

**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： 2017-10-31，18：30-21：50 实验地点： 南一楼804室**

**指导教师： 朱虹**

**专业班级：信息安全 201602班**

**学 号： U201614836 姓 名： 林晓斌**

**同组学生： 无 报告日期： 2017年 10 月 31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：2017.10.17

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

[实验三 子程序执行设计 3](#_Toc500881952)

[3.1 实验目的与要求 3](#_Toc500881953)

[3.2 实验内容 3](#_Toc500881954)

[3.2.1 任务1 3](#_Toc500881955)

[3.2.2 任务2 10](#_Toc500881956)

[3.3 实验小结 11](#_Toc500881957)

[3.4 参考文献 12](#_Toc500881958)

# 实验三 子程序执行设计

# 3.1 实验目的与要求

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；

理解高级语言程序与汇编语言程序之间的对应关系。

熟悉汇编语言指令的特点，掌握代码优化的基本方法；

# 3.2 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：

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）

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）

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？

3.2.1 任务1

3.2.1.1 实验思路及寄存器分配

与上次实验的思路大体一致，但是需将之前几个功能分几个子程序来写，并利用堆栈的方式实现数据的传输。但是由于子程序对寄存器的改变会影响主程序，故在进入子程序之后需要对使用到的寄存器进行保护。并且需要对堆栈的准确计算，才能取到所需的值。

1 存储单元分配

BUF1 db变量 用于存储所有学生的信息

IN\_NAME db变量 10个空间的0

POIN dw变量 10个空间的0

BUF3 db变量 存储提示找不到学生的字符串

BUF2 db变量 存储请输入要查找学生的字符串

2 寄存器分配

主程序：DX：存放in\_name，buf2的偏移地址，用于打印文字

DI:存放buf1首地址

SI:存放in\_name首地址

AX:用于寄存器传参，通过ax是否等于1来判断字符串是否一致

BX:用于存放找到的人的偏移地址

3.2.1.2 流程图

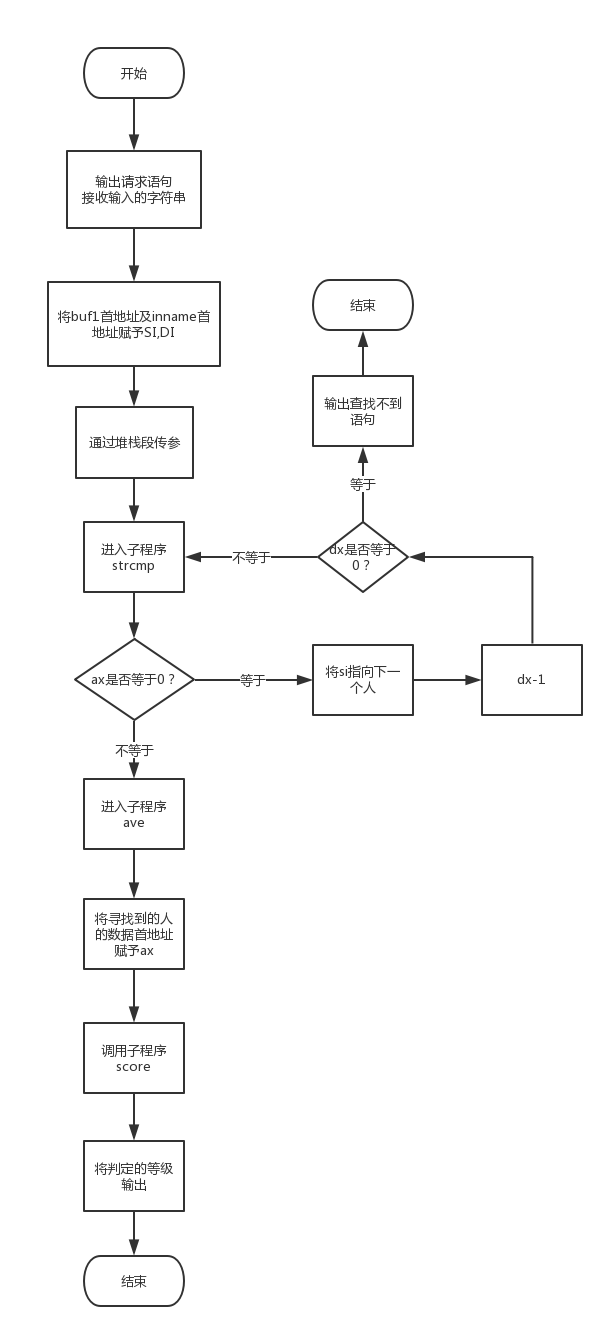


图3.1流程图

3.2.1.3 源代码

.386

STACK SEGMENT STACK USE16

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

N EQU 5

BUF1 DB 'zhangsan',0,0

DB 100,85,90,?

DB 'lisi',6 DUP(0)

DB 80,90,90,?

DB N-3 DUP('TempValue',0,80,90,95,?)

DB 'linxiaob',0,0

DB 85,95,100,?

in\_name DB 10,11 DUP(0)

POIN DW 10,11 dup(0)

BUF3 DB 0AH,0DH,'can not find the student$'

BUF2 DB 0AH,0DH,'Please enter your name:$'

DATA ENDS

CODE SEGMENT USE16

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

BEGIN: MOV AX,DATA

MOV DS,AX

LEA DX,BUF2

MOV AH,9

INT 21H

LEA DX,IN\_NAME

MOV AH,10

INT 21H

MOV DL,0AH

MOV AH,2H

INT 21H

MOV DL,0DH

MOV AH,2H

INT 21H

MOV SI,OFFSET BUF1

MOV DI,OFFSET IN\_NAME

MOV DX,N

LP1: CALL STRCMP

CMP AX,0

JE LOP ; 1

MOV CH,0

SUB SI,CX

MOV POIN,SI

CALL AVE

MOV BX,POIN

MOV AX,DS:[BX+13]

PUSH AX

CALL SCORE

MOV DH,0

MOV AH,2H

INT 21H

MOV AH,4CH ;EXIT

INT 21H

LOP: MOV CH,0

MOV BX,14

SUB BX,CX

ADD SI,BX

MOV DI,OFFSET IN\_NAME

INC DI

INC DI

DEC DX

JNZ LP1

LEA DX,BUF3

MOV AH,9H

INT 21H

MOV AH,4CH ;主程序

STRCMP PROC NEAR

PUSH SI

PUSH DI

PUSH BP

PUSH BX

PUSH CX

MOV BP,SP;

MOV DI,[BP+2]

MOV SI,[BP+4]

INC DI

INC DI

LP2: MOV CL,0

LP: MOV AL,[SI]

MOV BL,[DI]

CMP AL,BL

JNE LFALSE

INC SI

INC DI

INC CL

CMP CL,IN\_NAME+1

JNE LP

CMP BYTE PTR [SI],0

JNE LP

LRIGHT: MOV AX,1

JMP LOUT;

LFALSE: MOV AX,0

LOUT: POP CX

POP BX

POP BP

POP DI

POP SI

RET

STRCMP ENDP ; 比较的子程序

AVE PROC NEAR

PUSH SI

PUSH DI

MOV SI,OFFSET BUF1

MOV DI,N

LP3: MOV AL,[SI+10]

MOV BL,[SI+11]

MOV CL,[SI+12]

MOV AH,0

MOV BH,0

MOV CH,0

SAL AX,2

SAL BX,1

ADD AX,BX

ADD AX,CX

MOV CL,7

DIV CL

MOV [SI+13],AL

ADD SI,14

DEC DI

JNZ LP3

POP DI

POP SI

RET

AVE ENDP ; 求平均值的程序

SCORE PROC NEAR

MOV BP,SP

MOV AX,[BP+2]

MOV DX,0

CMP AL,90

JGE LEVEL1

CMP AL,80

JGE LEVEL2

CMP AL,70

JGE LEVEL3

CMP AL,60

JGE LEVEL4

MOV DL,'F'

JMP LAST

LEVEL1: MOV DL,'A'

JMP LAST

LEVEL2: MOV DL,'B'

JMP LAST

LEVEL3: MOV DL,'C'

JMP LAST

LEVEL4: MOV DL,'D'

LAST: RET

SCORE ENDP

CODE ENDS

END BEGIN

3.2.1.4 程序运行结果

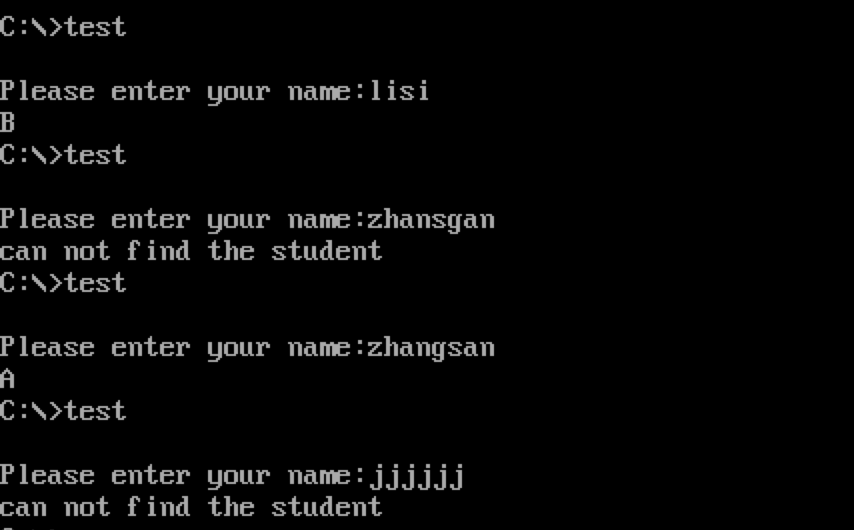


图3.2程序运行结果

输入zhangsan，输出结果为A，如图3.2所示，zhangsan的成绩为100，85，80，平均成绩为91，符合结果。

输入lisi，输出结果为B，如图3.2所示，lisi的成绩为85，90，90，平均成绩为88，符合结果。

输入jjjjjj，查无此人，故输出错误提示信息，如图3.2.

3.2.2 任务2

3.2.2.1 c程序源代码

#include <stdio.h>

#include <stdlib.h>

int strcmp (char \*,char \*);

int main(){

char a[10] = {0};

char b[10]={0};

int c;

scanf("%s",a);

scanf("%s",b);

c = strcmp(a,b);

printf("%d",c);

return 0;

}

int strcmp(char \* a,char \* b){

int i;

for(i=0;i<=9;i++)

if(a[i]!=b[i])return 0;

return 1;

}

3.2.2.2 反汇编

1. 函数调用语句翻译成机器指令有哪些？

调用语句在反汇编中为call strcmp（如图3.3），在此之前先对数据压栈，利用了堆栈传参的方法。

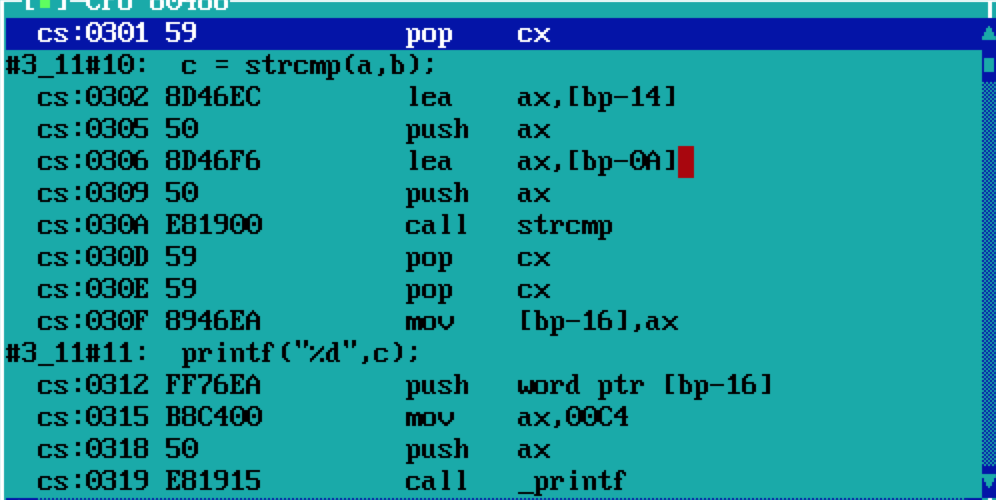


图3.3调用语句

2.刚进入函数时，堆栈中存放了一些什么信息？

刚进入strcmp函数中，堆栈存放了数组a与数组b的首地址（如图3.3与图3.4）

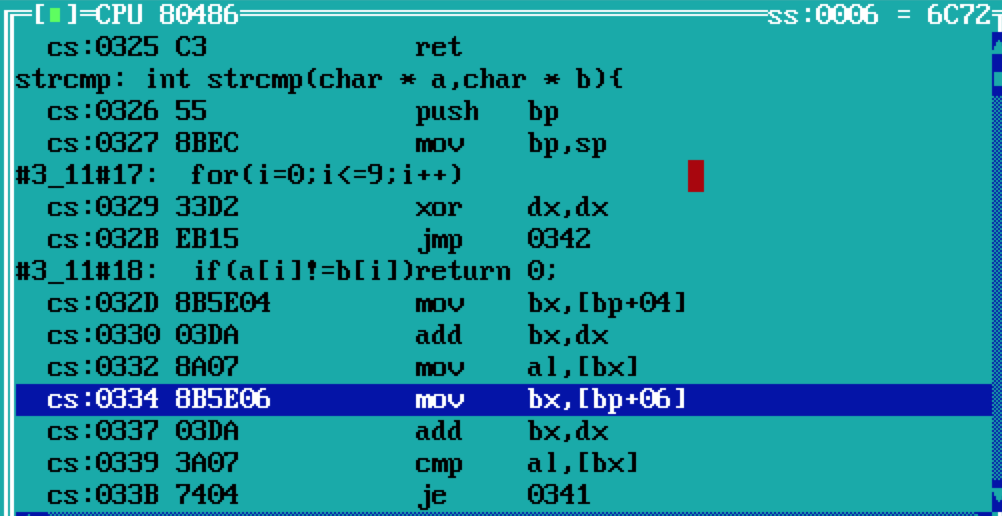


图3.4堆栈区参数

3.函数参数及局部变量的分配空间是什么？

函数参数与局部变量均在堆栈中分配了存储空间。[bp-0A]为a数组分配的空间的首地址，[bp-14]为b数组分配空间的首地址，[bp-16]为变量c的首地址如图3.5所示。

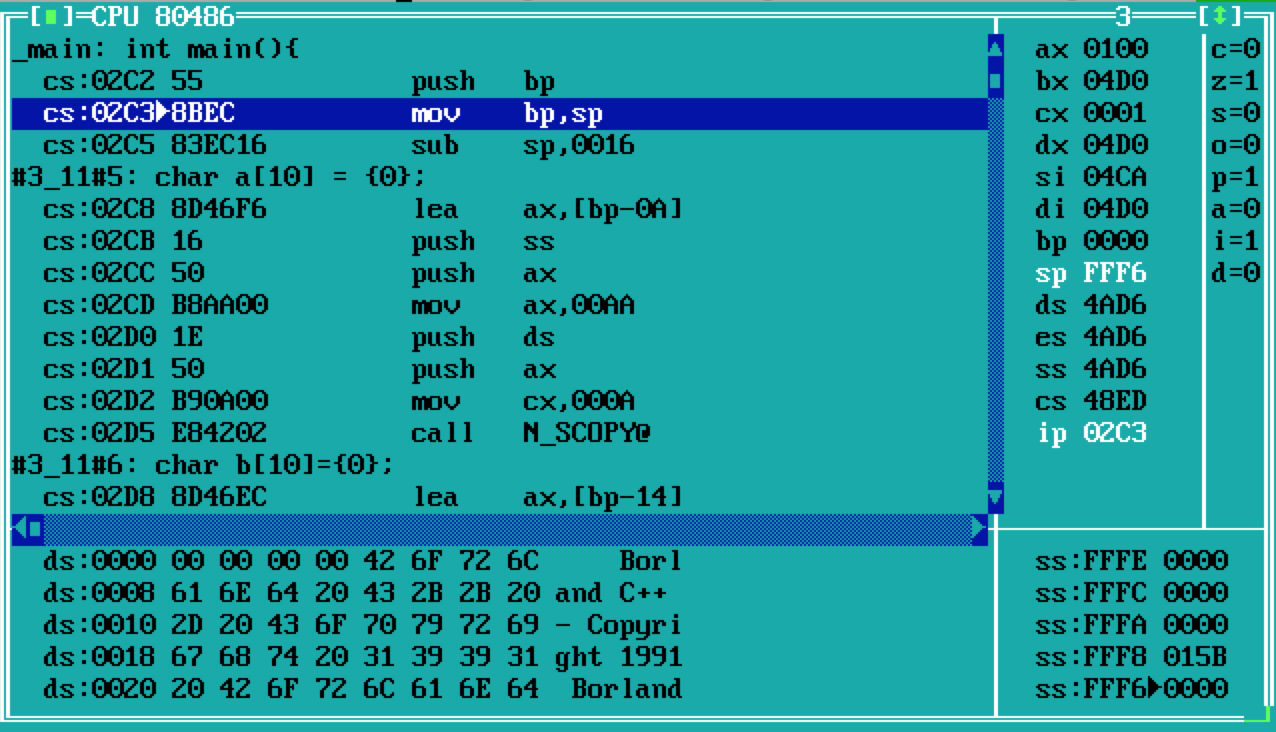


图3.5函数参数与局部变量的分配空间

# 3.3 实验小结

本次试验主要是对上次试验的优化，模块化。通过对子程序的调用，可用时主程序变得更加容易阅读，意义更加清晰，同时，子程序的调用也使得子程序更加专一，更加具有健壮性。但是一开始上手总是有点不太习惯，总是忘记进入子程序时会将ret的地址存在堆栈的顶端，所以在推出堆栈数据时总是出错，以至后来手忙脚乱。再一个就是寄存器的保存，倘若不使用寄存器的保护，极有可能会对主程序（上一级程序）中寄存器保存的数据进行破坏，故应该将子程序将会使用到的寄存器在进入子程序时推入堆栈段，对其进行保护，最后还得记得推出到原寄存器。

再就是见识到了c语言的反汇编程序，发现其也大部分是利用堆栈段对数据进行传参，并且其子程序strcmp十分注重寄存器的保护，虽然看起来很麻烦，但却也一丝不苟。

总体来说，注意三点：

1. 尽量让子程序功能专一，健壮
2. 要将子程序中可能使用的寄存器进行保护
3. 熟悉了解堆栈段中的数据情况

# 3.4 参考文献

[1] 王元珍、曹忠升、韩宗芬.80X86汇编语言程序设计(第一版).华中科技大学出版社，2005年:1-348

[2] 许向阳.80X86汇编语言程序设计上机指南(第一版).华中科技大学出版社,2007年:1-224