```
1   module LIFOstack (Din, clk, en, rst, rw, Dout);
2
3   input [15:0] Din; // Data being fed to stack
4   input clk; // clock signal input
5   input en; // disable stack when not in use
6   input rst; // reset pin to clear and reinitialise stack (active high)
7   input rw; // 0: read, 1: write
8
9   output reg [15:0] Dout; // Data being pulled from stack
10  reg empty; // goes high to indicate SP is at 0
11  reg full; // goes high to indicate SP is at (slots)
12
13  reg [5:0] SP; // Points to slot to save next value to
14  integer i;
15  reg [15:0] mem [31:0];
16
17  always @ (posedge clk) begin
18        if (!en); // if not enabled, ignore this cycle
19        else begin
20              if (rst) begin // if rst is high, clear memory and reset pointers/outputs
21                    Dout = 16'h0000;
22                    SP = 6'b000000;
23                    empty = 1'b1;
24                    for (i = 0; i < 32; i = i + 1) begin
25                       mem[i] = 16'h0000;
26                       end
27                 end
28              else begin
29                    if (!full && rw) begin // Write when NOT full & Writing
30                          mem[SP] = Din; // Store data into current slot
31                          SP = SP + 1'b1; // Increment stack pointer to next empty slot
32                          full = (SP == 6'b100000) ? 1 : 0; // Stack is full if SP is (slots)
33                          empty = 1'b0; // Stack is never empty after a push
34                       end
35                    else if (!empty && !rw) begin // Read when NOT empty & reading
36                          SP = SP - 1'b1; // Decrement stack pointer to last filled slot
37                          Dout = mem[SP]; // Output data from last filled slot
38                          mem[SP] = 16'h0000; // Clear slot after setting output
39                          full = 1'b0; // Stack is never full after a pop
40                          empty = (SP == 6'b000000) ? 1 : 0; // Stack is empty if SP is 0
41                       end
42                 end
43           end
44     end
45
46  endmodule
47
```