实验报告

mips 模拟器实验报告

一、 设计思路&原理阐述

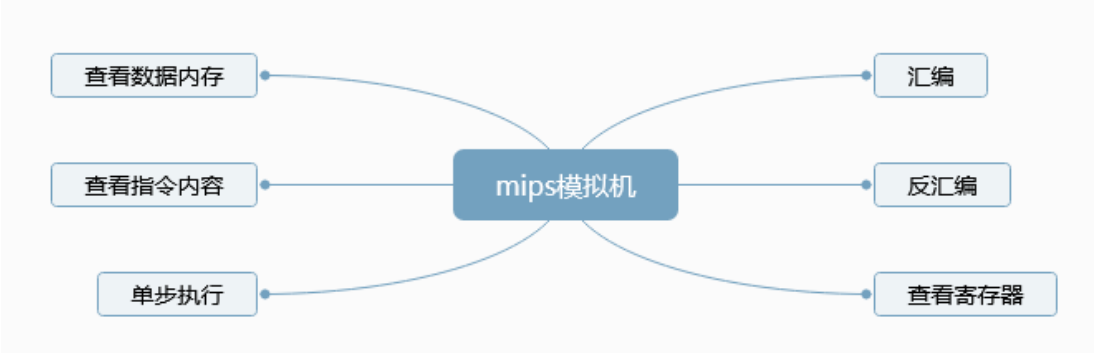
模拟器就是进行图形化的汇编和反汇编的转化，将 mips 汇编指令转化为机器码，并且在图形模式下进行执行、查看寄存器、查看内存等操作。

其实本实验最重要的就是做一个汇编模式的编译器，运用此编译器，进行一些简单的 mips 汇编代码的执行。

汇编的原理就是根据老师给的《zpc 汇编指令参考文档》，对不同的指令进行分类，进行不同的操作，以至于达到最终的目的。

二、 实现框图

根据实验原理，绘制实现框图如下：



三、 关键技术

模拟机的关键技术就在于对各种指令的分门别类的操作，把每条指令对应上一个机器码，再要学会从机器码里面找到对应的指令、寄存器、特征码等。

汇编代码转机器码需要的就是将一个指令正确分类，在一个字符串里对各个字符分类操作，找到正确有效的信息，进行机器码的转化，并将机器码存入我们的指令内存里面，方便进行读取。

反汇编就是我们执行指令的过程，将指令的机器码找出来，在里面找到对应的 rt，rs，dat 等变量，根据这些变量还原我们的代码，然后进行对应的模拟操

作。

图形界面，可视化操作。本次实验采用了 easyx 这一比较简单的图形库，因为其操作简单容易上手。之前准备尝试的 MFC 因准备不够充分而放弃了。

单步执行代码，这个要做到控制好流程，真正做到单步的执行，还要进行可视化的操作，让人可以看清在执行那一条操作。这也是涉及了图形化的操作界面的设计。

四、 实现结果

本次实验共涉及三种类型的 MIPS 指令，分别为 R 型、I 型和 J 型，共三种类型，基本上所有的 mips 直接指令都进行了囊括（伪指令实现较为复杂，考虑得不够多）。基本上都可以可视化运行。

具体代码和分析如下：

1. bool assembler(char* inst); //指令转机器码

通过将 inst 进行分类获得 rt、rs 等值，组成最后的机器码。

2. void fetch(char* inst, char* s);//截取对应部分字符串

通过截取部分字符串让我们获得 rt、rs 等部分的值

3. void delete_label(char* inst);//截取非@部分

4. int str2reg(char* op); //字符串转寄存器

将 op 字符串转成我们的寄存器编号

5. void UppToLow(string str); //小写转大写

为了方便进行汇编操作，将所有的大写都转化为小写，方便判断字母。

6. void DemToBin(dword machine); //机器码转二进制

7. void Createlabel(); //创建符号表

8. void ShowReg();　　//显示内存

图形化页面显示内存和寄存器的值

9. int GetType(int num);　　//判断指令格式

根据机器码判断指令的格式是三个中的那一种

10. void GetReg(int n, char* op);　//机器码转寄存器

把机器码转为寄存器，方便我们进行反汇编操作

11. void GetRInst(char* op, int func);//机器码转 R 指令操作

机器码转为 R 指令

12. void GetIInst(char* op, int n);//机器码转 I 指令操作

机器码转为 I 指令

13. void ExecuteByStep();　　//单步执行

调用函数进行单步执行汇编指令，并且进行可视化模拟

14. void getInput();　　//一行一行输入指令

调用 easyx 的 inputbox 操作，每次输入一个指令，不符合要求的指令会输出错误信息，符合的会被加入指令内存和进行汇编转化为机器码的操作。

15. void outIns();　　//在指定位置输出指令

在指定的地方输出指令，方便我们进行可视化查看

16. void initGraph();　　//进行图片准备

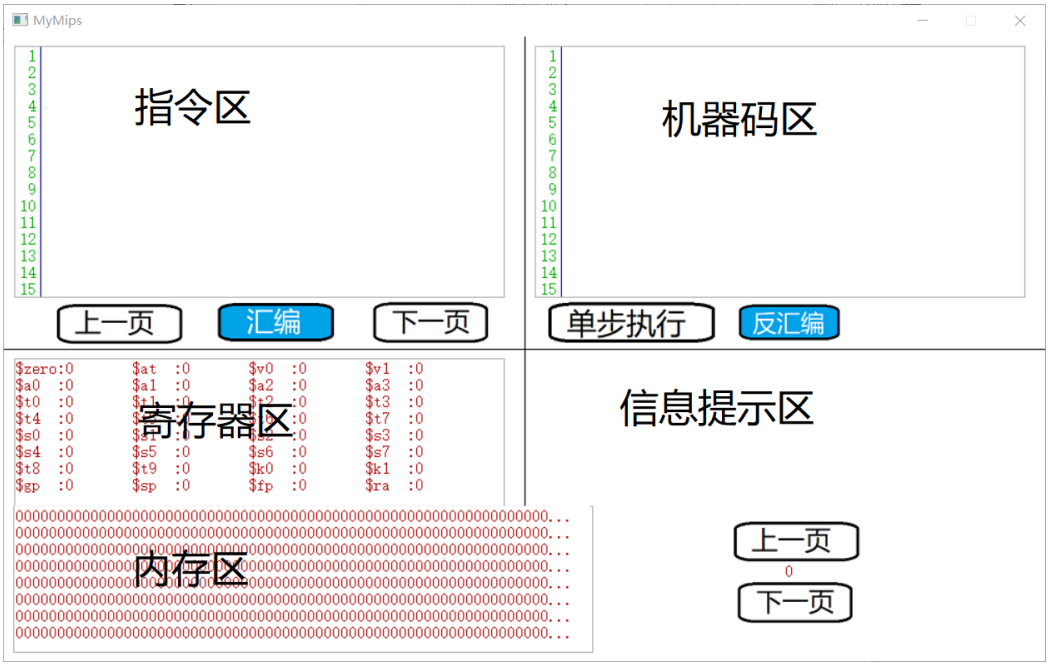
进行基础绘图的准备

17. void initReg();

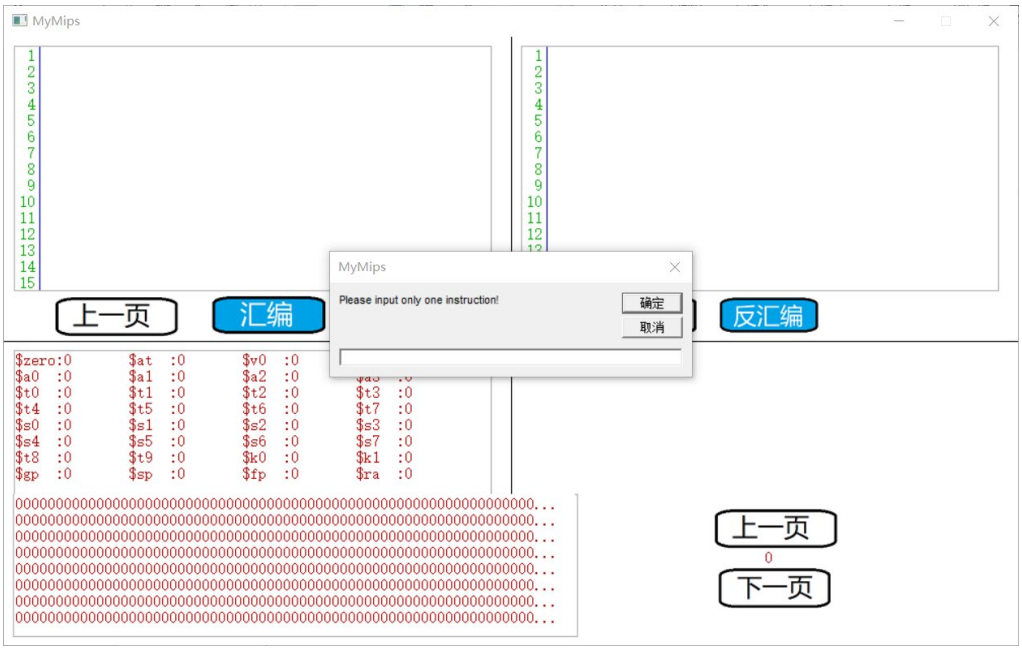
准备寄存器的字符串，方便后面进行可视化输出

五、 结果分析

我们的整体运行界面如下:

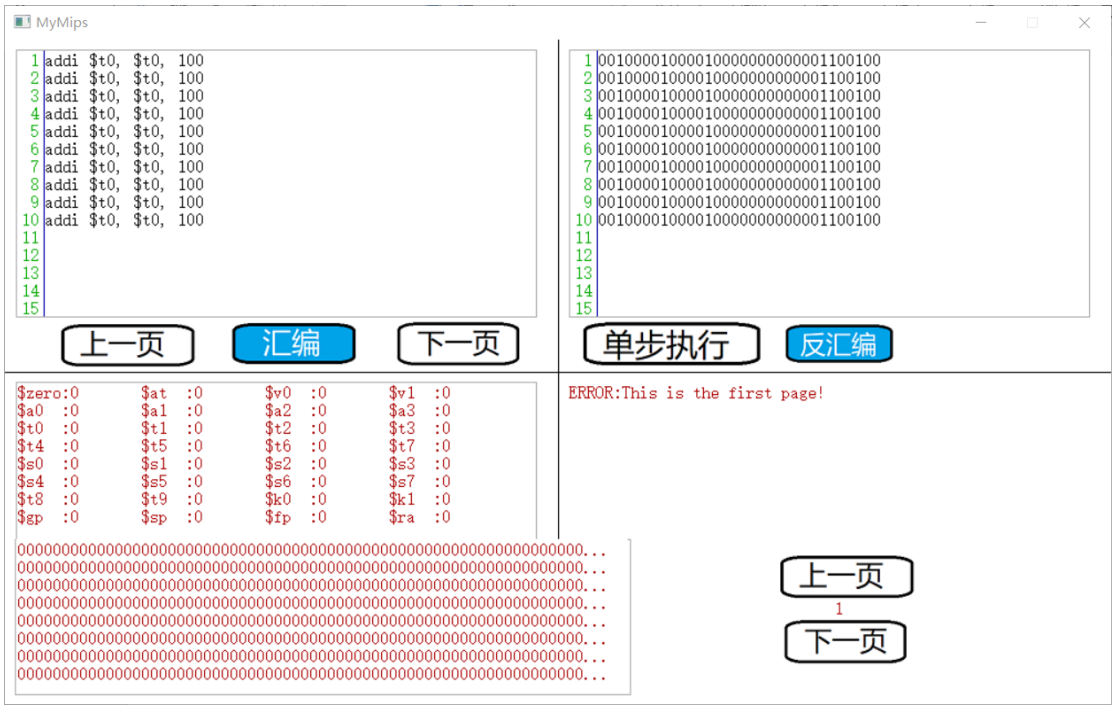


双击指令区空白可以输入指令：



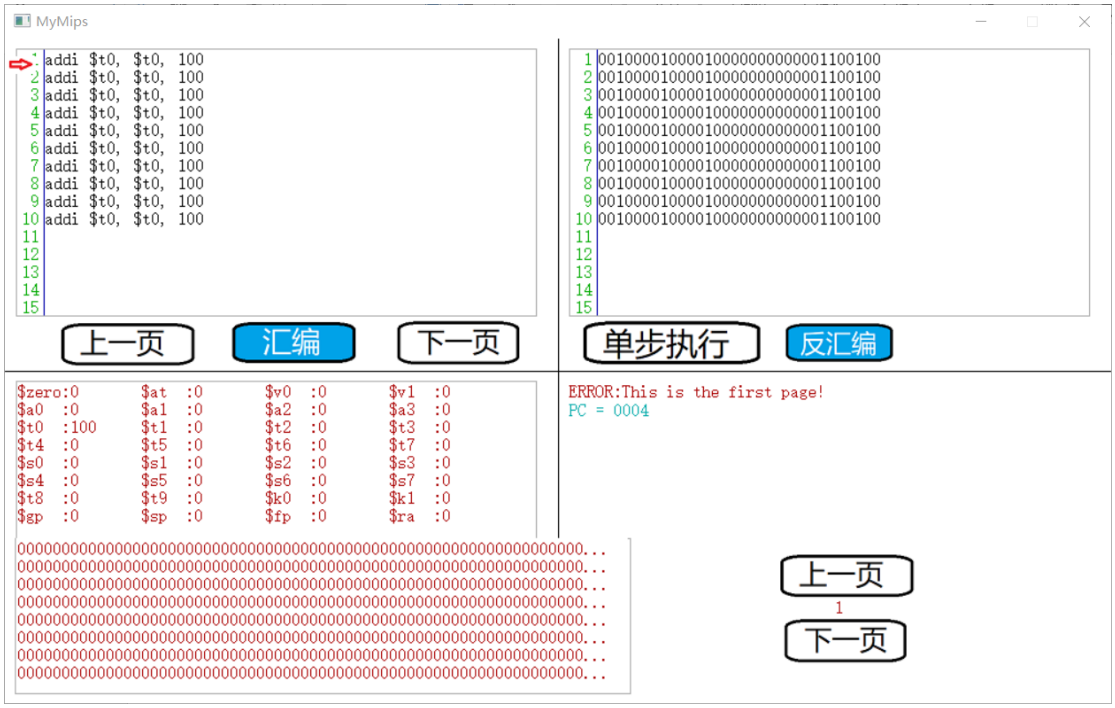
点击取消可以终止输入，每次仅允许输入一条指令

（做测试，我们输入了多条一样的指令，方便进行操作，其他指令也可正常运行，但是截图较为麻烦，可以自行测试）



点击上一页下一页（指令区下方的）可以在我们读入多条指令的时候进行翻页，查看我们读入的指令有哪些。如果页数不对，即将超页（第一页或最后一页，会再提示区提示是否超页）。右边就是我们的机器码区，会对应左边的汇编指令编译出对应的机器码，其他的指令均可以进行测试。

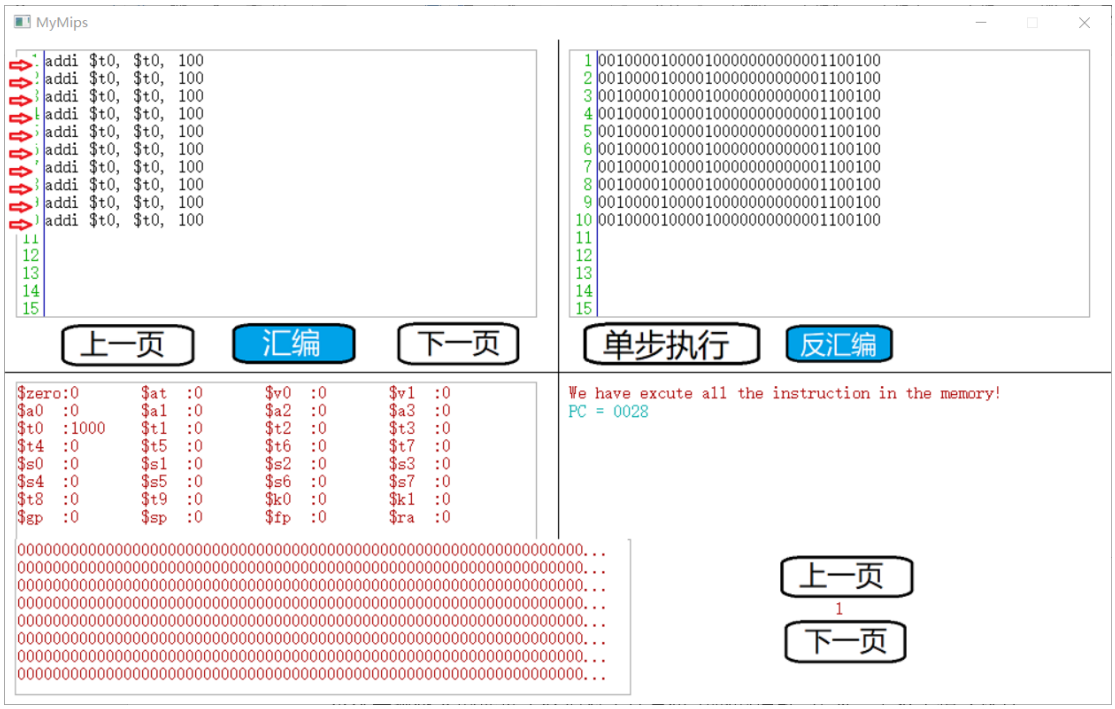
接下来我们单步执行指令：



运行时有箭头进行指示，说明是那一条指令正在运行。也会在信息区给出当前 PC 的偏移量。

执行一步后我们发现，t0 的值变了，也就是我们的指令加 100 起了作用，说明指令运行是正确的。

继续点击到后面，当执行完毕时会有错误提示信息：



我们执行了十次，t0 的值也变成了 1000，其他指令类似也可以进行这样的操作。

需要注意的是，因为数据内存是以 unsigned char 进行存储的，在进行字符输出的时候没有很好的输出办法，我只是采取了简单的 uc 转到 char 的操作，再加 ‘0’进行输出，这样会导致我们的内存有一些不对劲，很多时候会出现乱码，这个还值得改进。

六、 总结与展望

本次实验极大地促进了我们对于设计编译器的了解，让我们对字符串的理解更加深刻，也对 mips 汇编指令的构成更加清晰，这样的经验是非常有用的。

但是因为时间较为仓促，很多功能还没有一个很好的实现，或许进行更好的图形库的应用比较关键，对伪指令的进一步加深使用也是比较重要的一环，在后面应该对这两点进行深刻的学习。有待后续钻研。

组名：下岗工人再就业组

成员：令狐鹏（图形部分负责人及部分汇编代码），

黄仁泓（汇编代码），

熊迹远