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