**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： 数学科学学院**

**专 业： 信息与计算科学（数学与计算机实验班）**

**指 导 教 师： 罗秋明**

**报告人： 王曦 学号： 2021192010 班级： 数计**

**实 验 时 间： 2023年12月14日**

**实验报告提交时间： 2023年12月14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。

**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。**

1.1**添加 comb 指令的比特模式串**

comb 是 R 型指令，RISC-V 中 R 型指令的格式如下图所示。

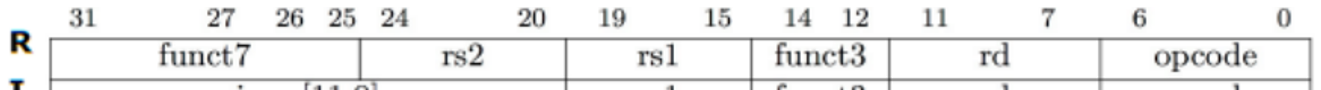


图 1.1 ：RISC-V 的 R 型指令格式

为避免 comb 指令与 RISC-V 已有的指令冲突，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。

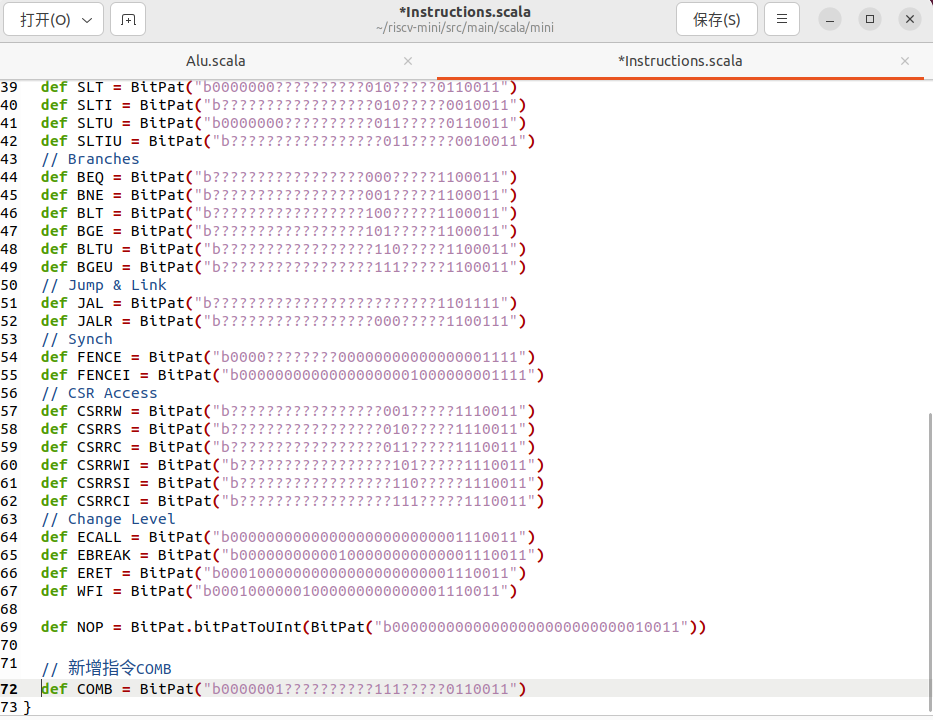


图 1.2 ：添加 comb 指令的比特模式串

1.2 **添加译码**

在 Alu.scala 的译码部分，按 comb 指令的模式串添加常量 ALU\_COMB ，以让译码器能输出正确的信号，如下图所示。

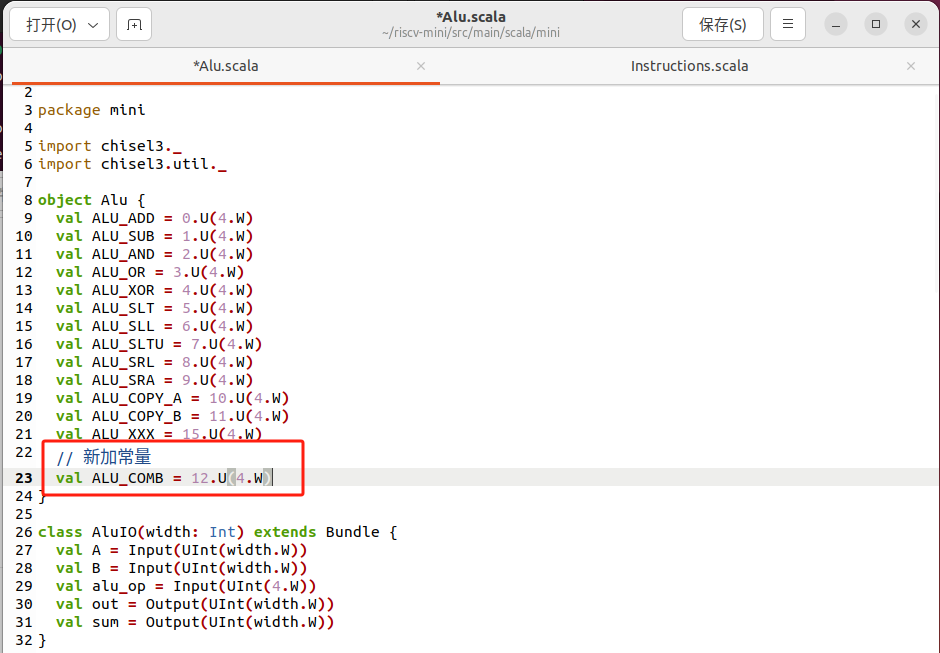


图 1.3 ：添加常量 ALU\_COMB

1.3 **添加译码映射**

在 Control.scala 中为 comb 指令添加译码映射。该指令执行后 PC 加 4 ，从寄存器文件读取数据 rs1 和 rs2 进行拼接，再将 ALU 输出的拼接结果写回寄存器文件中，如下图所示。

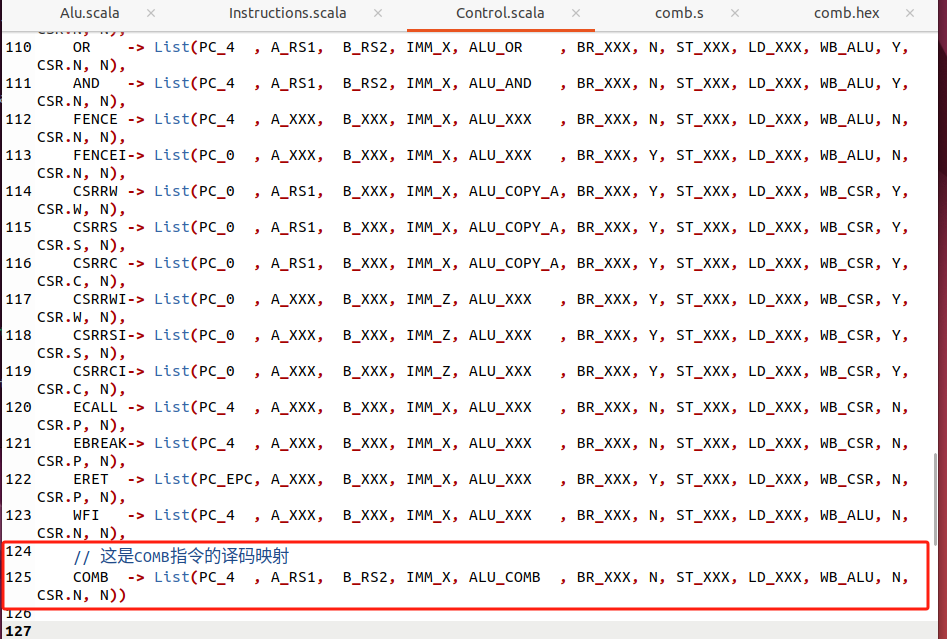


图 1.4 ：添加 comb 指令的译码映射

1.4 **实现执行**

在 Alu.scala 中实现 comb 指令的执行过程，即将 rs1 的高 16 位和 rs2 的低 16 位拼接为 32 位整数输出，如下图所示。

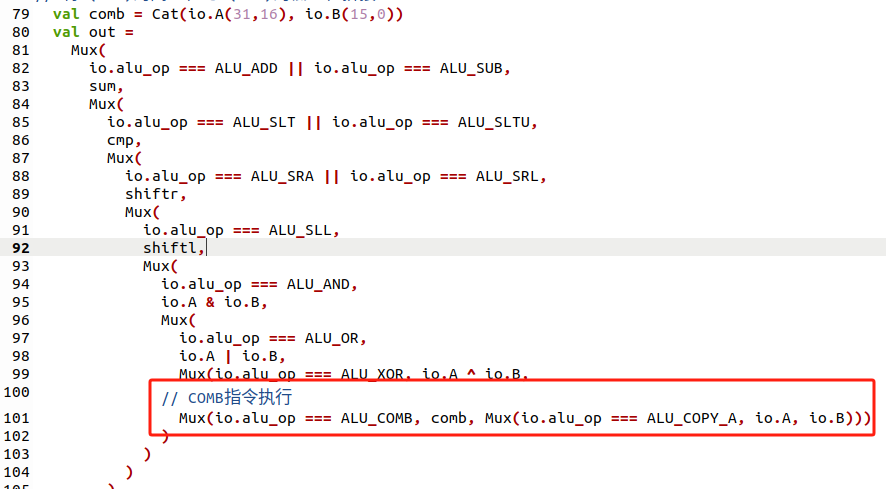
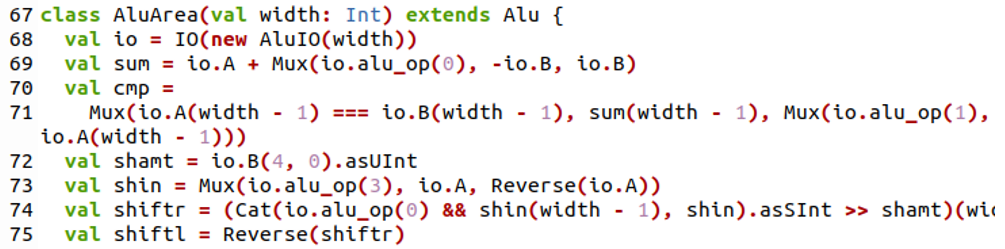
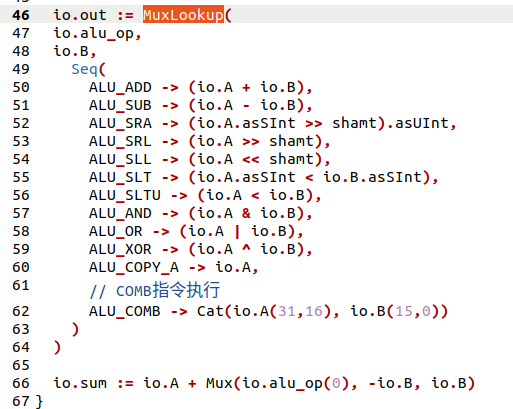


图 1.5 ：comb 指令的实现

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。**

2.1 **创建十六进制程序**

如下图创建测试 comb 指令的汇编代码 comb.s ，此处注释掉 comb 指令是因为它不能被编译器识别，可在生成该文件对应的十六进制文件后再加入 comb 指令对应的十六进制。

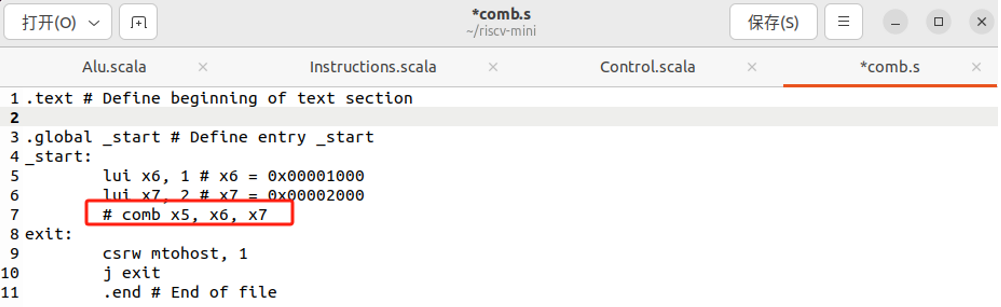


图 2.1 ：测试汇编程序

用如下命令编译 comb.s 。

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

在 comb.s 同目录下得到二进制文件 comb ，如下图所示。

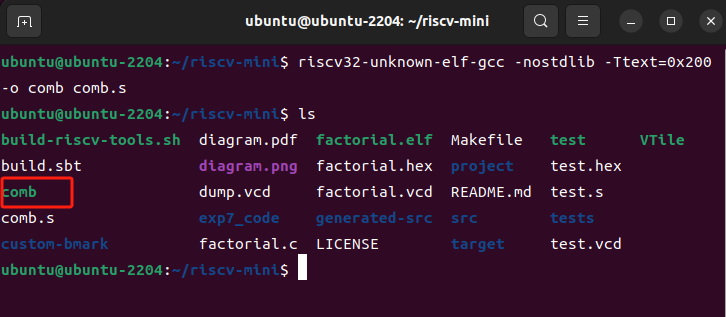


图 2.2 ：得到二进制文件 comb

用如下命令将 comb 生成十六进制文件 comb.hex 。

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

在 comb 同目录下得到十六进制文件 comb.hex ，如下图所示。

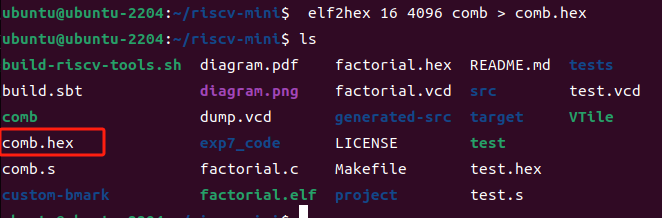


图 2.3 ：得到十六进制文件 comb.hex

在 comb.hex 的第 33 行找到指令 lui x6, 1 和指令 lui x7, 2 的机器码对应的十六进制，同时观察到它们以小端法存储，如下图所示。

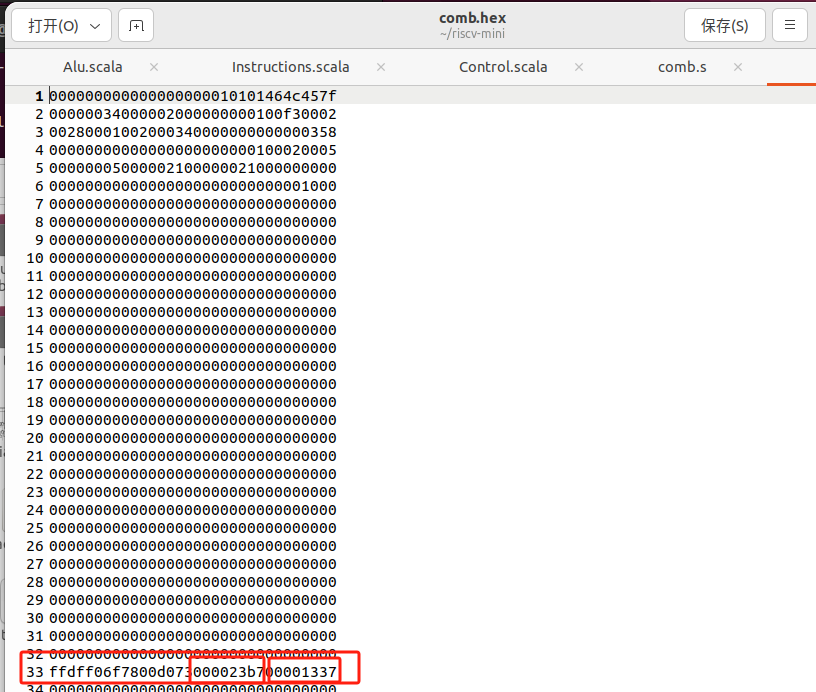


图 2.4 ：指令 lui x6, 1 和指令 lui x7, 2 的机器码对应的十六进制

指令 comb x5, x6, x7 对应的机器码的十六进制为 0x027372b3 ，因采用小端法存储，故需将其插入到上两条指令对应的机器码之前，如下图所示。



图 2.5 ：插入指令 comb x5, x6, x7 对应的机器码的十六进制

2.2 **生成波形**

执行如下的命令，生成 VTile 可执行文件和波形文件。

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

命令运行结果如下图所示。

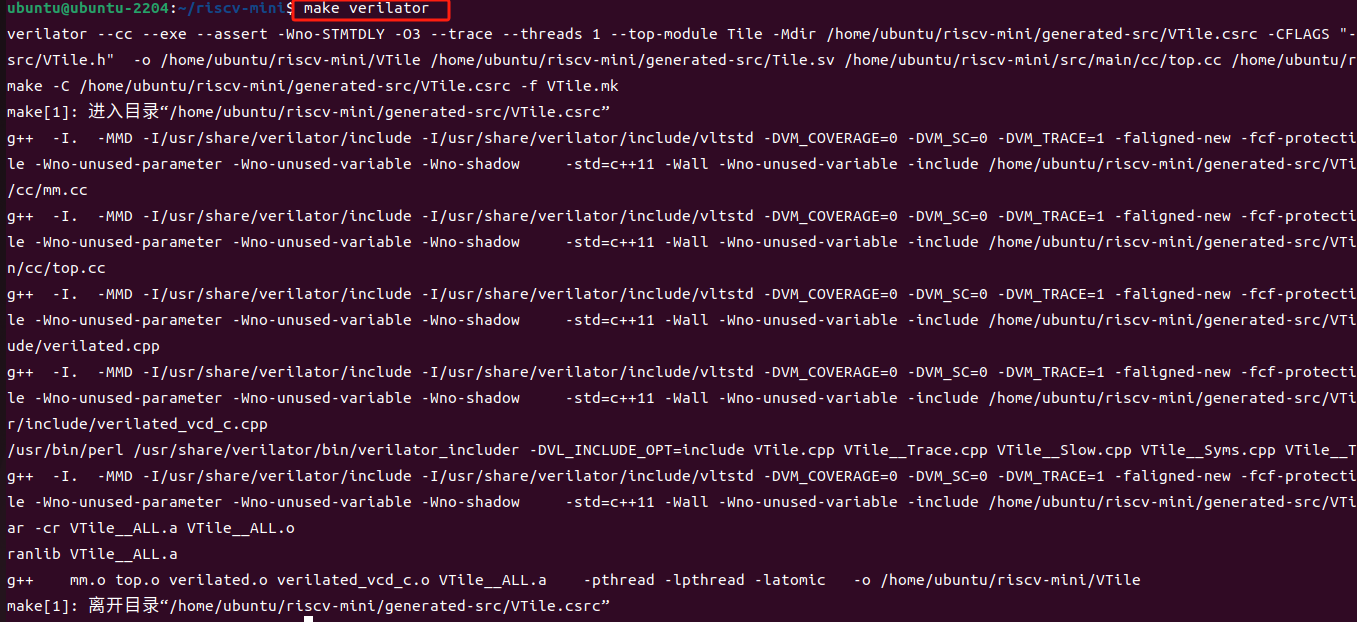
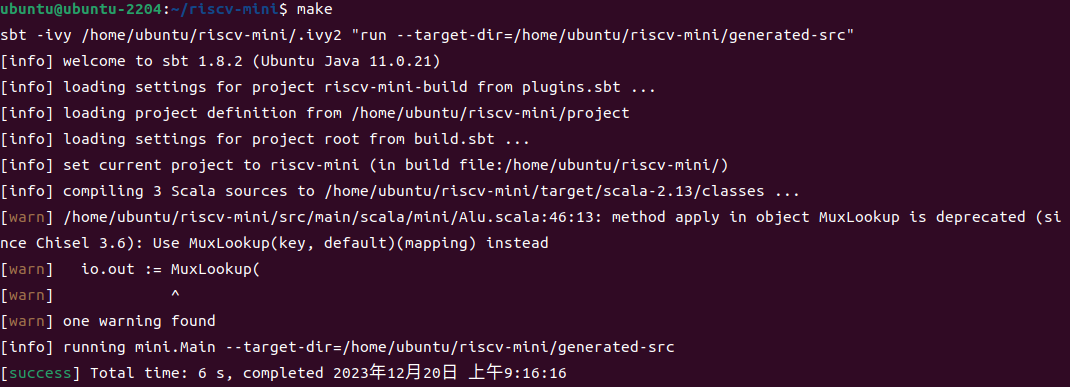


图 2.6 ：make 命令和 make verilator 命令的运行结果

在同目录下得到波形文件 comb.vcd ，如下图所示。

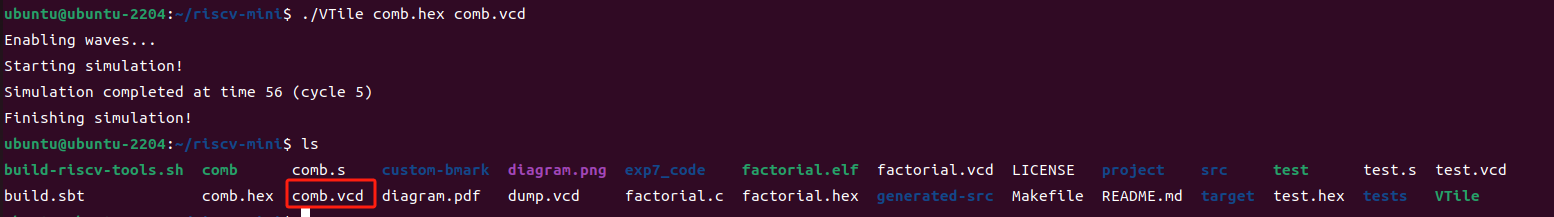


图 2.7 ：同目录下得到波形文件 comb.vcd

2.3 **观察波形**

执行如下命令，用 GTKWave 打开 comb.vcd 。

|  |
| --- |
| ./VTile comb.hex comb.vcd |

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, 1 | 0x00001337 | x6 = 0x00001000 |
| lui x7, 2 | 0x000023b7 | x7 = 0x00002000 |
| comb x5, x6, x7 | 0x027372b3 | x5 = cat(x6(31:16), x7(15:0)) = 0X00002000 |

波形如下图所示。

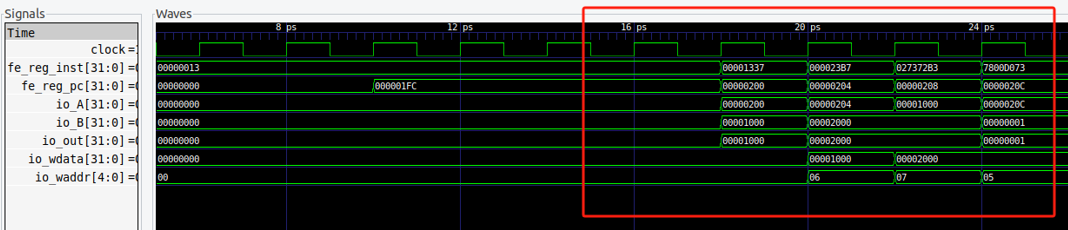


图 2.8 ：波形图

观察到程序读取 x6 和 x7 的值后进行拼接，结果 0x000020000 写回 x5 。

3. **自行设计其他功能指令，并验证设计是否正确。**

自定义指令：newadd 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 。

3.1 **添加 newadd 指令的比特模式串**

同 **1.1** ，参考 RISC-V 的 R 型指令的格式。

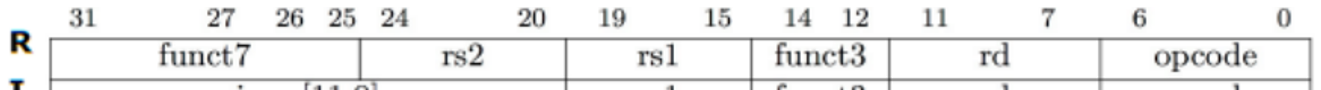


图 3.1 ：RISC-V 的 R 型指令格式

为避免 comb 指令与 RISC-V 已有的指令冲突，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。

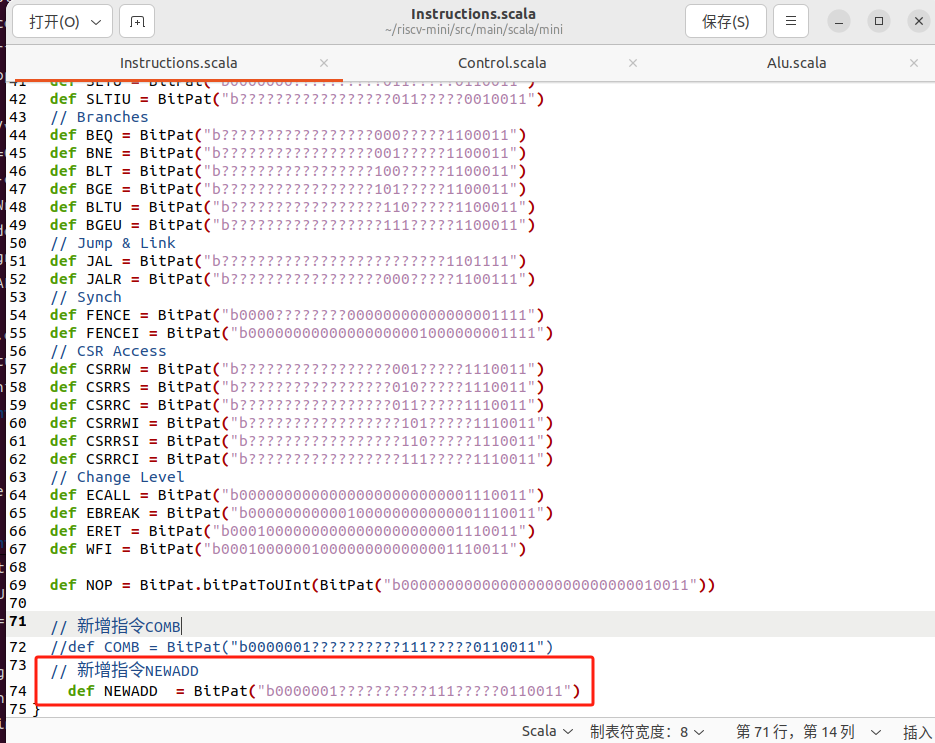


图 3.2 ：设置 newadd 指令的比特模式串

3.2 **添加译码常量和译码映射**

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



图 3.3 ：为 newadd 指令添加译码常量

同 **1.3** ，在 Control.scala 中为 newadd 指令添加译码映射，如下图所示。

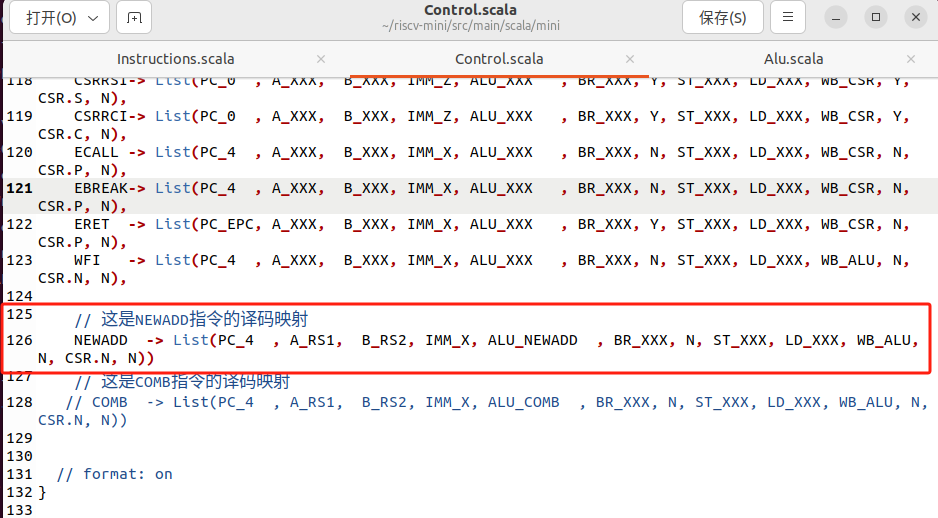


图 3.4 ：为 newadd 指令添加译码映射

3.3 **实现执行**

同 **1.4** ，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。

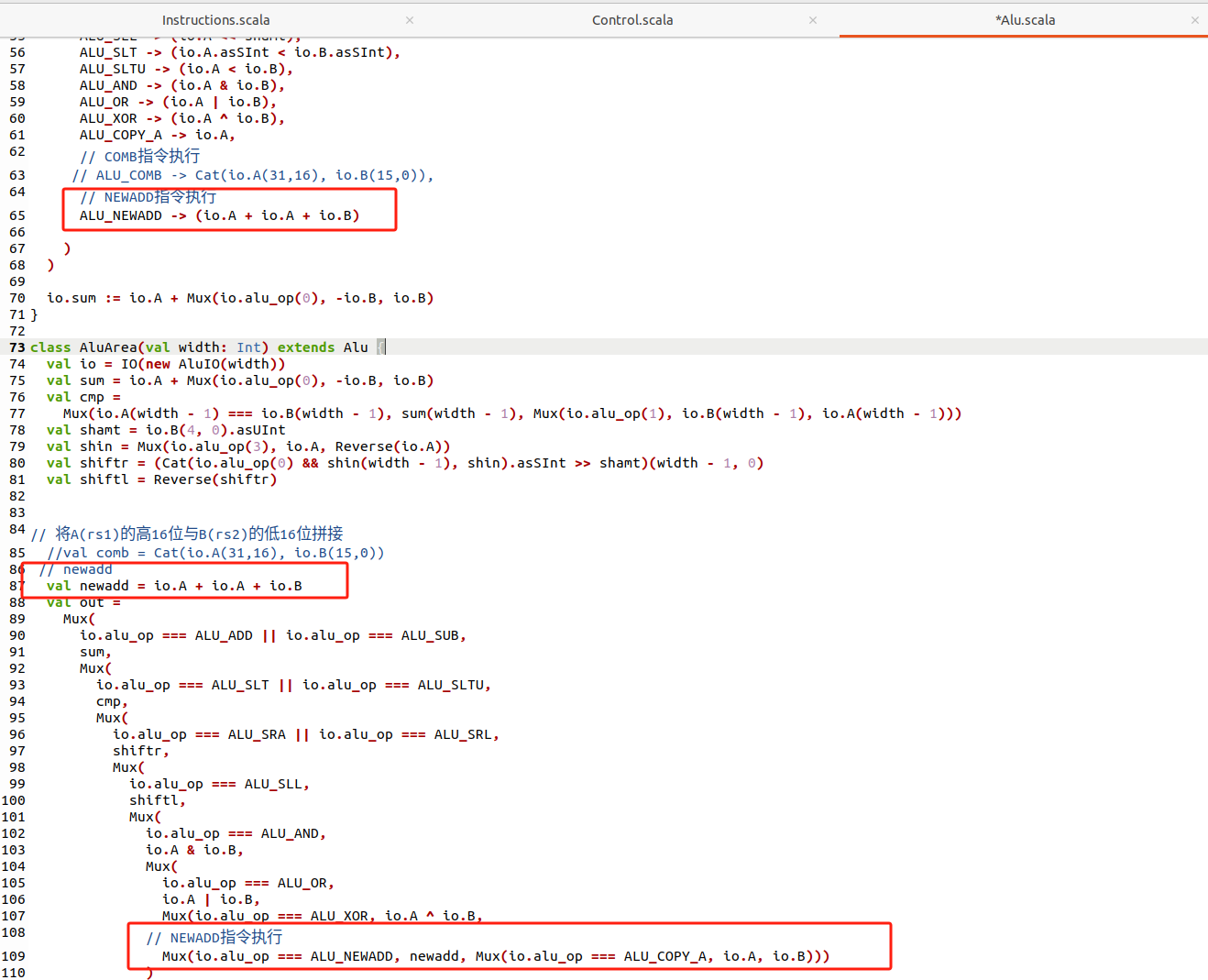


图 3.5 ：实现 newadd 指令的执行

3.4 **测试**

同 **2.1** ，创建汇编程序 newadd.s ，注释掉 newadd 指令，如下图所示。

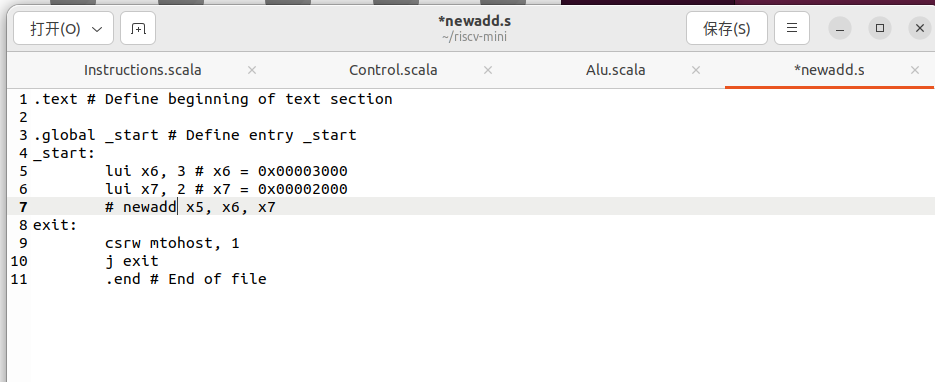


图 3.6 ：创建汇编程序 newadd.s

执行如下命令编译得到二进制文件 newadd ，并将其转化为十六进制文件 elf2hex 16 4096 newadd > newadd.hex 。

|  |
| --- |
| riscv32-unknown-elf-gcc -nostdlib -Ttext=0x200 -o newadd newadd.s  elf2hex 16 4096 newadd > newadd.hex |

在 newadd.hex 中找到指令 lui x6, 3 和指令 lui x7, 2 的机器码对应的十六进制，指令newadd x5, x6, x7 的十六进制机器码为 0x027372b3 ，采用小端法存储，将其插至上述两指令之前，如下图所示。

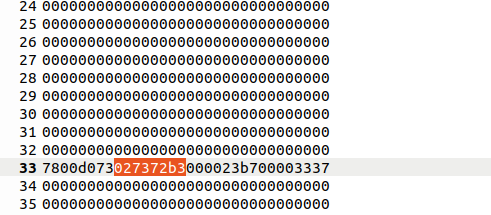


图 3.7 ：插入指令newadd x5, x6, x7 的十六进制机器码

执行如下命令生成 VTile 可执行文件和波形图 newadd.vcd 。

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

命令的执行结果如下图所示。

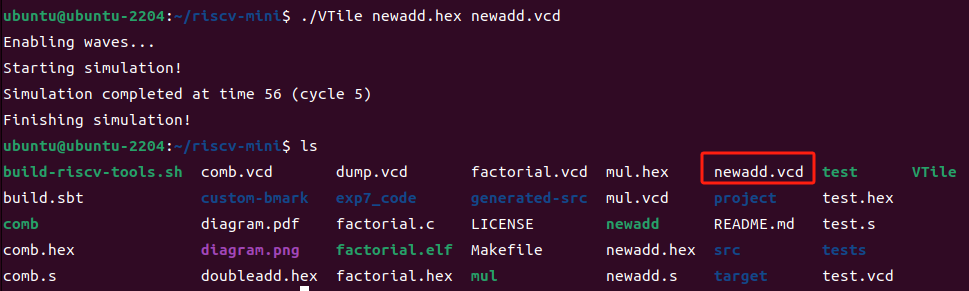
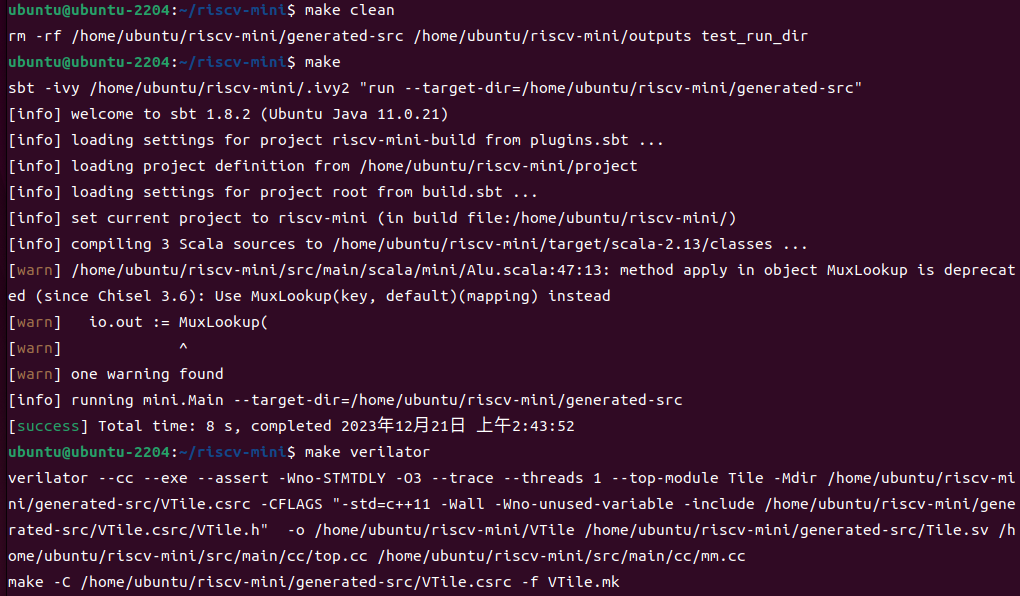


图 3.8 ：生成波形文件 newadd.vcd

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 |
| newadd x5, x6, x7 | 0x027372b3 | x5 = 2 \* x6 + x7 = 0x00008000 |

波形如下图所示。

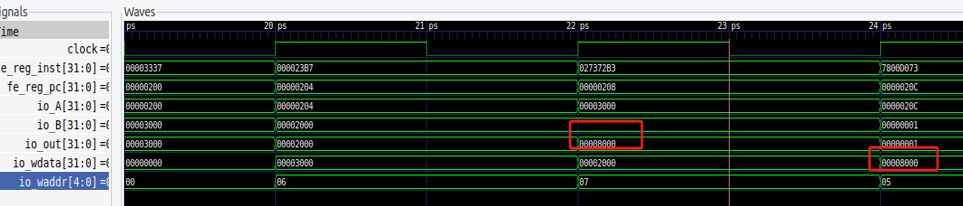


图 3.9 ：波形图

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**

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, 1 | 0x00001337 | x6 = 0x00001000 |
| lui x7, 2 | 0x000023b7 | x7 = 0x00002000 |
| comb x5, x6, x7 | 0x027372b3 | x5 = cat(x6(31:16), x7(15:0)) = 0X00002000 |

波形如下图所示。

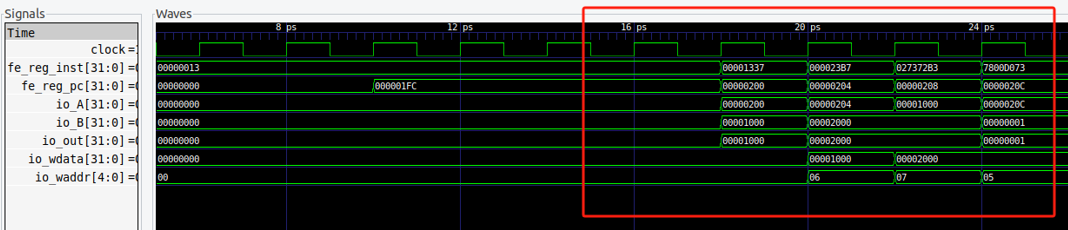


图 5.1 ：波形图

观察到程序读取 x6 和 x7 的值后进行拼接，结果 0x000020000 写回 x5 。

2. **测试 newadd 指令**

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 |
| newadd x5, x6, x7 | 0x027372b3 | x5 = 2 \* x6 + x7 = 0x00008000 |

波形如下图所示。

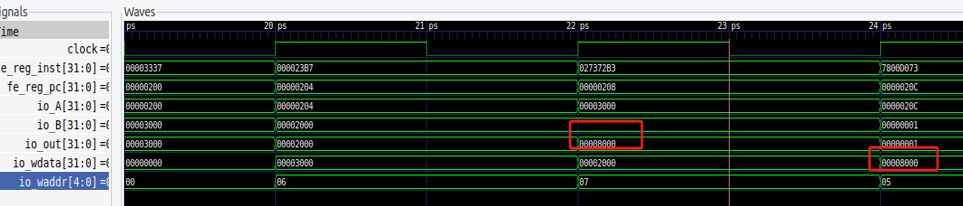


图 5.2 ：波形图

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 。

**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。以下是我对本次实验的心得体会：

**实验总结**：

（一）**修改数据通路**

在数据通路中新增指令comb rs1, rs2, rd，该指令采用R型指令格式。为了实现该指令的功能，需要在数据通路中添加逻辑单元，将rs1的高16位和rs2的低16位拼接成32位整数，并将结果保存到rd寄存器。这涉及到对ALU（算术逻辑单元）和寄存器文件的修改。

（二）**执行指令并观察仿真波形**

在处理器上加载修改后的指令集，特别是新增的comb指令。通过仿真工具观察指令的执行过程，检查波形图以确保新指令在处理器中的正确执行。这一步让我对数字逻辑和处理器内部结构有更深入的理解。

（三）**自定义指令**

除了实验要求的comb指令外，我设计了一个 R 型指令 newadd ，该指令的比特模式串、译码和 comb 指令基本相同，只有实现需要稍加修改。

**实验心得**：

（一）**深入理解RISC-V架构**

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。了解每个部分的作用和关系，对于理解计算机体系结构和指令集设计有了更直观的认识。

（二）**实践数字逻辑设计**

通过对数据通路的修改，我实践了数字逻辑设计的基本原理。理解如何在处理器中插入新指令，涉及到ALU、寄存器文件等关键组件的修改，增进了我的数字电路设计能力。

（三）**验证与调试能力**

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。这是一个关键的实际技能，对于在处理器设计和软硬件协同开发中都是至关重要的。

综上，本次新增指令实验使我对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。这样的实践性实验为我今后深入学习计算机体系结构和处理器设计打下了坚实的基础。

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