Lab 4: Memory stage

In this lab , I just show source code, testbench and simulation

A.Implementation

1. data memory

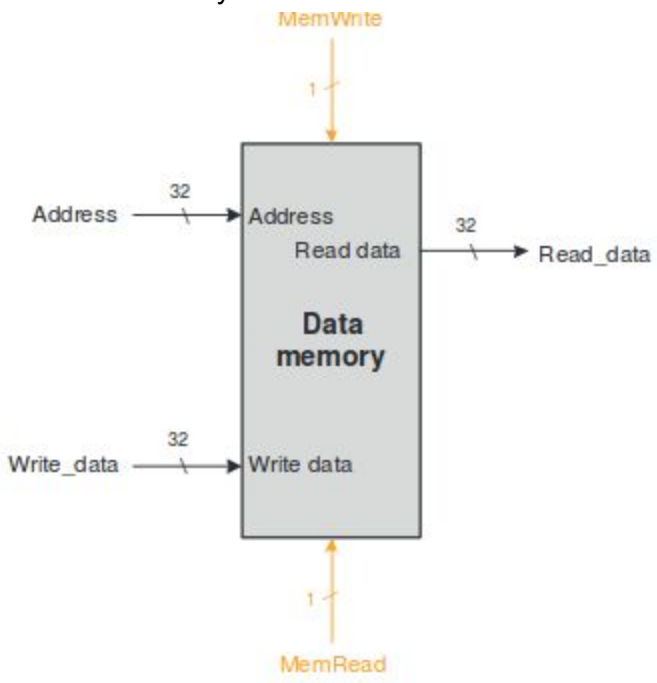


Figure 4.3: The data memory unit

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////

// Engineer: hau tao
//
// Create Date:01:50:27 12/06/2015
// Design Name:
// Module Name:        data_mem

//////////////////////////////////////////////////////////////////////////////////

module data_mem(input [31:0] address, inout[31:0] writedata, input memread, input memwrite,
output reg[31:0] readdata
        );
    reg[31:0] register[0:255];
    reg [31:0] data_in;
```

```
    integer i;
    initial
    begin
          data_in <= 'h0000000A;
          register[0] <= 'h002300AA;
           register[1] <= 'h10654321;
           register[2] <= 'h00100022;
           register[3] <= 'h8C123456;
           register[4] <= 'h8F123456;

           register[5] <= 'hAD654321;
           register[6] <= 'h13012345;
           register[7] <= 'hAC654321;
           register[8] <= 'h12012345;
           for ( i = 9; i < 255; i = i +1)
                    begin
                              register[i] <= 'h11111111;
                    end
           end
           assign writedata = (memread)? data_in:'hx;
           // write data
           always @ (address or writedata)

            begin
          if ( memwrite) begin
          register[address] = writedata;
          end

     //read data
     if ( memread) begin
          readdata = register[address] ;
          end
  end

endmodule
```

2. And gate

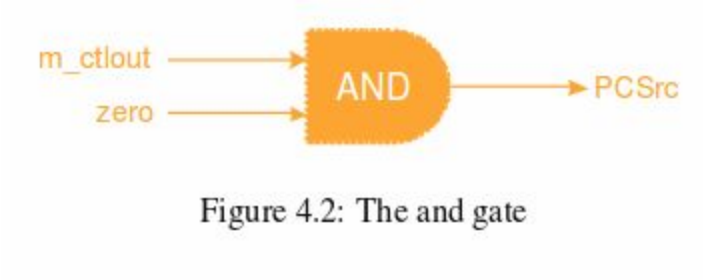


Figure 4.2: The and gate

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: hau tao
//
// Create Date:13:48:16 12/06/2015
// Design Name:
// Module Name:        and_gate
//////////////////////////////////////////////////////////////////////////////////
module and_gate( input zero, input branch, output reg PCSrc
            );
    always @*
    begin
    PCSrc = branch & zero;
    end


endmodule
```

3. Memory latch

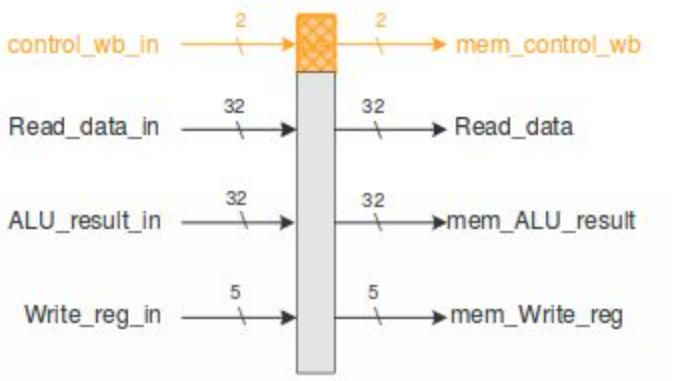
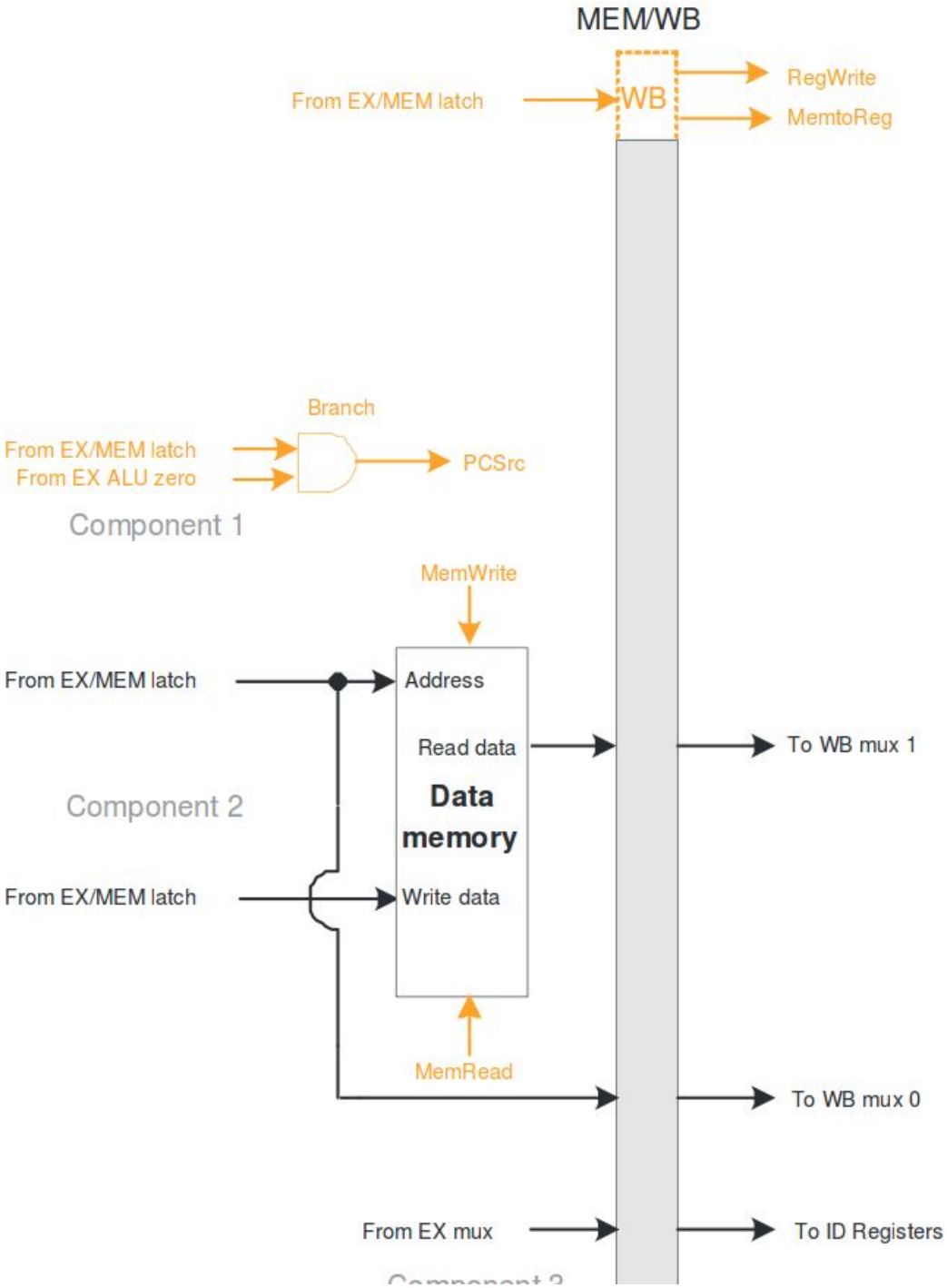


Figure 4.4: The MEM/WB pipeline register (latch)

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: hau tao
//
// Create Date:14:01:42 12/06/2015
// Design Name:
// Module Name:        mem_latch
//////////////////////////////////////////////////////////////////////////////////
module mem_latch(input [1:0] control_wb_in, input[31:0] Read_data_in, input[31:0]
ALU_result_in, input[4:0] Write_reg_in,
output reg[1:0]mem_control_wb, output reg[31:0]Read_data, output reg[31:0]mem_ALU_result,
output reg[4:0] mem_Write_reg
        );

    always @*
        begin
        #5  // clock cycle of the latch
        mem_control_wb <= control_wb_in;
                Read_data  <= Read_data_in;
                mem_ALU_result <= ALU_result_in;
                mem_Write_reg   <= Write_reg_in;


        end
```

endmodule

4. memory stage connection



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Engineer: hau tao
//
// Create Date:14:18:46 12/06/2015
// Design Name:
// Module Name:         memory_stage_connection

//////////////////////////////////////////////////////////////////////////////////
module memory_stage_connection(input [1:0] control_wb_in, input [2:0] ex_mem_latch, input
zero, input[31:0] ALU_result_in,
inout[31:0] To_mem_write_data, input[31:0] Write_reg_in, output
wire[1:0]mem_control_wb,output wire[31:0] Read_data, output wire[31:0]mem_ALU_result,
output wire[4:0] mem_Write_reg
          );
    wire PCSrc;

    wire [31:0] Read_data_in;
    and_gate and_gate_1 (
          .zero(zero),
          .branch(ex_mem_latch[2]),
          .PCSrc(PCSrc)
   );
   data_mem data_mem_1 (
          .address(ALU_result_in),
          .writedata(To_mem_write_data),
          .memread(ex_mem_latch[1]),
          .memwrite(ex_mem_latch[0]),
          .readdata(Read_data_in)
   );
   mem_latch mem_latch_1 (
          .control_wb_in(control_wb_in),
          .Read_data_in(Read_data_in),
          .ALU_result_in(ALU_result_in),
          .Write_reg_in(Write_reg_in),
          .mem_control_wb(mem_control_wb),
          .Read_data(Read_data),
          .mem_ALU_result(mem_ALU_result),
          .mem_Write_reg(mem_Write_reg)
   );
```

```
endmodule
```

B. Test bench

1. data memory

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: hau tao
//
// Create Date:   13:05:11 12/06/2015
// Design Name:   data_mem
// Module Name:   /home/hau/Desktop/lab4/test_mem_data.v
// Project Name:  lab4
////////////////////////////////////////////////////////////////////////////////

module test_mem_data;

    // Inputs
    reg [31:0] address;
    reg memread;
    reg memwrite;


    // Outputs
    wire [31:0] readdata;

    // Bidirs
    wire [31:0] writedata;

    // Instantiate the Unit Under Test (UUT)
    data_mem uut (
            .address(address),
            .writedata(writedata),
            .memread(memread),
            .memwrite(memwrite),
            .readdata(readdata)
    );

    initial begin
            // Initialize Inputs
            address = 0;
            memread = 0;
            memwrite = 0;
```

```
        // Wait 100 ns for global reset to finish

        #1;

        memread =1;
        #1;
        memwrite =1;
        #1;
        address =1;
        #1;
        address =2;
        #1
        address =3;
        memwrite =0;
        #1
        address =4;

        #1
        address =5;
        #1
        address =6;
        end


        // Add stimulus here
        always@(address or memread or memwrite)begin
        #1

$display("time=%0d\taddress=%0d\twritedata=%h\treaddata=%h",$time,address,writedata,read
data);
end


endmodule
```

2. and gate
timescale 1ns / 1ps

```
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: hau tao
//
// Create Date:   13:53:06 12/06/2015
// Design Name:   and_gate
// Module Name:   /home/hau/Desktop/lab4/test_gate.v
// Project Name:  lab4
//////////////////////////////////////////////////////////////////////////////////

module test_gate;

    // Inputs
    reg zero;
    reg branch;

    // Outputs
    wire PCSrc;

    // Instantiate the Unit Under Test (UUT)
    and_gate uut (
            .zero(zero),
            .branch(branch),
            .PCSrc(PCSrc)
    );

    initial begin
            // Initialize Inputs
            zero = 0;
            branch = 0;

            // Wait 100 ns for global reset to finish

            #1;
            branch =1;
            #1
            zero =1;
            #1
            branch = 0;
```

```
        end

        always @* begin

         #2$display("time=%0d\tbranch=%0b\tzero=%0b\tPCSrc=%0b\t",$time, branch, zero,
PCSrc);

        end



endmodule
```

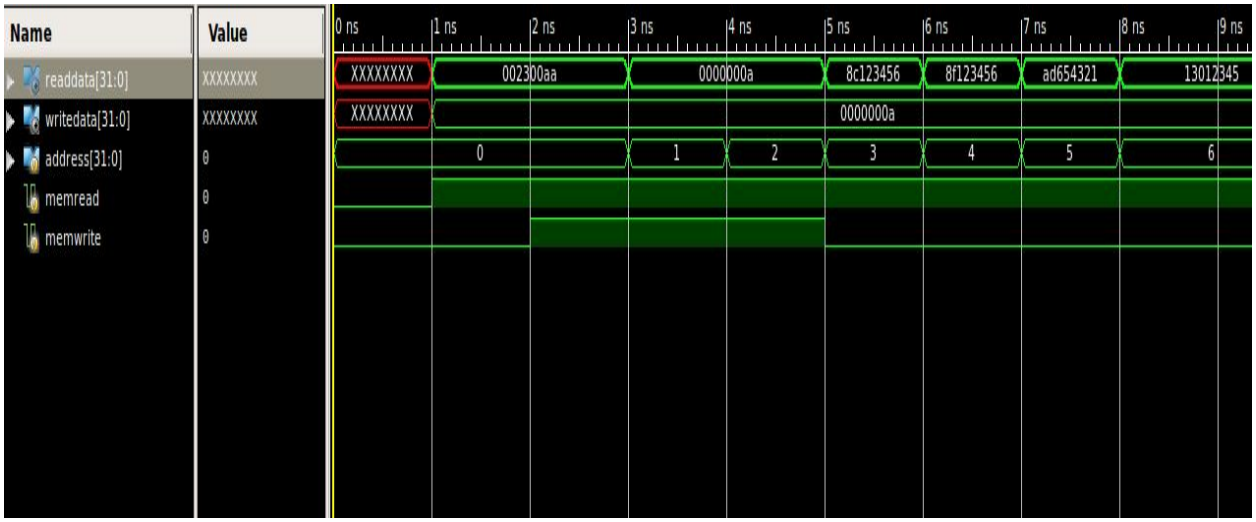
3. memory latch

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: hau tao
//
// Create Date:   14:16:05 12/06/2015
// Design Name:   mem_latch
// Module Name:   /home/hau/Desktop/lab4/test_mem_latch.v
// Project Name:  lab4
//////////////////////////////////////////////////////////////////////////////////

module test_mem_latch;

    // Inputs
    reg [1:0] control_wb_in;
    reg [31:0] Read_data_in;
    reg [31:0] ALU_result_in;
    reg [4:0] Write_reg_in;

    // Outputs
    wire [1:0] mem_control_wb;
    wire [31:0] Read_data;
    wire [31:0] mem_ALU_result;
    wire [4:0] mem_Write_reg;

    // Instantiate the Unit Under Test (UUT)
    mem_latch uut (
        .control_wb_in(control_wb_in),
```

```
        .Read_data_in(Read_data_in),
        .ALU_result_in(ALU_result_in),
        .Write_reg_in(Write_reg_in),
        .mem_control_wb(mem_control_wb),
        .Read_data(Read_data),
        .mem_ALU_result(mem_ALU_result),
        .mem_Write_reg(mem_Write_reg)
    );

    initial begin
        // Initialize Inputs
        control_wb_in = 0;
        Read_data_in = 0;
        ALU_result_in = 0;
        Write_reg_in = 0;

        // Wait 100 ns for global reset to finish
        #1;
        control_wb_in =1;
        Read_data_in = 2;
        ALU_result_in = 3;
        Write_reg_in = 4;


        // Add stimulus here

    end

endmodule
```

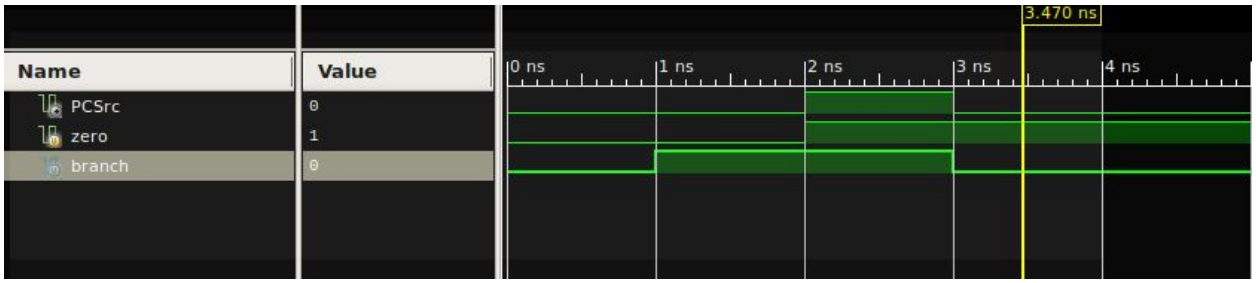
C. Simulation

1. data memory



time=1 address=0 writedata=xxxxxxxx readdata=xxxxxxxx
time=3 address=1 writedata=0000000a readdata=002300aa
time=5 address=3 writedata=0000000a readdata=0000000a
time=7 address=5 writedata=0000000a readdata=8f123456
time=9 address=6 writedata=0000000a readdata=13012345

2. and gate



time=2 branch=1 zero=0 PCSrc=0
time=4 branch=0 zero=1 PCSrc=0

3. memory latch