```
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
```

```
76                n=0;
77              end
78          end
79
80          4: // xor
81          begin
82              out = A ^ B;
83              co=0;
84              ovf=0;
85              z=(out == 0) ? 1 : 0;
86              if (out[W-1]==1) begin
87               n=1;
88              end else begin
89               n=0;
90              end
91          end
92
93          5: // clear
94          begin
95              out = 0;
96              co=0;
97              ovf=0;
98              z=(out == 0) ? 1 : 0;
99              if (out[W-1]==1) begin
100              n=1;
101             end else begin
102              n=0;
103             end
104         end
105
106         6: // shift
107         begin
108             out = A;
109             co=0;
110             ovf=0;
111             z=(out == 0) ? 1 : 0;
112             if (out[W-1]==1) begin
113              n=1;
114             end else begin
115              n=0;
116             end
117         end
118
119
120            default: out = A + B ;
121         endcase
122     end
123
124 endmodule
```