计算机系统结构实验 Lab4 实验报告

- 姓名：夏鸿驰
- 学号：520021910965

- 日期：2022年4月26日

目录

1. 实验概述

1.1 实验名称

- 简单的类 MIPS 单周期处理器存储部件的设计与实现（二）

1.2 实验目的

1. 理解寄存器、数据存储器、有符号扩展单元的 IO 定义
2. Registers 的设计实现
3. Data Memory 的设计实现
4. 有符号扩展部件的实现
5. 对功能模块进行仿真

2. 实验步骤

2.1 新建工程

2.1.1 用与前两个lab一样的步骤创建一个名为lab04的工程即可

2.2 构建寄存器组模块

2.2.1 模块描述

寄存器是指令操作的主要对象，MIPS 中一共有 32 个 32 位的寄存器，用作数据的缓存。
相应的I/O定义为：

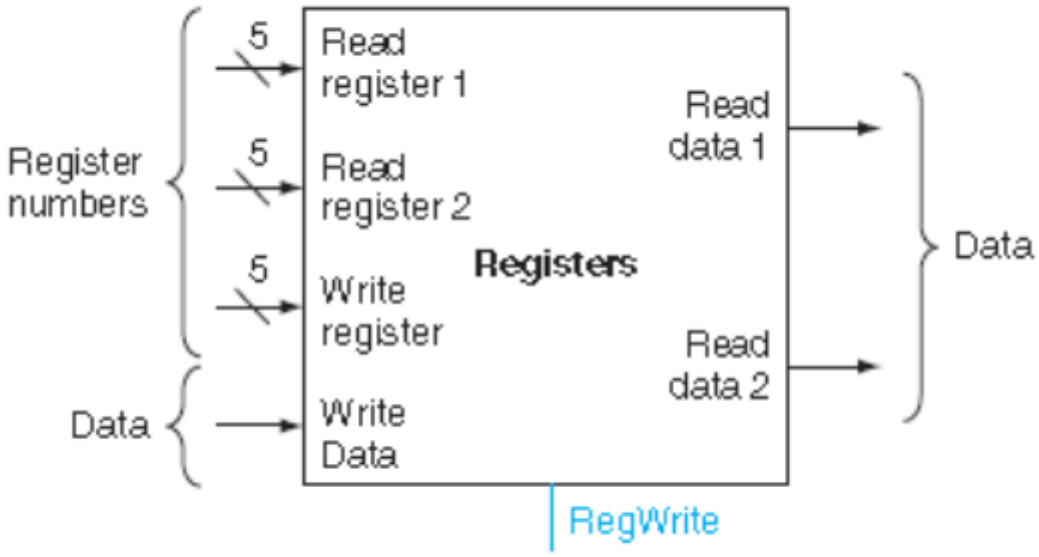


图 1. 寄存器模块的 IO 定义

2.2.2 编写寄存器组功能

新建source文件Registers.v，并输入以下verilog代码：

```
`timescale 1ns / 1ps
module Registers(
    input Clk,
    input [25:21] readReg1,
    input [20:16] readReg2,
    input [4:0] writeReg,
    input [31:0] writeData,
    input regWrite,
    output [31:0] readData1,
    output [31:0] readData2
    );

    reg [31:0] regFile[31:0];
    reg [31:0] rd1;
    reg [31:0] rd2;

    always @(readReg1 or readReg2 or writeReg)
        begin
            rd1 = regFile[readReg1];
            rd2 = regFile[readReg2];
        end

    assign readData1 = rd1;
    assign readData2 = rd2;

    always @(negedge Clk)
        begin
            if(regWrite) regFile[writeReg] = writeData;
```

```
        end


endmodule
```

2.2.3 寄存器组功能仿真

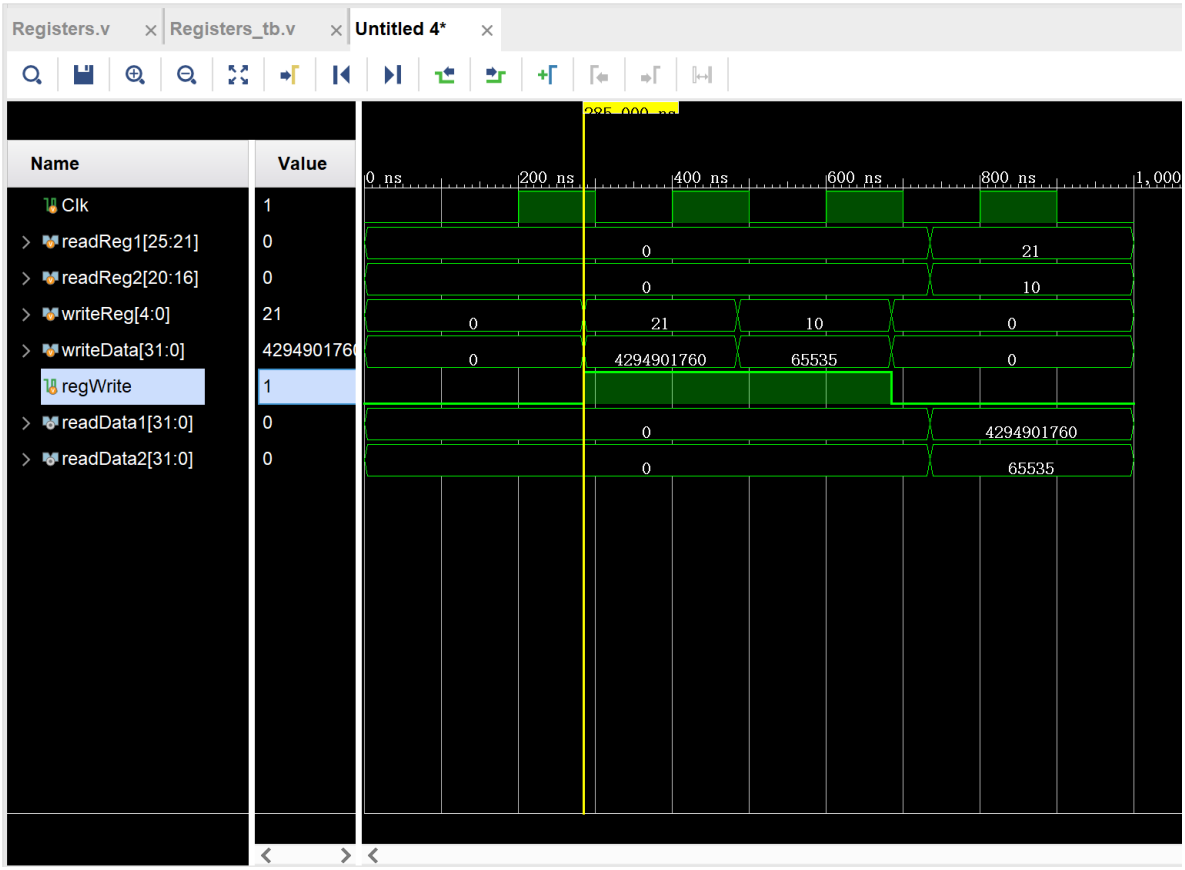
新建激励文件Registers_tb.v，并输入以下verilog代码：

```
`timescale 1ns / 1ps
module Registers_tb(

    );
    reg Clk;
    reg [25:21] readReg1;
    reg [20:16] readReg2;
    reg [4:0] writeReg;
    reg [31:0] writeData;
    reg regWrite;
    wire [31:0] readData1;
    wire [31:0] readData2;
    integer i;

    Registers regs(
        .Clk(Clk),
        .readReg1(readReg1),
        .readReg2(readReg2),
        .writeReg(writeReg),
        .writeData(writeData),
        .regWrite(regWrite),
        .readData1(readData1),
        .readData2(readData2)
    );

    always #100 if($time!=100) Clk = !Clk;
    initial begin
        Clk = 0;
        readReg1 = 0;
        readReg2 = 0;
        writeReg = 0;
        writeData = 0;
        regWrite = 0;
        for(i = 0;i < 32;i = i + 1)
        begin
            regs.regFile[i] = 0;
        end
        #285;
        regWrite = 1'b1;
        writeReg = 5'b10101;
        writeData = 32'b11111111111111110000000000000000;
        #200;
        writeReg = 5'b01010;
        writeData = 32'b00000000000000001111111111111111;
        #200;
        regWrite = 1'b0;
        writeReg = 5'b00000;
        writeData = 32'b00000000000000000000000000000000;
        #50;
```

```
50          readReg1 = 5'b10101;
51          readReg2 = 5'b01010;
52
53      end
54  endmodule
```

2.2.4 获取仿真图像

经过仿真，得到以下图像：



2.2.5 仿真结果分析

观察图像，可知仿真结果与逻辑相一致，目标模块达到了预想的结果，模块构建成功。

2.3 构建数据存储器模块

2.3.1 模块描述

Data memory 是用来存储运行完成的数据，或者初始化的数据。内存模块的编写与 register 类似，由于写数据也要考虑信号同步，因此也需要时钟。

相应的I/O定义为：

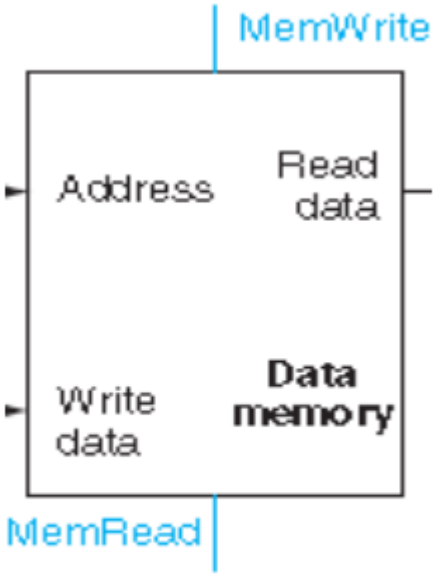


图 2. 内存模块的 IO 定义

2.3.2 编写数据存储器功能

新建source文件DataMemory.v，并输入以下verilog代码：

```
`timescale 1ns / 1ps
module DataMemory(
    input Clk,
    input [31:0] Address,
    input [31:0] WriteData,
    input MemWrite,
    input MemRead,
    output [31:0] ReadData
    );

    reg [31:0] MemFile[0:63];
    reg [31:0] rd;

    always @ (Address)
        begin
            rd = 0;
            if(MemRead)
                rd = MemFile[Address];
        end

    assign ReadData = rd;

    always @ (negedge Clk)
        begin
            if(MemWrite)
                MemFile[Address] = WriteData;
        end
endmodule
```

2.3.3 数据存储器功能仿真

新建激励文件DataMemory_tb.v，并输入以下verilog代码：

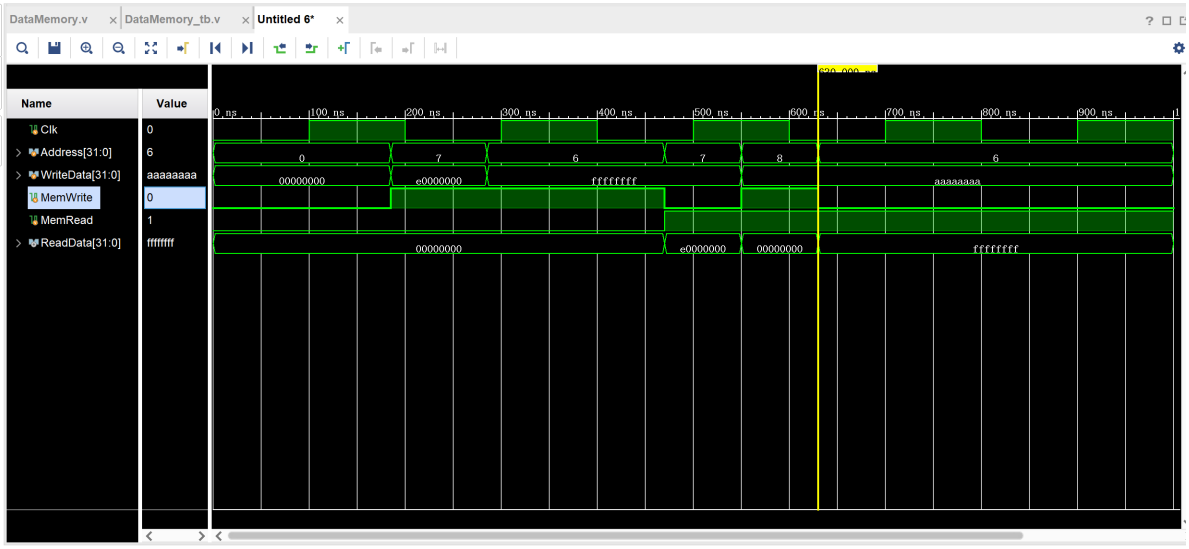
```
`timescale 1ns / 1ps
```

```
module DataMemory_tb(

    );
    reg Clk;
    reg [31:0] Address;
    reg [31:0] WriteData;
    reg MemWrite;
    reg MemRead;
    wire [31:0] ReadData;
    integer i;

    DataMemory dm(
        .Clk(Clk),
        .Address(Address),
        .WriteData(WriteData),
        .MemWrite(MemWrite),
        .MemRead(MemRead),
        .ReadData(ReadData)
    );

    always #100 Clk = !Clk;

    initial begin
        Clk = 0;
        Address = 0;
        WriteData = 0;
        MemWrite = 0;
        MemRead = 0;
        for(i = 0; i < 64; i = i + 1)
        begin
            dm.MemFile[i] = 0;
        end

        #185;
        //185ns
        MemWrite = 1'b1;
        Address = 32'b00000000000000000000000000000111;
        WriteData = 32'b11100000000000000000000000000000;

        #100;
        //285ns
        MemWrite = 1'b1;
        Address = 32'b00000000000000000000000000000110;
        WriteData = 32'hffffffff;

        #185;
        //470ns
        Address = 7;
        MemRead = 1'b1;
        MemWrite = 1'b0;

        #80;
        //550ns
        MemWrite = 1;
        Address = 8;
        WriteData = 32'haaaaaaaa;

        #80;
```

```
        //630ns
        Address = 6;
        MemWrite = 0;
        MemRead = 1;
    end
endmodule
```

2.3.4 获取仿真图像

经过仿真，得到以下图像：



2.3.5 仿真结果分析

观察图像，可知仿真结果与逻辑相一致，目标模块达到了预想的结果，模块构建成功。

2.4 构建带符号扩展模块

2.4.1 模块描述

将 16 位有符号数扩展为 32 位有符号数。

相应的I/O定义为：

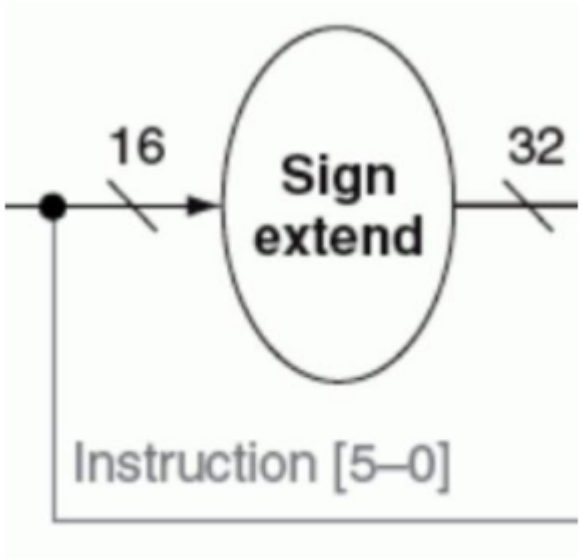


图 3. 带符合扩展单元

2.4.2 编写带符号扩展模块功能

新建source文件SignExt.v，并输入以下verilog代码：

```
`timescale 1ns / 1ps
module SignExt(
    input [15:0] Instr,
    output [31:0] Data
    );
    assign Data[31:0] ={{16{Instr[15]}},Instr[15:0]};
endmodule
```

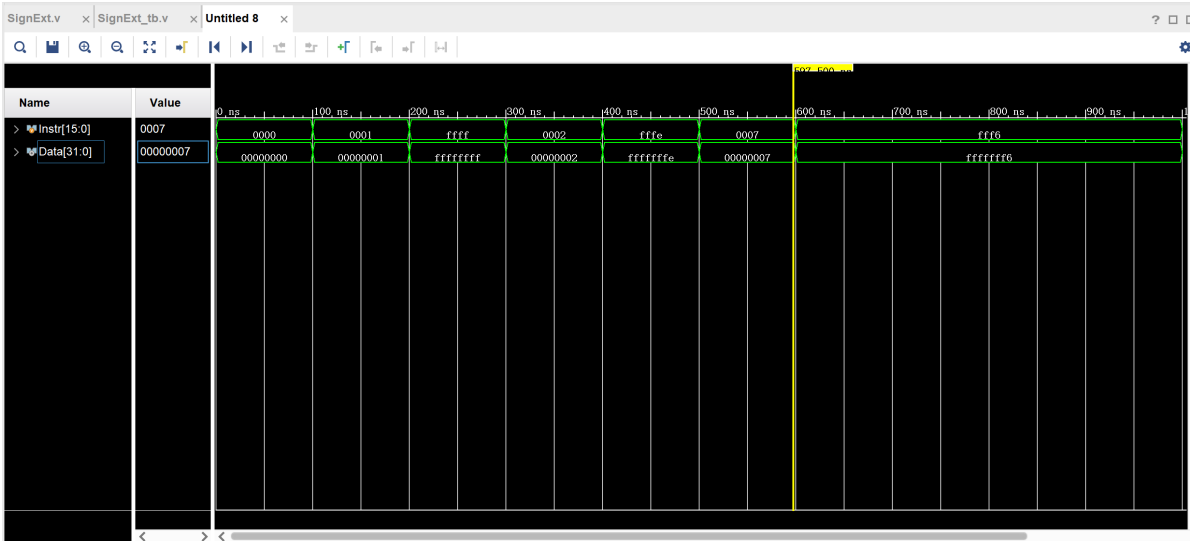
2.4.3 带符号扩展模块功能仿真

新建激励文件SignExt_tb.v，并输入以下verilog代码：

```
`timescale 1ns / 1ps
module SignExt_tb(

    );
    reg [15:0] Instr;
    wire [31:0] Data;

    SignExt se(
        .Instr(Instr),
        .Data(Data)
    );

    initial begin
        Instr = 0;
        #100;
        Instr = 1;
        #100;
        Instr = -1;
        #100;
        Instr = 2;
        #100;
        Instr = -2;
        #100;
        Instr = 7;
        #100;
        Instr = -10;
    end
endmodule
```

2.4.4 获取仿真图像

经过仿真，得到以下图像：



2.4.5 仿真结果分析

观察图像，可知仿真结果与逻辑相一致，目标模块达到了预想的结果，模块构建成功。

3. 实验心得

3.1 实验重难点

- 本实验的重难点在于通过lab1-lab3一路下来积累的经验独自完成寄存器组模块，数据存储器模块和带符号扩展模块的编写及独立完成相应激励代码的编写和调试。此外还有对于在激励文件中初始化的理解和应用。

3.2 实验感想

- 在本次实验过程中，从编写寄存器组模块，数据存储器模块和带符号扩展模块一路下来的实验经历让我感到一个个挑战接踵而至，让我充满动力，也给了我一种最终完成任务的成就感。此外，在实验中遇到的困难都会在微信群里得到老师及时的反馈，所以要在这里感谢老师的付出。

4. 参考资料

[1] 2022计算机系统结构实验指导书lab4