

**实验报告**



**题目： Linux环境和GCC工具链**

**班 级： 2022211313**

**学 号： 2022211388**

**姓 名： 陈韵涵**

**学 院： 计算机学院**

**2023年 10 月 12 日**

一、实验目的

1. 熟悉linux系统的常用命令；
2. 掌握gcc编译器的使用方法；
3. 掌握gdb的调试工具使用；
4. 掌握objdump反汇编工具使用；
5. 理解反汇编程序（对照源程序与objdump生成的汇编程序）。
6. 实验环境+-

列举所使用的软件工具

1. Ubuntu
2. Gcc、gdb

三、实验内容

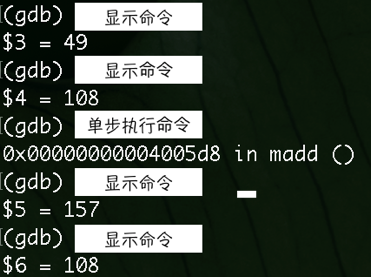
现有两个int型数组a[i]=i-50，b[i]=i+y，其中y取自于学生本人学号2022211x\*y的个位。登录bupt1服务器，在linux环境下使用vi编辑器编写C语言源程序，完成数组a+b的功能，规定数组长度为100，函数名为madd（），数组a，b均定义在函数内，采用gcc编译该程序（使用-g -fno-pie -fno-stack-protector选项），

1. 使用objdump工具生成汇编程序，找到madd函数的汇编程序，给出截图；
2. 用gdb进行调试，练习下列gdb命令，给出截图；

gdb、file、kill、quit、break、delete、clear、info break、run、continue、nexti、stepi、disassemble、list、print、x、info reg、watch

1. 找到a[i]+b[i]对应的汇编指令，指出a[i]和b[i]位于哪个寄存器中，给出截图；
2. 使用单步指令及gdb相关命令，显示a[xy]+b[xy]对应的汇编指令执行前后操作数寄存器十进制和十六进制的值，其中x，y取自于学生本人学号2022211x\*y的百位和个位。

学号2022211999，a[99]+b[99]单步执行前后的参考截图如下（实际命令未显示出）：



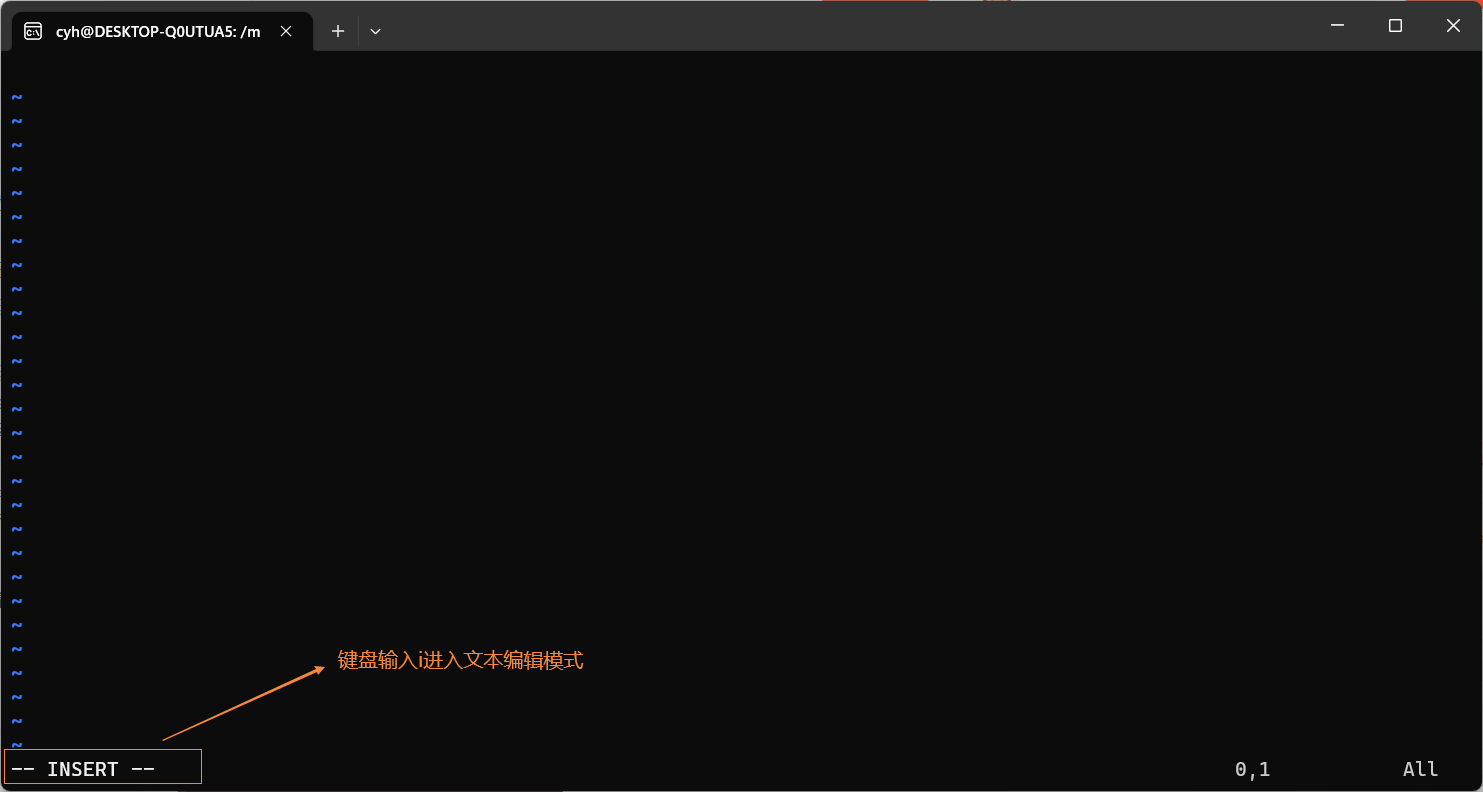
四、实验步骤及实验分析

需要给出操作步骤、运行截图、分析过程的内容

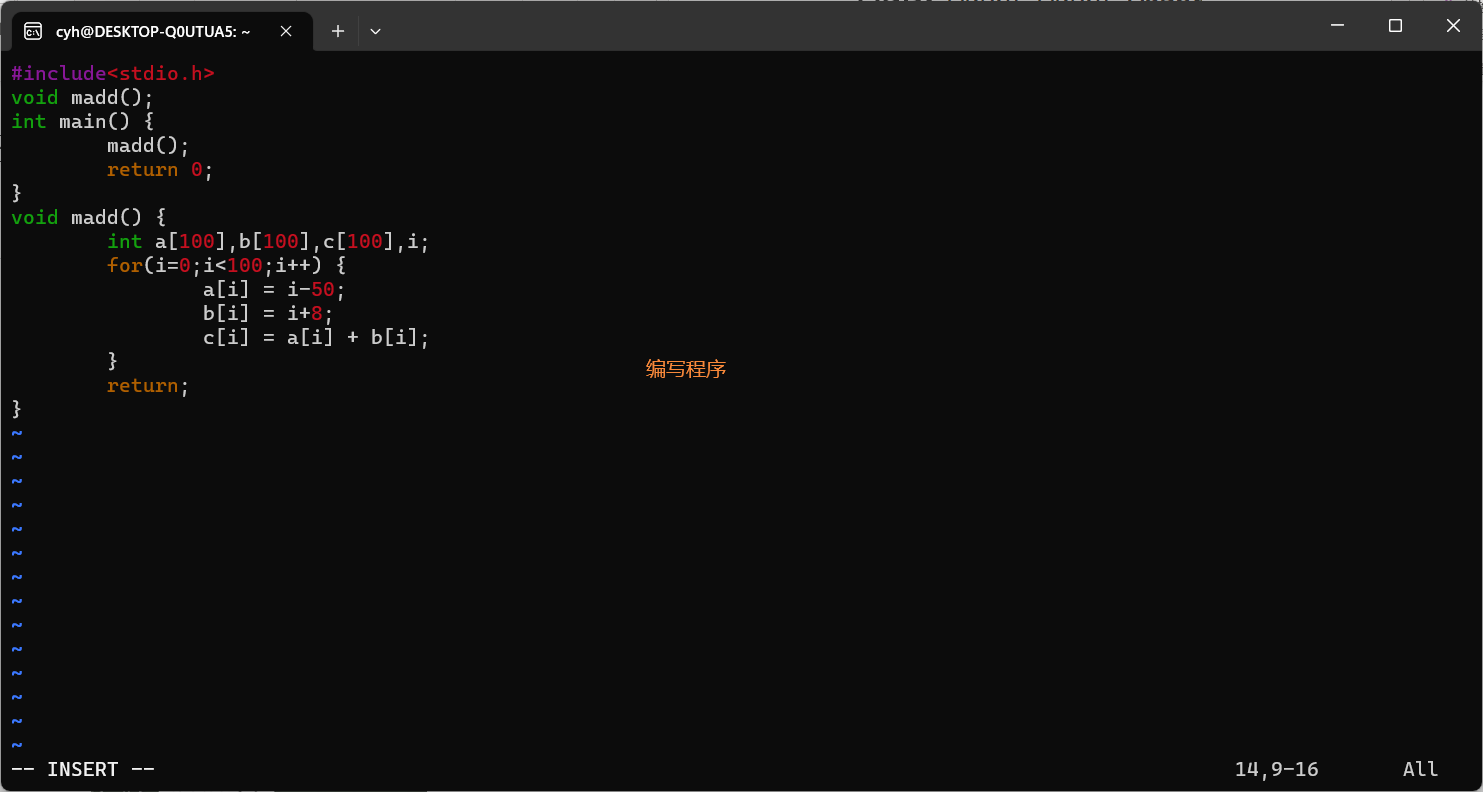
1. 使用objdump工具生成汇编程序，找到madd函数的汇编程序，给出截图

①输入vi Madd.c用vi个C语言源文件

②键盘输入i进入插入模式



③编写程序

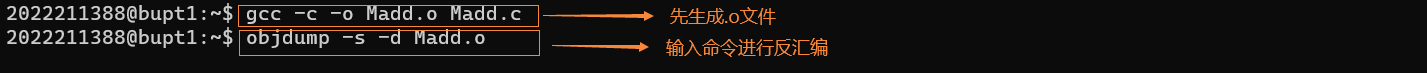


④编写程序结束后，按Esc然后输入:wq保存并退出编辑

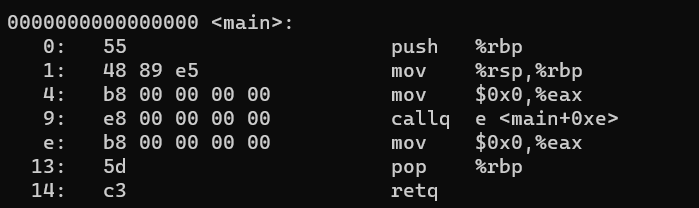
⑤编译文件

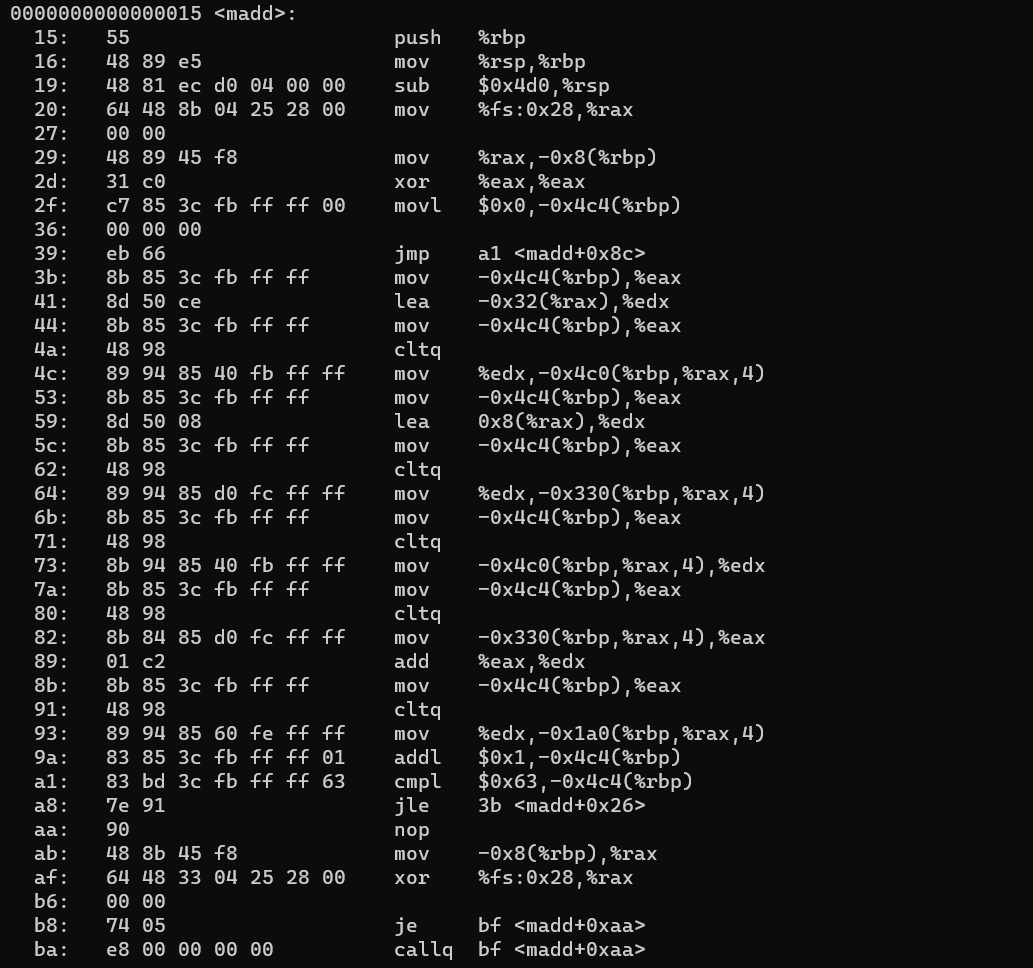


⑥



⑦

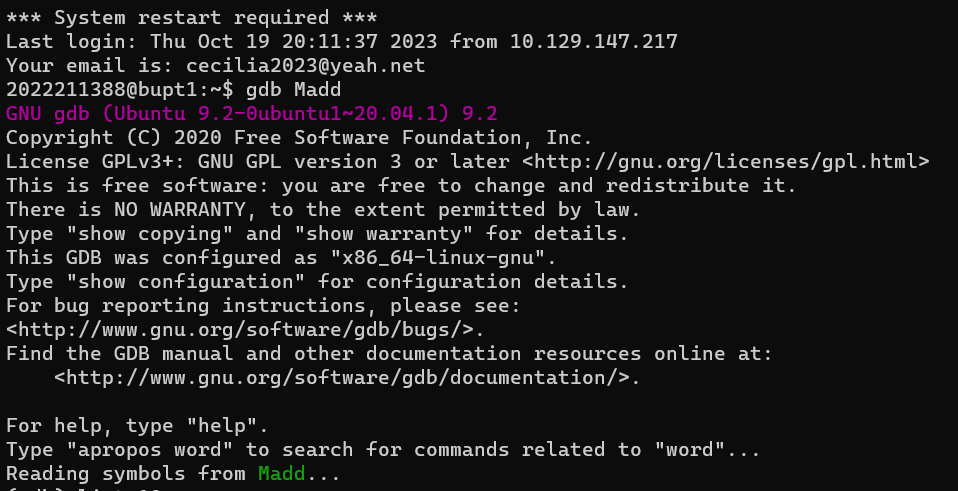




2. 用gdb进行调试，练习下列gdb命令，给出截图；

gdb、file、kill、quit、break、delete、clear、info break、run、continue、nexti、stepi、disassemble、list、print、x、info reg、watch

①gdb



②file



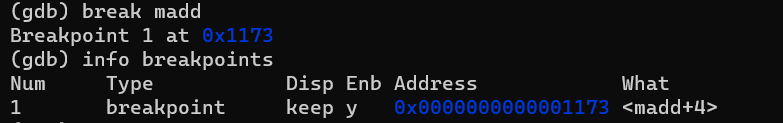
③kill



④quit 退出调试



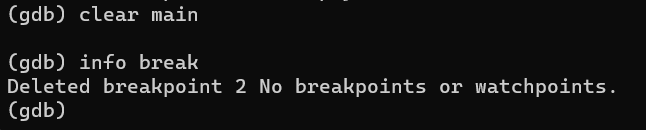
⑤break



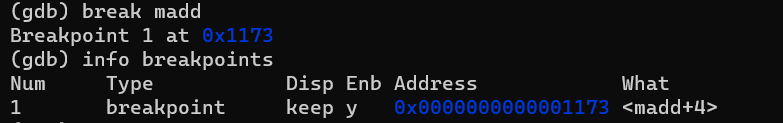
⑥delete 删除序号为1的断点



⑦clear

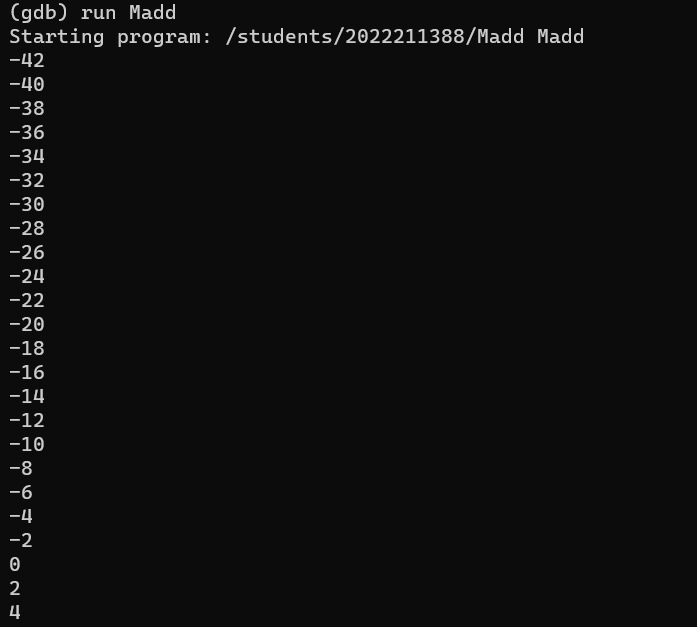


⑧info break

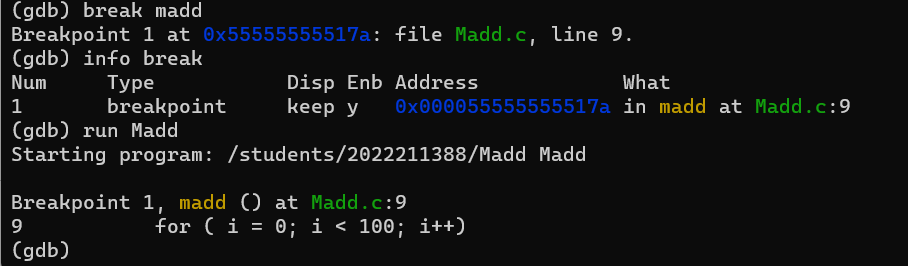


⑨run

在启动程序之前我在源文件里面又添加了打印c[i]的值

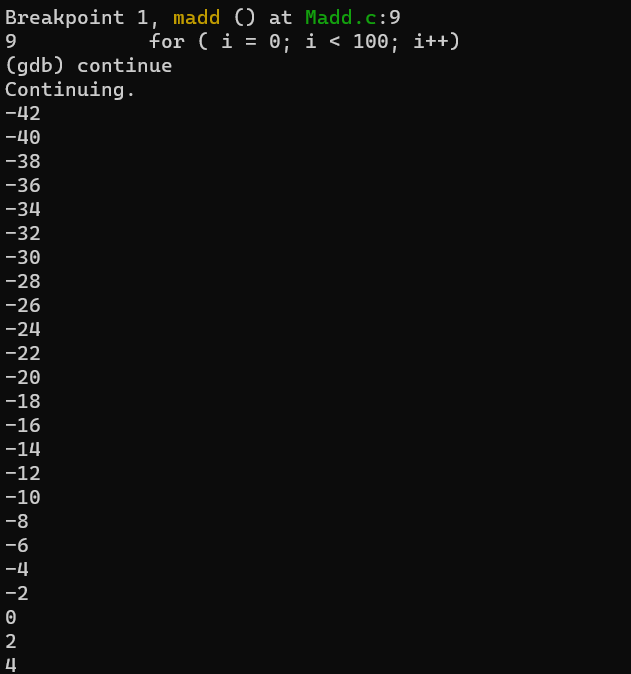


添加断点后：



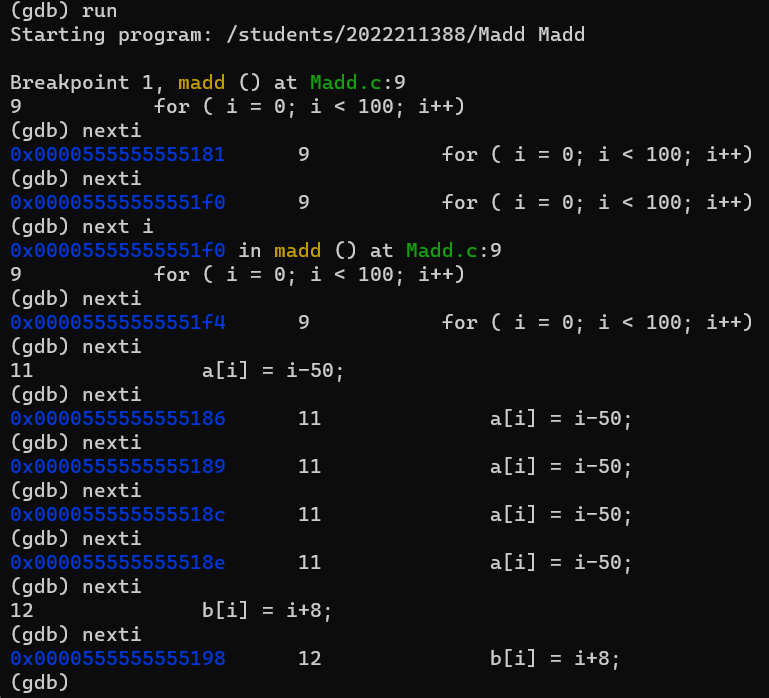
⑩continue

Gdb continue以后程序开始继续执行



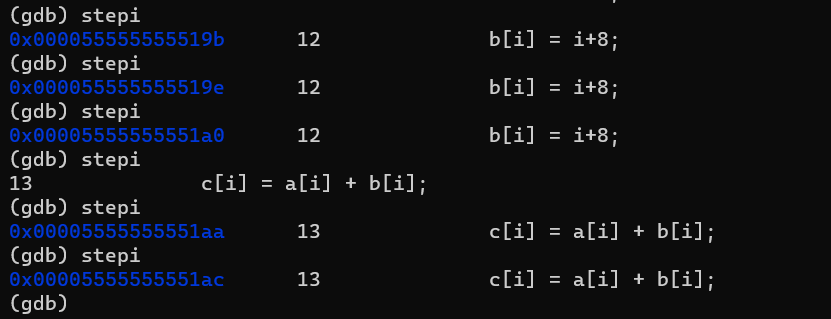
①①nexti

单步调试程序的一种方法。单步一条机器指令，不进入函数。



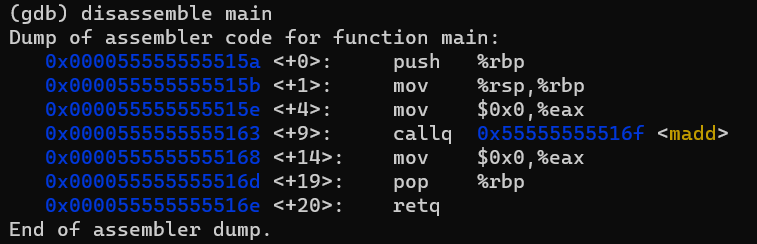
①②stepi

单步一条机器指令。

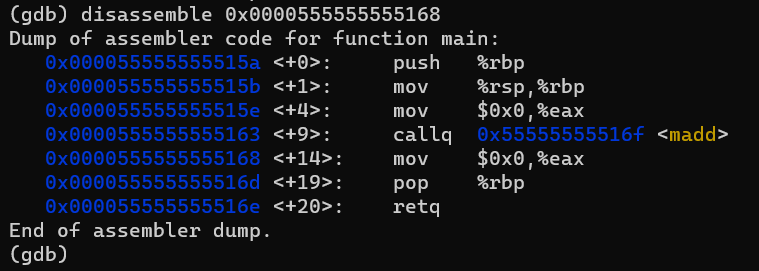


①③disassemble

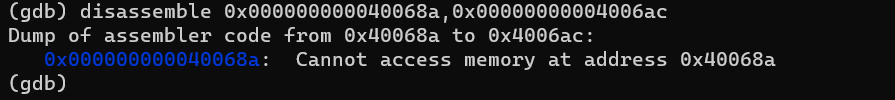
指定要反汇编的函数。如果指定，反汇编命令将产生整个函数的反汇编输出。



指定要反汇编的地址。请注意，当仅指定一个地址时，此命令将反汇编包含给定地址的整个函数，包括其上方的指令。

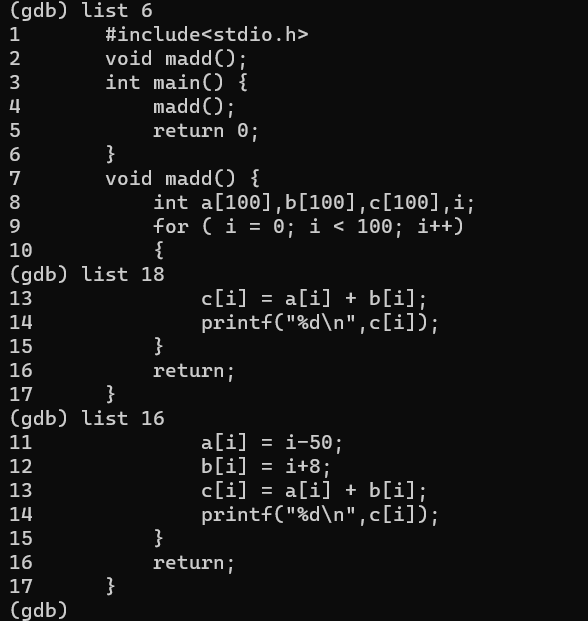


指定要反汇编的起始地址和结束地址。如果使用这种形式，该命令将不会反汇编整个函数，而只会反汇编起始地址和结束地址之间的指令。

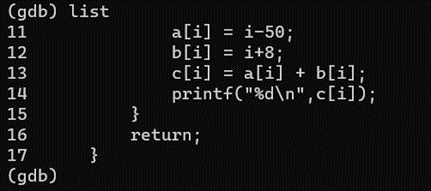


①④list

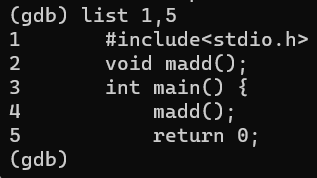
list <linenum>显示程序第linenum行的周围的源程序。



list显示当前行后面的源程序。

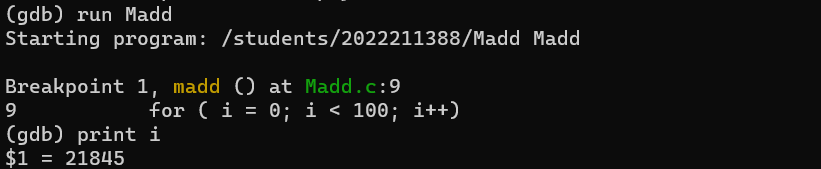


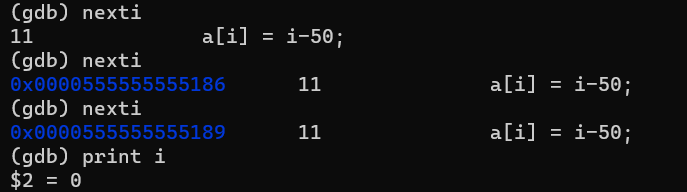
list <first>, <last>显示从first行到last行之间的源代码。

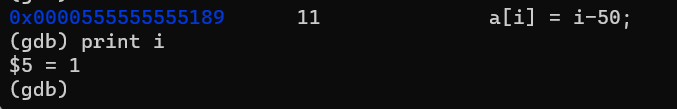


①⑤print

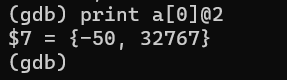
打印变量值





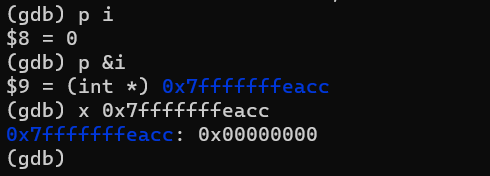


打印数组元素值



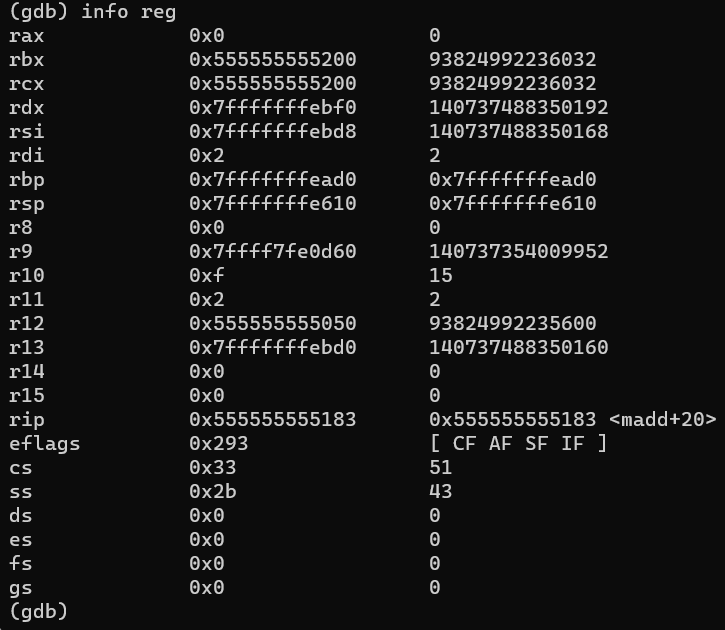
①⑤x

先查看i的地址，再用x命令查看该地址的值



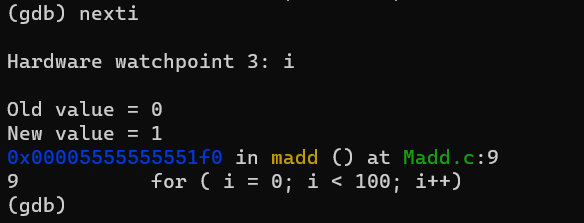
①⑥info reg

看到堆栈信息



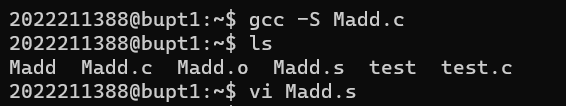
①⑦

当监视的东西发生变化时，程序会停下来



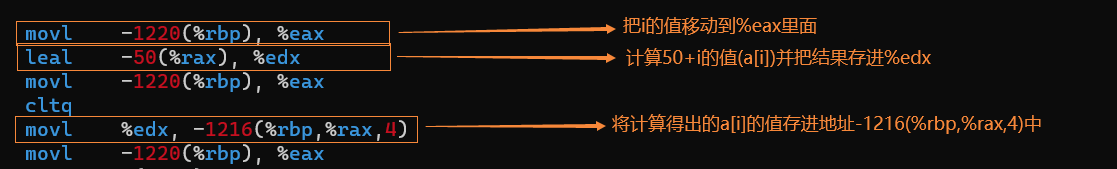
3. 找到a[i]+b[i]对应的汇编指令，指出a[i]和b[i]位于哪个寄存器中，给出截图

首先生成汇编文件，然后利用vi编辑器查看汇编代码。



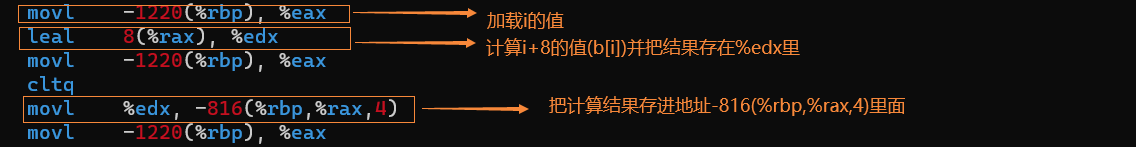
用vi编辑器打开汇编文件：

先查看跟给a[i]赋值相关的指令



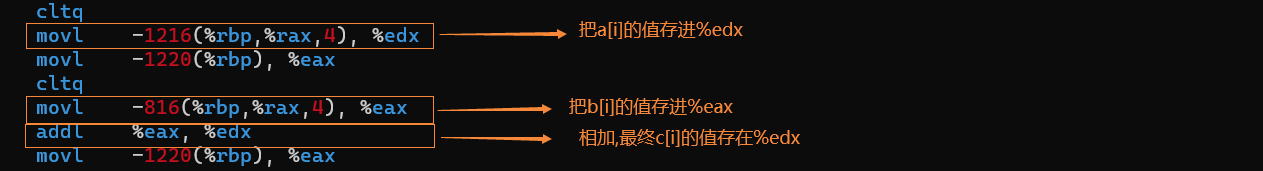
可以看到最开始a[i]的值是存在地址-1216(%rbx,%rax,4)中的

再查看跟给b[i]赋值相关的指令



可以看到最开始b[i]的值是存在地址-816(%rbp,%rax,4)中的

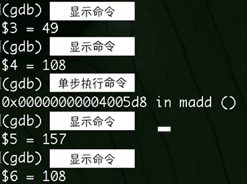
查看跟a[i]+b[i]相关的指令



因此，在执行a[i]+b[i]这条指令时，a[i]的值存在寄存器%edx，b[i]的值存在寄存器%eax

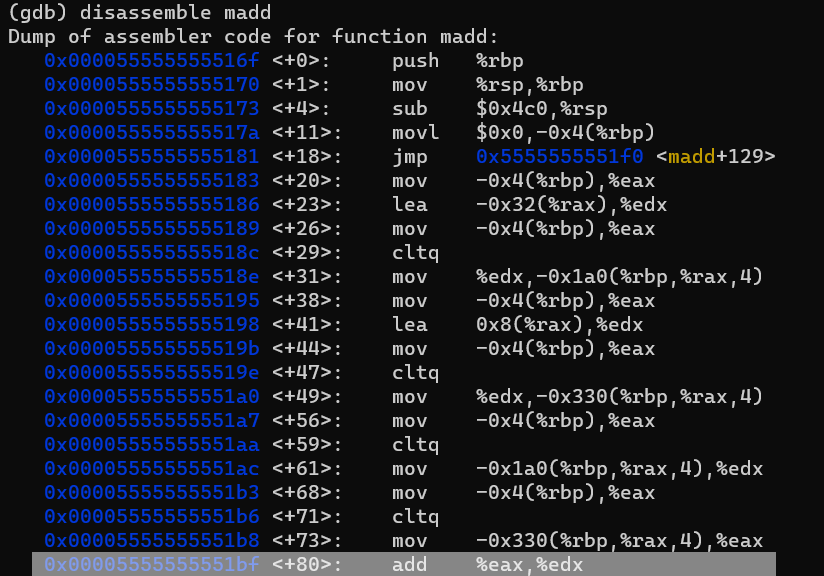
4. 使用单步指令及gdb相关命令，显示a[xy]+b[xy]对应的汇编指令执行前后操作数寄存器十进制和十六进制的值，其中x，y取自于学生本人学号2022211x\*y的百位和个位。

学号2022211999，a[99]+b[99]单步执行前后的参考截图如下（实际命令未显示出）：



我的xy = 38

先执行(gdb) disassemble madd查看madd函数的汇编指令，可以看出a[xy]+b[xy]对应的汇编指令是下图这一条：

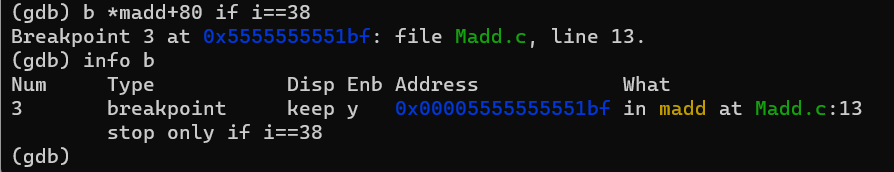


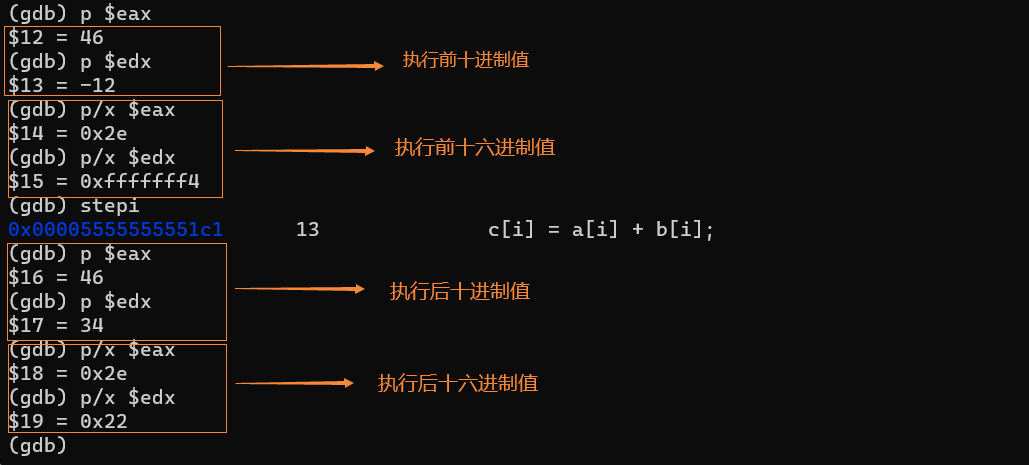
显示a[xy]+b[xy]对应的汇编指令执行前后操作数寄存器十进制和十六进制的值：

可以看到操作数的寄存器是%eax和%edx，其中%eax是目的操作数，%edx是源操作数。

由上一题可以得知：此时%eax存的是b[i]的值，%edx存的是a[i]的值

新建条件断点：





五、总结体会

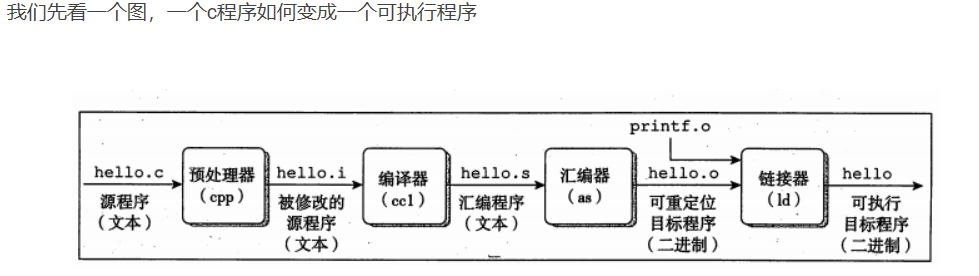
总结心得（包括实验过程中遇到的问题、如何解决的、意见和建议等）

1. vi编辑器最开始不会用，于是我就气急败坏地乱点，但是这并没有什么帮助，最后在网络上看了vi编辑器的详细使用方法，了解了它的三个状态，才知道如何输入以及如何保存。

这说明在遇到问题是，还是要查资料充分了解，才能进行下一步工作。

2. 还没有生成汇编文件就开始反汇编。

一开始不了解反汇编的原理，即：利用反汇编器对.o文件进行反汇编，还没有生成.o文件就用了objdump，自然是会报错的。说明在进行一个命令前还是要充分了解它到底是干嘛的，以及了解编译器的整个工作流程，才更利用以后的使用。



3. 在做第三题时，只关注了a[i]+b[i]语句的add指令那一条，就浅显地以为a[i]和b[i]单纯地就存在寄存器%edx和寄存器%eax里面，直到做第四题单步调试的时候才发现不对劲，因为在进行加法运算之前的赋值运算时，%eax里面存的一直是i的值，而%edx里面时而是a[i]的值时而是b[i]的值。解决方法是，单步调试+认真阅读汇编文件，这才搞清楚原来a[i]的值和b[i]的值是在进行加法运算时才分别移动到%edx和%eax里面的。