**深 圳 大 学 实 验 报 告**

**课 程 名 称： 计算机系统(3)**

**实验项目名称： 新增指令实验**

**学 院： 计算机与软件学院**

**专 业： 计算机与软件学院所有专业**

**指 导 教 师： 刘刚**

**报告人： 何泽锋 学号： 2022150221 班级： 高性能特色班**

**实 验 时 间： 2024年11月30日-2024年12月1日**

**实验报告提交时间： 2024年12月1日**

**教务处制**

**一、 实验目标：**

了解RISC-V mini处理器架构，在其基础之上新增一个指令，完成设计并观察指令执⾏。

**二、实验内容**

1) 修改数据通路，新增指令comb rs1,rs2,rd采用R型指令格式，实现将rs1高16位和rs2低16位拼接成32位整数，并且保存到rd寄存器。

2) 在处理器上执行该指令，观察仿真波形，验证功能是否正确。

3） 自行设计其他功能指令，并验证设计是否正确

**三、实验环境**

硬件：桌面PC

软件：Chisel开发环境

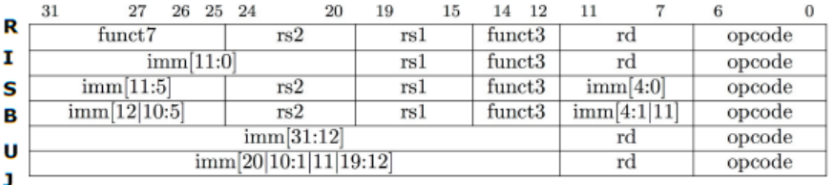
**四、****实验步骤及说明**

学习Chisel数据通路的Chisel描述，特别是指令译码部分和core核心代码。然后按照下面操作完成指令译码器的修改，以及数据通路的修改，

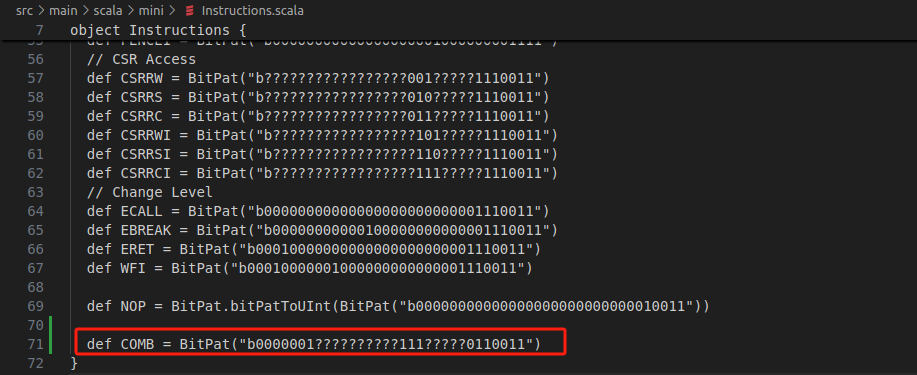
具体操作如下：按照参考文档完成comb指令的实现，自行设计新指令实现其功能并验证。

1. 在Instrutcions.scala文件中添加 comb 指令比特模式串

（1）分析comb指令，comb指令为R型指令，其格式如下

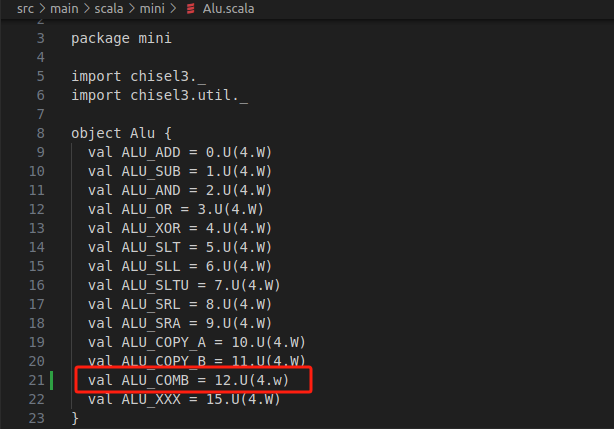


（2）此处为了避免与riscv-mini的指令冲突，将 comb 指令的opcode、funct3和funct7部分设置为0110011、111、0000001。并将比特模式串写入Instrutcions.scala文件中。

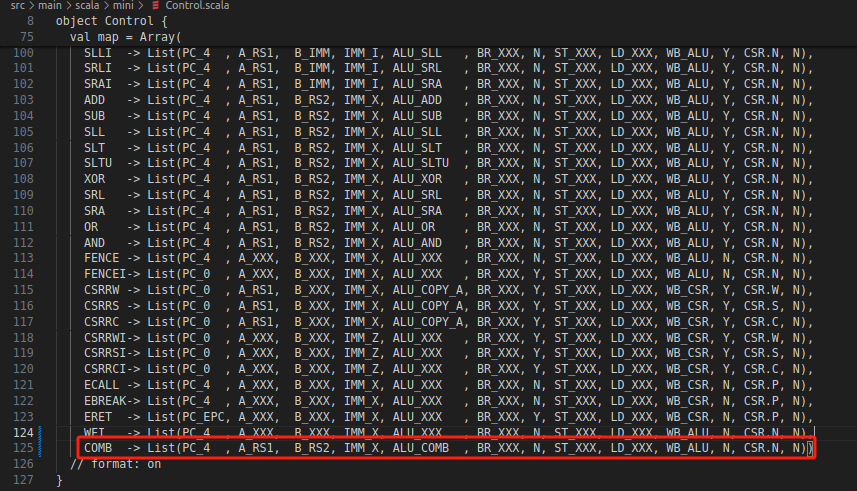


2. . 添加 comb 指令的译码

（1）comb指令的作用是将rs1的高16位与rs2的低16位拼接成32位整数，，在ALU.scala文件中添加常量ALU\_COMB，使得译码出正确信号

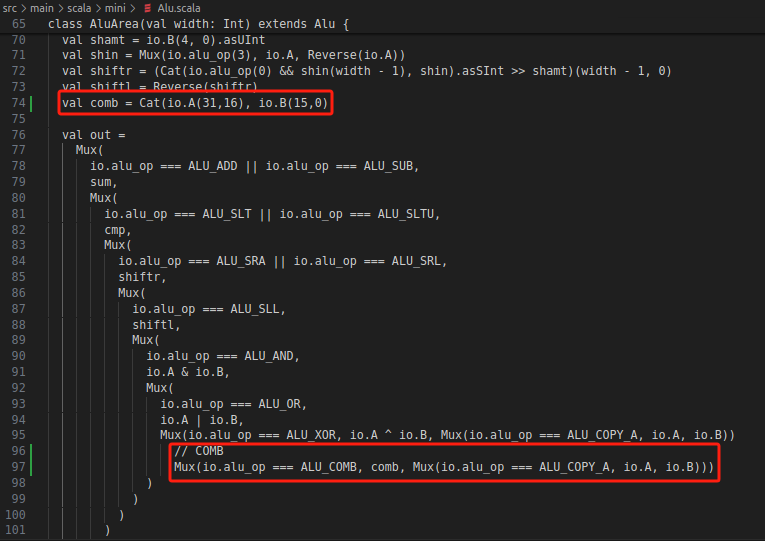
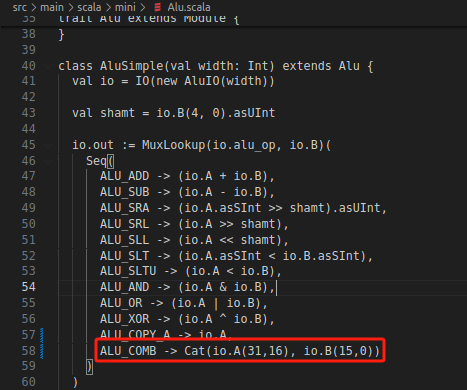


（2）然后为comb指令添加译码映射，在Control.scala文件中进行修改，当指令执行后PC+4，然后从寄存器中读取rs1和rs2，然后进行拼接，将ALU输出结果写回寄存器



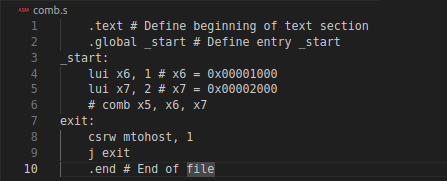
3. 实现 comb 指令的执行操作

（1）在Alu.scala文件添加将rs1高16位和rs2低16位拼接成32位整数的操作

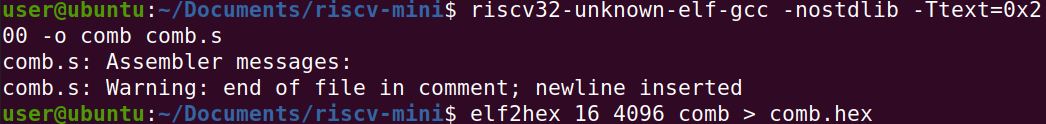


4. 对 comb 指令进行测试

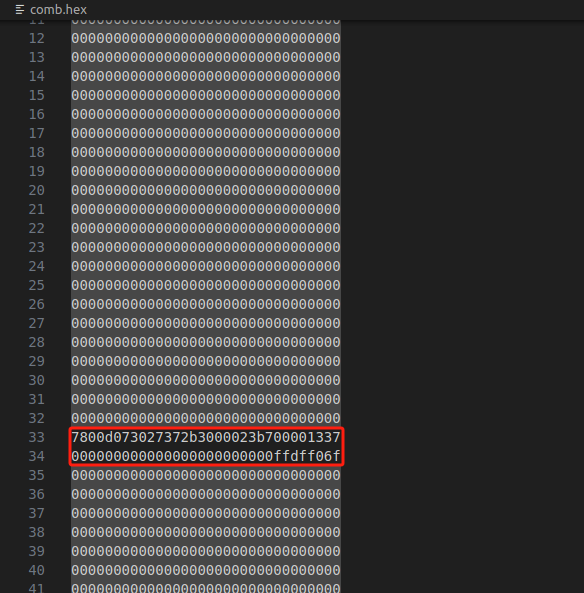
（1）创建测试文件comb.s，并写入汇编指令，需要注意comb为自己加入的指令，不能被汇编器汇编，所以这里将其注释掉，到后面生成的comb.hex文件中再将 comb x5, x6, x7 的二进制添加进去。



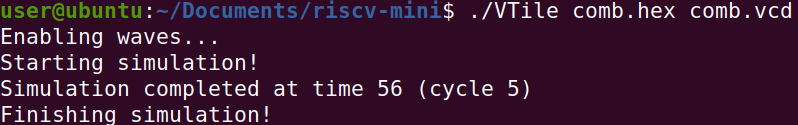
（2）编写完程序后，使用如下命令进行编译，然后使用 elf2hx 命令将comb二进制文件转换成十六进制：



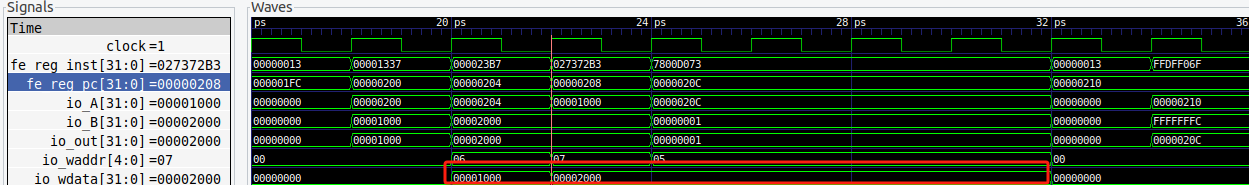
（3）在comb.hex文件中，可以找到 lui x6, 1 和 lui x7, 2 的机器码对应的十六进制形式，comb x5, x6, x7 转换成机器码的十六机制形式为 027372b3。因此处指令存储为小端模式，故我们需要将十六进制数插入到如图位置。修改后如下：



（4）执行下面命令，使mini处理器执行新建指令并产生波形文件。

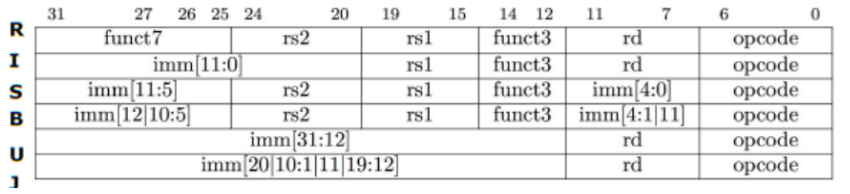


（5）使用GTKWave打开comb.vcd文件，其波形图如下，从波形图中可以看出， comb 指令将拼接后的结果0x00002000写回到了5号寄存器中，故该指令执行正常。



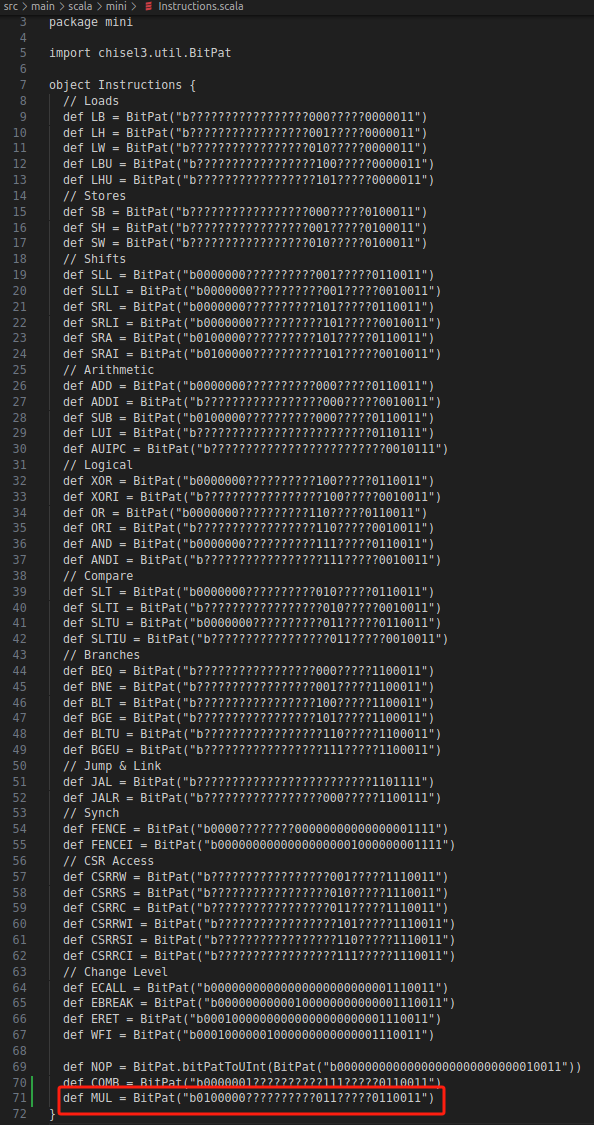
**新增指令**

新增加的指令为mul 乘法指令，读入两个寄存器数并将乘法计算结果写回第三个寄存器。根据R型指令规则，将funct7定义为0100000，funct3定义位011，opcode定义为0110011



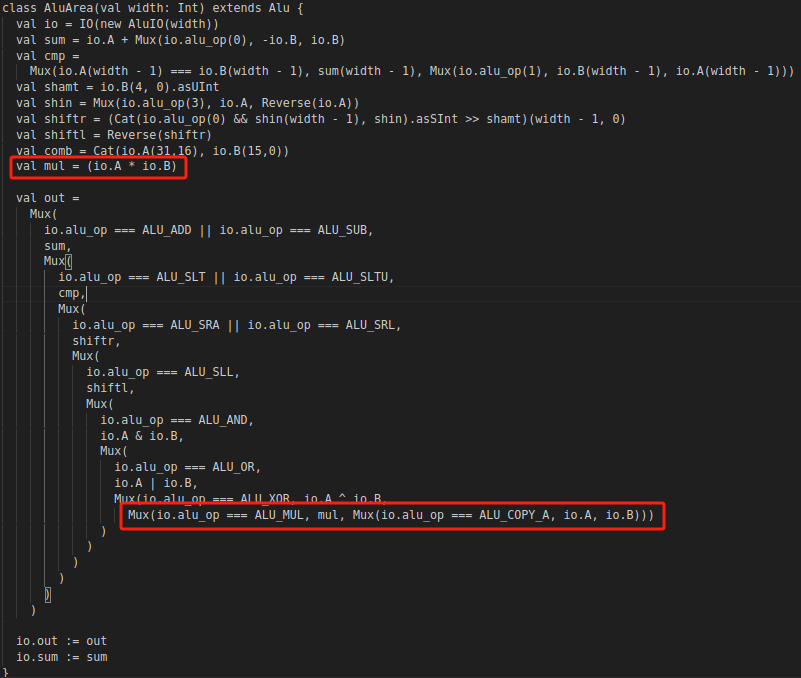
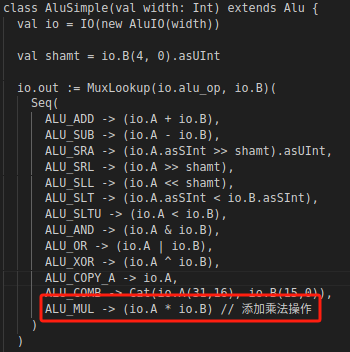
1.Instructions.scala：

（1）设置 mul 指令的比特模式。



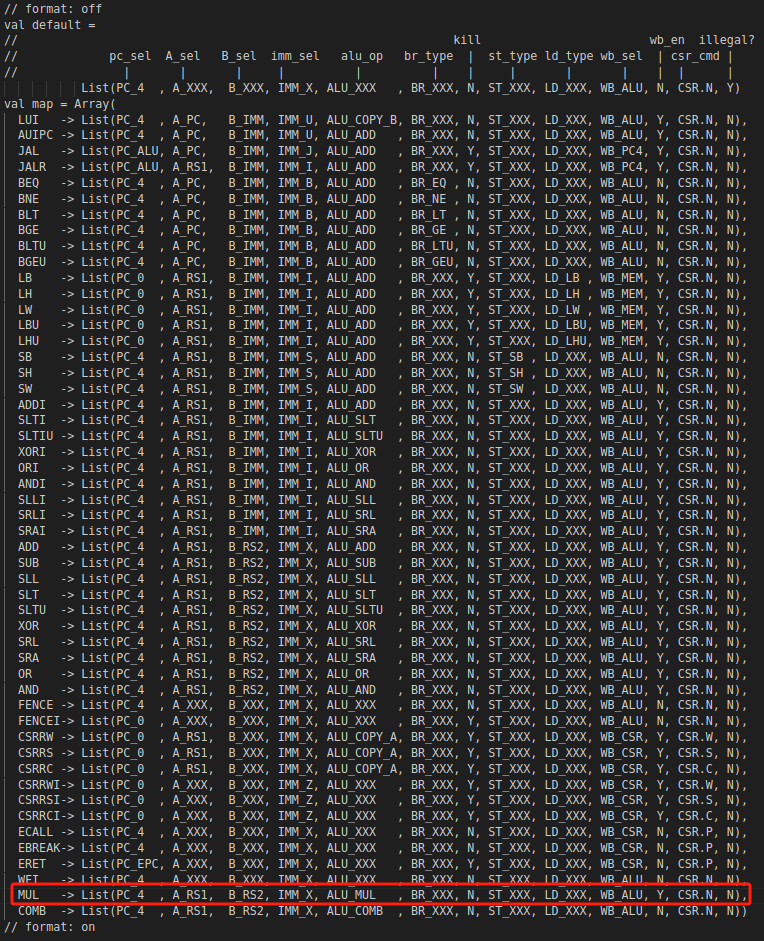
2.ALU.scala：

（1）添加常量 ALU\_MUL ，让译码器可以译码出正确的信号。



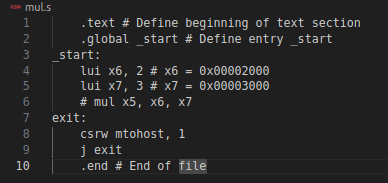
3.control.scala：

（1）为 mul 指令添加对应的译码映射。 mul 指令执行后pc需要加4，并将从寄存器文件中读取的数据rs1和rs2进行拼接操作，然后将ALU输出的拼接结果写回到寄存器文件中。



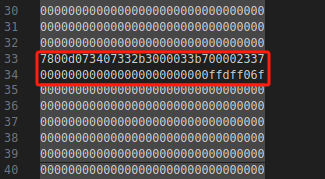
4.测试：

（1）首先编写mul.s文件并写入测试汇编代码，同理此处需要将新加入的指令注释掉，在生成.hex文件后在加入对应的二进制。

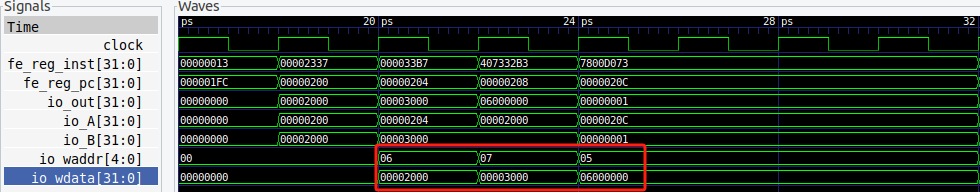


（2）在mul.hex文件中写入mul x5, x6, x7的二进制码，

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Funct7(31-25) | Rs2(24-20) | Rs1(19-15) | Funct3(14-12) | Rd(11-7) | Opcode(6-0) |
| 0100000 | 00111 | 00110 | 011 | 00101 | 0110011 |



（3）观察结果，在io\_wdata中，x6为00002000，x7为00003000，二者相乘结果为06000000，存储在x5中，如图所示结果正确，指令正确。



**五、实验结果**

本实验成功实现了两个新增指令：comb 和 mul，并通过仿真波形验证了其功能正确性。

（1）comb 指令：

①指令格式：R 型指令

②功能：将寄存器 rs1 的高 16 位和 rs2 的低 16 位拼接成 32 位整数，并保存到 rd 寄存器。

③仿真波形验证：comb 指令能够将拼接后的结果正确写入指定寄存器，例如将 rs1 = 0x00001000 (十六进制) 和 rs2 = 0x00002000 (十六进制) 拼接后，rd 寄存器值为 0x00002000 (十六进制)。

（2）mul 指令：

①指令格式：R 型指令

②功能：读取寄存器 rs1 和 rs2 进行乘法运算，将结果写回rd寄存器。

③仿真波形验证：mul 指令能够将乘法结果正确写入指定寄存器，例如将 rs1 = 0x00002000 (十六进制) 和 rs2 = 0x00003000 (十六进制) 相乘后，rd 寄存器值为 0x06000000 (十六进制)。

**六、实验总结与体会**

通过本次实验，我深入理解了 RISC-V mini 处理器架构，并掌握了如何使用 Chisel 语言进行指令集扩展，学习了以下核心知识：指令译码，将指令从二进制形式转换为可执行的操作，并确定指令类型、操作数来源等信息；数据通路设计，指令执行过程中数据流动的路径，包括寄存器、ALU、数据存储器等组件的连接方式；ALU 操作，完成指令指定的算术或逻辑运算，例如加法、减法、乘法、移位等。

通过亲自实现 comb 和 mul 两个新指令，我深刻体会到指令集扩展的复杂性和重要性。每个指令的实现都需要考虑指令格式、操作数来源、数据通路设计、ALU 操作等多个方面，并通过观察波形图确保指令的执行结果的正确性。

|  |
| --- |
| **指导教师批阅意见：**  **成绩评定：**  指导教师签字：  年 月 日 |
| 备注： |