C0945 Assignment01

目的

模拟 ALU 进行整数和浮点数的四则运算。

要求

1. 修改 ALUSimulator.zip 中的 ALU.java 文件，实现以下各个方法。注意事项参见最后注意。

2. 方法说明

1) public String calculation(String formula)

该方法用于模拟两个操作数的四则运算，操作数可以为整数或浮点数，运算类型包括加减乘除。当两个操作数均为整数时，将操作数表示为 32 位的整数，并调用相应的整数运算方法进行运算；当至少一个操作数为浮点数时，采用 32 位的 IEEE 754 表示操作数，并调用相应的浮点数运算方法进行运算。

输入：

- formula：字符串表示的计算公式，其形式为：操作数 操作符 操作数=，例如“5+(-7)=”。计算公式中有且仅有 2 个操作数，采用十进制表示，当操作数为负数时可能用括号括住；有且仅有+-*/中的 1 个作为操作符；以=号结束。

输出：

- 返回值：计算结果的真值。如果是负数，最左边为“-”；如果是正数或 0，不需要符号位。

2) public String integerRepresentation(String number, int length)

该方法用于生成十进制整数的补码表示。

输入：

- number：十进制整数。如果是负数，最左边为“-”；如果是正数或 0，不需要符号位。
- length：补码表示的长度。

输出：

- 返回值：number 的补码表示，长度为 length。

3) public String floatRepresentation(String number, int sLength, int eLength)

该方法用于生成十进制浮点数的二进制表示。

输入：

- number：十进制浮点数，其中包含小数点。如果是负数，最左边为“-”；如果是正数或 0，不需要符号位。
- sLength：尾数的长度，取值大于等于 8。
- eLength：指数的长度，取值大于等于 8。

输出：

- 返回值：number 的二进制表示，长度为 1+sLength+eLength。从左向右，依次为符号、指数（移码表示）、尾数（首位隐藏）。需要考虑 0、反规格化表示、无穷、NaN 等因素，具体借鉴 IEEE 754。舍入采用就近舍入。

4) public String ieee754(String number, int length)

该方法用于生成十进制浮点数的 IEEE 754 表示，要求调用 floatRepresentation 实现。

输入：

- number：十进制浮点数，其中包含小数点。如果是负数，最左边为“-”；如果是正数或 0，不需要符号位。
- length：二进制表示的长度，为 32 或 64。

输出：

- 返回值：number 的二进制表示，长度为 length。从左向右，依次为符号、指数（移码表示）、尾数（首位隐藏）。

5) public String integerTrueValue(String operand)

该方法用于计算二进制表示的整数的真值。

输入：

- operand：操作数用补码形式的二进制表示。

输出：

- 返回值：操作数的真值。如果是负数，最左边为“-”；如果是正数或 0，不需要符号位。

6) public String floatTrueValue(String operand, int sLength, int eLength)

该方法用于计算二进制表示浮点数的真值。

输入：

- operand：操作数，原码形式的二进制表示。
- sLength：尾数的长度，取值大于等于 8。
- eLength：指数的长度，取值大于等于 8。

输出：

- 返回值：操作数的真值。如果是负数，最左边为“-”；如果是正数或 0，不需要符号位。正负无穷分别表示为“+Inf”和“-Inf”，NaN 表示为“NaN”。

7) public String negation(String operand)

该方法用于模拟按位取反操作。

输入：

- operand：操作数的二进制表示。

输出：

- 返回值：将操作数按位取反。

8) public String leftShift(String operand, int n)

该方法用于模拟左移操作，通过对字符串操作实现。

输入：

- operand：操作数的二进制表示。
- n：左移的位数。

输出：

- 返回值：左移的结果。

9) public String rightAriShift(String operand, int n)

该方法用于模拟算术右移操作，通过对字符串操作实现。

输入：

- operand：操作数的二进制表示。
- n：右移的位数。

输出：

- 返回值：右移的结果，高位补符号位。

10) public String rightLogShift(String operand, int n)

该方法用于模拟逻辑右移操作，通过对字符串操作实现。

输入：

- operand：操作数的二进制表示。
- n：右移的位数。

输出：

- 返回值：右移的结果，高位补 0。

11) public String fullAdder(char x, char y, char c)

该方法用于模拟全加器，对两位及进位进行加法运算。

输入：

- x 和 y：相加的两位，取值为 0 或 1。
- c：后面的进位，取值为 0 或 1。

输出：

- 返回值：长度为 2 的字符串，从左向右，第 1 位为和，第 2 位为进位。

12) public String claAdder(String operand1, String operand2, char c)

该方法用于模拟 8 位的先行进位加法器，要求调用 fullAdder 方法实现。

输入：

- operand1：被加数，用补码表示。
- operand2：加数，用补码表示。
- c：后面的进位。

输出：

- 返回值：长度为 9 的字符串。从左向右，前 8 位为计算结果，用补码表示；最后 1 位为进位。

13) public String integerAddition(String operand1, String operand2, char c, int length)

该方法用于模拟部分先行进位加法器，要求调用 claAdder 方法实现。

输入：

- operand1：被加数，用补码表示。
- operand2：加数，用补码表示。
- c：后面的进位。
- length：存放操作数的寄存器的长度。length 不小于操作数的长度，当某个操作数的长度小于 length 时，需要在高位补符号位。当 length 不为 8 的整数倍，需要对高位补位并运算。

输出：

- 返回值：长度为 length+1 的字符串。从左向右，前 length 位计算结果，用补码表示；最后 1 位为是否溢出，其中溢出为 1，不溢出为 0。

14) public String integerSubtraction(String operand1, String operand2, int length)

该方法用于模拟减法，要求调用 integerAddition 方法来实现。

输入：

- operand1：被减数，用补码表示。
- operand2：减数，用补码表示。

- length：存放操作数的寄存器的长度。length 不小于操作数的长度，当某个操作数的长度小于 length 时，需要在高位补符号位。

输出：

- 返回值：长度为 length+1 的字符串。从左向右，前 length 位计算结果，用补码表示；最后 1 位为是否溢出，其中溢出为 1，不溢出为 0。

15) public String integerMultiplication(String operand1, String operand2, int length)

该方法用于模拟 Booth 乘法，要求调用 integerAddition 方法和 integerSubtraction 方法来实现。

输入：

- operand1：被乘数，用补码表示。
- operand2：乘数，用补码表示。
- length：存放操作数的寄存器的长度。length 不小于操作数的长度，当某个操作数的长度小于 length 时，需要在高位补符号位。

输出：

- 返回值：长度为 length*2，为计算结果，用补码表示。

16) public String integerDivision (String operand1, String operand2, int length)

该方法用于模拟恢复余数除法，要求调用 integerAddition 方法、integerSubtraction 等方法来实现。

输入：

- operand1：被除数，用补码表示。
- operand2：除数，用补码表示。
- length：存放操作数的寄存器的长度。length 不小于操作数的长度，当某个操作数的长度小于 length 时，需要在高位补符号位。

输出：

- 返回值：长度为 length*2 的字符串。从左向右，前 length 位为商，用补码表示；后 length 为余数，用补码表示。

17) public String floatAddition(String operand1, String operand2, int sLength, int eLength, int gLength)

该方法用于模拟浮点数的加法，要求调用 integerAddition、integerSubtraction 等方法来实现。

输入：

- operand1：被加数，用二进制表示。
- operand2：加数，用二进制表示。
- sLength：尾数的长度，取值大于等于 8。
- eLength：指数的长度，取值大于等于 8。
- gLength：保护位的长度。

输出：

- 返回值：长度为 1+sLength+eLength+1 的字符串。从左向右，依次为符号、指数（移码表示）、尾数（首位隐藏）；最后 1 位为是否溢出，其中溢出为 1，不溢出为 0。舍入采用就近舍入。

18) public String floatSubtraction(String operand1, String operand2, int sLength, int eLength, int gLength)

该方法用于模拟浮点数的减法，要求调用 floatAddition 等方法来实现。

输入：

- operand1：被减数，用二进制表示。
- operand2：减数，用二进制表示。
- sLength：尾数的长度，取值大于等于 8。
- eLength：指数的长度，取值大于等于 8。
- gLength：保护位的长度。

输出：

- 返回值：长度为 1+sLength+eLength+1 的字符串。从左向右，依次为符号、指数（移码表示）、尾数（首位隐藏）；最后 1 位为是否溢出，其中溢出为 1，不溢出为 0。舍入采用就近舍入。

19) public String floatMultiplication(String operand1, String operand2, int sLength, int eLength)

该方法用于模拟浮点数的乘法，要求调用 integerAddition、integerSubtraction 等方法来实现。

输入：

- operand1：被乘数，用二进制表示。
- operand2：乘数，用二进制表示。
- sLength：尾数的长度，取值大于等于 8。
- eLength：指数的长度，取值大于等于 8。

输出：

- 返回值：长度为 1+sLength+eLength，为积，用二进制表示。从左向右，依次为符号、指数（移码表示）、尾数（首位隐藏）。舍入采用就近舍入。

20) public String floatDivision(String operand1, String operand2, int sLength, int eLength)

该方法用于模拟浮点数的恢复余数除法，要求调用 integerAddition、integerSubtraction 等方法来实现。

输入：

- operand1：被除数，用补码表示。
- operand2：除数，用补码表示。
- sLength：尾数的长度，取值大于等于 8。
- eLength：指数的长度，取值大于等于 8。

输出：

- 返回值：长度为 1+sLength+eLength，为商，用二进制表示。从左向右，依次为符号、指数（移码表示）、尾数（首位隐藏）。舍入采用就近舍入。

注意事项

1. 在 ALUSimulator.zip 所包含工程的基础上，采用 Java 编程。
2. 只能修改 ALU.java 文件内容，可以新建方法，但不得新建其它文件。在 ALU.java 的开头处添加注释，注明学号和姓名。
3. 所有方法的实现必须采用指定算法。
4. 如果提交结果不能正常被测试，可以有一次解释机会，但不得修改。最终无法被测试的提交结果计 0 分。