

**实验报告**

**实 验（二）**

题 目 DataLab 数据表示

专 业 计算机大类

学　　 号 1191000606

班　　 级 1903003

学 生 陈一帆

指 导 教 师 郑贵滨

实 验 地 点 G709

实 验 日 期 2021.4.2

**计算机科学与技术学院**

**目 录**

[第1章 实验基本信息 - 4 -](#_Toc20402368)

[1.1 实验目的 - 4 -](#_Toc20402369)

[1.2 实验环境与工具 - 4 -](#_Toc20402370)

[1.2.1 硬件环境 - 4 -](#_Toc20402371)

[1.2.2 软件环境 - 4 -](#_Toc20402372)

[1.2.3 开发工具 - 4 -](#_Toc20402373)

[1.3 实验预习 - 4 -](#_Toc20402374)

[第2章 实验环境建立 - 6 -](#_Toc20402375)

[2.1 Ubuntu下CodeBlocks安装 - 6 -](#_Toc20402376)

[2.2 64位Ubuntu下32位运行环境建立 - 7 -](#_Toc20402377)

[第3章 C语言的数据类型与存储 - 8 -](#_Toc20402378)

[3.1 类型本质（1分） - 8 -](#_Toc20402379)

[3.2 数据的位置-地址（2分） - 8 -](#_Toc20402380)

[3.3 main的参数分析（2分） - 11 -](#_Toc20402381)

[3.4 指针与字符串的区别（2分） - 13 -](#_Toc20402382)

[第4章 深入分析UTF-8编码 - 15 -](#_Toc20402383)

[4.1 提交utf8len.c子程序 - 15 -](#_Toc20402384)

[4.2 C语言的strcmp函数分析 - 15 -](#_Toc20402385)

[4.3讨论：按照姓氏笔画排序的方法实现 - 16 -](#_Toc20402386)

[第5章 数据变换与输入输出 - 17 -](#_Toc20402387)

[5.1 提交](#_Toc20402388)**[cs\_atoi.c](#_Toc20402388)** [- 17 -](#_Toc20402388)

[5.2 提交](#_Toc20402389)**[cs\_atof.c](#_Toc20402389)** [- 17 -](#_Toc20402389)

[5.3 提交](#_Toc20402390)**[cs\_itoa.c](#_Toc20402390)** [- 18 -](#_Toc20402390)

[5.4 提交](#_Toc20402391)**[cs\_ftoa.c](#_Toc20402391)** [- 18 -](#_Toc20402391)

[5.5 讨论分析OS的函数对输入输出的数据有类型要求吗 - 19 -](#_Toc20402392)

[第6章 整数表示与运算 - 20 -](#_Toc20402393)

[6.1 提交fib\_dg](#_Toc20402394)**[.c](#_Toc20402394)** [- 20 -](#_Toc20402394)

[6.2 提交](#_Toc20402395)**[fib\_loop.c](#_Toc20402395)** [- 20 -](#_Toc20402395)

[6.3 fib溢出验证 - 21 -](#_Toc20402396)

[6.4 除以0验证： - 21 -](#_Toc20402397)

[第7章 浮点数据的表示与运算 - 23 -](#_Toc20402398)

[7.1 正数表示范围 - 23 -](#_Toc20402399)

[7.2浮点数的编码计算 - 23 -](#_Toc20402400)

[7.3特殊浮点数值的编码 - 24 -](#_Toc20402401)

[7.4浮点数除0 - 25 -](#_Toc20402402)

[7.5 Float的微观与宏观世界 - 25 -](#_Toc20402403)

[7.6 讨论：任意两个浮点数的大小比较 - 25 -](#_Toc20402404)

[第8章 舍尾平衡的讨论 - 27 -](#_Toc20402405)

[8.1 描述可能出现的问题 - 27 -](#_Toc20402406)

[8.2 给出完美的解决方案 - 27 -](#_Toc20402407)

[第9章 总结 - 29 -](#_Toc20402408)

[9.1 请总结本次实验的收获 - 29 -](#_Toc20402409)

[9.2 请给出对本次实验内容的建议 - 29 -](#_Toc20402410)

[参考文献 - 30 -](#_Toc20402411)

# 第1章 实验基本信息

## 1.1 实验目的

熟练掌握计算机系统的数据表示与数据运算

通过C程序深入理解计算机运算器的底层实现与优化

掌握VS/CB/GCC等工具的使用技巧与注意事项

## 1.2 实验环境与工具

### 1.2.1 硬件环境

X64CPU；1.6GHz；8G RAM；512G SSD

### 1.2.2 软件环境

Windows10 64位；Ubuntu 18.04.5 LTS 64位

### 1.2.3 开发工具

Clion2020.3；Vscode；gedit+gcc；

## 1.3 实验预习

上实验课前，必须认真预习实验指导书（PPT或PDF）

了解实验的目的、实验环境与软硬件工具、实验操作步骤，复习与实验有关的理论知识。

采用sizeof在Windows的VS/CB以及Linux的CB/GCC下获得C语言每一类型(Char /short int/int/long/float/double/long long/long double/指针)在32/64位模式下的空间大小

编写C程序，计算斐波那契数列在int/long/unsigned int/unsigned long类型时，n为多少时会出错；用递归程序实现，会出现什么问题？再用循环方式实现。

写出float/double类型最小的正数、最大的正数（非无穷）

按步骤写出float数-1.1在内存从低到高地址的字节值-16进制

按照阶码区域写出float的最大密度区域范围及其密度，最小密度区域及其密度（区域长度/表示的浮点个数）

# 第2章 实验环境建立

## 2.1 Ubuntu下Clion安装

Clion运行界面截图：编译、运行hellolinux.c

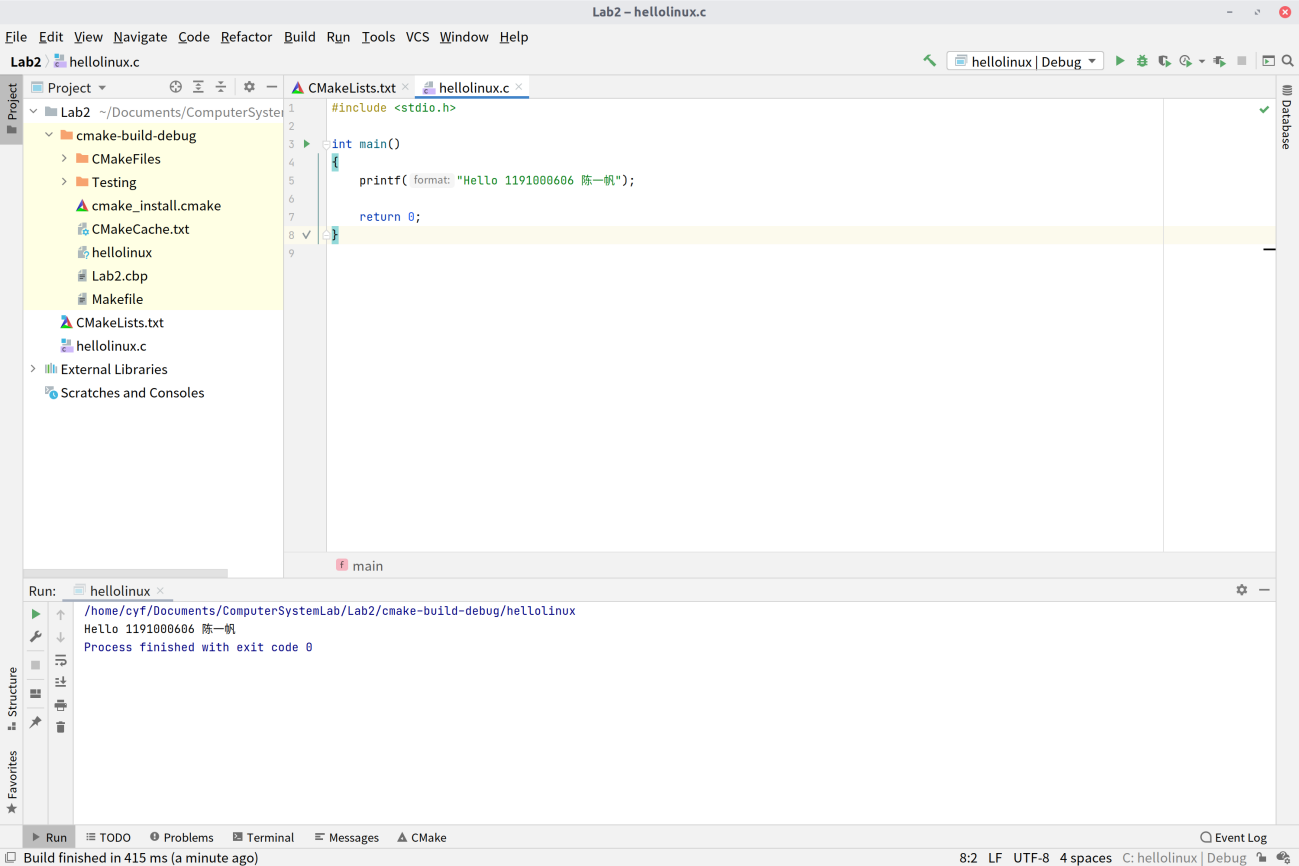


图2.1 Clion运行截图

## 2.2 64位Ubuntu下32位运行环境建立

在终端下，用gcc的32位模式编译生成hellolinux.c。执行此文件。

Linux及终端的截图。

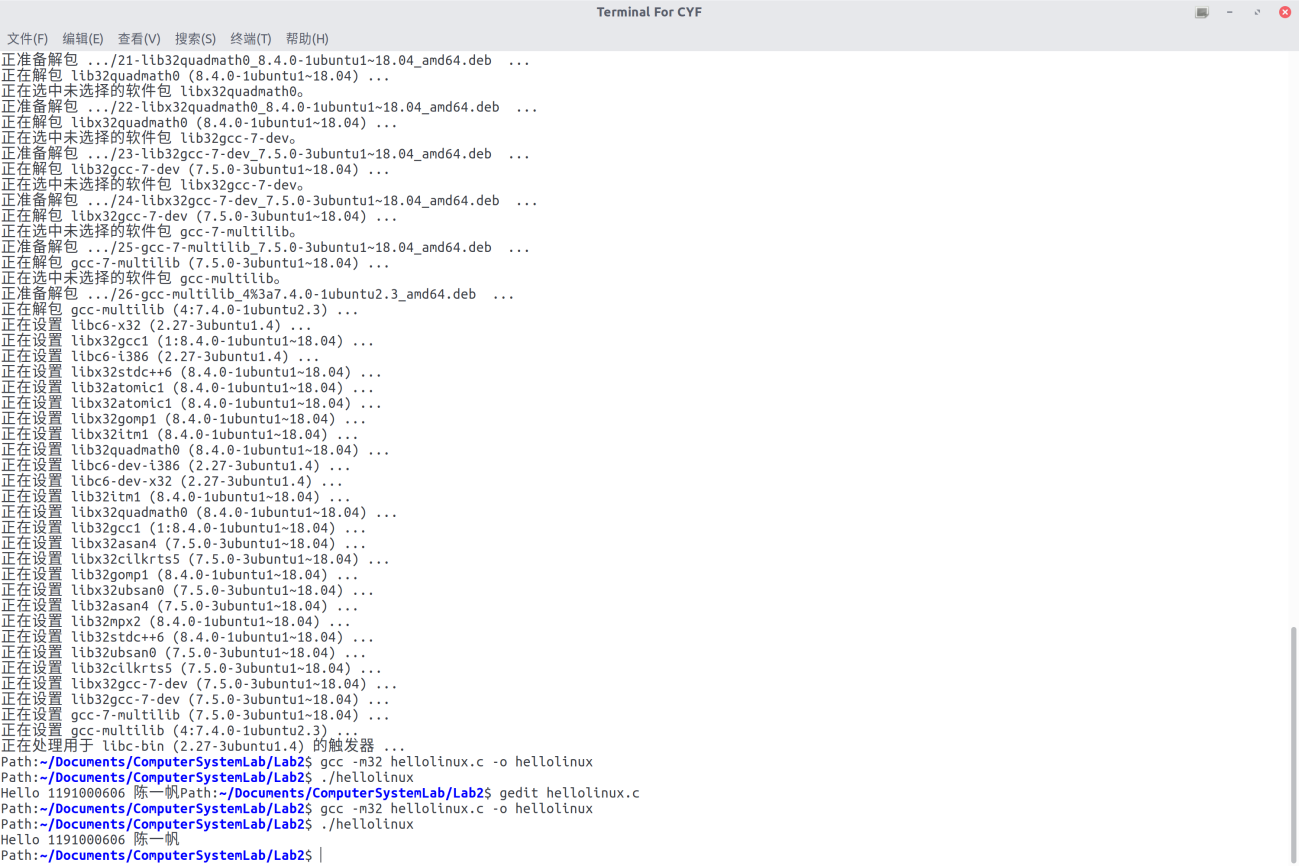


图2.2 Ubuntu 32运行

# 第3章 C语言的数据类型与存储

## 3.1 类型本质

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Win/VS/x86** | **Win/VS/x64** | **Win/CL/32** | **Win/CL/64** | **Linux/CL/32** | **Linux/CL/64** |
| char | 1 | 1 | 1 | 1 | 1 | 1 |
| short | 2 | 2 | 2 | 2 | 2 | 2 |
| int | 4 | 4 | 4 | 4 | 4 | 4 |
| long | 4 | 4 | 4 | 4 | 4 | 8 |
| long long | 8 | 8 | 8 | 8 | 8 | 8 |
| float | 4 | 4 | 4 | 4 | 4 | 4 |
| double | 8 | 8 | 8 | 8 | 8 | 8 |
| long double | 8 | 8 | 12 | 16 | 12 | 16 |
| 指针 | 4 | 8 | 4 | 8 | 4 | 8 |

C编译器对sizeof的实现方式：sizeof不是函数，也不是操作符。因为在编译运行时它没有被编译成机器指令，可以把它看做成是一个宏。

## 3.2 数据的位置-地址

打印x、y、z输出的值：如图3-2

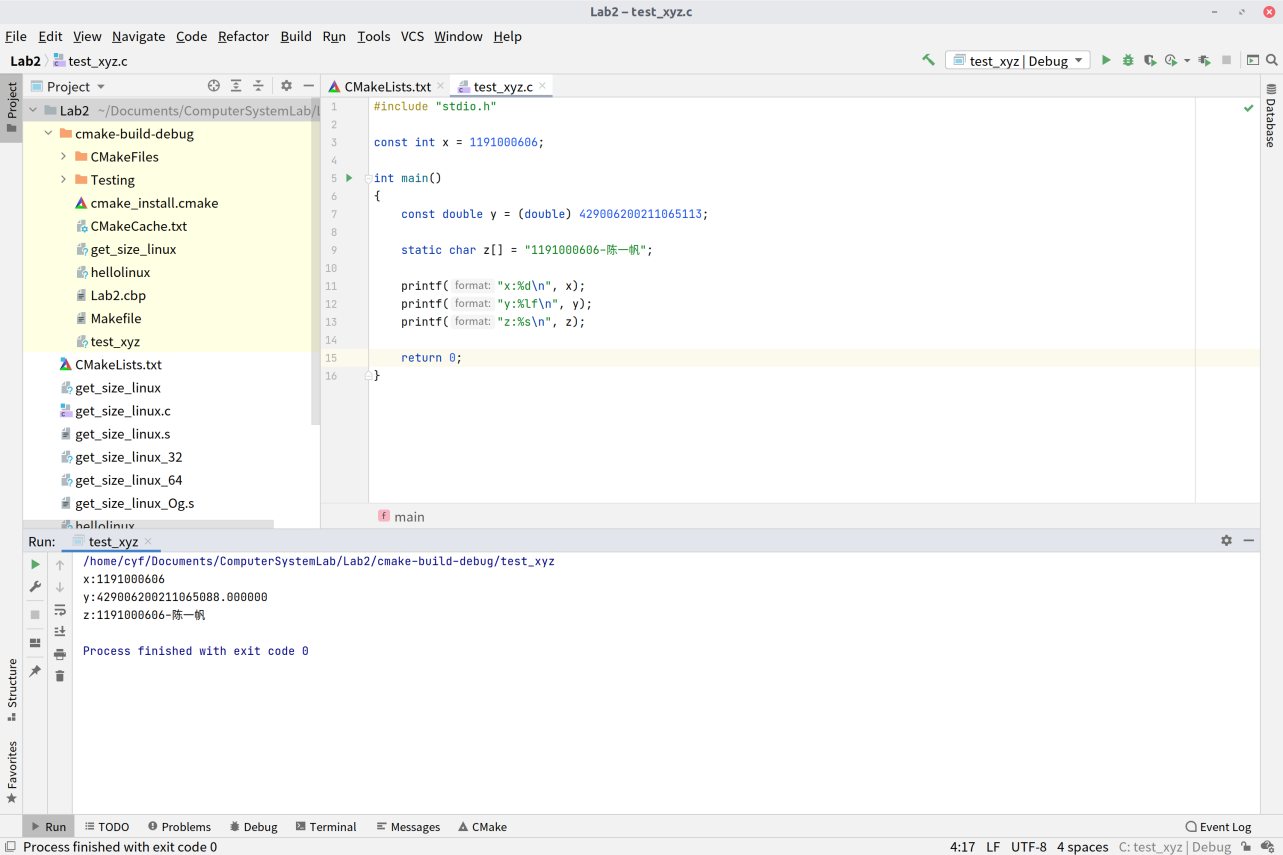


图3.2.1 x，y，z输出的值

反汇编查看x、y、z的地址，每字节的内容，如下图

反汇编查看x、y、z在代码段的表示形式。截图3，标注说明

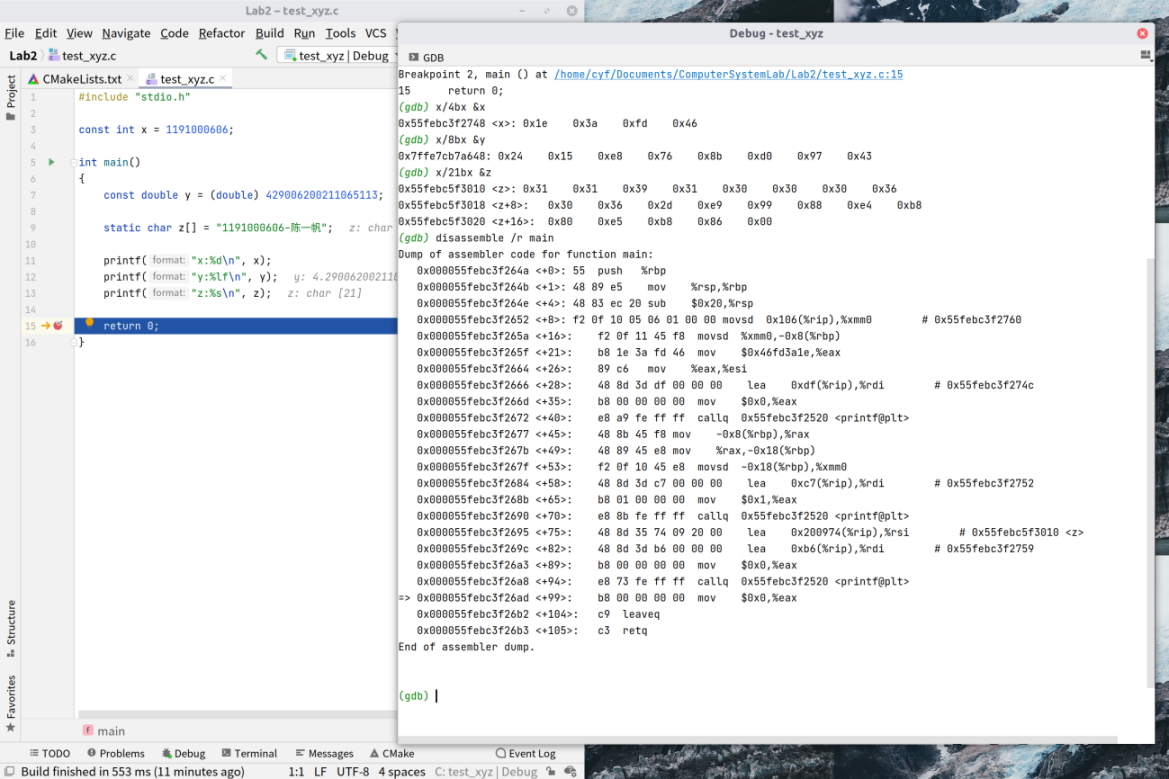


图3.2.2 x，y，z地址，字节值

运行时，采用gdb调试，地址，字节值为：

0x55febc3f2748 <x>: 0x1e 0x3a 0xfd 0x46

0x7ffe7cb7a648 <y>: 0x24 0x15 0xe8 0x76 0x8b 0xd0 0x97 0x43

0x55febc5f3010 <z>: 0x31 0x31 0x39 0x31 0x30 0x30 0x30 0x36

0x55febc5f3018 <z+8>:0x30 0x36 0x2d 0xe9 0x99 0x88 0xe4 0xb8

0x55febc5f3020 <z+16>: 0x80 0xe5 0xb8 0x86 0x00

使用objdump命令，得到反汇编文件如下

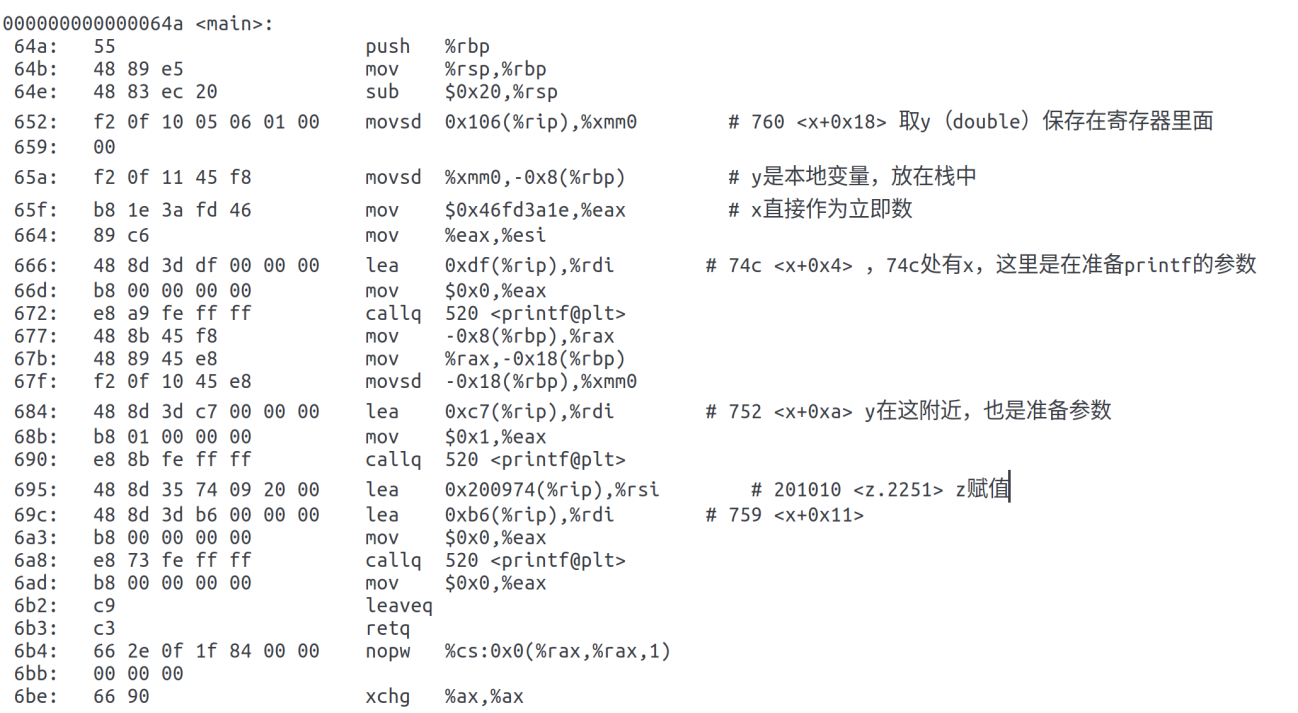


图3.2.3 反汇编文件中的main函数

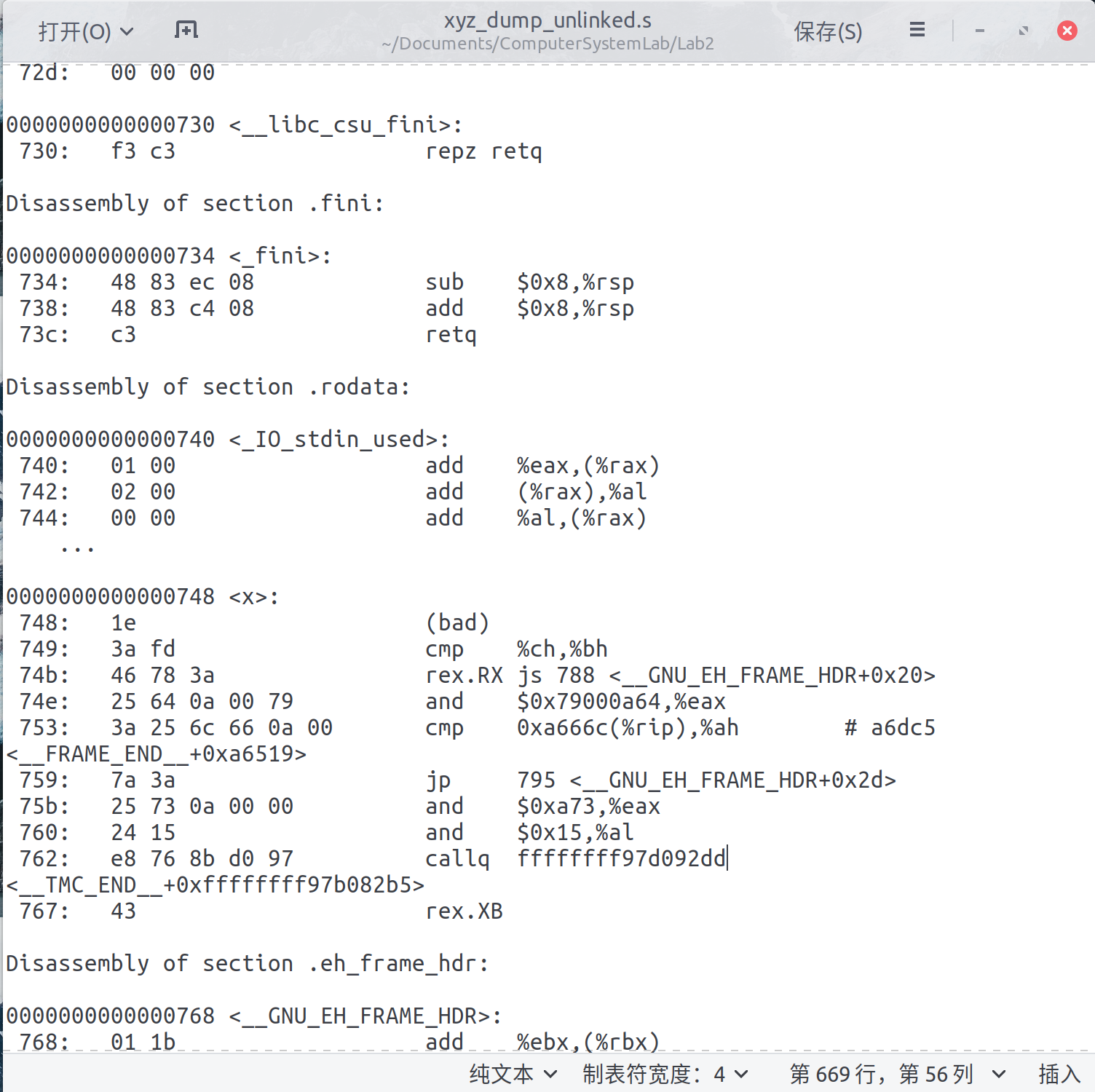


图3.2.4 反汇编文件中的x

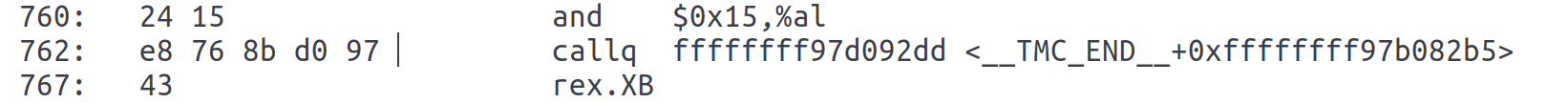


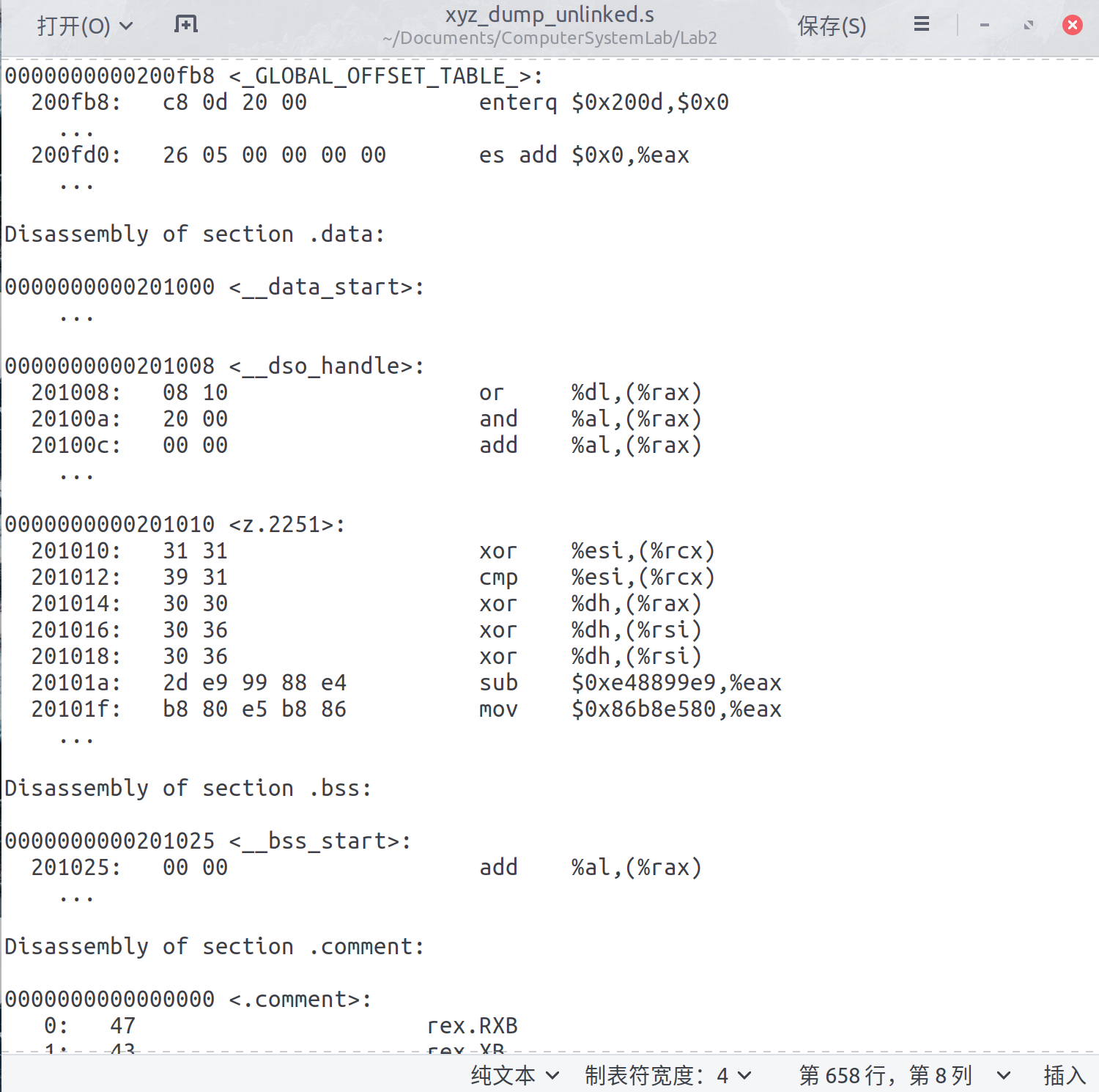
图3.2.5 反汇编文件中的y

图3.2.6 反汇编文件中的z

x与y在\_\_\_汇编\_\_\_阶段转换成补码与ieee754编码。

数值型常量与变量在存储空间上的区别是：数值型常量一般存放在.rodata段、常量存储区，有的立即数直接编码在指令里，存放在代码段(.text)中；数值型变量若为未初始化的全局变量将存放在.bss段（在运行时将为.bss段分配空间），若为初始化的全局变量以及初始化的静态变量将存放在.data段，函数内的局部变量（用户临时创建的）、传递的函数参数一般存放在栈上，动态分配的变量在堆中。

字符串常量与变量在存储空间上的区别是：指针方式创建的字符数组，是常量字符串；方括号（[]）方式创建的字符数组是字符串变量。未初始化的全局字符串存放在bss段；全局的初始化的字符串常量内容存储在rodata段；全局初始化后的字符串变量存储在data段；函数内的局部变量（用户临时创建的）或常量中，字符串内容存放在rodata段，变量或常量指针放入栈中；静态字符串变量存放在data段。

常量表达式在计算机中处理方法是：在编译阶段转换成常量来存储

**3.2提示：**

**①在linux下生成可执行程序，假设是a.out。然后用objdump -dx a.out > a-dump.s 生成反汇编文件a-dump.s，查看a-dump.s**

**②gdb ./a.out ↙ layout asm ↙ b main ↙ r ↙ disp argc ↙….**

## 3.3 main的参数分析

反汇编查看x、y、z的地址

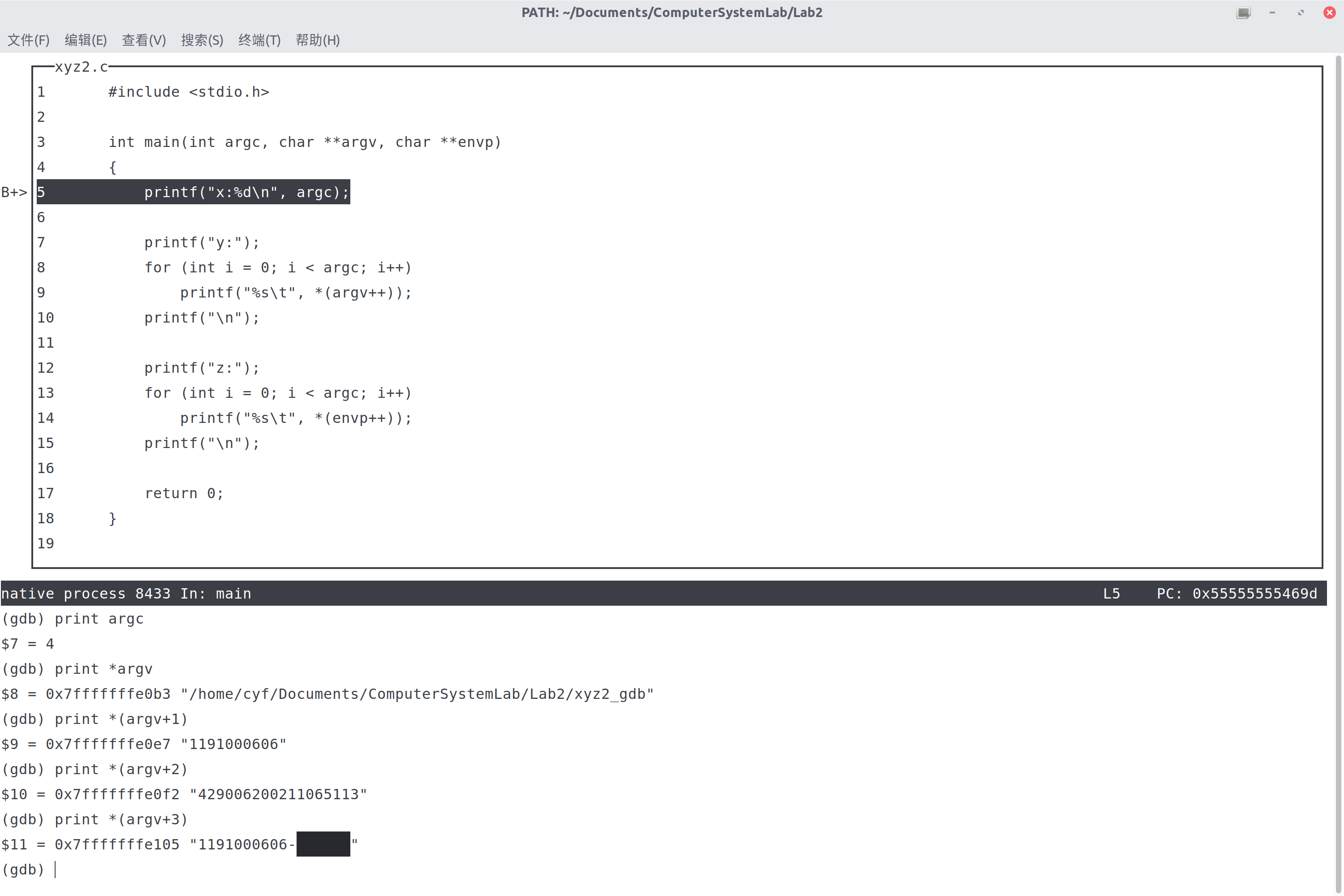


图3.2.6 x,y,z的地址

x: 0x7fffffffe0e7

y: 0x7fffffffe0f2

z: 0x7fffffffe105

命令行传递参数，反汇编观察argc、argv的地址与内容,

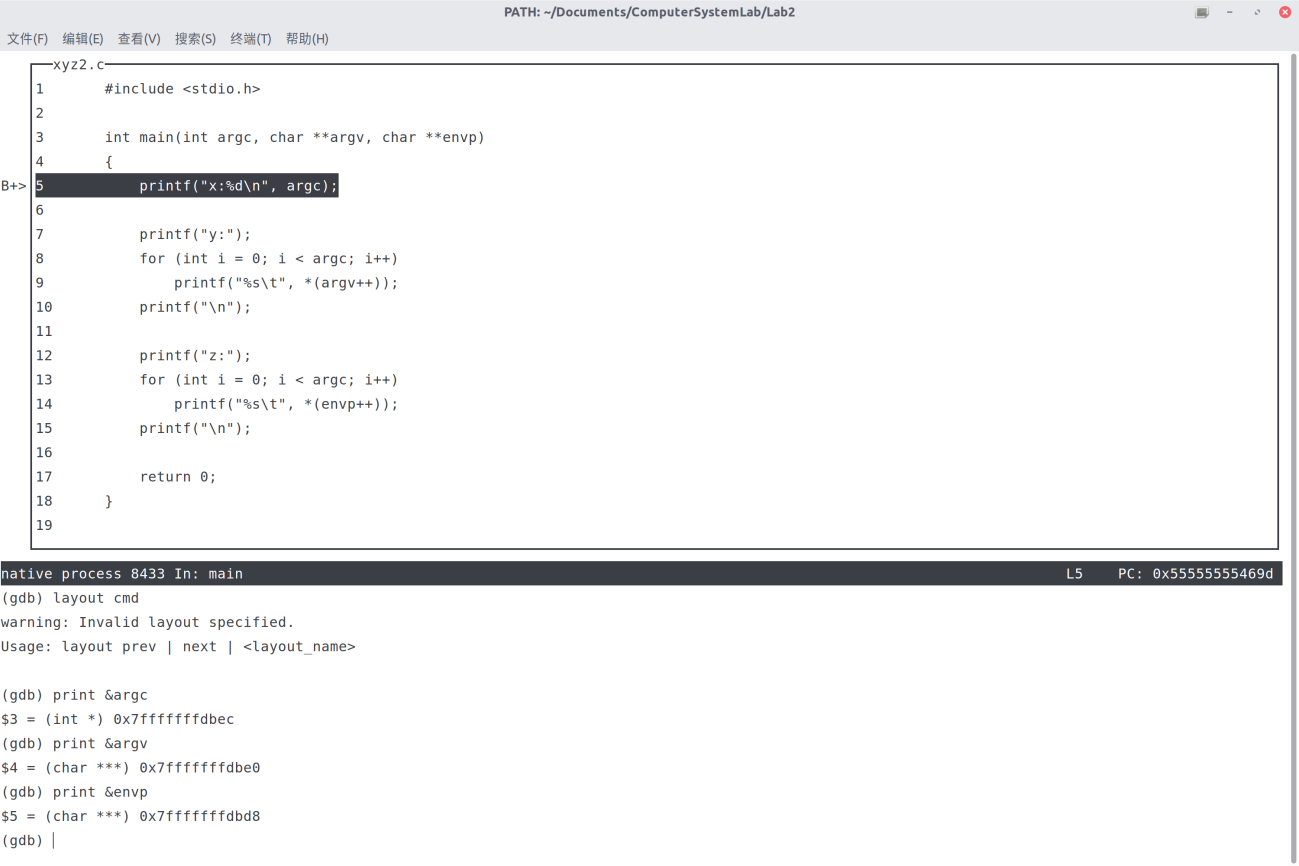


图3-3-2 argc,argc,envp的地址

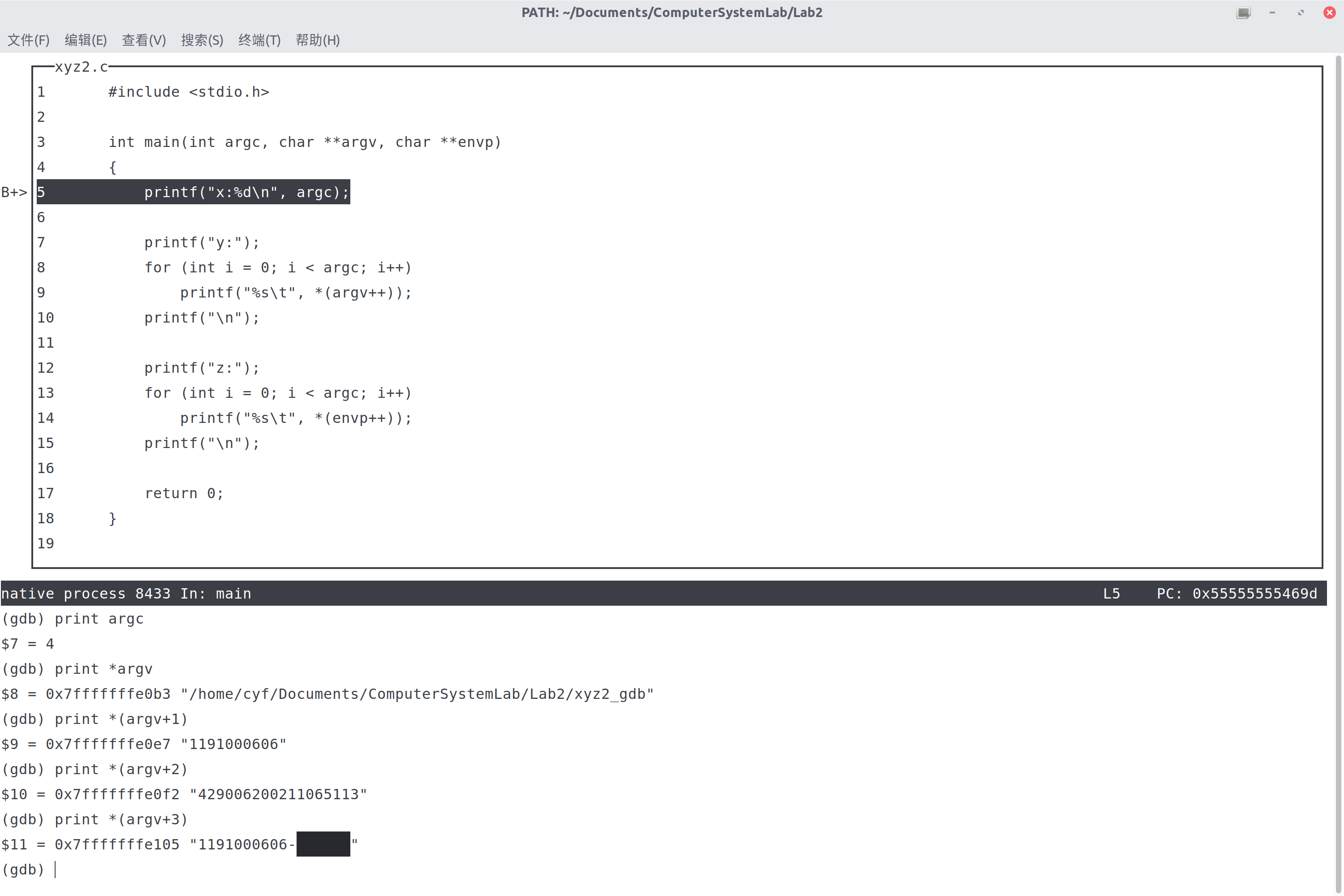


图3.3.3 argc,argc的内容

## 3.4 指针与字符串的区别

cstr的地址与内容截图，pstr的内容与截图，如图3-4-1。

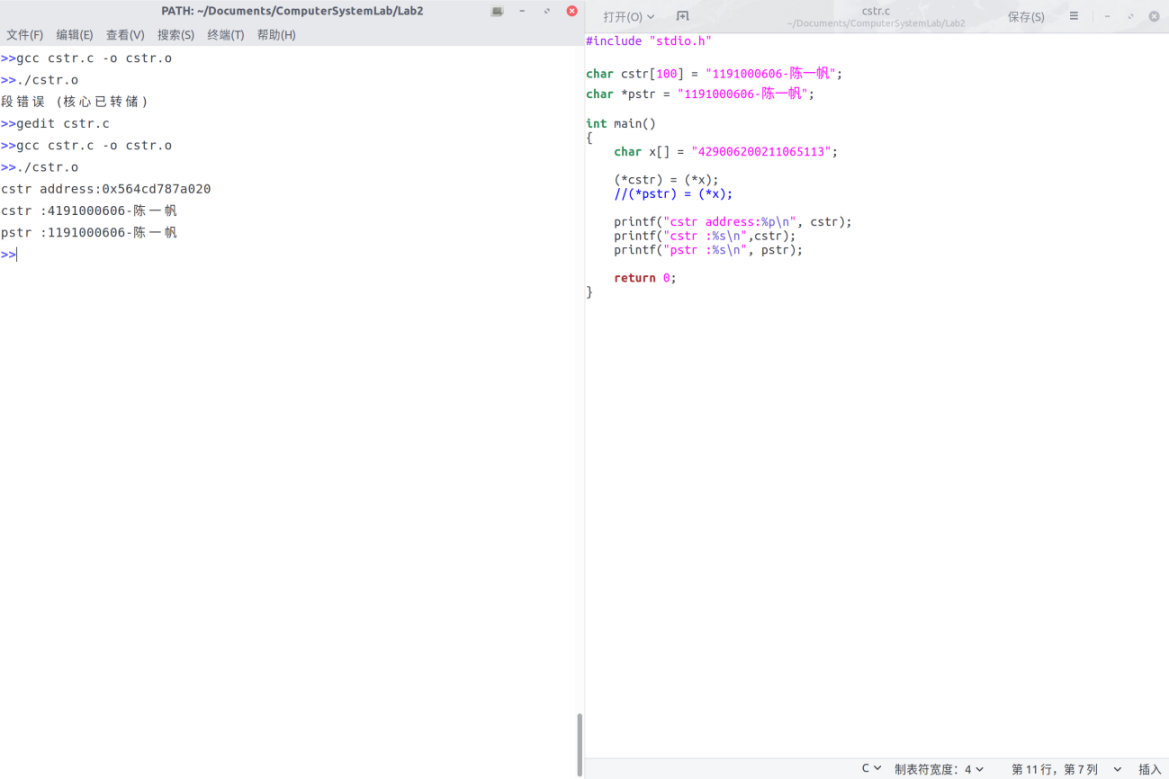


图3.4 cstr与pstr

pstr修改内容会出现什么问题：用学号+姓名初始化后，若以 \*pstr = “…”/…为身份证号/这一形式赋值，将会报错，因为原先pstr的值（学号+姓名）存储在常量区，这种赋值语句会意图修改常量区的值，不合法；若以pstr = “…”/…为身份证号/的形式赋值是合法的，但此时指针指向位置发生改变，即指针指向了常量区另一个地方，该地方存有后赋值的字符串常量。

# 第4章 深入分析UTF-8编码

## 4.1 提交utf8len.c子程序

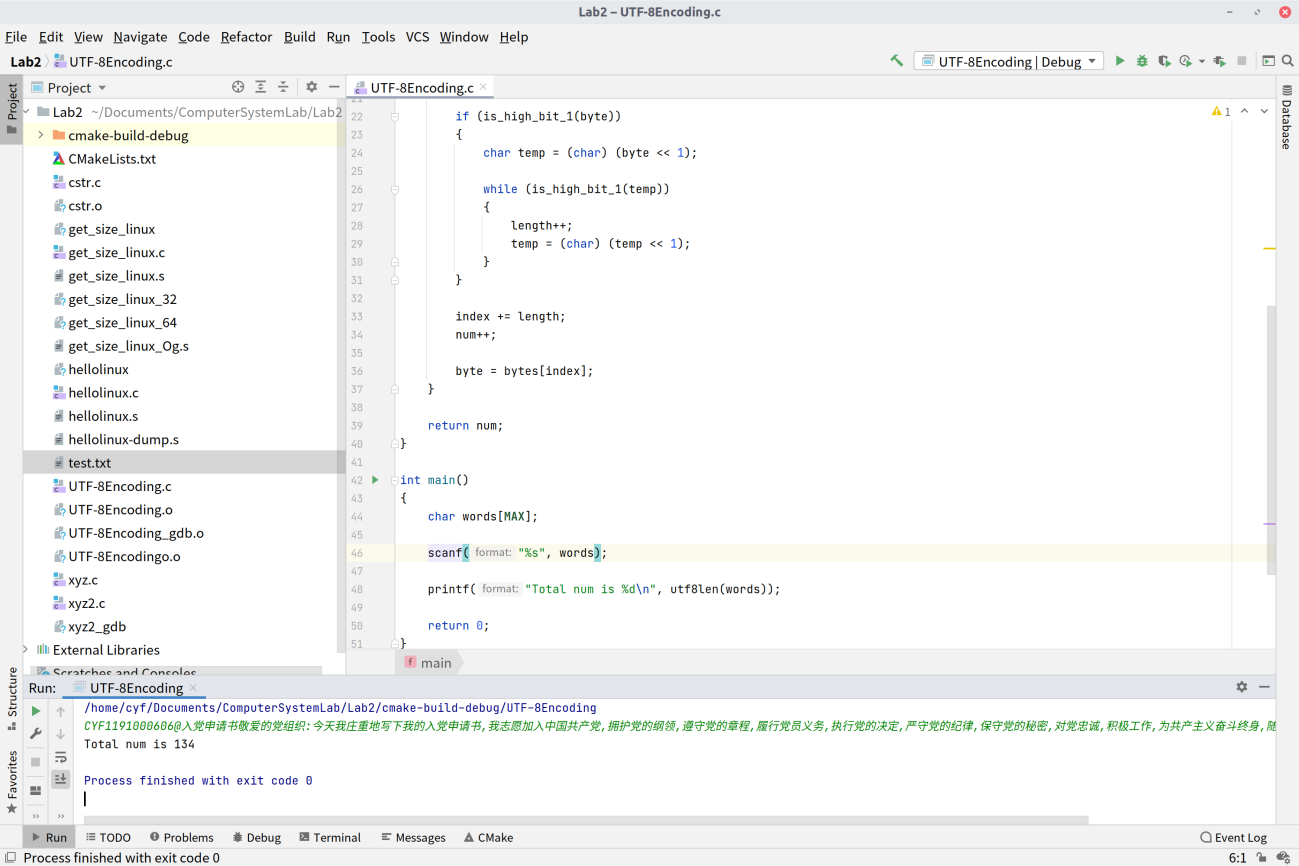


图4.1 utf8len.c

## 4.2 C语言的strcmp函数分析

分析论述：strcmp到底按照什么顺序对汉字排序

C语言中，Windows系统下汉字使用GB编码，一般为2字节，Linux下采用Unicode标准，一般使用UTF-8的编码方式，汉字一般为三个字节。对于两个字符串，strcmp逐个字节比较其有符号数值。数值大的，字符串就比较大。若前面字符串＞后面字符串，返回正数，=返回0，<返回负数。

## 4.3讨论：按照姓氏笔画排序的方法实现（选做，不做要求）

1. 首先按照笔画数由少到多排序的原则。姓氏笔画少的排在前面，姓氏笔画多的排在后面。
2. 笔画数相同的，比较其起笔的种类，不妨按照横竖撇捺的顺序，第一笔相同，则比较下一笔，直到不同为止。

# 第5章 数据变换与输入输出

## 5.1 提交**cs\_atoi.c**

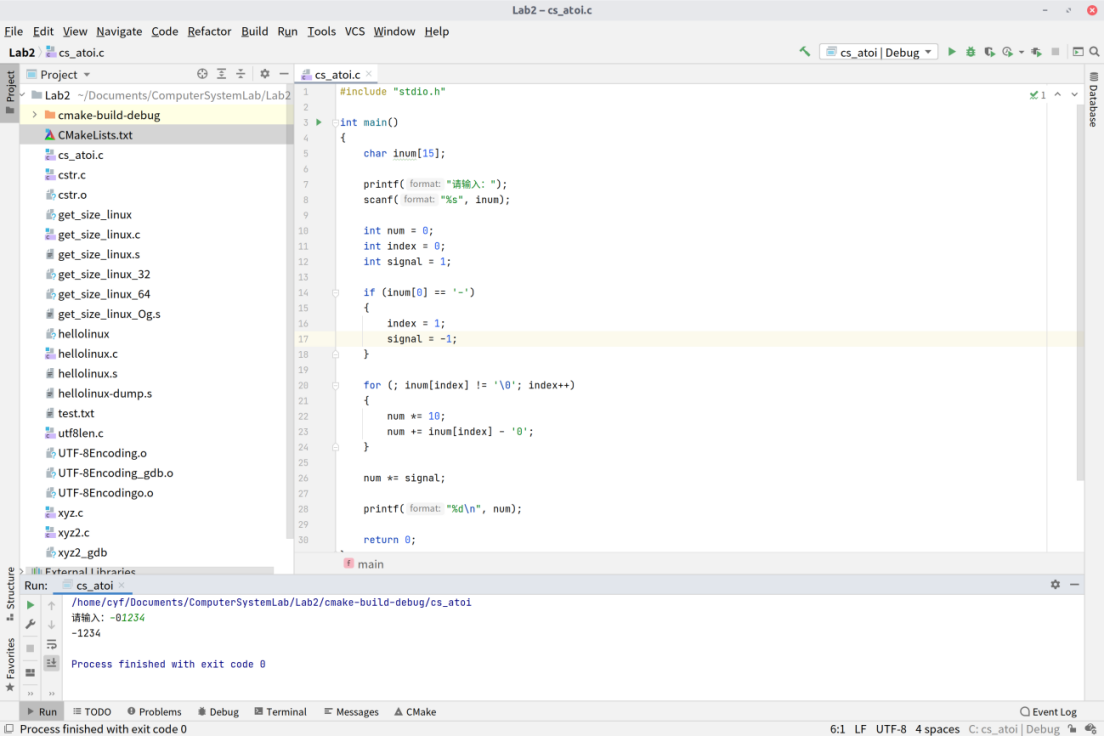


图5.1 cs\_atoi.c

## 5.2 提交**cs\_atof.c**

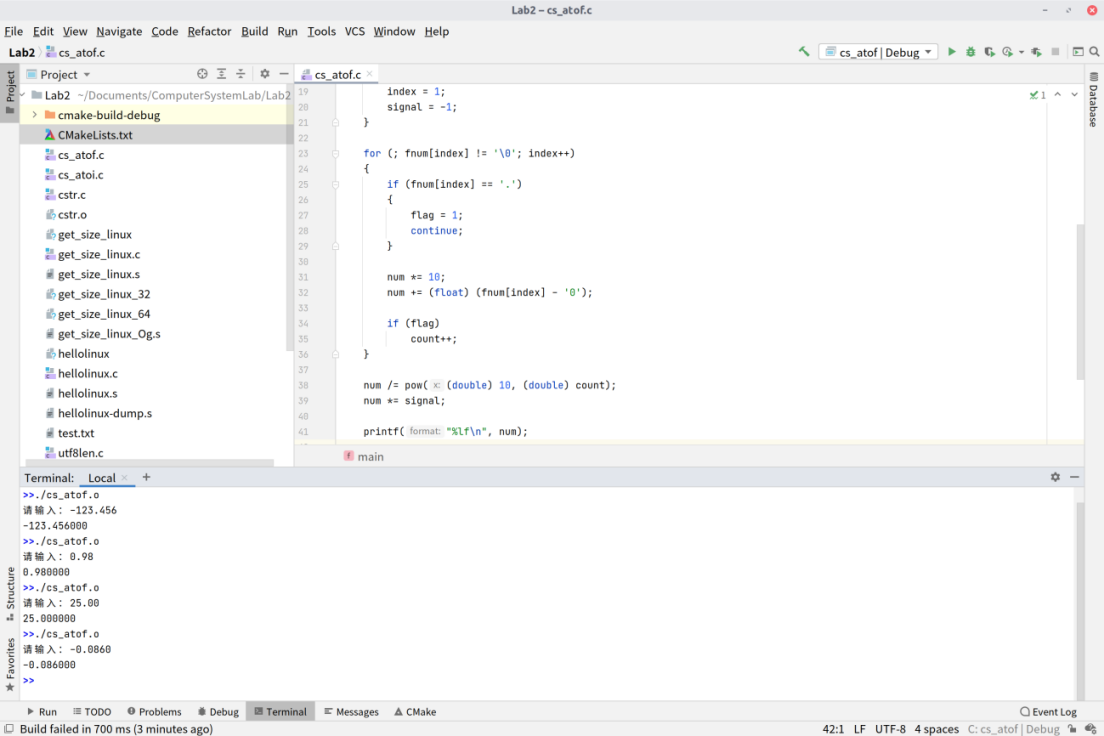


图5.2 cs\_atof.c

## 5.3 提交**cs\_itoa.c**

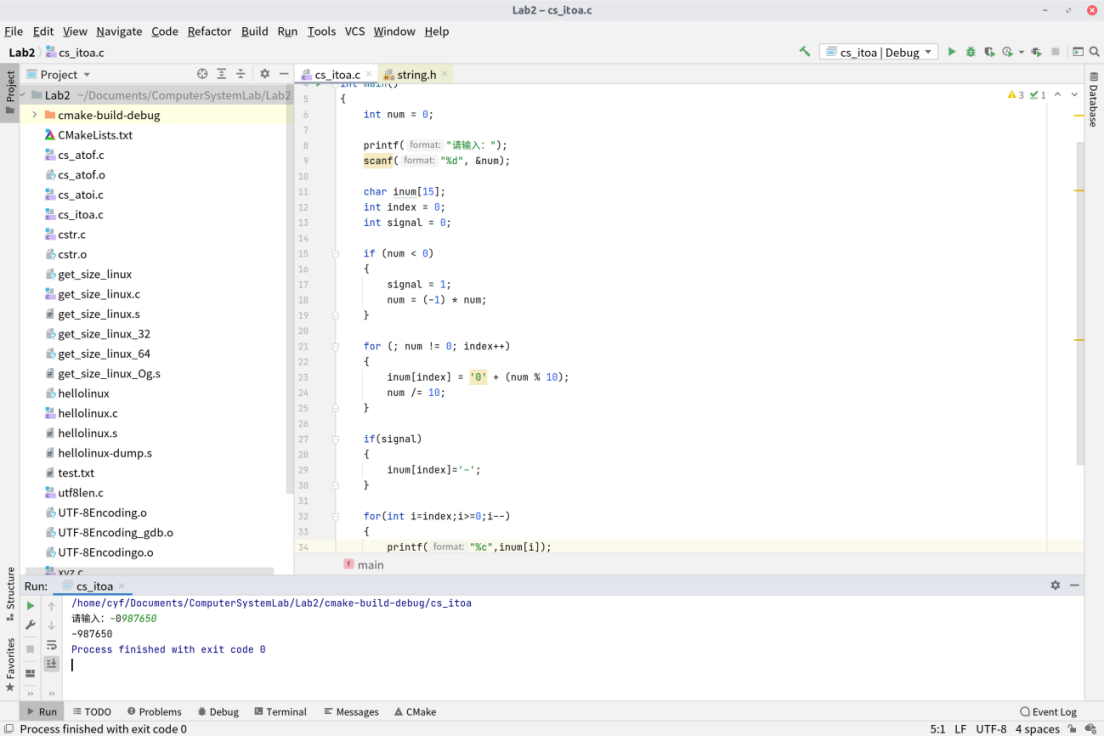


图5.3 cs\_itoa.c

## 5.4 提交**cs\_ftoa.c**

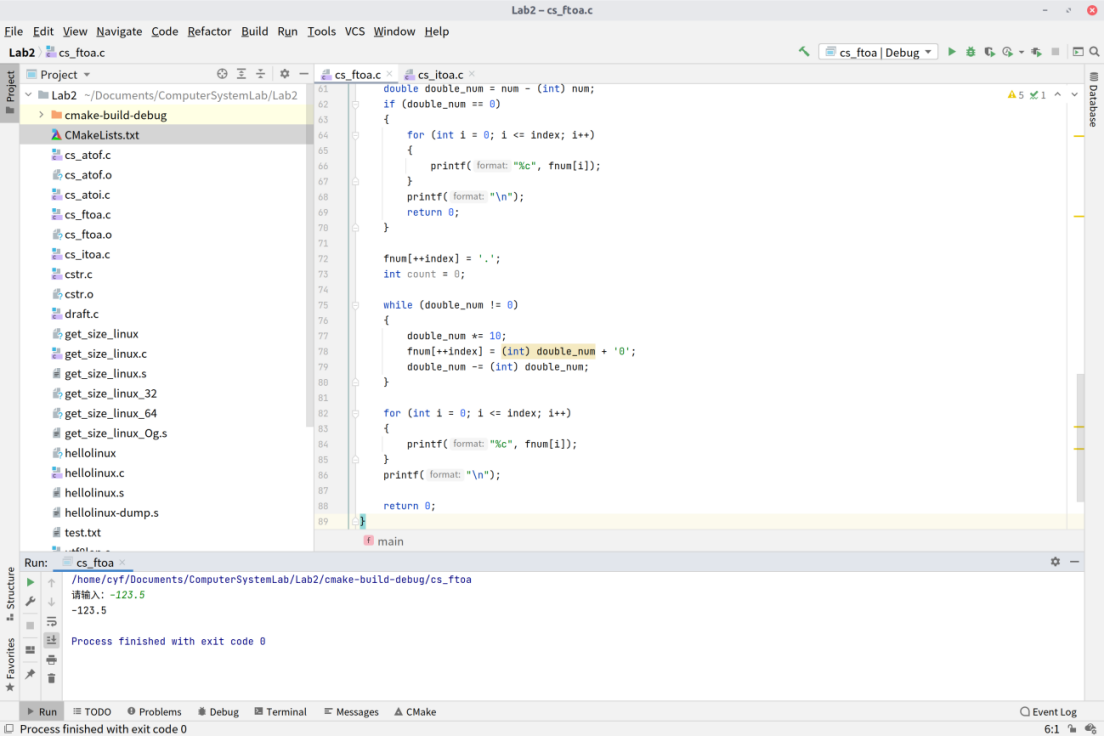


图5.3 cs\_ftoa.c

## 5.5 讨论分析OS的函数对输入输出的数据有类型要求吗

论述如下：

OS的函数将输入输出的数据都看成字符串来处理，以字符串的形式输入输出，通过输入输出流来实现，在Linux下标准输入流为stdin，标准的输入设备默认键盘；标准输出流为stdout，标准的输出设备默认屏幕；stderr 为标准错误流，只有程序出错时才会执行的流程。OS先接收输入的字符串，再进行进一步处理；对于输出，也就将要输出的数据转换成字符串输出的。

**第6章 整数表示与运算**

## 6.1 提交fib\_dg**.c**

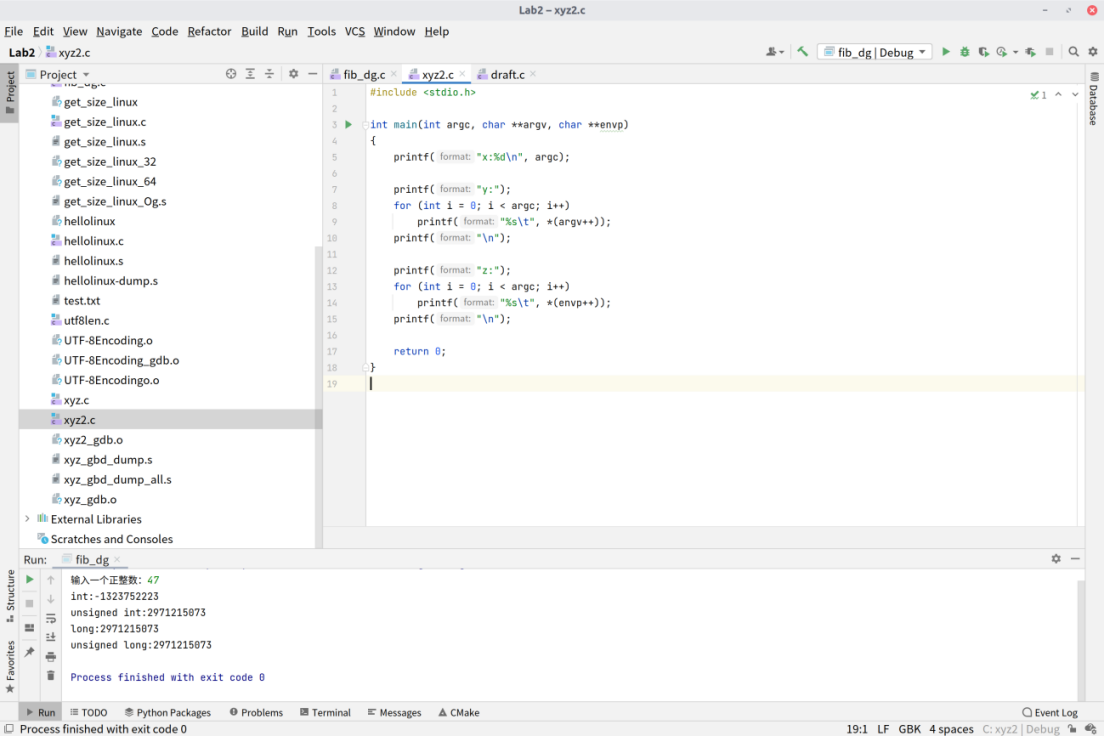


图6.1 fib\_dg.c

## 6.2 提交**fib\_loop.c**

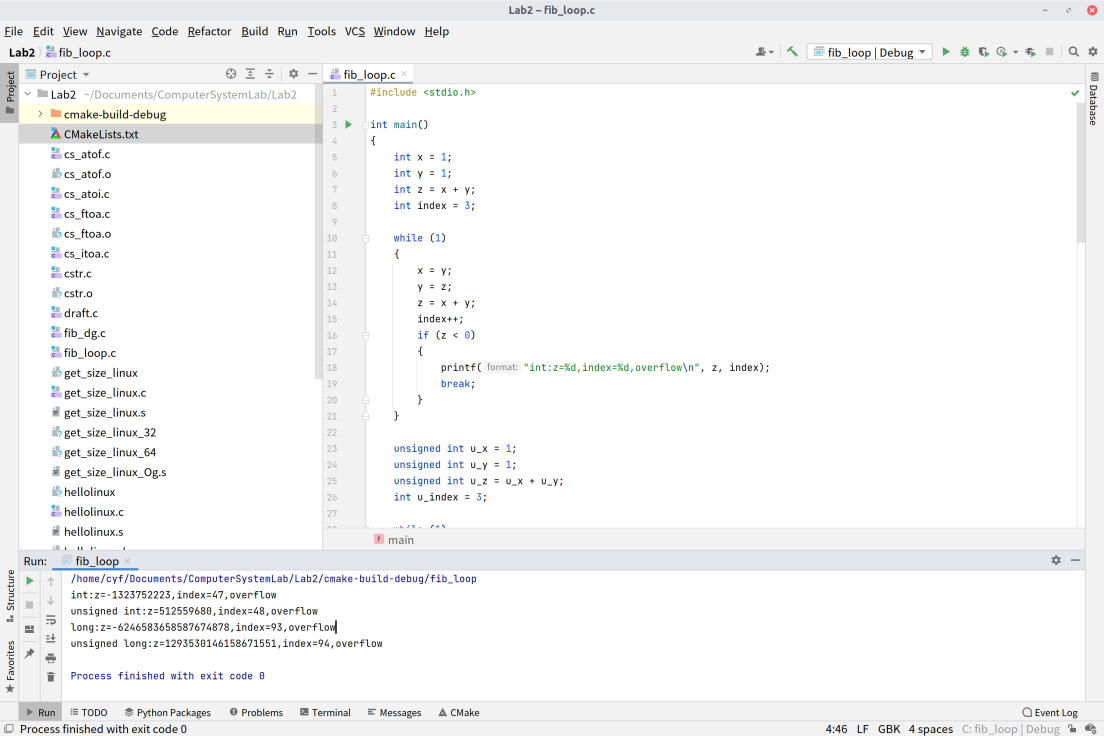
****

图6.2 fib\_loop.c

## 6.3 fib溢出验证

int 时从n=\_\_\_\_\_47\_\_\_\_时溢出，long时n=\_\_\_\_\_\_93\_\_\_\_\_\_时溢出。

unsigned int 时从n=\_\_\_48\_\_\_时溢出，unsigned long时n=\_\_\_\_94\_\_\_\_时溢出。

## 6.4 除以0验证：

除以0：截图1：报错：SIGFPE (Arithmetic exception)

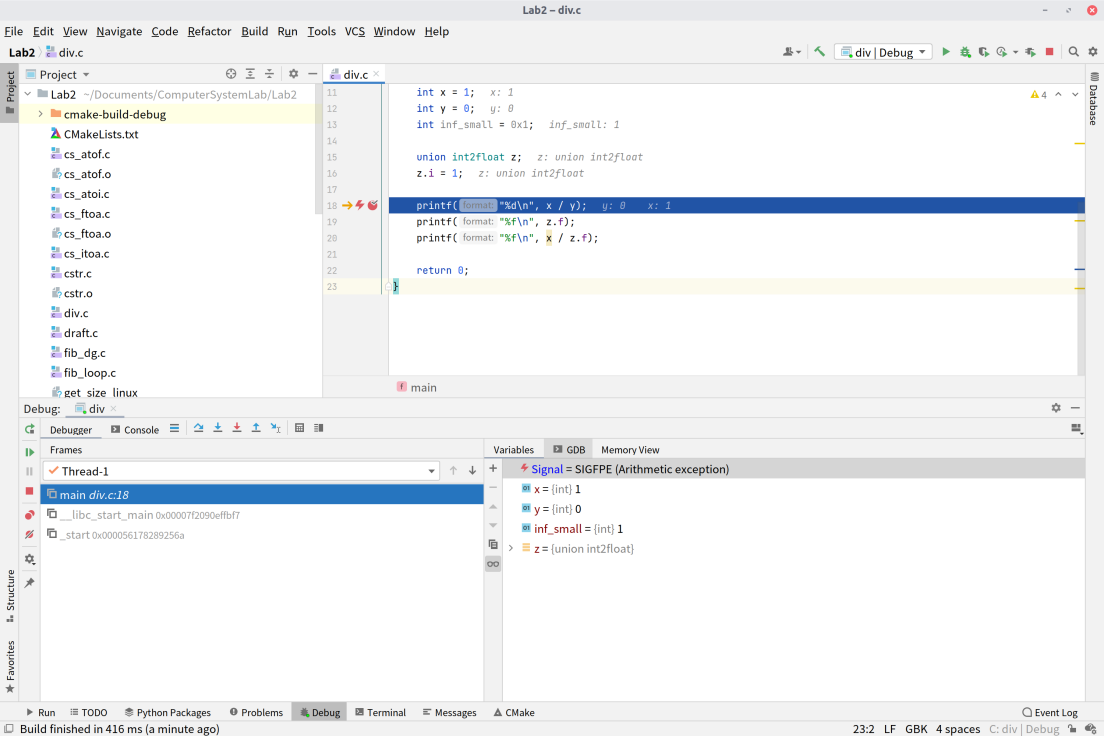


图6.4.1 1/0结果

除以极小浮点数，截图2：得到inf，表示正无穷

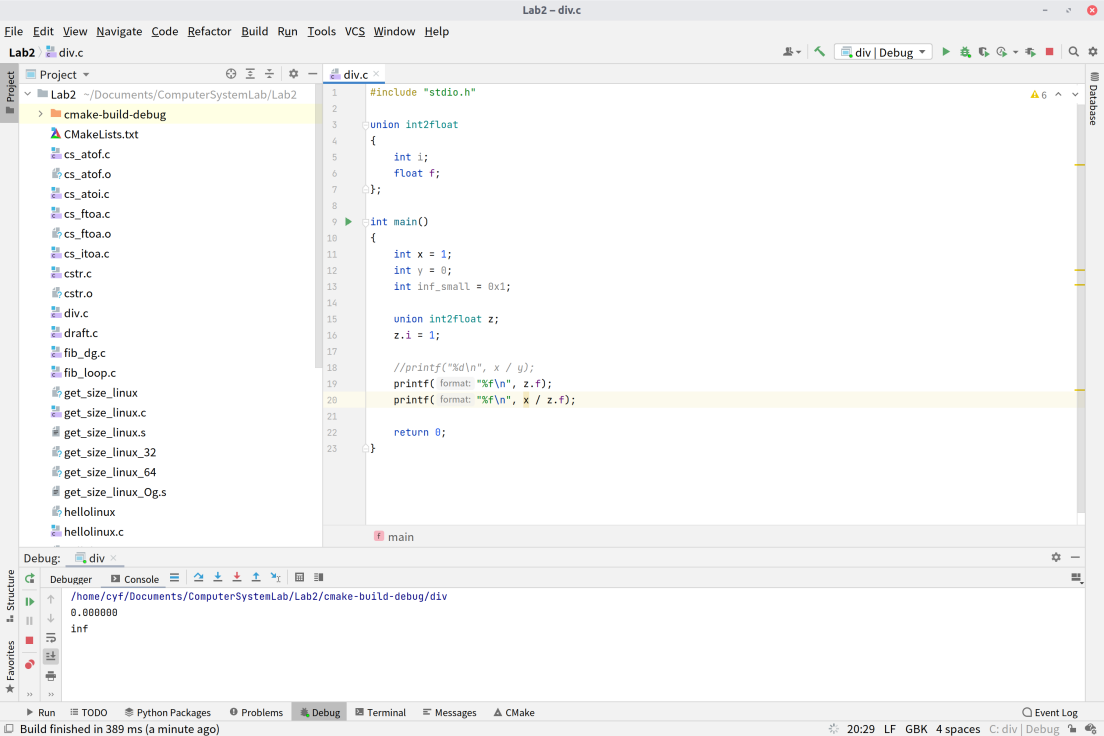


图6.4.2 1除以最小浮点数结果

# 第7章 浮点数据的表示与运算

## 7.1 正数表示范围

写出float/double类型最小的正数、最大的正数（非无穷）

float：最小：2^(-149)，最大：1^128-2^104

double：最小：2^(-1074)，最大：1^1024-2^971

## 7.2浮点数的编码计算

1. 按步骤写出float数-1.1的浮点编码计算过程，写出该编码在内存中从低地址字节到高地址字节的16进制数值

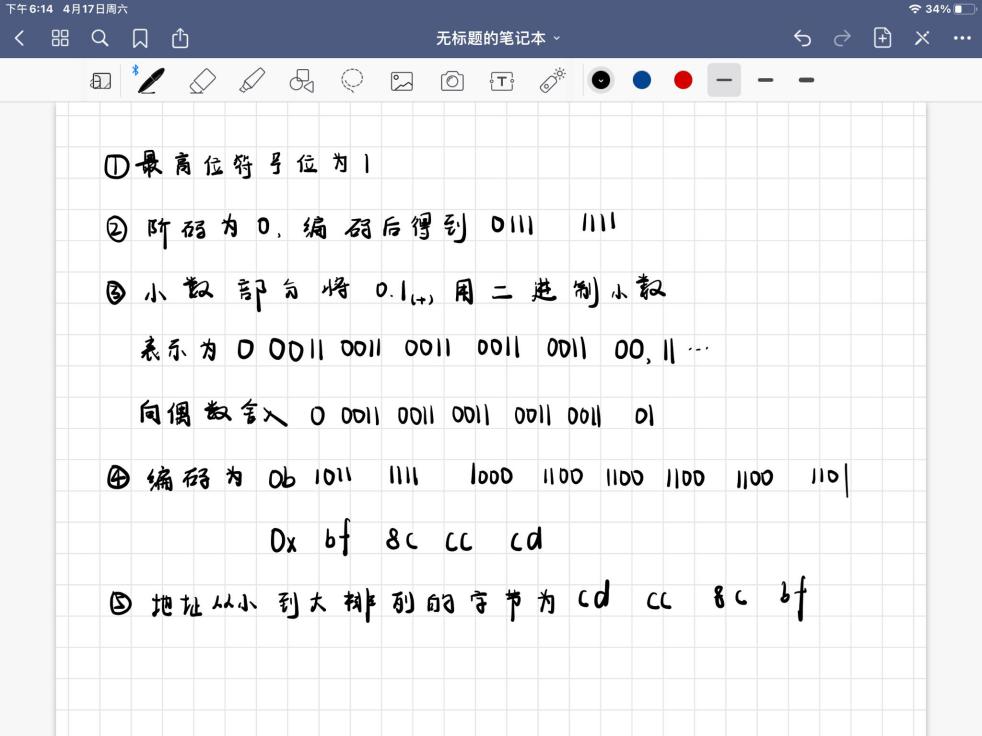


图7.2.1 计算步骤

1. 验证：编写程序，输出值为-1.1的浮点变量其各内存单元的数值，截图。

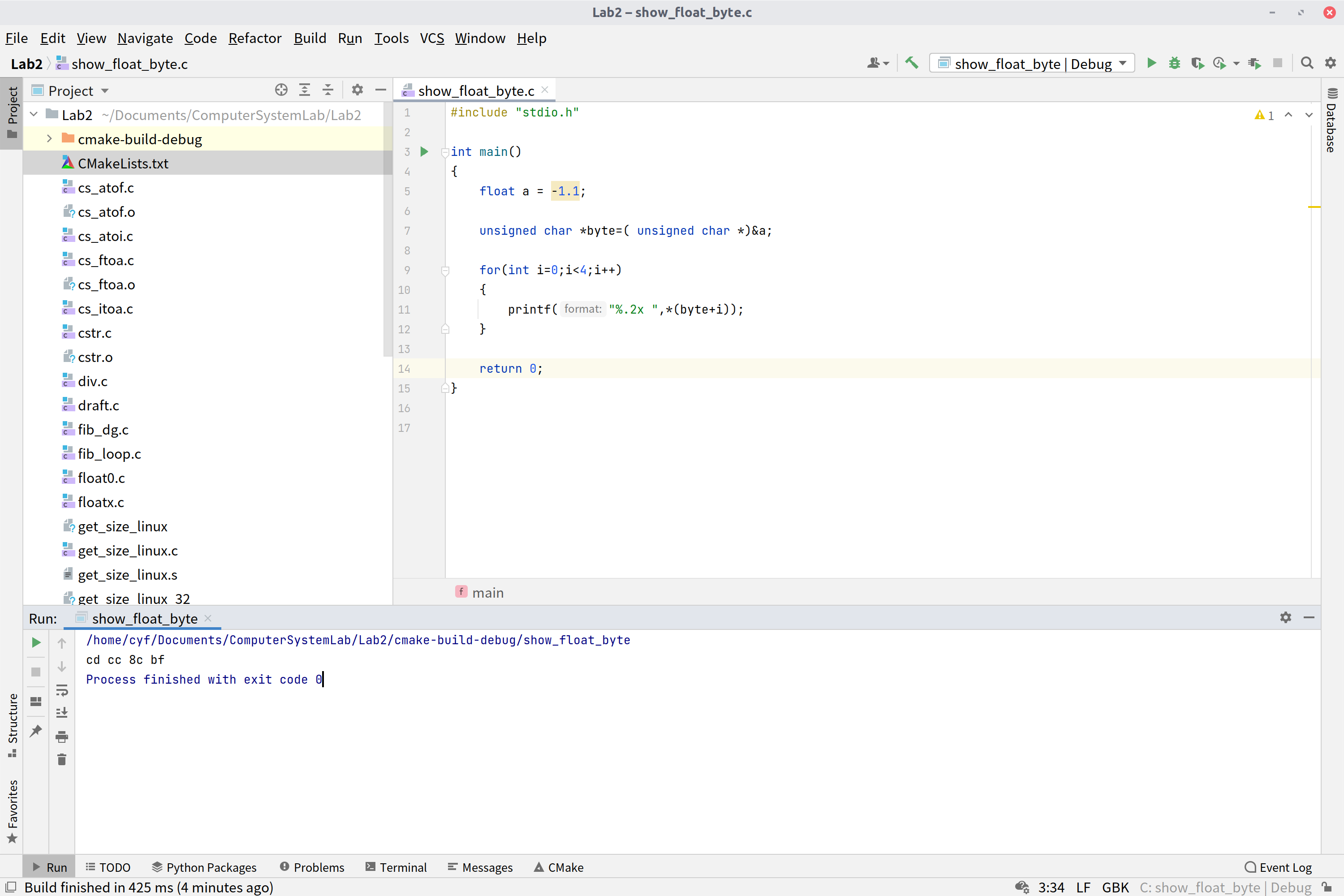


图7.2.2 验证

## 7.3特殊浮点数值的编码

1. 构造多float变量，分别存储+0-0，最小浮点正数，最大浮点正数、最小正的规格化浮点数、正无穷大、Nan,并打印最可能的精确结果输出（十进制/16进制）。

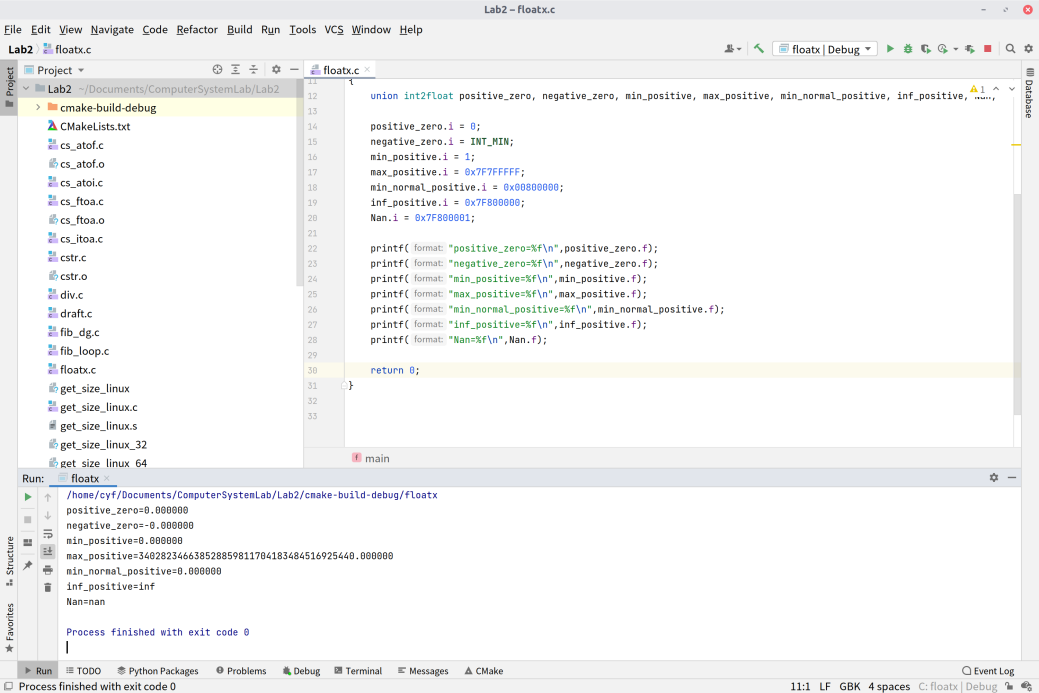


图7.3 floatx.c

（2）提交子程序floatx.c

## 7.4浮点数除0

1. 编写C程序，验证C语言中float除以0/极小浮点数后果，截图

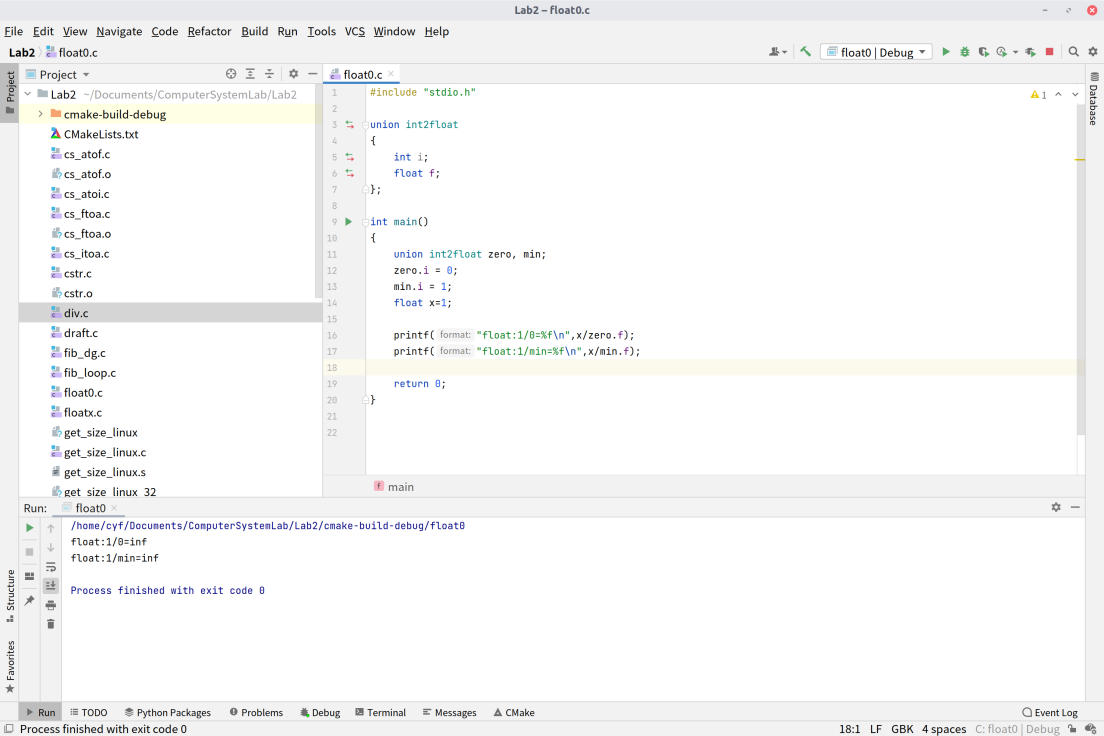


图7.4 float0.c

（2）提交子程序float0.c

## 7.5 Float的微观与宏观世界

按照阶码的数值区域，float编码最密集区域的阶码编码是：\_0000 0000\_\_, 该区域中相邻浮点数编码的数值的间距是：\_\_\_2^(-149)\_\_\_；float编码最稀疏区域的阶码编码是：\_\_\_\_1111 1110\_\_\_\_\_\_, 该区域中相邻浮点数编码的数值的间距是：\_\_\_\_2^104\_\_\_\_\_\_。

最小正数变成十进制科学记数法，最可能能精确到多少 1.401298\*10^-45

最大正数变成十进制科学记数法，最可能能精确到多少2.028241\*10^31

## 7.6 讨论：任意两个浮点数的大小比较

因为计算机存储的特性，任意两个浮点是不能用==直接比较。比较好的方法就是用两个数之间的差值小于某个最小值，比如对于两个浮点数a,b，如果要比较大小，那么常常会设置一个精度，如果fabs(a-b)<=1e-6，那么就是相等了。 fabs是求浮点数绝对值的函数。类似的判断大于的时候，就是if(a>b && fabs(a-b)>1e-6)。

判断小于的时候，就是if(a<b && fabs(a-b)>1e-6)。

对于以及转化为IEEE型的浮点数：

首先判断是否为NAN，若为两个数都为NAN，vb中默认相等，cb中不可比较。

再判断两个数符号位，如果一个为正数，一个负数，则正数大于负数，若为0，则正0和0相等

然后取出阶码部分，若阶码为正数，则阶码大的数大，若为负数，则解码小的数小

取出尾数比较，若为为负数，则尾数大的小，若为正数，尾数小的大。**第8章 舍尾平衡的讨论**

## 8.1 描述可能出现的问题

计量单位的改变 , 常常出现数据不平衡的现象

在报表处理过程中，有时需要对数据进行舍位平衡。舍位，就是将表格中的数据抹去尾数的一位或几位数，例如：将以元为单位的表格，改为以千元为单位的数据表格，就是舍去表格中数据的后三位。舍位处理往往会采取四舍五入计算，这时就会产生误差，而如果报表中有这些数据的合计数值，那么舍位时产生的误差就会积累，有可能导致舍位过的数据与其合计值无法匹配。例如，保留一位小数的原始的数据是4.5+4.5=9.0，而四舍五入只保留整数部分后，平衡关系就变为5+5=9了，看上去明显是荒谬的。

## 8.2 给出完美的解决方案

舍位后总计产生的误差，称为“平衡差”（真实数值-平衡后理论数值），舍位平衡其实就是消除平衡差的过程。

（1）单向舍位平衡，即如果在数据统计时，每个数据只用于一次合计，那么在处理舍位平衡时，只需要根据合计值的误差，调整使用的各项数据就可以了，这属于比较简单的情况。

考虑方法1：我们将平衡差直接加到第一个数据，可以解决舍位平衡。但这种方法很可能会使改动后的该数据变动较大，从而影响后续操作，故该方法不具备可行性。

考虑方法2：我们将正精度单位（例如保留到0.1/1，正精度单位即为0.1/1）（如果平衡差>0）或负精度单位（例如保留到-0.1/-1，正精度单位即为-0.1/-1）（如果平衡差<0）加到绝对值最大的数据中，这样使得每个数据没有更改或偏差不是很大，但是寻找绝对值最大的多个数据需要进行排序算法，对于大量数据不是很适用，所以该方法排除。

考虑方法3：我们将所有的非零数按顺序加上正或负精度单位。考虑到在四舍五入时，0并不会产生误差，而且如果修改数据中的0，这样的变动会比较明显，因此在调整时将保留原始数据中的0不变。这样调整平衡的结果比较合理。同时这种方案避免了排序操作，效率较高，因此这种舍位平衡的规则最为常用。

（2）双向舍位平衡，即如果数据在行向和列向两个方向同时需要计算合计值，同时还需要计算所有数据的总计值。

有一些平衡差只与合计值相关，这样的平衡差称为合计平衡差。在双向舍位平衡表中，只存在一横一纵两个合计平衡差。其它的平衡差都会和具体数据相关，如Feb这个月最下方的平衡差，这种平衡差称为非合计平衡差。

考虑情况1：横向与纵向的非合计平衡差符号相同。只需要调整交叉点处的数据，根据平衡差符号加减最小调整值。

考虑情况2：同向的2个非合计平衡差符号相反。任选另一方向平衡差为0的数据，将这两个方向的数分别根据按平衡差的符号加减最小调整值。

考虑情况3：某个合计平衡差与另一方向的非合计平衡差符号相反。调整交叉点处的合计数据，根据合计平衡差的符号加减最小调整值。

考虑情况4：某个合计平衡差与同方向的非合计平衡差符号相同。任选一另一方向平衡差为0的数据，同时调整这2个方向的数据

考虑情况5：两个方向合计平衡差的符号相同。此时，只有合计数据影响了结果的平衡。任选一个非合计值，根据合计平衡差的符号加减最小调整值，同样调整这个数据的横向和纵向合计值。

综上：

1.对于其它的情况，说明计算有误，是无法通过1次调整达成舍位平衡的。

2.上面的情况往往是混合出现的。

3.可以先处理第(1)种情况，即所有非合计行列平衡差符号相同的情况，再处理第(2)种情况，将非合计行/列中不同符号的平衡差消除。全部调整理完毕后，非合计行与非合计列的平衡差只能各为一种符号。此时再处理第(3)种和第(4)种情况，将非合计行/列的平衡差与合计行/列的平衡差配合消除。最后，如果两个方向行/列的平衡差仍未消除，再按照第(5)中情况处理。这样，就可以对一般性的数据组合完成双向舍位平衡处理了。

# 第9章 总结

## 9.1 请总结本次实验的收获

1.学会使用gbd调试，充分理解了各种IDE的debug功能背后更底层的操作

2.使用了联合Union，使得类型转换更加方便

3.对汇编代码有了更深刻的理解

## 9.2 请给出对本次实验内容的建议

感觉对于浮点数类型转换，与第二章CSAPP的作业大量重合。

更希望本次实验可以引导我们会简单阅读完整的汇编代码。

千年虫，舍尾平衡等拓展内容难度较高。

# 参考文献

**为完成本次实验你翻阅的书籍与网站等**

[1] 林来兴. 空间控制技术[M]. 北京：中国宇航出版社，1992：25-42.

[2] 辛希孟. 信息技术与信息服务国际研讨会论文集：A集[C]. 北京：中国科学出版社，1999.

[3] 赵耀东. 新时代的工业工程师[M/OL]. 台北：天下文化出版社，1998 [1998-09-26]. http://www.ie.nthu.edu.tw/info/ie.newie.htm（Big5）.

[4] 谌颖. 空间交会控制理论与方法研究[D]. 哈尔滨：哈尔滨工业大学，1992：8-13.

[5] KANAMORI H. Shaking Without Quaking[J]. Science，1998，279（5359）：2063-2064.

[6] CHRISTINE M. Plant Physiology: Plant Biology in the Genome Era[J/OL]. Science，1998，281：331-332[1998-09-23]. http://www.sciencemag.org/cgi/ collection/anatmorp.