UM–SJTU Joint Institute
Introduction to Computer Organization (VE370)

Project Two Report

Single-Cycle CPU

Name: Mingxuan Lu ID: 518021911166

November 7, 2020

1 Objective

In this project, we will Model both single cycle (individual work) and pipelined implementation (team work) of MIPS computer in Verilog that support a subset of MIPS instruction set including:

- The memory-reference instructions load word (lw) and store word (sw).
- The arithmetic-logical instructions add, addi, sub, and, andi, or, and slt.
- The jumping instructions branch equal (beq), branch not equal (bne), and jump (j).

  In this report, only individual work will be included.

2 Procedure

2.1 Single-Cycle CPU

For a Single-Cycle CPU, we refer to Figure 1 as a top-level block diagram.

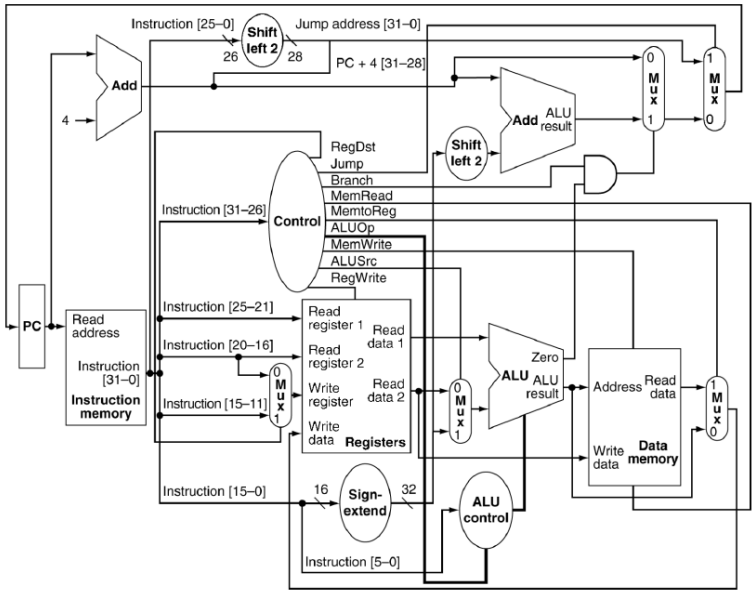

Figure 1. Single-Cycle Implementation of MIPS Architecture

All the modules implemented in the verilog to realize this diagram is listed in the table below (Table 1).

Block Name	**Module Name**
PC	PC.v
Instruction Memory	instructionMEM.v
Registers	RegFile.v
Data Memory	dataMEM.v
Sign-extend	SignExtend.v
ALU Control	ALUControl.v
ALU	ALU.v
Control	Control.v
5-bit Mux	Mux_5.v
32-bit Mux	Mux_32.v

Table 1. Block Name vs. Module Name

3 Simulation Result

The **concise** simulation result is listed below (Table 2). Only changed registers are listed. The **complete** simulation result is attached in Appendix B.

Cycle Number	PC	Changed Registers
0	0x00000000	[$t0] = 0x00000020
1	0x00000004	[$t1] = 0x00000037
2	0x00000008	[$s0] = 0x00000020
3	0x0000000c	[$s0] = 0x00000037
4	0x00000010	*sw*
5	0x00000014	*sw*
6	0x00000018	[$s1] = 0x00000057
7	0x0000001c	[$s2] = 0xfffffffe9
8	0x00000020	*Not-taken branch*
9	0x00000024	[$s1] = 0x00000037
10	0x00000028	[$s2] = 0x00000000
11	0x0000002c	*Not-taken branch*
12	0x00000030	[$s3] = 0x00000020
13	0x00000034	*Not-taken branch*
14	0x00000038	[$s4] = 0x00000001
15	0x0000003c	*Not-taken branch*
16	0x00000040	[$s2] = 0x00000037
17	0x00000044	*j Last*
18	0x00000038	[$s4] = 0x00000000
19	0x0000003c	*Taken branch to "Next"*
20	0x0000007c	*Not-taken branch*
21	0x00000080	*Taken branch to Exit*
22	0x000000c0	*Exit*

The change in data memory can be testified by later "lw" instructions to registers. For example, in the 4th cycle, the value of $s0 is stored in 4($zero). Then in the 9th cycle, the value of $s0 stored in 4($zero) is loaded into $s1. The equivalence of the values of $s0 and $s1 justify the correctness of "sw" instruction.

4 Conclusion

In this project, we use Verilog to design and implement a Single-Cycle CPU. All the instructions in the requirements, including add, addi, sub, and, andi, or, slt, lw, sw, beq, bne and j, have been designed, tested and verified.

For the Single-Cycle CPU, instructions are translated into binary codes to be dealt with the CPU. Since it only dealing with one instruction in one clock cycle, time latency is its biggest shortcoming. Therefore, the design and implementation will be covered in our group report.

Appendix A: Source code

Single_Cycle.v

```
`ifndef MODULE_SINGLE_CYCLE
`define MODULE_SINGLE_CYCLE
`timescale 1ns / 1ps
`include "ALU.v"
`include "ALUControl.v"
`include "Control.v"
```

```
`include "Mux_32.v"
`include "Mux_5.v"
`include "PC.v"
`include "SignExtend.v"
`include "DataMEM.v"
`include "instructionMEM.v"
`include "RegFile.v"



//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/29 14:33:04
// Design Name:
// Module Name: Single_Cycle
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Single_Cycle(clk);
    input clk;
    wire[31:0] PCIn;
    wire[31:0] PCOut;
    wire[31:0] PCPlus4;
    wire[31:0] instruction;
    wire[31:0] jumpAddress;
    wire[31:0] branchAddress;
    wire[31:0] addedAddress;
    wire jump;
    wire beq;
    wire bne;
    wire memRead;
    wire memReg;
    wire branch;
    wire PCHold;
    wire regDst;
    wire regWrite;
    wire memWrite;
    wire ALUSrc;
    wire[4:0] writeReg;
    wire[1:0] ALUOp;
    wire[31:0] extended;
    wire[31:0] ALUsecond;
    wire[31:0] ALUResult;
    wire[31:0] writeData;
    wire[31:0] regReadData1;
    wire[31:0] regReadData2;
    wire[31:0] memReadData;
    wire[3:0] ALUCtrlSignal;
    wire ALUZero;

    assign PCPlus4 = PCOut + 4;
    assign addedAddress = PCPlus4 + (extended << 2);
    assign branch = (beq & ALUZero) | (bne & ~ALUZero);
    assign jumpAddress = {PCPlus4[31:28], instruction[25:0], 2'b0};
    assign branchAddress = (branch == 0) ? PCPlus4 : addedAddress;
    assign PCIn = (jump == 0) ? branchAddress : jumpAddress;
    assign PCHold = 0;

    PC pc(PCIn, PCOut, PCHold, clk);

    RegFile RFile(.clk(clk), .RegWrite(regWrite), .ReadReg1(instruction[25:21]), .ReadReg2(instruction[20:16]),
    .WriteReg(writeReg), .ReadData1(regReadData1), .ReadData2(regReadData2), .WriteData(writeData));

    instructionMEM instruMEM(.address(PCOut), .instruction(instruction));

    Control ctrl(.Opcode(instruction[31:26]), .RegDst(regDst), .Jump(jump), .beq(beq), .bne(bne),
    .MemRead(memRead), .MemtoReg(memReg), .ALUOp(ALUOp), .MemWrite(memWrite), .ALUSrc(ALUSrc),
.RegWrite(regWrite));

    dataMEM DM(.clk(clk), .address(ALUResult), .DataWrite(regReadData2), .DataRead(memReadData),
     .WriteMem(memWrite), .ReadMem(memRead));

    SignExtend SE(.DataIn(instruction[15:0]), .DataOut(extended));
```

```
    ALUControl ALUctrl(.Funcode(instruction[5:0]), .Opcode(ALUOp), .ctrl(ALUCtrlSignal));

    ALU alu(.A(regReadData1), .B(ALUsecond), .ctrl(ALUCtrlSignal), .zero(ALUZero),
    .result(ALUResult));

    Mux_5 Reg_Mux(.A(instruction[20:16]), .B(instruction[15:11]), .ctrl(regDst), .out(writeReg));

    Mux_32 ALU_Mux(.A(regReadData2), .B(extended), .ctrl(ALUSrc), .out(ALUsecond));

    Mux_32 WB_Mux(.A(ALUResult), .B(memReadData), .ctrl(memReg), .out(writeData));

endmodule
`endif
```

DataMEM.v

```
`ifndef MODULE_DATAMEM
`define MODULE_DATAMEM
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/31 22:08:34
// Design Name:
// Module Name: DataMEM
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module dataMEM(clk, address, DataWrite, DataRead, WriteMem, ReadMem);
    input clk;
    input [31:0] address;
    input [31:0] DataWrite;
    input WriteMem;
    input ReadMem;
    output [31:0] DataRead;
    reg [31:0] memory[0:128]; //store 128 data totally
    integer i;

    initial begin
            for (i = 0; i < 128; i = i + 1)
                memory[i] = 32'b0;
            end

    always @(posedge clk)
        begin
            if(WriteMem == 1'b1) memory[address >> 2] = DataWrite;
        end

        assign DataRead = (ReadMem==1'b0) ? 32'b0:memory[address >> 2];
endmodule

`endif
```

ALU.v

```
`ifndef MODULE_ALU
`define MODULE_ALU
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/29 16:55:10
// Design Name:
// Module Name: ALU
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
```

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////

module ALU(A, B, ctrl, zero, result);
    input [31:0] A;
    input [31:0] B;
    input [3:0] ctrl;
    output [31:0] result;
    output zero;
    reg [31:0] result;

    assign zero = (result == 0);

    always @(ctrl or A or B) begin
        case(ctrl)
            4'b0000:
            begin
                result = A & B;
            end
            4'b0001:
            begin
                result = A | B;
            end
            4'b0010:
            begin
                result = A + B;
            end
            4'b0110:
            begin
                result = A - B;
            end
            4'b0111:
            begin
                if(A < B) result = 1;
                else result = 0;
            end
            4'b1100:
                result = ~(A | B);
       endcase
    end
endmodule

`endif
```

ALUControl.v

```
`ifndef MODULE_ALUCONTROL
`define MODULE_ALUCONTROL
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/29 16:59:13
// Design Name:
// Module Name: ALUControl
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module ALUControl(Funcode, Opcode, ctrl);
    input [5:0] Funcode;
    input [1:0] Opcode;
    output [3:0] ctrl;
    reg [3:0] ctrl;

    always @(Funcode or Opcode)
    begin
        case(Opcode)
            2'b00:
                ctrl = 4'b0010;
            2'b01:
```

```
                ctrl = 4'b0110;
            2'b10:
            begin
                case(Funcode)
                6'b100000:
                ctrl = 4'b0010;
                6'b100010:
                ctrl = 4'b0110;
                6'b100100:
                ctrl = 4'b0000;
                6'b100101:
                ctrl = 4'b0001;
                6'b101010:
                ctrl = 4'b0111;
                endcase
            end
            2'b11:
                ctrl = 4'b0000;
            endcase
    end
endmodule
`endif
```

Control.v

```
`ifndef MODULE_CONTROL
`define MODULE_CONTROL
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/29 17:04:37
// Design Name:
// Module Name: Control
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Control(Opcode, RegDst, Jump, beq, bne, MemRead, MemtoReg, ALUOp, MemWrite, ALUSrc, RegWrite);
    input[5:0] Opcode;
    output RegDst;
    output Jump;
    output beq;
    output bne;
    output MemRead;
    output MemtoReg;
    output MemWrite;
    output ALUSrc;
    output RegWrite;

    output[1:0] ALUOp;

    assign RegDst = (Opcode == 6'b000000) ? 1:0;
    assign Jump = (Opcode == 6'b000010) ? 1:0;
    assign beq = (Opcode == 6'b000100) ? 1:0;
    assign bne = (Opcode == 6'b000101) ? 1:0;
    assign MemRead = (Opcode == 6'b100011) ? 1:0;
    assign MemtoReg = (Opcode == 6'b100011) ? 1:0;
    assign MemWrite = (Opcode == 6'b101011) ? 1:0;
    assign ALUSrc = (Opcode == 6'b100011 || Opcode == 6'b101011 || Opcode == 6'b000010 || Opcode == 6'b001000 ||
Opcode == 6'b001100) ? 1:0;
    assign RegWrite = (Opcode == 6'b000000 || Opcode == 6'b100011 || Opcode == 6'b001000 || Opcode == 6'b001100) ?
1:0;
    assign ALUOp[0] = (Opcode == 6'b000100 || Opcode == 6'b000101 || Opcode == 6'b000010 || Opcode == 6'b001100) ?
1:0;
    assign ALUOp[1] = (Opcode == 6'b000000 || Opcode == 6'b001100) ? 1:0;

endmodule

`endif
```

RegFile.v

```
`ifndef MODULE_REGFILE
`define MODULE_REGFILE
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/31 22:48:03
// Design Name:
// Module Name: RegFile
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module RegFile(clk, RegWrite, ReadReg1, ReadReg2, WriteReg, ReadData1, ReadData2, WriteData);
    integer i;
    input clk, RegWrite;
    input[4:0] ReadReg1, ReadReg2;
    input[4:0] WriteReg;
    output[31:0] ReadData1, ReadData2;
    input[31:0] WriteData;

    reg [31:0] registers [0:31];

    initial begin
        for (i = 0; i < 32; i = i + 1)
            registers[i] = 32'b0;
    end

    always @ (posedge clk)
    begin
        if (RegWrite == 1)
            registers[WriteReg] <= WriteData;
    end

    assign ReadData1 = registers[ReadReg1];
    assign ReadData2 = registers[ReadReg2];

endmodule
`endif
```

SignExtend.v

```
`ifndef MODULE_SIGNEXTEND
`define MODULE_SIGNEXTEND
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/11/01 16:52:09
// Design Name:
// Module Name: SignExtend
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module SignExtend(DataIn,DataOut);
    input [15:0] DataIn;
    output [31:0] DataOut;

    assign DataOut[31:16] = (DataIn[15] == 1) ? 16'b1111111111111111 : 16'b0;
    assign DataOut[15:0] = DataIn[15:0];

endmodule
```

```
`endif
```

PC.v

```
`ifndef MODULE_PC
`define MODULE_PC
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/31 22:42:28
// Design Name:
// Module Name: PC
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module PC(PCIn, PCOut, PCHold, clk);
    input [31:0]PCIn;
    input PCHold;
    input clk;
    output [31:0]PCOut;
    reg [31:0]PCOut;

    initial PCOut <= 0;

    always@(posedge clk)
    begin
        if(PCHold == 0) PCOut <= PCIn;
    end

endmodule
`endif
```

instructionMEM.v

```
`ifndef MODULE_INSTRUCTIONMEM
`define MODULE_INSTRUCTIONMEM
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/31 22:24:21
// Design Name:
// Module Name: instructionMEM
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module instructionMEM(address, instruction);
    input [31:0] address;  //instruction addrress
    output reg [31:0] instruction; //get instruction's machine code

    always @(*)
        case (address >> 2)
            'd0:  instruction = 32'b00100000000001000000000000100000; //addi $t0, $zero, 0x20
            'd1:  instruction = 32'b00100000000001001000000000110111; //addi $t1, $zero, 0x37
            'd2:  instruction = 32'b00000001000010011000000000100100; //and $s0, $t0, $t1
            'd3:  instruction = 32'b00000001000010011000000000100101; //or $s0, $t0, $t1
            'd4:  instruction = 32'b10101100000100000000000000000100; //sw $s0, 4($zero)
            'd5:  instruction = 32'b10101100000001000000000000001000; //sw $t0, 8($zero)
```

```
            'd6:  instruction = 32'b00000001000010011000100000100000; //add $s1, $t0, $t1
            'd7:  instruction = 32'b00000001000010011001000000100010; //sub $s2, $t0, $t1
            'd8:  instruction = 32'b00010010001100100000000000001001; //beq $s1, $s2, error0
            'd9:  instruction = 32'b10001100000100010000000000000100; //lw $s1, 4($zero)
            'd10: instruction = 32'b00110010001100100000000001001000; //andi $s2, $s1, 0x48
            'd11: instruction = 32'b00010010001100100000000000001001; //beq $s1, $s2, error1
            'd12: instruction = 32'b10001100000100110000000000001000; //lw $s3, 8($zero)
            'd13: instruction = 32'b00010010000100110000000000001010; //beq $s0, $s3, error2
            'd14: instruction = 32'b00000010010100011010000000101010; //slt $s4, $s2, $s1 (Last)
            'd15: instruction = 32'b00010010100000000000000000001111; //beq $s4, $0, NEXT
            'd16: instruction = 32'b00000010001000001001000000100000; //add $s2, $s1, $0
            'd17: instruction = 32'b00001000000000000000000000001110; //j Last
            'd31: instruction = 32'b00010100000101000000000000001111; //bne $s4, $0, EXIT
            'd32: instruction = 32'b00010110010101000000000000001111; //bne $s4, $s2, EXIT
            default: instruction <= 32'b00100000000010000000000000100000; // (Exit)
        endcase

endmodule
`endif
```

Mux_32.v

```
`ifndef MODULE_MUX_32
`define MODULE_MUX_32
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/31 22:40:15
// Design Name:
// Module Name: Mux_32
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Mux_32(A, B, ctrl, out);
    input [31:0] A;
    input [31:0] B;
    input ctrl;
    output [31:0] out;
    assign out = (ctrl == 1'b0) ? A: B;
endmodule
`endif
```

Mux_5.v

```
`ifndef MODULE_MUX_5
`define MODULE_MUX_5
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/10/31 22:38:21
// Design Name:
// Module Name: Mux_5
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Mux_5(A, B, ctrl, out);
    input [4:0] A;
```

```
    input [4:0] B;
    input ctrl;
    output [4:0] out;
    assign out = (ctrl == 1'b0) ? A : B;
endmodule
`endif
```

Single_Clock_TestBench.v

```
`timescale 1ns / 1ps
`include "Single_Cycle.v"
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2020/11/03 19:46:14
// Design Name:
// Module Name: Single_Clock_TestBench
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Single_Clock_TestBench;
    integer i = 0; // Count the time
    integer j = 0; // Count the instruction number
    reg clk;
    Single_Cycle uut(.clk(clk));

    initial begin
    #0  clk = 0; $display("Start Simulation of Single Cycle! The result is:");
    #920 $stop;
    end

    always #20 begin
        $display("Time = %d, CLK = %d, PC = 0x%H", i, clk, uut.PCOut);
        $display("Instruction: %H", uut.instruction);
        $display("[$s0] = 0x%H, [$s1] = 0x%H, [$s2] = 0x%H", uut.RFile.registers[16], uut.RFile.registers[17],
uut.RFile.registers[18]);
        $display("[$s3] = 0x%H, [$s4] = 0x%H, [$t0] = 0x%H", uut.RFile.registers[19], uut.RFile.registers[20],
uut.RFile.registers[8]);
        $display("[$t1] = 0x%H, [$t2] = 0x%H, [$t3] = 0x%H", uut.RFile.registers[9], uut.RFile.registers[10],
uut.RFile.registers[11]);
        $display("-------------------------------------------------------------");
        clk = ~clk;
        i = i + 20;
        if(i == 900) $display("Exit! Simulation Done!");
    end
endmodule
```

Appendix B: Simulation Result

```
Start Simulation of Single Cycle! The result is:
Time =           0, CLK = 0, PC = 0x00000000
Instruction: 20080020
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000000
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =          20, CLK = 1, PC = 0x00000004
Instruction: 20090037
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =          40, CLK = 0, PC = 0x00000004
Instruction: 20090037
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000000, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
```

```
Time =          60, CLK = 1, PC = 0x00000008
Instruction: 01098024
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =          80, CLK = 0, PC = 0x00000008
Instruction: 01098024
[$s0] = 0x00000000, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         100, CLK = 1, PC = 0x0000000c
Instruction: 01098025
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         120, CLK = 0, PC = 0x0000000c
Instruction: 01098025
[$s0] = 0x00000020, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         140, CLK = 1, PC = 0x00000010
Instruction: ac100004
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         160, CLK = 0, PC = 0x00000010
Instruction: ac100004
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         180, CLK = 1, PC = 0x00000014
Instruction: ac080008
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         200, CLK = 0, PC = 0x00000014
Instruction: ac080008
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         220, CLK = 1, PC = 0x00000018
Instruction: 01098820
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         240, CLK = 0, PC = 0x00000018
Instruction: 01098820
[$s0] = 0x00000037, [$s1] = 0x00000000, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         260, CLK = 1, PC = 0x0000001c
Instruction: 01099022
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         280, CLK = 0, PC = 0x0000001c
Instruction: 01099022
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         300, CLK = 1, PC = 0x00000020
Instruction: 12320009
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         320, CLK = 0, PC = 0x00000020
Instruction: 12320009
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
-------------------------------------------------------------
Time =         340, CLK = 1, PC = 0x00000024
Instruction: 8c110004
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffff9
```

```
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         360, CLK = 0, PC = 0x00000024
Instruction: 8c110004
[$s0] = 0x00000037, [$s1] = 0x00000057, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         380, CLK = 1, PC = 0x00000028
Instruction: 32320048
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         400, CLK = 0, PC = 0x00000028
Instruction: 32320048
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0xfffffff9
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         420, CLK = 1, PC = 0x0000002c
Instruction: 12320009
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         440, CLK = 0, PC = 0x0000002c
Instruction: 12320009
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         460, CLK = 1, PC = 0x00000030
Instruction: 8c130008
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         480, CLK = 0, PC = 0x00000030
Instruction: 8c130008
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000000, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         500, CLK = 1, PC = 0x00000034
Instruction: 1213000a
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         520, CLK = 0, PC = 0x00000034
Instruction: 1213000a
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         540, CLK = 1, PC = 0x00000038
Instruction: 0251a02a
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         560, CLK = 0, PC = 0x00000038
Instruction: 0251a02a
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         580, CLK = 1, PC = 0x0000003c
Instruction: 1280000f
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         600, CLK = 0, PC = 0x0000003c
Instruction: 1280000f
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
Time =         620, CLK = 1, PC = 0x00000040
Instruction: 02209020
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
------------------------------------------------------------
```

```
Time =         640, CLK = 0, PC = 0x00000040
Instruction: 02209020
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000000
[$s3] = 0x00000020, [$s4] = 0x00000001, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         660, CLK = 1, PC = 0x00000044
Instruction: 0800000e
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         680, CLK = 0, PC = 0x00000044
Instruction: 0800000e
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         700, CLK = 1, PC = 0x00000038
Instruction: 0251a02a
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         720, CLK = 0, PC = 0x00000038
Instruction: 0251a02a
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000001, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         740, CLK = 1, PC = 0x0000003c
Instruction: 1280000f
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         760, CLK = 0, PC = 0x0000003c
Instruction: 1280000f
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         780, CLK = 1, PC = 0x0000007c
Instruction: 1414000f
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         800, CLK = 0, PC = 0x0000007c
Instruction: 1414000f
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         820, CLK = 1, PC = 0x00000080
Instruction: 1654000f
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         840, CLK = 0, PC = 0x00000080
Instruction: 1654000f
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         860, CLK = 1, PC = 0x000000c0
Instruction: 20080020
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Time =         880, CLK = 0, PC = 0x000000c0
Instruction: 20080020
[$s0] = 0x00000037, [$s1] = 0x00000037, [$s2] = 0x00000037
[$s3] = 0x00000020, [$s4] = 0x00000000, [$t0] = 0x00000020
[$t1] = 0x00000037, [$t2] = 0x00000000, [$t3] = 0x00000000
----------------------------------------------------------
Exit! Simulation Done!
```