```
module ULA (

input[31:0] operand1, operand2,
input wire [3:0] Unit_Control_ALU,

output wire [31:0] Output_Result,
output wire zero,
output wire Output_1bit
);


reg[31:0] result;
reg Reg_Zero;


initial begin
   result_1bit <= 1'd0;
   result <= 32'd0;
end


always @(operand1 or operand2 or Unit_Control_ALU) begin

      case (Unit_Control_ALU)
         4'b0000: result = operand1 & operand2; // AND
         4'b0001: result = operand1 | operand2; // OR
         4'b0010: result = operand1 + operand2; // ADD
         4'b0011: result = operand1 - operand2; // SUB
         4'b0100: begin

                     if (operand1 < operand2) begin // SLT
                        result = 32'd1;
                     end
                     else begin
                        result = 32'd0;
                     end
                  end
         4'b0101: begin

                     if (operand1 > operand2) begin // SGT
                        result = 32'd1;
                     end
                     else begin
                        result = 32'd0;
                     end
                  end
         4'b0110: begin

                     if (operand1 >= operand2) begin // SGET
                        result = 32'd1;
                     end
                     else begin
                        result = 32'd0;
                     end
                  end
         4'b0111: begin

                     if (operand1 <= operand2) begin // SLET
                        result = 32'd1;
                     end
                     else begin
                        result = 32'd0;
                     end
                  end
         4'b1000: result = operand1 * operand2; // MULT
         4'b1001: result = operand1 / operand2; // DIV
         4'b1010: result = ~(operand1 | operand2); // NOR
         4'b1011: result = operand1 >> operand2; //SLL
         4'b1100: result = operand1 << operand2; // SRL

      endcase

      if (Output_Result == 32'd0) begin
         Reg_Zero = 1'b1;
      end
      else begin
         Reg_Zero = 1'b0;
```

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