华 中 科 技 大 学

课 程 实 验 报 告

课程名称：汇编语言程序设计实验

实验名称：实验三 汇编语言与C语言混合编程

实验时间：**2019-10-14，19：00-22：20**

实验地点：南一楼

指导教师：鲁宏伟

专业班级：信息安全1802班

学 号：U201814864

姓 名：师天硕

报告日期：2019年10月15日

**成绩评定**

|  |  |  |
| --- | --- | --- |
| 实验完成质量（70分） | 报告撰写质量（30分） | 总成绩 |
| 实验步骤清晰、详细、深入，实验记录真实完整等 | 报告规范、完整、通顺、详实 |
|  |  |  |

**目 录**

[1实验目的 1](#_Toc22426426)

[2实验内容 1](#_Toc22426427)

[2.1 任务说明 1](#_Toc22426428)

[任务1：在C语言序中嵌入汇编语言指令语句序列 1](#_Toc22426429)

[任务2：在C语言程序中调用汇编语言实现的函数 1](#_Toc22426430)

[2.2 要求 1](#_Toc22426431)

[3实验过程 2](#_Toc22426432)

[3.1开发环境 2](#_Toc22426433)

[3.2设计思想 2](#_Toc22426434)

[3.2.1功能模块分析 2](#_Toc22426435)

[3.2.2数据结构分析 2](#_Toc22426436)

[3.2.3算法设计 2](#_Toc22426437)

[3.2.4汇编和C语言接口设计 3](#_Toc22426438)

[3.3流程图 3](#_Toc22426439)

[3.4源程序 4](#_Toc22426440)

[3.4.1使用内联汇编 4](#_Toc22426441)

[3.4.2调用汇编子程序 7](#_Toc22426442)

[3.5实验步骤 11](#_Toc22426443)

[3.6实验结果及分析 11](#_Toc22426444)

[4心得与体会 12](#_Toc22426445)

# 1实验目的

1. 掌握汇编语言程序与C语言程序混合编程的方法；
2. 熟悉C编译器的基本优化方法;
3. 了解C语言编译器的命名方法，主、子程序之间参数传递的机制。

# 2实验内容

## 2.1 任务说明

### 任务1：在C语言序中嵌入汇编语言指令语句序列

对于实验二的程序进行改造，主控程序、以及输入输出等功能用C语言实现，学生姓名搜索和成绩计算用C程序中嵌入汇编指令语句序列的方式实现。

### 任务2：在C语言程序中调用汇编语言实现的函数

对于实验二的程序进行改造，主控程序、以及输入输出等功能用C语言实现，学生姓名搜索和成绩计算用独立的汇编语言子程序的方式实现；在C语言程序中调用汇编语言子程序。

## 2.2 要求

（1）在不同的C语言开发环境中实现与汇编语言程序的混合编程，其操作方法有可能是不同的。请大家选择白己熟悉的C语言开发环境并查找相关的资料完成本实验。

（2）在实验报告中，详细地描述采用的开发环境及其实现方法。

（3）观察C语言编译器中对各种符号的命名规则（指编译器内部可以识别的命名规则，比如，符号名前面是否加下划线“\_”等），主、子程序之间参数传递的机制，通过堆栈传递参数后堆栈空间回收的方法。

（4）对混合编程形成的执行程序，用调试工具观察由C话言形成的程序代码与由汇编语言形成的程序代码之间的相互关系，包括段、偏移的值，汇编指令访问C的变量时是如何翻译的等。

（5）尝试在C话言程序中不合理地入汇编语言的指令语句，达到破坏C语言程序的正确性的目的。比如，在连续的几条C语言语句中间加入一条修改AX寄存器的汇编指令语句，而AX的内容在此处本不该被修改，这样就可观察到破坏C语言程序正确性的效果（该项实验表明：在C话言程序中，若不考虑上下话句翻译成怎样的机器码而随意嵌入汇编指令语句时，有可能存在出错的风险）。

（6）观察C编译器的优化策略对代码的影响。

（7）通过调试混合编程的程序，体会与纯粹汇编语言编写的程序的调试过程的差异。

（8）通过本次实验，希望大家明白：不同的编程语言是可以协同解决一个问题的，而且可以利用不同语言的特点来更好地解决问题；利用汇编语言的知识，能够更好地理解高级语言的内部处理原理与策略，为编写更好的C语言序、用好C编译器提供支持。

# 3实验过程

## 3.1开发环境

本次实验中使用的环境配置如下：

1. 虚拟机版本：DOSBox v0.74-3

打开后自动执行以下命令，挂载并进入目录，设置环境变量

**mount x C:\80x86-asm-learning**

**set PATH=%PATH%;x:\usr;x:\usr\masm615\BIN;**

**x:**

（2）编译器及其版本：Microsoft Visual C++ 2019

（3）调试工具：Microsoft Visual C++ 2019

（4）编程环境：Microsoft Visual C++ 2019

## 3.2设计思想

### 3.2.1功能模块分析

本程序要在实验二程序的基础上，用C实现主控程序、输入输出、成绩判断，而通过内联汇编和链接调用汇编程序的方法实现学生姓名搜索和成绩计算。因此首先我们将成绩计算和姓名搜索抽象为独立的模块，将其作为单独的C语言函数。

### 数据结构分析

程序所用数据结构和实验而相同，在具体实现上使用struct，并且考虑到之后要使用汇编语言进行寻址，使用#pragma pack(1)语句防止编译器对struct进行内存对齐。具体如下。

#pragma pack(1)

typedef struct \_student {

char name[10];

char a, b, c, avg;

}student;

### 3.2.3算法设计

C语言输入模块考虑到多种可能的输入和行缓冲对程序的影响，因此使用getchar()逐字符读入判断的方式读入带询问的学生姓名。并将其封装在函数int get\_in\_name()中，通过返回值来判断是否收到了合法的姓名，并且通过全局变量overflag判断程序是否终止。

### 3.2.4汇编和C语言接口设计

对任务一和任务二进行综合考量，最终决定用全局变量的形式进行C和汇编的数据共享，借此实现计算和查询的功能，其中，计算是对全局变量中的结构体数组进行操作，而查找也是在寻找到目标student结构体所在地址后将其存入student指针poin中。

对任务一，直接使用VS C++的内联汇编，使用\_\_asm {}将内联汇编段括起来，其中使用C的全局指针变量作为变量即可。对任务二，在C语言头文件中使用

extern void calc\_avg();

extern void get\_poin();

声明两个函数，待和汇编语言链接。在汇编程序中使用

public calc\_avg

public get\_poin

.code

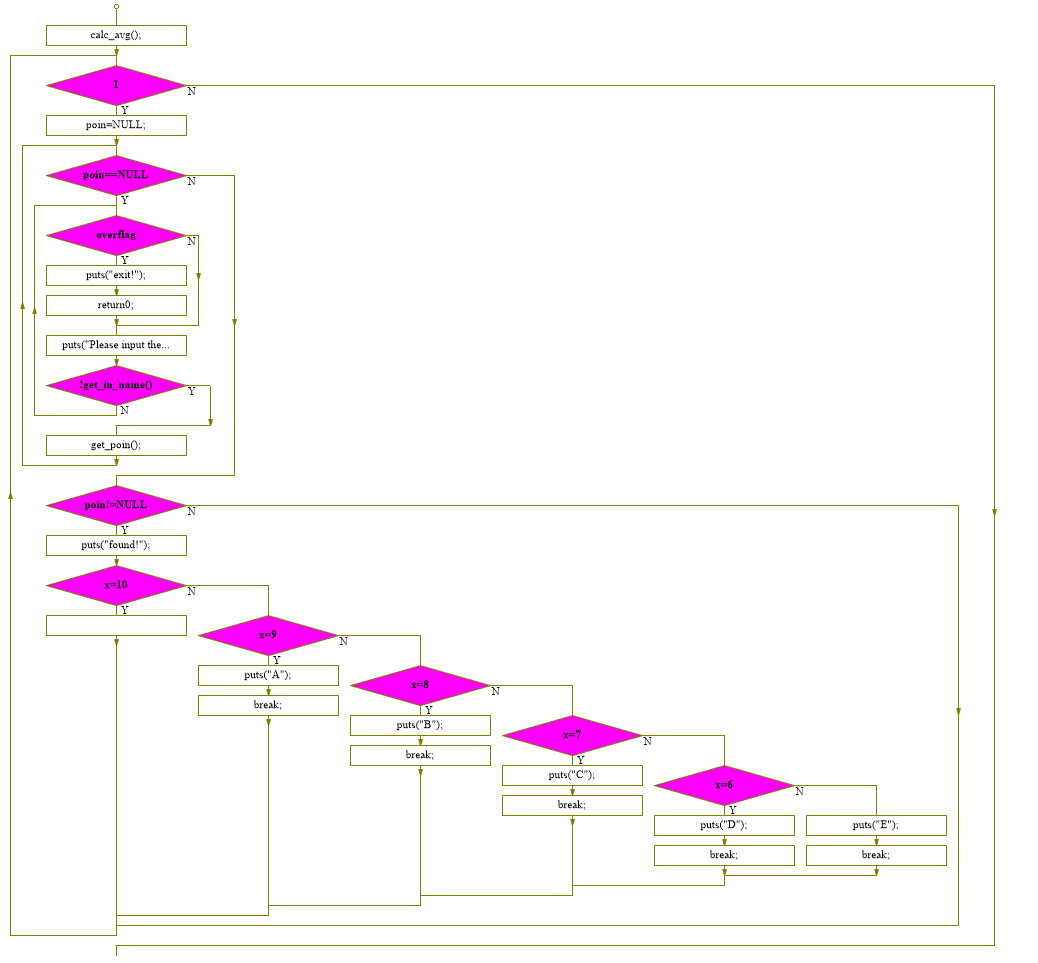
extern s:dword

extern poin:dword

extern in\_name:dword

将两个函数暴露给C语言程序，并且声明需要在程序中使用的全局变量。

## 3.3流程图



**图1程序流程图**

## 3.4源程序

### 3.4.1使用内联汇编

#include <stdio.h>

#define N 10

#pragma pack(1)

typedef struct \_student {

char name[10];

char a, b, c, avg;

}student;

student s[N] = {

{"ZhangSan",100,85,80},

{"LiSi",80,100,70},

{"mxtest",100,100,100},

{"mntest",0,0,0},

{"TempValue",80,90,95},

{"TempValue",80,90,95},

{"TempValue",80,90,95},

{"TempValue",80,90,95},

{"TempValue",80,90,95},

{"stonepage",85,85,100},

};

student\* poin;

char in\_name[11];

int overflag = 0;

void calc\_avg() {

\_\_asm {

mov ebx, offset s

mov ecx, N

avg\_iter :

xor ah, ah

mov al, 10[ebx]

shl ax, 1

add al, 11[ebx]

adc ah, 0

shl ax, 1

add al, 12[ebx]

adc ah, 0

mov dl, 7

div dl

mov 13[ebx], al

add bx, 14

loop avg\_iter

}

}

void get\_poin(){

\_\_asm {

mov ecx, N;

mov ebx, offset s

find\_iter :

mov esi, ebx

mov edi, offset in\_name

push ebx

push ecx

mov ecx, 10

strcmp\_iter :

mov dl, [esi]

cmp dl, [edi]

jnz next

inc esi

inc edi

loop strcmp\_iter

jmp found

next :

pop ecx

pop ebx

add ebx, 14

loop find\_iter

jmp not\_found

found :

mov poin, ebx

pop ecx

pop ebx

not\_found :

}

}

int get\_in\_name() {

int i;

for (in\_name[i = 0] = getchar(); i < 10; in\_name[++i] = getchar()) {

if (in\_name[i] == '\n') break;

}

if (!i) return 0;

if (in\_name[i] != '\n') {

puts("too long!");

while (getchar() != '\n');

return 0;

}

if (i == 1 && in\_name[0] == 'q') {

overflag = 1;

return 0;

}

for (; i < 10; i++) in\_name[i] = 0;

return 1;

}

int main() {

calc\_avg();

while (1) {

poin = NULL;

while (poin == NULL) {

do {

if (overflag) {

puts("exit!");

return 0;

}

puts("Please input the name of the student:");

} while (!get\_in\_name());

get\_poin();

}

if (poin != NULL) {

puts("found!");

printf("the level of %s is ", in\_name);

int x = poin->avg ? poin->avg / 10 : 0;

switch (x) {

case 10:

case 9: puts("A"); break;

case 8: puts("B"); break;

case 7: puts("C"); break;

case 6: puts("D"); break;

default: puts("E"); break;

}

}

}

}

### 3.4.2调用汇编子程序

b.h

#pragma once

extern void calc\_avg();

extern void get\_poin();

b.c

#include <stdio.h>

#include "b.h"

#define N 10

#pragma pack(1)

typedef struct \_student {

char name[10];

char a, b, c, avg;

}student;

student s[N] = {

{"ZhangSan",100,85,80},

{"LiSi",80,100,70},

{"mxtest",100,100,100},

{"mntest",0,0,0},

{"TempValue",80,90,95},

{"TempValue",80,90,95},

{"TempValue",80,90,95},

{"TempValue",80,90,95},

{"TempValue",80,90,95},

{"stonepage",85,85,100},

};

student \*poin;

char in\_name[11];

//int dwtest;

int overflag = 0;

int get\_in\_name() {

int i;

for (in\_name[i = 0] = getchar(); i < 10; in\_name[++i] = getchar()) {

if (in\_name[i] == '\n') break;

}

if (!i) return 0;

if (in\_name[i] != '\n') {

puts("too long!");

while (getchar() != '\n');

return 0;

}

if (i == 1 && in\_name[0] == 'q') {

overflag = 1;

return 0;

}

for (; i < 10; i++) in\_name[i] = 0;

return 1;

}

int main() {

calc\_avg();

while (1) {

poin = NULL;

while (poin == NULL) {

do {

if (overflag) {

puts("exit!");

return 0;

}

puts("Please input the name of the student:");

} while (!get\_in\_name());

get\_poin();

}

if (poin != NULL) {

puts("found!");

printf("the level of %s is ", in\_name);

int x = poin->avg ? poin->avg / 10 : 0;

switch (x) {

case 10:

case 9: puts("A"); break;

case 8: puts("B"); break;

case 7: puts("C"); break;

case 6: puts("D"); break;

default: puts("E"); break;

}

}

}

}

basm.asm

.386

.model flat, c

public calc\_avg

public get\_poin

.code

extern s:dword

extern poin:dword

extern in\_name:dword

N equ 10

calc\_avg proc

mov ebx, offset s

mov ecx, N

avg\_iter :

xor ah, ah

mov al, 10[ebx]

shl ax, 1

add al, 11[ebx]

adc ah, 0

shl ax, 1

add al, 12[ebx]

adc ah, 0

mov dl, 7

div dl

mov 13[ebx], al

add bx, 14

loop avg\_iter

ret

calc\_avg endp

get\_poin proc

mov ecx, N;

mov ebx, offset s

find\_iter :

mov esi, ebx

mov edi, offset in\_name

push ebx

push ecx

mov cl, 10

xor ch, ch

strcmp\_iter :

mov dl, [esi]

cmp dl, [edi]

jnz next

inc esi

inc edi

loop strcmp\_iter

jmp found

next :

pop ecx

pop ebx

add ebx, 14

loop find\_iter

jmp not\_found

found :

mov poin, ebx

pop ecx

pop ebx

not\_found:

ret

get\_poin endp

end

## 3.5实验步骤

1.阅读代码，查看资料，确定程序功能

2.准备上机实验环境，编写代码。

3.经MASM汇编，LINK连接后并且确认源程序正确无误后，直接运行，观察程序对输入的处理和输出有无明显错误。

4.测试各类输入，覆盖所有情况，

5.修改数据段内数据，重新编译链接进行测试。

## 3.6实验结果及分析

1. 编译完成，无报错。直接执行二进制程序，出现提示字符串’Please input the name of the student:’

2. 输入一个列表中已经存在的姓名stonepage程序显示该学生平均分所对应的等级， 并继续提示用户输入学生姓名。

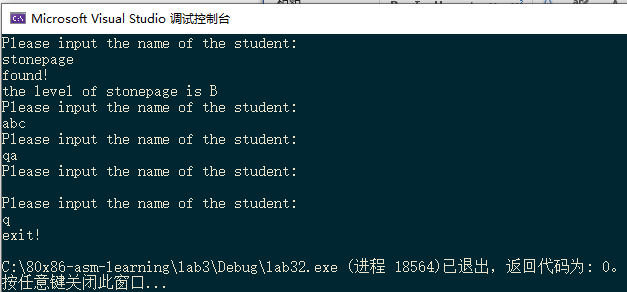
3. 输入一个列表中不存在的姓名abc，程序提示该学生不存在，并要求重新输入。

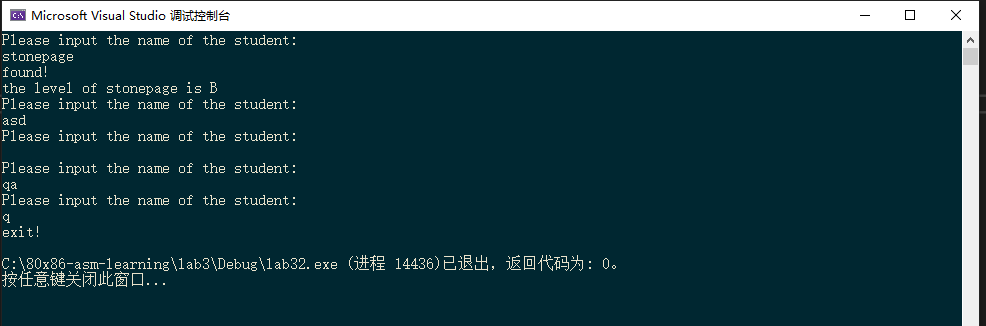
4. 输入一个列表中不存在，且以字母q为前缀的姓名qa，程序提示该学生不存在，并要求重新输入。

5. 直接输入回车，程序再次要求输入学生姓名。

6. 输入 q，程序直接退出。

1~6 步两程序的输出结果如图2所示，初步表明程序能够正常运行。





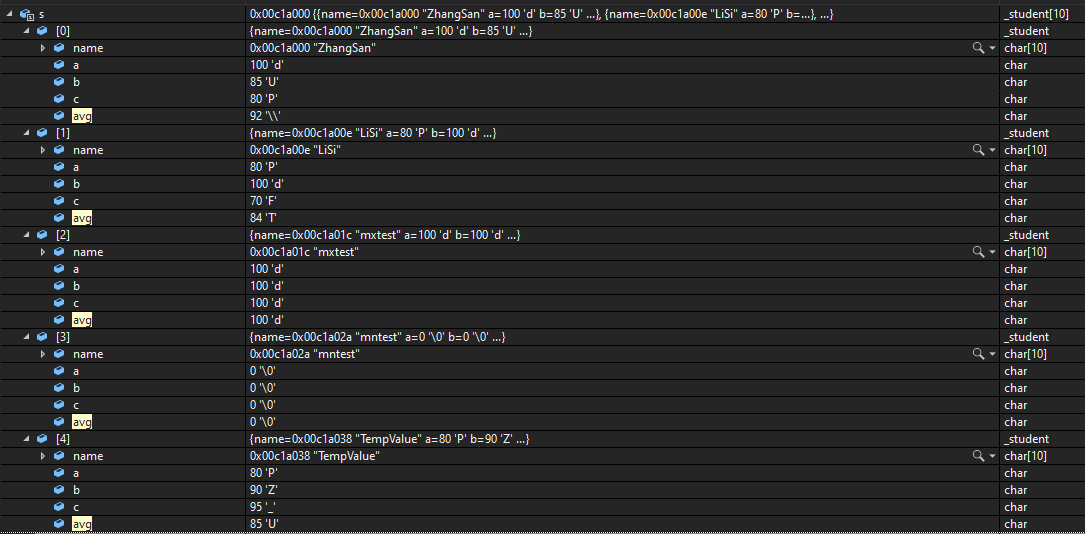
**图2初步测试记录**

7.测试计算部分：使用VS调试器，程序跳转到计算数据段中对应测试数据的位置。测试数据如下。

{"mxtest",100,100,100},

{"mntest",0,0,0},

单步执行程序，观察平均分的计算以及存储是否正常运行，级别判断是否正确以及程序是否溢出。测试表明，即使总分数全部取最大，即 100,100,100，在执行加权以后总分为 350，没有超过一个16位寄存器的表示范围，而乘法的积与除法的被除数均为16位寄存器，因此不会溢出。测试后相关平均分计算部分数据如图，答案正确



**图3极端数据测试后的内存**

# 4心得与体会

本次实验要要使用C语言和汇编语言进行混合编程，其中C语言和汇编语言的编程环境多种多样，在选择实现实验方案的过程中，我学习了AT&T汇编、了解了一些intel64位汇编，在查阅官方文档中，了解到了NASM和MASM汇编编译器的不同、看了Clang、GCC、VSC等不同C语言编译器对内联汇编语言的不同要求，在折腾了许许多多不同的技术，了解了各式各样的解决方案和其中的风险和问题之后，我为现代IDE的威力深深折服，最终选择了用VS进行项目的编译和链接，在这个过程中大大开拓了我的眼界，锻炼了我查找资料和学习的能力。这让我对程序编译、链接的过程更加熟悉和了解。