Lab A & Lab S: LC-3 Online Judge

1. 实验概述

在本实验中，你将不再仅仅是 LC-3 的使用者。你需要从零开始实现一套完整的工具链，能够将人类可读的汇编代码转换为机器码，并在虚拟环境中执行，最后通过一个自动化脚本将二者结合，实现一个能够自动判定代码正确性的“本地 OJ (Online Judge)”。核心目标：验证同学在 Lab1 和 Lab2 中编写的代码正确性。

1.1 实验目标

- 掌握汇编原理：实现双遍扫描，理解符号表与指令编码
- 理解指令周期：实现指令集模拟器，深入理解取值、译码、执行、访存、写回的全过程
- 系统集成能力：通过脚本或主程序将各个组件串连起来，实现完成的本地测试程序

1.2 开发环境

- 推荐语言：C/C++/Python/Rust/JavaScript ...
- 提交形式：源代码、实验报告（设计文档+测试报告）、线下验收

2. Part 1： 汇编器

2.1 任务描述

编写一个程序，它接受一个 `.asm` 文本文件作为输入，输出一个 `.bin/.txt` 可读的机器码文本文件

2.2 核心功能要求

- **预处理**：能够忽略注释（以 `;` 开头）和空白行，处理大小写不敏感的指令
- **符号表构建 (Pass 1)**：
    - 遍历代码，计算每行指令的内存地址（LC-3程序通常由 `.ORIG` 指定起始地址）。
    - 识别 Label（标签），建立 `Label -> Address` 的映射表。
- **代码生成 (Pass 2)**：
    - 再次遍历代码，将汇编指令翻译为 16-bit 机器码。
    - **指令集支持**：必须支持所有 LC-3 标准指令
    - **伪指令支持**：`.ORIG`，`.FILL`，`.BLKW`，`.STRINGZ`，`.END`
- **错误处理**：对于未定义的标签、非法的立即数范围、语法错误，应输出具体的报错信息。

2.3 输入/输出示例

- Input (`test.asm`):

```
.ORIG x3000
ADD R0, R0, #1  ; R0 = R0 + 1
HALT
.END
```

- Output (`test.txt`):

```
0011000000000000
0001000000100001
1111000000100101
```

注意起始地址这一行可以不翻译，因为如果有多个 .ORIG，汇编器应该直接将全部地址展开，直接填充空白的 0，实际模拟器运行的时候只能指定一次程序开始运行的地址，具体可以参考 lc3tools 编译后的地址空间

3. Part 2：模拟器

3.1 任务描述

编写一个程序，它加载汇编器/手动编写的机器码文件，在内存中模拟LC-3的硬件行为

3.2 核心要求

- **存储器 (Memory)**：一个大小为 2^{16} (65536) 的数组，每个单元存储 16-bit 数据。
- **寄存器文件 (Register File)**：
  - 8个通用寄存器 (R0-R7)
  - PC 程序计数器。
  - IR 指令寄存器
  - CC：N, Z, P 标志位。
- （选做）内存映射 I/O
  - 模拟键盘输入、模拟显示器输出
  - 中断调用子程序

3.3 执行循环

模拟器的主循环应严格遵循以下的逻辑：

- **Fetch**: 从 `Memory[PC]` 读取指令到 IR， `PC = PC + 1`
- **Decode**: 解析 IR 中的操作码（Opcode）
- **Execute**: 根据指令执行操作（如更新寄存器、访问内存）
- **Update CC**: 如果指令修改了通用寄存器，需根据结果更新 N/Z/P 标志位

4. Part 3： OJ 集成

4.1 任务描述

你需要编写一个评测主程序，将汇编器和模拟器连接起来，实现输入源码到判定结果的自动化流程，可以选择将汇编器和模拟器都编译成可执行程序，用命令行脚本实现完整的评测，也可以选择将汇编器和模拟器写在一起编译

4.2 工作流程

你得程序接受三个参数：原始代码（汇编/机器码），标准输入数据，期望输出数据

- 编译阶段（如果输入是汇编）：
  - 调用你的汇编器，将汇编码转换成机器码
  - 如果汇编器报错，OJ直接判定为 Compile Error
- 运行阶段
  - 调用你的模拟器，模拟运行你生成或提供的机器码
  - 由于 Lab1 和 Lab2 的输入输出需要，你的模拟器应当能够指定程序运行前寄存器以及指定内存地址的值，同时在运行结束后，支持输出寄存器和指定内存地址的值，以便于程序的验证
    - 超时限制：如果你的程序在规定指令数内未遇到 `HALT`，判定为 Time Limit Exceeded
- 验证阶段
  - 模拟器输出运行后的寄存器和内存信息，你的程序比对指定寄存器/内存地址的值和期望输出数据
  - 严格匹配：输出 Accepted
  - 否则：判定为 Wrong Answer

5. 验收标准

- 本班所有同学**必须**选择实现一个汇编器/模拟器
- 如果同时实现了汇编器和模拟器，则需要实现完整的本地 OJ 功能可获得加分
- 汇编器部分具体要求：
  - 程序结束后输出符号表（每一个 Label 代表的内存地址）
  - 程序结束后可选择输出调试信息，例如每一条指令的内存地址，形如：

```
x3000 : 0001000000100001
x3001 : 1111000000100101
```

- 模拟器部分具体要求：
  - 程序开始运行前，可指定寄存器和指定内存地址的值（便于评测）
  - 像 lc3tools 一样，模拟器应当可以实现单步调试功能，每一步输出当前的寄存器信息
  - 程序结束运行后，输出程序运行的指令数、寄存器信息和指定内存地址的值（便于评测）
- OJ集成部分的具体要求：
  - 集成部分的程序可以是纯命令行的，也可以是 GUI 的（不会额外加分）

- 由于 LC-3 的特殊性，输入输出文件格式需要有一定的规定以便于程序评测，这一部分的规定可以自己指定，例如：程序从 x3100 读取输入，并在程序结束计算之后将值存到 x3101

```
# input
R0, 0
R1, 0
R2, 0
R3, 0
R4, 0
R5, 0
R6, 0
R7, 0
x3100, 100
# output
R0, -
R1, -
R2, -
R3, -
R4, -
R5, -
R6, -
R7, -
x3101, 200
```

又比如，如果你使用的 JavaScript 这类解释性语言，可以将输入输出写成代码动态加载，例如我们希望输入在 R0，输出在 R7 寄存器，可以这么书写评测的函数体

```
# 输入 程序按逗号分隔输入
100, 200, 300
```

```
// 从分割好的一个 testcase 中获得输入
const input = parseInt(testcase)
// 初始化寄存器
lc3.r[0] = input
// 返回预计的正确答案
return (input + input) % 65536
```

```
// 返回程序输出的答案
return lc3.r[7]
```

- 由于模拟器不要求实现对IO，TRAP的模拟，所以 OJ 无需实现对 Lab3 Lab4 Lab5 的评测