一、设计题目

简单模型机的设计与实现

二、设计要求

将指令存储器和数据存储器分开,指令存储器的地址总线和数据总线宽度均为16位,数据存储器的地址总线宽度为16位,数据总线宽度为8位。

CPU使用流水线技术，流水级数为5级，分别是：取指、译码、执行、访存、写回。

CPU的主要寄存器及编址方式：PC寄存器，复位时的值为0，16位宽；16个通用寄存器（r0~r15），对应地址为0~15，复位时的值为0，8位宽；一个程序状态寄存器（PSW，对应地址为16），8位宽；通用寄存器、程序状态寄存器和数据存储器统一编址，通用寄存器既可以用寄存器号访问，也可以用地址空间的地址访问。

输入要求：模拟器从文件test.data读入汇编执行,先将汇编编译成二进制。

输出要求：模拟器用txt文件记录每一个周期CPU主要寄存器的值，总线数值，程序执行完毕后，用txt文件记录数据存储器的内容。记录数据时要注意对齐。同时界面显示。

基本功能：完成任务书基本指令串行执行过程描述。将指令分解成微指令。可视化展示指令执行过程，演示中测试用例要包含指令集中所有指令。

指令集：

```
Add Rd , Rr
Sub Rd , Rr
RJMP K
BRMI K
Mov  Rd , Rr
Ldi Rd , K
Ld  Rd , X
St  X, Rr
Nop
```

三、设计过程

3.1 设计架构

本项目设计共分为输入指令，指令转码，模拟执行，导出数据四个模块，如图1所示。

其中，输入指令模块负责从“test.data”文件中读取预先设置的指令内容，将其写入至汇编指令框中；

指令转码模块负责对读取到的指令内容进行转码，将其以机器码的形式写入机器码框中；

模拟执行模块负责将对每条指令的机器码进行拆解，分析，将每条指令分解成微指令序列，写入至微指令序列记录框中，同时模型机中代表程序计数器，总线，寄存器等的相应文本框实时显示其中存储内容，在有数据传递时高亮显示；

导出数据模块负责将执行完毕后模型机中各存储器的情况写入至“数据.txt”文件中，方便用户查看数据情况。

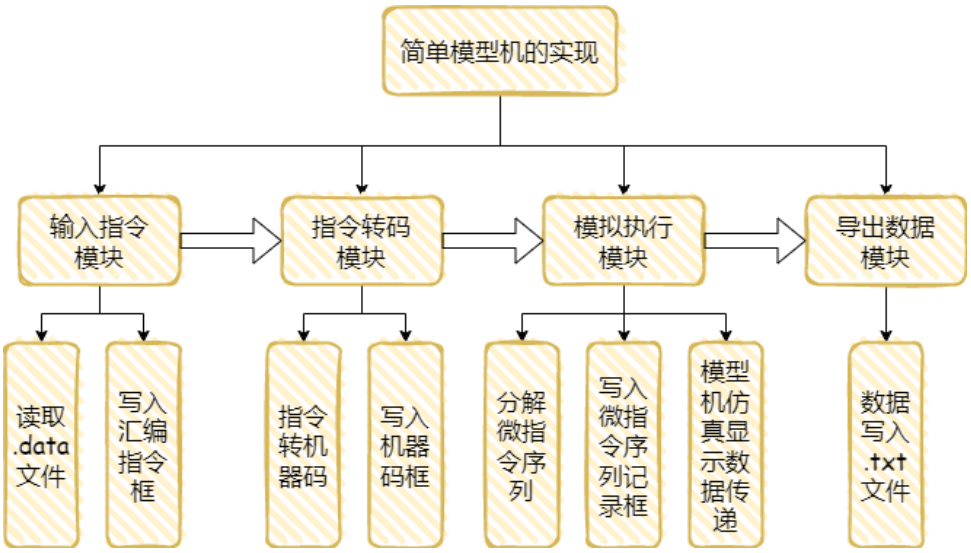

图 1 设计架构图

3.2 模块设计

1）输入指令模块 & 指令转码模块

点击.exe 可执行文件运行程序时，如图 2，首先展示程序主界面，在未导入文件时，其他按钮为禁用状态，仅导入文件按钮可用，此时所有数据寄存器等的值为复位状态，汇编指令，机器码，微指令序列记录，主存单元，指令存储单元框均为空，指令周期阶段框为未选中状态。

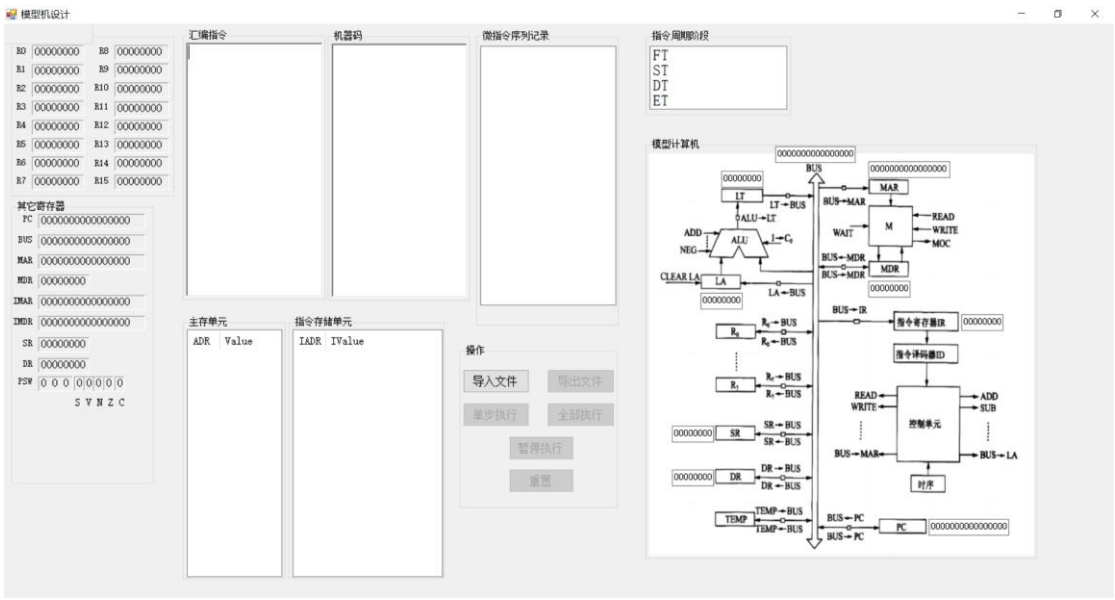

图 2 程序初始界面

当点击导入文件按钮后，如图 3，所有按钮变为可用状态，汇编指令框内写入文件中预先设置好的指令内容，机器码框中写入文件中指令转机器码后内容，主存单元框中写入模拟主存单元内容，指令存储单元框中写入指令在模拟存储中的相应信息。

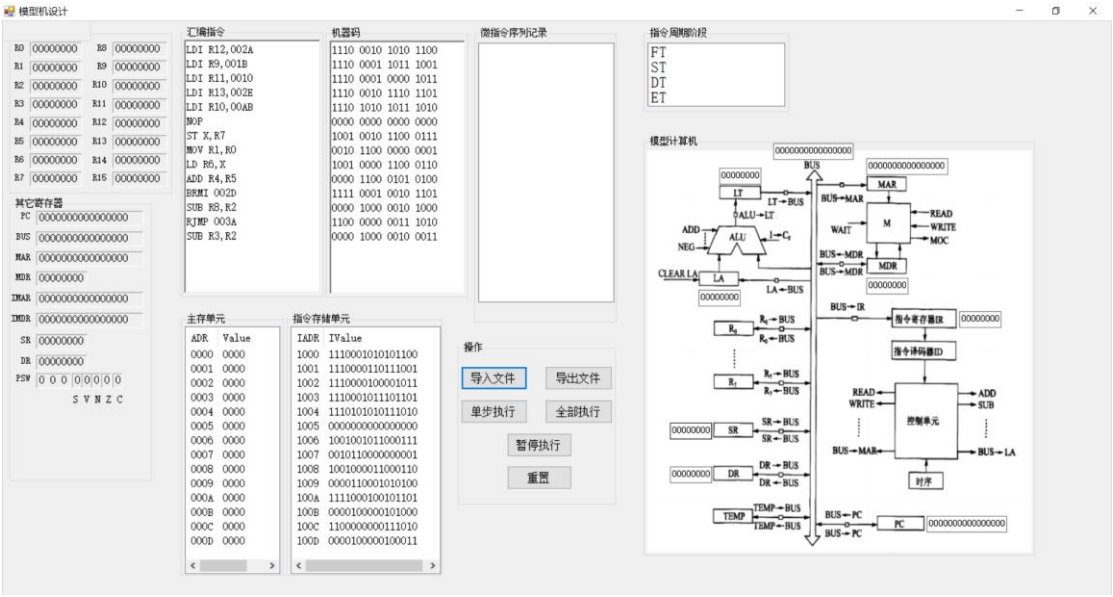

图 3 输入指令后界面

2）模拟执行模块

当程序导入文件后，此时可以对输入指令进行模拟执行，模拟执行分为两种形式，如图 4，点击单步执行，程序将往微指令序列记录框中写入一条由输入指令分解得到的微指令，如图 5，点击全部执行，程序将进行自动执行，无需用户手动操作，程序将不断自动分解输入指令，得到微指令序列，然后不断写入微指令序列记录框中，同时在执行过程中，指令周期阶段将实时显示当前指令的周期阶段，模型机将实时高亮显示数据传递情况，R0-R15 通用寄存器及其他寄存器将实时显示数据存储情况。

当用户点击全部执行按钮后，全部执行按钮禁用，此时用户可以通过暂停执行按钮，选择是否暂停执行或继续执行。

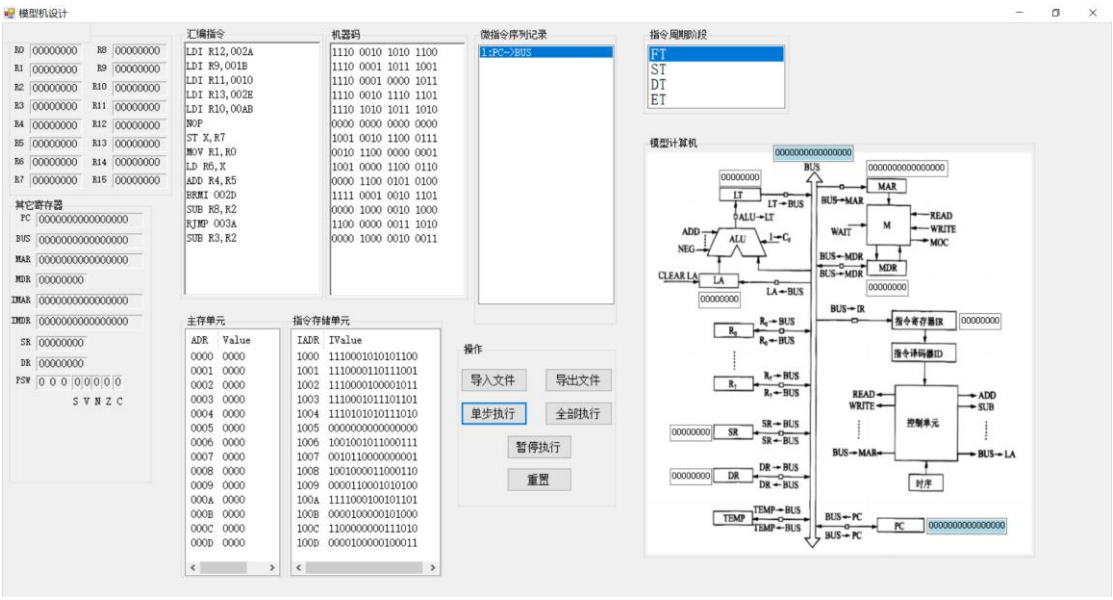

图 4 单步执行情况

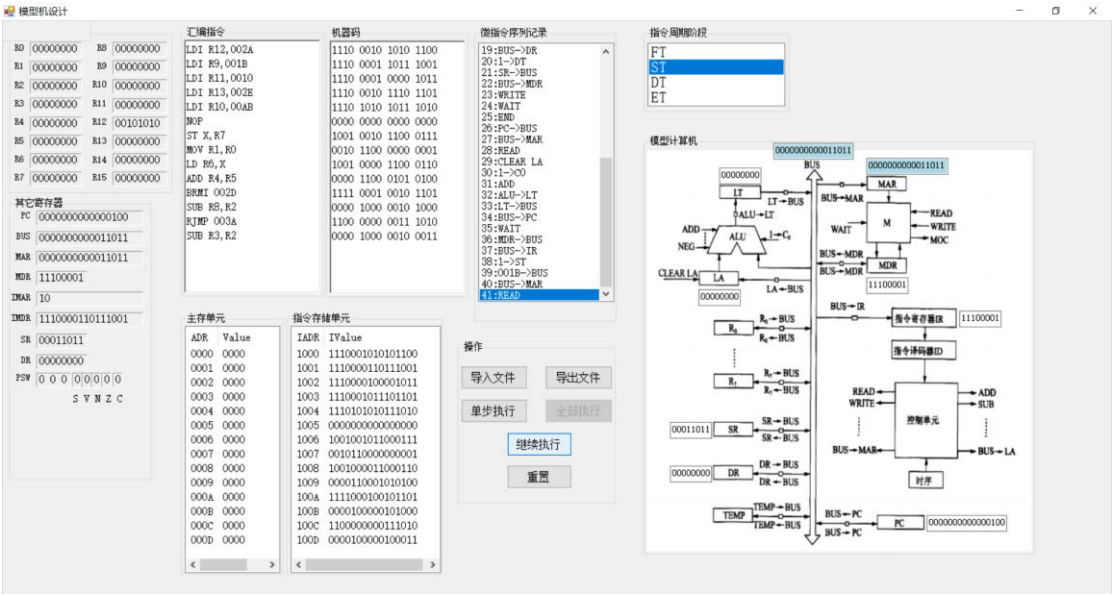

图 5 全部执行情况

3）导出文件模块

在程序执行完毕后，如图 6，点击导出文件按钮，程序将记录各通用寄存器及程序计数器的值，并将其情况导出至“数据.txt”文件中，当导出文件成功后，程序会弹出提示信息，提示文件导出成功。在程序 Debug 目录下找到“数据.txt”文件，发现程序运行各数据内容写入正确。

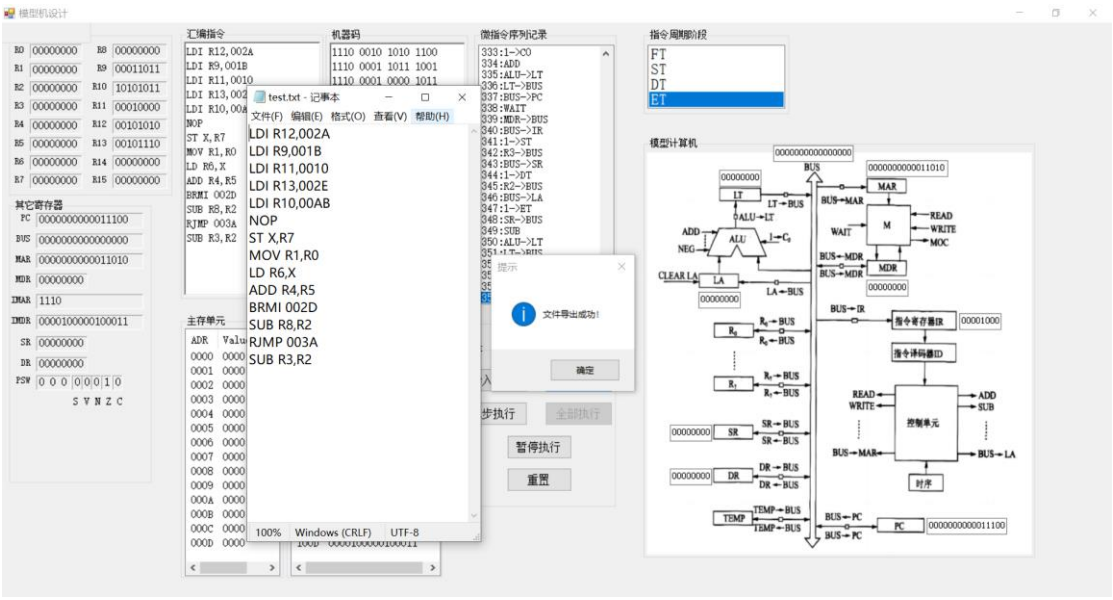

图 6 导出文件情况

3.3 测试文件执行过程

本次测试用到的测试文件为：test.data 其转 TXT 文本文件后内容如下图 7 所示：

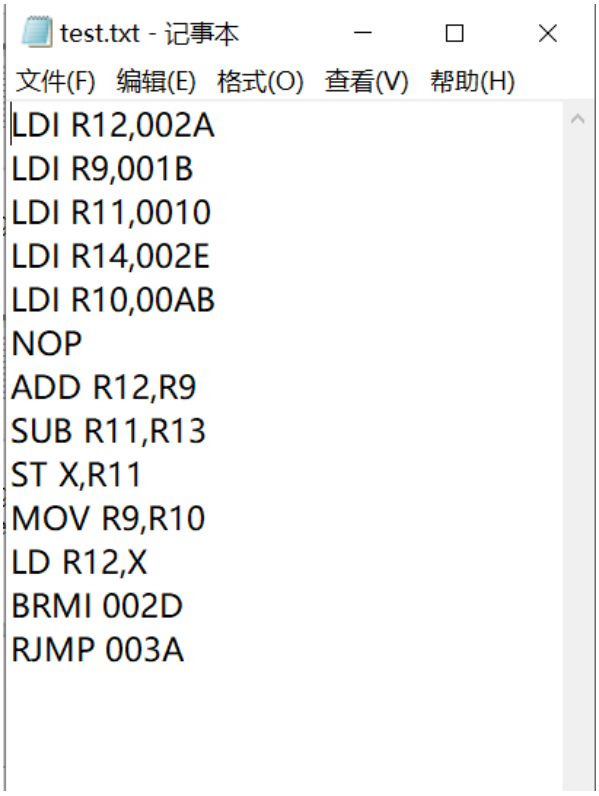

图 7 测试文件内容

1）在点击导入文件按钮后，如图 8，程序正确载入了文件，将文件内容正确显示在汇编指令框中，并将指令转机器码正确显示在机器码框中，同时程序进行了初始化，各项数据

存储器值归 0。

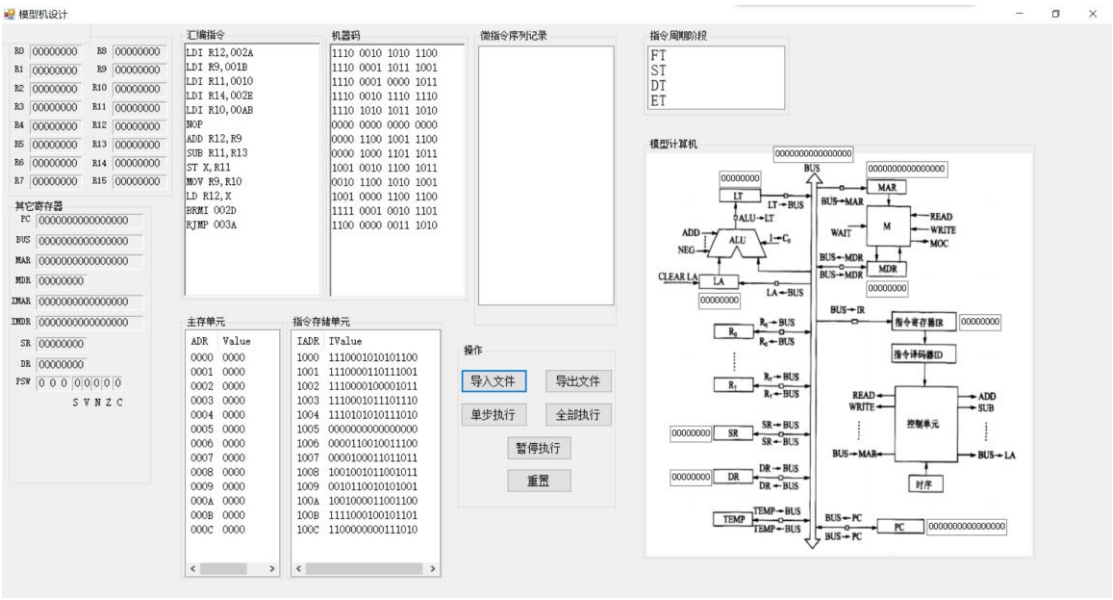

图 8 导入测试文件情况

2）测试文件中第一条指令为载入立即数指令，指令格式表示为：LDI，Rd，K，机器码（二进制表示）为 110 KKKK dddd KKKK，其中，dddd 为目的操作数的寄存器号，目的寄存器只能是 r8~r15，KKKK KKKK 是立即数。当第一条载入立即数指令执行完毕后，如图 9，可以看到程序中，第一条载入立即数指令正确被译码为 1110 0010 1010 1100，在执行完毕后，指定立即数被载入预先设置的 R12 寄存器中，同时，PC，BUS，MAR，MDR 等值显示正常，模型机动态运行正常。

该立即数指令被分解成微指令序列写入微指令序列记录框中，在该指令执行过程中将经历 FT，ST，DT，ET 四个周期阶段，FT 取指周期阶段包括：PC 送 BUS，BUS 送 MAR，READ，CLEAR LA，1 送 CO，ADD，ALU 送 LT，LT 送 BUS，WAIT，MDR 送 BUS，BUS 送 IR，1 送 ST，ST 取源操作数周期阶段包括：立即数送 BUS，BUS 送 MAR，READ，WAIT，MDR 送 BUS，BUS 送 DR，1 送 DT，此处载入立即数指令无目的操作数，故不进行 DT 取目的操作数周期阶段，ET 执行周期阶段包括 SR 送 BUS，BUS 送 MDR，WRITE，WAIT，END。

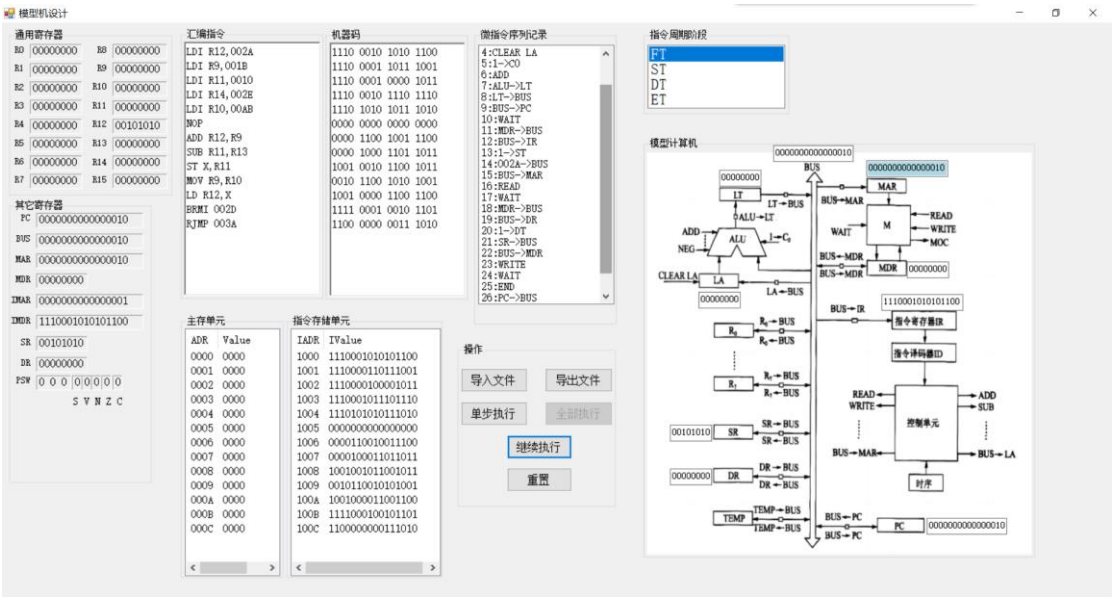

图 9 载入立即数指令执行情况

3）当多条载入立即数指令运行完毕，载入多个立即数后，接下来进行其他数据操作指令，首先是 ADD 加法指令，其指令格式表示为 ADD Rd，Rs，其功能为 Rd 寄存器与 Rs 寄存器中值相加后送 Rd 寄存器，其机器码（二进制表示）为 000 1100 dddd rrrr，其中，rrrr 为源操作数的寄存器号，dddd 为目的操作数的寄存器号。如图 10，可以看到 R12 与 R9 寄存器中数据正确相加，并存入目的寄存器。然后是 SUB 减法指令，其指令格式表示为 SUB Rd，Rs，其功能为 Rd 寄存器与 Rs 寄存器中值相减后送 Rd 寄存器，其机器码（二进制表示）为 0000 1000 dddd rrrr，其中，rrrr 为源操作数的寄存器号，dddd 为目的操作数的寄存器号。如图 11，可以看到 R11 与 R13 寄存器中数据正确相减，并存入目的寄存器中。接下来是 ST 存储指令，其指令格式表示为 ST，X，Rr，其功能为将 rrrr 寄存器的数值写入到 14 号寄存器的值做地址的内存单元。其机器码（二进制表示）为 1001 0010 rrrr 1100，其中，rrrr 源操作数的寄存器号， X 为 R14 如图 12，可以看到把 R11 寄存器的数值正确写入到 14 号寄存器的值做地址的内存单元。接下来是 MOV 数据传送指令，如图 13，可以看到将源寄存器 R9 的值写入目的寄存器 R10 中，其他指令类似，此处不过多赘述。

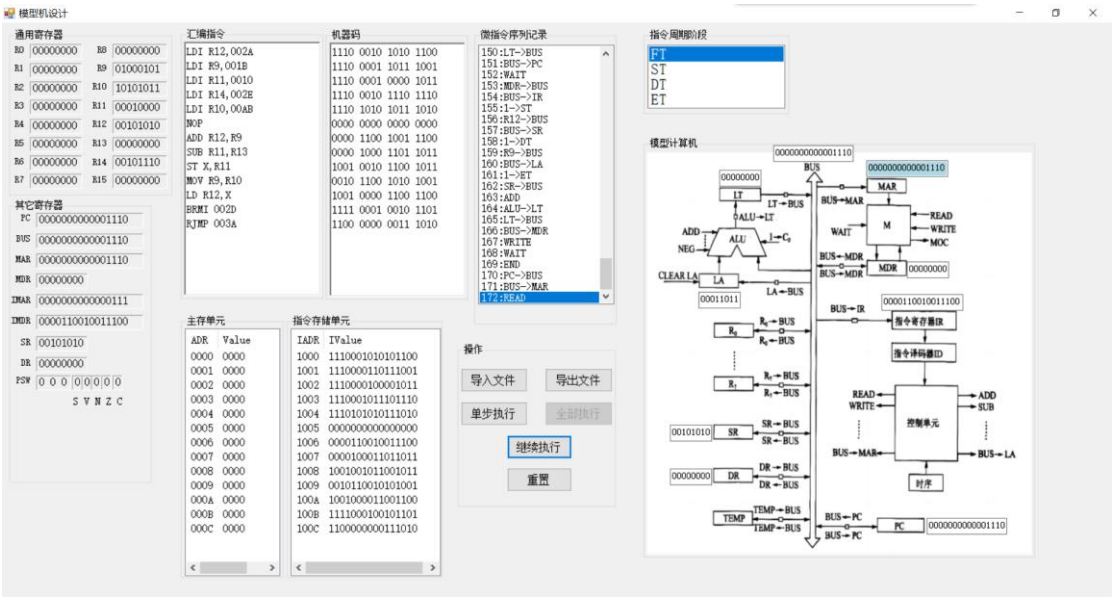

图 10 加法指令执行情况

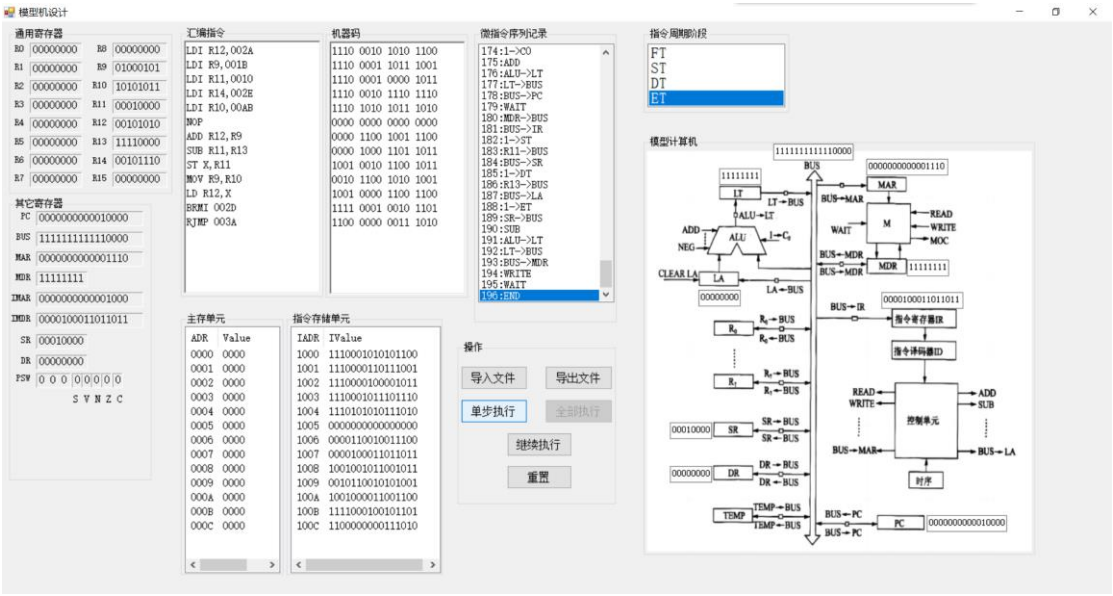

图 11 减法指令执行情况

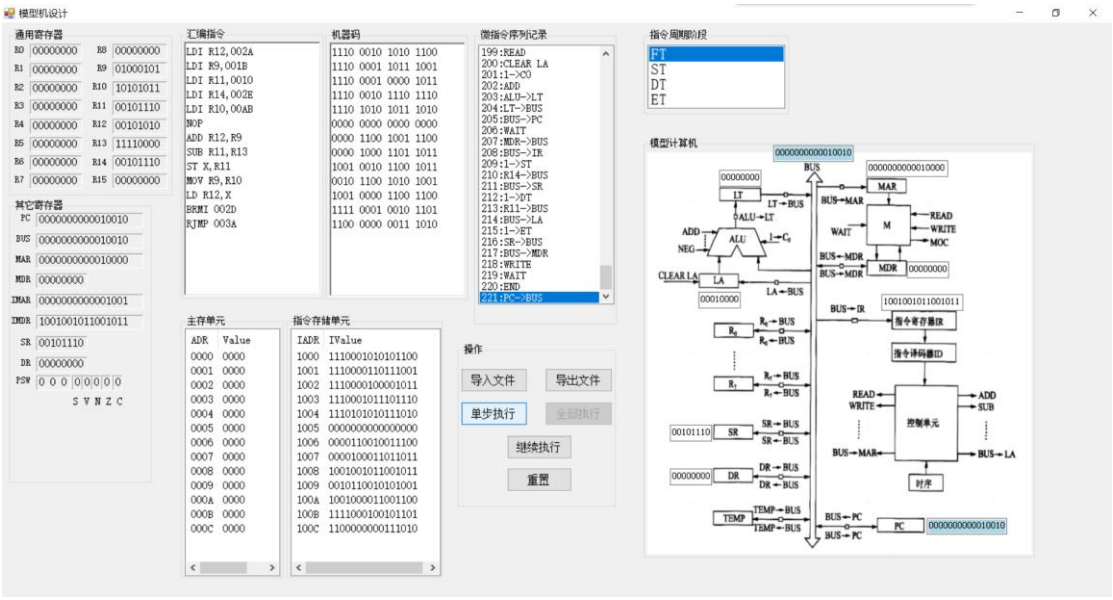

图 12 存储指令执行情况

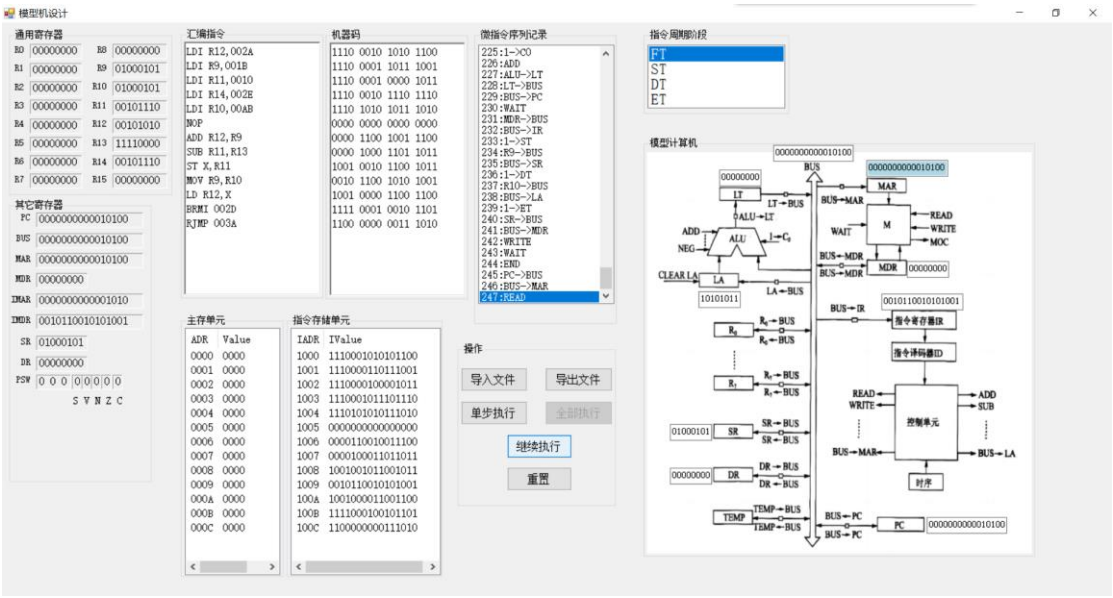

图 13 数据传送指令执行情况

四、设计中遇到的问题和解决方案

- 问题一：如何实现程序自律全部执行

  解决：在我实现完单步执行功能后，我想到设计一个循环，在每次执行完单步执行方法 singlestep() 方法后停顿几百毫秒再循环，实现程序的自律执行，开始时我计划使用 Thread.Sleep() 方法来实现延时。但在实际应用过程中我发现该方法会阻塞当前线程的执行。而我的程序又是基于 UI 界面的，阻塞当前线程的执行将导致模型机相关的控件无法在执行过程中实时刷新当前值。在查询资料后，我想到使用 async 和 await 关

键字来创建异步方法，用异步操作的方法避免阻塞 UI 线程，成功解决了问题。

- 问题二：switch 匹配报空字符串异常

  解决：在我实现分解指令添加微指令序列记录时，我多次使用到了 switch 匹配结构，但我有一次在使用中发现，switch 匹配 IR 文本框的值的前四位 textBoxIR.Text.Substring(0, 4)时发现，switch 匹配报错，随后我启用断点发现竟然匹配到的是空字符串，随后我在前一句进行打印发现其并不为空，显示正常。多次调试后发现原来是当 switch 匹配不到相应的 case，同时没有设置 default 语句时，即会显示匹配为空，可能这算是一个特性，在我加上相应的 default 语句后，成功解决了问题，也让我养成了以后规范使用 switch 的习惯。

- 问题三：指令存储单元和主存单元框的选用

  在设计指令存储单元和主存单元框时，我需要为其设置表头，分别对应表中设置的两列数据，但搜索资料后，查到的 DataGridView，对于没有提前准备数据源的表的设置使用并不方便。于是我对 Visual Studio2015 中的多个相似控件进行测试使用，最终发现 listView 成功适配要求，表头可以预先进行手动编辑进行显示，数据由代码句动态填充即可，成功解决了问题。

- 问题四：如何直观显示模型机的动态运行

  在此次课程设计任务基本实现功能后，在测试功能时我发现模型机的动态运行并不直观，虽然微指令序列不断在加，但给人的观感并不是非常直观，并不能直观的看到数据的传递和流向，在思考后，并且结合老师课上优秀作品的展示，我决定引入模型机的实现显示，将模型机图片使用 pictrueBox 进行加载，为了动态实时显示模型机 R0-R15、BUS、IMAR、IMDR、BUS 等数据存储器中值变化，又在模型机图片的对应位置旁加入了 相应的 textBox 用以显示对应值，在运行时给与相应控件高亮显示，表示数据的传递和流向，同时引入周期序列 ListBox 控件，根据微指令序列记录，实时显示当前指令的周期阶段。成功解决了问题。

五、设计感触

在完成此次计算机组成原理课程设计的过程当中，我深刻体会到了计算机的复杂性和魅力。通过自己动手去设计和实现一个简单的模型机系统，我对计算机的内部结构、指令执行流程和数据处理过程等有了更深入的理解。

此次课程设计让我从课上所学的抽象的计算机组成原理概念知识逐步转变为具体的实践，通过编写代码、设计模型机和模拟指令执行等工作，我深入探索了计算机的各个组成部分，如存储器、控制单元和运算单元等。我学会了如何处理指令的执行流程、数据的存取和操作，以及如何通过控制信号和时序控制来协调各个部件的工作。在此次设计过程中，我不仅学到了计算机硬件的知识，还锻炼了自己的问题解决能力和代码调试能力。在面对任务中出现的各种挑战和困难时，我学会了分析问题、查阅资料、与同学交流讨论，最终找到解决方案并将其实现。此次课程设计任务不仅让我计算机的原理有了更深入的理解，更提升了自己的编程能力和工程实践能力。我学会了将理论知识应用到实际项目中，并通过实践不断改

进和完善设计。这种实践经验将对我未来的学习和工作产生积极的影响。

同时，通过此次课程设计任务的完成，我也认识到了自己较牛人还存在着很多不足，例如在对课上知识的理解运用方面，在代码编写和调试方面等，同时这次任务也给了我一个宝贵的机会来反思和成长。我意识到自己在学习过程中应更加注重细节和深入理解原理。未来我要继续提升自己在这些方面的能力。我将坚持学习和实践，并持续改进自己的不足之处，以成为一名优秀的计算机工程师。

总的来说，此次计算机组成原理课程设计任务的完成是一次非常有意义和挑战性的经历。通过此次课程设计，我不仅加深了对计算机原理的理解，还培养了分析解决问题的能力。这将成为我未来在计算机领域发展的坚实基础，为我打开更广阔的职业发展道路。