P4 FPU

《计算机组织结构》课程助教组

作业要求

1. 计算浮点数真值的和：dest + src

```
public DataType add(DataType src, DataType dest)
```

2. 计算浮点数真值的差：dest - src

```
public DataType sub(DataType src, DataType dest)
```

3. 计算浮点数真值的乘积：dest * src

```
public DataType mul(DataType src, DataType dest)
```

4. 计算浮点数真值的商：dest / src

```
public DataType div(DataType src, DataType dest)
```

解题步骤

1. 处理边界情况（NaN, 0, INF）
2. 提取符号、阶码、尾数
3. 模拟运算得到中间结果
4. 规格化并舍入返回

1.处理边界情况

基于正则表达式处理NaN情况的方法已经在框架代码中给出，在加减乘除四个方法中直接调用即可。

```
String a = dest.toString();
String b = src.toString();
if (a.matches(IEEE754Float.NaN_Regular) ||
     b.matches(IEEE754Float.NaN_Regular)) {
    return new DataType(IEEE754Float.NaN);
}
```

1.处理边界情况

处理边界情况的方法cornerCheck()已经在框架代码中给出，在调用时注意替换不同的边界情况矩阵。

```
private String cornerCheck(String[][] cornerMatrix,
                           String oprA, String oprB) {
    for (String[] matrix : cornerMatrix) {
        if (oprA.equals(matrix[0]) && oprB.equals(matrix[1])) {
            return matrix[2];
        }
    }
    return null;
}
```

1.处理边界情况

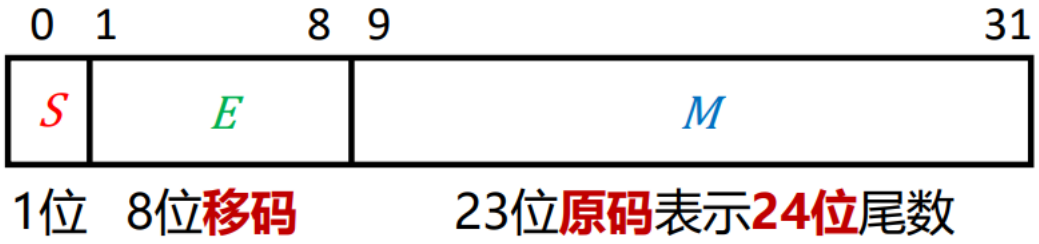
在完成除法时，需要注意被除数不为0，除数为0的情况。在我们的作业中，为了和整数除法保持一致，我们规定了当除数为0时抛出算数异常。

```
if(IEEE754Float.P_ZERO.equals(b) || IEEE754Float.N_ZERO.equals(b)) {
  if((!IEEE754Float.P_ZERO.equals(a)) &&
       (!IEEE754Float.N_ZERO.equals(a))){
                throw new ArithmeticException();
  }
}
```

请注意，在IEEE 754标准中，上述情况会返回无穷。

2.提取符号、阶码、尾数

本次实验采用IEEE 754标准中32位浮点数表示，提取符号、阶码、尾数的操作可以使用一系列字符串操作完成。



提取阶码的过程可以使用Java中的 Integer.valueOf(String s, int radix) 方法，提取出阶码后要进行3项特殊处理：

1. 取出的阶码全1（即阶码数值为255）：这个数表示无穷。对于加减法，返回本身。（无穷加减任何非无穷数还是无穷）对于乘除法，判断无穷的符号。
2. 取出阶码全0：这个数表示非规格化数。非规格化数真值是2的-126次方，因此需要将阶码加1保证对阶操作不会出错。
3. 尾数处理：根据是否为规格化数在尾数开头补1或0用于后续计算。同时采用直接补0的方式在末尾加上3位保护位。

3.模拟运算-加减

1. 对阶：小阶向大阶对齐，小阶增加至大阶。同时对小阶的尾数进行右移操作。请使用框架代码中提供的rightShift方法进行右移操作。

```
public String rightShift(String operand, int n)
```

2. 相加运算：可以使用ALU中的加法，但是请注意位数问题。计算时需要带上符号位，以获得正确的有符号计算结果。（这一步可以确定结果的符号位）

减法可以利用加法实现。与ALU中类似，将src符号取反后使用加法方法即可。

3.模拟运算-乘除

- 符号确定：相同为正，相异为负
- 阶码确定：乘除法不需要对阶。

  乘法阶码相加后减去偏置常数127，除法阶码相减后加上偏执常数127。
- 尾数确定：27位无符号数的乘除法

  注意：27位*27位无符号数得到的结果是54位，隐藏位为两位。（考虑小数1.xxx乘以1.xxx）因此需要将阶码加1。

4.规格化舍入并返回

需要规格化的情况：

- **尾数>27位**：尾数右移，阶码增加

  特殊情况：尾数增加到全1，指数上溢，返回无穷
- **尾数<27位**：尾数左移（删除尾数计算后得到的前导0直到隐藏位为1），阶码减小

  特殊情况：若阶码已经减到了0，尾数不需要再次左移。

  例如：阶码为 0000 0001，尾数为0.1000 0000 0000 0000 0000 0000 00

  规格化后阶码为全0，尾数是0.1000 0000 0000 0000 0000 0000 00

乘除还要考虑的情况：

- **阶码<0**：尾数右移，阶码增加，直到阶码为0或尾数前27位全为0

  如果阶码仍然小于0，此时发生阶码下溢，返回0

最后的尾数舍入操作请使用round方法，允许传入大于27位的尾数。