```
1   // Eugene Ngo
2   // 5/3/23
3   // EE 469
4   // Lab 3
5
6   // alu is a module capable of adding, subtracting, anding and oring
7   // depending on what the control input says,
8   // and reports any flags that may appear
9   `timescale 1ns/10ps
10  module alu(input  logic [31:0] a, b,
11             input  logic [1:0]  ALUControl,
12             output logic [31:0] Result,
13             output logic [3:0]  ALUFlags);
14     logic [31:0] ADD, AND, OR, c0, b1;
15     assign AND = a & b;
16     assign OR = a | b;
17
18     // converts to negative if we are subtracting
19     always_comb begin
20        if (ALUControl[0] == 0) begin
21           b1 <= b;
22        end else begin
23           b1 <= ~b;
24        end
25     end
26
27     // Adder for all bits
28     // does our subtraction or addition for every bit in
29     // out A and B
30     fullAdder firstbit (.A(a[0]),.B(b1[0]),.cin(ALUControl[0]),.sum(ADD[0]),.cout(c0[0]));
31     genvar i;
32     generate
33        for (i = 1; i < 32; i++) begin : gen
34           fullAdder otherbits(.A(a[i]),.B(b1[i]),.cin(c0[i-1]),.sum(ADD[i]),.cout(c0[i]));
35        end
36     endgenerate
37
38     // determines output based on our ALUControl
39     always_comb begin
40        if (ALUControl == 2'b11) begin
41           Result <= OR;
42        end else if (ALUControl == 2'b10) begin
43           Result <= AND;
44        end else begin
45           Result <= ADD;
46        end
47     end
48
49     // Sets all our flags for our computation
50     assign ALUFlags[3] = Result[31];
51     assign ALUFlags[2] = (Result == 0);
52     assign ALUFlags[1] = c0[31] & !ALUControl[1];
53     assign ALUFlags[0] = !(a[31] ^ b[31] ^ ALUControl[0]) & (a[31] ^ ADD[31]) & !ALUControl[1
   ];
54  endmodule
55
56  // alu_testbench tests a variety of different inputs and ouputs
57  // pulled from the alu.tv file.
58  module alu_testbench();
59     logic [31:0] a, b;
60     logic [1:0]  ALUControl;
61     logic [31:0] Result;
62     logic [3:0]  ALUFlags;
63     logic clk;
64     logic [103:0] testvectors [1000:0];
65
66     alu dut (.a,.b,.ALUControl,.Result,.ALUFlags);
67
68     parameter CLOCK_PERIOD=100;
69
70     initial clk = 1;
71     always begin
72        #(CLOCK_PERIOD/2);
```

```
73          clk <= ~clk;
74       end
75
76
77       initial begin
78          $readmemh("alu.tv", testvectors);
79          for (int i = 0; i < 20; i = i + 1) begin
80             {ALUControl,a,b,Result,ALUFlags} = testvectors[i]; @(posedge clk);
81          end
82       end
83    endmodule
```