**计算机系统结构实验实验报告 Lab04**

姓名：叶增渝

学号：519030910168

日期：2021年04月09日

**目录**

[1.实验概述 2](#_Toc70973178)

[1.1 实验名称 2](#_Toc70973179)

[1.2 实验目的 2](#_Toc70973180)

[2.实验描述 2](#_Toc70973181)

[2.1 寄存器 2](#_Toc70973182)

[2.1.1寄存器模块描述 2](#_Toc70973183)

[2.1.3仿真激励代码 4](#_Toc70973184)

[2.1.4仿真波形 5](#_Toc70973185)

[2.1.5实验结论 5](#_Toc70973186)

[2.2内存单元模块 5](#_Toc70973187)

[2.2.1内存单元模块描述 5](#_Toc70973188)

[2.2.2内存单元模块代码 6](#_Toc70973189)

[2.2.5实验结论 8](#_Toc70973190)

[2.3带符号扩展 8](#_Toc70973191)

[2.3.1带符号扩展模块描述 8](#_Toc70973192)

[2.3.2ALU模块代码 9](#_Toc70973193)

[2.3.3ALU仿真激励代码 9](#_Toc70973194)

[2.3.4仿真波形 9](#_Toc70973195)

[2.3.5实验结论 10](#_Toc70973196)

[3.实验心得与总结 10](#_Toc70973197)

[4.参考资料 10](#_Toc70973198)

1.实验概述

1.1 实验名称  
简单的类 MIPS 单周期处理器实现 –寄存器、 存储器与有符号扩展

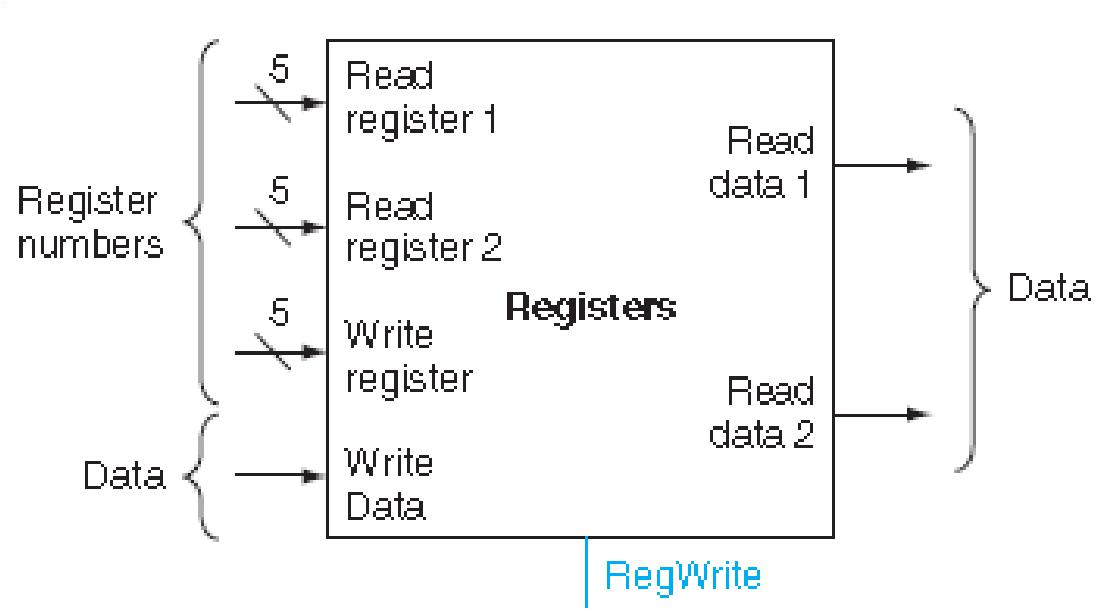
1.2 实验目的  
1.理解 CPU 的寄存器、 存储器、 有符号扩展  
1.Register 的实现  
2.Data Memory 的实现  
3.有符号扩展的实现  
4.使用行为仿真

2.实验描述

# 2.1 寄存器

## 2.1.1寄存器模块描述

寄存器是指令操作的主要对象，用于数据的存储与拿取。32位的 MIPS 中共有 32 个 32 位的寄存器。



寄存器模块

本小节运用Verilog硬件描述语言来模拟寄存器的运行流程，其中包含4个输入、1个控制信号与2个输出，其中写操作当且仅当在时钟下跳沿正式写入。

2.1.2 寄存器模块代码

由于不确定WriteReg,WriteData,RegWrite信号的先后次序，可采用时钟的下降沿作为写操作的同步信号，以防止发生错误。

Tips：在以非0情况下的ReadData无法正常返回值，所以应当在redReg这里增加判断成为

if(readReg1)

ReadData1 = regFile[readReg1];

else

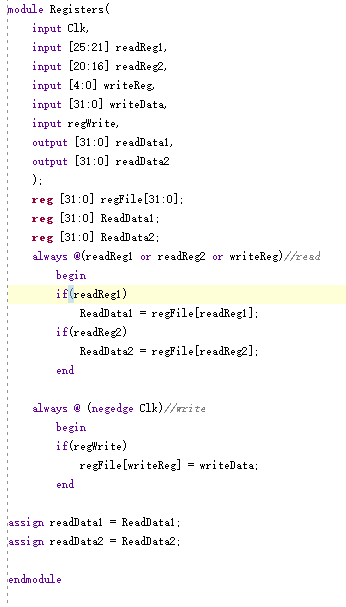
ReadData1 = 0;

if(readReg2)

ReadData2 = regFile[readReg2];

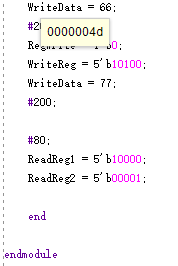
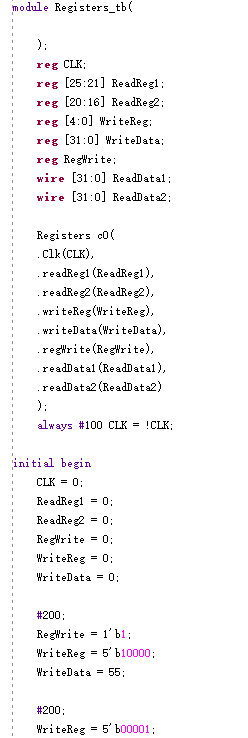
else

ReadData2 = 0;

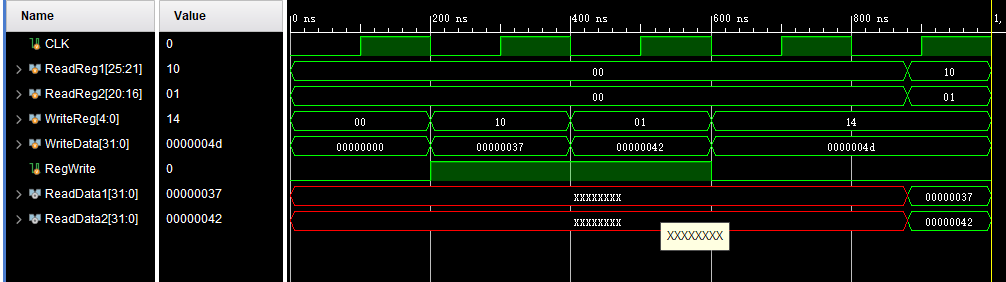


## 2.1.3仿真激励代码

激励文件主要通过每隔一段时间更换输入来模拟各类写与读操作，其中时钟Clk值每100ns改变一次



## 2.1.4仿真波形



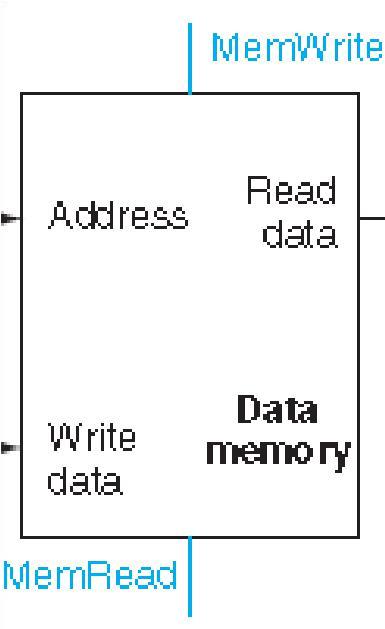
## 2.1.5实验结论

首先在200ns与400ns时向10与01这两个32位寄存器中写值后，在900ns两个读信号ReadReg1与ReadReg2转为1，可以看到成功读出了之前存入的值，与实验预期相同，说明我们能够通过编写代码很好地模拟单周期处理器中的寄存器模块的工作。

# 2.2内存单元模块

## 2.2.1内存单元模块描述

存储器本模块与register类似，由于写数据也要考虑信号同步，因此也需要时钟。内存单元的实现，也可用系统Block Memory来生成。

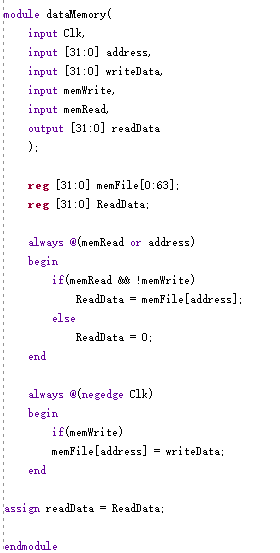


内存单元

这里为单周期处理器的取数据模块，也是多周期处理器的Mem的组件。

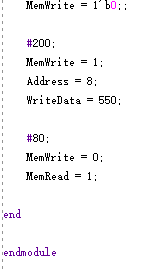
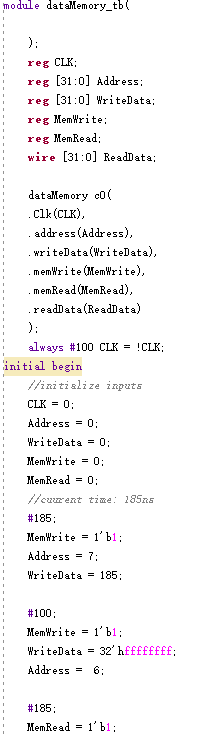
## 2.2.2内存单元模块代码

当且仅当memRead为1而memWrite为0时，读操作才能发生由于不知道什么时候需要开始读，所以要检测memRead与address的改变，而写操作应当在时钟下跳沿时受到memWrite为1的信号时写入。

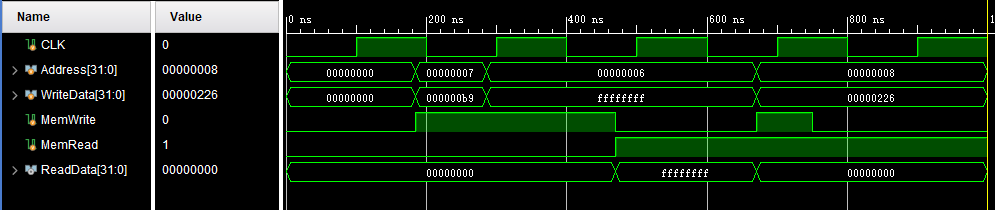


**2.2.3ALUCtr仿真激励代码**

设置时钟变换为100ns，在初始化后在不同时间点更改参数进行测试



**2.2.4仿真波形**



## 2.2.5实验结论

在200ns到470n成功s向6与8号寄存器中写入了数据，而在此后到570ns，memRead为1而memWrite为0，ReadData成功取出了对应寄存器的数据。说明通过上述代码可以很好地实现单周期处理器的内存单元模块的功能。

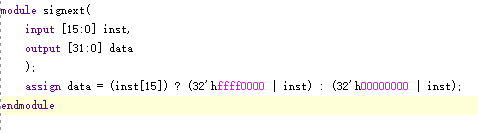
# 2.3带符号扩展

## 2.3.1带符号扩展模块描述

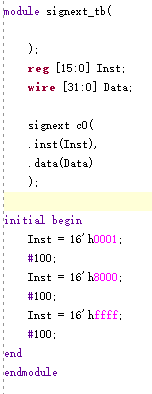
将16位有符号数扩展为32位有符号数，只需要将前十六位用符号位填充满即可。  
补码：  
（1） 正数的补码： 与原码相同。如+9 的补码是 00001001。  
（2） 负数的补码：符号位为 1， 其余位为该数绝对值的原码按位取反； 然后整个数加 1。  
举例：-7 因为是负数，则符号位为“1”。后七位： +7 的原码（0000111） →按位取反（1111000） →加 1（1111001），所以-7 的补码是 11111001。

## 2.3.2ALU模块代码

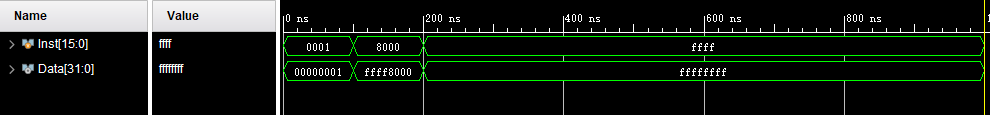
这里判断符号位为0还是1，，常数后16位为0，或操作对后16位不影响，如果是1，前16位与1或，则前16位全为1；如果是0，前16位与0或，则前16位全为0.



## 2.3.3ALU仿真激励代码



## 2.3.4仿真波形



## 2.3.5实验结论

当Inst的符号位为1时，如8000H与ffffH，可以看到前16位均为1，而符号位为0时，如0001H前16位为0。

3.实验心得与总结

实验4并没有上板测试过程仅有模拟仿真，相对实验3增加了时钟下跳沿的判断与读写操作的判断，难度略有增加。但有了实验3的铺垫，实际难度并不大。在调试dataMemory时需要同时判断memRead与address均需判断，在这里卡了很久，但也让我学到了条件判断的重要性。在带符号扩展时则发现无法在always @外直接用if，算是吃一堑长一智。最重要的是通过这个实验更加了解了单周期处理器各重要单元模块的实现过程，对单周期处理器有了更加深刻的了解。

在后续实验5的情况下，我同时发现了实验4在设计上存在的些许问题，例如寄存器模块，应当增加当读取的寄存器为0时的情况，否则readReg在判断时不成立，无法成功读取，在此时，应当增加条件判断直接返回0。

4.参考资料

2020计算机系统结构实验指导书-LAB04\_M