VE370 Introduction to Computer Organization

Project 2

Digital Design Using Verilog to Implement Single-cycle & Pipelined Processors

Team Members:

- *QI Fubo 516370910175*
- *HU Bingcheng 516021910219*
- *NG Jing Ni Ashley 515370990007*

Contents

9. Mux_N_bit.v
10. pc.v
11. pipeline.v
12. pipe_test.v
13. register.v
14. sign_extension.v
15. state_register.v
16. control.v

I. Objectives

- To build up both single-cycle and pipelined datapaths and construct a simple version of a processor sufficient to implement a subset of the MIPS instruction set that includes:
    1. The **memory-reference instructions** load word (`lw`) and store word (`sw`)
    2. The **arithmetic-logical instructions** `add`, `addi`, `sub`, `and`, `andi`, `or`, and `slt`
    3. The **jumping instructions** branch equal (`beq`), branch not equal (`bne`), and jump (`j`)
- To model and simulate the single-cycle and pipelined datapaths in Verilog HDL
- To synthesize the results by using Xilinx synthesis tools

Introduction

Aiming at enhancing our understanding of how the central processing unit (CPU) works, this project invites us to build a MIPS single-cycle processor and a MIPS pipelined processor using Verilog.

The single-cycle implementation executes all instructions in one clock cycle. This means that no datapath resource can be used more than once per instruction, so any element needed more than once must be duplicated. Therefore, a memory for instructions is separated from one for data. Although some of the functional units will need to be duplicated, many of the elements can be shared by different instruction flows. In order to share a datapath element between two or more different instruction classes, multiple connections to the input of an element are allowed, and multiplexors and control signals are designed carefully to select among the multiple inputs.

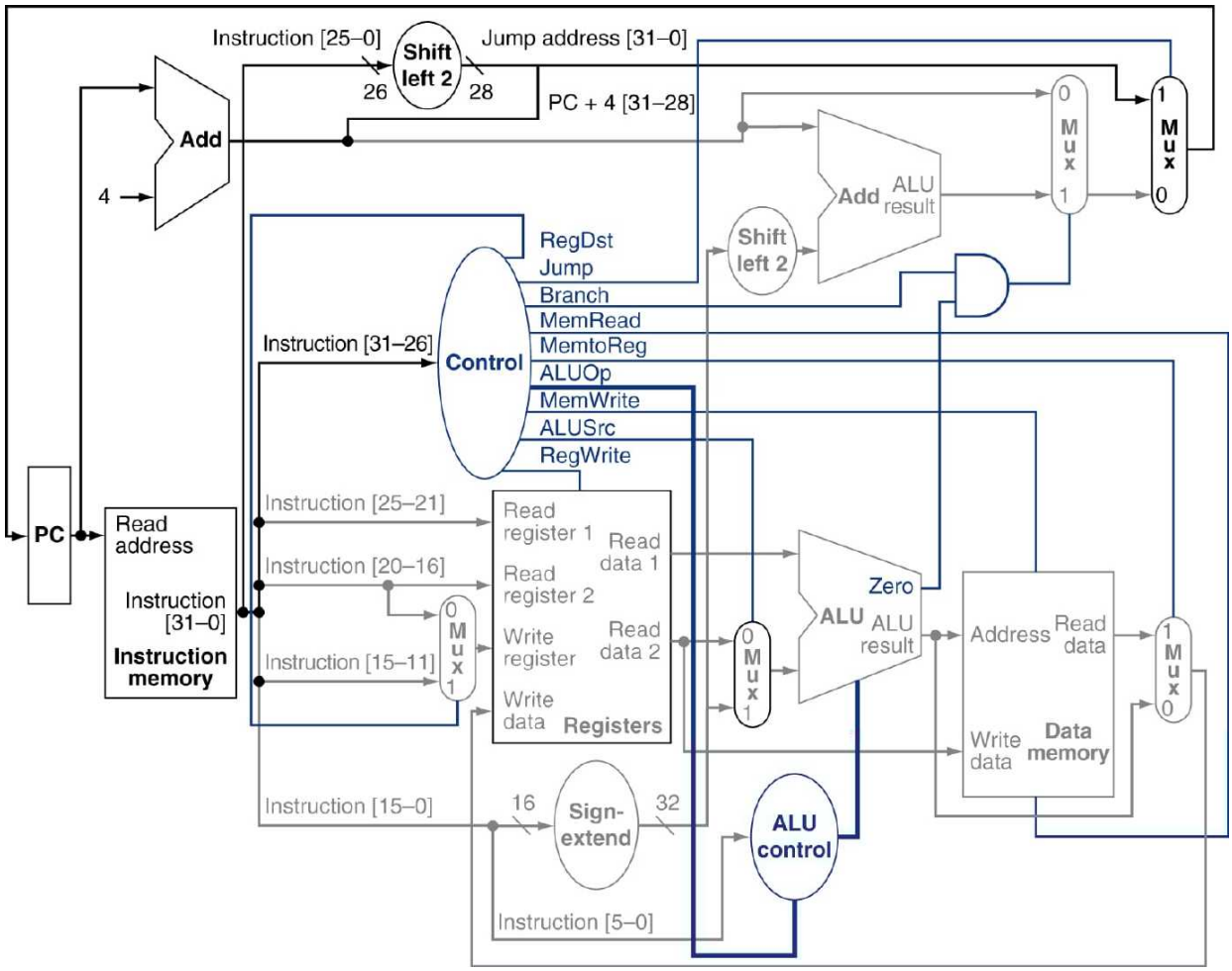
However, the single-cycle implementation has an unsatisfactory performance due to its inefficiency; the execution time of each instruction is one clock cycle, which is determined by the longest possible path in the processor. Using the same hardware components, the pipelined implementation allows different functional units of a system to run concurrently. As a result, pipelining technique significantly reduces the execution time of a same program compared to the single-cycle implementation.

In order to prevent data hazards in the pipeline and minimize the delay created by stalls, a forwarding unit and a hazard detection unit are implemented. The forwarding technique retrieves the missing data element from internal buffers rather than waiting for it to arrive from programmer-visible registers or memory. The hazard detection unit operates during the ID stage so that it can insert the stall between the load and its use; in this case, stalling is inevitable. For control hazards that might occur during the operation of jumping instructions, we assume branch is not taken and stall if the assumption is not correct during the ID stage via the hazard detection unit. For the case when there is a load before a branch, unfortunately, as stated before, stalls are added so that the word is successfully written in the instruction memory.

II. Circuit Design

i. Single-cycle datapath

Our project follows the following single-cycle processor schematic from the textbook. This version of the MIPS single-cycle processor can execute the following instructions: add, sub, and, or, slt, lw, sw, beq, bne, addi, andi and j.

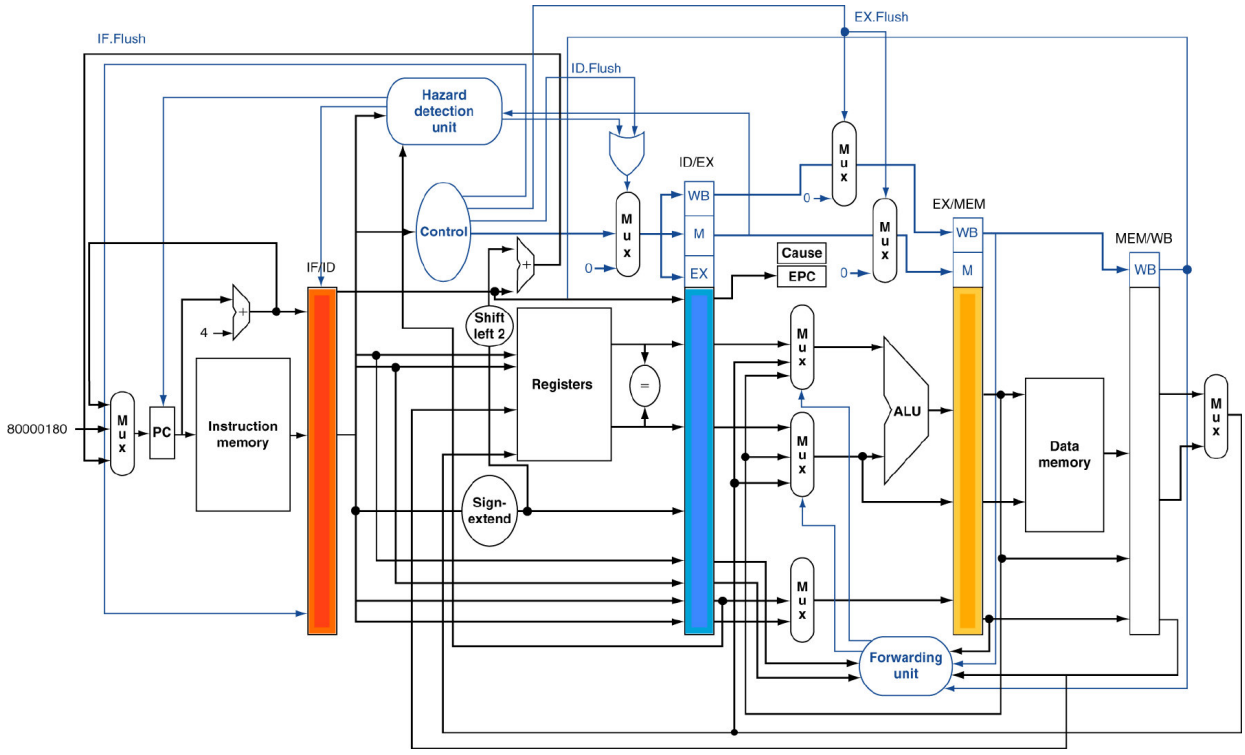


The model divides the machine into two main units: control and data paths. Each unit consists of various functional blocks, such as a 32-bit ALU, register file, sign extension logic, and five multiplexers to select the appropriate operand.

For the ALU control unit and the 32-bit ALU, our design follows six combinations of four control inputs in the textbook. The figure below shows how to set the ALU control bits according to the different function codes of the ALUOp control bit and the R type command.

ii. Pipelined datapath

Our project follows the following pipelined processor schematic from the textbook. This version of the MIPS single-cycle processor can execute the following instructions: add, sub, and, or, slt, lw, sw, beq, bne, addi, andi and j.



III. Design of Components

i. IF Stage

1. PC MUX

The PC multiplexor controls what value replaces the PC (PC+4, the branch destination address or the jump destination address). The Verilog code for the PC multiplexor is

```
module Mux_N_bit(in1,in2,out,select);
    parameter N = 32;
    input[N-1:0] in1,in2;
    input select;
    output[N-1:0] out;
    assign out = select?in2:in1;
endmodule
```

2. PC Register

The program counter is a 32-bit register that is written at the end of every clock cycle and thus does not need a write control signal. The Verilog code for PC register is

```
module PC (
    input               clk,
                        PCWrite,
    input       [31:0]  in,
    output reg  [31:0]  out
```

```
    );

        initial begin
            out = 32'b0;
        end

        always @ (posedge clk) begin
            if (PCWrite)
                out <= in;
        end

    endmodule
```

3. PC Adder 4

The PC adder is an ALU wired to always add its two 32-bit inputs and place the sum on its output. The Verilog code for the adder is

```
module adder(
        input [31:0] a,
        input [31:0] b,
        output [31:0] sum
        );
        reg [31:0] sum;
        always @(a or b)
            begin
                sum = a + b;
            end
    endmodule
```

4. Instruction Memory

The instruction memory only reads, we treat it as a combinational logic: the output at any time reflects the contents of the location specified by the address input, and no read control signal is needed. The Verilog code for the instruction memory is

```
module instruction_memory (
        input       [31:0]  address,
        output      [31:0]  instruction
    );

        parameter size = 128; // you can change here, size is the max size of memory
        integer i;
        // initialize memory
        reg [31:0] memory [0:size-1];
        // clear all memory to nop
        initial begin
            for (i = 0; i < size; i = i + 1)
                memory[i] = 32'b0;
            // include the instruction_memory
```

```
        `include "InstructionMem_for_P2_Demo.txt"
    end
    // Output the menmory at address
    assign instruction = memory[address >> 2];

endmodule
```

ii. EX Stage

1. Forwarding Unit and MUX

The forwarding unit detects the hazards in the EX and MEM stage and assigns the ALU forwarding control of the multiplexors. According to the conditions for the two kinds of data hazards, the Verilog code for the forwarding unit module is written as follow.

```
module Forward (
    input       [4:0]   registerRsID,
                        registerRtID,
                        registerRsEX,
                        registerRtEX,
                        registerRdMEM,
                        registerRdWB,
    input               regWriteMEM,
                        regWriteWB,
    output reg  [1:0]   forwardA,
                        forwardB,
    output reg          forwardC,
                        forwardD
);

    initial begin
        forwardA = 2'b00;
        forwardB = 2'b00;
        forwardC = 1'b0;
        forwardD = 1'b0;
    end

    always @ ( * ) begin
        if (regWriteMEM && registerRdMEM && registerRdMEM == registerRsEX)
            forwardA = 2'b10;
        else if (regWriteWB && registerRdWB && registerRdWB == registerRsEX)
            forwardA = 2'b01;
        else
            forwardA = 2'b00;

        if (regWriteMEM && registerRdMEM && registerRdMEM == registerRtEX)
            forwardB = 2'b10;
        else if (regWriteWB && registerRdWB && registerRdWB == registerRtEX)
```

```
            forwardB = 2'b01;
        else
            forwardB = 2'b00;

        if (regWriteMEM && registerRdMEM && registerRdMEM == registerRsID)
            forwardC = 1'b1;
        else
            forwardC = 1'b0;

        if (regWriteMEM && registerRdMEM && registerRdMEM == registerRtID)
            forwardD = 1'b1;
        else
            forwardD = 1'b0;
    end

endmodule // Forward
```

As the pipeline registers hold the data to be forwarding, we take inputs to the ALU from any pipeline register rather than jus ID/EX, so that the proper data can be forwarded to the next stage. By adding multiplexors to the input of the ALU, and with the proper controls determined by the forwarding unit showed in the following figure, the pipeline can run at full speed in the presence of the data dependences.

Mux control	Source	Explanation
ForwardA = 00	ID/EX	The first ALU operand comes from the register file.
ForwardA = 10	EX/MEM	The first ALU operand is forwarded from the prior ALU result.
ForwardA = 01	MEM/WB	The first ALU operand is forwarded from data memory or an earlier ALU result.
ForwardB = 00	ID/EX	The second ALU operand comes from the register file.
ForwardB = 10	EX/MEM	The second ALU operand is forwarded from the prior ALU result.
ForwardB = 01	MEM/WB	The second ALU operand is forwarded from data memory or an earlier ALU result.

The Verilog code for each 3-to-1 Mux is

```
module Mux_32bit_3to1 (in00, in01, in10, mux_out, control);
    input [31:0] in00, in01, in10;
    output [31:0] mux_out;
    input [1:0] control;
    reg [31:0] mux_out;
    always @(in00 or in01 or in10 or control)
    begin
        case(control)
        2'b00:mux_out<=in00;
        2'b01:mux_out<=in01;
        2'b10:mux_out<=in10;
        default: mux_out<=in00;
        endcase
    end
```

```
endmodule
```

2. ALU Control

The ALU control unit generates a 4-bit control input to the ALU with the function field of the instruction and a 2-bit control field ALUOp determined by the main control unit. Our design follows the that in the textbook that shows how the ALU control inputs are set based on the 2-bit ALUOp control and the 6-bit function code.

Instruction opcode	ALUOp	Instruction operation	Funct field	Desired ALU action	ALU control input
LW	00	load word	XXXXXX	add	0010
SW	00	store word	XXXXXX	add	0010
Branch equal	01	branch equal	XXXXXX	subtract	0110
R-type	10	add	100000	add	0010
R-type	10	subtract	100010	subtract	0110
R-type	10	AND	100100	AND	0000
R-type	10	OR	100101	OR	0001
R-type	10	set on less than	101010	set on less than	0111

The Verilog code for the ALU control module is

```
module ALU_control(
      funct,ALUOp,ALUCtrl
      );
      input [5:0] funct;
      input [1:0] ALUOp;
      output [3:0] ALUCtrl;
      reg [3:0] ALUCtrl;
      always @ (funct or ALUOp)
      begin
          case(ALUOp)
              2'b00: assign ALUCtrl = 4'b0010;
              2'b01: assign ALUCtrl = 4'b0110;
              2'b10:
                  begin
                      if (funct == 6'b100000)      assign ALUCtrl = 4'b0010;
                      else if (funct == 6'b100010) assign ALUCtrl = 4'b0110;
                      else if (funct == 6'b100100) assign ALUCtrl = 4'b0000;
                      else if (funct == 6'b100101) assign ALUCtrl = 4'b0001;
                      else if (funct == 6'b101010) assign ALUCtrl = 4'b0111;
                  end
              2'b11:
                  assign ALUCtrl = 4'b0000;
              // default: assign ALUCtrl = 4'b1111;
          endcase
      end
  endmodule
```

3. ALU

Depending on the ALU control lines, the ALU performs one of the functions shown in the following table.

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

For load word and store word instructions, ALU computes the memory address by addition. For the R-type instructions, ALU performs one of the five actions: and, or add, subtract, or set on less than. For the instruction beq, the ALU performs subtraction. The Verilog code for the ALU is

```
1  `ifndef MODULE_ALU
2  `define MODULE_ALU
3  `timescale 1ns / 1ps
4  module ALU(
5      ALUCtrl,a,b,zero,ALU_result
6      );
7      input [3:0] ALUCtrl;
8      input [31:0] a, b;
9      output zero;
10     output [31:0] ALU_result;
11     reg zero;
12     reg [31:0] ALU_result;
13     always @ (a or b or ALUCtrl)
14     begin
15         case (ALUCtrl)
16             4'b0000:
17             begin
18                 assign ALU_result = a & b;
19                 assign zero = (a & b == 0) ? 1:0;
20             end
21             4'b0001:
22             begin
23                 assign ALU_result = a | b;
24                 assign zero = (a | b == 0) ? 1:0;
25             end
26             4'b0010:
27             begin
28                 assign ALU_result = a + b;
```

```
                assign zero = (a + b == 0) ? 1:0;
            end
            4'b0110:
            begin
                assign ALU_result = a - b;
                assign zero = ( a == b) ? 1:0;
            end
            4'b0111:
            begin
                assign ALU_result = (a < b) ? 1:0;
                assign zero = (a < b) ? 0:1;
            end
            default:
            begin
                assign ALU_result = a;
                assign zero = (a == 0) ? 1:0;
            end
        endcase
    end
endmodule
`endif
```

iii. Memory Stage and Write Back Stage

1. Data Memory

This part is used to save the data to a data memory. The data memory must be written on store instructions; hence, data memory has read and write control signals, an address input, and an input for the data to be written into memory.

```
module data_memory (
    input           clk,
    input           MemRead,
                    MemWrite,
    input    [31:0] address,
                    write_data,
    output   [31:0] read_data
);
    parameter       size = 64; // size of data register_memory
    integer         i;
    wire     [31:0] index;
    reg      [31:0] register_memory [0:size-1];
    assign index = address >> 2; // address/4
    initial begin
        for (i = 0; i < size; i = i + 1)
            register_memory[i] = 32'b0;
        // read_data = 32'b0;
        // wire can not be set within a initial.
    end
```

```
    always @ ( posedge clk ) begin
        if (MemWrite == 1'b1) begin
            register_memory[index] = write_data;
        end
    end
    assign read_data = (MemRead == 1'b1)?register_memory[index]:32'b0;
endmodule
```

VI. Control and Data Hazard

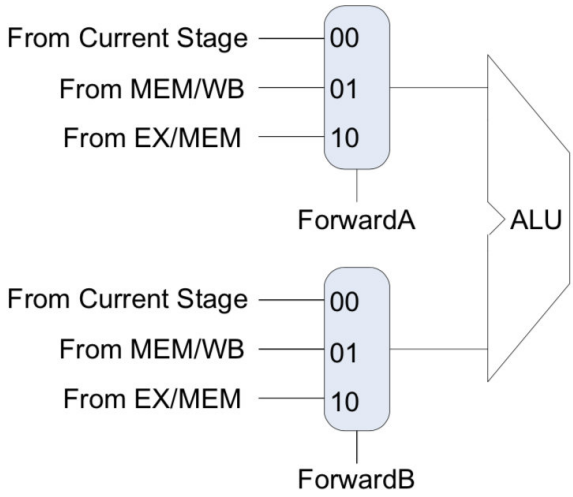
i. EXstage

Data Hazard

Branch prediction and forwarding help make a computer fast while still getting the right answers. This forwarding control will be in the EX stage, because the ALU forwarding multiplexors are found in that stage. Thus, we must pass the operand register numbers from the ID stage via the ID/EX pipeline register to determine whether to forward values.

```
if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0)
and (MEM/WB.RegisterRd = ID/EX.RegisterRs)
and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd =
ID/EX.RegisterRs)) )
ForwardA = 01

if (MEM/WB.RegWrite and (MEM/WB.RegisterRd ≠ 0) and (MEM/WB.RegisterRd = ID/EX.RegisterRt)
and not (EX/MEM.RegWrite and (EX/MEM.RegisterRd ≠ 0) and (EX/MEM.RegisterRd =
ID/EX.RegisterRt)) )
ForwardB = 01
```

According to the figure belowe and the logic above, we can get the verilog program belowe.



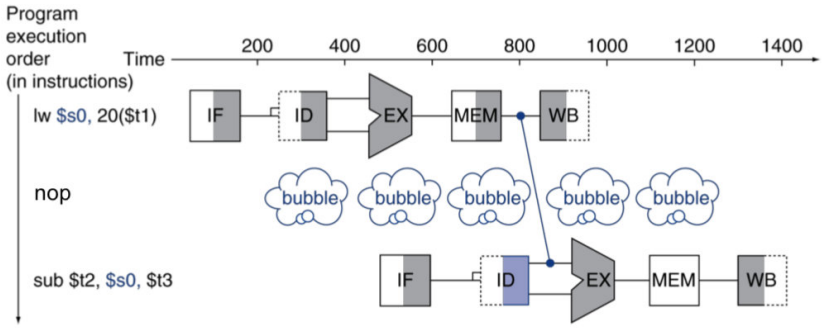

```
module Forwarding(
    // Input Register
    input   [4:0]   ID_EX_Reg_Rt,
                    ID_EX_Reg_Rs,
```

```
                    IF_ID_Reg_Rs,
                    IF_ID_Reg_Rt,
                    EX_MEM_Reg_Rd,
                    MEM_WB_Reg_Rd,
    // Input control signal
    input           EX_MEM_RegWrite,
                    MEM_WB_RegWrite,
    //  Forward for ALU input
    output  reg [1:0]   Forward_ALU_A,
                        Forward_ALU_B,
    //  Forward for state reg input
    output  reg     Forward_C,
                    Forward_D
);
    always @(*)
        begin
            if(EX_MEM_RegWrite && (EX_MEM_Reg_Rd != 0) && (EX_MEM_Reg_Rd == ID_EX_Reg_Rs))
                Forward_ALU_A = 2'b10;
            // ID_EX for ALU
            else if(MEM_WB_RegWrite && (MEM_WB_Reg_Rd != 0) && (MEM_WB_Reg_Rd == ID_EX_Reg_Rs))
                Forward_ALU_A = 2'b01;
            // MEM_WB for ALU
            else
                Forward_ALU_A = 2'b00;
            // Regsiter for ALU

            if(EX_MEM_RegWrite && (EX_MEM_Reg_Rd != 0) && (EX_MEM_Reg_Rd == ID_EX_Reg_Rt))
                Forward_ALU_B = 2'b10;
            else if(MEM_WB_RegWrite && (MEM_WB_Reg_Rd != 0) && (MEM_WB_Reg_Rd == ID_EX_Reg_Rt))
                Forward_ALU_B = 2'b01;
            else
                Forward_ALU_B = 2'b00;
            //Select the data input of ID_EX_State_reg
            if(EX_MEM_RegWrite && (MEM_WB_Reg_Rd != 0) && (MEM_WB_Reg_Rd == IF_ID_Reg_Rs))
                Forward_C = 1;
            else
                Forward_C = 0;

            if(EX_MEM_RegWrite && (MEM_WB_Reg_Rd != 0) && (MEM_WB_Reg_Rd == IF_ID_Reg_Rt))
                Forward_D = 1;
            else
                Forward_D = 0;
        end
endmodule
```

ii. ID stage

Data Hazard

We use the `Hazard Detection Unit` to detect the *Load-Use Hazard* and stall the pipline by one clock cycle.




```
ID/EX.MemRead and
((ID/EX.RegisterRt == IF/ID.RegisterRs) or
(ID/EX.RegisterRt == IF/ID.RegisterRt))
```

With the logic shown above, we can get the verilog program belowe.

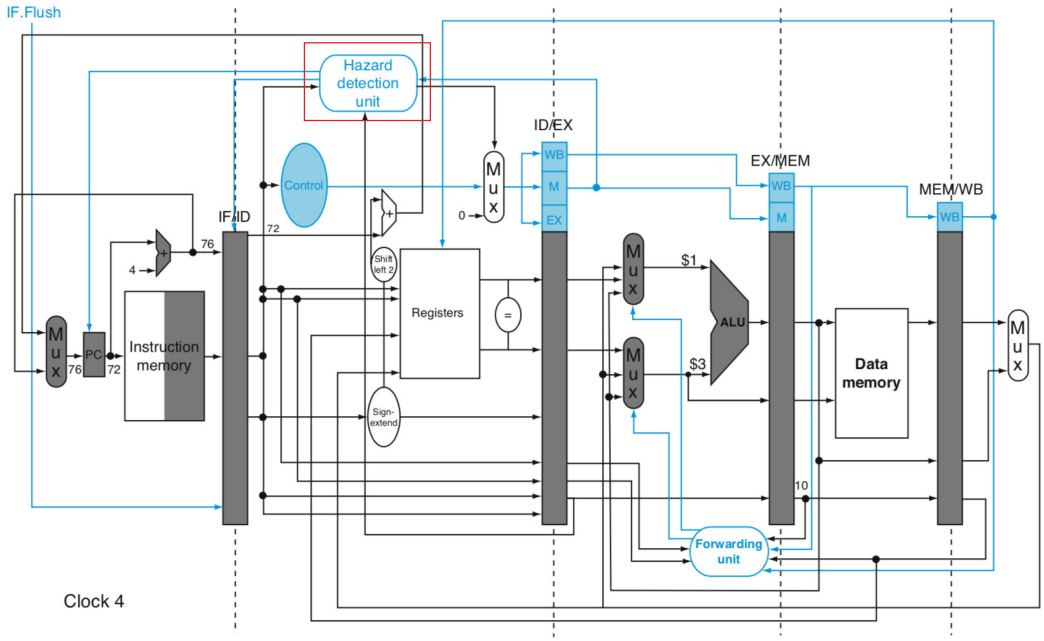
```
module hazard_detection(
    input [4:0] ID_Reg_Rt,
                IF_Reg_Rs,
                IF_Reg_Rt,

    input       EX_MemRead,
                EX_regWirte,
                MEM_MemRead,
                Ins_Beq,
                Ins_Bne,

    output  reg     stall,
                    flush
);
    initial begin
        stall = 1'b0;
        flush = 1'b0;
    end
    always @(*) begin
        if(EX_MemRead&& ID_Reg_Rt&& ((ID_Reg_Rt == IF_Reg_Rs) || (ID_Reg_Rt == IF_Reg_Rt)))
        begin
            stall = 1'b1;
            flush = 1'b1;
        end else if(Ins_Beq || Ins_Bne) begin
            if(EX_regWirte && ID_Reg_Rt&&((ID_Reg_Rt == IF_Reg_Rs) || (ID_Reg_Rt == IF_Reg_Rt))) begin
                stall = 1'b1;
                flush = 1'b1;
```

```
            end else if (MEM_MemRead &&ID_Reg_Rt&& ((ID_Reg_Rt == IF_Reg_Rs) || (ID_Reg_Rt
== IF_Reg_Rt))) begin
                stall = 1'b1;
                flush = 1'b1;
            end else begin
                stall = 1'b1;
                flush = 1'b1;
            end
        end else begin
            stall = 1'b0;
            flush = 1'b0;
        end
    end
endmodule
```

Control Hazard

We have limited our concern to hazards involving arithmetic operations and data transfers. However, there are also pipeline hazards involving branches. An instruction must be fetched at every clock cycle to sustain the pipeline, yet in our design the decision about whether to branch doesn't occur until the MEM pipeline stage. This delay in determining the proper instruction to fetch is called a control hazard or branch hazard, in contrast to the data hazards we have just examined.

We *assume branch not taken* and use dynamic branch prediction. We only change prediction on two successive mispredictions.




```
module HazardDetection (
    input               branchEqID,
                        branchNeID,
                        memReadEX,
                        regWriteEX,
                        memReadMEM,
    input       [4:0]   registerRsID,
                        registerRtID,
```

```
                        registerRtEX,
                        registerRdEX,
                        registerRdMEM,
    output reg          stall,
                        flush
);

    initial begin
        stall = 1'b0;
        flush = 1'b0;
    end

    always @ ( * ) begin
        if (memReadEX && registerRtEX && (registerRtEX == registerRsID || registerRtEX == registerRtID)) begin
            stall = 1'b1;
            flush = 1'b1;
        end else if (branchEqID || branchNeID) begin
            if (regWriteEX && registerRdEX && (registerRdEX == registerRsID || registerRdEX == registerRtID)) begin
                stall = 1'b1;
                flush = 1'b1;
            end else if (memReadMEM && registerRdMEM && (registerRdMEM == registerRsID || registerRdMEM == registerRtID)) begin
                stall = 1'b1;
                flush = 1'b1;
            end else begin
                stall = 1'b0;
                flush = 1'b0;
            end
        end else begin
            stall = 1'b0;
            flush = 1'b0;
        end
    end

endmodule
```

VII. Instruction Implementation

i. Data tables

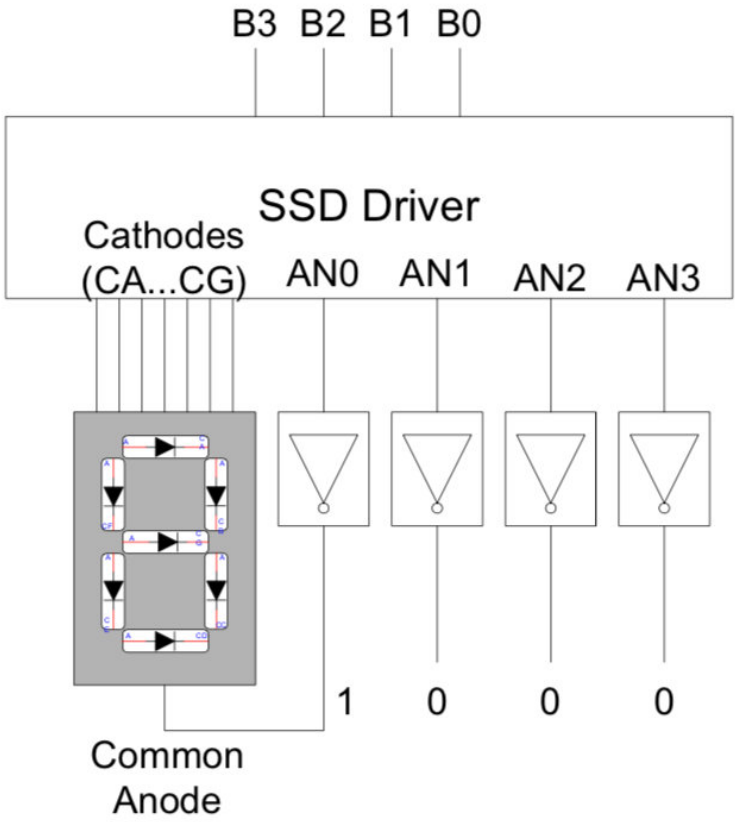
opcode	Operation	ALUOp	funct	ALU Control	ALU function
lw	load word	00	XXXXXX	0010	add
sw	store word				
beq	branch equal	01	XXXXXX	0110	subtract
R-type	add	10	100000	0010	add
	subtract		100010	0110	subtract
	AND		100100	0000	AND
	OR		100101	0001	OR
	set-on-less-than		101010	0111	set-on-less-than

The following table shows the setting of the control lines for each instruction in our design.

	Jump	ALUSrc	RegDst	ALUOp	MemWrite	MemRead	Branch	MemtoReg	RegWrite
R-type	0	0	1	10	0	0	0	0	1
addi	0	1	0	00	0	0	0	0	1
andi	0	1	0	00	0	0	0	0	1
slt	0	1	0	00	0	0	0	0	1
beq	0	0	0	01	0	0	1	0	0
bne	0	0	0	01	0	0	1	0	0
lw	0	1	0	00	0	1	0	1	1
sw	0	1	0	00	1	0	0	0	0
j	1	0	0	00	0	0	0	0	0

VIII. SSD and Top Module

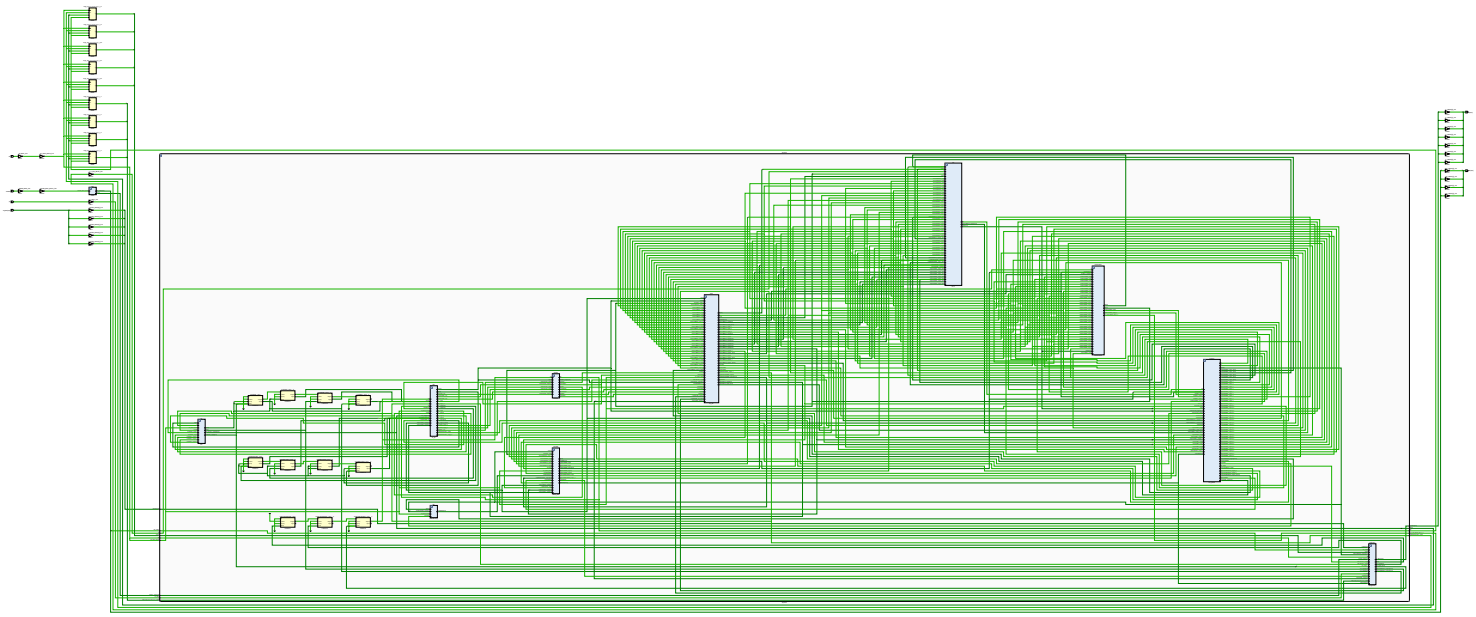
The Basys 3 board contains a four-digit common anode SSD. The anodes of the seven LEDs forming each digit are tied together into one “common anode” circuit node, but the LED cathodes remain separate. The common anode signals are available as four “digit enable” input signals to the 4-digit display. The cathodes of similar segments on all four displays are connected into seven circuit nodes labeled CA through CG (so, for example, the four d cathodes from the four digits are grouped together into a single circuit node called “CD”). These seven cathode signals are available as inputs to the 4-digit display. This signal connection scheme makes the cathode signals common to all digits but they can only illuminate the segments of a digit whose corresponding anode signal is asserted.



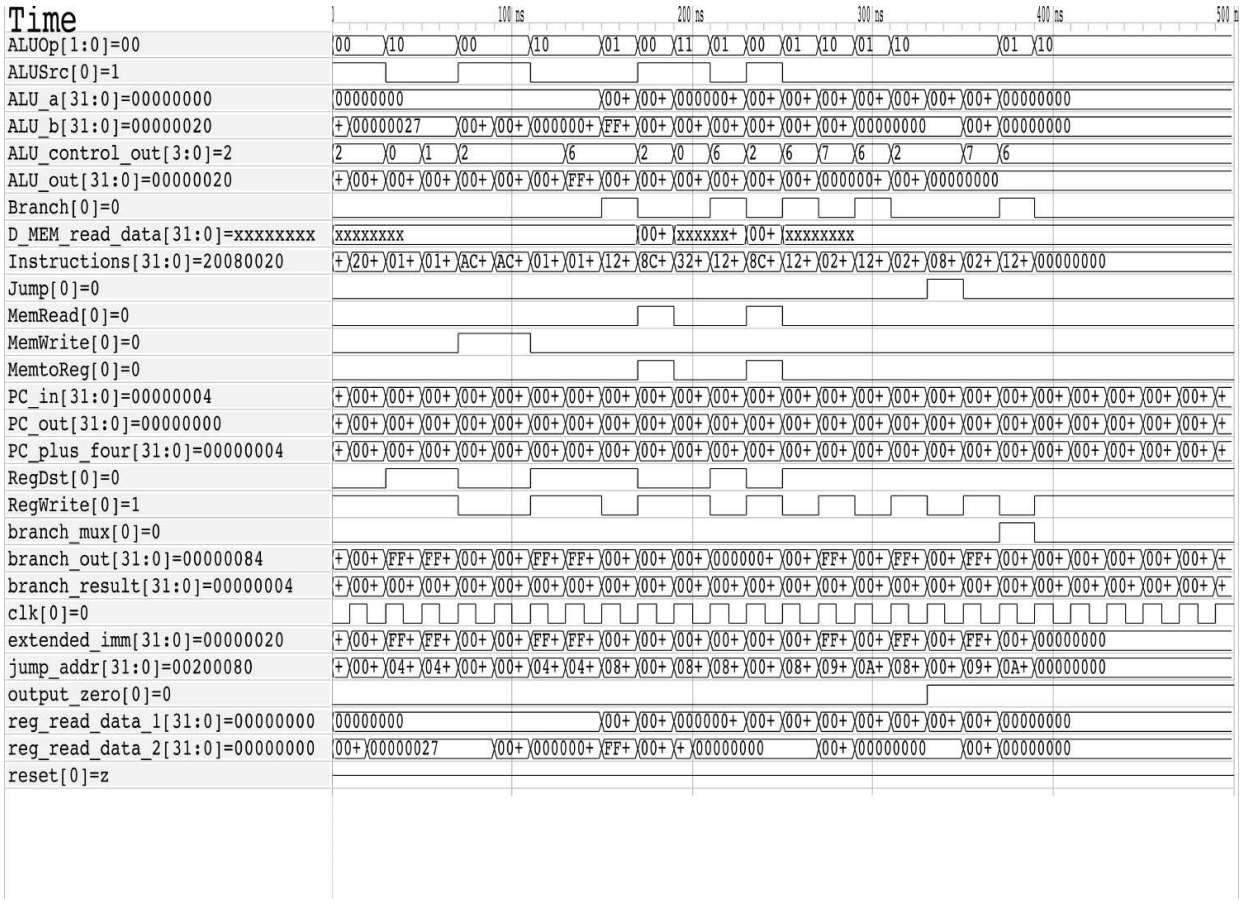

```
module SSD(Q,out);
    input [3:0] Q;
    output [6:0] out;
    reg [6:0] out;
    always @(Q) begin
        case(Q)
        4'b0000: out = 7'b0000001;
        4'b0001: out = 7'b1001111;
        4'b0010: out = 7'b0010010;
        4'b0011: out = 7'b0000110;
        4'b0100: out = 7'b1001100;
        4'b0101: out = 7'b0100100;
        4'b0110: out = 7'b0100000;
        4'b0111: out = 7'b0001111;
        4'b1000: out = 7'b0000000;
        4'b1001: out = 7'b0000100;
        default: out = 7'b1111110;
    endcase
end
endmodule
```

IX. RTL Schematic (Optional)

Please check the RTL schematic on the next two pages.



X. Textual Result



By running the instruction memory, we can achieve appendix part.

XI. Conclusion and Discussion

In this project, a MIPS single-cycle processor and a MIPS pipelined processor are modeled, implemented using Verilog and simulated in Vivado. The pipelined processor program is run on a Xilinx FPGA board, which is portal to the Vivado Design Suite. The results on board coincide with the simulation results.

Throughout the implementation process, some unexpected events occurred and time were spent on debugging the modules. The events can be divided into two categories. The following discusses the reasons that led to the events.

1. Simulation errors in Verilog HDL
    - For the implementation of the ‘and’ instruction in the ALU module, logical AND ‘&&’ should be used rather than bitwise AND ‘&’. Using bitwise AND will lead to problematic results. They are two different logical operations.
    - In the single cycle processor, due to the rising edge in the first clock cycle, if the first instruction is ‘lw’, error would occur. Therefore, the program counter starts from ‘-4’ instead of ‘0’.
    - When coding the main processor module, where a large number of wires are connected to run the program, the naming of wires was not paid attention to. This led to error in wire

connections. The reason is that Verilog is a case-sensitive language. For example, 'WireReg' and 'wireReg' actually refer to different wires, resulting in the error.
    - The naming of the pipeline register wasn't consistent for different components, some represented the output of a pipeline and some were the input of a pipeline. It led to some errors when simulating the processor too.
    - Besides, during the debug process, we learned that the 'assign' statement used for continuous assignment in Verilog can only drive a value to a net but it cannot assign a value to a register.
2. Unexpected results on FPGA board
    - Because the events in Verilog mostly happen at 'posedge Clock' or 'negedge Clock', when the switch spends more than half a clock cycle time in between the 'on' and 'off' position, the program counter will not be determined and hence leads to varying random numbers.

By accomplishing the project, our understanding of how the central processing unit (CPU) works is deepened through practice. The mechanisms of forwarding and flushing in order to prevent hazards are carefully comprehended during the design process. While theoretical knowledge is the foundation, as programmers, the use of programming language has a series of rules and conventions to follow. Especially for naming, the habit of consistency is important and will save a lot of time in the debug process, no matter what programming project we work on in the future.

X. Reference

Hennessy, & JohnL. (1998). Computer organization and design:the hardware/software interface. Morgan Kaufmann Publishers.

Patterson, D. A., & Hennessy, J. L. (2014). Computer organization and design, fourth edition. Tailieu Vn.

XII. Appendix

1. TextualResult

Single

```
LXT2 info: dumpfile single_cycle.vcd opened for output.
Texual result of single cycle:
=============================================================
Time:           0, CLK = 0, PC = 0xfffffffc
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
```

```
Time:           0, CLK = 1, PC = 0x00000000
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------------
Time:           1, CLK = 0, PC = 0x00000000
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------------
Time:           1, CLK = 1, PC = 0x00000004
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------------
Time:           2, CLK = 0, PC = 0x00000004
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------------
Time:           2, CLK = 1, PC = 0x00000008
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------------
Time:           3, CLK = 0, PC = 0x00000008
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------------
Time:           3, CLK = 1, PC = 0x0000000c
```

```
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------
Time:           4, CLK = 0, PC = 0x0000000c
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------
Time:           4, CLK = 1, PC = 0x00000010
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------
Time:           5, CLK = 0, PC = 0x00000010
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------
Time:           5, CLK = 1, PC = 0x00000014
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------
Time:           6, CLK = 0, PC = 0x00000014
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
--------------------------------------------------------
Time:           6, CLK = 1, PC = 0x00000018
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
```

```
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:           7, CLK = 0, PC = 0x00000018
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:           7, CLK = 1, PC = 0x0000001c
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:           8, CLK = 0, PC = 0x0000001c
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:           8, CLK = 1, PC = 0x00000020
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:           9, CLK = 0, PC = 0x00000020
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:           9, CLK = 1, PC = 0x00000024
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
```

```
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          10, CLK = 0, PC = 0x00000024
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          10, CLK = 1, PC = 0x00000028
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          11, CLK = 0, PC = 0x00000028
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          11, CLK = 1, PC = 0x0000002c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          12, CLK = 0, PC = 0x0000002c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          12, CLK = 1, PC = 0x00000030
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
```

```
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:           13, CLK = 0, PC = 0x00000030
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:           13, CLK = 1, PC = 0x00000034
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:           14, CLK = 0, PC = 0x00000034
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:           14, CLK = 1, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:           15, CLK = 0, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:           15, CLK = 1, PC = 0x0000003c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
```

```
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:          16, CLK = 0, PC = 0x0000003c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:          16, CLK = 1, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:          17, CLK = 0, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:          17, CLK = 1, PC = 0x00000044
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:          18, CLK = 0, PC = 0x00000044
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
----------------------------------------------------------
Time:          18, CLK = 1, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
```

```
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          19, CLK = 0, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          19, CLK = 1, PC = 0x0000003c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          20, CLK = 0, PC = 0x0000003c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          20, CLK = 1, PC = 0x0000007c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          21, CLK = 0, PC = 0x0000007c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
-------------------------------------------------------------
Time:          21, CLK = 1, PC = 0x00000080
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
------------------------------------------------------------
Time:          22, CLK = 0, PC = 0x00000080
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
------------------------------------------------------------
Time:          22, CLK = 1, PC = 0x00000084
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
------------------------------------------------------------
Time:          23, CLK = 0, PC = 0x00000084
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
------------------------------------------------------------
Time:          23, CLK = 1, PC = 0x00000088
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
------------------------------------------------------------
Time:          24, CLK = 0, PC = 0x00000088
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
------------------------------------------------------------
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 500000 ticks.
> finish
** Continue **
```

Pipline

```
VCD info: dumpfile pipeline.vcd opened for output.
Texual result of pipeline:
====================================================
Time:           0, CLK = 0, PC = 0x00000000
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           0, CLK = 1, PC = 0x00000004
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           1, CLK = 0, PC = 0x00000004
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           1, CLK = 1, PC = 0x00000008
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           2, CLK = 0, PC = 0x00000008
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           2, CLK = 1, PC = 0x0000000c
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
```

```
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           3, CLK = 0, PC = 0x0000000c
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           3, CLK = 1, PC = 0x00000010
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           4, CLK = 0, PC = 0x00000010
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           4, CLK = 1, PC = 0x00000014
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           5, CLK = 0, PC = 0x00000014
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           5, CLK = 1, PC = 0x00000018
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
================================================
Time:           6, CLK = 0, PC = 0x00000018
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           6, CLK = 1, PC = 0x0000001c
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           7, CLK = 0, PC = 0x0000001c
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           7, CLK = 1, PC = 0x00000020
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           8, CLK = 0, PC = 0x00000020
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           8, CLK = 1, PC = 0x00000024
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
```

```
Time:           9, CLK = 0, PC = 0x00000024
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           9, CLK = 1, PC = 0x00000024
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          10, CLK = 0, PC = 0x00000024
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          10, CLK = 1, PC = 0x00000028
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          11, CLK = 0, PC = 0x00000028
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xffffffe9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          11, CLK = 1, PC = 0x0000002c
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xffffffe9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          12, CLK = 0, PC = 0x0000002c
```

```
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffffe9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           12, CLK = 1, PC = 0x0000002c
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffffe9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           13, CLK = 0, PC = 0x0000002c
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffffe9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           13, CLK = 1, PC = 0x00000030
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffffe9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           14, CLK = 0, PC = 0x00000030
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0xfffffffe9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           14, CLK = 1, PC = 0x00000030
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0xfffffffe9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           15, CLK = 0, PC = 0x00000030
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0xfffffffe9
```

```
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           15, CLK = 1, PC = 0x00000034
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0xfffffffe9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           16, CLK = 0, PC = 0x00000034
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           16, CLK = 1, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           17, CLK = 0, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           17, CLK = 1, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           18, CLK = 0, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
```

```
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          18, CLK = 1, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          19, CLK = 0, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          19, CLK = 1, PC = 0x0000003c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          20, CLK = 0, PC = 0x0000003c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          20, CLK = 1, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          21, CLK = 0, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
```

```
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           21, CLK = 1, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           22, CLK = 0, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           22, CLK = 1, PC = 0x00000044
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           23, CLK = 0, PC = 0x00000044
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           23, CLK = 1, PC = 0x00000048
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           24, CLK = 0, PC = 0x00000048
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
```

```
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          24, CLK = 1, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          25, CLK = 0, PC = 0x00000038
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          25, CLK = 1, PC = 0x0000003c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          26, CLK = 0, PC = 0x0000003c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          26, CLK = 1, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          27, CLK = 0, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
```

```
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          27, CLK = 1, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          28, CLK = 0, PC = 0x00000040
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          28, CLK = 1, PC = 0x0000007c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          29, CLK = 0, PC = 0x0000007c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          29, CLK = 1, PC = 0x00000080
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          30, CLK = 0, PC = 0x00000080
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
```

```
================================================
Time:           30, CLK = 1, PC = 0x00000084
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           31, CLK = 0, PC = 0x00000084
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           31, CLK = 1, PC = 0x00000088
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           32, CLK = 0, PC = 0x00000088
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           32, CLK = 1, PC = 0x0000008c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:           33, CLK = 0, PC = 0x0000008c
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
```

```
Time:          33, CLK = 1, PC = 0x00000090
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
Time:          34, CLK = 0, PC = 0x00000090
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$s5] = 0x00000000
[$s6] = 0x00000000, [$s7] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
[$t4] = 0x00000000, [$t5] = 0x00000000, [$t6] = 0x00000000
[$t7] = 0x00000000, [$t8] = 0x00000000, [$t9] = 0x00000000
================================================
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 700000 ticks.
> finish
** Continue **
```

2. adder.v

```
module adder(
    input [31:0] a,
    input [31:0] b,
    output [31:0] sum
    );
    reg [31:0] sum;
    always @(a or b)
        begin
            sum = a + b;
        end
endmodule
```

3. ALU_control.v

```
module ALU_control(
    funct,ALUOp,ALUCtrl
    );
    input [5:0] funct;
    input [1:0] ALUOp;
    output [3:0] ALUCtrl;
    reg [3:0] ALUCtrl;
```

```
    always @ (funct or ALUOp)
    begin
        case(ALUOp)
            2'b00: assign ALUCtrl = 4'b0010;
            2'b01: assign ALUCtrl = 4'b0110;
            2'b10:
                begin
                    if (funct == 6'b100000)      assign ALUCtrl = 4'b0010;
                    else if (funct == 6'b100010) assign ALUCtrl = 4'b0110;
                    else if (funct == 6'b100100) assign ALUCtrl = 4'b0000;
                    else if (funct == 6'b100101) assign ALUCtrl = 4'b0001;
                    else if (funct == 6'b101010) assign ALUCtrl = 4'b0111;
                end
            2'b11:
                assign ALUCtrl = 4'b0000;
        endcase
    end
endmodule
```

4. ALU.v

```
module ALU(
    ALUCtrl,a,b,zero,ALU_result
    );
    input [3:0] ALUCtrl;
    input [31:0] a, b;
    output zero;
    output [31:0] ALU_result;
    reg zero;
    reg [31:0] ALU_result;
    always @ (a or b or ALUCtrl)
    begin
        case (ALUCtrl)
            4'b0000:
            begin
                assign ALU_result = a & b;
                assign zero = (a & b == 0) ? 1:0;
            end
            4'b0001:
            begin
                assign ALU_result = a | b;
                assign zero = (a | b == 0) ? 1:0;
            end
            4'b0010:
            begin
                assign ALU_result = a + b;
                assign zero = (a + b == 0) ? 1:0;
            end
            4'b0110:
```

```
            begin
                assign ALU_result = a - b;
                assign zero = ( a == b) ? 1:0;
            end
            4'b0111:
            begin
                assign ALU_result = (a < b) ? 1:0;
                assign zero = (a < b) ? 0:1;
            end
            default:
            begin
                assign ALU_result = a;
                assign zero = (a == 0) ? 1:0;
            end
        endcase
    end
endmodule
```

5. data_memory.v

```
module data_memory (
    input            clk,
    input            MemRead,
                     MemWrite,
    input    [31:0]  address,
                     write_data,
    output   [31:0]  read_data
);

    parameter        size = 64; // size of data register_memory
    integer          i;
    wire     [31:0]  index;
    reg      [31:0]  register_memory [0:size-1];

    assign index = address >> 2; // address/4


    initial begin
        for (i = 0; i < size; i = i + 1)
            register_memory[i] = 32'b0;
        // read_data = 32'b0;
        // wire can not be set within a initial.
    end

    always @ ( posedge clk ) begin
        if (MemWrite == 1'b1) begin
            register_memory[index] = write_data;
        end
    end
```

```
30
31      assign read_data = (MemRead == 1'b1)?register_memory[index]:32'b0;
32  endmodule
```

6. forwarding_unit.v

```
1   module Forward (
2       input       [4:0]   registerRsID,
3                           registerRtID,
4                           registerRsEX,
5                           registerRtEX,
6                           registerRdMEM,
7                           registerRdWB,
8       input               regWriteMEM,
9                           regWriteWB,
10      output reg  [1:0]   forwardA,
11                          forwardB,
12      output reg          forwardC,
13                          forwardD
14  );
15
16      initial begin
17          forwardA = 2'b00;
18          forwardB = 2'b00;
19          forwardC = 1'b0;
20          forwardD = 1'b0;
21      end
22
23      always @ ( * ) begin
24          if (regWriteMEM && registerRdMEM && registerRdMEM == registerRsEX)
25              forwardA = 2'b10;
26          else if (regWriteWB && registerRdWB && registerRdWB == registerRsEX)
27              forwardA = 2'b01;
28          else
29              forwardA = 2'b00;
30
31          if (regWriteMEM && registerRdMEM && registerRdMEM == registerRtEX)
32              forwardB = 2'b10;
33          else if (regWriteWB && registerRdWB && registerRdWB == registerRtEX)
34              forwardB = 2'b01;
35          else
36              forwardB = 2'b00;
37
38          if (regWriteMEM && registerRdMEM && registerRdMEM == registerRsID)
39              forwardC = 1'b1;
40          else
41              forwardC = 1'b0;
42
43          if (regWriteMEM && registerRdMEM && registerRdMEM == registerRtID)
```

```
            forwardD = 1'b1;
        else
            forwardD = 1'b0;
    end

endmodule // Forward
```

7. hazard_detection.v

```
module HazardDetection (
    input               branchEqID,
                        branchNeID,
                        memReadEX,
                        regWriteEX,
                        memReadMEM,
    input       [4:0]   registerRsID,
                        registerRtID,
                        registerRtEX,
                        registerRdEX,
                        registerRdMEM,
    output reg          stall,
                        flush
);

    initial begin
        stall = 1'b0;
        flush = 1'b0;
    end

    always @ ( * ) begin
        if (memReadEX && registerRtEX && (registerRtEX == registerRsID || registerRtEX == registerRtID)) begin
            stall = 1'b1;
            flush = 1'b1;
        end else if (branchEqID || branchNeID) begin
            if (regWriteEX && registerRdEX && (registerRdEX == registerRsID || registerRdEX == registerRtID)) begin
                stall = 1'b1;
                flush = 1'b1;
            end else if (memReadMEM && registerRdMEM && (registerRdMEM == registerRsID || registerRdMEM == registerRtID)) begin
                stall = 1'b1;
                flush = 1'b1;
            end else begin
                stall = 1'b0;
                flush = 1'b0;
            end
        end else begin
            stall = 1'b0;
```

```
            flush = 1'b0;
        end
    end

endmodule // HazardDetection
```

8. instruction_memory.v

```
module instruction_memory (
    input       [31:0]  address,
    output      [31:0]  instruction
);

    parameter size = 128; // you can change here, size is the max size of memory
    integer i;
    // initialize memory
    reg [31:0] memory [0:size-1];
    // clear all memory to nop
    initial begin
        for (i = 0; i < size; i = i + 1)
            memory[i] = 32'b0;
        // include the instruction_memory
        `include "InstructionMem_for_P2_Demo.txt"
    end
    // Output the menmory at address
    assign instruction = memory[address >> 2];

endmodule
```

9. Mux_N_bit.v

```
module Mux_N_bit(in1,in2,out,select);
    parameter N = 32;
    input[N-1:0] in1,in2;
    input select;
    output[N-1:0] out;
    assign out = select?in2:in1;
endmodule
```

10. pc.v

```
module PC (
    input               clk,
                        PCWrite,
    input       [31:0]  in,
    output reg  [31:0]  out
);

```

```
    initial begin
        out = 32'b0;
    end

    always @ (posedge clk) begin
        if (PCWrite)
            out <= in;
    end

endmodule
```

11. pipeline.v

```
module Pipeline(
    input clk
);

    wire            reset;
    assign          reset = 1'b0;
    // Wire in IF stage
    wire    [31:0]  PC_in_IF,
                    PC_out_IF,
                    PC_add4_IF,
                    instrustion_IF,
                    branch_result_IF;

    wire            stall,
                    IF_ID_Write,
                    IF_ID_flush;
    // wire in ID stage input of IDEX state regsiter, output of state regsiter start with
EX
    wire    [31:0]  ID_EX_PC_add4,
                    ID_EX_PC_add4_res,
                    ID_EX_instrustion,
                    ID_EX_reg_read_data_1,
                    ID_EX_reg_read_data_2,
                    ID_EX_reg_read_new_data_1,
                    ID_EX_reg_read_new_data_2,
                    ID_EX_sign_extend,
                    ID_EX_jump_addr;

    wire    [4:0]   ID_EX_Reg_Rs,
                    ID_EX_Reg_Rt,
                    ID_EX_Reg_Rd;

    wire    [1:0]   ID_EX_alu_op;
    // Control signals
    wire            ID_EX_regDst,
                    ID_EX_branchEq,
```

```
                ID_EX_branchNeq,
                ID_EX_memRead,
                ID_EX_memtoReg,
                ID_EX_memWrite,
                ID_EX_aluSrc,
                ID_EX_regWrite,
                ID_EX_branch,
                ID_EX_jump,
    //  comparasions
                ID_EX_equal_reg_read_data;

    // Wire in EX/MEM
    wire    [31:0]  EX_MEM_reg_read_data_1,
                    EX_MEM_reg_read_data_2,

                    EX_MEM_sign_extend,
                    // output of mux
                    EX_MEM_alu_in_one,
                    EX_MEM_alu_in_two,
                    EX_MEM_alu_temp_two,
                    EX_MEM_alu_result;

    wire    [4:0]   EX_MEM_reg_Rs,
                    EX_MEM_reg_Rt,
                    EX_MEM_reg_Rd,
                    EX_reg;

    wire    [3:0]   EX_MEM_alu_control;


    wire    [1:0]   EX_MEM_alu_op;
    wire            EX_MEM_regDst,
                    EX_MEM_memRead,
                    EX_MEM_memtoReg,
                    EX_MEM_memWrite,
                    EX_MEM_regWrite,
                    EX_MEM_aluSrc,
                    EX_MEM_aluZero;

    //  wire in MEM/WB
    wire    [31:0]  MEM_WB_alu_result,
                    MEM_WB_reg_read_data_2,
                    MEM_WB_D_MEM_read_data;

    wire    [4:0]   MEM_register;

    wire            MEM_WB_memRead,
                    MEM_WB_memtoReg,
                    MEM_WB_memWrite,
```

```
                MEM_WB_regWrite;

    // wire in WB

    wire    [31:0]  WB_alu_result,
                    WB_D_MEM_read_data,
                    WB_D_MEM_read_addr,
                    WB_write_data;

    wire    [4:0]   WB_register;
    wire            WB_memtoReg,
                    WB_regWrite;
    // Data hazard
    wire    [1:0]   forward_A,
                    forward_B;

    wire            forward_C,
                    forward_D;



    assign PC_add4_IF = PC_out_IF + 4;
    // assign PC_in_IF = PC_add4_IF;
    // IF stage
    assign pc_wirte = ~stall;
    PC program_counter(
        .clk(clk),
        .PCWrite(pc_wirte),
        .in(PC_in_IF),
        .out(PC_out_IF)
    );

    instruction_memory ins_mem(
        .address(PC_out_IF),
        .instruction(instrustion_IF)
    );

    // IF/ID
    wire IF_Flush;
    assign IF_ID_Write = ~stall;
    IF_ID_Reg State_IF_ID(
        .clk(clk),
        .reset(reset),
        .PC_plus4_in(PC_add4_IF),
        .instruction_in(instrustion_IF),
        .IF_ID_Write(IF_ID_Write),
        .IF_Flush(IF_Flush),
        .PC_plus4_out(ID_EX_PC_add4),
        .instruction_out(ID_EX_instrustion)
```

```
    );
    //  ID stage
    assign ID_EX_Reg_Rs = ID_EX_instrustion[25:21];
    assign ID_EX_Reg_Rt = ID_EX_instrustion[20:16];
    assign ID_EX_Reg_Rd = ID_EX_instrustion[15:11];

    wire ins_Beq,ins_Bne;

    Control control(
        .op_code(ID_EX_instrustion[31:26]),
        .ALUOp(ID_EX_alu_op),
        .RegDst(ID_EX_regDst),
        .Jump(ID_EX_jump),
        .Ins_Beq(ins_Beq),
        .Ins_Bne(ins_Bne),
        .MemRead(ID_EX_memRead),
        .MemtoReg(ID_EX_memtoReg),
        .MemWrite(ID_EX_memWrite),
        .ALUSrc(ID_EX_aluSrc),
        .RegWrite(ID_EX_regWrite)
    );
    Registers register(
        .clk(clk),
        .regWrite(WB_regWrite),
        // .write_data(WB_D_MEM_read_data),
        .read_register_1(ID_EX_Reg_Rs),
        .read_register_2(ID_EX_Reg_Rt),
        .write_register(WB_register),
        .write_data(WB_write_data),
        .read_data_1(ID_EX_reg_read_data_1),
        .read_data_2(ID_EX_reg_read_data_2)
    );

    sign_extension sign(
        .shortInput(ID_EX_instrustion[15:0]),
        .longOutput(ID_EX_sign_extend)
    );
    assign ID_EX_reg_read_new_data_1 = (forward_C)?
MEM_WB_alu_result:ID_EX_reg_read_data_1;
    assign ID_EX_reg_read_new_data_2 = (forward_D)?
MEM_WB_alu_result:ID_EX_reg_read_data_2;
    assign ID_EX_PC_add4_res = ID_EX_PC_add4 + (ID_EX_sign_extend<<2);
    assign ID_EX_jump_addr = {ID_EX_PC_add4[31:28],ID_EX_instrustion[25:0],2'b0};
    assign ID_EX_equal_reg_read_data = (ID_EX_reg_read_new_data_1 ==
ID_EX_reg_read_new_data_2);
    assign ID_EX_branch = (ins_Beq && ID_EX_equal_reg_read_data) || (ins_Bne && !
ID_EX_equal_reg_read_data);

    assign branch_result_IF = (ID_EX_branch == 1'b0)?PC_add4_IF:ID_EX_PC_add4_res;
```

```
179     assign PC_in_IF = (ID_EX_jump == 1'b0)?branch_result_IF:ID_EX_jump_addr;
180     assign IF_Flush = (ID_EX_jump||ID_EX_branch);
181     wire ID_EX_flush;
182     ID_EX_Reg idex(
183     //   Input
184         .clk(clk),
185         .reset(reset),
186         .flush(ID_EX_flush),
187         .RegDst(ID_EX_regDst),
188         .MemtoReg(ID_EX_memtoReg),
189         .MemRead(ID_EX_memRead),
190         .MemWrite(ID_EX_memWrite),
191         .ALUSrc(ID_EX_aluSrc),
192         .RegWrite(ID_EX_regWrite),
193         .ALUop(ID_EX_alu_op),
194         .reg_read_data_1(ID_EX_reg_read_data_1),
195         .reg_read_data_2(ID_EX_reg_read_data_2),
196         .extended_imm(ID_EX_sign_extend),
197         .IF_ID_Register_Rs(ID_EX_Reg_Rs),
198         .IF_ID_Register_Rt(ID_EX_Reg_Rt),
199         .IF_ID_Register_Rd(ID_EX_Reg_Rd),
200
201     //  output
202         .out_RegDst(EX_MEM_regDst),
203         .out_MemRead(EX_MEM_memRead),
204         .out_MemtoReg(EX_MEM_memtoReg),
205         .out_MemWrite(EX_MEM_memWrite),
206         .out_ALUSrc(EX_MEM_aluSrc),
207         .out_RegWrite(EX_MEM_regWrite),
208         .reg_read_data_1_out(EX_MEM_reg_read_data_1),
209         .reg_read_data_2_out(EX_MEM_reg_read_data_2),
210         .extended_imm_out(EX_MEM_sign_extend),
211         .IF_ID_Register_Rs_out(EX_MEM_reg_Rs),
212         .IF_ID_Register_Rd_out(EX_MEM_reg_Rd),
213         .IF_ID_Register_Rt_out(EX_MEM_reg_Rt),
214         .ALUop_out(EX_MEM_alu_op)
215     );
216
217     HazardDetection hazard(
218         .branchEqID(ins_Beq),
219         .branchNeID(ins_Bne),
220         .memReadEX(EX_MEM_memRead),
221         .regWriteEX(EX_MEM_regWrite),
222         .memReadMEM(MEM_WB_memRead),
223         .registerRsID(ID_EX_Reg_Rs),
224         .registerRtID(ID_EX_Reg_Rt),
225         .registerRtEX(EX_MEM_reg_Rt),
226         .registerRdEX(EX_reg),
227         .registerRdMEM(MEM_register),
```

```
228         .stall(stall),
229         .flush(ID_EX_flush)
230     );
231 
232 
233 
234     // EX
235     ALU_control alu_control(
236         .funct(EX_MEM_sign_extend[5:0]),
237         .ALUOp(EX_MEM_alu_op),
238         .ALUCtrl(EX_MEM_alu_control)
239     );
240 
241     Mux_32bit_3to1 forward_A_Mux(
242         .in00(EX_MEM_reg_read_data_1),
243         .in01(WB_write_data),
244         .in10(MEM_WB_alu_result),
245         .control(forward_A),
246         .mux_out(EX_MEM_alu_in_one)
247     );
248 
249     Mux_32bit_3to1 forward_B_Mux(
250         .in00(EX_MEM_reg_read_data_2),
251         .in01(WB_write_data),
252         .in10(MEM_WB_alu_result),
253         .control(forward_B),
254         .mux_out(EX_MEM_alu_temp_two)
255     );
256 
257     assign EX_MEM_alu_in_two = (EX_MEM_aluSrc == 1'b0)?
EX_MEM_alu_temp_two:EX_MEM_sign_extend;
258     // Register stored in EX/MEM
259     assign EX_reg = (EX_MEM_regDst == 0)?EX_MEM_reg_Rt:EX_MEM_reg_Rd;
260 
261     ALU alu(
262         .a(EX_MEM_alu_in_one),
263         .b(EX_MEM_alu_in_two),
264         .ALUCtrl(EX_MEM_alu_control),
265         .ALU_result(EX_MEM_alu_result),
266         .zero(EX_MEM_aluZero)
267     );
268 
269     // EX/MEM
270     EX_MEM_Reg exmem(
271         .clk(clk),
272         .reset(reset),
273         .ALU_result(EX_MEM_alu_result),
274         .reg_read_data_2(EX_MEM_alu_temp_two),
275         .ID_EX_Regsiter_Rd(EX_reg),
```

```
276         // output
277         .ALU_result_out(MEM_WB_alu_result),
278         .reg_read_data_2_out(MEM_WB_reg_read_data_2),
279         .EX_MEM_Regsiter_Rd_out(MEM_register)
280     );
281 
282     reg [31:0] buffer,buffer1,buffer2,buffer3;
283     always @(posedge clk)begin
284       buffer <= EX_MEM_regWrite;
285       buffer1 <= EX_MEM_memRead;
286       buffer2 <= EX_MEM_memtoReg;
287       buffer3 <= EX_MEM_memWrite;
288     end
289     assign MEM_WB_regWrite = buffer;
290     assign MEM_WB_memRead = buffer1;
291     assign MEM_WB_memtoReg = buffer2;
292     assign MEM_WB_memWrite = buffer3;
293     data_memory dm(
294         .clk(clk),
295         .MemRead(MEM_WB_memRead),
296         .MemWrite(MEM_WB_memWrite),
297         .address(MEM_WB_alu_result),
298         .write_data(MEM_WB_reg_read_data_2),
299         .read_data(MEM_WB_D_MEM_read_data)
300     );
301 
302     // MEM/WB
303     MEM_WB_Reg mem_wb(
304         .RegWrite(MEM_WB_regWrite),
305         .MemtoReg(MEM_WB_memtoReg),
306         .D_MEM_read_data_in(MEM_WB_D_MEM_read_data),
307         .ALU_result(MEM_WB_alu_result),
308         .EX_MEM_Reg_Rd(MEM_register),
309         .clk(clk),
310         .reset(reset),
311         .D_MEM_read_data_out(WB_D_MEM_read_data),
312         .D_MEM_read_addr_out(WB_D_MEM_read_addr),
313         .RegWrite_out(WB_regWrite),
314         .MemtoReg_out(WB_memtoReg),
315         .MEM_WB_Reg_Rd(WB_register)
316     );
317 
318 
319     assign WB_write_data = (WB_memtoReg == 0) ? WB_D_MEM_read_addr:WB_D_MEM_read_data;
320 
321     Forward forwad(
322         .registerRsID(ID_EX_Reg_Rs),
323         .registerRtID(ID_EX_Reg_Rt),
324         .registerRsEX(EX_MEM_reg_Rs),
```

```
        .registerRtEX(EX_MEM_reg_Rt),
        .registerRdMEM(MEM_register),
        .registerRdWB(WB_register),
        .regWriteMEM(MEM_WB_regWrite),
        .regWriteWB(WB_regWrite),
        .forwardA(forward_A),
        .forwardB(forward_B),
        .forwardC(forward_C),
        .forwardD(forward_D)
    );



// MEM_WB_RegWrite
endmodule

// 3-to-1 MUX for forwarding

module Mux_32bit_3to1 (in00, in01, in10, mux_out, control);
    input [31:0] in00, in01, in10;
    output [31:0] mux_out;
    input [1:0] control;
    reg [31:0] mux_out;
    always @(in00 or in01 or in10 or control)
    begin
        case(control)
        2'b00:mux_out<=in00;
        2'b01:mux_out<=in01;
        2'b10:mux_out<=in10;
        default: mux_out<=in00;
        endcase
    end
endmodule

module HazardDetection (
    input               branchEqID,
                        branchNeID,
                        memReadEX,
                        regWriteEX,
                        memReadMEM,
    input       [4:0]   registerRsID,
                        registerRtID,
                        registerRtEX,
                        registerRdEX,
                        registerRdMEM,
    output reg          stall,
                        flush
);

```

```
    initial begin
        stall = 1'b0;
        flush = 1'b0;
    end

    always @ ( * ) begin
        if (memReadEX && registerRtEX && (registerRtEX == registerRsID || registerRtEX == registerRtID)) begin
            stall = 1'b1;
            flush = 1'b1;
        end else if (branchEqID || branchNeID) begin
            if (regWriteEX && registerRdEX && (registerRdEX == registerRsID || registerRdEX == registerRtID)) begin
                stall = 1'b1;
                flush = 1'b1;
            end else if (memReadMEM && registerRdMEM && (registerRdMEM == registerRsID || registerRdMEM == registerRtID)) begin
                stall = 1'b1;
                flush = 1'b1;
            end else begin
                stall = 1'b0;
                flush = 1'b0;
            end
        end else begin
            stall = 1'b0;
            flush = 1'b0;
        end
    end

endmodule
```

12. pipe_test.v

```
module pipeline_tb;
    integer i = 0;
    // Inputs
    reg clk;
    // Instantiate the Unit Under Test (UUT)
    Pipeline uut (
        .clk(clk)
    );
    initial begin
        // Initialize Inputs
        clk = 0;
        $dumpfile("pipeline.vcd");
        $dumpvars(1, uut);
        $display("Texual result of pipeline:");
        $display("============================================================");
        #700;
```

```
        $stop;
    end
    always #10 begin
        $display("Time: %d, CLK = %d, PC = 0x%H", i, clk, uut.PC_out_IF);
        $display("[$s0] = 0x%H, [$s1] = 0x%H, [$s2] = 0x%H",
uut.register.register_memory[16], uut.register.register_memory[17],
uut.register.register_memory[18]);
        $display("[$s3] = 0x%H, [$s4] = 0x%H, [$s5] = 0x%H",
uut.register.register_memory[19], uut.register.register_memory[20],
uut.register.register_memory[21]);
        $display("[$s6] = 0x%H, [$s7] = 0x%H, [$t0] = 0x%H",
uut.register.register_memory[22], uut.register.register_memory[23],
uut.register.register_memory[8]);
        $display("[$t1] = 0x%H, [$t2] = 0x%H, [$t3] = 0x%H",
uut.register.register_memory[9], uut.register.register_memory[10],
uut.register.register_memory[11]);
        $display("[$t4] = 0x%H, [$t5] = 0x%H, [$t6] = 0x%H",
uut.register.register_memory[12], uut.register.register_memory[13],
uut.register.register_memory[14]);
        $display("[$t7] = 0x%H, [$t8] = 0x%H, [$t9] = 0x%H",
uut.register.register_memory[15], uut.register.register_memory[24],
uut.register.register_memory[25]);
        $display("================================================");
        clk = ~clk;
        if (~clk) i = i + 1;
    end
endmodule
```

13. register.v

```
module Registers(
    input       clk,
                regWrite,
    input       [4:0]   read_register_1,    read_register_2,
    input       [4:0]   write_register,
    input       [31:0]  write_data,
    output      [31:0]  read_data_1,read_data_2
);
    parameter size = 32;          // 32-bit CPU, $0 - $31
    reg [31:0] register_memory [0:size-1];
    integer i;

    initial begin
        for (i = 0; i < size; i = i + 1)
            register_memory[i] = 32'b0;
    end
    assign read_data_1 = register_memory[read_register_1];
    assign read_data_2 = register_memory[read_register_2];
    always @(negedge clk) begin
```

```
        if (regWrite == 1)
            register_memory[write_register] <= write_data;
    end
endmodule // registers
```

14. sign_extension.v

```
module sign_extension(
    shortInput, longOutput
    );
    input [15:0] shortInput;
    output [31:0] longOutput;
    // reg [31:0] longOutput;
    assign longOutput[15:0] = shortInput[15:0];
    assign longOutput[31:16] = shortInput[15]?16'b1111_1111_1111_1111:16'b0;
endmodule
```

15. state_register.v

```
module IF_ID_Reg(
        // Data input
        input   [31:0]    PC_plus4_in,instruction_in,
        // Control signal input
        input             IF_ID_Write, IF_Flush, clk, reset,
        // output
        output  [31:0]    PC_plus4_out,instruction_out
);
        reg [31:0] PC_plus4_out, instruction_out;

        initial begin
            PC_plus4_out <= 32'b0;
            instruction_out <= 32'b0;
        end
        always @(posedge clk or posedge reset)
        begin if (reset == 1'b1|| IF_Flush == 1'b1) begin
                PC_plus4_out <= 32'b0;
                instruction_out <= 32'b0;
            end else if(IF_ID_Write == 1'b1) begin
                PC_plus4_out <= PC_plus4_in;
                instruction_out <= instruction_in;
            end
        end
endmodule

// Since branch and jump are taken within ID stage, no need to keep Branch jump as well as the address to this register.
module ID_EX_Reg(
        // *****INPUT********
        input           clk,reset,
```

```
30          // hazard control signal
31          input           flush,
32          // Control signal
33          input           RegDst,
34                          MemtoReg,
35                          // Branch,
36                          MemRead,
37                          MemWrite,
38                          ALUSrc,
39                          RegWrite,
40          input   [1:0]   ALUop,
41          // input   [31:0]  PC_plus4_in,
42          // Read data
43          input   [31:0]  reg_read_data_1,reg_read_data_2,extended_imm,
44          // Register ID
45          input   [4:0]   IF_ID_Register_Rs,IF_ID_Register_Rt, IF_ID_Register_Rd,
46          //  ******OUTPUT******
47          output  reg     out_RegDst,
48                          out_MemRead,
49                          out_MemtoReg,
50                          out_MemWrite,
51                          out_ALUSrc,
52                          out_RegWrite,
53          output reg [31:0]  reg_read_data_1_out,reg_read_data_2_out, extended_imm_out,
54          // PC_plus4_out, jump_address_out,
55          output reg [4:0]
IF_ID_Register_Rs_out,IF_ID_Register_Rd_out,IF_ID_Register_Rt_out,
56          output reg [1:0]   ALUop_out
57
58  );
59      // For ID_Flush == 1, which means that lw occurs anc cannnot be solved with
forwarding, set WB, MEM, EX control signal to 0.
60      initial begin
61          out_RegDst      = 1'b0;
62          // out_Jump         = 1'b0;
63          // out_Branch       = 1'b0;
64          out_MemRead     = 1'b0;
65          out_MemtoReg    = 1'b0;
66          out_MemWrite    = 1'b0;
67          out_ALUSrc      = 1'b0;
68          out_RegWrite    = 1'b0;
69          ALUop_out       = 2'b0;
70          reg_read_data_1_out = 32'b0;
71          reg_read_data_2_out = 32'b0;
72          // extended_imm_out = 32'b0;
73          IF_ID_Register_Rd_out = 5'b0;
74          IF_ID_Register_Rs_out = 5'b0;
75          IF_ID_Register_Rt_out = 5'b0;
76          // funct_out              = 6'b0;
```

```
77      end
78
79      always @(posedge clk or posedge reset)
80      begin
81          if (reset == 1'b1) begin
82              out_RegDst      = 1'b0;
83              // out_Jump         = 1'b0;
84              // out_Branch       = 1'b0;
85              out_MemRead     = 1'b0;
86              out_MemtoReg    = 1'b0;
87              out_MemWrite    = 1'b0;
88              out_ALUSrc      = 1'b0;
89              out_RegWrite    = 1'b0;
90              ALUop_out       = 2'b0;
91              reg_read_data_1_out = 32'b0;
92              reg_read_data_2_out = 32'b0;
93              extended_imm_out    = 32'b0;
94              IF_ID_Register_Rd_out = 5'b0;
95              IF_ID_Register_Rs_out = 5'b0;
96              IF_ID_Register_Rt_out = 5'b0;
97            end else if(flush) begin
98              out_RegWrite = 1'b0;
99              out_MemtoReg = 1'b0;
100             // out_Branch    = 1'b0;
101             out_MemRead  = 1'b0;
102             out_MemWrite = 1'b0;
103             // out_Jump  = 1'b0;
104             out_ALUSrc   = 1'b0;
105             out_RegDst   = 1'b0;
106             ALUop_out    = 2'b0;
107           end else begin
108             out_RegWrite <= RegWrite;
109             out_MemtoReg <= MemtoReg;
110             // out_Branch = Branch;
111             out_MemRead <= MemRead;
112             out_MemWrite <= MemWrite;
113             // out_Jump = Jump;
114             out_RegDst <= RegDst;
115             out_ALUSrc <= ALUSrc;
116             ALUop_out <= ALUop;
117             // jump_addr_out = jump_address;
118             // PC_plus4_out = PC_plus4_in;
119             reg_read_data_1_out <= reg_read_data_1;
120             reg_read_data_2_out <= reg_read_data_2;
121             extended_imm_out <= extended_imm;
122             IF_ID_Register_Rs_out <= IF_ID_Register_Rs;
123             IF_ID_Register_Rt_out <= IF_ID_Register_Rt;
124             IF_ID_Register_Rd_out <= IF_ID_Register_Rd;
125             // funct_out = funct;
```

```
126             end
127         end
128 endmodule
129 
130 
131 module EX_MEM_Reg(
132         input       clk,reset,
133         // Hazard signal,
134         // Flush in ID/EX will keep the control signal here zero and do not need flush signal
    any more. For this stage, just store information every posedge.
135         // Control Signal
136                     RegWrite,
137                     MemtoReg,
138                     // Branch,
139                     MemRead,
140                     MemWrite,
141                     // Jump,
142         output  reg RegWrite_out,
143                     MemtoReg_out,
144                     // Branch_out,
145                     MemRead_out,
146                     MemWrite_out,
147                     // Jump_out,
148         // Data
149         input       [31:0]  ALU_result,reg_read_data_2,
150         output  reg [31:0]  ALU_result_out, reg_read_data_2_out,
151         input       [4:0]   ID_EX_Regsiter_Rd,
152         output  reg [4:0]   EX_MEM_Regsiter_Rd_out
153         // No need for ALU zero
154         // input               ALU_zero,
155         // output   reg         ALU_zero_out
156 );
157         initial begin
158             ALU_result_out = 32'b0;
159             reg_read_data_2_out = 32'b0;
160             EX_MEM_Regsiter_Rd_out = 5'b0;
161             // ALU_zero_out  = 1'b0;
162             RegWrite_out = 1'b0;
163             MemtoReg_out = 1'b0;
164             MemRead_out = 1'b0;
165             MemWrite_out = 1'b0;
166         end
167 
168         always @(posedge clk or posedge reset)
169         begin
170             if(reset == 1'b1)
171                 begin
172                     ALU_result_out = 32'b0;
173                     reg_read_data_2_out = 32'b0;
```

```
                EX_MEM_Regsiter_Rd_out = 5'b0;
                RegWrite_out = 1'b0;
                MemtoReg_out = 1'b0;
                MemRead_out = 1'b0;
                MemWrite_out = 1'b0;
            end
        else begin
                MemRead_out = MemRead;
                RegWrite_out = RegWrite;
                MemWrite_out = MemWrite;
                MemtoReg_out = MemtoReg;
                ALU_result_out = ALU_result;
                reg_read_data_2_out = reg_read_data_2;
                EX_MEM_Regsiter_Rd_out = ID_EX_Regsiter_Rd;
        end
        end
endmodule

module MEM_WB_Reg(
    // ******INPUT*******
    input           RegWrite,MemtoReg,
    input   [31:0]  D_MEM_read_data_in,ALU_result,
    input   [4:0]   EX_MEM_Reg_Rd,
    input           clk,reset,
    // ******OUTPUT*******
    output reg      RegWrite_out,MemtoReg_out,
    output reg [31:0] D_MEM_read_data_out,D_MEM_read_addr_out,
    output reg [4:0]  MEM_WB_Reg_Rd
);
    initial begin
        RegWrite_out = 1'b0;
        MemtoReg_out = 1'b0;
        MEM_WB_Reg_Rd = 5'b0;
        D_MEM_read_addr_out = 32'b0;
        D_MEM_read_data_out = 32'b0;
    end
    always @(posedge clk)
    begin
            RegWrite_out <= RegWrite;
            MemtoReg_out <= MemtoReg;
            MEM_WB_Reg_Rd <= EX_MEM_Reg_Rd;
            D_MEM_read_addr_out <= ALU_result;
            D_MEM_read_data_out <= D_MEM_read_data_in;
    end
endmodule
```

16. control.v

```
module Control (
    input       [5:0]   op_code,
    output reg  [1:0]   ALUOp,
    output reg  RegDst,
                Jump,
                Ins_Beq,
                Ins_Bne,
                MemRead,
                MemtoReg,
                MemWrite,
                ALUSrc,
                RegWrite
);

    initial begin
        RegDst      = 1'b0;
        Jump        = 1'b0;
        Ins_Beq     = 1'b0;
        Ins_Bne     = 1'b0;
        MemtoReg    = 1'b0;
        MemWrite    = 1'b0;
        ALUSrc      = 1'b0;
        RegWrite    = 1'b0;
        ALUOp       = 2'b00;
    end

    always @ ( op_code ) begin
        case (op_code)
            6'b000000: begin // R-type
                RegDst      <= 1'b1;
                Jump        <= 1'b0;
                Ins_Beq     <= 1'b0;
                Ins_Bne     <= 1'b0;
                MemRead     <= 1'b0;
                MemtoReg    <= 1'b0;
                MemWrite    <= 1'b0;
                ALUSrc      <= 1'b0;
                RegWrite    <= 1'b1;
                ALUOp       <= 2'b10;
            end
            6'b000010: begin // j
                RegDst      <= 1'b1;
                Jump        <= 1'b1;
                Ins_Beq     <= 1'b0;
                Ins_Bne     <= 1'b0;
                MemRead     <= 1'b0;
                MemtoReg    <= 1'b0;
                MemWrite    <= 1'b0;
                ALUSrc      <= 1'b0;
```

```
                RegWrite    <= 1'b0;
                ALUOp       <= 2'b10;
            end
            6'b000100: begin // beq
                RegDst      <= 1'b1;
                Jump        <= 1'b0;
                Ins_Beq     <= 1'b1;
                Ins_Bne     <= 1'b0;
                MemRead     <= 1'b0;
                MemtoReg    <= 1'b0;
                MemWrite    <= 1'b0;
                ALUSrc      <= 1'b0;
                RegWrite    <= 1'b0;
                ALUOp       <= 2'b01;
            end
            6'b000100: begin // bne
                RegDst      <= 1'b1;
                Jump        <= 1'b0;
                Ins_Beq     <= 1'b0;
                Ins_Bne     <= 1'b1;
                MemRead     <= 1'b0;
                MemtoReg    <= 1'b0;
                MemWrite    <= 1'b0;
                ALUSrc      <= 1'b0;
                RegWrite    <= 1'b0;
                ALUOp       <= 2'b01;
            end
            6'b001000: begin // addi
                RegDst      <= 1'b0;
                Jump        <= 1'b0;
                Ins_Beq     <= 1'b0;
                Ins_Bne     <= 1'b0;
                MemRead     <= 1'b0;
                MemtoReg    <= 1'b0;
                MemWrite    <= 1'b0;
                ALUSrc      <= 1'b1;
                RegWrite    <= 1'b1;
                ALUOp       <= 2'b00;
            end
            6'b001100: begin // andi
                RegDst      <= 1'b0;
                Jump        <= 1'b0;
                Ins_Beq     <= 1'b0;
                Ins_Bne     <= 1'b0;
                MemRead     <= 1'b0;
                MemtoReg    <= 1'b0;
                MemWrite    <= 1'b0;
                ALUSrc      <= 1'b1;
                RegWrite    <= 1'b1;
```

```
                ALUOp       <= 2'b11;
            end
            6'b100011: begin // lw
                RegDst      <= 1'b0;
                Jump        <= 1'b0;
                Ins_Beq     <= 1'b0;
                Ins_Bne     <= 1'b0;
                MemRead     <= 1'b1;
                MemtoReg    <= 1'b1;
                MemWrite    <= 1'b0;
                ALUSrc      <= 1'b1;
                RegWrite    <= 1'b1;
                ALUOp       <= 2'b00;
            end
            6'b101011: begin // sw
                RegDst      <= 1'b0;
                Jump        <= 1'b0;
                Ins_Beq     <= 1'b0;
                Ins_Bne     <= 1'b0;
                MemRead     <= 1'b0;
                MemtoReg    <= 1'b0;
                MemWrite    <= 1'b1;
                ALUSrc      <= 1'b1;
                RegWrite    <= 1'b0;
                ALUOp       <= 2'b00;
            end

            default: ;
        endcase
    end
endmodule
```