

**实验报告**



**题目： Linux系统及其相关软件环境**

**班 级： 2022211305**

**学 号： 2022211119**

**姓 名： 赵宇鹏**

**学 院： 计算机学院（国家示范性软件学院）**

**2022年 10 月 19 日**

1. 实验目的
2. 熟悉linux操作的基本操作；
3. 掌握gcc编译方法；
4. 掌握gdb的调试工具使用；
5. 掌握objdump反汇编工具使用；
6. 熟悉理解反汇编程序（对照源程序与objdump生成的汇编程序）。

二、实验环境（5分）

简述使用的工具

1. SecureCRT或其他远程登陆工具（服务器：10.120.11.12）

**SecureCRT 是一种远程登录工具，通常用于通过 SSH、Telnet 或其他协议与远程服务器或网络设备建立连接。也可以通过Windows terminal通过ssh进行连接。**

1. Linux

**Linux 是一种类 Unix 操作系统，Linux 提供了命令行界面，通过终端或控制台可以在系统上执行命令，Linux 是一个多功能、多用户、多任务的开源操作系统。**

1. Gcc编译器

**GCC是一套由 GNU 项目开发的开源编译器集合，用于编译多种编程语言的源代码为可执行程序。GCC编译器能将C、C++语言源程序、汇编程序编译、链接成可执行文件。在Linux系统中，可执行文件没有统一的后缀，系统从文件的属性来区分可执行文件和不可执行文件。**

1. GDB调试工具

**GDB是一款GNU开发组织并发布的UNIX/Linux下的程序调试工具，它使你能在程序运行时观察程序的内部结构和内存的使用情况。**

1. Objdump命令反汇编

**objdump 是一个用于反汇编可执行文件、目标文件和共享库的命令行工具。它是 GNU 工具链的一部分，用于分析二进制文件的内容，包括可执行代码、符号表、段信息等。**

三、实验概况

简述实验内容和基本设想

实验内容一（15分）

在linux环境下，编辑课件中源程序（注意程序的完整性）（包含源程序的开发环境截图），采用gcc编译该程序（要求分别采用-o和-O参数，并比较两者性能，编译指令截图），采用gdb进行调试，让程序运行到for函数语句（调试截图），运用objdump工具生成汇编程序（给出main函数的汇编程序截图）

#include<stdio.h>

int main(void)

{

double counter;

double result;

double temp;

for(counter=0;counter<2000.0\*2000.0\*2000.0/20.0+2020;

counter+=(5-1)/4){

temp=counter/1979;

result=counter;

}

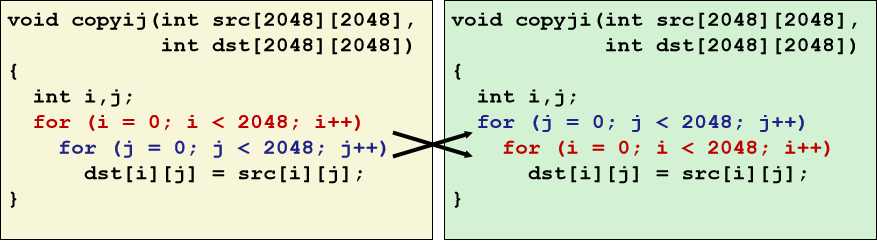
printf(Result is%lf\\n,result);

return 0；

}

实验内容二（15分）

在linux环境下，分别打印输出如下算法所需时间



分别设置不同优化参数，给出运行时间

实验内容三（30分）

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；

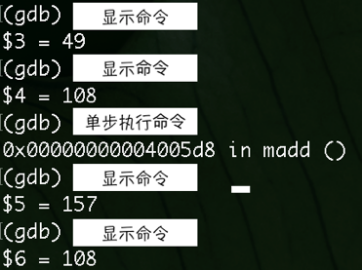
gdb、file、kill、quit、break、delete、clear、info break、run、continue、nexti、stepi、disassemble、list、print、x、info reg、watch

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；

a[i]和b[i]位于eax以及edx寄存器中

1. 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）：



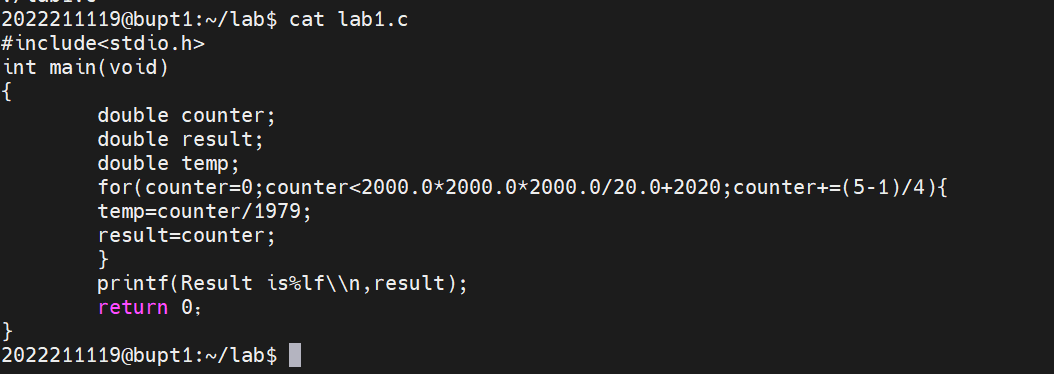
实验内容四（加分项，20分）

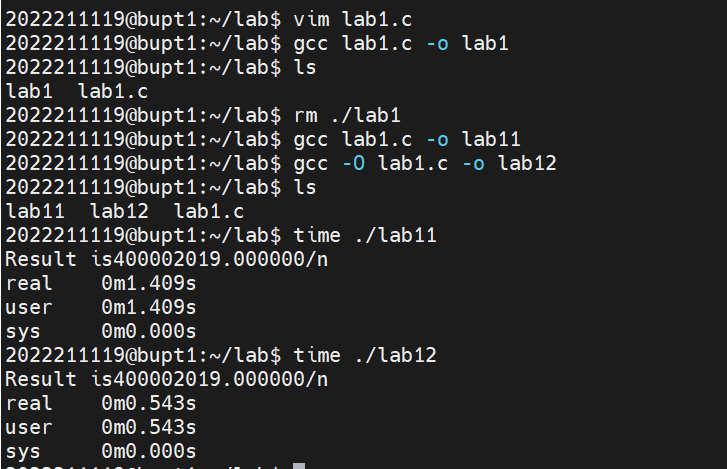
任选高复杂度算法（具体算法自选，类型分为高计算量类型和高内存需求类型2类算法），通过设置不同优化参数，分析算法的运行效率

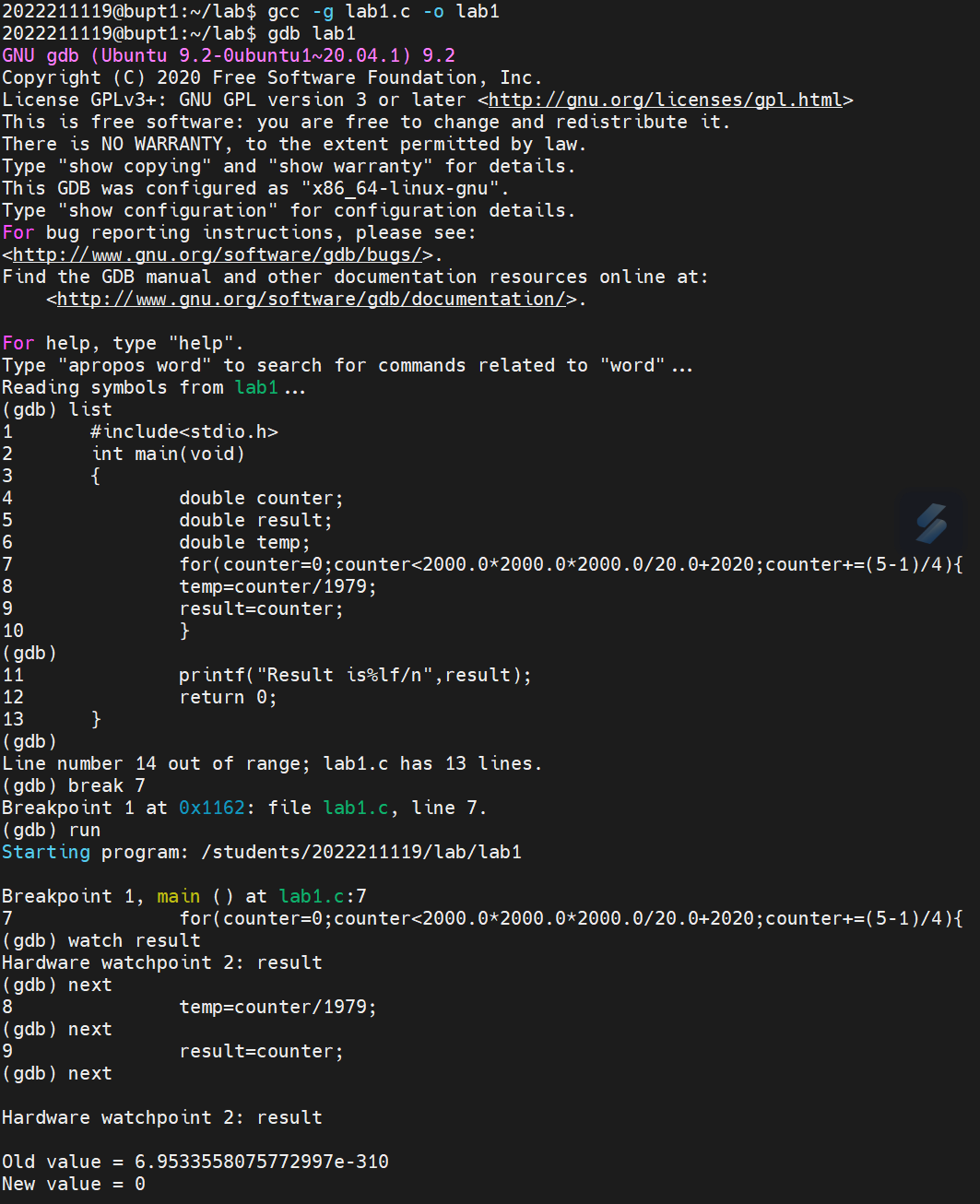
本算法为计算斐波那契数列的高计算量算法，可见在优化后算法的执行效率 大大提升。

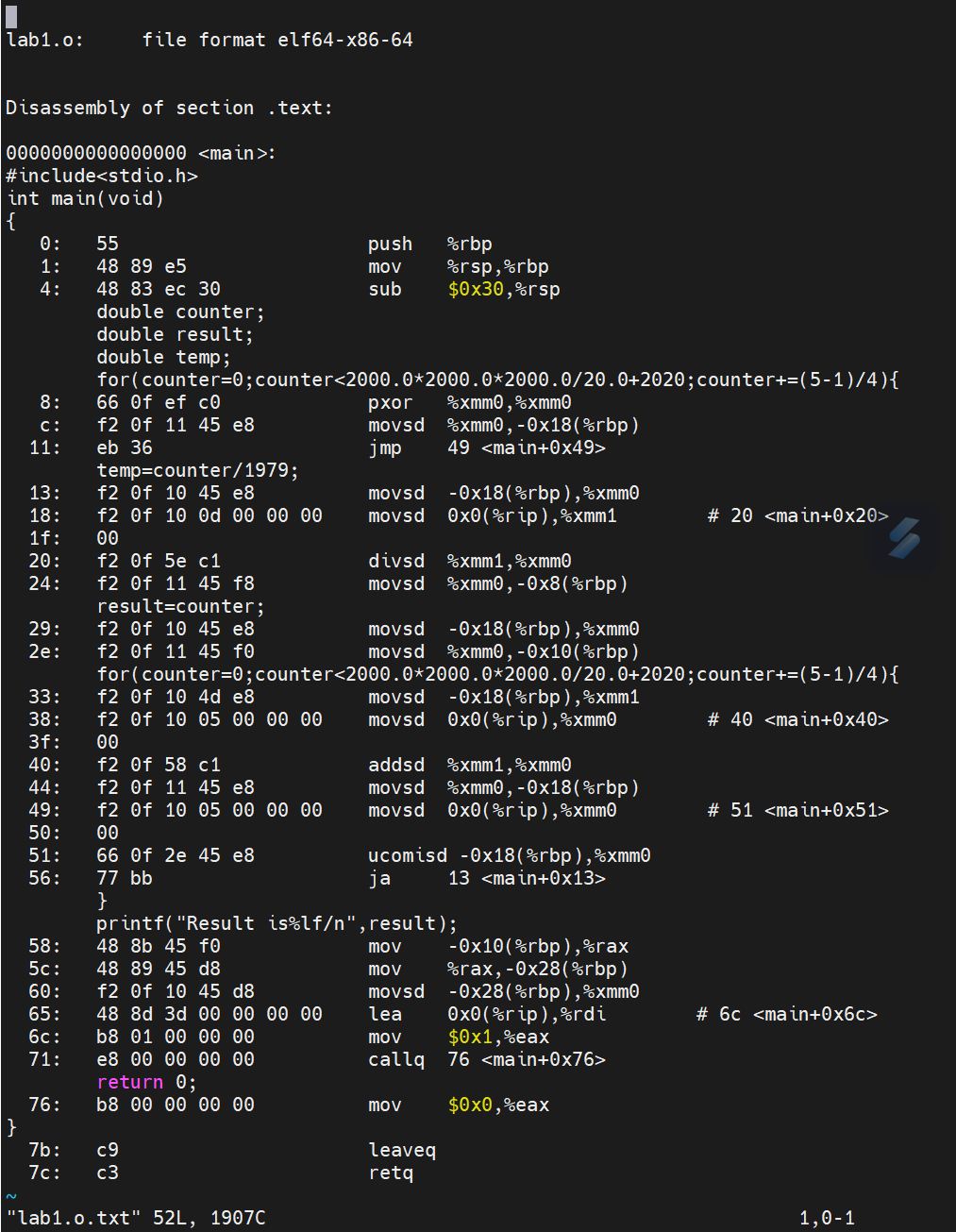
四、实验步骤（60-80分）

**实验一**

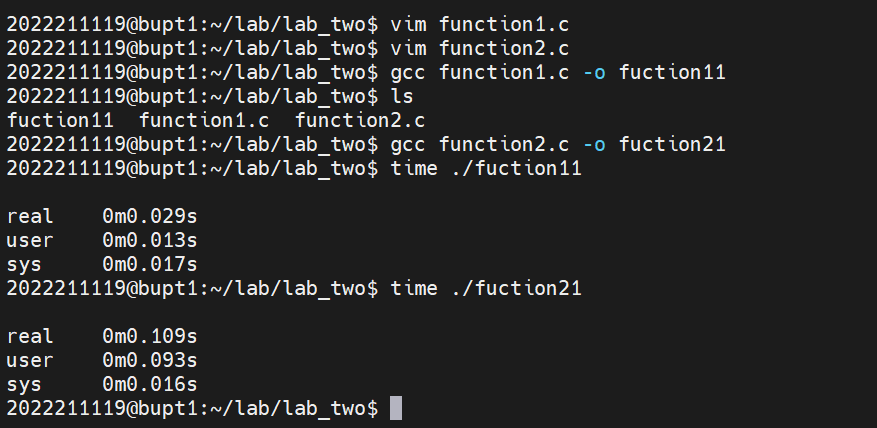


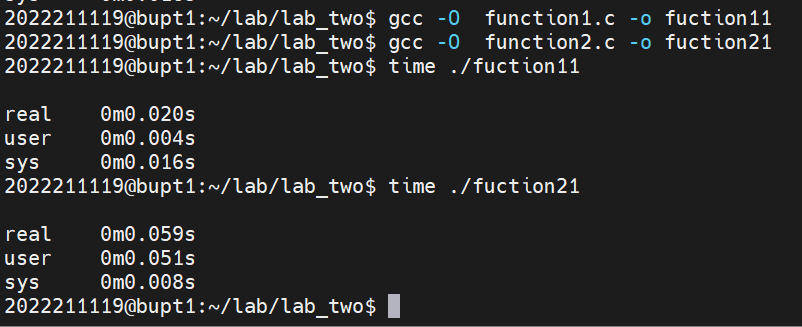


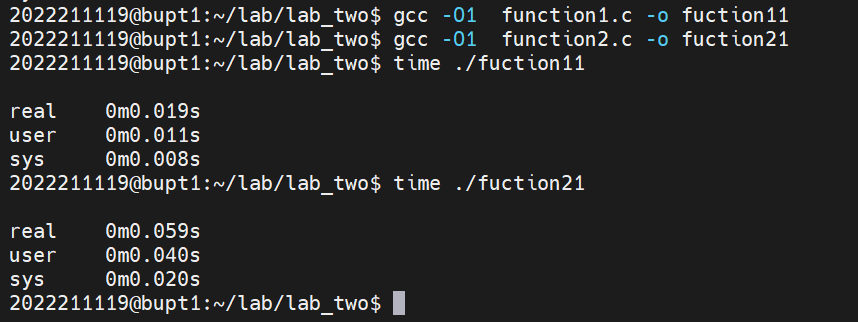


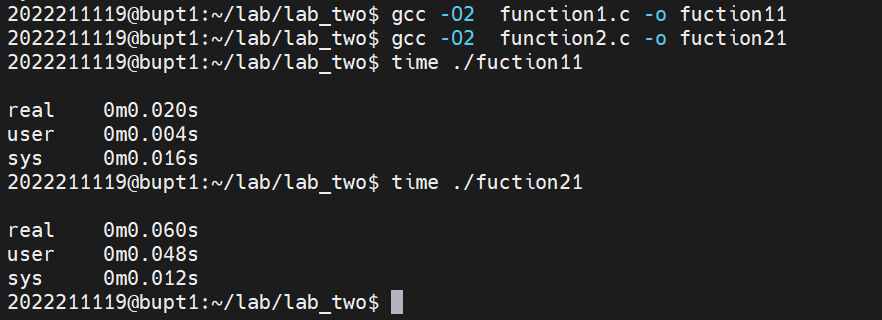


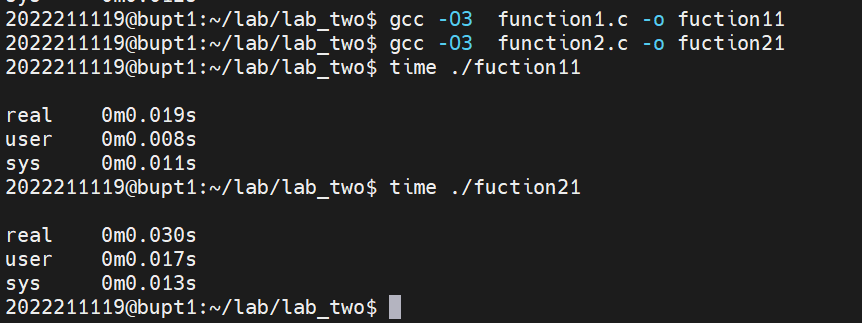
**实验二**

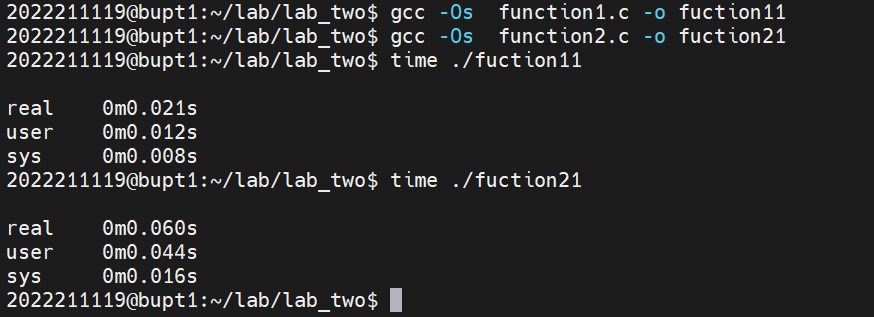






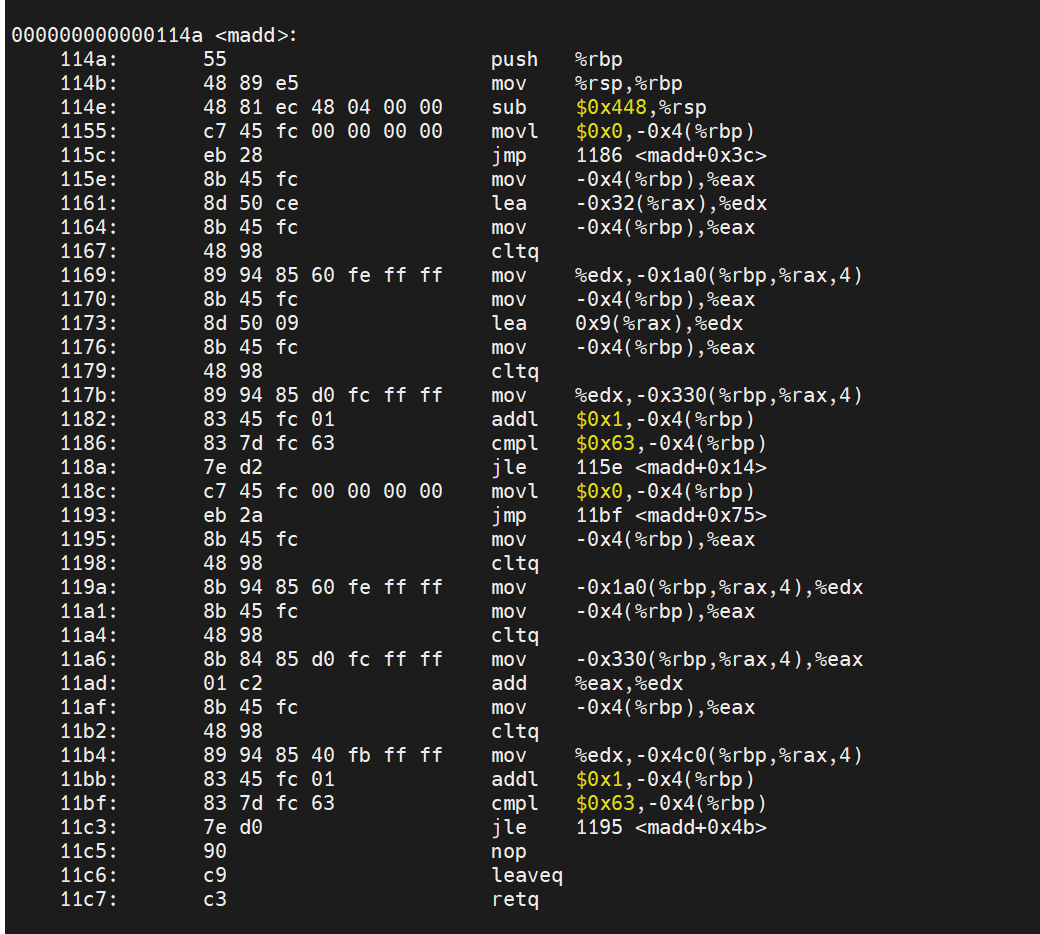




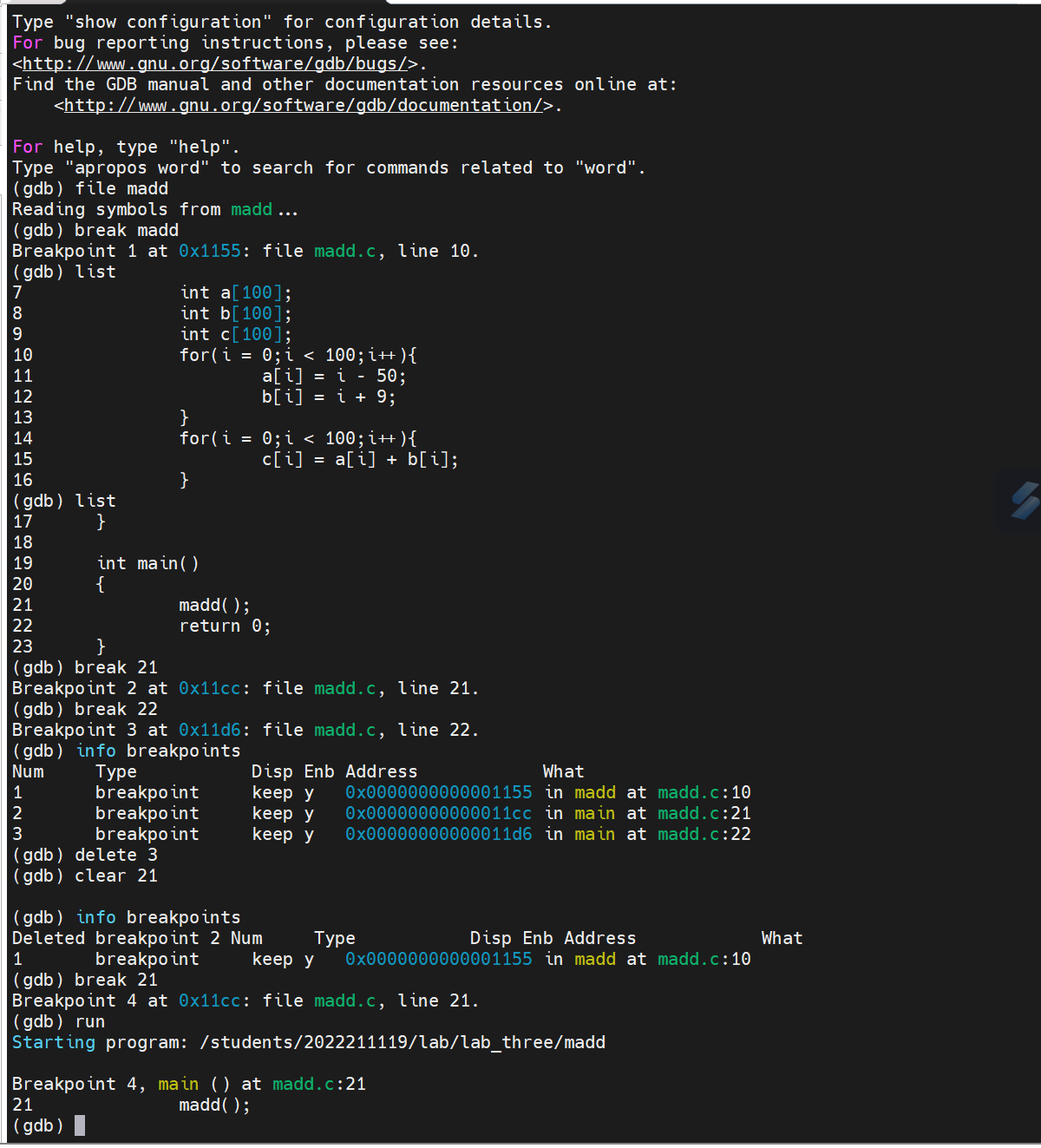


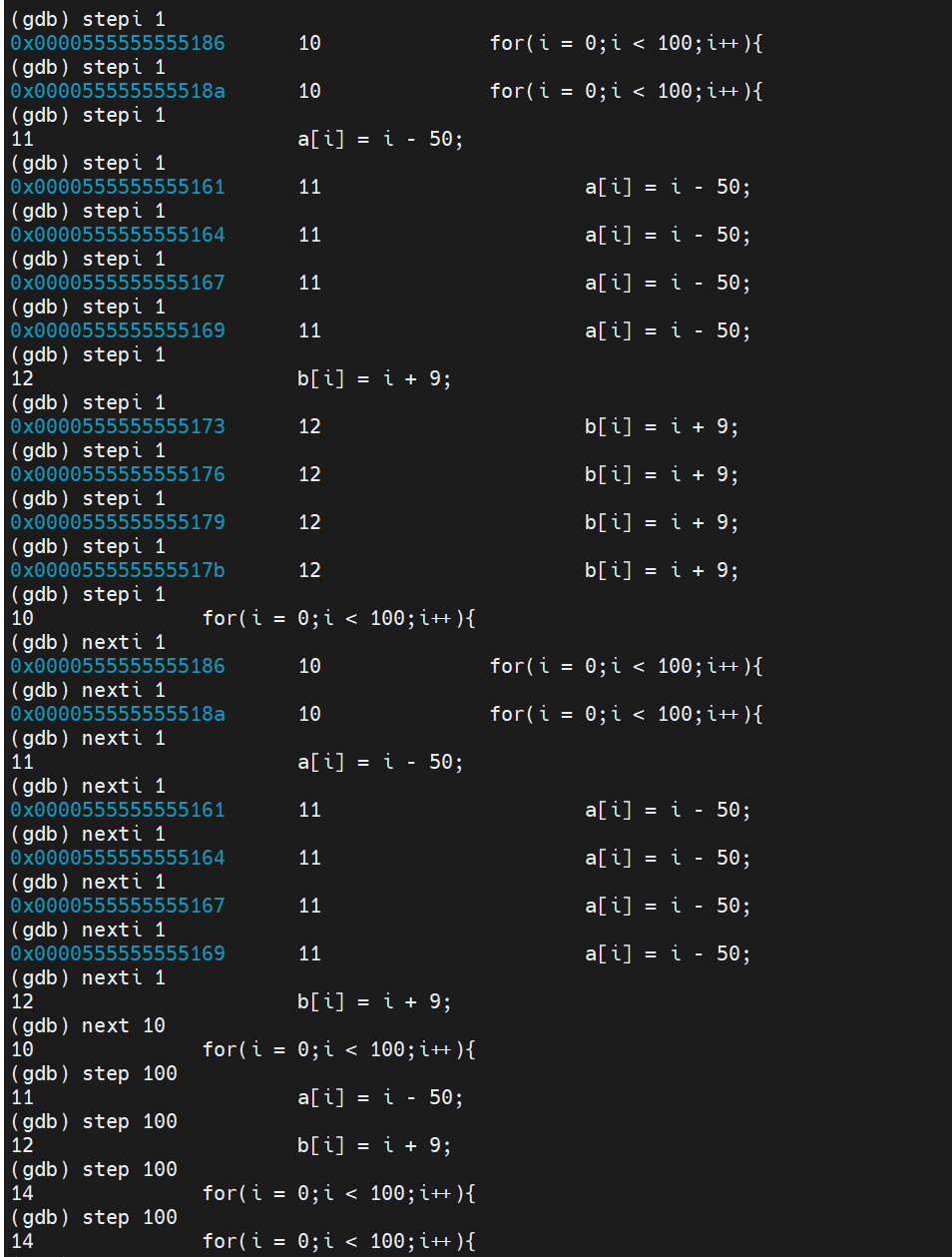
**实验三**

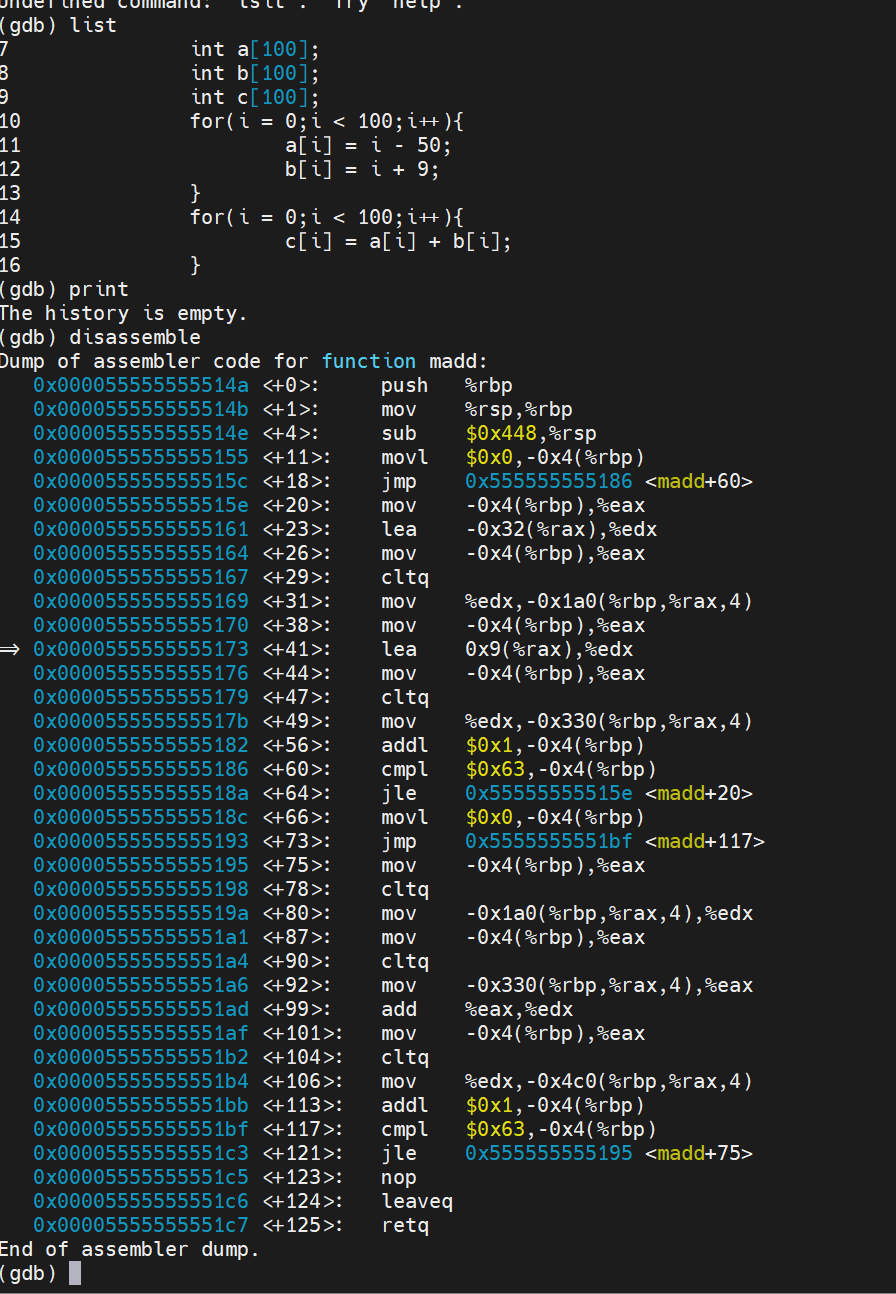
**1、**

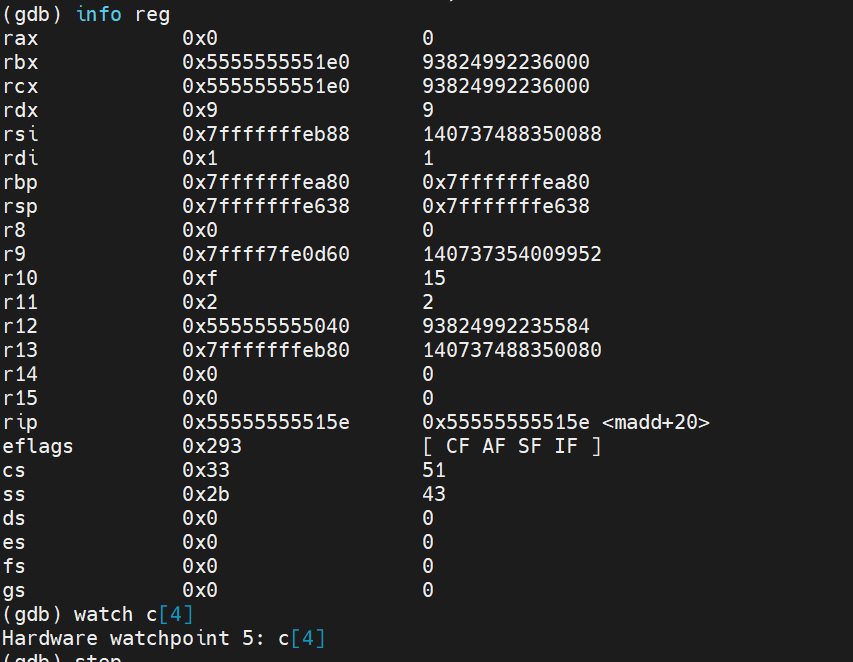


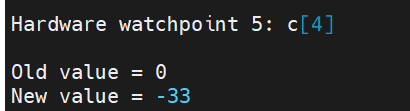
2、

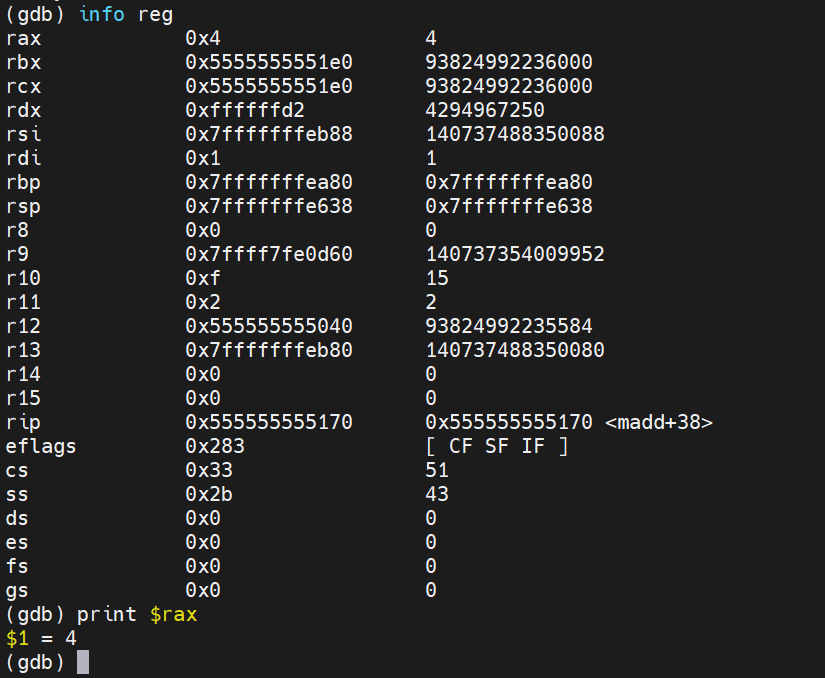




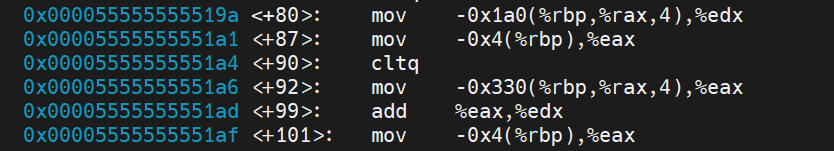






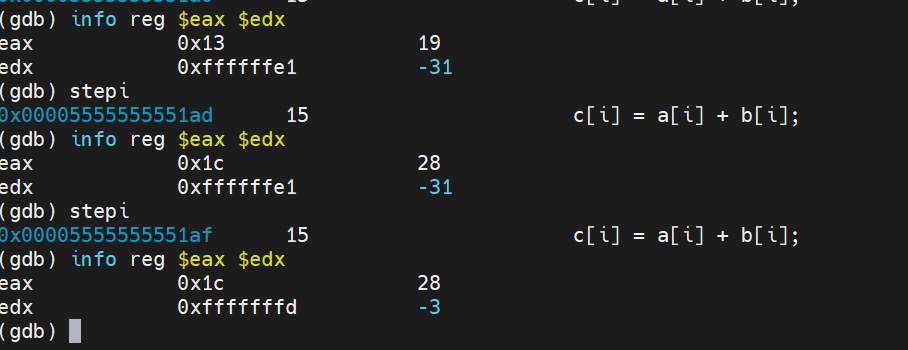


3、

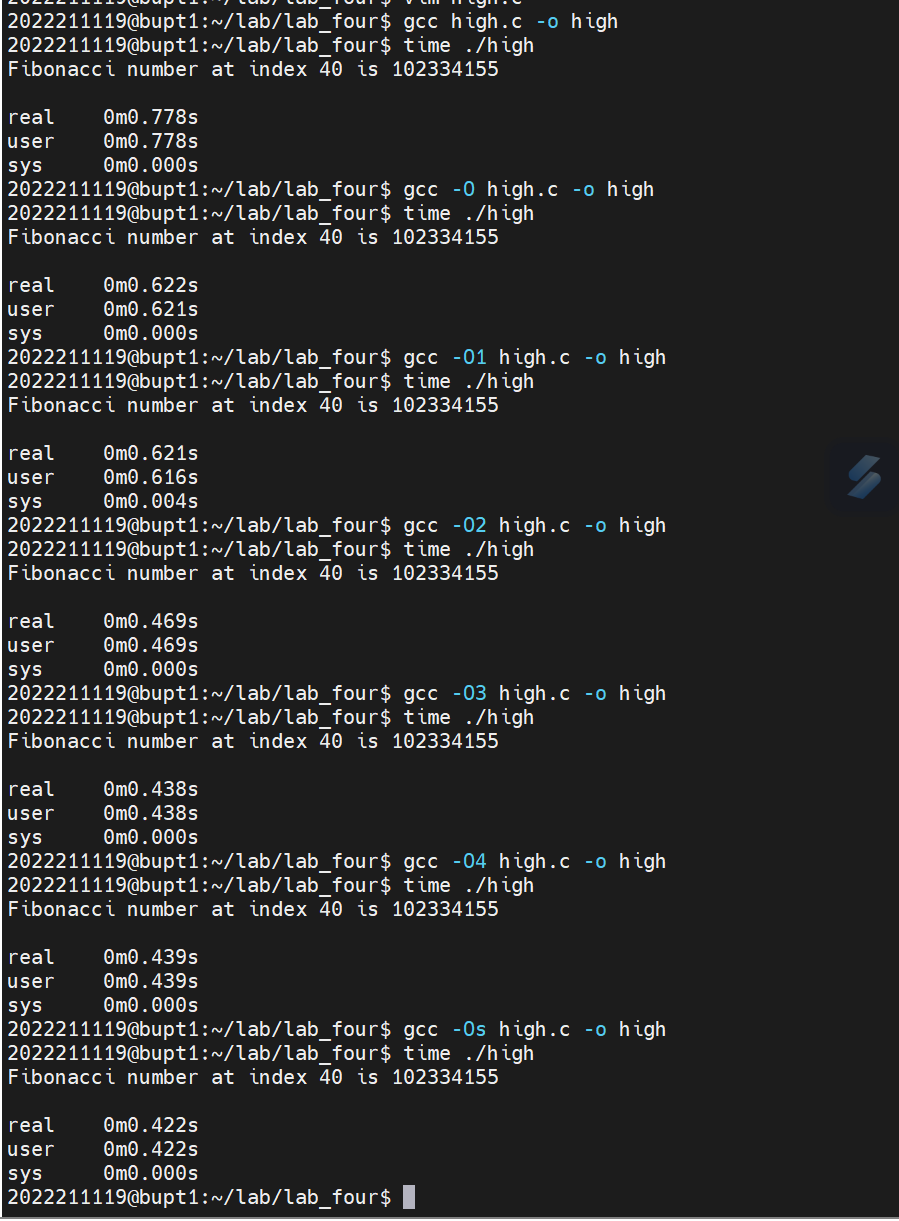


**a[i]+b[i]位于eax寄存器和edx寄存器中**

4、



**实验四**



本算法为计算斐波那契数列的高计算量递归算法，通过设置不同优化参数，可以发现算法的执行效率大大提高，其中O1优化效果和Os优化效果不太明显，因为O1优化级别不够高，优化不足，Os最后可再次优化的范围有限。

五、实验分析（20分）

1、通过MobaXterm远程连接Linux操，通过ppt学习Linux操作系统的基本使用方法，逐个完成上述试验任务，按照指导编写和调试程序，并使用objdump的反汇编功能来了解汇编语言指令以及程序的执行流程。

2、在完成实验任务后熟悉 Linux 下的编程环境，包括代码编写、编译和运行，同时能够深入了解程序的汇编代码，对代码分析有更多的了解。

六、实验总结（5分）

总结心得（包括遇到的困难，自己一些不成功的设计和设想）

1. 进行实验二操作时。需要声明的二维数组太大，会造成栈溢出，在主函数里面无法操作，会发生段错误，设想使用malloc进行申请内存，设计失败后与唐浦瑞同学进行讨论后创建了一个全局数组，成功运行，了解后发现全局变量是存在静态内存中的，需要时再调用，所以不会栈溢出。
2. 刚开始时不清楚stepi与step的区别，对这两个指令使用比较随意，导致不能达到想要的代码片段。

七、诚信声明（不签扣10分）

需要填写如下声明，并在底部给出手写签名的电子版。

在完成本次实验过程中，我曾分别与以下各位同学就以下方面做过交流：

1、简单描述交流内容，例如：来自唐浦瑞的建议，采用申请全局变量的方式声明一个大型二维数组。

此外，我还参考了以下资料：

1. 网址

<https://blog.csdn.net/Aliven888/article/details/118888960>

了解不同编译优化参数间的联系和区别。

在我提交的程序中，还在对应的位置以注释形式记录了具体的参考内容。

我独立完成了本次实验除以上方面之外的所有工作，包括分析、设计、编码、调试与测试。

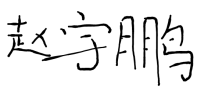
我清楚地知道，从以上方面获得的信息在一定程度上降低了实验的难度，可能影响起评分。

我从未使用他人代码，不管是原封不动地复制，还是经过某些等价转换。

我未曾也不会向同一课程（包括此后各届）的同学复制或公开我这份程序的代码，我有义务妥善保管好它们。

我编写这个程序无意于破坏或妨碍任何计算机系统的正常运行。

我清楚地知道，以上情况均为本课程纪律所禁止，若违反，对应的实验成绩将按照0分计。



（签名）