VE370 Project Report

Project 2 - Fall 2020

Gengchen Yang (518370910088)

Table of Contents

VE370 Project 2 Individual Report

Introduction

Processors are commonly used in our daily life. In almost every electronic device, there is one or multiple processors. Among the ways to implement processors, single-stage-processor is an easy approach. The figure below shows an single-stage implementation of MIPS architecture.

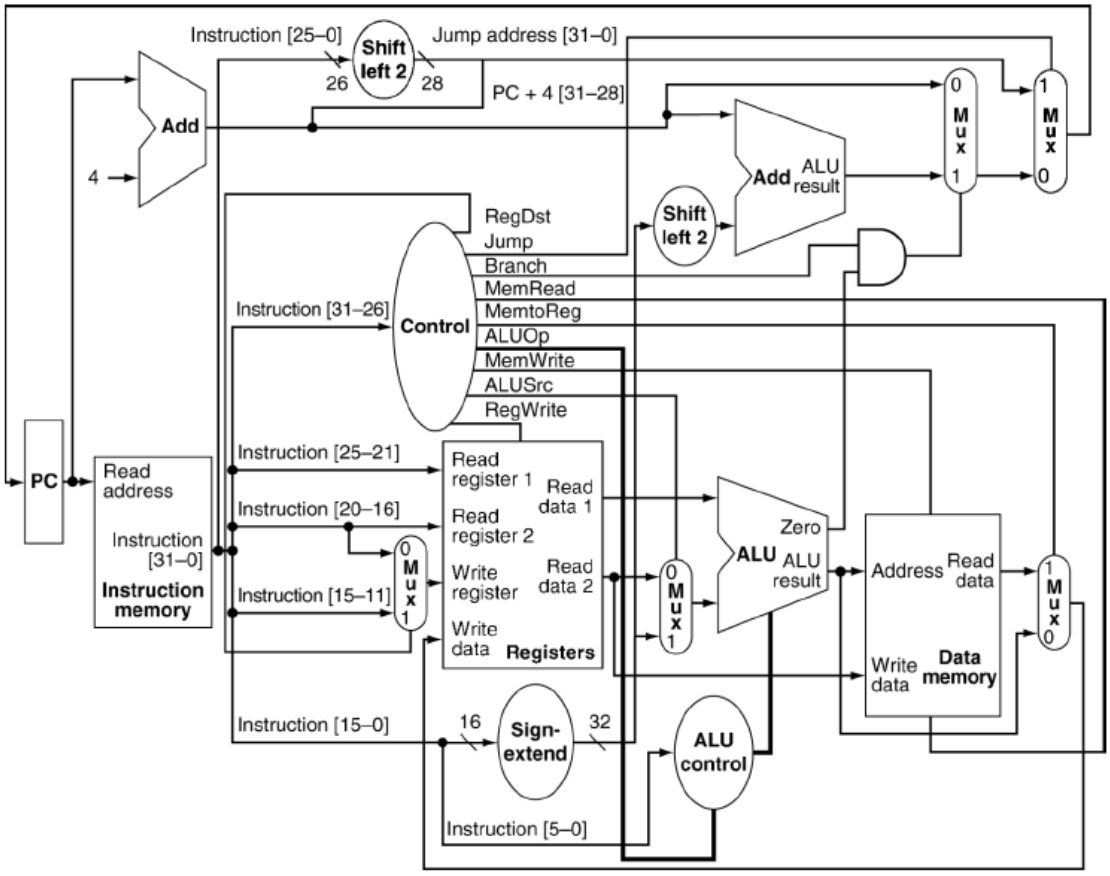

Figure 1. Single stage implementation of MIPS architecture

Implementation

The codes of the implementation of the Single-Stage-Processor is shown below:

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

```
// Company:
// Engineer:
//
// Create Date: 2020/11/01 17:18:05
// Design Name:
// Module Name: Single_cycle_processor
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//


module Single_cycle_processor(clock, reset, PC_addr, PC_input, instruction,
addr, RegDst, Jump, Branch, Branch_not_equal, MemRead, MemtoReg, MemWrite,
ALUSrc, RegWrite, ALUOp, ALU_result, ALU_1, ALU_2, mem0, mem4, mem8,
 reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7, reg8, reg9, reg10, reg11,
reg12, reg13, reg14, reg15, reg16, reg17, reg18, reg19, reg20, reg21, reg22,
reg23, reg24, reg25, reg26, reg27, reg28, reg29, reg30, reg31);
    input clock, reset;
    output [31:0] PC_addr, PC_input, instruction;
    output [31:0] reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7, reg8, reg9,
reg10, reg11, reg12, reg13, reg14, reg15, reg16, reg17, reg18, reg19, reg20,
reg21, reg22, reg23, reg24, reg25, reg26, reg27, reg28, reg29, reg30, reg31;

    // PC block
    wire [31:0] PC_in, PC_out; // PC_in_temp;
    wire clk;
    clock clk1(clock, clk);
    //init temp(clock, reset, PC_in, PC_in_temp);
    PC PC_block(clk, reset, PC_in, PC_out);
    assign PC_addr = PC_out;
    assign PC_input = PC_in;

    //Instruction Memory
    wire [31:0] instruction_code;
    output [5:0] addr;
    Instruction_Mem Instruction_block(PC_out, instruction_code, reset, addr);
    assign instruction = instruction_code;

    //Control Block
    output RegDst, Jump, Branch, Branch_not_equal, MemRead, MemtoReg, MemWrite,
ALUSrc, RegWrite;
    output [1:0] ALUOp;
    control Control_block(instruction_code[31:26], RegDst, Jump, Branch,
Branch_not_equal, MemRead, MemtoReg, ALUOp, MemWrite, ALUSrc, RegWrite);

    //Register File
    wire [31:0] read_data_1, read_data_2, Write_data;
    wire [4:0] write_reg;
```

```
    MUX_reg MUX_1(instruction_code[20:16], instruction_code[15:11], write_reg,
RegDst);
    Reg_file Register_File(clock, reset, instruction_code[25:21],
instruction_code[20:16], read_data_1, read_data_2, write_reg, RegWrite,
Write_data,
    reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7, reg8, reg9, reg10, reg11,
reg12, reg13, reg14, reg15, reg16, reg17, reg18, reg19, reg20, reg21, reg22,
reg23, reg24, reg25, reg26, reg27, reg28, reg29, reg30, reg31);

    //Sign extention
    wire [31:0] sign_extended;
    Sign_extention Sign_Extention(instruction_code[15:0], sign_extended);

    //ALU block
    wire [31:0] ALU_input_2, ALU_out;
    wire zero;
    wire [3:0] ALU_control;
    output [31:0] ALU_result;
    output [31:0] ALU_1, ALU_2;
    ALUControl ALU_Control_block(ALUOp, instruction_code[5:0], ALU_control);
    MUX_32 MUX_ALU(read_data_2, sign_extended, ALU_input_2, ALUSrc);
    ALU ALU_block(read_data_1, ALU_input_2, ALU_out, zero, ALU_control);
    assign ALU_result = ALU_out;
    assign ALU_1 = read_data_1;
    assign ALU_2 = ALU_input_2;

    //Data Memory block
    wire [31:0] Memory_Read_data;
    output [31:0] mem0, mem4, mem8;
    Data_memory Data_Memory_block(clock, reset, ALU_out[7:2], MemRead, MemWrite,
Memory_Read_data, read_data_2, mem0, mem4, mem8);
    MUX_32 MUX_datamem(ALU_out, Memory_Read_data, Write_data, MemtoReg);

    //PC+4
    wire [31:0] PC_add_4;
    ALU_add PC_add4(PC_out, 32'b00000000000000000000000000000100, PC_add_4);

    //Shift left 2
    wire [31:0] Jump_address;
    Shift_left_two_jump Jump_addr (instruction_code[25:0], Jump_address);

    //branch shift
    wire [31:0] relative_address;
    Shift_left_two_branch Branch_addr(sign_extended, relative_address);

    //Branch ALU
    wire [31:0] Branch_address;
    ALU_add Branch_ALU(PC_add_4, relative_address, Branch_address);

    //Branch
    wire Branch_select;
    wire [31:0] PC_address_1;
    assign Branch_select = (Branch & zero) ? 1 : 0;
    MUX_32 MUX_branch(PC_add_4, Branch_address, PC_address_1, Branch_select);

    //Branch Not Equal
    wire Branch_not_equal_select;
    wire [31:0] PC_address_2;
```

```
    assign Branch_not_equal_select = (Branch_not_equal & (~zero)) ? 1 : 0;
    MUX_32 MUX_branch_not_equal(PC_address_1, Branch_address, PC_address_2,
Branch_not_equal_select);

    //Jump
    MUX_32 MUX_jump(PC_address_2, Jump_address, PC_in, Jump);

endmodule

//initialization delay
module init(clock, reset, in, out);
    input clock, reset;
    input [31:0] in;
    output [31:0] out;
    reg count;
    reg [31:0] out;

    always @(posedge clock or posedge reset) begin
        if (reset == 1) begin
            count = 1;
            out = 0;
        end
        else begin
            if (count == 1) begin
                count <= 0;
                out = 0;
            end
            else out <= in;
        end
    end
endmodule

//mux before Reg_File
module MUX_reg(input_1, input_2, out, select);
    input [4:0] input_1, input_2;
    input select;
    output [4:0] out;
    assign out = select ? input_2 : input_1;
endmodule

//program counter
module PC(clock, reset, PC_in, PC_out);
    input clock, reset;
    input [31:0] PC_in;
    output [31:0] PC_out;
    reg [31:0] PC_out;

    always @(posedge clock or posedge reset) begin
        if (reset == 1) begin
            PC_out = 32'd0;
        end
        else begin
            PC_out = PC_in;
        end
    end
endmodule

//sign extention
```

```
module Sign_extention(sign_original, sign_extended);
    input [15:0] sign_original;
    output [31:0] sign_extended;
    assign sign_extended[31:16] = sign_original[15] ? 16'b1111_1111_1111_1111 :
0;
    assign sign_extended[15:0] = sign_original[15:0];
endmodule

//multiply by four in branch
module Shift_left_two_branch(sign_original, sign_shifted);
    input [31:0] sign_original;
    output [31:0] sign_shifted;
    assign sign_shifted[31:2] = sign_original[29:0];
    assign sign_shifted[1:0] = 2'b00;
endmodule

//multiply by four in jump
module Shift_left_two_jump(sign_original, sign_shifted);
    input [25:0] sign_original;
    output [31:0] sign_shifted;
    assign sign_shifted[27:2] = sign_original[25:0];
    assign sign_shifted[1:0] = 2'b00;
    assign sign_shifted[31:27] = 0;
endmodule

//Memory block
module Data_memory(clock, reset, address, mem_read, mem_write, mem_out,
data_write, data0, data4, data8);
    input clock, reset, mem_read, mem_write;
    input [5:0] address;
    input [31:0] data_write;
    output [31:0] mem_out;
    output [31:0] data0, data4, data8;
    reg [31:0] Memory [15:0];
    reg [31:0] mem_out;
    reg [4:0] i;

    // wire number = address[7:2];

    assign data0 = Memory[0];
    assign data4 = Memory[1];
    assign data8 = Memory[2];

    always @ (posedge clock or posedge reset) begin
        if (reset == 1) begin
            for (i = 0; i < 16; i = i + 1) begin
                Memory[i] = 0;
            end
        end
        else begin
            if (mem_write == 1) begin
                Memory[address] = data_write;
            end
            if (mem_read == 1) begin
                mem_out = Memory[address];
            end
            else begin
                mem_out = 32'bx;
```

```
            end
        end
    end
endmodule

//Register File block
module Reg_file(clock, reset, read_reg_1, read_reg_2, read_data_1, read_data_2,
write_reg, RegWrite, Write_data,
reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7, reg8, reg9, reg10, reg11, reg12,
reg13, reg14, reg15, reg16, reg17, reg18, reg19, reg20, reg21, reg22, reg23,
reg24, reg25, reg26, reg27, reg28, reg29, reg30, reg31);
    input clock, reset, RegWrite;
    input [4:0] read_reg_1, read_reg_2, write_reg;
    input [31:0] Write_data;
    output [31:0] read_data_1, read_data_2;
    output [31:0] reg0, reg1, reg2, reg3, reg4, reg5, reg6, reg7, reg8, reg9,
reg10, reg11;
    output [31:0] reg12, reg13, reg14, reg15, reg16, reg17, reg18, reg19, reg20,
reg21;
    output [31:0] reg22, reg23, reg24, reg25, reg26, reg27, reg28, reg29, reg30,
reg31;
    reg [31:0] Registers[31:0];
    reg [31:0] read_data_1, read_data_2;
    reg [5:0] i;

    assign reg0 = Registers[0];
    assign reg1 = Registers[1];
    assign reg2 = Registers[2];
    assign reg3 = Registers[3];
    assign reg4 = Registers[4];
    assign reg5 = Registers[5];
    assign reg6 = Registers[6];
    assign reg7 = Registers[7];
    assign reg8 = Registers[8];
    assign reg9 = Registers[9];
    assign reg10 = Registers[10];
    assign reg11 = Registers[11];
    assign reg12 = Registers[12];
    assign reg13 = Registers[13];
    assign reg14 = Registers[14];
    assign reg15 = Registers[15];
    assign reg16 = Registers[16];
    assign reg17 = Registers[17];
    assign reg18 = Registers[18];
    assign reg19 = Registers[19];
    assign reg20 = Registers[20];
    assign reg21 = Registers[21];
    assign reg22 = Registers[22];
    assign reg23 = Registers[23];
    assign reg24 = Registers[24];
    assign reg25 = Registers[25];
    assign reg26 = Registers[26];
    assign reg27 = Registers[27];
    assign reg28 = Registers[28];
    assign reg29 = Registers[29];
    assign reg30 = Registers[30];
    assign reg31 = Registers[31];
```

```
    always @ (posedge clock or posedge reset) begin
        if (reset == 1) begin
            for (i = 0; i < 32; i = i + 1) begin
                Registers[i] = 0;
            end
        end
        else if (RegWrite == 1) begin
            Registers[write_reg] = Write_data;
        end
        read_data_1 = Registers[read_reg_1];
        read_data_2 = Registers[read_reg_2];
    end

endmodule

//ALU unit for arithmatic
module ALU(input_1, input_2, ALU_out, zero, control);
    input [31:0] input_1, input_2;
    input [3:0] control;
    output zero;
    output [31:0] ALU_out;
    reg zero;
    reg [31:0] ALU_out;

    always @ (control or input_1 or input_2) begin
        case (control)
        4'b0000: begin //add
            zero <= 0;
            ALU_out <= input_1 + input_2;
        end
        4'b0001: begin //equal(slt, beq, bne) and sub
            zero <= (input_1 == input_2) ? 1 : 0;
            ALU_out = input_1 - input_2;
        end
        4'b0011: begin // and
            zero <= 0;
            ALU_out = input_1 & input_2;
        end
        4'b0010: begin //or
           zero <= 0;
            ALU_out <= input_1 | input_2;
        end
        4'b0101: begin //slt
            zero <= (input_1 == input_2) ? 1 : 0;
            ALU_out = (input_1 < input_2) ? 32'd1 : 32'd0;
        end
        default: begin
            zero <= 0;
            ALU_out = 32'bx;
        end
        endcase
    end

endmodule

//ALU control
module ALUControl(ALUOp, funct, control);
    input [1:0] ALUOp;
```

```
    input [5:0] funct;
    output [3:0] control;

    assign control[3] = 0;
    assign control[2] = (ALUOp == 2'b10) & (funct == 6'b101010);
    assign control[1] = (ALUOp == 2'b11) | ((ALUOp == 2'b10) & ((funct ==
6'b100101)  | (funct == 6'b100100)));
    assign control[0] = (ALUOp == 2'b01) | (ALUOp == 2'b11) | ((ALUOp == 2'b10)
& ((funct[1] == 1) | (funct == 6'b100100)));
endmodule

//Control unit
module control(Instruction, RegDst, Jump, Branch, Branch_not_equal, MemRead,
MemtoReg, ALUOp, MemWrite, ALUSrc, RegWrite);
    input [5:0] Instruction;
    output RegDst, Jump, Branch, Branch_not_equal, MemRead, MemtoReg, MemWrite,
ALUSrc, RegWrite;
    output [1:0] ALUOp;

    assign ALUOp[1] = (Instruction == 0) | (Instruction == 6'b001100);
    assign ALUOp[0] = (Instruction == 6'b000100) | (Instruction == 6'b000101) |
(Instruction == 6'b001100);
    assign RegDst = (Instruction == 6'b000000);
    assign Jump = (Instruction == 6'b000010);
    assign Branch = (Instruction == 6'b000100);
    assign Branch_not_equal = (Instruction == 6'b000101);
    assign MemRead = (Instruction == 6'b100011);
    assign MemtoReg = (Instruction == 6'b100011);
    assign MemWrite = (Instruction == 6'b101011);
    assign ALUSrc = (Instruction == 6'b100011) | (Instruction == 6'b101011) |
(Instruction == 6'b001000) | (Instruction == 6'b001100);
    assign RegWrite = (!(Instruction == 6'b101011)) & (!(Instruction ==
6'b000100)) & (!(Instruction == 6'b000101)) & (!(Instruction == 6'b000010));

endmodule

//MUX_32
module MUX_32(input_1, input_2, out, select);
    input [31: 0] input_1, input_2;
    output [31:0] out;
    input select;
    assign out = select ? input_2 : input_1;
endmodule

//Instruction Memory
module Instruction_Mem(address, instr_code, reset, show);
    input reset;
    input [31:0] address;
    output [31:0] instr_code;
    output [5:0] show;
    reg [31:0] Instructions[41:0];
    wire [5:0] addr = address[7:2];
    assign instr_code = Instructions[addr];
    assign show = addr;

    always @(posedge reset) begin
        Instructions[0] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
```

```
        Instructions[1] = 32'b00100000000010010000000000110111; //addi $t1,
$zero, 0x37
        Instructions[2] = 32'b00000001000010011000000000100100; //and $s0, $t0,
$t1
        Instructions[3] = 32'b00000001000010011000000000100101; //or $s0, $t0,
$t1
        Instructions[4] = 32'b10101100000100000000000000000100; //sw $s0,
4($zero)
        Instructions[5] = 32'b10101100000010000000000000001000; //sw $t0,
8($zero)
        Instructions[6] = 32'b00000001000010011000100000100000; //add $s1, $t0,
$t1
        Instructions[7] = 32'b00000001000010011001000000100010; //sub $s2, $t0,
$t1
        Instructions[8] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[9] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[10] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[11] = 32'b00010010001100100000000000010010; //beq $s1, $s2,
error0
        Instructions[12] = 32'b10001100000100010000000000000100; //lw $s1,
4($zero)
        Instructions[13] = 32'b00110010001100100000000001001000; //andi $s2,
$s1, 0x48
        Instructions[14] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[15] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[16] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[17] = 32'b00010010001100100000000000001111; //beq $s1, $s2,
error1
        Instructions[18] = 32'b10001100000100110000000000001000; //lw $s3,
8($zero)
        Instructions[19] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[20] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[21] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[22] = 32'b00010010000100110000000000001101; //beq $s0, $s3,
error2
        Instructions[23] = 32'b00000010010100011010000000101010; //slt $s4, $s2,
$s1 (Last)
        Instructions[24] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[25] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[26] = 32'b00100000000010000000000000100000; //addi $t0,
$zero, 0x20
        Instructions[27] = 32'b00010010100000000000000000001111; //beq $s4, $0,
EXIT
        Instructions[28] = 32'b00000001000100000100100000100000; //add $s2, $s1,
$0
        Instructions[29] = 32'b00001000000000000000000000010111; //j Last
```

```
        Instructions[30] = 32'b00100000000010000000000000000000; //addi $t0, $0,
0(error0)
        Instructions[31] = 32'b00100000000010010000000000000000; //addi $t1, $0,
0
        Instructions[32] = 32'b00001000000000000000000000111111; //j EXIT
        Instructions[33] = 32'b00100000000010000000000000000001; //addi $t0, $0,
1(error1)
        Instructions[34] = 32'b00100000000010010000000000000001; //addi $t1, $0,
1
        Instructions[35] = 32'b00001000000000000000000000111111; //j EXIT
        Instructions[36] = 32'b00100000000010000000000000000010; //addi $t0, $0,
2(error2)
        Instructions[37] = 32'b00100000000010010000000000000010; //addi $t1, $0,
2
        Instructions[38] = 32'b00001000000000000000000000111111; //j EXIT
        Instructions[39] = 32'b00100000000010000000000000000011; //addi $t0, $0,
3(error3)
        Instructions[40] = 32'b00100000000010010000000000000011; //addi $t1, $0,
3
        Instructions[41] = 32'b00001000000000000000000000111111; //j EXIT
    end
endmodule

//32 bit Add
module ALU_add(input_1, input_2, out);
    input [31:0] input_1, input_2;
    output [31:0] out;
    assign out = input_1 + input_2;
endmodule

//clock divider
module clock(clk, clock);
    input clk;
    output clock;
    reg [100:0] counter = 0;
    reg clock;
    always @(posedge clk) begin
        if (counter == 4) begin
            counter <= 0;
            clock <= 1;
        end
        else begin
            clock <= 0;
            counter <= counter + 1;
        end
    end
endmodule
```

Simulation Results

We used the following set of code to test the processor:

```
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00100000 00001001 00000000 00110111 //addi $t1, $zero, 0x37
00000001 00001001 10000000 00100100 //and $s0, $t0, $t1
```

```
00000001 00001001 10000000 00100101 //or $s0, $t0, $t1
10101100 00010000 00000000 00000100 //sw $s0, 4($zero)
10101100 00001000 00000000 00001000 //sw $t0, 8($zero)
00000001 00001001 10001000 00100000 //add $s1, $t0, $t1
00000001 00001001 10010000 00100010 //sub $s2, $t0, $t1
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00010010 00110010 00000000 00010010 //beq $s1, $s2, error0
10001100 00010001 00000000 00000100 //lw $s1, 4($zero)
00110010 00110010 00000000 01001000 //andi $s2, $s1, 0x48
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00010010 00110010 00000000 00001111 //beq $s1, $s2, error1
10001100 00010011 00000000 00001000 //lw $s3, 8($zero)
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00010010 00010011 00000000 00001101 //beq $s0, $s3, error2
00000010 01010001 10100000 00101010 //slt $s4, $s2, $s1 (Last)
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00100000 00001000 00000000 00100000 //addi $t0, $zero, 0x20
00010010 10000000 00000000 00001111 //beq $s4, $0, EXIT
00000010 00100000 10010000 00100000 //add $s2, $s1, $0
00001000 00000000 00000000 00010111 //j Last
00100000 00001000 00000000 00000000 //addi $t0, $0, 0(error0)
00100000 00001001 00000000 00000000 //addi $t1, $0, 0
00001000 00000000 00000000 00111111 //j EXIT
00100000 00001000 00000000 00000001 //addi $t0, $0, 1(error1)
00100000 00001001 00000000 00000001 //addi $t1, $0, 1
00001000 00000000 00000000 00111111 //j EXIT
00100000 00001000 00000000 00000010 //addi $t0, $0, 2(error2)
00100000 00001001 00000000 00000010 //addi $t1, $0, 2
00001000 00000000 00000000 00111111 //j EXIT
00100000 00001000 00000000 00000011 //addi $t0, $0, 3(error3)
00100000 00001001 00000000 00000011 //addi $t1, $0, 3
00001000 00000000 00000000 00111111 //j EXIT
```

The simulation results are shown below:

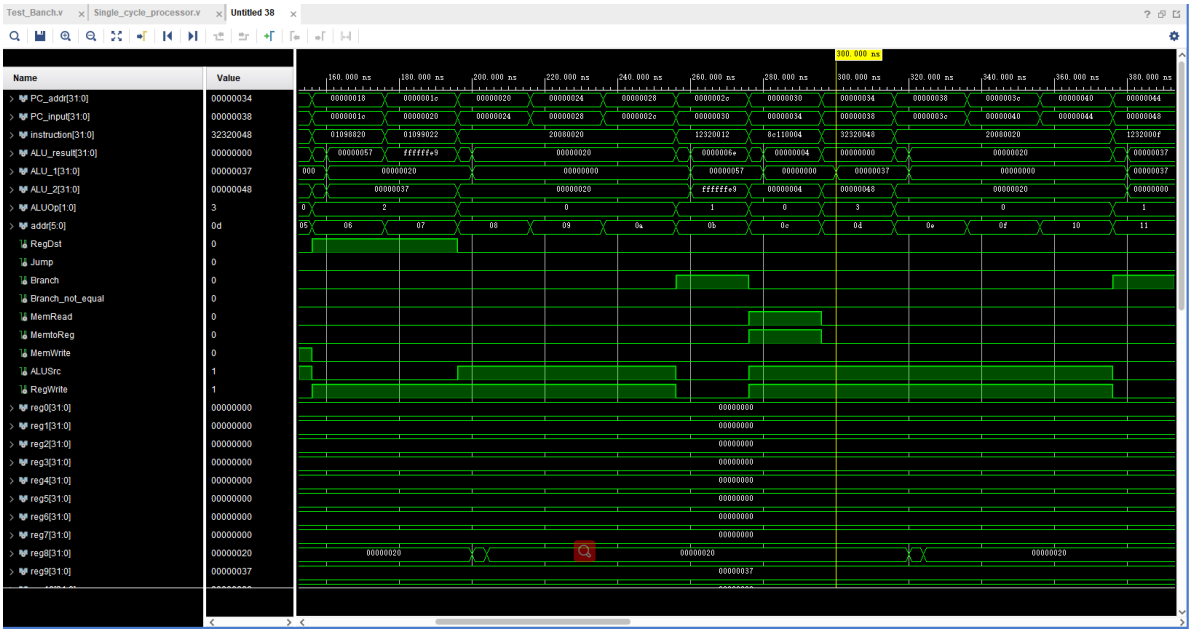

Figure 2. Part of the simulation results

The full results are shown below:

```
TIME = 60, CLK = 0, PC = 00

[$s0] = 00000000, [$s1] = 00000000, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000000, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 80, CLK = 1, PC = 01

[$s0] = 00000000, [$s1] = 00000000, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 100, CLK = 2, PC = 02

[$s0] = 00000020, [$s1] = 00000000, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020
```

```
[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 120, CLK = 3, PC = 03

[$s0] = 00000037, [$s1] = 00000000, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 140, CLK = 4, PC = 04

[$s0] = 00000037, [$s1] = 00000000, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 160, CLK = 5, PC = 05

[$s0] = 00000037, [$s1] = 00000000, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 180, CLK = 6, PC = 06

[$s0] = 00000037, [$s1] = 00000057, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000
```

```
[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 200, CLK = 7, PC = 07

[$s0] = 00000037, [$s1] = 00000057, [$s2] = fffffffe9

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 220, CLK = 8, PC = 08

[$s0] = 00000037, [$s1] = 00000057, [$s2] = fffffffe9

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 240, CLK = 9, PC = 09

[$s0] = 00000037, [$s1] = 00000057, [$s2] = fffffffe9

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 260, CLK = 10, PC = 0a

[$s0] = 00000037, [$s1] = 00000057, [$s2] = fffffffe9

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000
```

```
[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 280, CLK = 11, PC = 0b

[$s0] = 00000037, [$s1] = 00000057, [$s2] = fffffffe9

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 300, CLK = 12, PC = 0c

[$s0] = 00000037, [$s1] = 00000037, [$s2] = fffffffe9

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 320, CLK = 13, PC = 0d

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 340, CLK = 14, PC = 0e

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000
```

```
TIME = 360, CLK = 15, PC = 0f

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 380, CLK = 16, PC = 10

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 400, CLK = 17, PC = 11

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000000, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 420, CLK = 18, PC = 12

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 440, CLK = 19, PC = 13
```

```
[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 460, CLK = 20, PC = 14

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 480, CLK = 21, PC = 15

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 500, CLK = 22, PC = 16

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 520, CLK = 23, PC = 17

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000
```

```
[$s3] = 00000020, [$s4] = 00000001, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 540, CLK = 24, PC = 18

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000020, [$s4] = 00000001, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 560, CLK = 25, PC = 19

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000020, [$s4] = 00000001, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 580, CLK = 26, PC = 1a

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000020, [$s4] = 00000001, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 600, CLK = 27, PC = 1b

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000000

[$s3] = 00000020, [$s4] = 00000001, [$s5] = 00000000
```

```
[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 620, CLK = 28, PC = 1c

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000037

[$s3] = 00000020, [$s4] = 00000001, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 640, CLK = 29, PC = 1d

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000037

[$s3] = 00000020, [$s4] = 00000001, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 660, CLK = 30, PC = 17

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000037

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 680, CLK = 31, PC = 18

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000037

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020
```

```
[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 700, CLK = 32, PC = 19

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000037

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 720, CLK = 33, PC = 1a

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000037

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 740, CLK = 34, PC = 1b

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000037

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000

[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000

TIME = 760, CLK = 35, PC = 2b

[$s0] = 00000037, [$s1] = 00000037, [$s2] = 00000037

[$s3] = 00000020, [$s4] = 00000000, [$s5] = 00000000

[$s6] = 00000000, [$s7] = 00000000, [$t0] = 00000020

[$t1] = 00000037, [$t2] = 00000000, [$t3] = 00000000
```

```
[$t4] = 00000000, [$t5] = 00000000, [$t6] = 00000000

[$t7] = 00000000, [$t8] = 00000000, [$t9] = 00000000
```

(Note: All PC values are divided by four, for convenience of comparing to the instructions)

Peer evaluation

Name	Level of contribution	Description of contribution
Gengchen Yang	5	Forwarding unit, FPGA implementation
Jiaying Xu	5	Patch for pipelined system
Qinhang Wu	5	Synthesis, FPGA implementation, RTL schematic
Yuru Liu	5	Hazard detection unit, bonus

Conclusions

We can see that the results are just as we expected, and hence it perfectly supports MIPS instructions. Also, compared with pipeline processor, we do not need to worry about the hazards and data dependencies.

Appendix

(None)