```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 04/20/2021 03:07:01 PM
// Design Name:
// Module Name: lab3_simulation
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module lab3_simulation();
    reg [15:0] sw;
    wire [6:0] seg;
    wire [3:0] an;
    reg btnU, btnR, clkin;
    wire dp;
    lab3_top UUT (.sw(sw), .btnU(btnU), .btnR(btnR), .clkin(clkin), .seg(seg),
.dp(dp), .an(an));

    wire [7:0] D7Seg3, D7Seg2, D7Seg1, D7Seg0; // Change the Radix of these signals
to ASCII
    show_7segDisplay  showit (.seg(seg),.dp(dp),.an(an),
     .D7Seg0(D7Seg0),.D7Seg1(D7Seg1),.D7Seg2(D7Seg2),.D7Seg3(D7Seg3));

    parameter PERIOD = 10;
    parameter real DUTY_CYCLE = 0.5;
    parameter OFFSET = 2;

    initial    // Clock process for clkin
    begin
        #OFFSET
          clkin = 1'b1;
        forever
        begin
```

```
        #(PERIOD-(PERIOD*DUTY_CYCLE)) clkin = ~clkin;
    end
 end

initial
begin
  // add your stimuli here
  // to set signal foo to value 0 use
  // foo = 1'b0;
  // to set signal foo to value 1 use
  // foo = 1'b1;
  //always advance time my multiples of 100ns
  // to advance time by 100ns use the following line
  btnR = 1'b0;
  #1000;
  //Skip first 1000ns.
  sw[15:8] = 8'b00000000;
  sw[7:0] = 8'b00000000;
  btnU = 1'b0;
  //btnR = 1'b0;
  //0 + 0 = 0.
  #200;
  sw[15:8] = 8'b00000001;
  sw[7:0] = 8'b00000001;
  btnU = 1'b1;
  //1 - 1 = 0.
  #200;
  sw[15:8] = 8'b11111111;
  sw[7:0] = 8'b00000001;
  //-1 - 1 = -2.
  #200;
  sw[15:8] = 8'b00000010;
  sw[7:0] = 8'b00000001;
  //2 - 1 = 1.
  #200;
  sw[15:8] = 8'b10000000;
  sw[7:0] = 8'b00000001;
  btnU = 1'b0;
  //-128 + 1 = -127.
  #200;
  //Trying 1-F on all bits now.
  sw[15:8] = 8'b00000001;
  sw[7:0] = 8'b00010000;
  //11
  #200;
  sw[15:8] = 8'b00000010;
```

```
sw[7:0] = 8'b00100000;
//22
#200;
sw[15:8] = 8'b00000011;
sw[7:0] = 8'b00110000;
//33
#200;
sw[15:8] = 8'b00000100;
sw[7:0] = 8'b01000000;
//44
#200;
sw[15:8] = 8'b00000101;
sw[7:0] = 8'b01010000;
//55
#200;
sw[15:8] = 8'b00000110;
sw[7:0] = 8'b01100000;
//66
#200;
sw[15:8] = 8'b00000111;
sw[7:0] = 8'b01110000;
//77
#200;
sw[15:8] = 8'b00001000;
sw[7:0] = 8'b10000000;
//88
#200;
sw[15:8] = 8'b00001001;
sw[7:0] = 8'b10010000;
//99
#200;
sw[15:8] = 8'b00001010;
sw[7:0] = 8'b10100000;
//AA
#200;
sw[15:8] = 8'b00001011;
sw[7:0] = 8'b10110000;
//BB
#200;
sw[15:8] = 8'b00001100;
sw[7:0] = 8'b11000000;
//CC
#200;
sw[15:8] = 8'b00001101;
sw[7:0] = 8'b11010000;
//DD
```

```
    #200;
    sw[15:8] = 8'b00001110;
    sw[7:0] = 8'b11100000;
    //EE
    #200;
    sw[15:8] = 8'b00001111;
    sw[7:0] = 8'b11110000;
    //FF
    #200;
    //Also include 8 tests where the two numbers have the same and opposite signs an
the input sub alternates.
    sw[15:8] = 8'b00000100;
    sw[7:0] = 8'b00000101;
    //4 + 5 = 9.
    #200;
    sw[15:8] = 8'b00000100;
    sw[7:0] = 8'b00000101;
    btnU = 1'b1;
    //4 - 5 = -1.
    #200;
    sw[15:8] = 8'b11111100;
    sw[7:0] = 8'b00000101;
    btnU = 1'b0;
    // -4 + 5 = 1.
    #200;
    sw[15:8] = 8'b11111100;
    sw[7:0] = 8'b00000101;
    btnU = 1'b1;
    // -4 - 5 = -9.
    #200;
    sw[15:8] = 8'b11111100;
    sw[7:0] = 8'b11111011;
    btnU = 1'b0;
    // -4 + -5 = -9.
    #200;
    sw[15:8] = 8'b11111100;
    sw[7:0] = 8'b11111011;
    btnU = 1'b1;
    // -4 - -5 = 1.
    #200;
    sw[15:8] = 8'b00000100;
    sw[7:0] = 8'b11111011;
    btnU = 1'b0;
    //4 + -5 = -1.
    #200;
    sw[15:8] = 8'b00000100;
```

```
    sw[7:0] = 8'b11111011;
    btnU = 1'b1;
    //4 - -5 = 9.
    #200;
    //Make sure to have some values that cause overflows.
    sw[15:8] = 8'b01111111;
    sw[7:0] = 8'b00001000;
    btnU = 1'b0;
    //127 + 8 = overflow.
    #200;
    sw[15:8] = 8'b01000000;
    sw[7:0] = 8'b11000000;
    btnU = 1'b1;
    //64 - (-64) = overflow.
    #200;
    sw[15:8] = 8'b10000000;
    sw[7:0] = 8'b11111000;
    btnU = 1'b0;
    //-128 + -8 = overflow.
    #200;
    sw[15:8] = 8'b10000000;
    sw[7:0] = 8'b00001000;
    btnU = 1'b1;
    //-128 - 8 = overflow.

   end

endmodule
```