Chapter 2 HomeworK (2)

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　　addi x0, x0, 0

2) ret　　jalr x0, 0(x1)

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[h], x0

6) sext.w rd, rs　　addiw rd, rs, 0

△ csrrs 作用是将 CSR寄存器的值读入rd，然后将rs1的值与CSR的值按位或后写CSR寄存器。

△ offset为属性操作符，表示应该把其后跟着的符号地址的值作为操作数。

△ auipc是PC高位立即数加法指令，将12位立即数与PC的高20位的和放入目标寄存器，用于跳转

△ CSR寄存器：控制和状态寄存器

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：

若 t_0 和 t_1 都是有符号数，请在横线处填上正确的指令，使得当 t_0 和 t_1 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 t_0~t_4 以外的任何寄存器)

```
add t0, t1, t2          # t1, t2为有符号数
slti t3, t2, 0          # 判断 t2>0还是<0
slt t4, t0, t1          # t2小于0 则 t0小于t1; t2大于0，t0大于t1
bne t3, t4, overflow    # t3 != t4 跳转 overflow
```

2)当 t_0 和 t_1 都是无符号数时，请尽量给出简单的检测 add t0, t1, t2 指令加法是否溢出的指令序列。

```
add t0, t1, t2
bltu t0, t1, overflow
```

3) 调研其它指令集架构(如x86、ARM等)是如何检测加法溢出的。

ARM: ①进位与借位(无符号数)。状态寄存器的C位表示进位或非借位标志，加法指令(包括比较指令CMN)，根据两数相加的结果进行比较，当结果产生了进位，则C=1，其余情况C=0。

②溢出：当操作数与运算结果为二进制的补码表示的带符号数，用CSPR CPSR的V位，V=1表示符号位溢出

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^L-1	X	-1	X

(L is the width of the operation in bits)

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？

NV, invalid operation 无效操作

DZ, divide by zero 除零异常

OF, overflow 溢出异常

UF, underflow 借位溢出

NX, inexact 不精确

不会使处理器陷入系统调用？

3) 调研其它指令集架构(如x86、ARM等)是如何处理除数为0的。

ARM会先将除数进行判断，不为0才继续执行除法运算。

12. 写出以下程序在RISC-V中应当处于的特权等级。

1) Linux Kernel S

2) BootROM M

3) BootLoader M

4) USB Driver S

5) Vim U

△如何更好的区分？

S 操作系统内核，设备驱动程序

U 用于运行用户程序

13. 写出实现以下C程序的32位RISC-V汇编代码。假设A和B的起始地址存放于寄存器 t_0 和 t_1，C的地址存放于寄存器 t_2。

```
int vecMul(int *A, int *B, C)
{
  for(int i=0; i<100; ++i)
  {
    A[i] = B[i]*C;
  }
  return A[0];
}
```

```
# Assume t0 holds pointer to A
# Assume t1 holds pointer to B
# Assign t3 = i, t4 = 100
Loop:
  add t3, x0, x0 # i=0
  add t4, x0, 100 # t4=100
Loop:
  bge t3, t4, exit
  sll t5, t3, 2 # i*4
  add t0, t0, t5
  add t1, t1, t5
  lw t1, 0(t1)
  mul t0, t1, t2
  addi t3, t3, 1
  j Loop #Iterate
exit: ret
```

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 a_0、a_1 和 a_2。

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

```
blt a0, a1, If  # a0≤a1, a≤b
sub a2, a0, a1
j end
If: add a2, a0, a1
j end
end: ret # Dumping point is here
```

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t_0中，a存放于t_1中。

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

```
add t0, t0, x0
addi t1, x0, 3
add t2, x0, x0
sll t3, t2, 2
add t4 t0, t0, x0
```

```
add  t0 , t4 , t3
add  t0 , t1 , x0
addx mul t3 , t3 , t1
add  t0 , t4 , t3
add  t0 , t1 , x0
```

lw指令的应用?

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

```
swap: lw t0 , 0(t0)
    add t2 , x0 , t0
    lw t1 , 0(t1)
    add t0 , x0 , t1
    sw t0 , 0(t0).
    add t1 , t2 , x0
    sw t1 , 0(t1)
    ret
```

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  # (a0!=a2)
    slli  a1 , a1 , 1   # a1*2
    addi  a0 , a0 , 1   # a0=a0+1
    jloop
done : ~
```

```
int i=0 , j=1 , K=30 ;
for(i=0 , i<K(30) , i++)
{ j=j*2 ; }
# done
```

计算 2^{30} = ?