Lab 2: The MIPS datapath in Verilog: The ID stage

A. Introduction

The objective of this lab is to implement and test the instruction decoder (ID) pipeline stage of the MIPS five stages pipeline. In this lab, I build up and test successfully 4 components of ID stages including control, register, sign extend, and ID latch.

B. Interface

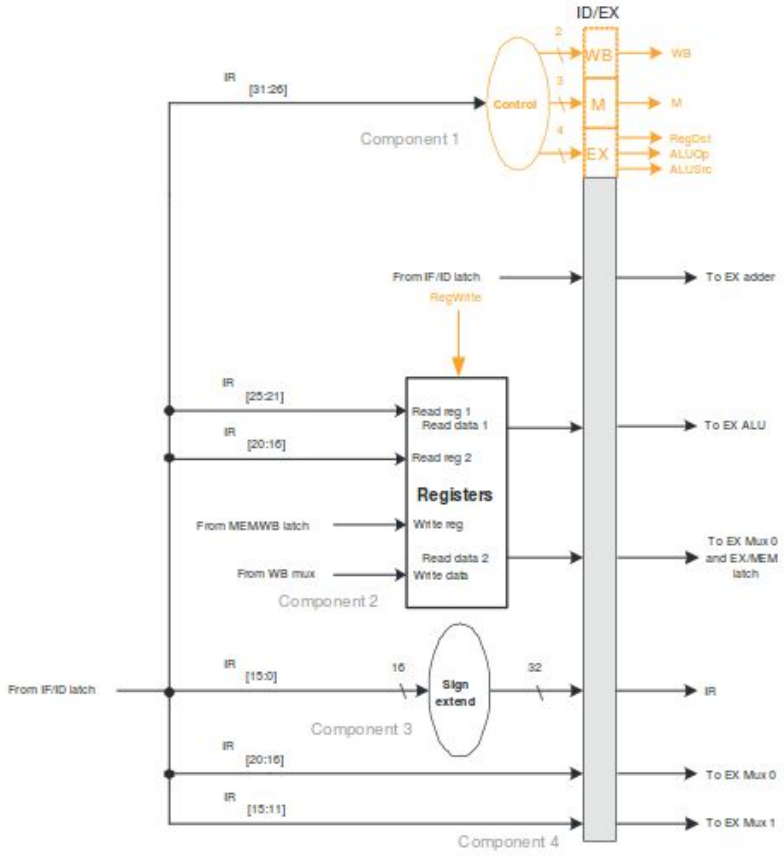


Figure 2.1: The ID stage

The Instruction decode (ID) stage figures it out that what instructions say to do and get the value from the registers, and the control unit will define what the output is . These outputs include 3 signals: M, WB, EX. We also need a sign extended unit to extend from 16 to 32-bit output

C. Design

The module I_DECODE instantiates the modules CONTROL, REG, S_EXTEND, and ID_EX

- The CONTROL module has input the opcode field of the IF_ID instr and output is the 9-bit control bits which are shown in figure which are shown in figure 2.2
- .The register file REG, which has 32 general purpose registers, and has input the rs and rt fields of IF_ID_ instr, MEM_WB_Writereg, MEM_WB_Writedata, and RegWrite (for the time being it can be from anywhere). Outputs are the contents of register rs and register rt.
- The combinational module S_EXTEND receives as input the 16-bit immediate field of IF_ID_instr and output is the 32 bit sign-extended value.
- The ID_EX module includes the pipeline register ID/EX and inputs the outputs of the CONTROL, REG, S_EXTEND modules, as well as the IF_ID_NPC, IF_ID_Instr[20-16] and IF_ID_Instr[15-11]. Outputs are the control bits (9 bits) NPC, Reg[rs], Reg[rt],signExtended (32 bit), Instr[20-16], and Instr[15-11].

	Execution/Address Calculation stage control lines				Memory access stage control lines			Write-back stage control lines	
Instruction	Reg Dst	ALU Op1	ALU Op0	ALU Src	Branch	Mem Read	Mem Write	Reg Write	Mem to Reg
R-format	1	1	0	0	0	0	0	1	0
lw	0	0	0	1	0	1	0	1	1
sw	X	0	0	1	0	0	1	0	X
beq	X	0	1	0	1	0	0	0	X

Figure 2.2: ALUOp Control Bit and Function Code Sets (after [4])

The content of the register

```
002300AA
10654321
00100022
8C123456
8F123456
AD654321
13012345
AC654321
12012345
```

D. Implementation

The sign-extended unit

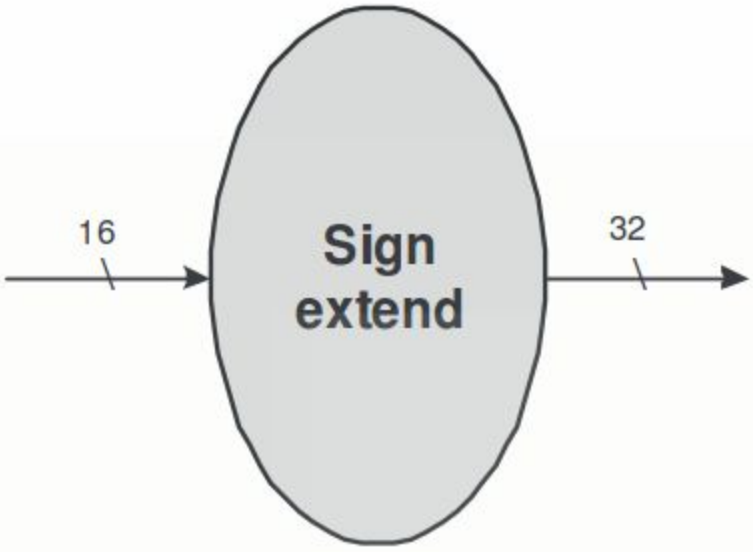


Figure 2.3: The sign-extend unit

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Engineer: hau tao
// Create Date:    14:07:13 11/07/2015
// Design Name:    extended_register
// Module Name:
/home/hau/Desktop/labcse401/lab2/ID_stage/test_extended_signal.v
// Project Name:  ID_stage
//////////////////////////////////////////////////////////////////////////////////

module test_extended_signal;
```

```
    // Inputs
    reg [15:0] unextended_signal;

    // Outputs
    wire [31:0] extended_signal;

    // Instantiate the Unit Under Test (UUT)
    extended_register uut (
        .unextended_signal(unextended_signal),
        .extended_signal(extended_signal)
    );

    initial begin
        // Initialize Inputs
        unextended_signal = 0;

        // Wait 1ns for global reset to finish

        #2 unextended_signal = 16'b1110000100111111;
        #3 unextended_signal = 16'b0000000100111111;
       end
        // Add stimulus here
        always @*
        begin
        #2 $display("Time = %0d\t unextended_signal =%0b\t
extended_signal=%0b\t", $time, unextended_signal, extended_signal);
        #1 $display("Time = %0d\t unextended_signal =%0b\t
extended_signal=%0b\t", $time, unextended_signal, extended_signal);
end
endmodule
```

- The control unit

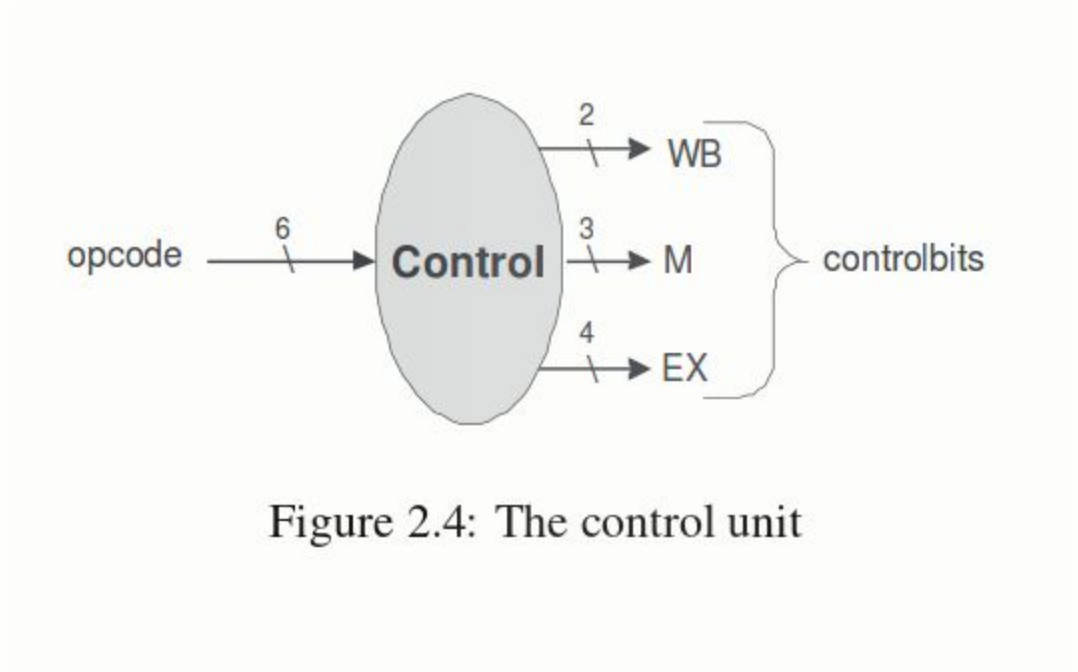


Figure 2.4: The control unit

```
`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////
// Engineer: Hau Tao
//
// Create Date:   14:35:07 11/07/2015
// Design Name:   control_bit
// Module Name:   /home/hau/Desktop/labcse401/lab2/ID_stage/test_control_bit.v
// Project Name:  ID_stage
//////////////////////////////////////////////////////////////////////////////////

module test_control_bit;

    // Inputs
    reg [5:0] opcode;

    // Outputs
    wire [3:0] EX;
    wire [2:0] MEM;
```

```
    wire [1:0] WB;

    // Instantiate the Unit Under Test (UUT)
    control_bit uut (
            .opcode(opcode),
            .EX(EX),
            .MEM(MEM),
            .WB(WB)
    );

    initial begin
            // Initialize Inputs
            opcode = 6'b000000;
    #2 opcode = 6'b100011;
    #4 opcode = 6'b101011;
    #6 opcode = 6'b000100;
    end

    always @*
    begin
    #2 $display("Time = %0d\t opcode =%0b\t EX=%0b\t MEM=%0b\t WB=%0b\t", $time,
opcode, EX, MEM, WB);
    #4 $display("Time = %0d\t opcode =%0b\t EX=%0b\t MEM=%0b\t WB=%0b\t", $time,
opcode, EX, MEM, WB);
    #6 $display("Time = %0d\t opcode =%0b\t EX=%0b\t MEM=%2b\t WB=%0b\t", $time,
opcode, EX, MEM, WB);
    #4 $display("Time = %0d\t opcode =%0b\t EX=%0b\t MEM=%0b\t WB=%0b\t", $time,
opcode, EX, MEM, WB);
    end

endmodule
```

- The register File

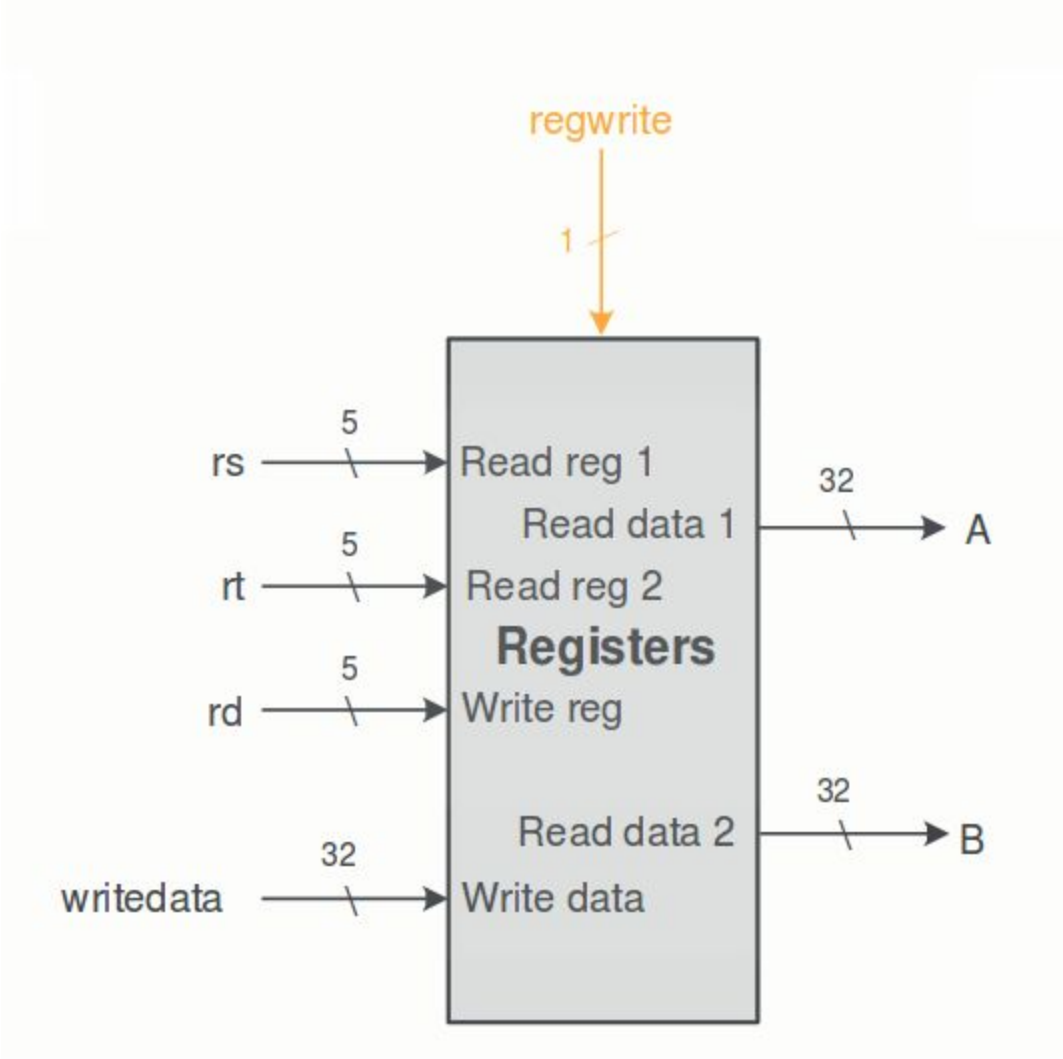


Figure 2.5: The register file

`timescale 1ns / 1ps

//
// Company:

```
// Engineer: hau tao
//
// Create Date:   15:34:02 11/07/2015
// Design Name:   register_file
// Module Name:   /home/hau/Desktop/labcse401/lab2/ID_stage/test_reg.v
// Project Name:  ID_stage
////////////////////////////////////////////////////////////////////////////////

module test_reg;

    // Inputs
    reg reg_write;
    reg [4:0] rs;
    reg [4:0] rt;
    reg [4:0] rd;
    reg [31:0] write_data;
    reg [31:0]register[0:4];

    integer i;

    // Outputs
    wire [31:0] A;
    wire [31:0] B;

    // Instantiate the Unit Under Test (UUT)
    register_file uut (
            .reg_write(reg_write),
            .rs(rs),
            .rt(rt),
            .rd(rd),
            .write_data(write_data),
            .A(A),
            .B(B)
    );

    initial begin
            // Initialize Inputs
            register[0] <= 'h002300AA;
            register[1] <= 'h10654321;
```

```
        register[2] <= 'h00100022;



        // 2ns delay
        #2
        begin
        reg_write = 1;
        rs = 1;
        rt = 2;
        rd = 2;
        write_data = 5;
        end


    end

    always@*
    begin
    #3 $display("Time = %0d\t rs =%0h\t rt=%0h\t rd=%0h\t write_data=%0h\t register[rd]
=%0h\t  A=%0h\t B=%1h\t", $time, rs, rt, rd, write_data,register[rd], A, B);

    end
```

- The ID/EX latch

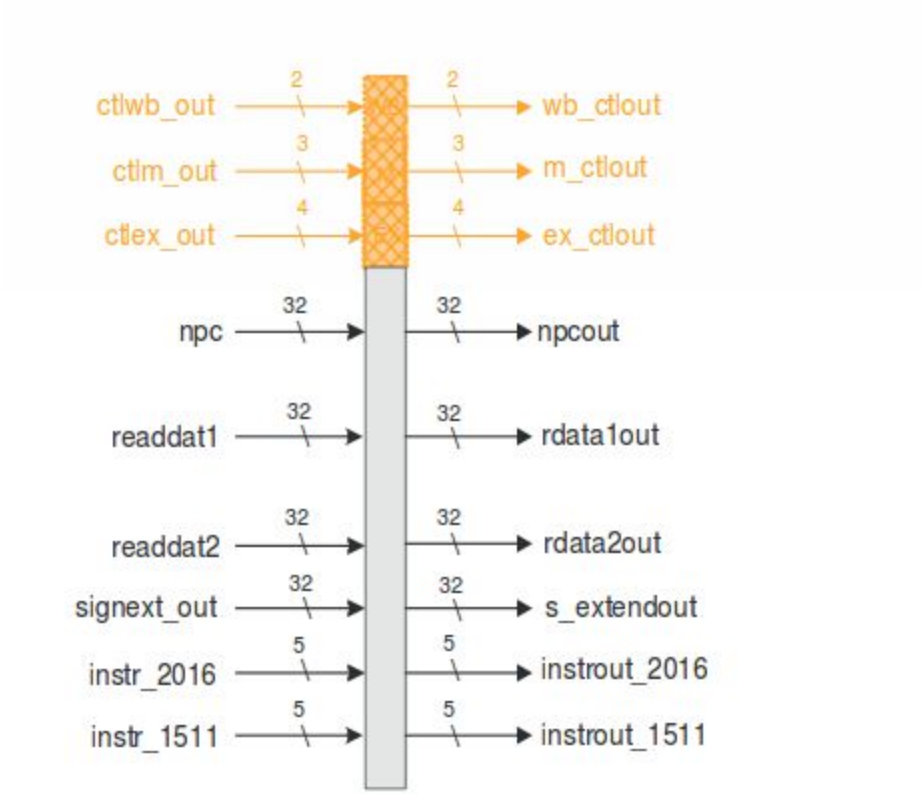


Figure 2.6: The ID/EX pipeline register (latch)

```
module ID_latch (
        input[1:0]  ctlwb_out,
        input[2:0]  ctlm_out,
        input[3:0]  ctlex_out,
        input[31:0] npc,
        input[31:0] readdat1,
        input[31:0] readdat2,
        input[31:0] signext_out,
        input[4:0]  instr_2016,
        input[4:0]  instr_1511,

        output reg[1:0]  wb_ctlout,
```

```
        output reg[2:0]  m_ctlout,
          output reg[3:0]  ex_ctlout,
        output reg[31:0] npcout,
        output reg[31:0] rdata1out,
        output reg[31:0] rdata2out,
        output reg[31:0] s_extendout,
        output reg[4:0]  instrout_2016,
        output reg[4:0]  instrout_1511
        );



        initial begin

    wb_ctlout <= 0;
        m_ctlout  <= 0;
    ex_ctlout <= 0;
        npcout <= 0;
        rdata1out <= 0;
        rdata2out <= 0;
        s_extendout<= 0;
        instrout_2016 <= 0;
        instrout_1511 <= 0;
    end
    always @*
        begin
        #1  // clock cycle of the latch
        wb_ctlout <= ctlwb_out;
                m_ctlout  <= ctlm_out;
                ex_ctlout <= ctlex_out;
                npcout<= npc;
                rdata1out <= readdat1;
                rdata2out <= readdat2;
                s_extendout<= signext_out;
                instrout_2016 <= instr_2016;
                instrout_1511 <= instr_1511;
        end
endmodule
```

- The completed ID stage

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Engineer: Hau Tao
// Create Date: 16:23:46 11/12/2015
// Module Name:        final_ID_stage
//////////////////////////////////////////////////////////////////////////////////
module final_ID_stage(input[31:26] IR_control, input[31:0] npc_in, input[25:21] IR_reg1,
input[20:16] IR_reg2, input[15:0] unextended_bit,
        input[20:16] IR_mux0, input[15:11] IR_mux1, input WB, input[31:0]
mux_output_MEM_WB, input[4:0] MEM_WB,
          output wire [1:0] wb_ctlout,
        output wire [2:0] m_ctlout,
        output wire[3:0] ex_ctlout,
        output wire[31:0] npcout,
        output wire[31:0] rdata1out,
        output wire[31:0] rdata2out,
        output wire [31:0] s_extendout,
        output wire [4:0] instrout_2016,
        output wire [4:0] instrout_1511



        );


    wire [31:0] read_data_1;
    wire [31:0] read_data_2;
    wire [31:0] extended_signal_1;
    wire [3:0] EX_out;
    wire [2:0] MEM_out;
    wire [1:0] WB_out;
    register_file register1 (
          .reg_write(WB),
```

```
        .rs(IR_reg1),
        .rt(IR_reg2),
        .rd(MEM_WB),
        .write_data(mux_output_MEM_WB),
        .A(read_data_1),
        .B(read_data_2)
        );

extended_register extended_register_1 (
        .unextended_signal(unextended_bit),
        .extended_signal(extended_signal_1)
);

control_bit control_bit_1(
        .opcode(IR_control),
        .EX(EX_out),
        .MEM(MEM_out),
        .WB(WB_out)
);

ID_latch ID_latch_1 (
        .ctlwb_out(WB_out),
        .ctlm_out(MEM_out),
        .ctlex_out(EX_out),
        .npc(npc_in),
        .readdat1(read_data_1),
        .readdat2(read_data_2),
        .signext_out(extended_signal_1),
        .instr_2016(IR_mux0),
        .instr_1511(IR_mux1),
        .wb_ctlout(wb_ctlout),
        .m_ctlout(m_ctlout),
        .ex_ctlout(ex_ctlout),
        .npcout(npcout),
        .rdata1out(rdata1out),
        .rdata2out(rdata2out),
        .s_extendout(s_extendout),
        .instrout_2016(instrout_2016),
        .instrout_1511(instrout_1511)
```

```
    );


endmodule
```

- Connection IF_ID

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Hau Tao
//
// Create Date: 17:02:49 11/13/2015
// Design Name:
// Module Name:    connectionID_IF
/
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module connectionID_IF(input  PCSrc, input [31:0] EX_MEM,
input WB,
        output wire [1:0] wb_ctlout,
       output wire [2:0] m_ctlout,
       output wire[3:0] ex_ctlout,
       output wire[31:0] npcout,
       output wire[31:0] rdata1out,
       output wire[31:0] rdata2out,
       output wire [31:0] s_extendout,
       output wire [4:0] instrout_2016,
```

```
    output wire [4:0] instrout_1511

    );
 wire [31:0] npcout_IF_ID;
 wire [31:0] instrout_IF_ID;



IF_stage IF_1 (
     .PCSrc(PCSrc),
     .EX_MEM(EX_MEM),
     .npcout(npcout_IF_ID),
     .instrout(instrout_IF_ID)
);



// Instantiate the Unit Under Test (UUT)

final_ID_stage uut (
    .IR_control(instrout_IF_ID[31:26]),
    .npc_in(npcout_IF_ID),
    .IR_reg1(instrout_IF_ID[25:21]),
    .IR_reg2(instrout_IF_ID[20:16]),
    .unextended_bit(instrout_IF_ID[15:0]),
```

```
        .IR_mux0(instrout_IF_ID[20:16]),
        .IR_mux1(instrout_IF_ID[15:11]),
        .WB(WB),
        .mux_output_MEM_WB(mux_output_MEM_WB),
        .MEM_WB(MEM_WB),
        .wb_ctlout(wb_ctlout),
        .m_ctlout(m_ctlout),
        .ex_ctlout(ex_ctlout),
        .npcout(npcout),
        .rdata1out(rdata1out),
        .rdata2out(rdata2out),
        .s_extendout(s_extendout),
        .instrout_2016(instrout_2016),
        .instrout_1511(instrout_1511)
        );

    always@*
    # 20 $display("time =%0d\t instrout_IF_ID=%0b\t,
npcout_IF_ID=%0b\t", $time, instrout_IF_ID, npcout_IF_ID);



endmodule
```

E. Test bench design

1.The sign-extended unit

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Engineer: hau tao
// Create Date:   14:07:13 11/07/2015
// Design Name:   extended_register
// Module Name:
/home/hau/Desktop/labcse401/lab2/ID_stage/test_extended_signal.v
// Project Name:  ID_stage
////////////////////////////////////////////////////////////////////////////////

module test_extended_signal;

    // Inputs
    reg [15:0] unextended_signal;

    // Outputs
    wire [31:0] extended_signal;

    // Instantiate the Unit Under Test (UUT)
    extended_register uut (
        .unextended_signal(unextended_signal),
        .extended_signal(extended_signal)
    );

    initial begin
        // Initialize Inputs
```

```
        unextended_signal = 0;

        // Wait 1ns for global reset to finish

        #1 unextended_signal = 16'b1110000100111111;
        #2 unextended_signal = 16'b0000000100111111;
       end
        // Add stimulus here
        always @*
        begin
        #2 $display("Time = %0d\t unextended_signal =%0b\t
extended_signal=%0b\t", $time, unextended_signal, extended_signal);
        #1 $display("Time = %0d\t unextended_signal =%0b\t
extended_signal=%0b\t", $time, unextended_signal, extended_signal);
       end
endmodule
```

2. The control unit

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Engineer: Hau Tao
//
// Create Date:   14:35:07 11/07/2015
// Design Name:   control_bit
// Module Name:   /home/hau/Desktop/labcse401/lab2/ID_stage/test_control_bit.v
// Project Name:  ID_stage
////////////////////////////////////////////////////////////////////////////////

module test_control_bit;

    // Inputs
    reg [5:0] opcode;

    // Outputs
    wire [3:0] EX;
    wire [2:0] MEM;
    wire [1:0] WB;

    // Instantiate the Unit Under Test (UUT)
    control_bit uut (
            .opcode(opcode),
            .EX(EX),
            .MEM(MEM),
            .WB(WB)
    );

    initial begin
            // Initialize Inputs
            opcode = 6'b000000;
    #2 opcode = 6'b100011;
    #4 opcode = 6'b101011;
    #6 opcode = 6'b000100;
```

```
    end

    always @*
    begin
    #2 $display("Time = %0d\t opcode =%0b\t EX=%0b\t MEM=%0b\t WB=%0b\t", $time,
opcode, EX, MEM, WB);
    #4 $display("Time = %0d\t opcode =%0b\t EX=%0b\t MEM=%0b\t WB=%0b\t", $time,
opcode, EX, MEM, WB);
    #6 $display("Time = %0d\t opcode =%1b\t EX=%0b\t MEM=%0b\t WB=%1b\t", $time,
opcode, EX, MEM, WB);

    end

endmodule
```

3. The register file

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: hau tao
//
// Create Date:   15:34:02 11/07/2015
// Design Name:   register_file
// Module Name:   /home/hau/Desktop/labcse401/lab2/ID_stage/test_reg.v
// Project Name:  ID_stage
////////////////////////////////////////////////////////////////////////////////

module test_reg;

    // Inputs
    reg reg_write;
    reg [4:0] rs;
    reg [4:0] rt;
    reg [4:0] rd;
    reg [31:0] write_data;
    reg [31:0]register[0:4];

    integer i;

    // Outputs
    wire [31:0] A;
    wire [31:0] B;

    // Instantiate the Unit Under Test (UUT)
    register_file uut (
            .reg_write(reg_write),
            .rs(rs),
            .rt(rt),
            .rd(rd),
            .write_data(write_data),
            .A(A),
```

```
        .B(B)
    );

    initial begin
        // Initialize Inputs
        register[0] <= 'h002300AA;
        register[1] <= 'h10654321;
        register[2] <= 'h00100022;



        // 2ns delay
        #2
        begin
        reg_write = 1;
        rs = 1;
        rt = 2;
        rd = 2;
        write_data = 5;
        end


    end

    always@*
    begin
    #3 $display("Time = %0d\t rs =%0h\t rt=%0h\t rd=%0h\t write_data=%0h\t register[rd]
=%0h\t  A=%0h\t B=%1h\t", $time, rs, rt, rd, write_data,register[rd], A, B);

    end


endmodule
```

4. The ID/EX latch

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Engineer:Hau Tao
//
// Create Date:   15:49:27 11/12/2015
// Design Name:   ID_latch
// Module Name:   /home/hau/Desktop/labcse401/lab2/ID_stage/test_ID.v
////////////////////////////////////////////////////////////////////////////////

module test_ID;

    // Inputs
    reg [1:0] ctlwb_out;
    reg [2:0] ctlm_out;
    reg [3:0] ctlex_out;
    reg [31:0] npc;
    reg [31:0] readdat1;
    reg [31:0] readdat2;
    reg [31:0] signext_out;
    reg [4:0] instr_2016;
    reg [4:0] instr_1511;

    // Outputs
    wire [1:0] wb_ctlout;
    wire [2:0] m_ctlout;
    wire [3:0] ex_ctlout;
    wire [31:0] npcout;
    wire [31:0] rdata1out;
    wire [31:0] rdata2out;
    wire [31:0] s_extendout;
    wire [4:0] instrout_2016;
    wire [4:0] instrout_1511;

    // Instantiate the Unit Under Test (UUT)
    ID_latch uut (
```

```
        .ctlwb_out(ctlwb_out),
        .ctlm_out(ctlm_out),
        .ctlex_out(ctlex_out),
        .npc(npc),
        .readdat1(readdat1),
        .readdat2(readdat2),
        .signext_out(signext_out),
        .instr_2016(instr_2016),
        .instr_1511(instr_1511),
        .wb_ctlout(wb_ctlout),
        .m_ctlout(m_ctlout),
        .ex_ctlout(ex_ctlout),
        .npcout(npcout),
        .rdata1out(rdata1out),
        .rdata2out(rdata2out),
        .s_extendout(s_extendout),
        .instrout_2016(instrout_2016),
        .instrout_1511(instrout_1511)
);

initial begin
        // Initialize Inputs
        ctlwb_out = 0;
        ctlm_out = 0;
        ctlex_out = 0;
        npc = 0;
        readdat1 = 0;
        readdat2 = 0;
        signext_out = 0;
        instr_2016 = 0;
        instr_1511 = 0;

        // Wait 10 ns for global reset to finish

        #10
        ctlwb_out = 1;
        ctlm_out = 2;
        ctlex_out = 3;
        npc = 4;
```

```
        readdat1 = 5;
        readdat2 = 6;
        signext_out =7;
        instr_2016 = 8;
        instr_1511 = 9;

        //#10;
        //$finish;
        end



endmodule
```

5. The completed ID stage

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:   16:29:56 11/13/2015
// Design Name:   final_ID_stage
// Module Name:   /home/hau/Desktop/labcse401/lab2/ID_stage/testing_ID_stage.v
// Project Name:  ID_stage
// Target Device:
// Tool versions:
// Description:
//
// Verilog Test Fixture created by ISE for module: final_ID_stage
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////

module testing_ID_stage;

    // Inputs
    reg [31:26] IR_control;
    reg [31:0] npc_in;
    reg [25:21] IR_reg1;
    reg [20:16] IR_reg2;
    reg [15:0] unextended_bit;
    reg [20:16] IR_mux0;
    reg [15:11] IR_mux1;
    reg WB;
    reg [31:0] mux_output_MEM_WB;
    reg [4:0] MEM_WB;
```

```
// Outputs
wire [1:0] wb_ctlout;
wire [2:0] m_ctlout;
wire [3:0] ex_ctlout;
wire [31:0] npcout;
wire [31:0] rdata1out;
wire [31:0] rdata2out;
wire [31:0] s_extendout;
wire [4:0] instrout_2016;
wire [4:0] instrout_1511;

// Instantiate the Unit Under Test (UUT)
final_ID_stage uut (
	.IR_control(IR_control),
	.npc_in(npc_in),
	.IR_reg1(IR_reg1),
	.IR_reg2(IR_reg2),
	.unextended_bit(unextended_bit),
	.IR_mux0(IR_mux0),
	.IR_mux1(IR_mux1),
	.WB(WB),
	.mux_output_MEM_WB(mux_output_MEM_WB),
	.MEM_WB(MEM_WB),
	.wb_ctlout(wb_ctlout),
	.m_ctlout(m_ctlout),
	.ex_ctlout(ex_ctlout),
	.npcout(npcout),
	.rdata1out(rdata1out),
	.rdata2out(rdata2out),
	.s_extendout(s_extendout),
	.instrout_2016(instrout_2016),
	.instrout_1511(instrout_1511)
);

initial begin
	// Initialize Inputs
	IR_control = 0;
	npc_in = 0;
```

```
        IR_reg1 = 0;
        IR_reg2 = 0;
        unextended_bit = 0;
        IR_mux0 = 0;
        IR_mux1 = 0;
        WB = 0;
        mux_output_MEM_WB = 0;
        MEM_WB = 0;

        // Wait 100 ns for global reset to finish
        // Wait 100 ns for global reset to finish
        #10
        IR_control = 6'b000000;
        npc_in = 2;
        IR_reg1 = 0;
        IR_reg2 = 2;
        unextended_bit = 16'b1010111111101100;
        IR_mux0 = 9;
        IR_mux1 = 10;
        WB = 1;
        MEM_WB = 2;
        mux_output_MEM_WB = 5;

        // Add stimulus here

    end

endmodule
```

6. Connection IF_ID

```
`timescale 1ns / 1ps

////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer: Hau Tao
//
// Create Date:   17:37:38 11/13/2015
// Design Name:   connectionID_IF
// Module Name:   /home/hau/Desktop/labcse401/connection_ID_IF/testing_ID_IF.v
////////////////////////////////////////////////////////////////////////////////

module testing_ID_IF;

    // Inputs
    reg PCSrc;
    reg [31:0] EX_MEM;
    reg WB;

    // Outputs
    wire [1:0] wb_ctlout;
    wire [2:0] m_ctlout;
    wire [3:0] ex_ctlout;
    wire [31:0] npcout;
    wire [31:0] rdata1out;
    wire [31:0] rdata2out;
    wire [31:0] s_extendout;
    wire [4:0] instrout_2016;
    wire [4:0] instrout_1511;

    // Instantiate the Unit Under Test (UUT)
    connectionID_IF uut (
            .PCSrc(PCSrc),
            .EX_MEM(EX_MEM),
            .WB(WB),
            .wb_ctlout(wb_ctlout),
            .m_ctlout(m_ctlout),
            .ex_ctlout(ex_ctlout),
```

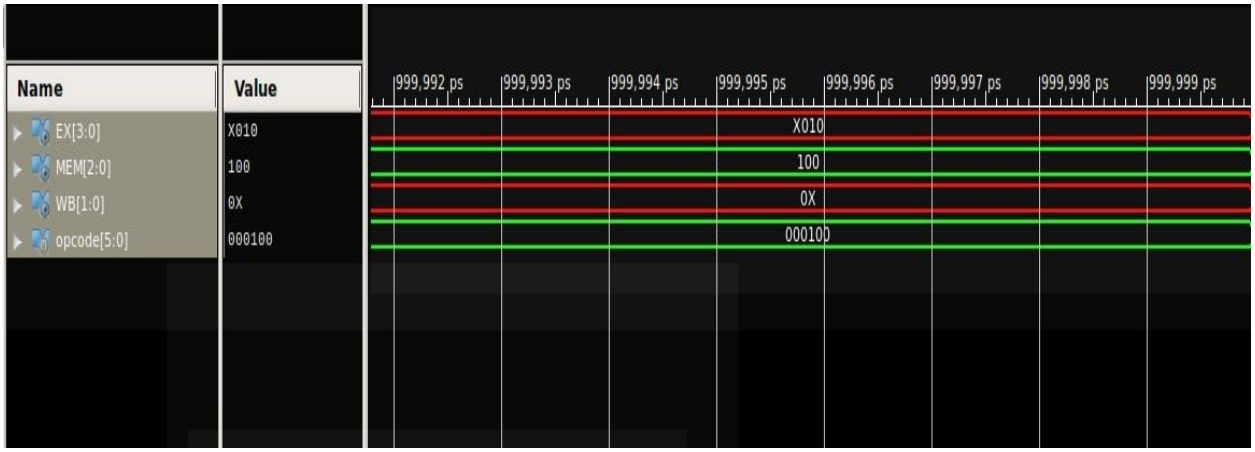
```
        .npcout(npcout),
        .rdata1out(rdata1out),
        .rdata2out(rdata2out),
        .s_extendout(s_extendout),
        .instrout_2016(instrout_2016),
        .instrout_1511(instrout_1511)
    );

    initial begin
        // Initialize Inputs
        PCSrc = 0;
        EX_MEM = 0;
        WB = 0;

        // Wait 100 ns for global reset to finish
        #10;
        PCSrc = 1;
        EX_MEM = 1;
        WB = 1;

        // Add stimulus here

    end

endmodule
```

E. Time Simulation

1.The sign-extended unit


```
Time = 2    unextended_signal =1110000100111111        extended_signal=11111111111111111110000100111111
Time = 3    unextended_signal =100111111     extended_signal=11111111111111111110000100111111
```

2. The control unit



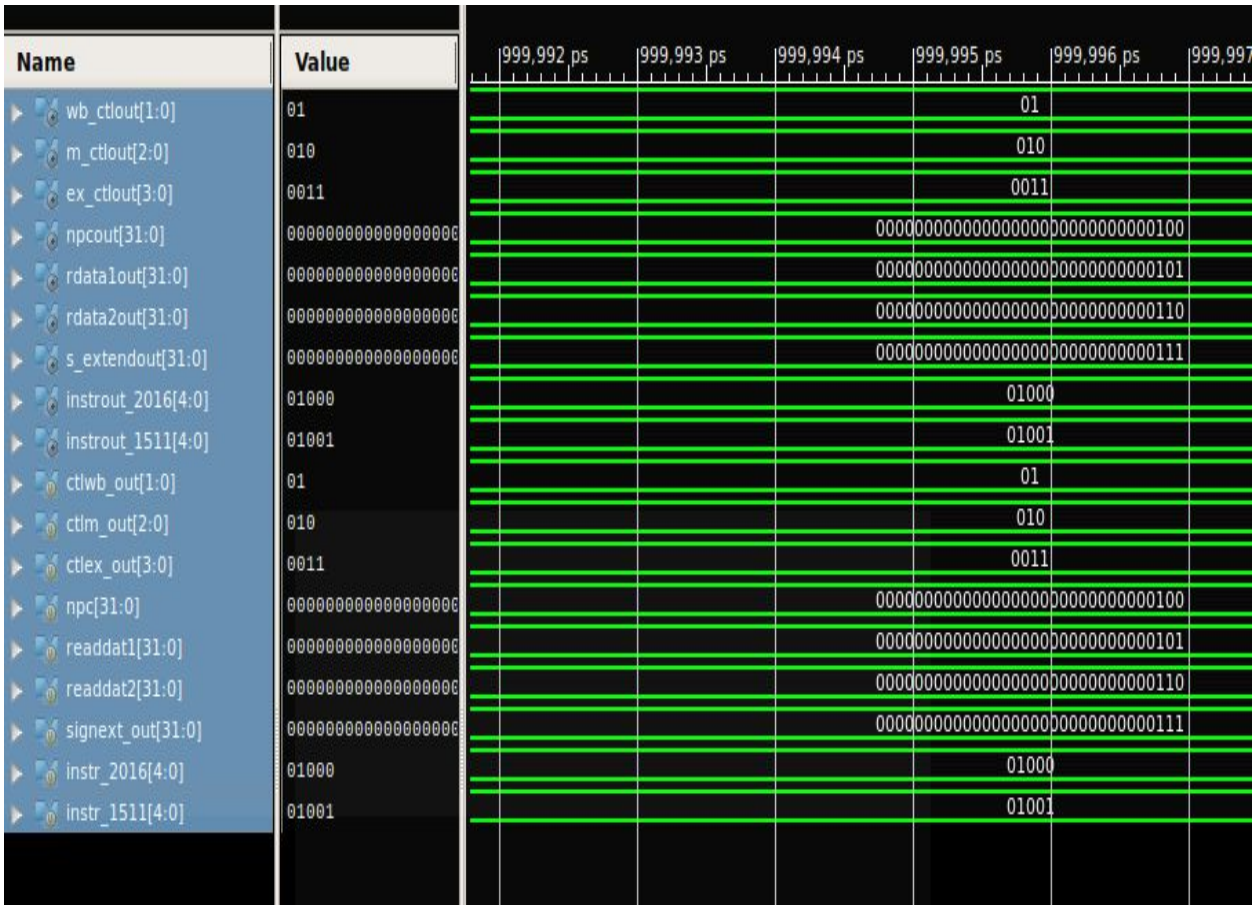
Time = 2	opcode =100011	EX=1100	MEM=000	WB=10
Time = 6	opcode =101011	EX=0001	MEM=010	WB=11
Time = 24	opcode =000100	EX=x010	MEM=100	WB=0x

3. The register file

Time = 3 rs =1 rt=2 rd=2 write_data=5 register[rd] =100022 A=10654321 B=00000005

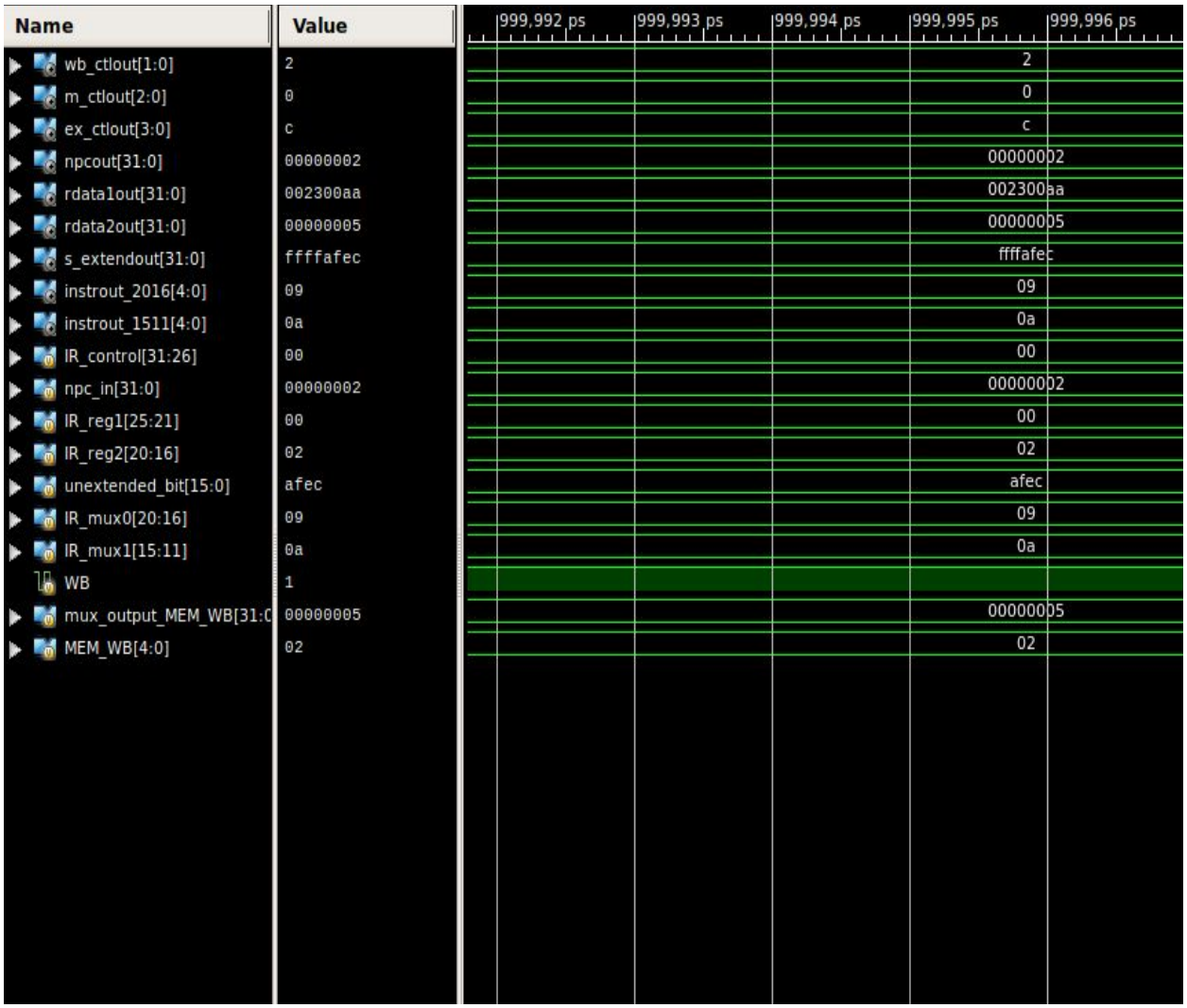
Note: all values are hex-presented

4. The ID/EX latch



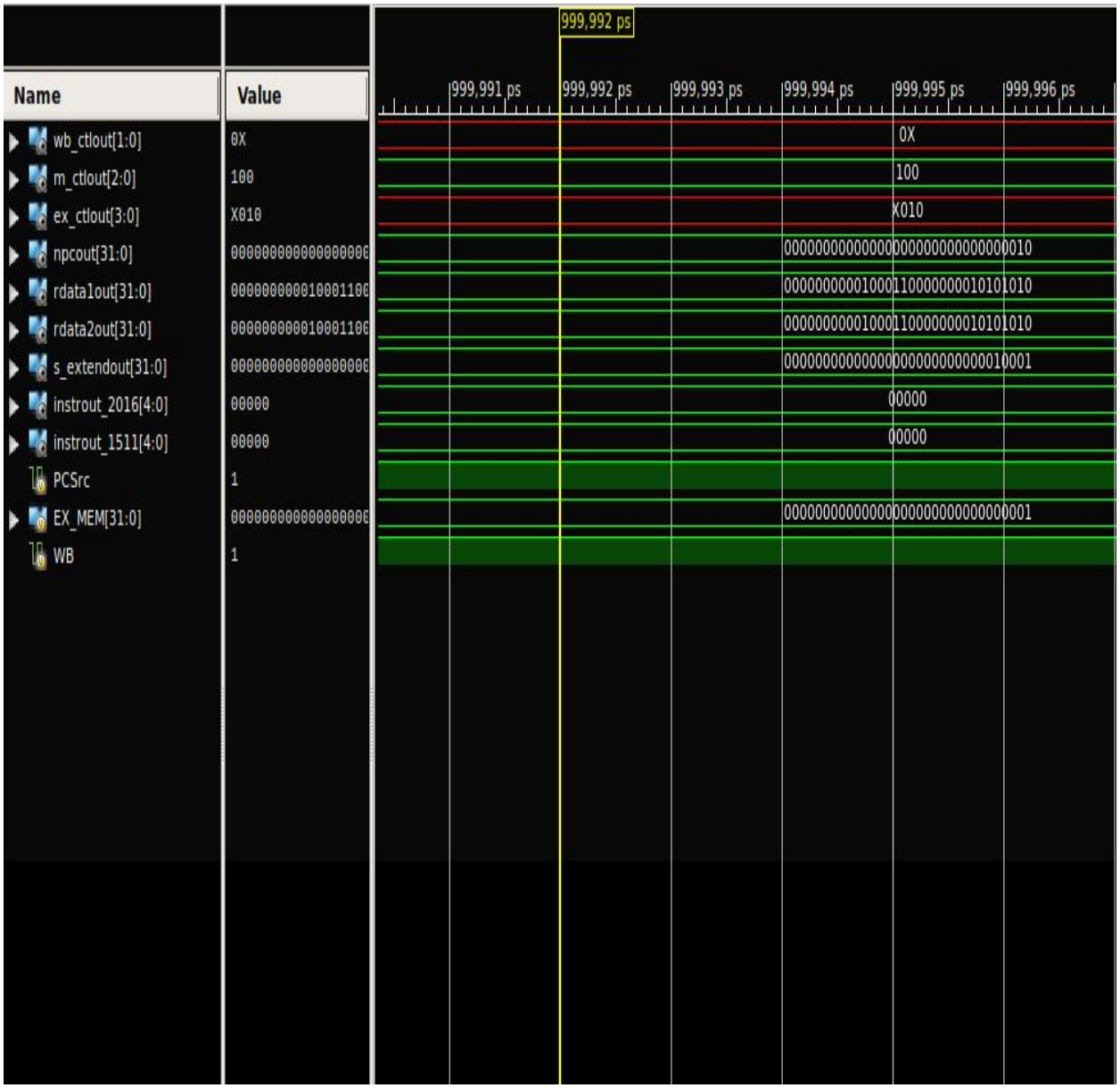
Note: The first 9 variables are the values of output, the last 9 variables are values of input. It's apparent that the outputs are same exactly with the inputs. All values are hex represented

5. The completed ID/EX stage



Explanation: All values are hex representation
when IR_control opcode = 0x0, thw wb_ctilout =0x2, m_ctlout =0x0, ex_ctlout =0xC
npc output = npc_in = 0x2
IR_mux0 = instrout_2016 =0x9
IR_mux1 =instr_1511 =0xA
unextended_bit = 0xafec => extended_bit = 0xffffafec
with WB= 1, it is enable register file and ready to read data from register file. rdata1out = 0x2300a which is reading content at register[0], and rdata2out = 0x10654321 which read from register[2]. at this time rd =2, the content of register[rd] = register[2] = mux_ouput_MEM_WB =0x5. Therefore, rda2out = 0x5

6. Connection IF and ID



time =35 instrout_IF_ID=10000000000000000000000010001 , npcout_IF_ID=10

G. Conclusion

The purpose of this project is to implement ID module. I completed it successfully and the test cases is good enough .The most difficult part of this module is to implement register file, but due to clear explanation from professor and TA, I could finish it