

جبرانی پایانترم

طراحی سیستم‌های دیجیتال



محمد داودآبادی فراهانی

401110331

# سوال 1 میانترم

## الف)

در این بخش از سوال ماژول STACK\_BASED\_ALU را پیاده‌سازی می‌کنیم که در ادامه این پیاده‌سازی را خواهیم دید.

**module** STACK\_BASED\_ALU #(parameter N = 8) (

    input signed [N-1:0] input\_data,

    input [2:0] opcode,

    output reg signed [N-1:0] output\_data,

    output reg overflow

);

*// Internal stack for ALU operations*

    reg [N-1:0] stack [0:N-1];

    integer top = -1;

    always @(\*) begin

        overflow = 0;

        output\_data = 0;

        case (opcode)

            3'b100: *// Addition*

                if (top > 0) begin

                    output\_data = stack[top] + stack[top-1];

                    overflow = ((stack[top] > 0 && stack[top-1] > 0 && output\_data <= 0) || (stack[top] < 0 && stack[top-1] < 0 && output\_data >= 0)) ? 1 : 0;

                end

            3'b101: *// Multiplication*

                if (top > 0) begin

                    output\_data = stack[top] \* stack[top-1];

                    if (stack[top] == 0 || stack[top-1] == 0) begin

                        overflow = 0;

                    end

                    else begin

                        overflow = (stack[top] == output\_data / stack[top-1]) ? 0 : 1;

                    end

                end

            3'b110: *// PUSH*

                if (top < N-1) begin

                    top = top + 1;

                    stack[top] = input\_data;

                end

            3'b111: *// POP*

                if (top >= 0) begin

                    output\_data = stack[top];

                    top = top - 1;

                end

            default: *// No Operation*

                output\_data = 0;

        endcase

    end

**endmodule**

در این کد همانطور که مشخص است 5 عمل ضرب و جمع و پوش و پاپ و Nop انجام می‌شود و در حالت جمع و ضرب درصورتی که اورفلو رخ داده باشد گفته می‌شود.

حال تست برای این ماژول می‌نویسیم.

`timescale 1ns / 1ps

**module** TB;

    reg [3:0] input\_data\_4;

    reg [7:0] input\_data\_8;

    reg [15:0] input\_data\_16;

    reg [31:0] input\_data\_32;

    reg [2:0] opcode;

    wire signed [3:0] output\_data\_4;

    wire signed [7:0] output\_data\_8;

    wire signed [15:0] output\_data\_16;

    wire signed [31:0] output\_data\_32;

    wire overflow\_4, overflow\_8, overflow\_16, overflow\_32;

*// Instantiate the module with N = 4*

    STACK\_BASED\_ALU #(.N(4)) alu\_4 (

        .input\_data(input\_data\_4),

        .opcode(opcode),

        .output\_data(output\_data\_4),

        .overflow(overflow\_4)

    );

*// Instantiate the module with N = 8*

    STACK\_BASED\_ALU #(.N(8)) alu\_8 (

        .input\_data(input\_data\_8),

        .opcode(opcode),

        .output\_data(output\_data\_8),

        .overflow(overflow\_8)

    );

*// Instantiate the module with N = 16*

    STACK\_BASED\_ALU #(.N(16)) alu\_16 (

        .input\_data(input\_data\_16),

        .opcode(opcode),

        .output\_data(output\_data\_16),

        .overflow(overflow\_16)

    );

*// Instantiate the module with N = 32*

    STACK\_BASED\_ALU #(.N(32)) alu\_32 (

        .input\_data(input\_data\_32),

        .opcode(opcode),

        .output\_data(output\_data\_32),

        .overflow(overflow\_32)

    );

    initial begin

*// Test the ALU with some operations*

        input\_data\_32 = -32'd1;

        input\_data\_16 = -16'd1;

        input\_data\_8 = -8'd1;

        input\_data\_4 = 4'd7;

        opcode = 3'b110; *// PUSH*

        #10

        input\_data\_32 = 32'd2;

        input\_data\_16 = 16'd2;

        input\_data\_8 = 8'd2;

        input\_data\_4 = 4'd1;

        #10

        opcode = 3'b100; *// Addition*

        #10

        $display("after addition 4 -> output\_4: %d\_%b, overflow\_4: %d\_%b\n", output\_data\_4, output\_data\_4, overflow\_4, overflow\_4);

        $display("after addition 8 -> output\_8: %d\_%b, overflow\_8: %d\_%b\n", output\_data\_8, output\_data\_8, overflow\_8, overflow\_8);

        $display("after addition 16 -> output\_16: %d\_%b, overflow\_16: %d\_%b\n", output\_data\_16, output\_data\_16, overflow\_16, overflow\_16);

        $display("after addition 32 -> output\_32: %d\_%b, overflow\_32: %d\_%b\n", output\_data\_32, output\_data\_32, overflow\_32, overflow\_32);

        opcode = 3'b101; *// Multiplication*

        #10

        $display("after multiplication 4 -> output\_4: %d\_%b, overflow\_4: %d\_%b\n", output\_data\_4, output\_data\_4, overflow\_4, overflow\_4);

        $display("after multiplication 8 -> output\_8: %d\_%b, overflow\_8: %d\_%b\n", output\_data\_8, output\_data\_8, overflow\_8, overflow\_8);

        $display("after multiplication 16 -> output\_16: %d\_%b, overflow\_16: %d\_%b\n", output\_data\_16, output\_data\_16, overflow\_16, overflow\_16);

        $display("after multiplication 32 -> output\_32: %d\_%b, overflow\_32: %d\_%b\n", output\_data\_32, output\_data\_32, overflow\_32, overflow\_32);

        opcode = 3'b111; *// POP*

        #10

        $display("after pop 4 -> output\_4: %d\_%b, overflow\_4: %d\_%b\n", output\_data\_4, output\_data\_4, overflow\_4, overflow\_4);

        $display("after pop 8 -> output\_8: %d\_%b, overflow\_8: %d\_%b\n", output\_data\_8, output\_data\_8, overflow\_8, overflow\_8);

        $display("after pop 16 -> output\_16: %d\_%b, overflow\_16: %d\_%b\n", output\_data\_16, output\_data\_16, overflow\_16, overflow\_16);

        $display("after pop 32 -> output\_32: %d\_%b, overflow\_32: %d\_%b\n", output\_data\_32, output\_data\_32, overflow\_32, overflow\_32);

        opcode = 3'b000; *// No Operation*

        #10;

        $finish;

    end

**endmodule**

در این تست 4 instance از ماژول می‌گیریم با سایزهای مورد نیاز و در 4 بیتی ضرب و جمع را روی 1 و 7 انجام می‌دهیم و برای بقیه روی 2 و منفی 1 انجام می‌دهیم.

در ابتدا همه‌ی این اعداد گفته شده را روی استک پوش می‌کنیم سپس عملیات ضرب و جمع را انجام می‌دهیم و در آخر یکی از اعداد را پاپ می‌کنیم و یک Nop انجام می‌دهیم تا از صحت عملکرد این دستورها مطمئن شویم.

انتظار داریم که جمع 1 و 7 و ضرب 2 و منفی 1 اورفلو کند ولی برای باقی حالت‌ها اورفلو رخ ندهد.

خروجی برنامه در عکس زیر مشخص است.

