# P3 ALU

《计算机组织结构》课程助教组

# 作业要求

1. 在ALU类中实现实现整数的二进制乘法(要求使用布斯乘法实现)

```
public static String floatToBinary(String
floatStr)
```

2. 实现整数的二进制除法 (dest ÷ src)，使用恢复余数除法和不恢复余数除法均可。输入为32位二进制补码，输出为32位商，并且将32位余数正确储存在余数寄存器remainderReg中。

```
public static String binaryToFloat(String binStr)
```

# 解题步骤

1. 在ALU类中实现实现整数的二进制乘法(要求使用布斯乘法实现)

public DataType mul(DataType src, DataType dest)

实现思路：只需按照布斯算法描述严格实现即可

## 补码一位乘法：布斯算法

$[X \times Y]_c = 2^n \times [P_n]_c$ **即约定部分积的小数点到最右侧**

$[P_{i+1}]_c = [2^{-1} \times (P_i + X \times (y_i - y_{i+1}))]_c$ **求得$[P_i]_c$后，根据两位即可求得$[P_{i+1}]_c$**

$y_{i+1}y_i = 01$ 则 $[P_{i+1}]_c = [2^{-1} \times (P_i + X)]_c$

$y_{i+1}y_i = 10$ 则 $[P_{i+1}]_c = [2^{-1} \times (P_i - X)]_c$

} 执行$[P_i]_c + [\pm X]_c$然后右移一位

$y_{i+1}y_i = 00$ 则 $[P_{i+1}]_c = [2^{-1} \times P_i]_c$

$y_{i+1}y_i = 11$ 则 $[P_{i+1}]_c = [2^{-1} \times P_i]_c$

} 右移一位

**运算步骤：**

1. 增加 $y_0 = 0$
2. 根据 $y_{i+1}y_i$决定是否执行$[P_i]_c + [\pm X]_c$
3. 右移部分积
4. 重复步骤 2和步骤 3共 n 次，得到最终结果

推荐使用StringBuilder进行计算，deleteCharAt函数和insert函数等可以帮助你更快实现右移等操作。

# 解题步骤

2. 实现整数的二进制除法 (dest ÷ src)，使用恢复余数除法和不恢复余数除法均可。输入为32位二进制补码，输出为32位商，并且将32位余数正确储存在余数寄存器remainderReg中。

public DataType div(DataType src, DataType dest)

余数寄存器：X
除数寄存器：Y
商寄存器：Z

实现逻辑（以不恢复余数除法为例）：

1.输入检查、特殊情况处理（除法错、0做被除数等）

2.判断 X和Y符号：

同号：执行 X - Y

异号：执行 X + Y

3.判断 X和Y符号：

同号：X和Z左移一位最低位补1，X=2X-Y

异号：X和Z左移一位最低位补0，X=2X+Y

重复此步骤直到完成第 32 次循环

# 解题步骤

2. 实现整数的二进制除法 (dest ÷ src)，使用恢复余数除法和不恢复余数除法均可。输入为32位二进制补码，输出为32位商，并且将32位余数正确储存在余数寄存器remainderReg中。

```
public static String binaryToFloat(String binStr)
```

4.商修正：

Z左移一位，最低位：为X和Y同号补1，异号补0

被除数和Y同号：无操作

被除数和Y异号：Z最低位加1

5.余数修正：

X和被除数同号：不修正

X和被除数异号 & 被除数和Y同号：X=X + Y

X和被除数异号 & 被除数和Y异号：X=X - Y

注意除法算法固有的Bug（对于不恢复余数法），对于整除结果要进行特判处理