

**实验报告**



**题目： Linux系统及其相关软件环境**

**班 级： 2023211311**

**学 号： 2023211198**

**姓 名： 杜昊阳**

**学 院： 计算机学院**

**2022年 10 月 12 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. MobaXterm Personal Edition v22.1（服务器：10.120.11.12）
2. Linux
3. Gcc编译器
4. GDB调试工具
5. Objdump命令反汇编

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;

}

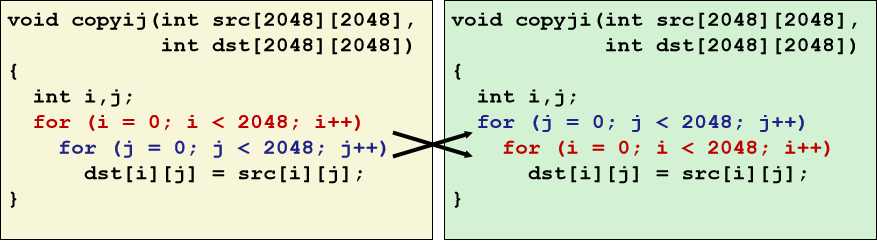
printf(Result is%lf\\n,result);

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）

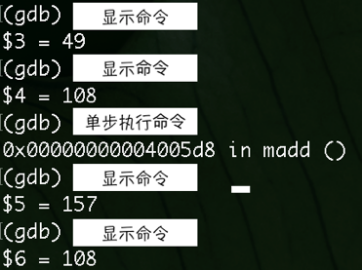
现有两个int型数组a[i]=i-50，b[i]=i+y，其中y取自于学生本人学号2022211x\*y的个位。登录bupt1服务器，在linux环境下使用vi编辑器编写C语言源程序，完成数组a+b的功能，规定数组长度为100，函数名为madd（），数组a，b均定义在函数内，采用gcc编译该程序（使用-g -fno-pie -fno-stack-protector选项），

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；
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）：



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

四、实验步骤（60-80分）

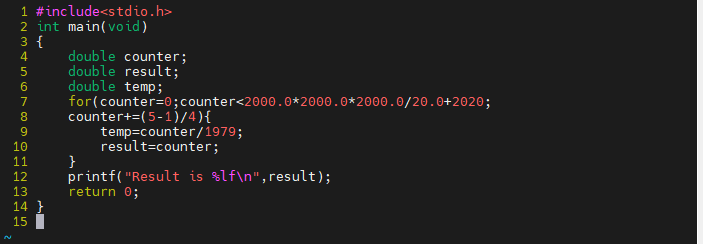
实验内容一（15分）

1.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）

（1）创建文件exp1.c



（2）进行编辑代码（已输入修改后的代码）



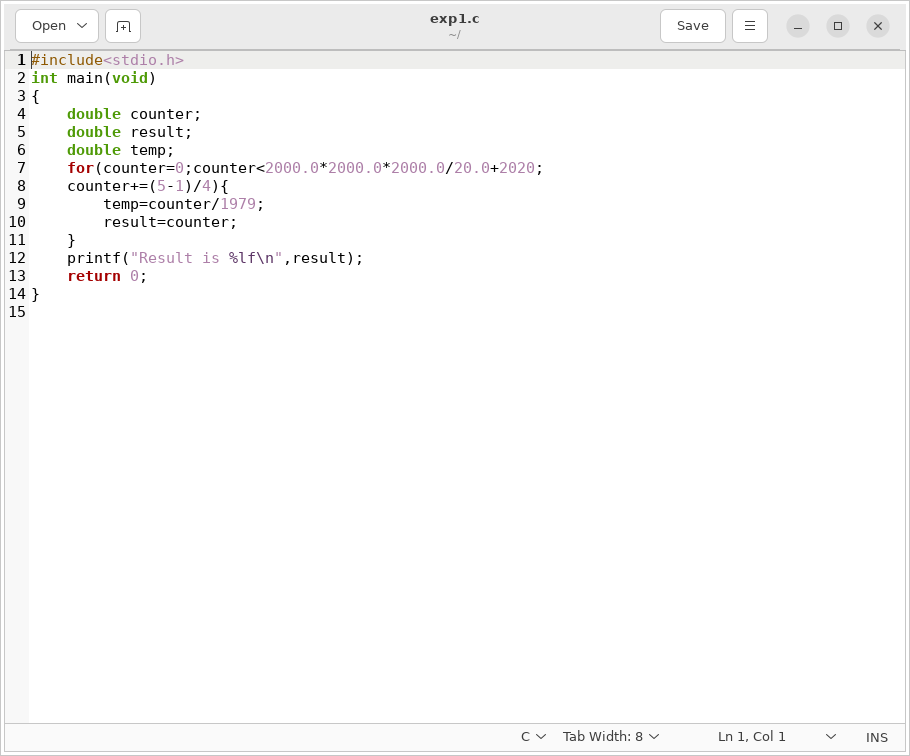
（3）Esc，保存退出



（4）打开exp1.c文件



（5）进行修改（实际并未进行修改）



（6）save保存

2.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）

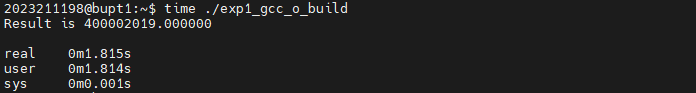
（1）采用-o0参数编译exp1.c，生成exp1\_gcc\_o\_build文件



（2）执行exp1\_gcc\_o\_build文件



（3）观察-o0参数下exp1\_gcc\_o\_build文件的执行时间



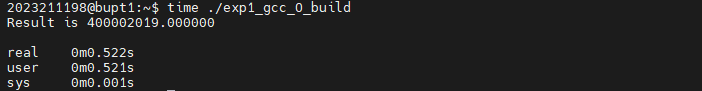
（4）采用-O参数编译exp1.c，生成exp1\_gcc\_O\_build文件



（5）执行exp1\_gcc\_O\_build文件



（6）观察-O参数下exp1\_gcc\_O\_build文件的执行时间



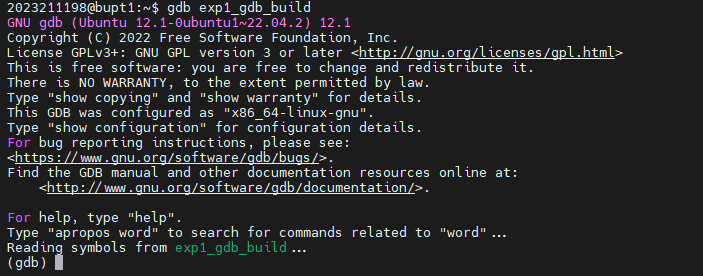
（7）经比对，-O参数下exp1.c执行速度快于-o参数下exp1.c执行速度

3.采用gdb进行调试，让程序运行到for函数语句（调试截图）

（1）编译exp1.c



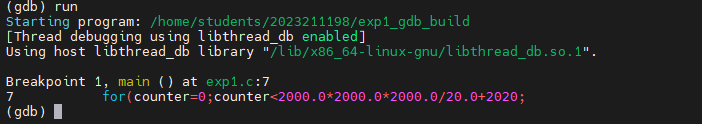
（2）启动gdb



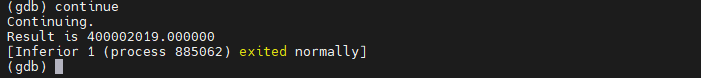
（3）经查，for函数语句在第7行，所以在line7打断点



（4）执行程序到断点



（5）继续执行，直至结束



（6）退出gdb

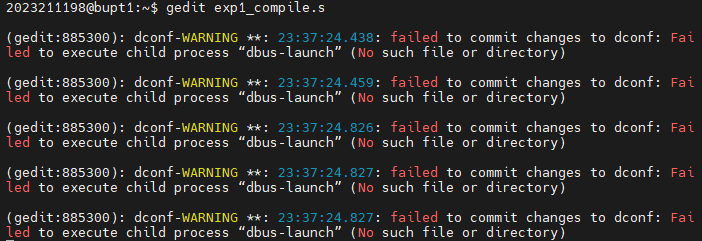


4．运用objdump工具生成汇编程序（给出main函数的汇编程序截图）

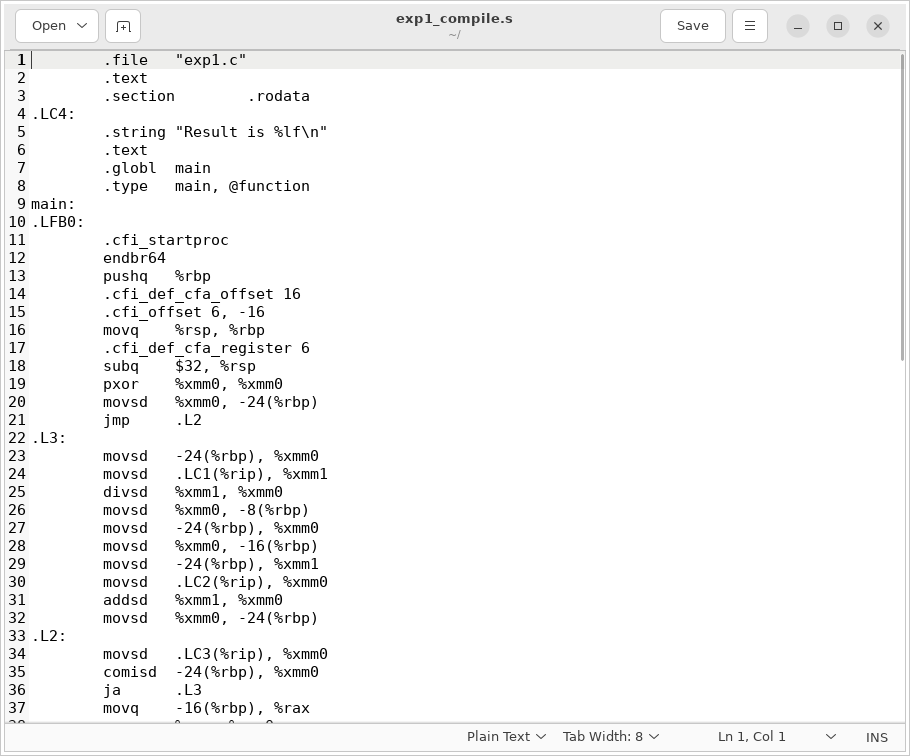
（1）生成exp1.c的汇编文件exp1\_compile.s



（2）打开exp1\_compile.s

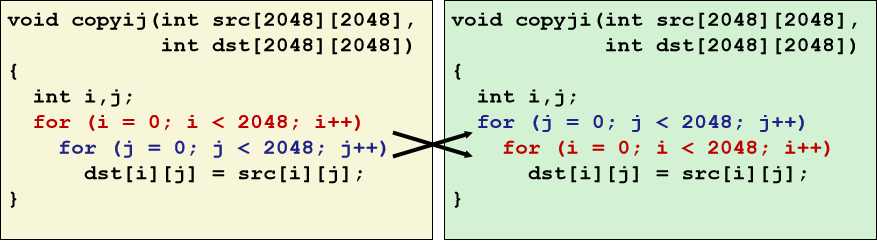


（3）弹出汇编文件内容



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

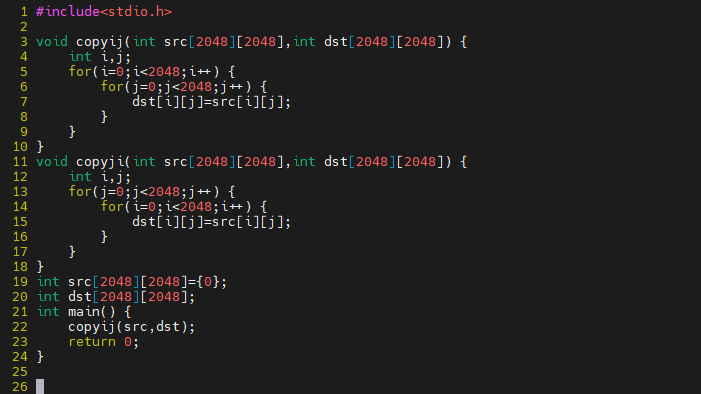


分别设置不同优化参数，给出运行时间

（1）创建exp2\_ij.c文件



（2）编写exp2\_ij.c代码



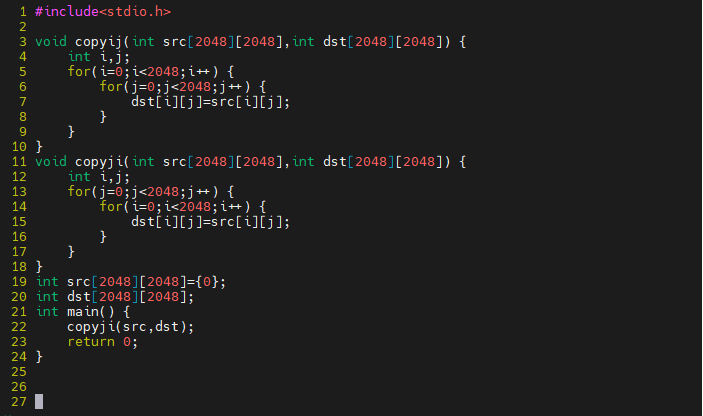
（3）Esc，输入:wq保存退出



（4）创建exp2\_ji.c文件



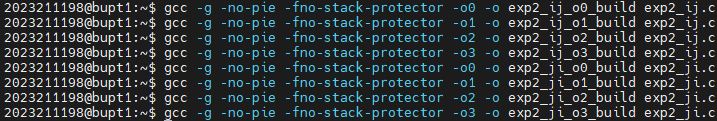
（5）编写exp2\_ji.c代码



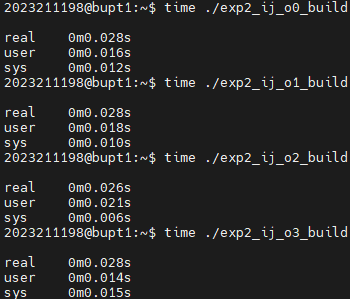
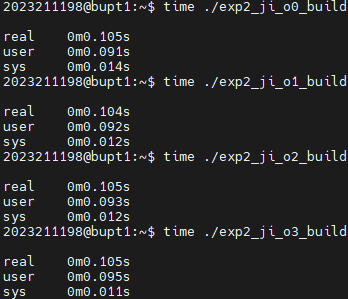
（6）Esc，输入:wq保存退出



（7）分别以o0、o1、o2、、o3参数对exp2\_ij.c和exp2\_ji.c进行编译



（8）观察两组文件在4个参数下执行时间

实验内容三（30分）

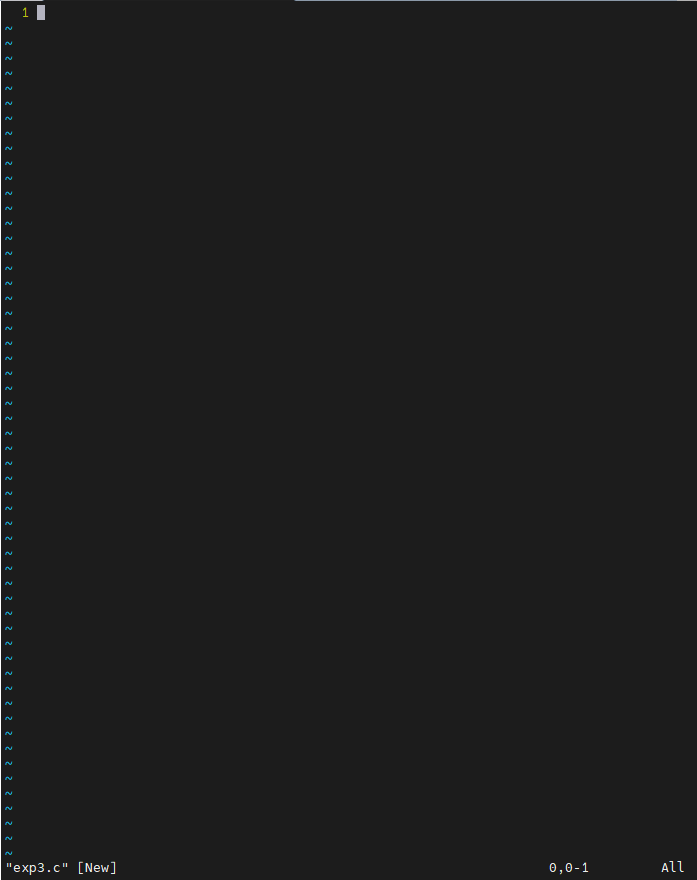
现有两个int型数组a[i]=i-50，b[i]=i+y，其中y取自于学生本人学号2022211x\*y的个位。登录bupt1服务器，在linux环境下使用vi编辑器编写C语言源程序，完成数组a+b的功能，规定数组长度为100，函数名为madd（），数组a，b均定义在函数内，采用gcc编译该程序（使用-g -fno-pie -fno-stack-protector选项），

（1）创建文件exp3.c

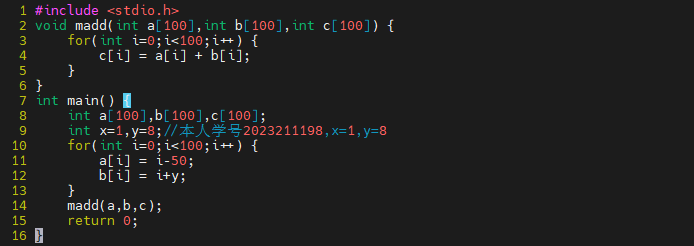


（2）在vi编辑器中编写代码

编写前：



编写后：



（3）Esc，输入:wq保存退出



（4）编译exp3.c文件，生成exp3\_build文件

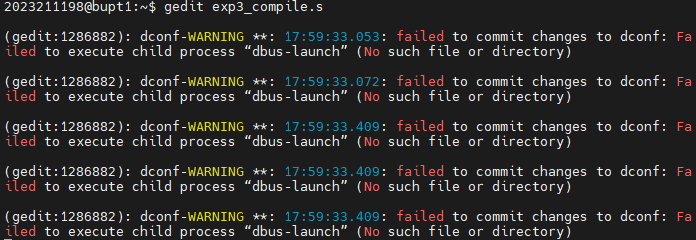


1. 使用objdump工具生成汇编程序，找到madd函数的汇编程序，给出截图；

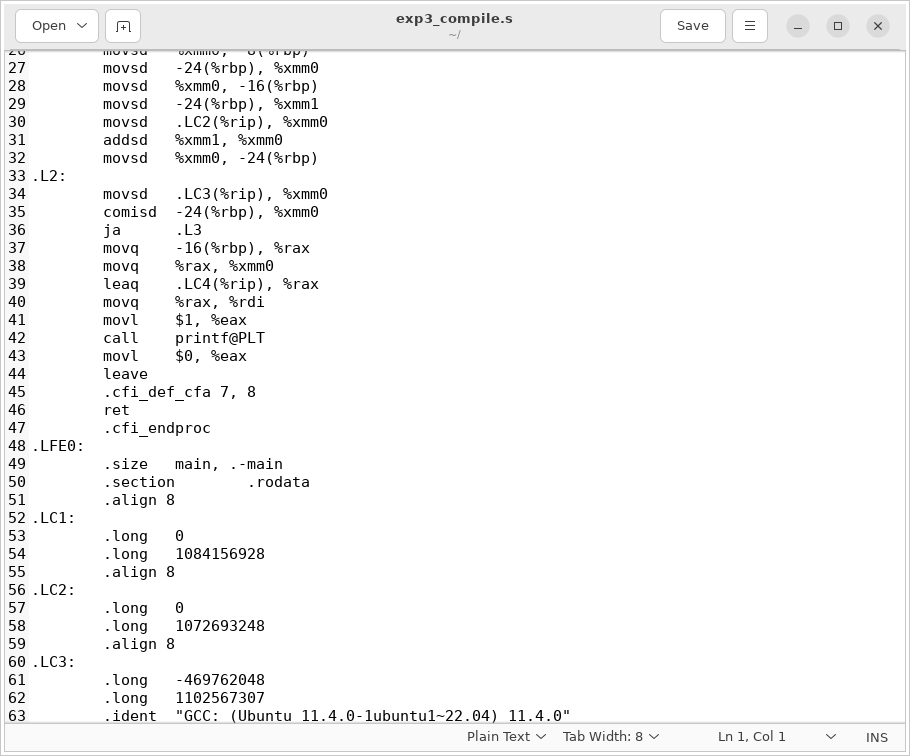
（1）生成exp3.c的汇编文件exp3\_compile.s



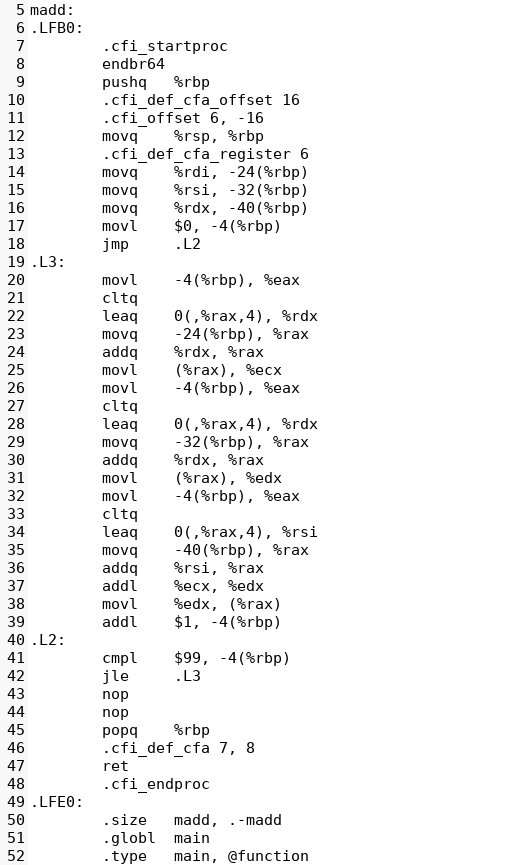
（2）打开exp3\_compile.s



（3）弹出汇编文件内容



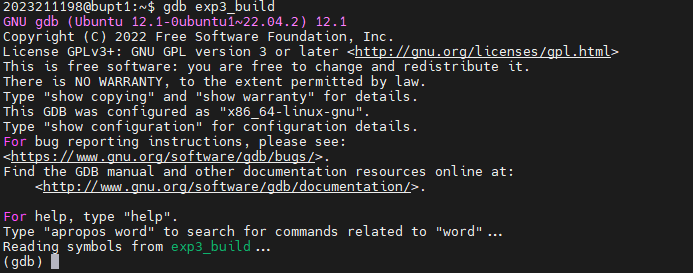
（4）找到madd函数的汇编程序



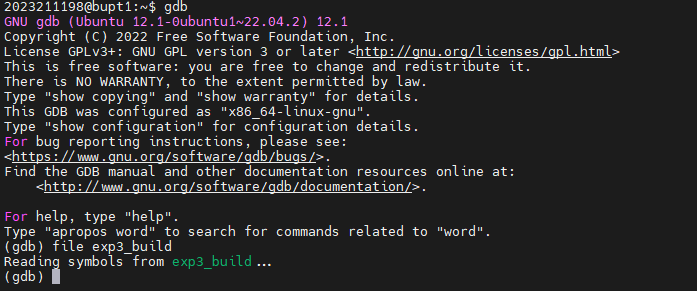
1. 用gdb进行调试，练习下列gdb命令，给出截图；

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

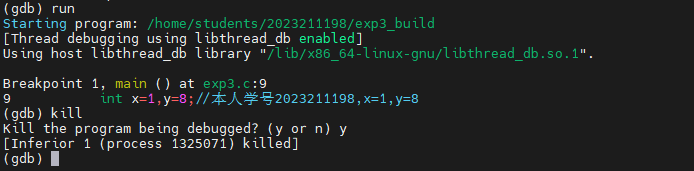
1. gdb



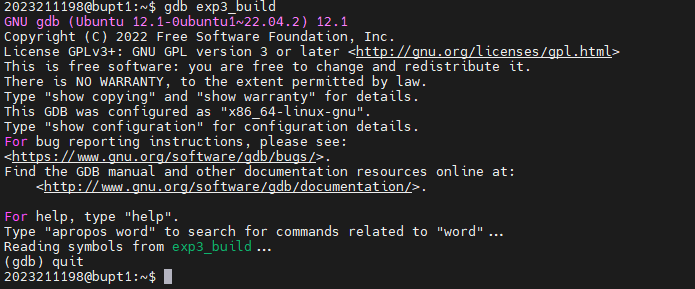
1. file



1. kill



1. quit



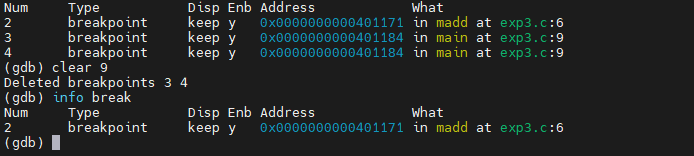
1. break



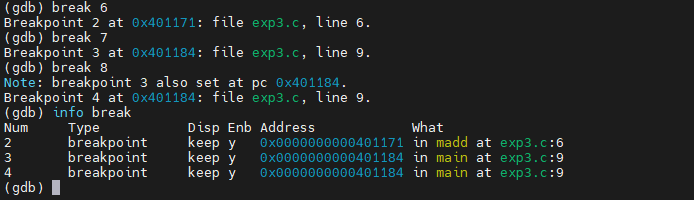
1. delete



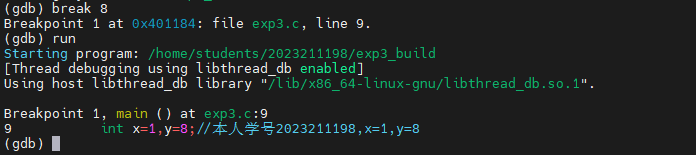
1. clear



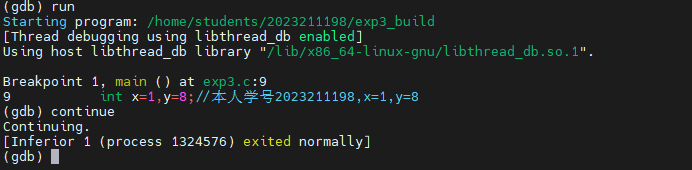
1. info break



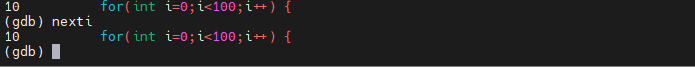
1. run



1. continue



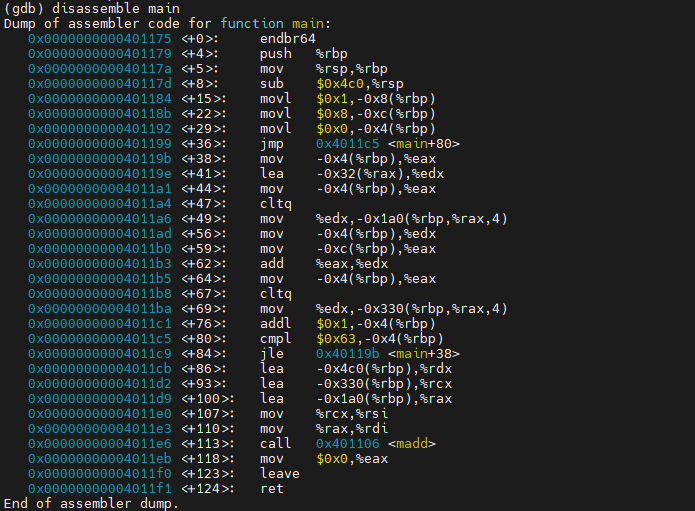
1. nexti



1. stepi



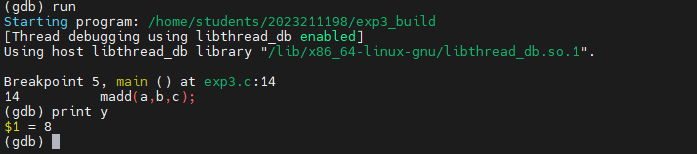
1. disassemble



1. list



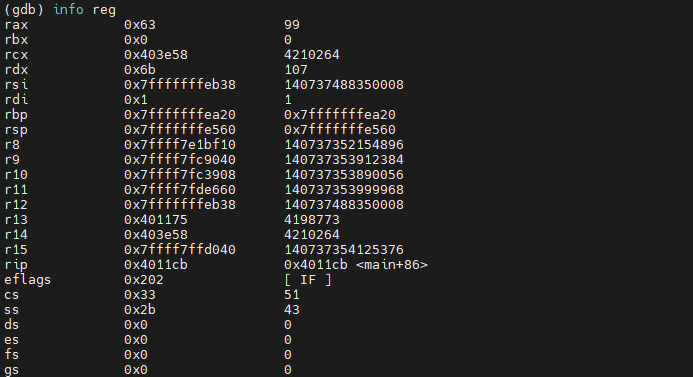
1. print



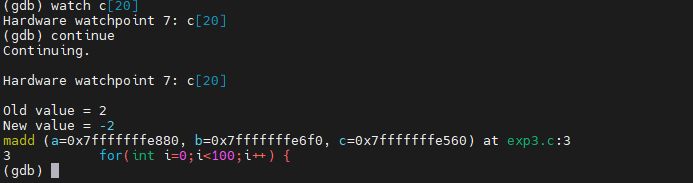
1. x



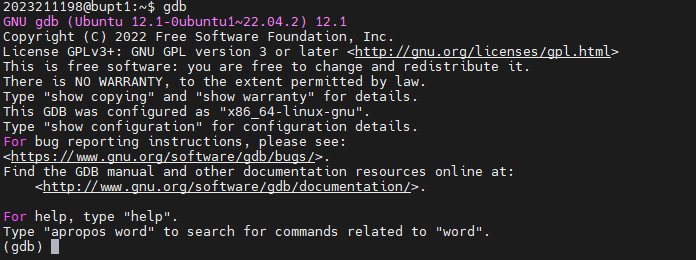
1. info reg



1. 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. 启动gdb



（2）载入exp3\_build文件



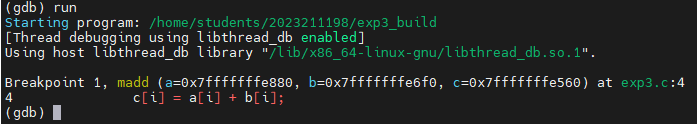
（3）清除所有断点，将文件初始化



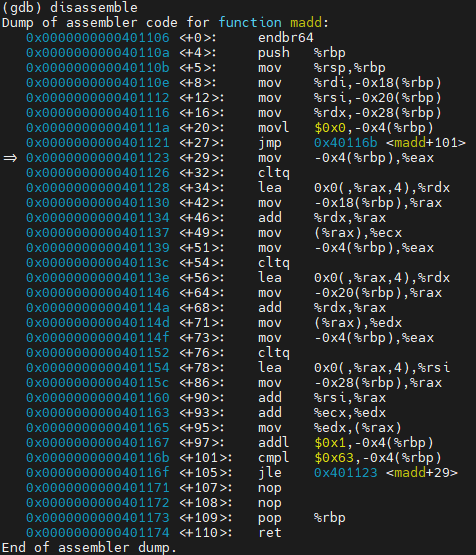
（4）在第四行（c[i] = a[i] + b[i];）加入断点



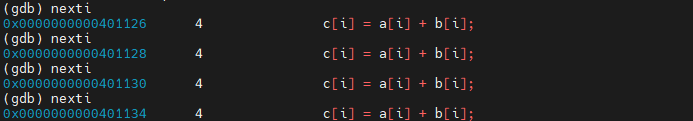
（5）执行到断点



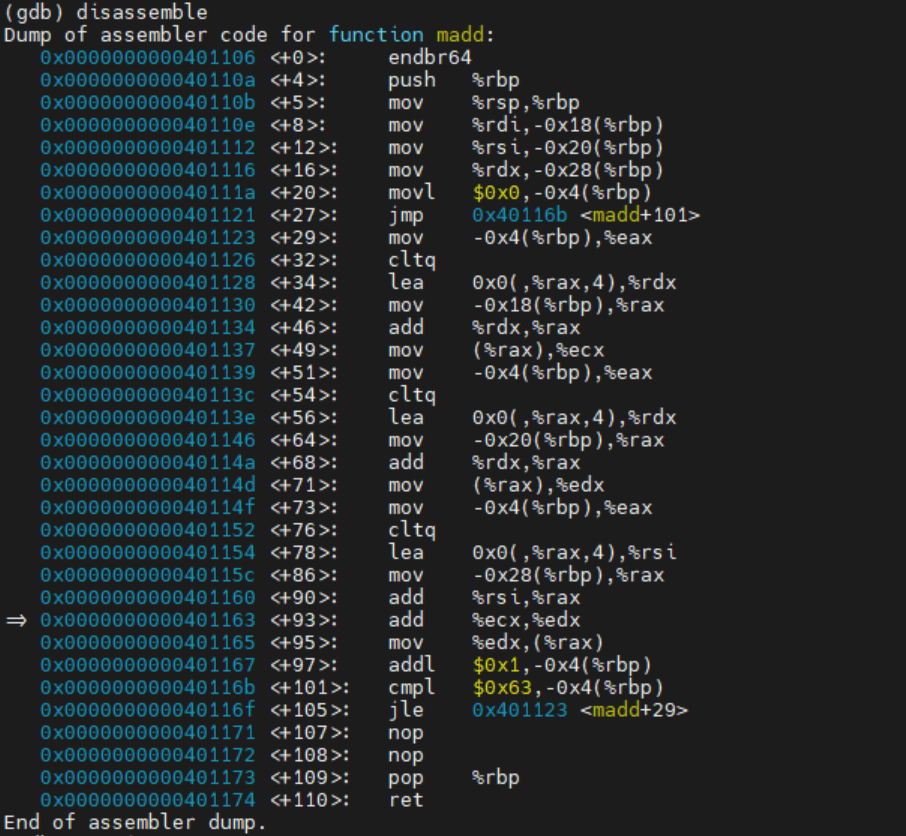
（6）使用disassemble指令，查看当前汇编指令



（7）多次使用nexti指令，执行到a[i] + b[i]



（8）使用disassemble指令，查看当前汇编指令



可知a[i] + b[i]对应add %ecx,%edx

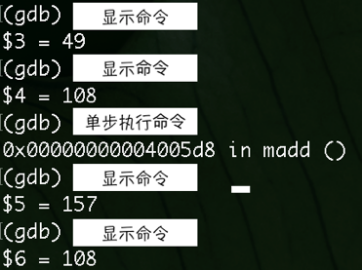
（9）通过print指令查看寄存器内的值



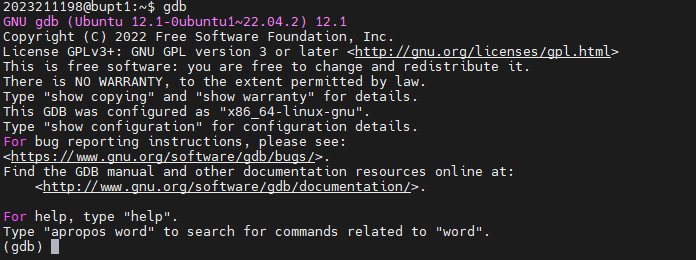
已知a[0]=-50,b[0]=-42,可知a[i]在%ecx，b[i]在%edx

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）：



（1）启动gdb



（2）载入exp3\_build文件



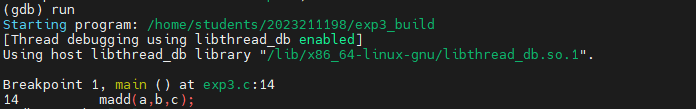
（3）清除所有断点，将文件初始化



（4）在第13行标上断点



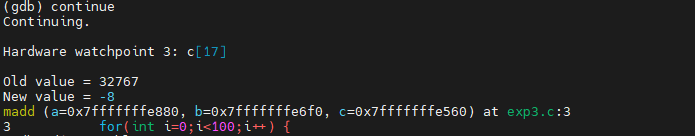
（5）开始执行程序到断点



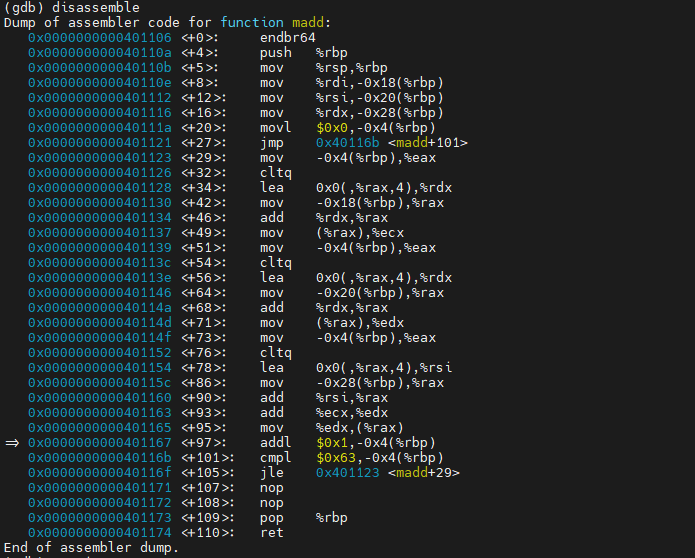
（6）添加对c[17]的监视



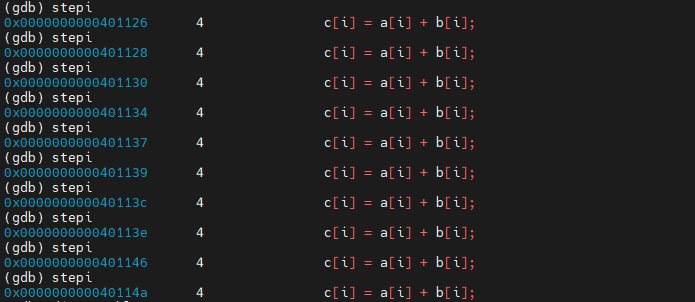
（7）继续执行到监视变化

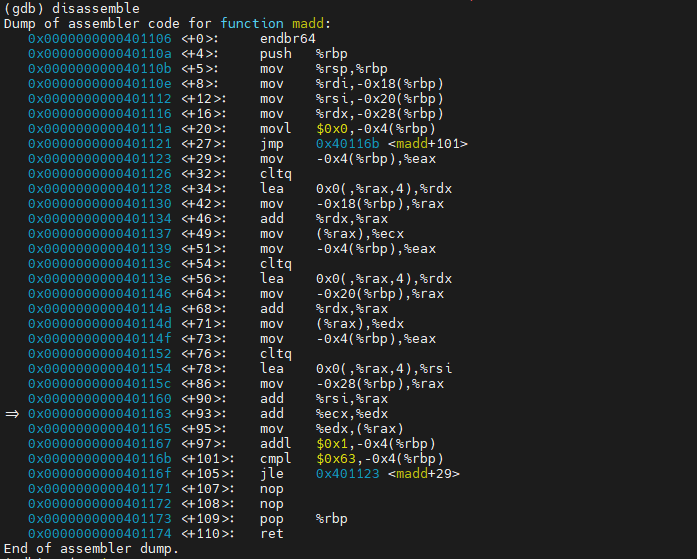


（8）查看当前汇编指令



（9）通过stepi逐步执行，直至a[18]+b[18]之前(<+93>)





（10）分别以十进制和十六进制查看相加之前ecx、edx的值



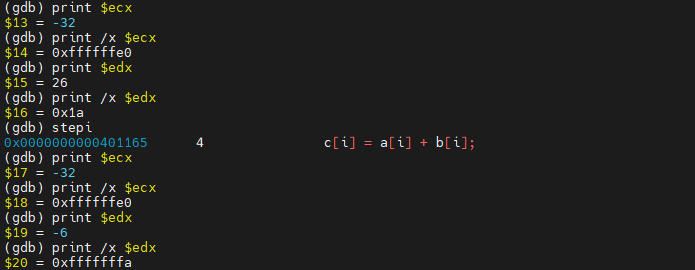
（11）单步执行



（12）分别以十进制和十六进制查看相加之后ecx、edx的值



（13）单步执行完整截图



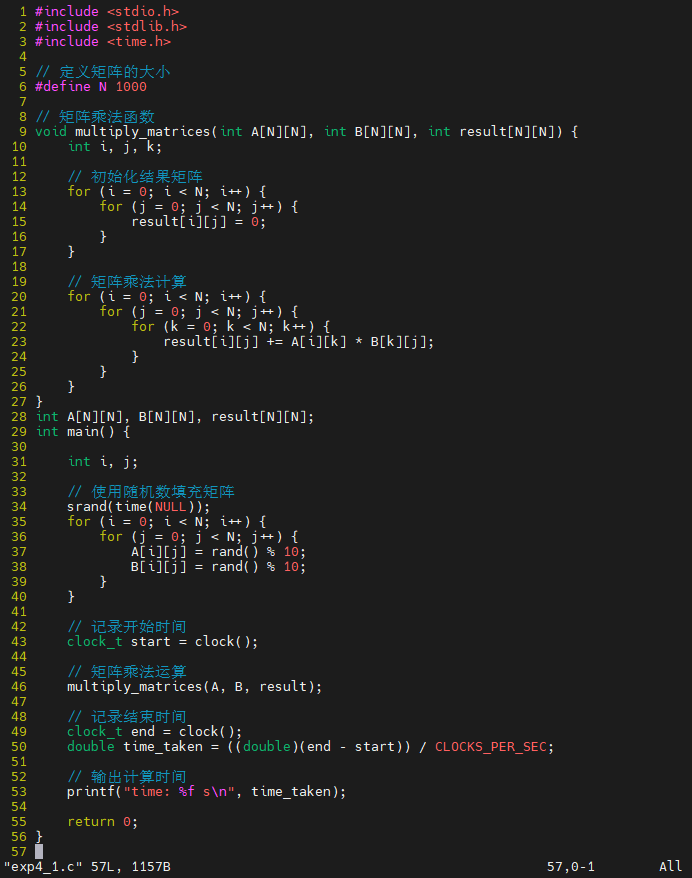
实验内容四（加分项，20分）

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

（1）创建exp4\_1.c文件



（2）编写exp4\_1.c代码



（3）Esc，输入:wq保存退出



（4）创建exp4\_2.c文件



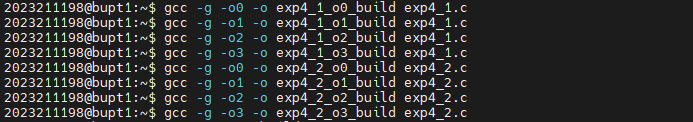
（5）编写exp4\_2.c代码



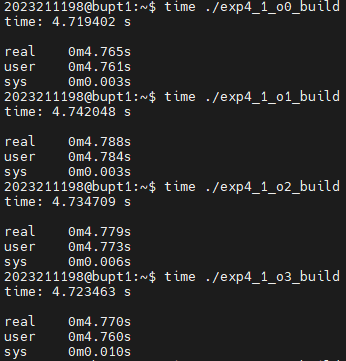
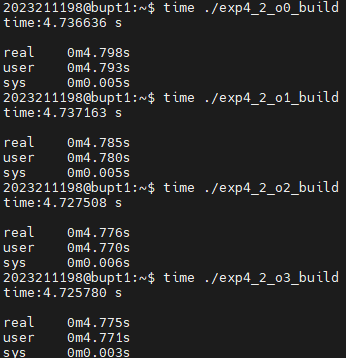
（6）Esc，输入:wq保存退出



（7）分别以o0、o1、o2、、o3参数对exp4\_1.c和exp4\_2.c进行编译



（8）观察两组文件在4个参数下执行时间

五、实验分析（20分）

实验时的工作思路、设想、效果等综合分析

工作思路：

每一步流程为：

（1）先浏览题目

（2）查询对应资料（工具书，网站）

（3）使用伪代码或文字等方式，记录解题思路

（4）在自己的Linux系统中测试执行情况，部分代码先在Windows系统中测试

（5）如果遇到问题或错误，通过交流，查书，上网查错误代码等方式修正错误，返回第3步

（6）在远程系统上完成相应步骤

（7）截图，填写实验报告

（8）核对远程系统文件、截图、实验报告，确保填写无误

设想：

（1）通过本实验熟悉Linux系统基本使用方法，建立自己的Linux系统；

（2）成功在Linux系统下编写C语言程序并编译和执行；

（3）了解汇编程序、汇编指令等

效果：

基本达到目标，熟悉Linux系统基本使用方法，建立自己的Linux系统；成功在Linux系统下编写C语言程序并编译和执行；了解汇编程序、汇编指令等。

但仍需加强对指令的记忆

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.在Linux环境中，不会使用vi编辑器，写完代码不会保存退出，通过上网查询vi编辑器使用方法得以解决。

2.编译时没有带-g，使得编译程序无法被gdb执行，通过删除文件再次编译解决

3.在gdb中不会设置断点，不会逐行执行，通过查资料学会

4.不会阅读汇编程序，通过查阅资料学会

5.实验内容二，int[2048][2048]溢出，自己设想采用malloc动态数组仍报错，后来通过与同学交流决定采用全局变量的方式

6.不了解汇编指令，通过逐行执行，观察效果的方式找到每个指令的作用

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、来自富思涵(311-2023211218)的建议，实验内容二采用全局变量的方式完成执行代码

此外，我还参考了以下资料：

1. ChatGPT(<https://chatgpt.com>)

用于生成实验内容四中“高计算量类型算法”和“高内存需求类型算法”。

2、CSDN(<https://blog.csdn.net/>)

(1) 【Linux C】GCC编译 && GDB调试 从入门到放弃 （gcc调试选项详解、gdb调试、条件断点、远程调试、脚本化调试）(<https://blog.csdn.net/weixin_43764974/article/details/130787032>)

用于学习gcc和gdb调试指令

(2) gcc 中-O -O1 -O2 -O3 -Os -Ofast -Og优化的原理(<https://blog.csdn.net/Windgs_YF/article/details/106567011>)

用于研究不同优化参数的作用

(3) 汇编基础-----mov与常见寻址方式(<https://blog.csdn.net/a1309525802/article/details/137461445>)

用于研究汇编指令

3、Bilibili(<https://www.bilibili.com>)

速学150个Linux常用命令(<https://www.bilibili.com/video/BV12L411a7Ne/?spm_id_from=333.337.search-card.all.click>)

用于学习Linux常用指令

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）