0 是否会引起 RISC-V 抛出异常？试分析为什么采取这样的设计。

指令	rs1	rs2	Op=DIVU 时 rd 值	Op=REMU 时 rd 值	Op=DIV 时 rd 值	Op=REM 时 rd 值
Op rd,rs1,rs2	x	0				

2) 对浮点除法，除数为 0 将会引起 fcsr 控制寄存器中的相关标志位被置位。下图给出了 fcsr 的构成，请说明 fflags 的各位分别代表什么含义。fflags 被置位是否会使得处理器陷入系统调用？



3) 调研其他指令集架构（如 x86、ARM 等）是如何处理除数为 0 的。

12. 写出以下程序在 RISC-V 中应当处于的特权等级。

1) Linux Kernel
2) BootROM
3) BootLoader
4) USB Driver
5) vim

13.

```
    }
    return A[0];
}
```

```
addi t3, x0, 0
loop: addi t3, t3, 1
      beq t3, 100, end
      lw t4, 400)
      lw t5, 4(a1)
      mul t4, t5, t4
      j loop
end: jal t0, 0
```

14. 写出实现以下 C 程序的 32 位 RISC-V 汇编代码。假设 a、b 和 c 分别对应寄存器 a0、a1 和 a2。

```
int a, b, c;
if(a > b){
    c = a + b;
}
else{
    c = a – b;
}
```

```
blt a0, a1, end
sub a2, a0, a1
end: add a2, a0, a1
```

15. 写出实现以下 C 程序的 32 位 RISC-V 汇编代码。假设指针 p 已经通过程序 int *p = (int *) malloc(4*sizeof(int))得到，且 p 存放于 t0 中，a 存放于 t1 中。

```
p[0] = p;
int a = 3;
p[1] = a;
p[a] = a;
```

```
addi t1, t1, 3
sw t1, 4(t0)
mul t2, t1, 4
sw t1, t2(t0)
```

16. 写出实现以下 C 程序的 32 位 RISC-V 汇编代码。假设指针 a 和 b 分别存放于 t0 和 t1 中。

```
void swap(int *a, int *b) {
    int tmp = *a;
    *a = *b;
    *b = tmp;
    return;
}
```

```
addi t2, t0, 0
addi t0, t1, 0
addi t1, t2, 0
ra
```

17. 解释以下 RISC-V 汇编代码实现的功能。

```
        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
        addi a0,a0,1
        j loop
done:   # exit code
```

得到 $a1=2^{30}$

18. 有一组操作码，它们的出现几率如下表所示。