```
1   // Eugene Ngo
2   // 4/7/2023
3   // CSE 469
4   // Lab 1 Task 3
5
6   // singleALU implements ALU logic for single bits which can be combined to
7   // make up large ALUs.
8   module singleALU (a, b, carryIn, ALUControl, Result, carryOut);
9      input logic a, b, carryIn;
10     input logic [1:0] ALUControl;
11     output logic Result, carryOut;
12
13
14     logic [2:0] outputs;
15
16     and andValue (outputs[1], b, a);
17     or  orValue  (outputs[2], b, a);
18
19     // Selecting B (Addition = A+B+0, Subtraction = A+(~B)+1)
20     // logic subtractSelector = ALUControl[0];
21
22     // MUX to select B (ALUControl[0] == 1 = select ~B)
23     wire middleValues[1:0];
24     and selectB (middleValues[0], ~ALUControl[0], b);
25     and selectNotB (middleValues[1], ALUControl[0], ~b);
26
27     logic selectedB;
28     or selectedBValue (selectedB, middleValues[0], middleValues[1]);
29
30     fullAdder fullAddedValue (.a(a), .b(selectedB), .carryIn(carryIn),
31                               .Result(outputs[0]), .carryOut(carryOut));
32
33     // MUX to select between computed values (add, sub, and, or)
34
35     always_comb begin
36        case (ALUControl)
37           2'b00: Result = outputs[0];
38           2'b01: Result = outputs[0];
39           2'b10: Result = outputs[1];
40           2'b11: Result = outputs[2];
41           default: Result = 1'bX;
42        endcase // end case statements
43     end // end comb block
44     // End of module
45
46  endmodule
47
```