



NMJ31804 – Principles of Computer Architecture
SEMESTER 2, 2021/22

LAB 4) Design of Memory Module RAM and ROM

Name: TAN YI JIE

Matric Number: 191020976

Program code: RK20 - Computer Engineering

Task 1

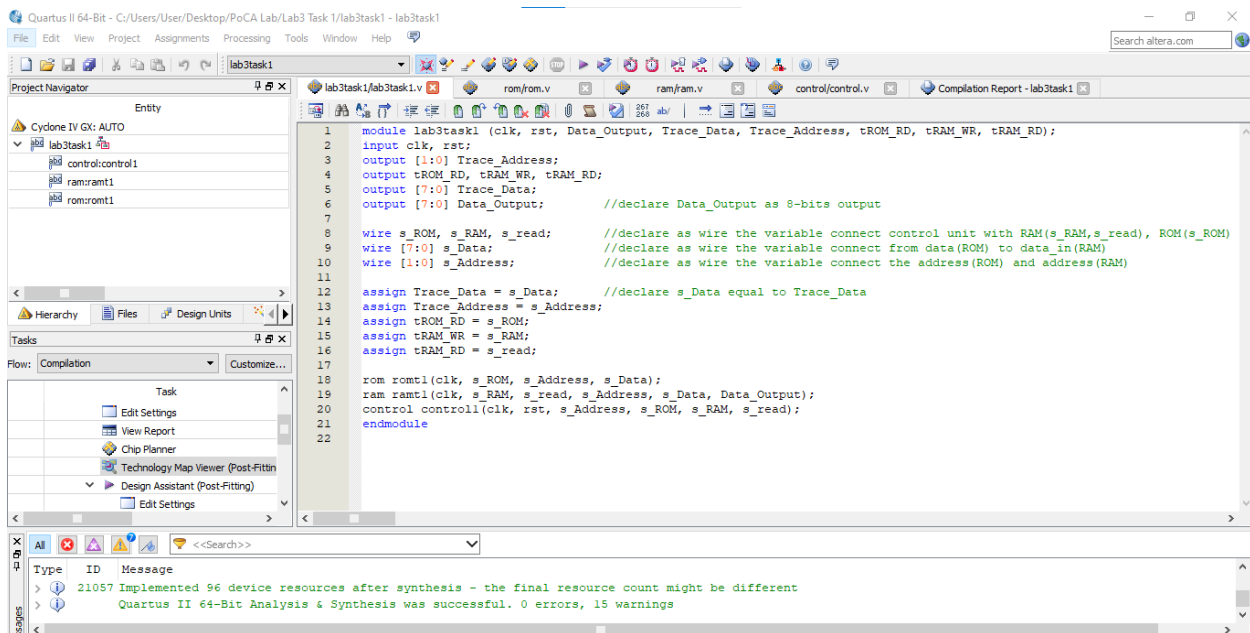


Figure 1: Verilog Code for lab3task1 Module

Figure 1 shows that the compilation of the lab3task1 module is successful. This module is the top level module which is the integration of all module.

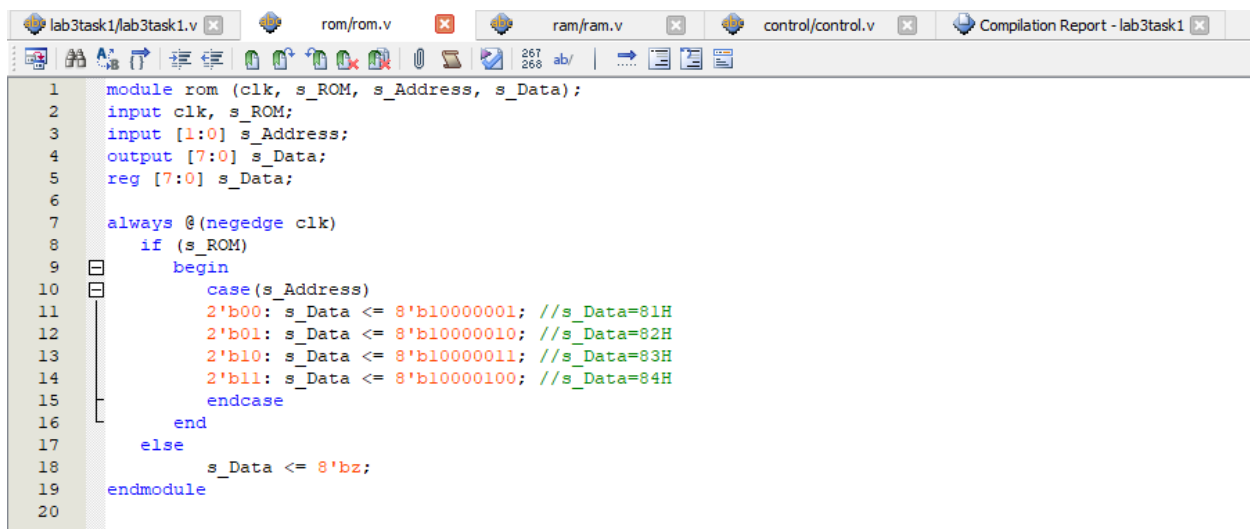
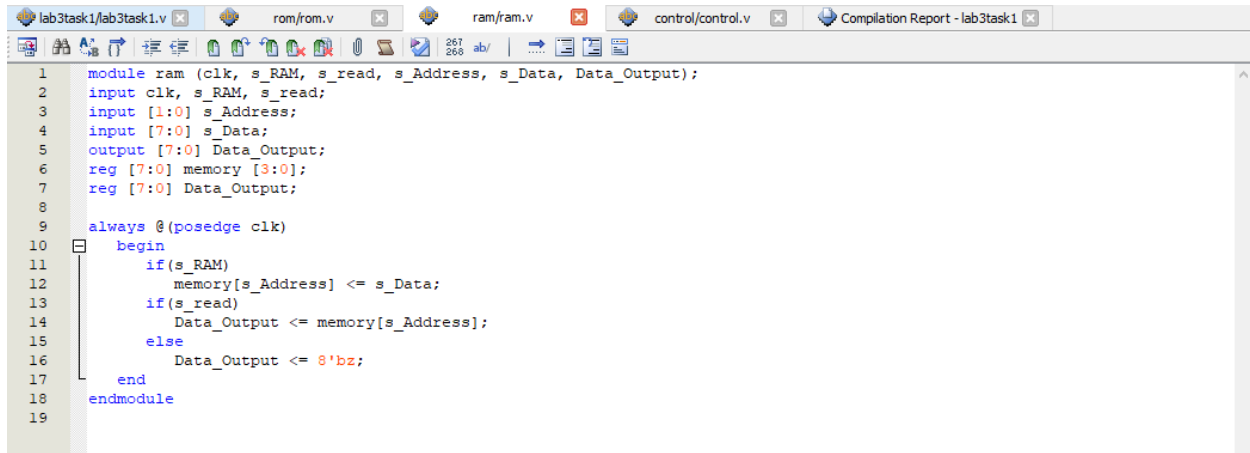


Figure 2: Verilog Code for rom Module

Figure 2 shows the Verilog code for module of ROM.



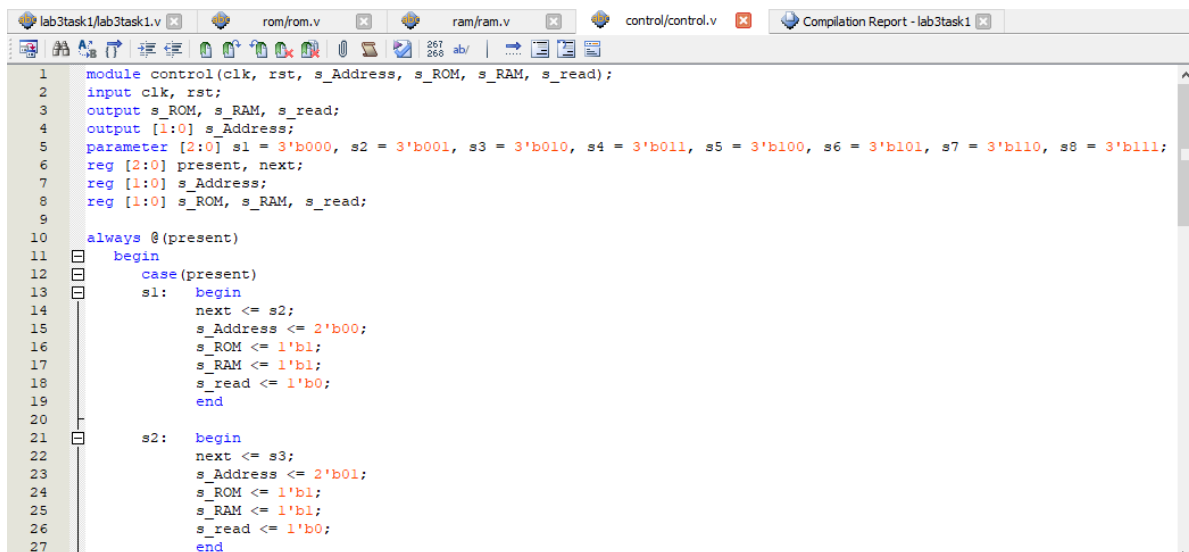
```

1 module ram (clk, s_RAM, s_read, s_Address, s_Data, Data_Output);
2 input clk, s_RAM, s_read;
3 input [1:0] s_Address;
4 input [7:0] s_Data;
5 output [7:0] Data_Output;
6 reg [7:0] memory [3:0];
7 reg [7:0] Data_Output;
8
9 always @(posedge clk)
10 begin
11     if(s_RAM)
12         memory[s_Address] <= s_Data;
13     if(s_read)
14         Data_Output <= memory[s_Address];
15     else
16         Data_Output <= 8'bz;
17 end
18 endmodule
19

```

Figure 3: Verilog Code for ram Module

Figure 3 shows the Verilog code for module of RAM.



```

1 module control(clk, rst, s_Address, s_ROM, s_RAM, s_read);
2 input clk, rst;
3 output s_ROM, s_RAM, s_read;
4 output [1:0] s_Address;
5 parameter [2:0] s1 = 3'b000, s2 = 3'b001, s3 = 3'b010, s4 = 3'b011, s5 = 3'b100, s6 = 3'b101, s7 = 3'b110, s8 = 3'b111;
6 reg [2:0] present, next;
7 reg [1:0] s_Address;
8 reg [1:0] s_ROM, s_RAM, s_read;
9
10 always @(present)
11 begin
12     case(present)
13     s1: begin
14         next <= s2;
15         s_Address <= 2'b00;
16         s_ROM <= 1'b1;
17         s_RAM <= 1'b1;
18         s_read <= 1'b0;
19     end
20
21     s2: begin
22         next <= s3;
23         s_Address <= 2'b01;
24         s_ROM <= 1'b1;
25         s_RAM <= 1'b1;
26         s_read <= 1'b0;
27     end
28

```

```

28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87

s3: begin
    next <= s4;
    s_Address <= 2'b10;
    s_ROM <= 1'b1;
    s_RAM <= 1'b1;
    s_read <= 1'b0;
end

s4: begin
    next <= s5;
    s_Address <= 2'b11;
    s_ROM <= 1'b1;
    s_RAM <= 1'b1;
    s_read <= 1'b0;
end

s5: begin
    next <= s6;
    s_Address <= 2'b00;
    s_ROM <= 1'b0;
    s_RAM <= 1'b0;
    s_read <= 1'b1;
end

s6: begin
    next <= s7;
    s_Address <= 2'b01;
    s_ROM <= 1'b0;
    s_RAM <= 1'b0;
    s_read <= 1'b1;
end

s7: begin
    next <= s8;
    s_Address <= 2'b10;
    s_ROM <= 1'b0;
    s_RAM <= 1'b0;
    s_read <= 1'b1;
end

s8: begin
    next <= s1;
    s_Address <= 2'b11;
    s_ROM <= 1'b0;
    s_RAM <= 1'b0;
    s_read <= 1'b1;
end
endcase
end

always @(posedge clk or negedge rst)
begin
    if(rst == 0)

        present <= s1;
    else
        present <= next;
    end
end
endmodule

```

Figure 4: Verilog Code for control Module

Figure 4 shows the Verilog code for module of control unit which is the finite state machine.

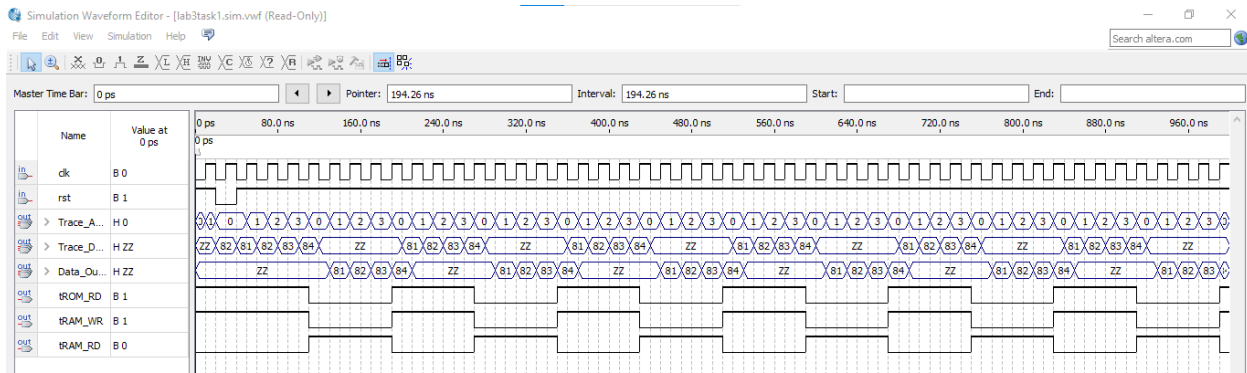


Figure 5 shows the simulation result for the lab3task1 module. All the output data for Trace_Address, Trace_Data, and Data_Output will have specific result in hexadecimal radix and the result will reflect it during each of the positive edge of the clock. During the second negative edge cycle of the clock, the system will read the value of Trace_Data from ROM and store in RAM, then it will read the RAM value after the sixth positive edge cycle of the clock such as 80, 81, 82, and 83. Otherwise, the output data for Trace_Data and Data_Output will show “ZZ”.

Task 2

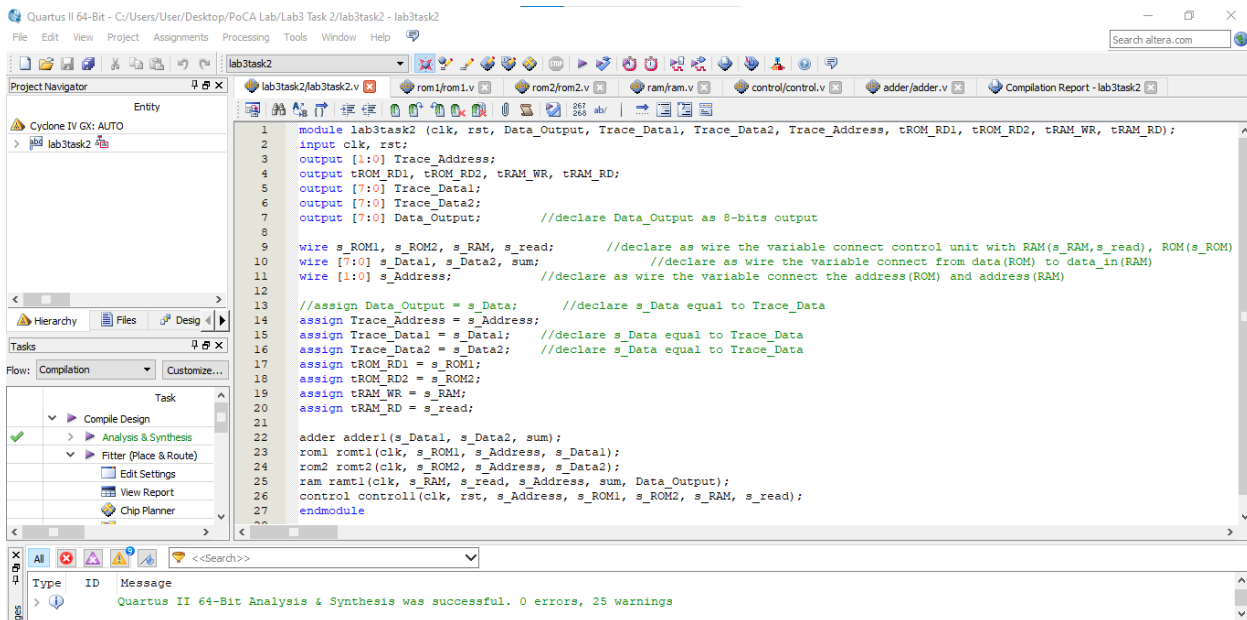


Figure 6 shows that the compilation of the lab3task2 module is successful. This module is the top level module which is the integration of all module.

```

1 module rom1(clk, s_ROM1, s_Address, s_Data1);
2 input clk, s_ROM1;
3 input [1:0] s_Address;
4 output [7:0] s_Data1;
5 reg [7:0] s_Data1;
6
7 always @(negedge clk)
8     if (s_ROM1)
9     begin
10         case(s_Address)
11             2'b00: s_Data1 <= 8'b01000000; //s_Data=40H
12             2'b01: s_Data1 <= 8'b01000001; //s_Data=41H
13             2'b10: s_Data1 <= 8'b01000010; //s_Data=42H
14             2'b11: s_Data1 <= 8'b01000011; //s_Data=43H
15         endcase
16     end
17 else
18     s_Data1 <= 8'bz;
19 endmodule
20
21

```

Figure 7: Verilog Code for rom1 Module

Figure 7 shows the Verilog code for module of ROM1.

```

1 module rom2(clk, s_ROM2, s_Address, s_Data2);
2 input clk, s_ROM2;
3 input [1:0] s_Address;
4 output [7:0] s_Data2;
5 reg [7:0] s_Data2;
6
7 always @(negedge clk)
8     if (s_ROM2)
9     begin
10         case(s_Address)
11             2'b00: s_Data2 <= 8'b01100101; //s_Data=65H
12             2'b01: s_Data2 <= 8'b01100110; //s_Data=66H
13             2'b10: s_Data2 <= 8'b01100111; //s_Data=67H
14             2'b11: s_Data2 <= 8'b01101000; //s_Data=68H
15         endcase
16     end
17 else
18     s_Data2 <= 8'bz;
19 endmodule
20

```

Figure 8: Verilog Code for rom2 Module

Figure 8 shows the Verilog code for module of ROM2.

```

1 module ram (clk, s_RAM, s_read, s_Address, s_Data, Data_Output);
2 input clk, s_RAM, s_read;
3 input [1:0] s_Address;
4 input [7:0] s_Data;
5 output [7:0] Data_Output;
6 reg [7:0] memory [3:0];
7 reg [7:0] Data_Output;
8
9 always @(posedge clk)
10     begin
11         if(s_RAM)
12             memory[s_Address] <= s_Data;
13         if(s_read)
14             Data_Output <= memory[s_Address];
15         else
16             Data_Output <= 8'bz;
17     end
18 endmodule
19

```

Figure 9: Verilog Code for ram Module

Figure 9 shows the Verilog code for module of RAM.

```

1  module control(clk, rst, s_Address, s_ROM1, s_ROM2, s_RAM, s_read);
2  input clk, rst;
3  output s_ROM1, s_ROM2, s_RAM, s_read;
4  output [1:0] s_Address;
5  parameter [2:0] s1 = 3'b000, s2 = 3'b001, s3 = 3'b010, s4 = 3'b011, s5 = 3'b100, s6 = 3'b101, s7 = 3'b110, s8 = 3'b111;
6  reg [2:0] present, next;
7  reg [1:0] s_Address;
8  reg [1:0] s_ROM1, s_ROM2, s_RAM, s_read;
9
10 always @(present)
11 begin
12 case(present)
13 s1: begin
14     next <= s2;
15     s_Address <= 2'b00;
16     s_ROM1 <= 1'b1;
17     s_ROM2 <= 1'b1;
18     s_RAM <= 1'b1;
19     s_read <= 1'b0;
20 end
21
22 s2: begin
23     next <= s3;
24     s_Address <= 2'b01;
25     s_ROM1 <= 1'b1;
26     s_ROM2 <= 1'b1;
27     s_RAM <= 1'b1;
28
29 s3: begin
30     next <= s4;
31     s_Address <= 2'b10;
32     s_ROM <= 1'b1;
33     s_RAM <= 1'b1;
34     s_read <= 1'b0;
35 end
36
37 s4: begin
38     next <= s5;
39     s_Address <= 2'b11;
40     s_ROM <= 1'b1;
41     s_RAM <= 1'b1;
42     s_read <= 1'b0;
43 end
44
45 s5: begin
46     next <= s6;
47     s_Address <= 2'b00;
48     s_ROM <= 1'b0;
49     s_RAM <= 1'b0;
50     s_read <= 1'b1;
51 end
52
53 s6: begin
54     next <= s7;
55
56     s_Address <= 2'b01;
57     s_ROM <= 1'b0;
58     s_RAM <= 1'b0;
59     s_read <= 1'b1;
60 end
61
62 s7: begin
63     next <= s8;
64     s_Address <= 2'b10;
65     s_ROM <= 1'b0;
66     s_RAM <= 1'b0;
67     s_read <= 1'b1;
68 end
69
70 s8: begin
71     next <= s1;
72     s_Address <= 2'b11;
73     s_ROM <= 1'b0;
74     s_RAM <= 1'b0;
75     s_read <= 1'b1;
76 end
77 endcase
78 end
79
80 always @(posedge clk or negedge rst)
81 begin
82     if(rst == 0)

```


```

82         present <= s1;
83     else
84         present <= next;
85     end
86 endmodule
87

```

Figure 10: Verilog Code for control Module

Figure 10 shows the Verilog code for module of control unit which is the finite state machine.



```
1 module adder(s_Data1, s_Data2, sum):
2
3     input [7:0] s_Data1;
4     input [7:0] s_Data2;
5     output [7:0] sum;
6
7     assign sum = s_Data1 + s_Data2;
8
9 endmodule
10
```

Figure 11: Verilog Code for adder Module

Figure 11 shows the Verilog code for module of adder.

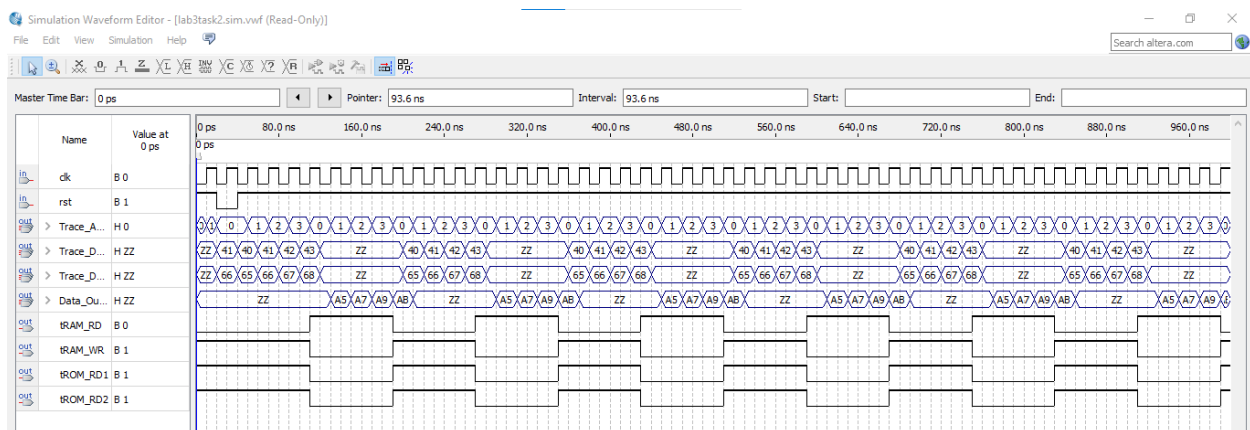


Figure 12: Simulation Result for lab3task2 Module

Figure 12 shows the simulation result for the lab3task2 module. All the output data for Trace_Address, Trace_Data1, Trace_Data2, and Data_Output will have specific result in hexadecimal radix and the result will reflect it during each of the negative edge of the clock. During the third negative edge cycle of the clock, the system will read the value of Trace_Data1 from ROM1 and Trace_Data2 from ROM2 then under the operation of adder and the result will store in RAM, then it will read the RAM value after the seventh positive edge cycle of the clock such as A5, A7, A9, and AB. Otherwise, the output data for Trace_Data1, Trace_Data2, and Data_Output will show "ZZ".

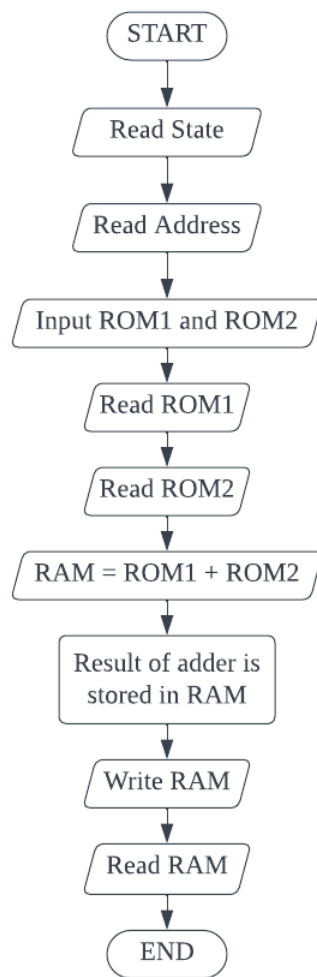


Figure 13: Flowchart of the system

Figure 13 shows the figure of the system for lab3task2 module.