

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验四** 模块化程序设计

**实验时间： 2017-11-7，18：30-21：50 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： 龙洲鸣 报告日期： 2017年 11 月 7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

[实验四模块化程序设计 3](#_Toc500882076)

[4.1实验目的与要求 3](#_Toc500882077)

[4.2实验任务 3](#_Toc500882078)

[4.3实验内容 4](#_Toc500882079)

[4.3.1任务一 4](#_Toc500882080)

[4.3.2 任务二 20](#_Toc500882081)

[4.4实验小结 26](#_Toc500882082)

[4.5参考文献 26](#_Toc500882083)

# 实验四模块化程序设计

# 4.1实验目的与要求

掌握子程序设计的方法与技巧，熟悉子程序的参数传递方法和调用原理；

掌握宏指令、模块化程序的设计方法;

掌握较大规模程序的合作开发与调试方法；

掌握汇编语言程序与C语言程序混合编程的方法；

熟悉C编译器的基本优化方法;

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。

# 4.2实验任务

任务1 宏与子程序设计

进一步修改与增强实验三的学生成绩查询程序的功能。程序执行时首先显示一个功能菜单: 然后，根据用户选择的菜单项，完成相应的功能。

1=录入学生姓名和各科考试成绩

2=计算平均分

3=按平均分排序从高到低排序，排序结果仍存放在原缓冲区中

4=输出成绩单

5=程序退出。

要求：

2人一组，一人负责包括菜单显示、程序退出在内的主程序，以及菜单中的功能1和2；另一人负责菜单中的功能3和4。各自汇编自己的模块，然后连接生成一个程序。

录入学生姓名和各科考试成绩时，首先显示录入的是第几个学生的信息，然后分别在提示之后输入姓名和各科成绩。所有学生信息录入完毕后回到菜单显示的位置。姓名及考试成绩的存放、平均分的计算，按照实验三的要求。

按平均分排序从高到低排序，排序结果仍存放在原缓冲区中。

输出成绩单时，依次显示每个学生的姓名、各科成绩，平均成绩。可在排序前调用，也可在排序后调用。

将9号和10号DOS系统功能调用定义成宏指令并调用。使用子程序完成各模块的功能。实现串、数转换时，也应使用子程序来完成。

任务2：在C语言程序中调用 汇编语言实现的函数

对于任务1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，其他功能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。

要求与提示：

(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。请大家选择自己熟悉的C语言开发环境并查找相关的资料完成本实验。

观察主、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。

对混合编程形成的执行程序，用调试工具观察由C语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。

# 4.3实验内容

4.3.1任务一

4.3.1.1设计思想及存储单元分配

通过主程序调用四个子程序来完成任务，子程序一二只需在文件中调用，而子程序三四需要在外部文件进行调用，故需将两个文件进行联合汇编，并且在编写子程序时需要注意对寄存器的保护，以及子程序的对主程序的数据调用。

存储单元分配：

K:存储了常数0

X：存储了常数16，作为一个学生的总数据长度

Buf：buf段定义了每个学生的所有数据，按前十个字节为学生姓名，之后三个字节为三门学科成绩，再往后一个字节存储为学生的平均成绩，最后一个字存储了学生的排名。

Crlf：用于存储换行操作

Score：用于暂时存储10号输入的成绩值

in\_name：用于暂时存储10号输入的学生姓名

msg：用于存储提示信息

str：用于存储提示信息及错误提示

主程序寄存器分配：

DX: 用于存储10号调用所需的偏移地址

SI,DI:用于将输入数据缓冲区的数据复制到buf数据段

AL:用于1号调用，并进行比对以进行子程序的跳转

BX:用于对子程序所需的数据进行压栈

子程序一寄存器分配：

cx：用于学生人数循环计数，当cx=0时跳出循环

cx：用于存储score中的数据，并将其处理赋予buf段中的相应位置

bx：用于存储buf的偏移地址

子程序二寄存器分配：

si：用于存储学生数据段的偏移量

bx：用于存储buf段首地址

cl：用于存储除数

dl：用于取出score中的数据

ax：用于保存被除数

al：保存除法完成后的结果

4.3.1.2 流程图

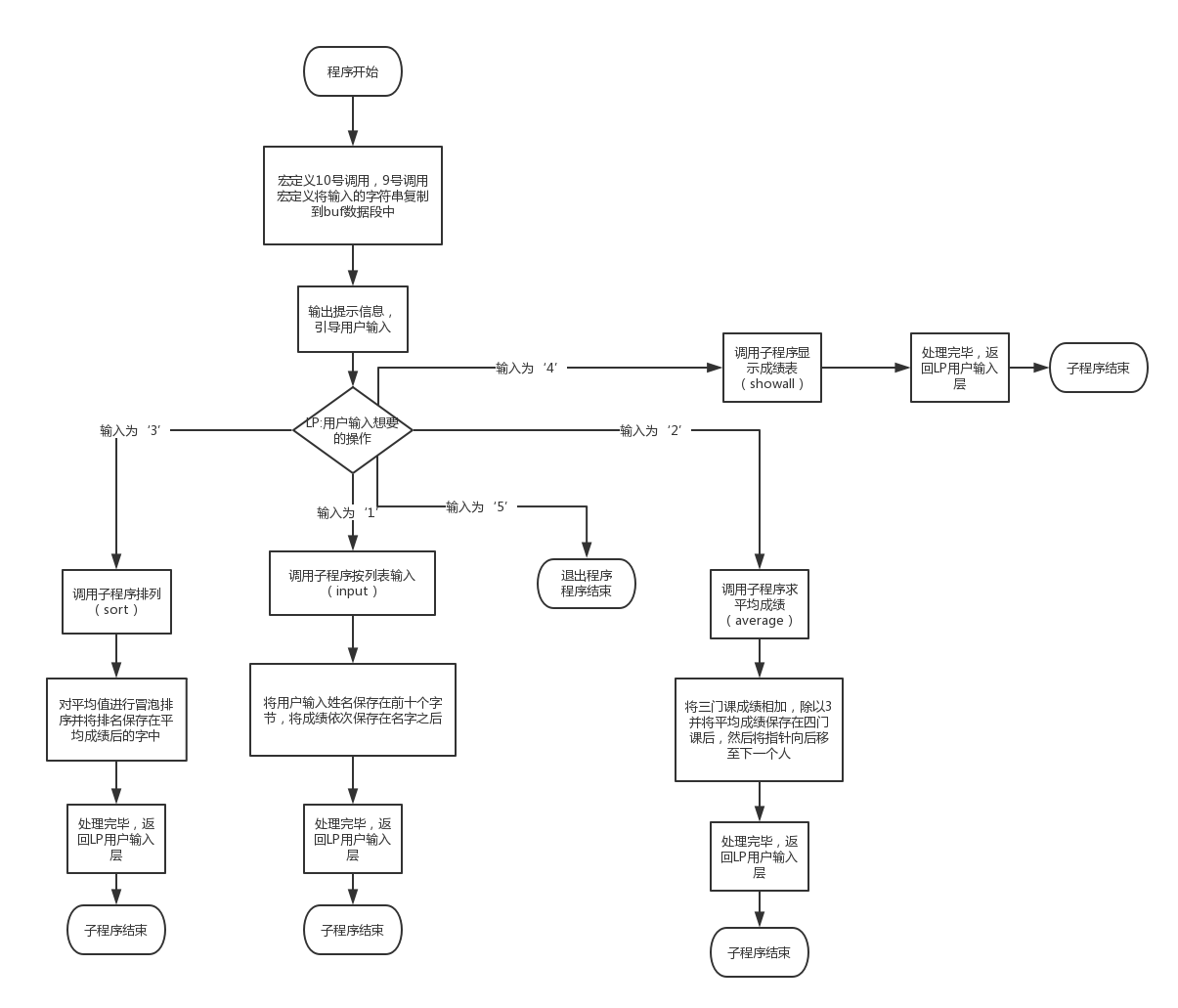


图 1实验4程序流程图

4.3.1.3 源程序

林晓斌（任务一二主程序以及两文件的汇编链接）：

PUBLIC CRLF

PUBLIC BUF

PUBLIC X

PUBLIC IN\_NAME

EXTERN SHOWALL:NEAR

EXTERN SORT:NEAR

.386

STACK SEGMENT STACK USE16 PARA STACK 'STACK'

DB 200 DUP(0)

STACK ENDS

DATAS SEGMENT USE16 PARA PUBLIC 'DATAS'

NUM DB 1,?,1 DUP(0)

N EQU 5

K EQU 0

X EQU 16

BUF DB 10 DUP(0) ;学生姓名，不足10个字节的部分用0填充

DB 100,85,80,40 ; 平均成绩还未计算

DW 0

DB 10 DUP(0)

DB 80,100,70,20

DW 0

DB N-3 DUP(10 DUP(0),80,90,95,30,0,0) ;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 10 DUP(0) ;最后一个必须是自己名字的拼音

DB 85,85,100,70

DW 0

POIN DW 0

CRLF DB 0AH,0DH,'$';回车换行

CHAR DB 0

SCORE DB 3,?,3 DUP(0)

in\_name DB 20,?,20 DUP(0)

MSG1 DB 'Please choose the number you want.$'

MSG2 DB 0AH,0DH,'1=input,2=calculate the ave,3=rank,4=output,5=quit$'

STR1 DB 'Please input student name:$'

STR2 DB 'Unexpected character or too long string detected!$'

STR3 DB 'A variate has not been initialized!$'

STR4 DB 'Can not find match name!$'

STR5 DB 'The average score is $$$$'

STR6 DB 'Overflow score detected!$'

STR7 DB 'Score validation succeed.$'

STR8 DB 'You are inputing no.1 student.$'

STR9 DB 'Please input scores.$'

STR10 DB 'You are inputing no.2 student.$'

STR11 DB 'You are inputing no.3 student.$'

STR12 DB 'You are inputing no.4 student.$'

STR13 DB 'You are inputing no.5 student.$'

STR14 DB "Score: Average score:$$$$"

Temp DB " "

Temp1 DB " "

DATAS ENDS

CODES SEGMENT USE16 PARA PUBLIC 'CODE'

ASSUME CS:CODES,DS:DATAS,SS:STACK

BEGIN: MOV AX,DATAS

MOV DS,AX

MOV ES,AX

WRT MACRO A

LEA DX,A

MOV AH,9

INT 21H

ENDM

READ MACRO A

LEA DX,IN\_NAME

MOV AH,10

INT 21H

ADD DX,2

MOV SI,DX

LEA DI,A

MOV CX,9

CLD

REP MOVSB

ENDM

READ1 MACRO A

LEA DX,A

MOV AH,10

INT 21H

ENDM

SAT: WRT MSG1

WRT MSG2

WRT CRLF

MOV AH,1

INT 21H

CMP AL,'5'

JGE QUIT

CMP AL,'4'

JE SHOW

CMP AL,'3'

JE SOT

CMP AL,'2'

JE AVE

CMP AL,'1'

JE LI

AVE: CALL AVERAGE

JMP SAT

LI: CALL INPUT

JMP SAT

SHOW: MOV BX,OFFSET BUF

PUSH BX

MOV BX,5

PUSH BX

CALL SHOWALL

JMP SAT

SOT: MOV BX,OFFSET BUF

PUSH BX

MOV BX,5

PUSH BX

CALL SORT

JMP SAT

INPUT PROC NEAR

PUSH DI

PUSH AX

PUSH CX

PUSH BX

MOV CX,5

MOV BX,OFFSET BUF

INPUT1: PUSH CX

WRT CRLF

CMP CX,5

JE S8

CMP CX,4

JE S10

CMP CX,3

JE S11

CMP CX,2

JE S12

CMP CX,1

JE S13

S8: WRT STR8

WRT CRLF

WRT STR1

READ BUF

JMP LPP

S10: WRT STR10

WRT CRLF

WRT STR1

READ BUF+16

JMP LPP

S11: WRT STR11

WRT CRLF

WRT STR1

READ BUF+32

JMP LPP

S12: WRT STR12

WRT CRLF

WRT STR1

READ BUF+48

JMP LPP

S13: WRT STR13

WRT CRLF

WRT STR1

READ BUF+64

JMP LPP

LPP: WRT CRLF

WRT STR9

READ1 SCORE

MOV CX,0

MOV CH,SCORE+3

CMP CH,0DH

JE LPP1

MOV CL,SCORE+2

SUB CL,30H

MOV CH,0

IMUL CX,10

MOV CH,SCORE+3

SUB CH,30H

ADD CL,CH

MOV CH,0

MOV [BX+10],CX

JMP LPPP1

LPP1: MOV CL,SCORE+2

SUB CL,30H

MOV CH,0

MOV [BX+10],CX

LPPP1: WRT CRLF

WRT STR9

READ1 SCORE

MOV CX,0

MOV CH,SCORE+3

CMP CH,0DH

JE LPP2

MOV CL,SCORE+2

SUB CL,30H

MOV CH,0

IMUL CX,10

MOV CH,SCORE+3

SUB CH,30H

ADD CL,CH

MOV CH,0

MOV [BX+11],CX

JMP LPPP2

LPP2: MOV CL,SCORE+2

SUB CL,30H

MOV CH,0

MOV [BX+11],CX

LPPP2: WRT CRLF

WRT STR9

READ1 SCORE

MOV CX,0

MOV CH,SCORE+3

CMP CH,0DH

JE LPP3

MOV CL,SCORE+2

SUB CL,30H

MOV CH,0

IMUL CX,10

MOV CH,SCORE+3

SUB CH,30H

ADD CL,CH

MOV CH,0

MOV [BX+12],CX

JMP LPPP3

LPP3: MOV CL,SCORE+2

SUB CL,30H

MOV CH,0

MOV [BX+12],CX

LPPP3: ADD BX,X

POP CX

DEC CX

JNZ INPUT1

POP BX

POP CX

POP AX

POP DI

RET

INPUT ENDP

AVERAGE PROC NEAR

PUSH BX

PUSH AX

PUSH SI

PUSH DX

PUSH CX

MOV SI,0

MOV CL,3

MOV BX,OFFSET BUF

LP: MOV AX,0

MOV DX,0

MOV AL,[BX+SI+10]

MOV DL,[BX+SI+11]

ADD AX,DX

MOV DL,[BX+SI+12]

ADD AX,DX

DIV CL

MOV AH,0

MOV [BX+SI+13],AL

ADD SI,16

CMP SI,80

JNE LP

POP CX

POP DX

POP SI

POP AX

POP BX

RET

AVERAGE ENDP

QUIT: MOV AH,4CH ;EXIT

INT 21H

CODES ENDS

END BEGIN

龙洲鸣（任务三四以及输入合法检验）：

NAME FUNC

PUBLIC ShowALL,SORT

.386

DATAS SEGMENT USE16 PARA PUBLIC 'D1'

N EQU 5

BUF DB 10 DUP(0) ;学生姓名，不足10个字节的部分用0填充

DB 100,85,80,89 ; 平均成绩还未计算

DW 0

DB 10 DUP(0)

DB 80,100,70,92

DW 0

DB N-3 DUP(10 DUP(0),80,90,95,73,0,0) ;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 10 DUP(0) ;最后一个必须是自己名字的拼音

DB 85,85,100,84

DW 0

BUFFER DB 16 DUP(0)

POIN DW 0

CRLF DB 0AH,0DH,"$";回车换行

CHAR DB 0

in\_name DB 20,?,20 DUP(0)

STR1 DB "Please input student name:$"

STR2 DB "Unexpected character or too long string detected!$"

STR3 DB "A variate hasn't been initialized!$"

STR4 DB "Can't find match name!$"

STR5 DB "The average score is $$$$"

STR6 DB "Overflow score detected!$"

STR7 DB "Score validation succeed.$"

STR8 DB "Name:$$$$$$$$$$"

STR9 DB "Score: Average score:$$$$"

STR10 DB "Score: Average score:$$$$"

Temp DB " "

Temp1 DB " "

DATAS ENDS

STACKS SEGMENT STACK PARA USE16

DB 100 DUP(0)

STACKS ENDS

CODES SEGMENT USE16 PARA PUBLIC 'CODE'

ASSUME CS:CODES,DS:DATAS,SS:STACKS

START:

MOV AX,DATAS

MOV DS,AX

MOV ES,AX

LEA SI,BUF

PUSH SI

PUSH 5

CALL ShowALL

;要求栈中传入2个参数，首先压入的是成绩表首地址，后压入的是成绩表的人数

SORT PROC NEAR

;形成栈帧

PUSH BP

MOV BP,SP

;保护环境，开辟空间

PUSHA

;SI为成绩表的首地址

MOV SI,[BP + 6]

MOV CX,[BP + 4]

DEC CX

MOV DX,1

;AL用于存放临时成绩变量

;+13后指向平均成绩

ADD SI,13

BubbleSort:

CMP DX,1

JNE SortExit

XOR DX,DX

PUSH CX

For1:

ADD SI,16

PUSH SI

MOV AL,[SI]

LEA DI,[SI - 16]

MOV AH,[DI]

CMP AL,AH

JBE For1Continue

PUSH CX

PUSH DI

PUSH SI

PUSH DI

SUB SI,13

LEA DI,BUFFER

MOV CX,16

CLD

REP MOVSB

POP SI

POP DI

SUB SI,13

SUB DI,13

MOV CX,16

CLD

REP MOVSB

POP DI

SUB DI,13

LEA SI,BUFFER

MOV CX,16

CLD

REP MOVSB

MOV DX,1

POP CX

For1Continue:

POP SI

DEC CX

JNZ For1

ForEnd:

POP CX

DEC CX

MOV SI,[BP + 6]

ADD SI,13

JMP BubbleSort

MOV CX,[BP + 4]

MOV BX,0

PM:

LEA SI,BUF

ADD SI,14

MOV WORD PTR [SI],BX

INC BX

CMP BX,CX

JNZ PM

SortExit:

POPA

MOV SP,BP

POP BP

RETN 4

SORT ENDP

;首先压入的参数是成绩表的首地址，之后是人数

ShowALL PROC NEAR

;形成栈帧

PUSH BP

MOV BP,SP

;保护环境

PUSHA

MOV SI,[BP + 6]

MOV CX,[BP + 4]

Show1:

;SI为成绩表的首地址

PUSH CX

PUSH SI

PUSH SI

;换行

LEA DX,CRLF

MOV AH,09H

INT 21H

;显示姓名

LEA DX,STR8

ADD DX,5

MOV DI,DX

MOV CX,9

CLD

REP MOVSB

LEA DX,STR8

POP SI

MOV AH,09H

INT 21H

;换行

LEA DX,CRLF

MOV AH,09H

INT 21H

;显示成绩和平均成绩

LEA AX,[SI + 10]

LEA DX,[STR9 + 7]

PUSH AX

PUSH DX

CALL IntToString

INC AX

ADD DX,4

PUSH AX

PUSH DX

CALL IntToString

INC AX

ADD DX,4

PUSH AX

PUSH DX

CALL IntToString

INC AX

LEA DX,[STR9 + 35]

PUSH AX

PUSH DX

CALL IntToString

LEA DX,STR9

MOV AH,09H

INT 21H

;换行

LEA DX,CRLF

MOV AH,09H

INT 21H

LEA SI,STR10

LEA DI,STR9

MOV CX,39

CLD

REP MOVSB

POP SI

ADD SI,16

POP CX

DEC CX

JNZ Show1

POPA

MOV SP,BP

POP BP

RETN 4

ShowALL ENDP

;第一个参数为字节数据，第二个压入的参数为要写入的位置

IntToString PROC NEAR

;形成栈帧

PUSH BP

MOV BP,SP

;保护环境

PUSHA

;SI为要写入的位置

XOR AX,AX

MOV SI,[BP + 6]

MOV AL,[SI]

XOR CX,CX

;做除法得商和余数，AX放商，DX放余数

DIVF:

INC CX

XOR DX,DX

MOV BX,0AH

DIV BX

PUSH DX

CMP AX,0

JNE DIVF

LEA BX,Temp

IntToChar1:

POP AX

DEC CX

ADD AX,'0'

MOV BYTE PTR [BX],AL

INC BX

CMP CX,0

JNE IntToChar1

LEA SI,Temp

MOV DI,[BP + 4]

MOV CX,3

Copy:

MOV AL,[SI]

CMP AL,32

JE CopyE

MOV [DI],AL

CopyE:

INC SI

INC DI

DEC CX

JNZ Copy

LEA SI,Temp1

LEA DI,Temp

MOV CX,3

CLD

REP MOVSB

POPA

MOV SP,BP

POP BP

RETN 4

IntToString ENDP

CODES ENDS

END START

4.3.1.4程序运行结果及测试

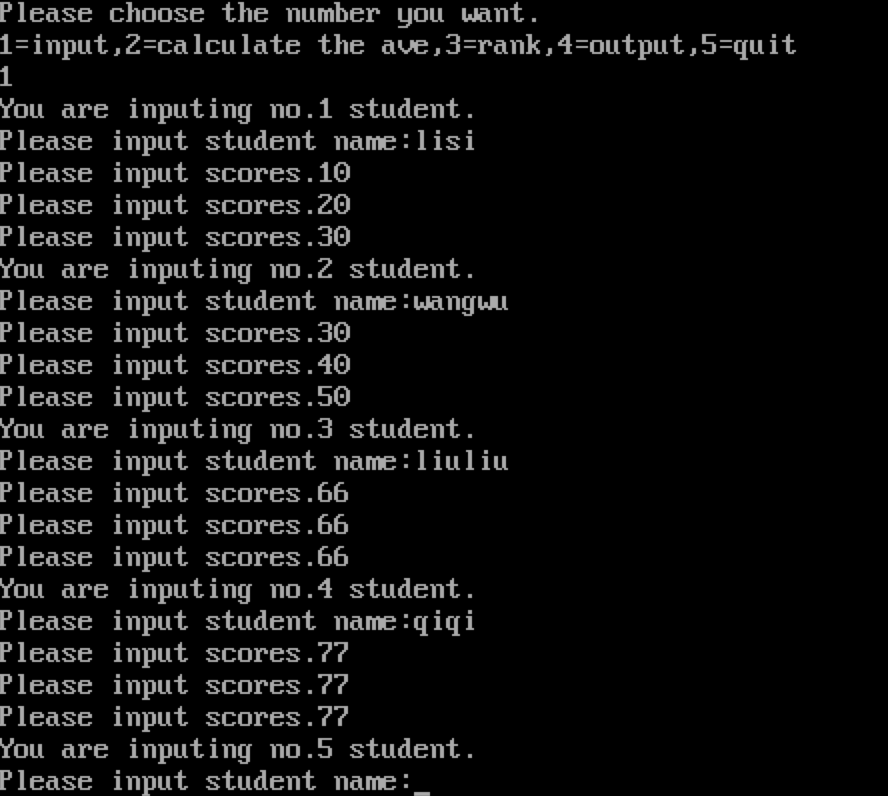


图4.1程序输入界面

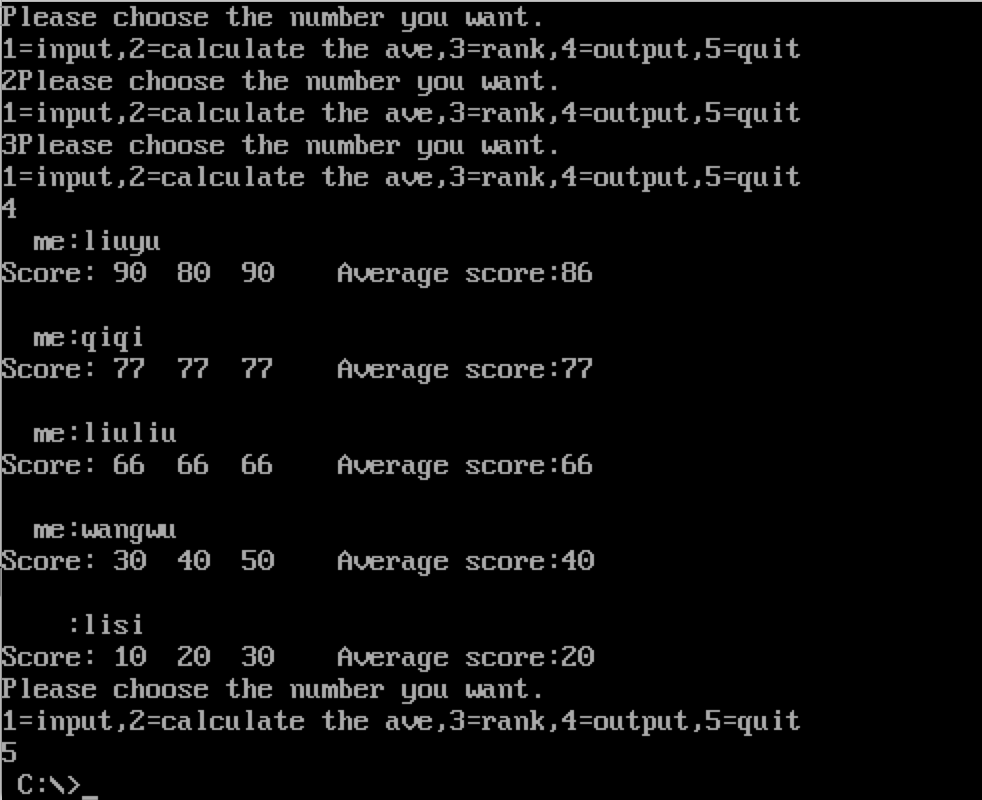


图4.2程序输出界面

4.3.2 任务二

4.3.2.1设计思想及存储单元分配

在c语言文件中实现输入，然后再调用汇编文件，使其实现之前演示的内容。编写两个文件最大的问题就是两个文件之间的传参，一开始想用堆栈传参，但是太复杂，写着写着经常出问题，于是后来就改用公共变量传值。演示系统借用了数据结构试验使用的演示系统。

4.3.2.2 源程序：

C语言程序：

#include "stdio.h"

#include "stdlib.h"

#define N 5

int PARA1;

struct student {

char rank;

char name[10];

char chinese;

char english;

char math;

char average;

};

struct student stu[N]={' '};

extern void ZBB();

extern void SOR();

void input(){

int a,i;

for (a=1; a<=5; a++) {

printf("please input no.%d students information\n",i);

i--;

scanf("%s",stu[i].name);

scanf("%d",(int\*)&stu[i].chinese);

scanf("%d",(int\*)&stu[i].english);

scanf("%d",(int\*)&stu[i].math);

}

}

void output(){

int i,j=0;

int a=5;

for (a=5; a>=1; a--) {

while (i<=4) {

if(stu[j].rank==i+1){

printf("%-5s",stu[j].name);

printf("%-5d",stu[j].chinese);

printf("%-5d",stu[j].english);

printf("%-5d",stu[j].math);

printf("%-5d\n",stu[j].average);

j=0;

i++;

}

else j++;

}

}

}

int main(void) {

char c;

rintf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" 1=输入学生成绩\n");

printf(" 2=计算平均数\n");

printf(" 3=按成绩排序\n");

printf(" 4=输出学生成绩\n");

printf(" 5=退出程序\n");

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

c=getchar();

getchar();

while(c!='5'){

switch(c){

case('1'):{input();PARA1=&stu[0];};break;

case('2'):AVERAGE();break;

case('3'):RANKS();break;

case('4'):output();break;

default:break;

}

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf(" 1=输入学生成绩\n");

printf(" 2=计算平均数\n");

printf(" 3=按成绩排序\n");

printf(" 4=输出学生成绩\n");

printf(" 5=退出程序\n");

printf(" \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

getchar();

c=getchar();

getchar();

}

return 0;

}

汇编子程序一：

NAME N4

EXTRN \_PARA1:WORD

PUBLIC \_SOR

.386

INCLUDE MARCO.LIB

STACK0

STUDENT STRUCT

DB 11,?,11 DUP(0)

DB 0,0,0,?

DW 0

STUDENT ENDS

DATA4 SEGMENT USE16 PARA PUBLIC 'D5'

CM STUDENT <>

N EQU 5

DATA4 ENDS

\_TEXT SEGMENT USE16 PUBLIC 'CODE'

ASSUME CS:\_TEXT,ES:DATA4,SS:STACK

\_SOR PROC

MOV SI,DS:\_PARA1

MOV BX,OFFSET CM

MOV CH,N

SIZ2:

MOV DI,SI

MOV CL,CH

DEC CL

SIZ1:

ADD DI,19

MOV AL,16[SI]

CMP AL,16[DI]

JGE NEXC

PUSH SI

PUSH BX

PUSH DI

CALL EXCH

NEXC:

DEC CL

JNZ SIZ1

ADD SI,19

DEC CH

CMP CH,1

JA SIZ2

MOV SI,DS:\_PARA1

MOV CX,1

SIZ3:

MOV 17[SI],CX

ADD SI,19

INC CX

CMP CX,N

JA ENDSO

JMP SIZ3

ENDSO:

RET

\_SOR ENDP

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

EXCH PROC

PUSH AX

PUSH BX

PUSH CX

PUSH SI

PUSH DI

PUSH BP

MOV BP,SP

MOV DI,14[BP]

MOV BX,16[BP]

MOV SI,18[BP]

MOV CX,19

CH1:

MOV AL,BYTE PTR [SI]

MOV BYTE PTR ES:[BX],AL

INC SI

INC BX

DEC CX

JNZ CH1

MOV BX,16[BP]

MOV SI,18[BP]

MOV CX,19

CH2:

MOV AL,BYTE PTR [DI]

MOV BYTE PTR [SI],AL

INC SI

INC DI

DEC CX

JNZ CH2

MOV SI,18[BP]

MOV DI,14[BP]

MOV CX,19

CH3:

MOV AL,BYTE PTR ES:[BX]

MOV BYTE PTR [DI],AL

INC DI

INC BX

DEC CX

JNZ CH3

POP BP

POP DI

POP SI

POP CX

POP BX

POP AX

RET 6

EXCH ENDP

\_TEXT ENDS

END

子程序二：

NAME N6

EXTRN \_PARA1:word

PUBLIC \_ZBB

.386

INCLUDE MARCO.LIB

STACK0

DATA6 SEGMENT USE16 PARA PUBLIC 'D7'

DATA6 ENDS

\_TEXT SEGMENT USE16 PARA PUBLIC 'CODE'

ASSUME ES:DATA6,CS:\_TEXT,SS:STACK

\_ZBB PROC NEAR

MOV CX,5

MOV BX,\_PARA1

LOPA:

ADD BX,13

XOR AX,AX

XOR DX,DX

MOV DL,[BX]

INC BX

ADD AX,DX

ADD AX,AX

MOV DL,[BX]

ADD AX,DX

ADD AX,AX

INC BX

MOV DL,[BX]

ADD AX,DX

MOV DH,7

DIV DH

INC BX

MOV [BX],AL

ADD BX,3

DEC CX

JNZ LOPA

RET 2

\_ZBB ENDP

\_TEXT ENDS

END

4.3.2.3程序运行截图：

将学生成绩输入程序中，并选择功能进行使用，如图4.3所示

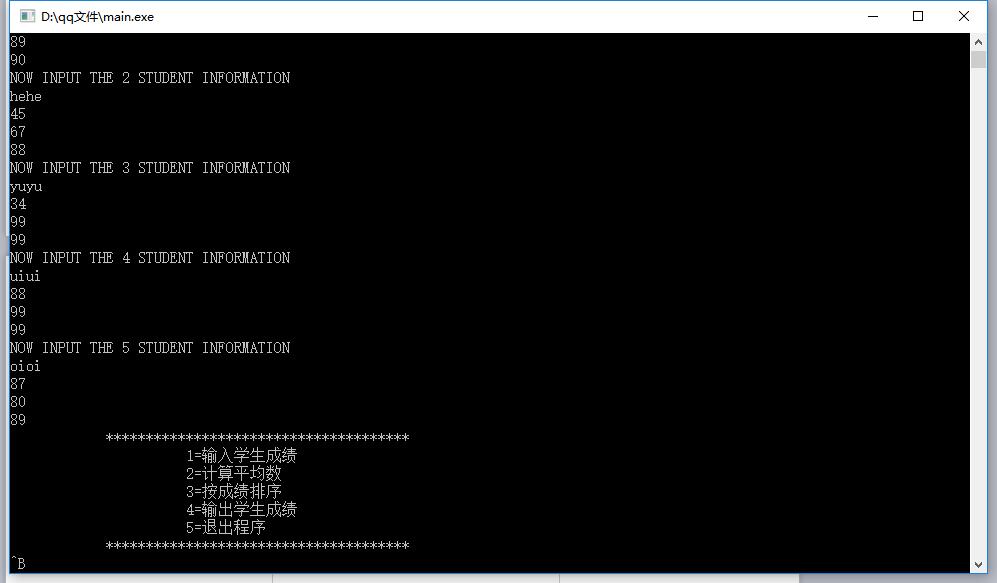


图4.3程序运行截图

# 4.4实验小结

本次试验感觉难度剧增。一方面是试验内容难，特别是c语言相互调用的内容，对于公用数据段和公共变量的传递有了更深理解。另一方面是由于没能跟搭档事前沟通好，大家都自己写自己的内容，导致最后链接的时候发现两人数据段定义并不一样，程序也基本上得重改，这大大拖慢了我们的进度。再有就是子程序保护的问题，所谓模块化程序设计，模块要能想积木一样，既能够自己独立，也要有相应的链接模块，一开始我们写的内容都依赖对方有这样那样的条件，这显然是不利于我们进行模块化编程的。吸取教训，下次一要跟合作的人商量好公用部分内容，二对于子程序要尽量能做到模块化。

# 4.5参考文献

[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