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：汇编语言程序设计实验一

实验时间：**2019-9-16，19：00-22：20**

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9月16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](#_Toc22420517)

[2实验内容 1](#_Toc22420518)

[2.1任务一 1](#_Toc22420519)

[2.2任务二 2](#_Toc22420520)

[3实验过程 3](#_Toc22420521)

[3.1开发环境 3](#_Toc22420522)

[3.2任务一 3](#_Toc22420523)

[3.2.1 实验步骤 3](#_Toc22420524)

[3.2.4 修改后的源程序 3](#_Toc22420525)

[3.2.3 实验记录 4](#_Toc22420526)

[3.3任务二 6](#_Toc22420527)

[3.3.1实验步骤 6](#_Toc22420528)

[3.3.2修改后的源程序 6](#_Toc22420529)

[3.3.3实验记录 7](#_Toc22420530)

[4心得与体会 8](#_Toc22420531)

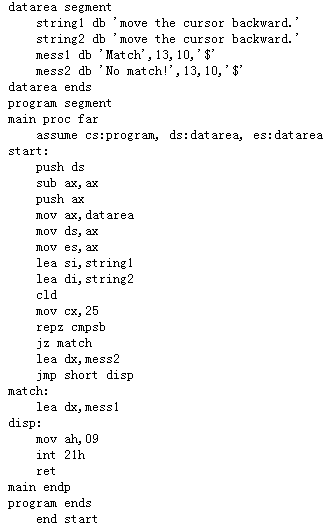
# 1实验目的

1. 掌握汇编源程序编辑工具、汇编程序、连接程序、调试工具 TD 的使用；
2. 理解数、符号、寻址方式等在计算机内的表现形式；
3. 理解指令执行与标志位改变之间的关系；
4. 熟悉常用的 DOS 功能调用；
5. 熟悉分支、循环程序的结构及控制方法，掌握分支、循环程序的调试方法；
6. 加深对转移指令及一些常用的汇编指令的理解。

# 2实验内容

## 2.1任务一

阅读以下程序，根据指令的执行流程，说明程序实现的目标，并采用汇编程序对改代码进行编辑、编译、连接和调试。



要求：

1. 查阅资料，说明指令“cld”和“repz cmpsb”的功能；
2. 查阅资料，说明“int 21h”的功能；
3. 借助TD工具，观察指令“repz cmpsb”完成后，哪些标志位发生了变化。
4. 程序中的两条“push”指令的作用是什么，有没有感觉代码中缺少与“push”指令相关的操作指令？如果有，尝试完善改短代码，并观察程序运行效果。
5. 运行程序，说明当前程序运行的结果，然后修改字符串string2，让程序产生不同的结果。

## 2.2任务二

阅读下列程序, 并指出程序执行之后, 以BUF2、BUF3、BUF4 为首址的3 个字节存储区中存放的数据。

|  |  |
| --- | --- |
|  |  |

要求：

1. 分别记录执行到“ MOV CX，10”和“INT 21H”之前的(BX)、(BP)、(SI)、(DI)各是多少。
2. 记录程序执行到退出之前数据段开始，40个字节的内容，指出程序运行结果是否与设想的一致。
3. 在标号LOPA前加上一段程序，实现新的功能：先显示提示信息“Press any key to begin!”, 然后，在按了一个键之后继续执行 LOPA 处的程序。

操作提示：使用 TD.EXE 调试程序时，应先单步执行各个语句，每执行一条语句，都应观察数据段中的内容以及相应寄存器的变化。首先注意观察对DS寄存器的赋值过程，并在TD的数据窗口定位待观察的数据区位置。其次，单步执行循环体两遍且正确理解了循环体语句的含义后，可在“MOV AH, 4CH”处设置断点，然后直接执行到断点处，回答(1)和(2)的问题。完成（3）的内容，涉及到“INT 21H”相关的输入和输出操作，参考任务1中显示字符串的方法。

## 

# 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：masm6.11

（3）调试工具：TD

（4）编程环境：Visual Studio Code

## 3.2任务一

### 3.2.1 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，使用TD对连接后的EXE程序进行调试，打断点，在“repz cmpsb”指令前后记录标志位。

④查阅资料，修改程序，观察结果是否相同

⑤修改程序使得结果不同

### 3.2.4 修改后的源程序

datarea segment

string1 db 'Move the cursor backwardx'

string2 db 'Move the cursor backward.'

mess1 db 'Match.',13,10,'$'

mess2 db 'No Match!',13,10,'$'

datarea ends

prognam segment

main proc far

assume cs:prognam,ds:datarea,es:datarea

start:

mov ax,datarea

mov ds,ax

mov es,ax

lea si,string1

lea di,string2

cld

mov cx,25

repz cmpsb

jz match

lea dx,mess2

jmp short disp

match:

lea dx,mess1

disp:

mov ah,09

int 21h

mov ax 4c00h

int 21h;

main endp

prognam ends

end start

### 3.2.3 实验记录

1. 查阅资料，说明指令“cld”和“repz cmpsb”的功能；

8086CPU提供了一组处理主存中连续存放的数据串的指令——串操作指令。每执行一次串操作指令，作为源地址指针的SI和作为目的地址指针的DI将自动修改。而修改的方向由标志位DF决定。

执行指令CLD指令后，DF=0，表示地址指针增，执行STD指令，DF=1，地址指针减。

CMPSB 指令是串比较指令，它可以比较原串和目的串是否相同，并根据减法结果设置标志位。修改SI、DI寄存器的值，指向下一个元素。B表示是字节串。

REPZ指令是重复前缀指令，把后面的CMPS指令循环执行。每执行一次指令CX减1并判断标志位。CX=1或ZF=1时结束。

在之前令CX为25，因此repz就是遍历判断两个串是否相等。讲结果存在ZF标志位中。

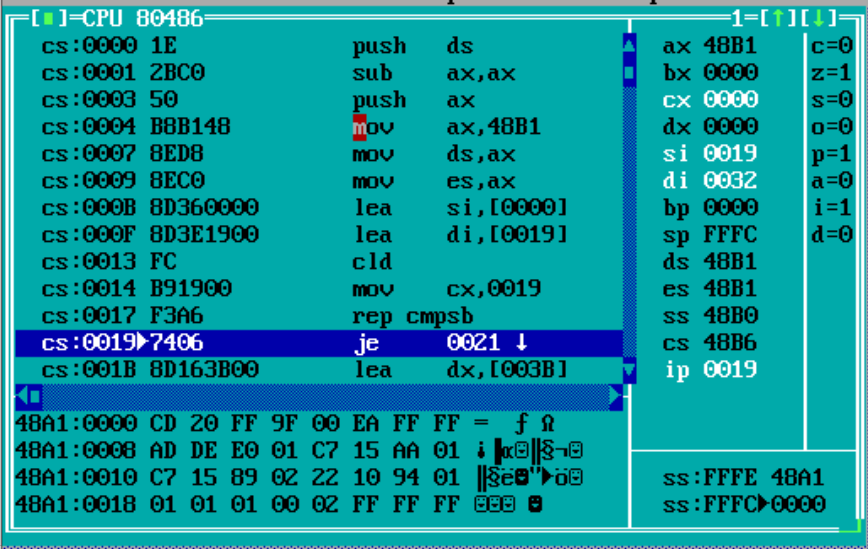
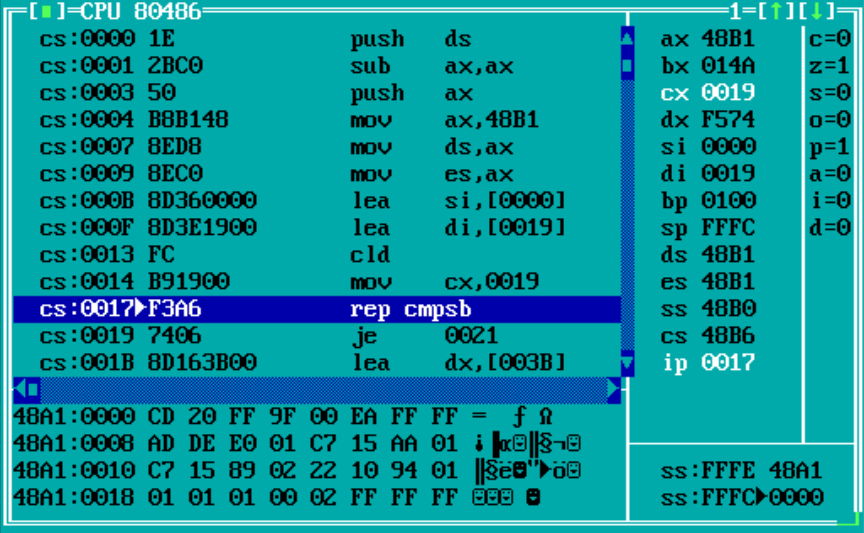
1. 查阅资料，说明“int 21h”的功能；

Int 21h是由DOS提供的系统功能调用，一组实现特殊功能的子程序供程序员在编写自己的程序时调用，以减轻编程的工作量。涉及屏幕显示、文件管理、I/O管理等等，每个子程序都有一个功能号，在调用int 21h之前传入寄存器AH中。

这里之前讲AH置为9，表示输出DS:DX的以“$”结尾的字符串，

1. 借助TD工具，观察指令“repz cmpsb”完成后，哪些标志位发生了变化。

如下图1所示，指令“repz cmpsb”完成前后标志位没有发生变化，说明匹配过程中，字节串都相同。



**图1 指令“repz cmpsb”完成前（左）、完成后（右）的标志位**

1. 程序中的两条“push”指令的作用是什么，有没有感觉代码中缺少与“push”指令相关的操作指令？如果有，尝试完善改短代码，并观察程序运行效果。

程序加载后，DS存放着程序所在内存的段地址，这个内存区域的前256个字节存放在的是PSP（程序段前缀）。PSP是DOS操作系统在执行程序时为程序所建立的一个信息块，里面包括了传递给待运行程序的命令行参数，程序运行结束时返回DOS所需的地址等有用的信息。

这里讲DS和0压入栈内，栈里存储的是程序结束后应该返回DOS的代码逻辑地址，在程序最后执行指令ret，即pop ip, pop cs时，通过修改CS、IP寄存器返回DOS系统，这是一种存储上下文的方式。

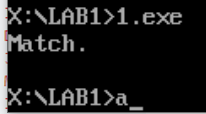
现在通常使用的方式是，在最后直接调用int 21h;DOS系统调用返回DOS。用如下语句

mov ax 4c00h

int 21h;

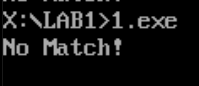
1. 运行程序，说明当前程序运行的结果，然后修改字符串string2，让程序产生不同的结果。

运行程序结果如图3。两个字符串是相同的，因此输出mess1



**图2程序1的运行结果**

将string1 db 'Move the cursor backward.'改为string1 db 'Move the cursor backwardx'在未改变字符串长度的情况下，使两个字符串不同。运行结果如图4。



**图3程序1修改后的运行结果**

## 3.3任务二

### 3.3.1实验步骤

①准备上机环境，编写实验程序

②经MASM汇编，LINK连接后并且确认源程序正确无误后，使用TD对连接后的EXE程序进行调试，打断点，执行至mov cx,10后记录(BX)，(BP)，(SI)，(DI)的值。继续打断点至int 21h处，记录(BX)，(BP)，(SI)，(DI)的值。然后在内存数据窗口跳转至DS段，记录DS段的前40个字节的数据。

③然后重新更改程序，通过系统21H中断的9号调用显示相应的字符串，然后再通过系统21H中断的1号调用读取输入字符，中断返回，继续执行LOPA处的程序。

### 3.3.2修改后的源程序

.386

stack segment use16 stack

db 200 dup(0)

stack ends

data segment use16

buf1 db 0,1,2,3,4,5,6,7,8,9

buf2 db 10 dup (0)

buf3 db 10 dup (0)

buf4 db 10 dup (0)

string db 'Press any key to begin!', '$'

data ends

code segment use16

assume cs:code, ds:data, ss:stack

start: mov ax, data

mov ds, ax

mov si, offset buf1

mov di, offset buf2

mov bx, offset buf3

mov bp, offset buf4

mov cx, 10

mov dx, offset string

mov ah, 9

int 21h

mov ah, 1

int 21h

lopa: mov al, [si]

mov [di], al

inc al

mov [bx], al

add al, 3

mov ds:[bp], al

inc si

inc di

inc bp

inc bx

dec cx

jnz lopa

mov ah, 4ch

int 21h

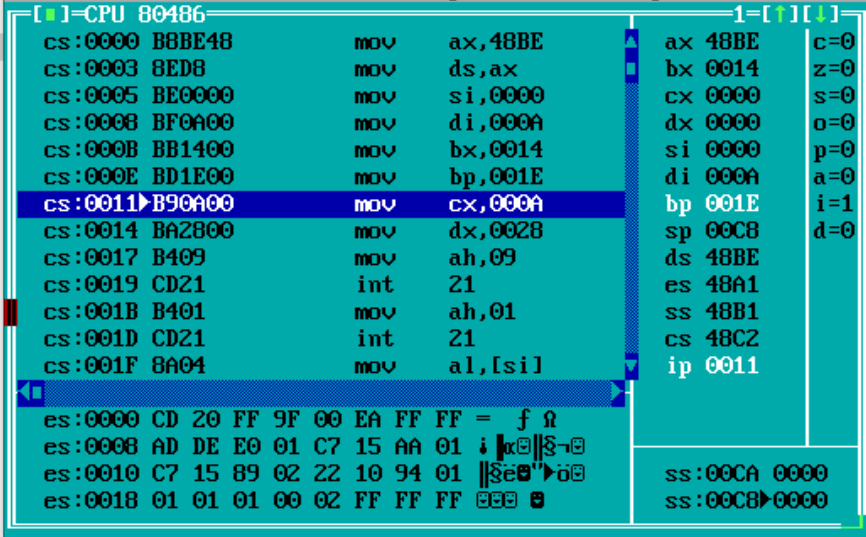
code ends

end start

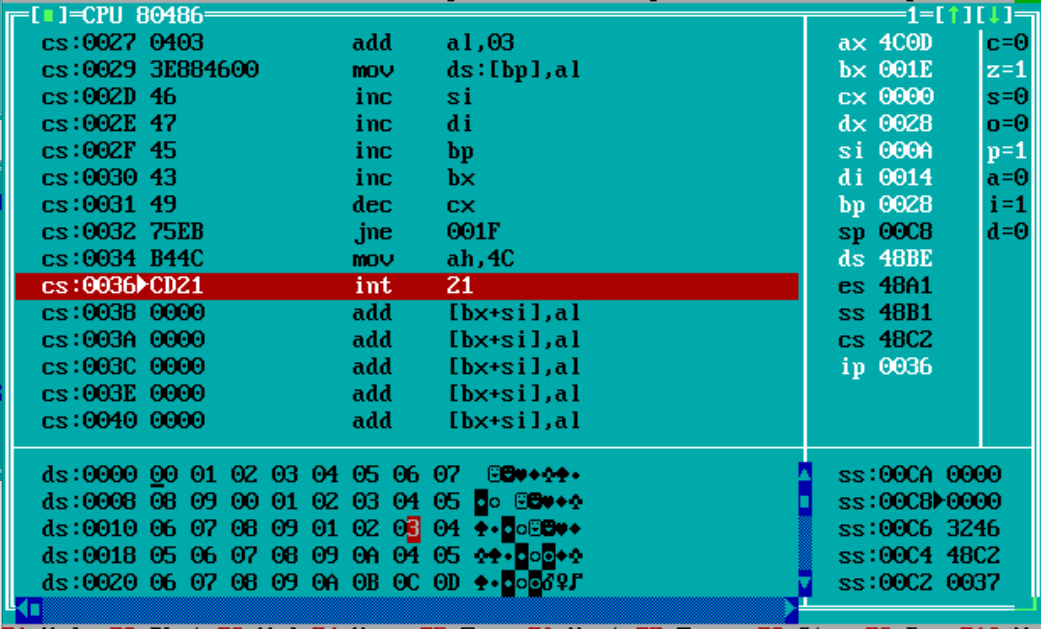
### 3.3.3实验记录

1. 分别记录执行到“ MOV CX，10”和“INT 21H”之前的(BX)、(BP)、(SI)、(DI)各是多少。

如图所示

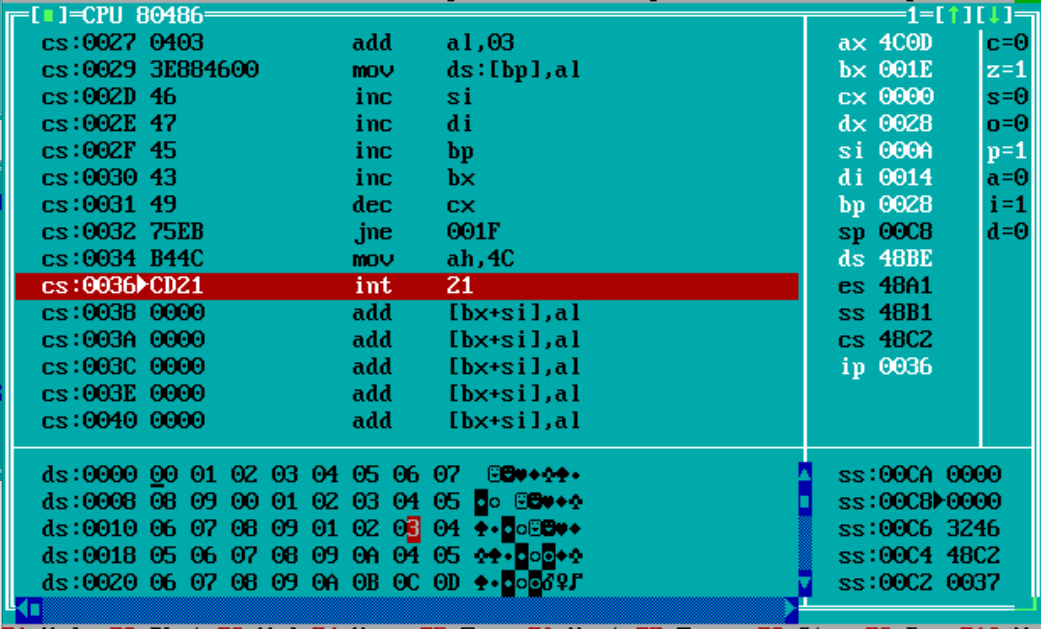


**图4 执行到“ MOV CX，10”之前寄存器的值**



**图5 执行到“INT 21H”之前寄存器的值**

1. 记录程序执行到退出之前数据段开始，40个字节的内容，指出程序运行结果是否与设想的一致。



**图6 执行到“INT 21H”之前数据段开始40个字节的值**

1. 在标号LOPA前加上一段程序，实现新的功能：先显示提示信息“Press any key to begin!”, 然后，在按了一个键之后继续执行 LOPA 处的程序。

在lopa之前增加如下程序。利用

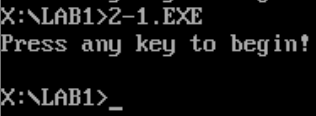
mov ah, 9

int 21h

mov ah, 1

int 21h

实现功能，如下图所示



**图6 修改后程序的运行结果**

# 4心得与体会

本次实验是学习汇编语言的第一次实验。在这次实验中，我第一次编写了完整的汇编语言程序，对汇编程序的基本结构有了了解，学会了一些伪指令的使用方法。学会利用masm对程序进行编译链接，在dosbox虚拟机中运行。同时学习了td调试工具的使用方法，之前学习的抽象的程序运行过程具现的变成了寄存器和内存中的数据的变化，这大大加深了我对汇编语言的理解，在以后的学习中，对于不明白的汇编语句可以使用调试，实验每一个指令每一段代码的实际表现。具体的新学到的知识方面，我了解到了x86汇编对字符串的串指令、与DOS系统交互的系统调用、DOS系统中程序运行中PSP(程序段前缀)的相关知识等。

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：实验二分支程序、循环程序的设计

实验时间：**2019-9-30，19：00-22：20**

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9月30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](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861972)

[2实验内容 1](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861973)

[2.1 学生成绩查询程序 1](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861974)

[2.1.1 实验说明 1](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861975)

[2.1.2 功能一：提示并输入学生姓名 1](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861976)

[2.1.3功能二：以学生姓名查询有无该学生 2](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861977)

[2.1.4功能三：计算所有学生的平均成绩 2](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861978)

[2.1.5功能四：将功能二查到的学生的平均成绩进行等级判断，并显示判断结果 2](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861979)

[3实验过程 2](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861980)

[3.1开发环境 2](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861981)

[3.2设计思想 3](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861982)

[3.2.1功能模块分析 3](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861983)

[3.2.2 数据结构分析 3](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861984)

[3.2.3算法设计 4](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861985)

[3.3流程图 4](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861986)

[3.4源程序 4](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861987)

[3.5实验步骤 7](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861988)

[3.6实验结果及分析 7](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861989)

[4心得与体会 8](file:///C:\80x86-asm-learning\汇编语言程序设计课程实验二.docx#_Toc20861990)

# 1实验目的

1. 熟悉分支、循环程序的结构及控制方法，掌握分支、循环程序的调试方法；
2. 加深对转移指令及一些常用的汇编指令的理解；
3. 加深对常用DOS功能调用指令的理解，进一步熟悉TD的使用。

# 2实验内容

## 2.1 学生成绩查询程序

设计实现一个学生成绩查询的程序

### 2.1.1 实验说明

在以BUF为首地址的字节数据存储区中，存放着N个学生的课程成绩表（百分制），每个学生的相关信息包括：10名（占10个字节），结束符为数值0；语文成绩（1个字节）；数学成绩（1个字节）；英语成绩（1个字节）；平均成绩（1个字节）。

例如：

N equ 1000

BUF db ‘ZhangSan’, 0, 0

;学生姓名，不足10个字节的部分用0填充

db 100, 85, 80, ？

;平均成绩还未计算

db ‘LiSi’, 6 dup(0)

db 80, 100, 70, ？

db N-3 dup(‘TempValue ’, 0, 80, 90, 95, ?)

; 除了已经定义了的学生信息及成绩表外，其他学生的暂时成绩假定是一样的

db ‘WangWu’, 4 dup(0)

;最后一个必须修改为自己名字的拼音

db 85, 85, 100, ？

### 2.1.2 功能一：提示并输入学生姓名

1. 使用9号DOS系统功能调用，提示用户输入学生姓名。
2. 使用10号DOS系统功能调用，输入学生姓名。输入的学生姓名字符串存放在以in\_name为首地址的存储区中。
3. 若只是输入了回车，则回到“（1）”处重新提示并输入；若仅仅输入字符q，则程序退出，否则，准备进入下一步处理。

### 2.1.3功能二：以学生姓名查询有无该学生

1. 使用循环程序结构，在成绩表中查找该学生。
2. 若未找到，提示用户该学生不存在，并回到“功能一（1）”的位置，提示并重新输入姓名。
3. 若找到，则将该学生课程成绩表的起始偏移地址保存到POIN字变量中。

提示：字符串比较时，当采用输入串的长度作为循环次数时，若因循环次数减为0而终止循环，则还要去判断成绩表中名字串的下一个字符是否是结束符0，若是，才能确定找到了。

### 2.1.4功能三：计算所有学生的平均成绩

1. 使用算术运算相关指令计算并保存每一个学生的平均成绩。
2. 平均成绩计算公式：（A\*2+B+C/2）/ 3.5，即将语文成绩A乘以权重2、英语成绩C除以权重2后，再与数学成绩B一起求和，再计算该生的平均成绩。要求避免溢出。

提示：使用循环程序结构，注意寻址方式的灵活运用。把小数3.5转换成分数后再运算以避免使用浮点指令。

### 2.1.5功能四：将功能二查到的学生的平均成绩进行等级判断，并显示判断结果

1. 平均成绩等级显示方式：若平均成绩大于等于90分，显示“A”；大于等于80分，显示“B”；大于等于70分，显示“C”；大于等于60分，显示“D”；小于60分，显示“F”。

提示：使用分支程序结构，采用2号DOS系统功能调用显示结果。

1. 使用转移指令回到“功能一（1）”处（提示并输入姓名）。

# 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：masm6.11

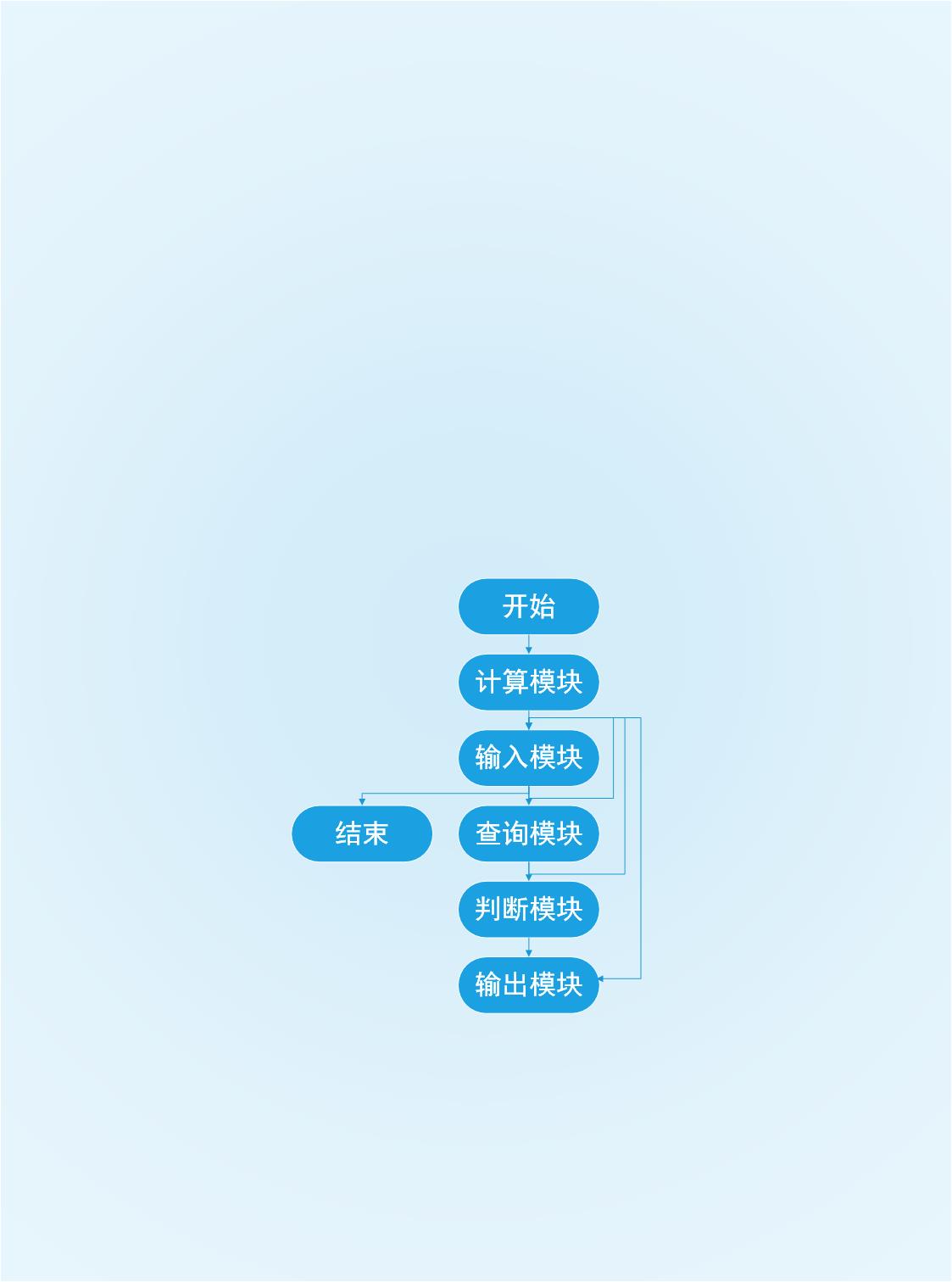
（3）调试工具：TD

（4）编程环境：Visual Studio Code

## 3.2设计思想

### 3.2.1功能模块分析

本程序要求支持4个功能，我们将其对应的大致分为5个部分：输入部分，查找部分，成绩计算部分，等级判断部分和输出部分。仔细分析相应功能后，发现成绩计算部分和其它部分耦合较少，在进入程序时统一计算即可。而其它四个部分是明显的顺序结构。细节上，在输入后需要由输入内容控制跳转，而在查找之前需要对输入的数据进行处理。综上，程序大致模块如图1所示。



**图1 程序模块图**

### 3.2.2 数据结构分析

程序所用程序记录在同一数据段中，主要分三段。

1. 程序运行中用于指示的字符串，用$号结尾的字节床表示。包括以下四条。

'Please input the name of the student: $'

'Student not found, please re-input ... ',0dh,0ah,'$'

'the level of the student are: $'

'exit!',0dh,0ah,'$'

1. 程序运行时的缓存，用以存储读入的查询学生姓名。由头上的meta信息和之后的数据你组成。一共12字节，第一个字节存缓冲区长度，第二个字节存缓冲区读入字节数，第3到12个字节存放读入的字符。
2. 学生的信息记录，每条记录有14字节，由十字节长度的姓名，和4字节的成绩组成，如图2所示。



**图2学生记录内存分布示意图**

### 3.2.3算法设计

总体上，所有的字符串输出使用 int 21H 的 09H 号中断，需要输出的内容全部保存在数 据段；所有需要输出的字符使用 int 21H 的 02H 号中断，需要输出的字符使用立即数表 示；需要输入数据则使用 int 21H 的 0AH 号中断，需要输入的数据存在数据区的数据缓 冲区中。

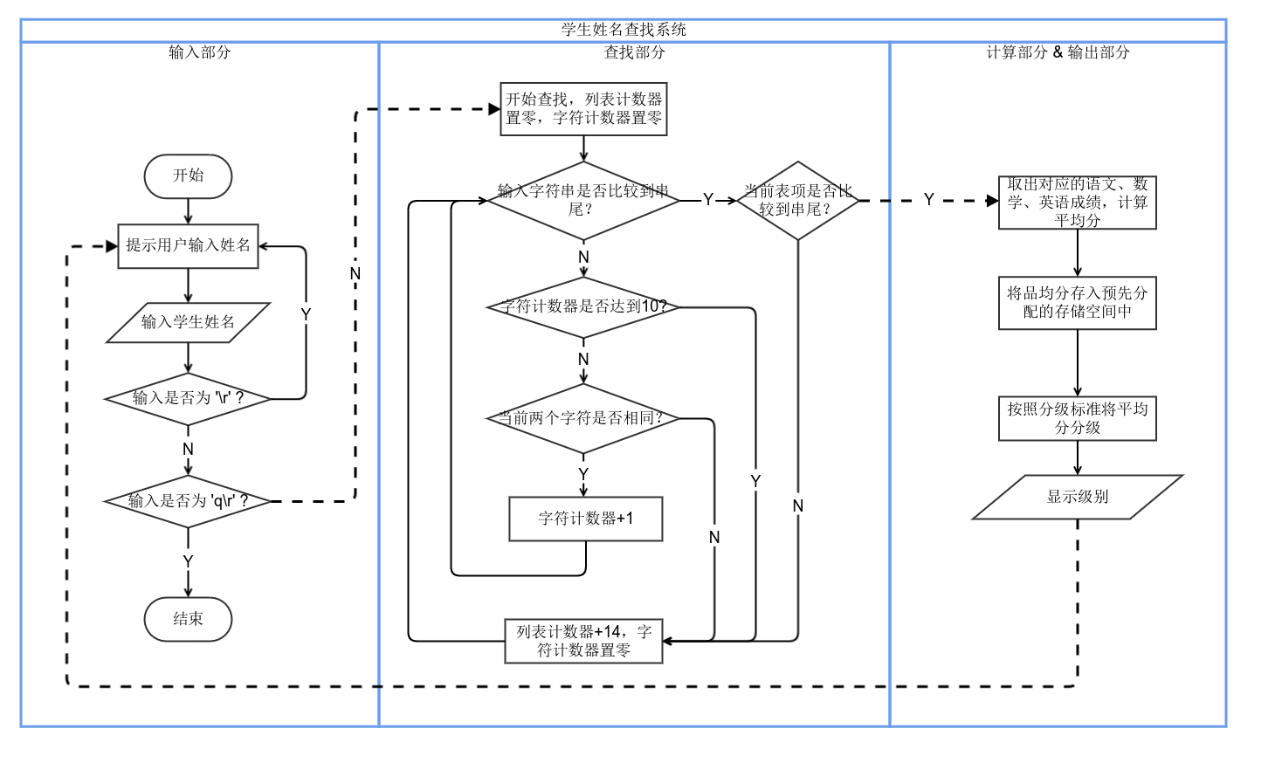
计算部分，发现计算平均成绩公式的特性，计算过程可以优化为一系列\*2（左移）和相加操作，最后利用div指令计算平均成绩，存入内存。

输入部分，对用户输入的数据进行判断，若为回车字符 (DH)，则判定为非法输入，提示 重新输入；若为’q\n’̊(71H, 0DH)，则判断为退出，直接终止程序；否则判断为姓名，进入查找部分继续执行。

查找部分，利用二重循环，遍历每条学生记录。在嵌套循环中：对姓名的每一个字进行逐字匹配，一旦发现失配则匹配不成功，进入下一个循环；若同时匹配到结尾，表明匹配成功，跳转进入判断部分；若所有的姓名均遍历完成仍没有匹配成功者，则表明姓名不存在，重新进入输入部分。

## 3.3流程图

如图3所示



**图3程序流程图**

## 3.4源程序

.386

BR macro

mov ah, 02h ; \r

mov dl, 0dh

int 21h

mov ah, 02h ; \n

mov dl, 0ah

int 21h

endm

stackseg segment use16 stack

dw 10 dup(0)

stackseg ends

dataseg segment use16

s1 db 'Please input the name of the student: $'

s2 db 'Student not found, please re-input ... ',0dh,0ah,'$'

s3 db 'the level of the student are: $'

s4 db 'exit!',0dh,0ah,'$'

in\_name\_head db 10, ?

in\_name db 10 dup(0)

N equ 10

s\_name equ 0

s\_mx\_name equ 9

s\_a equ 10

s\_b equ 11

s\_c equ 12

s\_avg equ 13

s\_len equ 14

BUF db 'ZhangSan', 2 dup(0) ;学生姓名，不足10个字节的部分用0填充

db 100, 85, 80, ? ;平均成绩还未计算

db 'LiSi', 6 dup(0)

db 80, 100, 70, ?

db 'mxtest', 4 dup(0)

db 100, 100, 100, ?

db 'mntest', 4 dup(0)

db 0, 0, 0, ?

db N-5 dup('TempValue', 0, 80, 90, 95, ?); 除了已经定义了的学生信息及成绩表外，其他学生的暂时成绩假定是一样的

db 'stonepage', 0 ;最后一个必须修改为自己名字的拼音

db 85, 85, 100, ?

poin dw 0

dataseg ends

codeseg segment use16

assume ss:stackseg, ds:dataseg, cs:codeseg

start:

mov ax, dataseg

mov ds, ax

mov ax, stackseg

mov ss, ax

mov sp, 20

avg\_all:

mov bx, offset BUF

mov cx, N

avg\_iter:

xor ah, ah

mov al, s\_a[bx]

shl ax, 1

add al, s\_b[bx]

adc ah, 0

shl ax, 1

add al, s\_c[bx]

adc ah, 0

mov dl, 7

div dl

mov s\_avg[bx], al

add bx, s\_len

loop avg\_iter

input:

mov ah, 09h

mov dx, offset s1

int 21h

mov ah, 0ah

mov in\_name\_head, s\_mx\_name+1

lea dx, in\_name\_head

int 21h

BR

;0dh=='\n' 71h=='q'

cmp in\_name, 0dh

jz input

cmp in\_name, 71h

jnz filter

cmp in\_name+01h, 0dh

jnz filter

jmp quit

filter:

mov bl, in\_name\_head[1]

xor bh, bh

mov byte ptr in\_name[bX], 0

find:

mov cx, n

mov bx, offset BUF

find\_iter:

lea si, s\_name[bx]

mov di, offset in\_name

push bx

push cx

mov cl, in\_name\_head[1]

xor ch, ch

strcmp\_iter:

mov dl, [si]

cmp dl, [di]

jnz next

inc si

inc di

loop strcmp\_iter

jmp found

next:

pop cx

pop bx

add bx, s\_len

loop find\_iter

not\_found:

mov ah, 09h

mov dx, offset s2

int 21h

jmp input

found:

mov poin, bx

mov al, s\_avg[bx]

outs3:

mov dx, offset s3

mov ah, 09H

int 21H

judge:

mov dl, 'A'

cmp al, 90

jge outlevel

inc dl

cmp ax, 80

jge outlevel

inc dl

cmp ax, 70

jge outlevel

inc dl

cmp ax, 60

jge outlevel

add dl, 2

outlevel:

mov ah, 02h

int 21h

BR

jmp input

quit:

mov ah, 09h

mov dx, offset s4

int 21h

mov ax, 4c00h

int 21h

codeseg ends

end start

## 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。

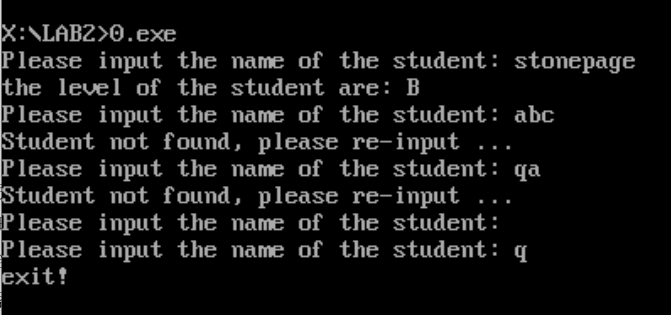
3. 输入一个列表中不存在的姓名abc，程序提示该学生不存在，并要求重新输入。

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 步输出结果如图4所示，初步表明程序能够正常运行。



**图4初步测试记录**

7.测试计算部分：使用TD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。

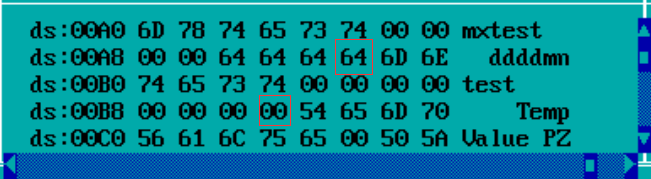
db 'mxtest', 4 dup(0)

db 100, 100, 100, ?

db 'mntest', 4 dup(0)

db 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的内存数据如图，答案正确



**图5极端数据测试后的内存**

# 4心得与体会

本次实验要完成一个完整的小程序。从分析问题，设计算法和数据结构，到编码和调试，经历了用汇编语言开发的完整流程。

一方面，这个程序让我几乎使用了迄今为止学到的所有汇编语法，这让我对汇编语言的各种指令、寻址方式等更加熟悉。在实际的编程中，出现的bug让我必须关注平日学习中有意无意忽略的细节，让我对指令和语法有更深的理解。与此同时我也更加习惯，在有限的寄存器下进行编程，只有较为简单的流程控制，等汇编语言这种极低抽象的语言特性。

另一方面，在不断的对错误的调试中，我对Turbo Debugger 的各种便于调试的功能和调试的方法（如打断点，监视，查找，跳 转，录制宏等）越发熟悉，举例来讲，此次调试中我就了解到了F7 Trace和F8 Step单步指令的区别。这些都大大提高了我的综合能力。

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

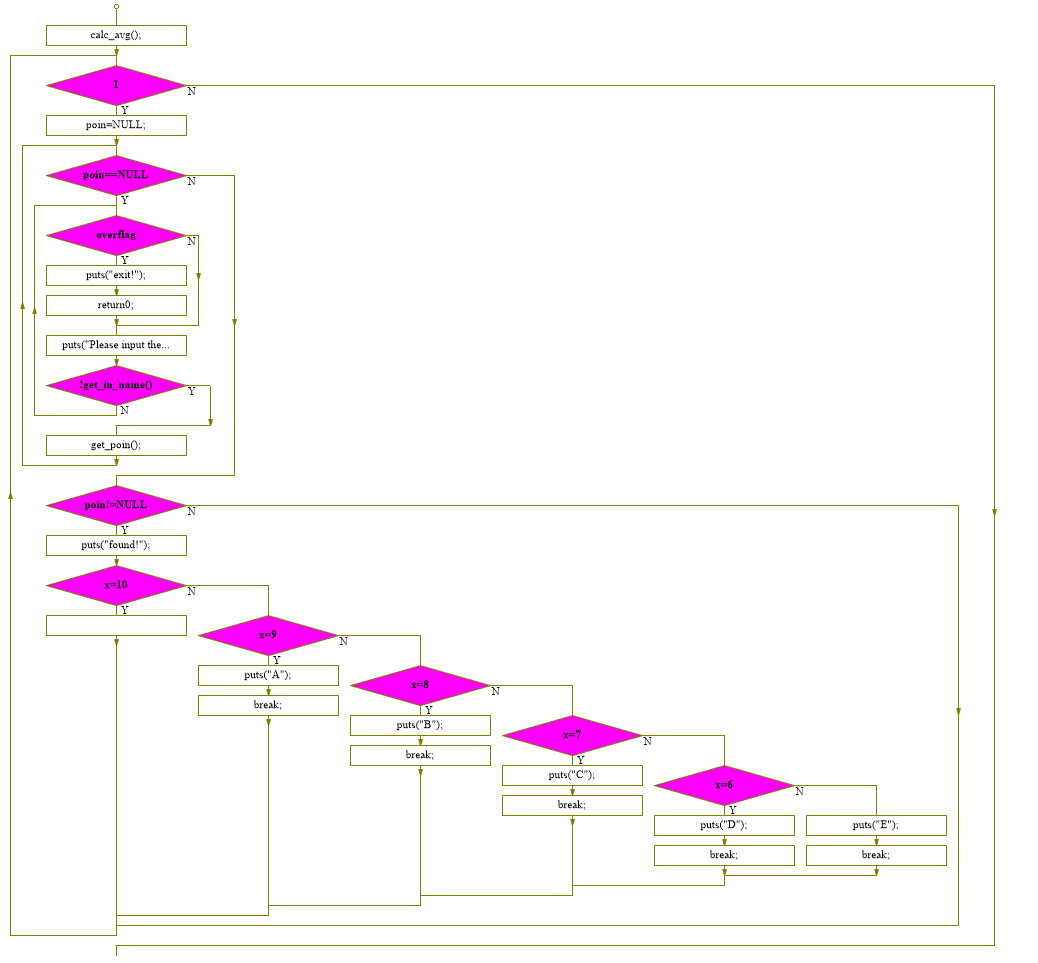
extern s:dword

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。

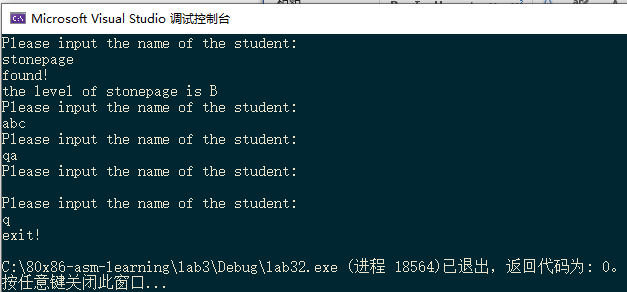
3. 输入一个列表中不存在的姓名abc，程序提示该学生不存在，并要求重新输入。

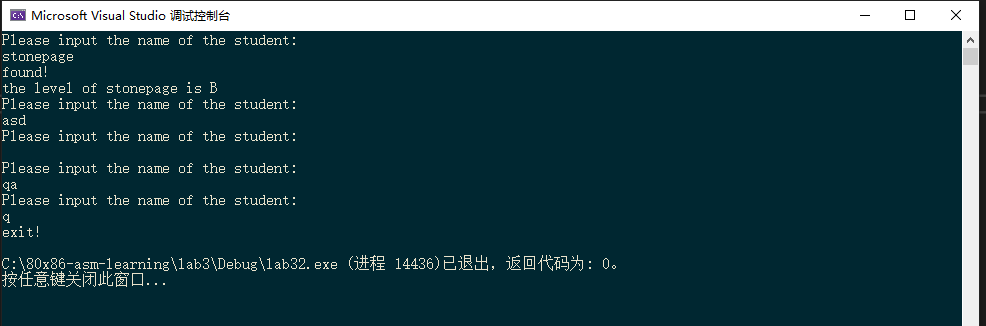
4. 输入一个列表中不存在，且以字母q为前缀的姓名qa，程序提示该学生不存在，并要求重新输入。

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。





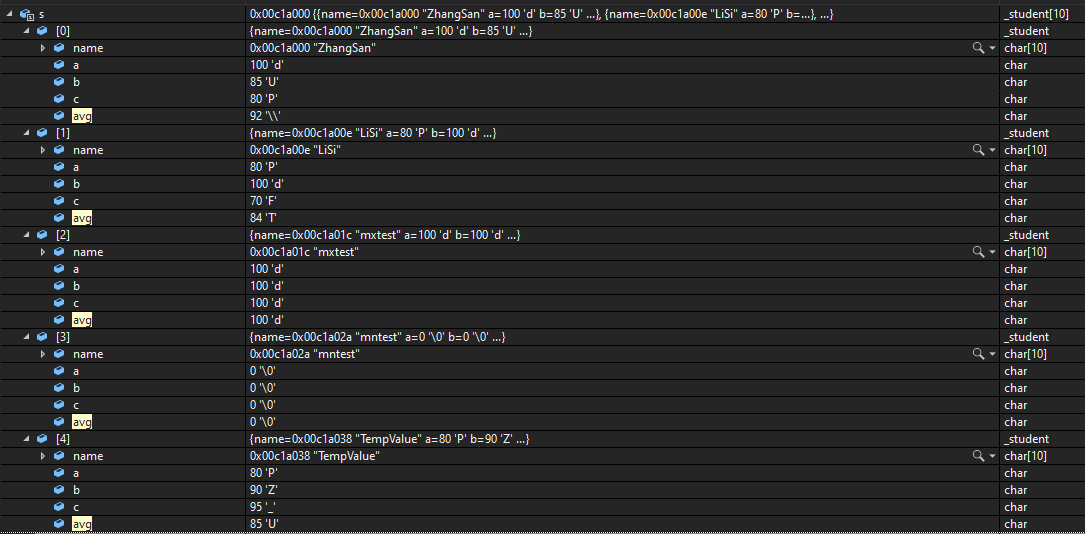
**图2初步测试记录**

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。