

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实践**

**专业班级：计算机科学与技术ACM1801班**

**学 号： U2018414593**

**姓 名： 程迎松**

**指导教师： 曹忠升**

**实验时段： 2020年3月19日~5月7日**

**实验地点： 江西省婺源县紫阳镇**

**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：程迎松

报告日期：2020.5.10

实验报告成绩评定：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 实验完成质量（70%），报告撰写质量（30%），每次满分20分。 |  |  |  |  |  |
| 合计（100分） |  | | | | |

备注：实验完成质量从实验目的达成程度，设计方案、实验方法步骤、实验记录与结果分析论述清楚等方面评价；报告撰写质量从撰写规范、完整、通顺、详实等方面评价。

指导教师签字：

                    日期：

目录

[课程总体说明 - 5 -](#_Toc41395411)

[0.1 课程目标 - 5 -](#_Toc41395412)

[0.2 成绩构成 - 5 -](#_Toc41395413)

[0.3 实验任务的总体描述 - 5 -](#_Toc41395414)

[1 编程基础 1](#_Toc41395415)

[1.1 实验目的与要求 1](#_Toc41395416)

[1.2 实验内容 1](#_Toc41395417)

[1.3 任务1.1至1.3实验过程 3](#_Toc41395418)

[1.3.1 实验步骤 3](#_Toc41395419)

[1.3.2实验记录与分析 3](#_Toc41395420)

[1.4 任务1.4的实验过程 5](#_Toc41395421)

[1.4.1 设计思想 5](#_Toc41395422)

[1.4.2 流程图 6](#_Toc41395423)

[1.4.3 源程序 6](#_Toc41395424)

[1.4.4 实验步骤 13](#_Toc41395425)

[1.4.5 实验记录与分析 13](#_Toc41395426)

[1.5 小结 16](#_Toc41395427)

[2 程序优化 17](#_Toc41395428)

[2.1 实验目的与要求 17](#_Toc41395429)

[2.2 实验内容 17](#_Toc41395430)

[2.3 任务2.1实验过程 18](#_Toc41395431)

[2.3.1 设计思想 18](#_Toc41395432)

[2.3.2 流程图 18](#_Toc41395433)

[2.3.3 源程序 18](#_Toc41395434)

[2.4 任务2.2实验过程 21](#_Toc41395435)

[2.4.1 设计思想 21](#_Toc41395436)

[2.4.2 具体实现细节 21](#_Toc41395437)

[2.4.3 源程序 21](#_Toc41395438)

[2.4.4 实验记录与分析 24](#_Toc41395439)

[2.5 小结 25](#_Toc41395440)

[3 模块化程序设计 27](#_Toc41395441)

[3.1 实验目的与要求 27](#_Toc41395442)

[3.2 实验内容 27](#_Toc41395443)

[3.3 任务3.1实验过程 28](#_Toc41395444)

[3.3.1 设计思想 28](#_Toc41395445)

[3.3.2 模块调用关系 28](#_Toc41395446)

[3.3.3 源程序 28](#_Toc41395447)

[3.3.4 实验记录和分析 32](#_Toc41395448)

[3.4 任务3.2实验过程 34](#_Toc41395449)

[3.4.1 设计思想 34](#_Toc41395450)

[3.4.2 实验方法说明 34](#_Toc41395451)

[3.4.3 实验记录和分析 35](#_Toc41395452)

[3.5 小结 37](#_Toc41395453)

[4 中断与反跟踪 38](#_Toc41395454)

[4.1 实验目的和要求 38](#_Toc41395455)

[4.2 实验内容 38](#_Toc41395456)

[4.3 任务4.1实验内容 40](#_Toc41395457)

[4.3.1 设计思想 40](#_Toc41395458)

[4.3.2 流程图 40](#_Toc41395459)

[4.3.3 源代码 40](#_Toc41395460)

[4.3.4 实验记录和分析 43](#_Toc41395461)

[4.4 任务4.2，4.3实验内容 45](#_Toc41395462)

[4.4.1 设计思想 45](#_Toc41395463)

[4.4.2 实验记录和分析 46](#_Toc41395464)

[4.5 小结 48](#_Toc41395465)

[5 WIN32编程 49](#_Toc41395466)

[5.1 实验目的与要求 49](#_Toc41395467)

[5.2 实验内容 49](#_Toc41395468)

[5.3 任务5.1实验内容 50](#_Toc41395469)

[5.3.1 实验思路 50](#_Toc41395470)

[5.3.2 排序步骤 50](#_Toc41395471)

[5.3.3 实验记录和分析 50](#_Toc41395472)

[5.4 小结 52](#_Toc41395473)

[参考文献 54](#_Toc41395474)

# 课程总体说明

## 0.1 课程目标

下表是本课程的目标及与支撑的毕业要求指标点之间的关系。请大家关注下表中最后一列“实验中的注意事项”的内容，以便更有针对性的满足课程目标的要求。

|  |  |  |
| --- | --- | --- |
| 课程目标 | 支撑的毕业要求指标点 | 实验中的注意事项 |
| 掌握汇编语言程序设计的全周期、全流程的基本方法与技术，通过程序调试、数据记录和分析，了解影响设计目标和技术方案的多种因素。 | 3.1掌握与计算机复杂工程问题有关的工程设计和软硬件产品开发全周期、全流程的基本设计/开发方法和技术，了解影响设计目标和技术方案的多种因素。 | 不能只写代码完成功能，还要有设计、调试、记录、分析等部分的内容。 |
| 掌握编写、调试汇编语言程序的基本方法与技术，能根据实验任务要求,设计出较充分利用了汇编语言优势的软件功能部件或软件系统。 | 3.2能为计算机复杂工程问题解决方案设计满足特定需求的软/硬件模块。 | 要思考与运用汇编语言的优势编写某些程序。 |
| 熟悉支持汇编语言开发、调试以及软件反汇编的主流工具的功能、特点与局限性及使用方法。 | 5.1了解计算机专业常用的现代仪器、信息技术工具、工程工具和模拟软件的使用原理和方法，并理解其局限性。 | 熟悉实验中使用的工具，把对工具的看法记录在案。 |

## 0.2 成绩构成

实验课程综合成绩由实验过程成绩和实验报告成绩二部分构成。**实验过程成绩**：30%。主要考察各实验完成过程中的情况，希望大家做到预习准备充分，操作认真熟练，在规定的时间内完成实验任务，结果正确，积极发现和提出问题，交流讨论时描述问题准确、清晰。实验报告成绩：70%。主要考核报告体现的实验完成质量(含问题的分析、设计思想与程序、针对问题的实验方法与步骤、实验记录、实验结果分析等方面)和报告格式规范等撰写质量方面的内容。

## 0.3 实验任务的总体描述

本课程安排了8次4学时的课内实验课时，将实现一个具有一定复杂程度的系统。对该系统的相关要求被划分成了**5个主题**：1）搭建原型系统；2）在原型系统基础上探索程序指令级别的优化；3）通过模块化调整与优化原型系统的程序结构；4）通过中断、内存数据和地址操纵、跟踪与反跟踪、加密等措施增强系统安全性；5）程序在不同平台上的移植。

针对这5个主题，对应地布置了5次实验。**实验1（编程基础）**安排8个课内学时熟悉汇编语言程序设计的基本方法、技术与工具，设计实现指定原型系统的主要功能。针对原型系统的搭建，实验报告中要有全周期、全流程的描述。**实验2（程序优化）**安排4个课内学时探索如何通过选择不同的指令及组合关系来优化程序的性能或代码长度。**实验3（模块化程序设计）**安排8个课内学时，利用子程序、模块化程序设计方法、与C语言混合编程等，调整与优化程序结构。**实验4（中断与反跟踪）**安排8个课内学时，通过利用中断机制、内存数据和地址操纵技术、跟踪与反跟踪技巧、加密等措施增强系统安全性。**实验5（WIN32程序设计）**安排4个课内学时，熟悉在不同操作系统平台上移植实现已有系统功能的基本方法。每次实验的侧重面有所不同，但都会涉及到课程目标的三个方面，因此，需要大家在实验过程中以及实验报告中有所注意和体现。

**本次课程所涉及的原型系统是一个网店商品信息管理系统。下面描述该系统的基本需求，后续每次实验都是以这个基本需求为背景而展开的。**

有一个老板在网上开了1个网店SHOP，网店里有n种商品销售。每种商品的信息包括：商品名称（最长名称9个字节，其后加一个数值0表示名称结束），折扣（字节类型，取值0~10；0表示免费赠送，10表示不打折，1~9为折扣率；实际销售价格=销售价\*折扣/10），进货价(字类型)，销售价（字类型），进货总数（字类型），已售数量（字类型），推荐度【=（进货价/实际销售价格+已售数量/（2\*进货数量））\*128，字类型】。老板管理网店信息时需要输入自己的名字（最长名字9个字节，其后加一个数值0表示结束）和密码（最长密码6个字节，其后加一个数值0表示结束），老板登录后可查看商品的全部信息；顾客（无需登录）可以查看网店中每个商品除了进货价以外的信息，可以对指定商品下单预定。

该系统被执行后，首先显示一个菜单界面，菜单界面信息包括：

当前用户名：（老板名称或顾客）

当前浏览商品名称：（没有时空缺）

请输入数字1…9选择功能：

1.登录/重新登录

2.查找指定商品并显示其信息

3.下订单

4.计算商品推荐度

5.排名

6.修改商品信息

7.迁移商店运行环境

8.显示当前代码段首址

9.退出

当用户输入某一个有效数字后，就进入到指定的功能中执行，执行完之后再回到该菜单界面。如果选择的是退出功能，则程序退出。该菜单中每项菜单的具体功能要求详见每次的实验任务描述。

# 1 编程基础

## 实验目的与要求

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

1. 掌握汇编源程序编辑工具、汇编程序、连接程序、调试工具TD的使用；
2. 理解数、符号、寻址方式等在计算机内的表现形式；
3. 理解指令执行与标志位改变之间的关系；
4. 熟悉常用的DOS功能调用；
5. 熟悉分支、循环程序的结构及控制方法，掌握分支、循环程序的调试方法；
6. 加深对转移指令及一些常用的汇编指令的理解；
7. 掌握设计实现一个原型系统的基本方法。

## 实验内容

任务1.1：《80X86汇编语言程序设计》教材中 P31的 1.14题。要求：

(1) 直接在TD中输入指令，完成两个数的求和、求差的功能。求和/差后的结果放在(AH)中。

(2) 请事先指出执行指令后(AH)、标志位 SF、OF、CF、ZF的内容。

(3) 记录上机执行后的结果，与（2）中对应的内容比较。

任务1.2 《80X86汇编语言程序设计》教材中 P45的 2.3题。要求：

（1）分别记录执行到“MOV CX，10”和“INT 21H”之前的(BX),(BP), (SI), (DI)是多少。

（2）记录程序执行到退出之前数据段开始40个字节的内容，指出程序运行结果是否与设想的一致。

任务1.3 《80X86汇编语言程序设计》教材中 P45的 2.4题的改写。要求：

(1) 实现的功能不变，但对数据段中变量访问时所用到的变址寄存器采用32位寄存器。

(2) 记录程序执行到退出之前数据段开始40个字节的内容，检查程序运行结果是否与设想的一致。

(3)在TD代码窗口中观察并记录机器指令代码在内存中的存放形式，并与TD中提供的反汇编语句及自己编写的源程序语句进行对照，也与任务1.2做对比。（相似语句记录一条即可，重点理解机器码与汇编语句的对应关系，尤其注意操作数寻址方式的编码形式，比如寄存器间接寻址、变址寻址、32位寄存器与16位寄存器编码的不同、段前缀在代码里是如何表示的等）。

（4）观察连续存放的二进制串在反汇编成汇编语言语句时，从不同字节位置开始反汇编，结果怎样？理解 IP/EIP指明指令起始位置的重要性。

任务1.4：实验主要是利用分支、循环程序的结构，实现该系统的基本功能，并能熟悉全周期、全流程地设计实现一个原型系统的基本方法。本次实验要具体实现的功能要求如下：

0.主菜单界面

完整显示“实验任务的总体描述”中给出的界面信息。等待用户输入数字（可使用1号DOS系统功能调用）。对用户输入的字符进行判断，看是否是1~9的数字；是的话就转移到对应功能的程序标号，不是的话就提示错误，回到主菜单界面。

1.登录/重新登录

（1）先后分别提示用户输入姓名和密码（可使用9号DOS系统功能调用）。

（2）分别获取输入的姓名和密码（可使用10号DOS系统功能调用）。输入的姓名字符串放在以in\_name为首址的存储区中，密码放在以in\_pwd为首址的存储区中。

（3）若输入姓名时只是输入了回车，则将0送到AUTH字节变量中，回到主菜单界面。

（4）进行身份认证：

（a）使用循环程序结构，比较姓名是否正确。若不正确，则跳到（c）。

（b）若正确，再比较密码是否相同，若相同，跳到（d）。

（c）若名字或密码不对，则提示登录失败，并转到“（3）”的位置。

（d）若名字和密码均正确，则将1送到AUTH变量中，回到主菜单界面。

2.查找指定商品并显示其信息

（1）提示用户输入商品名称。

（2）在商店中寻找是否存在该商品。

（3）若存在，则将商品名称或地址记录到GOOD字段中。商品信息的显示暂时不做。返回到主菜单界面。

（4）若没有找到，提示没有找到，返回到主菜单界面。

3.下订单

（1）判断当前浏览商品是否有效（GOOD不为空），若有效，判断其剩余数量是否为0，不为0则将已售数量加1，重新计算所有商品的推荐度（目前不是用子程序实现的，所以，跳转之前，要把返回地址送到指定变量中），返回主菜单界面。

（2）若无效或剩余数量为0，则提示错误，回到主菜单界面。

4.计算商品推荐度

按照给出的公式计算所有商品的推荐度，返回到指定的位置（JMP 含返回地址的指定变量）。

5.排名

暂不实现，直接返回主菜单界面。

6.修改商品信息

暂不实现，直接返回主菜单界面。

7.迁移商店运行环境

暂不实现，直接返回主菜单界面。

8.显示当前代码段首址

将当前代码段寄存器CS里面的内容按照16进制的方式显示到屏幕上，返回主菜单界面。

9.退出

退出本系统（可使用4CH号DOS系统功能调用）

## 任务1.1至1.3实验过程

### 实验步骤

1. 准备上机实验环境，对实验用到的软件进行安装、运行，通过试用初步了解软件的基本功能、操作等。

2. 任务1.1：在TD的代码窗口中的当前光标下输入第一个运算式对应的两个8位数值对应的指令语句MOV AH,01001. 准备上机实验环境，对实验用到的软件进行安装、运行，通过试用初步了解软件的基本功能、操作等。

1101B；MOV AL,-01110010B；ADD AH,AL；观察代码区显示的内容与自己输入字符之间的关系；然后确定CS:IP指向的是自己输入的第一条指令的位置，单步执行三次，观察寄存器内容的变化，记录标志寄存器的结果。

3.任务1.2：复制教材代码，并使用TD进行调试，单步执行到指定位置停止，并观察相应寄存器的值。并在程序最后设置断点，直接执行到断点观察存储区

4.任务1.3：使用变址寻址代替寄存器寻址，只使用一个寄存器来存储BUF首址，然后按字节偏移使用变址寻址。

5.观察TD中的反汇编代码，并观察程序单步执行过程的IP/EIP值。

6. 尝试按照自己想的其他语句及输入格式等进行操作，积累更多的经验。

### 1.3.2实验记录与分析

1. 实验环境条件：P3 1GHz，256M内存；WINDOWS 10下DOSBox0.72； TD.EXE 5.0。

2. 任务1.1，观察各标志位的值

表1.1 各标志位表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 表达式 | SF | OF | CF | ZF |
| 11H+09H = 1AH | 0 | 0 | 0 | 0 |
| FF00H+0H=FF00H | 1 | 0 | 0 | 0 |
| 40H+40H = 80H | 1 | 1 | 0 | 0 |
| 80H+80H = 00H | 0 | 1 | 1 | 1 |
| 0H-80H = 80H | 1 | 1 | 0 | 0 |
| 00H–01H = FFH | 1 | 0 | 1 | 0 |

3.任务1.2，开始在CODE段没有写USE16,会导致CODE段循环之前的MOV指令的语句出问题。

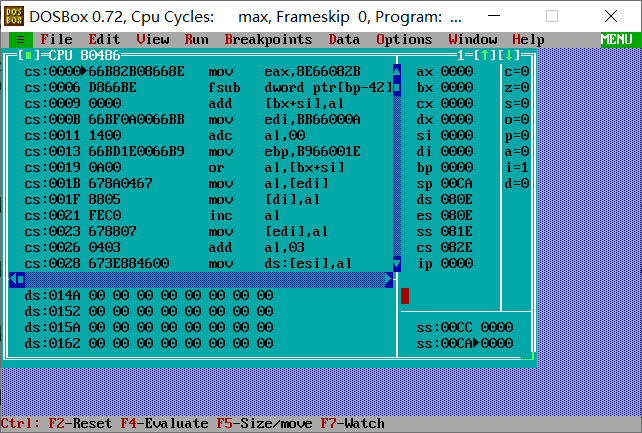


图1.1 语句翻译错误图

4. 任务1.2的结果和任务1.3的结果都相同。

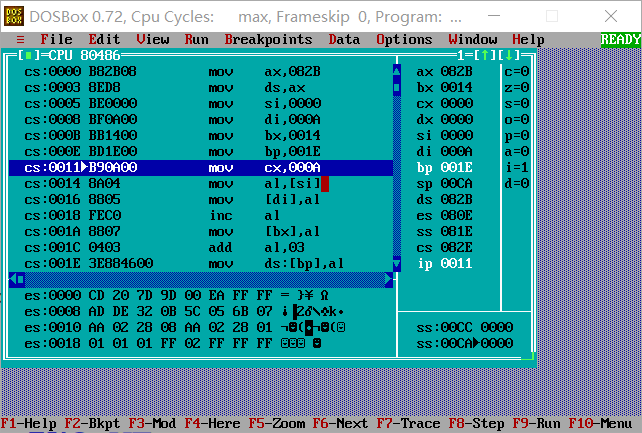


图1.2 执行到MOV CX,10时寄存器值

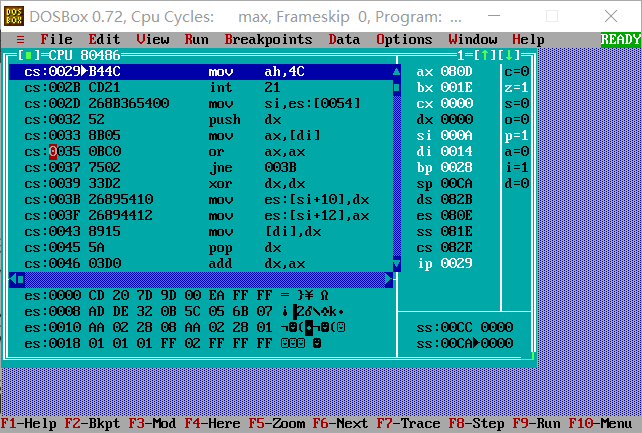


图1.3 执行到INT 21H的各寄存器值

5. 可以看出，反汇编代码是将源操作数和目的操作数的位置进行互换。变址寻址会在前面自动加上段前缀。32位寄存器的会有e做前缀，并且赋值时会扩充为32位数。IP值会指向当前语句的下一条语句的代码地址。

## 任务1.4的实验过程

### 设计思想

1.显示主菜单，将提示信息存储为字符串放在存储器中，使用DOS9号调用输出到屏幕中。其中如果没有登录，就在用户名位置直接换行输出。

2.检测用户输入数字，首先检查是不是0-9的数字，如果不是提示错误信息，并回到主界面（使用转移指令）。如果是使用分支结构进入各个功能区

3. 功能1，登录；首先输出提示用户输入用户名的信息（把提示信息存储在存储器中），读取用户输入的用户名，使用循环逐字符的比对用户输入和存储器中的老板名称，如果有字符不相同，把ATUH置0，返回主程序。如果直到出现’$’还是相同，就用相同的办法提示用户输入密码，比对密码，还是相同就把AUTH置1返回主程序。

4. 功能2，查找商品。首先输出提示用户输入要查询的商品名称，然后循环在存储器中寻找商品。（对每个商品的商品名进行字符串比对），每个外循环地址偏移21个字节，也就是跳转到下一个商品位置。如果查到商品，把该商品的所有信息存储到GOOD中。

5. 功能3，首先检查当前浏览商品是否有效，还有是否售完，如果无效或售完输出错误信息返回主界面，不然就将售出数目加1，重新计算所有商品的推荐度

6. 功能4 按照公式计算每个商品的推荐度，其中把128放入公式中先乘，避免出现除法结果取整是0之后计算出错。还有在进行字除法前要把DX清零，避免溢出。在调用功能4前保存指令地址，现在跳转回去

7.功能8，获取当前代码段首址，然后每4位取出，如果是0-9，就转换为字符‘0’-‘9’,A-F同样处理，输出

### 流程图

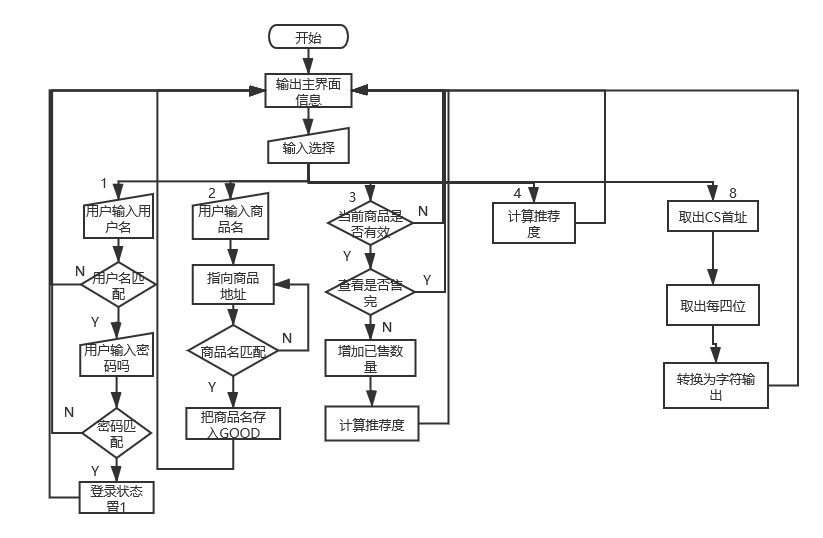


图1.4 商品系统流程图

### 源程序

INCLUDE IO\_M.LIB

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

BNAME DB 'CYINGSONG','$' ;老板的姓名

BPASS DB 'test00',0 ;密码6个字节

ATUH DB 0 ;当前登录状态

GOOD DB '$',9 DUP(0) ;当前浏览商品名称

;GOOD DB 0000H

N EQU 30

M EQU 21 ;每个商品占据字节数

SNAME DB 'SHOP',0 ;网店名称

GA1 DB 'PEN$',6 DUP(0),10 ;商品名称和折扣

DW 35,56,70,25,?

GA2 DB 'BOOK$',5 DUP(0),9

DW 12,30,25,5,?

GAN DB N-2 DUP('TempValue$',8,15,0,20,0,30,0,2,0,?,?)

;GAN DB N-2 DUP('TempValue$',9,12,0,30,0,25,0,5,0,?,?)

OUTPUT1 DB '----------------------------',0AH,0DH

DB 'current user name:$'

OUTPUT2 DB 'Currently browsing the product name:$'

OUTPUT3 DB 'Please input the number 1... 9 to selection function:',0AH,0DH

DB '1.Login/login again',0AH,0DH

DB '2.Finds the specified item and displays its information',0AH,0DH

DB '3.place an order',0AH,0DH

DB '4.Calculate the product recommendation',0AH,0DH

DB '5.ranking',0AH,0DH

DB '6.Modify product information',0AH,0DH

DB '7.Migrate the store runtime environment',0AH,0DH

DB '8.Displays the current code snippet header',0AH,0DH

DB '9.Quit',0AH,0DH,"$"

INPUTNAME DB 'Please input a user name:$'

INPUTPW DB 'Please input password:$'

INPUTGOOD DB 'Please enter the product name:$'

NOTFIND DB 0AH,0DH,'NOT FIND!',0AH,0DH,'$'

NOGOODS DB 0AH,0DH,'NOGOODS!',0AH,0DH,'$'

in\_name DB 10

DB 0

DB 10 DUP(0)

in\_pwd DB 10

DB 0

DB 10 DUP(0)

in\_good DB 10

DB 0

DB 10 DUP(0)

Temp DW 0

T DB '0000H','$'

DATA ENDS

CODE SEGMENT USE16

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

START: ;显示主界面

MOV AX, DATA

MOV DS, AX

CRLF ;换行

WRITE OUTPUT1 ;输出用户名

CMP ATUH,0 ;如果没有登录就直接换行

JE CHANGELINE1

WRITE BNAME

CHANGELINE1:

CRLF

WRITE OUTPUT2 ;输出当前浏览商品名称

WRITE GOOD

CRLF

WRITE OUTPUT3 ;输出界面提示信息

MAIN: ;主菜单界面

;接受用户输入一个字符

MOV AH,1

INT 21H

CMP AL,'9' ;检查是否是0-9的数字

JA MAIN

CMP AL,'1'

JB MAIN

;根据输入转移到各自的功能

CMP AL,'1'

JE Login

CMP AL,'2'

JE FindItem

CMP AL,'3'

JE Order

MOV SI,OFFSET MAIN

CMP AL,'4'

JE Calculate

CMP AL,'5'

JE Rank

CMP AL,'6'

JE Modify

CMP AL,'7'

JE Migrate

CMP AL,'8'

JE Display

CMP AL,'9'

JE Quit

Login: ;用户登录

CRLF

WRITE INPUTNAME

;用户输入名字

READ in\_name

LEA SI,in\_name+2

MOV AL,13

CMP [SI],AL ;如果输入的是回车直接返回主界面

JNE A

BACK: ;匹配失败，回到主界面

MOV ATUH,0

JMP START

A:

MOV AL,'$'

LEA DI,BNAME

LOOP1: ;检查用户名

MOV BL,[DI]

CMP [SI],BL

JNE BACK

INC SI

INC DI

CMP [DI],AL

JNE LOOP1

;提示并输入密码

CRLF

WRITE INPUTPW

READ in\_pwd

CRLF

;检查密码

LEA SI,in\_pwd+2

LEA DI,BPASS

MOV CX,6 ;密码6个字节

;CMP CX,in\_pwd+1 ;检查输入位数是否相同

;JNE BACK

LOOP2:

MOV BL,[DI]

CMP [SI],BL

JNE BACK

INC SI

INC DI

LOOP LOOP2

MOV ATUH,1

JMP START

FindItem: ;查找指定商品并显示其信息

;提示用户输入商品名称

CRLF

WRITE INPUTGOOD

;用户输入商品名称

READ in\_good

;在商店中寻找是否存在该商品

LEA SI,in\_good+2 ;SI中存储用户的输入

LEA DI,GA1-M ;DI存储商品信息首址

;每个商品占21个字节，一共有N个商品

MOV CX,N+1 ;循环次数

MOV AL,'$'

LOOP3:

DEC CX

CMP CX,0

JNE D ;没有找到商品，返回主界面

WRITE NOTFIND

JMP MAIN

D: ADD DI,M ;每次移动M个字节

LEA BX,[DI]

LOOP4: ;检查用户名

MOV DL,[BX]

CMP [SI],DL

JNE LOOP3 ;匹配失败，跳出内循环

INC SI

INC BX

CMP [BX],AL

JNE LOOP4

;找到商品把信息记录到GOOD中

MOV BX,0

MOV DX,'$' ;作为循环的计数器

COPY:

MOV AL,[DI]

MOV GOOD[BX],AL

INC BX

INC DI

CMP [DI],DX

JNE COPY

MOV GOOD[BX],'$'

JMP START

;JMP MAIN

Order: ;下订单

;判断GOOD是否为空

MOV AL,'$'

CMP GOOD,AL

JNE E

WRONG:

WRITE NOGOODS

JMP START

E:

LEA DI,GA1-M

LEA SI,GOOD

MOV CX,N+1 ;循环次数

MOV AL,'$'

OUTLOOP:

ADD DI,M ;每次移动M个字节

LEA BX,[DI]

INLOOP: ;检查用户名

MOV DL,[BX]

CMP [SI],DL

JNE OUTLOOP ;匹配失败，跳出内循环

INC SI

INC BX

CMP [BX],AL

JNE INLOOP

;检查剩余数目是否为0

MOV AX, WORD PTR 15[DI] ;取出进货总数

MOV BX, WORD PTR 17[DI] ;取出已经销售的数量

SUB AX,BX

JZ WRONG ;如果售完就返回

INC WORD PTR 17[DI]

LEA SI,OFFSET FUNC

FUNC:

JMP MAIN

Calculate: ;计算商品推荐度

;推荐度=(（进货价/实际销售价格+已售数量/（2\*进货数量）\*128)

; = (进货价\*1280/（销售价\*折扣）+64\*已售数量/进货数量

LEA DI,GA1-M

MOV CX,N

Cal\_Re:

ADD DI,M

MOV BL,10[DI] ;取出折扣

MOV AX,WORD PTR 13[DI] ;取出销售价格

MUL BL

MOV BX,AX ;AX中是(销售价\*折扣）

MOV AX,WORD PTR 11[DI] ;BX中是进货价

MOV DX,1280

MUL DX ;现在AX中是进货价\*1280

MOV DX,0

DIV BX

MOV Temp,AX ;把第一部分结果存入Temp

MOV AX,WORD PTR 17[DI] ;取出已经销售的数量

MOV DL,64

MUL DL ;AX中是64\*已售数量

MOV BX,WORD PTR 15[DI] ;取出进货总数

MOV DX,0 ;要将DX清零再进行字除法，防止溢出

DIV BX

MOV BX,Temp

ADD BX,AX

MOV WORD PTR 19[DI],BX

LOOP Cal\_Re

JMP SI

Rank:

JMP MAIN

Modify:

JMP MAIN

Migrate:

JMP MAIN

Display: ;将当前代码段首址显示在屏幕上

MOV AX,CS

MOV SI,OFFSET T+3

XOR CX,CX

MOV CL,4

Dis:

MOV DH,AL

shr AX,1

shr AX,1

shr AX,1

shr AX,1 ;让AX逻辑右移4位

AND DH,0FH

ADD DH,30H

CMP DH,':'

JA isLetter

JB NO

IsLetter:

ADD DH,7H

No:

MOV [SI],DH

DEC SI

LOOP Dis

WRITE T

JMP MAIN

Quit:

MOV AX,4C00H

INT 21H

CODE ENDS;

END START

### 实验步骤

1. 准备上机实验环境。

2. 使用notepad++编辑源程序，存盘文件后缀名为ASM。使用MASM 6.0汇编源文件。观察提示信息，若出错，则用编辑程序修改错误，存盘后重新汇编，直至不再报错为止。

3. 使用连接程序LINK.EXE将汇编生成的OBJ文件连接成执行文件。若连接时报错，则依照错误信息修改源程序。之后重新汇编和连接，直至不再报错并生成EXE文件。

4. 执行该程序，观察执行现象。进行测试各分支的功能，观察执行的结果并进行调试。

5. 使用TD.EXE观察各功能使用时寄存器和标志位的变化。

### 实验记录与分析

1.实验环境条件：P3 1GHz，256M内存；WINDOWS 10下DOSBox0.72； TD.EXE 5.0。

2.要求0，显示主界面，利用宏简化代码，输出字符串。汇编界面不能输出中文字符，所以提示信息改成英文。

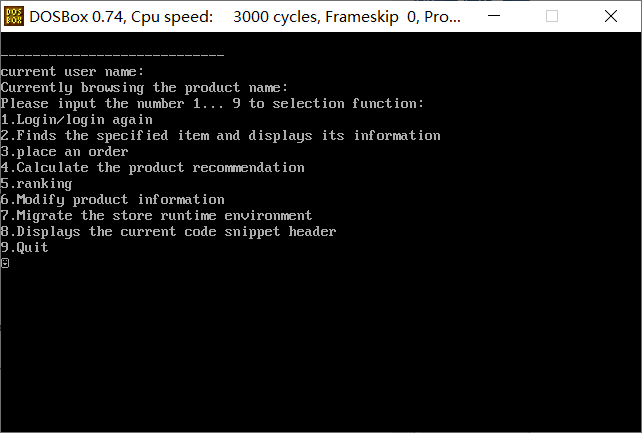


图1.5 主界面图

3. 要求1，登录或者重新登录。如果是顾客直接输入回车，回到主界面。如果是老板，输入用户名，利用循环进行字符串匹配，如果匹配失败返回主界面，成功就把ATUH置1.字符串比对时，使用CMP指令逐个字符比对。

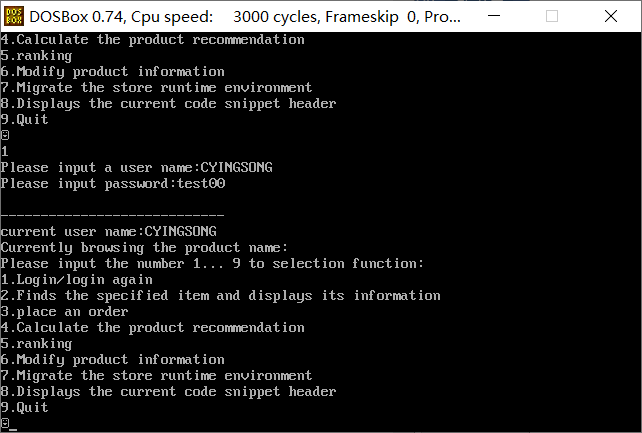


图1.6 用户登录成功图

4. 要求2，查看商品。使用双重循环来查找商品，内循环比对字符串，外循环将寄存器DI指向增加21个字节，来指向下一个商品。观察得知，DB类型一个字节，DW类型两个字节，DATA段存储是连续存放的，可以使用变址寻址来定位数据。

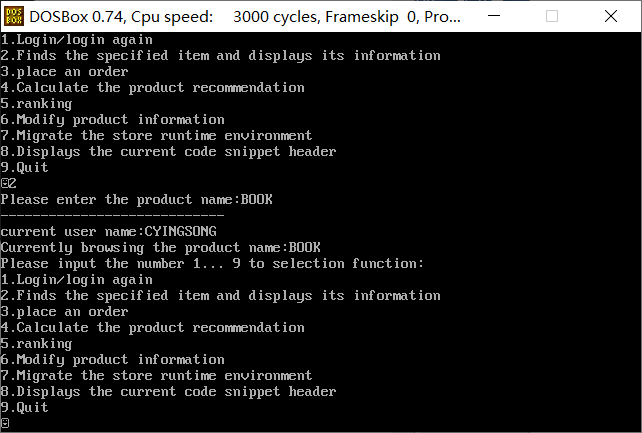


图1.7 查找商品成功图

5. 要求3，下单。首先检查GOOD字段是否’$’,不是就在查找商品，并使用变址寻址来检查商品的剩余数目。INC指令把已售出数目加一，使用WORD PTR 17[DI]按照字类型来增加。然后调用4功能计算所有商品的推荐度。使用一个标号在下一个语句的位置，把标号的偏移地址送入SI中。最后4功能的时候JMP SI回去。实现类似子程序的调用。

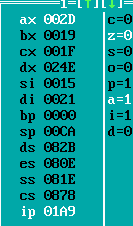


图1.8 AX中存储该商品剩余数目

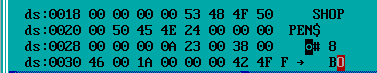


图1.9 已售数目增加1对比图

6. 要求4， 计算所有商品的推荐度，把128分别乘以两部分避免会出现除法结果是0，然后除法前把DX置0，防止出现溢出。最后JMP SI。

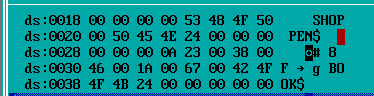
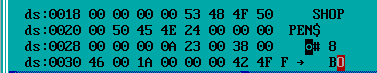


图1.10 计算所得推荐度存入存储区图

7. 要求5，把CS段首址转换成十六进制输出。使用shr移位运算和AND按位逻辑运算，四位四位的取出值，然后按照是数字和字母分别加上合适值转成ASCII码，输出字符。

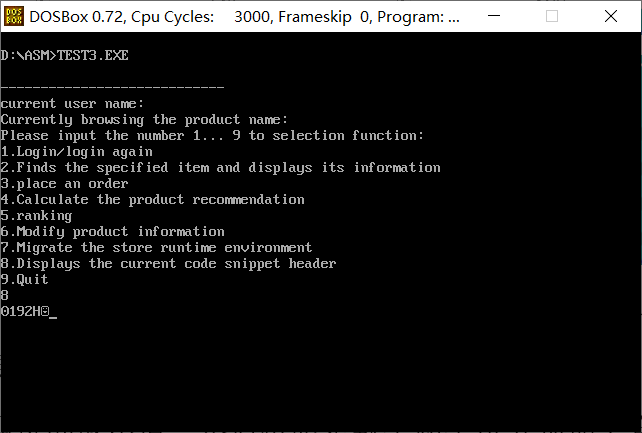


图1.11 输出CS首址

## 小结

直接上手构建一个系统在开始的时候确实一时手足失措，但是有其他类型的编程经验，首先把整体构成用其他语言或者流程图框架图写出来，然后使用条件转移指令来构建汇编系统的框架，并且使用宏指令，大大简化了程序，增加了可读性，运算方面要仔细考虑每个寄存器，考虑不同的寻址方式和地址的值。

虽然写的很难，但是最终还是写出来了。纸上得来终觉浅，得知此事要躬行。进行实验可以非常快的熟练掌握各种汇编中的细节和技巧。

汇编相较于高级语言确实在可读性和书写的难度上比较劣势，但是另一方面，贴近硬件的指令也大大减少了高级语言里为了书写方便不必要的调用和维护，在速度上占有优势。而这次的收获主要就是掌握了汇编基本语法包括程序流程的写法转移指令和寻址的使用以及宏的使用。

# 程序优化

## 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。

## 实验内容

**任务2.1 观察多重循环对CPU计算能力消耗的影响。**

请通过适当修改任务1.4的程序，完成如下研究：

1.请描述并实现对一段代码的执行时间进行测量的方法。该方法应能观察到程序中的一条指令发生修改时，程序完成同样功能时的执行时间的变化。

2.通过在不同软硬件运行环境下运行同一个程序，观察程序执行时间是否会随之发生变化。

对于任务1.4，现假设在双十一零点时，SHOP网店中的“Bag”商品共有m件，有m个顾客几乎同时下单购买了该商品。请模拟后台处理上述信息的过程并观察执行的时间。

上述场景的后台处理过程，可以理解为在同一台电脑上有m个请求一起排队使用任务1.4的程序。为了观察从第1个顾客开始进入购买至第m个顾客购买完毕之间到底花费了多少时间，我们让任务1.4的“下订单”调整后的代码重复执行m次，通过计算这m次循环执行前和执行后的时间差，来感受其影响。

调整后的下订单的描述：

【前提条件】用户查询到了商品“Bag”，在主菜单界面下选择了“3”对应的下订单功能。

《下订单》

【此处可插入计时开始、m次循环开始的控制代码】

（1）判断当前浏览商品是否有效（GOOD不为空），若有效，判断其剩余数量是否为0，不为0则将已售数量加1，重新计算所有商品的推荐度。

【此处进行m次循环的判断，循环未结束则继续执行“（1）”，结束时计时结束】

返回主菜单界面

（2）若无效或剩余数量为0，则提示错误，回到主菜单界面。

通过在不同的计算机硬件、不同的虚拟机环境、以及调试工具TD等下运行该程序，观察环境对程序执行时间的影响。

**任务2.2 对任务2.1中的汇编源程序进行优化**。

优化工作包括代码长度的优化和执行效率的优化，本次优化的重点是执行效率的优化。请通过优化m次循环体内的程序，使程序的执行时间尽可能减少10%以上（注意，在编写任务2.1的程序时，尽量不要考虑代码优化的问题）。

注意：（1）由于本课程关注的是指令级别的影响，因此，请大家不要只考虑纯算法级别的优化。（2）在优化思想的描述中，首先要对任务2.1中可能需要优化的做法或程序片段进行分析说明，然后提出自己的优化思路。（3）对于优化问题，其实验结果的记录与分析中，必须有相同条件下优化前后的数据对比。

## 任务2.1实验过程

### 设计思想

任务2.1的要求是对一段程序运行时间进行测量，可以使用老师提供的子程序文件。并且因为子程序测量的单位是ms，为了让输出更加明显，是m和n的值都为10000，且将‘BAG’商品放在最后，以增加其搜索时间。

### 流程图

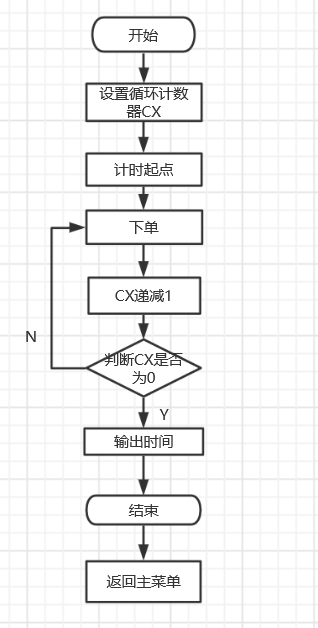


图2.1 下单流程图

### 源程序

; 2750

Order: ;下订单，并记录时间

MOV AX,0

CALL TIMER

;MOV AX,0

MOV CX,10000 ;使用CX做计数器

;判断GOOD是否为空

MOV AL,'$'

CMP GOOD,AL ;检查GOOD的第一位是不是'$'

;CMP GOOD,0

JNE E

WRONG:

WRITE NOGOODS

;JMP TIMEFINAL

JMP START

E: ;寻找到当前浏览商品的地址,如果不是无效，肯定会找到地址

;WRITE B

LEA DI,GA1-M

LEA SI,GOOD

;MOV DI,GOOD

;MOV CX,N+1 ;循环次数

MOV DH,'$'

OUTLOOP:

ADD DI,M ;每次移动M个字节

LEA BX,[DI]

INLOOP: ;检查用户名

MOV DL,[BX]

CMP [SI],DL

JNE OUTLOOP ;匹配失败，跳出内循环

INC SI

INC BX

CMP [BX],DH

JNE INLOOP

;检查剩余数目是否为0

MOV DX,15[DI] ;取出进货总数

MOV BX,17[DI] ;取出已经销售的数量

SUB DX,BX

JZ WRONG ;如果售完就返回

INC WORD PTR 17[DI]

LEA SI,OFFSET FUNC

JMP Calculate

FUNC:

DEC CX

;CMP CX,0

JNE E

TIMEFINAL:

MOV AX, 1

CALL TIMER ;终止计时并显示计时结果(ms)

;JMP INPUT

JMP START

Calculate: ;计算商品推荐度

;推荐度=(（进货价/实际销售价格+已售数量/（2\*进货数量）\*128)

; = (进货价\*1280/（销售价\*折扣）+64\*已售数量/进货数量

PUSH AX

PUSH BX

PUSH CX

PUSH DX

PUSH SI

PUSH DI

LEA DI,GA1-M

MOV CX,N

Cal\_Re:

ADD DI,M

MOV BX,10[DI] ;取出折扣

MOV AX,13[DI] ;取出销售价格

MUL BX

MOV BX,AX ;BX中是(销售价\*折扣）

MOV AX,11[DI] ;取出进货价

MOV DX,1280

MUL DX ;现在AX中是进货价\*1280

MOV DX,0

DIV BX

MOV Temp,AX ;把第一部分结果存入Temp

MOV AX,17[DI] ;取出已经销售的数量

MOV DX,64

MUL DX ;AX中是64\*已售数量

MOV BX,15[DI] ;取出进货总数

MOV DX,0 ;要将DX清零再进行字除法，防止溢出

DIV BX ;AX中存储的是第二部分的值

MOV BX,Temp

ADD BX,AX

MOV WORD PTR 19[DI],BX

LOOP Cal\_Re

POP DI

POP SI

POP DX

POP CX

POP BX

POP AX

JMP SI

## 任务2.2实验过程

### 设计思想

任务2.2要求对程序进行优化，减少程序运行时间。具体想法是

1. 把乘除指令转换成移位指令、加法指令等；
2. 其内循环体（推荐度计算）中每减少一条指令，就相当于减少了m\*n条指令的执行时间，需要仔细斟酌；
3. 在寻址方式中尽量把16位寄存器换成32位寄存器，能有更多的机会和技巧提高指令执行效率。

### 具体实现细节

(1) 在推荐度计算中为了保护和恢复现场，把所有的寄存器入栈了，但是分析只有CX寄存器需要被保护，所以可以把其他寄存器的入栈出栈命令删除

(2) 在查找商品的时候返回商品名，需要在下单时循环比对查找当前商品的地址，而直接返回商品首地址，可以节省很多循环指令

(3) 在内循环（推荐度计算）内，可以尽可能的减少指令，并且使用移位代替乘法

### 源程序

更改部分：

;GOOD DB '$',9 DUP(0) ;当前浏览商品名称

GOOD DW 0000H

GA1 DB 'PEN$',6 DUP(0),10 ;商品名称和折扣

DW 35,56,100,25,?

;GA2 DB 'BAG$',6 DUP(0),10

; DW 35,56,10000,0,?

;GA3 DB 'BOOK$',5 DUP(0),9

; DW 12,30,25,5,?

GAN DB N-2 DUP('TempValue$',8,15,0,20,0,30,0,2,0,?,?)

GA2 DB 'BAG$',6 DUP(0),10

DW 35,56,10000,0,?

FindItem: ;查找指定商品并显示其信息

;提示用户输入商品名称

CRLF

WRITE INPUTGOOD

;用户输入商品名称

READ in\_good

;在商店中寻找是否存在该商品

LEA SI,in\_good+2 ;SI中存储用户的输入

LEA DI,GA1-M ;DI存储商品信息首址

;每个商品占21个字节，一共有N个商品

MOV CX,N+1 ;循环次数

MOV AL,'$'

LOOP3:

DEC CX

CMP CX,0

JNE D ;没有找到商品，返回主界面

WRITE NOTFIND ;输出没有找到商品

JMP START

D: ADD DI,M ;每次移动M个字节

LEA BX,[DI]

LOOP4: ;检查用户名

MOV DL,[BX]

CMP BYTE PTR [SI],DL

JNE LOOP3 ;匹配失败，跳出内循环

INC SI

INC BX

CMP [BX],AL

JNE LOOP4

;找到商品把信息记录到GOOD中

MOV BX,0

MOV DL,'$'

COPY: ;把商品名复制进GOOD中

MOV GOOD,DI

JMP START

Order: ;下订单，并记录时间

MOV AX,0

CALL TIMER

MOV CX,10000 ;使用CX做计数器

;判断GOOD是否为空

CMP GOOD,0

JNE E

WRONG:

WRITE NOGOODS

JMP START

E: ;寻找到当前浏览商品的地址,如果不是无效，肯定会找到地址

MOV DI,GOOD

;检查剩余数目是否为0

MOV DX,15[DI] ;取出进货总数

MOV BX,17[DI] ;取出已经销售的数量

SUB DX,BX

JZ WRONG ;如果售完就返回

INC WORD PTR 17[DI]

LEA SI,OFFSET FUNC

JMP Calculate

FUNC:

DEC CX

;CMP CX,0

JNE E

TIMEFINAL:

MOV AX, 1

CALL TIMER ;终止计时并显示计时结果(ms)

;JMP INPUT

JMP START

Calculate: ;计算商品推荐度

;推荐度=(（进货价/实际销售价格+已售数量/（2\*进货数量）\*128)

; = (进货价\*1280/（销售价\*折扣）+64\*已售数量/进货数量

PUSH CX

LEA EDI,GA1-M

MOV CX,N

Cal\_Re:

ADD EDI,M

MOV AX,13[EDI] ;取出销售价格

MUL WORD PTR 10[EDI]

MOV BX,AX ;BX中是(销售价\*折扣）

MOV AX,11[EDI] ;取出进货价

MOV DX,1280

MUL DX ;现在AX中是进货价\*1280

MOV DX,0

DIV BX

MOV BX,AX ;把第一部分结果存入Temp

MOV AX,17[EDI] ;取出已经销售的数量

;使用移位代替乘法，乘64代表左移5位

SHL AX,5

MOV DX,0 ;要将DX清零再进行字除法，防止溢出

;AX中存储的是第二部分的值

DIV WORD PTR 15[EDI]

ADD AX,BX

MOV WORD PTR 19[EDI],AX

LOOP Cal\_Re

POP CX

JMP SI

### 实验记录与分析

任务2.1：

* 改变循环程序的设计和循环次数，对CPU资源消耗的影响有多大？

很大，不同结构不同循环次数的程序，其运行指令的数目不同，对于CPU资源的消耗是不同的。原始程序运行2760ms，但是在推荐计算中减少了一条指令，相当于在内循环中减少一条指令，可以减少30ms的运行时间。

* 内循环体中若有信息显示的代码（比如2号或9号功能调用），程序执行时间会有多大影响？

会大大增加程序运行的时间。因为输出功能是比较耗时的，而且缓冲区的输出有时还需要排队，是串行处理的。

* m和n的最大值受到哪些限制？当商品种类的数量很多时（比如10000以上），如何处理一个数据段空间不够的问题？

m,n的值要在字段表示整数范围内，可以设置已售数量第一次等于m则将已售数量置0，并且把n重新设置初值。设置一个标识的变量来记录置0的个数。

任务2.2：

汇编语言程序的优化可以从哪些方面进行？总结不同类别的优化措施对效率的影响程度。

1. 减少循环内部的指令数

2. 简化推荐度的计算公式

3. 乘除法尽量用移位指令代替

4. 16位寄存器改用32位寄存器

简化推荐度的计算公式是对程序运行时间影响最大的，因为其处理内循环，一条指令要运行m\*n次。而16位寄存器改用32位寄存器并不能马上节省时间，只是更大的寻址空间可以有更多的机会去优化程序。

* 哪些指令是需要优化的关键性指令？

乘除法指令，还有循环内的指令（特别是推荐度计算）

循环体中用子程序调用与不用子程序调用会有多大影响？

调用子程序，需要传递参数和保护恢复参数，有很多的寄存器操作和堆栈操作，而不用子程序调用，直接使用代码的可扩展性降低，但是减少了很多指令，运行时间减少。

* 利用32位寄存器书写寻址方式具有较多的灵活性。你是否有通过此途径优化指令语句的例子

可以减少使用寄存器的个数，把32位寄存器当作两个16位寄存器使用。

程序段循环10000次，优化率为36%。实验优化结果如下

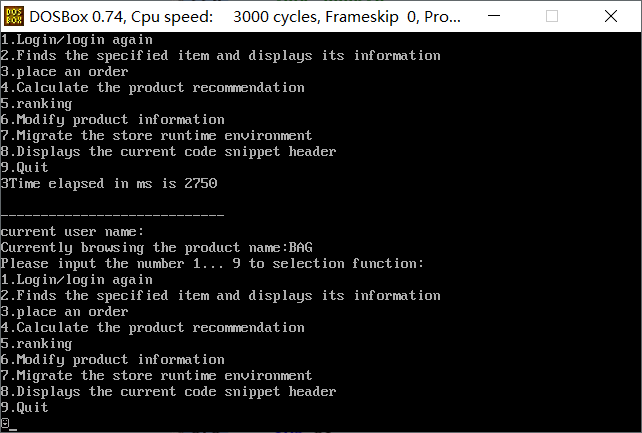


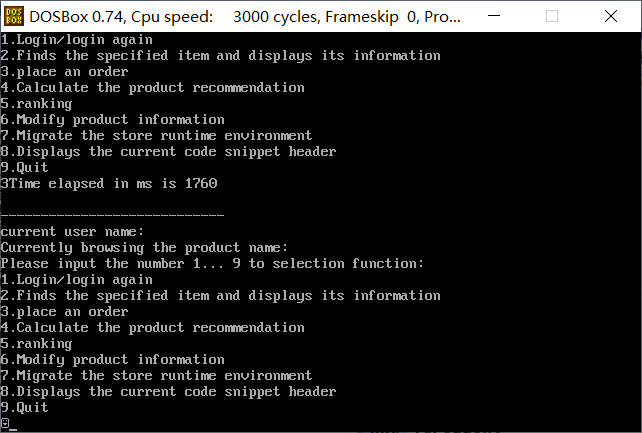
图2.2 优化前运行时间2750ms

图2.3 优化前运行时间1760ms

## 小结

使用计时来关注程序指令级别的优化，让我更加深刻的了解了指令操作。了解不同指令操作的区别，比如乘法如果是2的幂次做乘数，等价于移位操作，而移位操作显然更加简单。还有有些指令是不必要的，可以减少。比如在DEC CX,JZ MAIN中就不应该插入CMP CX,0因为递减后标志位中已经可以得知CX是否为0.还有循环结构的改变，使用LOOP指令可以不需要自己递减计数器。使用变量名加计数器的变址寻址，也可以不需要使用寄存器存址，减少了指令也节省了寄存器。还有运算公式的化简，指令格式的改变都可以加快程序运行的速度。

总之，让我注意到了更加精细，更加底层的编程思想。和高级语言的算法优化在首位，对于具体每个步骤在机器中的实现并不是很感兴趣相比，汇编更加关注机器中的事。

# 模块化程序设计

## 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3.1 宏与子程序设计

1.把网店商品信息管理系统的子功能尽量改成子程序的方式实现。

2.将任务1.4中重复使用的程序段尽量改成宏（至少定义一个宏指令）或子程序的方式来实现。

3.在网店商品信息管理系统中新增如下功能：

1）在“2.查找指定商品并显示其信息”的功能中，实现商品信息的显示功能。即：在找到指定商品之后，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。

2）实现“6.修改商品信息”的具体功能。

只有老板登录后可以使用本功能。若当前浏览商品无效，则返回；若有效，则按照：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，回到主菜单界面。

4.将本次新增功能的子程序放到另外单独的模块中，按照模块化程序设计的方法搭建系统。

任务3.2 C语言混合编程

将网店商品信息管理系统中菜单界面的功能用C语言实现，其他子功能可以仍采用汇编语言实现。在C语言程序中调用汇编语言子程序

## 任务3.1实验过程

### 设计思想

（1）将原来的跳转地址选择不同功能改成子程序调用，比较输入然后转入各个子程序中即可。而子程序只需要保护AX以防止输入被修改，使用RET返回调用处

（2）将字符串比对改成宏指令实现，参数有五个，分别是两个字符串首址，第一个串长度，比对失败跳转的地址和用来中转的寄存器。使用宏指令可以增加程序的可读性

（3）新增功能涉及输入一个数字并检查其合法性，显示十进制数和修改。

（4）输入数字并检查有无非法字符使用宏指令完成，思想就是逐个字符读入，如果是非法字符写入标志退出，不然就将已输入的值乘10加上当前输入的值，直到输入回车输入结束。

（5）显示十进制数调用F2T10模块的F2T10函数，把AX中的数转成十进制并输出

（6）使用宏指令完成多个项目的显示和修改，让程序更加简便

（7）使用模块构建系统，模块之间共享一定的变量和需要用到的函数，存储空间使用PARA模式，把同名的代码段，数据段都合并起来。同时写一个额外的test.asm文件来测试各个模块的通讯和功能有无问题，通过测试再接入系统中。

### 模块调用关系

一共有三个模块，分别是主模块test4.asm,两个子模块F2T10.asm,Mo.asm.其中主模块调用Mo模块来修改值，其中Mo引用商品首址，和当前浏览商品还有登录状态变量，而在Mo中再调用F2T10模块，显示当前浏览商品的值和修改后v的值。

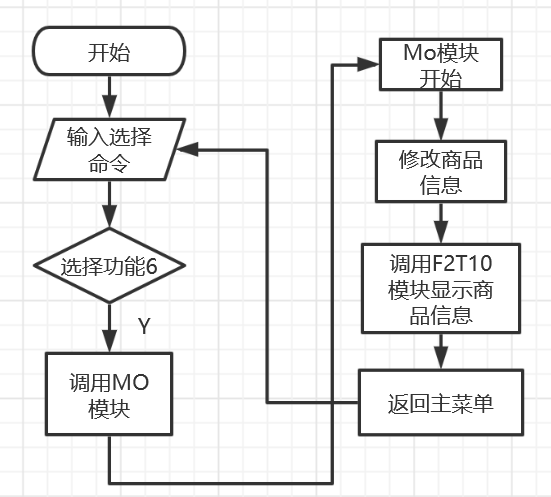


图3.1 调用关系流程图

### 源程序

新增的功能6

EXTRN ATUH:BYTE,GOOD:BYTE,GA1:BYTE

EXTRN F2T10:FAR

PUBLIC ModifyFun

INCLUDE IO\_M.LIB

Modi MACRO A,B

;A是输出信息首址，B是存储位置

LOCAL P1

LOCAL P2

P1:

WRITE A

MOV AX,B

CALL F2T10

WRITE Separator

INPUTNUM

CMP AL,0

JZ P1

CMP BX,13

JE P2

MOV BX,DX

WRITE A

MOV WORD PTR B,BX

MOV AX,BX

MOV DX,0

CALL F2T10

CRLF

P2:

ENDM

.386

STACK SEGMENT USE16 PARA STACK 'STACK'

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16 PARA PUBLIC 'A1'

discount DB 'discount:$'

Separator DB '>>$'

ModifyWrong DB 'ModifyWrong!$'

M EQU 21

purchase DB 'purchase:$'

Sale DB 'Sale:$'

Total DB 'Total:$'

Saled DB 'Saled:$'

Recommaned DB 'Recommaned:$'

T DB '0000$'

DATA ENDS

CODE SEGMENT USE16 PARA PUBLIC 'B1'

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

ModifyFun PROC FAR

;子程序：修改商品信息

;入口参数： ATUH,GOOD,GA1

;无出口参数

;只有ATUH为1才能使用该功能

;若GOOD中无商品则返回

;不然，逐一显示折扣，进货价，进货总数，使用DisNum

;输入新的值，并检查，重新修改和显示该项。

;MOV AL,ATUH

;MOV AH,0

;CALL DisNum

MOV AL,ATUH

CMP AL,0

JZ ModifyWrong1

;WRITE discount

MOV AL,'$'

CMP GOOD,AL

JE ModifyWrong1

;寻找到当前浏览商品的地址,如果不是无效，肯定会找到地址

LEA DI,GA1-M

LEA SI,GOOD

MOV DH,'$'

OUTL:

ADD DI,M ;每次移动M个字节

LEA BX,[DI]

INL: ;检查用户名

MOV DL,[BX]

CMP [SI],DL

JNE OUTL ;匹配失败，跳出内循环

INC SI

INC BX

CMP [BX],DH

JNE INL

;显示并修改折扣

Discount1:

WRITE discount

MOV AX,0

MOV AL,10[DI]

CALL F2T10

WRITE Separator

;输入一个数字

INPUTNUM

;检查数字

CMP AL,0 ;输入出错，重新显示

JZ Discount1

CMP BX,13 ;输入直接就是回车，不需要修改

JE Purchase1

CMP DL,10

JA Discount1

CMP DL,1

JB Discount1

MOV BL,DL

WRITE discount

MOV BYTE PTR 10[DI],BL

MOV AL,BL

MOV AH,0

MOV DX,0

CALL F2T10

CRLF

;显示并修改进货价

Purchase1:

Modi purchase,11[DI]

Modi Sale,13[DI]

Modi Total,15[DI]

;Modi Saled,17[DI]

;Modi Recommaned,19[DI]

JMP BACK4

ModifyWrong1:

WRITE Modifywrong

CRLF

BACK4:

RET

ModifyFun ENDP

;DisNum PROC

;将AX中的数以十进制形式输出

;mod 10取得末位，并转换为ASCII码，然后反序存储

CODE ENDS

输入数字的宏指令：

INPUTNUM MACRO

;AX中存储的是标志位，如果是输入有问题就AX中是0，DX中存储的是输入的数字

LOCAL P1

LOCAL P2

LOCAL P3

LOCAL EN

MOV DX,0

MOV BX,13 ;如果直接输入的就是空格，BX就是13

P1:

MOV AH,1 ;输入一个字符

INT 21H

CMP AL,13 ;如果输入的是一个回车，输入终止，返回

JE P2

;如果输入的不是数字，输出异常

CMP AL,'0'

JB P3

CMP AL,'9'

JA P3

SUB AL,'0' ;把输入转成数字

MOV CL,AL

MOV CH,0

MOV AX,DX ;DX中存储的是之前的值

MOV BX,10

MUL BX ;原来的值乘10，加上刚刚输入的值

ADD AX,CX

MOV DX,AX

JMP P1

P3:

MOV AL,0

JMP EN

P2:

MOV AL,1

EN:

ENDM

### 实验记录和分析

（1）程序运行需要使用LINK工具将不同模块连接在一起。

（2）PARA模式是将不同模块的同名段连接在一起，NEAR调用是在段内-128到127之间的跳转，如果CODE段都连接在一起是可以直接调用的（代码不长）

下图展示了数据段的合并：

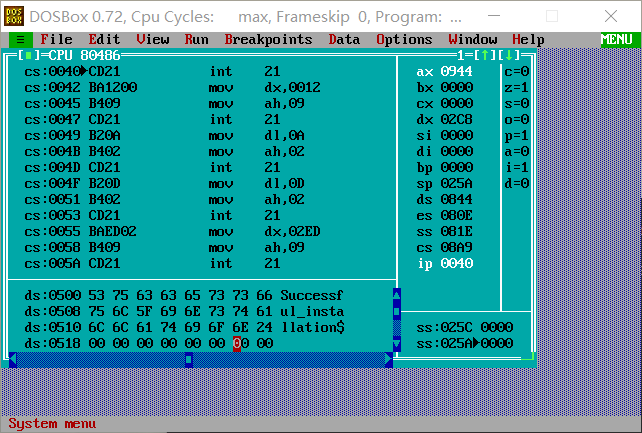


图3.2 主模块数据区

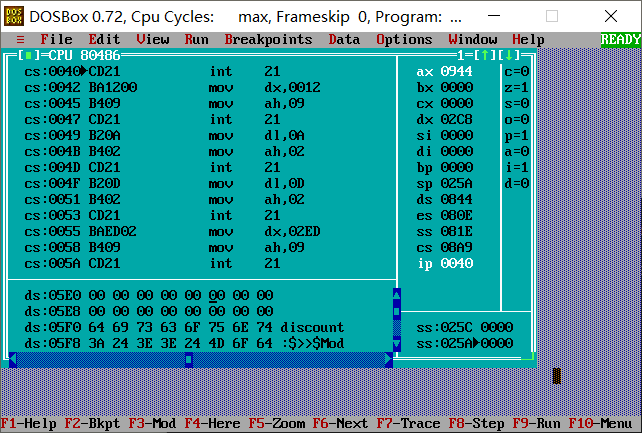


图3.3 Mo模块数据区

（3）不同模块之间公共变量，使用PUBLIC和EXTERN连接，类似与C语言的声明和引用。如果要公用宏，需要编写宏库文件，然后同时INCLUDE

（4）宏指令可以使用LOCAL命令来指定标识，汇编的时候会自动编号不会重叠。

（5）子程序参数可以使用寄存器传入，可以使用特定约定的变量传入还可以使用堆栈传入，但是堆栈要注意返回前释放，取得返回的地址。而且在子程序中要注意保护现场，对必要的寄存器入栈保存值。

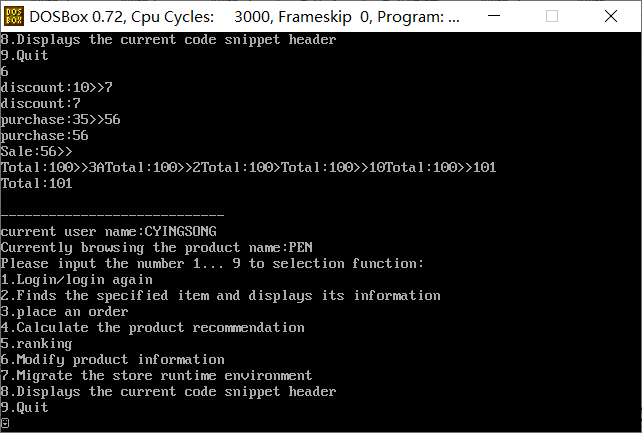


图3.4 使用模块的修改程序

## 任务3.2实验过程

### 设计思想

1. 使用dosbox+BC环境编程，使用tasm对asm文件汇编，使用BC集成环境创建工程对混合文件进行汇编链接并生成可执行文件。
2. 了解C语言函数的参数传递机制。
3. 体会混合编程中数据内存的相互调用。
4. 在C语言程序中嵌入汇编语言。
5. 尝试不合适的插入汇编语句，修改寄存器，观察C语言可能出现的错误。
6. 观察混合编程代码数据的偏移
7. 体会C语言和汇编的效率差别。
8. 观察汇编程序和Ｃ语言程序的变量命名的特殊规则。

### 实验方法说明

1. 使用C语言完成菜单和控制流程，调用汇编函数来实现某些功能
2. 使用C语言来完成登录函数，并传入参数ＢＮＡＭＥ和ＢＰＡＳＳ。了解C语言函数的参数传递机制
3. 在C语言中调用汇编中的内存数据，来完成菜单的输出。
4. 使用ａｓｍ关键字嵌入汇编语句，也可以用来让C语言程序出错。
5. 使用ＢＣ中的调试工具来观察数据段和代码段的偏移。
6. 使用ＢＣ中的工具查看C语言翻译成的汇编代码，并与手写的汇编做对比。

### 实验记录和分析

1. 在汇编程序中，如果需要被Ｃ语言调用的数据，命名时需要加上＿｛下划线｝，C语言中如果参数小于4个会默认由四个寄存器保存参数，大于四个会依次入栈传递参数。综合之后确实会有数据代码段的偏移。
2. 使用C语言调用汇编程序输出菜单效果如下：

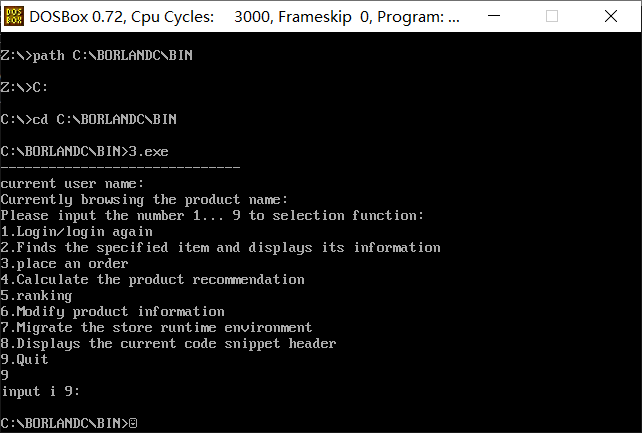


图3.5 菜单输出效果

1. 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，因为C语言翻译成汇编也是使用一些寄存器来达到比如赋值之类的目的，如果随意修改寄存器的值，很可能让程序出现错误。
2. 使用C语言完成函数登录，传递了2个参数，并调用和修改了汇编程序中内存值。效果如下：

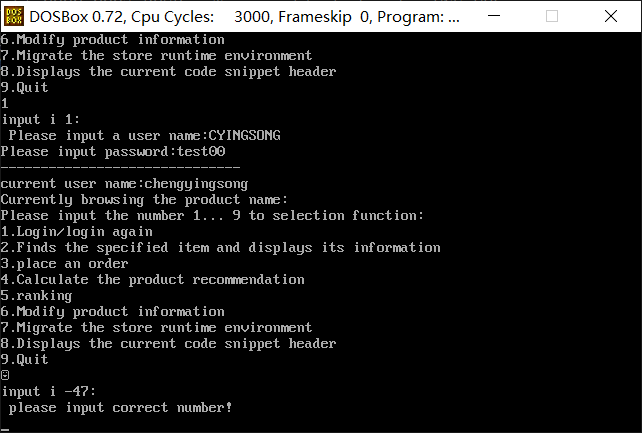


图3.6 登录成功

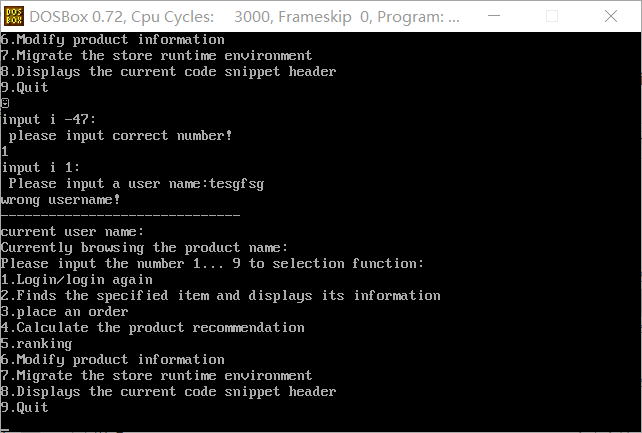


图3.7 登录失败

1. C语言的效率要低于汇编程序。一个i= i- 304，需要首先赋值给寄存器，然后加上一个模值，然后再赋值回去。汇编可以直接进行减法。

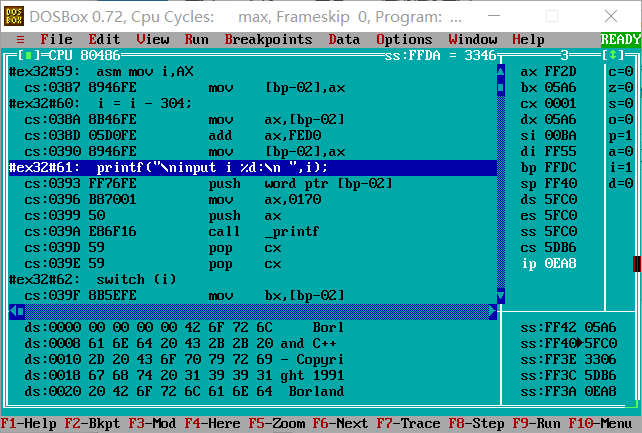


图3.8 效率对比

## 小结

模块化程序设计和C语言混合编程都涉及到了不同文件之间调用函数和调用一些数据内存，同时也了解了多文件编译重定位链接并生成可执行文件的流程，在C语言混合编程中，还花了时间在配置和了解环境上，顺便还了解了一下脚本文件的用法。汇编的模块化程序设计和高级的模块化程序设计并无很大不同，C语言混合编程就需要按照C语言传递参数的规则来书写函数。还需要注意变量和段的命名问题。

这次实验更加深入的了解了汇编文件的翻译和执行过程，对整个汇编的逻辑有了更深的认识。

# 中断与反跟踪

## 实验目的和要求

(1) 熟悉I/O访问，BIOS功能调用方法；

(2） 掌握中断矢量表的概念；

(3) 掌握实方式下中断处理程序的编制与调试方法；

(4) 进一步熟悉内存的一些基本操纵技术；

(5) 熟悉跟踪与反跟踪的技术以及相关的反汇编工具；

(6) 提升对计算机系统的理解与分析能力。

## 实验内容

任务4.1：实现“7.迁移商店运行环境”的功能。

在操作系统和虚拟机中，经常要进行内存的调度迁移。这里的迁移运行环境的含义是指将“网店商品信息管理系统”当前的数据段、堆栈段、代码段切换到另外一套数据段、堆栈段和代码段中去，并保证切换前后程序的状态一致（比如，切换前正在浏览某个商品的信息，切换后也应保留该浏览状态）。本次实验只要求切换任务3.1程序的堆栈段。切换的操作是在指定时间下，由中断服务程序完成。

另，为便于观察，需要调整“8.显示当前代码段首址”的功能为：“8.显示当前段寄存器SS的内容”（即按照16进制方式显示这个段寄存器的内容）。

“7.迁移商店运行环境”具体实现的提示如下：

【前提条件】在网店商品信息管理系统（任务3.1对应的程序）中新定义了一个与堆栈段STACK大小一样的数据段空间STACKBAK作为两个堆栈段相互切换的空间。

（1）设定切换的时钟时刻（可以通过界面输入，也可以在变量里直接给定）。为便于观察，可以只设定时钟的分钟和秒（甚至只设定秒），也就是当时钟走到任一小时的对应分钟和秒的时刻时就切换一次（如果只设定秒，那么，时钟走到任一分钟的对应秒的时刻就会切换一次）。

（2）检查中断服务程序是否被安装，已经安装的话就返回主菜单界面。

（3）设置当前堆栈段信息和待转移的堆栈段信息，安装中断处理程序（接管8号时钟中断），返回到主菜单界面。

中断处理程序的主要流程：

（1）执行原中断服务程序的功能（使用PUSH+CALL组合）。

（2）用IO指令从CMOS芯片中读取当前时钟的分钟和秒信息，判断是否与设定的分钟和秒时间相同（注意CMOS分钟和秒信息的编码为BCD码），不相同则中断返回；相同则执行（3）。

（3）获取必要的当前堆栈段信息和待转移的堆栈段信息，并记录本次转移操作的状态（用于下次转移时判断转移的方向是转移到STACK还是STACKBAK）。

（4）把当前堆栈段里的有效内容复制到新的堆栈段。

（5）切换堆栈段指针。

（6）中断返回。

**任务4.2：数据加密与反跟踪**

在任务4.1的网店商品信息管理程序的基础上，老板的密码采用密文的方式存放在数据段中，各种商品的进货价也以密文方式存放在数据段中。加密方法自选（但不应选择复杂的加密算法）。

可以采用计时、中断矢量表检查、堆栈检查、间接寻址等反跟踪方法中的几种方法组合起来进行反跟踪（建议采用不少于两种反跟踪方法，重点是深入理解和运用好所选择的反跟踪方法）。

为简化录入和处理的工作量，只需要定义三种商品的信息即可。

任务4.3：跟踪与数据解密

解密同组同学的加密程序，获取各个商品的进货价。

建议尽量使用到以下的技术：

1）利用静态反汇编工具（如SOFTICE, OLLYDBG等）将执行程序反汇编成源程序，观察源程序的特点。

2）利用二进制文件编辑工具，直接观察和修改执行文件中的信息（如老板名字信息等）。

3）动态跟踪调试，注意观察和跳过反跟踪的代码。

4）有余力的学生可以设计实现：(a)一个暴力猜解密码的程序；(b)接管键盘的中断服务程序，驻留该程序之后再运行网店商品信息管理程序，截取用户输入用户名之后的字符串信息，保存在指定内存中；退出网店商品信息管理程序之后，用TD去观察中断服务程序记录的字符串信息。

## 任务4.1实验内容

### 设计思想

1. 在代码区定义一些变量，比如转移方向和是否安装新中断程序的标识
2. 在功能7修改中断矢量表，使用35H和25H号调用，其中25H要修改DS为CODE，并且恢复是要重新设置为0.
3. 在新增的中断程序里转移堆栈数据，从Capaity到sp为有效内容，使用数据转移指令MOVSB转移数据，并切换堆栈段指针SS

### 流程图

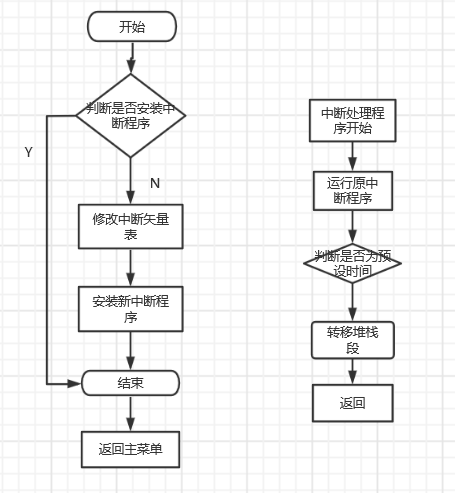


图4.1 安装中断程序流程图

### 源代码

CMP CS:Successful\_installation,1

JE Migrate\_end

MOV AX,3508H ;获取中段矢量

INT 21H

MOV OLD\_INT,BX

MOV OLD\_INT+2,ES

MOV DX,OFFSET NEW08H

MOV BX,CODE

MOV DS,BX

MOV AX,2508H ;设置新的中断矢量

INT 21H

MOV BX,DATA

MOV DS,BX

MOV CS:Successful\_installation,1

MOV BX,SS

MOV STACK1,BX

;OUT1 '1'

;WRITE is\_Successful\_installation

;CRLF

JMP START

Migrate\_end:

;OUT1 '2'

WRITE al\_Successful\_installation

CRLF

JMP START

NEW08H PROC FAR

PUSHF

CALL DWORD PTR CS:OLD\_INT

MOV AL,0 ;读取秒的信息

OUT 70H,AL

JMP $+2

IN AL,71H

;获取当前时间

MOV BL,AL

MOV BH,BL

AND BL,0FH

SHR BH,4

ADD BX,3030H

XCHG BH,BL

CMP BX,CS:Time

JNZ O

DEC CS:COUNT

JNZ O

Trans:

;转移堆栈段

MOV CS:COUNT,18

STI

PUSHA

PUSH DS

PUSH ES

MOV AL,Positive\_Trans

ADD AL,30H

OUT1 AL

CALL DisPlay

CMP Positive\_Trans,1

JNZ Negetive\_trans

MOV Positive\_Trans,0

;将STACK转入STACKBAK

MOV AX,STACK1

MOV DS,AX

MOV AX,OFFSET STACKBAK

MOV ES,AX

MOV SI,200

MOV DI,200

MOV CX,200

SUB CX,SP

ADD CX,2

STD

REP MOVSB

CLD

MOV AX,DATA

MOV DS,AX

MOV ES,AX

MOV AX,OFFSET STACKBAK

MOV SS,AX

JMP OU

Negetive\_trans:

MOV Positive\_Trans,1

;将STACKBAK转入STACK

MOV AX,OFFSET STACKBAK

MOV DS,AX

MOV AX,STACK1

MOV ES,AX

MOV SI,200

MOV DI,200

MOV CX,200

SUB CX,SP

ADD CX,2

STD

REP MOVSB

CLD

MOV AX,DATA

MOV DS,AX

MOV ES,AX

MOV AX,STACK1

MOV SS,AX

OU:

POP ES

POP DS

POPA

O:

IRET

NEW08H ENDP

### 实验记录和分析

IO:

1. CMOS里的时间信息是按照压缩BCD码的形式存放的，举例说明压缩BCD码的格式是什么？

BCD码是使用4位二进制来表示0-9的整数，把十进制数字按照每一位存储，比如秒“35”，3的BCD码是0011,5的BCD码是0101，所以35的BCD存储就是00110101.

中断矢量表：

1. 打开TD之后，如何在数据区切换到中断矢量表所在内存区域？

直接查看ES区

1. 如何计算某个中断入口在中断矢量表内的偏移地址？

N号中断[n\*4]和[n\*4+2]

1. 程序中如何使用系统功能调用获取中断入口地址？可以在TD中录入指令语句或编写完整程序来尝试。

使用MOV AX,3508H, INT 21H.指令

1. 用TD把中断矢量表里的中断矢量的值随意改成其他值（或改成其他中断的中断矢量）会有什么现象发生？（比如修改21H,1H,3H等的中断矢量，修改后再做些其他操作，比如打开一个执行程序等）

没有恢复中断矢量表会出现故障。

1. 对整个中断矢量表中的入口地址进行观察，是否能看出什么特点？

每个中断占4个字节，入口地址是连续存放的。

中断服务程序：

1. 为避免未调试好的中断服务程序接管时钟、键盘中断时使系统时间或键盘操作失灵，可以先用其他方法，请给出你采用的“其他”调试方法的具体描述并实施一下。

直接把堆栈转移写成子程序，把接管中断的程序中只写一个简单输出，两方面都调试好之后再结合

1. 如何判断中断服务程序已经安装过了？

使用一个变量记录安装状态。还可以查看中断矢量表中的值是否是原来的值。

内存操纵：

1. 如果要迁移切换数据段和代码段，是否会有新的问题需要注意？

切换数据段指针和代码段指针。代码段不能直接改变CS寄存器，可以把偏移值和SS入栈，然后IERT。

1. 如果给你的是一片连续的存储空间，你如何将它们定义成合适的堆栈段、数据段和代码段（即给出段首址和分配的空间大小），以满足程序的相应段的迁移要求？

首先确定各段需要的总空间，分配各自的首址入CS，SS等寄存器，然后计算偏移值去获得总的地址。

堆栈转移效果图如下：

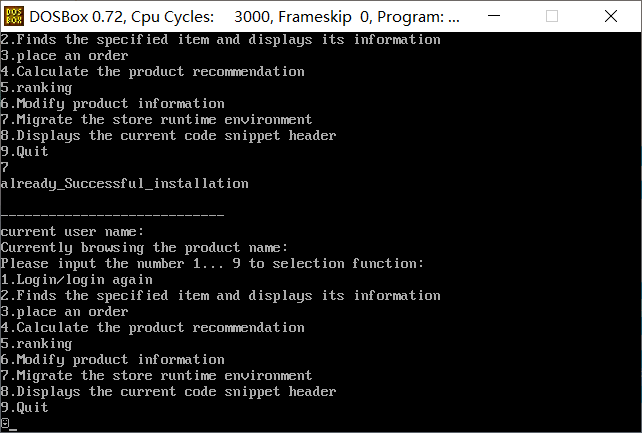


图4.2 安装新中断

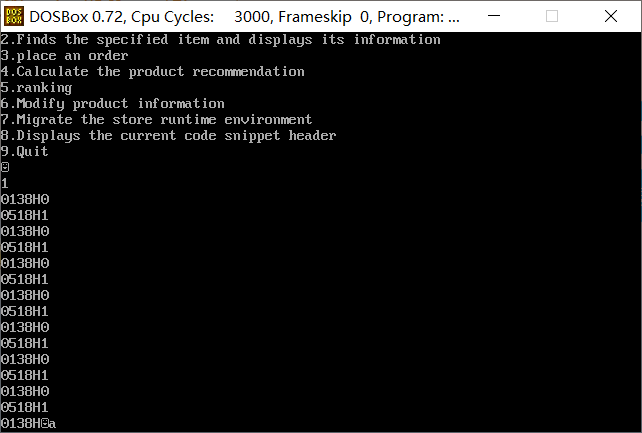


图4.3 堆栈转移

## 任务4.2，4.3实验内容

### 设计思想

* 1. 密文采用函数加密的方法加密，密码是每个字符减16H,然后乘以二 ，使用了计时、中断矢量表检查、堆栈检查、间接寻址方法来反跟踪。计时是查看指令运行时间的数量级，防止单步调试，二中断矢量表检查是防止中断，堆栈检查也是通过中断时会对堆栈进行改变来防止中断，间接寻址方法使用地址表和在在代码段写入一些无意义的数据定义都是混淆视线，让程序的可读性下降。
  2. 跟踪是采用反汇编的方式。观察代码并有时配合断点进行单步调试。对代码中的重点代码进行阅读，并尝试着暴力破解。

### 实验记录和分析

同组：汪先奇

1. 若密码是用明文存放在数据段中的，如何更快地获取密码？

使用TD直接观察数据段相应位置，或者在程序中编写额外的代码输出数据段相应位置。

1. 若商品进货价是用明文存放在数据段中的，如何更快地获取进货价？

将执行程序文件用二进制编辑工具打开，直接在文件里寻找所定义的商品信息

1. 如何对密码实现快速的暴力破解？（可以编写程序，也可以描述一下实现的具体思路）

密码有大写字母小写字母和数字，首先还是要使用反汇编观察代码，看出代码位数以及对密码的取值有一定的猜想，然后进行一定程度的枚举。如果没有输入次数的限制，就使用穷举法，把所有可能都试一遍。

1. 如何综合利用静态反汇编和动态反汇编的信息破解程序？（要使用反汇编工具得到汇编源程序，对其进行观察分析）

静态反汇编观察代码，动态反汇编还可以进行一定程度的单步调试。

1. 举例说明如何观察到程序中存在反跟踪的代码?举例说明如何应对反跟踪程序? （在记录分析里具体描述）

首先是熟悉常见的反跟踪方法，在观看反汇编代码时就可以有所预料。比如可以看出其使用了堆栈检查反跟踪。



4.4反汇编图

知道其反跟踪方法后就可以使用相应的方法应对，比如可以设置断点进行单步调试，在相应段不中断。

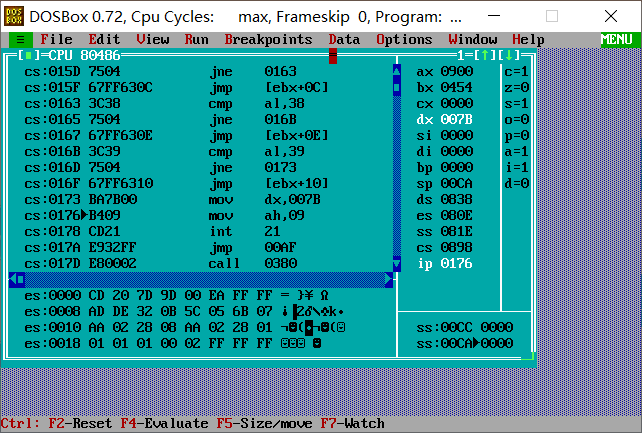


图4.5利用TD单步调试

1. 当存在修改中断矢量表的代码时，一般会先关掉中断（也即执行CLI指令）。如果不想因为关中断指令的出现让跟踪者容易判断出后续存在反跟踪代码，应如何设计修改中断矢量表的代码，达到不用关中断的目的？

使用INT 21H的设置中断功能。

1. 是否可以通过修改AUTH的值来达到获取进货价的目的？是否可以通过观察该程序计算推荐度的过程来获取进货价？

不可以，修改AUTH的值老板看的也是密文，同样计算推荐度是也是使用密文的。

## 小结

进行跟踪和反跟踪，使用了一些针对性的混淆技巧，总的来说，这是一个很灵活的东西，可以针对对方的手段做出应对。然后这些技巧其实也是从另外一个角度去了解汇编本身，只有对语言的运作和反汇编等方法了解清楚，才能正确的实现自己的目的。在实际的应用时也确有作用。

# WIN32编程

## 实验目的与要求

（1）熟悉WIN32/64程序的设计和调试方法；

（2）了解不同操作系统环境下开发工具的特点；

（3）熟悉宏汇编语言中INVOKE、结构变量、简化段定义等功能；

（4）进一步理解机器语言、汇编语言、高级语言之间以及实方式、保护方式之间的一些关系；了解16位段程序移植到32/64位段程序时需要注意的问题。

## 实验内容

任务5.1 编写一个基于窗口的WIN32/64程序，实现网店商品信息管理系统的部分功能。也即：以任务3.1为基础，将其部分功能移植过来，具体要求如下描述。

1编写一个基于窗口的WIN32/64程序的菜单框架，具有以下的下拉菜单项：

File Action Help

Exit Recommendation About

List Sort

点菜单File下的Exit选项时结束程序；点菜单Help下的选项About，弹出一个消息框，显示本人信息，类似图5.1所示。点菜单Action下的选项Recommendation、List Sort将分别实现计算推荐度或显示排序后的SHOP所有商品信息的功能（详见要求“2”的描述）。

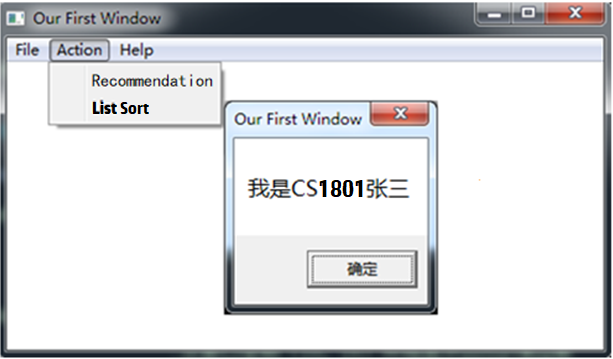


图5.1 菜单示例

2.要求采用结构变量存放商品的相关信息。商品数定义5种左右。

（1）点菜单项Recommendation时，按照任务3.1的方法计算所有商品的推荐度。

（2）点菜单项List Sort时，先对所有商品按照推荐度从高到低排序，然后按照排序结果在窗口中列出SHOP的所有商品的信息。具体显示格式自行定义，可以参照图5.2的样式（不要求用中文）。



图5.2 商品信息显示示意图

## 任务5.1实验内容

### 实验思路

1.配置WIN32编程环境，使用MASM编程

2 修改menu.rc文件，修改菜单栏选项以符合要求

3 使用结构变量存放商品信息

4 计算推荐度

5 对推荐度进行排序并输出

6 将写好的子程序调用写入相应的信息分发程序处。

### 排序步骤

1使用地址表存储每个商品的首地址

2 找出从当前循环变量置表尾代表的商品中推荐度最高的商品地址

3 将其与当前位置的地址交换

4 当前位置后移

5 如果移到了表尾终止，排序完成

6 按照地址表引用商品，然后输出。

思路就是使用选择排序，以地址表作为引用结构的指针，并交换指针而不是实际的信息位置。

### 实验记录和分析

1. 观察32/64位下调试工具与16位TD的异同。

都可以单步调试，观察数据区和一些寄存器的值。32/64位下可以使用源码级调试，可以使用更加强大的功能，直接在源码上进行调试。

2. 调试WIN32/64程序与16位段程序的主要差异是什么？

段的定义可以使用简化段定义，然后地址的位数是32位而不是16位。WIN32可以在windows系统中运行，而16位段在dosbox中运行。

3. 尝试使用一下汇编语言程序的源码级调试工具和方法，与非源码级调试做个对比。

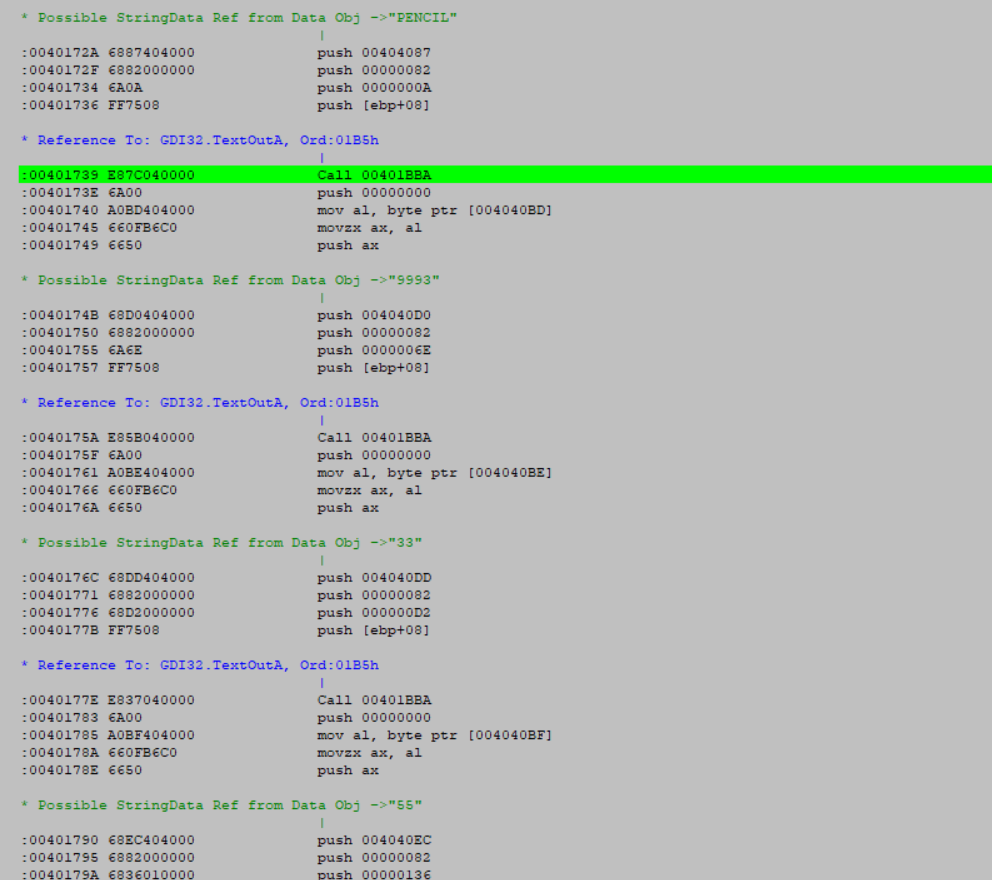


图5.3 反汇编图

源码级调试可以直接清楚的在原来的源码上进行调试，直观性提升了许多，而非源码级测试就要观察复杂的反汇编代码，其中的间接引用和直接引用地址都是直接显示的，如果没有源码作为对比很难理解。

4. 观察收到的消息，记录每个菜单项或按键等操作所对应的消息信息。

IDM\_FILE\_EXIT 10001

IDM\_ACTION\_LIST 10101

IDM\_ACTION\_RECOMMAND 10102

IDM\_HELP\_ABOUT 10201

5. 比较DOS、Windows输出方式，观察Win32/64程序的几种字符串输出方式所用函数的原型。

DOS是使用系统功能调用输出，Windows是调用函数输出。

6. 观察结构变量的推荐度等字段的偏移，体会结构变量优点。

结构变量可以简单的通过.命令获取字段的偏移，使得程序的可读性大大提高

7. 观察Invoke语句翻译成机器码后的特点，记录参数压栈顺序。单步跟踪到调用系统API函数的位置，观察相关代码的特点

Invoke是将参数按照顺序压栈，而在被调用的函数中，又会自动出栈写入原型定义时的对应参数（按照定义的数据类型给字节）。

运行程序结果：

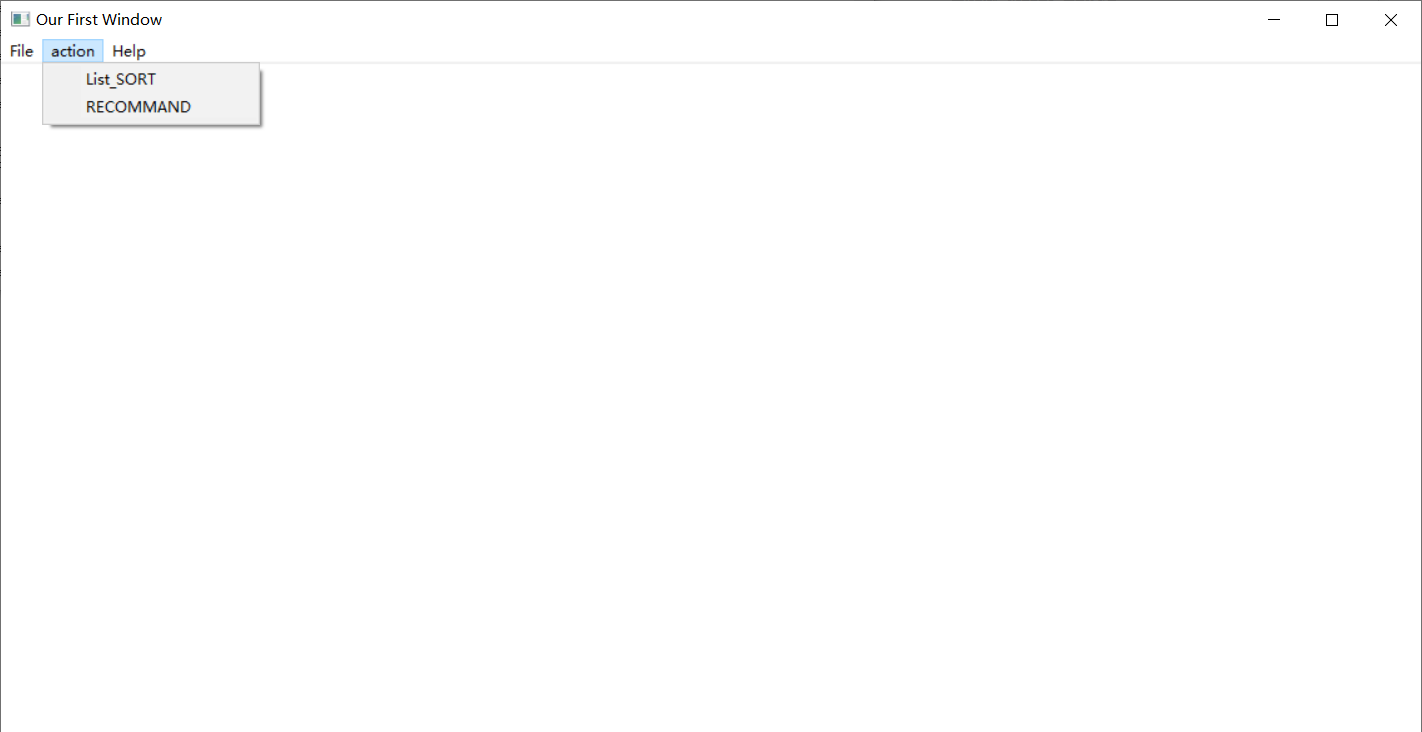


图5.4 菜单栏运行效果图

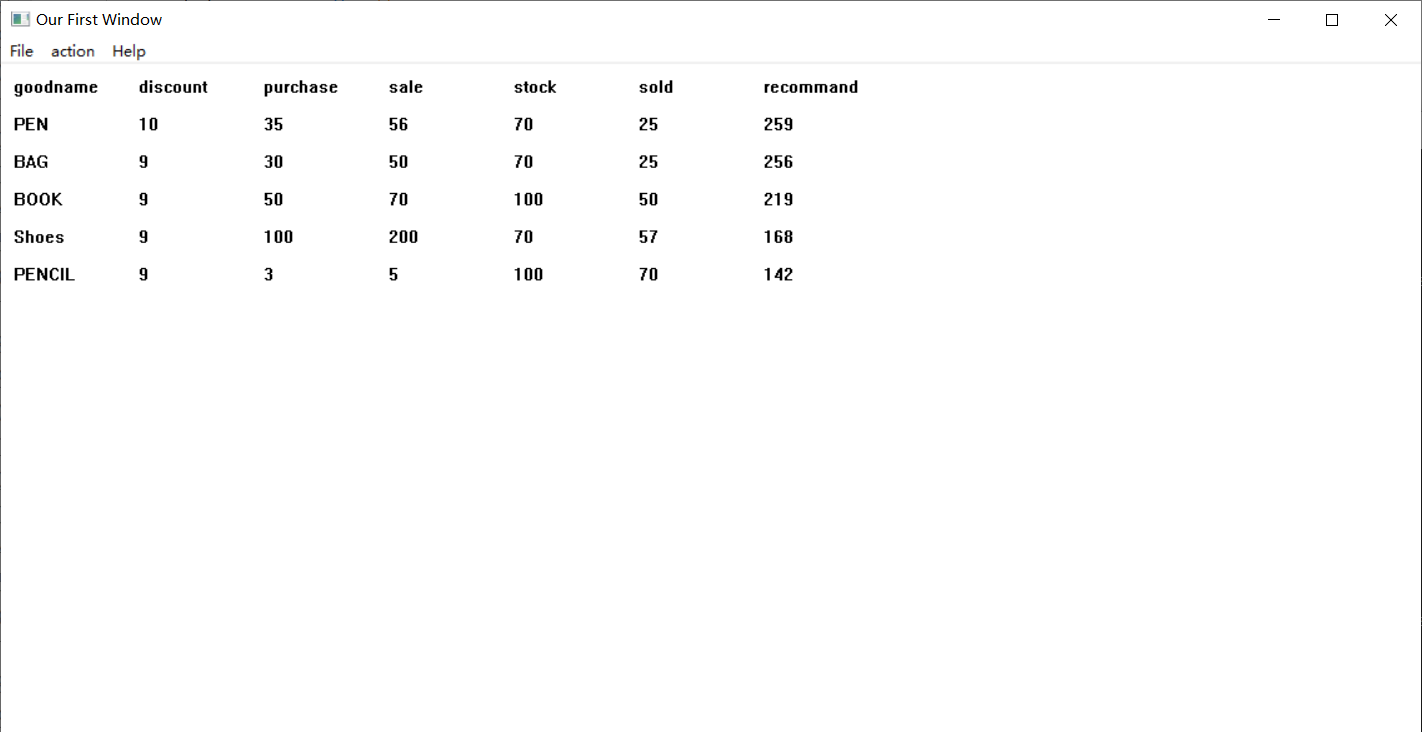


图5.5 排序后效果图

## 小结

WIN32编程，在32位WINDOWS环境中配置编程环境，然后编程的特点就是使用了大量的内置的句柄函数，使用invoke调用，还有就是整个框架处理是比较模式化的，包括收集信息，处理信息，显示窗口之类的。在框架内特定的位置写入自己的信息处理函数就可以个性化的开发WIN32窗口程序。同时，在menu.rc文件夹中修改配置来改变菜单和接受特定信息。

参考文献  
[1] 王元珍 等.80X86汇编语言程序设计.2005-4(第1版).华中科技大学出版社.1-348.  
[2] 许向阳.80X86汇编语言程序设计上机指南.2007-4(第1版).华中科技大学出版社.1-224.