**学 号：201814600205**





**2019-2020春季学期**

**计算机原理课程设计**

**作业题目：模型机设计**

**学生姓名：马昕**

**专业班级：18计算机2班**

**学 院：人工智能学院**

**任课教师：于复兴 副教授**

**2020年7月4日**项目基本信息

|  |  |
| --- | --- |
| 项目概述 | 本项目使用python语言开发，UI界面使用pyqt设计而成，整个设计独自一人完成，并且打包为exe文件，可以在任何windows系统下完美运行。  本项目可以将事先编写好的汇编代码文件导入并展现给用户，然后将汇编代码转化为机器码，再根据机器码跳转到相应的微指令入口，然后依次执行并记录位操作，最后可以点击保存按钮将本次运行的微操作保存下来。  通过本项目，使用者可以更加深入的学习计算机内部数据的流通方式，了解到计算机最底层的工作原理。本项目还可以实时观测到各个寄存器的值，更加方便了使用者学习。 |

# 1系统需求分析

## 1.1 功能需求分析

1. 导入文件：导入事先编写好的汇编代码文件。
2. 转机器码：根据导入的汇编文件，将每一条指令转化为机器码。
3. 微指令入口：根据操作码找到相应微指令入口。
4. 数据显示：实时显示各个寄存器的值。
5. 位操作记录：记录每一步位操作，并且可以导出为文件。

## 1.2系统总体结构设计

计算机模拟机的设计,主要划分为以下三个模块:

1. 读入：读入事先写好的汇编文件。
2. 运行：根据微指令模拟计算机底层的运行状态。
3. 显示：实时显示各个寄存器的数值以及微操作。

根据需求分析的结果,总体结构如图2-1所示。

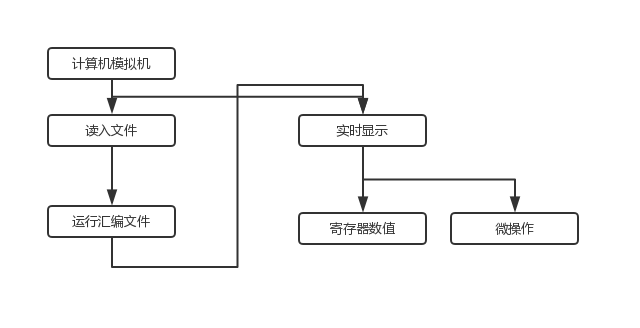


图2.1系统总体结构

# 2详细设计

## 2.1 数据结构设计

1. 类的设计

**表3-1** ToolFunction**类成员列表**

|  |
| --- |
| class ToolFunction: |
| 无 |
| def analyze(line): |
| def readFile(fileName): |
| def replace\_char(string, char, \*args): |

**表3-2** ToolFunction**类成员列表**

|  |
| --- |
| class Hardware(metaclass=ABCMeta): |
| 寄存器变量，如self.R0，self.R1 |
| def PLA(self):（抽象方法） |
| def run(self):（抽象方法） |

**表3-3** CPU**类成员列表**

|  |
| --- |
| class CPU(Hardware): |
| self.instruction\_set  self.control\_memory |
| def compile(self, assembly\_code): |
| def PLA(self, machine\_code):（复写父类方法） |
| def run(self, machine\_code):（复写父类方法） |
| def set\_register\_value(self, source\_operand, value): |
| def judgment\_register(self, register): |

**表3-4** MyWindow**类成员列表**

|  |
| --- |
| class MyWindow(QMainWindow): |
| def time\_call\_function(self): |
| def open\_file\_success\_click\_button(self): |
| def single\_cycle\_execution\_click\_button(self): |
| def continuous\_cycle\_execution\_click\_button(self): |
| def set\_all\_register(self): |
| def set\_control\_memory(self): |
| def save\_file\_success\_click\_button(self): |

（2）系统包含的类库

import sys 系统相关模块

Import pyqt5 UI界面相关模块

import re 正则匹配模块

import abc 抽象方法模块

## 2.2系统主要函数的组成、功能、参数说明、相互调用关系

**主函数 if \_\_name\_\_ == "\_\_main\_\_":**

初始化UI界面，创建UI对象，创建对象时即完成pyQT的槽函数以及信号函数的绑定，然后使用 show.show()显示UI界面，并且绑定程序结束按钮为界面右上角的关闭选项栏。

**编码函数 def compile(self, assembly\_code):**

本函数的作用是将传入的汇编文件依次编码成机器码，其中包括寻址方式。

**指令跳转函数 def PLA(self, machine\_code):**

这个函数可以根据机器码的前四位，即操作码来找到对应微指令的入口。

**运行函数 def run(self, machine\_code):**

本函数即为核心部分，用来模拟cpu的运行过程，并且计算各个寄存器的值。

**寄存器设置函数def set\_register\_value(self, source\_operand, value):**

本函数第一个参数为寄存器的地址，第二个参数为要设置的值，作用是将对应的寄存器的值设置为第二个参数的值。

**寄存器值返回函数def judgment\_register(self, register):**

这个函数的功能根据字符串返回对应的寄存器。

**汇编指令分解函数def analyze(line):**

主要功能是分离指令与操作数，最终以列表形式返回，用于后边转化为机器码。

**文件读入函数def readFile(fileName):**

这是文件读入函数，读入文件的内容，将每一行的数据保存在列表中，最后将整个列表返回，是程序实现的第一步。

**字符串替换函数def replace\_char(string, char, \*args):**

由于python 的字符串为不可变类型，所以本函数的作用是将特定位置的字符串进行替换，并且返回替换后的字符串。

# 3设计测试流程

1. 打开本程序。如图3.1所示。

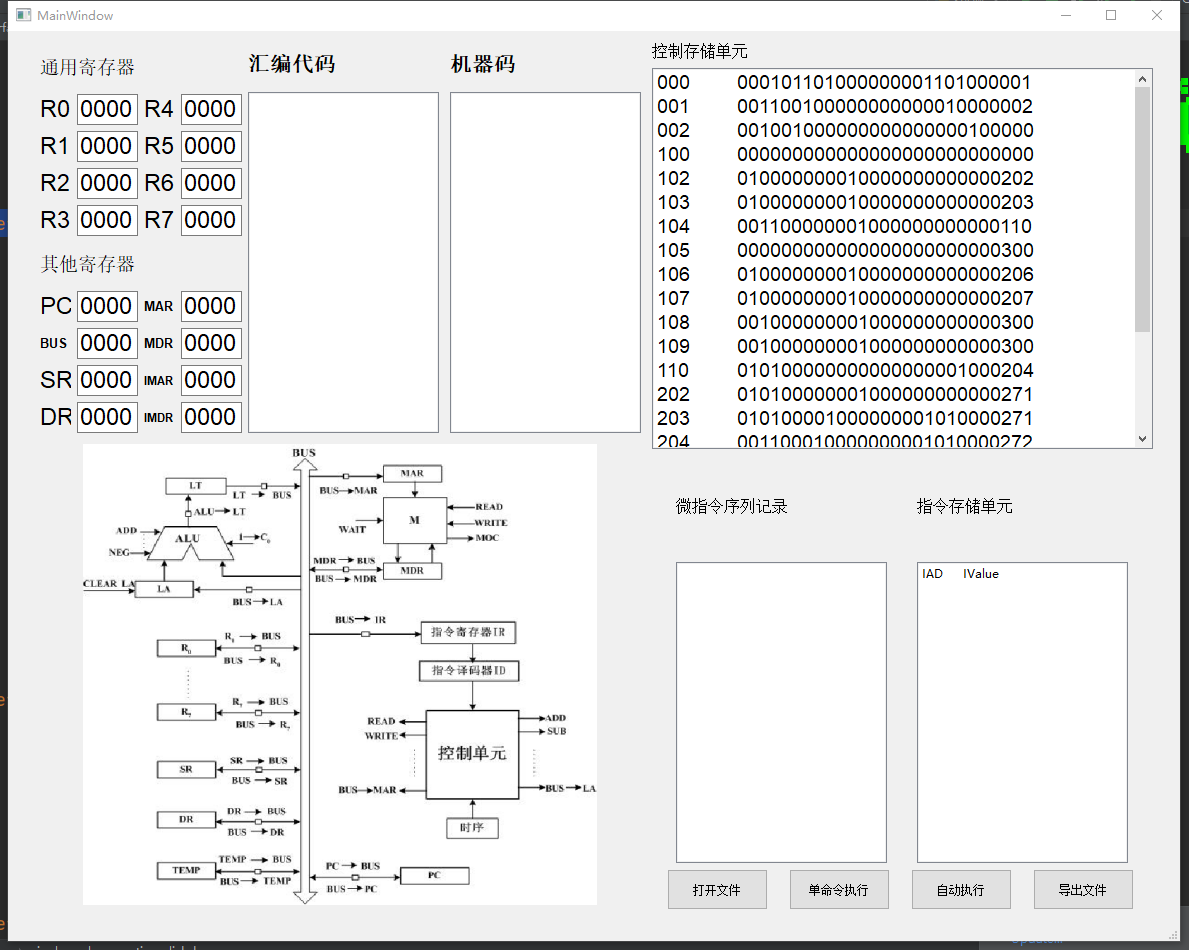


图3.1 系统主页

1. 点击打开文件，并选择文件。如图5.2所示。

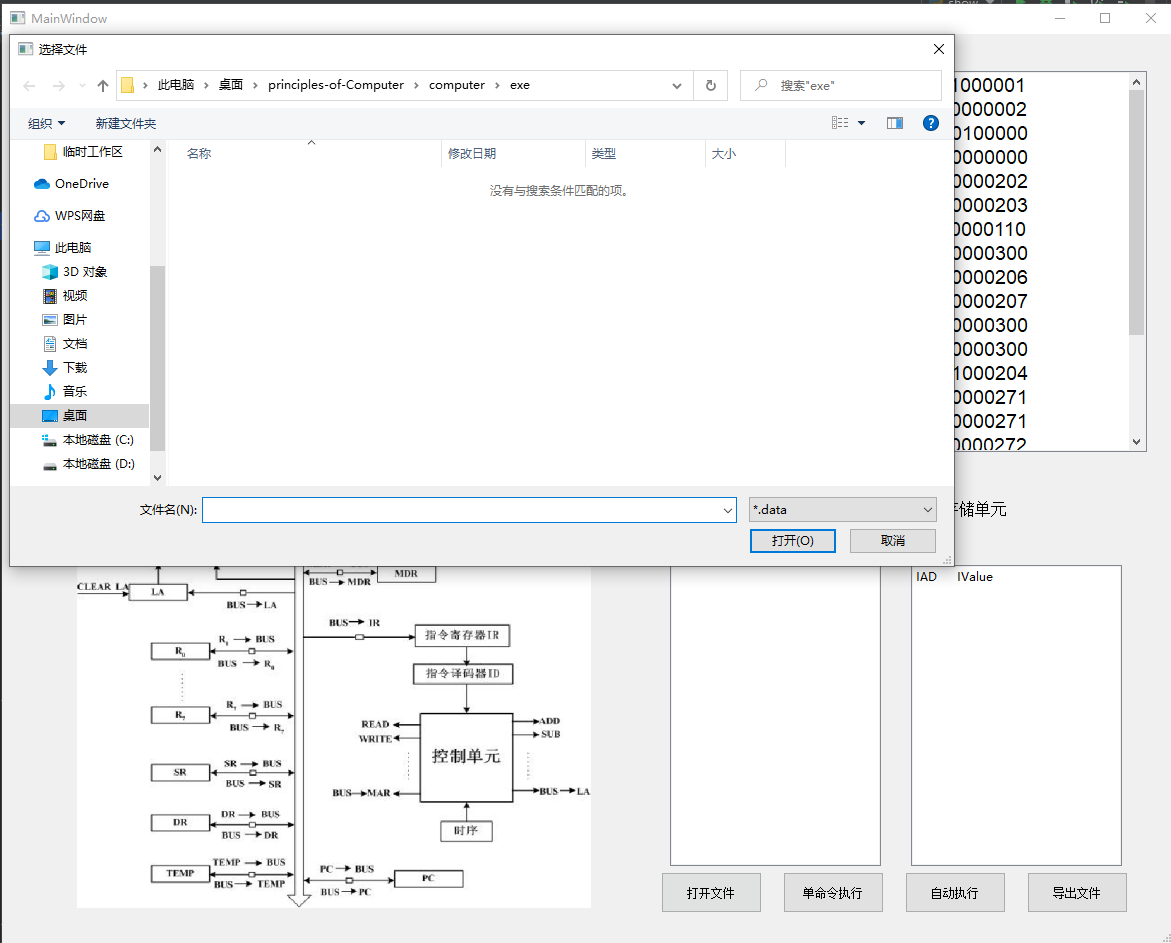


图3.2 文件选择

3.文件读入成功。如图3.3所示。

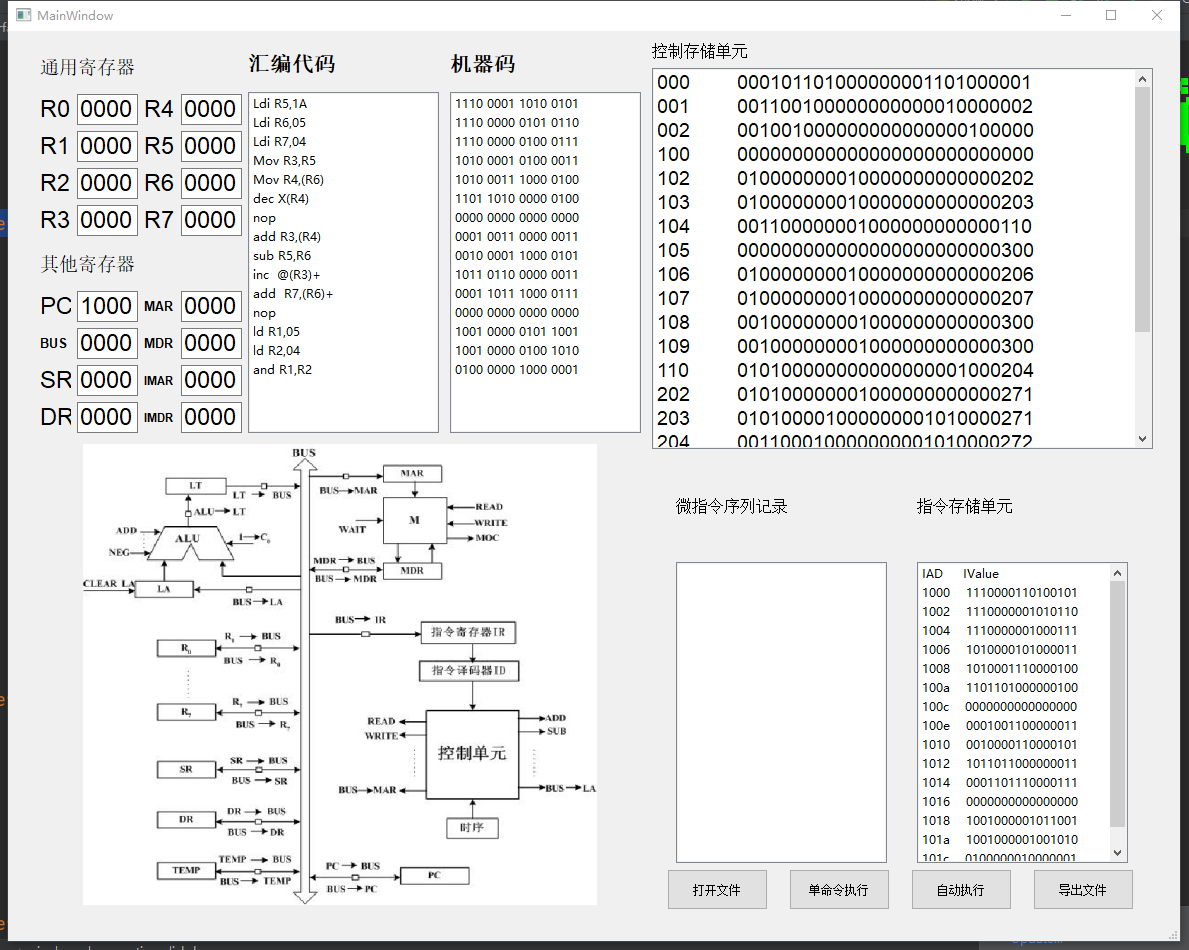


图3.3 读入成功

3.点击单步执行，会自动执行下一条命令。如图3.4所示。

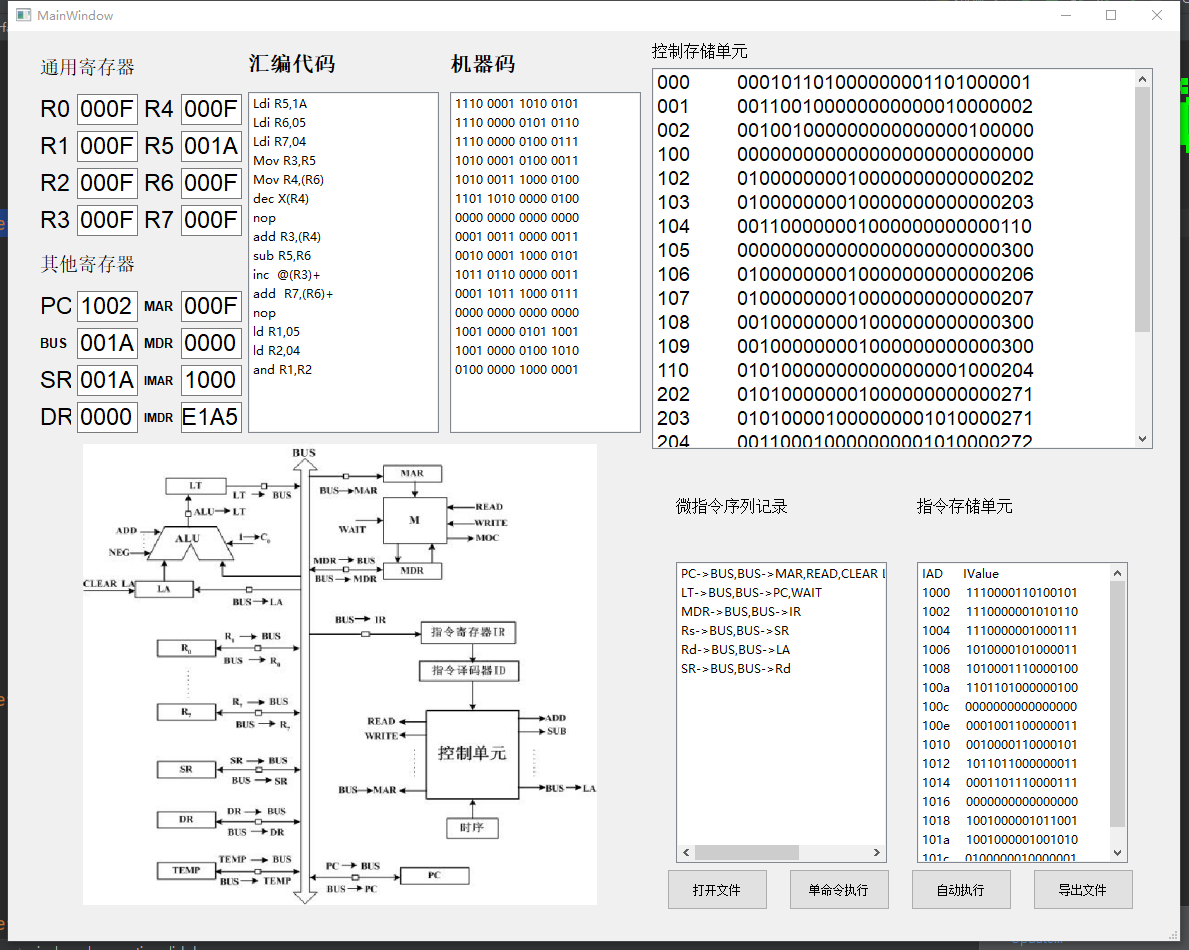


图3.4 单步执行

5.点击自动执行，可以自动执行命令。运行结果如图3.5所示。

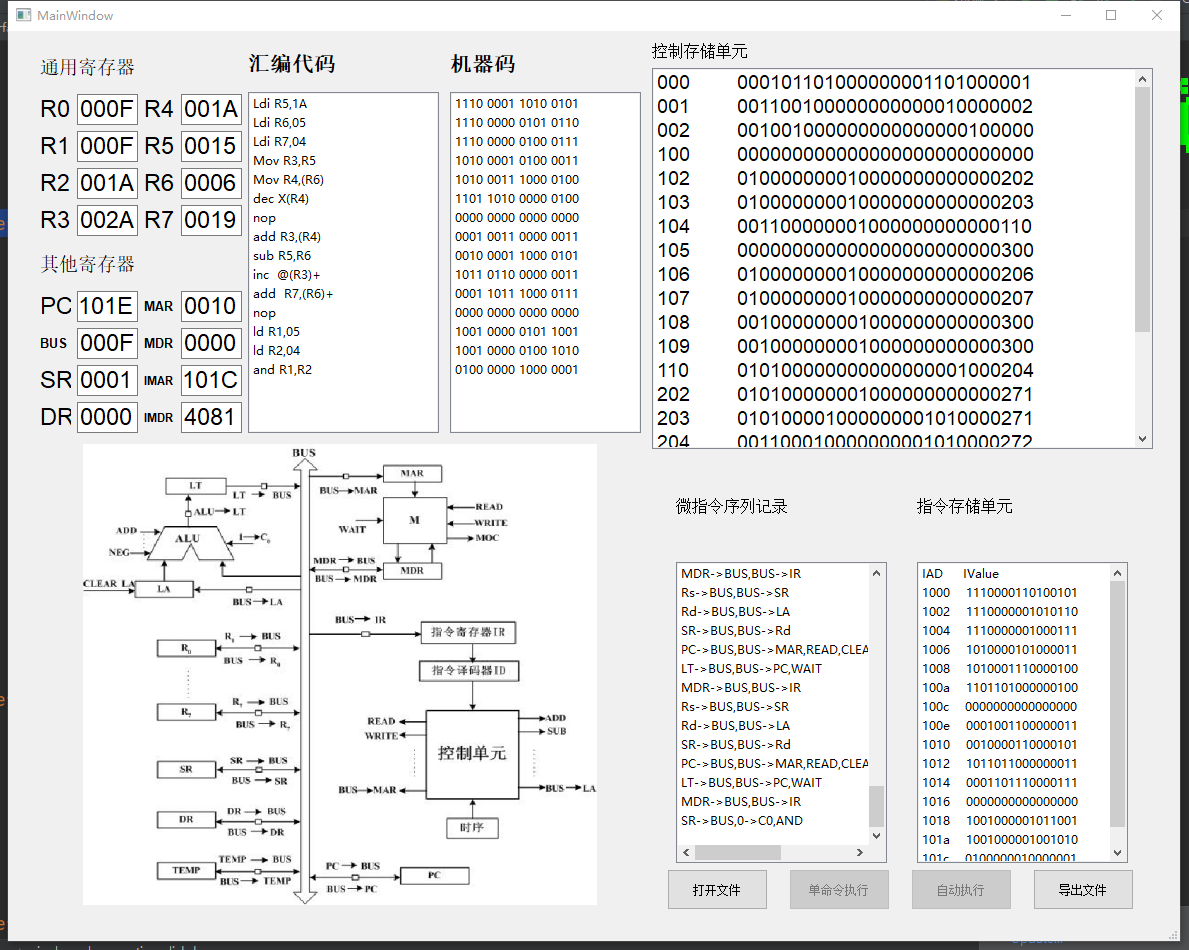


图3.5 自动执行

6.点击导出文件，可以将微命令导出。运行结果如图3.6所示。

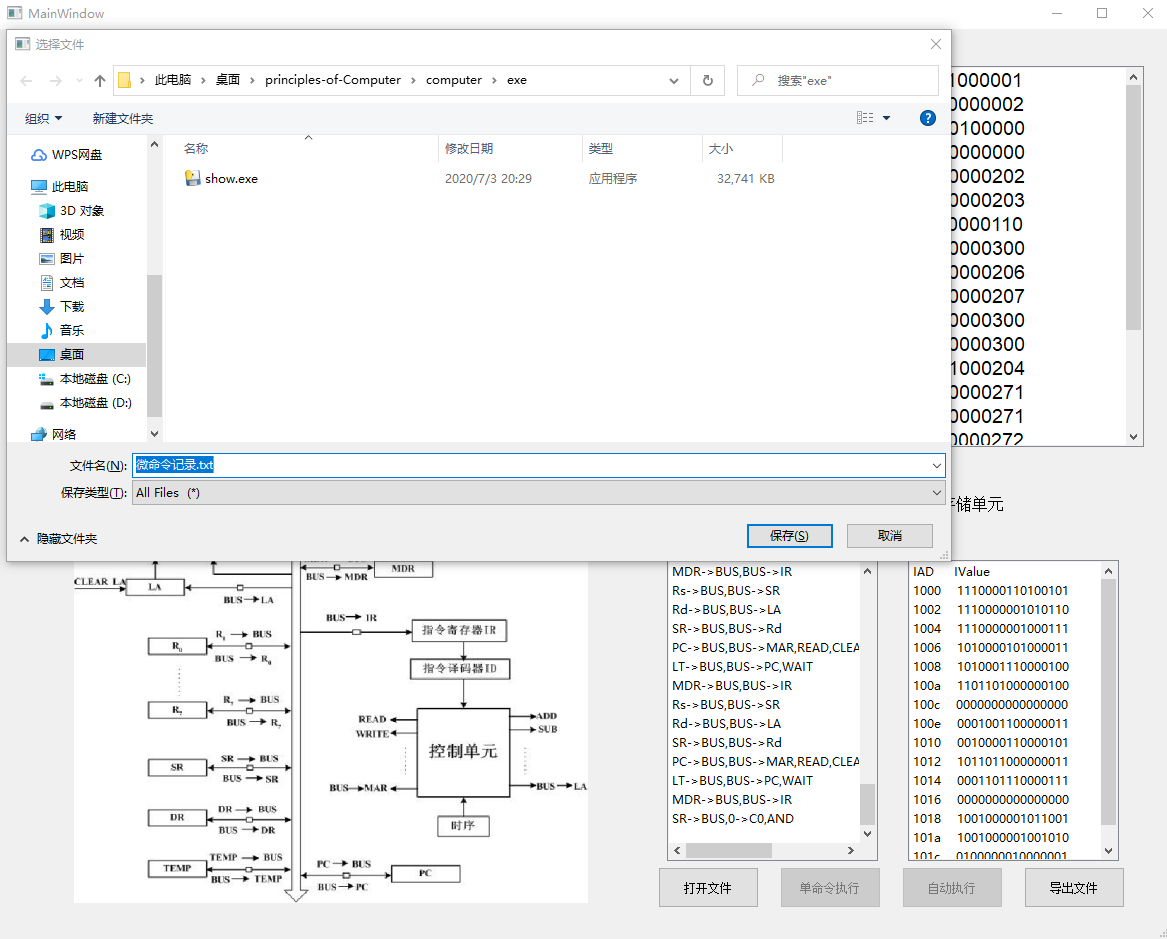


图3.6 导出文件

# 4 作品设计、实现难点分析

## 1.难点分析

一个项目的开发过程并不总是一帆风顺的，其中难免会遇到困难，使编程无法继续下去。例如汇编文件的机器码转化。

## 2.解决方案

针对程序设计过程中遇到的困难，我们采取了积极有效的解决方法：上网搜寻相关知识，并且认真细读课本内容，最后明白了机器码的指令格式是：操作码（4位） 目的操作数寻址方式（3位） 目的操作数（3位） 源操作数寻址方式（3位） 源操作数（3位），并且按照这种格式进行编码，其中涉及大量的字符串替换，而且在分析中发现因为中英文符号的不同而导致的各种错误，最终经过反复的调试及测试，解决了上述的难点，使得程序可以正常运转。

## 3.测试结果

程序在pycharm以及多台windows系统电脑中运行测试，能实现预期的功能，项目完工！

# 5 心得体会

# 通过本次计算机原理课设的设计，让我深深感受到了计算机的魅力，对于底层的最接近硬件的设计，才是整个计算机设计的最核心部位，我也通过这期间的资料查询以及思考，慢慢掌握了计算机的原理，明白的计算设计的难点以及重点，同时也学习到了非常多python的知识，在之前的小项目或者爬虫项目中，并未使用python中类的相关知识，通过本次课程设计我也更加巩固了我的编程水平。

# 我相信这些基本的知识，无论在后续的课设中，还是今后对于计算机的学习，都起到了无可挑剔的作用。我也会继续学习相关的理论知识，为将来的学习、工作铺平道路。

## 因为原码过多且分散于不同文件，所以源码以文件形式上传