**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： 软件工程（腾班）**

**指 导 教 师： 王 毅**

**报告人： 黄亮铭 学号： 2022155028 班级： 腾班**

**实 验 时 间： 2024年12月13日**

**实验报告提交时间： 2025年1月3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。然后按照下面指引完成指令译码器的修改，以及数据通路的修改。

分析：添加新指令 comb ，首先需要根据riscv指令格式，设置该指令各个字段的值，并在相应文件中添加该指令的比特模式。然后设置该指令的译码结果，接着在ALU中实现该指令的功能。最后让该指令在处理器上执行，验证功能是否正确。以下操作均在实验六下载下来的riscv-mini项目工程上进行。

**1. 添加 comb 指令比特模式串**

comb 为R型指令，riscv的R型指令格式如图1所示：

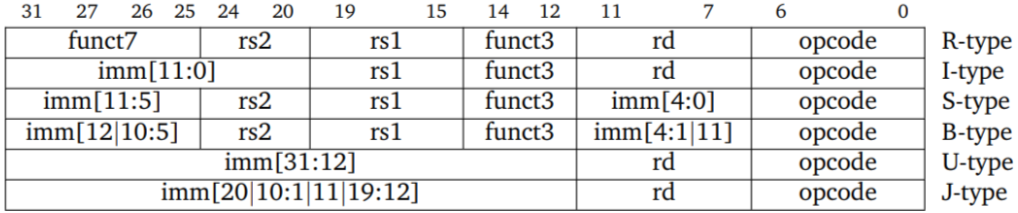


图 1 riscv指令格式

为了避免新加指令与riscv-mini已有指令冲突，这里我们将 comb 指令的opcode、funct3和funct7部分设置为0110011、111、0000001。然后使用 BitPat() 函数设置 comb 指令的比特模式。请在Instrutcions.scala中添加comb指令比特模式串，并截图。

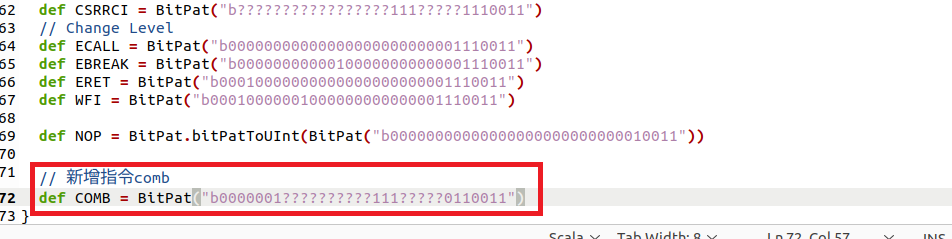


图 2 在Instructions.scala中添加comb指令比特模式串

**2. 添加 comb 指令的译码**

comb 指令需要在ALU中将rs1高16位和rs2低16位拼接成32位整数，因此需要在Alu.scala文件中添加常量 ALU\_COMB ，让译码器可以译码出正确的信号。请在Alu.scala中添加该常量，并截图。

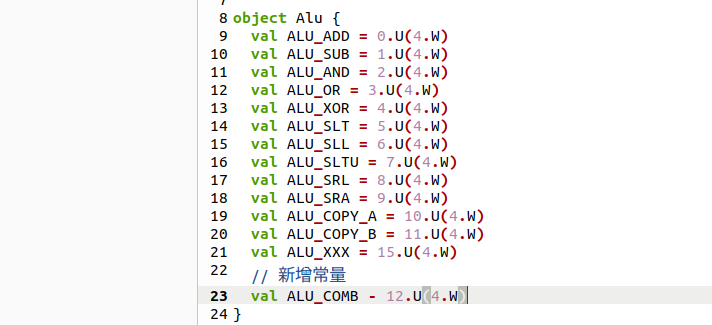


图 3 在Alu.scala中添加ALU\_COMB常量

接下来在Control.scala文件中为 comb 指令添加对应的译码映射。comb 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。请在Control.scala中添加comb指令的译码映射，并截图。

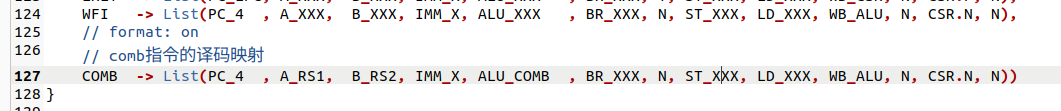


图 4 在Control.scala中添加comb指令的译码映射

**3. 实现 comb 指令的执行操作**

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。请在Alu.scala中实现该操作，并截图。

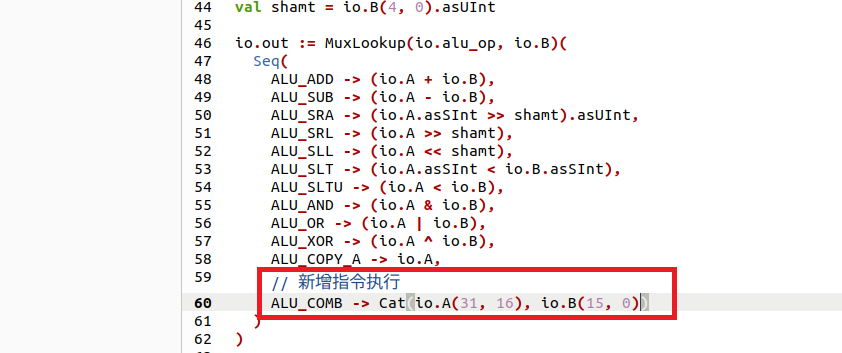


图 5 在Alu.scala中的AluSimple添加comb指令执行的逻辑

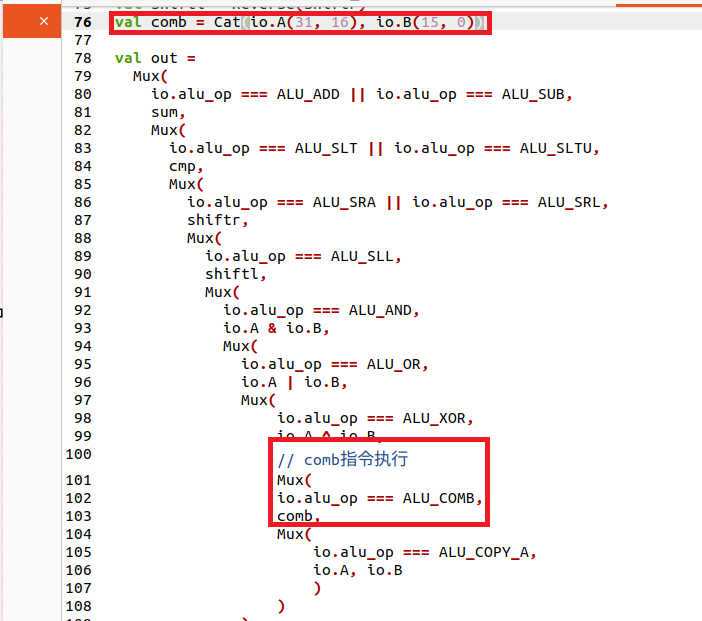


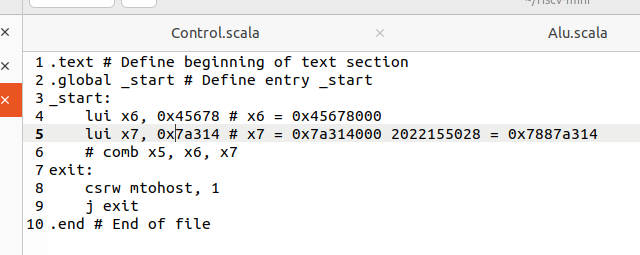
图 6 在Alu.scala中的AluArea添加comb指令执行的逻辑和多选器逻辑

**4. 对comb指令进行测试**

参照实验六第二部分第一节，尝试编写一个汇编程序comb.s，对comb指令进行测试。这里给出一个样例代码。在下面代码中，x6寄存器中的数据为0x00001000，x7寄存器中的数据为0x00002000，comb命令将x6的高16位（即0x0000）和x7的低16位（即0x2000）拼接成一个32位的整数（即0x00002000）。

1. .text *# Define beginning of text section*
2. .global \_start *# Define entry \_start*
3. \_start:
4. lui x6, 1 *# x6 = 0x00001000*
5. lui x7, 2 *# x7 = 0x00002000*
6. *# comb x5, x6, x7*
7. exit:
8. csrw mtohost, 1
9. j exit
10. .end *# End of file*

**我随机生成了一个随机数0x12345678，装载进x6寄存器中，因为是lui的功能是把一个20位的立即数装载进寄存器的高20位。因此，在编写汇编代码的时候可以将0x12345678写成0x45678（满足该数大于0xFFFF的要求）；我的学号为2022155028，16进制表示为0x7887a314，与上同理，在编写汇编代码的时候可以将0x7887a314写成0x7a314。**



请注意，因为 comb 为自己加入的指令，不能被汇编器汇编，所以这里先将其注释掉，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。

编写完程序后，使用riscv32-unknown-elf-gcc编译得到comb二进制文件：

**我的编译指令：**

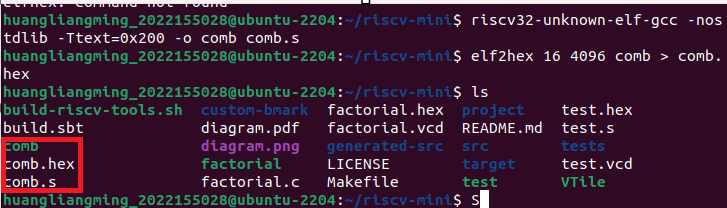
|  |
| --- |
| **riscv32-unknown-elf-gcc -nostdlib -Ttext=0x200 -o comb comb.s** |

然后使用elf2hx将编译得到的comb二进制文件转换成16进制文件comb.hex：

**我的转换指令：**

|  |
| --- |
| **elf2hex 16 4096 comb > comb.hex** |

执行完上述两条指令之后，对应文件夹的目录如下，可以看到对应文件已经生成。



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，如图7所示：

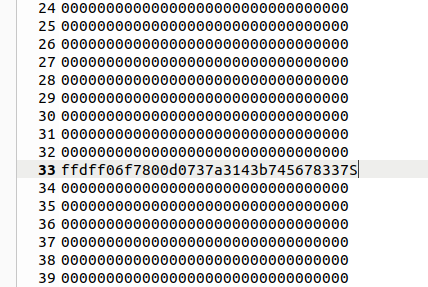


图 7 在comb.hex中找到lui x6,1和lui x7,2的十六进制机器码

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第一个红线的前面。修改后如图8所示：

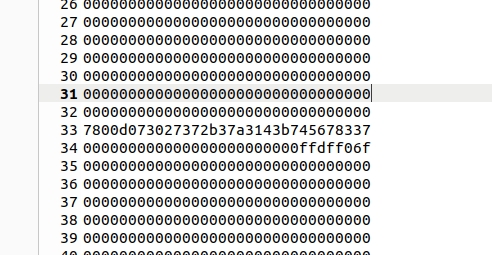


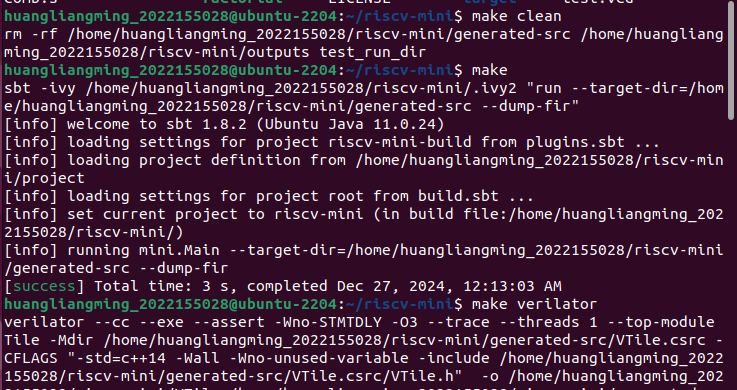
图 8 将comb x5, x6, x7汇编的十六进制机器码插入到comb.hex中

接着需要在主目录下一次执行 make 和 make verilator 命令（若之前已经执行过，则在此次操作之前需要执行 make clean），执行后会产生VTile可执行文件。然后执行下面命令，使mini处理器执行新建指令并产生波形文件。

**我执行了如下命令：**

|  |
| --- |
| **make clean**  **make**  **make verilator** |

执行结果如下图所示，说明命令运行成功。



**使用GTKWave打开comb.vcd文件（对应指令为./VTile comb.hex comb.vcd， gtkwave comb.vcd**）**，执行指令结果如下图所示。其波形图如图9所示。**

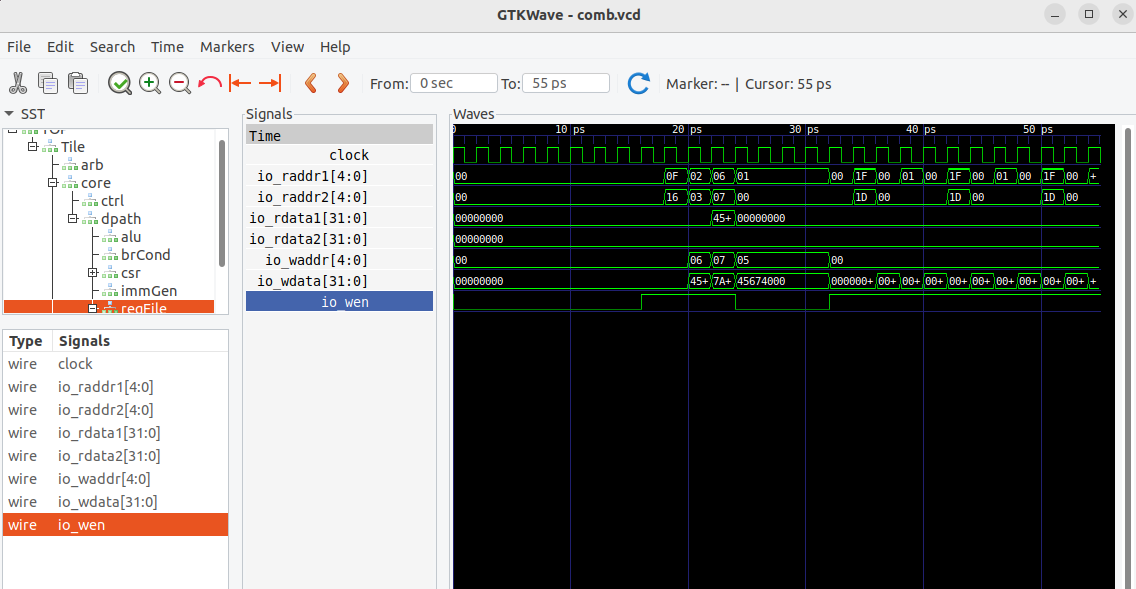
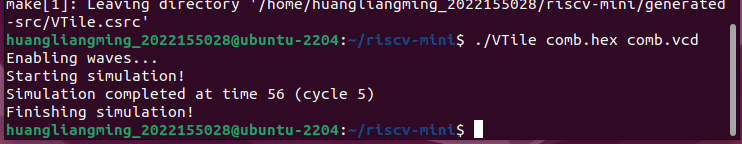


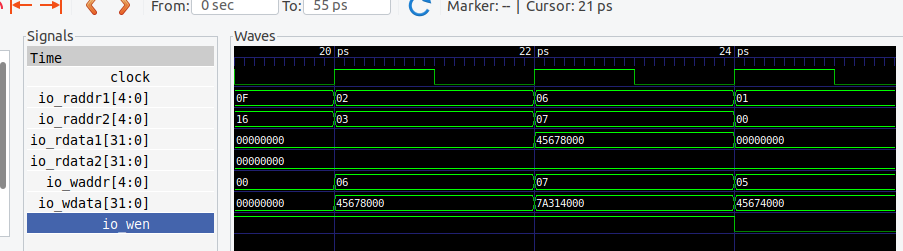
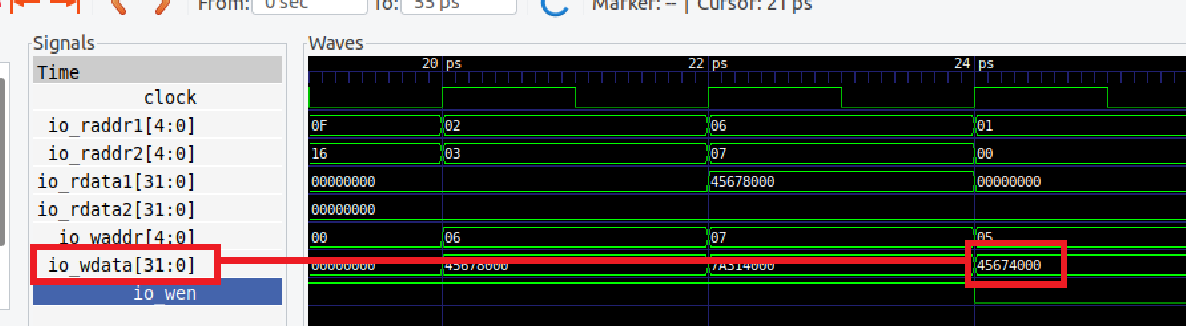
图 9 comb测试程序的波形图

指令对应的十六进制形式见表1所示：

表 1 核心指令对应的十六进制形式

|  |  |  |
| --- | --- | --- |
| **指令** | **十六进制表示** | **说明** |
| lui x6, 1 | 45678337 | x6 = 0x45678000 |
| lui x7, 2 | 7a31423b7 | x7 = 0x7a314000 |
| comb x5, x6, x7 | 027372b3 | x5 = cat(x6(31:16), x7(15:0)) |

从波形图中可以看出， comb 指令将拼接后的结果0x45674000写回到了5号寄存器中，故该指令执行正常。



可以看到，io\_wdata分别获得x6和x7的值，然后将其拼接，最后写入x5中。

**五、实验结果**

1. 本次实验我成功地添加了一条新增指令，用于将两个寄存器的数值进行拼接。
2. 本次实验，我在处理器上执行该指令，观察仿真波形，验证功能的正确性。

**六、实验总结与体会**

1. 在本次实验中，通过对数据通路的修改和指令新增，我更深入地理解了RISC-V处理器的架构。
2. 通过观察仿真波形图，我学会了如何验证和调试新增的指令。
3. 通过对数据通路的修改，我了解了数字逻辑设计的基本原理。
4. 通过本次实验，我理解了如何在处理器中插入新指令。

本次实验我将文档中给定的Terminal图替换成我自己的Terminal图，图中的绿色的用户名为我的姓名的拼音+学号，即huangliangming\_2022155028。

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