```java
/*
 * CSC252 - Sim 3
 * File: Sim3_ALU.java
 * Author: Rithvik
 * Purpose: Build an N-bit ALU from individual ALU elements.
 */

public class Sim3_ALU {
    // ALU operation select bits
    public RussWire[] aluOp;
    // Flag to invert the b input (used for subtraction)
    public RussWire bNegate;
    // Inputs to the ALU
    public RussWire[] a;
    public RussWire[] b;
    // Results of the ALU operation
    public RussWire[] result;

    // The ALU elements and the ALU width
    private Sim3_ALUElement[] elements;
    private int n;

    // Constructor: Prepare arrays and ALU elements for given bit-width
    public Sim3_ALU(int size) {
        n = size;

        aluOp = new RussWire[3];
        for (int i = 0; i < 3; i++)
            aluOp[i] = new RussWire();

        bNegate = new RussWire();

        a = new RussWire[n];
        b = new RussWire[n];
        result = new RussWire[n];
        elements = new Sim3_ALUElement[n];

        for (int i = 0; i < n; i++) {
            a[i] = new RussWire();
            b[i] = new RussWire();
            result[i] = new RussWire();
            elements[i] = new Sim3_ALUElement();
        }
    }

    /**
     * Execute the ALU operation across all bits.
     */
    public void execute() {
        // Start with carry-in from bNegate signal
        boolean prevCarry = bNegate.get();

        // For each bit, propagate aluOp and inputs to ALU elements
        for (int i = 0; i < n; i++) {
            elements[i].aluOp[0].set(aluOp[0].get());
            elements[i].aluOp[1].set(aluOp[1].get());
            elements[i].aluOp[2].set(aluOp[2].get());

            // Set bInvert based on bNegate (used for subtraction)
            elements[i].bInvert.set(bNegate.get());

            // Set inputs a and b for this ALU element
            elements[i].a.set(a[i].get());
            elements[i].b.set(b[i].get());

            // Set carry input from previous element
            elements[i].carryIn.set(prevCarry);

            // Execute the first pass to compute sum and carry
            elements[i].execute_pass1();

            // Update carry for next element
            prevCarry = elements[i].carryOut.get();
        }

        // Connect the 'less' input of the first element to the last
addResult
        elements[0].less.set(elements[n - 1].addResult.get());

        // Clear 'less' input for all other elements
        for (int i = 1; i < n; i++) {
            elements[i].less.set(false);
        }

        // Execute second pass on all elements and collect the final
results
        for (int i = 0; i < n; i++) {
            elements[i].execute_pass2();
            result[i].set(elements[i].result.get());
        }
    }
}
```