

[illegible]


```

1  //////////////////////////////////////
2  // Engineer:
3  // Create Date: 11/10/2024 05:24:45 AM
4  // Module Name: alu_8bit_tb
5  //////////////////////////////////////
6
7
8  `timescale 1ns / 1ps
9
10 module tb_alu_8bit;
11
12     // Inputs
13     reg [7:0] A;
14     reg [7:0] B;
15     reg [3:0] opcode;
16
17     // Outputs
18     wire [7:0] result;
19     wire carry;
20     wire zero;
21     wire overflow;
22     wire sign;
23
24     // Instantiate the ALU
25     alu_8bit uut (
26         .A(A), .B(B), .opcode(opcode), .result(result), .carry(carry), .zero(zero), .overflow(overflow), .sign(sign)
27     );
28
29     // Task to display results
30     task display_results;
31         input [7:0] A;
32         input [7:0] B;
33         input [3:0] opcode;
34         input [7:0] expected_result;
35         input expected_carry;
36         input expected_zero;
37         input expected_overflow;
38         input expected_sign;
39         begin
40             // Display the results
41             $display("A = %b, B = %b, Opcode = %b | Result = %b, Carry = %b, Zero = %b, Overflow = %b, Sign = %b",
42                 A, B, opcode, result, carry, zero, overflow, sign);
43             // Check the results
44             if (result !== expected_result)
45                 $display("Error: Expected Result = %b", expected_result);
46             if (carry !== expected_carry)
47                 $display("Error: Expected Carry = %b", expected_carry);
48             if (zero !== expected_zero)
49                 $display("Error: Expected Zero = %b", expected_zero);

```

```

module tb_alu_8bit;
30     task display_results;
39         begin
48             if (zero !== expected_zero)
50                 if (overflow !== expected_overflow)
51                     $display("Error: Expected Overflow = %b", expected_overflow);
52                 if (sign !== expected_sign)
53                     $display("Error: Expected Sign = %b", expected_sign);
54             end
55         endtask
56
57     initial begin
58         // Test Addition
59         A = 8'b00001111; B = 8'b00000001; opcode = 4'b0000; // 15 + 1
60         #10 display_results(A, B, opcode, 8'b00010000, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 16, no carry, no zero, no overflow, positive
61
62         A = 8'b01111111; B = 8'b00000001; opcode = 4'b0000; // 127 + 1
63         #10 display_results(A, B, opcode, 8'b10000000, 1'b0, 1'b0, 1'b1, 1'b1); // Expected: 128, no carry, no zero, overflow, negative
64
65         // Test Subtraction
66         A = 8'b00001000; B = 8'b00000100; opcode = 4'b0001; // 8 - 4
67         #10 display_results(A, B, opcode, 8'b00000100, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 4, no carry, no zero, no overflow, positive
68
69         A = 8'b00001000; B = 8'b00001000; opcode = 4'b0001; // 8 - 8
70         #10 display_results(A, B, opcode, 8'b00000000, 1'b0, 1'b1, 1'b0, 1'b0); // Expected: 0, no carry, zero, no overflow, positive
71
72         A = 8'b00000000; B = 8'b00000001; opcode = 4'b0001; // 0 - 1
73         #10 display_results(A, B, opcode, 8'b11111111, 1'b1, 1'b0, 1'b0, 1'b1); // Expected: 255, carry (borrow), no zero, no overflow, negative
74
75         // Test Logical Operations
76         A = 8'b11001100; B = 8'b10101010; opcode = 4'b0010; // A AND B
77         #10 display_results(A, B, opcode, 8'b10001000, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 136, no carry, no zero, no overflow, positive
78
79         A = 8'b11001100; B = 8'b10101010; opcode = 4'b0011; // A OR B
80         #10 display_results(A, B, opcode, 8'b11101110, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 238, no carry, no zero, no overflow, positive
81
82         A = 8'b11001100; B = 8'b10101010; opcode = 4'b0100; // A XOR B
83         #10 display_results(A, B, opcode, 8'b01100110, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 102, no carry, no zero, no overflow, positive
84
85         A = 8'b00001111; opcode = 4'b0101; // NOT A
86         #10 display_results(A, B, opcode, 8'b11110000, 1'b0, 1'b0, 1'b0, 1'b1); // Expected: 240, no carry, no zero, no overflow, negative
87
88         // Test Shift Operations
89         A = 8'b00001111; opcode = 4'b0110; // A LSHIFT
90         #10 display_results(A, B, opcode, 8'b00011110, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 30, no carry, no zero, no overflow, positive
91
92         A = 8'b00001111; opcode = 4'b0111; // A RSHIFT
93         #10 display_results(A, B, opcode, 8'b00000111, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 7, no carry, no zero, no overflow, positive
94         $display("Error: Expected zero = %b", expected_zero);

```

```

10 module tb_alu_8bit;
57     initial begin
58         A = 8'b00001111; opcode = 4'b0111; // A < B
93         #10 display_results(A, B, opcode, 8'b00000111, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 7, no carry, no zero, no overflow, positive
94
95         // Test Comparison Operations
96         A = 8'b00001000; B = 8'b00000100; opcode = 4'b1000; // A < B
97         #10 display_results(A, B, opcode, 8'b00000000, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 0, no carry, no zero, no overflow, positive
98
99         A = 8'b00000100; B = 8'b00001000; opcode = 4'b1000; // A < B
100        #10 display_results(A, B, opcode, 8'b00000001, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 1, no carry, no zero, no overflow, positive
101
102        A = 8'b00001000; B = 8'b00001000; opcode = 4'b1001; // A == B
103        #10 display_results(A, B, opcode, 8'b00000001, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 1, no carry, no zero, no overflow, positive
104
105        // Test Increment and Decrement
106        A = 8'b00001111; opcode = 4'b1010; // Increment A
107        #10 display_results(A, B, opcode, 8'b00010000, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 16, no carry, no zero, no overflow, positive
108
109        A = 8'b00000001; opcode = 4'b1011; // Decrement A
110        #10 display_results(A, B, opcode, 8'b00000000, 1'b0, 1'b1, 1'b0, 1'b0); // Expected: 0, no carry, zero, no overflow, positive
111
112        A = 8'b00000000; opcode = 4'b1011; // Decrement A
113        #10 display_results(A, B, opcode, 8'b11111111, 1'b1, 1'b0, 1'b0, 1'b1); // Expected: 255, carry, no zero, no overflow, negative
114
115        // Test Multiplication
116        A = 8'b00000010; B = 8'b00000010; opcode = 4'b1100; // 2 * 2
117        #10 display_results(A, B, opcode, 8'b00000100, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 4, no carry, no zero, no overflow, positive
118
119        A = 8'b00001000; B = 8'b00000010; opcode = 4'b1100; // 8 * 2
120        #10 display_results(A, B, opcode, 8'b00010000, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 16, no carry, no zero, no overflow, positive
121
122        // Test Division
123        A = 8'b00001000; B = 8'b00000010; opcode = 4'b1101; // 8 / 2
124        #10 display_results(A, B, opcode, 8'b00000100, 1'b0, 1'b0, 1'b0, 1'b0); // Expected: 4, no carry, no zero, no overflow, positive
125
126        A = 8'b00000000; B = 8'b00000001; opcode = 4'b1101; // 0 / 1
127        #10 display_results(A, B, opcode, 8'b00000000, 1'b0, 1'b1, 1'b0, 1'b0); // Expected: 0, no carry, zero, no overflow, positive
128
129        A = 8'b00001000; B = 8'b00000000; opcode = 4'b1101; // Division by zero
130        #10 display_results(A, B, opcode, 8'b00000000, 1'b1, 1'b0, 1'b0, 1'b0); // Expected: 0, carry for division by zero
131
132        // End of simulation
133        $finish;
134    end
135 endmodule
136

```