華中科技大學

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　模块化程序设计

实验时间： 2018-4-16，14：00-17：30，2018-4-23，14：00-17：30

实验地点： 南一楼 804 室

指导教师： 朱虹

专业班级 计算机 201601 班

学　　号： U201614532　　**姓　　名：** 吕鹏泽

同组学生： 吴阳民　　**报告日期：** 2018 年　4 月 24 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）了解程序计时的方法以及运行环境对程序执行情况的影响。

（2）熟悉汇编语言指令的特点，掌握代码优化的基本方法。

2. 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四的**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=计算平均利润率，

4=计算利润率排名，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。若未能在第一个网店中找到该商品，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：“SHOP1，商品名称，销售价，进货总数，已售数量”顺序显示该商品的信息，同时还要将“SHOP2”中该商品的信息也显示出来。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（先指定网店名称）。[若把接下来的处理步骤写成子程序，则网店名称和商品名称（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]。若未能在指定网店中找到该商品，重新提示输入网店名称和商品名称。若只输入回车，则回到功能三（1）。

找到该商品之后，按照：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：进货价：25》24　　//符号“》”仅作为分隔符，也可以选择其他分隔符号

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

当对三项信息都处理完毕后，回到功能三（1）。

（4）计算平均利润率

首先计算 SHOP1 中第一个商品的利润率 PR1，然后在 SHOP2 网店中寻找到该商品，也计算其利润率 PR2。最后求出该商品的平均利润率 APR=(PR1+PR2)/2，并保存到 SHOP1 的利润率字段中。重复上述步骤，依次将每个商品的平均利润率计算出来。回到功能三（1）。

（5）计算利润率排名

对 SHOP2 中的每个商品按照平均利润率的大小排名，排名信息存放到 SHOP2 中商品的利润率字段中。回到功能三（1）。

（6）输出全部商品信息

将 SHOP1 和 SHOP2 中的所有商品信息显示到屏幕上，包括平均利润率和排名（替代了商品原有的利润率字段）。具体的显示格式自行定义（可以分网店显示，也可以按照商品排名显示，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。

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加的代码，若有重复的模块（如：两个人都会使用进制转换子程序，各自模块中可能都有相同的进制转换程序），也需要去掉重复的部分。

建议分组方法：按照学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平均利润率从高到低计算名次，也可以考虑按照指定字段（比如已售数量等）排名。相同平均利润率时排名相同，下一个相邻平均利润率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。

3. 实验过程

3.1 任务 1

本次任务我负责菜单显示、程序退出在内的主程序，以及菜单中的功能（1）和（2），同组的吴阳民同学负责菜单中的功能（3）、（4）和（5）。

3.1.1 设计思想及存储单元分配

设计思想:

功能 1、2、4 设计思想同实验 1。功能 3 根据用户登陆状态，利用 9 号调用输出相应的菜单选项，写一个子程序获取用户输入并纠错知道用户输入对了选项为止，由用户选择功能并执行。模块 1 循环查询商品，找到输出对应商品信息，未找到则让用户重新输入商品名。模块 2 先选择商店，然后查询出正确的商品，利用将字符串转化为数字的子程序获取用户输入，然后根据用户的输入修改对应商品信息。

存储单元分配：同实验 1

寄存器分配：

CX：控制循环；

BX：基址寄存器；

BP：存放缓冲区基地址；

AL：临时读取数据域的值；

AX，DX，SI：临时寄存器；

3.1.2 流程图

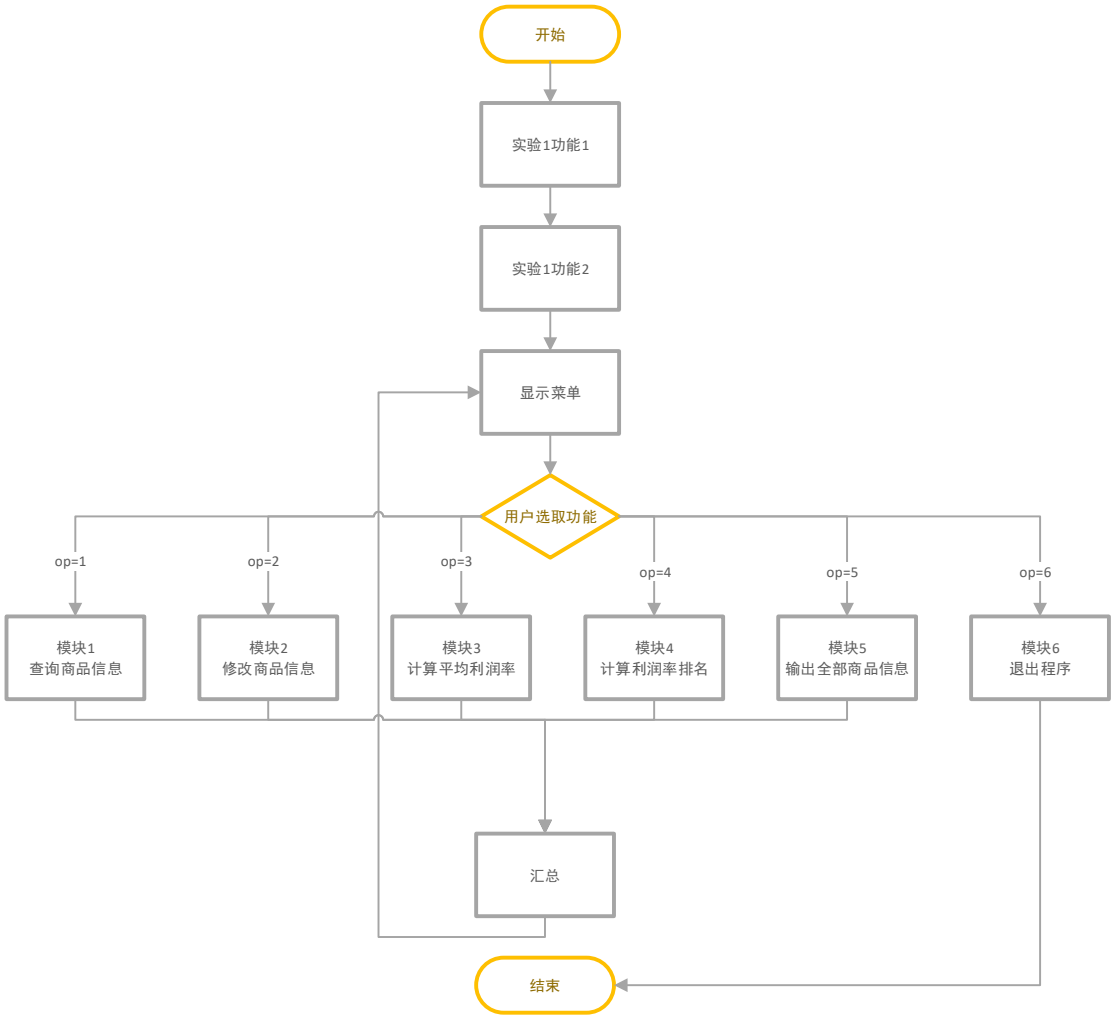


图 3.1.1 程序框架图

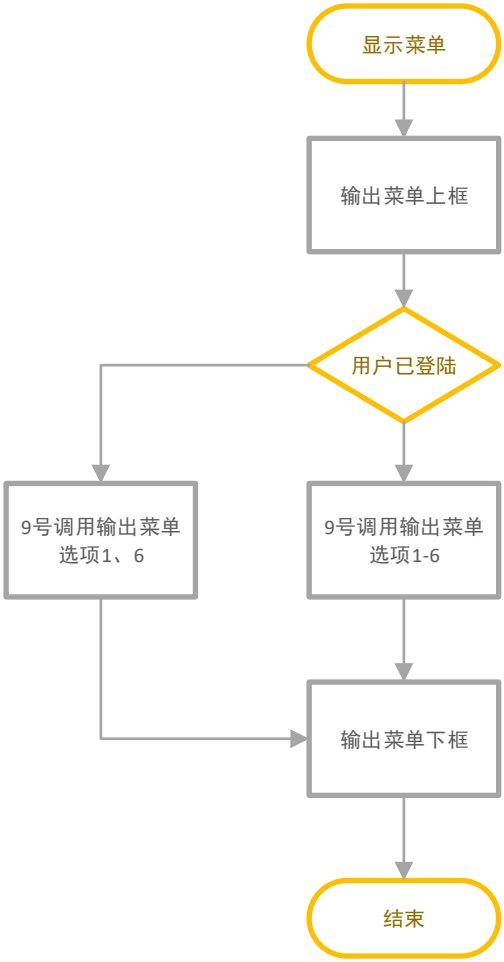


图 3.1.2 菜单显示子程序流程图

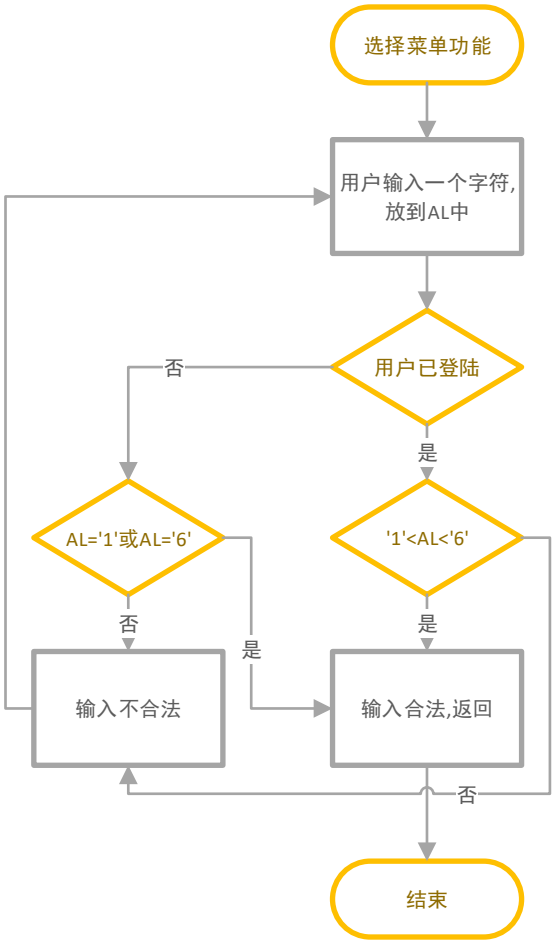


图 3.1.3 获取菜单功能子程序流程图

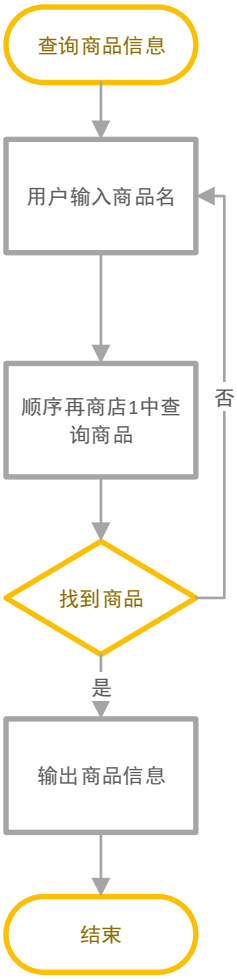


图 3.1.4 模块 1 商品信息查询功能流程图

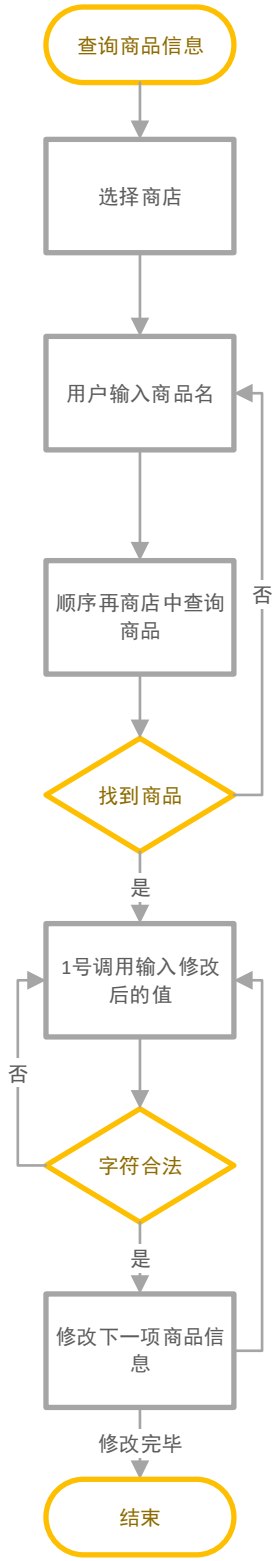


图 3.1.5 模块 2 商品信息修改功能流程图

3.1.3 源程序

3.1.3.1 吕鹏泽负责模块源代码(T.ASM)

```
NAME WAN1
EXTRN AVERAGR_PROFIT:NEAR,PROFIT_RANKING:NEAR,PRINT_IFO:NEAR
PUBLIC PRINTAX,PrintASCII,S1,S2,GA1,GB1
.386
INCLUDE MACRO.LIB
STACK SEGMENT USE16 PARA STACK 'STACK'
DB 200 DUP(0)
STACK ENDS
DATA SEGMENT USE16 PARA PUBLIC 'D1'
BNAME DB 'LVPENGZE',0,0                    ;管理员姓名
BPASS DB 'test',0,0                        ;密码
AUTH DB ?                                  ;标记登陆状态
N EQU 10
S1 DB 'SHOP1','$'                          ;网店 1
GA1 DB 'PEN',7 DUP(0)
 DW 10,56,70,50,?                  ;进货价、销售价、进货总数、已售数量、利润率
GA2 DB 'BOOK',6 DUP(0)
 DW 12,30,25,10,?
GA3 DB 'BAG',7 DUP(0)
 DW 15,30,30,0,?
GA4 DB 'CUP',7 DUP(0)
 DW 10,23,40,12,?
GA5 DB 'HAT',7 DUP(0)
 DW 23,40,50,13,?
GA6 DB 'JUICE',5 DUP(0)
 DW 1,3,50,40,?
GA7 DB 'RULER',5 DUP(0)
 DW 3,8,30,10,?
GA8 DB 'ERASER',4 DUP(0)
 DW 1,2,30,12,?
GAN DB N-8 DUP('TempValue',0,15,0,20,0,30,0,30,0,?,?)

S2 DB 'SHOP2','$'                          ;网店 2
GB1 DB 'PEN',7 DUP(0)
 DW 10,56,70,50,?                  ;进货价、销售价、进货总数、已售数量、利润率
GB2 DB 'BOOK',6 DUP(0)
 DW 12,30,25,10,?
GB3 DB 'BAG',7 DUP(0)
 DW 15,30,30,0,?
GB4 DB 'CUP',7 DUP(0)
 DW 10,23,40,12,?
GB5 DB 'HAT',7 DUP(0)
 DW 23,40,50,13,?
GB6 DB 'JUICE',5 DUP(0)
 DW 1,3,50,40,?
GB7 DB 'RULER',5 DUP(0)
 DW 3,8,30,10,?
GB8 DB 'ERASER',4 DUP(0)
 DW 1,2,30,12,?
GBN DB N-8 DUP('TempValue',0,15,0,20,0,30,0,30,0,?,?)

PR1 DW 0                                   ;商店 1 利润率
PR2 DW 0                                   ;商店 2 利润率
APR DW 0                                   ;平均利润率
MSG1 DB 0AH,0DH,'Input your account:',0DH,0AH,'$'
MSG2 DB 'Input your password:',0DH,0AH,'$'
```

```
MSG3 DB 'WRONG ACCOUNT',0DH,0AH,'$'
MSG4 DB 'Enter the name of the item:',0DH,0AH,'$'
MSG5 DB 0AH,0DH,'Input Your Option',0DH,0AH,'$'

MENU1 DB '1.Query commodity information','$'
MENU2 DB '2.Modify commodity information','$'
MENU3 DB '3.Calculate average profit margin','$'
MENU4 DB '4.Calculate profit rate ranking','$'
MENU5 DB '5.Export all product information','$'
MENU6 DB '6.Program exit','$'

SORT_BUF DW N DUP(0)
SORT_RAN DW N DUP(0)
Product_name DB 'Product name', '$'
Purchase_price DB 'Purchase price', '$'
Selling_price DB 'Selling price', '$'
Total_purchases DB 'Total purchases', '$'
Sold_quantity DB 'Sold Quantity', '$'
Profit_rate DB 'Profit rate', '$'
Ranking DB 'Ranking', '$'

SelectShop DB '1.SHOP1 2.SHOP2','$'
in_name   DB 11                    ;姓名缓冲区
          DB ?
          DB 11 DUP(0)
in_pwd      DB 7                       ;密码缓冲区
          DB ?
          DB 7 DUP(0)
in_goods DB 11                     ;商品名缓冲区
         DB ?
         DB 11 DUP(0)
DATA ENDS
CODE SEGMENT USE16 PARA PUBLIC 'CODE'
ASSUME CS:CODE,DS:DATA,SS:STACK
START:
 MOV AX,DATA
 MOV DS,AX
GONG_NENG1:                              ;----------------------------------------------------功能 1，提示
登陆
 CALL ClearS
 WRITE MSG1
 READ in_name
 CRLF

 WRITE MSG2
 READ in_pwd
 CRLF

 MOV BL,in_name[2]
 CMP BL,'q'
 JE    EXIT                       ;输入'q'，退出程序

 MOV BL,in_name[1]
 CMP BL,0
 JE UNLOGIN                       ;输入回车,转未登录
 JMP GONG_NENG2                   ;否则转登录

UNLOGIN:
 MOV AH,0
 MOV AUTH,AH                      ;标记登陆状态
 JMP GONG_NENG3                   ;未登录状态，转功能 3

GONG_NENG2:                              ;----------------------------------------------------功能 2，判断登陆
```

```
        XOR ECX,ECX
        MOV CL,in_name[1]          ;循环比较姓名
        MOV ESI,0
 LOOP1:
        MOV BL,in_name[ESI+2]
        CMP BL,BNAME[ESI]
        JNE ACCOUNTWRONG           ;姓名不匹配转 ACCOUNTWRONG
        INC ESI
        DEC ECX
        JNZ LOOP1

        MOV BL,0
        CMP BL,BNAME[ESI]
        JNZ ACCOUNTWRONG           ;输入的字符串是定义字符串的子集

        XOR ECX,ECX
        MOV CL,in_pwd[1]
        MOV ESI,0                  ;循环条件初始化
 LOOP2:
        MOV BL,in_pwd[ESI+2]
        CMP BL,BPASS[ESI]
        JNE ACCOUNTWRONG           ;密码不匹配转 ACCOUNTWRONG
        INC ESI
        DEC ECX
        JNZ LOOP2

        MOV BL,0
        CMP BL,BPASS[ESI]
        JNZ ACCOUNTWRONG
        JMP  SUCCESS               ;账号密码都相同转登陆成功

 ACCOUNTWRONG:                     ;登陆失败
        WRITE MSG3
        JMP GONG_NENG1

SUCCESS:                           ;登陆成功
 MOV AH,1
 MOV AUTH,AH
 JMP GONG_NENG3                    ;已登录状态，转功能 3

GONG_NENG3:                              ;-------------------------------------------------功能 3

 CALL ClearS
 CALL DisplayMenu
 CALL SelectFun

 CMP AL,'1'
 JE _FUN1
 CMP AL,'2'
 JE _FUN2
 CMP AL,'3'
 JE _FUN3
 CMP AL,'4'
 JE _FUN4
 CMP AL,'5'
 JE _FUN5
 CMP AL,'6'
 JE _FUN6
_FUN1:
 CALL QueryCommodity
 JMP _FUN_NEXT
_FUN2:
 CALL ModifyCommodity
```

```
 JMP _FUN_NEXT
_FUN3:
 CALL AVERAGR_PROFIT
 JMP _FUN_NEXT
_FUN4:
 CALL PROFIT_RANKING
 JMP _FUN_NEXT
_FUN5:
 CALL PRINT_IFO
 JMP _FUN_NEXT
_FUN6:
 JMP EXIT
_FUN_NEXT:
 MOV AH,01H
 INT 21H
 JMP GONG_NENG3


GONG_NENG4:                          ;-----------------------------------------------------功能 4，商品等级
判断
 MOV AX,PR1
 CMP AX,90
 JGE A_GRADE                  ;利润率大于 90%
 CMP AX,50
 JGE B_GRADE                  ;利润率大于 50%
 CMP AX,20
 JGE C_GRADE                  ;利润率大于 20%
 CMP AX,0
 JGE D_GRADE                  ;利润率大于 0%
 JMP F_GRADE

A_GRADE:
 OUT1 'A'
 CRLF
 JMP T
B_GRADE:
 OUT1 'B'
 CRLF
 JMP T
C_GRADE:
 OUT1 'C'
 CRLF
 JMP T
D_GRADE:
 OUT1 'D'
 CRLF
 JMP T
F_GRADE:
 OUT1 'F'
 CRLF
 JMP T
T:
 JMP GONG_NENG1

EXIT:
 MOV AH,4CH
      INT 21H                         ;退出程序

;------------------------------
;以 10 进制输出有符号整数
;EAX——需要输出的数
PRINTAX PROC
 PUSH EBX
```

```
        PUSH CX
    PUSH EDX                        ;保护现场
    MOV EBX,10
    XOR CX,CX                       ;计数器清 0
    CMP EAX,0
    JNL _LOP1
    NOT EAX
    ADD EAX,1
    PUSH EAX
    OUT1 '-'                    ;输出负号
    POP EAX
    _LOP1:                              ;(EAX)除以 P，所得商->EAX，余数入栈，CX++，记录余
数个数
    XOR EDX,EDX
    DIV EBX
    PUSH DX
    INC CX
    OR EAX,EAX
    JNZ _LOP1
    _LOP2:                              ;从栈中弹出一位 P 进制数，并将该数转换成 ASCII 码后输
出
    POP AX
    CMP AL,10
    JB _L1
    ADD AL,7
    _L1:                            ;输出 P 进制数
    ADD AL,30H
    MOV DL,AL
    MOV AH,2
    INT 21H
    LOOP _LOP2
    POP EDX
    POP CX
    POP EBX                         ;恢复现场
    RET
PRINTAX ENDP

;-------------------------------
;重复输出字符
;AL——需要重复输出的字符
;AH——需要重复输出的字符数
PrintASCII PROC
 PUSH ECX
 MOVSX ECX,AH
_PUTCHAR_:
 OUT1 AL
 LOOP _PUTCHAR_
 POP ECX
 RET
PrintASCII ENDP

;模块功能:显示菜单
;模块名称:DisplayMenu
;传入参数:无
;参数传入方式：无
;传出参数:无
;参数传出方式：无
;备注:负责人：吕鹏泽  同组同学：吴阳民
DisplayMenu PROC
 PUSH AX

 MOV AL,' '
```

```
 MOV AH,10
 CALL PrintASCII
 MOV AL,'-'
 MOV AH,60
 CALL PrintASCII
 CRLF

 MOV AH,AUTH
 CMP AH,0
 JE _DM_UNLOGIN_
 JMP _DM_LOGIN_
_DM_LOGIN_:
 MOV AL,' '
 MOV AH,20
 CALL PrintASCII
 WRITE MENU1
 CRLF
 MOV AL,' '
 MOV AH,20
 CALL PrintASCII
 WRITE MENU2
 CRLF
 MOV AL,' '
 MOV AH,20
 CALL PrintASCII
 WRITE MENU3
 CRLF
 MOV AL,' '
 MOV AH,20
 CALL PrintASCII
 WRITE MENU4
 CRLF
 MOV AL,' '
 MOV AH,20
 CALL PrintASCII
 WRITE MENU5
 CRLF
 MOV AL,' '
 MOV AH,20
 CALL PrintASCII
 WRITE MENU6
 CRLF
 JMP _DM_R_
_DM_UNLOGIN_:
 MOV AL,' '
 MOV AH,20
 CALL PrintASCII
 WRITE MENU1
 CRLF
 MOV AL,' '
 MOV AH,20
 CALL PrintASCII
 WRITE MENU6
 CRLF
 JMP _DM_R_
_DM_R_:
 MOV AL,' '
 MOV AH,10
 CALL PrintASCII
 MOV AL,'-'
 MOV AH,60
 CALL PrintASCII
 CRLF
 POP AX
```

```
 RET
DisplayMenu ENDP
;------------------------------
;根据登录状态选择功能
;AL——记录并返回选择的功能
SelectFun PROC
_S_RESTART:
 WRITE MSG5
 MOV AH,01H
 INT 21H
 MOV AH,AUTH
 CMP AH,0
 JE _S_UNLOGIN_
 JMP _S_LOGIN_
_S_LOGIN_:
 CMP AL,'1'
 JL _S_RESTART
 CMP AL,'6'
 JG _S_RESTART
 JMP _S_R_
_S_UNLOGIN_:
 CMP AL,'1'
 JE _S_R_
 CMP AL,'6'
 JE _S_R_
 JMP _S_RESTART
_S_R_:
 PUSH AX
 CRLF
 POP AX
 RET
SelectFun ENDP

;-------------------------------清屏
ClearS PROC
 PUSH AX
 MOV AH,00H
      MOV AL,03H
      INT 10H
 POP AX
 RET
ClearS ENDP

;功能  1
;模块功能:查询商品信息
;模块名称:QueryCommodity
;传入参数:无
;参数传入方式：无
;传出参数:无
;参数传出方式：无
;备注:负责人：吕鹏泽  同组同学：吴阳民
QueryCommodity PROC
 PUSH ECX
 PUSH EBX
 PUSH ESI
 PUSH EDX
_QC_START:
 WRITE MSG4
 READ in_goods
 CRLF
 MOV ECX,N
 MOV EBX,0
_QC_LOOP1:                              ;第一层循环，顺序查询 N 件商品
```

```
MOV ESI,0
MOV AL,in_goods[1]
CBW
MOV DX,AX                       ;DX 控制第二层循环
_QC_LOOP2:                      ;第二层循环，比较商品名称
      MOV AH,GA1[EBX][ESI]
      CMP AH,in_goods[ESI+2]
      JNE _QC_UNFIND                ;名称不同
      INC ESI
      DEC DX
      JZ _QC_JUDGE              ;找到商品,转 JUDGE
      JMP _QC_LOOP2

_QC_UNFIND:
ADD EBX,20
DEC ECX
JNZ _QC_LOOP1
JMP _QC_START                   ;未找到商品，重新输入商品名称
_QC_JUDGE:
MOV AH,GA1[EBX][ESI]
CMP AH,0
JNE _QC_UNFIND                      ;商品名称为子集,属于未找到
JMP _QC_FIND
_QC_FIND:
;显示商店 1 商品
MOV AH,40
MOV AL,'-'
CALL PRINTASCII
CRLF
WRITE S1
OUT1 ':'
OUT1 ' '

MOVSX SI,BYTE PTR in_goods[1]
MOV AH,'$'
    MOV in_goods[SI+2],AH
WRITE in_goods[2]
CRLF

WRITE Purchase_price
OUT1 ':'
MOVSX EAX,WORD PTR GA1[EBX+10]
CALL PRINTAX
CRLF

WRITE Selling_price
OUT1 ':'
MOVSX EAX,WORD PTR GA1[EBX+12]
CALL PRINTAX
CRLF

WRITE Total_purchases
OUT1 ':'
MOVSX EAX,WORD PTR GA1[EBX+14]
CALL PRINTAX
CRLF

WRITE Sold_quantity
OUT1 ':'
MOVSX EAX,WORD PTR GA1[EBX+16]
CALL PRINTAX
CRLF
```

```
;显示商店 2 商品
 MOV AH,40
 MOV AL,'-'
 CALL PRINTASCII
 CRLF
 WRITE S2
 OUT1 ':'
 OUT1 ' '
 WRITE in_goods[2]
 CRLF

 WRITE Purchase_price
 OUT1 ':'
 MOVSX EAX,WORD PTR GB1[EBX+10]
 CALL PRINTAX
 CRLF

 WRITE Selling_price
 OUT1 ':'
 MOVSX EAX,WORD PTR GB1[EBX+12]
 CALL PRINTAX
 CRLF

 WRITE Total_purchases
 OUT1 ':'
 MOVSX EAX,WORD PTR GB1[EBX+14]
 CALL PRINTAX
 CRLF

 WRITE Sold_quantity
 OUT1 ':'
 MOVSX EAX,WORD PTR GB1[EBX+16]
 CALL PRINTAX
 CRLF
 POP EDX
 POP ESI
 POP EBX
 POP ECX
 RET
QueryCommodity ENDP

;功能  2
;模块功能:修改商品信息
;模块名称:ModifyCommodity
;传入参数:无
;参数传入方式：无
;传出参数:无
;参数传出方式：无
;备注:负责人：吕鹏泽  同组同学：吴阳民
ModifyCommodity PROC
 PUSHAD
_MC_START:
 WRITE SelectShop
 CRLF
 MOV AH,01H
 INT 21H
 CMP AL,'1'                              ;选择商店 1
 JE _MC_1
 CMP AL,'2'
 JE _MC_2                            ;选择商店 2
 JMP _MC_START                           ;输入错误
_MC_1:
```

```
LEA EBX,GA1
JMP _MC_3
_MC_2:
LEA EBX,GB1
JMP _MC_3
_MC_3:
CRLF
WRITE MSG4
READ in_goods
CRLF
CMP in_goods[1],0
JE _MC_R                    ;输入回车

MOV ECX,N                   ;查询商品
_MC_LOOP1:
MOV ESI,0
MOV AL,in_goods[1]
CBW
MOV DX,AX                   ;DX 控制第二层循环
_MC_LOOP2:                  ;第二层循环，比较商品名称
    MOV AH,DS:[EBX][ESI]
    CMP AH,in_goods[ESI+2]
    JNE _MC_UNFIND              ;名称不同
    INC ESI
    DEC DX
    JZ _MC_JUDGE            ;找到商品,转 JUDGE
    JMP _MC_LOOP2

_MC_UNFIND:
ADD EBX,20
DEC ECX
JNZ _MC_LOOP1
JMP _MC_START                   ;未找到商品，重新输入商品名称
_MC_JUDGE:
MOV AH,DS:[EBX][ESI]
CMP AH,0
JNE _MC_UNFIND                      ;商品名称为子集,属于未找到
JMP _MC_FIND
_MC_FIND:

_MC_T1:
CRLF
WRITE Purchase_price
OUT1 ':'
MOVSX EAX,WORD PTR DS:[EBX+10]
CALL PRINTAX
OUT1 '>'
CALL INPUT_NEW_INFO
CMP CX,-1
JE _MC_T1
CMP CX,-2
JE _MC_T2
MOV WORD PTR DS:[EBX+10],CX
CRLF
_MC_T2:
CRLF
WRITE Selling_price
OUT1 ':'
MOVSX EAX,WORD PTR DS:[EBX+12]
CALL PRINTAX
OUT1 '>'
CALL INPUT_NEW_INFO
CMP CX,-1
```

```
 JE _MC_T2
 CMP CX,-2
 JE _MC_T3
 MOV WORD PTR DS:[EBX+12],CX
 CRLF
_MC_T3:
 CRLF
 WRITE Total_purchases
 OUT1 ':'
 MOVSX EAX,WORD PTR DS:[EBX+14]
 CALL PRINTAX
 OUT1 '>'
 CALL INPUT_NEW_INFO
 CMP CX,-1
 JE _MC_T3
 CMP CX,-2
 JE _MC_T4
 MOV WORD PTR DS:[EBX+14],CX
 CRLF
_MC_T4:
 CRLF
 WRITE Sold_quantity
 OUT1 ':'
 MOVSX EAX,WORD PTR DS:[EBX+16]
 CALL PRINTAX
 OUT1 '>'
 CALL INPUT_NEW_INFO
 CMP CX,-1
 JE _MC_T4
 CMP CX,-2
 JE _MC_R
 MOV WORD PTR DS:[EBX+16],CX
 CRLF

_MC_R:
 POPAD
 RET
ModifyCommodity ENDP

;--------------------------------
;从键盘输入 10 进制数，结果存在 CX 中
;传出参数：CX,输入的 10 进制数,如果输入错误则 CX=-1,如果输入回车 CX=-2
INPUT_NEW_INFO PROC
 PUSH AX
 PUSH BX
 MOV CX,0
 MOV AH,01H
 INT 21H
 CMP AL,0DH
 JE _INI_0DH                    ;只输入了回车
 JMP _INI_JUDGE
_INI_START:
 MOV AH,01H
 INT 21H
 CMP AL,0DH
 JE    _INI_R
_INI_JUDGE:
 CMP AL,'0'
 JL _INI_WRONG                  ;输入字符不合法
 CMP AL,'9'
 JG _INI_WRONG                  ;输入字符不合法
 SUB AL,'0'                     ;转换成数字
 MOVSX BX,AL
```

```
 IMUL CX,10
 ADD CX,BX
 JMP _INI_START
_INI_WRONG:
 MOV CX,-1
 JMP _INI_R
_INI_0DH:
 MOV CX,-2
 JMP _INI_R
_INI_R:
 POP BX
 POP AX
 RET
INPUT_NEW_INFO ENDP

CODE ENDS
END START
```

3.1.3.2 吴阳民负责模块源代码(T1.ASM)

```
;corporation@tortoiselala
;
;data:20180413
;partner:pengzeLv

 INCLUDE MACRO.LIB
 EXTRN    GA1:BYTE, GB1:BYTE, S1:BYTE, S2:BYTE, PrintASCII:NEAR, PRINTAX:NEAR
 PUBLIC AVERAGR_PROFIT, PROFIT_RANKING, PRINT_IFO

.386
DATA SEGMENT USE16 PARA PUBLIC 'D1'
 N EQU 10

 SORT_BUF DW N DUP(0)
 SORT_RAN DW N DUP(0)
 Product_name DB 'Product name', '$'
 Purchase_price DB 'Purchase price', '$'
 Selling_price DB 'Selling price', '$'
 Total_purchases DB 'Total purchases', '$'
 Sold_quantity DB 'Sold Quantity', '$'
 Profit_rate DB 'Profit rate', '$'
 Ranking DB 'Ranking', '$'
DATA ENDS
STACK SEGMENT USE16 PARA STACK 'STACK'
 DB 200 DUP(0)
STACK ENDS

CODE SEGMENT USE16 PARA PUBLIC 'CODE'
      ASSUME CS:CODE, DS:DATA, SS:STACK
START:
      MOV AX, DATA
      MOV DS, AX


;功能  3
;模块功能:计算平均利润率
;模块名称:AVERAGR_PROFIT
;传入参数:无
;参数传入方式
;传出参数:无
;参数传出方式：无
```

```
;备注:无
AVERAGR_PROFIT PROC
 PUSHA
                    ;����������Ĵ����������
 MOV EBX, 0
 LEA AX, GA1
 MOV SI, N
                    ;��h��ъ��Ïn������3���n����3
LOOP_1:
 MOVSX EAX, WORD PTR GA1[EBX][12]
 MOVSX EDX, WORD PTR GA1[EBX][16]
 IMUL EAX, EDX

 MOVSX ECX, WORD PTR GA1[EBX][10]
 MOVSX EDX, WORD PTR GA1[EBX][14]
 IMUL ECX, EDX

 SUB EAX, ECX
 IMUL EAX, 100
 CDQ
 IDIV ECX

 MOV WORD PTR GA1[EBX][18], AX

 MOVSX EAX, WORD PTR GB1[EBX][12]
 MOVSX EDX, WORD PTR GB1[EBX][16]
 IMUL EAX, EDX

 MOVSX ECX, WORD PTR GB1[EBX][10]
 MOVSX EDX, WORD PTR GB1[EBX][14]
 IMUL ECX, EDX

 SUB EAX, ECX
 IMUL EAX, 100
 CDQ
 IDIV ECX

 MOV WORD PTR GB1[EBX][18], AX

 MOV AX, WORD PTR GA1[EBX][18]
 ADD AX, WORD PTR GB1[EBX][18]

 SAR AX, 1
 MOV WORD PTR GA1[EBX][18], AX

 ADD EBX, 20
 DEC SI

 JNZ LOOP_1

 POPA
 RET

AVERAGR_PROFIT ENDP

;功能 4
;模块功能:计算利润率排名
;模块名称:PROFIT_RANKING
;传入参数:无
;参数传入方式:无
;传出参数:无
;参数传出方式：无
;备注:使用冒泡排序，使用两个一维数组来分别保存平均利润率和原偏移地址，之后再根据偏移地址写
```

入排名　　perfect

```
PROFIT_RANKING PROC
 PUSHA
        ;保护现场

 MOV AL, N
 MOV EBX, 0
 LOOP_2:
    MOV AH, N          ;内层循环计数
    MOV CX, 0          ;计数器，大于该利润的商品数量
    MOV EBP, 0

    LOOP_3:
        MOV DX, WORD PTR GA1[EBP][18]
        CMP DX, WORD PTR GA1[EBX][18]

        JNG CONTINUE
        INC CX

    CONTINUE:
        ADD EBP, 20

        DEC AH
        JNZ LOOP_3
    INC CX
    MOV WORD PTR GB1[EBX][18], CX
    ADD EBX, 20

    DEC AL
    JNZ LOOP_2

 POPA
 RET

PROFIT_RANKING ENDP


;功能 5
;模块功能:输出商品信息
;模块名称:PRINT_IFO
;传入参数:0,1,(0 代表输出 shop1 信息，1 代表输出 shop2 信息）
;参数传入方式:al
;传出参数:无
;参数传出方式：无
;备注:;如果入口参数为 0，输出 shop1 信息（利润率），入口参数为 1， 输出 shop2 信息（排名）,al 为
传参寄存器

PRINT_IFO PROC
 PUSHA

 CMP AL, 0
 JZ CALL_1
 JMP CALL_2
CALL_1:
 CALL PRINT_SHOP1
 JMP EXIT_FUN
CALL_2:
 CALL PRINT_SHOP2
 JMP EXIT_FUN
EXIT_FUN:
```

```
 POPA
 RET
PRINT_IFO ENDP

PRINT_SHOP1 PROC
 PUSH EAX
 PUSH SI
 PUSH BP
 ;输出店铺名称
 WRITE S1
 CRLF
 ;输出标题栏
 WRITE Product_name
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII

 WRITE Purchase_price
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII

 WRITE Selling_price
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII

 WRITE Total_purchases
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII

 WRITE Sold_quantity
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII

 WRITE    Profit_rate
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII
 ;输出标题栏的换行
 CRLF

 ;循环输出 shop1 信息，循环条件为 bp
 MOV BP, N
 LEA SI, GA1
 START_PRINT_IFO:
        ;首先输出商品名称
        MOV BX, SI
        CALL WRITE_TEN_CHAR
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII
        ;将 5 个数字有符号数字转换为 ascii，并输出
        MOVSX EAX, WORD PTR [SI][10]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        MOVSX EAX, WORD PTR [SI][12]
        CALL PRINTAX
```

```
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        MOVSX EAX, WORD PTR [SI][14]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        MOVSX EAX, WORD PTR [SI][16]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        MOVSX EAX, WORD PTR [SI][18]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        ;输出空格
        CRLF
        ADD SI, 20
        DEC BP
        JNZ START_PRINT_IFO


 POP BP
 POP SI
 POP EAX
 RET
PRINT_SHOP1 ENDP



PRINT_SHOP2 PROC
 PUSHA
 ;输出店铺名称
 WRITE S2
 CRLF
 ;输出标题栏

 WRITE Product_name
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII

 WRITE Purchase_price
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII

 WRITE Selling_price
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII

 WRITE Total_purchases
 MOV AL, ' '
 MOV AH, 5
 CALL PrintASCII
```

```
WRITE Sold_quantity
MOV AL, ' '
MOV AH, 5
CALL PrintASCII

WRITE    Ranking
MOV AL, ' '
MOV AH, 5
CALL PrintASCII
;输出标题栏的换行
CRLF

;循环输出 shop1 信息，循环条件为 bp
MOV BP,   N
LEA SI, GB1
START_PRINT_IFO_:
        ;首先输出商品名称
        MOV BX, SI
        CALL WRITE_TEN_CHAR
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        ;将 5 个数字有符号数字转换为 ascii，并输出
        MOVSX EAX, WORD PTR [SI][10]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        MOVSX EAX, WORD PTR [SI][12]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        MOVSX EAX, WORD PTR [SI][14]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        MOVSX EAX, WORD PTR [SI][16]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        MOVSX EAX, WORD PTR [SI][18]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 5
        CALL PrintASCII

        ;输出空格
        CRLF
        ADD SI, 20
        DEC BP
        JNZ START_PRINT_IFO_

 POPA
 RET
PRINT_SHOP2 ENDP
```

```
;函数 write_ten_char 用于输出连续的 10 个字符，bx 为起始字符偏移地址
;调用方法
;MOV BX, OFFSET GA1
;LEA BX, GA1
;（GA1 为 ds 段的变量名）
WRITE_TEN_CHAR PROC
 ;现场保护
 PUSH DX
 PUSH AX
 ;初始化循环条件
 MOV DH, 10
 ;开始循环
 START_PRINT:
        ;调用系统功能输出
        MOV DL, DS:[BX]
        MOV AH, 2
        INT 21H

        INC BX
        DEC DH
        JNZ START_PRINT
 POP AX
 POP DX
 RET
WRITE_TEN_CHAR ENDP

CODE ENDS
      END START
```

3.1.3.3 LIB 文件源码(MACRO.LIB)

```
READ        MACRO A
        LEA DX,A
        MOV AH,10
        INT 21H
        ENDM
WRITE       MACRO A
        LEA DX,A
        MOV AH,9
        INT 21H
        ENDM
CRLF MACRO
        MOV AH,2
        MOV DL,0AH
        INT 21H
        MOV DL,0DH
        INT 21H
        ENDM
OUT1 MACRO A
        MOV DL,A
        MOV AH,2
        INT 21H
        ENDM
STACK0     MACRO A
STACK       SEGMENT USE16 PARA STACK 'STACK'
        DB A
STACK       ENDS
        ENDM
```

3.1.4 实验步骤

1.准备上机环境，编辑、汇编、文件 T

2.LINK 文件 T 和同组同学的文件 T1

3.以游客身份登陆系统，测试功能 1、6

4.输入姓名、密码登陆系统

5.测试功能 1-6

5.1 美化菜单

5.2 利用宏定义增强程序可读性

6.观察并记录程序运行信息

7.观察子程序调用堆栈的变化

8.观察 FAR、NEAR 调用的区别

9.尝试多种子程序间传递参数的方法

10.测试公共符号和外部引用符号不一致时会发生什么。

3.1.5 实验记录与分析

实验环境条件：WINDOWS 10 下 DOSBox0.72； TD.EXE 5.0

1.以游客身份登录后的界面显示如图 3.1.6 所示，只有功能 1 和功能 6 可用。

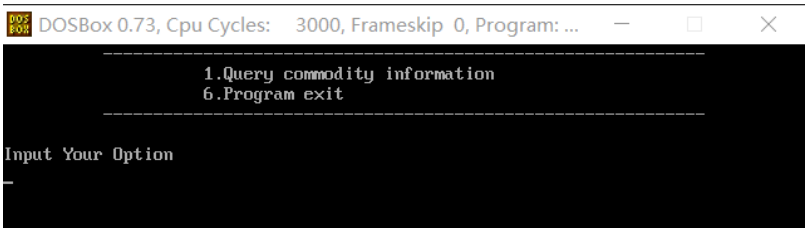


图 3.1.6 游客界面

当输入不合法的字符时，系统会让用户重新输入，直到输入了正确的字符，演示过程如图 3.1.7 所示。

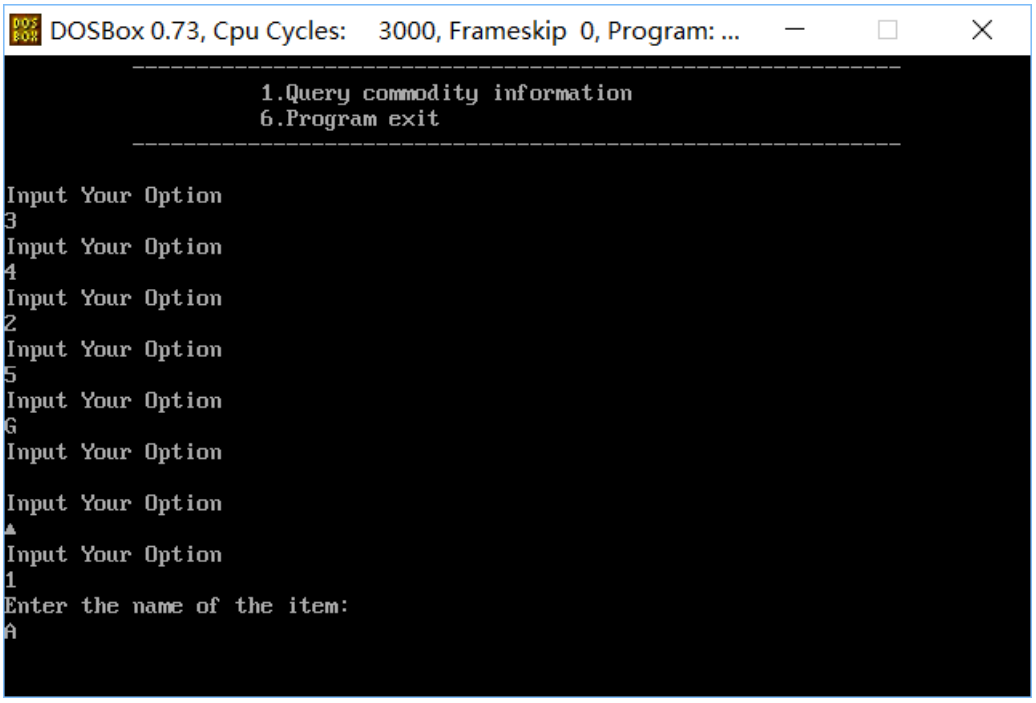


图 3.1.7 选项输入

只有在输入了正确的商品名后，系统会显示商店 1 和商店 2 的商品信息，结果如图 3.1.8 所示。

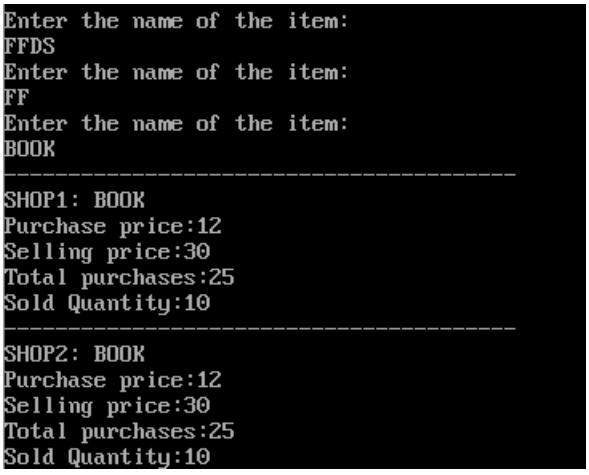


图 3.1.8 查询商品信息

然后按任意键回到菜单界面，输入 6 后退出系统，演示过程如图 3.1.9 所示

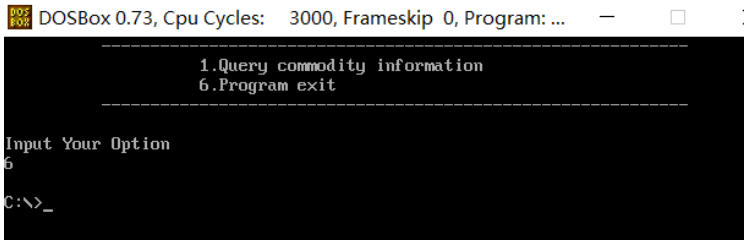


图 3.1.9 退出系统

2.以账号密码登录后的菜单如图 3.1.10 所示，有功能 1-6 可选，其中功能 1、6 同未登录状态的功能 1、6.

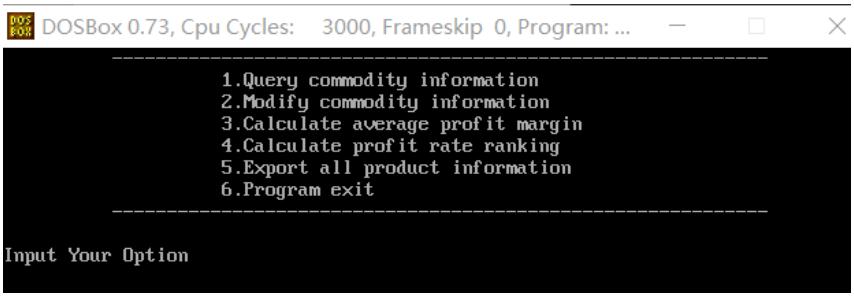


图 3.1.10 登录状态的系统界面

输入 2 后进入功能 2，选择商店和商品后进行信息修改，输入回车不修改，输入非法字符系统会让用户重新输入，结果如图 3.1.11 所示。

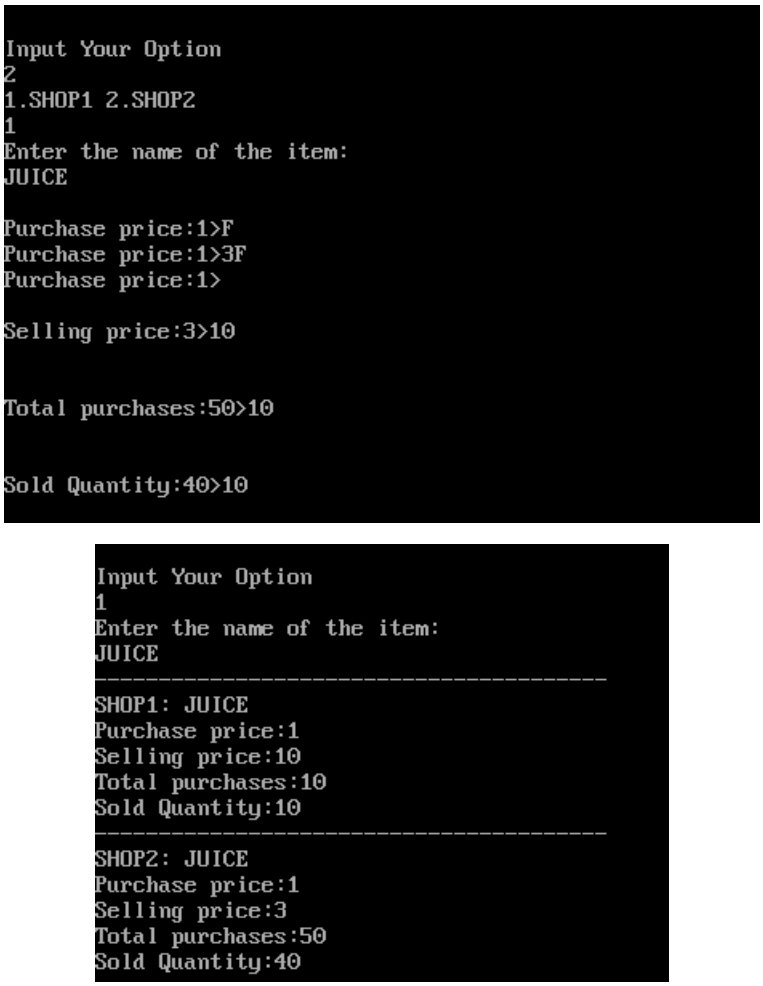


图 3.1.11 商品信息修改

运行功能 3、4 后使用功能 5 输出所有商品信息，结果如图 3.1.12 所示

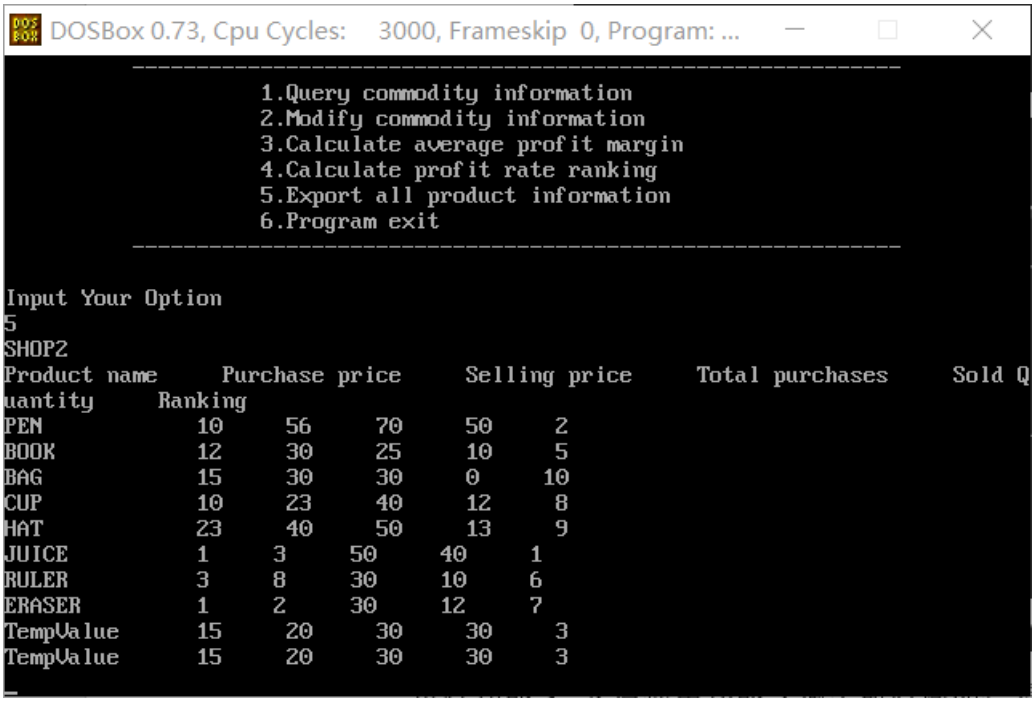


图 3.1.12 所有商品信息

3.1.6 思考题

1. 在 TD 中跟踪到子程序内部有几种方法？在 TD 中观察子程序调用和返回时堆栈的变化。

（1）如图 3.1.13 所示，TD 跟踪到子程序内部有三种方法，分别为 Run,Go to cursor,Step over

方法一：先在子程序内部设置断点（快捷键 F2 设置断点），然后选取 Run 方式（快捷键 F9）运行程序，在断电处会自动停下。

方法二：将光标放在子程序内部的指令处，选取 Go to cursor 方式（快捷键 F4）运行程序，在光标处会自动停下。

方法三：在执行到 call 子程序的指令时使用 Trace into 方式运行程序（快捷键 F7），会进入到子程序内部。

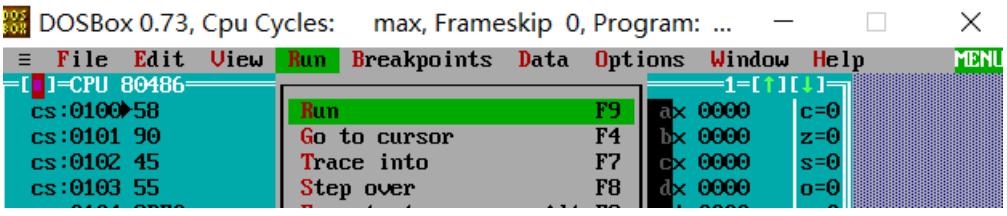


图 3.1.13 跟踪到子程序

（2）如图 3.1.13 a 和 3.1.13 b 所示，调用子程序前 sp 的值位 03E8H，调用后 sp 的值为

03E6H，发现有两个字节的数据进栈了，观察 SS:[SP]的数据，为 ss:03E6 C9 00，正好是调用的子程序的下一步指令的 IP 地址，如图 3.1.13c 所示子程序返回后 ip 的值正好是 00C9H，显然在调用子程序的时候，程序将子程序的出口的 IP 压入堆栈段，等到子程序运行完毕 RET 的时候，将 IP 地址出栈，作为执行下一步语句的 IP。

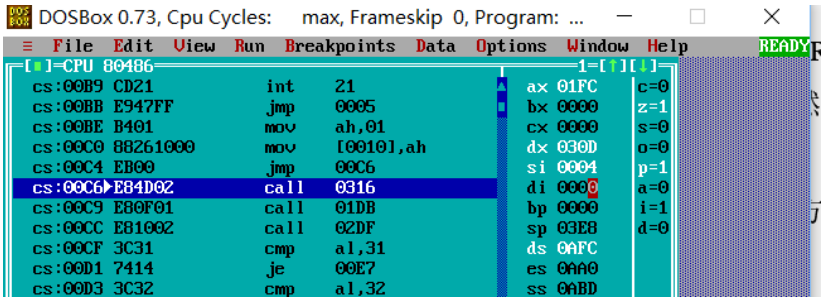


图 3.1.13 a 调用子程序前

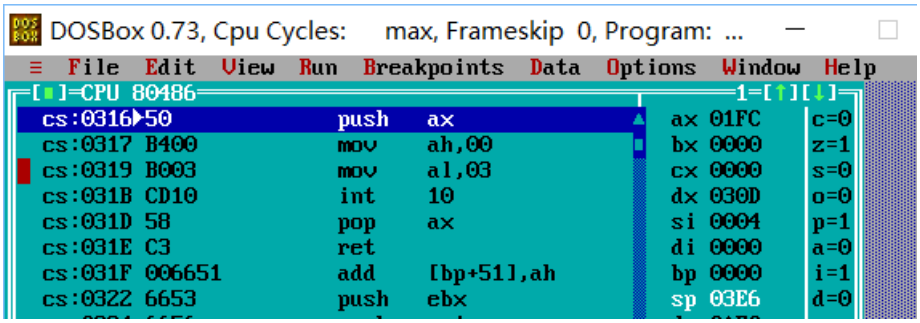


图 3.1.13 b 进入子程序后

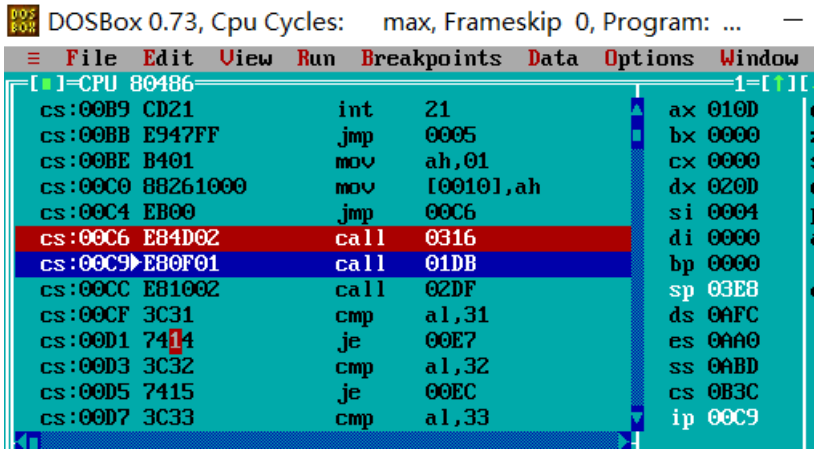


图 3.1.13 c 子程序返回

2. 注意观察 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。

（1）FAR 类型子程序 RETF 指令的机器码为 0CBH，NEAR 类型子程序的 RET 指令机器码为 0C3H.

（2）如图 3.1.14a 和图 3.1.14b，通过观察子程序调用前后可知堆栈有 4 个字节的数据

进栈，进一步观察堆栈数据域可以发现进栈的值是主程序的 CS 的内容 0B19 和调用子程序的下一步指令的 IP 的内容 00D1。如图 3.1.14c，子程序返回后按照 IP、CS 的顺序出栈 4 个字节的内容，作为下一步指令的执行地址。

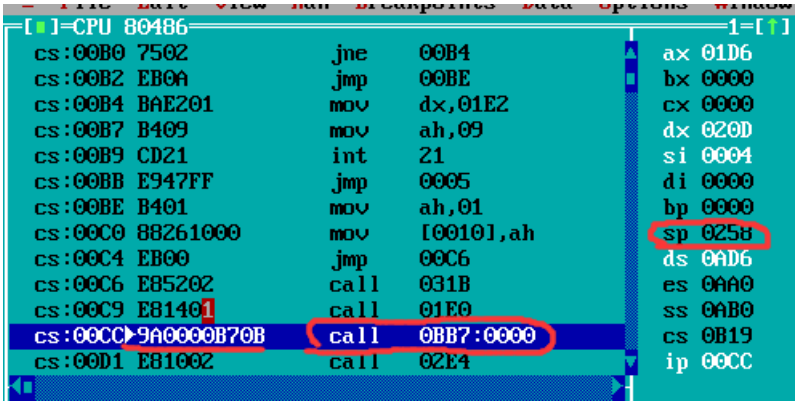


图 3.1.14a 调用子程序前

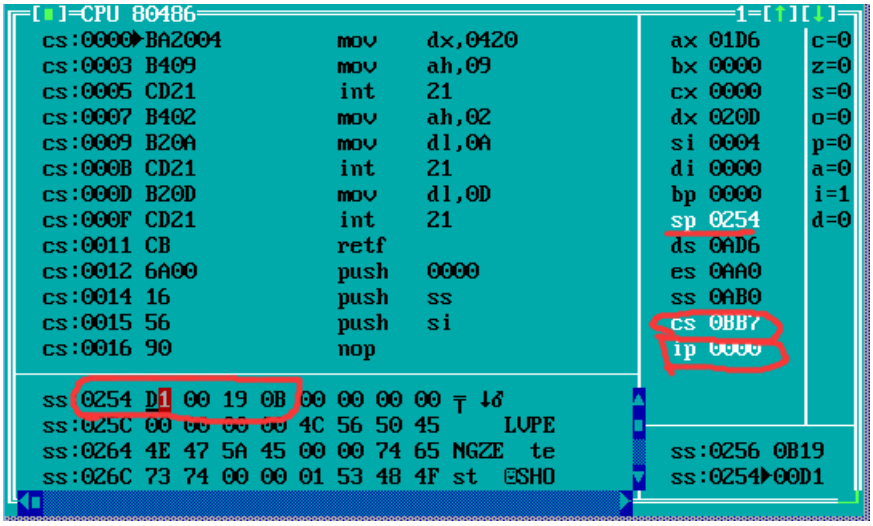


图 3.1.14b 调用子程序后

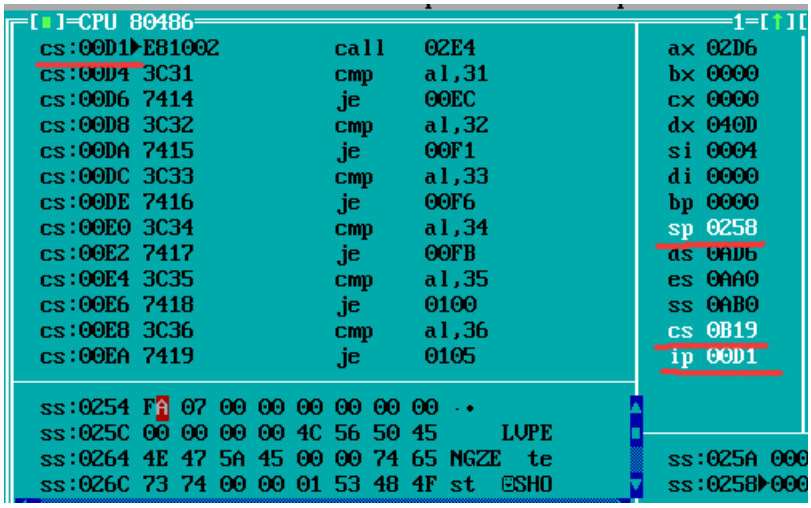


图 3.1.14c 子程序返回

3. 观察模块间的参数的传递方法，包括公共符号的定义和外部符号的引用，若符号名

不一致或类型不一致会有什么现象发生？

在编译时正常，但在连接时会报错，如图 3.1.15 所示。

图 3.1.15 链接报错

4. 通过 TD 观察宏指令在执行程序中的替换和扩展，解释宏和子程序的调用有何不同。

宏会在汇编时进行展开，多次调用宏则会展开多次，而子程序在使用时会跳转到子程序执行，执行完毕后再跳转回主程序。

5. EXTRN 说明语句放在.386 之前或者之后有什么区别？

经测试发现放在.386 之前或之后均能正常运行。

6.如何利用宏功能使汇编语言的程序变得更加直观易读？

将常用的 2 好号调用、9 号调用、10 号调用等常用 DOS 调用写成宏汇编的形式

3.2 任务 2

3.2.1 设计思想及存储单元分配

使用 C 语言写用户登录界面和菜单功能选择界面和功能 1，将商品在菜单中使用 switch 语句调用任务 1 中用汇编写的 4 个功能模块，调试成功后将功能 1(查询商品信息)用 C 语言实现，继续调试。

存储单元分配：汇编源程序的变量同任务 1，C 程序使用 char 型变量 AUTH，用于记录登录状态。char []变量 in_name,in_pwd 用于缓存用户输入的账号密码。char 型变量 op 用于记录用户选择，商店和商品信息定义成结构体的形式，其中商品名为 10 个字节，商品进价、售价、进货量、售出量、利润率各为 2 个字节，共 10 个字节。

3.2.2 流程图

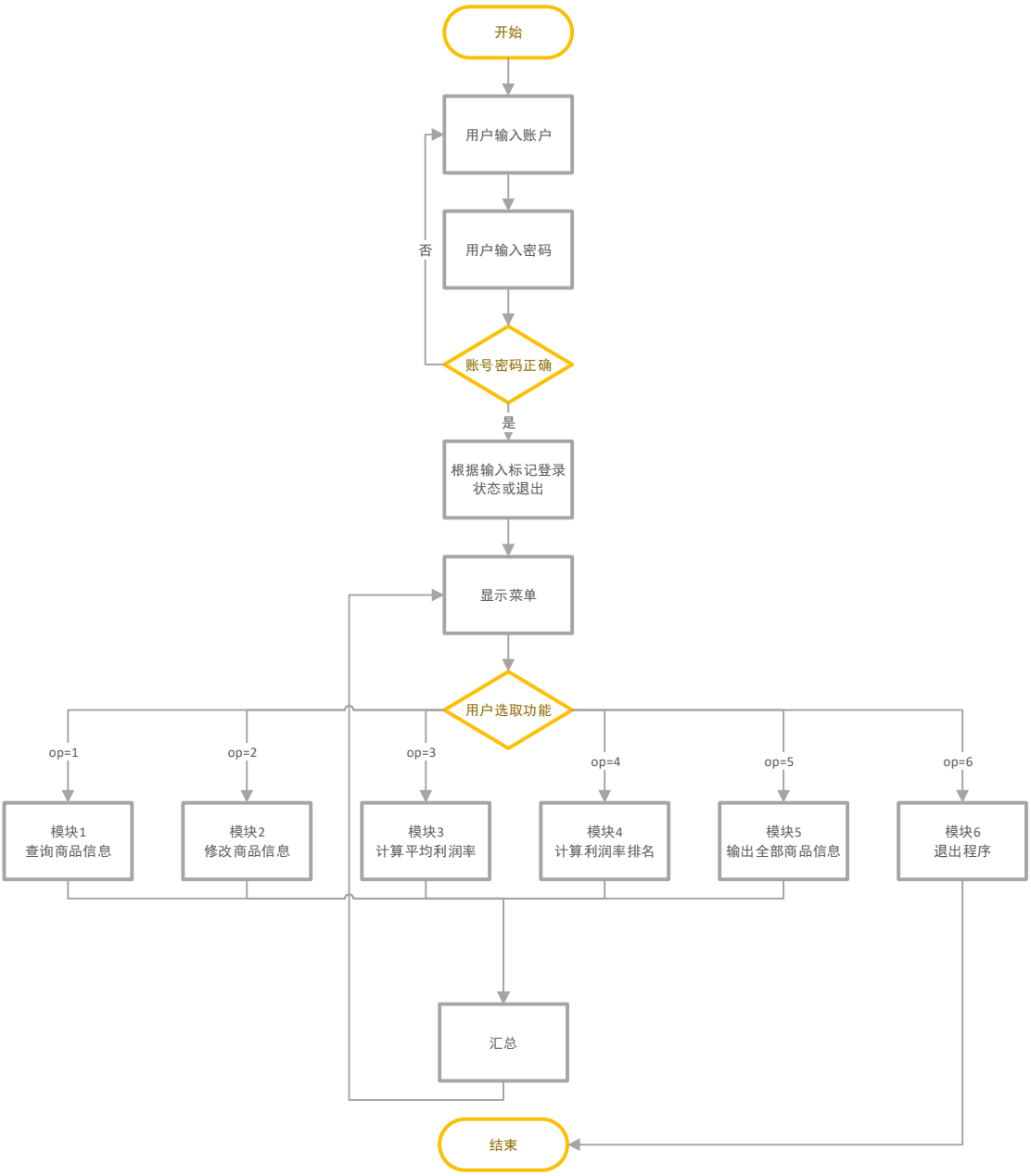


图 3.2.1 主程序流程图

3.2.3 源程序

商品数据定义、用户登录、菜单界面、功能 1——查询商品信息用 C 语言实现，功能 2-5 用汇编语言实现。

3.2.3.1 C 程序源代码

```
#include <stdio.h>
#include <string.h>
#include<stdlib.h>
#define BNAME "LVPENGZE"
#define BPASS "test"
#define N 10
extern   void   QueryCommodity(struct commodity *,struct commodity *,short);
extern   void   ModifyCommodity(struct commodity *,struct commodity *,short);
extern   void   AVERAGR_PROFIT(struct commodity *,struct commodity *,short);
extern   void   PROFIT_RANKING(struct commodity *,struct commodity *,char);
extern   void   PRINT_IFO(struct commodity *,struct commodity *,short);
extern   void   ClearS(void);

void menu1(void);
void menu2(void);
void C_query_commodity(struct commodity*,struct commodity *,short);
struct commodity
{
 char cname[10];//商品名
 short purchase_price;//购价
 short selling_price;//售价
 short total;//进货总数
 short sell;//售出数
 short prt;//利润率
};
struct shop
{
 char sname[6];//商店名
 struct commodity good[N];//商品信息
};
struct shop S[2];
int i, j;
struct commodity *pc,*temp1,*temp2;

int flat;
char AUTH;
char in_name[11];
char in_pwd[7];
char in_commodity[11];
char c;
char op;
int main(void)
{
 strcpy(S[0].sname, "SHOP1");
 strcpy(S[1].sname, "SHOP2");


 for (i = 0; i <= 1; i++)
 {
        pc = &(S[i].good[0]);
        strcpy(pc->cname, "PEN");
        pc->purchase_price = 10;
        pc->selling_price = 56;
        pc->total = 70;
        pc->sell = 50;
 }
 for (i = 0; i <= 1; i++)
 {
        pc = &(S[i].good[1]);
        strcpy(pc->cname, "BOOK");
```

```
        pc->purchase_price = 12;
        pc->selling_price = 30;
        pc->total = 25;
        pc->sell = 10;
    }
    for (i = 0; i <= 1; i++)
    {
        pc = &(S[i].good[2]);
        strcpy(pc->cname, "BAG");
        pc->purchase_price = 15;
        pc->selling_price = 30;
        pc->total = 30;
        pc->sell = 0;
    }
    for (i = 0; i <= 1; i++)
    {
        pc = &(S[i].good[3]);
        strcpy(pc->cname, "CUP");
        pc->purchase_price = 10;
        pc->selling_price = 23;
        pc->total = 40;
        pc->sell = 12;
    }
    for (i = 0; i <= 1; i++)
    {
        for (j = 4; j < N; j++)
        {
            pc = &(S[i].good[j]);
            strcpy(pc->cname, "TempValue");
            pc->purchase_price = 10;
            pc->selling_price = 10;
            pc->total = 10;
            pc->sell = 10;
        }
    }
Start:
    printf("Input your account:\n");
    gets(in_name);
    printf("Input your password:\n");
    gets(in_pwd);
    if (in_name[0] == 'q')//输入‘q'
    {
        return 0;
    }

    if (strlen(in_name) == 0)//输入回车
    {
        AUTH = 0;
    }
    else
    {
        while (!(strcmp(BNAME, in_name) == 0 && strcmp(BPASS, in_pwd) == 0))
        {
            printf("WRONG ACCOUNT!\n");
            goto Start;
        }
        printf("Login success\n");
        AUTH = 1;
    }
    if (AUTH == 1)
    {
        while (1)
        {
            ClearS();
```

```
                menu1();
                while (!scanf("%d", &op))
                {
                        getchar();
                }
                getchar();
                switch (op)
                {
                case 1:C_query_commodity(&S[0].good[0],&S[1].good[0],N); break;
                case 2:ModifyCommodity(&S[0].good[0],&S[1].good[0],N); break;
                case 3:AVERAGR_PROFIT(&S[0].good[0],&S[1].good[0],N); break;
                case 4:PROFIT_RANKING(&S[0].good[0],&S[1].good[0],N); break;
                case 5:PRINT_IFO(&S[0].good[0],&S[1].good[0],N); break;
                case 6:return 0;
                default:printf("WRONG INPUT!\n");
                        break;
                }
                getchar();
        }
 }
 else
 {
        while (1)
        {
                ClearS();
                menu2();
                while (!scanf("%d", &op))
                {
                        getchar();
                }
                getchar();
                switch (op)
                {
                case 1: C_query_commodity(&S[0].good[0], &S[1].good[0], N); break;
                case 6:return 0;
                default:printf("WRONG INPUT!\n");
                        break;
                }
                getchar();
        }
 }

}
void menu1(void)
{
 printf("                ------------------------------------------------------\n");
 printf("                |               Select your op:                      |\n");
 printf("                |               1.Query commodity information        |\n");
 printf("                |               2.Modify commodity information       |\n");
 printf("                |               3.Calculate average profit margin    |\n");
 printf("                |               4.Calculate profit rate ranking      |\n");
 printf("                |               5.Export all product information     |\n");
 printf("                |               6.Program exit                       |\n");
 printf("                ------------------------------------------------------\n");
}
void menu2(void)
{
 printf("                ------------------------------------------------------\n");
 printf("                |               Select your op:                      |\n");
 printf("                |               1.Query commodity information        |\n");
 printf("                |               6.Program exit                       |\n");
 printf("                ------------------------------------------------------\n");
}
void C_query_commodity(struct commodity* p1,struct commodity* p2,short num)
```

```
{
 flat = 0;
 while (!flat)
 {
        printf("Input the name of the commodity('q' to quit):\n\n");
        scanf("%s", in_commodity);
        temp1 = p1; temp2 = p2;
        if (in_commodity[0] == 'q')
                return;
        for (i = 0; i < num&&flat == 0; i++)
        {
                if (strcmp(in_commodity, temp1->cname) == 0)
                        flat = 1;
                temp1++; temp2++;
        }
        if(flat==0)
                printf("Can't find the commodity,please input again\n\n");
 }
 temp1--;temp2--;
 printf("-------------------------------------\n");
 printf("Commodity Name: %s\n", temp1->cname);
 printf("Purchase Price: %d\n", temp1->purchase_price);
 printf("Selling Price: %d\n", temp1->selling_price);
 printf("Total Purchases: %d\n", temp1->total);
 printf("Sold Quantity: %d\n", temp1->sell);
 printf("Profit Rate: %d\n", temp1->prt);
 printf("-------------------------------------\n");
 printf("Commodity Name: %s\n", temp2->cname);
 printf("Purchase Price: %d\n", temp2->purchase_price);
 printf("Selling Price: %d\n", temp2->selling_price);
 printf("Total Purchases: %d\n", temp2->total);
 printf("Sold Quantity: %d\n", temp2->sell);
 printf("Ranking: %d\n", temp2->prt);
 putchar('\n');
 system("pause");
 return;
}
```

3.2.3.2 ASM 程序源代码

```
        PUBLIC
_QueryCommodity,_ModifyCommodity,_AVERAGR_PROFIT,_PROFIT_RANKING,_PRINT_IFO,_ClearS,_SE
TDS
        INCLUDE MACRO.LIB
        .386
        _STACK1 SEGMENT USE16 PARA STACK 'STACK'
         DB 400 DUP(0)
        _STACK1 ENDS
        _DATA SEGMENT USE16 PARA PUBLIC 'DATA'

        MSG4 DB 'Enter the name of the item:',0DH,0AH,'$'

        SelectShop DB '1.SHOP1 2.SHOP2','$'
        Product_name DB 'Commodity', '$'
        Purchase_price DB ' cost ', '$'
        Selling_price DB ' price ', '$',0
        Total_purchases DB 'Total Quantity', '$'
        Sold_quantity DB 'Sold Quantity', '$'
        Profit_rate DB 'Profit rate', '$'
        Ranking DB 'Ranking', '$'
        S1 DB 'SHOP1','$'
        S2 DB 'SHOP2','$'
```

```
in_goods DB 11                          ;商品名缓冲区
         DB ?
         DB 11 DUP(0)

_DATA ENDS
_TEXT SEGMENT USE16 PARA PUBLIC 'CODE'
ASSUME CS:_TEXT,DS:_DATA,SS:_STACK1
_SETDS PROC
 MOV AX,_DATA
 MOV DS,AX
 _SETDS ENDP
;------------------------------
;以 10 进制输出有符号整数
;EAX——需要输出的数
PRINTAX PROC
 PUSH EBX
      PUSH CX
 PUSH EDX                               ;保护现场
 MOV EBX,10
 XOR CX,CX                              ;计数器清 0
 CMP EAX,0
 JNL _LOP1
 NOT EAX
 ADD EAX,1
 PUSH EAX
 OUT1 '-'                           ;输出负号
 POP EAX
 _LOP1:                                     ;(EAX)除以 P，所得商->EAX，余数入栈，CX++，记录余
数个数
 XOR EDX,EDX
 DIV EBX
 PUSH DX
 INC CX
 OR EAX,EAX
 JNZ _LOP1
 _LOP2:                                     ;从栈中弹出一位 P 进制数，并将该数转换成 ASCII 码后输
出
 POP AX
 CMP AL,10
 JB _L1
 ADD AL,7
 _L1:                                   ;输出 P 进制数
 ADD AL,30H
 MOV DL,AL
 MOV AH,2
 INT 21H
 LOOP _LOP2
 POP EDX
 POP CX
 POP EBX                                ;恢复现场
 RET
PRINTAX ENDP

;--------------------------------
;重复输出字符
;AL——需要重复输出的字符
;AH——需要重复输出的字符数
PRINTASCII PROC
 PUSH ECX
 MOVSX ECX,AH
_PUTCHAR_:
 OUT1 AL
```

```
 LOOP _PUTCHAR_
 POP ECX
 RET
PRINTASCII ENDP
;------------------------------清屏
_ClearS PROC
 PUSH AX
 MOV AH,00H
      MOV AL,03H
      INT 10H
 POP AX
 RET
_ClearS ENDP

;功能  1
;模块功能:查询商品信息
;模块名称:_QueryCommodity
;传入参数:无
;参数传入方式：无
;传出参数:无
;参数传出方式：无
;备注:使用了 DB 类型，10 个字节的缓冲区 in_name
_QueryCommodity PROC C GA1:WORD,GB1:WORD,N:WORD
 MOV AX, _DATA
 PUSHAD
_QC_START:
 WRITE MSG4
 READ in_goods
 CRLF
 MOV CX,N
 MOV BX,GA1
_QC_LOOP1:                              ;第一层循环，顺序查询 N 件商品
 MOV SI,0
 MOV AL,in_goods[1]
 CBW
 MOV DX,AX                          ;DX 控制第二层循环
 _QC_LOOP2:                         ;第二层循环，比较商品名称
       MOV AH,DS:[BX][SI]
       CMP AH,in_goods[SI+2]
       JNE _QC_UNFIND                   ;名称不同
       INC SI
       DEC DX
       JZ _QC_JUDGE                 ;找到商品,转 JUDGE
       JMP _QC_LOOP2

_QC_UNFIND:
 ADD BX,20
 DEC CX
 JNZ _QC_LOOP1
 JMP _QC_START                          ;未找到商品，重新输入商品名称
_QC_JUDGE:
 MOV AH,DS:[BX][SI]
 CMP AH,0
 JNE _QC_UNFIND                             ;商品名称为子集,属于未找到
 JMP _QC_FIND
_QC_FIND:
;显示商店 1 商品
 MOV AH,40
 MOV AL,'-'
 CALL PRINTASCII
 CRLF
 WRITE S1
```

```
OUT1 ':'
OUT1 ' '

MOVSX SI,BYTE PTR in_goods[1]
MOV AH,'$'
    MOV in_goods[SI+2],AH
WRITE in_goods[2]
CRLF

WRITE Purchase_price
OUT1 ':'
MOVSX EAX,WORD PTR DS:[BX+10]
CALL PRINTAX
CRLF

WRITE Selling_price
OUT1 ':'
MOVSX EAX,WORD PTR DS:[BX+12]
CALL PRINTAX
CRLF

WRITE Total_purchases
OUT1 ':'
MOVSX EAX,WORD PTR DS:[BX+14]
CALL PRINTAX
CRLF

WRITE Sold_quantity
OUT1 ':'
MOVSX EAX,WORD PTR DS:[BX+16]
CALL PRINTAX
CRLF

WRITE Profit_rate
OUT1 ':'
MOVSX EAX,WORD PTR DS:[BX+18]
CALL PRINTAX
CRLF


;显示商店 2 商品
MOV AH,40
MOV AL,'-'
CALL PRINTASCII
CRLF
WRITE S2
OUT1 ':'
OUT1 ' '
WRITE in_goods[2]
CRLF

WRITE Purchase_price
OUT1 ':'
SUB BX,GA1
ADD BX,GB1
MOVSX EAX,WORD PTR DS:[BX+10]
CALL PRINTAX
CRLF

WRITE Selling_price
OUT1 ':'
MOVSX EAX,WORD PTR DS:[BX+12]
CALL PRINTAX
CRLF
```

```
WRITE Total_purchases
OUT1 ':'
MOVSX EAX,WORD PTR DS:[BX+14]
CALL PRINTAX
CRLF

WRITE Sold_quantity
OUT1 ':'
MOVSX EAX,WORD PTR DS:[BX+16]
CALL PRINTAX
CRLF

WRITE Ranking
OUT1 ':'
MOVSX EAX,WORD PTR DS:[BX+18]
CALL PRINTAX
CRLF

POPAD
RET
_QueryCommodity ENDP
;功能 2
;模块功能:修改商品信息
;模块名称:_ModifyCommodity
;传入参数:无
;参数传入方式：无
;传出参数:无
;参数传出方式：无
;备注:无
_ModifyCommodity PROC C GA1:WORD,GB1:WORD,N:WORD
PUSHAD
_MC_START:
WRITE SelectShop
CRLF
MOV AH,01H
INT 21H
CMP AL,'1'                          ;选择商店 1
JE _MC_1
CMP AL,'2'
JE _MC_2                            ;选择商店 2
JMP _MC_START                       ;输入错误
_MC_1:
MOV BX,GA1
JMP _MC_3
_MC_2:
MOV BX,GB1
JMP _MC_3
_MC_3:
CRLF
WRITE MSG4
READ in_goods
CRLF
CMP in_goods[1],0
JE _MC_R                            ;输入回车

MOV CX,N                            ;查询商品
_MC_LOOP1:
MOV SI,0
MOV AL,in_goods[1]
CBW
MOV DX,AX                           ;DX 控制第二层循环
```

```
_MC_LOOP2:                          ;第二层循环，比较商品名称
        MOV AH,DS:[BX][SI]
        CMP AH,in_goods[SI+2]
        JNE _MC_UNFIND                      ;名称不同
        INC SI
        DEC DX
        JZ _MC_JUDGE                ;找到商品,转 JUDGE
        JMP _MC_LOOP2

_MC_UNFIND:
 ADD BX,20
 DEC CX
 JNZ _MC_LOOP1
 JMP _MC_START                              ;未找到商品，重新输入商品名称
_MC_JUDGE:
 MOV AH,DS:[BX][SI]
 CMP AH,0
 JNE _MC_UNFIND                                 ;商品名称为子集,属于未找到
 JMP _MC_FIND
_MC_FIND:

_MC_T1:
 CRLF
 WRITE Purchase_price
 OUT1 ':'
 MOVSX EAX,WORD PTR DS:[BX+10]
 CALL PRINTAX
 OUT1 '>'
 CALL INPUT_NEW_INFO
 CMP CX,-1
 JE _MC_T1
 CMP CX,-2
 JE _MC_T2
 MOV WORD PTR DS:[BX+10],CX
 CRLF
_MC_T2:
 CRLF
 WRITE Selling_price
 OUT1 ':'
 MOVSX EAX,WORD PTR DS:[BX+12]
 CALL PRINTAX
 OUT1 '>'
 CALL INPUT_NEW_INFO
 CMP CX,-1
 JE _MC_T2
 CMP CX,-2
 JE _MC_T3
 MOV WORD PTR DS:[BX+12],CX
 CRLF
_MC_T3:
 CRLF
 WRITE Total_purchases
 OUT1 ':'
 MOVSX EAX,WORD PTR DS:[BX+14]
 CALL PRINTAX
 OUT1 '>'
 CALL INPUT_NEW_INFO
 CMP CX,-1
 JE _MC_T3
 CMP CX,-2
 JE _MC_T4
 MOV WORD PTR DS:[BX+14],CX
 CRLF
```

```
_MC_T4:
 CRLF
 WRITE Sold_quantity
 OUT1 ':'
 MOVSX EAX,WORD PTR DS:[BX+16]
 CALL PRINTAX
 OUT1 '>'
 CALL INPUT_NEW_INFO
 CMP CX,-1
 JE _MC_T4
 CMP CX,-2
 JE _MC_R
 MOV WORD PTR DS:[BX+16],CX
 CRLF

_MC_R:
 POPAD
 RET
_ModifyCommodity ENDP
;--------------------------------
;从键盘输入 10 进制数，结果存在 CX 中
;传出参数：CX,输入的 10 进制数,如果输入错误则 CX=-1,如果输入回车 CX=-2
INPUT_NEW_INFO PROC
 PUSH AX
 PUSH BX
 MOV CX,0
 MOV AH,01H
 INT 21H
 CMP AL,0DH
 JE _INI_0DH                    ;只输入了回车
 JMP _INI_JUDGE
_INI_START:
 MOV AH,01H
 INT 21H
 CMP AL,0DH
 JE     _INI_R
_INI_JUDGE:
 CMP AL,'0'
 JL _INI_WRONG                  ;输入字符不合法
 CMP AL,'9'
 JG _INI_WRONG                  ;输入字符不合法
 SUB AL,'0'                     ;转换成数字
 MOVSX BX,AL
 IMUL CX,10
 ADD CX,BX
 JMP _INI_START
_INI_WRONG:
 MOV CX,-1
 JMP _INI_R
_INI_0DH:
 MOV CX,-2
 JMP _INI_R
_INI_R:
 POP BX
 POP AX
 RET
INPUT_NEW_INFO ENDP
;功能  3
;模块功能:计算平均利润率
;模块名称:_AVERAGR_PROFIT
;传入参数:无
;参数传入方式
;传出参数:无
```

```
;参数传出方式：无
;备注:无
_AVERAGR_PROFIT PROC C GA1:WORD,GB1:WORD,N:WORD
 PUSHA
 MOV BX, GA1
 MOV SI, N

LOOP_1:
 MOVSX EAX, WORD PTR [BX][12]   ;售价
 MOVSX EDX, WORD PTR [BX][16]   ;售出数
 IMUL EAX, EDX                  ;销售额

 MOVSX ECX, WORD PTR [BX][10]   ;进价
 MOVSX EDX, WORD PTR [BX][14]   ;进货量
 IMUL ECX, EDX                  ;成本

 SUB EAX, ECX
 IMUL EAX, 100
 CDQ
 IDIV ECX

 PUSH AX                        ;商店 1 商品利润率进栈

 SUB BX,GA1
 ADD BX,GB1
 MOVSX EAX, WORD PTR [BX][12]
 MOVSX EDX, WORD PTR [BX][16]
 IMUL EAX, EDX

 MOVSX ECX, WORD PTR [BX][10]
 MOVSX EDX, WORD PTR [BX][14]
 IMUL ECX, EDX

 SUB EAX, ECX
 IMUL EAX, 100
 CDQ
 IDIV ECX

 POP CX                         ;商店 2 商品利润率出栈
 ADD AX,CX

 SAR AX, 1                      ;计算平均利润率

 SUB BX,GB1
 ADD BX,GA1
 MOV WORD PTR [BX][18], AX

 ADD EBX, 20
 DEC SI

 JNZ LOOP_1
 POPA
 RET
_AVERAGR_PROFIT ENDP

;功能 4
;模块功能:计算利润率排名
;模块名称:_PROFIT_RANKING
;传入参数:无
;参数传入方式:无
;传出参数:无
;参数传出方式：无
```

```
;备注:无


_PROFIT_RANKING PROC C GA1:WORD,GB1:WORD,N:BYTE
 PUSHA          ;保护现场
 MOV AL, N
 MOV BX, GA1
 LOOP_2:
      MOV AH, N          ;内层循环计数
      MOV CX, 0          ;计数器，大于该利润的商品数量
      MOV DI, GA1

      LOOP_3:
           MOV DX, WORD PTR DS:[DI][18]
           CMP DX, WORD PTR [BX][18]

           JNG CONTINUE
           INC CX

      CONTINUE:
           ADD DI, 20

           DEC AH
           JNZ LOOP_3
      INC CX
      PUSH BX
      SUB BX,GA1
      ADD BX,GB1
      MOV WORD PTR [BX][18], CX
      POP BX
      ADD BX, 20

      DEC AL
      JNZ LOOP_2
 POPA
 RET

_PROFIT_RANKING ENDP
;功能 5
;模块功能:输出商品信息
;模块名称:_PRINT_IFO
;传入参数:0,1,(0 代表输出 shop1 信息，1 代表输出 shop2 信息）
;参数传入方式:al
;传出参数:无
;参数传出方式：无
;备注:;如果入口参数为 0，输出 shop1 信息（利润率），入口参数为 1， 输出 shop2 信息（排名）,al 为
传参寄存器

_PRINT_IFO PROC C GA1:WORD,GB1:WORD,N:WORD
 PUSHA
_PT_START:
 WRITE SelectShop
 CRLF
 MOV AH,01H
 INT 21H
 CMP AL,'1'                    ;选择商店 1
 JE CALL_1
 CMP AL,'2'
 JE CALL_2                     ;选择商店 2
 JMP _PT_START                 ;输入错误
CALL_1:
 ;输出店铺名称
```

```
WRITE S1
CRLF
;输出标题栏
WRITE Product_name
MOV AL, ' '
MOV AH, 2
CALL PRINTASCII

WRITE Purchase_price
MOV AL, ' '
MOV AH, 2
CALL PRINTASCII

WRITE Selling_price
MOV AL, ' '
MOV AH, 2
CALL PRINTASCII

WRITE Total_purchases
MOV AL, ' '
MOV AH, 2
CALL PRINTASCII

WRITE Sold_quantity
MOV AL, ' '
MOV AH, 2
CALL PRINTASCII

WRITE    Profit_rate
MOV AL, ' '
MOV AH, 2
CALL PRINTASCII
;输出标题栏的换行
CRLF

;循环输出 shop1 信息，循环条件为 bp
MOV CX, N
MOV BX, GA1
START__PRINT_IFO:
    ;首先输出商品名称
    CALL WRITE_TEN_CHAR
    MOV AL, ' '
    MOV AH, 3
    CALL PRINTASCII
    ;将 2 个数字有符号数字转换为 ascii，并输出
    MOVSX EAX, WORD PTR DS:[BX][10]
    CALL PRINTAX
    MOV AL, ' '
    MOV AH, 7
    CALL PRINTASCII

    MOVSX EAX, WORD PTR DS:[BX][12]
    CALL PRINTAX
    MOV AL, ' '
    MOV AH, 10
    CALL PRINTASCII

    MOVSX EAX, WORD PTR DS:[BX][14]
    CALL PRINTAX
    MOV AL, ' '
    MOV AH, 12
    CALL PRINTASCII
```

```
        MOVSX EAX, WORD PTR DS:[BX][16]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 12
        CALL PRINTASCII

        MOVSX EAX, WORD PTR DS:[BX][18]
        CALL PRINTAX


        ;输出空格
        CRLF
        ADD BX, 20
        DEC CX
        JNZ START__PRINT_IFO
 JMP EXIT_FUN
CALL_2:
 ;输出店铺名称
 WRITE S2
 CRLF
 ;输出标题栏

 WRITE Product_name
 MOV AL, ' '
 MOV AH, 2
 CALL PRINTASCII

 WRITE Purchase_price
 MOV AL, ' '
 MOV AH, 2
 CALL PRINTASCII

 WRITE Selling_price
 MOV AL, ' '
 MOV AH, 2
 CALL PRINTASCII

 WRITE Total_purchases
 MOV AL, ' '
 MOV AH, 2
 CALL PRINTASCII

 WRITE Sold_quantity
 MOV AL, ' '
 MOV AH, 2
 CALL PRINTASCII

 WRITE    Ranking
 MOV AL, ' '
 MOV AH, 2
 CALL PRINTASCII
 ;输出标题栏的换行
 CRLF

 ;循环输出 shop1 信息，循环条件为 bp
 MOV CX,  N
 MOV BX, GB1
 START__PRINT_IFO_:
        ;首先输出商品名称
        CALL WRITE_TEN_CHAR
        MOV AL, ' '
        MOV AH, 3
        CALL PRINTASCII
```

```
        ;将 2 个数字有符号数字转换为 ascii，并输出
        MOVSX EAX, WORD PTR DS:[BX][10]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 7
        CALL PRINTASCII

        MOVSX EAX, WORD PTR DS:[BX][12]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 10
        CALL PRINTASCII

        MOVSX EAX, WORD PTR DS:[BX][14]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 12
        CALL PRINTASCII

        MOVSX EAX, WORD PTR DS:[BX][16]
        CALL PRINTAX
        MOV AL, ' '
        MOV AH, 12
        CALL PRINTASCII

        MOVSX EAX, WORD PTR DS:[BX][18]
        CALL PRINTAX


        ;输出空格
        CRLF
        ADD BX, 20
        DEC CX
        JNZ START__PRINT_IFO_

 JMP EXIT_FUN
EXIT_FUN:
 POPA
 RET
_PRINT_IFO ENDP

;函数 write_ten_char 用于输出连续的 10 个字符，bx 为起始字符偏移地址
;调用方法
;MOV BX, OFFSET GA1
;LEA BX, GA1
;（GA1 为 ds 段的变量名）
WRITE_TEN_CHAR PROC
 ;现场保护
 PUSH BX
 PUSH DX
 PUSH AX
 ;初始化循环条件
 MOV DH, 10
 ;开始循环
 START_PRINT:
        ;调用系统功能输出
        MOV DL, DS:[BX]
        MOV AH, 2
        INT 21H

        INC BX
        DEC DH
```

```
        JNZ START_PRINT
 POP AX
 POP DX
 POP BX
 RET
WRITE_TEN_CHAR ENDP
_TEXT ENDS
 END
```

3.2.4 实验步骤

1.准备上机环境，编辑汇编文件和 C 文件

2.再 BC31 中创建新的工程 3.PRJ，将汇编文件和 C 文件加入到工程 3.PRJ 中

3.编译工程，生成 3.EXE 文件

4.运行 3.EXE，进行功能测试

5.使用 BC31 中的 TD 进行调试

6.观察调用汇编子程序时的反汇编代码。

7. 在 C 语言源程序中不合理地嵌入汇编语言的指令语句,观察结果

3.2.5 实验记录与分析

实验环境条件：WINDOWS 10 下 BC3.1

1.以游客身份登录后的界面显示如图 3.1.6 所示，只有功能 1 和功能 6 可用。

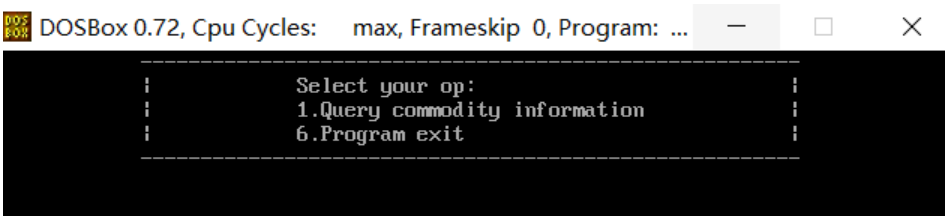
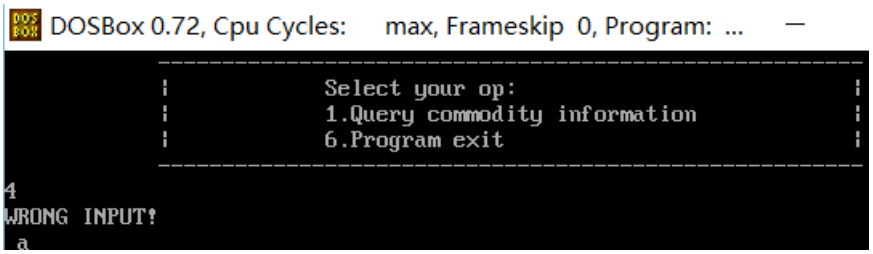


图 3.2.2 游客界面

当输入不合法的字符时，系统会让用户重新输入，直到输入了正确的字符，演示过程如图 3.1.7 所示。



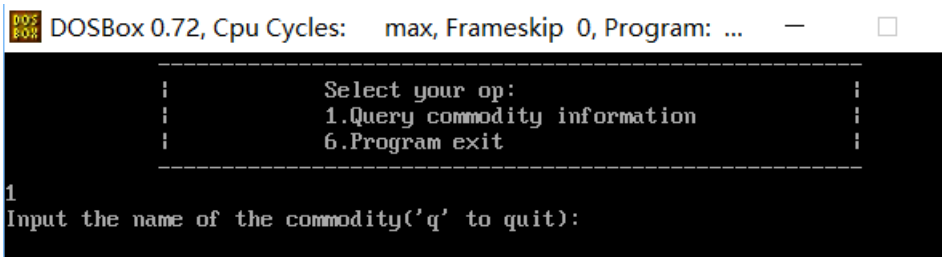


图 3.2.3 选项输入

只有在输入了正确的商品名后，系统会显示商店 1 和商店 2 的商品信息，结果如图 3.1.8 所示。

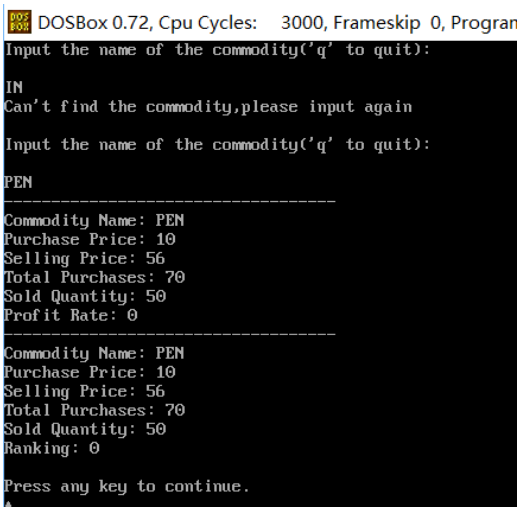


图 3.2.4 查询商品信息

然后按任意键回到菜单界面，输入 6 后退出系统，演示过程如图 3.1.9 所示

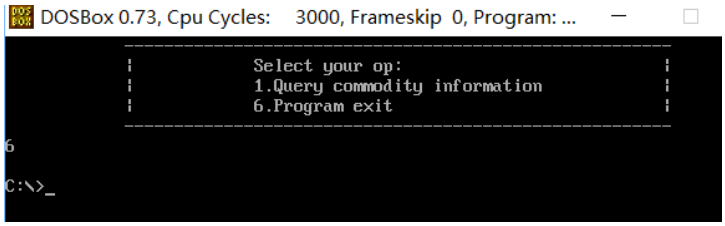


图 3.2.5 退出系统

2.以账号密码登录后的菜单如图 3.1.10 所示，有功能 1-6 可选，其中功能 1、6 同未登录状态的功能 1、6.

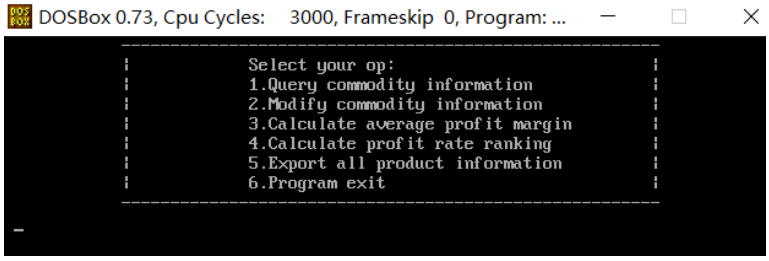
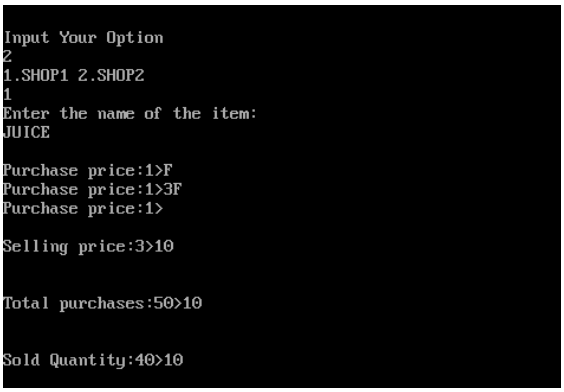


图 3.2.6 登录状态的系统界面

输入 2 后进入功能 2，选择商店和商品后进行信息修改，输入回车不修改，输入非法字符系统会让用户重新输入，结果如图 3.1.11 所示。



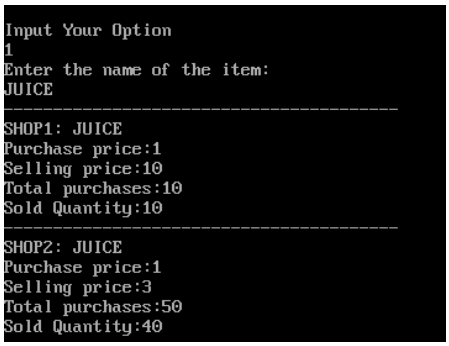
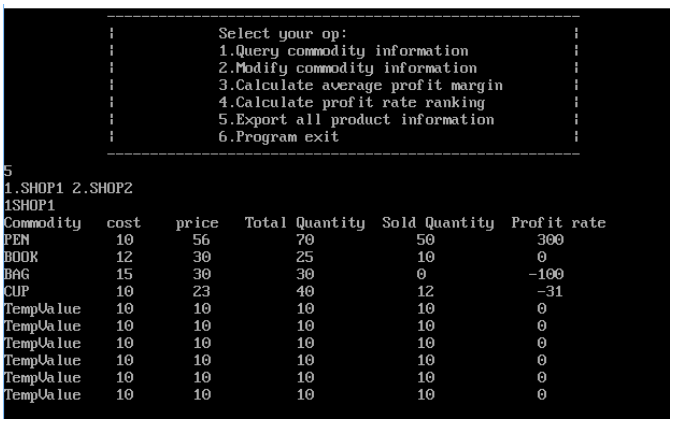


图 3.2.7 商品信息修改

运行功能 3、4 后使用功能 5 输出所有商品信息，结果如图 3.1.12 所示



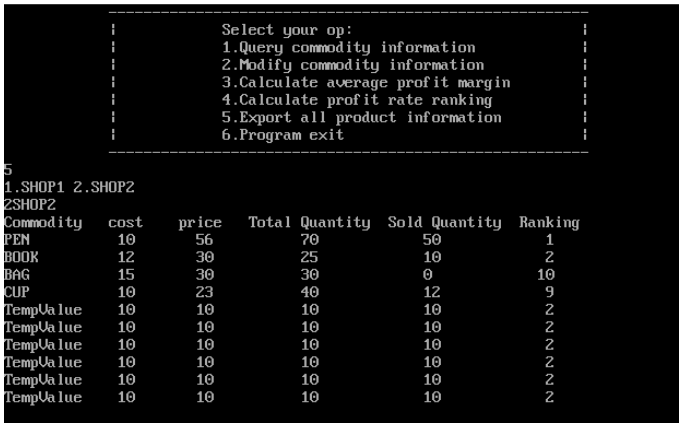


图 3.2.8 所有商品信息

3.2.6 思考题

1.请尝试在 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。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（或 DS 等其他寄存器）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

再调用功能函数时加上 asm mov ax,100 asm mov ds,ax mov cs,ax 三句句汇编指令来修改 ds 和 cs 的值,运行结果如图 3.2.9 所示：

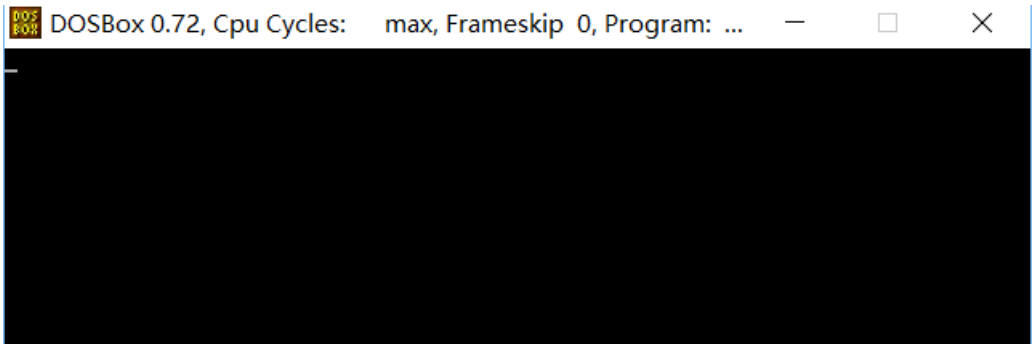
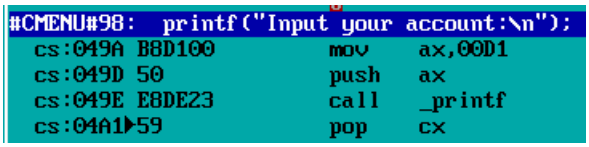


图 3.2.9 被破坏的程序执行情况

通过 TD 调试发现程序进入了一个死循环，可见在程序运行过程中修改寄存器的值会破坏程序。

2. 观察 C 编译器的优化策略对代码的影响。通过实际观察与分析，记录本实验中汇编语言程序的效率会优于 C 语言程序的实例（至少给出一处的观察结果）。

在输出字符串时，C 函数 printf 实现该功能需要接近 20 条汇编指令，而在汇编程序中只需行指令的 9 号调用就能实现相同的功能。



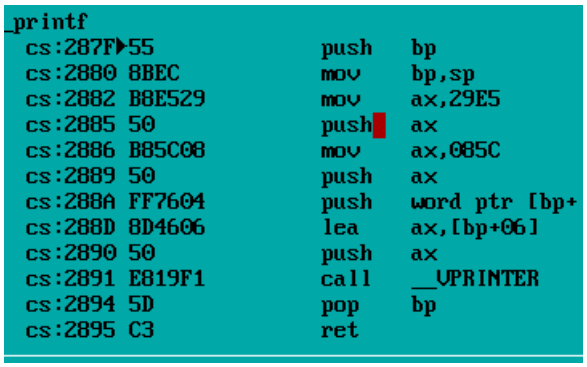


图 3.2.10 printf 的反汇编指令

4. 总结与体会

通过任务 1，我初步了解了模块化的设计思想，在面临一个大型工程时，采用模块化设计可以极大的提高效率，同样也方便测试，可以极大的提高我们的编程效率。这次实验中，我和我的同伴分别负责不同的功能，然后将生成的 obj 文件交给对方，使用 link 程序链接生成 exe 文件，在这过程中，由于是第一次进行模块化设计，因此遇到了许多模块通信的问题，比如如何生命 PUBLIC 变量，如何用 EXTRN 引用另一模块的变量，通过阅读课本以及请教老师，我们成功实现了程序链接。在测试功能时，却出现了生成的 EXE 文件在我的电脑运行正常，而在我的同伴的电脑上运行不正常的问题，通过 TD 调试发现，在使用 INT 21H 指令时，AL 的值在我同伴的电脑上被改变了，然后我在调用 21H 时通过入栈保护 AX 的值成功解决了问题。

任务 2 拓宽了我的视野，让我学会了 C 和汇编的混合编程，让我懂得了高级语言和低级语言的关系。更重要的时，这次实验让我看到了代码优化的方法，高级语言虽然便于编程，但它同样是臃肿的，往往一个简单的功能却需要许多行汇编指令才能实现，而利用汇编编程可以极大地提高效率，因此在某个问题算法方面无法优化时利用汇编进行优化也不失为一种方法。

本次实验中最大的一个问题就是 C 和汇编地连接问题。首先是 32 位寄存器的使用，由于 BC31 是默认 16 位段的程序，而我的程序中使用了 GA1[EBX]这样的寻址方式，因此在链接时报了 32 位寄存器冲突的问题，为了解决这个问题，我将汇编语句全部注释了，然后逐句取消注释观察是否报错，最终找到了错误原因，于是我就使用 16 位寄存器进行寻址，如 GA1[BX]，解决了问题。实验中遇到的另一个问题是段类别，一开始我的汇编程序和 C 程序的段类别不同，因此在 NEAR 方式调用子函数和传递变量是就出现了错误，通过调试发现，当汇编程序的 DATA 段位’DATA’类别，CODE 段为’CODE’类别时汇编程序和 C 程序有相同的段首址，即它们的 DS,CS 的值是一样的。

5. 参考文献

[1] 王元珍,曹忠升,韩宗芬.80X6 汇编语言程序设计.版本(第 1 版).武汉.华中科技大学出版社,2005 年 4 月.