

**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 中断与反跟踪**

**实验时间： 2019-4-24，14：00-17：30 实验地点： 南一楼804室30号实验台**

**指导教师： 曹忠升**

**专业班级：计算机科学与技术ACM1701班**

**学 号： U201714780 姓 名： 刘晨彦**

**同组学生： 张瀚元 报告日期： 2019年5月4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： 纸质版再签名

日期：2019.5.4

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 3](#_Toc7809388)

[2 实验内容 3](#_Toc7809389)

[3 实验过程 5](#_Toc7809390)

[3.1 任务1 5](#_Toc7809391)

[3.1.1 设计思想及存储单元分配 5](#_Toc7809392)

[3.1.2 流程图 6](#_Toc7809393)

[3.1.3 源程序 11](#_Toc7809394)

[3.1.4 实验步骤 17](#_Toc7809395)

[3.1.5 实验记录与分析 18](#_Toc7809396)

[3.2 任务二 23](#_Toc7809397)

[3.2.1 设计思想及存储单元分配 23](#_Toc7809398)

[3.2.2 流程图 24](#_Toc7809399)

[3.2.3 源程序 24](#_Toc7809400)

[3.2.4 实验步骤 29](#_Toc7809401)

[3.2.5 实验记录与分析 30](#_Toc7809402)

[4 总结与体会 35](#_Toc7809403)

[参考文献 36](#_Toc7809404)

参考文献

# 实验目的与要求

1. 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；
2. 掌握宏指令、模块化程序的设计方法;
3. 掌握较大规模程序的合作开发与调试方法；
4. 掌握汇编语言程序与C语言程序混合编程的方法；
5. 了解C编译器的基本优化方法;
6. 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。

# 实验内容

**任务1 宏与子程序设计**

进一步修改与增强实验一任务4的网店商品信息管理程序的功能，主要调整功能三。

1.调整后的功能三的描述

（1）首先显示一个功能菜单（格式自行定义。若是未登录状态，只显示菜单“1”和“6”）:

1=查询商品信息，2=修改商品信息，3=计算推荐度，

4=计算推荐度排名，5=输出全部商品信息，6=程序退出。

输入1-6的数字进入对应的功能。

（2）查询商品信息

提示用户输入要查询的商品名称。若未能在网店中找到该商品，重新提示输入商品名称。若只输入回车，则回到功能三（1）。（思考一下模糊查询如何实现）

找到该商品之后，按照：“商品名称，折扣，销售价，进货总数，已售数量，推荐度”顺序显示该商品的信息。显示之后回到功能三（1）。

（3）修改商品信息

提示用户输入要修改信息的商品名称。[若把接下来的处理步骤写成子程序，则商品名称（或其偏移地址）就是子程序的入口参数，是否找到、是否是回车或者修改成功的信息是出口参数]。若未能在网店中找到该商品，重新提示输入商品名称。若只输入回车，则回到功能三（1）。

找到该商品之后，按照：折扣，进货价，销售价，进货总数的次序，逐一先显示原来的数值，然后输入新的数值（若输入有错，则重新对该项信息进行显示与修改。若直接回车，则不修改该项信息）。

如：折扣：9》8 //符号“》”仅作为分隔符，也可以选择其他分隔符号

进货价：25》24

销售价：46》5A6 //输入了非法数值，下一行重新显示和输入

销售价：46》56

进货总数：30》 //直接回车时，对这项信息不做修改

当对这些信息都处理完毕后，回到功能三（1）。

（4）计算推荐度

从头到尾依次将每个商品的推荐度计算出来。回到功能三（1）。

（5）计算推荐度排名

对SHOP中的每个商品按照推荐度的大小排名，排名信息可以存放到自行定义的一组结构变量中。回到功能三（1）。

（6）输出全部商品信息

将SHOP中的所有商品信息显示到屏幕上，包括排名。具体的显示格式自行定义（可以按照存放次序显示，也可以按照商品推荐度排名的次序显示，等等，显示方式可以作为子程序的入口参数）。回到功能三（1）。

**2.其他要求**

（1）**两人一组**，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）、（4）和（5）。各自汇编自己的模块，设计测试方法，测试通过；然后把自己的模块交给对方，各自把对方的程序整合到自己的程序中，连接生成一个程序，再进行整体调试。

实验报告中只需要描述自己负责的相关功能的设计思想、流程图、源程序。但在设计思想中要描述整体框架（包括整体的模块结构图、功能模块与子程序之间的对应关系等）和分工说明（包括模块的分配，两人协商一致的函数名、变量名等信息）。实验步骤和记录中要描述自己功能的实现与测试以及与同组模块整合后的联调与测试。

**注意，**在每个模块的开始，注明编写者的名字以及同组同学的名字。整合到一起时，要注意删掉自己测试时额外增加的代码，若有重复的模块（如：因两个人都会使用进制转换程序，导致各自模块中可能都有相同的进制转换子程序），也需要去掉重复的部分。

**建议分组方法：**按照学号（或前后左右相邻座位号）顺序依次两人一组，若班级人数为奇数，则最后三人一组（其中两人的分工是相同的，第三人只需要选择其中一个同学的模块与自己模块整合即可）。

（2）排名的基本要求是按照推荐度从高到低计算名次，也可以考虑按照指定字段（比如已售数量等）排名。相同推荐度排名相同，下一个相邻推荐度的名次应该是排名在前的所有商品种类“和”的下一个数值。

（3）将9号和10号DOS系统功能调用定义成宏指令并调用。功能（1）-（5）应尽量采用子程序方式实现。需要借鉴书上（或网上）的进制转换程序：十进制转二进制的子程序F10T2和二进制转十进制的子程序F2T10。

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、以及输入输出较多的某一个功能（如功能（1）、（2）、（5）中的某一个）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。

# 实验过程

## 任务1

### 设计思想及存储单元分配

模块任务分配：如图3.1所示。

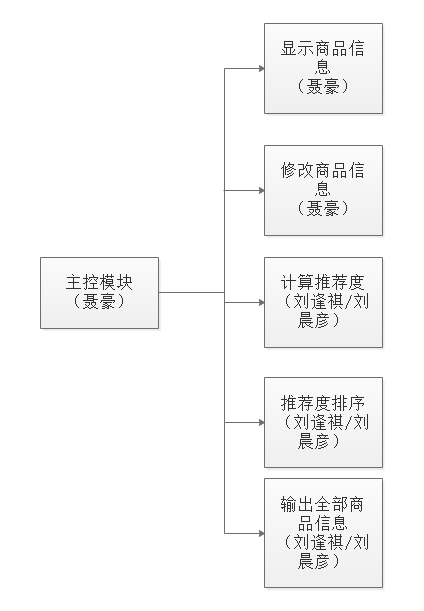


图3.1 程序模块任务分配

设计思想：

（1）程序总体逻辑如图3.2所示：

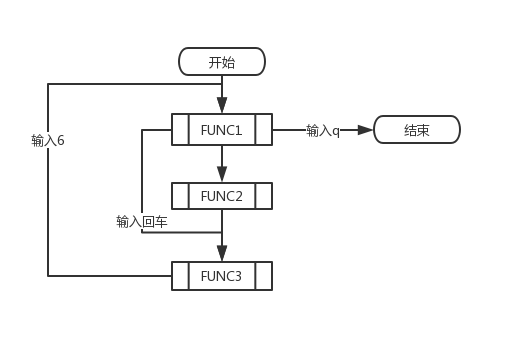


图3.2 代码总体逻辑流程图

（2）协商使用的函数名：INQURE: NEAR, ALTER: NEAR, COUNT\_RECOM: NEAR, RANK\_RECOM: NEAR, LDISPLAY: NEAR, RADIX: NEAR

（3）协商使用的三个变量名：RANK: WORD, GA1: BYTE

（4）协商使用的段的定义：CS: CODE, DS: DATA, SS: STACK

（4）计算推荐度时：移植前两次实验中的子程序代码使用。

（5）计算推荐度排名时，创建一大于商品种类的存储空间RANK，将商品首地址依次存入。排序时使用冒泡排序，通过商品首地址获得推荐度的首地址进行推荐度的比较。

（6）输出全部商品信息时，通过RANK依次获得商品首地址，通过循环依次输出商品信息

存储单元分配：

新增了一长度大于商品数的DW类型缓冲区，变量名为RANK，用于推荐度排序

寄存器分配：

BX：内层循环计数器  
CX：外层循环计数器

SI：存放商品首地址

DI：存放RANK串地址

### 流程图

功能三总程序流程图如图3.3所示：

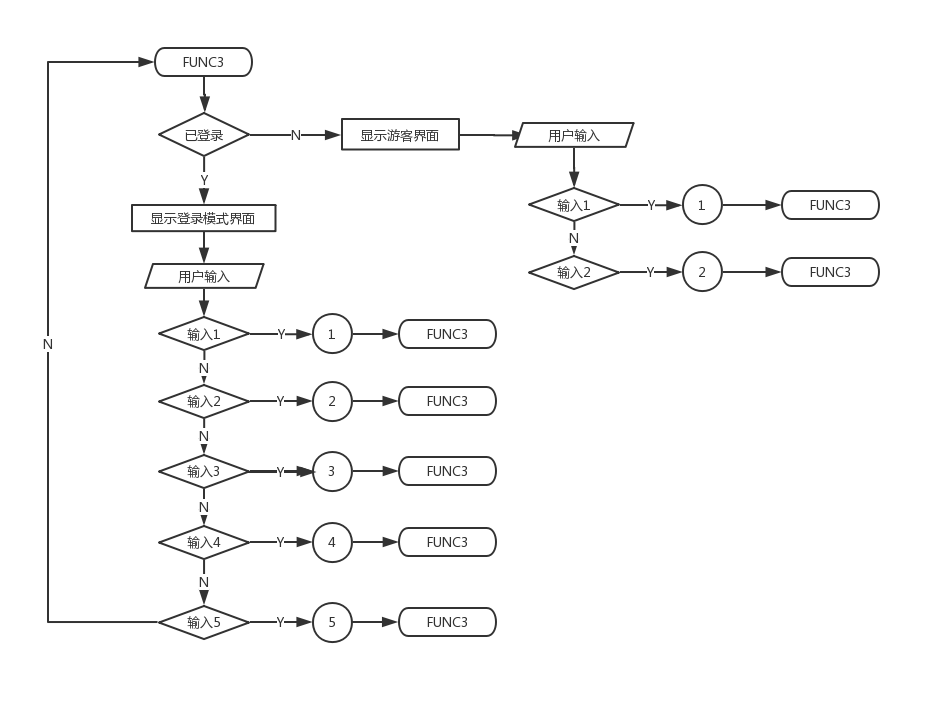


图3.3 功能三总程序流程图

功能三（4）：计算推荐度的流程图如图3.4所示：

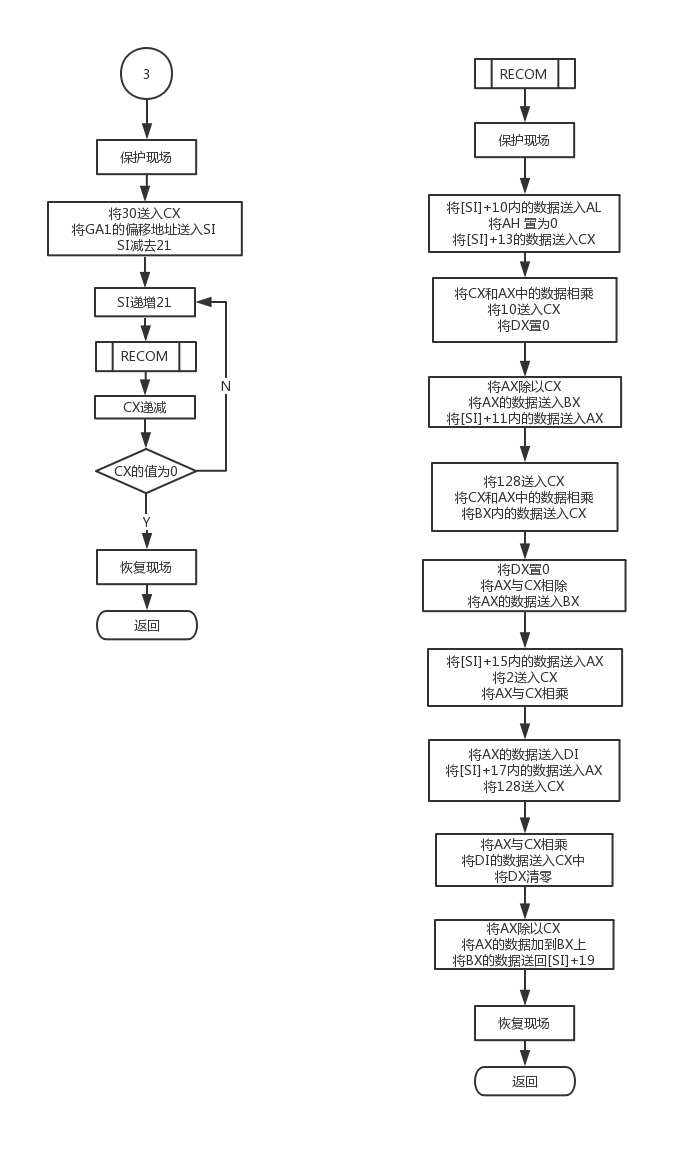


图3.4计算推荐度子程序的流程图

计算推荐度排名模块的流程图如图3.5所示：

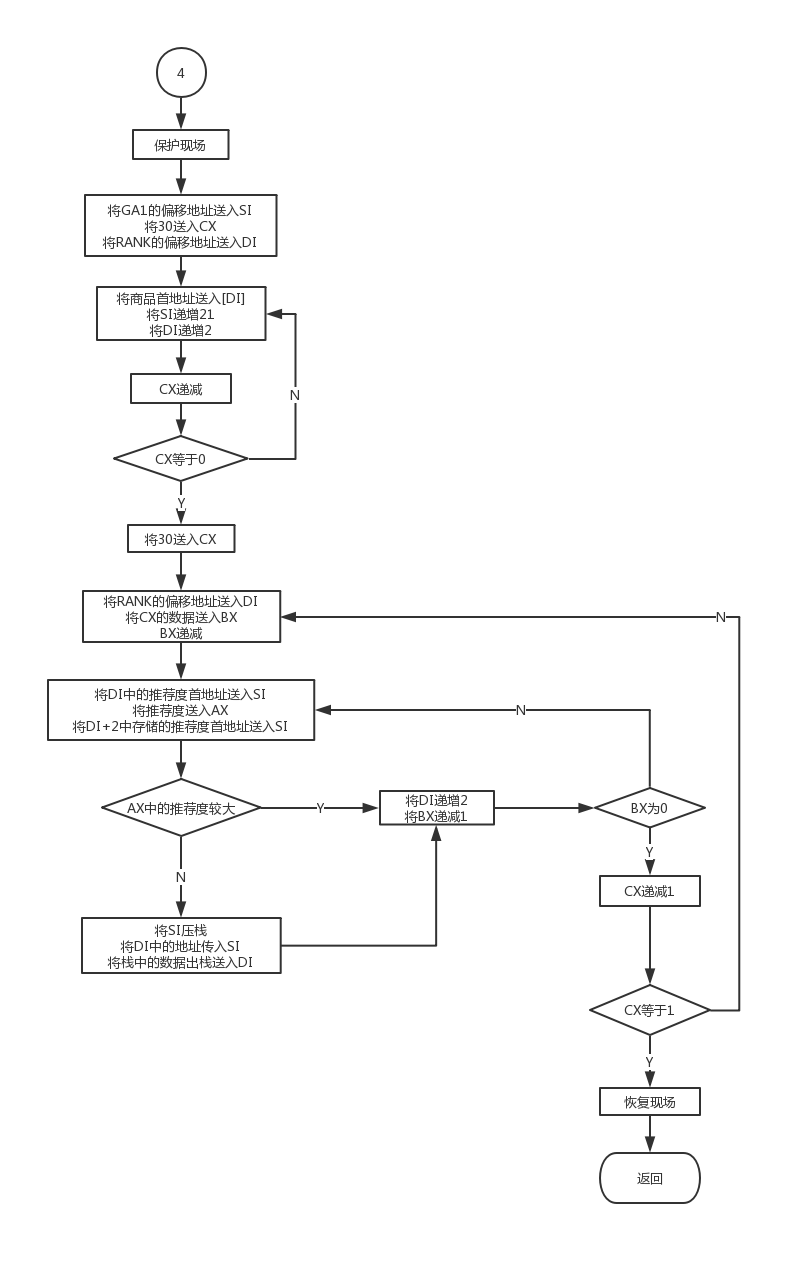


图3.5计算推荐度排名子程序的流程图

功能三（6）输出全部商品信息的流程图如图3.6所示

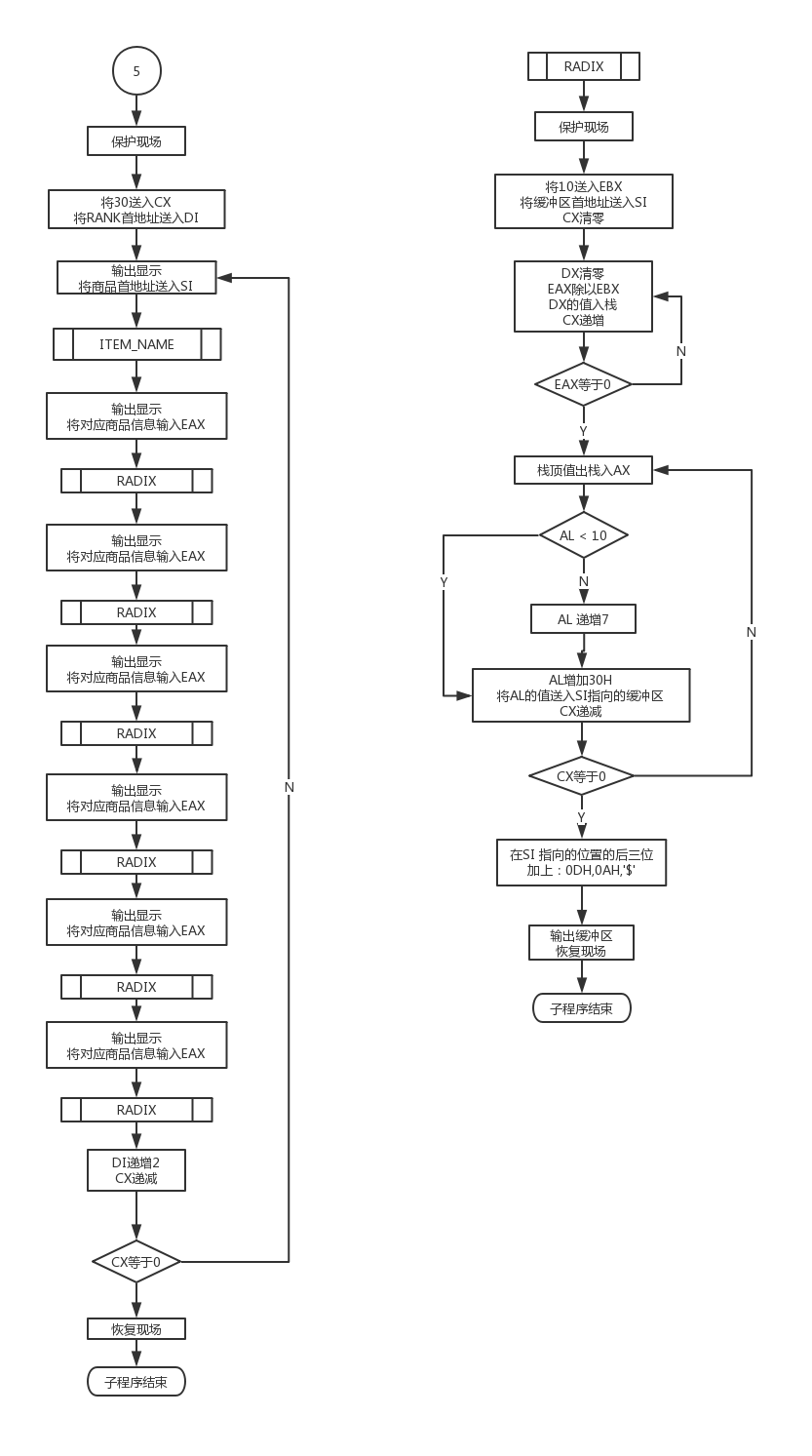


图3.6输出全部商品信息子程序的流程图

### 源程序

任务三（4）（5）（6）模块：

;编写： 刘晨彦

;同组同学：聂豪，刘逢祺

NAME SHOP\_EXTERNAL1

EXTRN GA1: BYTE, RANK: WORD

PUBLIC COUNT\_RECOM, RANK\_RECOM, LDISPLAY, RADIX

INCLUDE MACRO.LIB

.386

STACK SEGMENT USE16 STACK 'STACK'

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16 PUBLIC 'DATA'

BUF13 DB 'NAME OF ITEM:', '$'

BUF14 DB 'DISCOUNT:', '$'

BUF15 DB 'PURCHASE PRICE:', '$'

BUF16 DB 'SALE PRICE:', '$'

BUF17 DB 'PURCHASE NUMBER:', '$'

BUF18 DB 'SALE NUMBER:', '$'

BUF19 DB 'RECOMMENDATION:', '$'

BUF20 DB 'RECOMMENDATION LIST:', 0AH, 0DH, '$'

BUFA DB 15 DUP(0)

DATA ENDS

CODE SEGMENT USE16 PUBLIC 'CODE'

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

COUNT\_RECOM PROC ;A LOOP TO COUNT RECOMMENDATION

PUSH CX

PUSH SI

HERE: MOV CX, WORD PTR 30

MOV SI, OFFSET GA1

SUB SI, 21

COUNT\_ALL\_RECOM: ;LOOP FOR COUNTING RECOMMENDATION

ADD SI, 21

CALL RECOM

DEC CX

CMP CX, 0

JNE COUNT\_ALL\_RECOM

POP SI

POP CX

RET

COUNT\_RECOM ENDP

RECOM PROC ;ACTUALLY COUNTING RECOMMENDATION

PUSH AX

PUSH BX

PUSH CX

PUSH DX

PUSH DI

MOV AL, [SI + 10] ;DISCOUNT IN AX

MOV AH, 0

MOV CX, [SI + 13] ;SALE PRICE IN CX

MUL CX ;SALE \* DISCOUNT IN AX

MOV CX, 10

MOV DX, 0

DIV CX ;ACTUAL SALE PRICE IN AX

MOV BX, AX ;ACTUAL SALE PRICE IN BX

MOV AX, [SI + 11] ;PURCHASE PRICE

MOV CX, 128

MUL CX ;PURCHASE PRICE \* 128 IN AX

MOV CX, BX ;ACTUAL SALE PRICE IN CX

MOV DX, 0

DIV CX ;PURCHASE PRICE \* 128 / ACTUAL SALE PRICE IN AX

MOV BX, AX ;PURCHASE PRICE \* 128 / ACTUAL SALE PRICE IN BX

MOV AX, [SI + 15] ;NUM OF PURCHASE IN AX

MOV CX, 2 ;2 IN CX

MUL CX ;2 \* NUM OF PURCHASE IN AX

MOV DI, AX ;2 \* NUM OF PURCHASE IN DI

MOV AX, [SI + 17] ;NUM OF SALE

MOV CX, 128

MUL CX ;NUM OF SALE \* 128 IN AX

MOV CX, DI ;2 \* NUM OF PURCHASE IN CX

MOV DX, 0

DIV CX ;NUM OF SALE \* 128 / 2 \* NUM OF PURCHASE IN AX

ADD BX, AX

MOV WORD PTR [SI + 19], BX

POP DI

POP DX

POP CX

POP BX

POP AX

RET

RECOM ENDP

RANK\_RECOM PROC ;RANK ALL ITEM'S RECOMMENDATION

PUSH AX

PUSH BX

PUSH CX

PUSH DX

PUSH SI

PUSH DI

MOV SI, OFFSET GA1

MOV CX, 30

MOV DI, OFFSET RANK

;LOOP1:PUT ALL THE RECOMMENDATION IN TO RANK ARRAY

LOOP1: MOV [DI], SI ;MOVE THE ADDRESS OF RECOM INTO RANK

ADD SI, 21

ADD DI, 2

DEC CX

CMP CX, 0

JNE LOOP1

MOV CX, 30

LOOP2: MOV DI, OFFSET RANK

MOV BX, CX

DEC BX

LOOP3: MOV SI, [DI]

MOV AX, [SI + 19]

MOV SI, [DI + 2]

CMP AX, [SI + 19]

JA NEX

PUSH SI

MOV SI, [DI]

MOV [DI + 2], SI

POP SI

MOV [DI], SI

NEX: ADD DI, 2

DEC BX

JNZ LOOP3

DEC CX

CMP CX, 1

JNZ LOOP2

ENDF: POP DI

POP SI

POP DX

POP CX

POP BX

POP AX

RET

RANK\_RECOM ENDP

LDISPLAY PROC ;A LOOP TO DISPLAY ALL ITEM'S INFORMATION

PUSH AX

PUSH BX

PUSH CX

PUSH DX

PUSH DI

PUSH SI

MOV CX, 30

MOV DI, OFFSET RANK

LOOP6: WRITE BUF13

MOV SI, [DI]

CALL ITEM\_NAME

CRLF

WRITE BUF14

MOVZX EAX, BYTE PTR [SI] + 10

CALL RADIX

WRITE BUF15

MOVZX EAX, WORD PTR [SI] + 11

CALL RADIX

WRITE BUF16

MOVZX EAX, WORD PTR [SI] + 13

CALL RADIX

WRITE BUF17

MOVZX EAX, WORD PTR [SI] + 15

CALL RADIX

WRITE BUF18

MOVZX EAX, WORD PTR [SI] + 17

CALL RADIX

WRITE BUF19

MOVZX EAX, WORD PTR [SI] + 19

CALL RADIX

CRLF

ADD DI, 2

DEC CX

JNE LOOP6

POP SI

POP DI

POP DX

POP CX

POP BX

POP AX

RET

LDISPLAY ENDP

ITEM\_NAME PROC ;PRINT ITEM'S NAME: PUT ADDRESS INTO SI AND CALL THIS FUNCTION

PUSH BX

PUSH AX

PUSH DX

MOV BX,0

OPITEM: MOV DH, [SI]+[BX]

CMP DH, 0

JE OPITEM1

OUT1 DH

INC BX

CMP BX, 10

JE OPITEM1

JMP OPITEM

OPITEM1:

POP DX

POP AX

POP BX

RET

ITEM\_NAME ENDP

RADIX PROC

PUSH CX

PUSH EDX

PUSH SI

PUSH EBX

MOV EBX, 10

LEA SI, BUFA

XOR CX, CX

LOP1: XOR EDX, EDX

DIV EBX

PUSH DX

INC CX

OR EAX,EAX

JNZ LOP1

LOP2: POP AX

CMP AL, 10

JB L1

ADD AL,7

L1: ADD AL, 30H

MOV [SI], AL

INC SI

LOOP LOP2

MOV BYTE PTR [SI], 0DH

MOV BYTE PTR [SI + 1], 0AH

MOV BYTE PTR [SI + 2], '$'

WRITE BUFA

POP EBX

POP SI

POP EDX

POP CX

RET

RADIX ENDP

CODE ENDS

END

### 实验步骤

1.准备上机实验环境。

2.使用VISUAL STUDIO修改实验一中的程序，要求满足本次实验要求，保存至SHOPE1.ASM。使用MASM6.0汇编源文件，观察提示信息，若出错则返回重新编辑SHOPE1.ASM，保存后重新汇编，直至不再报错为止。汇编同组同学完成的SHOP.ASM, SHOPE2.ASM, SHOPE3.ASM文件。

3.使用连接程序LINK.EXE将生成的SHOP.OBJ文件、SHOPE1.OBJ、SHOPE2.OBJ、SHOPE3.OBJ连接成执行文件。

4.执行程序。在未连接其他模块的情况下按照程序设计要求进行交互，检查是否达到程序设计要求。

5.综合测试程序功能运行情况：

（1）用游客模式查找商品CAKE

（2）登录模式下修改商品BOOK的信息（输入2 3 4 1），使用任务三（4）（5）（6），查看输出是否符合要求。

6. 使用TD.EXE或直接运行程序，观察执行情况:

（1）通过TD观察宏指令在执行程序中的替换和扩展

（2）观察FAR、NEAR类型子程序的RET指令的机器码有何不同。观察FAR类型子程序被调用时堆栈的变化情况。

（3）观察EXTRN说明语句放在.386之前或者之后有什么区别？

### 实验记录与分析

1. 实验环境条件：i7-7700HQ 2.80GHz，8G内存；WINDOWS 10下DOSBox0.72；TD.EXE 5.0。

2. 汇编源程序时未发生异常。

3. 连接过程中未发生异常

4. 在未连接其他模块的情况下检查设计模块是否达到程序设计要求：

任务三（4）：将每个商品推荐度计算出来后回到功能三，测试结果如图3.7(a),(b)，(c)所示：

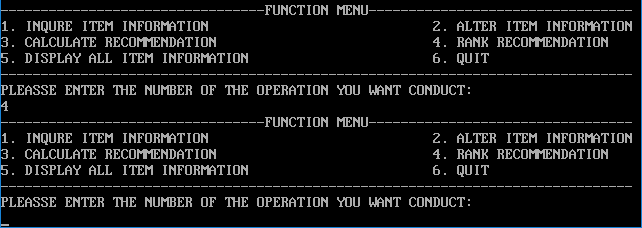


图3.7(a) 推荐度计算并回到功能三测试截图

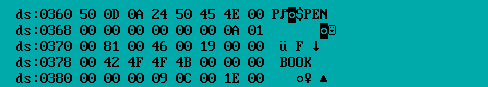


图3.7(b) 推荐度计算前商品存储空间截图

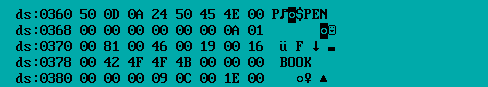


图3.7(c) 推荐度计算后商品存储空间截图

根据测试截图3.7(b),(c)所示，可见推荐度被计算后存入了对应的存储空间。故该功能正常。

任务三（5）：将推荐度排名并存放到自定义的结构变量中，功能结束后回到任务三。测试截图如图3.8(a),(b),(c)所示。

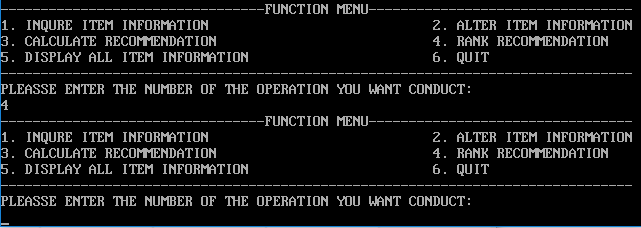


图3.8(a) 推荐度排名后回到功能三测试截图

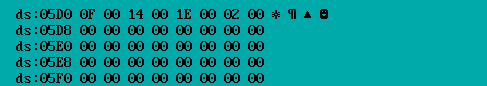


图3.8(b) 推荐度计算前RANK存储空间截图

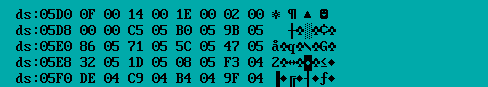


图3.8(c) 推荐度计算后RANK存储空间截图

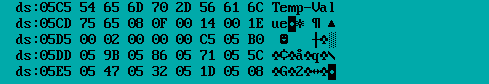


图3.8(d) DS：05C5对应的存储内容截图

根据测试截图3.8(b),(c),(d)所示，RANK中存储了对应的根据推荐度排序的商品首地址，在RANK中排名第一的地址DS：05C5中存储的是商品Temp-Value的信息。根据测试，该项功能正常。

任务三（5）：将所有商品信息按照推荐度顺序输出，并回到功能三。程序测试截图如图3.9所示

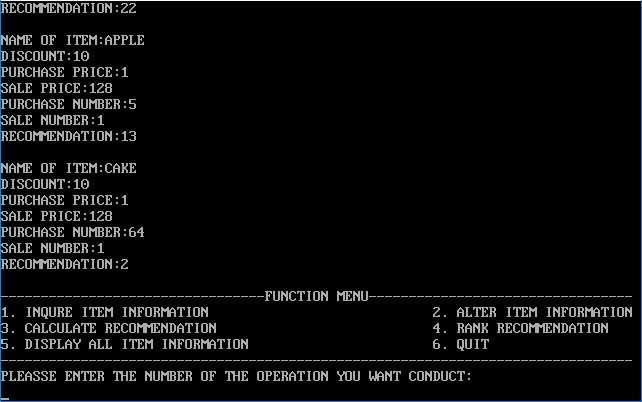


图3.9 所有商品信息按照推荐度顺序输出并回到功能三测试截图

5.综合测试

（1）用游客模式查找商品CAKE：测试截图如图3.10所示：

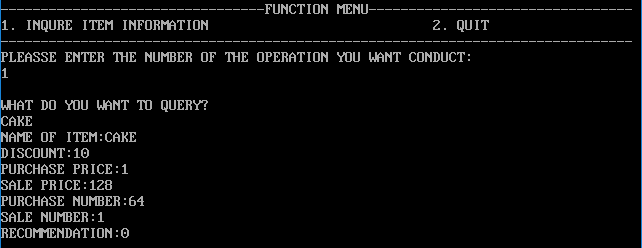


图3.10 用游客模式查找商品CAKE测试截图

测试结果显示功能正常。

（2）登录模式下修改商品BOOK的信息，使用任务三（4）（5）（6），查看输出是否符合要求。(输出隐去商品的部分信息)，如图3.11(a)、(b)所示。截图显示功能正常。

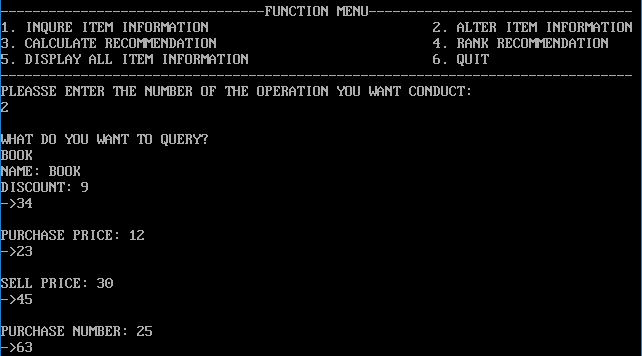


图3.11(a) 修改商品信息截图

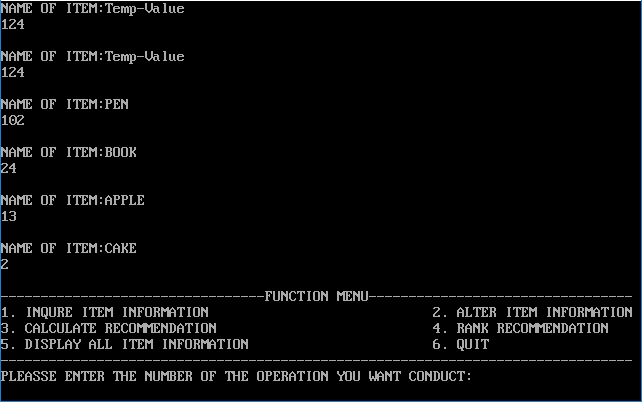


图3.11(b) 修改后推荐度排序截图

6.使用TD观察程序：

（1）通过TD观察宏指令在执行程序中的替换和扩展：由图3.12(a)、(b)、(c)所示，可知宏指令在程序中直接替换为了对应的指令

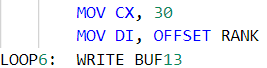


图3.12(a) 使用宏指令的代码

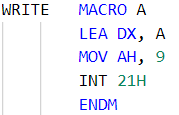


图3.12(b) 宏定义

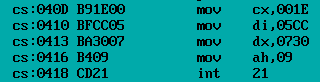


图3.12(c) 实际的宏指令替换

（2）观察FAR、NEAR类型子程序的RET指令的机器码有何不同。如图3.13所示。FAR类型子程序被调用时堆栈的变化情况如图3.14(a),(b)所示：

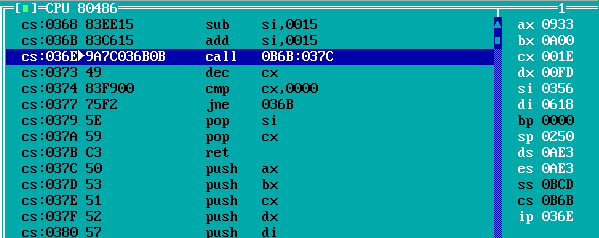


图3.13 调用FAR类型子程序反汇编截图

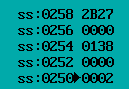


图3.14(a)调用前堆栈截图

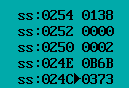


图3.14(b) 调用后堆栈截图

由截图可知，FAR类型相比NEAR类型多了OB6B，而0B6B是代码段CS的内容，进入子程序后，先将CS的数据压栈，然后将子程序结束后返回地址压栈。

（3）观察EXTRN说明语句放在.386之后的运行情况如图3.15所示：

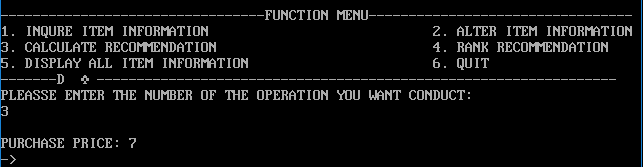


图3.15 将EXTRN语句放在.386之后的运行截图

此时生成的是32位的段程序，由截图可知程序无法正常运行。

## 任务二

### 设计思想及存储单元分配

设计思想：程序总体流程如图3.16所示：

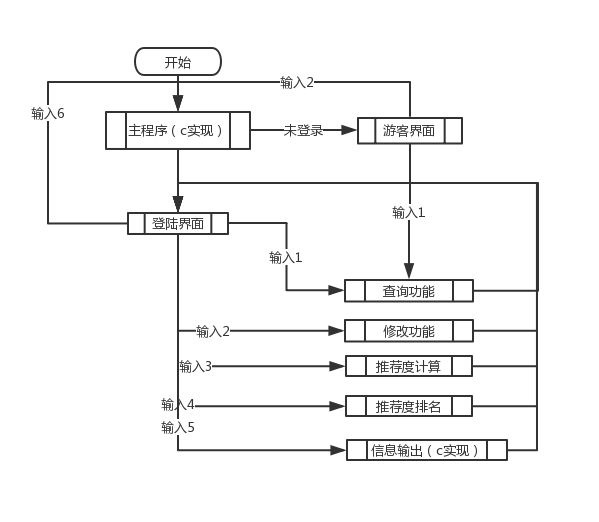


图3.16 程序设计流程图

根据设计要求，将主程序和一子程序使用c语言复现，其余子程序仍使用任务一中的子程序。

存储单元分配：

新增了一长度大于商品数的DW类型缓冲区，变量名为RANK，用于推荐度排序

寄存器分配：

BX：内层循环计数器  
CX：外层循环计数器

SI：存放商品首地址

DI：存放RANK串地址

### 流程图

本任务主程序流程图如图3.3所示，本任务子程序流程图如图3.4，3.5，3.6所示。

### 源程序

使用C复现的主程序和查询商品信息子程序如下：

#include <stdio.h>

#include <string.h>

#pragma inline

#define N 30

extern void ALTER(void);

extern void COUNT\_RECOM(void);

extern void RANK\_RECOM(void);

extern void LDISPLAY(void);

extern char GA1;

char aname[] = {'l','i','u','c','h','e','n','y','a','n','\0'};

char password[] = {'t','e','s','t','\0'};

char bname[10];

char bpwd[10];

int login(void){

int i;

char in;

begin:

printf(" ------------------------------------------------------------------------------\n");

printf("| SHOP |");

printf(" ------------------------------------------------------------------------------\n");

printf("Please log in:\nname:\n");

i = 0;

while(1){//get name

in = getchar();

bname[i] = in;

i ++;

if(in == '\n'){

break;

}

}

fflush(stdin);

bname[i - 1] = '\0';

if(bname[0] == 'q' && bname[1] == '\0'){

return 0;

}

if(bname[0] == '\0'){

return 2; //fail to login, into consumer mode

}else{

if(strcmp(aname, bname) == 0){//if the user name is correct

printf("PASSWORD:\n");

i = 0;

while(1){//get password

in = getchar();

bpwd[i] = in;

i ++;

if(in == '\n'){

break;

}

}

fflush(stdin);

bpwd[i - 1] = '\0';

if (strcmp(password, bpwd) == 0){

return 1;

}else{

printf("WRONG PASSWORD\n");

goto begin;

}

}else{

printf("WRONG NAME\n");

goto begin;

}

}

}

int display(int auth){

int choice;

if(auth == 1){

printf(" --------------------------------FUNCTION MENU-------------------------------\n");

printf("|1. INQURE ITEM INFORMATION 2. ALTER ITEM INFORMATION|\n");

printf("|3. CALCULATE RECOMMENDATION 4. RANK RECOMMENDATION |\n");

printf("|5. DISPLAY ALL ITEM INFORMATION 6. QUIT |\n");

printf(" ----------------------------------------------------------------------------\n");

printf("PLEASSE ENTER THE NUMBER OF THE OPERATION YOU WANT CONDUCT:\n");

scanf("%d", &choice);

fflush(stdin);

}else if(auth == 2){

printf(" --------------------------------FUNCTION MENU--------------------------------\n");

printf("|1. INQURE ITEM INFORMATION 2. QUIT |\n");

printf(" -----------------------------------------------------------------------------\n");

printf("PLEASSE ENTER THE NUMBER OF THE OPERATION YOU WANT CONDUCT:\n");

scanf("%d", &choice);

fflush(stdin);

}else if(auth == 0){

return 0;

}

return choice;

}

void func1(void) {

char stuff[11];

short \*n;

char \*p=NULL;

char \*find=NULL;

int i,j;

printf("WHAT DO YOU WANT TO QUERY?\n");

while (1){

asm mov p,offset GA1;

gets(stuff);

for (j=0;j<30;j++){

for (i=0;i<10;i++){

if (stuff[i]!=\*(p+i)){

break;

}

}

if (j==10){

find=p;

break;

}

else if(\*(p+j)==0){

find=p;

break;

}

}

if (find==NULL){

printf("THINGS NOT FOUND! INPUT AGAIN.\n");

continue;

}

else {

printf("NAME:");

printf("%s\n",stuff);

printf("DISCOUNT:%d\n",\*(find+10));

n=(short \*)(find+11) ;

printf("PURCHASE\_PRICE:%d\n",\*n);

printf("SALE\_PRICE:%d\n",\*(n+1));

printf("PURCHASE\_NUMBER:%d\n",\*(n+2));

printf("SALE\_NUMBER:%d\n",\*(n+3));

printf("RECOMMENDATION:%d\n",\*(n+4));

break;

}

}

}

int main(void){

char choice;

int auth;

FUNC1:

auth = login();

CYCLE:

choice = display(auth);

if(auth == 2){

switch(choice){

case 1:

func1();

goto CYCLE;

case 2:

goto FUNC1;

default:

goto CYCLE;

}

}else if(auth == 1){

switch(choice){

case 1:

func1();

goto CYCLE;

case 2:

ALTER();

goto FUNC1;

case 3:

COUNT\_RECOM();

goto CYCLE;

case 4:

RANK\_RECOM();

goto CYCLE;

case 5:

LDISPLAY();

goto CYCLE;

case 6:

goto FUNC1;

default:

goto CYCLE;

}

}

return 0;

}

### 实验步骤

1.准备上机实验环境。

2.使用VISUAL STUDIO用c语言复现任务一中的一子程序和主程序，要求满足本次实验要求，保存至SHOP.C。将SHOP.C和包含所需要的其余子程序的SHOPE1C.ASM放入文件夹BIN中。

3.打开Borland C++ 3.1，创建新PROJECT，将文件SHOP.C和SHOPE1C.ASM加入PROJECT，点击BUILD ALL编译链接，若存在错误则返回重新编辑，否则直接点击RUN运行程序。

4.执行程序。检查是否达到程序设计要求。

5. 尝试在C语言源程序中不合理地嵌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。

6. 观察C的主程序调用汇编子程序时寄存器、堆栈变化。

7. 观察C的主程序调用C的子程序时寄存器、堆栈变化。

### 实验记录与分析

1. 实验环境条件：i7-7700HQ 2.80GHz，8G内存；WINDOWS 10下Borland C++ 3.1， DOSBox0.72和TD.EXE 5.0。

2. 编写时未发生异常。

3. 编译链接时未发生异常。

4. 检查是否达到设计要求：

（1）未登录直接进入访客模式，测试截图如图3.17所示，测试结果显示功能正常。



图3.17 直接进入访客模式

（2）访问模式下查询商品信息，测试截图如图3.18所示，测试结果显示功能正常。

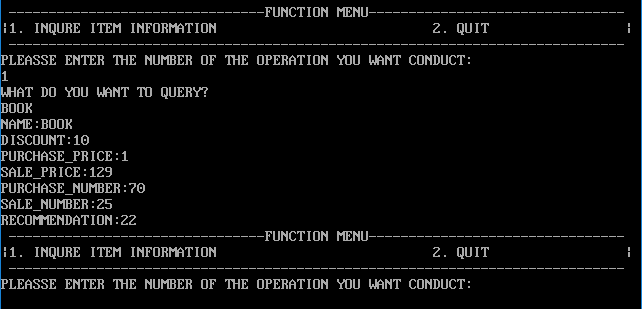


图3.18 访客模式下查询商品信息

（3）店主能正确登录网点测试，测试截图如图3.19(a)、(b) 、(c)所示，测试结果显示功能正常。

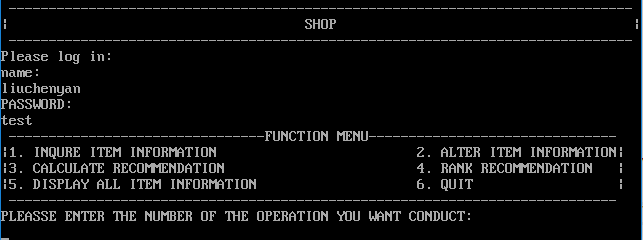


图3.19(a) 正确登陆测试截图



图3.19(b) 错误输入用户名测试截图



图3.19(c) 错误输入密码测试截图

（4）登录模式下查询商品信息测试，测试如图3.20所示，测试结果显示功能正常。

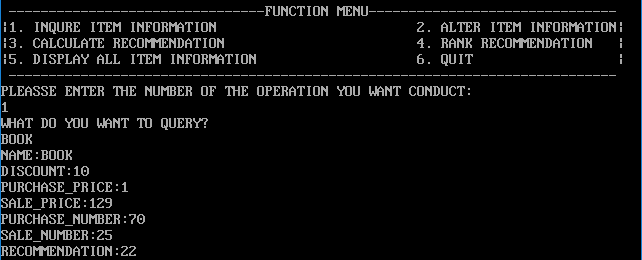


图3.20 登陆模式下查询商品信息测试截图

（5）登陆模式下修改商品信息测试，测试如图3.21所示，测试结果显示功能正常。

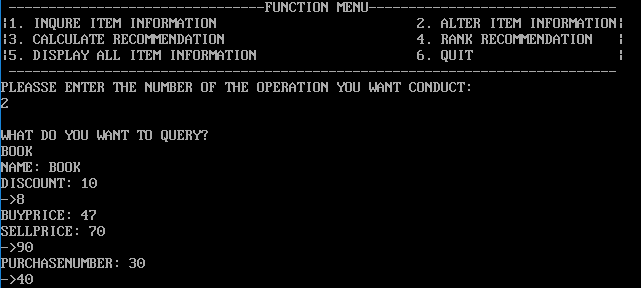


图3.21 登陆模式下修改商品信息测试截图

（6）登陆模式下计算商品推荐度测试，测试结果如图3.22所示，测试结果显示功能正常。

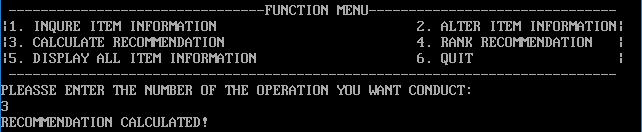


图3.22 登陆模式下计算商品推荐度测试

（7）登陆模式下商品推荐度排序测试，测试结果如图3.23所示，测试结果显示功能正常。

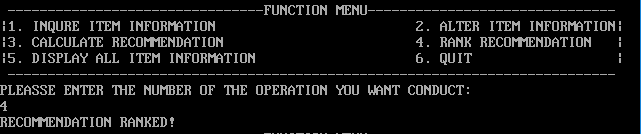


图3.23 登陆模式下商品推荐度排序测试截图

（8）登陆模式下输出全部商品推荐度排序测试,测试截图如图3.24所示，测试结果显示功能正常。

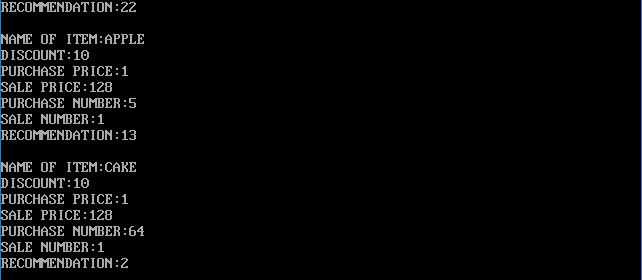


图3.24 登陆模式下输出全部商品推荐度排序测试截图

5.尝试在C语言源程序中不合理地嵌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。C程序更改如图3.25所示，更改后程序运行状态如图3.26所示，由此可知不合理地嵌入汇编语言的指令语句破坏了C语言程序的正确性

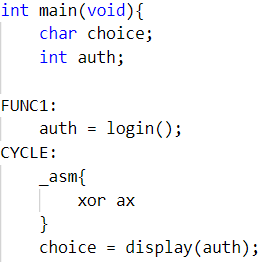


图3.25 C程序中嵌入不合理的汇编语句指令

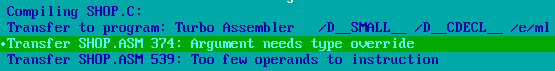


图3.26 嵌入不合理汇编指令后运行结果截图

6. 观察C的主程序调用汇编子程序时寄存器、堆栈变化。

如图3.27、3.28所示，主程序进入子程序后，首先将主程序的下一条执行至指令的地址送入堆栈，SP所存栈顶地址值减小了2。同时段未发生变化。

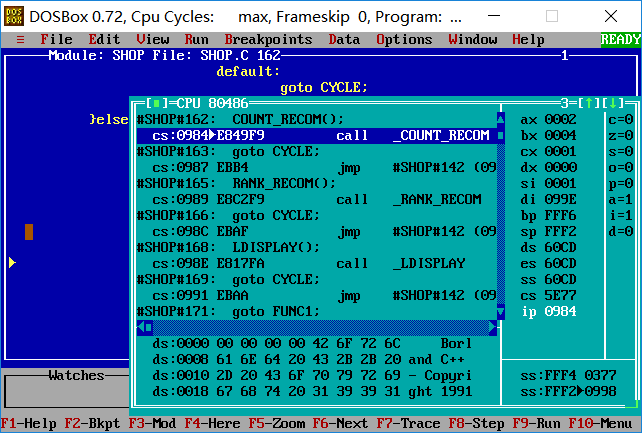


图3.27 主程序进入子程序前截图

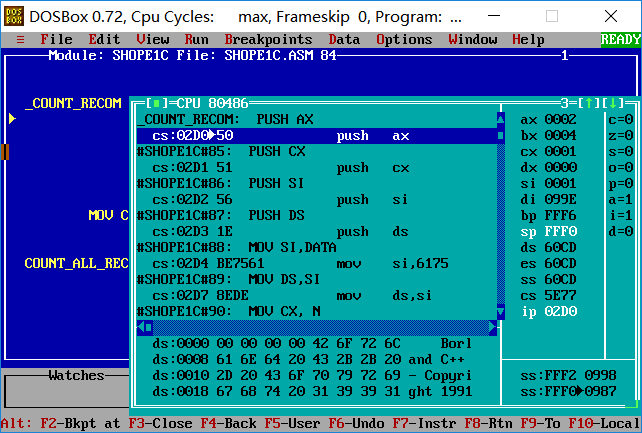


图3.28 进入子程序截图

7. 观察C的主程序调用C的子程序时寄存器、堆栈变化。

如图3.29、3.30所示，主程序进入子程序后，首先将主程序的下一条执行至指令的地址送入堆栈，SP所存栈顶地址值减小了2。同时段首址不改变。

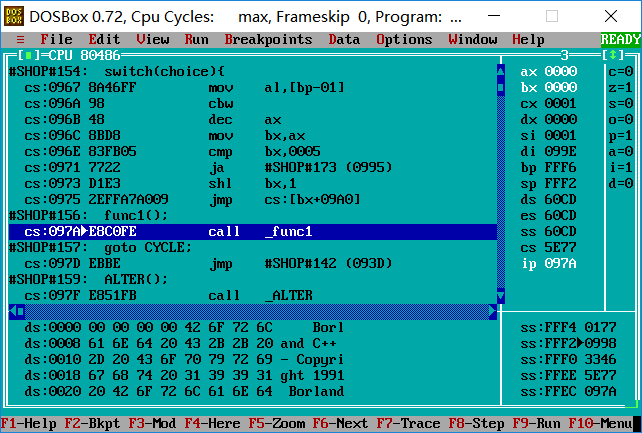


图3.29 主程序进入子程序前截图

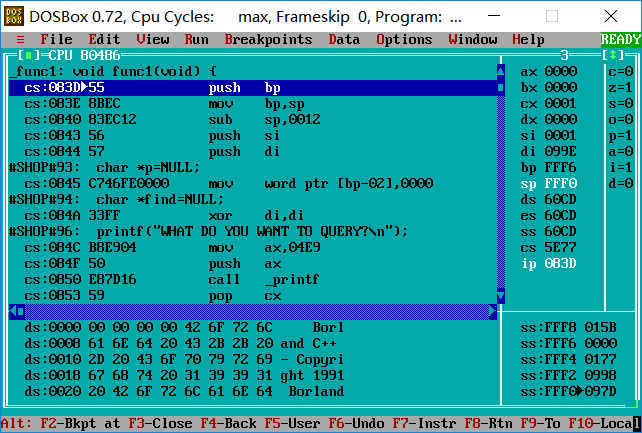


图3.30 进入子程序截图

# 总结与体会

通过这次实验我学习并掌握了汇编语言的模块化编程和C语言与汇编的混合编程。在任务一中，我学习并了解了多个不同模块的汇编程序之间如何进行通讯如何交流与共享数据，虽然在这个过程中遇到了许多问题，不过在老师的帮助下我解决了这些问题，并且加深了我对汇编语言的理解与认识。

在任务二中由于对新的编程环境BC不太熟悉，中间也遇到了不少问题，不过主要还是因为自己不过仔细，没有仔细看实验指导造成的，这也给我以警醒。

总而言之，这两次实验较之前更具难度，不过也让我学会了更多。

# 参考文献

[1] 许向阳.80X86汇编语言程序设计上机指南.武汉:华中科技大学出版社,2007：1-61