```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 02.09.2024 00:08:12
// Design Name:
// Module Name: ALU
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ALU(
    input [5:0] opcode,
    input [5:0] funct,
    input [31:0] in1,
    input [31:0] in2,
    output reg [31:0] result,
    output reg rw,  // A signal that decides whether to write into file (if 1 ->
write)
    input clk
);

    wire [31:0] sum;
    wire [31:0] diff;
    wire [31:0] product;
    wire [31:0] sum_or;

    // Instantiate the adder, subtractor, AND, and OR modules
    thirtytwobitadder ADD (.a(in1), .b(in2), .carryout(), .sum(sum),
.carryin(1'b0));
    thirtytwobitsubtractor SUBTRACT (.a(in1), .b(in2), .carry(), .diff(diff),
.borrowin(1'b0));
    AND prod (.a(in1), .b(in2), .result(product));
    OR orop (.a(in1), .b(in2), .result(sum_or));

    // ALU operation selection
    always @(*)
    begin
```

```
        rw = 1'b0;  // Default to no write

        case (opcode)
            6'b000000: begin // R-type instructions
                case (funct)
                    6'b100000: result = sum;         // ADD
                    6'b100010: result = diff;        // SUB
                    6'b100100: result = product;     // AND
                    6'b100101: result = sum_or;      // OR
                    default: result = 32'b0;         // Default case for unhandled
funct
                endcase
                rw = 1'b1; // Set rw for R-type operations
            end

            6'b100011: begin // LW
                result = sum;
                rw = 1'b1;
            end

            6'b101011: begin // SW
                result = sum;
                rw = 1'b0; // Typically no write enable for store word
            end

            6'b000100: begin // BEQ
                if (diff == 32'b0) begin
                    result = diff;
                end else begin
                    result = 32'b0; // Ensure result is set to a default value if
condition fails
                end
                rw = 1'b0; // Typically no write enable for branch equal
            end

            default: begin
                result = 32'b0; // Default case for unhandled opcodes
                rw = 1'b0;
            end
        endcase
    end

endmodule
```