labA/labS:LC3

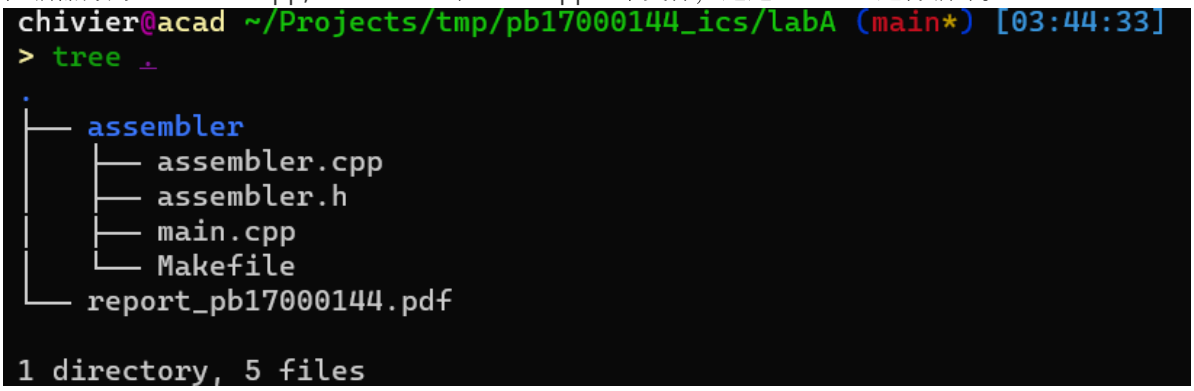
实验目的

- 实现将汇编语言翻译为机械码的汇编器
- 实现将机械码进行模拟的模拟器

实验过程

汇编器

汇编器分为assember.cpp,assember.h和main.cpp三个文件，通过makefile进行编译。



assember.h

- 首先包含了各种c++自带的库
- 定义或声明了各种常量，如指令长度，不同的指令操作码
- 一些枚举类型和内敛函数
- 定义了一个值的类，定义了一个map类，用于把标签和值一一对应
- 定义了一个assember类，用于翻译汇编语言

assember.cpp

主要是对assember.h中定义的类内的各种函数的定义，其中最主要的是汇编过程，即assember函数。
过程大致如下：

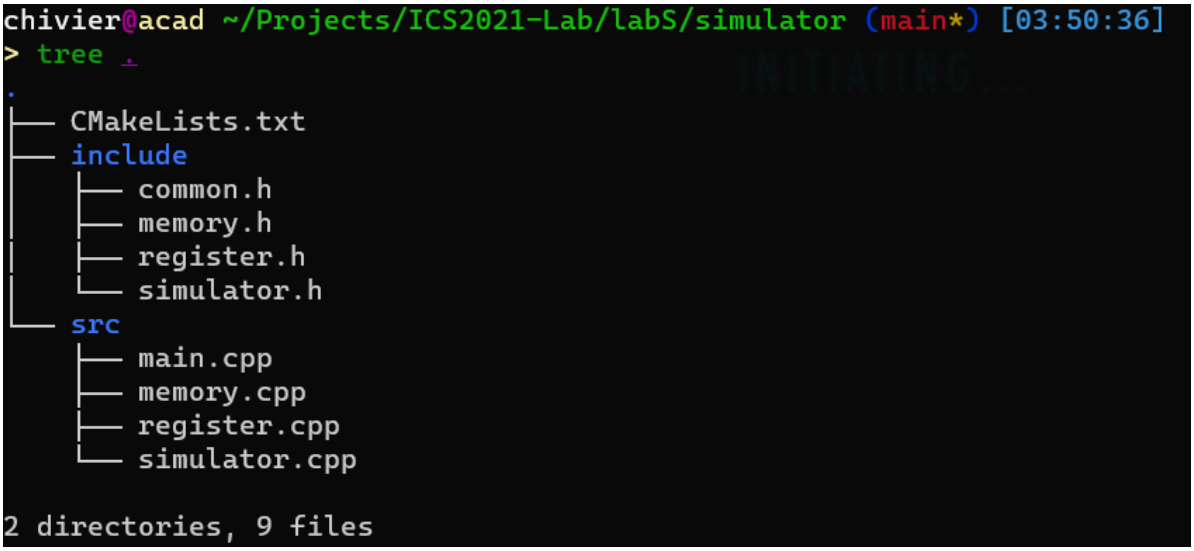
1. 首先将文件打开，并整行的读入文件，若该行为空行，或者没有指令与注释，就不读入。
2. 将读入的文件分为内容部分（即指令），原文件行，文件的类型（注释，指令，伪代码，不确定等），注释部分，地址（初始为0）
3. 然后就是相当于预处理一遍文件，确定各个汇编指令的起始地址，和代码的具体种类（将上一步中中不确定的代码种类确定下来），将标签与对应的代码匹配起来构成一个“map”。
4. 之后，先是确定输出的各种模式，然后打开输出文件，准备将汇编语言转化为的机械码输入到文件中，根据上面确定的指令的不同类别来具体逐个按照格式输入并翻译指令，翻译途中要去除标签，若遇到标签，则需要在上一步中构造的“map”中寻找具体得指令
5. 关闭文件，返回即可

main.cpp

根据终端传来的各种参数，确定不同的模式和输入，输出文件的名称，最后调用assember函数，并输出最终状态。

模拟器

模拟器部分大体上分为四个头文件和四个cpp文件，通过cmake来编译多个文件。



common.h

其中包含了#progma once 表示该文件只会执行一次，这样可以避免宏名冲突，提高程序的效率，还有include各种c++中的库，额外包含了一个boost中的库，声明了各种全局变量。

memory.h

定义了一个命民空间virtual_machine_nsp,其中包含把大小为16的字符串翻译为int16_t类型的内联函数，定义了一个memeory类，有0xFFFF大小的内存空间和各种函数成员，并对其内存初始化为0。

register.h

同样在virtual_machine_nsp的命民空间中，包含了10个寄存器的枚举类型和一个寄存器的数组与其重载输出的声明

simulator.h

在virtual_machine_nsp的命民空间中，包含了操作码的枚举类型，和一个virtual_machine_tp 类，含有寄存器和内存成员，与各种函数成员的声明。virtual_machine_tp 是最主要的一个类。

memory.cpp

是对在头文件中声明的memory_tp类中的函数的定义，包括从文件中读取内存指令，和获取或某个内存指令。

register.cpp

是对重载输出reg寄存器的定义

simulator.cpp

是主要的一个cpp文件

1. 首先定义了符号位扩展和更新条件码寄存器的函数操作。
2. 针对各种不同的指令设计了不同各种操作的函数
3. 从输入文件中读取其中的内容读取，并打开读入预设的寄存器文件，由此来初始化寄存器的值
4. 从刚刚读入的文件指令中一个一个的读，判断操作码，并调用对应的函数，最后返回寄存器的状态

main.cpp

先初始定义全局变量，再设置一些命令行参数对应的不同操作，最后是一步一步执行程序，并计算周期，输出最终寄存器的状态，模拟结束。

实验总结

- 本次实验在助教给出的框架下完成，读懂框架的构造后，完成代码的补充就相对与从零开始就简单很多。但需要学习一些c++的知识
- 实验本身进行的比较顺利，结果也符合LC3模拟器和汇编器的要求。