**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： 罗秋明**

**报告人： 岳海涛 学号： 2022152020 班级： 22计科**

**实 验 时 间： 2024年12月9日**

**实验报告提交时间： 2024年12月9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

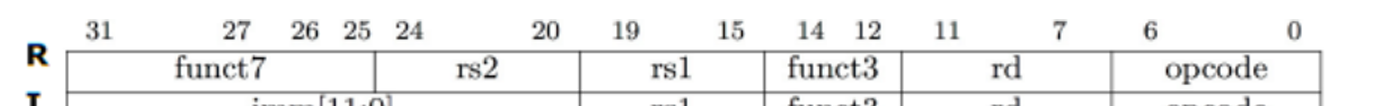
**四、****实验步骤及说明**

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，

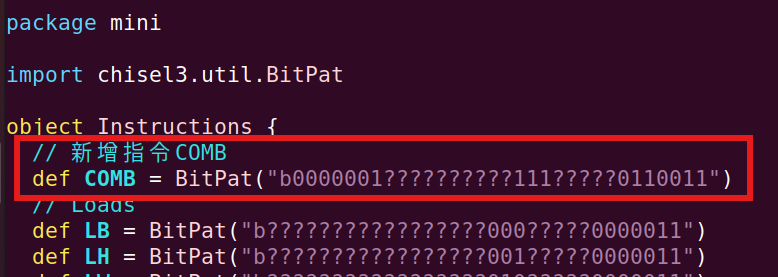
具体操作如下：按照参考文档完成comb指令的实现，自行设计新指令实现其功能并验证。

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

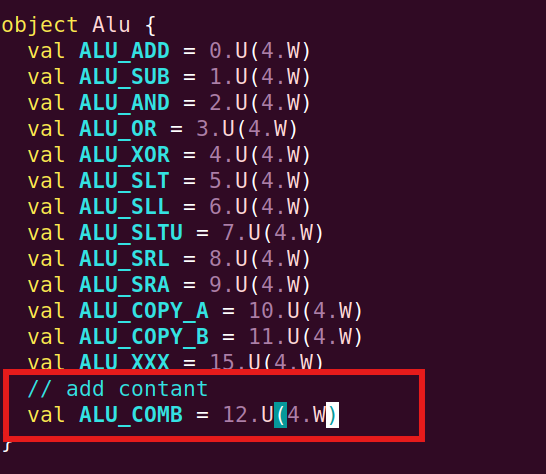
comb 为R型指令，riscv的R型指令格式如下：



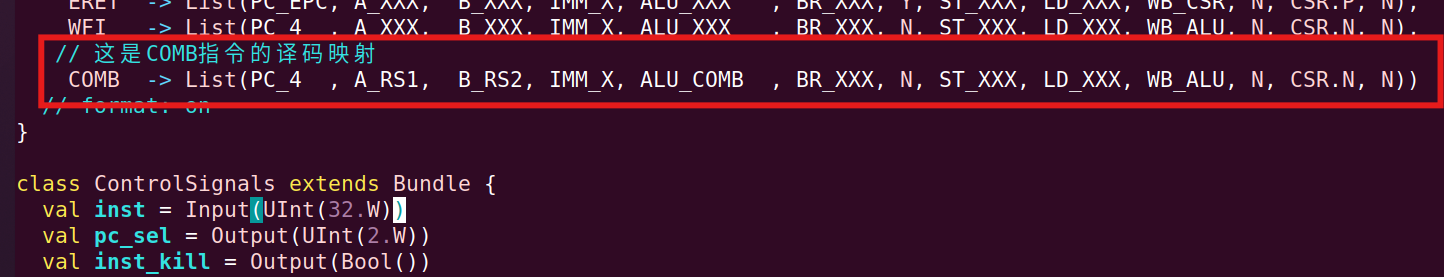
将 comb 指令的opcode、funct3和funct7部分设置为0110011、111、0000001。然 后使用 BitPat() 函数设置 comb 指令的比特模式



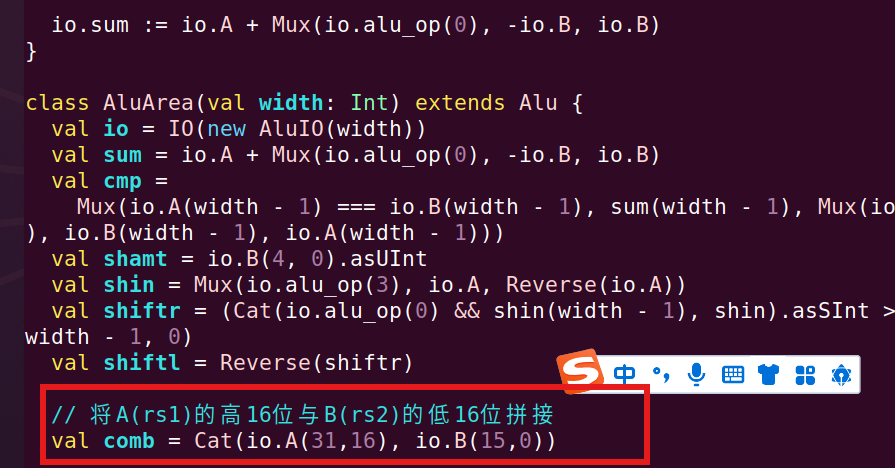
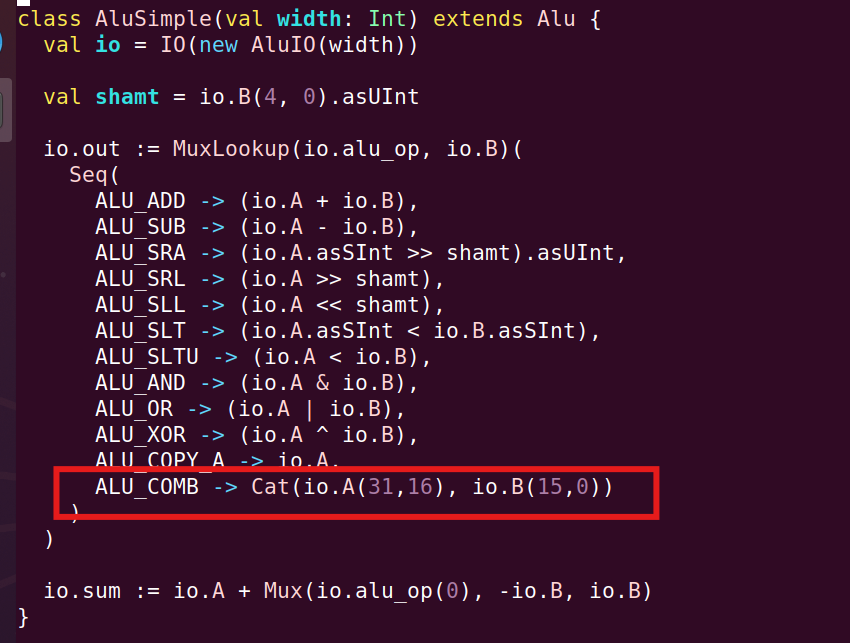
**2.1 添加 comb 指令的译码 comb 指令需要在ALU中将rs1高16位和rs2低16位拼接成32位整数，因此需要在Alu.scala文件中添加常量 码出正确的信号**



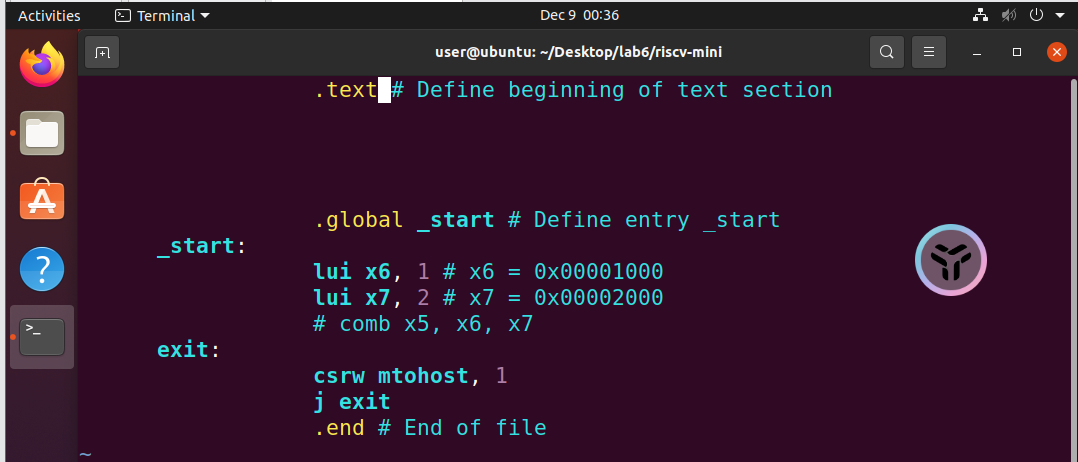
**2.2为comb指令添加对应的译码映射。comb指令执行后pc需要加4，并将从寄存器文件中读取的数据rs1和rs2进行拼接操作，然后 将ALU输出的拼接结果写回到寄存器文件中。**



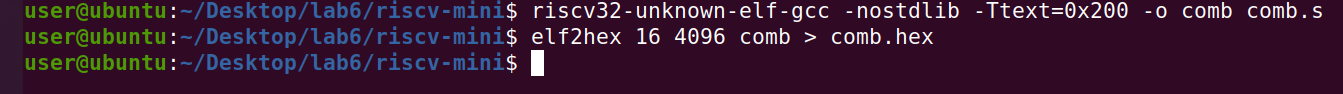
**2.3. 实现comb指令的执行操作 在Alu.scala文件添加将rs1高16位和rs2低16位拼接成32位整数的操作**



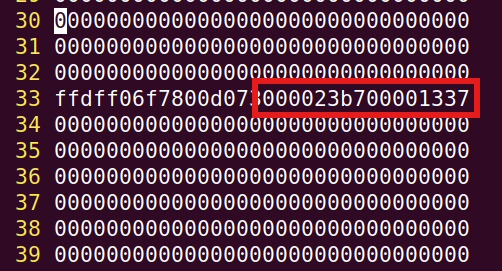
**3. 1对comb指令进行测试 首先创建comb.s文件**



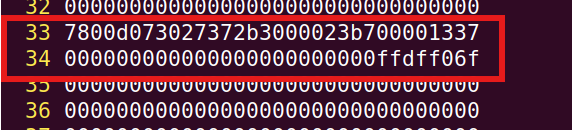
**3.2编写完程序后，使用如下命令进行编译，然后使用elf2hx命令将comb二进制文件转换成十六进制：**



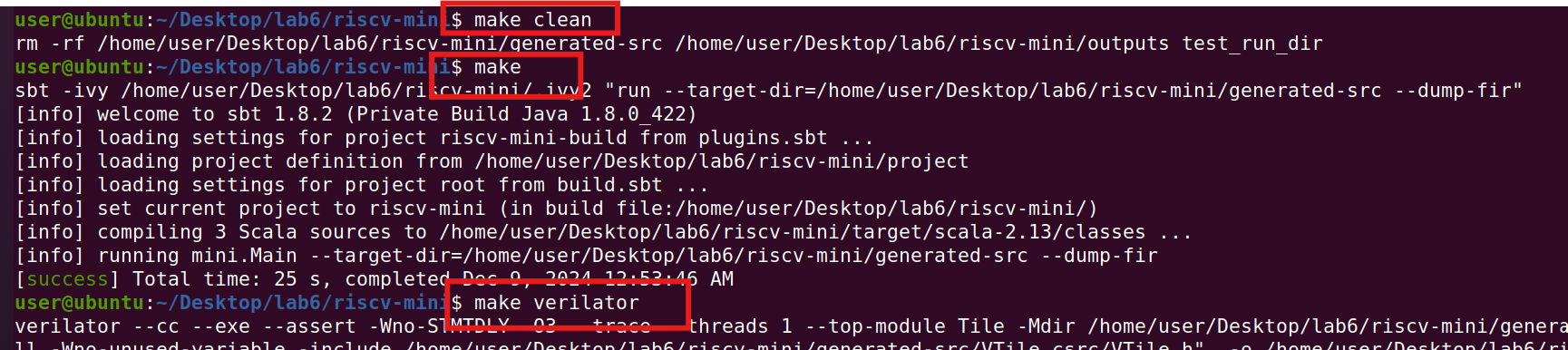
**3.3 comb.hex文件中，可以找到 lui x6, 1和 lui x7, 2的机器码对应的十六进制形式：**



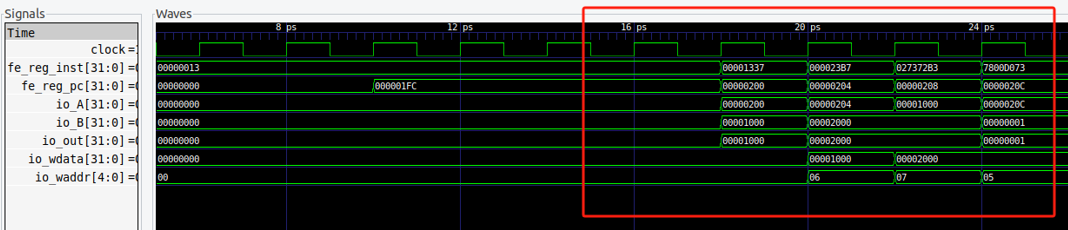
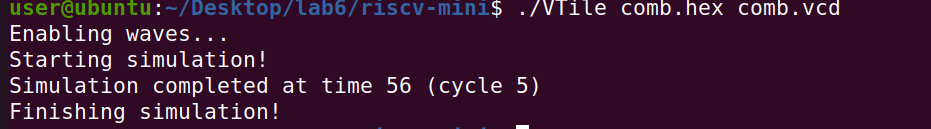
**3.4 comb x5, x6, x7转换成机器码的十六机制形式为 027372b3。因此处指令存储为小端模式，故我们需要将十六进制数插入到第一个红线 的前面。**



**3.5接着需要在主目录下一次执行 make 和 make verilator命令**



**4.1生成波形文件**



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

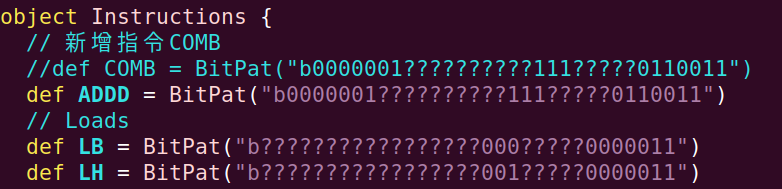
**5.自行设计其他功能指令，并验证设计是否正确**

自定义指令：addd rd, rs, rt

功能：R 型指令，读取寄存器 rs 和 rt 的值，计算 2 \* rs + rt ，将结果写回寄存器 rd

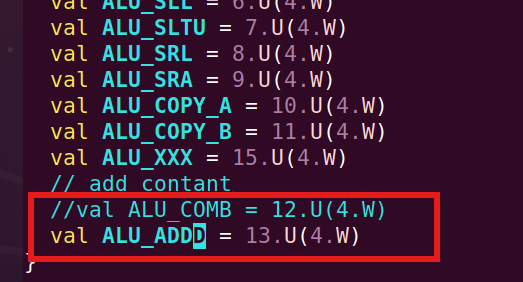
按照上面的步骤进行设计：

1. 在 Instructions.scala 中将该指令的 opcode 、funct3 、funct7 分别设置为 0110011 、111 、0000001 ，并使用 BitPat() 函数设置 comb 指令的比特模式，如下图所示。

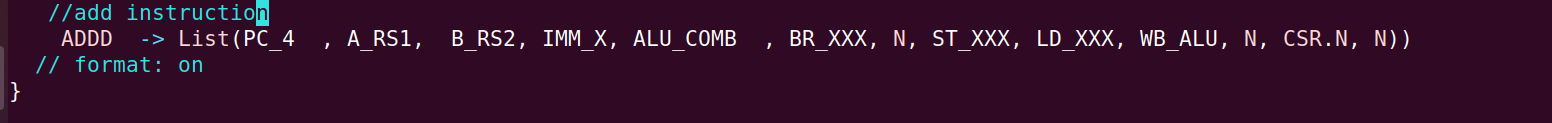


**2. 添加译码常量和译码映射**

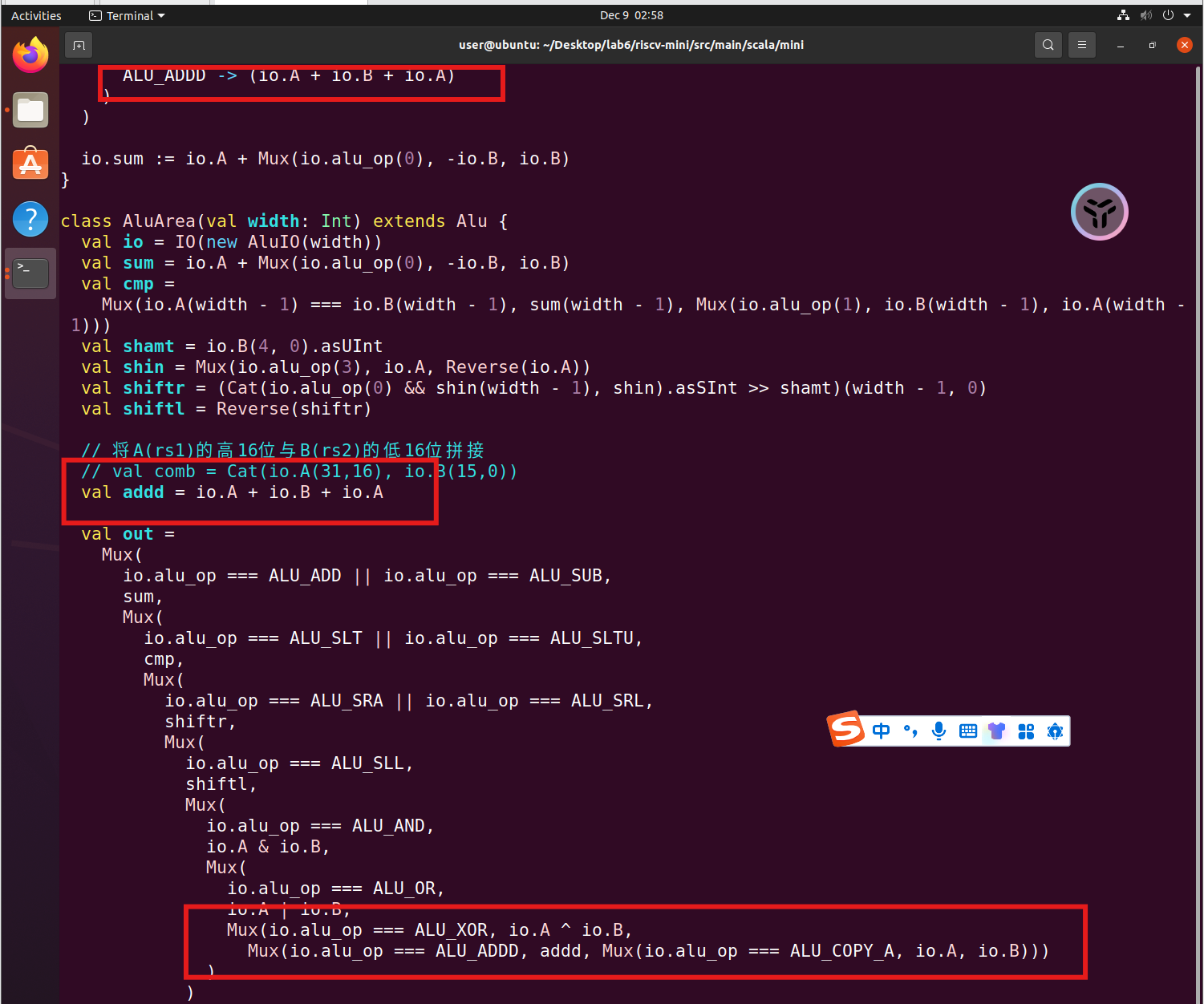
同 **1.2** ，在 Alu.scala 中为 addd 指令添加译码常量，如下图所示。



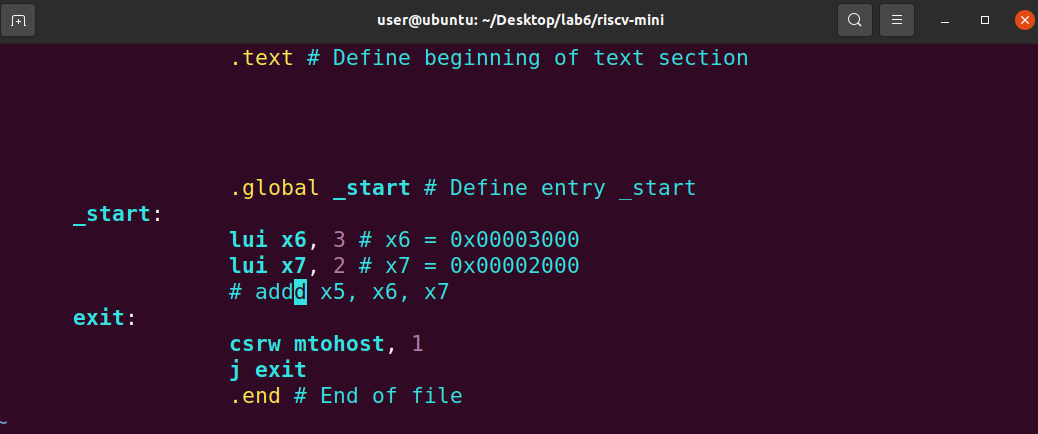
3．在 Control.scala 中为 addd 指令添加译码映射，如下图所示



4. 在 Alu.scala 中实现 newadd 指令的执行，即读取寄存器 rs 和 rt 的值，计算 2 \* rs + rt ，将结果写回寄存器 rd ，如下图所示。



**进行测试**



**生成.hex文件后进行修改生成vcd文件**

**最后的结果为：**

程序中 3 条主要的指令的十六进制如下。

|  |  |  |
| --- | --- | --- |
| **指令** | **十六进制机器码** | **说明** |
| lui x6, 3 | 0x000033337 | x6 = 0x00003000 |
| lui x7, 2 | 0x000023b7 | x7 = 0x00002000 |
| addd x5, x6, x7 | 0x027372b3 | x5 = 2 \* x6 + x7 = 0x00008000 |

波形如下图所示。

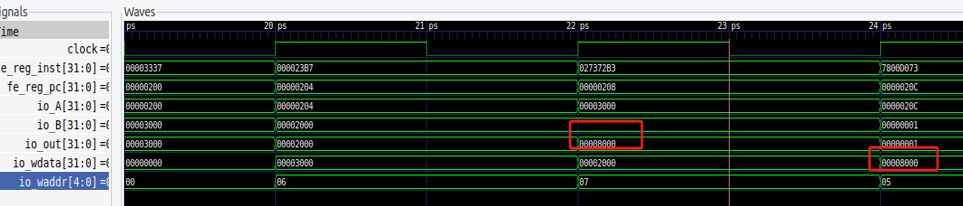


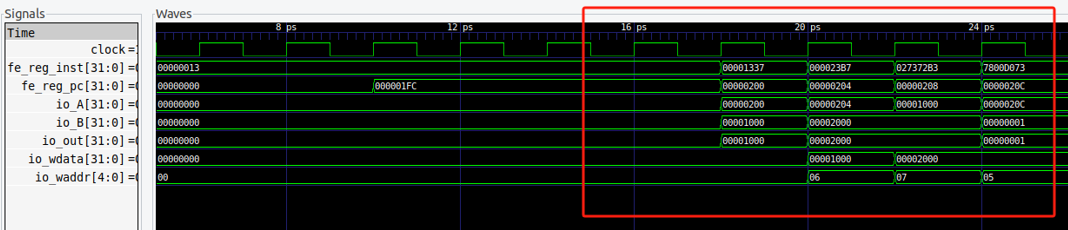
图 3.9 ：波形图

观察到程序读取 x6 和 x7 的值后，计算 2 \* x6 + x7 的值，将结果写回 x5 。

**五、实验结果**

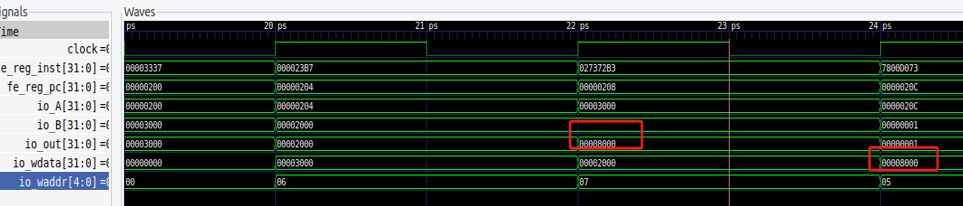
1. **测试 comb 指令**

|  |  |  |
| --- | --- | --- |
| **指令** | **十六进制机器码** | **说明** |
| lui x6, 1 | 0x00001337 | x6 = 0x00001000 |
| lui x7, 2 | 0x000023b7 | x7 = 0x00002000 |
| comb x5, x6, x7 | 0x027372b3 | x5 = cat(x6(31:16), x7(15:0)) = 0X00002000 |



2. **测试 addd 指令**

|  |  |  |
| --- | --- | --- |
| **指令** | **十六进制机器码** | **说明** |
| lui x6, 3 | 0x000033337 | x6 = 0x00003000 |
| lui x7, 2 | 0x000023b7 | x7 = 0x00002000 |
| addd x5, x6, x7 | 0x027372b3 | x5 = 2 \* x6 + x7 = 0x00008000 |



观察到程序读取 x6 和 x7 的值后，计算 2 \* x6 + x7 的值，将结果写回 x5 。

**六、实验总结与体会**

本次新增指令实验是对RISC-V mini处理器架构进行修改，以支持自定义指令。实验目的在于深入了解RISC-V处理器的内部结构，并通过新增指令的设计与实现，验证对指令集架构的理解程度, 除了实验要求的comb指令外，我设计了一个 R 型指令 addd ，该指令的比特模式串、译码和 comb 指令基本相同, 我对RISC-V处理器的设计和指令集有了更深入的了解。通过实际操作，我不仅学到了数字逻辑设计的实际应用，还提升了对计算机体系结构的理解.

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