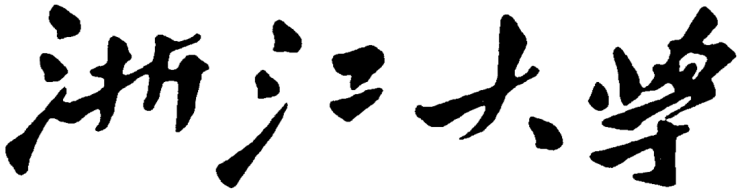

本科实验报告

RISC-V 指令集仿真器

课程名称：	计算机组成与设计
姓　　名：	箫宇
学　　院：	信息与电子工程学院
系：	
专　　业：	信息工程
学　　号：	
指导老师：	刘鹏

2023 年 7 月 2 日

浙江大学实验报告

专业：信息工程
姓名：箫宇
学号：
日期：2023 年 7 月 2 日
地点：教 7-108

课程名称：计算机组成与设计　指导老师：刘鹏　成绩：
实验名称：RISC-V 指令集仿真器　实验类型：代码编写　同组学生姓名：无

一　代码实现思路

1. instforms.cpp

我们在 informs.cpp 文件中的主要工作为实现要求指令的编码函数，即依据函数传入的参数（即寄存器的编号或者立即数），为对应指令类型的结构体进行赋值。

在赋值的过程中，以 encodeLb 函数为例讲解代码思路：

Listing 1: decodeLb

```
bool
IFormInst::encodeLb(unsigned rdv, unsigned rs1v, int offset)
{
  //check the reg number
  if(rdv > 31 or rs1v > 31)
    return false;

  if(offset > (1<<11) or -offset < -(1<<11))
    return false;

  fields.opcode = 0x03;
  fields.rd = rdv & 0x1f;
  fields.funct3 = 0x0;
  fields.rs1 = rs1v & 0x1f;

  #pragma GCC diagnostic push
  #pragma GCC diagnostic ignored "-Wconversion"
    fields.imm = offset;
  #pragma GCC diagnostic pop

  return true;
```

可以看到，在函数起始处，需要先对传入的寄存器操作数范围进行检查，因为 RISC-V 中寄存器数目为 32，所以寄存器操作数的值不能超过 31。在完成了对寄存器操作数范围的检查后，我们需要检查传入的偏移量 offset 的范围。

通过查看 union IFormInst 的定义可以发现其中定义了结构体 fields，然后查阅 lb 指令的指令结构对 fields 中的各个成员进行赋值，此处的取与操作是为了再次防止操作数越界。

代码最后以 #pragma 开始的三行代码也是为了再次确保立即数未超出范围。

2. decode.cpp

decode.cpp 文件主要关注从 1368 行开始的 decode 函数，输入一条指令 inst，需要给出其对于的操作数 op 和指令条目 InstEntry。而需要我们完成的部分为 l5，l8，l13，l24，l25，l27 和 l12 的部分内容，下面将以 l5 部分为例讲解代码思路

Listing 2: l5 部分

```
l5:  // 00101   U-form
    {
        UFormInst uform(inst);
        op0 = uform.bits.rd;
        op1 = uform.immed();
        return instTable_.getEntry(InstId::auipc);
    }
    return instTable_.getEntry(InstId::illegal);
```

通过查阅 instforms.hpp 中对不同命令的映射，得到 op0,op1 对应的传入值，将其赋值；如果有 funct3，还需要根据 funct3 值的不同给出不同的 InstEntry。如果最终 funt3 没有与所有指令匹配，则返回 illegal。

3. Hart.cpp

Hart.cpp 文件主要是功能是实现指令的执行过程，下面将以 execAnd 为例分析此部分代码的实现思路。

```
template <typename URV>
void
Hart<URV>::execAnd(const DecodedInst* di)
{
  URV v = intRegs_.read(di->op1()) & intRegs_.read(di->op2());
  intRegs_.write(di->op0() , v);

}
```

在这部分代码里面，变量的类型主要分为两种，有符号数 SRV 以及无符号数 URV。

主要需要调用的函数有：

(1) intRegs_.read(): 读取寄存器的值

(2) intRegs_.write(): 在寄存器中写入值

(3) 处理立即数的 op0As(URV/SRV)

我们需要利用上述函数在 Hart.cpp 中实现各条指令的执行方式。例如 and：将两个源寄存器中的值取出取与在写入目标寄存器中。

二 调试过程

1. 单步运行输出信息意义

如下图所示，利用交互模式进行单步调试时输出的信息含义如下：

图 1: 输出信息

2. 调试思路

(1) 可以手动对机器码进行译码操作，看是否和后面的译码结果一样; 如果不一致，多半是 decode.cpp 里面对应的指令有问题

(2) 观察指令对各类资源的修改是否正确；如果不对，有可能是 Hart.cpp 对应指令有问题

(3) PC 指针相关的就看指令地址是否正确跳转

3. 调试结果

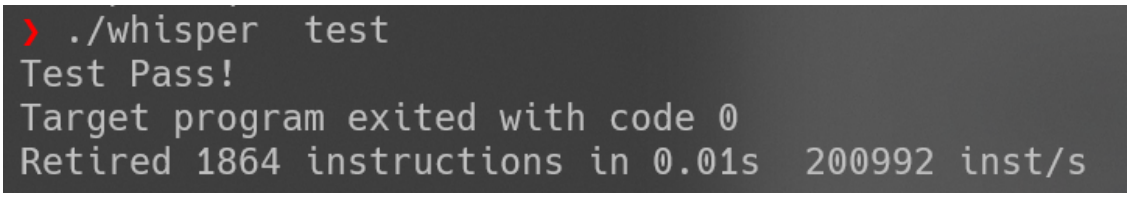

图 2: 测试结果

三 收获

在本次实验中，通过实现部分指令的编码、执行过程，极大地加深了自己对于各种指令的熟悉程度：不仅更加深入地掌握了它的功能，同时也对指令编码有了更深的了解。

不过对于本次实验也有一些小建议：Project 可以搭配 Docker 镜像一起发布。因为在实验过程中，大部分同学都遇到了诸如 boost 库无法安装好之类的环境配置问题，花费了大量的时间在编译环境的搭建上。而且最后大家遇到的问题也是千奇百怪，而 Docker 镜像则是一个很好的解决方案，实测利用镜像可在半小时内搭好环境。可以让大家从环境搭建中解放出来，将精力投入到编码过程中。