```
module ALU #(parameter W=8)(ALUCtrl,A,B,out,co,ovf,z,n);
    input [(W-1):0] A,B;
    input [2:0] ALUCtrl;
    output [(W-1):0] out;
    output reg co,ovf,z,n;
    reg [(W-1):0] out;
    reg [15:0]temp;
    reg [7:0]    multiplier_copy;
    reg [15:0]    multiplicand_copy;
    reg              negative_output;
    reg [15:0] product_temp;

    reg [5:0]     bit;

    initial bit = 0;
    initial negative_output = 0;


    always @(*)
    begin
        case(ALUCtrl)
       0: // Add
       begin
          {co,out} = A + B ;
          z=(out == 0) ? 1 : 0;
          if ((A[W-1] == 0 && B[W-1] == 0 && out[W-1] == 1) || (A[W-1] == 1 && B[W-1] == 1 && out[W-1] == 0)) begin
                  ovf = 1;
                  n=0;
               end else begin
                  ovf = 0;
                  if (out[W-1]==1) begin
                     n=1;
                  end else begin
                     n=0;
                  end
               end
      end
       1: // SubAB
       begin
           {co,out} = A + (~B + 1)  ;  // (~B + 1)
           z=(out == 0) ? 1 : 0;
           if ((A[W-1] == 1 && B[W-1] == 0 && out[W-1] == 0) || (A[W-1] == 0 && B[W-1] == 1 && out[W-1] == 1)) begin
                  ovf = 1;
                  n=0;
               end else begin
                  ovf = 0;
                  if (out[W-1]==1) begin
                     n=1;
                  end else begin
                     n=0;
                  end
               end
       end
       2: //and
       begin
           out = A & B;
           co=0;
           ovf=0;
           z=(out == 0) ? 1 : 0;
           if (out[W-1]==1) begin
            n=1;
           end else begin
            n=0;
           end
       end

       3: // or
       begin
           out = A | B;
           co=0;
           ovf=0;
           z=(out == 0) ? 1 : 0;
           if (out[W-1]==1) begin
            n=1;
           end else begin
```

```
            n=0;
           end
       end

       4: // xor
       begin
           out = A ^ B;
           co=0;
           ovf=0;
           z=(out == 0) ? 1 : 0;
           if (out[W-1]==1) begin
            n=1;
           end else begin
            n=0;
           end
       end

       5: // clear
       begin
           out = 0;
           co=0;
           ovf=0;
           z=(out == 0) ? 1 : 0;
           if (out[W-1]==1) begin
            n=1;
           end else begin
            n=0;
           end
       end

       6: // shift
       begin
           out = A;
           co=0;
           ovf=0;
           z=(out == 0) ? 1 : 0;
           if (out[W-1]==1) begin
            n=1;
           end else begin
            n=0;
           end
       end


          default: out = A + B ;
        endcase
    end

endmodule
```