Name: Joseph Hany **ID:900182870**

Architecture Lab5 Report

2) A technical summary of experiments conducted in the lab (steps, results, components, code functionality, etc.)

- **Experiment 1:**
  This experiment aims to implement a word addressable instruction memory
- In order to implement such a circuit, we should do the following:
  1- Create one new module: InstMem
  2- InstMem module simply receives a six-bit address for the desired word in the instruction memory and outputs the instruction corresponding to the provided address.

  For example:

```
module InstMem(
input [5:0] addr, output [31:0] data_out
    );

    reg [31:0] mem [0:63];
    initial begin  // initializing the instruction memory with some data in order to test it
        mem[0]=32'd17;
        mem[1]=32'd9;
        mem[2]=32'd25;
        mem[20]=32'd5000;
        mem[30]=32'd6000;
        mem[40]=32'd7000;
    end
    assign data_out = mem[addr];  // here we assign the instruction found in the
instruction memory to the data out bus that is composed of 32 wires
endmodule
```

- **Experiment 2:**
  This experiment aims to implement a word addressable Data Memory
- In order to implement such a circuit, we should do the following:
  1- Create one new module: DataMem
  2- DataMem module simply checks whether MemRead or MemWrite are enabled. If the MemRead is enabled it outputs the data corresponding to the provided address. If the MemWrite is enabled it writes the provided data to the input address in the data memory.

  For example:

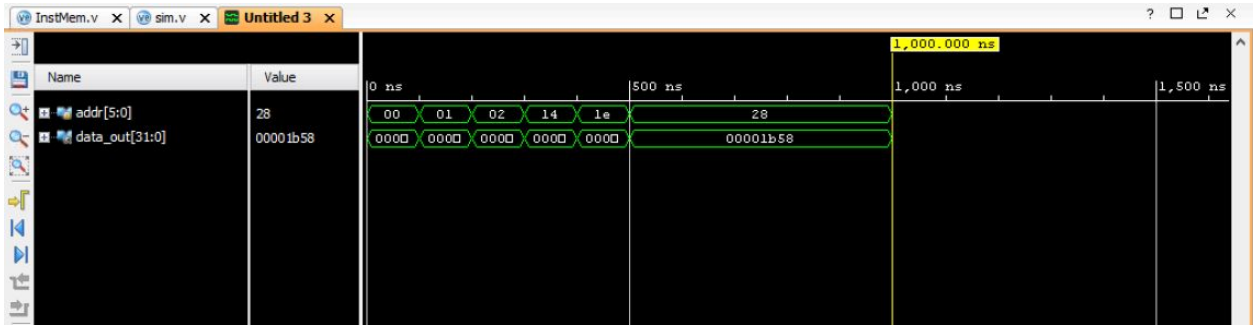
```
module DataMem
(input clk, input MemRead, input MemWrite,
input [5:0] addr, input [31:0] data_in, output reg [31:0] data_out);
reg [31:0] mem [0:63];

always @(addr, MemRead) begin
if(MemRead)   // here we check whether the MemRead is set to 1 or not
data_out=mem[addr]; // if set to 1, we set data_out to the data corresponding to the
provided address
end

always @(posedge clk) begin
if(MemWrite)begin      // here we check whether the MemWrite is set to 1 or not
mem[addr]=data_in;    // if set to 1, we set the provided address in the memory to the
input data
end
end
endmodule
```

3) Verilog code for all modules of each experiment (including testbenches).

- **Verilog code of each experiment is attached in the zip file**

4) Snapshot of simulation output:

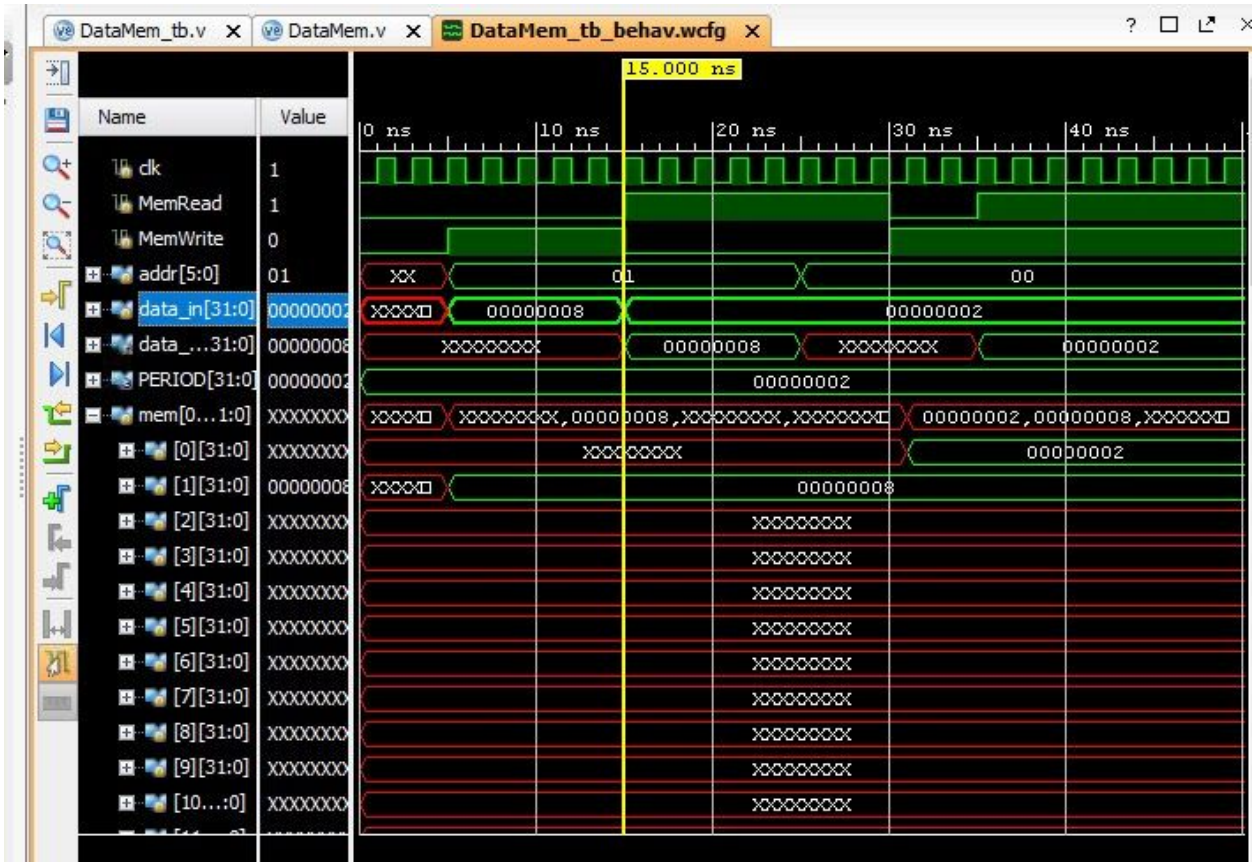
- **EXP1 Simulation Snapshot:**



Here, we can see that when:

- The provided address is 0 (0 in Hexa), the data_out equal to 32'd17
- The provided address is 1 (1 in Hexa), the data_out equal to 32'd9
- The provided address is 2 (2 in Hexa), the data_out equal to 32'd25
- The provided address is 20 (14 in Hexa), the data_out equal to 32'd5000
- The provided address is 30 (1e in Hexa), the data_out equal to 32'd6000
- The provided address is 40 (2e in Hexa), the data_out equal to 32'd7000

- **EXP2 Simulation Snapshot:**



Here, we can see that:

- We begin by initializing MemRead and MemWrite to 0
- When MemRead=0, MemWrite=1, data_in=8, addr=1 → we write to address 1 data 8
- MemWrite=0, MemRead=1, data_in=2, addr=1 → we read data found in addr 1
- MemWrite=0, MemRead=1, data_in=2, addr=0 → we read data found in addr 0
- MemWrite=1, MemRead=0, data_in=2, addr=0 → we write 2 to addr 0
- MemWrite=1, MemRead=1, data_in=2, addr=0 → we neither write nor read from the data memory