HW1: 4_1 Mux, Program counter and Memory

Summer 2021

Thiha Myint

Demo:

https://youtu.be/gCpGRKknvNA

Top Level Block Diagram

Part 1

Figure 1 show the block diagram for the HW1 assignment. It is made to connect between mux, program counter and memory. The circuit take input from JALR, Branch, Jump which are connected to output from Mux. Then, DOUT go to program counter which goes toward memory that store data and instruction to be executed.

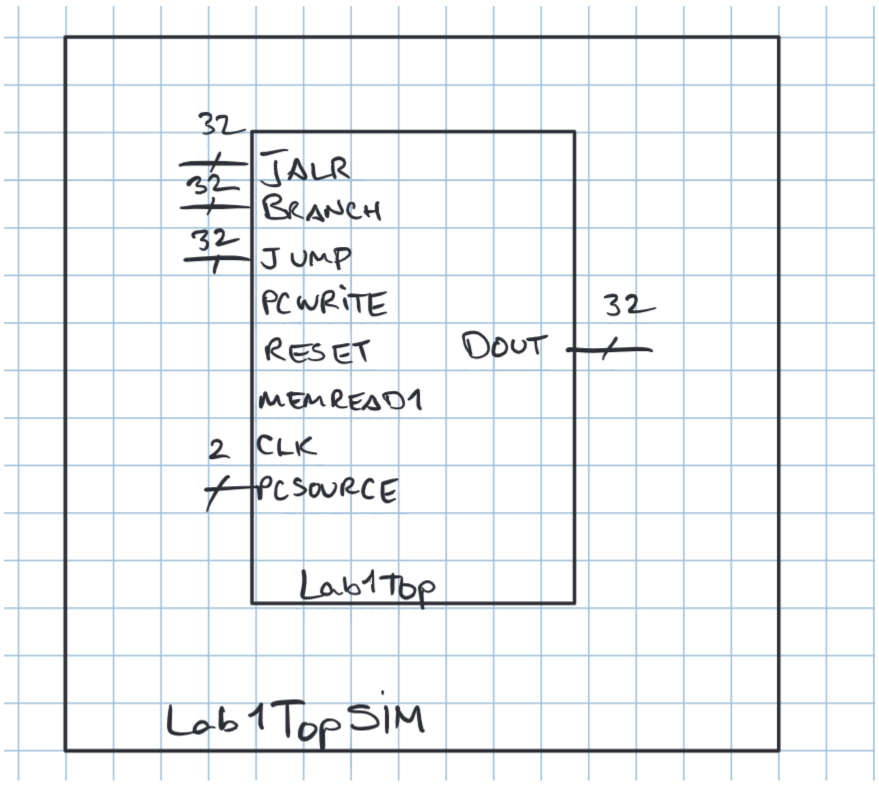


Fig 1 : Top level block diagram of HW1

Part 2

Structural Model

The circuit consists of three parts, mux, program counter and memory register file. Mux takes input from top level module and output the results. Program counter then take the results from mux to show the current instruction process to execute. The data set is store in memory.

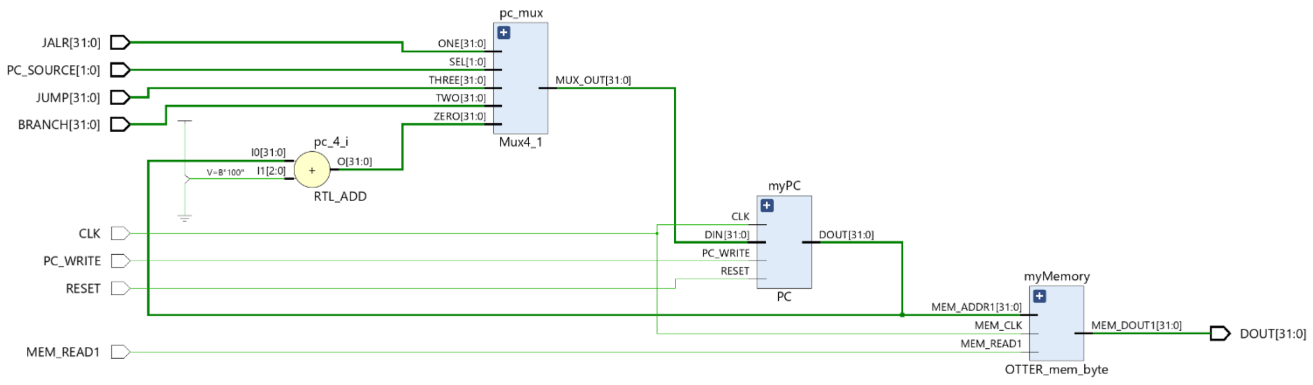
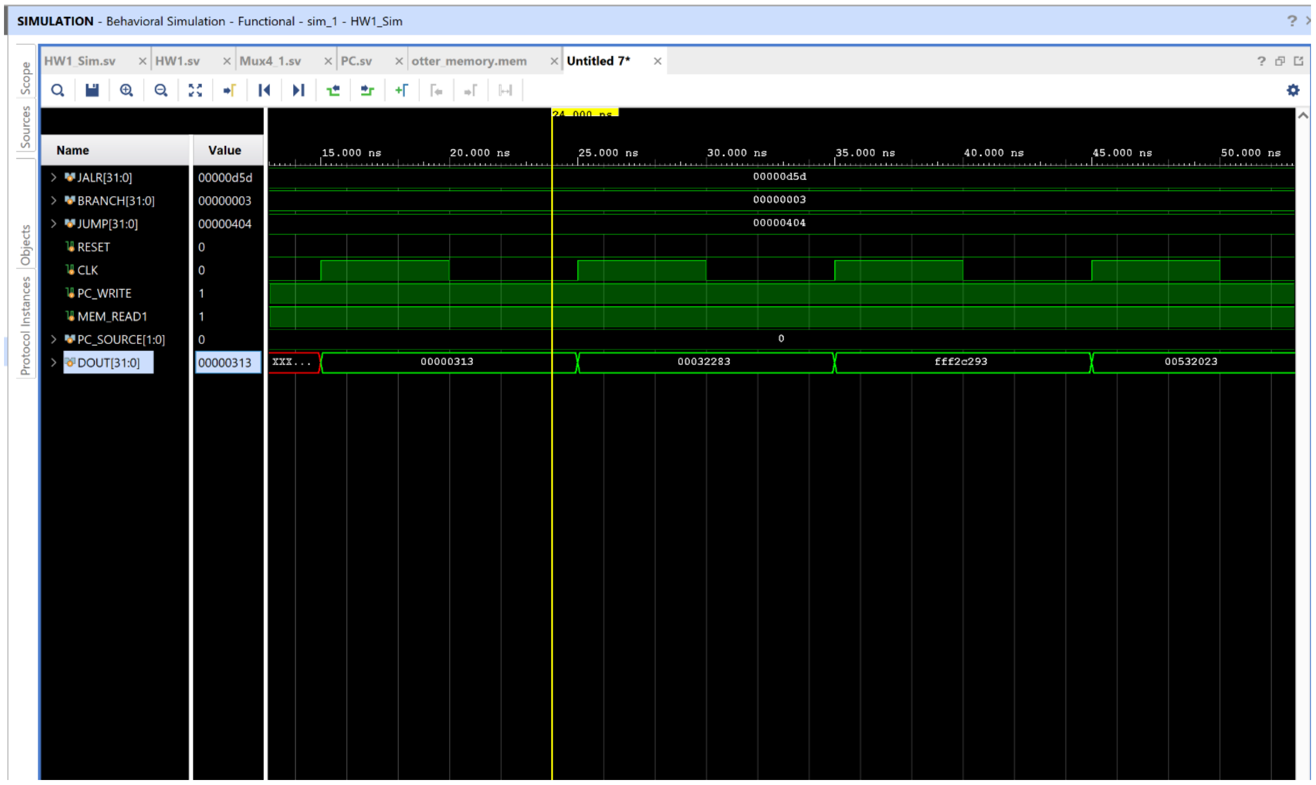


Fig 2 : Structural model of HW1

Simulation



Name	Value
JALR[31:0]	00000d5d
BRANCH[31:0]	00000003
JUMP[31:0]	00000404
RESET	0
CLK	0
PC_WRITE	1
MEM_READ1	1
PC_SOURCE[1:0]	0
DOUT[31:0]	00000313

```
JALR[31:0]       00000d5d
BRANCH[31:0]     00000003
JUMP[31:0]       00000404
PC_SOURCE[1:0]   0
DOUT[31:0]       XXX...  00000313  00032283  fff2c293  00532023
```

Fig 3: Sim waveform which show the memory address

Conclusion

This lab shows the basic of how to construct the computer. The knowledge of how computer operate was observed. Now, I understand it that computer needs register file to store 32 registers for values. I see that how circuit are connected and built on each other. Everything has its purpose to become a working computer. I was able to translate machine code to assembler and vice versa.

Code

```
`timescale 1ns / 1ps

module HW1(
    input [31:0] JALR,
    input [31:0] JUMP,
    input [31:0] BRANCH,
    input CLK,
    input RESET,
    input PC_WRITE,
    input MEM_READ1,
    input [1:0] PC_SOURCE,
    output logic [31:0] DOUT
    );

    logic [31:0] pc_4, pc_in, pc_out;

    assign pc_4 = pc_out + 4;

  Mux4_1 pc_mux (.ZERO(pc_4), .ONE(JALR), .TWO(BRANCH), .THREE(JUMP),
.SEL(PC_SOURCE), .MUX_OUT(pc_in));

  PC myPC (.DIN(pc_in), .DOUT(pc_out), .CLK(CLK), .PC_WRITE (PC_WRITE),
.RESET(RESET));

  OTTER_mem_byte myMemory (.MEM_ADDR1(pc_out), .MEM_READ1(MEM_READ1),
.MEM_CLK(CLK), .MEM_DOUT1(DOUT));

  endmodule
```

```
`timescale 1ns / 1ps

module Mux4_1(
  input [31:0] ZERO,
  input [31:0] ONE,
  input [31:0] TWO,
  input [31:0] THREE,
  input [1:0] SEL,
  output logic [31:0] MUX_OUT
  );


  always_comb
    begin
      case(SEL)
        0 : MUX_OUT <= ZERO;
        2'b01   : MUX_OUT <= ONE;
        2 : MUX_OUT <= TWO;
        3 : MUX_OUT <= THREE;
        default : MUX_OUT <= ZERO;
      endcase
    end

endmodule
```

```
`timescale 1ns / 1ps

module PC(
  input [31:0] DIN,
  output logic [31:0] DOUT = 0, //initialize register to zero
  input CLK,
  input RESET,
  input PC_WRITE
);

  always_ff @ (posedge CLK)
    begin
      if (RESET) //sychronous reset
        DOUT = 0;
      else if (PC_WRITE)
        DOUT = DIN;
    end

endmodule
```

```
`timescale 1ns / 1ps

module HW1_Sim();

    logic [31:0] JALR;
    logic [31:0] BRANCH;
    logic [31:0] JUMP;
    logic [31:0] DOUT;
    logic RESET;
    logic CLK;
    logic PC_WRITE;
    logic MEM_READ1;
    logic [1:0] PC_SOURCE;

    HW1 HW1_Test (.*);

    always
    begin
    CLK = 0; #5; CLK = 1; #5;
    end

    initial
    begin
        RESET = 1; PC_SOURCE = 0;
        JALR = 3421;
        BRANCH = 3;
        JUMP = 1028;
        PC_WRITE = 0;
        MEM_READ1 = 0;
        #10
        RESET = 0; PC_WRITE = 1; MEM_READ1 = 1;
        #50;
        PC_SOURCE = 1;
        #10;
        PC_SOURCE = 2;
        #10;
        PC_SOURCE = 3;
        #10;
        RESET = 1;
        #10;
        RESET = 0; PC_WRITE = 0;
        #10;
        PC_WRITE = 1; MEM_READ1 = 0;

     end
endmodule
```