《计算机组成原理》实验报告

年级、专业、班级	2021 级计算机科学与技术(卓越)02 班	**姓名**	文红兵
实验题目	实验三简易单周期 CPU 实验		
实验时间	2023 年 5 月 6 日	**实验地点**	DS1410
实验成绩	优秀/良好/中等	**实验性质**	□验证性 ☑设计性 □综合性

教师评价：

□算法/实验过程正确； □源程序/实验内容提交； □程序结构/实验步骤合理;

□实验结果正确； □语法、语义正确； □报告规范;

其他:

评价教师: 钟将

实验目的

(1)掌握不同类型指令在数据通路中的执行路径。

(2)掌握 Vivado 仿真方式。

报告完成时间: 2023 年 5 月 6 日

1 实验内容

阅读实验原理实现以下模块：

(1) Datapath，其中主要包含 alu(实验一已完成)，PC(实验二已完成)，adder、mux2、signext、sl2(其中 adder、mux2 数字逻辑课程已实现，signext、sl2 参见实验原理)，
(2) Controller(实验二已完成)，其中包含两部分，分别为 main_decoder，alu_decoder。
(3) 指令存储器 inst_mem(Single Port Ram)，数据存储器 data_mem(Single Port Ram)；使用 Block Memory Generator IP 构造指令，注意考虑 PC 地址位数统一。(参考实验二)
(4) 参照实验原理，将上述模块依指令执行顺序连接。实验给出 top 文件，需兼容 top 文件端口设定。
(5) 实验给出仿真程序，最终以仿真输出结果判断是否成功实现要求指令。

2 实验设计

2.1 数据通路

2.1.1 功能描述

数据通路的具体功能为根据指令进行取指、译码、执行、访存和写回操作，实现 CPU 基本的数据处理流程。

2.1.2 接口定义

3 实验过程记录

3.1 问题 1:ALU 编码不一致

问题描述：设定的 ALUControl 编码的实现与假设不一致。

解决方案：修改 ALU 模块。

3.2 问题 2:Data Memory 结果异常

问题描述：Data Memory 无法写入数据，导致读出数据始终为 0。

解决方案：经过不断调试，发现 data memory 的使能端口理解错误，理解成为了读使能端口，经过修正，问题解决。

表 1: datapath 模块接口定义

信号名	方向	位宽	功能描述
CLK	Input	1-bit	时钟信号
RST	Input	1-bit	复位信号
instr	Input	32-bit	输入指令
ReadData	Input	32-bit	data memory 读出的数据
ALUControl	Input	3-bit	ALU 控制信号
PCSrc	Input	1-bit	启用分支地址信号
MemtoReg	Input	1-bit	写入寄存器堆数据的选择信号
ALUSrc	Input	1-bit	启用立即数信号
RegDst	Input	1-bit	启用 rd 寄存器信号
RegWrite	Input	1-bit	寄存器堆写使能信号
Jump	Input	1-bit	无条件跳转指令
MemWrite	Input	1-bit	data memory 的写使能信号
MemRead	Input	1-bit	data memory 的读使能信号
ALUResult	Output	32-bit	ALU 计算结果
rd2	Output	32-bit	Register File 第二个读出的数据
PC	Output	32-bit	地址
Zero	Output	1-bit	ALU 输出的零信号

4 实验结果及分析

4.1 仿真图

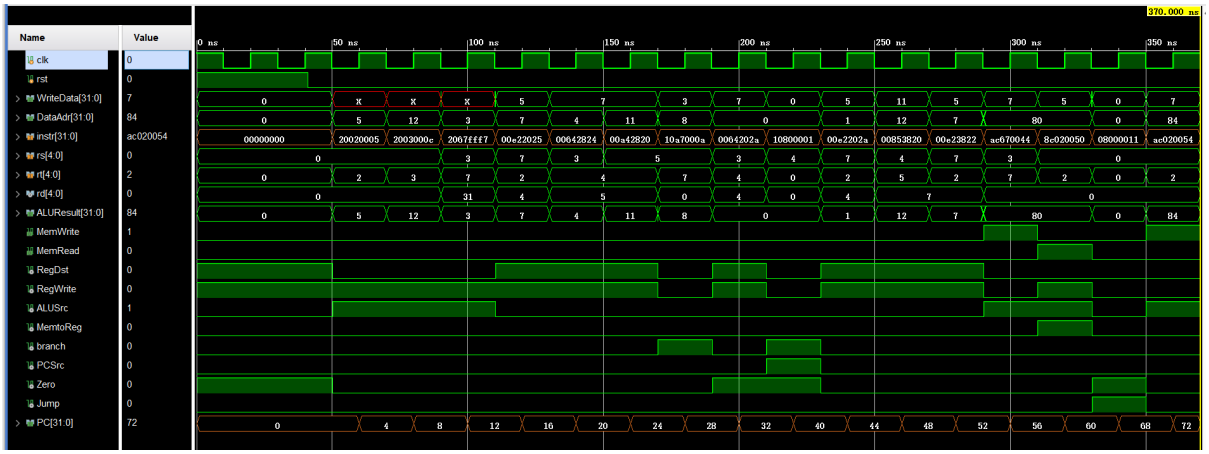

图 1: 仿真图

4.2 控制台输出

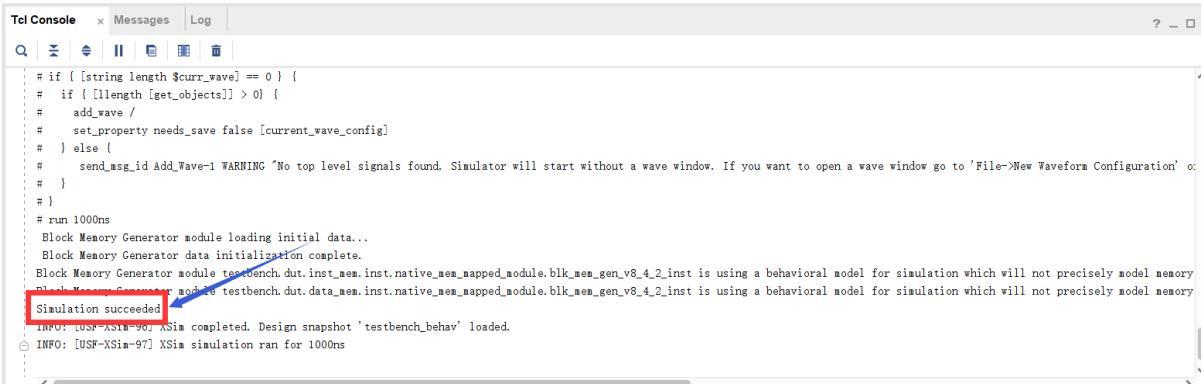

图 2: 控制台输出图

A Datapath 代码

```
module datapath(
        input wire CLK, RST,
        input wire[31:0] instr, // 输入指令
        input wire[31:0] ReadData, // data memory的读出数据
        input wire [2:0] ALUControl, // ALU的控制信号
        input wire PCSrc , // 启用分支地址信号
        input wire MemtoReg , // 写入寄存器堆数据的选择信号, 1选择ReadData
        input wire ALUSrc , // 启用立即数信号
        input wire RegDst , // 启用rd寄存器信号
        input wire RegWrite , // 寄存器堆写使能信号
```

```
    input wire Jump , // 无条件跳转指令
    input wire MemWrite, // 控制信号, data memory的写使能信号
    input wire MemRead, // 控制信号, data memory的读使能信号
    output wire[31:0] ALUResult, // ALU计算结果
    output wire[31:0] rd2, // Register File第二个读出的数据
    output wire[31:0] PC, // 地址
    output wire Zero // ALU输出的零信号
);

// wire
wire[31:0] PCPLUS4 ;// pc + 4
wire[31:0] SignImm ; // 立即数符号扩展的数字
wire[31:0] rd1 ; // Register File第一个读出的数据

// PCSrc and Jump
wire[31:0] PCSeclected ;
wire[31:0] PCBranch ;
assign PCBranch = PCPLUS4 + {SignImm[29:0],2'b00} ;
assign PCSeclected = (Jump == 1)? {PCPLUS4[31:28],{instr[25:0], 2'b00}}:
                                  (PCSrc == 1)? PCBranch : PCPLUS4 ;
// pc
pc pc(
    .clk(CLK),
    .reset(RST),
    .d(PCSeclected),
    .q(PC)
);

// pc + 4
assign PCPLUS4 = PC + 4 ;

// RegDst
wire[4:0] WriteReg ; // register file 写入端口的地址选择, 从rs和rt中选
wire[4:0] rd = instr[15:11] ;
assign WriteReg = (RegDst == 1) ? rd : instr[20:16] ;

// MemtoReg
wire[31:0] WriteBackResult ; // 写回数据的结果
assign WriteBackResult = (MemtoReg == 0)? ALUResult : ReadData ;

// regfile
regfile regfile(
    .clk(~CLK), // 时钟
    .we3(RegWrite), // 写入端口的使能信号
    .ra1(instr[25:21]),
    .ra2(instr[20:16]),
    .wa3(WriteReg), // 两个读入端口的地址, 一个写入端口的地址
    .wd3(WriteBackResult), // 写入的数据
    .rd1(rd1),
```

```
        .rd2(rd2)  // 两个端口读出的数据
    );

    // sign_extend
    signExtension signExtension(
        .a(instr[15:0]),
        .y(SignImm)
    );

    // ALUSrc
    wire[31:0] SrcB ; // ALU的第二个操作数
    assign SrcB = (ALUSrc == 1)? SignImm : rd2 ;

    // ALU
    alu alu(
        .a(rd1),
        .b(SrcB),
        .op(ALUControl),
        .s(ALUResult),
        .zero(Zero)
        );

endmodule
```