

```
1  module ULA (
2
3  input[31:0] operand1, operand2,
4  input wire [3:0] Unit_Control_ALU,
5
6  output wire [31:0] Output_Result,
7  output wire zero,
8  output wire Output_1bit
9  );
10
11
12  reg[31:0] result;
13  reg Reg_Zero;
14
15  initial begin
16      result_1bit <= 1'd0;
17      result <= 32'd0;
18  end
19
20
21  always @(operand1 or operand2 or Unit_Control_ALU) begin
22
23      case (Unit_Control_ALU)
24          4'b0000: result = operand1 & operand2; // AND
25          4'b0001: result = operand1 | operand2; // OR
26          4'b0010: result = operand1 + operand2; // ADD
27          4'b0011: result = operand1 - operand2; // SUB
28          4'b0100: begin
29
30              if (operand1 < operand2) begin // SLT
31                  result = 32'd1;
32              end
33              else begin
34                  result = 32'd0;
35              end
36          end
37          4'b0101: begin
38
39              if (operand1 > operand2) begin // SGT
40                  result = 32'd1;
41              end
42              else begin
43                  result = 32'd0;
44              end
45          end
46          4'b0110: begin
47
48              if (operand1 >= operand2) begin // SGET
49                  result = 32'd1;
50              end
51              else begin
52                  result = 32'd0;
53              end
54          end
55          4'b0111: begin
56
57              if (operand1 <= operand2) begin // SLET
58                  result = 32'd1;
59              end
60              else begin
61                  result = 32'd0;
62              end
63          end
64          4'b1000: result = operand1 * operand2; // MULT
65          4'b1001: result = operand1 / operand2; // DIV
66          4'b1010: result = ~(operand1 | operand2); // NOR
67          4'b1011: result = operand1 >> operand2; //SLL
68          4'b1100: result = operand1 << operand2; // SRL
69
70      endcase
71
72      if (Output_Result == 32'd0) begin
73          Reg_Zero = 1'b1;
74      end
75      else begin
76          Reg_Zero = 1'b0;
77      end
78  end
```

```
78         end
79
80     end
81
82     assign Output_Result = result;
83     assign zero = Reg_Zero;
84
85 endmodule
86
87
```