# 《微控制器原理、接口与应用》编程实践报告

|  |  |
| --- | --- |
| **项目名称：** | **多功能矩阵运算处理平台** |
| **项目时间：** | **2020—2021学年秋冬学期** |
| **指导老师：** | **沈海斌 罗小华** |
| **小组成员：** | **温晨怡 蔡悦 林炬乙（24组）** |
| **报告学生：** | **林炬乙** |
| **提交日期：** | **2020年12月29日** |

目 录

[1 项目概况 3](#_Toc60060789)

[1.1 项目目标 3](#_Toc60060790)

[1.2 组员分工 3](#_Toc60060791)

[2 软硬件平台与算法 4](#_Toc60060792)

[2.1 微控制器介绍 4](#_Toc60060793)

[2.2 软件开发平台介绍 5](#_Toc60060794)

[2.2.1 软件开发平台 5](#_Toc60060795)

[2.2.2 与目前最先进平台功能的对比 6](#_Toc60060796)

[2.3 矩阵处理算法 7](#_Toc60060797)

[2.3.1 矩阵求逆算法 7](#_Toc60060798)

[2.3.2 矩阵求特征值算法 7](#_Toc60060799)

[3 设计、检查与调试 8](#_Toc60060800)

[3.1 设计技巧 8](#_Toc60060801)

[3.2 检查技巧 9](#_Toc60060802)

[3.2.1检查表 9](#_Toc60060803)

[3.2.2 测试样例 10](#_Toc60060804)

[3.2.3代码覆盖率 10](#_Toc60060805)

[3.3 调试技巧 11](#_Toc60060806)

[3.3.1单步调试 11](#_Toc60060807)

[3.3.2.交互性强 12](#_Toc60060808)

[3.3.3. assert宏 12](#_Toc60060809)

[3.3.4.断点 12](#_Toc60060810)

[3.3.5. 寄存器 13](#_Toc60060811)

[3.3.6. 存储区 13](#_Toc60060812)

[3.3.7.变量窗口 14](#_Toc60060813)

[4 项目成果展示与项目总结 15](#_Toc60060814)

[4.1 项目功能与性能 15](#_Toc60060815)

[4.1.1 程序实现结果 15](#_Toc60060816)

[4.1.2 求逆性能测试 18](#_Toc60060817)

[4.1.3 求特征值性能测试 21](#_Toc60060818)

[4.2 各组员成果与代码量 24](#_Toc60060819)

[[参考文献] 24](#_Toc60060820)

[5 个人工作总结 24](#_Toc60060821)

[5.1 分工与完成情况 24](#_Toc60060822)

[5.1.1 个人分工内容 24](#_Toc60060823)

[5.1.2 个人工作完成情况 24](#_Toc60060824)

[5.2 工作亮点 25](#_Toc60060825)

[5.3 各环节技巧 26](#_Toc60060826)

[5.3.1 代码编写技巧 26](#_Toc60060827)

[5.3.2 调试技巧 28](#_Toc60060828)

[5.4 计划时间表与完成时间表 28](#_Toc60060829)

[5.5 遇到的问题与解决 29](#_Toc60060830)

[5.6 体会与建议 30](#_Toc60060831)

## 1 项目概况

### 1.1 项目目标

本项目希望能为同学们的日常学习、科研中的矩阵运算处理过程提供实用工具。将利用keil5进行算法实现，proteus8进行硬件仿真与交互。

我们将实现对三阶矩阵的加法、减法、乘法、求逆、求特征值等五项功能。在交互中，使用虚拟键盘进行数据输入，并将输入过程、计算结果等实时显示在LCD屏上。

### 1.2 组员分工

|  |  |  |  |
| --- | --- | --- | --- |
| 组员 | 温晨怡 | 蔡悦 | 林炬乙 |
| 分工 | 乘法功能实现  求逆功能实现 | 输入功能实现  输出功能实现 | 求特征值功能实现  加法减法功能实现 |
| 讨论机制 | 自由讨论（自由提出意见）与少数服从多数相结合 | | |
| 协作过程评价 | 三人协作效率高，沟通顺畅愉快，对分工内容尽心完成，  联调整合时能相互帮助、共同面对问题。  在三人的齐心努力下，本项目较好地达成了目标，效果良好。 | | |

## 2 软硬件平台与算法

### 2.1 微控制器介绍

LPC2132是基于一个支持实时仿真和嵌入式跟踪的32位ARM7TDMI-STM CPU的微控制器，并带有64KB的嵌入的高速Flash存储器。128位宽度的存储器接口和独特的加速模式使32位代码能够在最大时钟速率下运行。对代码规模有严格控制的应用可使用16位Thumb®模式将代码规模降低超过30%，而性能的损失却很小。

较小的封装和极低的功耗使LPC2132可理想地用于小型系统中，如访问控制和POS机。宽范围的串行通信接口和片内16KB的SRAM使LPC2132非常适用于通信网关、协议转换器、软modem、声音辨别和低端成像，为它们提供巨大的缓冲区空间和强大的处理功能。多个32位定时器、1个10位8路ADC、10位DAC、PWM通道和47个GPIO以及多达9个边沿或电平触发的外部中断使它们特别适用于工业控制和医疗系统。

LPC2132具有以下特性：

* 小型LQF64封装的32位ARM7TDMI-S微控制器。
* 16KB片内静态RAM。
* 片内Boot装载软件实现在系统/在应用中编程（ISP/IAP）。扇区擦除或整片擦除的时间为400ms，1ms可编程256字节。
* EmbeddedICE®RT和嵌入式跟踪接口可实时调试（利用片内RealMonitor软件）和高速跟踪执行代码。
* 1个8路10位A/D转换器共包含16个模拟输入，每个通道的转换时间低至2.44us。
* 1个10位D/A转换器，可提供不同的模拟输出。
* 2个32位定时器/计数器（带4路捕获和4路比较通道）、PWM单元（6路输出）和看门狗。
* 实时时钟具有独立的电源和时钟源，在节点模式下极大地降低了功耗。
* 多个串行接口，包括2个16C550工业标准UART、2个高速I2C接口（400kbit/s）、SPITM和SSP（具有缓冲功能，数据长度可变）。
* 向量中断控制器。可配置优先级和向量地址。
* 多达47个5V的通用I/O口（LQFP64封装）。
* 9个边沿或电平触发的外部中断引脚。
* 通过片内PLL可实现最大为69MHz的CPU操作频率，PLL的稳定时间为100us。
* 片内晶振频率范围：1~30MHz。
* 2个低功耗模式：空闲和掉电。
* 可通过个别使能/禁止外部功能和降低外部时钟来优化功耗。
* 通过外部中断将处理器从掉电模式中唤醒。
* 单个电源供电，含有上电复位（POR）和掉电检测（BOD）电路：-CPU操作电压范围：3.0~3.6V（3.3V+/- 10%），I/O口可承受5V的最大电压。

### 2.2 软件开发平台介绍

#### 2.2.1 软件开发平台

Keil MDK-ARM（旧称RealView MDK）开发工具源于德国Keil公司，被全球上百万的嵌入式开发工程师验证和使用，是ARM公司目前推出的针对各种嵌入式处理器的软件开发工具。

Keil MDK集成了业内最领先的技术，包括uVision3、uVision4、uVision5集成开发环境与ARM编译器。支持ARM7、ARM9、Cortex-M0、Cortex-M0+、Cortex-M3-、Cortex-M4、Cortex-R4内核处理器。

Keil MDK可以自动配置代码，集成Flash烧写模块，强大的Simulation设备模拟、性能分析等功能，与ARM之前的工具包ADS等相比，ARM编译器的最新版本可将性能改善超过20%以上。

Proteus是著名的EDA软件，从原理图布局、代码调试到单片机与外围电路协同仿真，一键切换到PCB设计，真正实现了从概念到产品的完整设计。迄今为止是世界上唯一将电路仿真软件、PCB设计软件和虚拟模型仿真软件三合一的设计平台，其处理器模型支持8051、HC11、PIC10/12/16/18/24/30/DsPIC33、AVR、ARM、8086和MSP430等，2010年增加Cortex和DSP系列处理器，并持续增加其他系列处理器模型。在编译方面，它也支持IAR、Keil和MATLAB等多种编译。

Proteus具有以下几个优势：1. 提供软件调试功能。2. 提供丰富的外围接口器件及其仿真。如RAM、ROM、键盘、马达等等。3. 提供丰富的虚拟仪器，利用虚拟仪器在仿真过程中可以测量外围电路的特性。4. 具有强大的原理图绘制功能。5. 支持与Keil的联合仿真。

#### 2.2.2 与目前最先进平台功能的对比

|  |  |  |
| --- | --- | --- |
| Feature | RVDS | MDK |
| 编译器 | RVCT | RVCT（只支持部分选项） |
| 支持ARM内核 | 全部ARM内核 | ARM7、ARM9、Cortex |
| 开发环境 | Eclipse | uVision |
| 调试软件 | uVision Debugger | RealView Debugger |
| 仿真器 | RealView ICE | ULINK2 & ULINK-ME |
| Flash烧写 | √ | √ |
| 多核调试 | √ | × |
| 外设虚拟 | √ | × |
| Trace支持 | √ | × |
| DSP & Cache | √ | × |
| Linux调试 | √ | × |

其他功能区别：

1. 器件库：RVDS只能选择具体的ARM Core，MDK只能通过器件库的方式选择具体的ASIC集成芯片。因此MDK在调试时不需要做多余的工作就可以观察到芯片的外设，但只能支持其中一部分的芯片，并且器件库需要不断的升级和维护。RVDS可以支持所有的ARM ASIC芯片，可以通过观察BCD文件达到观察芯片外设的功能，且这个文件可以保存。

2. 初始化代码：MDK可以根据所选择的具体ASIC芯片，自动生成相关的初始化代码，包括基本外设的初始化、堆栈的初始化、头文件的定义。而RVDS没有这个功能。

3. 调试：MDK支持条件断点、数据访问断点、执行断点、单步、观察变量、符号表加载等常见的功能。RVDS的调试功能比它丰富。

因此，MDK仅适合MCU的开发，适合于程序规模和复杂度小的工控行业。当程序规模和复杂度增加时，RVDS可以提供更快捷、更细致、更丰富的功能。

### 2.3 矩阵处理算法

#### 2.3.1 矩阵求逆算法

在选择矩阵求逆算法时，了解了伴随矩阵法、LU分解法和高斯-约旦消元法，最后选择了高斯-约旦消元法。

使用伴随矩阵法时，需要分别算出其伴随矩阵和行列式，再算出逆矩阵。此法的时间复杂度主要来源于计算行列式，由于计算行列式的函数为递归形式，需要占用栈空间。且此法需要进行较多乘法与除法，稳定性不佳。

LU分解法中，考虑到A-1=(LU)-1=U-1L-1，需要将A分解为LU后，对L和U分别求逆，再相乘。此法主要是分解过程耗时，求解三角矩阵的时间复杂度是O(n2)，且分解的算法也非常复杂。为了求A的逆，还需要求L与U的逆，过于曲折。

高斯-约旦消元法中，把A和单位矩阵I同行放置，形成增广矩阵，对A进行加减消元使A化成单位矩阵。高斯-约旦消元法相对数值稳定，求解过程清晰，使用的乘法运算较少，占用存储空间小，代码量较合适。

综合考虑稳定性、算法实现难度、代码量后，我们选择了**高斯-约旦消元法**。

#### 2.3.2 矩阵求特征值算法

在选择矩阵求特征值的算法时，了解了求解矩阵方程法、雅可比法和QR变换法，最后选择了雅可比法。

从原始矩阵出发，求出其特征多项式及其根，即得到矩阵的特征值，这就是求解矩阵方程法的过程。但高次多项式求根问题尚有困难，而且重根的计算精度较低。另外，原始矩阵求特征多项式系数的过程，对舍入误差非常敏感，对最终结果影响很大。

雅可比正交相似变换，适用于实对称矩阵求特征值，且计算结果很准确；但用于非对称矩阵时收敛效果并不好。

QR正交相似变换，一般认为对任意中小型矩阵都可求特征值，实际上最适合非对称矩阵，计算结果准确。但对称矩阵用QR正交相似变换时，收敛效果反而不理想。

考虑到雅可比法运行较稳定，算法实现较容易，最终决定采用**雅可比法**。

## 3 设计、检查与调试

### 3.1 设计技巧

**（1）自顶向下进行设计。**

根据项目目标，使用流程图描述项目流程，再根据项目流程进行任务分工。

各个模块以函数形式实现功能，函数之间的接口需要提前进行设计，以确保联合调试时的效率。

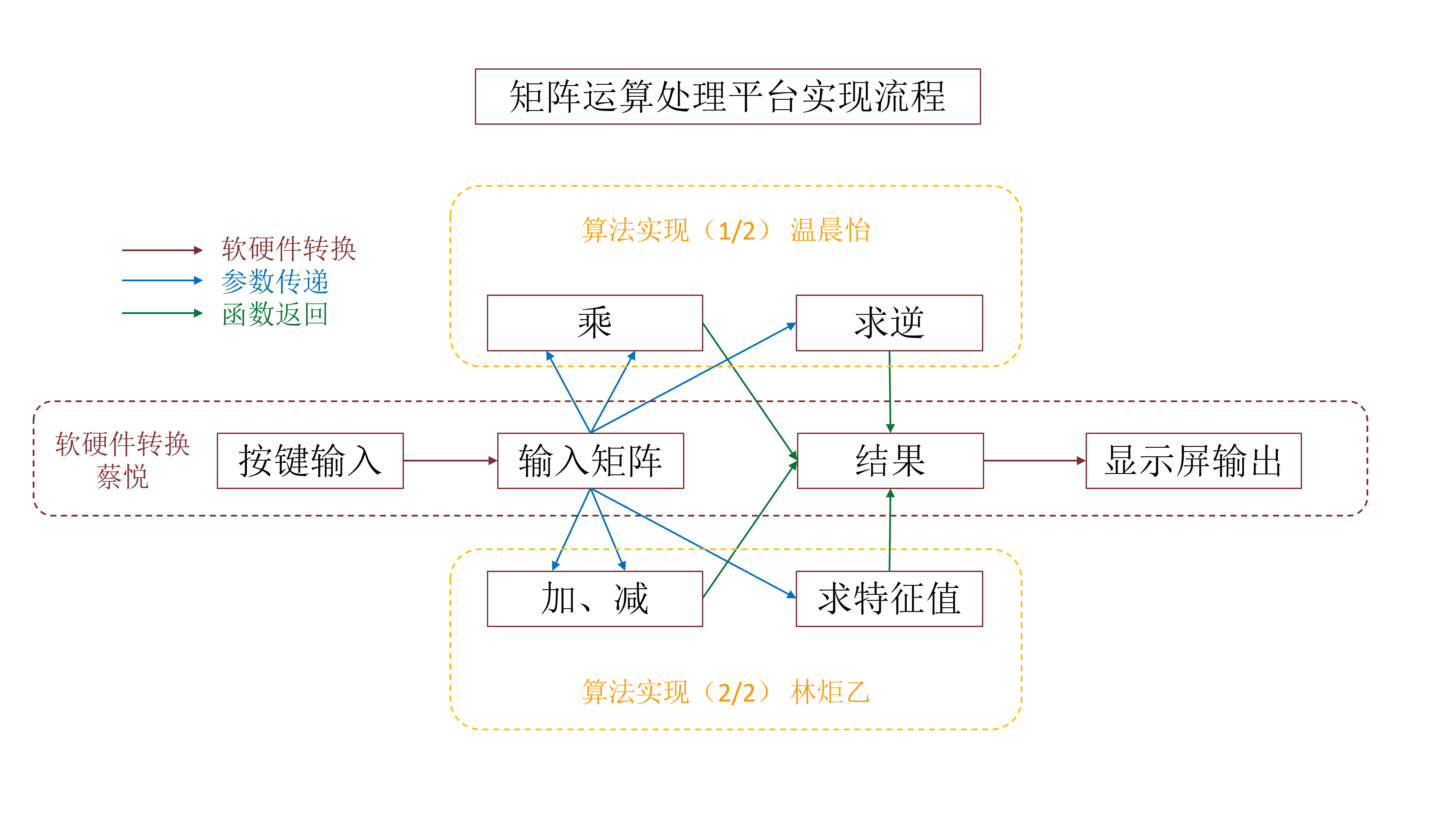


图1 项目实现流程

**（2）巧用全局变量与头文件进行工程组织。**

编写C语言代码时，设置适合的全局变量供汇编子程序调用。同时全局变量可用于函数参数传递、结果传递等，如运算结果可存储在全局变量中以便转化为显示屏的输出。

使用头文件进行工程文件的组织，通过#ifndef、#define、#endif来防止重复定义函数或未定义函数的现象出现。

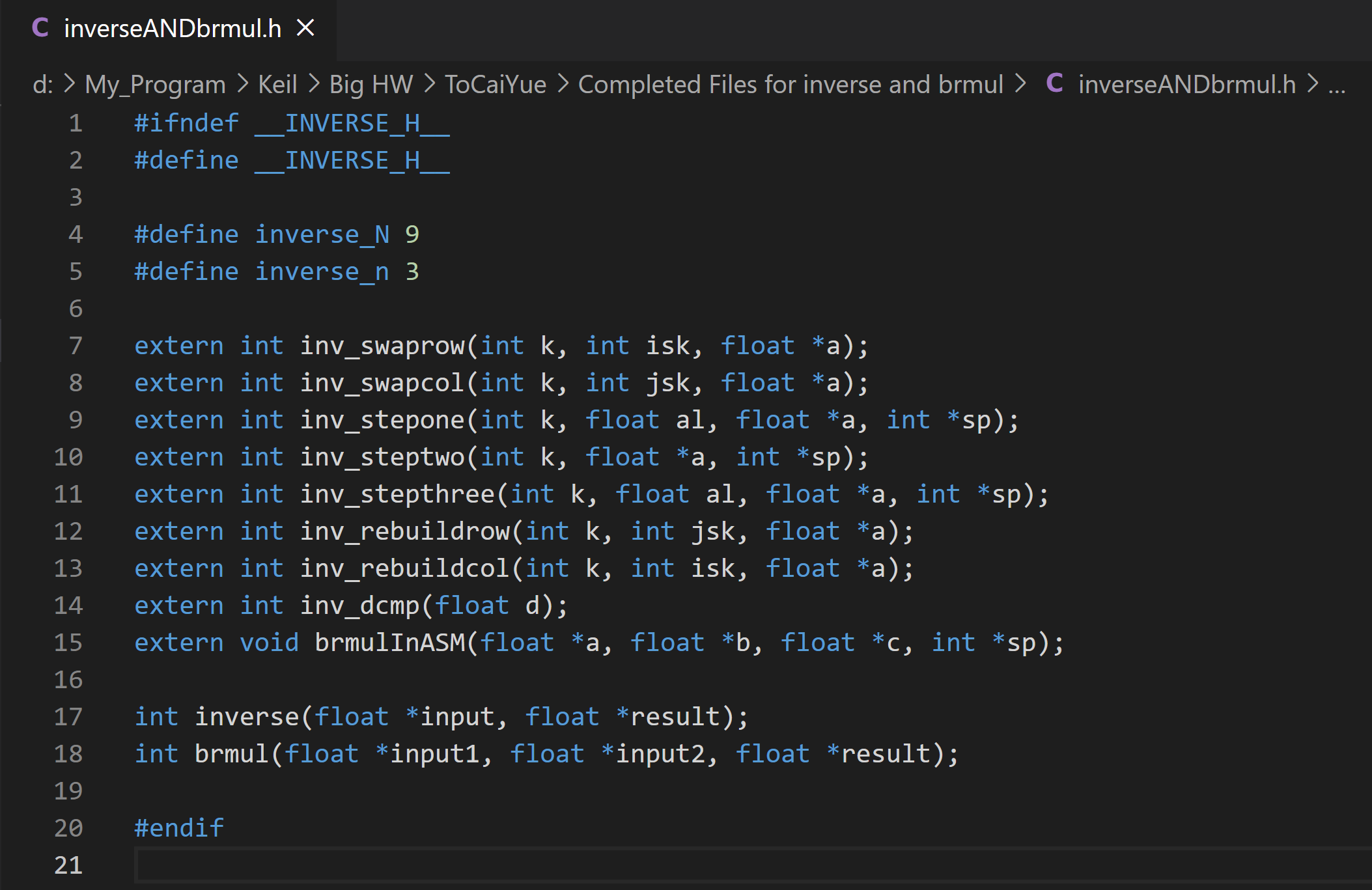


图2 求逆、乘法相关函数的头文件

### 3.2 检查技巧

#### 3.2.1检查表

把要检查的事项先罗列在一张表中，在实施代码审查的时候，依据所列出的检查项逐一确认代码中是否存在对应的问题。

检查表中的检查项通常依据编码规范来制定，如变量是否初始化，命名是否一致，字符串是否正确解码，逻辑操作符是否正确，、{}对是否一致等。也根据以往的软件代码中常见的错误适时地调整。列好清单,提高在代码审查过程中发现的缺陷个数，帮助团队成员更好更快的进行代码审查。

检查表内容主要有

​ 1)常规项，比如代码能否正常运行、是否简单易懂、是否存在多余或者重复的代码、是否尽可能的模块化、是否有被注释掉的代码等等。

​ 2)安全。是否所有的输入输出值都进行了检查、无效的参数值是否进行了检查并编码、使用第三方工具返回的错误是否被捕捉等。

​ 3)文档。是否有注释并描述代码意图、是否所有函数都有注释、第三方库的使用和函数是否有文档、非常规行为和边界处理是否有描述等。

​ 4)测试。代码是否可以测试、是否可以被理解、是否检查数组的“越界”错误、是否有可以被已存在API替代的测试代码等。

#### 3.2.2 测试样例

**求特征值测试集**

测试1

[3.0, 2.0, 4.0, 2.0,0.0, 2.0,4.0 ,2.0 ,3.0]

目标结果：特征值 λ1 = 8 , λ2 = λ3 = -1

测试2

[2.0, -2.0, 0.0, -2.0, 1.0, -2.0, 0.00, -2.0 ,0.00]

目标结果：特征值 λ1 = 4 , λ2 = -2, λ3 = 1

测试3

[1.0 ,2.0, 3.0 ,2.0, 1.0, 3, 3, 3, 6.0]

目标结果：特征值 λ1 = 9 , λ2 = 0, λ3 = -1

**求逆测试集**

测试1

[1.0 ,0.0 ,1.0 ,1.0 ,1.0 ,2.0 ,3.0 ,4.0 ,2.0]

测试2

[1.0, -1.0, 3.0, 2.0, -1.0, 4.0, -1.0, 2.0, -4.0]

测试3

[1.0, 2.0, 3.0, 4.0, 0.0, 1.0, 2.0, -5.0, 1.0]

测试4

[1.0 ,0.0 ,1.0 ,1.0 ,1.0 ,2.0 ,3.0 ,4.0 ,2.0]

该矩阵为奇异矩阵，不可求特征值

#### 3.2.3代码覆盖率

用pip 安装gcovr, 在官网下载安装了Mingw的gcc, 然后用命令行

g++ -fprofile-arcs -ftest-coverage -fPIC -O0 needTest.cpp -o program

./program

gcovr -r .

就可以产生代码覆盖率的测试结果.

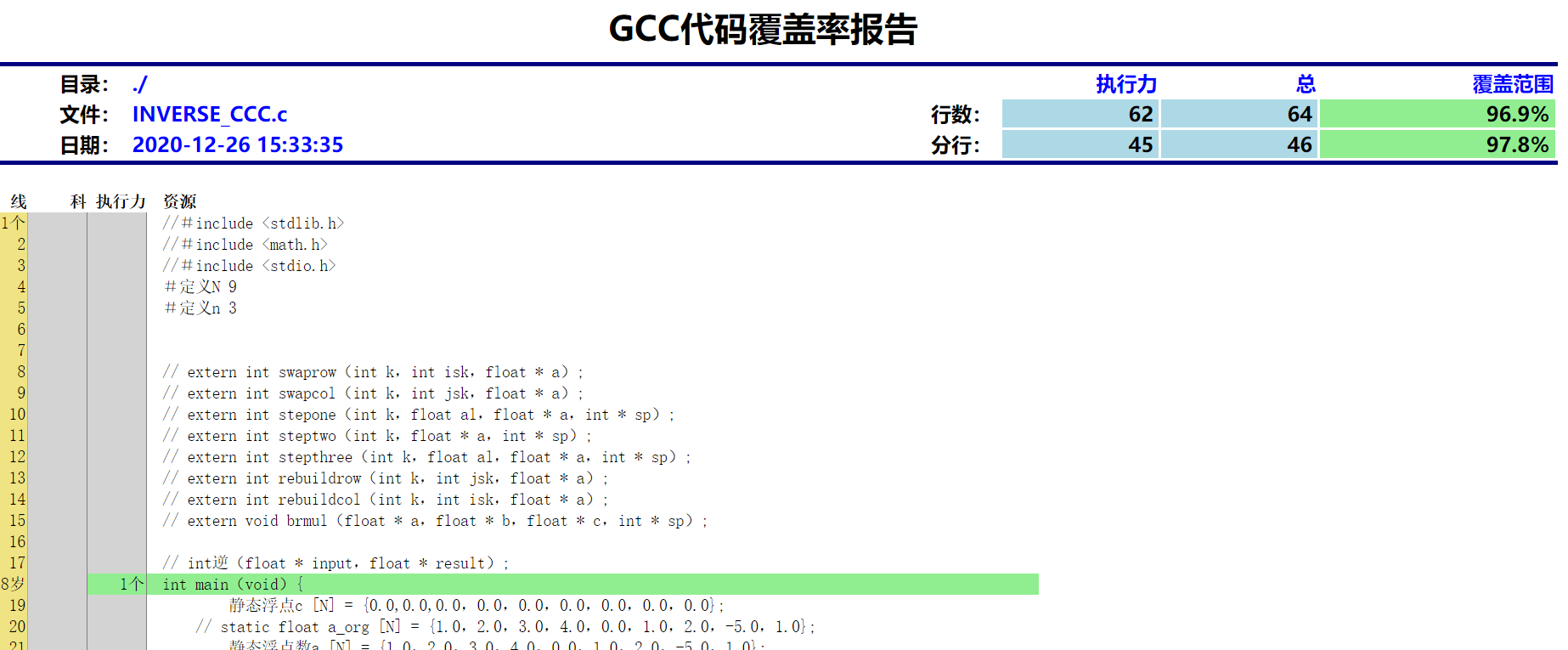


图3求逆代码覆盖率报告

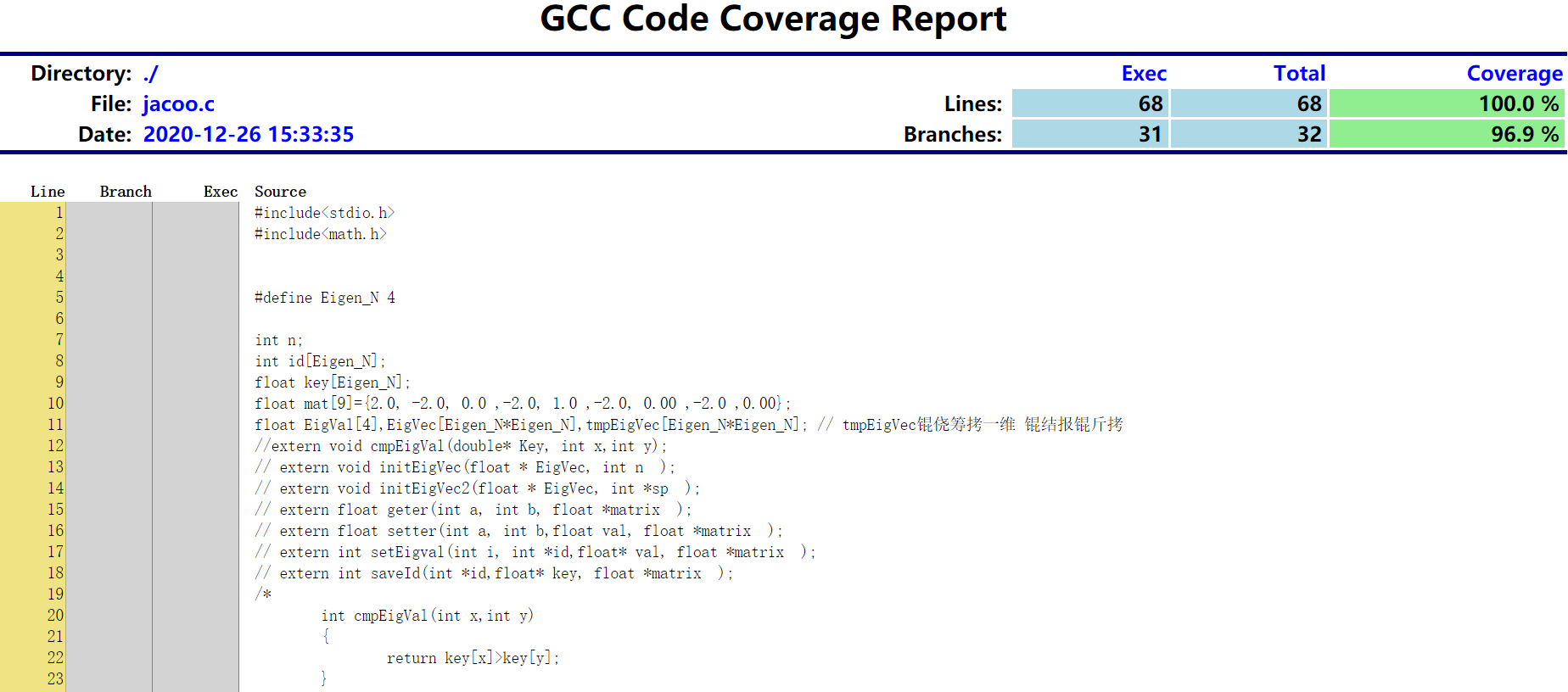


图4求特征值代码覆盖率报告

### 3.3 调试技巧

#### 3.3.1单步调试

当输出与自己的预期不符合时，在不确定是否正确的**代码较短**的情况下，可以将原本的输入按照代码**进行单步调试**，看看哪一步不符合自己的预期，比较慢但是可以准确地找出出错的地方。如下图所示:

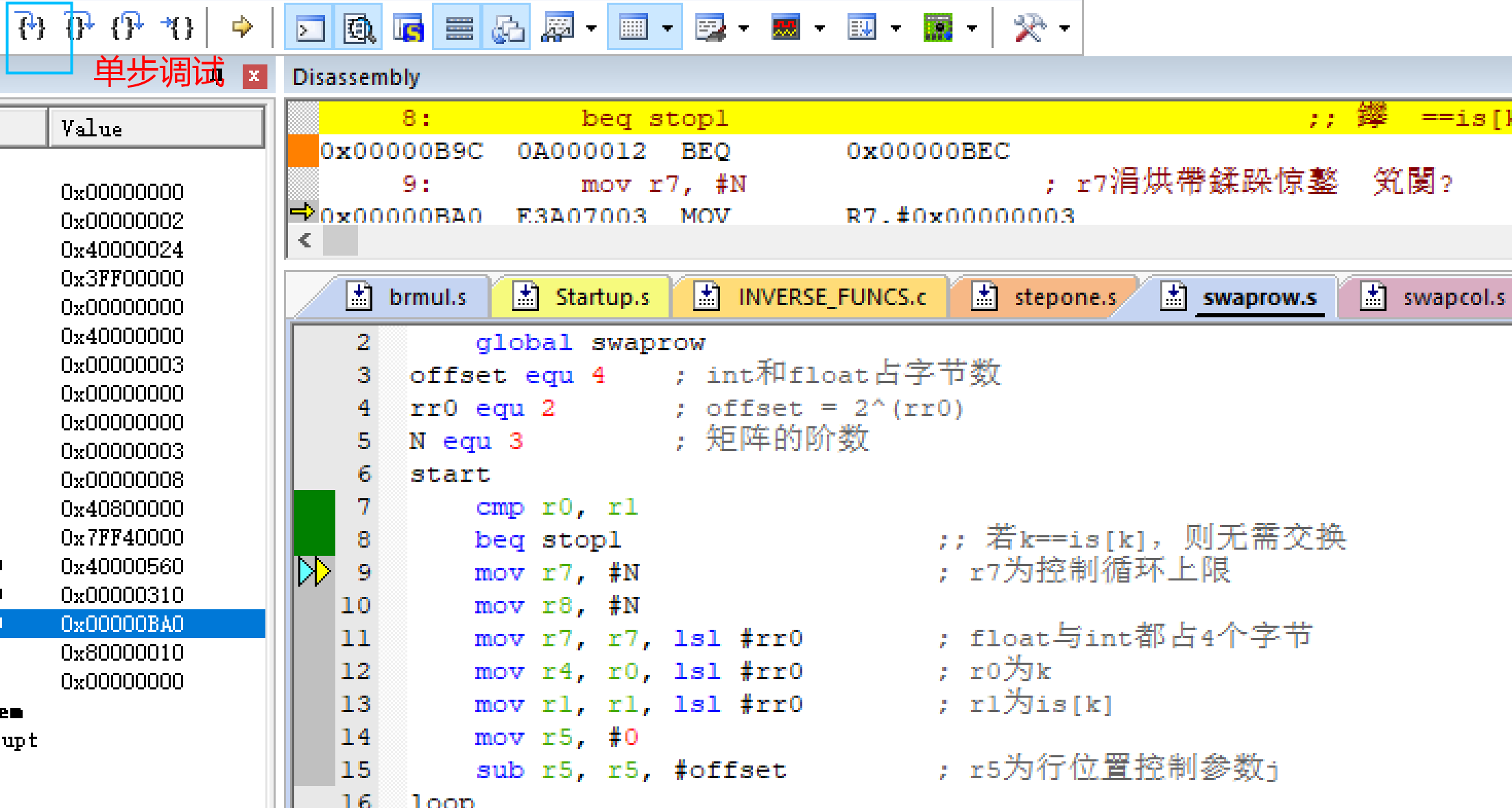


图5 单步调试

#### 3.3.2.交互性强

在**代码较长**时，可以在不同的部分后面或中间添加一个指示语句，比如C语言中在每个if分支中直接printf一个提示信息，表示本次运行到了该部分代码，也**可以直接将部分变量输出**，可以比较快地找到长代码中出错的步骤。

#### 3.3.3. assert宏

C语言可以**使用assert宏来判断某一数据是否出错**，它的作用是：如果它的条件为false，则输出一条错误信息并终止程序。在编程中如果有需要也可以对assert进行自定义，如当其发生错误时不终止程序，而是进入一个调试程序。在debug完成后发布代码时，可以利用ifdef对assert宏进行开启和关闭，而不需要一行行将assert删除。

#### 3.3.4.断点

通过直接运行至断点处，快速查看某一模块的功能是否正确。

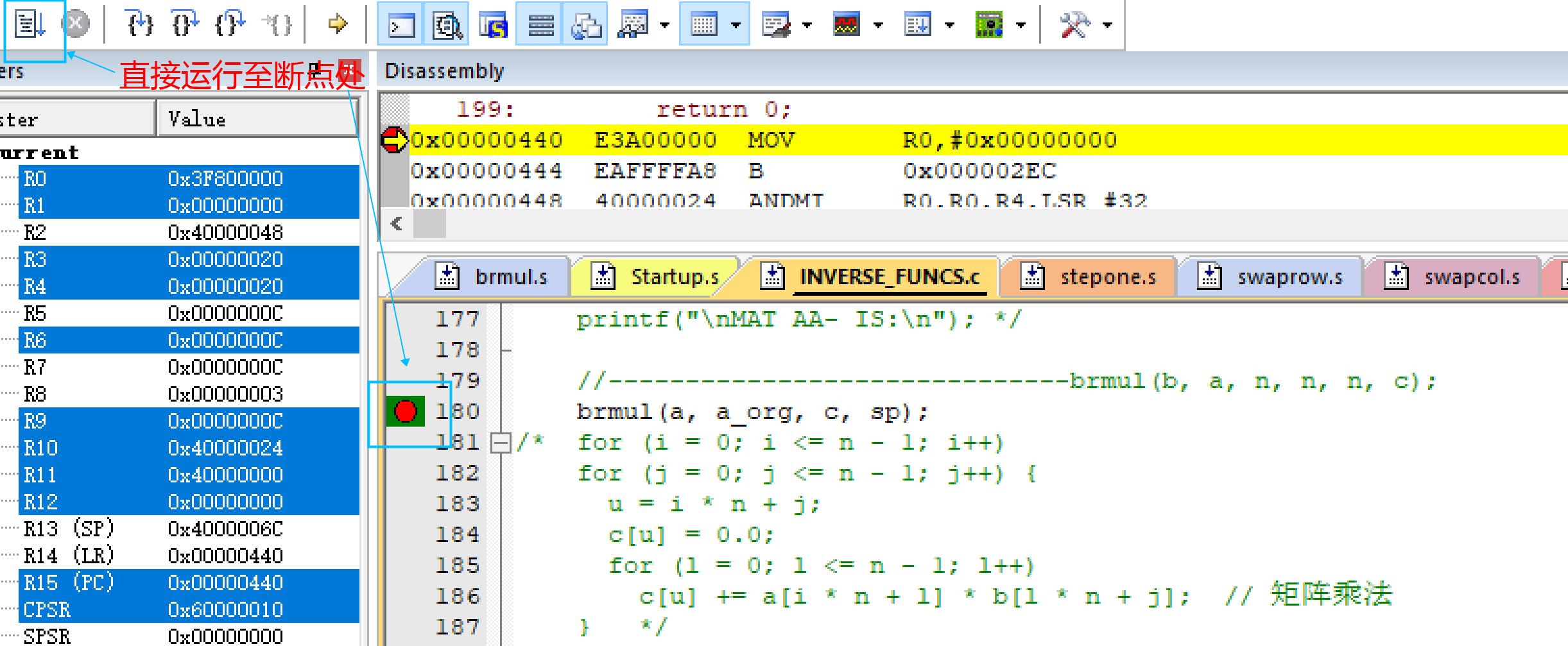


图 6运行至断点

#### 3.3.5. 寄存器

通过寄存器窗口查看寄存器状态

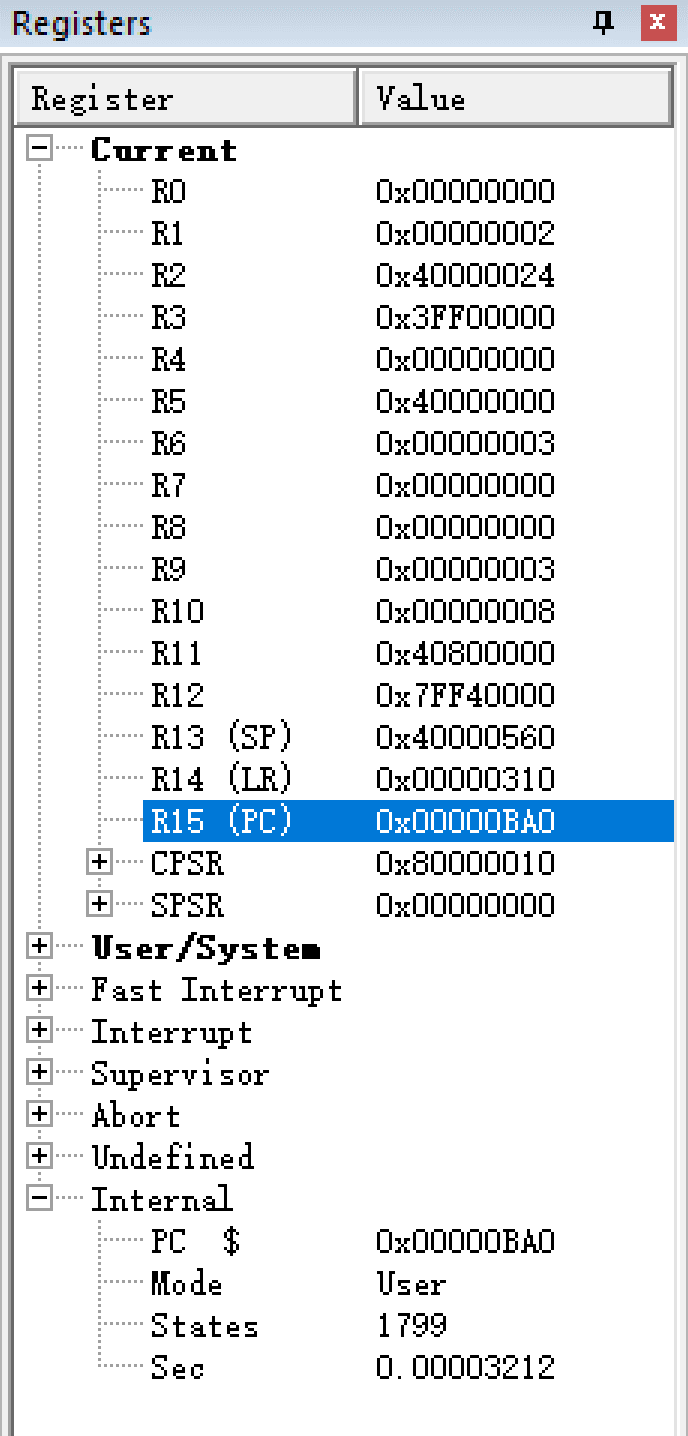


图7 寄存器窗口

#### 3.3.6. 存储区

调试汇编函数时，查看存储区窗口，观察读写操作

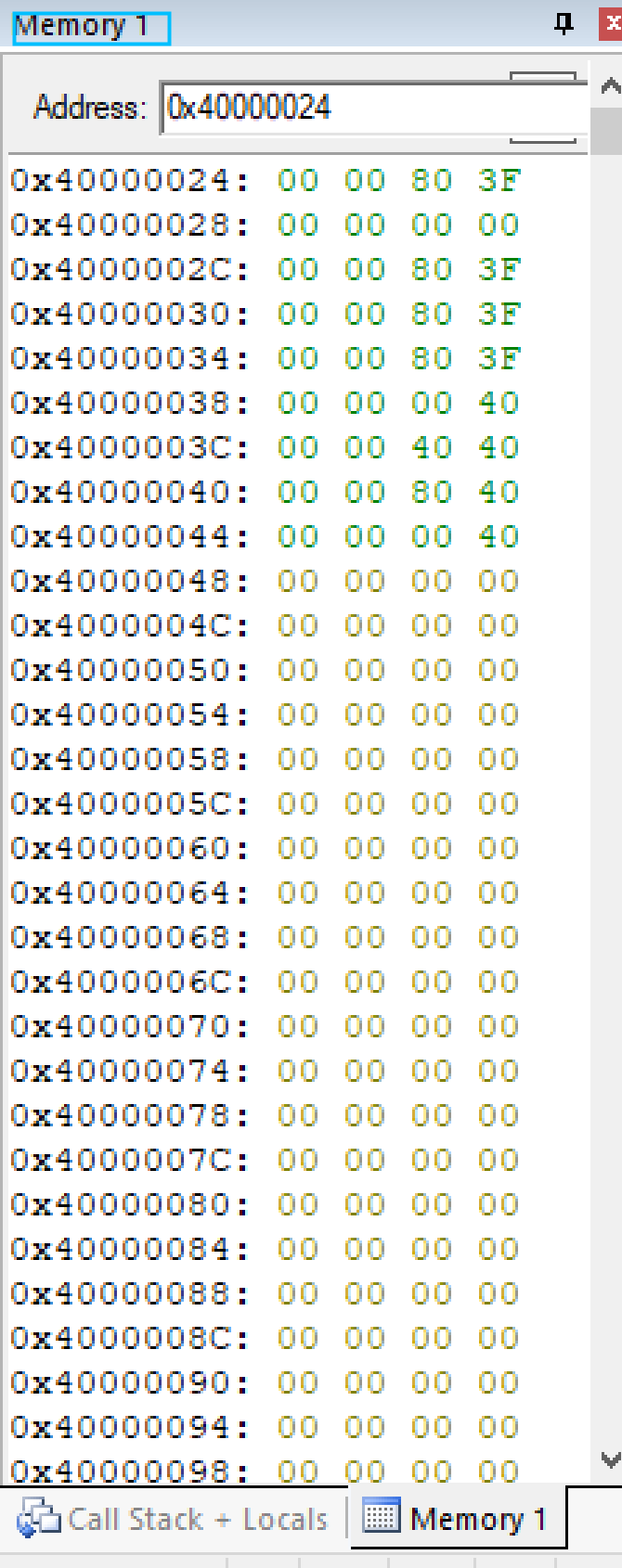


图8存储区窗口

#### 3.3.7.变量窗口

调试C语言的时候，查看变量窗口确定各个变量当前的状态。

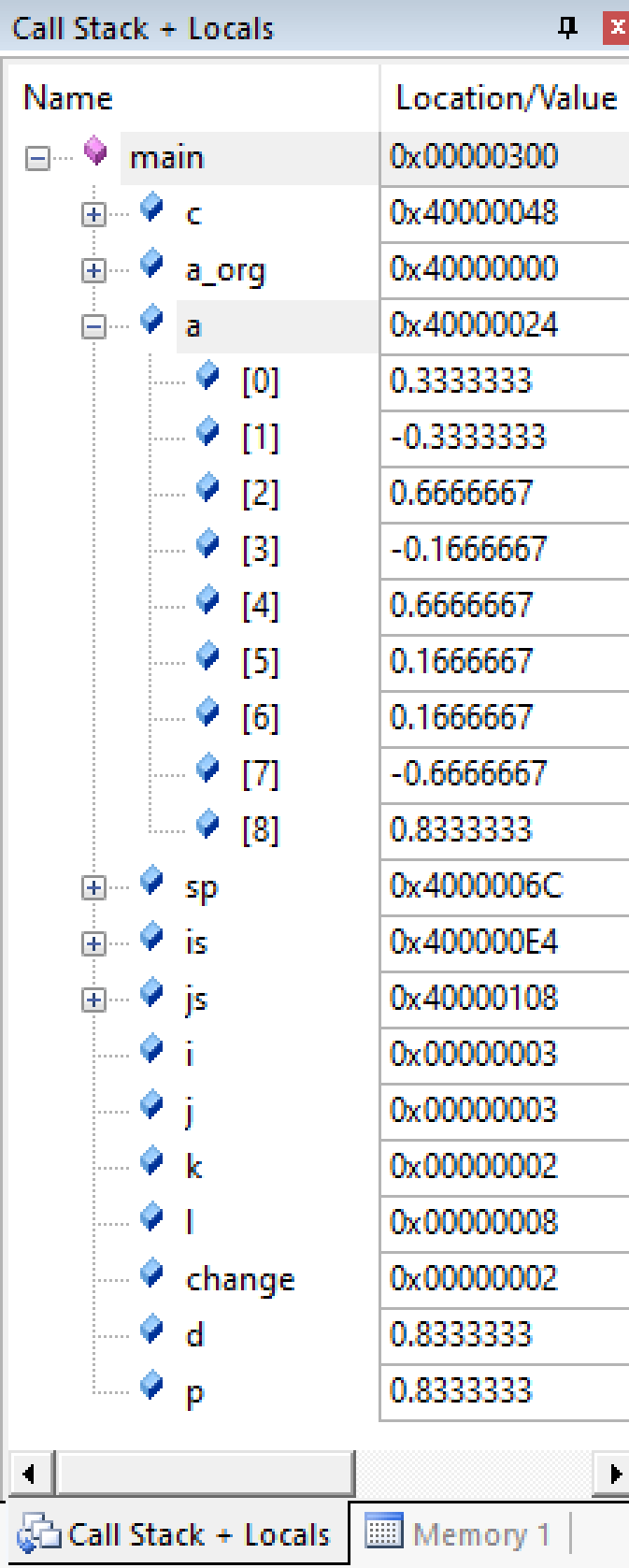


图9 变量区

## 4 项目成果展示与项目总结

### 4.1 项目功能与性能

#### 4.1.1 程序实现结果

|  |
| --- |
| **计算器整体功能介绍** |
| 目前支持的功能：  ① 支持四位以内的浮点数（0.100~9999，小数点前后加起来共四位）输入与显示。  ② 支持矩阵加、减、乘以及求逆、求特征值五种运算。  ③ 可以使用“Backspace”撤销上一步，如删除上一位数字，返回上一个数字的输入，撤销操作符输入等。  ④ 可以使用“C”进行状态重置，重新开始一轮输入与运算。  目前存在的一些缺陷与功能缺失：  ① 不支持负数输入。  ② 没有游标显示，只能通过显示内容判断目前的输入进度。  ③ 不支持形似0.001、1.02这类小数点后接0的数字输入。  ④ 操作符没有具体的符号显示在显示屏上，仅用数字进行表示：1-加，2-减，3-乘，4-求逆，5-求特征值。  ⑤ 没有对显示屏进行checkbusy。  ⑥ 由于浮点数存储的精度问题，有些小数会有偏差，如输入9.99时会存储并显示为9.989。 |
| **Proteus整体电路图** |
|  |
| **矩阵键盘** |
|  |
| 程序使用了4\*5的矩阵键盘，并使用反转法获取按下的键的信息。各个键位功能如图中标注所示。 |
| **矩阵及操作符的顺序输入（使用说明）** |
| 考虑到矩阵的复杂性，计算器的输入方式比较严苛：  ① 从第一个矩阵的第一个数字开始输入，数字输入完毕后按下“OK”键确定当前数字并跳转到下一数字，直到第一个矩阵的九个数字全部输入完毕。  ② 然后要求输入一个操作符。如果为双目操作符则开始第二个矩阵的输入。  ③ 第二个矩阵输入完毕/输入单目操作符之后，要求输入“=”。之后显示对应的计算结果。  ④ 按下“C”键重置状态，回到①。 |
| **显示** |
| C:\Users\vasua\AppData\Roaming\Tencent\Users\1832472182\QQ\WinTemp\RichOle\V7Y1]14A0Q(4`DOLU`Q~WWE.png |
| 可以看到显示屏被大致划分成了四个区域：左上角显示第一个矩阵，右上角显示第二个矩阵，左下角显示输出的结果（如果是求特征值只会输出三个数字），右下角显示选择的操作符。 |
| **功能展示** |
| 加法 |
|  |
| 减法 |
|  |
| 乘法 |
|  |
| 求逆 |
|  |
| 求特征值 |
|  |

#### 4.1.2 求逆性能测试

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **速度测试** | | | | | | |
| 运行时间 | 测试1 | | 测试2 | 测试3 | | 测试4 |
| 带汇编的程序 | 114.98us | | 112.92us | 111.80us | | 116.68us |
| 纯C程序 | 99.97us | | 97.83us | 95.88us | | 101.47us |
| **速度测试——结论与分析**  **结论：**  带汇编的程序与纯C程序的求解速度都在0.1ms左右，能很好地满足学生日常学习、科研的矩阵处理需求。（没有查询到计算三阶矩阵的逆的最快性能，可能是因为三阶矩阵太简单了。）  纯C程序比带汇编的程序快了接近15%。汇编没有起到优化作用，甚至因为调用外部汇编函数、代码不紧凑等原因导致了更大的时间开销。  **分析：为何汇编没有优化甚至还起了反作用？**  ——这一部分虽然用汇编替代了核心的循环乘法等内容，但汇编代码的编写思路依旧遵循的是原本C程序的思路，并没有进行代码流程上的优化；且由于计算内容为浮点运算，我们只能调用外部的浮点库进行运算，对于外部浮点库的性能我们无法把控。因此这一部分的汇编代码并没有起到优化作用。  ——因为带汇编的函数多次从C跳到汇编函数中、从汇编函数跳到外部浮点运算库中，造成了额外跳转开销；同时自己编写的汇编代码或许也不如C编译器编译得到的汇编精炼、紧凑，这也导致了时间的延长。  **反思：**  考虑到求逆算法的实现过程，改变汇编编写思路并不容易。但如果我们能使用FPU运算单元以硬浮点的方式进行浮点运算处理，则算法核心的浮点运算速度将大大提升。（遗憾的是我们本次选择的微控制器并没有FPU单元。）  虽然汇编并没有带来项目功能的性能提升，但随着汇编代码量的增加，我们对汇编的熟悉、理解也进一步加深。运用C与汇编联调的方式正确地实现了功能也是一种锻炼和经验积累。在处理外部浮点库、比对编译器给出的汇编代码时，我们还能学习成熟汇编代码的结构与语句，对增强自身的代码编写能力有很大的益处。 | | | | | | |
| **精度测试** | | | | | | |
| 测试1  [1.0 ,0.0 ,1.0 ,1.0 ,1.0 ,2.0 ,3.0 ,4.0 ,2.0] | | | | | | |
| MATLAB结果 | | 带汇编的程序 | | | 纯C程序 | |
|  | |  | | |  | |
| 测试2  [1.0, -1.0, 3.0, 2.0, -1.0, 4.0, -1.0, 2.0, -4.0] | | | | | | |
| MATLAB结果 | | 带汇编的程序 | | | 纯C程序 | |
|  | |  | | |  | |
| 测试3  [1.0, 2.0, 3.0, 4.0, 0.0, 1.0, 2.0, -5.0, 1.0] | | | | | | |
| MATLAB结果 | | 带汇编的程序 | | | 纯C程序 | |
|  | |  | | |  | |
| 测试4  [1.0 ,0.0 ,1.0 ,1.0 ,1.0 ,2.0 ,3.0 ,4.0 ,2.0]  该矩阵为奇异矩阵，不可求逆 | | | | | | |
| MATLAB结果 | | 带汇编的程序 | | | 纯C程序 | |
|  | | 无法判断其为奇异矩阵，  会持续计算，  但计算结果为奇异矩阵，  可人工判断。 | | | 无法判断为奇异矩阵，  结果与带汇编的程序一致。 | |
| **精度测试——结论与分析**  **结论：**  1.当矩阵为可逆矩阵时，我们的求逆功能准确度好，精度较高（可达8位小数精度）。若保留前四位，则结果与matlab计算结果一致。带汇编的程序与纯C程序的计算结果没有区别。  2.当矩阵为不可逆矩阵时，我们的求逆功能无法正确判断，会持续计算，但计算结果为奇异矩阵，可人工判断。但不可逆矩阵并不会使程序崩溃，算法稳定性较好。  该计算结果的精度与准确度都非常可观，足以满足学生的日常学习、科研需求。 | | | | | | |
| **求逆性能测试——总结**  无论是运算速度、精度、正确性还是稳定性，求逆这一功能都可以很好地满足学生的日常学习、科研需求，达到了项目目标。  但遗憾的是C与汇编联调并没有带来性能上的优化，若能使用硬浮点方式进行计算将使求逆性能进一步提升。 | | | | | | |

#### 4.1.3 求特征值性能测试

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **速度测试** | | | | | | |
| 运行时间 | 测试1 | | 测试2 | 测试3 | |
| 带汇编的程序 | 7.947ms | | 8.12ms | 7.05ms | |
| 纯C程序 | 7.89ms | | 8.07ms | 7.00ms | |
| **速度测试——结论与分析**  **结论：**  带汇编的程序与纯C程序的求解速度都在7-8ms左右，能很好地满足学生日常学习、科研的矩阵处理需求。  该计算结果的精度与准确度都非常可观，足以满足学生的日常学习、科研需求。纯C程序比带汇编的程序快了接近0.05ms。汇编没有起到优化作用，甚至因为调用外部汇编函数、代码不紧凑等原因导致了更大的时间开销。  **分析：为何汇编没有优化甚至还起了反作用？**  ——这一部分虽然用汇编替代了核心的循环求迹等内容，但汇编代码的编写思路依旧遵循的是原本C程序的思路，并没有进行代码流程上的优化；且由于计算内容为浮点运算，我们只能调用外部的浮点库进行运算，对于外部浮点库的性能我们无法把控。因此这一部分的汇编代码并没有起到优化作用。  ——因为带汇编的函数多次从C跳到汇编函数中、从汇编函数跳到外部浮点运算库中，造成了额外跳转开销；同时自己编写的汇编代码或许也不如C编译器编译得到的汇编精炼、紧凑，这也导致了时间的延长。  **反思：**  考虑到特征值算法的实现过程，改变汇编编写思路并不容易。但如果我们能使用FPU运算单元以硬浮点的方式进行浮点运算处理，则算法核心的浮点运算速度将大大提升。（遗憾的是我们本次选择的微控制器并没有FPU单元。）  虽然汇编并没有带来项目功能的性能提升，但随着汇编代码量的增加，我们对汇编的熟悉、理解也进一步加深。运用C与汇编联调的方式正确地实现了功能也是一种锻炼和经验积累。在处理外部浮点库、比对编译器给出的汇编代码时，我们还能学习成熟汇编代码的结构与语句，对增强自身的代码编写能力有很大的益处。 | | | | | | |
| **精度测试** | | | | | | |
| 测试1  [3.0, 2.0, 4.0, 2.0,0.0, 2.0,4.0 ,2.0 ,3.0] | | | | | | |
| MATLAB结果 | | 带汇编的程序 | | | 纯C程序 | |
|  | |  | | |  | |
| 测试2  [2.0, -2.0, 0.0 ,-2.0, 1.0 ,-2.0, 0.00 ,-2.0 ,0.00] | | | | | | |
| MATLAB结果 | | 带汇编的程序 | | | 纯C程序 | |
|  | |  | | |  | |
| 测试3  [1.0 ,2.0, 3.0 ,2.0, 1.0 , 3, 3, 3, 6.0] | | | | | | |
| MATLAB结果 | | 带汇编的程序 | | | 纯C程序 | |
|  | |  | | |  | |
| **精度测试——结论与分析**  **结论：**  当矩阵为普通矩阵时，我们的求特征值功能准确度好，精度较高（可达8位小数精度）。若保留前四位，则结果与matlab计算结果一致。带汇编的程序与纯C程序的计算结果没有区别。  该计算结果的精度与准确度都非常可观，足以满足学生的日常学习、科研需求。 | | | | | | |
| **求特征值性能测试——总结**  无论是运算速度、精度、正确性还是稳定性， 求特征值这一功能都可以很好地满足学生的日常学习、科研需求，达到了项目目标。  但遗憾的是C与汇编联调并没有带来性能上的优化，若能使用硬浮点方式进行计算将使求特征值性能进一步提升。 | | | | | | |

### 4.2 各组员成果与代码量

|  |  |  |  |
| --- | --- | --- | --- |
| 组员 | 分工 | 成果 | 代码量 |
| 温晨怡 | 求逆、乘法 | 功能正确，速度、精度、正确性、稳定性满足要求 | C语言200行，  汇编327行 |
| 蔡悦 | 输入、输出 | 完成了基本功能，可以满足基础应用 | C语言300行，汇编约400行 |
| 林炬乙 | 求特征值、加减和求迹 | 功能正确，速度、精度、正确性、稳定性满足要求 | C语言200行  汇编247行 |

### [参考文献]

[1] ARM. (2001). *ARM7TDMI-S (Revision: r4p3) Technical Reference Manual.*

[2] Philips Semiconductors. (2004). *LPC2131/2132/2138 Preliminary data sheet.*

[3] ARM. (2013). *Cortex-A7 Floating-Point Unit (Revision: r0p5) Technical Reference Manual.*

[4] AMPIRE 128X64显示器使用手册及常用代码

## 5 个人工作总结

### 5.1 分工与完成情况

#### 5.1.1 个人分工内容

项目目标为实现对三阶矩阵的加法、减法、乘法、除法、求迹、求逆、求特征值等七项功能，我需要完成对三阶矩阵的**求迹、求特征值、加法、减法**这四项功能的实现。

#### 5.1.2 个人工作完成情况

|  |  |
| --- | --- |
| 序号 | 事项 |
| 1 | 确定项目目标。确定求特征值算法。 |
| 2 | 调试软件平台——完成C与汇编函数的联调。 |
| 3 | 完成矩阵求特征值与求迹的C语言实现，共计209行。 |
| 4 | 将求迹,加法,减法功能完全转化为汇编 |
| 5 | 将求特征值算法核心部分转化为汇编，共7个汇编文件，共计215行。 |
| 6 | 设计实践报告的格式与内容。 |
| 总结 | 在本次项目中较好地完成了分工内容，与队友积极沟通、共同进步。 |

### 5.2 工作亮点

**（1）实现代码覆盖率**

第一步, 我了解到Visual studio 中有代码测试, 然而需要enterprise 企业版,我们用的是community版本.

第二步,我发现完全Linux 环境下可以直接用gcc,我因为有math.h头文件在window, 移过去还有很多依赖, 所以决定在window 使用gcov.

第三步: 我用pip 安装了gcovr, 在官网下载安装了Mingw的gcc, 然后用命令行

g++ -fprofile-arcs -ftest-coverage -fPIC -O0 needTest.cpp -o program

./program

gcovr -r .

就可以产生代码覆盖率的测试结果.

**（2）求迹,求特征值功能得以完整实现，核心运算部分基本转化为汇编代码。**

在使用C语言实现了求特征值、求迹后，将求迹功能完全转化为了汇编，求特征值算法地核心部分也转化为了汇编。总计编写汇编文件8个，汇编代码量达247行。

总体而言，求特征值功能可以在8ms左右完成，求迹功能在0.2ms左右完成，这两项功能的运算速度能够很好地满足项目目标；同时运算精度达8位小数，与matlab的实验结果相一致，求特征值算法能稳定求特征值，两项功能的精度、正确性和稳定性都得到了很好的保证。

虽然在性能分析中发现汇编并不能优化现有的程序，但随着汇编代码量的增加、不断对汇编代码进行调试，我对汇编的熟悉、掌握程度不断上升。在处理外部浮点库、比对编译器给出的汇编代码时，我们还能学习成熟汇编代码的结构与语句，对增强自身的代码编写能力有很大的益处。

### 5.3 各环节技巧

#### 5.3.1 代码编写技巧

**（1）自顶向下设计**

在设计代码时需要对算法流程有一个全局性的掌握，可以通过算法流程图理清算法执行过程。首先需要用高级语言对算法进行完整实现，再将核心运算部分转化为更底层的汇编函数。



图10 求特征值算法流程图

**（2）将具体功能封装在各个函数中，main函数调用各个函数**

每个函数互相解耦, 采用面向接口的方式编程，而不是用直接的类型引用, 内部自行依赖，对外只提供必须的接口或消息对象，那么由此即可达成较低的耦合度。模块化的设计方式能够代码结构更加清晰，代码编写难度下降，调试更加方便，可拓展性增强。且将C函数转化为汇编函数更加容易。

**（3）命令规范，注释详细**

变量和函数名最好能直观地表现出它们的功能，这将大大增加程序的可读性，而且在与他人的代码进行联调时不容易出现重名的情况。注释有利于代码维护和团队协作。我采用小驼峰式命名法,如initEigVec,setEigVal.

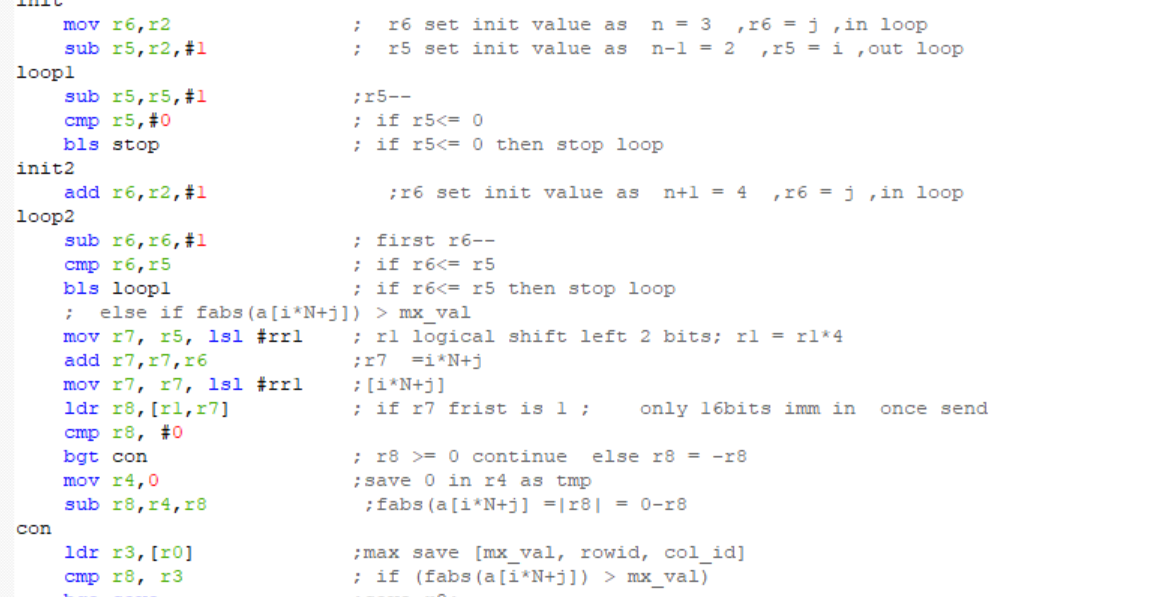


图3 大量注释的汇编代码

**(4) 杂项技巧**

汇编代码中的for循环,不要按我们一般C语言的写法从0到n,从 n到0再和0比较更好, 因为可以用立即数。

采用git 的回滚功能, 每一次小模块成功都commit一次. 程序出错后可以快速恢复。

#### 5.3.2 调试技巧

**（1）分治调试。**

每个功能都应该调试通过后再加入, 不要盲目加入到大文件，以确保其功能逻辑正确、运行稳定。每个汇编模块,用一个test来调试,这将有力避免由于一个函数错误导致程序主体运行时崩溃的情况，也降低了程序调试的难度和复杂度。

**（2）善于运用调试工具。**

Keil平台的调试功能十分齐全，包括单步调试、直接运行至断点处、跳过函数执行等功能，并且能实时监测寄存器情况、变量值、存储区情况等。通过这些功能, 我们可以在汇编中每一步观察各个变量的值, 及时发现问题并调试。

### 5.4 计划时间表与完成时间表

|  |  |  |  |
| --- | --- | --- | --- |
| 事项 | 计划时间 | 完成时间 | 是否  在计划时间内完成 |
| 确定算法与硬件平台 | 10月23日前完成 | 10月24日完成 | √ |
| 软件平台调试 | 11月6日前完成 | 11月23日完成 | × |
| 代码编写 | 11月30日前完成 | 11月30日完成 | √ |
| 代码检查 | 12月3日前完成 | 12月2日完成 | √ |
| 个人代码调试 | 12月17日前完成 | 12月10日完成 | √ |
| 代码联合调试 | 12月20日前完成 | 12月14日完成 | √ |
| 工作整理与问题分析 | 12月24日前完成 | 12月17日完成 | √ |
| 小组共同完成  “项目总结” | 12月26日前完成 | 12月22日完成 | √ |
| 个人工作总结 | 12月28日前完成 | 12月27日完成 | √ |
| **项目管理过程总结：**  **前期**进展较慢，尤其是软件平台调试花费了非常多的时间，找了很多资料也没能解决这与我们自身对微控制器编程平台和汇编语言的不熟悉有关，在课程基本过半的时候才能得心应手地使用平台；  **中期**进展迅速，在平台问题解决后，以代码实现算法并不是一件困难的事，事实上汇编语言许多都是实现C语言的一条语句, 格式较为固定,只要投入足够的耐心与时间即可；  由于项目中期基本实现了功能、完成了项目目标，**后期**的总结时间比较宽裕，但我们还是开始总结,在总结上多花时间。完成项目的过程很重要，但汲取成功经验与失败教训也非常关键。 | | | |

### 5.5 遇到的问题与解决

|  |  |  |
| --- | --- | --- |
|  | **遇到的问题** | **解决** |
| **平台配置相关** | | |
| 1 | Keil5运行c代码, 非常费cpu而且很慢. | 1）调整编译器的设置  2）在include path下面的misc control里面添加--cpp11 可以编译cpp |
| 2 | Ads 一次没装成功,后面就很麻烦, 安装到100%卡住了 | 重装一次,安装路径不能有括号, 一般安装目录默认(x86) ,但是ads是不能装在这里的 |
| **代码编写相关** | | |
| 2 | Non-RWPI Section libspace.o(.bss) cannot be assigned to PI Exec region ER\_ZI. | 设置工程属性,找到了官方文档, 用相对应的cpu和内存就可以了. |
| 3 | Error: Encountered an improper argument | 使用中文字符时keil5调试时会崩溃,使用英语注释不容易出错. |
| 4 | Debug时代码无法正常运行 | 我的debug用的cpu不是project的2132 ,是2130 ,选择对应的debug CPU |
| 5 | float compare with 0的问题 | 如果不知道精度的情况下，可以这样写： if((a>=-0.0001)&&a<=0.00001)) ,如果这个if的条件为真，就代表a是等于0的。（因为float型是保证5位小数有效的，double是保证13位小数有效的。） |

### 5.6 体会与建议

大作业总是磨人的。但就是通过这一漫长、煎熬的过程，我们才能收获成就感和成长。

这次写项目的过程对锻炼我的汇编代码能力、加深我对微控制器系统的理解有着极显著的作用。虽然汇编的部分并没有使整体性能得到提升，但编写一行行汇编代码的过程、检查浮点库结构的过程、比对编译器给出的汇编程序的过程，使我明白了汇编代码编写逻辑，看到了成熟精妙的汇编如何做到高级语言无法完成的工作。底层语言虽然难懂，但也有其独特的魅力。

感谢沈老师和罗老师在项目过程中给予的中肯意见与指导。在我们为浮点运算头痛不已的时候，沈老师指出可以尝试寻找外部浮点处理库来进行运算处理，这给了我们坚持项目的希望。之前做项目的时候, 很多时候完全不明白应该做些什么, 只是靠自己的直觉去行动。沈老师非常详细的ppt ,通过一个个问题, 让我们知道了应该怎么去规划一个项目, 知道了哪些地方是容易出问题的. 三次展示的安排给了我们一种节奏感与紧迫感，这让项目能有序推进。

感谢队长的精密筹划和队友的倾心付出。这次的项目完成过程是比较顺利的一次大作业完成过程。虽然在技术、工程上我们常常会遇到难题，但大家齐心协力、共同进步、积极向上、主动担当的团队精神使项目不断向前推进。我们不仅完成了项目,而且收获了友谊.团队沟通流畅，每个人都尽心完成分工，确定了整体目标后就一致朝着目标前进，这种相互支持的氛围给了我很多的动力。

我们的项目目标并不是那么困难，但在完成后发现，原来简单的事认真做、完整地做也能达到非常好的效果。这使我感到巨大的满足与幸福。

通过这一过程我学到了许多，也希望能提出我自己的一些建议，希望我们的微控制器课程越办越好：

第一个建议是希望老师能邀请参与过微控制器项目研究的研究生学长学姐来担任助教,给我们讲讲微控制器代码的编写中难点有哪些, 他们是怎么思考, 怎么解决的. 这也可以让我们少走一些弯路. 很多时候我们找不到解决平台调试问题的方法, 或许可以邀请研究所的学长学姐编写一个简单的一两页文档, 来方便之后的同学配置微控制器工程文件。

第二个建议是希望能在学在浙大的讨论区开一个问题板块,同学们可以把自己的问题发在上面 ,在讨论区相互解答相互交流, 因为在群聊中消息容易遗漏, 比较零碎不利于学习, 讨论区也可以让下一年的学弟学妹能够学习进步。

第三个建议是个人感觉该门课的知识点是很多的，虽则老师在每章节开始之前都会提及本章的主要内容，但随着课程的深入，我们没有像老师那样清晰的脉络，会感觉到内容很冗杂。如果老师能在课后稍微总结一下该节课的主要知识点，对于我们的课后消化是极好的。

最后, 感谢老师这一学期的辛苦付出!