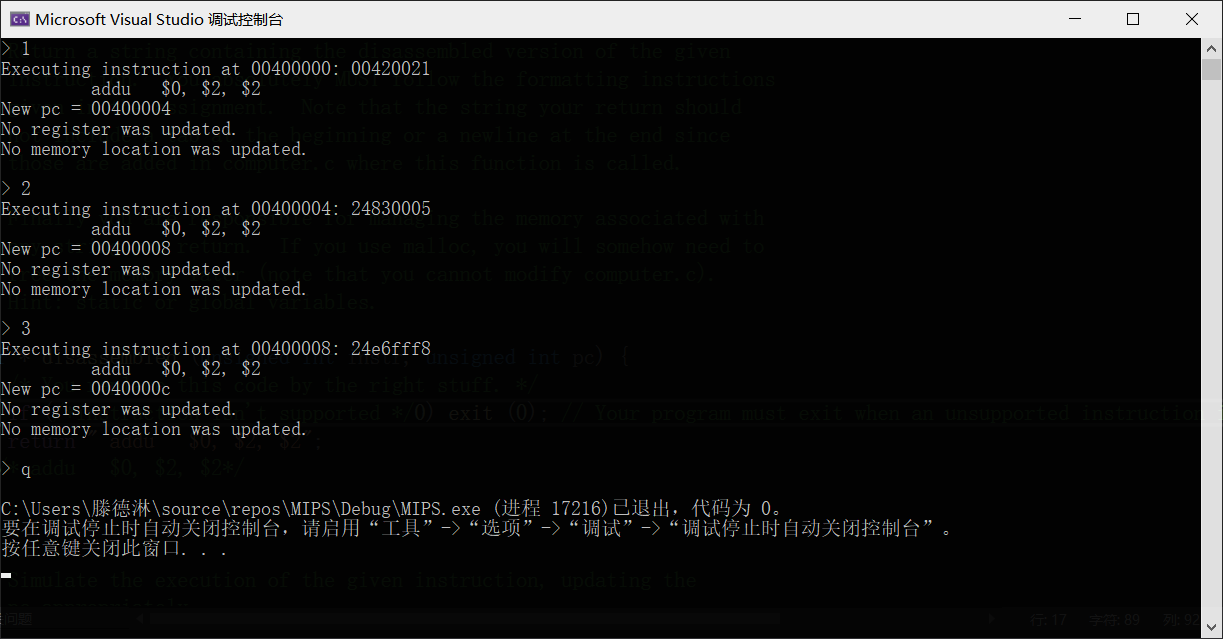
**第一步：熟悉所给代码架构**

熟悉整体框架之后，将程序运行起来；

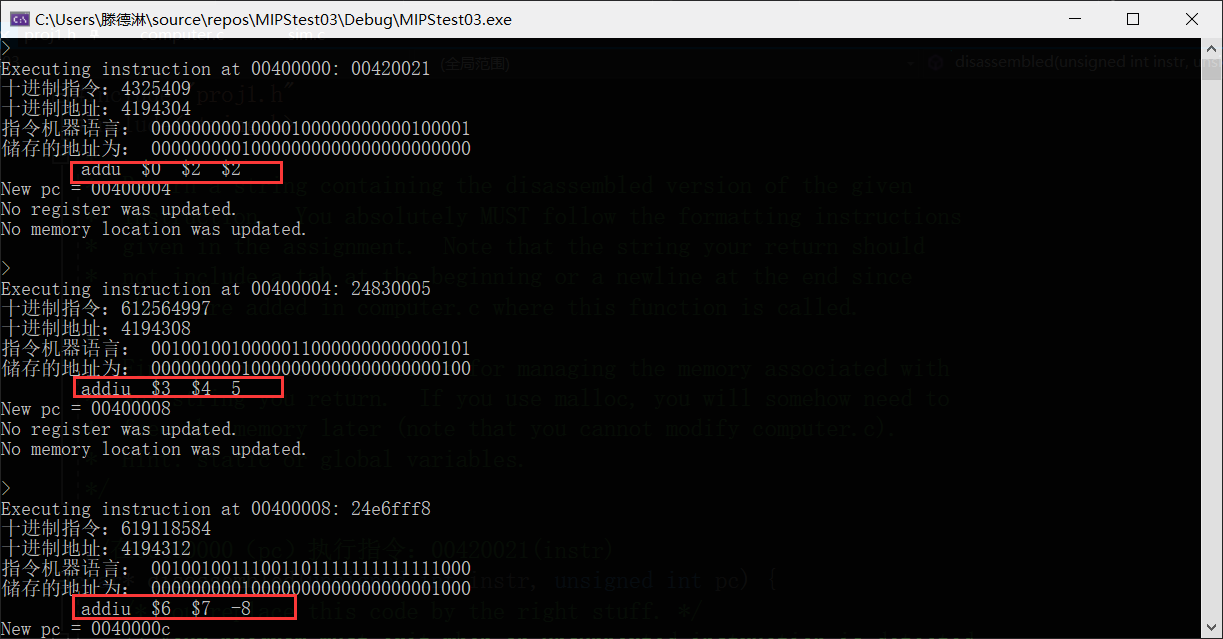
需要将sample. dump文件放入工程文件下，并在命令行参数添加-i sample.dump；调整框架中的语法错误（if处和函数不安全问题）并添加return “addu $0, $2, $2”后可以正常运行框架内容；



**第二步：编写disasmbled函数并测试**

这一步的目的就是要输出汇编指令；

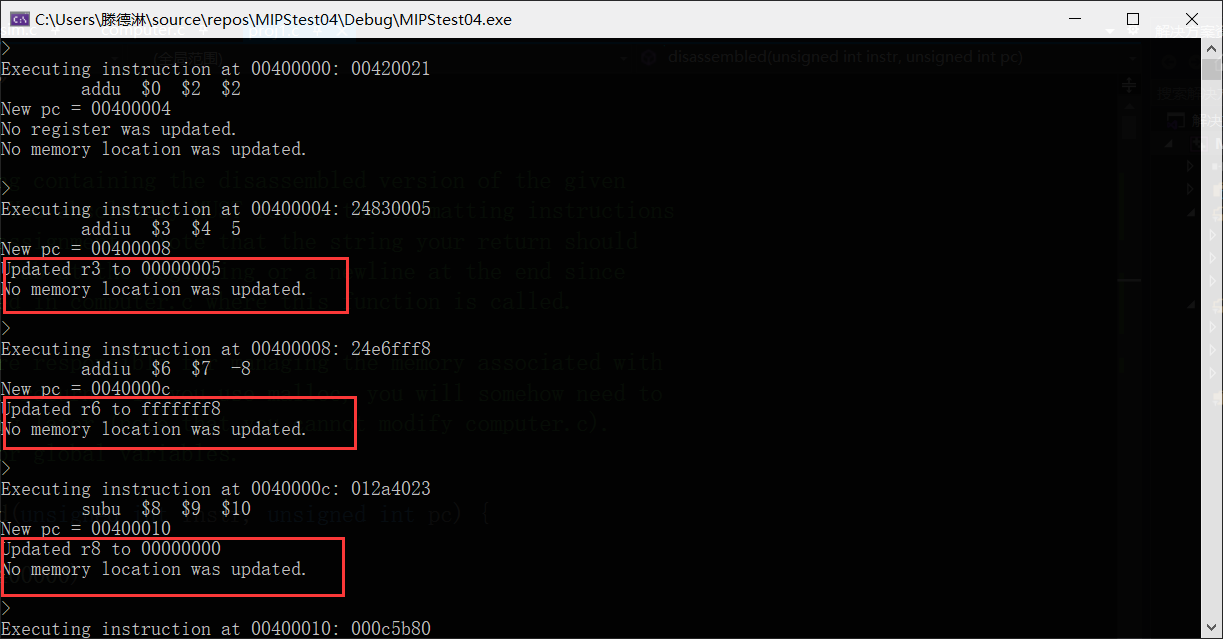
根据反汇编知识，并通过查表，一一确定条指令的二进制数；首先通过判断opcode字段分为R I J三类指令格式，再在老师的要求上分别补充18条指令；然后不同的格式分为不同的输出方式，**R格式**首先通过移位运算确定**func ，rs ，rt ，rd ，shamt**段，**I格式**分为**opcode，rs，rt，rd，imm**段，**J格式**分为**op，imm**段；最后，用多个**if，else判断**，并输出相应的指令；



**第三步：编写simulateInstr函数并测试**

这一步的目的是根据相应的指令改变相应寄存器或者内存的值；

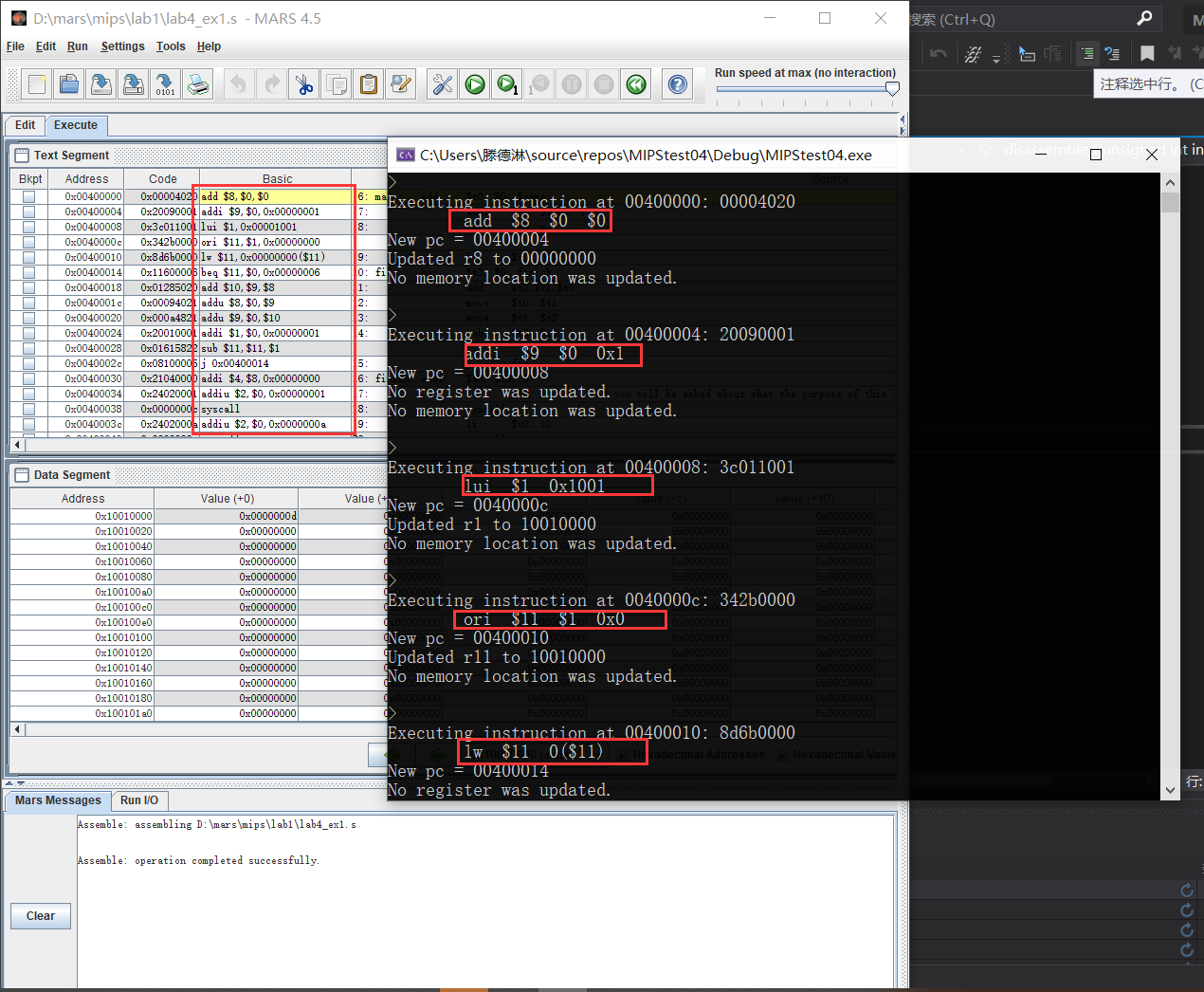
根据不同的指令在simulatelnstr函数中按照案例文件中的指令需求，并且通过指令的意思判断该如何进行更新寄存器和地址（**字段之间的更新、地址之间的跳转、字段和寄存器号数的输出等一些细节问题**）；**这一步需要深刻理解指令的含义并且用代码表示出来**；也是我认为最难的一部分；值得注意的是需要特别关注十六进制和二进制之间的区别，在更新寄存器的时候需要一步一步调试和监视；在通过百般心理斗争之后终于写出了能输出案例文件的指令的寄存器更新的方法；



**第四步：编写自己的测试程序**

这一步的目的是通过一个实例来检测前面的设计有没有问题；

首先用Mars打开一个MIPS程序（我使用的是第一次作业的第一个程序），然后汇编之后，点击存为test01.dump二进制文件；如果好奇，可以用notepad++的hex插件查看；然后再将test01.dump文件放到工程文件下面；将命令行参数改为：-i test01.dump 之后就可以运行并与Mars中的汇编指令进行对比；



**第五部：满足一些特殊的要求并打包作业**

1、满足当测试程序里面有汇编反汇编程序中没有写入的指令就退出；（if，exit）

2、配置作业包，将反汇编程序需要用到的文件放入；并且把test01.s和test01.dump放入；

3、提交；