“计算机组织结构”编程练习 02

目的

模拟 ALU 进行整数、浮点数和十进制数的四则运算。

要求

1. 类名为 ALU。
2. 包含 2 个枚举类型的成员变量：
   a) 操作类型 Operation，可以取的值包括：ADDITION，SUBTRACTION，MULTIPLICATION，DIVISION。
   b) 操作数类型 Type，可以取的值包括：INTEGER，FLOAT，DECIMAL。十进制整数不需进行转换为二进制，直接进行运算。
3. 包含以下方法：
   a) public String Complement(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。

   b) 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 4。
   - eLength：指数的长度，取值大于等于 4。

   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，其中 10…0 时进位。

   c) 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）。

d) public String NBCD(String number)

该方法用于生成十进制整数的 NBCD 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。

输出：

- 返回值：number 的 NBCD 表示。“+”表示为 1100，“-”表示为 1101。

e) public String TrueValue(String operand, Type type, int[] length)

该方法用于计算二进制表示的操作数的真值。

输入：

- operand：操作数。
  - 当 type 为 INTEGER 时，用补码形式的二进制表示。
  - 当 type 为 FLOAT 时，用原码形式的二进制表示。
  - 当 type 为 DECIMAL 时，用 NBCD 表示。
- type：操作数类型。
- length：
  - 当 type 为 INTEGER 时，忽略该参数。
  - 当 type 为 FLOAT 时，包含 2 个元素，依次为尾数的长度和指数的长度。
  - 当 type 为 DECIMAL 时，忽略该参数。

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”。

f) public String Negation(String operand)

该方法用于模拟取反操作。

输入：

- operand：操作数，用补码表示。

输出：

- 返回值：将操作数按位取反。

g) 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，用补码表示。
- n：左移的位数。

输出：

- 返回值：左移的结果。

h) public String RightShift(String operand, int n)

该方法用于模拟右移操作，通过对字符串操作实现。

输入：

- operand：操作数，用补码表示。
- n：右移的位数。

输出：

- 返回值：右移的结果，高位补符号位。

i) 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。

j) public String CLAAdder(String operand1, String operand2, char c, int length)

该方法用于模拟进位先行加法器，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。
- 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。length 的取值不大于 8。

输出：

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

k) public String Addition(String operand1, String operand2, char c, int length)

该方法用于模拟部分进位先行加法器，要求调用 CLAAdder 方法来模拟 8 位的进位先行加法器，并实现其串联。

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。

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。

l) public String AdditionF(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 Addition、Subtraction 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 4。
- eLength：指数的长度，取值大于等于 4。
- 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，其中 10…0 时进位。

m) public String AdditionD(String operand1, String operand2)

该方法用于模拟十进制整数加法，要求调用 Addition 方法来实现。

输入：

- operand1：被加数，用 NBCD 码表示。
- operand2：加数，用 NBCD 码表示。

输出：

- 返回值：和的 NBCD 码表示。

n) public String Subtraction(String operand1, String operand2, int length)

该方法用于模拟减法，要求调用 Addition 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。

o) public String SubtractionF(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 AdditionF 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 4。
- eLength：指数的长度，取值大于等于 4。
- 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，其中 10…0 时进位。

p) public String SubtractionD(String operand1, String operand2)

该方法用于模拟十进制数减法，要求调用 Negation、Addition 方法来实现。

输入：

- operand1：被减数，用 NBCD 码表示。
- operand2：减数，用 NBCD 码表示。

输出：

- 返回值：差的 NBCD 码表示。

q) public String Multiplication(String operand1, String operand2, int length)

该方法用于模拟 Booth 乘法，要求调用 Addition 方法和 Subtraction 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。

r) public String MultiplicationF(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 Addition、Subtraction 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 4。
- eLength：指数的长度，取值大于等于 4。

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，其中 10…0 时进位。

s) public String Division(String operand1, String operand2, int length)

该方法用于模拟不恢复余数除法，要求调用 Addition 方法和 Subtraction 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。

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。

t) public String DivisionF(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 Addition、Subtraction 方法来实现。

输入：

- operand1：被除数，用二进制表示。
- operand2：除数，用二进制表示。
- sLength：尾数的长度，取值大于等于 4。
- eLength：指数的长度，取值大于等于 4。

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，其中 10…0 时进位。

u) public String Calculation(String number1, String number2, Type type, Operation operation, int[] length)

该方法用于模拟两个整数或两个浮点数的四则运算，要求调用 Addition 方法、Subtraction 方法、Multiplication 方法、Division 方法、AdditionF 方法、SubtractionF 方法、MultiplicationF 方法和 DivisionF 方法。目前只需要实现 type 为 INTEGER 和 FLOAT 时的情形。

输出：

- number1：操作数 1，用十进制表示，为被加数/被减数/被乘数/被除数。

- number2：操作数 2，用十进制表示，为加数/减数/乘数/除数。
- type：操作数类型。
- operation：操作类型。
  - 当 type 为 INTEGER 时：包含 ADDITION，SUBTRACTION，MULTIPLICATION，DIVISION。
  - 当 type 为 FLOAT 时：包含 ADDITION，SUBTRACTION，MULTIPLICATION，DIVISION。
  - 当 type 为 DECIMAL 时：ADDITION，SUBTRACTION。
- length：存放长度信息的数组。
  - 当 type 为 INTEGER 时：包含 1 个元素，操作数将会转换为该长度的补码表示。
  - 当 type 为 FLOAT 时：包含 3 个元素，依次为尾数长度、指数长度和保护位长度（乘法和除法中保护位长度被自动忽略）。
  - 当 type 为 DECIMAL 时：忽略该参数。

输出：

- 返回值：计算结果，用十进制表示。其中，加法返回和，减法返回差，乘法返回积，除法返回商。如果是负数，最左边为“-”；如果是正数或 0，不需要符号位。

注意

1. 在 Programming02 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，标明学号和姓名。
3. 所有方法的实现必须采用指定算法，其过程必须与课件 Lecture13 上一致。
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。