

**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 实验地点： 南一楼804室90号实验台**

**2018-4-23，14：00-17：30**

**指导教师： 李海波**

**专业班级：校交 201601班**

**学 号： U201612696 姓 名： 陈淏睿**

**同组学生： 景常进 报告日期： 2017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：2018.4.24

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](#_Toc515195831)

[2 实验内容 1](#_Toc515195832)

[3 实验过程 3](#_Toc515195833)

[3.1 任务1 3](#_Toc515195834)

[3.1.1 设计思想及存储单元分配 3](#_Toc515195835)

[3.1.2 实验步骤 4](#_Toc515195836)

[3.1.3 流程图 4](#_Toc515195837)

[3.1.4 源程序 5](#_Toc515195838)

[3.1.5 实验记录与分析 12](#_Toc515195839)

[3.2 任务2 16](#_Toc515195840)

[3.2.1 设计思想及变量说明 16](#_Toc515195841)

[3.2.2 源程序 16](#_Toc515195842)

[3.2.3 实验步骤 23](#_Toc515195843)

[3.2.4 实验记录与分析 23](#_Toc515195844)

[4 总结与体会 25](#_Toc515195845)

[参考文献 27](#_Toc515195846)

# 实验目的与要求

本次实验的主要目的与要求有下面6点，所有的任务都会围绕这6点进行，希望大家事后检查自己是否达到这些目的与要求。

1. 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. 熟悉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 （尽量在第一次4个课内学时阶段完成，上机前实验报告应完成该任务中除实验记录与分析、总结与体会之外的内容）

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

（主要在第二次4个课内学时阶段完成，上机前实验报告应完成该任务中除实验记录与分析、总结与体会之外的内容）

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。

**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（建议用BC31，其功能与操作方法相对比较简单）。

（2）在实验报告中，比较详细的给出你的开发环境及其实现方法。

（3）观察C语言编译器中对各种符号的命名规则（指编译器内部可以识别的命名规则，比如，符号名前面是否加下划线“\_”，等），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（要设计一个有多个参数需要传递的C函数）。

（4）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。

（5）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（该项实验表明：在C语言程序中，若不考虑上下语句翻译成怎样的机器码而随意嵌入汇编指令语句时，有可能存在出错的风险）。

（6）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）。

（7）通过调试混合编程的程序，体会与纯粹汇编语言编写的程序的调试过程的差异。

（8）通过本次实验，希望大家明白：不同的编程语言是可以协同解决一个问题的，而且可以利用不同语言的特点来更好地解决问题；利用汇编语言的知识，能够更好地理解高级语言的内部处理原理与策略，为编写更好的C语言程序、用好C编译器提供支持。

# 实验过程

## 任务1

### 设计思想及存储单元分配

功能3的实现方式同实验二功能三（2），此处不再赘述。

在功能4中，首先按照有效商品的个数初始化临时数组相同个元素的序号，对商店一中的每个商品，在商店二中找到相同商品，将商店一中利润的EA分别存入临时数组与商店二中商品的对应位置。根据临时数组个元素的EA对应的元素的大小按降序对数组中元素进行排序，之后若有前后两元素指向的数值相同，则也更改其序号为相同的。对商店二中每个商品，按其对应位置保存的EA在临时数组中找到有相同EA的元素，将该元素的序号放进之前存EA的位置，返回。

在功能五中，以商店一中每个元素为准，首先在商店二中找到相同的商品，逐字输出商品名称。调用进制转换子程序MRADIX将商店一的商品中保存的平均利润逐字转换为10进制数的ASCII码并输出，最后将其排名转换为对应的ASCII码，输出。

1.存储单元分配

TIP\_12：长度较小，以$结尾的串，用于在功能5中输出不同类数据时做间隔。

M：存放商店中有效商品的个数，可依具体情况进行修改。

TAD：在功能4中使用的，长90字节的临时数组，其中3字节作为一个元素，每个元素中第一个字节存放当前元素的序号（可能不被初始化），后两个字节存放16位的偏移地址。

HKU：存放数100，在功能3做乘法时使用。

2.寄存器分配：3个功能合起来使用了所有寄存器。具体使用方式详见源程序中每个子程序前的注释部分。

### 实验步骤

1. 准备上机实验环境,将实验二任务二的拷贝重命名t1-,在此基础上修改程序。

2. 按照3.1.1的思路写功能3-功能5。

3. 反复进行汇编-修改过程，直至汇编不再报错。

4. 进入td单步调试，检查程序中的逻辑错误并修改。

5. 提取t1-中的功能3至功能5形成新文件，与同组同学提供的目录与剩下两个功能进行适配，修改段名，添加函数与变量属性。

6. 反复进行汇编-修改过程，直至汇编不再报错。

7. 进入td单步调试，检查程序中的逻辑错误并修改。

### 流程图

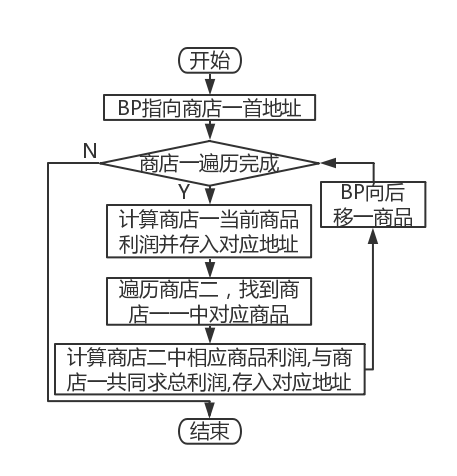
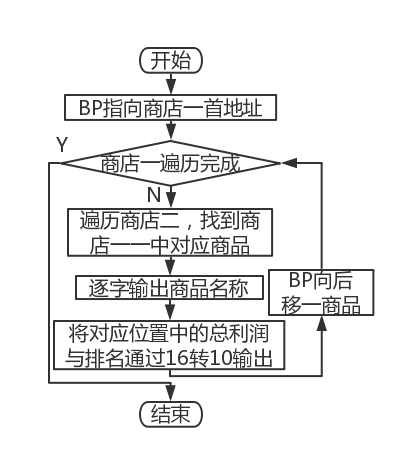
 

图3.1.1任务3流程图 图3.1.2任务5流程图

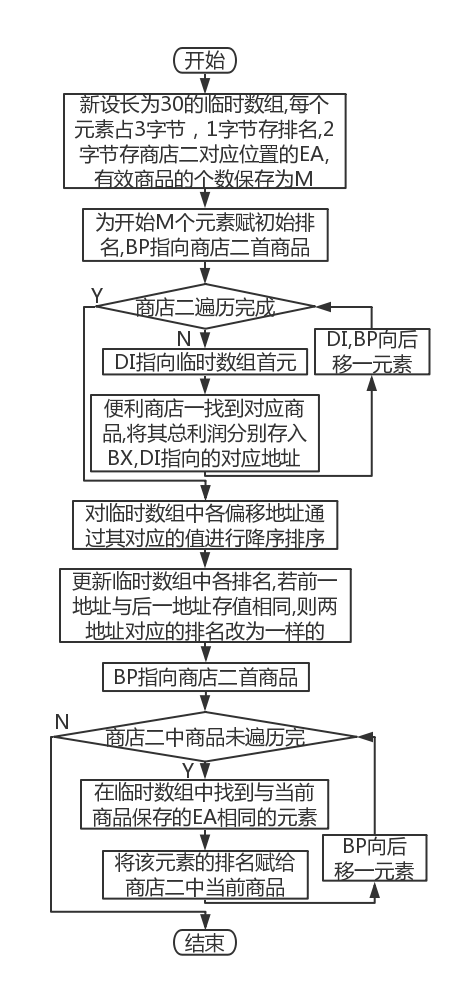


图3.1.3任务4流程图

### 源程序

T1-.ASM:

.386

INTR MACRO T ;参数为寄存器名的一部分

MOV SI,DS:[B&T&+10] ;进货价

IMUL SI,DS:[B&T&+14] ;进货数量

MOV AX,DS:[B&T&+12] ;销售价

IMUL AX,DS:[B&T&+16] ;已售数量

SUB AX,SI

IMUL HKU

IDIV SI

ENDM

FND MACRO COUT,PIN,A,B,N ;参数为成功时的返回标号

MOV SI,0

LP&A&: ;获得商店一中当前商品串长，存入AX(AL)

CMP BYTE PTR DS:[BP+SI],0

JE BRK&N&

INC SI

JMP LP&A&

BRK&N&:

MOV AX,SI

MOV CH,M ;CH负责商店二的遍历

LEA BX,PIN

LP&B&:

CALL CMPST ;比较两商品串是否相同,不同时AH为0（默认值）

OR AH,AH

JNZ COUT

ADD BX,20 ;BX移向商店二中下一商品，继续比较

DEC CH

JNZ LP&B&

JMP STEP1 ;商店二中商品已遍历完毕，仍未找到

ENDM

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

STACK SEGMENT STACK USE16

┇ ；此行之后为实验二相关数据定义

┇ ；此行之前为实验二相关数据定义

AUTH DB ? ;判断登录方式

TAD DB 90 DUP(0) ;存放排名与偏移地址的临时数组

TNM DB 3 DUP(0) ;存放十进制利润的临时数组

HKU DW 100

┇ ；此行之前为实验二相关数据定义

┇ ；此行之前为实验一任务五功能二相关代码

STEP3:

LEA DX,TIP\_13 ;菜单

MOV AH,9

INT 21H

MOV AH,1 ;输选项

INT 21H

LEA DX,TIP\_9

MOV AH,9

INT 21H

CMP AL,'3'

JE STEP3\_3

CMP AL,'4'

JE STEP3\_4

CMP AL,'5'

JE STEP3\_5

JMP STEP4

MOV CL,M ;外循环计数器

LEA BP,GA1

LPB:

LEA BX,IN\_GOOD

INC BX

MOV DX,[BX]

MOV DH,0

LPBA: ;比较商店一中当前字符串与输入串

LEA BX,IN\_GOOD

INC BX

MOV AX,[BX]

MOV AH,0

SUB AX,DX

MOV SI,AX

MOV BX,DS:[BP+SI]

MOV BH,0

MOV DI,BX

LEA BX,IN\_GOOD

ADD BX,2

MOV AX,SI

XLAT

CMP DI, AX

JNE BRK1 ;当前字符不同时直接退出循环

DEC DX

JNE LPBA ;SI非0时继续下一重循环

BRK1:

CMP DX,0 ;SI不为0说明对输入字符串的遍历未到头，即字符串不同，应继续比较下一字符串；为0则跳出外循环

JNE BRK13

INC SI

MOV BX,DS:[BP+SI]

CMP BX,0

JE BRK2

BRK13:

ADD BP,20 ;BP移向商店一中下一货物

DEC CL

JNE LPB ;商品未查找完

BRK2:

CMP CL,0 ;CL不为0说明已找到商品

JNE BRK3

LEA DX,TIP\_4 ;显示提示:重新输入待查商品名称

MOV AH,9

INT 21H

JMP STEP3

BRK3:

CMP AUTH,1

JNE BRK6

MOV CX,DS:[BP+16] ;已售数量（用于比较）

CMP CX,DS:[BP+14] ;进货数量

JB BRK15 ;已售数量低于进货数量

LEA DX,TIP\_10 ;显示提示:已售数量高于进货数量

MOV AH,9

INT 21H

JMP STEP1

BRK15:

INC WORD PTR DS:[BP+16] ;已售数量加一

STEP3\_3:

CALL PART3

JMP STEP3

STEP3\_4:

CALL PART4

JMP STEP3

STEP3\_5:

CALL PART5

JMP STEP3

┇ ；此行之后为实验一任务五功能三相关代码

┇ ；此行之前为实验二CMPST相关代码

;子程序名：PART3

;功能：计算两商店中每种商品的总利润率

;入口参数：无

;出口参数：无

;所使用的寄存器：

;CL——外循环计数器

;BP——指向商店一中商品的指针

PART3 PROC USES AX CX SI BP

MOV CL,M ;CL负责商店一的遍历

LEA BP,GA1

LPC: ;之前所有寄存器的值可作废

FND BRK20,GB1,N,O,21 ;循环比较BP与BX指向的串，相同时转到BRK20

BRK20: ;更新各商品均利润率（CL,BX,BP被占用）

INTR P ;计算BP所指向的商品的利润率

MOV DS:[BP+18],AX ;保存BP利润

INTR X ;计算BX所指向的商品的利润率

ADD AX,DS:[BP+18] ;两利润相加

CWD

SAR AX,1 ;求平均利润

MOV DS:[BP+18],AX ;平均利润存入SHOP1的利润字段

DEC CL

OR CL,CL

JZ BAR5

ADD BP,20 ;BP移向商店一中下一商品

JMP LPC

BAR5:

RET

PART3 ENDP

;子程序名：PART4

;功能：对两商店中的商品按总利润率排序，放在商店二中对应商品的相应EA中

;入口参数：无

;出口参数：无

;所使用的寄存器：

;BX——指向商店一中商品的指针

;BP——指向商店二中商品的指针

;CL——外循环计数器

;CH——内循环计数器

;DI——指向临时数组的指针/做变址寻址寄存器

;SI——做变址寻址寄存器

PART4 PROC USES AX BX CX DX BP SI DI

MOV BX,OFFSET TAD

MOV CL,M ;CL负责LPD中商店二的遍历

XOR CH,CH

LPF: ;初始化临时数组序号

INC CH

MOV [BX],CH

ADD BX,3 ;BX移向下一结构

CMP CH,CL ;按有效商品种数初始化标号

JNE LPF

LEA BP,GB1

LEA DI,[TAD+1]

LPP: ;找到商店一中对应商品，将总利润地址存入SHOP2和临时数组

FND BRK4,GA1,L,M,17 ;循环比较BP与BX指向的串，相同时转到BRK4

BRK4:

ADD BX,18

MOV DS:[BP+18],BX ;将商品总利润在商店一中的偏移地址存入商店二

MOV [DI],BX ;将商品总利润在商店一中的偏移地址存入临时数组

ADD DI,3 ;DI移向临时数组中下一元素

DEC CL

OR CL,CL

JZ BRK5

ADD BP,20

JMP LPP

BRK5: ;临时数组中元素排序

LEA BX,[TAD+1] ;取数组的首地址

MOV CL,M ;控制循环次数

XOR CH,CH

XOR SI, SI ;将SI清零

MOV DI,3

MOV AL,M ;偏移上限

XOR AH,AH

IMUL AX,3

L1: MOV BP, [BX+SI] ;用基变址寻址取操作数，L1为外循环，(SI)为循环变量，;相当于i

L2: CMP BP, [BX+DI] ;L2为内循环，(DI)为循环变量，相当于j

JGE L3

XCHG BP, [BX+DI] ;[BX+SI]<[BX+DI]，交换

MOV [BX+SI], BP ;BP新值送回[BX+SI]

L3: ADD DI,3 ;AH>=AL，不需交换，(AH)直接和后一个数比较，相当于j++

CMP DI,AX ;判断内层循环是否结束

JB L2

ADD SI,3 ;外层变量SI加3

MOV DI,SI ;相当于j=i

ADD DI,3

LOOP L1 ;通过寄存器实现两个存储器数据间的交换

LEA BX,TAD ;利润相等时修改排名数值

MOV SI,0

LPG:

MOV AX,[BX+SI]

MOV CX,[BX+SI+3]

MOV DX,[EAX]

CMP DX,[ECX] ;比较前后两利润率大小关系

JNE BRK11

MOV CL,[BX+SI]

MOV [BX+SI+3],CL ;序号变成一样的

BRK11:

ADD SI,3

MOV DL,M

XOR DH,DH

DEC DX

IMUL DX,3

CMP SI,DX

JB LPG

LEA BP,GB1 ;比对商店二与临时数组中相应位置保存的地址，若相同则将临时数组中对应序号赋给商店二中对应商品

XOR SI,SI ;SI置0

XOR CH,CH ;外循环计数器

LPH:

XOR DI,DI

XOR CL,CL ;内循环计数器

LPH\_1:

MOV AX,[BX+DI+1]

CMP AX,DS:[BP+SI+18]

JNE BRK12

MOV AL,[BX+DI]

XOR AH,AH

MOV DS:[BP+SI+18],AX

JMP BRK14

BRK12:

ADD DI,3 ;DI指向临时数组中下一地址

INC CL

MOV AL,M

CMP CL,AL

JNE LPH\_1 ;当前商品排名未产生

MOV WORD PTR DS:[BP+SI+18],0 ;保险语句，万一在临时数组中未找到与目标商品相同的地址则将其排名置0

BRK14:

ADD SI,20 ;SI指向商店二中下一商品

INC CH

MOV AL,M

CMP CH,AL

JNE LPH ;商店二未遍历完成

RET

PART4 ENDP

;子程序名：PART5

;功能：输出各商品的名称，平均利润与排名

;入口参数：无

;出口参数：无

;所使用的寄存器：

;BX——指向商店二中商品的指针

;BP——指向商店一中商品的指针

;CL——外循环计数器

;CH——内循环计数器

;DI——指向临时数组的指针/做变址寻址寄存器

;SI——做变址寻址寄存器

;DL——输出字符用

;AX——取得平均利润

PART5 PROC USES AX BX CX DX SI

MOV CL,M ;CL负责商店一的遍历

LEA BP,GA1

LPI: ;之前所有寄存器的值可作废

FND BRK16,GB1,J,K,18 ;循环比较BP与BX指向的串，相同时转到BRK16

BRK16:

XOR SI,SI

MOV CH,AL

LPI\_A: ;循环输出商品串

MOV DL,[BX+SI]

MOV AH,2

INT 21H

INC SI

DEC CH ;商品串未输出完毕

JNZ LPI\_A

LEA DX,TIP\_12 ;制表

MOV AH,9

INT 21H

MOV AX,DS:[BP+18] ;获取平均利润

MOV DX,AX

AND DX,8000H

CMP DX,0

PUSH BX

MOV BX,10 ;目标进制

JZ BRK22

DEC AX ;从补码返回源码（脱符号）

NOT AX

PUSH AX

LEA SI,TNM ;SI指向临时数组

MOV DL,'-' ;输出负号

MOV AH,2

INT 21H

POP AX

BRK22:

CALL MRADIX ;转换利润为十进制并输出ASCII字符

LEA DX,TIP\_12 ;制表

MOV AH,9

INT 21H

POP BX

MOV DL,[BX+18] ;提取排名

ADD DL,30H

MOV AH,2

INT 21H

LEA DX,TIP\_9 ;换行

MOV AH,9

INT 21H

DEC CL

JZ BAR6 ;商品未遍历完

ADD BP,20

JMP LPI

BAR6:

RET

PART5 ENDP

;子程序名：MRADIX

;功能：将AX中的16位无符号二进制数转换为P进制数（16位段）

;入口参数：

;AX——存放待转换的16位无符号二进制数

;BX——存放要转换进制的基数

;出口参数：无

;所使用的寄存器：

;CX——P进制数字入栈出栈时的计数器

;DX——做除法时存放被除数高位或余数

MRADIX PROC USES CX DX

XOR CX,CX

LPQ:

XOR DX,DX

DIV BX

PUSH DX

INC CX ;保存位数

OR AX,AX

JNZ LPQ

LPR:

POP AX

CMP AL,10

JB BRK23

ADD AL,7

BRK23:

ADD AL,30H

MOV DL,AL ;直接输出当前字符

MOV AH,2

INT 21H

LOOP LPR

RET

MRADIX ENDP

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

CODE ENDS

END START

Wan1.asm

NAME WAN1

EXTERN CRLF:BYTE,GA1:BYTE,GB1:BYTE

PUBLIC PART3,PART4,PART5,SET

.386

WRITE MACRO OUT\_BUF;宏定义代替9号调用

LEA DX,OUT\_BUF

MOV AH,9

INT 21H

ENDM

┇ ；此行之后为t1-相关宏定义

┇ ；此行之前为t1-相关宏定义

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

DATA1 SEGMENT USE16 PARA PUBLIC 'D1'

TIP\_12 DB ' $'

M DB 3 ;有效商品个数

TAD DB 90 DUP(0) ;存放排名与偏移地址的临时数组

HKU DW 100

TNM DB 3 DUP(0) ;存放十进制利润的临时数组

DATA1 ENDS

STACK SEGMENT USE16 PARA STACK 'STACK'

DB 200 DUP(0)

STACK ENDS

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

CODE SEGMENT USE16 PARA PUBLIC 'CODE'

ASSUME CS:CODE,ES:DATA1,SS:STACK

;子程序名：SET

;功能：将数据段DATA1装入ES

;入口参数：无

;出口参数：无

;所使用的寄存器：

;AX——作为中转寄存器

SET PROC

MOV AX,DATA1

MOV ES,AX

RET

SET ENDP

┇ ；此行之后为t1-各子程序

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU 2.60GHz，2.86G内存；WINDOWS 7下DOSBox0.74； notepad++ 7.55;MASM.EXE 6.0;LINK.EXE 5.2;TD.EXE 5.0。

2. 按3.1.1的思路编写程序，汇编，连接，过程中未出现问题，如图3.1.4所示。

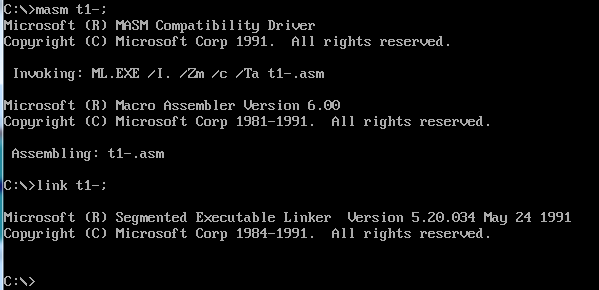


图3.1.4汇编连接过程正常

3.先后运行功能3,4,5运行进行检查，运行结果如图3.1.5所示。第一列为商品名称，第二列为总利润（未输出百分号），第三列为排名。总利润与排名与针对各商品利润手动计算的结果一致，可见单独编译运行时三个功能运行正常。需要注意的是最好按照功能3-功能4-功能5的顺序运行，否则存储器中未初始化的数据会得出错误结果。

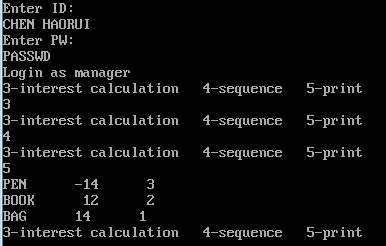


图3.1.5单独运行时的结果

4.将t1-.asm中的功能三相关各部分抽出，写成文件wan1.asm,同时加上段落名以及对PUBLIC和EXTERN的说明，同时在menu.asm中进行功能3-功能5的函数调用，补充由wan1.asm新增的PUBLIC和EXTERN说明。汇编，连接，期间未发生异常情况，如图3.1.6和3.1.7所示。



图3.1.6汇编menu与wan1

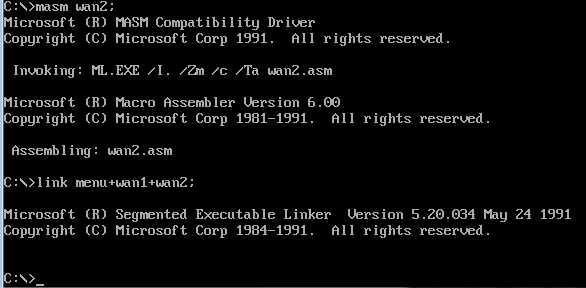


图3.1.7汇编menu与wan1

5.功能1,2已由另一位同学测试通过，因此直接测试功能3-功能5。测试功能3，4时，发现程序卡死，如图3.1.8和3.1.9所示；测试功能5时，发现输出乱码，如图3.1.10所示。经td单步调试检查后，发现wan1.asm中的数据段并未被装载至DS中，导致程序卡死/崩溃。采用用menu中主程序调用wan1中子程序的方式将wan1中数据段装入DS，同时修改相关PUBLIC和EXTERN声明。

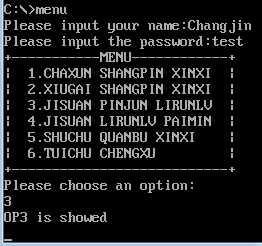
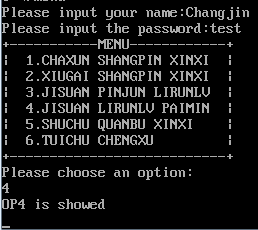
 

图3.1.8功能3卡死 图3.1.9功能4卡死



图3.1.10功能5卷屏

6. 重新汇编连接，再次执行功能五时，发现图3.1.11所示错误，猜想功能四利润排序出错。检查后发现首先在排序时排序对象是商店一中保存利润的地址而非保存的利润，即应进行二次取址；其次在进行比较时CL应赋M-1而非M。

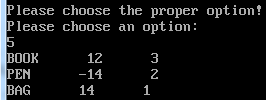


图3.1.11连接时报错

7.再次编译连接后输出结果如图3.1.12所示，与图3.1.5中的对应关系相同，证明结果正确。

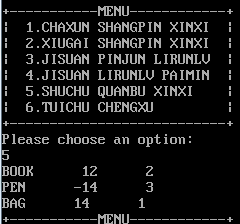


图3.1.12结果正确

## 任务2

### 设计思想及变量说明

将菜单界面及功能一，二，五用C语言改写，对相关变量及函数添加extern和public的说明；参照样例为汇编语言部分使用的变量及函数统一加上下划线前缀并在每个函数中添加相应的数据段装载语句。

在C语言中定义商品的数据结构，将汇编语言中定义的各商品变量在C语言中用结构改写。

1. 存储单元分配

将商品写成结构，两个商店中所有商品构成两个长为30，各有3个元素被初始化的结构数组：

typedef struct GOOD {

char name[10]; //名称

short ipr, opr, inm, onm, //进货价，销售价，进货数量，已售数量

int itr; //（存储利润的位置）

}GOOD

GOOD shop1[30] = { { "PEN",35,56,70,25 },{ "BOOK",12,30,25,5 },{ "BAG",25,35,20,12 } };

GOOD shop2[30] = { { "BOOK",12,28,20,15 },{ "PEN",35,50,30,24 },{ "BAG",20,35,42,35 } };

char tad[150]:存储序号及32位地址的临时数组

char id[10] = "CHR", pw[6] = "test", cbuf[10]：分别存储姓名，密码及输入串

char jhj[] = "进货价", xsj[] = "销售价", jhzs[] = "进货总数"：作为参数便于在函数中调用

2.寄存器分配

详见汇编语言部分中各函数的注释。

### 源程序

Forvs.asm

INTR MACRO T ;参数为寄存器名的一部分

MOV SI,DS:[ED&T&+10] ;进货价

IMUL SI,DS:[ED&T&+14] ;进货数量

MOV AX,DS:[ED&T&+12] ;销售价

IMUL AX,DS:[ED&T&+16] ;已售数量

SUB AX,SI

MOV BX,100

PUSH EDX

IMUL BX

IDIV SI

POP EDX

ENDM

FND MACRO COUT,PIN,A,B,N ;参数为成功时的返回标号

MOV ESI,0

LP&A&: ;获得商店一中当前商品串长，存入AX(AL)

CMP BYTE PTR DS:[EDX+ESI],0

JE BRK&N&

INC SI

JMP LP&A&

BRK&N&:

MOV AX,SI

MOV CH,M ;CH负责商店二的遍历

MOV EDI,PIN

LP&B&:

CALL CMPST ;比较两商品串是否相同,不同时AH为0（默认值）

OR AH,AH

JNZ COUT

ADD DI,24 ;DI移向商店二中下一商品，继续比较

DEC CH

JNZ LP&B&

ENDM

.386

.model flat, c

.code

public ITRST,QUEUE

;子程序名：ITRST

;功能：计算两商店中每种商品的总利润率

;入口参数：无

;出口参数：无

;所使用的寄存器：

;CL——外循环计数器

;EDX——指向商店一中商品的指针

ITRST PROC M:BYTE,shop1:DWORD,shop2:DWORD

MOV CL,M ;CL负责商店一的遍历

MOV EDX,shop1

LPC: ;之前所有寄存器的值可作废

FND BRK20,shop2,N,O,21 ;循环比较DX与DI指向的串，相同时转到BRK20

BRK20: ;更新各商品均利润率（CL,BX,BP被占用）

INTR X ;计算DX所指向的商品的利润率

MOV DS:[EDX+20],AX ;保存DX利润

INTR I ;计算DI所指向的商品的利润率

ADD AX,DS:[EDX+20] ;两利润相加

SAR AX,1 ;求平均利润

MOV DS:[EDX+20],AX ;平均利润存入SHOP1的利润字段

DEC CL

OR CL,CL

JZ BAR5

ADD EDX,24 ;BP移向商店一中下一商品

JMP LPC

BAR5:

RET

ITRST ENDP

;子程序名：QUEUE

;功能：对两商店中的商品按总利润率排序，放在商店二中对应商品的相应EA中

;入口参数：无

;出口参数：无

;所使用的寄存器：

;BX——

;BP——指向商店二中商品的指针

;CL——外循环计数器

;CH——内循环计数器

;DI——指向临时数组的指针

;BP——指向商店一中商品的指针

;CL——外循环计数器

;CH——内循环计数器

QUEUE PROC M:BYTE,shop1:DWORD,shop2:DWORD,TAD:DWORD

MOV EBX,TAD

MOV CL,M ;CL负责LPD中商店二的遍历

XOR CH,CH

LPF: ;初始化临时数组序号

INC CH

MOV [EBX],CH

ADD EBX,5 ;BX移向下一结构

CMP CH,CL ;按有效商品种数初始化标号

JNE LPF

MOV EDX,shop2

MOV EBX,TAD

INC EBX

LPP: ;找到商店一中对应商品，将总利润地址存入SHOP2和临时数组

PUSH EBX

FND BRK4,shop1,L,M,17 ;循环比较BP与BX指向的串，相同时转到BRK4

BRK4:

POP EBX

ADD EDI,20

MOV DS:[EDX+18],EDI ;将商品总利润在商店一中的偏移地址存入商店二

MOV [EBX],EDI ;将商品总利润在商店一中的偏移地址存入临时数组

ADD EBX,5 ;DI移向临时数组中下一元素

DEC CL

OR CL,CL

JZ BRK5

ADD EDX,24

JMP LPP

BRK5: ;临时数组中元素排序

MOV EBX,TAD

INC EBX ;取数组的首地址

MOV CL,M ;控制循环次数

XOR CH,CH

XOR ESI, ESI ;将SI清零

MOV EDI,5

MOV AL,M ;偏移上限

XOR AH,AH

IMUL AX,5

L1: MOV EDX, [EBX+ESI] ;用基变址寻址取操作数，L1为外循环，(SI)为循环变量，;相当于i

L2: CMP EDX, [EBX+EDI] ;L2为内循环，(DI)为循环变量，相当于j

JGE L3

XCHG EDX, [EBX+EDI] ;[BX+SI]<[BX+DI]，交换

MOV [EBX+ESI], EDX ;BP新值送回[BX+SI]

L3: ADD EDI,5 ;AH>=AL，不需交换，(AH)直接和后一个数比较，相当于j++

CMP EDI,EAX ;判断内层循环是否结束

JB L2

ADD ESI,5 ;外层变量SI加5

MOV EDI,ESI ;相当于j=i

ADD EDI,5

LOOP L1 ;通过寄存器实现两个存储器数据间的交换

MOV EBX,TAD ;利润相等时修改排名数值

XOR ESI,ESI

LPG:

MOV EAX,[EBX+ESI+1]

MOV ECX,[EBX+ESI+6]

MOV DX,[EAX]

CMP DX,[ECX] ;比较前后两利润率大小关系

JNE BRK11

MOV CL,[EBX+ESI]

MOV [EBX+ESI+5],CL ;序号变成一样的

BRK11:

ADD ESI,5

MOV DL,M

XOR DH,DH

DEC DX

IMUL DX,5

CMP SI,DX

JB LPG

MOV EDX,shop2 ;比对商店二与临时数组中相应位置保存的地址，若相同则将临时数组中对应序号赋给商店二中对应商品

XOR SI,SI ;SI置0

XOR CH,CH ;外循环计数器

LPH:

XOR DI,DI

XOR CL,CL ;内循环计数器

LPH\_1:

MOV AX,[EBX+EDI+1]

CMP AX,DS:[EDX+ESI+18] ;两地址比较

JNE BRK12

MOV AL,[EBX+EDI]

XOR AH,AH

MOV DS:[EDX+ESI+20],AX ;获得的排名放入商店二相应位置

JMP BRK14

BRK12:

ADD DI,5 ;DI指向临时数组中下一地址

INC CL

MOV AL,M

CMP CL,AL

JNE LPH\_1 ;当前商品排名未产生

MOV WORD PTR DS:[EDX+ESI+20],0 ;保险语句，万一在临时数组中未找到与目标商品相同的地址则将其排名置0

BRK14:

ADD SI,24 ;SI指向商店二中下一商品

INC CH

MOV AL,M

CMP CH,AL

JNE LPH ;商店二未遍历完成

RET

QUEUE ENDP

;子程序名：COMST

;功能：比较字符串

;入口参数：无

;出口参数：无

;所使用的寄存器：

;AX——作为中转寄存器

CMPST PROC ;比较字符串函数

MOV ESI,0

LPE:

MOV BH,DS:[EDI+ESI]

CMP BH,DS:[EDX+ESI]

JNE BRK19

INC SI

CMP SI,AX

JNE LPE

INC ESI

CMP BYTE PTR DS:[EDI+ESI],0 ;判断BX是否同样遍历完成

JNE BRK19

INC AH ;字符串相同

BRK19:

RET

CMPST ENDP

End

Main.c

#include<stdio.h>

#include<string.h>

#pragma warning(disable:4996)

typedef struct GOOD {

char name[10];

short ipr, opr, inm, onm;

int itr;

}GOOD;

char jhj[] = "进货价", xsj[] = "销售价", jhzs[] = "进货总数", flag;

;

extern void ITRST(char, GOOD\*, GOOD\*);

extern void QUEUE(char, GOOD\*, GOOD\*, char\*);

short alt(GOOD\* p, char\* buf);

short alt(GOOD\* p, char\* buf) {

char cbuf[4];

unsigned i;

do {

short n = 0;

flag = 1;

printf("%s:%hd >>",buf, p->ipr);

scanf("%s", cbuf); getchar();

if (!strlen(cbuf)) {

if (!strcmp(buf, jhj)) return p->ipr;

if (!strcmp(buf, xsj)) return p->opr;

if (!strcmp(buf, jhzs)) return p->inm;

}

for (i = 0;i < strlen(cbuf);i++) {

if (cbuf[i]<'0' || cbuf[i]>'9') {

flag = 0; break;

}

n = n \* 10 + cbuf[i] - '0';

}

if (flag == 0) continue;

return n;

}while (1);

}

void main() {

char id[10] = "CHR", pw[6] = "test", op, cbuf[10], \*buf2 = ":", \*buf3 = "access failed\n", tad[150];

GOOD \*p = NULL;

GOOD shop1[30] = { { "PEN",35,56,70,25 },{ "BOOK",12,30,25,5 },{ "BAG",25,35,20,12 } };

GOOD shop2[30] = { { "BOOK",12,28,20,15 },{ "PEN",35,50,30,24 },{ "BAG",20,35,42,35 } };

char n = 3, auth, count, i, j;

do {

do {

printf("input your name:");

scanf("%s", cbuf); getchar();

if (!strlen(cbuf)) {

auth = 0; break;

}

if (strcmp(id, cbuf)) continue;

printf("enter password:");

scanf("%s", cbuf); getchar();

if (!strcmp(pw, cbuf)) {

auth = 1; break;

}

} while (1);

count = 0;

do {

if (count) printf("请重新输入选项\n");

printf("选择操作:\n1.查询商品信息");

if (auth) printf("\t2.修改商品信息\t3.计算平均利润率\n4.计算利润率排名\t5.输出全部商品信息");

printf("\t6.退出\n");

op = getchar(); getchar();

} while (op < '1' || op>'6');

if (op == '6') break;

switch (op) {

case '1':

flag = 1;

do {

printf("input name of good:");

scanf("%s", cbuf); getchar();

if (!strlen(cbuf)) {

flag = 0; break;

}

for (i = 0;i < 3;i++)

if (!strcmp(shop1[i].name, cbuf)) break;

if (i == 3) {

printf("请重新输入\n"); continue;

}

break;

} while (1);

if (flag == 0) break;

printf("shop1\n商品名称:%s\t销售价:%hd\t进货总数:%hd\t销售总量:%hd\n", shop1[i].name, shop1[i].opr, shop1[i].inm, shop1[i].onm);

for (i = 0;i < 3;i++)

if (!strcmp(shop2[i].name, cbuf)) break;

printf("shop2\n商品名称:%s\t销售价:%hd\t进货总数:%hd\t销售总量:%hd\n", shop2[i].name, shop2[i].opr, shop2[i].inm, shop2[i].onm);

break;

case '2':

flag = 1;

do {

printf("输入网店名称:");

scanf("%s", cbuf); getchar();

if (cbuf[4] == '1') p = shop1;

if (cbuf[4] == '2') p = shop2;

printf("输入商品名称");

scanf("%s", cbuf); getchar();

if (!strlen(cbuf)) {

flag = 0; break;

}

for (i = 0;i < 3;i++)

if (!strcmp(shop1[i].name, cbuf)) break;

if (i == 3) {

printf("请重新输入\n"); continue;

}

break;

} while (1);

if (flag == 0) break;

p->ipr = alt(p, jhj);

p->opr = alt(p, xsj);

p->inm = alt(p, jhzs);

break;

case '3':

ITRST(n,&(shop1[0]),&(shop2[0]));

break;

case '4':

QUEUE(n, &(shop1[0]), &(shop2[0]),&(tad[0]));

break;

case '5':

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

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

if (strcmp(shop1[i].name, shop2[j].name)) continue;

printf("%s\t%hd\t%d\n", shop1[i].name, shop1[i].itr, shop2[j].itr);

}

}

break;

}

} while (1);

}

### 实验步骤

1. 准备上机实验环境。

2. 在VS2015中新建项目，将菜单与功能五用C语言改写。

3. 将任务1中的功能三，四提取，整理为forvs.asm，在变量名与函数名前加下划线，同时对相关变量与函数加EXTERN与PUBLIC声明。

4. 不断尝试生成解决方案，根据报错进行修改，直至生成成功。

5. 单步调试程序，直至每个功能均能返回正确结果。

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU @ 2.60GHz，2.86G内存；WINDOWS 7；Notepad++7.5.5.0；VS2015.

2. 在VS2015中建立空工程aasm，遵从VS环境下C与汇编混合编程教程设置环境变量，按照课内教授的C语言写法在main.c中实现主材单与功能一、二、五并导入工程，编译，发现按图3.2.1报错。请教同学后发现VS对函数比较挑剔，添加#pragma warning(disable:4996)报错解除。

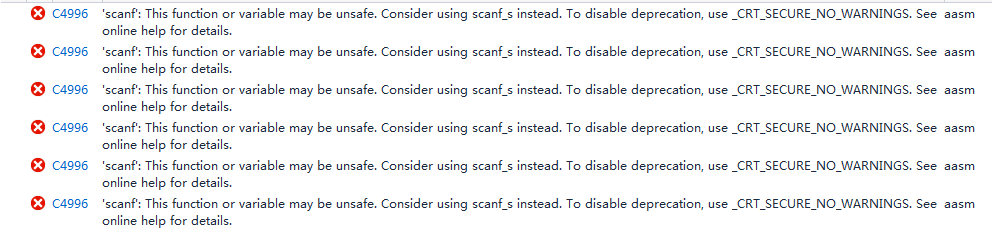


图3.2.1编译报错

3. VS环境下汇编参与混合编程不需要在变量名与函数名前加下划线。删除任务一wan1中原有的数据段，将原来定义的变量转移到C语言的变量定义中，作为参数传入汇编语言子程序，按Win32编程要求添加存储模型定义。

4. 由于WIN32环境下地址为32位，因此将所有寻址相关寄存器使用由“XX” 改为”EXX”，由于在进行功能四时需要将某些地址保存至内存中，因此临时数组tad长度应该为150（由WORD存地址改为DWORD存地址）。同时，由于C语言函数定义中参数采用传址方式，因此汇编语言中变量保存的是地址，所有的LEA语句应全部改为MOV语句。汇编源文件导入工程aasm，编译/汇编成功，如图3.2.2所示。

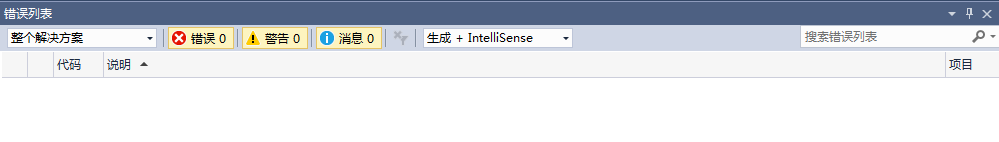


图3.2.2编译/汇编成功

5. 运行功能三时，发现函数无法返回。请教同学后学会了在VS下查看寄存器与内存的方式。单步执行功能三，发现在执行MOV EBP,shop1前后变量的值均改变，导致后面出现运行故障，如图3.2.3和图3.2.4所示。请教老师后发现局部变量实际是使用EBP作为指针的，把EBP的值改了对应变量的地址和值就应该变化，因此返回forvs.asm将所有EBP改为其他寄存器。由于32位下寻址时对寄存器的要求宽松许多，因此只需考虑寄存器是否被占用的问题。修改后重新观察执行情况，发现正确的值被存入正确的地址。

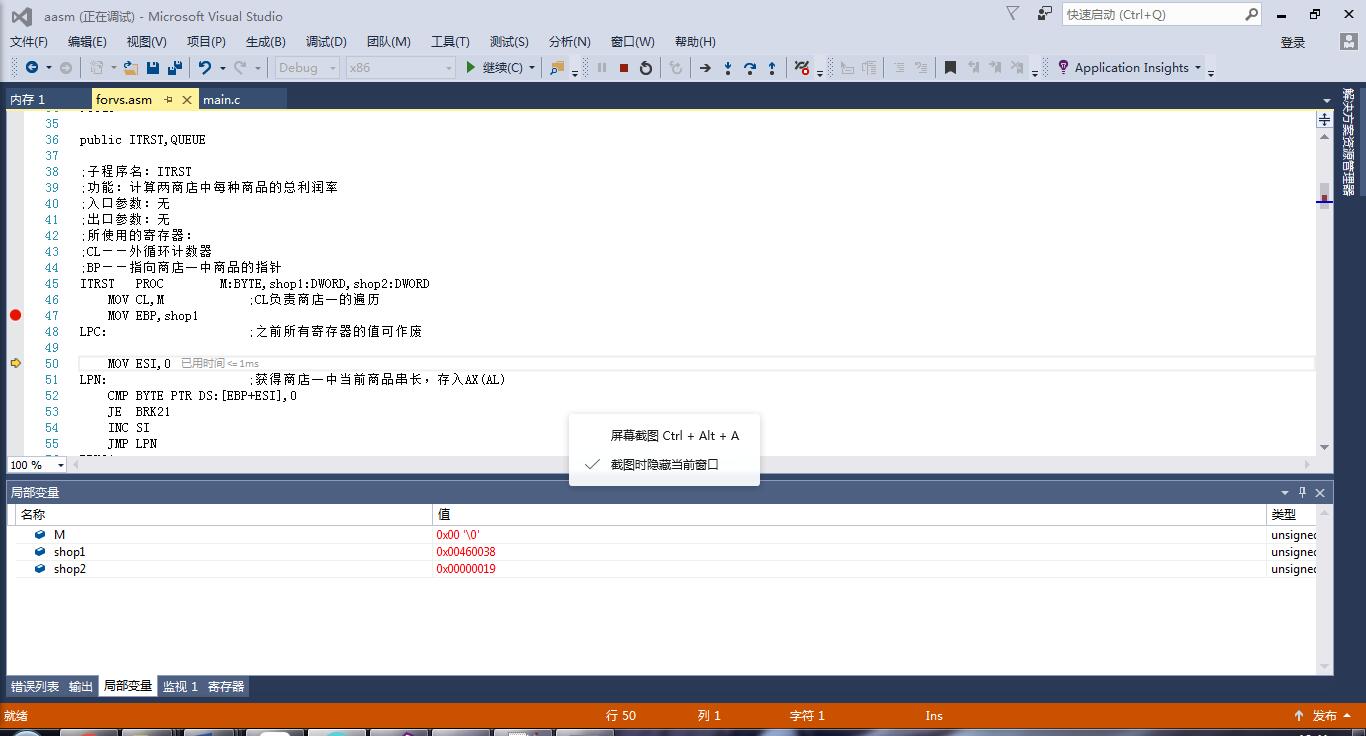
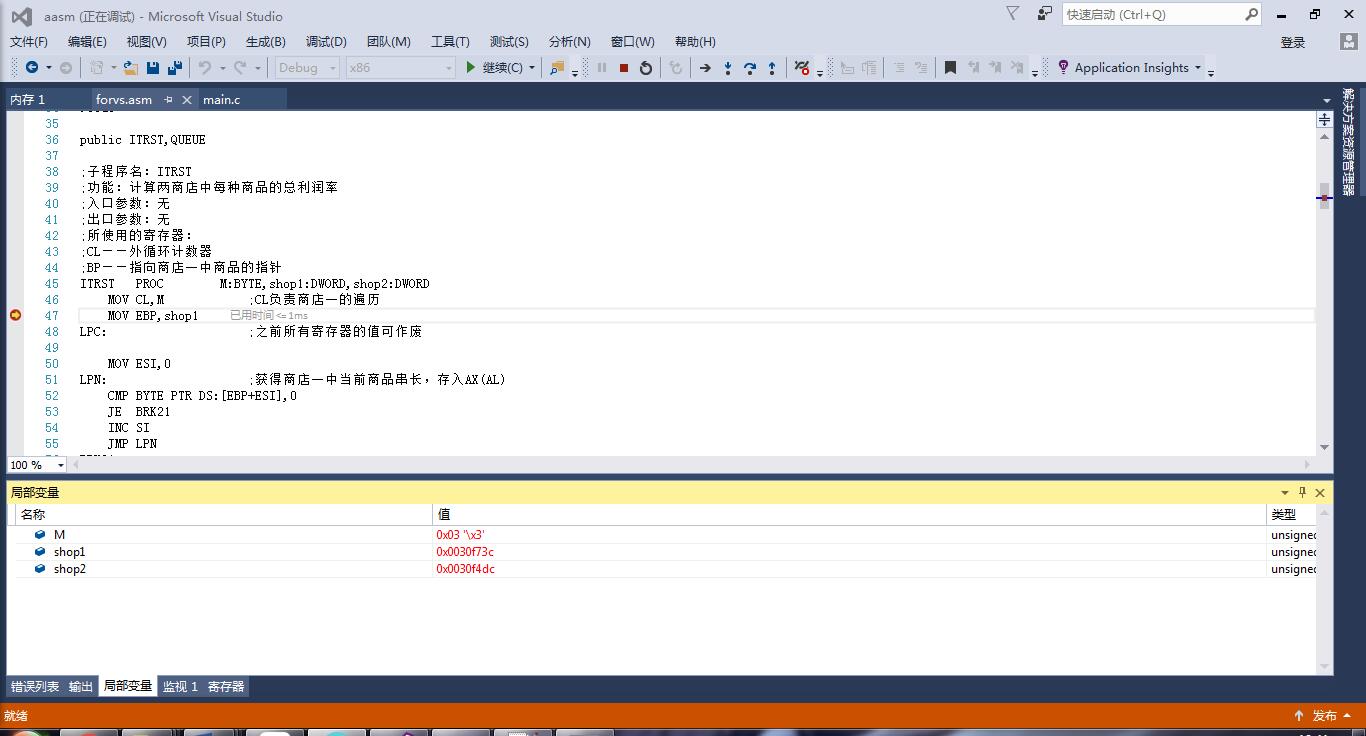


图3.2.3&图3.2.4 执行MOV EBP,shop1前后变量值对比

6. 在确保功能四运行正常的前提下进入功能五，发现输出结果错误，如图3.2.5所示。

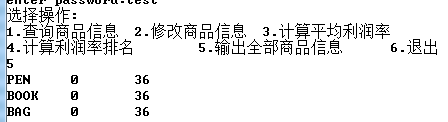


图3.2.6 GOOD类型在内存中的表示

7. 观察GOOD结构在内存中的表示形式，可观察到在最后的int类型和前面的short类型之间有两字节的cc，如图3.2.6所示。请教同学后得知int的范围为后面四个字节，两个cc用于字节对齐。修改利润与排名在结构体中存放的位置（18改成20），使其落在int的范围内，再次运行。



图3.2.6 GOOD类型在内存中的表示

8. 执行功能五时，发现除pen的利润外其他结果完全正确，而65522的二进制表示正好是正确值-14的补码，如图3.2.7所示。回忆起在计算利润时是以两字节（short）形式放入内存，而此处在输出时依然采用了int型，即连同高16位的0一起形成十进制数65522。将利润的输出方式改为short，再次运行。

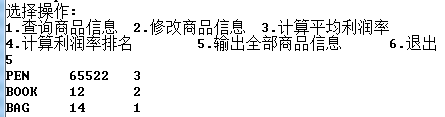


图3.2.7 PEN的利润错误

9. 功能五输出正确结果，如图3.2.8所示。

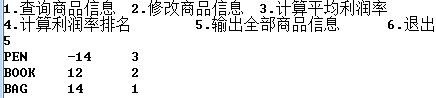


图3.2.8输出正确

# 总结与体会

通过任务1的实验，我基本了解了DOS下多文件汇编的操作，子程序，参数传递与宏定义相关操作与知识，同时开始有意识通过数组使用内存与寄存器之间的数据交互。同时这是我第一个与他人协作完成的编程作业，我体会到了成员之间提前协调好接口，约定好公共变量的重要性。这次我们是在编号自己的部分以后在进行的如上操作，发现将自己的程序与对方的程序的变量一一进行比对着实要占用不少时间。

任务二的实验中我面临的第一大问题便是编程环境的选取。本次实验之前我由于VS体积太大，速度较慢很少使用，这次可以说是我第一次认真考虑使用VS编程。然而对环境的陌生，写C语言部分时只在VS下出现的莫名其妙的报错使我一度转投Borland,而诸多同学对Borland C的控诉又使我开始纠结于究竟选用什么编译器。最后请教同学后才终于用pragma语句解决奇怪的报错，开始进入汇编语言部分的修改。在本任务中我熟悉了之前完全没用过的VS2015的基本操作，第一次看到了IDE中的数据对齐，了解了汇编与C混合编程时EBP由于传参而被占用，但要承认的是对于模型与环境变量的使用还要多加练习，增加熟练度。

我得到的教训是对于在常见IDE下的常见错误应该有基本的了解并准备好解决方案，以免在做任务的时候干着急，同时在该给电脑更新换代的时候也要及时换代，不能自己给自己造出电脑带不动IDE的困难。

思考题：

一、

1.用F7可以进入单步调试进入子程序;也可用光标选中子程序的语句，F4一步到位。

2.near类型函数中ret不变；far类型函数中ret变成retf。Far类型函数被call时先压入当前段CS再压入IP。

3.模块调用关系图、子程序功能说明、输入/输出说明可显著增加代码的可读性，不仅便于别人理解自己的代码，更利于自己在debug时对各个寄存器的情况心里有数。

7.宏指令在反汇编时会被展开进代码段的相应位置，而子程序则保持与主程序的相对位置不变。

8.可以给菜单加边框使其更加醒目，也可以对菜单中的各个选项通过制表使其排列更整齐。

11.可将经常使用又不便写成子程序的程序段写成宏以精简代码。

二、

3.VS2015中汇编语言使用的变量不需要加下划线；主，子程序之间通过堆栈传递参数，其中需要用到EBP。

4.在每次调试时发现数据段的位置各不相同，比如同一个变量shop1，在多次执行中有各种各样的值。若变量在汇编中定义为shop1:DWORD，则反汇编后MOV EDX,shop1变为00D4279E mov edx,dword ptr [shop1]。

7.多语言混合编程时，相对于单语言观察的角度更多，比如本次在调试C语言时可观察内存或反汇编语句检查逻辑正确性。

# 参考文献

[1]许向阳，《80X86汇编语言程序设计上机指南》“第十一章 汇编语言程序与C程序的连接”。