C1025 Assignment01

目的

模拟 ALU 进行整数和浮点数的四则运算。

要求

1. 解压 ALUSimulator.zip，修改 ALUSimulator/src/ALU.java 文件，使用 Java 实现以下各个方法。
2. 方法说明

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

生成十进制整数的二进制补码表示。

参数：

- number：十进制整数。若为负数；则第一位为“-”；若为正数或 0，则无符号位
- length：二进制补码表示的长度

返回值：

- number 的二进制表示，长度为 length

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

生成十进制浮点数的二进制表示。需要考虑 0、反规格化、正负无穷（“+Inf”和“-Inf”）、NaN 等因素，具体借鉴 IEEE 754。舍入策略为向 0 舍入。

参数：

- number：十进制浮点数（可能为不包含小数点的整数，例如 5）。若为负数；则第一位为“-”；若为正数或 0，则无符号位
- eLength：指数的长度，取值大于等于 4
- sLength：尾数的长度，取值大于等于 4

返回值：

- number 的二进制表示，长度为 1+eLength+sLength。从左向右，依次为

符号、指数（移码表示）、尾数（首位隐藏）

3) **public String ieee754 (String number, int length)**

生成十进制浮点数的 IEEE 754 表示，要求调用 floatRepresentation 实现。

参数：

- number：十进制浮点数，包含小数点。若为负数；则第一位为“-”；若为正数或 0，则无符号位
- length：二进制表示的长度，为 32 或 64

返回值：

- number 的 IEEE 754 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）

4) **public String integerTrueValue (String operand)**

计算二进制补码表示的整数的真值。

参数：

- operand：二进制补码表示的操作数

返回值：

- operand 的真值。若为负数；则第一位为“-”；若为正数或 0，则无符号位

5) **public String floatTrueValue (String operand, int eLength, int sLength)**

计算二进制原码表示的浮点数的真值。

参数：

- operand：二进制表示的操作数
- eLength：指数的长度，取值大于等于 4
- sLength：尾数的长度，取值大于等于 4

返回值：

- operand 的真值。若为负数；则第一位为“-”；若为正数或 0，则无符号位。正负无穷分别表示为“+Inf”和“-Inf”，NaN 表示为“NaN”

6) **public String negation (String operand)**

按位取反操作，需要采用非门模拟。

参数：

- operand：二进制表示的操作数

返回值：

- operand 按位取反的结果

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

左移操作。

参数：

- operand：二进制表示的操作数
- n：左移的位数

返回值：

- operand 左移 n 位的结果

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

逻辑右移操作。

参数：

- operand：二进制表示的操作数
- n：右移的位数

返回值：

- operand 逻辑右移 n 位的结果

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

算术右移操作。

参数：

- operand：二进制表示的操作数
- n：右移的位数

返回值：

- operand 算术右移 n 位的结果

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

全加器，对两位以及进位进行加法运算，需要采用与门、或门、异或门等模拟。

参数：

- x：被加数的某一位，取 0 或 1
- y：加数的某一位，取 0 或 1

- c：低位对当前位的进位，取 0 或 1

返回值：

- 相加的结果，用长度为 2 的字符串表示，第 1 位表示进位，第 2 位表示和

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

4 位先行进位加法器，要求采用 fullAdder 来实现。

参数：

- operand1：4 位二进制表示的被加数
- operand2：4 位二进制表示的加数
- c：低位对当前位的进位，取 0 或 1

返回值：

- 长度为 5 的字符串表示的计算结果，其中第 1 位是最高位进位，后 4 位是二进制表示的相加结果，其中进位不可以由循环获得

12) public String oneAdder (String operand)

加一器，实现操作数加 1 的运算。需要采用与门、或门、异或门等模拟，不可以直接调用 fullAdder、claAdder、adder、integerAddition 方法。

参数：

- operand：二进制补码表示的操作数

返回值：

- operand 加 1 的结果，长度为 operand 的长度加 1，其中第 1 位指示是否溢出（溢出为 1，否则为 0），其余位为相加结果

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

加法器，要求调用 claAdder 方法实现。

参数：

- operand1：二进制补码表示的被加数
- operand2：二进制补码表示的加数
- c：最低位进位
- length：存放操作数的寄存器的长度，为 4 的倍数。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

返回值：

- 长度为 length+1 的字符串表示的计算结果，其中第 1 位指示是否溢出（溢出为 1，否则为 0），后 length 位是相加结果

14) **public String integerAddition (String operand1, String operand2, int length)**

整数加法，要求调用 adder 方法实现。

参数：

- operand1：二进制补码表示的被加数
- operand2：二进制补码表示的加数
- length：存放操作数的寄存器的长度，为 4 的倍数。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

返回值：

- 长度为 length+1 的字符串表示的计算结果，其中第 1 位指示是否溢出（溢出为 1，否则为 0），后 length 位是相加结果

15) **public String integerSubtraction (String operand1, String operand2, int length)**

整数减法，要求调用 adder 方法实现。

参数：

- operand1：二进制补码表示的被减数
- operand2：二进制补码表示的减数
- length：存放操作数的寄存器的长度，为 4 的倍数。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

返回值：

- 长度为 length+1 的字符串表示的计算结果，其中第 1 位指示是否溢出（溢出为 1，否则为 0），后 length 位是相减结果

16) **public String integerMultiplication (String operand1, String operand2, int length)**

整数乘法，使用 Booth 算法实现，可调用 adder 等方法。

参数：

- operand1：二进制补码表示的被乘数

- operand2：二进制补码表示的乘数
- length：存放操作数的寄存器的长度，为 4 的倍数。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

返回值：

- 长度为 length+1 的字符串表示的计算结果，其中第 1 位指示是否溢出（溢出为 1，否则为 0），后 length 位是相乘结果

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

整数的不恢复余数除法，可调用 adder 等方法实现。

参数：

- operand1：二进制补码表示的被除数
- operand2：二进制补码表示的除数
- length：存放操作数的寄存器的长度，为 4 的倍数。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

返回值：

- 长度为 2*length+1 的字符串表示的相除结果，其中第 1 位指示是否溢出（溢出为 1，否则为 0），其后 length 位为商，最后 length 位为余数

18) public String signedAddition (String operand1, String operand2, int length)

带符号整数加法，可以调用 adder 等方法，但不能直接将操作数转换为补码后使用 integerAddition、integerSubtraction 来实现。

参数：

- operand1：二进制原码表示的被加数，其中第 1 位为符号位
- operand2：二进制原码表示的加数，其中第 1 位为符号位
- length：存放操作数的寄存器的长度，为 4 的倍数。length 不小于操作数的长度（不包含符号），当某个操作数的长度小于 length 时，需要将其长度扩展到 length

返回值：

- 长度为 length+2 的字符串表示的计算结果，其中第 1 位指示是否溢出（溢出为 1，否则为 0），第 2 位为符号位，后 length 位是相加结果

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

浮点数加法，要求调用 signedAddition 等方法实现。

参数：

- operand1：二进制表示的被加数
- operand2：二进制表示的加数
- eLength：指数的长度，取值大于等于 4
- sLength：尾数的长度，取值大于等于 4
- gLength：保护位的长度

返回值：

- 长度为 2+eLength+sLength 的字符串表示的相加结果，其中第 1 位指示是否指数上溢（溢出为 1，否则为 0），其余位从左到右依次为符号、指数（移码表示）、尾数（首位隐藏）。舍入策略为向 0 舍入

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

浮点数减法，要求调用 floatAddition 方法实现。

参数：

- operand1：二进制表示的被减数
- operand2：二进制表示的减数
- eLength：指数的长度，取值大于等于 4
- sLength：尾数的长度，取值大于等于 4
- gLength：保护位的长度

返回值：

- 长度为 2+eLength+sLength 的字符串表示的相减结果，其中第 1 位指示是否指数上溢（溢出为 1，否则为 0），其余位从左到右依次为符号、指数（移码表示）、尾数（首位隐藏）。舍入策略为向 0 舍入

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

浮点数乘法，可调用 integerMultiplication 等方法实现。

参数：

- operand1：二进制表示的被乘数

- operand2：二进制表示的乘数
- eLength：指数的长度，取值大于等于 4
- sLength：尾数的长度，取值大于等于 4

返回值：

- 长度为 2+eLength+sLength 的字符串表示的相乘结果，其中第 1 位指示是否指数上溢（溢出为 1，否则为 0），其余位从左到右依次为符号、指数（移码表示）、尾数（首位隐藏）。舍入策略为向 0 舍入

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

浮点数除法，可调用 integerDivision 等方法实现。

参数：

- operand1：二进制表示的被除数
- operand2：二进制表示的除数
- eLength：指数的长度，取值大于等于 4
- sLength：尾数的长度，取值大于等于 4

返回值：

- 长度为 2+eLength+sLength 的字符串表示的相除结果，其中第 1 位指示是否指数上溢（溢出为 1，否则为 0），其余位从左到右依次为符号、指数（移码表示）、尾数（首位隐藏）。舍入策略为向 0 舍入

注意事项

1. 请在 ALU.java 文件开头的注释中填写自己的学号和姓名。
2. 不得修改上述方法的方法名和参数，允许在 ALU.java 文件中创建新的方法，但是不允许创建新的文件。
3. 所有方法必须采用指定方式实现。
4. 可以调用 String、Integer、Math 等类。
5. 如果采用 sublime text 之类的编辑器来打开 ALU.java 文件，需要采用 GB 2312、GB18030 或者 GBK 编码，用 UTF-8 打开时中文注释会呈现乱码。
6. 不需要进行输入检查，假设调用各个方法时所传入的参数都是符合要求的，

例如在方法 3) public String ieee754 (String number, int length)中，参数 length 取 32 或 64，则不考虑 length 取其他值的情况。

7. 测试时会对所有的方法进行测试，以下为部分测试示例：
   - integerRepresentation("9", 8)
   - floatRepresentation("11.375", 8, 11)
   - ieee754("11.375", 32)
   - integerTrueValue("00001001")
   - floatTrueValue("01000001001101100000", 8, 11)
   - negation("00001001")
   - leftShift("00001001", 2)
   - logRightShift("11110110", 2)
   - ariRightShift("11110110", 2)
   - fullAdder('1', '1', '0')
   - claAdder("1001", "0001", '1')
   - oneAdder("00001001")
   - adder("0100", "0011", ‘0’, 8)
   - integerAddition("0100", "0011", 8)
   - integerSubtraction("0100", "0011", 8)
   - integerMultiplication("0100", "0011", 8)
   - integerDivision("0100", "0011", 8)
   - signedAddition("1100", "1011", 8)
   - floatAddition("00111111010100000", "00111111001000000", 8, 8, 4)
   - floatSubtraction("00111111010100000", "00111111001000000", 8, 8, 4)
   - floatMultiplication("00111110111000000", "00111111000000000", 8, 8)
   - floatDivision("00111110111000000", "00111111000000000", 8, 8)