بسمه تعالی

آزمایش شماره ۱۰

استاد

دکتر علیرضا اجلالی

سینا مظاهری

متین داغیانی

دانشگاه صنعتی شریف

۱۴۰۰-۱۳۹۹

مقدمه و اهداف آزمایش

هدف از انجام این آزمایش طراحی یک پردازنده ی پشته ای ساده می باشد که در مجموعه دستورالعمل های خود، ۸ دستور مرتبط با معماری پشته ای در آن تعریف شده است و از عملیات I/O و همچنین نمایش اعداد بر روی Seven Segment نیز پشتیبانی می کند.

تئوری آزمایش

به منظور پیاده سازی این معماری پشته ای ۸ ثبات ، ۸ بیتی به عنوان پشته در نظر گرفته شده که در مسیر داده وجود دارد و نیز برای آن ماژول جداگانه ای در نظر گرفته شده تا عملکرد آن را توصیف نماید. این ماشین چهار ثبات به نام های IR ، PC ، A و B دارد که به ترتیب دستور العمل ،آدرس دستور العملی که می خواهد اجرا شود، عملوند اول و عملوند دوم برای جمع و تفریق می باشند. همچنین یک حافظه 12×256 برای آن درنظر گرفته شده است. کلیه عملیات های مورد نظر در ماژولBasicCPU تعریف و دیکد می شوند. از سیگنال های T_i نیزبه منظور رعایت توالی عملیات های ثباتی برای یک دستور العمل خاص، نیز درنظر گرفته شده است. این ماژول ، دو ماژول پایین دستی به نام BCDConverter و SevenSegment در خود دارد که حاصل عملیات دودویی در مبنای ۲ را پس از پاپ کردن به BCDتبدیل نموده و آن را برروی SevenSegment نمایش می دهد.

کد های توصیف سخت افزار

کد توصیف BasicCPU.v:

```
module BasicCPU(Clock, reset, write ,din, err, digit_0, digit_1, digit_2, stack_input, T, inst,
push, pop);


        output [6:0] digit_0, digit_1, digit_2;
        output wire err;
        output reg [7:0] stack_input;
        output wire [7:0] T;
        output wire [11:0] inst;
        output push, pop;
        input Clock, reset, write;
        input [7:0] din;

        reg [11:0] ram [255:0];
        reg [11:0] IR;
        reg [7:0] PC, A, B, result;
        reg push, pop, S, Z;
        reg T0, T1, T2, T3, T4;
        wire full, empty, reset_not;
        wire [7:0] stack_out;
        wire [3:0] d0, d1, d2;

        integer i;
```

```
BCD bcd(stack_input,d2, d1, d0);
Stack stack(reset_not, stack_input, push, pop, stack_out, full, empty, T);
SevenSegment s0(digit_0, d0), s1(digit_1, d1), s2(digit_2, d2);


assign inst = ram[PC];
assign reset_not = ~reset;
assign err = (stack_input >127) || (stack_input[7] == 1);
always @(posedge Clock, posedge reset) begin
        if(reset) begin
                ram[0] <= 12'b_0000_00010111; //push constant 23
                ram[1] <= 12'b_0001_11111111; //push input data (saved in FF) to stack
                ram[2] <= 12'b_0110_00000000; //add
                ram[3] <= 12'b_0110_00000000; //add again (multiply 2)
                ram[4] <= 12'b_0010_00000000; //save to memory
                ram[5] <= 12'b_0000_00001100; //push constant 12
                ram[6] <= 12'b_0001_00000000; //load saved number
                ram[7] <= 12'b_0111_00000000; //subtract
                for(i = 8; i < 8'hFF; i = i+1)
                        ram[i] <= 12'hF_00;
                ram[255] <= (write)? {4'b0, din} : 12'b_1111_00000101;

                T0 <= 1;  //first clock
                T1 <= 0;  //second clock
                T2 <= 0;  //third clock
                T3 <= 0;  //fourth clock
                T4 <= 0;  //fifth clock

                S <= 0;  //sign flag
                Z <= 0;         //zero flag
                PC <= 0;
                IR <= ram[0];
                stack_input <= 0;
        end else begin
                        case(IR[11:8])
                                4'b0000: begin //push constant
                                        pop <= 0;
                                        push <= 1;
                                        stack_input <= IR[7:0];
                                        IR <= ram[PC+1];
                                        PC <= PC+1;
                                end
                                4'b0001: begin //push from memory
                                        if(T0) begin
                                            push <= 0;
                                            pop <= 0;
                                                stack_input <= ram[IR[7:0]];
                                                T0 <= 0;
                                                T1 <= 1;
                                        end else if (T1) begin
                        push <= 1;
                        pop <= 0;
                                                IR <= ram[PC+1];
                                                PC <= PC+1;
                                                T0 <= 1;
                                                T1 <= 0;
                                        end
                                end
                                4'b0010: begin //pop to memory
                                        if(T0) begin
                                                pop <= 1;
                                                push <= 0;
                                                T0 <= 0;
                                                T1 <= 1;
                                        end else if (T1)begin
                                            push <= 0;
                                            pop <= 0;
                        ram[IR[7:0]] <= stack_out;
                                                T0 <= 1;
                                                T1 <= 0;
                                                IR <= ram[PC+1];
                                                PC <= PC+1;
```

```
                end
            end
            4'b0011: begin //jump
                if(T0) begin
                        pop <= 1;
                        push <= 0;
                        T0 <= 0;
                        T1 <= 1;
                end else if (T1)begin
                    push <= 0;
                    pop <= 0;
PC <= stack_out;
                        T0 <= 1;
                        T1 <= 0;
                        IR <= ram[PC+1];
                        PC <= PC+1;
                end
            end
            4'b0100: begin //jump if zero
                if(Z)begin
                        if(T0) begin
                                pop <= 1;
                                push <= 0;
                                T0 <= 0;
                                T1 <= 1;
                        end else if (T1)begin
                                push <= 0;
                            pop <= 0;
                                PC <= stack_out;
                                T0 <= 1;
                                T1 <= 0;
                                IR <= ram[PC+1];
                                PC <= PC+1;
                        end
                end else begin
                        pop <= 0;
                        push <= 0;
                        IR <= ram[PC+1];
                        PC <= PC+1;
                end
            end
            4'b0101: begin //jump if negative
                if(S)begin
                        if(T0) begin
                                pop <= 1;
                                push <= 0;
                                T0 <= 0;
                                T1 <= 1;
                        end else if (T1)begin
                            push <= 0;
                         pop <= 0;
                                PC <= stack_out;
                                T0 <= 1;
                                T1 <= 0;
                                IR <= ram[PC+1];
                                PC <= PC+1;
                        end
                end else begin
                        pop <= 0;
                        push <= 0;
                        IR <= ram[PC+1];
                        PC <= PC+1;
                end
            end
            4'b0110: begin //add
                if(T0) begin
                        pop <= 1;
                        push <= 0;
                        T0 <= 0;
                        T1 <= 1;
                end else if (T1)begin
                    push <= 0;
```

```
                pop <= 0;
                A <= stack_out;
                    T1 <= 0;
                    T2<= 1;
            end else if (T2) begin
                pop <= 1;
                    push <= 0;
                    T2 <= 0;
                    T3<= 1;
            end else if(T3) begin
                push <= 0;
                pop <= 0;
                B <= stack_out;
                    T3 <= 0;
                    T4 <= 1;
            end else if (T4) begin
                    pop <= 0;
                    push <= 0;
                    stack_input <= A+B;
                    T4 = 0;
            end
            else begin
                push = 1;
                pop = 0;
                    T0 <= 1;
                    if((A+B) == 0)
                            Z <= 1;
                    else
                            Z <= 0;
                    if((A+B) < 0)
                            S <= 1;
                    else
                            S <= 0;
                    IR <= ram[PC+1];
                    PC <= PC+1;
            end
    end

    4'b0111: begin //sub
                    if(T0) begin
                    pop <= 1;
                    push <= 0;
                    T0 <= 0;
                    T1 <= 1;
            end else if (T1)begin
                push <= 0;
                pop <= 0;
                A <= stack_out;
                    T1 <= 0;
                    T2<= 1;
            end else if (T2) begin
                pop <= 1;
                    push <= 0;
                    T2 <= 0;
                    T3<= 1;
            end else if(T3) begin
                push <= 0;
                pop <= 0;
                B <= stack_out;
                    T3 <= 0;
                    T4 <= 1;
            end else if (T4) begin
                    pop <= 0;
                    push <= 0;
                    stack_input <= A - B;
                    T4 = 0;
            end
            else begin
                push = 1;
                pop = 0;
                    T0 <= 1;
                    if((A+B) == 0)
```

```
                                                        Z <= 1;
                                                else
                                                        Z <= 0;
                                                if((A+B) < 0)
                                                        S <= 1;
                                                else
                                                        S <= 0;
                                                IR <= ram[PC+1];
                                                PC <= PC+1;
                                        end

                                end
                        endcase

                end
        end
```

کد توصیف Stack:

```
module Stack(reset not,din,Push,Pop,dout,full,empty, T);
        input reset_not, Push, Pop;
        input [7:0] din;
        output reg [7:0] dout;
        output full, empty;
        output wire [7:0] T;

        reg [31:0] top;
        reg [7:0] mem[7:0];
        integer i;
        assign T = top;
        always @(Push, Pop, negedge reset_not) begin
                if(!reset_not)begin
                        for(i = 0; i < 8; i = i + 1) begin
                                mem[i] <= 0;
                        end
                        dout <= 0;
                        top <= 0;
                end else begin
                        case({Push, Pop})
                                2'b10:begin
                                        if(!full) begin
                                                mem[top] <= din;
                                                top <= top+1;
                                        end
                                end
                                2'b01:begin
                                        if(!empty)begin
                                                dout <= mem[top-1];
                                                top <= top-1;
                                        end
                                end
                        endcase
                end
        end

        assign full = (top == 8);
        assign empty = (top == 0);
endmodule
```

کد توصیف **BinaryToBCD** :

```
module BCD(number, hundreds, tens, ones);
   input  [7:0] number;
   output reg [3:0] hundreds;
   output reg [3:0] tens;
   output reg [3:0] ones;

   reg [19:0] shift;
   integer i;

   always @(number)
   begin
      shift[19:8] = 0;
      shift[7:0] = number;

      // Loop eight times
      for (i=0; i<8; i=i+1) begin
         if (shift[11:8] >= 5)
            shift[11:8] = shift[11:8] + 3;

         if (shift[15:12] >= 5)
            shift[15:12] = shift[15:12] + 3;

         if (shift[19:16] >= 5)
            shift[19:16] = shift[19:16] + 3;

         shift = shift << 1;
      end
      hundreds = shift[19:16];
      tens     = shift[15:12];
      ones     = shift[11:8];
   end

endmodule
```

کد توصیف **SevenSegment**:

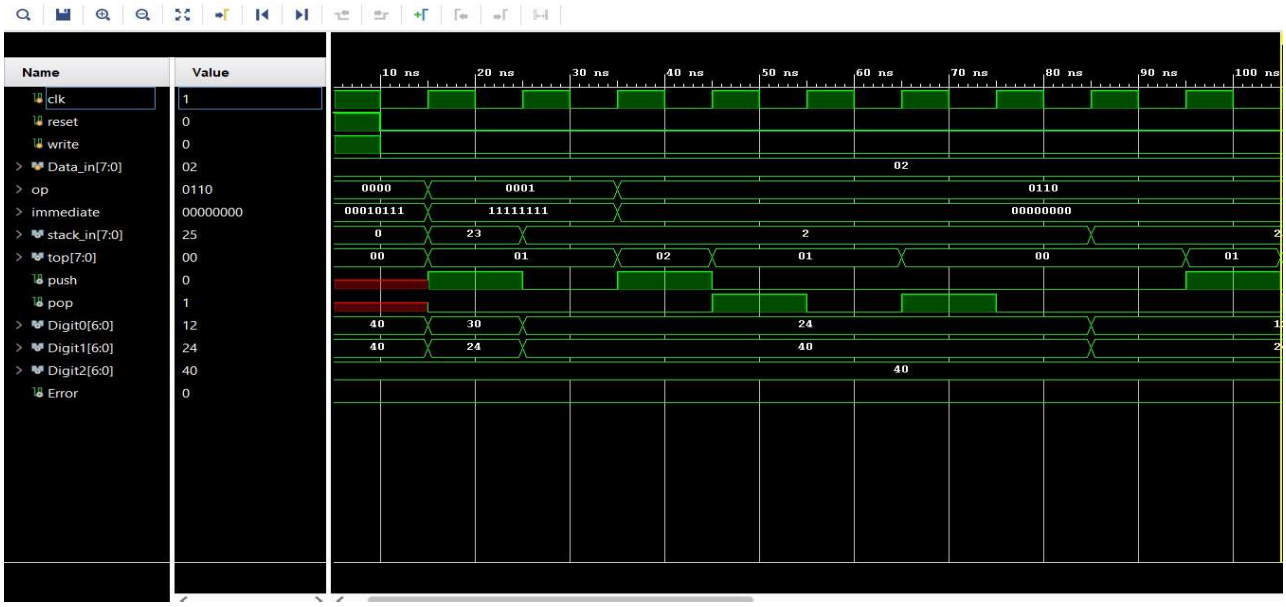
```
module SevenSegment(ssOut, nIn);
        output reg [6:0] ssOut;
        input [3:0] nIn;

        always @(nIn)
                case (nIn)
                        4'h0: ssOut = 7'b1000000;
                        4'h1: ssOut = 7'b1111001;
                        4'h2: ssOut = 7'b0100100;
                        4'h3: ssOut = 7'b0110000;
                        4'h4: ssOut = 7'b0011001;
                        4'h5: ssOut = 7'b0010010;
                        4'h6: ssOut = 7'b0000010;
                        4'h7: ssOut = 7'b1111000;
                        4'h8: ssOut = 7'b0000000;
                        4'h9: ssOut = 7'b0011000;
                        4'hA: ssOut = 7'b0001000;
                        4'hB: ssOut = 7'b0000011;
                        4'hC: ssOut = 7'b1000110;
                        4'hD: ssOut = 7'b0100001;
                        4'hE: ssOut = 7'b0000110;
                        default: ssOut = 7'b0110110;
    endcase
endmodule
```

شبیه سازی آزمایشگاهی:

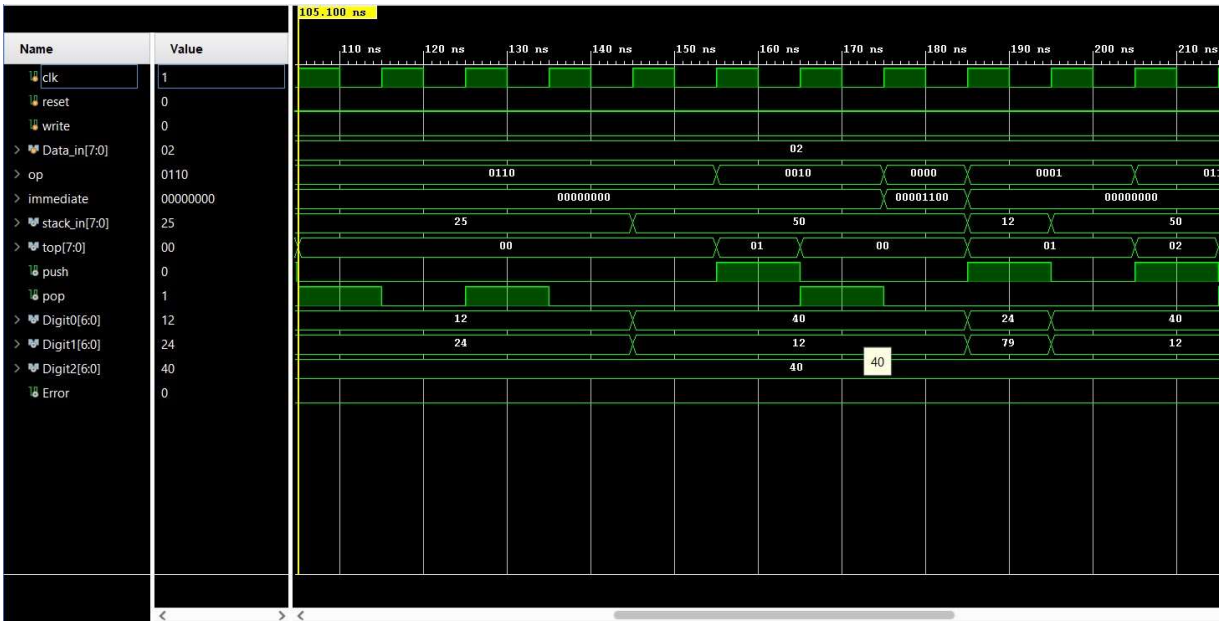
برای چنین توصیفی یک تست طرح نوشته شده به انضمام این که تمامی کد ها در پیوست فایل مورد نظر آمده اند.

```
module TestBench();
	reg clk, reset, write;
	reg [7:0] Data_in;

	wire [7:0] stack_in;
	wire [7:0] top;
	wire [11:0] inst;
	wire [6:0] Digit0, Digit1, Digit2;
	wire Error;
	wire push, pop;

	BasicCPU myCPU(clk, reset, write, Data_in, Error, Digit0, Digit1, Digit2, stack_in, top,
inst, push, pop);

	initial begin
		clk = 0;
		reset = 1;
		Data_in = 2;
		write = 1;
		#10
		reset = 0;
		write = 0;
		#10
		write = 0;
		#300
		$stop;
	end

	always #5 clk = ~clk;
endmodule
```

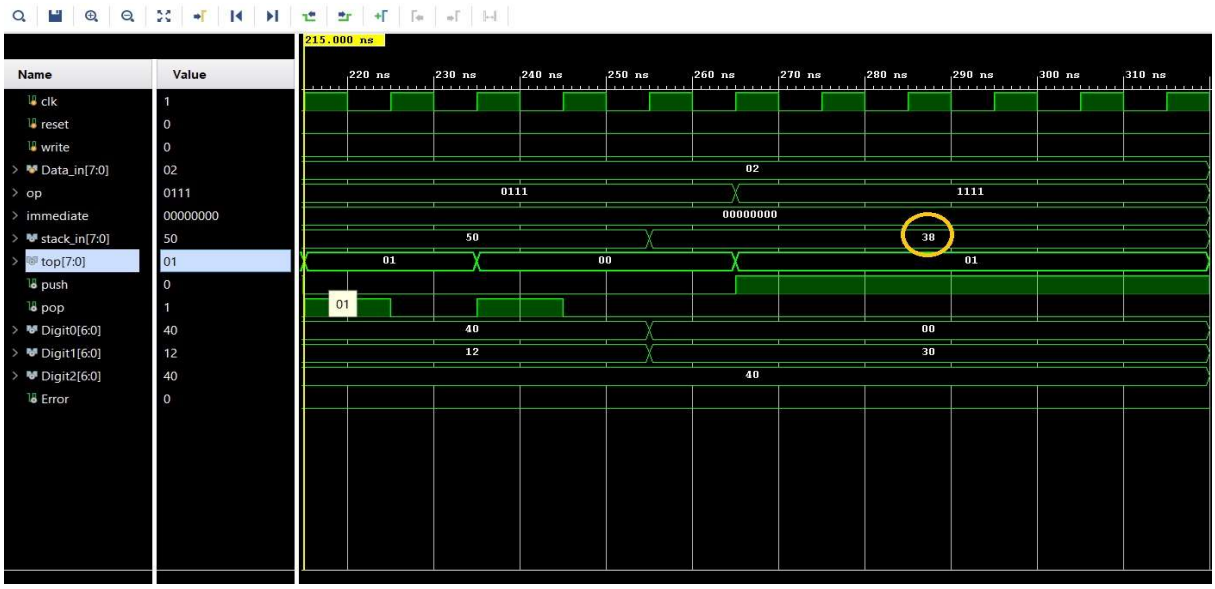
نتایج شکل موج:



شکل ۲


105.100 ns
Name
Value
clk
1
reset
0
write
0
Data_in[7:0]
02
op
0110
immediate
00000000
stack_in[7:0]
25
top[7:0]
00
push
0
pop
1
Digit0[6:0]
12
Digit1[6:0]
24
Digit2[6:0]
40
Error
0
110 ns
120 ns
130 ns
140 ns
150 ns
160 ns
170 ns
180 ns
190 ns
200 ns
210 ns
02
0110
0010
0000
0001
00000000
00001100
00000000
25
50
12
50
00
01
00
01
02
12
40
24
40
24
12
79
12
40
40

شکل ۳


215.000 ns
Name
Value
clk
1
reset
0
write
0
Data_in[7:0]
02
op
0111
immediate
00000000
stack_in[7:0]
50
top[7:0]
01
push
0
pop
1
Digit0[6:0]
40
Digit1[6:0]
12
Digit2[6:0]
40
Error
0
220 ns
230 ns
240 ns
250 ns
260 ns
270 ns
280 ns
290 ns
300 ns
310 ns
02
0111
1111
00000000
50
38
01
00
01
01
40
00
12
30
40