上海交通大學

《计算机组成与系统结构》实验 4 报告

学生姓名　　：梁展诚 Zhin Sheng Leong

学生学院（系）　：船舶海洋与建筑工程学院（土木工程）

学生学号　　：517010990022

实验名称　　：简单的类 MIPS 单周期处理器部件实现——寄存器与存储器

目录

简单的类 *MIPS* 单周期处理器部件实现 ——寄存器与存储器

1. 实验目的

- 理解 *CPU* 寄存器与存储器的操作原理
- 实现 *Register*
- 实现 *Data Memory*
- 实现有符号数的扩展
- 使用 *Verilog HDL* 进行逻辑设计
- 使用功能仿真

2. 实验原理

2.1 寄存器 *Registers* 的设计

寄存器是 *CPU* 内部用来存放数据的一些小型存储区域，用来暂时存放参与运算的数据和运算结果，同时也是指令操作的主要对象。*MIPS* 中以共有 32 个 通用的 32 位寄存器。

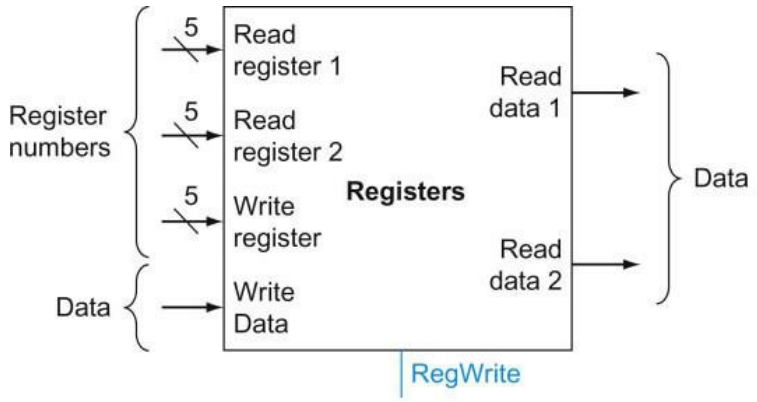

图1：*MIPS* 寄存器模块

一个简单 *MIPS* 寄存器模块主要包含的输入、输出信号及其含义如下：

信号	具体含义
Read Register 1	访问第 1 目标寄存器的选择
Read Register 2	访问第 2 目标寄存器的选择

Write Register	写有效时目标寄存器的选择
Write Data	欲写入目标寄存器的数据
Reg Write	判断能否写入寄存器的信号，高电位时为写有效
Read Data 1	从第 1 目标寄存器取出的数据
Read Data 2	从第 2 目标寄存器取出的数据

表 1：*MIPS* 寄存器模块的控制信号及其含义

2.2 内存单元模块 *Memory* 的设计

内存是外存与 CPU 进行沟通的桥梁，计算机中所有程序的运行都在内存中进行。只要计算机开始运行，操作系统就会把需要运算的数据从内存调到 CPU 中进行运算。当运算完成，CPU 将结果传送出来。

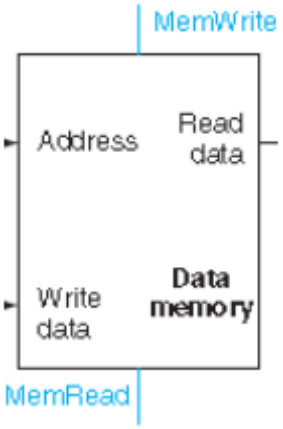

图 2：*MIPS* 内存单元模块

一个简单 *MIPS* 内存单元模块主要包含的输入、输出信号及其含义如下：

信号	具体含义
Address	为要访问或写入的内存地址
Write Data	欲写入内存的数据
MemRead	判断能否访问内存的信号，高电位时为访问有效
MemWrite	判断能否写入内存的信号，高电位时为写有效
Read Data	从相关内存地址中取出的数据

表 2：*MIPS* 内存单元模块的真值表

2.3 带符号数扩展单元的设计

MIPS 处理器在对指令进行译码时，有时需要将 16 位的带符号数扩展成 32 位的带符号数，以便能够传送至运算控制单元 *ALU*，与其他的 32 位数进行相关的运算和操作。若给定的符号数为正数，其符号位为 0，反之为 1，而带符号的扩展只需在前面补足符号即可。

3. 实验设计

3.1 寄存器 *Registers* 的实现

寄存器根据相应的输入信号决定其访问或写入的寄存器。由于不确定 *WriteReg*,*WriteData*,*RegWrite* 信号的先后次序，可通过 *Verilog* 中时钟（此处名为 *clock*）的下降沿作为写操作的同步信号，实现其功能，代码如下：

```
module Registers(
    input clock,
    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];
    integer temp1,temp2,temp3;

    always @(readReg1 or readReg2 or writeReg)
    begin
        temp1 = readReg1;
        temp2 = readReg2;
        temp3 = writeReg;
    end

    always @(negedge clock)
    begin
        if (RegWrite == 1)
            regFile[temp3] = writeData;
    end

    assign ReadData1 = regFile[temp1];
    assign ReadData2 = regFile[temp2];

endmodule
```

3.2 内存单元模块 *Memory* 的实现

通过寄存器的方式实现内存单元，一般实际应用中内存较大，此处设计的内存单元模块只能存放 64 个 32 位的数据。内存单元模块与上述寄存器模块相似，由于写入内存时也需要考虑信号同步，也通过 *Verilog* 中时钟（此处名为 *clock*）的下降沿作为写操作的同步信号，实现其功能，代码如下：

```
module Memory(
    input clock,
    input [31:0] Address,
    input [31:0] WriteData,
    input memWrite,
    input memRead,
    output [31:0] readData
    );

    reg [31:0] memFile [0:63];
    integer temp;
    reg [31:0] temp2;

    always @(Address)
    begin
        temp = Address;
        if (memRead)
            temp2 = memFile[temp];
        else
            temp2 = 32'bxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx;
    end

    always @(negedge clock)
    begin
        if (memWrite)
            memFile[temp] = WriteData;
    end

    assign readData = temp2;

endmodule
```

3.3 带符号数扩展单元的实现

带符号数扩展单元只需要根据带符号数的正负值即可判定其符号位的值，可通过 *Verilog* 中如下的代码实现其功能：

```
module signext(
    input [15:0] inst,
    output [31:0] data
    );

    integer temp;
```

```
7.      reg [31:0] result;
8.
9.      always @(inst)
10.     begin
11.         if (inst>=16'b1000000000000000)
12.             result = {16'b1111111111111111,inst};
13.         else
14.             result = {16'b0000000000000000,inst};
15.     end
16.
17.     assign data = result;
18.
19. endmodule
```

4. 结果仿真

对于上述的 3 个单元，采用软件仿真的方式进行校核测试。通过 *Verilog* 编写相应的激励文件，进行仿真，观察其结果，与预期的结果进行对比。

4.1 寄存器 *Registers* 的仿真

对寄存器 *Registers* 进行表 3 的测试，得出如下的仿真波形图：

图 3：寄存器 *Registers* 的仿真结果

时间 *ns*	*RegWrite*	*writeReg*	*writeData*	*readReg1*	*readReg2*
200	1	01110	*d8c5786f*	00000	00000
400	1	10110	*1eec4c6a*	00000	00000
600	0	00000	00000000	01110	10110

表 3：寄存器 *Registers* 的测试项目

图 3 中红色的输出为“x”值，表示输出信号为不确定。由上图可看出，初始时由于寄存器的内容为空，所以访问得出的数据均为“x”不确定值。可看出设计的模块对输入指令所输出的结果均与预期的一致，表示该模块的实现正确。

4.2 内存单元模块 *Memory* 的实现

对内存单元模块 *Memory* 进行表 4 的测试，得出如下的仿真波形图：

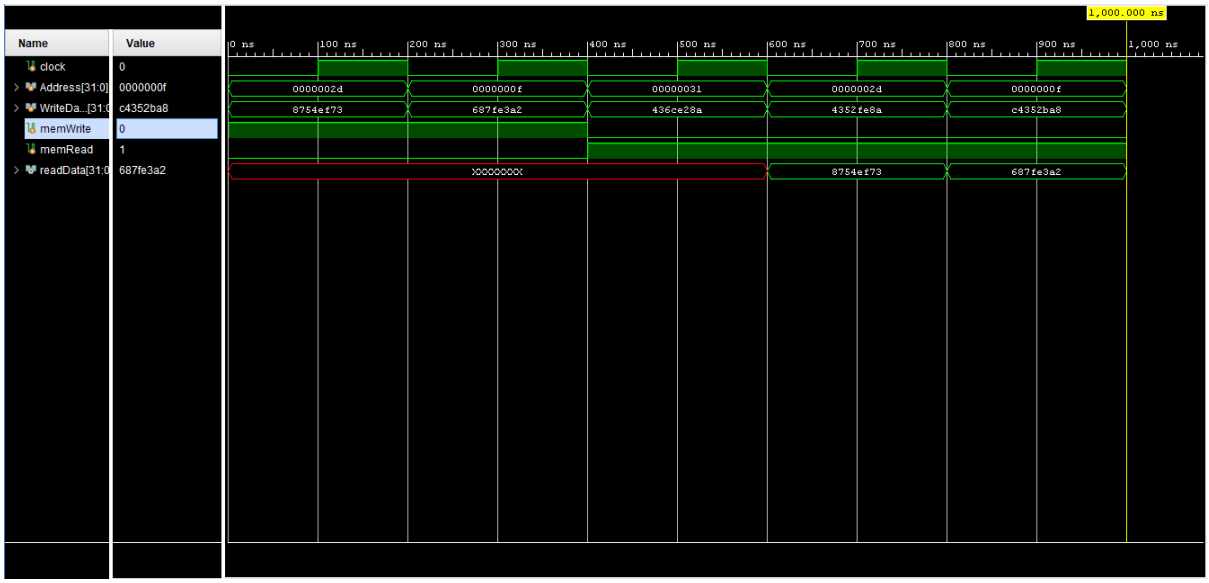

图 4：内存单元模块 *Memory* 的仿真结果

时间 ns	$memWrite$	$memRead$	$writeData$	$Address$
0	1	0	$8754ef73$	$0000002d$
200	1	0	$687fe3a2$	$0000000f$
400	0	1	$436ce28a$	00000031
600	0	1	$4352fe8a$	$0000002d$
800	0	1	$c4352ba8$	$0000000f$

表 4：内存单元模块 *Memory* 的测试项目

图 4 中红色的输出为“x”值，表示输出信号为不确定。由上图可看出，当内存欲访问地址为“00000031”的数据时，由于仍处于初始状态，未被写入任何数据，所以访问

得出的数据为“x”不确定值。可看出所设计的模块对输入指令所输出的结果均与预期的一致，表示该模块的实现正确。

4.3 带符号数扩展单元的仿真

对带符号数扩展单元进行表 5 的测试，得出如下的仿真波形图：

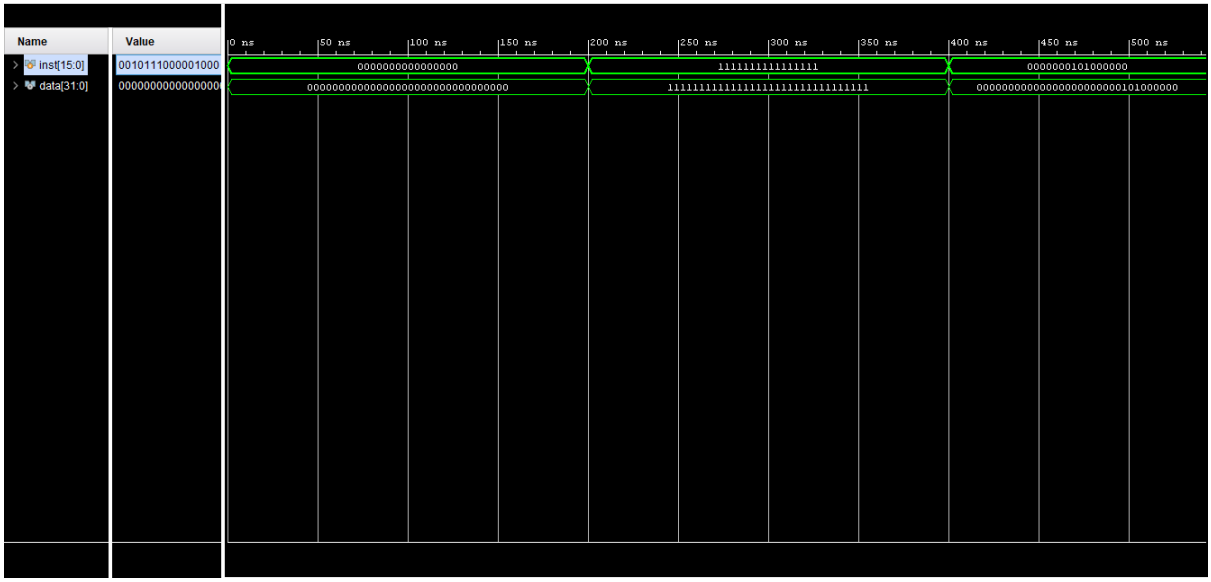

图 5：带符号数扩展单元的仿真结果 1

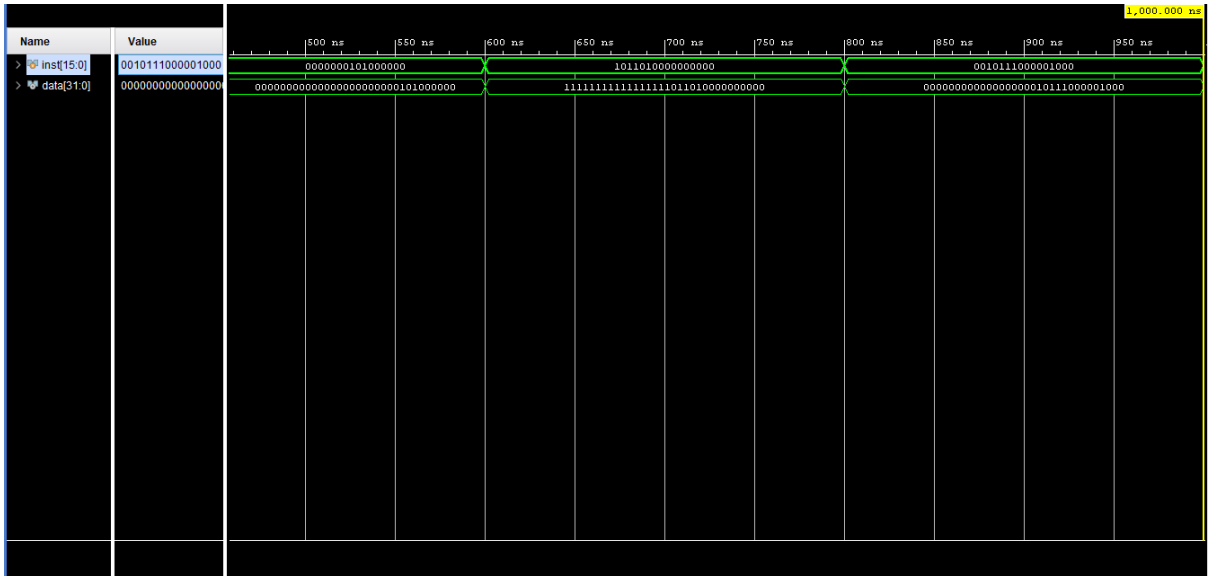

图 6：带符号数扩展单元的仿真结果 2

时间 ns	16 位的带符号数
0	0000000000000000
200	1111111111111111
400	0000000101000000
600	1011010000000000
800	0010111000001000

表 5：带符号数扩展单元的测试项目

由图 5 和图 6 可看出，所设计的模块对所有指令所输出的结果均与预期的一致，表示该模块的实现未正确。

5. 实验总结

本次实验主要着重于 $MIPS$ 处理器中的寄存器、内存模块单元以及带符号数扩展单元。本次实验所要实现单元模块的功能总体来是还是比较简单的，因此相对而言难度还并不是很大；但相比于上一次的实验，本次实验没有相应的真值表，不能只依样画葫芦，需要自己思考如何设计相关电路的逻辑关系，并通过 $Verilog$ 语句实现相应的功能。本次实验的实验令我对$MIPS$ 处理器内部部件的操作原理更为收悉，通过动手设计、实践的方法更能将课堂中所学的理论知识灵活运用到实际应用中。