虚拟 CPU 仿真实验报告

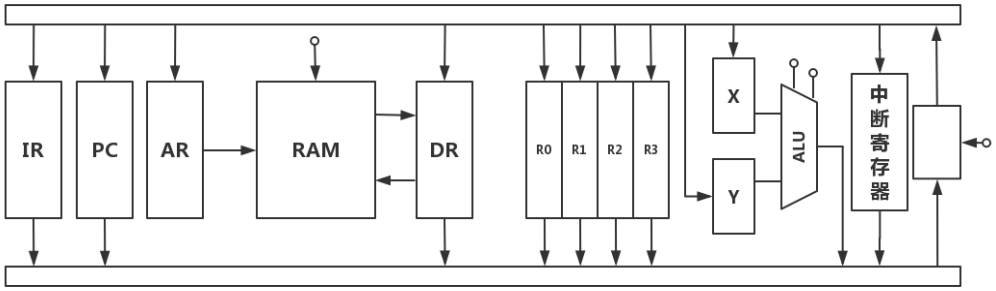
姓名：石宇辉　班级：2018211301　学号：2018210715　日期：2019.12.13

一、指令系统设计

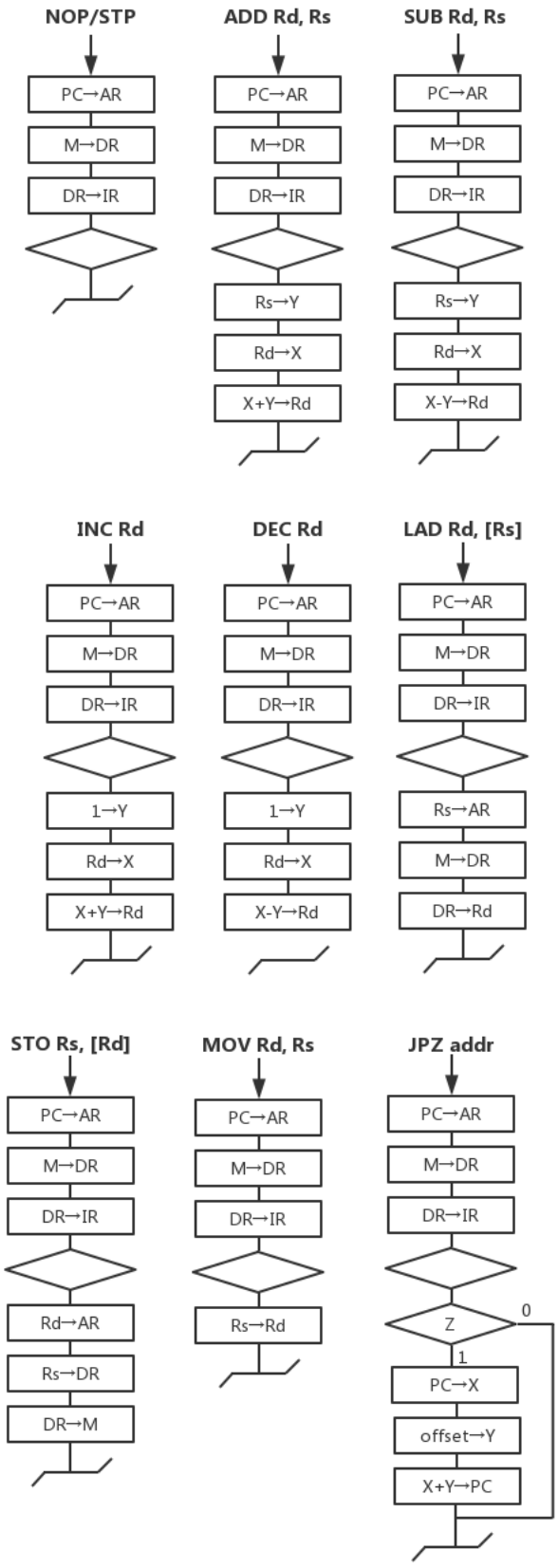
名称	助记符	功能	指令格式		
			IR7-IR4	IR3-IR2	IR1-IR0
空操作	NOP	无	0000	xx	
加法	ADD Rd, Rs	Rd←Rd+Rs	0001	Rd	Rs
减法	SUB Rd, Rs	Rd←Rd-Rs	0010	Rd	Rs
加 1	INC Rd	Rd←Rd+1	0011	Rd	xx
减 1	DEC Rd	Rd←Rd-1	1011	Rd	xx
取数	LAD Rd, [Rs]	Rd←[Rs]	0100	Rd	Rs
存数	STO Rs, [Rd]	Rs→[Rd]	0101	Rd	Rs
传送	MOV Rd, Rs	Rd←Rs	1000	Rd	Rs
Z 条件转移	JPZ offset	若 Z=1(运算结果为 0)，则 PC←PC+offset	1001	offset	
无条件转移	JMP offset	PC←PC+offset	1010	offset	
清零	CLA	重置寄存器	1110	xx	
停止	STP	停止程序	1111	xx	
开中断	EIT	允许中断	0110	xx	
关中断	DIT	禁止中断	0111	xx	
中断返回	RET	返回断点	1100	xx	

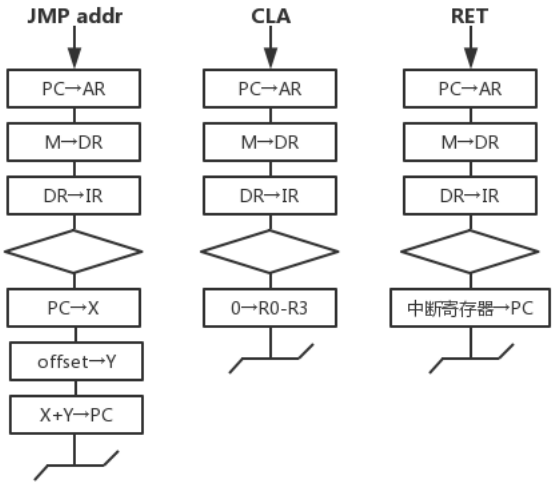
寻址方式说明：LAD 和 STO 指令为寄存器寻址，转移指令 JPZ 和 JMP 采用偏移寻址中的相对寻址方式，即 EA=offset+(PC)，操作数 offset 为偏移量。

二、数据通路设计



三、指令流程图


NOP/STP
PC→AR
M→DR
DR→IR
ADD Rd, Rs
PC→AR
M→DR
DR→IR
Rs→Y
Rd→X
X+Y→Rd
SUB Rd, Rs
PC→AR
M→DR
DR→IR
Rs→Y
Rd→X
X-Y→Rd
INC Rd
PC→AR
M→DR
DR→IR
1→Y
Rd→X
X+Y→Rd
DEC Rd
PC→AR
M→DR
DR→IR
1→Y
Rd→X
X-Y→Rd
LAD Rd, [Rs]
PC→AR
M→DR
DR→IR
Rs→AR
M→DR
DR→Rd
STO Rs, [Rd]
PC→AR
M→DR
DR→IR
Rd→AR
Rs→DR
DR→M
MOV Rd, Rs
PC→AR
M→DR
DR→IR
Rs→Rd
JPZ addr
PC→AR
M→DR
DR→IR
Z
0
1
PC→X
offset→Y
X+Y→PC



四、程序设计细节与使用方法

本程序是使用C#语言开发的基于.NET框架的Windows窗体应用，开发环境为Microsoft Visual Studio 2019。程序的运行依赖.NET Framework运行环境（Windows 7系统预装.NET Framework 3.5，Windows 10系统预装.NET Framework 4.6，框架向下兼容）。

在文件夹中已经生成了两个可执行文件，分别对应的.NET框架的3.5版本和4.5版本，请根据计算机中安装的.NET框架的版本选择可执行文件。若都无法执行，请安装最新的.NET框架。

主窗体中由内存模块、寄存器模块、编辑模块、运行模块共4部分组成。

写入汇编程序有两种方法：第一种是点击编辑模块中的“编辑内存单元”按钮将汇编程序或数据逐一写入RAM中；第二种是点击“打开”按钮打开预先写好的汇编程序（.asm或.txt文件），指令会自动写入RAM并汇编为机器指令。

运行汇编程序有两种方式：第一种是单步运行，即每次只执行一条指令；第二种是连续运行，即一次性执行全部指令至程序结束（STP指令）或发生中断。

用户可以通过编辑模块在任何时候修改任意内存单元(RAM)、程序计数器（PC）和通用寄存器（R0-R3）的值。

五、示例程序详解

在“示例程序”文件夹中有两个示例汇编程序，即**主程序.asm**和**中断服务程序.asm**。

主程序.asm中所写的程序是一个死循环，用于测试中断：

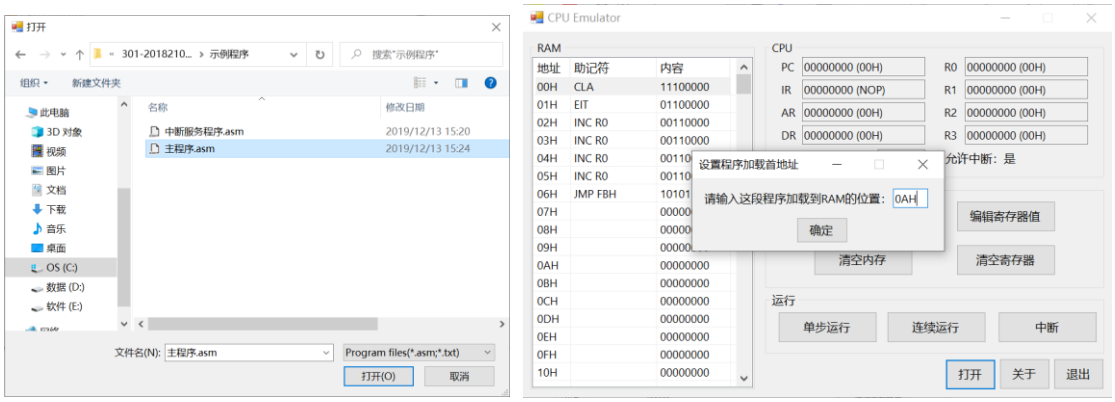
```
1.  CLA              # 清空寄存器
2.  EIT              # 开中断，即允许程序中断
3.  INC R0           # R0自加
4.  INC R0
5.  INC R0
6.  INC R0
7.  JMP FBH          # PC←PC-5（FBH为-5的补码），即跳转至第三行“INC R0”
```

中断服务程序.asm 的功能是将内存中的字符串倒序，其中 R0 和 R1 的初始值分别为字符串首末元素的地址：

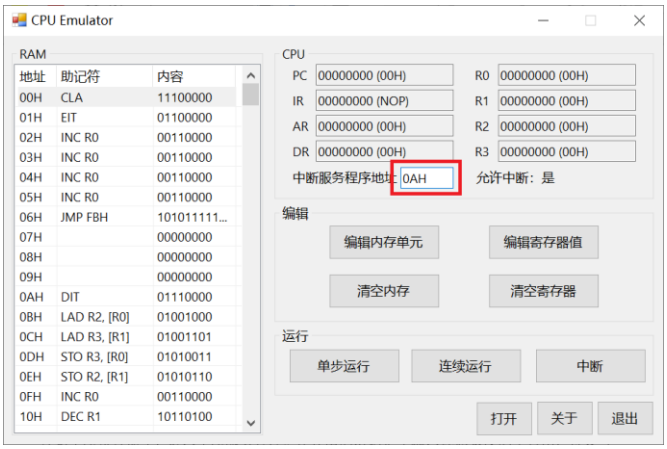
```
1.  DIT                 # 关中断，禁止程序产生中断
2.  LAD R2, [R0]
3.  LAD R3, [R1]
4.  STO R3, [R0]
5.  STO R2, [R1]        # 这四行实现 R0、R1 指向的字符互换
6.  INC R0              # R0 自加
7.  DEC R1              # R1 自减
8.  MOV R3, R0
9.  SUB R3, R1
10. JPZ 03H             # 若 Z 有效，则说明 R0=R1，即两指针相遇，翻转完成，跳转至 14 行
11. DEC R3
12. JPZ 01H             # 若 Z 有效，则说明 R0=R1+1，两指针穿越，翻转完成，跳转至 14 行
13. JMP F4H             # 执行到该行说明翻转未结束，跳转回第 2 行继续循环（F4H 即-12）
14. EIT                 # 开中断，即允许程序中断
15. RET                 # 返回主程序断点
```

五、示例程序演示及操作方法

1. 单击“打开”按钮，分别将两段程序加载到 RAM 的不同位置；

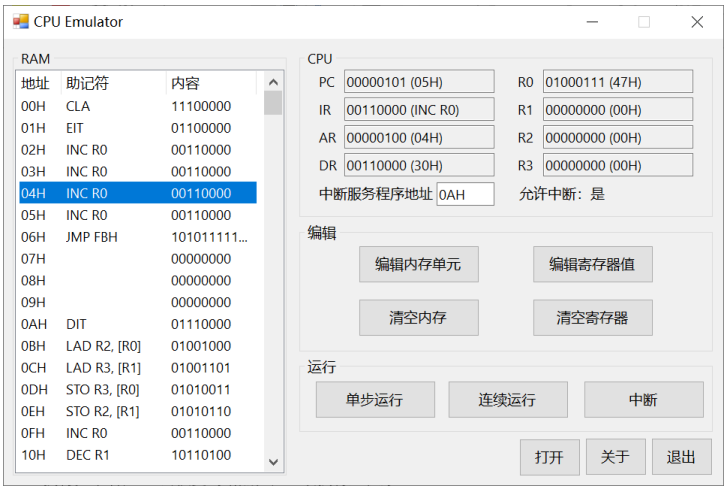


2. 将右侧CPU 框中的“中断服务程序地址”文本框中的值修改为中断服务程序所加载的位置；

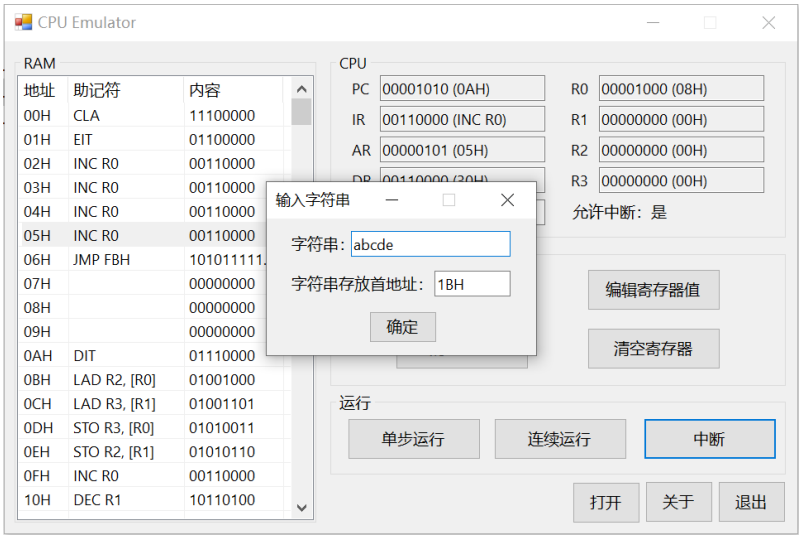


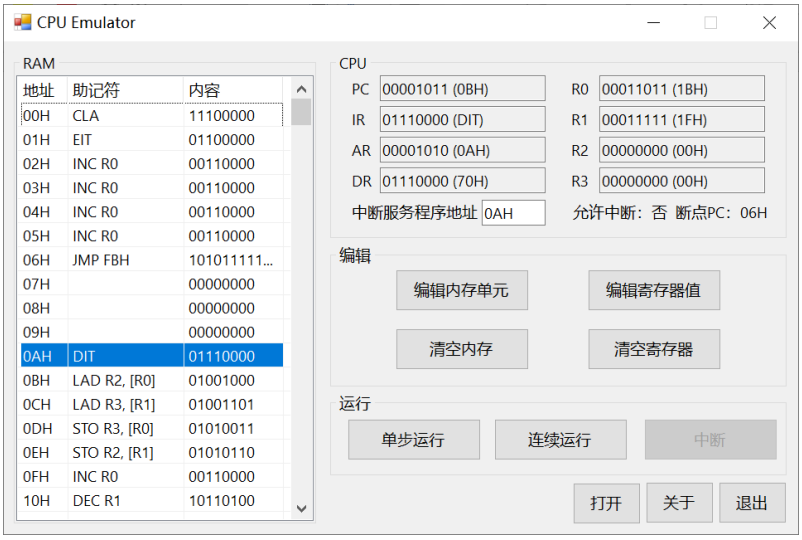
3. 点击“编辑寄存器值”按钮，将 PC 修改到主程序所加载的位置（若主程序加载到 00H 则无需修改 PC）；

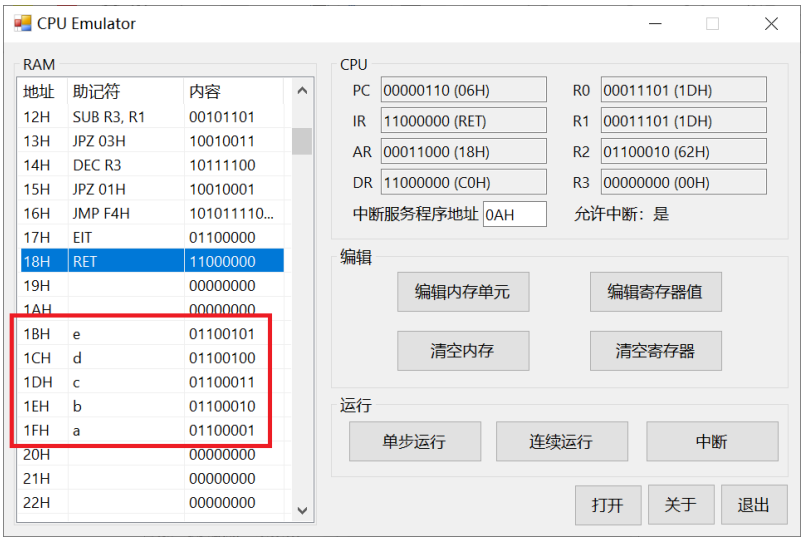
4. 点击“连续运行”。此时可以看到程序处于循环状态，R0 不断自加；



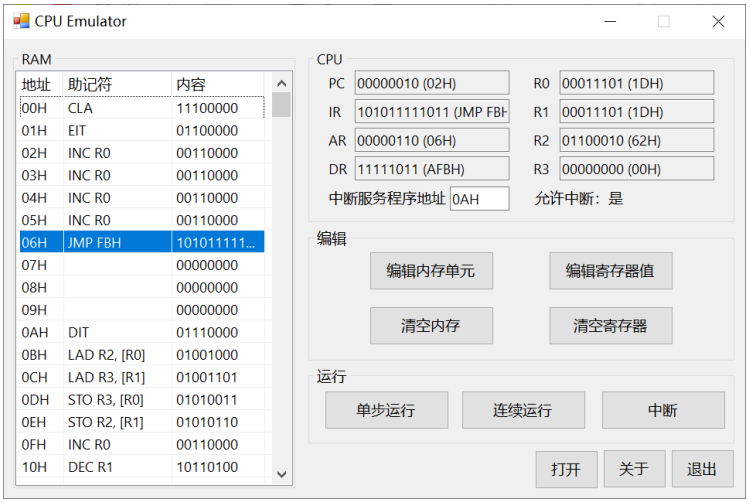
5. 当按下“中断”按钮时，产生中断信号，当前指令执行结束后会检测到中断，此时 PC 值被暂存到中断寄存器中，进入中断服务程序。在弹出的对话框中输入任意长度的字符串，将其写入 RAM 的任意位置，然后单步执行，通过左侧“RAM”框可以看到程序将字符串倒序的过程；







6. 中断服务程序执行完毕后，PC 会恢复到断点值，继续执行主程序。



六、收获与体会

通过这次的作业，我对计算机的指令系统、数据通路、总线系统、中断原理和 CPU 有了更为深刻的理解，为了设计这样一个虚拟 CPU 仿真器，我必须将计算机组成原理的诸多部分融会贯通，明白数据的流向与各寄存器值的变化，形成计算机执行程序的一个完整流程。同时，在程序的编写过程极大地锻炼了我的编码能力与逻辑思维能力。因此，本次作业全方位地提升了我的能力，使我受益匪浅。