Lab 3: Execution stage

In this lab, I just show the timing diagram, source code, and test bench of each component in the stage

1. Implementation

A. 32-bit Mux

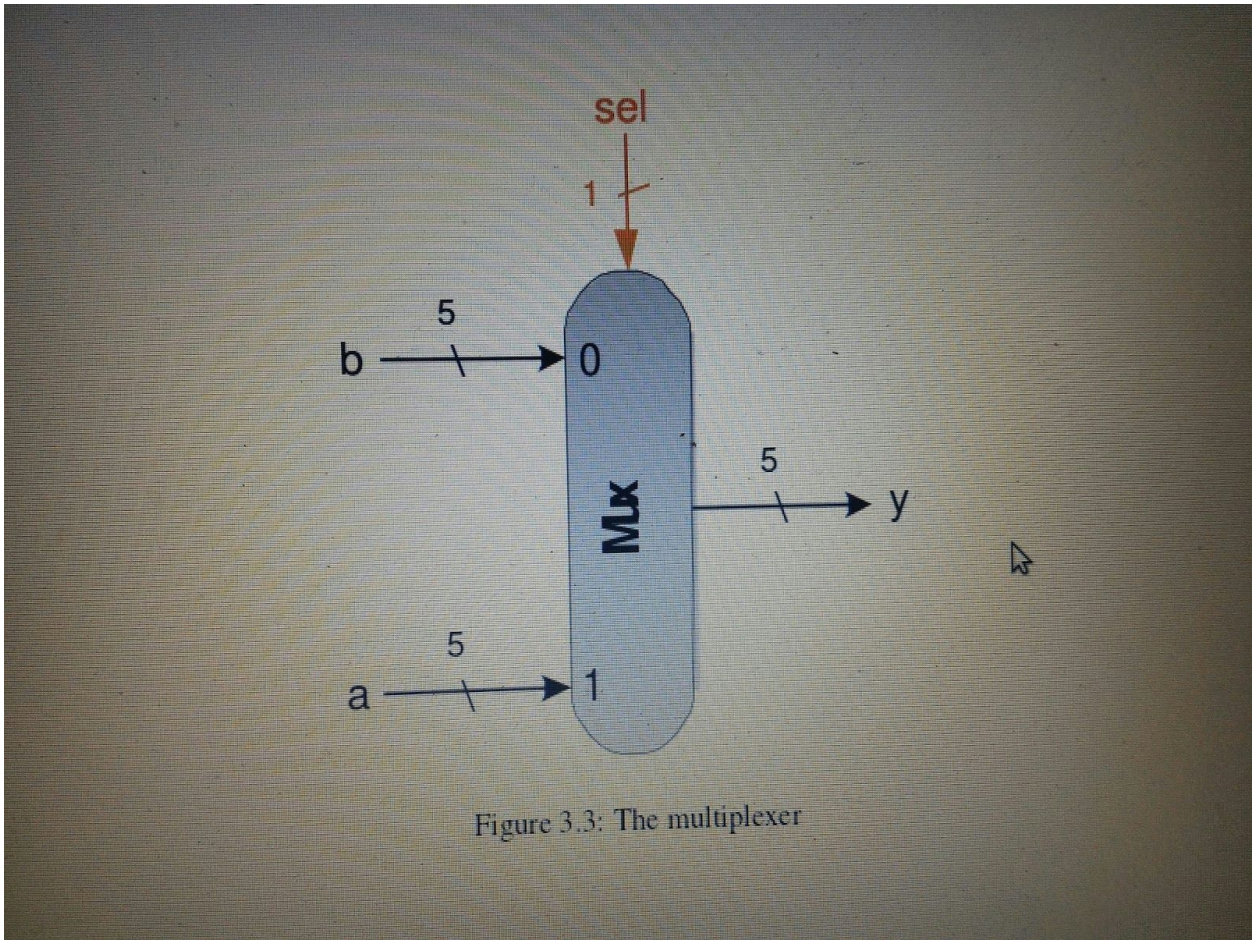


Figure 3.3: The multiplexer

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Hau Tao
// Create Date: 11:12:01 11/25/2015
// Module Name:    mux_32bit
//////////////////////////////////////////////////////////////////////////////////
module mux_32bit(a,b,sel,y
    );
input[31:0] a, b;
```

```
input sel;
output[31:0] y;
assign y = sel ? a: b;

endmodule
```

B. 5-bit mux

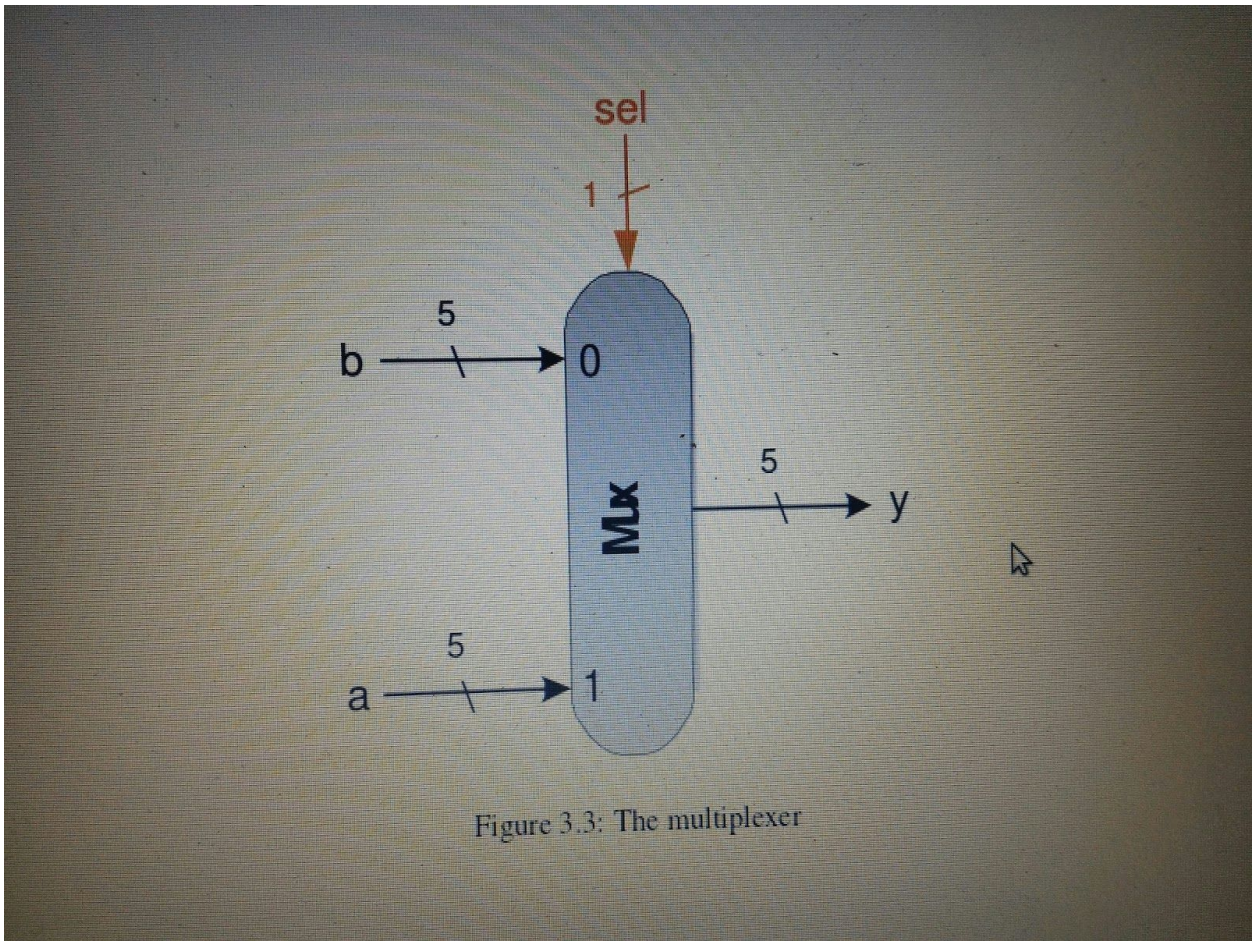


Figure 3.3: The multiplexer

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Engineer: Hau Tao
// Create Date: 09:44:04 11/16/2015
// Module Name:        mux_EX
//////////////////////////////////////////////////////////////////////////////////

module mux_EX( a, b, sel, y);
        input[4:0] a, b;
        input sel;
        output[4:0] y;
        assign y = sel ? a: b;
endmodule
```

C. ALU control unit

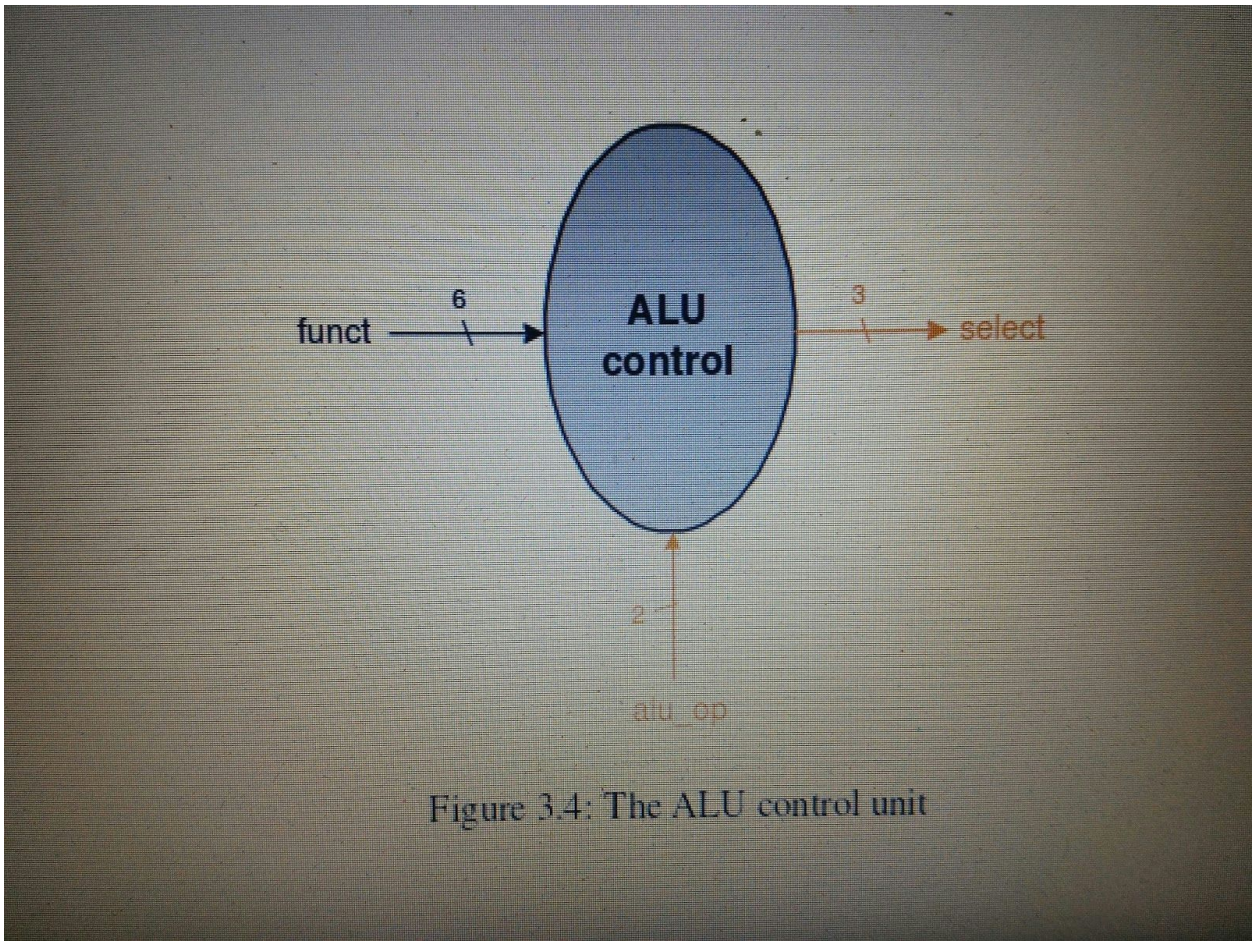


Figure 3.4: The ALU control unit

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Hau Tao
//
// Create Date: 10:08:06 11/16/2015
//////////////////////////////////////////////////////////////////////////////////
module ALU_control_unit(input[5:0] function_bit, input[1:0] ALUop, output reg[2:0] select
        );
    //ALU op
    parameter LW = 2'b00;
    parameter SW = 2'b00;
    parameter branch_equal = 2'b01;
```

```
parameter R_type = 2'b10;
parameter not_valid =2'b11;

//functionality
parameter loadword_func = 6'bxxxxxx;
parameter storeword_func = 6'bxxxxxx;
parameter branch_equal_func =6'bxxxxxx;
parameter add_func = 6'b100000;
parameter subtract_func=6'b100010;
parameter AND_func =6'b100100;
parameter OR_func =6'b100101;
parameter set_on_less_than_func =6'b101010;

//alu_control_input
parameter ADD_control = 3'b010 ;
parameter SUBTRACT_control =3'b110;
parameter OR_control = 3'b001;
parameter AND_control =3'b000;
parameter SET_ON_LESS_THAN_control = 3'b111;
initial begin
                        if( loadword_func == 6'bxxxxxx || storeword_func == 6'bxxxxxx)
                        begin
                        select = 3'b010;
                        end

                        else if (branch_equal_func == 6'bxxxxxx)
                        begin
                        select = 3'b110;
                        end

    end
always@*
case (ALUop)
LW: select = ADD_control;
SW: select = ADD_control;

branch_equal: select = SUBTRACT_control;

R_type:begin
```

```
                case(function_bit)
                add_func: begin
                select = ADD_control;
                end

                subtract_func: begin
                select = SUBTRACT_control;
                end

                AND_func:     begin
                select = AND_control ;
                end
                OR_func: begin
                select = OR_control ;
                end
                set_on_less_than_func: begin
                elect = SET_ON_LESS_THAN_control ;
                end
                endcase
                end
   default: $display("Not available instruction");
   endcase
endmodule
```

D. ALU

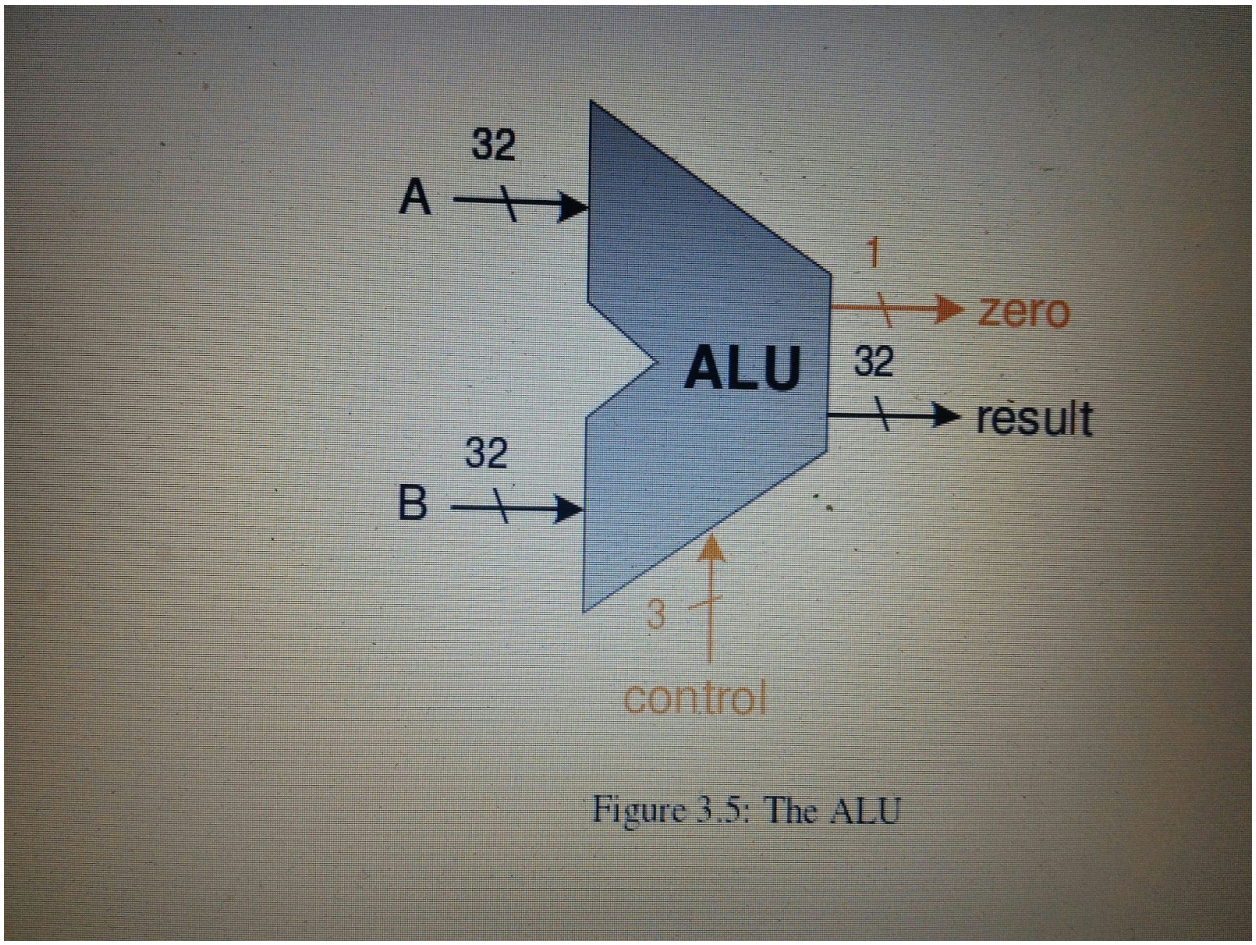


Figure 3.5: The ALU

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Hau tao
//
// Create Date: 20:53:46 11/24/2015
// Module Name:         alu
//////////////////////////////////////////////////////////////////////////////////
module alu(
    input wire [31:0] A,
    input wire [31:0] B,
    input wire [2:0]  control,
```

```
    output reg              zero,
    output reg [31:0] result
          );

     parameter add = 3'b010,
          subtract  = 3'b110,
          and_alu  = 3'b000,
          or_alu   = 3'b001,
          set_on_less_than  = 3'b111;

          always @* begin
          case (control)
          add:  result <= A + B;
          subtract:  result <= A - B;
          and_alu:  result <= A&B;
          or_alu:   result <= A|B;
          set_on_less_than:  result <= (A<B)? 1:0;
          default: result <= 32'bx;
          endcase
                    begin
                                        if (result==0)
                                        zero <=0;
                                        end

          end
endmodule
```

E. Adder

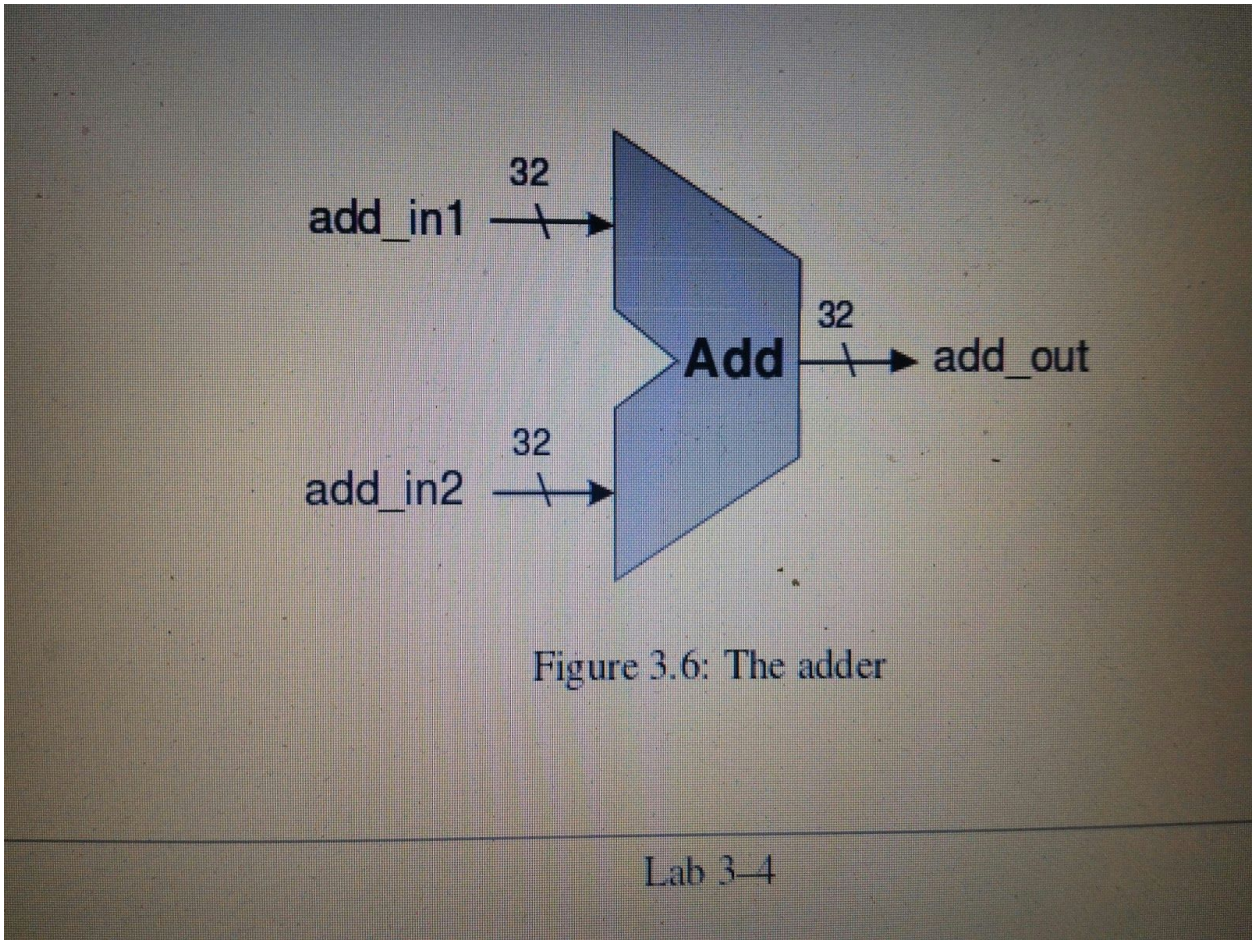


Figure 3.6: The adder


```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Engineer: Hau Tao
//
// Create Date: 09:35:04 11/16/2015
// Module Name:         adder
//////////////////////////////////////////////////////////////////////////////////
module adder( input [31:0] add_in1, input[31:0] add_in2, output reg[31:0] add_out
            );
    always @ *
    begin
```

```
    add_out = add_in1 + add_in2;
    end


endmodule

`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: hau Tao
//
// Create Date: 11:18:57 11/25/2015
//////////////////////////////////////////////////////////////////////////////////
module The_EX_stage(
   input  [1:0]  ctlwb_out,
   input [2:0]  ctlm_out,
   input [31:0]  adder_out,
   input          aluzero,
   input  [31:0]  alu_out,
   input  [31:0]  readdat2,
   input  [4:0]  mux_out,

   output reg[1:0]  wb_ctlout,
   output reg [2:0]  m_ctlout,
   output reg [31:0] add_result,
   output reg              zero,
   output reg [31:0] alu_result,
   output reg [31:0] rdata2out,
   output reg [4:0]  five_bit_muxout
         );



    always @*
         begin
         #5  // clock cycle of the latch
         wb_ctlout <= ctlwb_out;
                  m_ctlout  <= ctlm_out;
                  add_result <= adder_out;
```

```
                zero   <= aluzero;
                alu_result <= alu_out;
                rdata2out <= readdat2;
                five_bit_muxout<= mux_out;

        end

endmodule
```

F. EX latch

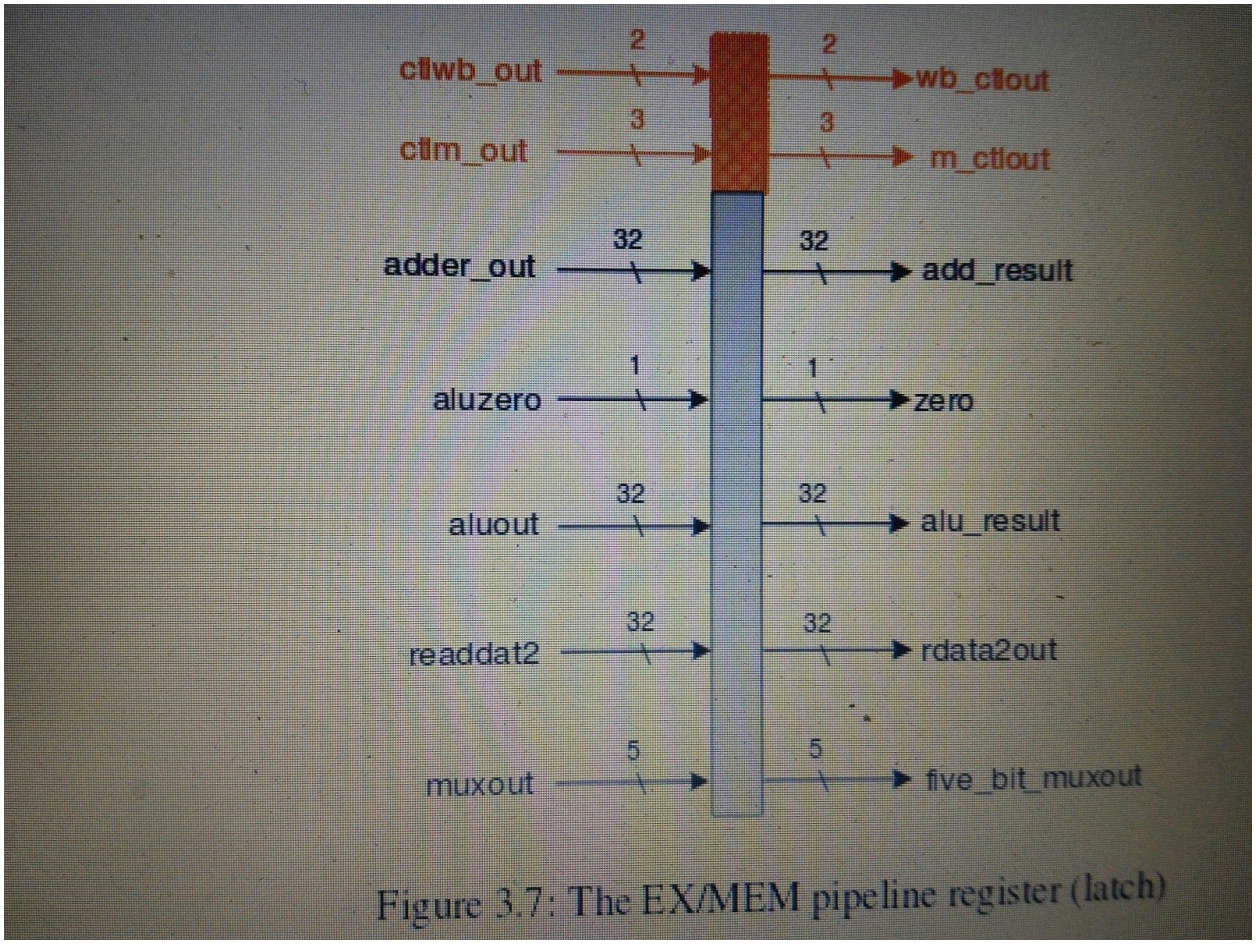


Figure 3.7: The EX/MEM pipeline register (latch)

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: hau Tao
//
// Create Date: 11:18:57 11/25/2015
//////////////////////////////////////////////////////////////////////////////////
module The_EX_stage(
    input  [1:0]  ctlwb_out,
    input [2:0]  ctlm_out,
    input [31:0]  adder_out,
    input          aluzero,
    input  [31:0]  alu_out,
    input  [31:0]  readdat2,
    input  [4:0]  mux_out,
```

```
    output reg[1:0]  wb_ctlout,
    output reg [2:0]  m_ctlout,
    output reg [31:0] add_result,
    output reg              zero,
    output reg [31:0] alu_result,
    output reg [31:0] rdata2out,
    output reg [4:0]  five_bit_muxout
          );



     always @*
          begin
          #5  // clock cycle of the latch
          wb_ctlout <= ctlwb_out;
                    m_ctlout  <= ctlm_out;
                    add_result <= adder_out;
                    zero   <= aluzero;
                    alu_result <= alu_out;
                    rdata2out <= readdat2;
                    five_bit_muxout<= mux_out;

          end

endmodule
```

G. EX stage combination

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Hau Tao
//
// Create Date: 11:37:10 11/25/2015
// Design Name:
// Module Name:        EX_stage_combination
//////////////////////////////////////////////////////////////////////////////////
module EX_stage_combination(
    input  [1:0]  wb_ctlout,
    input  [2:0]  m_ctlout,
    input  [3:0]  ex_ctlout,
    input  [31:0]  npcout,
    input  [31:0]  rdata1out,
    input  [31:0]  rdata2out,
    input  [31:0]  s_extendout,
    input  [4:0]  instrout_2016,
    input  [4:0]  instrout_1511,

    output wire [1:0]  EX_MEM_wb_ctlout,
    output wire [2:0]  EX_MEM_m_ctlout,
    output wire [31:0] EX_MEM_add_result,
    output wire                    EX_MEM_zero,
    output wire [31:0] EX_MEM_alu_result,
    output wire [31:0] EX_MEM_rdata2out,
    output wire [4:0]  EX_MEM_five_bit_muxout
        );
    wire [4:0] mux_5bit_output;
    wire [31:0] mux_32bit_output;
    wire [2:0] alu_input;
    wire alu_zero_output;
    wire [31:0] alu_result_output;
    wire [31:0] adder_output;

     mux_32bit my_mux_32bit (
            .a(s_extendout),
            .b(rdata2out),
```

```
        .sel(ex_ctlout[0]),
        .y(mux_32bit_output)
);

mux_EX my_mux_EX_5bit (
        .a(instrout_1511),
        .b(instrout_2016),
        .sel(ex_ctlout[3]),
        .y(mux_5bit_output)
);
ALU_control_unit my_alu_control_unit (
        .function_bit(s_extendout[5:0]),
        .ALUop(ex_ctlout[2:1]),
        .select(alu_input)
);
alu my_alu (
        .A(rdata1out),
        .B(mux_32bit_output),
        .control(alu_input),
        .zero(alu_zero_output),
        .result(alu_result_output)
);
adder my_adder (
        .add_in1(npcout),
        .add_in2(s_extendout),
        .add_out(adder_output)
);
The_EX_stage EX_register (
        .ctlwb_out(wb_ctlout),
        .ctlm_out(m_ctlout),
        .adder_out(adder_output),
        .aluzero(alu_zero_output),
        .alu_out(alu_result_output),
        .readdat2(rdata2out),
        .mux_out(mux_5bit_output),
        .wb_ctlout(EX_MEM_wb_ctlout),
        .m_ctlout(EX_MEM_m_ctlout),
        .add_result(EX_MEM_add_result),
        .zero(EX_MEM_zero),
```

```
        .alu_result(EX_MEM_alu_result),
        .rdata2out(EX_MEM_rdata2out),
        .five_bit_muxout(EX_MEM_five_bit_muxout)
    );

endmodule
```

2. Test Bench

A. 32-bit Mux

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Hau Tao
//
// Create Date:   11:13:30 11/25/2015
// Design Name:   mux_32bit
// Module Name:   /home/hau/Desktop/lab3/test_mux_32bit.v
// Project Name:  lab3
////////////////////////////////////////////////////////////////////////////////

module test_mux_32bit;

    // Inputs
    reg [31:0] a;
    reg [31:0] b;
    reg sel;

    // Outputs
    wire [31:0] y;

    // Instantiate the Unit Under Test (UUT)
    mux_32bit uut (
            .a(a),
            .b(b),
            .sel(sel),
            .y(y)
    );

    initial begin
            a = 32'b01010;
            b = 32'b10101;
            sel = 1'b1;
            #10;
            a = 32'b00000;
```

```
        #10
        sel = 1'b1;
        #10;
        b = 32'b11111;
        #5;
        a= 32'b00101;
        #5;
        sel = 1'b0;
        b = 32'b11101;
    #5;
    sel = 1'bx;
        end
     always @(a or b or sel)

        #1 $display("At t = %0d sel =%b a =%b b=%b y=%b",$time, sel, a, b, y);


endmodule
```

B. 5-bit Mux

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Hau Tao
//
// Create Date:   09:59:42 11/16/2015
// Design Name:   mux_EX
// Module Name:   /home/hau/Desktop/lab3/test_mux_EX.v
//////////////////////////////////////////////////////////////////////////////////

module test_mux_EX;

    // Inputs
    reg [4:0] a;
    reg [4:0] b;
    reg sel;

    // Outputs
    wire [4:0] y;

    // Instantiate the Unit Under Test (UUT)
    mux_EX uut (
            .a(a),
            .b(b),
            .sel(sel),
            .y(y)
    );

    initial begin
            a = 5'b01010;
            b = 5'b10101;
            sel = 1'b1;
            #10;
            a = 5'b00000;
            #10
            sel = 1'b1;
```

```
        #10;
        b = 5'b11111;
        #5;
        a= 5'b00101;
        #5;
        sel = 1'b0;
        b = 5'b11101;
    #5;
    sel = 1'bx;
        end
     always @(a or b or sel)

        #1 $display("At t = %0d sel =%b a =%b b=%b y=%b",$time, sel, a, b, y);


endmodule
```

C. ALU control unit

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Engineer: Hau Tao
// Create Date:   11:27:34 11/23/2015
// Design Name:   ALU_control_unit
// Project Name:  lab3
//////////////////////////////////////////////////////////////////////////////////

module test_alu_control_unit;

    // Inputs
    reg [5:0] function_bit;
    reg [1:0] ALUop;

    // Outputs
    wire [2:0] select;

    // Instantiate the Unit Under Test (UUT)
    ALU_control_unit uut (
            .function_bit(function_bit),
            .ALUop(ALUop),
            .select(select)
    );

    initial begin
            // Initialize Inputs
            function_bit = 6'b100000;
            ALUop = 2'b00;
            $monitor("ALUop =%b\t function_bit =%b\tselect=%b", ALUop, function_bit, select);

            #1
            ALUop = 2'b01;
            function_bit = 6'b100000;
            #1
            ALUop = 2'b10;
            function_bit = 6'b100000;
```

```
        #1
        function_bit = 6'b100010;
        #1
        function_bit = 6'b100100;
        #1
        function_bit = 6'b100101;
        #1
        function_bit = 6'b101010;
        #1
        $finish;
        end

endmodule
```

D. ALU

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:hau tao
// Create Date:   21:13:13 11/24/2015
// Design Name:   alu
//////////////////////////////////////////////////////////////////////////////////

module test_alu;

    // Inputs
    reg [31:0] A;
    reg [31:0] B;
    reg [2:0] control;

    // Outputs
    wire zero;
    wire [31:0] result;

    // Instantiate the Unit Under Test (UUT)
    alu uut (
            .A(A),
            .B(B),
            .control(control),
            .zero(zero),
            .result(result)
    );

    initial begin
            // Initialize Inputs
            A <= 'b1010;
            B <= 'b0111;
            control <= 'b011;
            $display("A=%b\tB=%b",A,B);
            $monitor("ALUOp=%b\tresult=%b",control, result);
            #1
```

```
        control <= 'b100;
        #1
        control <= 'b010;
        #1
        control <= 'b111;
        #1
        control <= 'b011;
        #1
        control <= 'b110;
        #1
        control <= 'b001;
        #1
        control <= 'b000;
        #1
        $finish;



    end

endmodule
```

E. Adder

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Engineer: Hau Tao
//
// Create Date:   09:39:54 11/16/2015
// Design Name:   adder
// Module Name:   /home/hau/Desktop/lab3/test_adder.v
// Project Name:  lab3
////////////////////////////////////////////////////////////////////////////////

module test_adder;

   // Inputs
   reg [31:0] add_in1;
   reg [31:0] add_in2;

   // Outputs
   wire [31:0] add_out;

   // Instantiate the Unit Under Test (UUT)
   adder uut (
         .add_in1(add_in1),
         .add_in2(add_in2),
         .add_out(add_out)
   );

   initial begin
         // Initialize Inputs
         add_in1 = 0;
         add_in2 = 0;

         // Wait 100 ns for global reset to finish
         #10;
         add_in1 =5;
         add_in2 = 12;
```

```
        // Add stimulus here
        #2
        $display("time = %0d\tadd_in1=%0d\t add_in2=%0d\tadd_out=%0d\t",$time, add_in1,
add_in2, add_out);

    end

endmodule
```

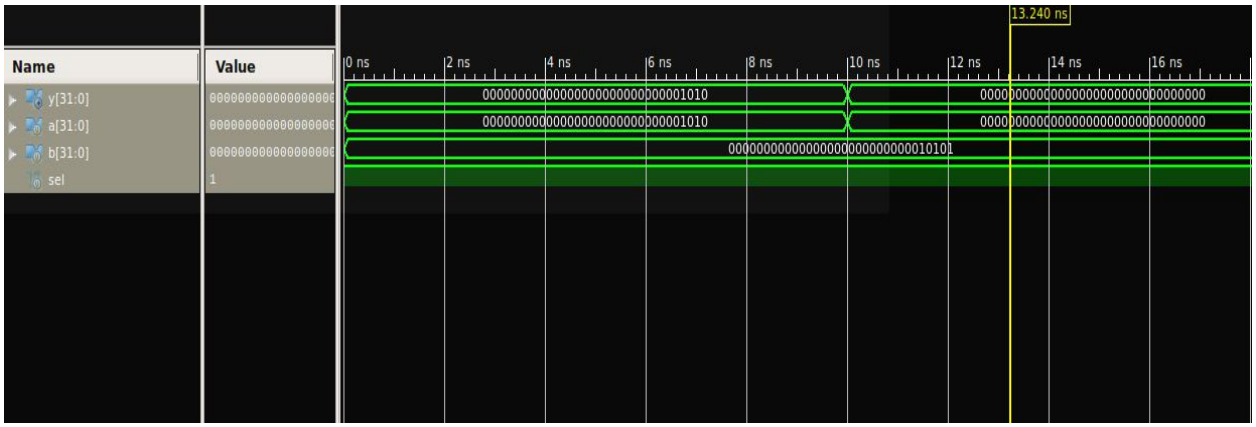
E.EX latch

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Hau Tao
//
// Create Date:   18:21:21 11/25/2015
// Design Name:   The_EX_stage
// Module Name:   /home/hau/Desktop/lab3/testing_EX_latch.v
////////////////////////////////////////////////////////////////////////////////

module testing_EX_latch;

    // Inputs
    reg [1:0] ctlwb_out;
    reg [2:0] ctlm_out;
    reg [31:0] adder_out;
    reg aluzero;
    reg [31:0] alu_out;
    reg [31:0] readdat2;
    reg [4:0] mux_out;

    // Outputs
    wire [1:0] wb_ctlout;
    wire [2:0] m_ctlout;
    wire [31:0] add_result;
    wire zero;
    wire [31:0] alu_result;
    wire [31:0] rdata2out;
    wire [4:0] five_bit_muxout;

    // Instantiate the Unit Under Test (UUT)
    The_EX_stage uut (
            .ctlwb_out(ctlwb_out),
            .ctlm_out(ctlm_out),
            .adder_out(adder_out),
            .aluzero(aluzero),
```

```
        .alu_out(alu_out),
        .readdat2(readdat2),
        .mux_out(mux_out),
        .wb_ctlout(wb_ctlout),
        .m_ctlout(m_ctlout),
        .add_result(add_result),
        .zero(zero),
        .alu_result(alu_result),
        .rdata2out(rdata2out),
        .five_bit_muxout(five_bit_muxout)
    );

    initial begin
        // Initialize Inputs
        ctlwb_out = 0;
        ctlm_out = 0;
        adder_out = 0;
        aluzero = 0;
        alu_out = 0;
        readdat2 = 0;
        mux_out = 0;



        #10
        ctlwb_out = 1;
        ctlm_out = 2;
        adder_out = 3;
        aluzero = 0;
        alu_out = 5;
        readdat2 = 6;
        mux_out = 7;



    end

endmodule
```

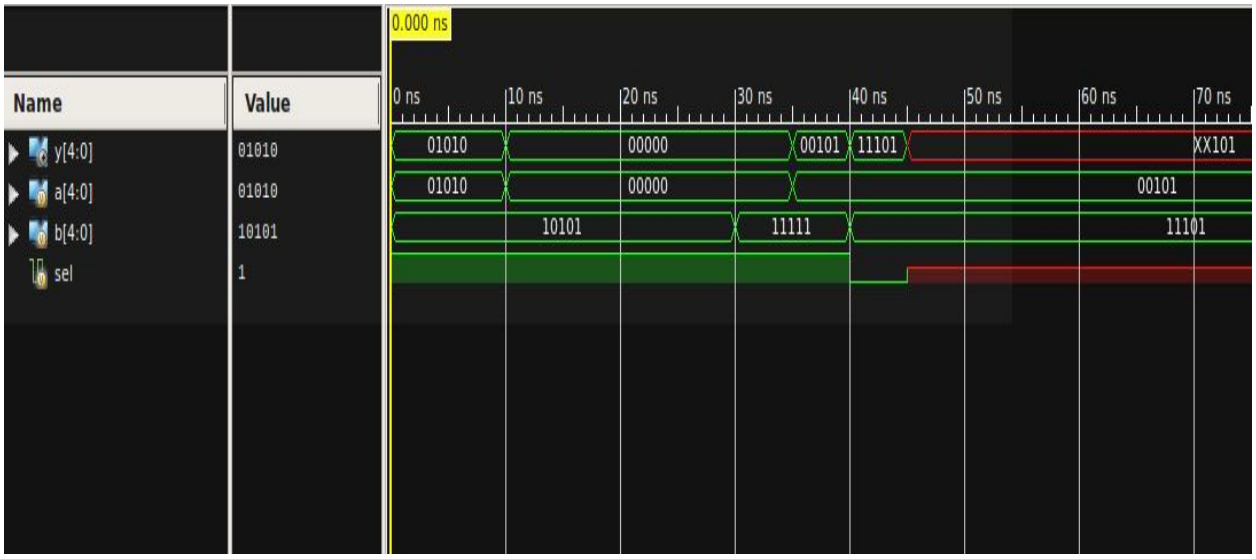
3. Simulation

A. 32-bit mux



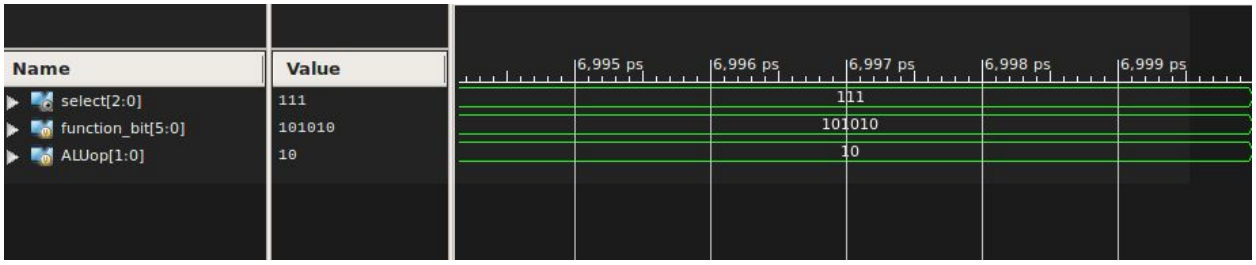
At t = 1 sel =1 a =00000000000000000000000000001010
b=00000000000000000000000000010101 y=00000000000000000000000000001010
At t = 11 sel =1 a =00000000000000000000000000000000
b=00000000000000000000000000010101 y=00000000000000000000000000000000
At t = 31 sel =1 a =00000000000000000000000000000000
b=00000000000000000000000000011111 y=00000000000000000000000000000000
At t = 36 sel =1 a =00000000000000000000000000000101
b=00000000000000000000000000011111 y=00000000000000000000000000000101
At t = 41 sel =0 a =00000000000000000000000000000101
b=00000000000000000000000000011101 y=00000000000000000000000000011101
At t = 46 sel =x a =00000000000000000000000000000101
b=00000000000000000000000000011101 y=0000000000000000000000000000xx101

B. 5-bit mux



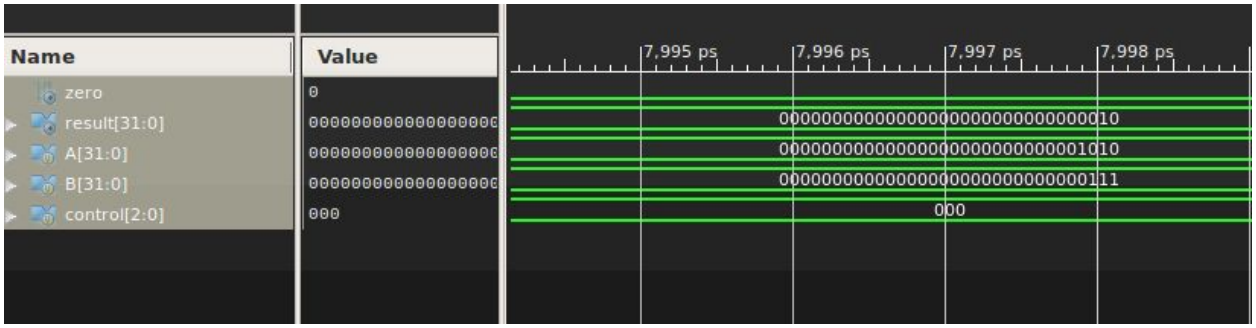
At t = 11 sel =1 a =00000 b=10101 y=00000
At t = 31 sel =1 a =00000 b=11111 y=00000
At t = 36 sel =1 a =00101 b=11111 y=00101
At t = 41 sel =0 a =00101 b=11101 y=11101
At t = 46 sel =x a =00101 b=11101 y=xx101

C.alu control unit



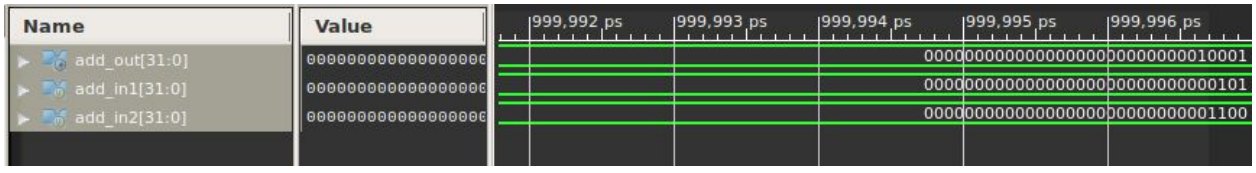
ALUop =10 function_bit =100010select=110
ALUop =10 function_bit =100100select=000
ALUop =10 function_bit =100101select=001
ALUop =10 function_bit =101010select=111

D. ALU



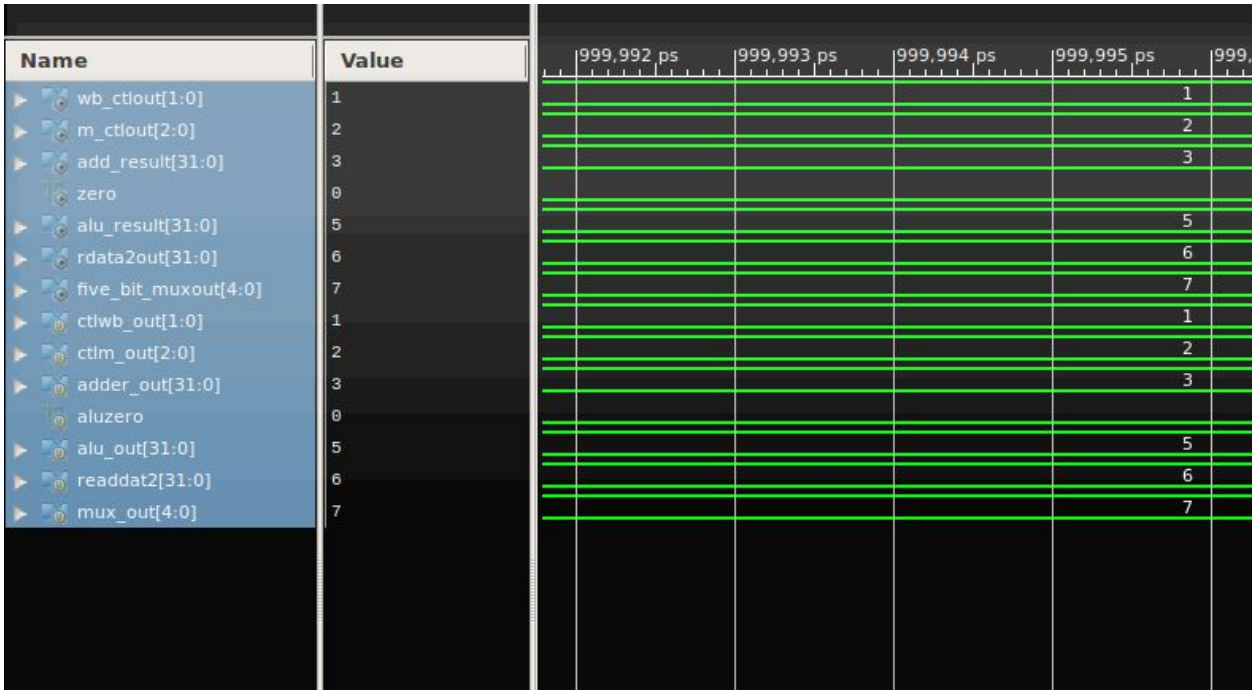
ALUOp=011 result=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ALUOp=100 result=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ALUOp=010 result=00000000000000000000000000010001
ALUOp=111 result=00000000000000000000000000000000
ALUOp=011 result=xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
ALUOp=110 result=00000000000000000000000000000011
ALUOp=001 result=00000000000000000000000000001111
ALUOp=000 result=00000000000000000000000000000010

E. Adder



time = 12 add_in1=5 add_in2=12 add_out=17

F. EX latch



It’s hard to test entire ex stage separately. As a mention of professor, I will test it when connecting all stages together