《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05,03班	姓名	张梓健，任俊璇
实验题目	实验三简易单周期CPU实验		
实验时间	2023 年 4 月 24 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: 冯永

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 5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

数据通路模块，用于连接pc,alu,regfile,sl2,signext等模块

2.1.2 接口定义

表 1: 数据通路接口定义

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
memtoreg	input	1-bit	回写的数据来自于 ALU 计算的结果还是存储器读取的数据
pcsrc	input	1-bit	程序计数器(PC)的选择信号
alusrc	input	1-bit	ALU的第二个操作数的选择信号
regdst	input	1-bit	寄存器目标地址的选择信号
regwrite	input	1-bit	是否写入寄存器的使能信号
jump	input	1-bit	是否进行跳转的使能信号
alucontrol	input	3-bit	ALU运算类型的控制信号
zero	output	1-bit	ALU计算结果是否为0的标志位
pc	output	32-bit	程序计数器的值
instr	input	32-bit	指令存储器读取的指令
aluout	output	32-bit	ALU计算结果的输出
writedata	output	32-bit	写入寄存器的数据
readdata	input	32-bit	从寄存器读取的数据
pc_in	wire	32-bit	程序计数器输入信号
pcplus4	wire	32-bit	PC加4的结果
writereg	wire	5-bit	写入寄存器的目标寄存器地址
result	wire	32-bit	存储器读取的数据或ALU计算结果
rd1	wire	32-bit	第一个操作数(从寄存器读取)
extout	wire	32-bit	符号扩展的输出
srcB	wire	32-bit	ALU的第二个操作数
slout	wire	32-bit	移位后的输出

2.1.3 逻辑控制

该模块是一个基本的 MIPS 处理器数据通路，实现了 MIPS 指令的基本执行功能。它包括一个程序计数器(PC)、一个指令存储器、一个寄存器文件、一个 ALU、一个符号扩展模块、一个移位模块、一个加法器、以及一些复用器和选择器等基本组件。该模块根据输入的指令和控制信号，完成对应的指令执行功能

2.2 控制器

2.2.1 功能描述

接收存储器传来的32位指令，调用Main decoder和Alu decoder，生成memtoreg,memwrite, branch,alusrc,regdst,regw
号。

2.2.2 接口定义

表 2: 控制器接口定义

信号名	方向	位宽	功能描述
op	input	6-bit	指令op码。
funct	input	6-bit	指令funct码。
zero	input	-bit	ALU输出是否为0
memtoreg	output	1-bit	回写的数据来自于 ALU 计算的结果或是存储器读取的数据。
memwrite	output	1-bit	是否需要写数据存储器。
alusrc	output	1-bit	送入 ALU B 端口的值是立即数的 32 位扩展还是寄存器堆读取的值。
regdst	output	1-bit	写入寄存器堆的地址是 rt 还是 rd,0 为 rt,1 为 rd。
regwrite	output	1-bit	是否需要写寄存器堆。
jump	output	1-bit	是否为 jump 指令。
pcsrc	output	1-bit	程序计数器(PC)输入选择是 PC+4 还是分支跳转地址
alucontrol	output	1-bit	ALU 控制信号，代表不同的运算类型。

2.2.3 逻辑控制

controller模块接收指令的op码和funct码，先调用Main decoder，根据op得到memtoreg,memwrite,branch,alusrc,regdst,regwrite,jump,aluop，然后调用Alu decoder，根据aluop和funct得到alucontrol。

3 实验过程记录

3.1 实验过程

导入已有模块：从实验一中导入 alu 模块，从实验二中导入 PC、Controller 模块，从数字逻辑实验中导入多路选择器、加法器模块，从本次实验中导入寄存器堆、移位，有符号扩展、顶层模块和仿真文件。

构造inst_mem和data_mem：使用 Block Memory，导入 coe 文件构造inst_mem和data_mem的IP。

设计实现mux2模块：根据实验要求设计mux2模块，实验二选一选择器的功能。

设计实现datapath模块：根据实验要求设计datapath模块，连接各底层模块实现CPU相应功能。

运行仿真：运行仿真文件，测试仿真结果是否正确。

3.2 遇到的问题

问题一：从实验二中导入的controller模块与本次实验要求有所不同，实验二中controller模块输出的branch信号，在本次实验中需要与从datapath模块输出的zero信号进行相与操作得到pcsrc信号，并输出。

问题二：根据RTL图发现判断目的寄存器的二选一选择器的输入值不对，应该输入指令的[20:16]和[15:11],但错误输入为了[25:21]和[20:16]。

问题三：运行仿真发现寄存器堆32个寄存器值一直为XXXXXXXX。

问题四：运行仿真发现结果不正确，多变量值为红色X值。

3.3 解决方案

解决一：修改controller模块，增加输入zero信号，assign pcsrc = branch & zero，删除输出branch信号，增加输出pcsrc信号。

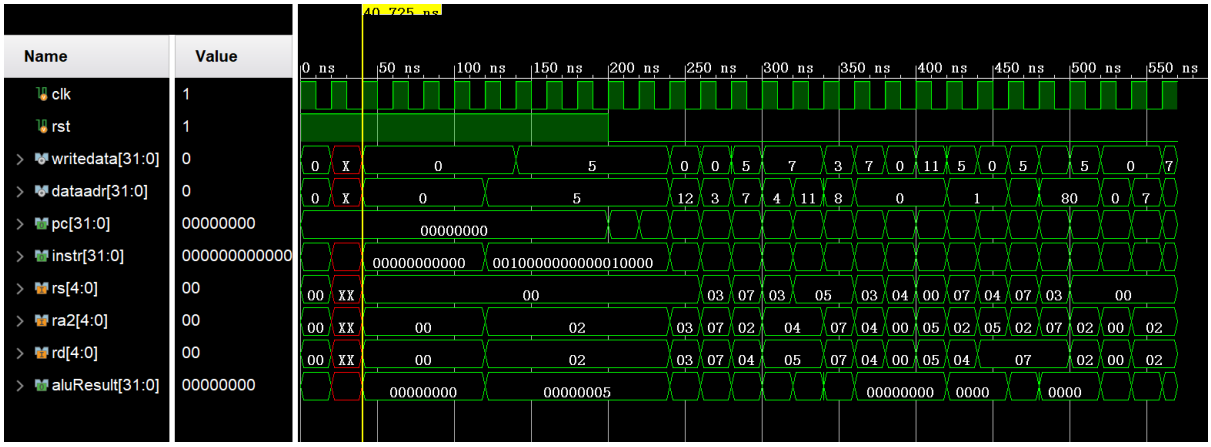
解决二：在datapath模块中修正该二选一选择器的输入。

解决三：在regfile模块中对寄存器堆进行初始化，将32个寄存器全部初始化为0。

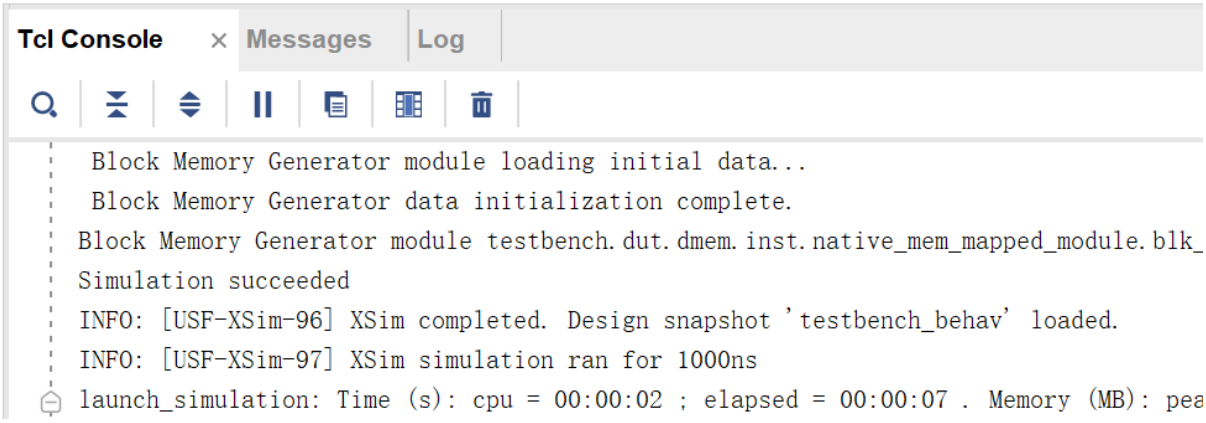
解决四：经过漫长的不断增加测试变量进行仿真，终于找出问题所在，原来是从实验一导入的alu模块中时序控制逻辑为always @（aluControl），当aluControl的值没有发生改变时，aluResult无法被赋值，导致其值为X，再连锁反应导致后续变量都为X，而将控制逻辑改为always @(*)后，问题解决，仿真运行结果正确。

4 实验结果及分析

4.1 仿真波形图



4.2 控制台输出图



A Datapath代码

```
module datapath(clk,rst,memtoreg,pcsrc,alusrc,
    regdst,regwrite,jump,alucontrol,zero,pc,instr,aluout,writedata,readdata);
    input clk,rst;
    input memtoreg,pcsrc,alusrc,regdst,regwrite,jump;
    input [2:0]alucontrol;
    output zero;
    output [31:0]pc;
    input [31:0]instr;
    output[31:0] aluout;
    output[31:0] writedata;
    input [31:0]readdata;
    wire[31:0]pc_in;
    pc pc_0(clk,rst,pc_in,pc);
    wire[31:0]pcplus4;
    pc_add pa_0(pc,pcplus4);
    wire[4:0]writereg;
    mux2x5 m25_0(instr[20:16],instr[15:11],regdst,writereg);
    wire[31:0]result;
    mux2x32 m32_0(aluout,readdata,memtoreg,result);
    wire[31:0]rd1;
    regfile rf(clk,regwrite,instr[25:21],instr[20:16],writereg,result,rd1,
        writedata);
    wire[31:0]extout;
    wire[31:0]srcB;
    mux2x32 m32_1(writedata,extout,alusrc,srcB);
    alu alu_0(rd1,srcB,alucontrol,aluout,zero);
    signext ext_0(instr[15:0],extout);
    wire[31:0]slout;
    sl2 sl_0(extout,slout);
    wire [31:0]pcbranch;
```

```
    adder a0(slout,pcplus4,pcbranch);
    mux2x32 pcmux(pcplus4,pcbranch,pcsrc,pc_in);
endmodule
```