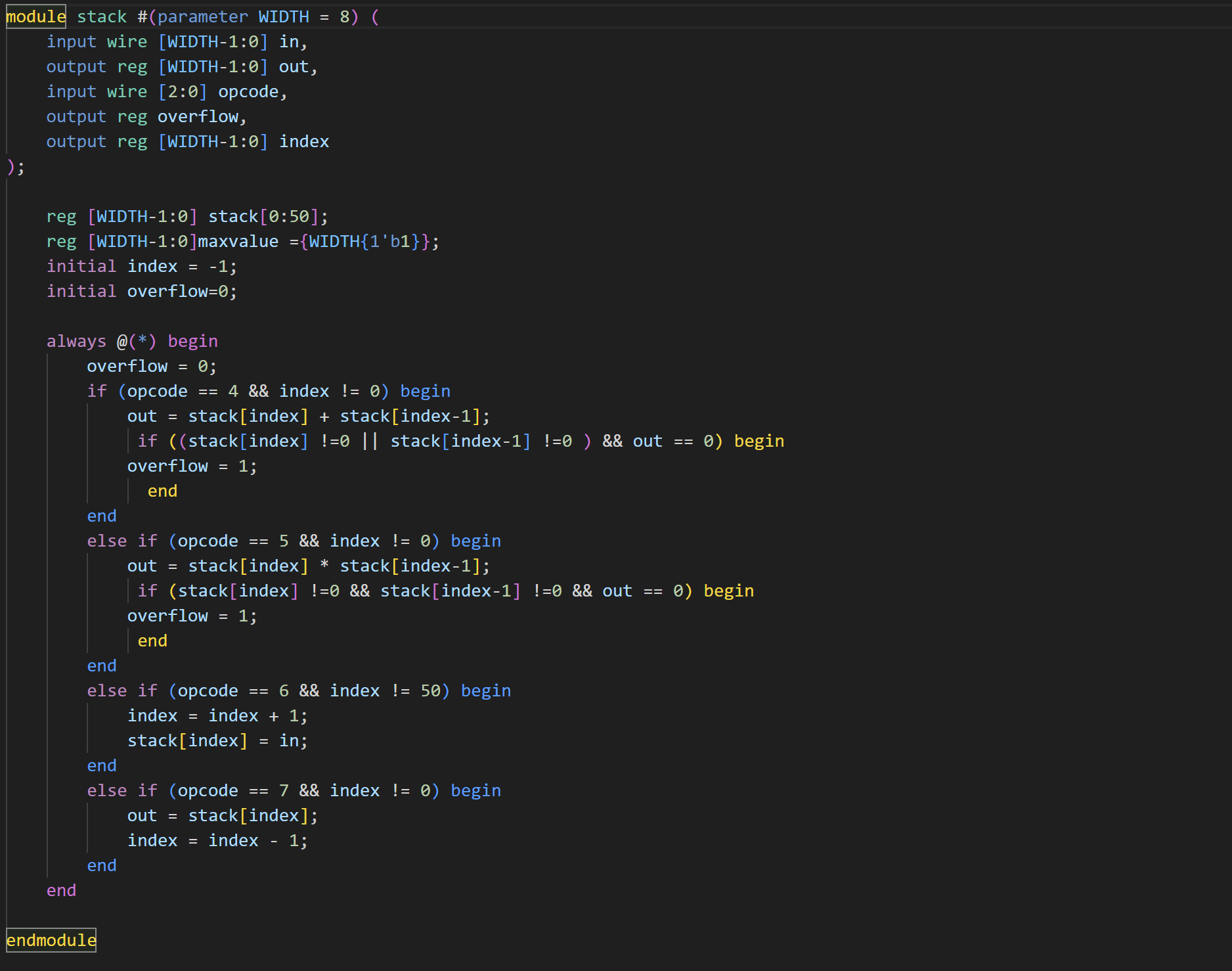
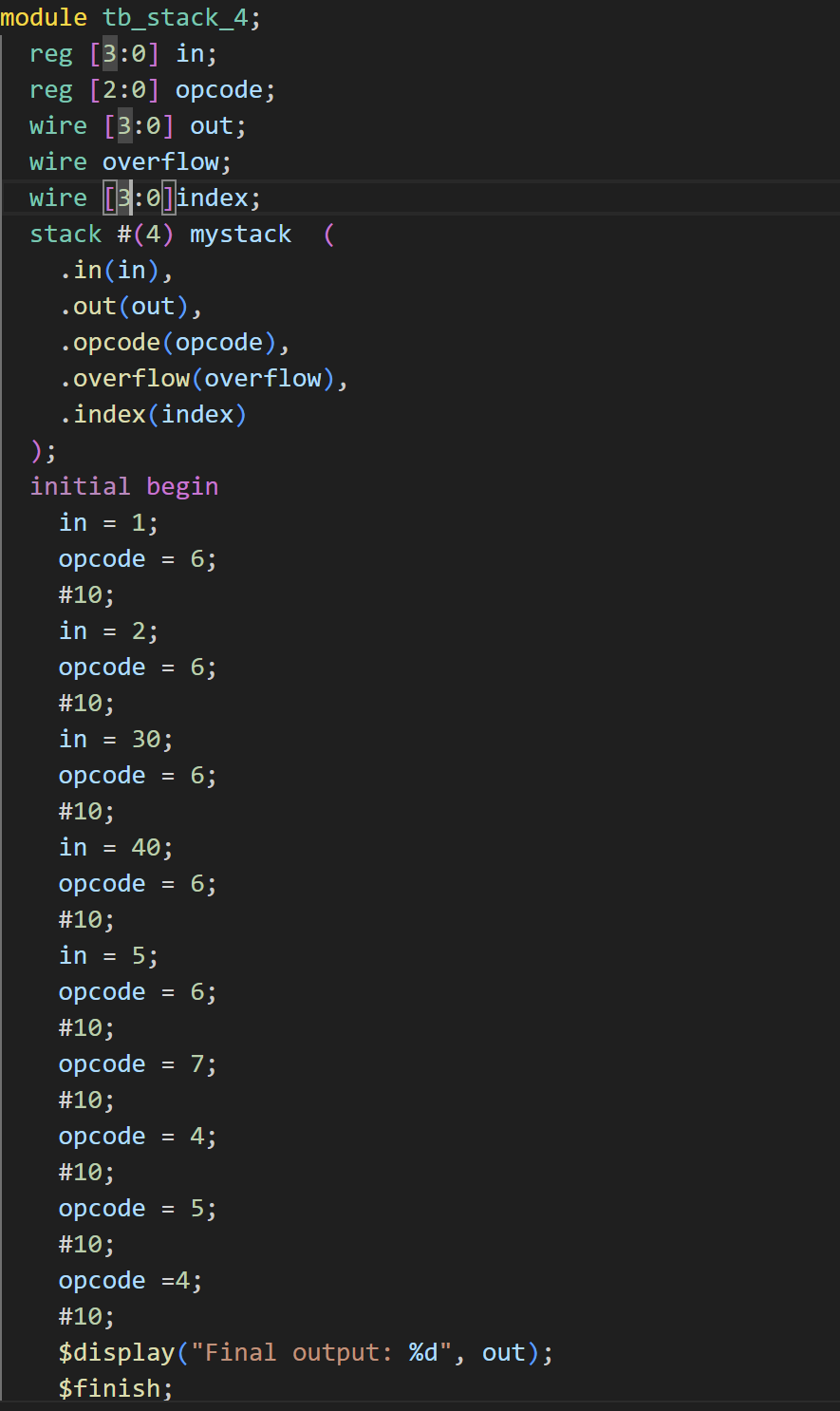
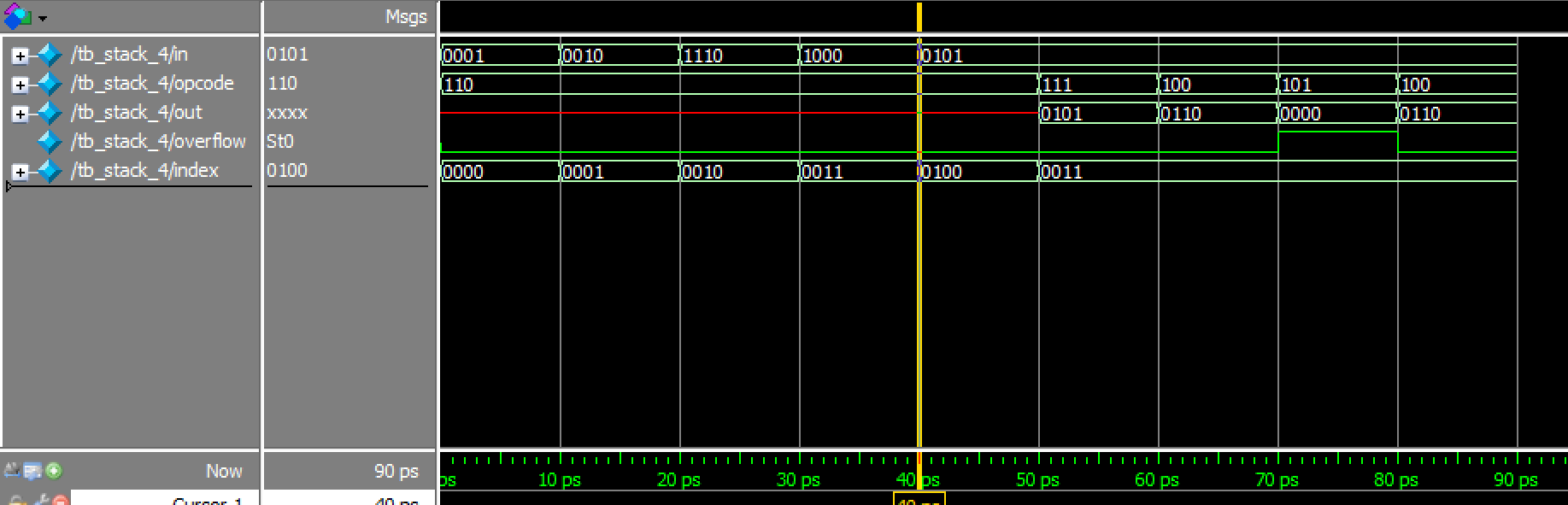
Module stack:



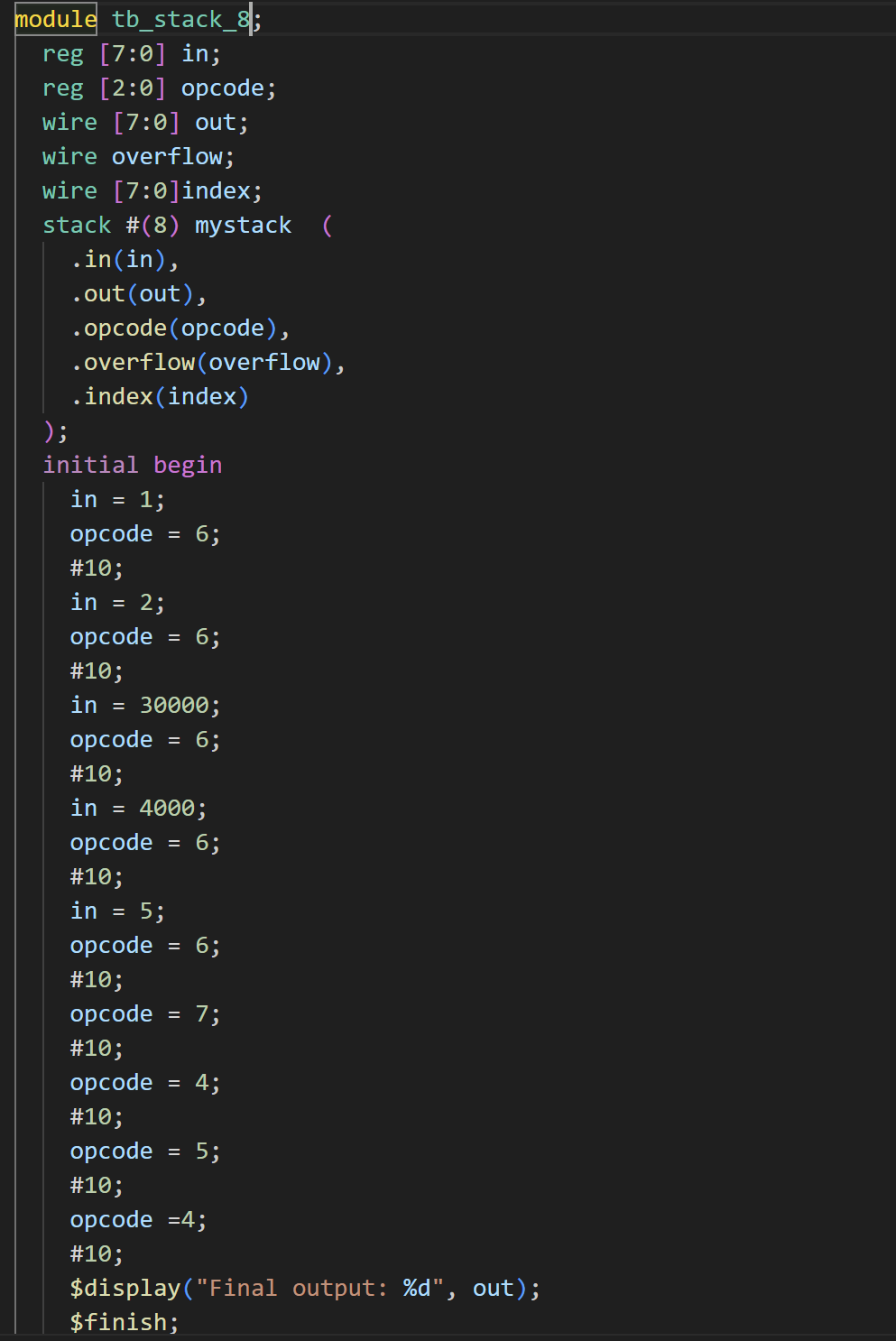
Test bench 4-bit:



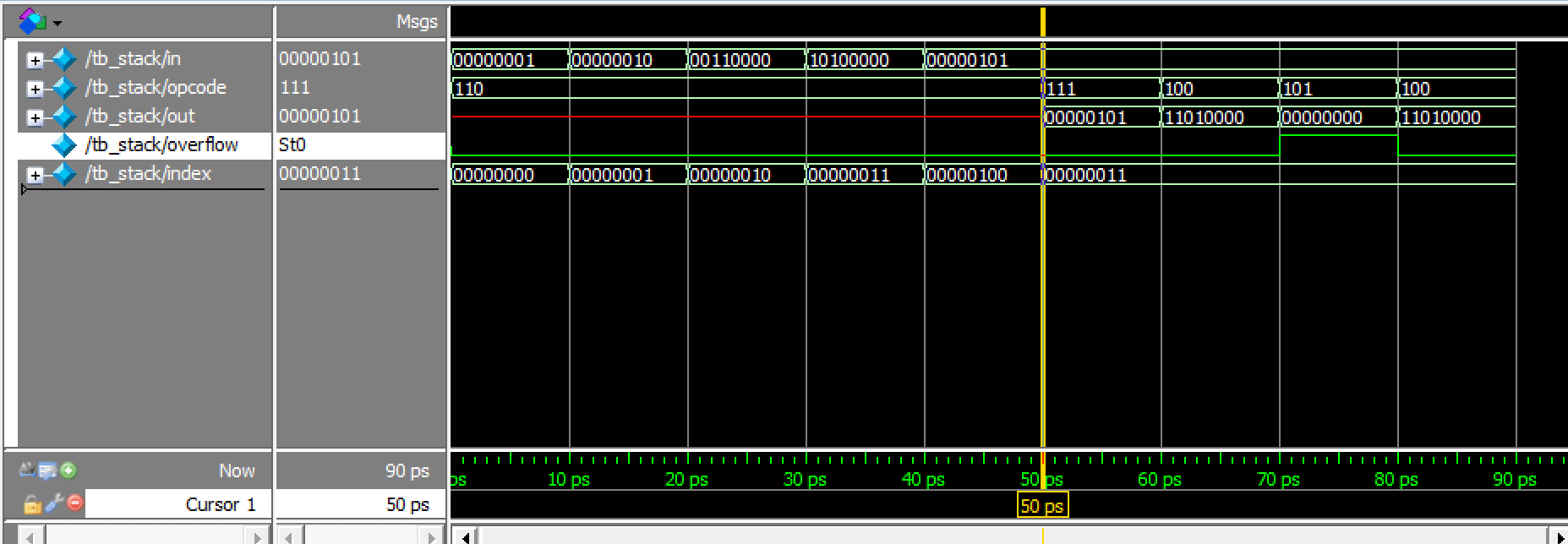
Wave form:



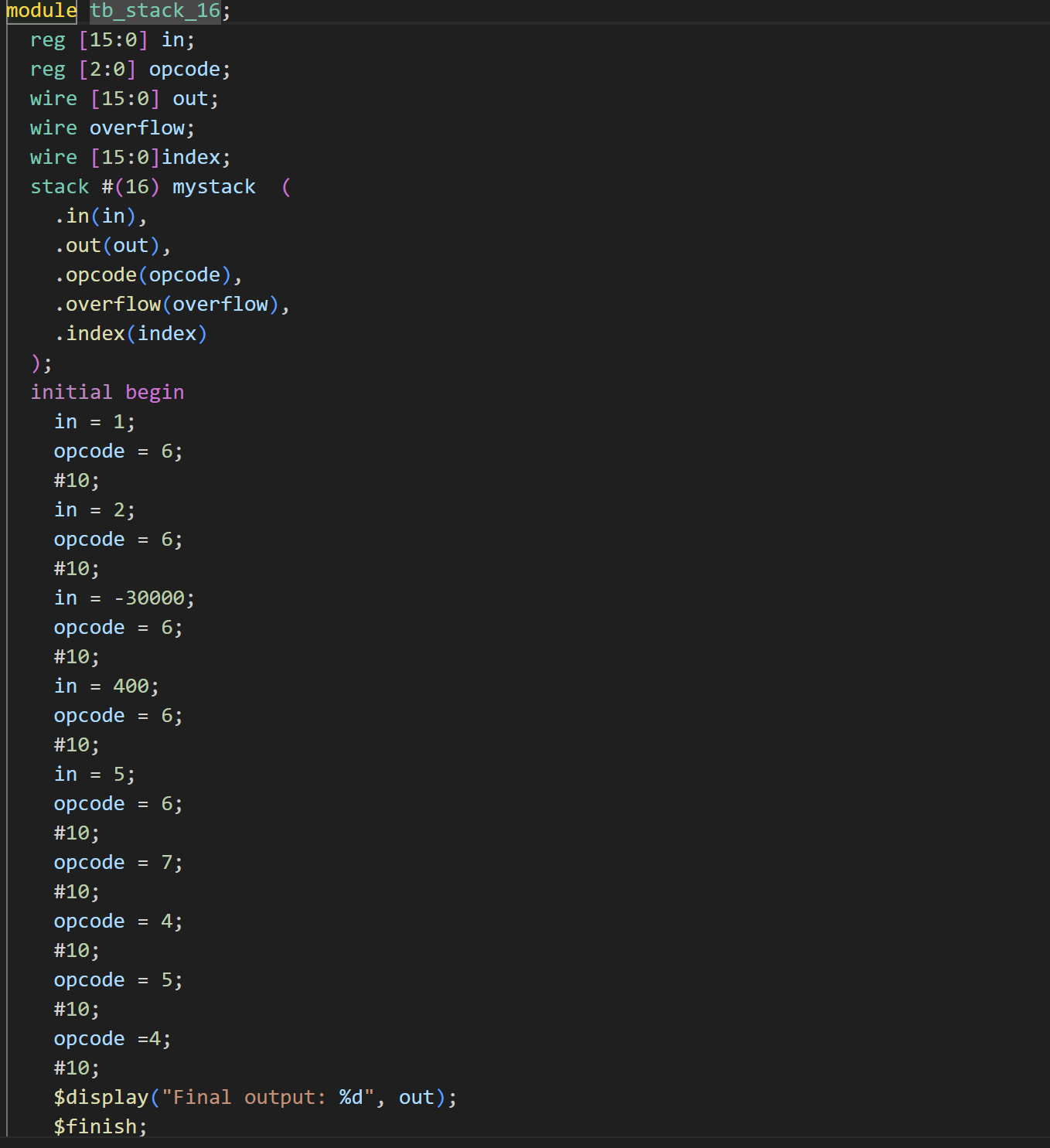
Testbench 8-bit:



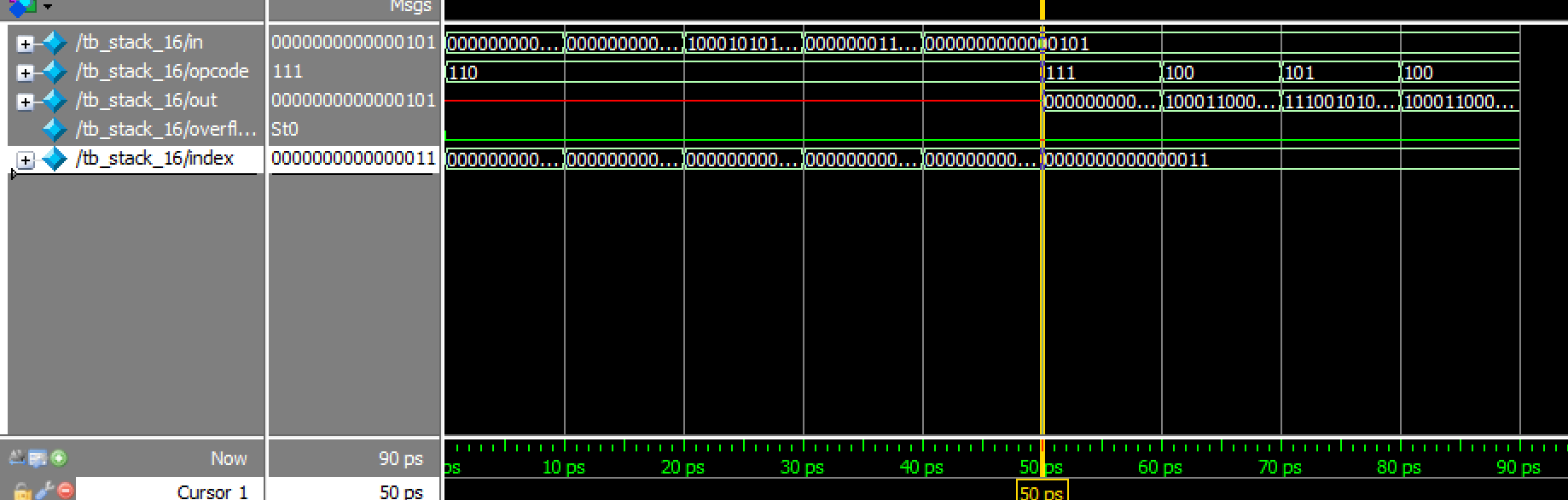
Wave form:



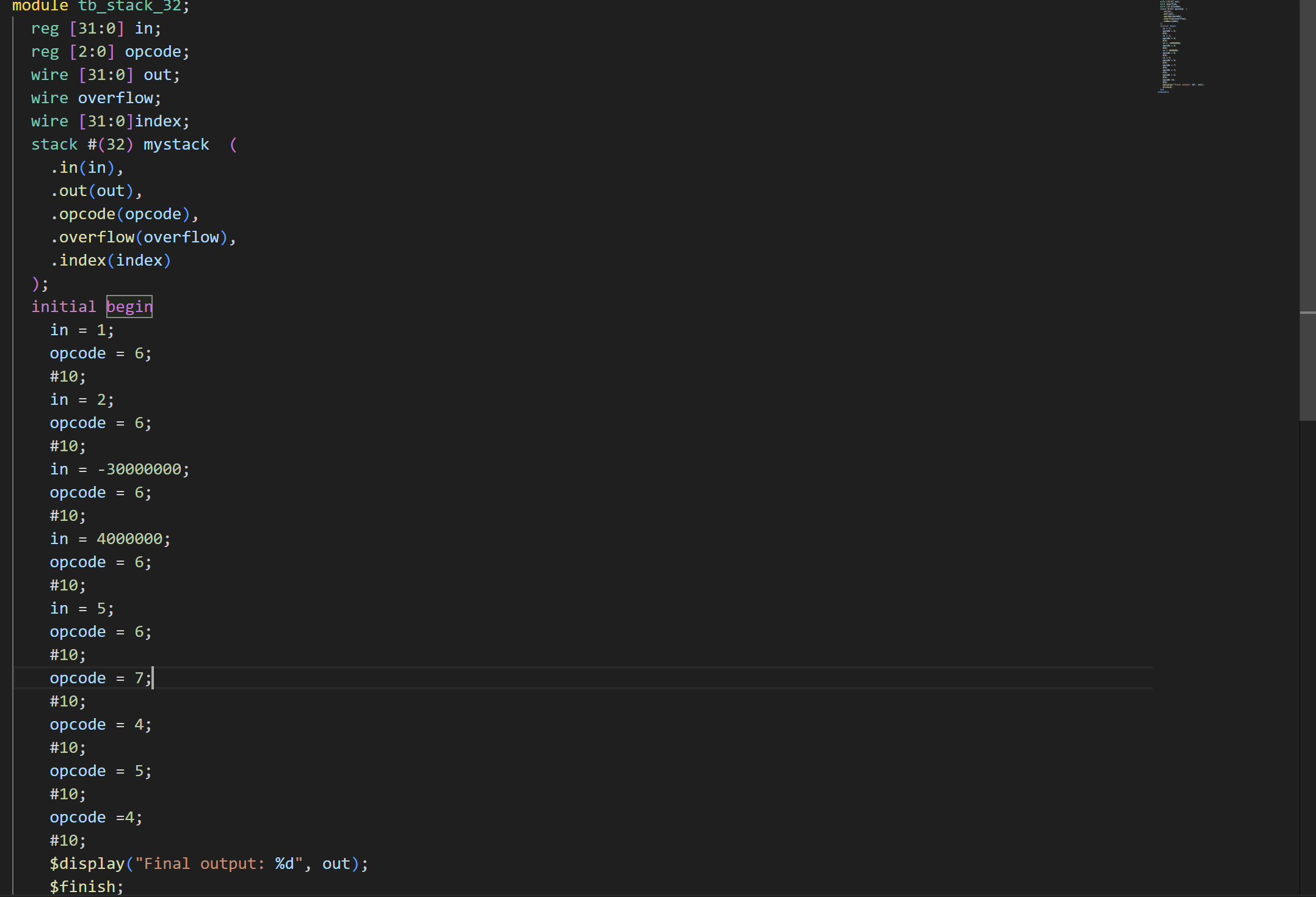
Testbench 16-bit:



Wave form:



Testbech 32bit:



Wave form:



In the stack module, we utilize a vector array with a variable size. Within an always block, we perform operations such as push, pop, addition, and multiplication based on the specified opcode. To test this module, we set parameters according to the desired bit value. We know that the module’s input ports are of type reg, and the outputs must be of type wire. Then, in an initial block, which executes sequentially and only once, we verify test cases that we suspect might be incorrect.

For the second part, the code is not complete, but the process is as follows: First, the infix expression needs to be converted to postfix. I’ve used two stacks for this purpose. All numbers are entered into one stack, while the + and \* operators are placed in another stack. When a + operator needs to be added to the stack, we consider the \* operator with higher precedence. In that case, we pop all contents from the stack2 (from top to bottom) and select two numbers from the stack1 . We then perform the operation on those two numbers. For further understanding, please refer to the two additional links provided on the GitHub homepage

Infix to postfix module:

`timescale 1ns / 1ps

module postfix(input [7:0]infix,input clk,

                    output reg [7:0]postfix);

reg [7:0] in,in2 ;

wire [7:0] out,out2;

reg [2:0] opcode,opcode2;

wire overflow,overflow2;

wire [7:0]index,index2;

stack #(8) s1 (in, out,opcode,overflow,index);

stack #(8) s2 (in2, out2,opcode2,overflow2,index2);

integer l=0;

integer k=0;

reg [7:0] temp;

always @ (infix)

begin

    if (infix != "+"&& infix!="\*"&&infix!="=")

    begin

      in= infix;

      opcode=6;

      postfix=in;

    end

    else if (infix == "\*")

    begin

        in2=infix;

        opcode2=6;

        temp="\*";

    end

    else if(infix=="+")

    begin

        if(temp=="\*")k=1;

            in2 = infix;

            opcode=6;

        temp="+";

    end

    else if(infix =="=")begin

            k=1;

    end

end

always @(posedge clk)

begin

    if( k==1)

    begin

        begin

            if(index2>=0)

            begin

                opcode2=7;

                opcode=6;

                in=out2;

                postfix=out2;

            end

            else

            begin

                k=0;

            end

        end

    end

end

endmodule

test bench:

