

**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 子程序执行设计**

**实验时间： 2017-10-24，18：30-21：50 实验地点： 南一楼804室6号实验台**

**指导教师： 许向阳**

**专业班级：信息安全 201601班**

**学 号： U201614804 姓 名： 黄永恒**

**同组学生： 无 报告日期： 2017年 10 月 25日**

**原创性声明**

  本人郑重声明：本报告的内容由本人独立完成，有关观点、方法、数据和文献等的引用已经在文中指出。除文中已经注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品或成果，不存在剽窃、抄袭行为。

特此声明！

学生签名：

日期：

成绩评定

|  |  |  |
| --- | --- | --- |
| 实验完成质量得分（70分）（实验步骤清晰详细深入，实验记录真实完整等） | 报告撰写质量得分（30分）（报告规范、完整、通顺、详实等） | 总成绩（100分） |
|  |  |  |

指导教师签字：

日期：

目 录

[1 实验目的与要求 1](#_Toc500539623)

[2 实验内容 1](#_Toc500539624)

[3 实验过程 2](#_Toc500539625)

[3.1 任务一 2](#_Toc500539626)

[3.1.1 设计思想及存储单元分配 2](#_Toc500539627)

[3.1.2 流程图 3](#_Toc500539628)

[3.1.3 源程序 5](#_Toc500539629)

[3.1.4 实验步骤 9](#_Toc500539630)

[3.1.5 实验记录与分析 10](#_Toc500539631)

[3.2 任务二 13](#_Toc500539632)

[3.2.1 源程序 13](#_Toc500539633)

[3.2.2 实验步骤 13](#_Toc500539634)

[3.2.3 实验数据记录与分析 13](#_Toc500539635)

[3.3 选做题 15](#_Toc500539636)

[3.3.1 实验步骤 15](#_Toc500539637)

[3.3.2 实验数据记录及分析 15](#_Toc500539638)

[4 总结与体会 16](#_Toc500539639)

[参考文献 17](#_Toc500539640)

# 实验目的与要求

本次实验的主要目的与要求有下面3点，所有的任务都会围绕这3点进行，希望大家事后检查自己是否达到这些目的与要求。

1. 学习子程序的设计方法，主程序与子程序之间的参数传递方法；
2. 理解高级语言程序与汇编语言程序之间的对应关系。
3. 熟悉汇编语言指令的特点，掌握代码优化的基本方法；

# 实验内容

**任务1. 用子程序实现第二次实验中的程序进行改写**

要求：

1. 用子程序 STRCMP 实现两个串的比较。两个串的首地址要通过堆栈方式传递。若两个串相同，子程序返回(AX)=1; 否则返回(AX)=0。（在调用本子程序前，先将将输入串中的0DH改为 00H,这样，两个串都是以0结束，与C语言中的strcmp函数一致）
2. 用一个子程序实现计算所有学生的平均成绩（无入口参数和出口参数）
3. 用一个子程序实现平均成绩的等级判断和判断结果显示（入口参数为分数，通过堆栈传递）

**任务2.** **观察C语言程序中函数参数的传递方法、调用方法及变量的存储空间分配方法**

在C语言程序中，仿照strcmp自己写一个函数实现两个字符串比较。对实现的程序进行调试，采用反汇编方法观察C语言语句对应生成的执行程序。

举例说明，函数调用语句翻译成机器指令有哪些？ 刚进入函数时，堆栈中存放了一些什么信息？函数参数及局部变量的分配空间是什么？

**选做题. 对任务1中的汇编源程序进行优化**

优化工作包括代码长度的优化和执行效率的优化，本次优化的重点是执行效率的优化。

**优化方法提示：**首先是通过选择执行速度较快的指令来提高性能，比如，把乘除指令转换成移位指令、加法指令等；其次，尽可能减少内循环体中的指令；第三，尽量采用32位寄存器寻址，能有更多的机会提高指令执行效率。

为展现效率的提高，可以在执行某一功能前后计时，并显示时间。另外，由于机器速度快，执行程序的时间短。为增强优化效果，可以多次（循环）执行某一功能，如多次计算所有人的平均成绩。汇编教学网站的资料下载（软件栏）中提供了“[汇编自动计时程序](http://222.20.79.217/huibian1/site/assembly/courseMaterial.files/materialInfo.jsp?id=160)”、“秒和百分秒显示disptime”子程序。

# 实验过程

## 任务一

### 设计思想及存储单元分配

1. 子程序一：字符串比较函数

设计思想：利用堆栈方式传递两个字符串首字母地址，在子程序中用两个寄存器指向这两串字符串，并用字节寄存器取出相对应的字符，逐个比较，若比较不相等，则令AX=0，并退出子程序，若比较到两个字符都是0，则令AX=1，并退出子程序，其余情况寄存器移动到下一字符，再进行比较，与以上情况相同。

存储单元分配

BUF：字节变量，存储N个学生的姓名以及成绩，每个学生分配14个字节，其中10个字节存储名字，若名字小于10个字节，则置0,剩下4个字节存储成绩，分别是语文、数学、英语以及平均值。

in\_name：字节变量，作为输入缓冲区，缓冲区分配10个字节

寄存器分配

AX：作为字符串是否相同的标志，相同为1，不同为0

SI：存储缓冲区in\_name中名字的偏移地址

DI：存储BUF的偏移地址

DH：存储输入名字的字符

DL：存储BUF中名字的字符

BP：指向栈顶，传递堆栈区的变量

1. 子程序二：计算所有学生的平均成绩

设计思想：无需传递参数，直接调用子程序。在子程序中用一个寄存器指向存储单元，并计算成绩存储地址，用字节寄存器逐个取出成绩，乘以权重，存到另一个寄存器中，最后将计算后的平均值存到相应单元后，开始计算下一个同学，直到所有同学成绩计算完，退出程序。

寄存器分配

AX：存储成绩加权和

CX：计数器，赋值N+1，判断是否计算完所有同学成绩

DI： 存储BUF的偏移地址

BX：作为公差，循环一次加一次14，计算每个同学名字的偏移地址

DL：取每项成绩

1. 子程序三：判断等级并显示

设计思想：先在主程序中开辟一块存储单元，构造一个等级表。利用堆栈方式传递需比较的分数，调用子程序，在子程序中用一个寄存器指向表地址，再将分数除以10，取商作为偏移量，加到寄存器中，直接利用九号调用显示等级。

寄存器分配

AX：存放查找学生的加权平均成绩

SI：指向存储单元等级表

DL：赋值为10，作为除数

CX：存储商，作为偏移量

### 流程图

图3.1.1是主程序流程图（精简）。

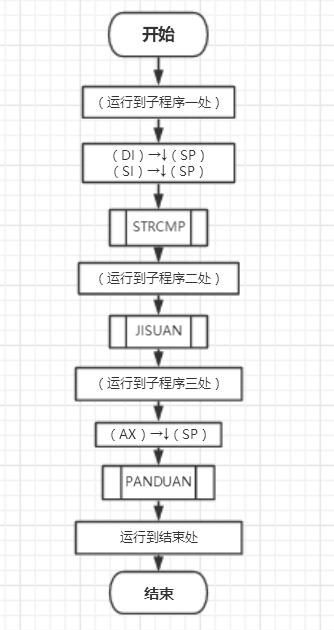


图3.1.1 主程序流程图（精简）

图3.1.2是子程序一的流程图

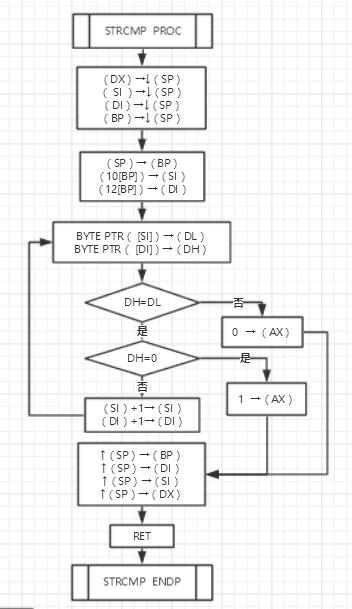


图3.1.2 子程序一流程图

图3.1.3是子程序二的流程图

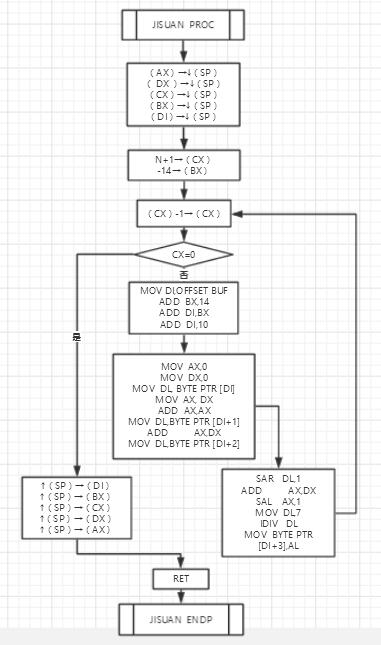


图3.1.3 子程序二流程图

图3.1.4是子程序三的流程图

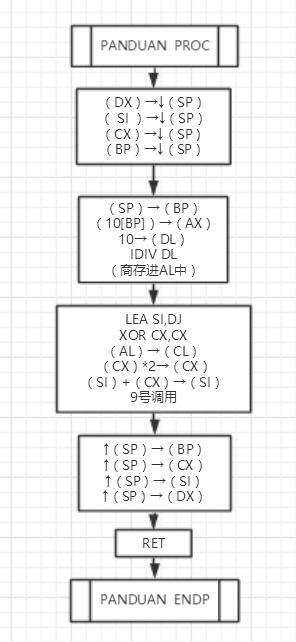


图3.1.4 子程序三流程图

### 源程序

;-------------------------------------------------------------------------

DATA SEGMENT USE16

N EQU 4

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3个已经具体定义了学生 信息的成绩表以外，其他学生的信息暂时假

定为一样的。

DB 'yongheng',0,0 ;最后一个必须是自己名字的拼音

DB 85, 85, 100, ?

PR DB 'Please input name:(# = quit) $' ;提醒输入，输入'#'退出

in\_name DB 10

DB ?

DB 10 DUP(0)

PR2 DB 'Error！ Please input again: $'

PR3 DB 'NO EXIST! $'

PR4 DB 'Grades error! $'

PR5 DB '100 $' ;若平均成绩为100，则通过字符串输出

POIN DW ?

DJ DB 'F$','F$','F$','F$','F$','F$','D$','C$','B$','A$','A$';等级表，根据成绩除以10后得

;到的商进行输出等级

DATA ENDS

;-------------------------------------------------------------------------

STACK SEGMENT STACK

DB 200 DUP(0)

STACK ENDS

;-------------------------------------------------------------------------

CODE SEGMENT USE16

ASSUME CS:CODE,DS:DATA,SS:STACK

START:

;--------------------------------------------------------------------------;附加功能1：判断成绩是否合法

MOV AX,DATA

MOV DS,AX

MOV BP,-14

MOV CX,N+1

WX2: DEC CX

JZ TSSR

LEA DI, BUF

ADD BP,14

ADD DI,BP

ADD DI,10

MOV AX,3

NX2: CMP BYTE PTR[DI], 0

JB ERR2

CMP BYTE PTR[DI], 100

JA ERR2

INC DI

DEC AX

JNZ NX2

JMP WX2

;--------------------------------------------------------------------------;功能一：输入姓名

TSSR:LEA DX, PR ;若成绩正常，则提示输入姓名

MOV AH, 9

INT 21H

SR: LEA DX, in\_name ;键盘输入名字

MOV AH,10

INT 21H

MOV DL, 0AH ;换行

MOV AH, 2

INT 21H

MOV SI, OFFSET in\_name+2 ;将SI指向缓冲区姓名起始地址

CMP BYTE PTR [SI], '#' ;附加功能：若输入‘#’，则退出程序

JZ EXIT

MOV CL,[SI-1]

LOPA: CMP BYTE PTR [SI],'A' ;附加功能：判断输入合法性

JB ERR

CMP BYTE PTR [SI],'z'

JA ERR

INC SI

DEC CX

JNZ LOPA

;---------------------------------------------------------------------------;功能二：判断有无该姓名

MOV BX,OFFSET in\_name+1

MOV CL,[BX]

MOV SI,CX

INC BX

MOV BYTE PTR [BX+SI],00H ;修改输入串中最后一个字符为’\0’

MOV CX,N+1

MOV DX,-14

WX:MOV DI,OFFSET BUF

MOV SI,OFFSET in\_name+2

DEC CX

JZ NEH ;如果查找完所有仍未找到，则提示不存在

ADD DX,14

ADD DI,DX

PUSH DI ;将两串地址压入堆栈

PUSH SI

CALL STRCMP ;调用子程序，判断两串是否相同

CMP AX,0 ;若AX为0，表示未找到，继续调用找

JZ WX

MOV DI,OFFSET BUF

ADD DI,DX

ADD DI,10

MOV POIN,DI ;将找到学生的成绩起始地址存到POIN变量中

;------------------------------------------------------------------------;功能三：计算每个学生平均成绩

CALL JISUAN ;调用子程序，计算所有同学平均成绩

;-------------------------------------------------------------------------;功能四：显示学生等级及成绩

PRINT: MOV BX,POIN

XOR AX,AX

MOV AL,BYTE PTR [BX+3]

PUSH AX ;AX中存储了需判断的成绩，压入堆栈

CALL PANDUAN ;调用子程序，判断成绩等级并显示

PRINT2: MOV DL,' ' ;附加功能2：显示学生成绩

MOV AH,2

INT 21H

CMP AL,100

JZ GOOD

MOV AH, 0

MOV DL,10

IDIV DL

ADD AL,'0'

ADD AH,'0'

MOV BL, AL

MOV BH, AH

MOV DL,BL

MOV AH,2

INT 21H

MOV DL,BH

MOV AH,2

INT 21H

MOV DL,0AH

MOV AH,2

INT 21H

JMP TSSR

GOOD: LEA DX,PR5

MOV AH,9

INT 21H

MOV DL,0AH

MOV AH,2

INT 21H

JMP TSSR ;若该同学相应操作处理完毕，重新提示输入名字

;------------------------------------------------------------------------;结束程序

EXIT: MOV AH,4CH

INT 21H

ERR: LEA DX,PR2 ;提示输入名字错误

MOV AH,9

INT 21H

JMP SR

ERR2: LEA DX,PR4 ;显示成绩错误，并结束程序

MOV AH,9

INT 21H

JMP EXIT

NEH: LEA DX,PR3 ;提示不存在

MOV AH,9

INT 21H

MOV DL,0AH

MOV AH,2

INT 21H

JMP TSSR

;-----------------------------------------------------------------------------;子程序一：字符串比较

STRCMP PROC

PUSH DX

PUSH SI

PUSH DI

PUSH BP ;保护现场

MOV BP, SP

MOV SI, 10[BP]

MOV DI, 12[BP]

BJ: MOV DH, BYTE PTR [SI]

MOV DL, BYTE PTR [DI]

CMP DH, DL

JNE A0

CMP DH, 0

JE A1

INC SI

INC DI

JMP BJ

A0: MOV AX,0

POP BP

POP DI

POP SI

POP DX ;恢复现场

RET

A1: MOV AX,1

POP BP

POP DI

POP SI

POP DX

RET

STRCMP ENDP

;-------------------------------------------------------------------------------;子程序二：计算平均成绩

JISUAN PROC

PUSH AX

PUSH DX

PUSH CX

PUSH BX

PUSH DI ;保护现场

MOV CX, N+1

MOV BX, -14

JISUAN1:MOV DI, OFFSET BUF

DEC CX

JZ S ;若所有成绩均计算完毕，则跳到显示区

ADD BX, 14

ADD DI, BX

ADD DI, 10

XOR AX, 0

XOR DX, 0

MOV DL, BYTE PTR [DI]

MOV AX, DX

ADD AX, AX

MOV DL, BYTE PTR [DI+1]

ADD AX, DX

MOV DL, BYTE PTR [DI+2]

SAR DL,1

ADD AX, DX

SAL AX, 1

MOV DL,7

IDIV DL

MOV BYTE PTR [DI+3], AL

JMP JISUAN1

S：POP DI

POP BX

POP CX

POP DX

POP AX ;恢复现场

RET

JISUAN ENDP

;-------------------------------------------------------------------------- -;子程序三：等级输出

PANDUAN PROC

PUSH DX

PUSH SI

PUSH CX

PUSH BP ;保护现场

MOV BP, SP

MOV AX, 10[BP]

LEA SI, DJ

MOV DL, 10

IDIV DL ;此时AL中存放商

XOR CX, CX

MOV CL, AL

SAL CX, 1

ADD SI, CX

MOV DX, SI ;9号调用，直接输出相应等级

MOV AH,9

INT 21H

POP BP

POP CX

POP SI

POP DX ;恢复现场

RET

PANDUAN ENDP

CODE ENDS

END START

### 实验步骤

1. 准备上机环境，编辑、汇编、连接文件DEMO1。
2. 输入TD DEMO1，进入调试窗口
3. 输入‘lisi’，看数据段in\_name缓冲区变化
4. 在PRINT: MOV BX,POIN行设置断点，运行到此处，看数据段BUF区平均值是否正确，预期为5C（92）、54（84）、54（84）、57（87）。
5. 继续调试，看成绩等级是否正确。预期输出 B
6. 继续调试，看是否输出平均成绩以及是否正确。预期输出 84
7. 退出程序，重新运行，分别输入 ‘lisik’、‘lisi66’、回车、以及‘#’，检查程序功能是否正确。预期分别输出 NO EXIST! 并提示输入名字

Error Please input again:

Error Please input again:

退出程序

### 实验记录与分析

1. 实验环境条件：i7-5500U 3.0GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。
2. 编译连接正常，如图3.1.5所示

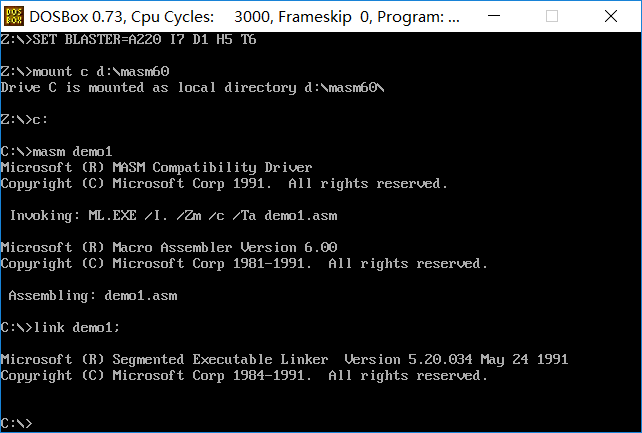


图3.1.5 正常编译连接

1. 输入td demo，进入调试器，输入lisi，观察缓冲区的数据，如图3.1.6、图3.1.7所示

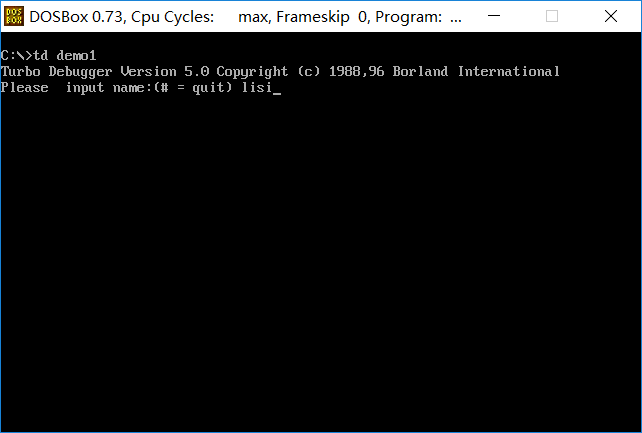


图3.1.6 输入‘lisi’

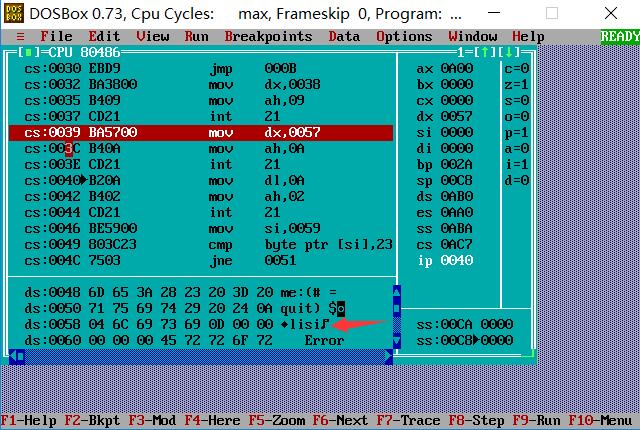


图3.1.7 缓冲区数据

1. 在PRINT: MOV BX,POIN行设置断点，运行到此处，看BUF平均值，如图3.1.8、图3.1.9所示

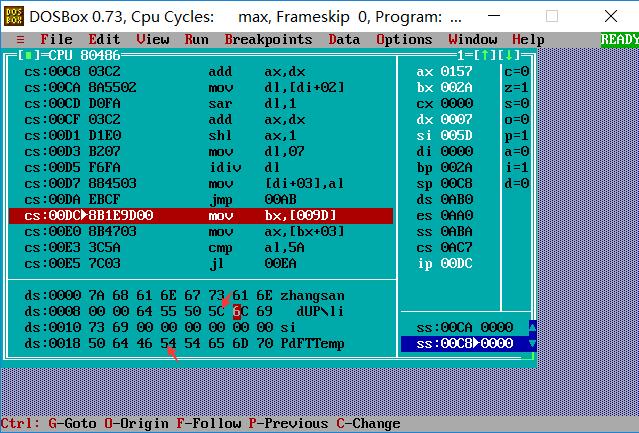


图3.1.8 BUF区数据

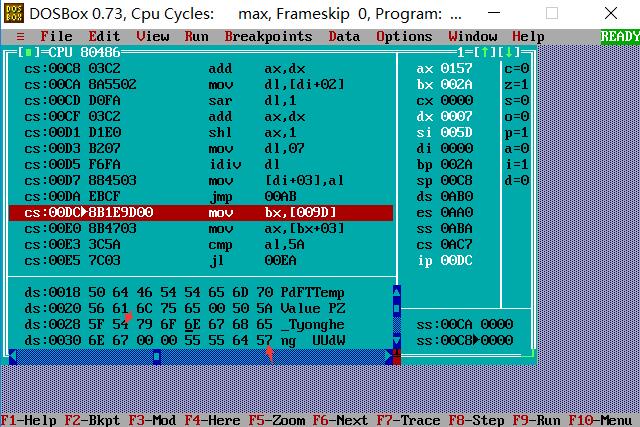


图3.1.9 BUF区数据

1. 继续调试，看成绩等级是否正确。预期输出 B，实际如图3.1.10所示

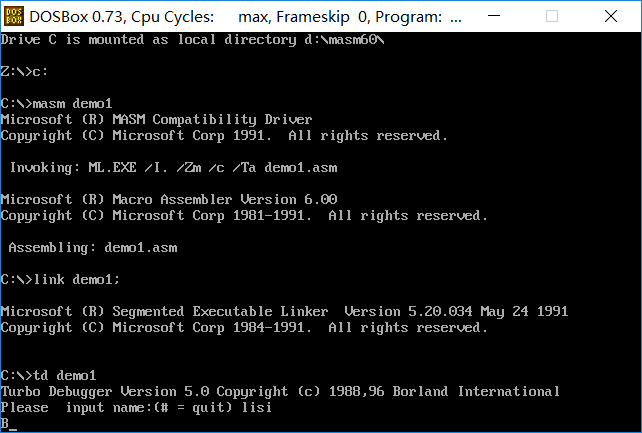


图3.1.10 成绩等级输出

1. 继续调试，看是否输出平均成绩以及是否正确。预期输出 84，实际如图3.1.11所示

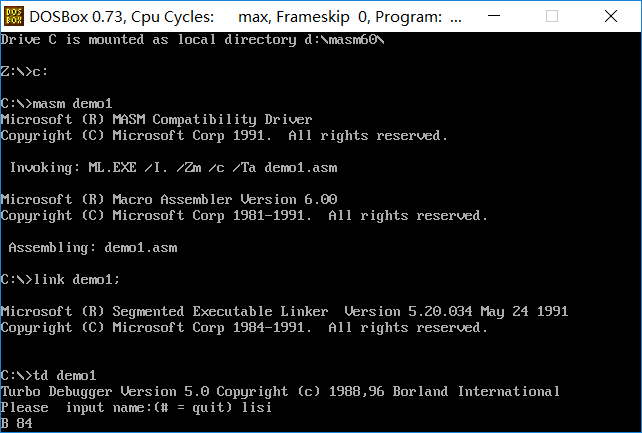


图3.1.11 成绩输出

1. 退出程序，重新运行，分别输入 ‘lisik’、‘lisi66’、回车、以及‘#’，检查程序功能是否正确。实际如图3.1.12所示

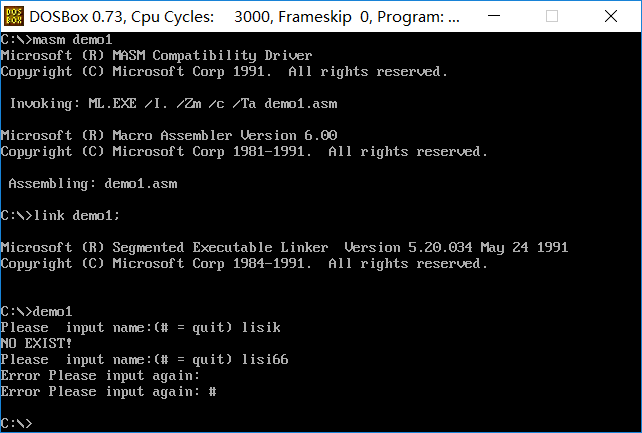


图3.1.12 输出字符正确性检查

由以上各图可知，程序执行结果与预期一致。

## 任务二

### 源程序

#include <stdio.h>

#include <stdlib.h>

int strcmp1(char \*s, char \*t)

{

int i=0;

while((s[i]==t[i])&&s[i]!='\0')

i++;

if (s[i]=='\0')

return 1;

else

return 0;

}

int main()

{

char s[]={'a','b','c','d'},t[10];

int h;

printf("Please input characters:\n");

scanf("%s",t);

h=strcmp1(s,t);

printf("%d",h);

return 0;

}

### 实验步骤

1. 在DOSBOX界面输入C语言代码，编译运行。
2. 退出BC，再次打开，点击File\DOS shell，输入TD NONAME00，进入调试界面，点击View\CPU，观察反汇编程序语言
3. 输入字符串abce
4. 调试程序，程序运行到子程序处，进入子程序，观察堆栈区以及数据区
5. 运行完程序，观察结果

### 实验数据记录与分析

1. 实验环境条件：i7-5500U 3.0GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。
2. 输入td NONAME00，进入调试器，点击View\CPU，观察反汇编程序语言，如图3.2.1所示

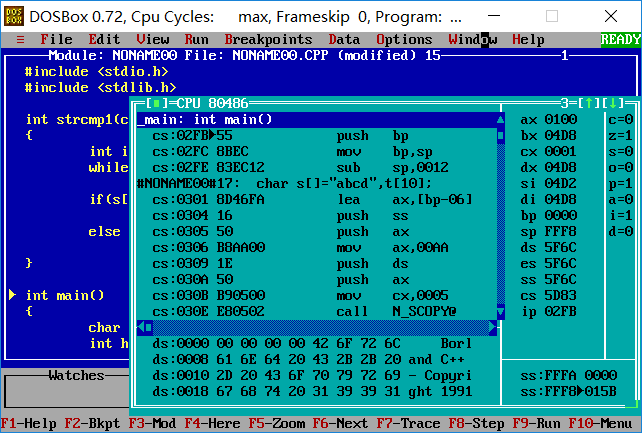


图3.2.1 C语言对应的反汇编语言

1. 输入字符串abce，如图3.2.2所示

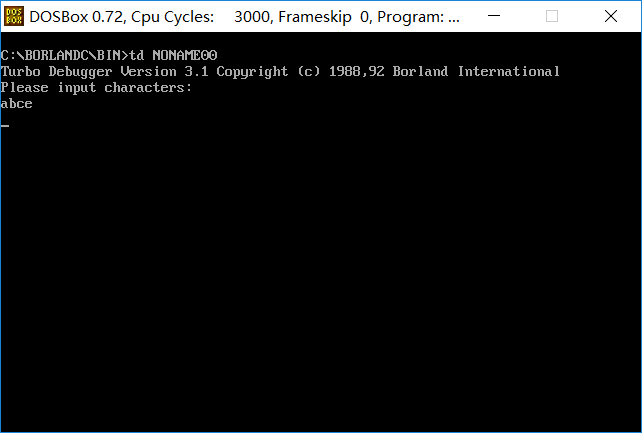


图3.2.2 输入字符串

1. 调试程序，程序运行到子程序处，进入子程序，观察堆栈区以及数据区，如图3.2.3所示

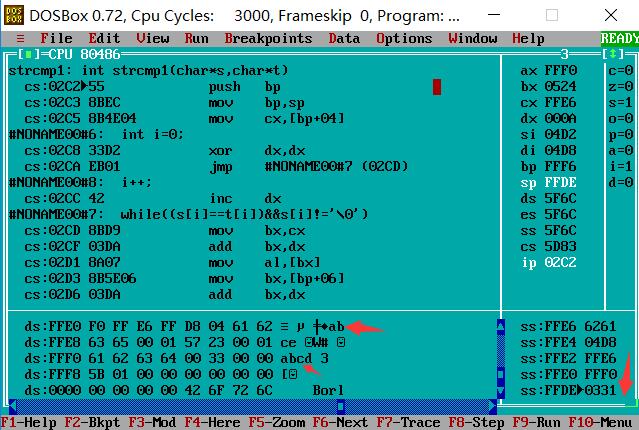


图3.2.3 进入函数时堆栈区存储值（两个串的首地址以及断点IP）

以及数据区存储值

1. 运行完程序，结果如图3.2.4所示

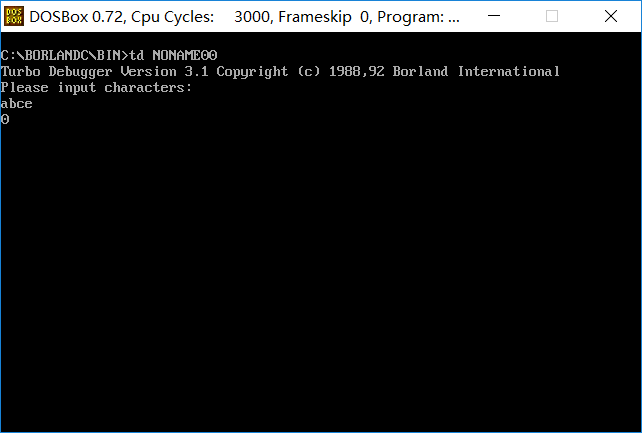


图3.2.4 运行结果

## 选做题

### 实验步骤

1. 为了使计时效果显著，可以考虑将N设为30，令循环次数为5000，调用计算平均值子程序。首先在程序未优化前调用计时功能。
2. 将子程序中乘除指令改为移位或者加减指令，将寄存器设为32位
3. 程序优化后再次调用计时功能

### 实验数据记录及分析

1. 程序未优化前调用计时功能，如图3.3.1所示

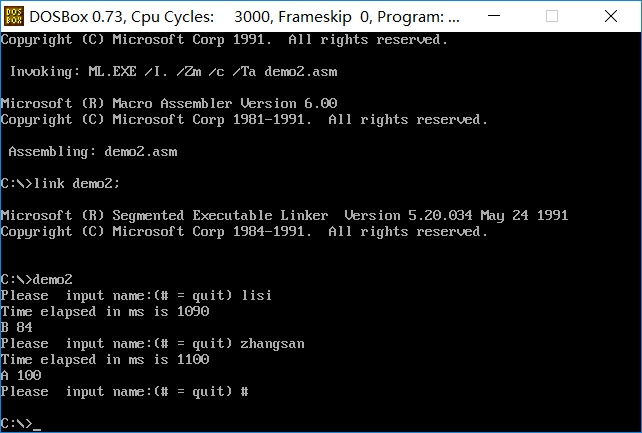


图3.3.1 程序未优化前

1. 程序优化后，如图3.3.2所示

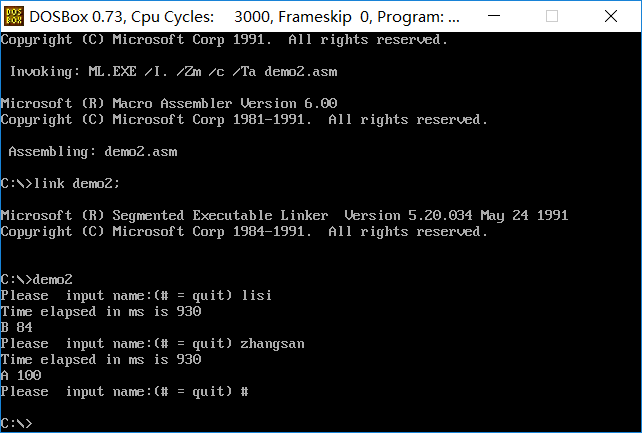


图3.3.2 程序优化后

# 总结与体会

通过本次实验，我基本掌握了子程序的设计方法，并且了解了三种传递参数方式，而且应用了堆栈传递方式。在首次编写时，我并没有完全保护现场，导致有些寄存器值在子程序中发生了变化，从而输出不了正确的值，这让我更加认识到保护现场的重要性，并且明白了C语言中主函数与功能函数的界限，即某些值仅在功能函数体中变化，并不影响在主函数中的值。

除此之外，我还通过编写C语言函数再转换为反汇编语言，了解了高级语言程序与汇编语言程序之间的关系，通过观察堆栈段的信息，我也了解了C语言调用函数是使用堆栈方式传递参数。

通过选做题，我基本掌握了优化代码的方法，如减少乘除指令，利用移位指令，减少循环体操作次数，同时也可以利用32位寄存器，加快程序运行效率。

总之，我相信自己会越来越熟练，越来越顺利！

# 参考文献

[1] 王元珍 曹忠升 韩宗芬.《80x86汇编语言程序设计》 “第三章 宏汇编语言”，“第四章 程序设计的基本方法”.华中科技大学出版社