

**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201804班**

**学 号： U201814606**

**姓 名： 岳靖**

**指导教师： 李海波**

**实验时段： 2020年3月19日~5月7日**

**实验地点： 南一楼804室**

**原创性声明**

  本人郑重声明：本报告的内容由本人独立完成，有关观点、方法、数据和文献等的引用已经在文中指出。除文中已经注明引用的内容外，本报告不包含任何其他个人或集体已经公开发表的作品或成果，不存在剽窃、抄袭行为。

特此声明！

学生签名： 纸质版再签名

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*（这一节内容要仔细阅读，保留在实验报告中）* - 2 -](#_Toc35182379)

[0.1 课程目标 - 2 -](#_Toc35182380)

[0.2 成绩构成 - 2 -](#_Toc35182381)

[0.3 实验任务的总体描述 - 2 -](#_Toc35182382)

[1 编程基础*（这之后内容只参考写法与格式，内容可改）* 1](#_Toc35182383)

[1.1 实验目的与要求 1](#_Toc35182384)

[1.2 实验内容 1](#_Toc35182385)

[1.3 任务1.1实验过程 *（这里展示“验证性、研究性”任务的写法）* 1](#_Toc35182386)

[1.3.1 实验方法说明 1](#_Toc35182387)

[1.3.2 实验记录与分析 1](#_Toc35182388)

[1.4 任务1.2的实验过程 *（这里展示“设计型”任务的写法）* 2](#_Toc35182389)

[1.4.1 设计思想及存储单元分配 2](#_Toc35182390)

[1.4.2 流程图 2](#_Toc35182391)

[1.4.3 源程序 3](#_Toc35182392)

[1.4.4 实验步骤 4](#_Toc35182393)

[1.4.5 实验记录与分析 4](#_Toc35182394)

[1.5 小结 5](#_Toc35182395)

[1.5.1 主要收获 6](#_Toc35182396)

[1.5.2 主要看法 6](#_Toc35182397)

[2 程序优化 7](#_Toc35182398)

[2.1 实验目的与要求 7](#_Toc35182399)

[2.2 实验内容 *（研究性任务）* 7](#_Toc35182400)

[2.3 任务2.1实验过程 7](#_Toc35182401)

[2.3.1 实验方法说明 7](#_Toc35182402)

[2.3.2 实验记录与分析 7](#_Toc35182403)

[2.4 任务2.2实验过程 7](#_Toc35182404)

[2.4.1 实验方法说明 7](#_Toc35182405)

[2.4.2 实验记录与分析 7](#_Toc35182406)

[2.5 小结 7](#_Toc35182407)

[2.5.1 主要收获 7](#_Toc35182408)

[2.5.2 主要看法 7](#_Toc35182409)

[3 模块化程序设计 7](#_Toc35182410)

[3.1 实验目的与要求 7](#_Toc35182411)

[3.2 实验内容 *（设计型、综合性任务）* 7](#_Toc35182412)

[。。。。。。。。 7](#_Toc35182413)

[4 中断与反跟踪 8](#_Toc35182414)

[4.1 实验目的与要求 8](#_Toc35182415)

[4.2 实验内容 *（研究性、综合性任务）* 8](#_Toc35182416)

[。。。。。。。。 8](#_Toc35182417)

[5 WIN32程序设计 8](#_Toc35182418)

[5.1 实验目的与要求 8](#_Toc35182419)

[5.2 实验内容 *（研究性、综合性任务）* 8](#_Toc35182420)

[。。。。。。。。 8](#_Toc35182421)

[参考文献 9](#_Toc35182422)

# 课程总体说明*（这一节内容要仔细阅读，保留在实验报告中）*

## 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。

# 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

（上机实验环境说明：计算机配置较好时使用VMWare Workstation的虚拟机环境，或者DOSBox虚拟机；在普通便携计算机上建议使用DOSBox虚拟机。源程序编辑工具可以使用记事本、EDIT、或C语言的编辑器；汇编程序使用MASM 6.0;连接程序使用LINK；调试工具使用TD；具体介绍可参见教材第7章及《80X86汇编语言程序设计上机指南》）

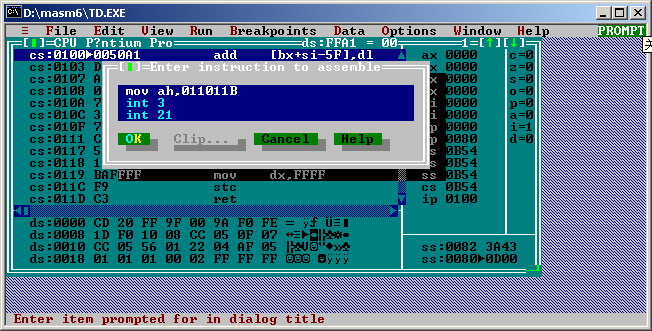
**任务1.1 《80X86汇编语言程序设计》教材中 P31的 1.14题。要求：**

(1) 直接在TD中输入指令（不需要编写完整程序），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。

**在TD中输入指令语句的操作提示：**将TD中的代码显示区置为当前区域，光标移到期望修改的行后，直接输入汇编指令；当输入了第一个字符时，TD自动弹出如下图所示的指令编辑窗口。每输入完一条指令，按回车键，这时输入的指令即可出现在光标处，同时光标自动下移一行，以便输入下一条指令。



**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)各是多少。

（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。

**操作提示：**使用TD.EXE调试程序时，应先单步执行各个语句，每执行一条语句，都应观察数据段中的内容以及相应寄存器的变化。首先注意观察对DS寄存器的赋值过程，并在TD的数据窗口定位待观察的数据区位置。其次，单步执行循环体两遍且正确理解了循环体语句的含义后，可在“MOV AH, 4CH”处设置断点，然后直接执行到断点处，回答(1)和(2)的问题。

**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。

**操作提示：**要让TD从任意指定地址开始反汇编，需要使用TD在代码显示区的Goto功能，即鼠标选中代码显示区，点击右键将显示带有Goto的菜单，选中Goto菜单项，输入CS:XXXX即可（XXXX是你希望录入的偏移地址）。

**任务1.4 设计实现一个网店商品信息管理系统。**

该系统的基本需求见“2020实验报告样例”文档中的“实验任务的总体描述”。

根据系统的基本需求，可以制定如下的数据段的定义（供参考）：

BNAME DB ‘ZHANG SAN’,0 ；老板姓名（本实验要求必须是自己名字的拼音）

BPASS DB ‘test’，0，0 ，0 ；密码

AUTH DB 0 ；当前登录状态，0表示顾客状态

GOOD DB/DW … ；当前浏览商品名称或地址（自行确定）

N EQU 30

SNAME DB ‘SHOP’,0 ；网店名称，用0结束

GA1 DB ‘PEN’, 7 DUP(0) ，10 ；商品名称及折扣

DW 35，56，70，25，？ ； 推荐度还未计算

GA2 DB ‘BOOK’, 6 DUP(0) ，9 ； 商品名称及折扣

DW 12，30，25，5，？ ；推荐度还未计算

GAN DB N-2 DUP( ‘TempValue’ ,0,8，15，0，20，0，30，0，2，0，？，？) ;除了2个已经具体定义了的商品信息以外，其他商品信息暂时假定为一样的。

本次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。

**提示：**字符串比较时，当采用输入串的长度作为循环次数时，若因循环次数减为0而终止循环，则还要去判断网店中定义的字符串的下一个字符是否是结束符0，若是，才能确定找到了（这样做是为了避免输入的字符串仅仅是数据段中所定义字符串的子集的误判情况）。

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）。

**要求尽量避免溢出。**结果只保留整数部分。

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实验过程

### 实验方法说明

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. 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, xxxx；MOV AL, xxxx；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，观察寄存器内容的变化，记录标志寄存器的结果。重复4次。

3.实验结果预测：输入MOV AH,00110011B;MOV AL,01011010B;预计ADD执行之后（AH）=8DH；SF=1、OF=1、CF=0、ZF=0。输入MOV AH，10101001B；MOV AL，11011101B；ADD执行之后（AX）=7AH；SF=1、OF=0、CF=1、ZF=0。输入MOV AH，01100101B；MOV AL，11011101B；预计ADD执行之后（AH）=0008H；SF=0、OF=0、CF=1、ZF=0。

### 实验记录与分析

1. 实验环境条件：16G内存；Windows10下DOSBox0.73； TD.EXE 5.0。
2. 实验过程

(1)输入指令MOV AH,0110011B并单步执行后，得如下结果：

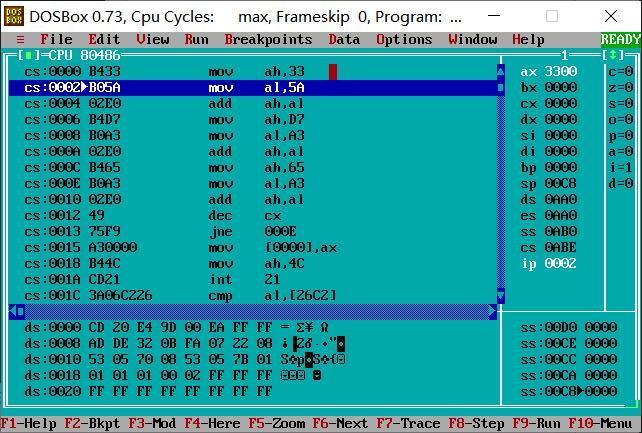


图3.1.1 指令MOV AH,0110011B结果

(2)输入指令MOV AL, 1011010B并单步执行后，得如下结果：

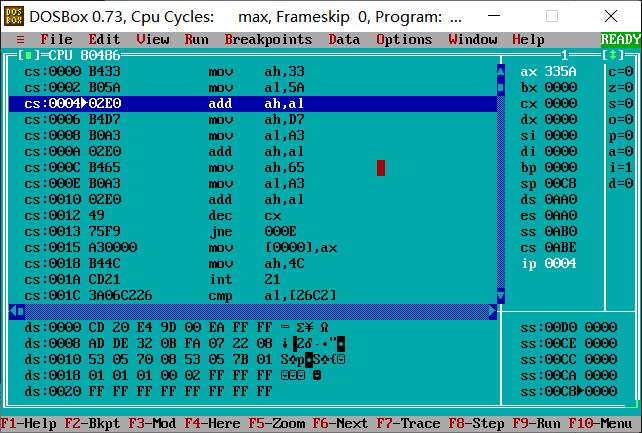


图3.1.2 指令MOV AL, 1011010B结果

(3)输入指令ADD AH, AL并单步执行后，得如下结果：

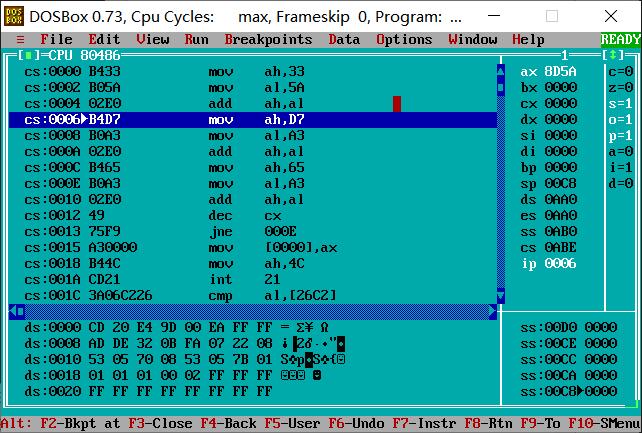


图3.1.3 指令ADD AH, AL结果

(4)输入指令MOV AH,-0101001B并单步执行后，得如下结果：

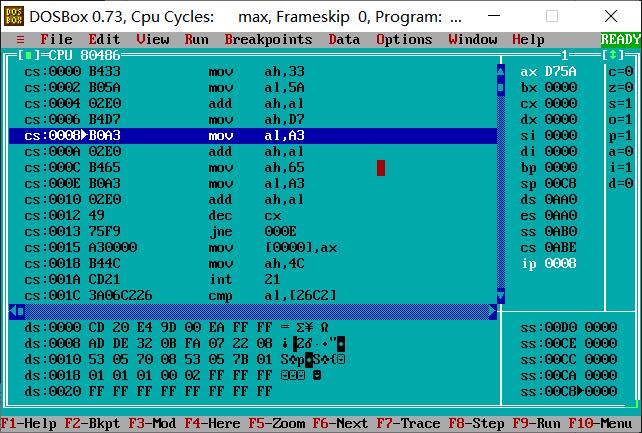


图3.1.4 指令MOV AH,-0101001B结果

(5)输入指令MOV AL,-1011101B并单步执行后，得如下结果：

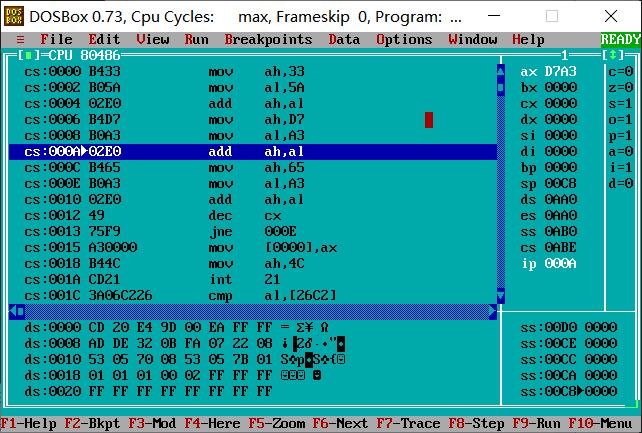


图3.1.5 指令MOV AL,-1011101B结果

(6)输入指令ADD AH, AL并单步执行后，得如下结果：

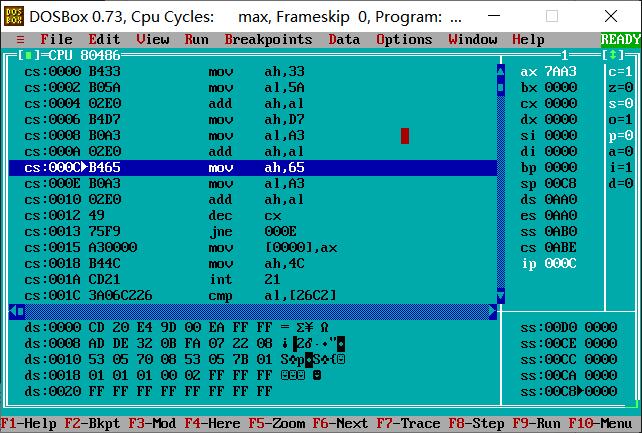


图3.1.6 指令ADD AH, AL结果

(7)输入指令MOV AH,01100101B并单步执行后，得如下结果：

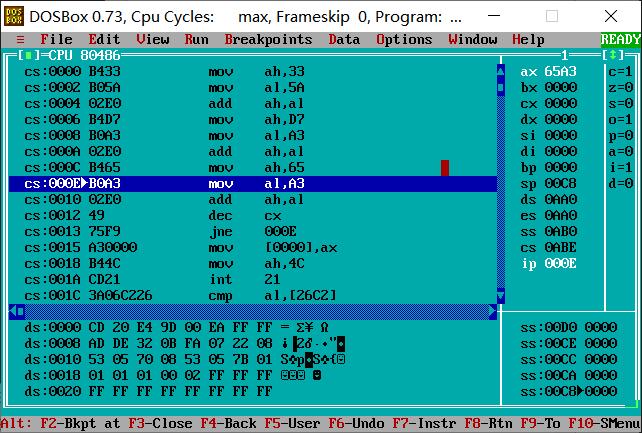


图3.1.7 指令MOV AH,01100101B结果

(8)输入指令MOV AL,-1011101B并单步执行后，得如下结果：

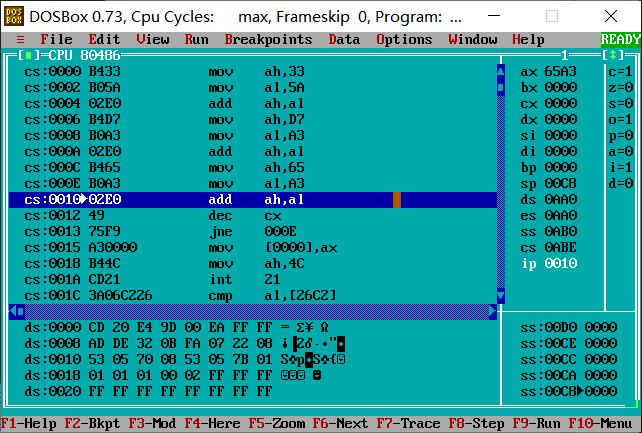


图3.1.8 指令MOV AL,-1011101B结果

(9)输入指令ADD AH, AL并单步执行后，得如下结果：

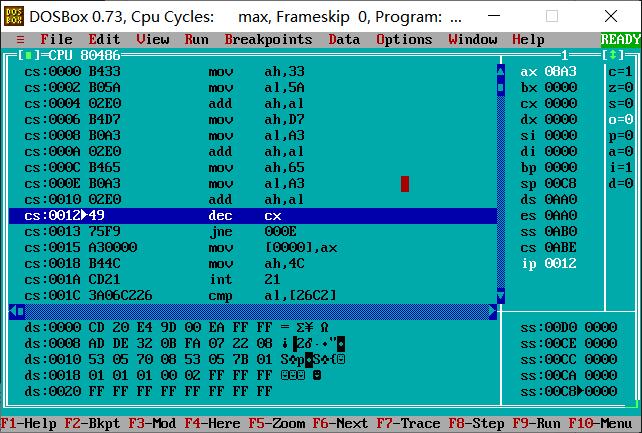


图3.1.9 指令ADD AH, AL结果

## 任务1.2实验过程

### 实验方法说明

1.准备上机环境，编辑、汇编、连接文件并使用TD观察执行情况。

2.记录初始(BX), (BP),(SI),(DI)。

3.记录执行到“MOV CX，10”之前时，(BX), (BP),(SI),(DI)。

4.记录执行到“INT 21H”之前时，(BX), (BP),(SI),(DI)。

5.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。

6.退出前40个字节数据段的预计值

00，01，02，03，04，05，06，07，

08，09，00，01，02，03，04，05，

06，07，08，09，01，02，03，04，

05，06，07，08，09，0A，04，05，

06，07，08，09，0A，0B，0C，0D，

### 源程序

### 源程序

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

SATCK ENDS

DATA SEGMENT USE16

BUF1 DB 0,1,2,3,4,5,6,7,8,9

BUF2 DB 10 DUP(0)

BUF3 DB 10 DUP(0)

BUF4 DB 10 DUP(0)

DATA ENDS

CODE SEGMENT USE16

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

START: MOV AX,DATA

MOV DS,AX

MOV SI,OFFSET BUF1

MOV DI,OFFSET BUF2

MOV BX,OFFSET BUF3

MOV BP,OFFSET BUF4

MOV CX,10

LOPA: MOV AL,[SI]

MOV [DI],AL

INC AL

MOV [BX],AL

ADD AL,3

MOV DS:[BP],AL

INC SI

INC DI

INC BP

INC BX

DEC CX

JNZ LOPA

MOV AH,4CH

INT 21H

CODE ENDS

END START

### 实验记录与分析

1. 实验环境条件：16G内存；Windows10下DOSBox0.72； TD.EXE 5.0。

2. 实验过程

1. 进入调试界面，观察(BX), (BP),(SI),(DI)初始值，得如下结果：

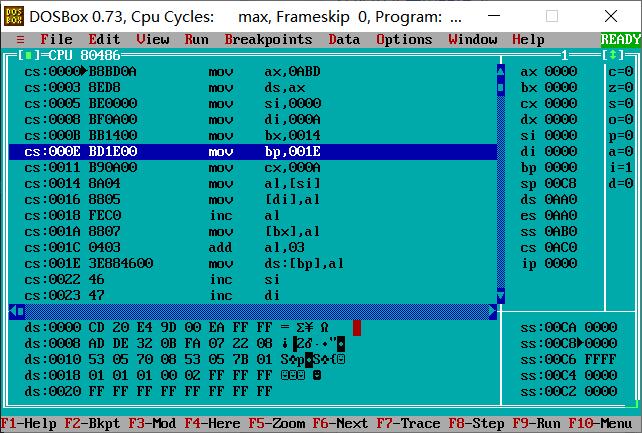


图3.2.1 (BX), (BP),(SI),(DI)初始值

(2) 执行到“MOV CX，10”之前时，观察(BX), (BP),(SI),(DI)的值，得如下结果：

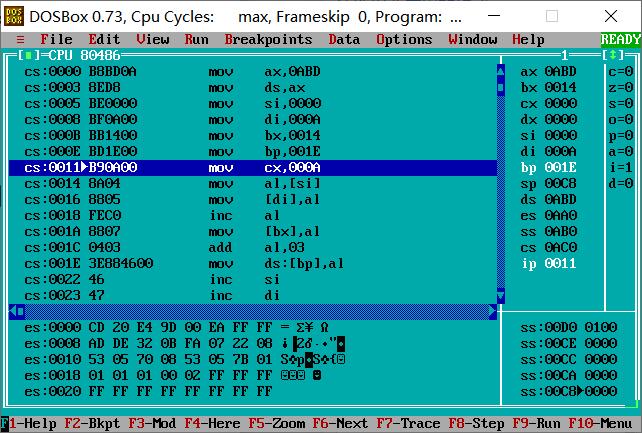


图3.2.2 执行到“MOV CX，10”之前时(BX), (BP),(SI),(DI)值

(3) 执行到“INT 21H”之前时，观察(BX), (BP),(SI),(DI)的值，得如下结果：

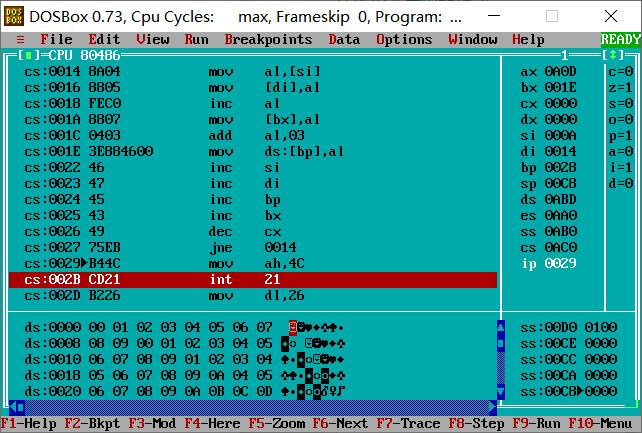


图3.2.3 执行到“INT 21H”之前时(BX), (BP),(SI),(DI)值

(4) 观察程序执行到退出之前数据段开始40个字节的内容，得如下结果：

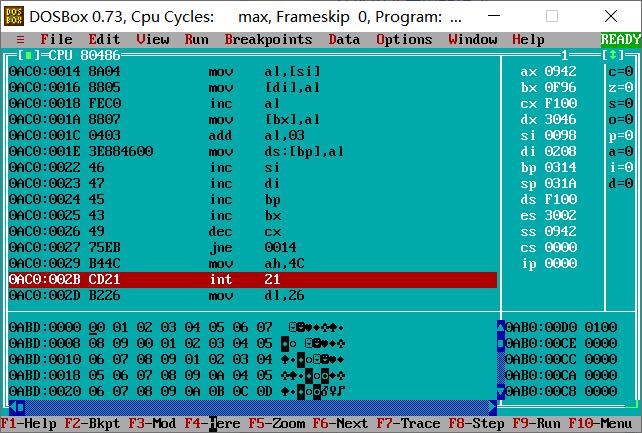


图3.2.4 数据段开始40个字节的内容

## 任务1.3实验过程

### 实验方法说明

1.准备上机环境，编辑、汇编、连接文件，使用TD观察执行情况。

2.观察数据段开始40个字节的内容。

3.在程序运行的过程中，注意指令在内存中的存放方式以及源程序代码与反汇编代码的区别。

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的结果。

5.退出前40个字节的预计值

00，01，02，03，04，05，06，07，

08，09，00，01，02，03，04，05，

06，07，08，09，01，02，03，04，

05，06，07，08，09，0A，04，05，

06，07，08，09，0A，0B，0C，0D，

### 源代码

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

BUF1 DB 0,1,2,3,4,5,6,7,8,9

BUF2 DB 10 DUP(0)

BUF3 DB 10 DUP(0)

BUF4 DB 10 DUP(0)

DATA ENDS

CODE SEGMENT USE16

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

START: MOV EAX,DATA

MOV DS,EAX

MOV ESI,OFFSET BUF1

MOV EDI,OFFSET BUF2

MOV EBX,OFFSET BUF3

MOV EBP,OFFSET BUF4

MOV ECX,10

LOPA: MOV AL,[ESI]

MOV [EDI],AL

INC AL

MOV [EBX],AL

ADD AL,3

MOV DS:[EBP],AL

INC ESI

INC EDI

INC EBP

INC EBX

DEC ECX

JNZ LOPA

MOV AH,4CH

INT 21H

CODE ENDS

END START

### 实验记录与分析

1. 实验环境条件：16G内存；Windows10下DOSBox0.72； TD.EXE 5.0。
2. 实验过程
3. 进入调试界面，设置断点，运行到程序退出前，观察到数据段开始40个字节的内容，得如下结果：

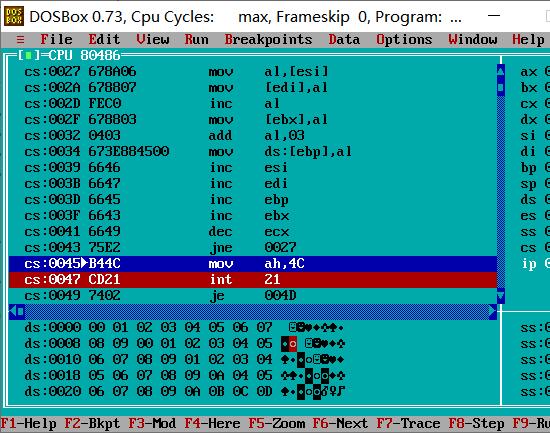


图3.3.1 数据段开始40个字节的内容

1. 32位寄存器对应的机器码与反汇编语言如下：

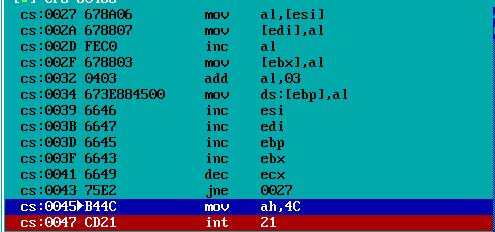


图3.3.2 32位寄存器对应的机器码与反汇编语言

1. 16位寄存器对应的机器码与反汇编语言如下：

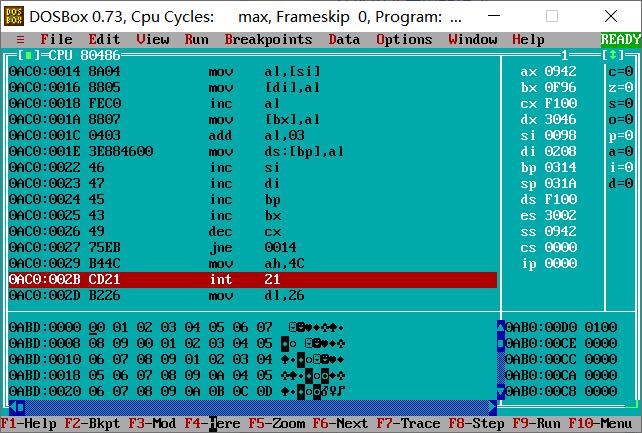


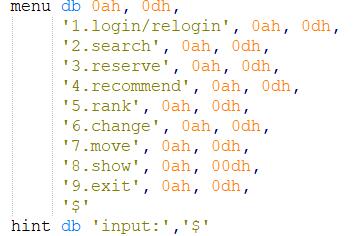
图3.3.3 16位寄存器对应的机器码与反汇编语言

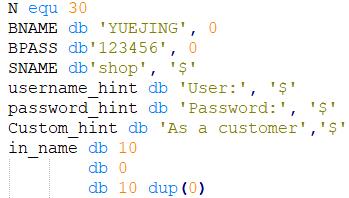
1. 连续存放的二进制串在反汇编成汇编语言语句时，从不同字节位置开始反汇编的结果也不一样。

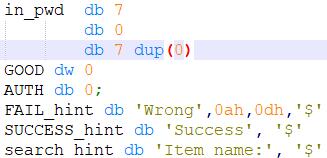
## 任务1.4的实践过程

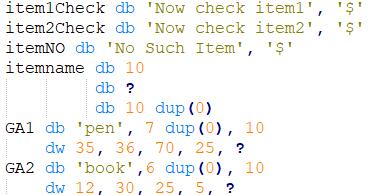
### 实验方法说明

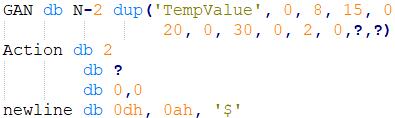
(一)本次实验内存分配











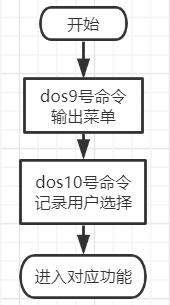
(二)功能说明

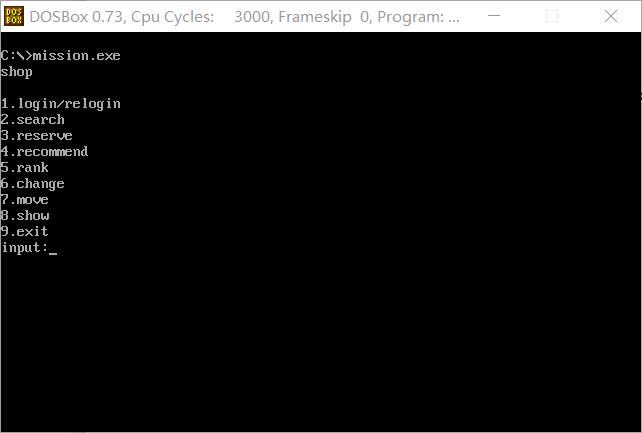
菜单：

（1）使用9号DOS系统功能，提示用户输入操作。

（2）使用10号DOS系统功能，输入操作，保存在action中

（3）跳至对应代码





菜单显示

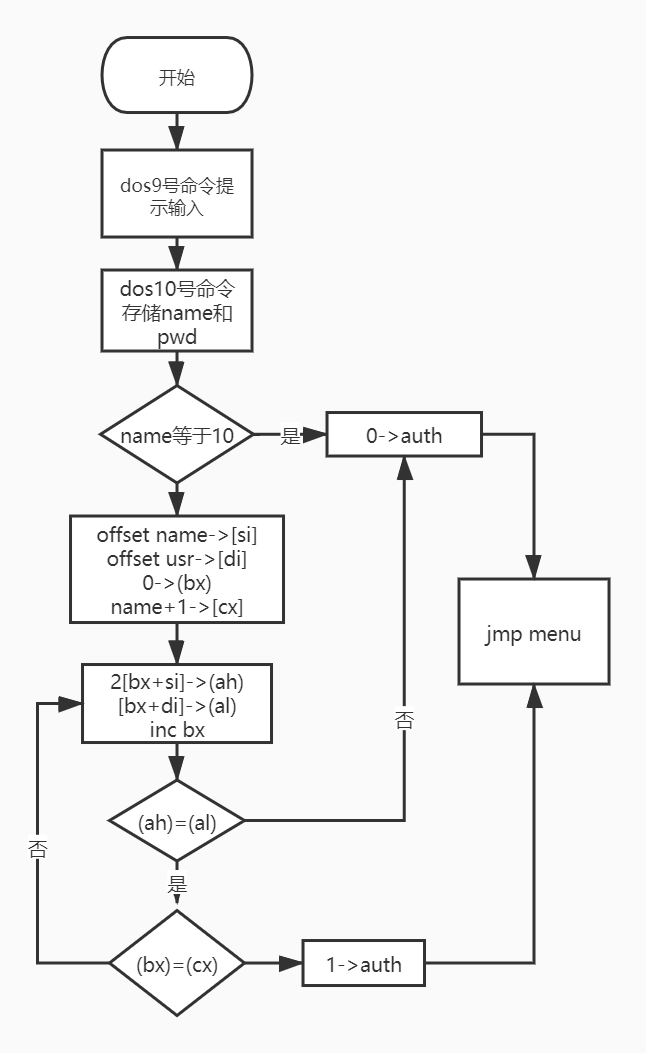
功能1：

（1）使用9号DOS系统功能，提示用户输入姓名和密码。

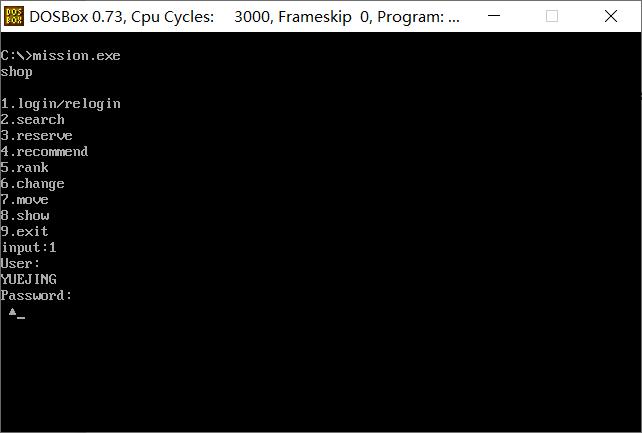
（2）使用10号DOS系统功能，输入姓名和密码。输入的姓名字符串放在IN\_NAME中，密码放在以IN\_PWD中

（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，再次返回主菜单界面

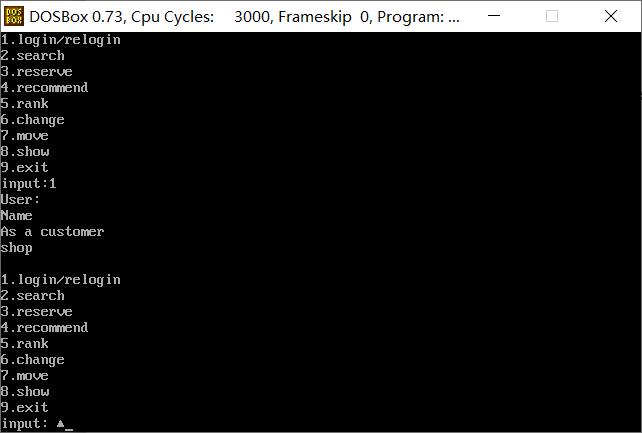
（4）使用循环结构，比较姓名是否正确。若不正确，则使用9号DOS系统功能，提示错误，同时返回主菜单界面。若正确，则比较密码。若二者都正确，则将1送到auth变量，返回主菜单。



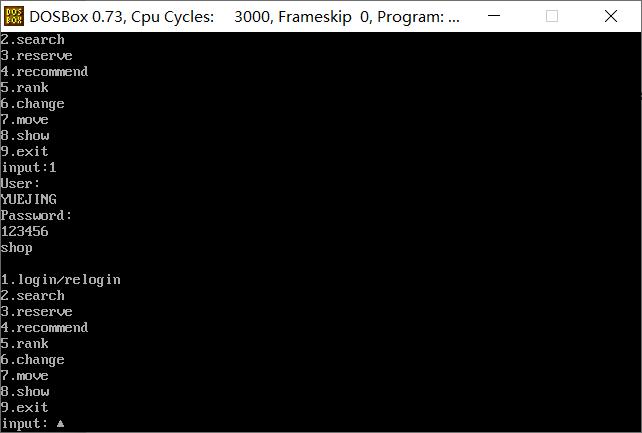
流程图



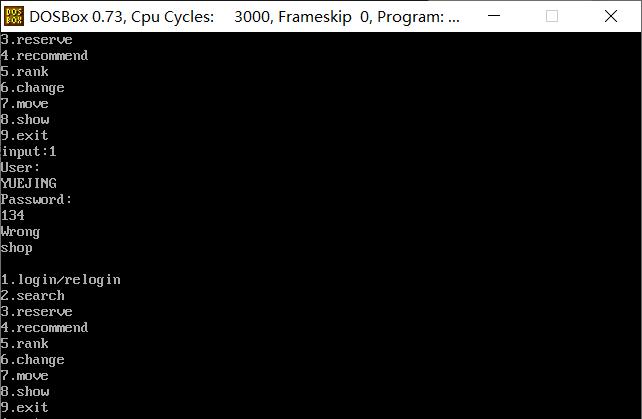
姓名正确时



姓名错误时



姓名正确且密码正确时



用户名正确但密码不正确

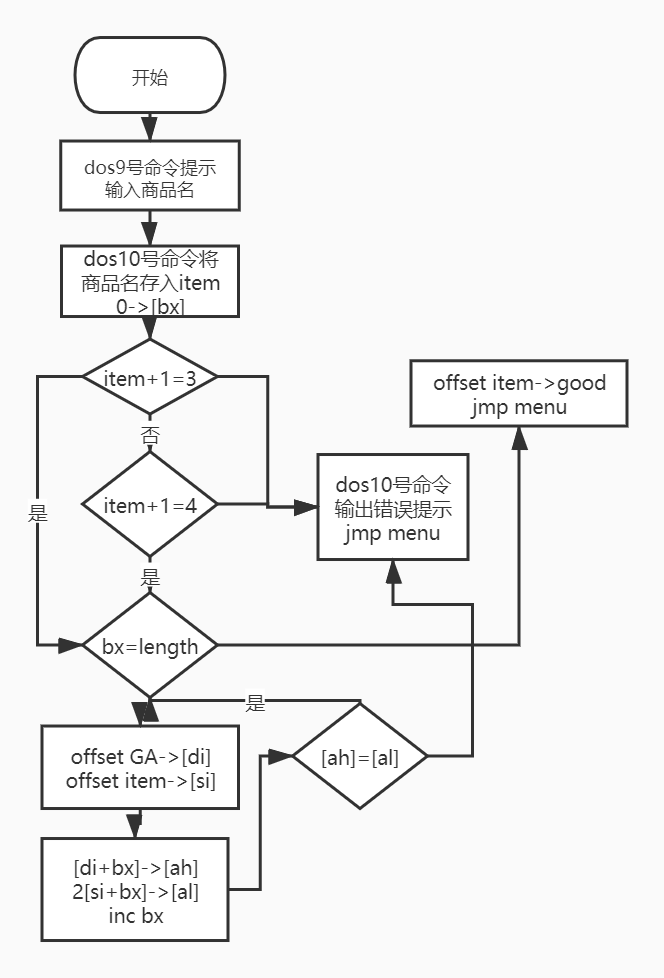
功能二：查找指定商品并显示其信息

（1）使用9号DOS系统功能，提示用户输入商品名称。

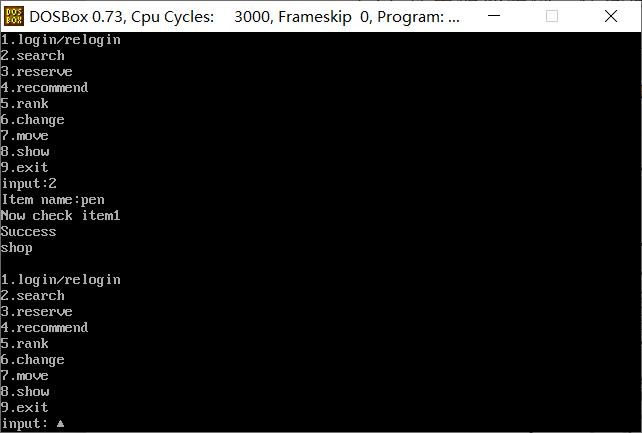
（2）使用10号DOS系统功能，输入商品名称。

（3）使用循环结构，在商店中寻找是否存在该商品。若存在，则将商品名称或地址记录到GOOD字段中,返回到主菜单界面。

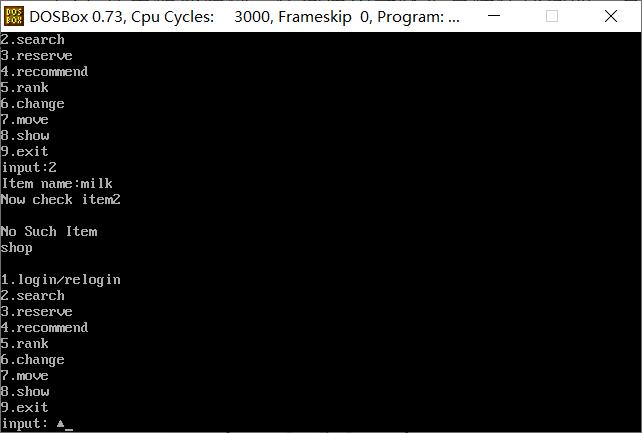
（4）若没有找到，则使用9号DOS系统功能，提示没有找到，返回到主菜单界面。



流程图



查找pen时

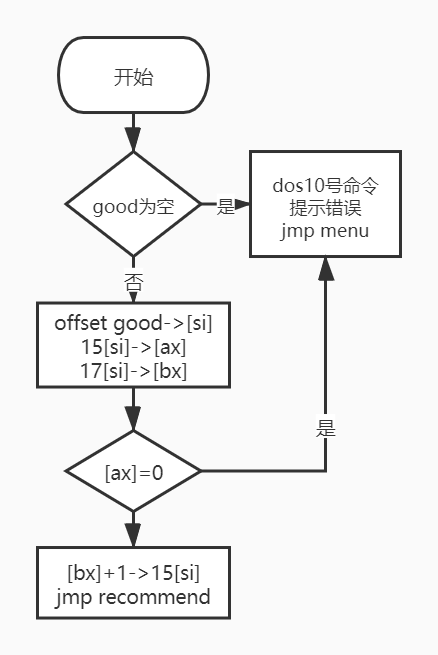


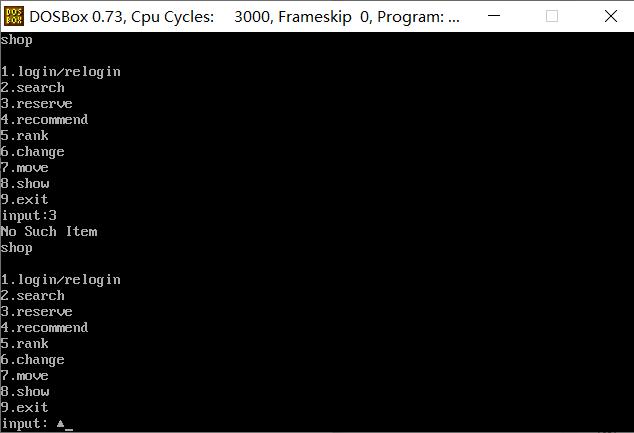
查找物品不存在时

功能三：下订单

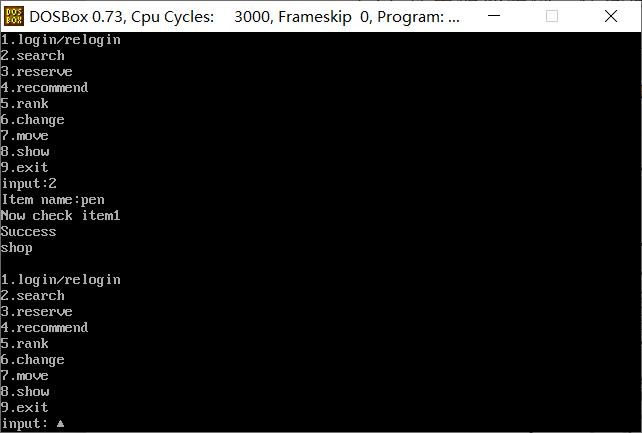
1. 判断变量GOOD是否为空，若不为空，判断其剩余数量是否为0，不为0则将已售数量加1。
2. 重新计算所有商品的推荐度，跳转功能四，返回主菜单界面。

（3）若无效或剩余数量为0，则提示错误，回到主菜单界面。

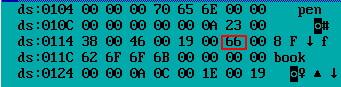




GOOD为空时



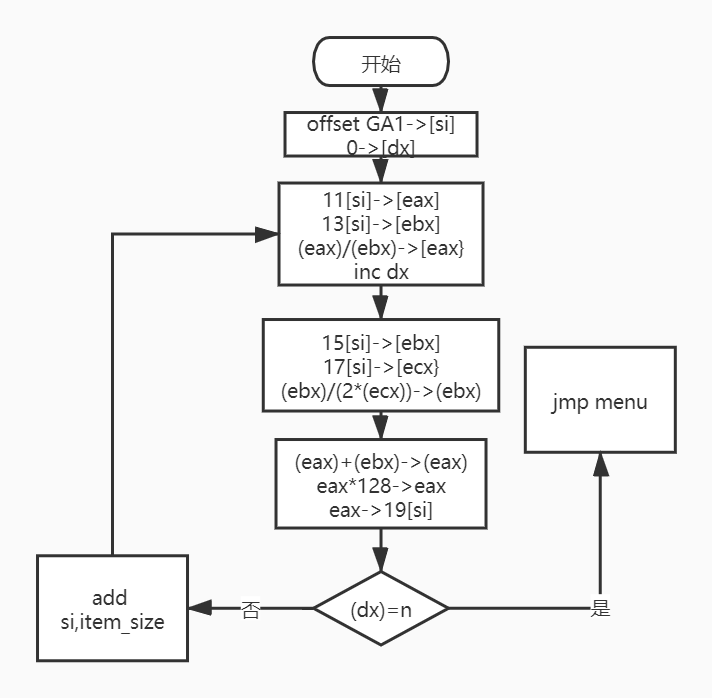
下单成功

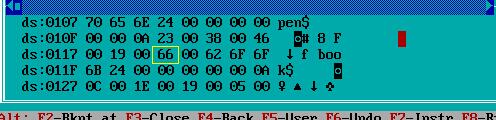


推荐度

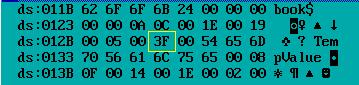
功能四：推荐度计算

1. 通过偏移值的到进货价，实际销售价格，已售数量，进货数量的值
2. 通过公式计算推荐度=（进货价/实际销售价格+已售数量/（2\*进货数量））\*128，并保存指定位置
3. 回到菜单



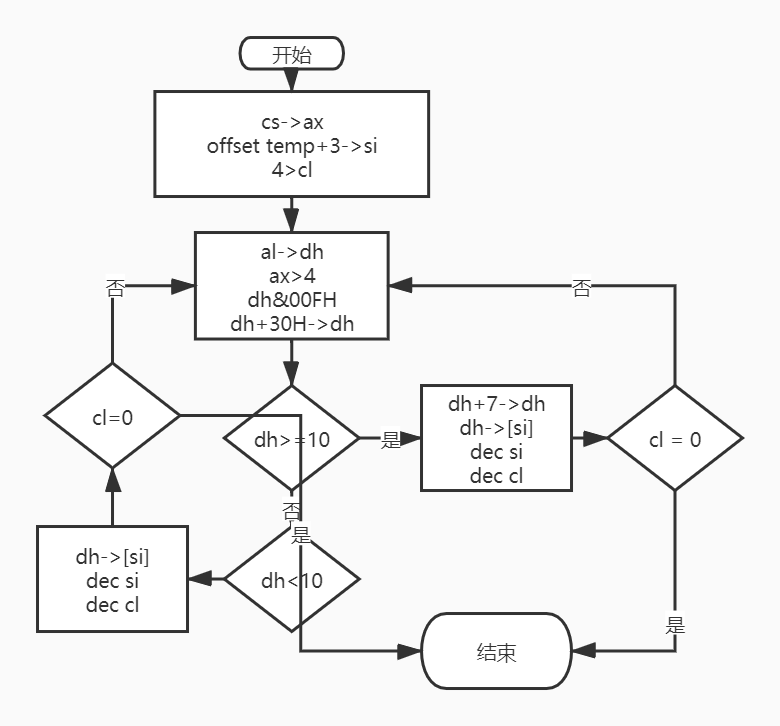


Pen的推荐度

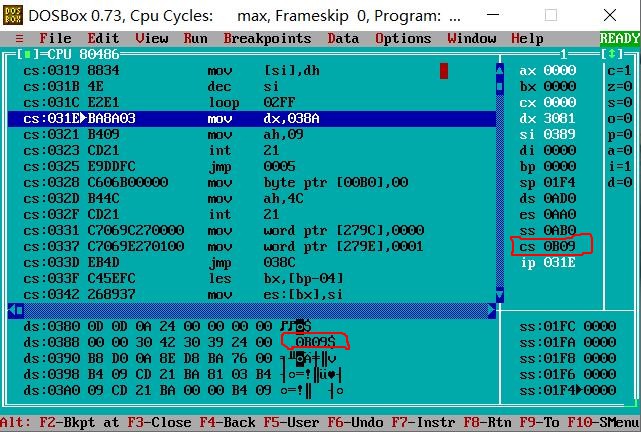


Book的推荐度

功能八：显示首段地址



流程图



将段首址转为字符串

(三)源代码

.386

STACK segment use16 STACK

db 500 dup(0)

STACK ends

DATA segment use16

menu db 0ah, 0dh,

'1.login/relogin', 0ah, 0dh,

'2.search', 0ah, 0dh,

'3.reserve', 0ah, 0dh,

'4.recommend', 0ah, 0dh,

'5.rank', 0ah, 0dh,

'6.change', 0ah, 0dh,

'7.move', 0ah, 0dh,

'8.show', 0ah, 00dh,

'9.exit', 0ah, 0dh,

'$'

hint db 'input:','$'

N equ 30

itemSize equ 21

BNAME db 'YUEJING', 0

BPASS db'123456', 0

SNAME db'shop', '$'

username\_hint db 'User:', '$'

password\_hint db 'Password:', '$'

Custom\_hint db 'As a customer','$'

in\_name db 10

db 0

db 10 dup(0)

in\_pwd db 7

db 0

db 7 dup(0)

GOOD dw 0

AUTH db 0;

NUM db 0

FAIL\_hint db 'Wrong',0ah,0dh,'$'

SUCCESS\_hint db 'Success', '$'

search\_hint db 'Item name:', '$'

item1Check db 'Now check item1'

item2Check db 'Now check item2', '$'

itemNO db 'No Such Item', '$'

itemname db 10

db ?

db 10 dup(0)

GA1 db 'pen', '$', 6 dup(0), 10

dw 35, 56, 70, 25, ?

GA2 db 'book', '$',5 dup(0), 10

dw 12, 30, 25, 5, ?

GAN db N-2 dup('TempValue', 0, 8, 15, 0,

20, 0, 30, 0, 2, 0,?,?)

Action db 2

db ?

db 0,0

newline db 0dh, 0ah, '$'

address dw 0

temp db 0, 0, 0, 0, '$'

DATA ends

code segment use16

assume cs:code, ds:DATA, ss:STACK

start:

mov ax,DATA

mov ds,ax

SHOWMENU:

lea dx, Sname

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

;MENU

lea dx, menu

mov ah,9

int 21h

lea dx,hint

mov ah,9

int 21h

;input action

lea dx, Action

mov ah, 10

int 21h

;action

mov NUM, 0

mov si, offset Action

cmp byte ptr 2[si],'1'

je Login

cmp byte ptr 2[si],'2'

je Search

cmp byte ptr 2[si],'3'

je Reserve

cmp byte ptr 2[si],'4'

je ALLRecommend

cmp byte ptr 2[si],'5'

je ALLRecommend

cmp byte ptr 2[si],'6'

je ALLRecommend

cmp byte ptr 2[si],'7'

je ALLRecommend

cmp byte ptr 2[si],'8'

je Show

cmp byte ptr 2[si],'9'

je exit

Login:

lea dx, newline

mov ah, 9

int 21h

;name

lea dx, username\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

;input name

lea dx, in\_name

mov ah, 10

int 21h

lea dx, newline

mov ah, 9

int 21h

mov si, offset in\_name

cmp byte ptr 2[si], 0dh

je Custom

call name\_check

;password

lea dx, password\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

lea dx, in\_pwd

mov ah, 10

int 21h

lea dx, newline

mov ah, 9

int 21h

call pwd\_check

jmp SHOWMENU

Custom:

lea dx, Custom\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

mov auth,0

jmp SHOWMENU

AUTH\_FAIL:

MOV AUTH, 0

lea dx, FAIL\_hint

mov ah, 9

int 21h

jmp SHOWMENU

name\_check proc

;the right namelength = 8

mov cl, in\_name + 1

mov ch, 0

mov bx, 0 ;length flag

cmp cx, 7

jne Custom

mov si, offset in\_name

mov di, offset BNAME

loopa: mov ah, byte ptr 2[bx+si]

mov al, byte ptr [bx+di]

cmp ah, al

jne AUTH\_FAIL

inc bx

cmp cx, bx

jne loopa

ret

name\_check endp

pwd\_check proc

;the right pwdlength = 6

mov cl, in\_pwd+1

mov ch, 0

mov bx, 0 ;length flag

cmp cx, 6

jne AUTH\_FAIL

mov si, offset in\_pwd

mov di, offset BPASS

loopb: mov ah, byte ptr 2[bx+si]

mov al, byte ptr [bx+di]

cmp ah, al

jne AUTH\_FAIL

inc cx

cmp cx, bx

jne loopb

mov auth, 1

ret

pwd\_check endp

SEARCH:

;input the item to search

lea dx, newline

mov ah, 9

int 21h

lea dx, search\_hint

mov ah, 9

int 21h

lea dx, itemname

mov ah, 0ah

int 21h

lea dx, newline

mov ah, 9

int 21h

;now campare the length

mov si, offset itemname

cmp byte ptr 1[si], 3

je ITEM1CMP

cmp byte ptr 1[si], 4

je ITEM2CMP

ITEM1CMP:

lea dx, item1Check

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

mov bx, offset GA1

mov di, bx ;offset GA1

mov si, offset itemname

mov bl, itemname+1

mov bh, 0

mov byte ptr itemname + 2[bx], '$'

mov bx, 0 ;flag

loopc:

cmp bx, 4

je GOTIT1

mov ah, byte ptr [di + bx]

mov al, byte ptr 2[si + bx]

inc bx

cmp ah, al

je loopc

jne ITEMNOTEXIST

GOTIT1:

mov GOOD, offset GA1

lea dx, SUCCESS\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

jmp SHOWMENU

ITEM2CMP:

lea dx, item2Check

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

mov bx, offset GA2

mov di, bx ;offset GA2

mov si, offset itemname

mov bl, itemname+1

mov bh, 0

mov byte ptr itemname + 2[bx], '$'

mov bx, 0 ;flag

loops:

cmp bx, 5

je GOTIT2

mov ah, byte ptr [di + bx]

mov al, byte ptr 2[si + bx]

inc bx

cmp ah, al

je loops

jne ITEMNOTEXIST

GOTIT2:

mov GOOD, offset GA2

lea dx, SUCCESS\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

jmp SHOWMENU

ITEMNOTEXIST:

lea dx, newline

mov ah, 9

int 21h

lea dx, itemNO

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

jmp SHOWMENU

reserve:

cmp GOOD, 0

je ITEMNOTEXIST

mov bx, GOOD

mov ax, 15[bx]

mov cx, 17[bx]

sub ax, cx

cmp ax, 0

je ITEMNOTEXIST

mov cx, 17[bx]

inc cx

mov 17[bx], WORD ptr cx

lea dx, SUCCESS\_hint

mov ah, 9

int 21

lea dx, newline

mov ah, 9

int 21

jmp RecommendGood

RecommendGood: ;good用

add bx,10; 跳到折扣

mov ax, 3[bx];销售价

mov dl, byte ptr [bx];

mov dh, 0

;实际售价=折扣\*销售价/10

mul dx

mov cx, 10

div cx

mov cx, ax;cx中是实际售价

mov ax, 1[bx];进货价

mov dx, 128

mul dx

mov dx,0

div cx;进货价\*128/实际售

mov cx, ax; 前半个式子存入cx

mov ax, word ptr 7[bx];已售数量

mov dx, 64;128/4

mul dx

mov dx, 0

div word ptr 5[bx];进货总数

add ax, cx

mov word ptr 9[bx], a

lea dx, SUCCESS\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

jmp SHOWMENU

Recommend: ;功能4用

add bx,10; 跳到折扣

mov ax, 3[bx];销售价

mov dl, byte ptr [bx];

mov dh, 0

;实际售价=折扣\*销售价/10

mul dx

mov cx, 10

div cx

mov cx, ax;cx中是实际售价

mov ax, 1[bx];进货价

mov dx, 128

mul dx

mov dx,0

div cx;进货价\*128/实际售价

mov cx, ax; 前半个式子存入cx

mov ax, word ptr 7[bx];已售数量

mov dx, 64;128/4

mul dx

mov dx, 0

div word ptr 5[bx];进货总数

add ax, cx

mov word ptr 9[bx], a

lea dx, SUCCESS\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

jmp ALLRecommend

ALLRecommend: ;功能4用

mov cl, NUM

cmp cl, N

je SHOWMENU

mov bx, offset GA1

mov al, NUM

mov dl, itemSize

mul dl

add bx, ax

cmp word ptr 1[bx], 0

je ALLRecommend

inc cl

mov NUM, cl

jmp Recommend

Show:

lea dx, newline

mov ah, 9

int 21h

lea dx, SUCCESS\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

CS1:

MOV AX,CS

MOV SI,OFFSET TEMP+3

MOV CX,0

MOV CL,4

CS2:

MOV DH,AL

SHR AX,4

AND DH,000FH

ADD DH,30H

CMP DH,':'

JA ISLETTER

JB ISNUM

ISLETTER:

ADD DH,7H

MOV [SI],DH

DEC SI

LOOP CS2

ISNUM:

MOV [SI],DH

DEC SI

LOOP CS2

PRINT:

MOV DX,OFFSET TEMP

MOV AH,09

INT 21H

lea dx, newline

mov ah, 9

int 21h

JMP SHOWMENU

exit:

mov auth,0

mov ah,4ch

int 21h

code ends

end start

## 思考题回答：

任务1.1：

打开TD之后，如何在代码区输入一条指令，并且执行这条指令？

直接键盘输入即可，F7单步执行

如何在代码区输入若干条指令后，再从输入的第一条指令开始执行？

光标移动到指定代码处后alt+f8，进入GOTO语句，输入CS：0000，同时alt+f8设置此句为操作句

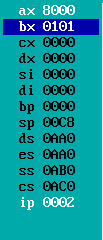
在输入一条指令中的数据时，若以16进制输入，需要注意什么问题？

注意数据的后缀H不能省略

输入指令MOV AH, -128并执行 查看AH寄存器的内容

D8

执行一条指令后，如何查看寄存器的值（含32位寄存器）？如何修改寄存器的值？

点击想要修改的寄存器，右键点击change

执行一条指令后，如何查看标志寄存器的值？



任务1.2和任务1.3：

如何使程序运行到光标的当前点？（假设活动光标在代码区，指向某一条指令）

变换IP的值

如何设置断点并运行到断点？

设置断点快捷键F2

运行到断点 快捷键F9

如何单步执行一条指令？（多种方法）

快捷键F8

任务1.4：

1.10号功能调用时，输入的字符数超过定义的数量时，它是如何处理的？

测试中输入值无法回车输入进系统

2.如果在9号功能调用时，带显示字符串的结尾没有“$”结束符会怎样？

会将数据段中从该字符串开始的所有数据一直输出直到遇见结束符

3.循环或转移时，是否有多种指令的组合方式实现？

可以使用loop+定义cx的值实现循环，也可以直接设置一个标志，每次检查标志，结合je,jne,ja等操作实现循环

## 1.8总结

1. 任务1.1-1.3总结：前三个任务的基本理解难度不高，属于比较轻松的类型，但是第一次在TD上一步步测试在刚开始让我吃了点苦头，一开始由于下意识依赖鼠标的关系经常在DOSBox出现问题鼠标左右键失灵过后就只能重启程序，非常非常不方便，同时一开始对TD的快捷键分布也十分不熟悉，在摸索了一段时间过后才算是能够比较轻松地使用TD这门工具了。同时另一个问题是对于汇编代码的直观感受还是比较陌生，一眼看完会感觉头疼。同时对于寄存器部分知识也没有完全记住，有问题的地方还比较多。
2. 任务1.4总结：该任务的难度比较大，最困难的两个部分是1.不知道如何分配寄存器，该实验中有不少需要循环比较的地方，而实现循环时对于不同的数据的分批处理就很麻烦，经常是写代码和流程图的时候觉得分配对了，但是一调试就发现中途被重复使用或者使用错了。2.是对于不少汇编的函数功能很是陌生，只能对着ppt边思考需要的功能边搜索是否有相应的函数，比较耗费时间，同时也没能实现最优的情况。关于汇编程序设计需要学习的地方还有很多。

# 程序优化

## 一、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。

## 二、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。

**提示** 1：程序计时方法有多种，比较直观的方法是在待测程序执行前获取一个时间标签，执行之后再获取一个时间标签，然后通过计算时间标签的差值来得到程序的执行时间。在QQ群里上传了一个计时程序timer.asm（里面有使用方法的注释），供大家学习参考。但该计时程序的精度在ms级，不能直接观察到几条指令对程序执行时间的影响。可以通过大量重复执行几条指令的方法来解决此问题。在复杂的数据处理、大量用户同时访问（如双十一购物）、大数据时代等场景下，某个功能的程序被大量地循环执行的情形还是比较常见的，因此，上述计时的做法也具有实用性，可以用来观察评估程序的执行效率。

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。

**提示** 2：通过修改m和商品数量n值（比如都大于1000等），来观察程序执行时间的变化。

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。**

**优化方法提示：**首先是通过选择执行速度较快的指令来提高性能，比如，把乘除指令转换成移位指令、加法指令等；其次，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；第三，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。

## 任务2.1实验过程

### 实验方法说明

1. 设置好m的值后先进行查找操作。
2. 判断完选择成功与否后选择使用倒计数法进行下订单的操作。
3. 在循环中通过进货量-销售量判断是否还有货。
4. 销售量加1后进行计算所有物品的推荐度。

流程图如下:

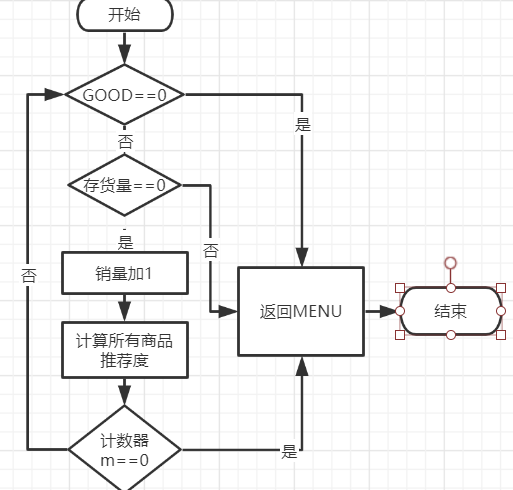


图2-1流程图

### 实验记录与分析

1.实验环境条件：16G内存；Windows10下DOSBox0.72； TD.EXE 5.0;

2.实验过程

cpucycles为100左右时：

当m=5000时：



图2-1

当m=10000时：



图2-2

## 任务2.2实验过程

### 实验方法说明

优化思路有三条，如下：

1. 使用移位指令处理推荐度计算中的乘128。
2. 计算所有物品推荐度时除开bag以外的物品都只计算一次。
3. 寻址方式采用32位。

### 关键代码如下

1. 移位乘法指令

优化前：

mov ax, 1[bx];进货价

mov dx, 128

mul dx

优化后：

sal ax, 7

1. 寻址方式

优化后:

mov ax, 3[ebx];销售价

### 实验记录与分析

本次实验将2.4.1中的三种优化思路在cpu100转下m=1000时都进行了尝试，结果如下：

32位寻址：

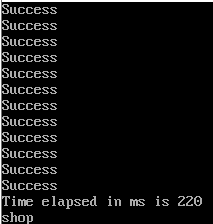


图2-3

移位乘法：

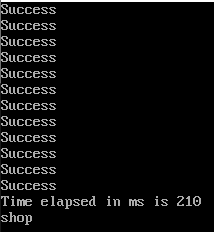


图2-4

算法上减少推荐度的计算:

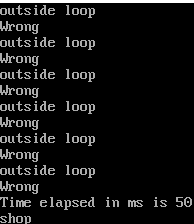


图2-5

效率比较：m大小为1000，cpucycles为100时：

|  |  |  |
| --- | --- | --- |
| 优化方式 | 运行时间 | 优化效率 |
| 未优化 | 270 | 0 |
| 32位寄存器 | 220 | 18％ |
| 移位乘法 | 210 | 22％ |
| 算法优化 | 50 | 81％ |

## 思考题回答

2.1.2插入2.9号命令影响：

内循环体中有信息显示的代码会大幅度增加程序执行时间。我在程序中插入了2号功能调用，观察到程序时间增加了一倍以上。

2.2.1：优化思路:

1. 尽量多使用寄存器来进行操作2.寻址方式采用32位方式3.对程序流程进行化简，去掉多余的指令。

2.2.2关键指令：

乘法指令和除法指令等可以用算数移位处理，循环标志等可以使用寄存器处理。

## 总结

本次实验让我体会到了硬件执行程序优化的思路与直接使用高级语言编程的优化思路的一些区别，面对汇编这门向程序员开放了寄存器的语言，灵活地使用寄存器是肯定能给程序执行带来好处的，只是目前我经验尚浅，还不能很好的融会贯通，需要之后继续学习相关内容。

# 模块化程序设计

## 实验目的与要求

掌握子程序设计的方法与技巧，熟悉子程序的参数传递方法和调用原理；

掌握宏指令、模块化程序的设计方法;

掌握较大规模程序的开发与调试方法；

掌握汇编语言程序与C语言程序混合编程的方法；

了解C编译器的基本优化方法;

了解C语言编译器的命名方法，主、子程序之间参数传递的机制。

## 实验内容

任务3.1 宏与子程序设计 （本次实验为设计型实验，实验报告应按照任务1.4的章节结构撰写，但不需要给出完整的流程图，只需要说明模块结构，子程序之间的调用关系即可；不需要给出完整的源程序，只需要给出新增子程序对应的源程序代码即可。预习报告应完成该任务“实验记录与分析”之前的全部内容）

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。注意：

1）在每个模块的开始，注明编写者的名字以及本模块中包含哪些子程序（只写名称）。

2）在程序设计思想中，要说明模块之间的通信“协议”，包括相互有关联的子程序名及其类型与参数、变量名及其类型、段的定义要求等信息。

3）对每个模块设计一套简单的测试方法，包括一个简单明了的测试框架和一组必要的测试数据，能验证被测模块符合模块之间的通信“协议”，以及该模块的功能基本正确。每个模块单独测试通过后，再一起连接生成一个程序，然后继续进行整体调试（虽然在一开始的时候就可以直接把编好的几个模块连接成一个程序，进行测试和调试，但这并不是实际中采用的最好的方法，而对单个模块先单独进行独立的测试是实际中比较有效的）。

4）需要借鉴书上（或网上）的进制转换程序：十进制转二进制的子程序F10T2和二进制转十进制的子程序F2T10（直接借鉴的程序不需要在实验报告中描述其详细内容）。

上述任务3.1中，需要解决的问题提示：

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。若执行RET前把栈顶的数值改掉，那么RET执行后程序返回到何处？

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。对一个NEAR类型子程序强制使用FAR调用（即CALL FAR PTR 子程序名）会怎样？反之，对一个FAR类型的子程序（子程序可以与主程序在同一个代码段，也可以在不同的代码段）强制使用NEAR调用又会怎样？

通过把一个模块拆成多个模块或反之，体会子程序和模块化程序设计的方法，体会模块调用关系图、子程序功能说明、输入/输出说明在程序设计中的作用。注意 “功能”描述、“模块”描述、“子程序”说明等的区别与联系。

观察不同模块的可合并段合并后变量偏移地址的变化情况。观察不同段在内存里的放置次序。体会模块间段的定义及其对应的装配方法。

在编程中尝试使用不同的子程序参数传递方法来编写子程序。

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？

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。

EXTRN说明语句放在.386之前或者之后有什么区别？(需要用TD观察)

EXTRN说明的变量的段与段寄存器的关联关系（ASSUME伪指令所表达的信息）是否能带入到本模块中？

如何利用宏功能使汇编语言的程序变得更加直观易读？

例：下面是一个利用宏功能直观化后的完整代码段程序，请写出对应的宏定义，并模仿该方式对自己编写的某段程序进行类似的改写。

StartProgram code，data，stack，start

Initial\_ds

GetStringTo BUF

DisplayStringFrom BUF

ExitToDOS

EndProgram code，start

## 任务一过程

### 实验方法说明

主程序中将控制输入用户名/密码/商品名部分设置为宏，其余部分使用子函数。

功能2的增强部分通过在主程序中将GOOD数据传给模块程序wan2.asm中来遍历商品的所有属性结合进制转换函数实现。

功能6也是通过将GOOD数据传给模块程序FUNC6.asm中来修改对应数据段。

新定义的数据如下:

Iname db 'Name:', 4 dup(0), '$'

off db 'off:', 5 dup(0), '$'

price db 'price:', 3 dup(0), '$'

db 'stock:', 3 dup(0), '$'

db 'sold:', 4 dup(0), '$'

db 'rec:', 5 dup(0), '$'

bufa db 15 dup(0)

### 模块结构说明

主程序模块引入引出：

NAME WAN1

EXTRN FUNC:FAR, FUNC6:FAR

PUBLIC GOOD, AUTH

Wan2模块引入引出：

NAME WAN2

EXTRN GOOD:WORD

public FUNC

Lab6模块引入引出:

NAME WAN3

EXTRN GOOD:WORD, AUTH:BYTE, FUNC:FAR

PUBLIC FUNC6

### 流程图

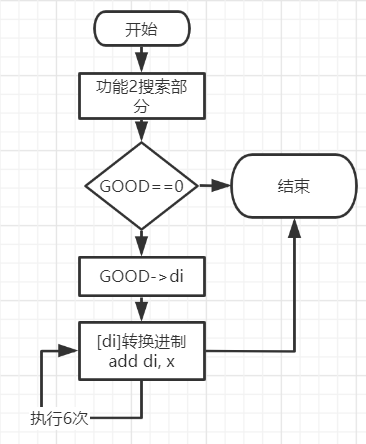


图3-1功能2附加部分

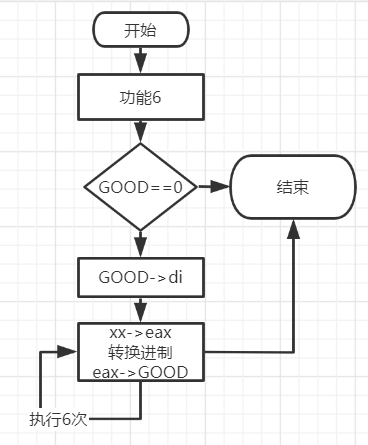


图3-2功能6

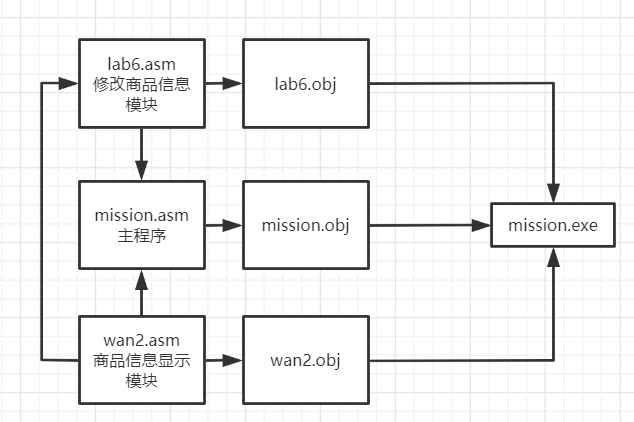


图3-3模块关系

### 关键代码

将输入姓名和密码以及商品名的部分改为了宏：

INUSER MACRO in\_name,newline,username\_hint

lea dx, newline

mov ah, 9

int 21h

;name

lea dx, username\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

;input name

lea dx, in\_name

mov ah, 10

int 21h

lea dx, newline

mov ah, 9

int 21h

endm

INPWD MACRO in\_pwd, newline, password\_hint

lea dx, password\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

lea dx, in\_pwd

mov ah, 10

int 21h

lea dx, newline

mov ah, 9

int 21h

Endm

INITEM MACRO itemname, newline, search\_hint

lea dx, newline

mov ah, 9

int 21h

lea dx, search\_hint

mov ah, 9

int 21h

lea dx, itemname

mov ah, 0ah

int 21h

lea dx, newline

mov ah, 9

int 21h

Endm

进制转换函数借鉴了教材p136的RADIX函数，此处不作展示。

### 实验记录与分析

进入程序，未登陆情况下选择功能6：

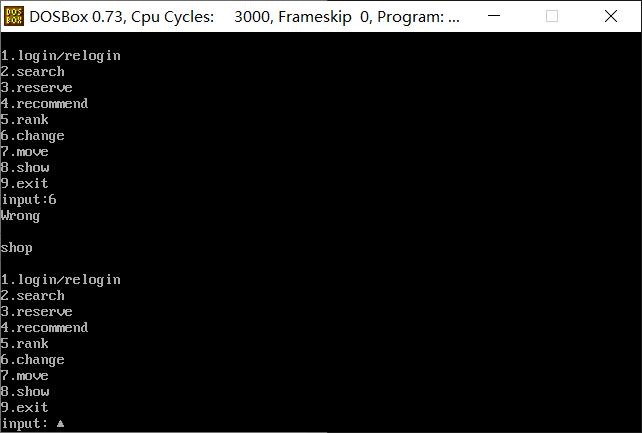


图3-4

使用功能2查找不存在的商品：

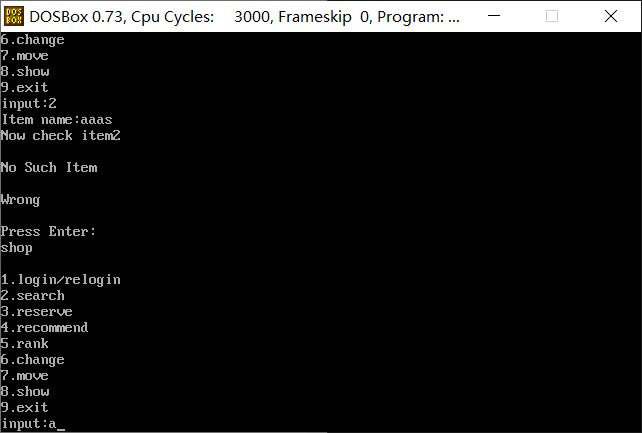


图3-5

使用功能2查找pen的信息：

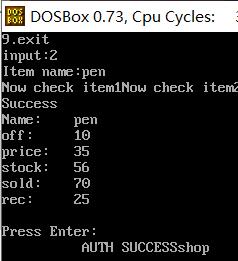


图3-6

与源程序初始值比较：

pen初始信息

图3-7

使用功能6：

先是显示当前浏览商品的信息:

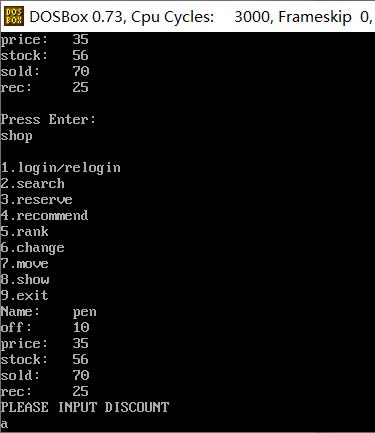


图3-8

修改信息：

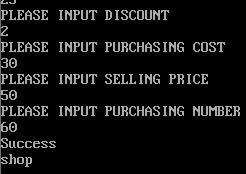


图3-9

检查是否修改成功:

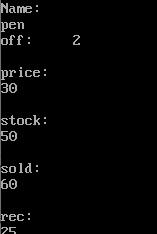


图3-10

## 任务二过程

### 实验方法说明

在不同的环境进行测试过后选择了使用C语言代码中嵌入汇编代码的方式。

本次实验选择了在DosBox上安装BorlandC进行编程。

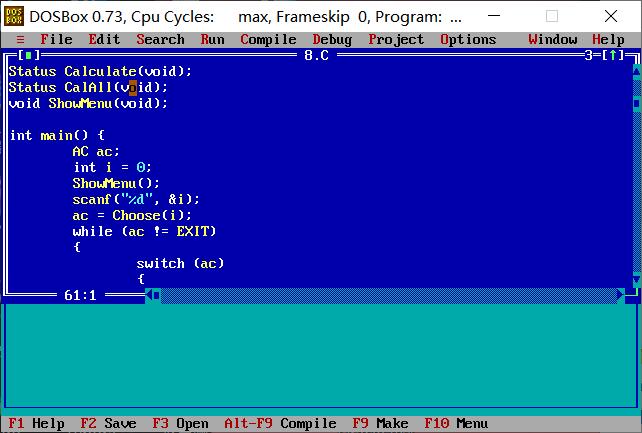


图3-11BorlandC界面

在任务3.1的代码基础上重构了:

1. 菜单的显示和功能选择完全通过C语言的printf和scanf实现
2. 将登录输入用户名和密码的汇编代码嵌入进了登陆函数中
3. 将查找商品的汇编代码嵌入进查找函数
4. 将计算推荐度代码侵入进计算函数中

使用每一段嵌入汇编代码时都保护了现场，保护了所有的寄存器。

### 关键代码

查找商品函数如下：

Status Search(void){

char search\_hint[] = "Item name:";

char item1Check[] = "Now check item1";

char item2Check[] = "Now check item2";

char itemNO[] = "No such item";

char itemname[10];

printf("%s", search\_hint);

scanf("%s", itemname);

getchar();

if( (strlen(itemname)!=3)&& (strlen(itemname)!=4))

{

printf("%s", FAIL\_hint);

\_\_asm{

mov GOOD, 0

}

return ERROR;

}

\_\_asm{

newline db 0dh, 0ah, '$';

push ax

push dx

push cx

push dx

push di

push si

mov si, offset itemname

cmp byte ptr 1[si], 3

je ITEM1CMP

cmp byte ptr 1[si], 4

je ITEM2CMP

ITEM1CMP:

lea dx, item1Check

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

mov bx, offset GA1

mov di, bx

mov si, offset itemname

mov bl, itemname+1

mov bh, 0

mov byte ptr itemname + 2[bx], '$'

mov bx, 0

loopc:

cmp bx, 4

je GOTIT1

mov ah, byte ptr [di + bx]

mov al, byte ptr 2[si + bx]

inc bx

cmp ah, al

je loopc

jne ITEMNOTEXIST

GOTIT1:

mov GOOD, offset GA1

lea dx, SUCCESS\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

jmp TOMENU

ITEM2CMP:

lea dx, item2Check

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

mov bx, offset GA2

mov di, bx

mov si, offset itemname

mov bl, itemname+1

mov bh, 0

mov byte ptr itemname + 2[bx], '$'

mov bx, 0

loops:

cmp bx, 5

je GOTIT2

mov ah, byte ptr [di + bx]

mov al, byte ptr 2[si + bx]

inc bx

cmp ah, al

je loops

jne ITEMNOTEXIST

GOTIT2:

mov GOOD, offset GA2

lea dx, SUCCESS\_hint

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

jmp TOMENU

ITEMNOTEXIST:

lea dx, newline

mov ah, 9

int 21h

lea dx, itemNO

mov ah, 9

int 21h

lea dx, newline

mov ah, 9

int 21h

jmp TOMENU

TOMENU:

pop si

pop di

pop dx

pop cx

pop bx

pop ax

}

if(GOOD == &GA1)

{

printf("%s\t%s\n", Iname, GA1.name);

printf("%s\t%c\n", off, GA1.off);

printf("%s\t$d\n", Price, GA1.price);

printf("%s\t%d\n", stock, GA1.stock);

printf("%s\t%d\n", cost, GA1.cost);

printf("%s\t%d\n", rec, GA1.rec);

}

else if(GOOD == &GA2)

{

printf("%s\t%s\n", Iname, GA2.name);

printf("%s\t%c\n", off, GA2.off);

printf("%s\t$d\n", Price, GA2.price);

printf("%s\t%d\n", stock, GA2.stock);

printf("%s\t%d\n", cost, GA2.cost);

printf("%s\t%d\n", rec, GA2.rec);

}

else

{

printf("%s\n", FAIL\_hint);

}

return OK;

}

### 实验记录与分析

实验环境:Windows10，DosBox0.74，BorlandC3.1

编译完C语言代码后在dosbox中执行exe文件后如图3-12:

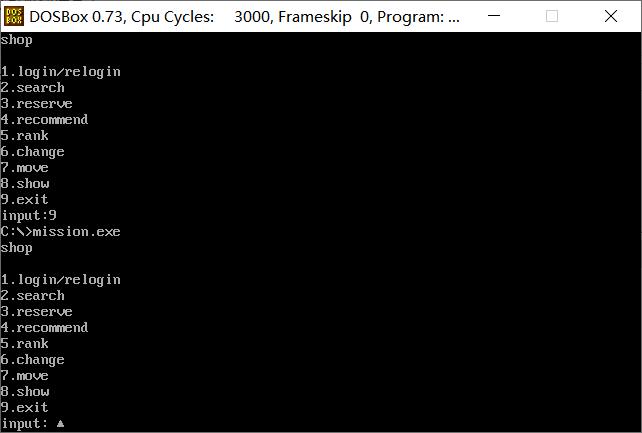


图3-12进入程序

测试登陆功能如图3-13

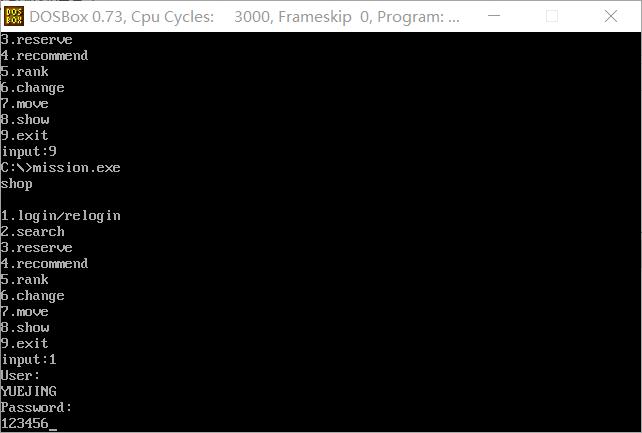


图3-13登陆

测试查找功能如图3-14

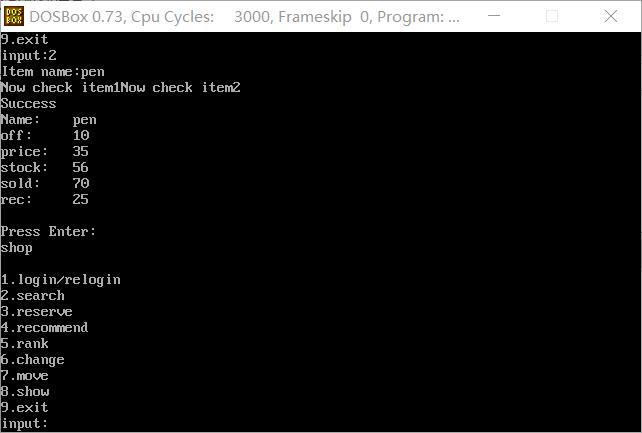


图3-14查找商品

测试下订单功能如图3-15

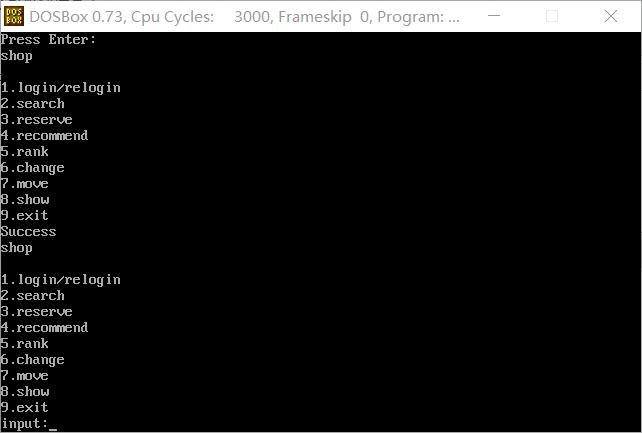


图3-15下订单

测试计算推荐度如图3-16

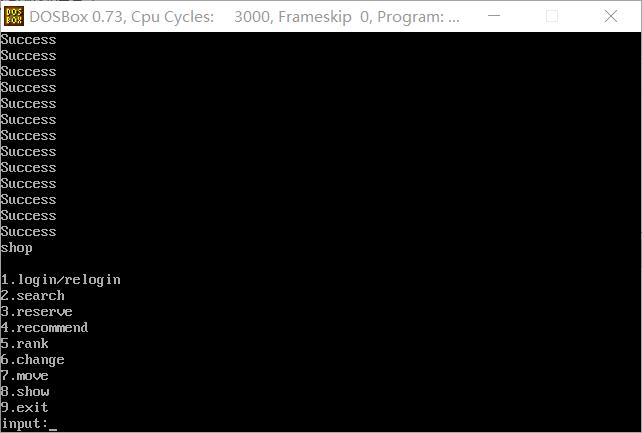


图3-16计算推荐度

测试修改商品属性如图3-17

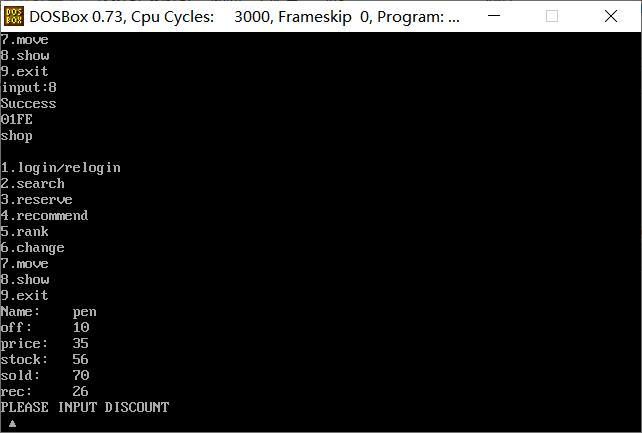


图3-17修改商品属性

## 思考题回答

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。若执行RET前把栈顶的数值改掉，那么RET执行后程序返回到何处？  
   答:使用f7可进入子程序内部；回返回到修改后的数值处
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。对一个NEAR类型子程序强制使用FAR调用（即CALL FAR PTR 子程序名）会怎样？反之，对一个FAR类型的子程序（子程序可以与主程序在同一个代码段，也可以在不同的代码段）强制使用NEAR调用又会怎样？  
   答:当调用该部分功能时会出现程序卡住无法进行任何操作的情况。
3. 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。  
   答:宏会生成对应功能的代码，子程序是直接进入相应代码段工作后便退出。  
   4.主、子程序之间参数传递的机制

通过观察反汇编代码可以得知，系统会先将需要传递的参数push进堆栈段，然后再在子程序段里面使用

5.在错误位置插入汇编代码

经过测试，我在一处传递寄存器ax的地方修改了ax中的内容，最后导致了我在之后的输出字符部分输出了乱码。

## 总结

通过本次实验的模块化设计，我对C语言的多文件项目的工作过程有了更深的理解，摸清了这种设计方式的工作过程是怎么一回事，同时这也是对模块化处理复杂问题的又一次尝试。同时汇编语言的宏命令也让我理解了C语言的宏是怎么一回事。通过这些灵活的处理手段可以让源代码文件变得长度适中，可读性良好。

混合编程实验写代码写得比较快，因为我的很多逻辑部分不需要有什么修改只需要保证在C中调用不会引起寄存器的恶性变化就好，但是在代码的调试阶段却出现了不少奇怪的问题，让我印象最为深刻的就是我在代码里面的注释会导致程序编译失败，还有就是BorlandC这种上了年纪的产品使用起来在不少地方比较反人类。这次实验我通过比较反汇编代码和之前的任务代码认识到了C和汇编的这种对应关系，让我对计算机编译过程有了更深的理解。

# 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。***（设计型实验。报告中仅需对该功能的设计实现给出完整描述）*

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。

上述任务中，需要解决的问题提示：

|  |  |  |
| --- | --- | --- |
| 任务编号 | 序号 | 需要尝试的操作 |
| 任务4.1  （IO） | 1 | 如何直接在TD下使用IN/OUT指令获取CMOS数据？ |
| 2 | 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？ |
| （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？ |
| 2 | 如何计算某个中断入口在中断矢量表内的偏移地址？ |
| 3 | 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。 |
| 4 | 程序中如何通过直接内存访问的方法获取中断入口地址？ |
| 5 | 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） |
| 6 | 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？ |
| 7 | 选择几个中断服务程序的入口地址，观察其对应的中断服务程序的代码（比如：看一下0号、2号、21号中断的处理程序干了些什么？） |
| （中断服务程序） | 1 | 有哪两种方式进入原中断服务程序？(CALL和JMP) |
| 2 | 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（比如：先不安装到中断矢量表中，仅当作子程序调用来调试；或者先安装到其他非硬件的中断号上，利用软中断来调试等。基本思路是：先把中断服务程序中与时钟、键盘操作没有直接相关的部分调试好，最后再把时钟、键盘操作相关的部分加上去）调试该中断服务程序，调试好后再安装成接管时钟或键盘中断的状态。  请给出你采用的“其他”调试方法的具体描述并实施一下。 |
| 3 | 如何判断中断服务程序已经安装过了？ |
| （内存操纵 | 1 | 如果要切换数据段和代码段，是否会有新的问题需要注意？ |
| 2 | 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？（比如，待迁移程序的堆栈段200字节，数据段509字节，代码段321字节；给你的一片存储空间为首址为2000H，大小为1100字节。） |

任务4.2：数据加密与反跟踪 （研究性实验）

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。

提示：为了使源程序的数据段中定义的密码、进货价等在汇编之后变成密文（也就是在最后交付出去的执行程序中看不到明文），可以使用数值运算符（参见教材P48）对变量的初始值进行变换。例如，如果想使进货价50变成密文，加密算法是与老板密码中的字符“W”做异或运算，则可写成：

DB 50 XOR ‘W’

任务4.3：跟踪与数据解密 （研究性实验）

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。

注意：两人一组，每人实现一套自己选择的加密与反跟踪方法，把执行程序交给对方解密（解密时间超过半小时的，说明反跟踪方法基本有效）。如何设计反跟踪程序以及如何跟踪破解，是本次实验报告中重点需要突出的内容。（分组可以按照学号顺序依次构成两人一组。也可以自行调整。如果班上人数是奇数，则三人一组，甲解密乙的，乙解密丙的，丙解密甲的。分组信息需要在实验报告的记录中给出）

上述任务4.2和4.3中，需要解决的问题提示：

若密码是用明文存放在数据段中的，如何更快地获取密码？

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？（除了用调试工具在内存中去看，还可以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）

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）

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）

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）

思考一下，如何用C语言（不嵌入汇编语言）实现反跟踪？是否能发现汇编语言的特殊之处? （在体会里具体描述）

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？

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？

接管键盘中断的方法可以用于跟踪程序，但若将接管键盘中断的方法用到反跟踪中，是否有好处？（输入密码前接管键盘中断，密码输入完毕后恢复原中断矢量。把密码字符串从明文转换成密文的转换算法写到接管后的键盘中断服务程序中，转换后的结果直接存在指定变量中，但不用破坏原键盘中断的返回值）

## 实验一过程

### 任务一实验方法说明

取出和放入中断：通过25H和35H功能实现

交换堆栈段方式：通过分别获取两个堆栈段的段首址(分别放入ss，es中)，以及偏移地址，结合sp的数值可以判断出需要复制的元素的个数，即可完成复制，之后将ss和es交换，同时sp赋正确的值。

新定义的数据如下：

SECOND DB 0

SETSECOND DB 10

TEMP DB 0

STATUS DB 0;0:stack->stackbak 1:stackbak->stack

OLD\_INT DW ?,?

;stack,stackbak的栈顶,给sp

TOPS DW ?

TOPB DW ?

TOP DW ?

;stack,stackbak偏移地址,栈底

ORIGIN1 DW ?

ORIGIN2 DW ?

AD1 DW ?

AD2 DW ?

AD DW ?;段首址

另新定义的堆栈段如下：

STACKBAK SEGMENT USE16 STACK

DB 200 DUP(1)

STACKBAK ENDS;

### 流程图

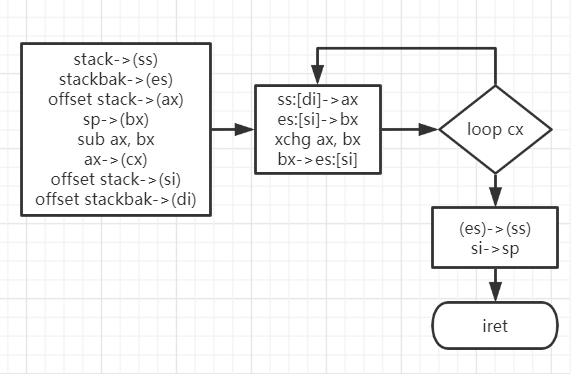


图4-1

### 关键代码

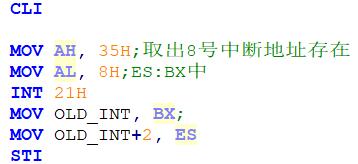


图4-2取8号向量

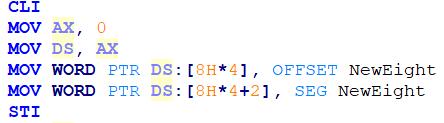


图4-3存8号向量

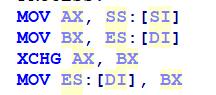


图4-4复制堆栈元素

### 实验记录与分析

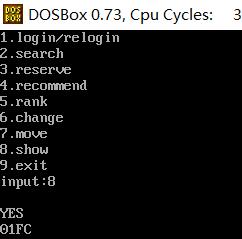


图4-5交换前

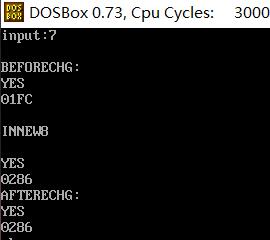


图4-6实现交换

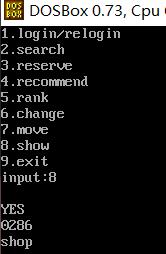


图4-7交换后

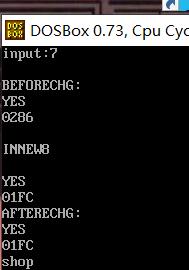


图4-8交换回来

## 实验二过程

### 实验方法说明

1. 使用xor方式将姓名/进货价加密，使用减法乘法方式将密码加密。
2. 用间接转移和计时的方法反跟踪。

### 流程图

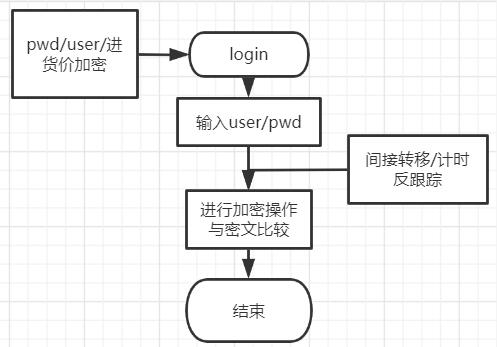


图4-9流程图

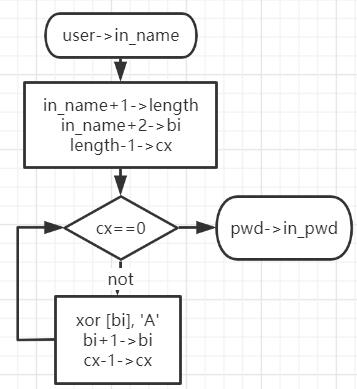
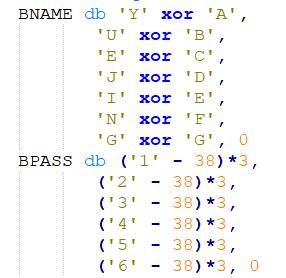
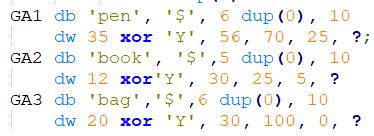


图4-10姓名加密过程

### 关键代码

数据段中定义：





代码段中计时反跟踪：

cli

;计时反跟踪

MOV AH, 2CH

INT 21H

PUSH DX;RESTORE SEC AND SEC/100

;the right namelength = 8

mov cl, in\_name + 1

mov ch, 0

cmp cx, 7

jne Custom1

pusha

push ax

pop ax

popa

mov si, offset in\_name

mov di, offset BNAME

MOV AH, 2CH

INT 21H

STI

CMP DX, [ESP]

POP DX

JNZ Custom1

mov bx, 0 ;length flag

xor in\_name + 2, 'A'

xor in\_name + 3, 'B'

xor in\_name + 4, 'C'

xor in\_name + 5, 'D'

xor in\_name + 6, 'E'

xor in\_name + 7, 'F'

xor in\_name + 8, 'G'

间接转移：

PUSHA

PUSH cx

pop ax

popa

mov ax, name\_check

call ax

MOV AH, AUTH

CMP Ah, 0

JE Custom

;password

INPWD in\_pwd, newline, password\_hint

PUSHA

PUSH cx

pop ax

popa

mov bx, pwd\_check

### 实验记录与分析

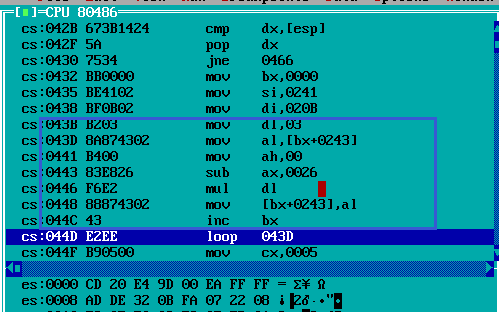


图4-11密码加密部分

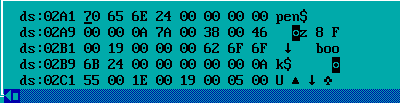


图4-12数据段加密部分

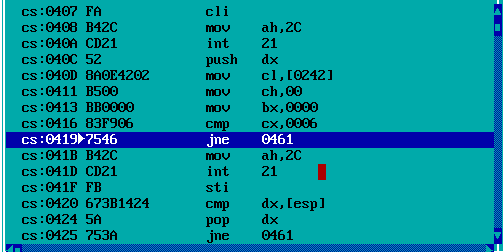


图4-13计时反跟踪

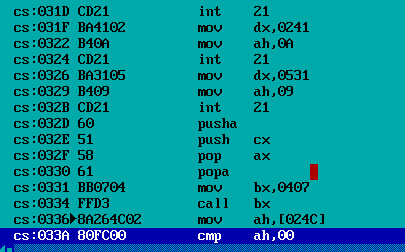


图4-14间接调用

## 实验三过程

### 实验方法说明

搭档:黄俊淇

实验方法：使用TD调用lab04.exe文件，使用单步调试进入到输入密码的部分代码段，比较其中的数据和对应的字符查找可能的密文。

### 实验记录与分析

通过查找到姓名可以大概判断姓名后的字符串应该就是密文。

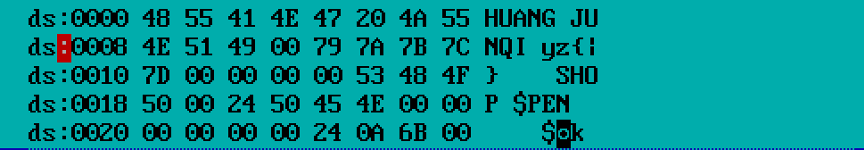


图4-15暴力搜索数据段

通过f7单步执行到输入姓名和密码部分后可以追踪到其进行姓名比较和密码比较的代码段，就可以搜索到其进行加密的部分。

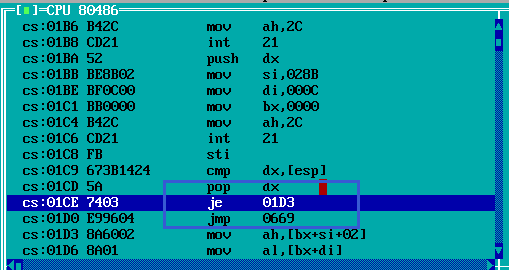


图4-16查找到其计时反跟踪位置

同过将条件跳转转换为直接的jmp 01D3就可以访问到其进行加密的程序部分。

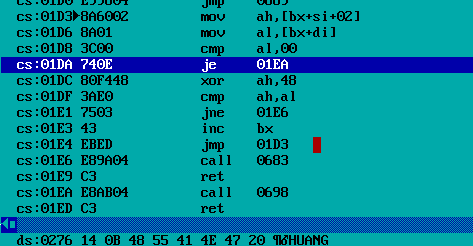


图4-17获得起加密方式

由此可推断出其加密方式是xor 48h，所以可以算出其密码为12345

通过对功能2显示商品信息部分的追踪，发现进货价的加密方式也是如此。

使用解密后的密码进行登陆:

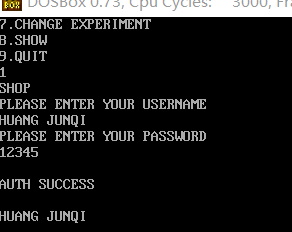


图4-18登陆成功

## 思考题

### 任务4.1

1.进入td查看中断向量位置

通过goto功能进入0:num\*4即可查看到其位置

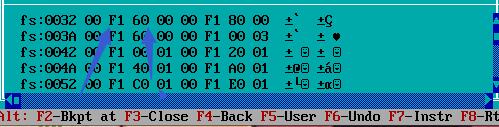


图4-9 8号位置

2.修改中断向量后执行其他功能

修改8号向量表后如果使用dos10号命令会进入新的8号向量中断程序中造成输入错误

3.判断是否被安装  
 程序开始时记录旧的地址中的值，在之后比较现在地址中的值和其是否相等，不等则说明已经安装成功

### 任务4.2和4.3

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？

可以，因为当AUTH=1时程序会示为老板，则可以直接输出商品信息。

2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？

用二进制工具打开后通过将已经商品名信息转换为二进制进行查找商品名，之后则可根据进货价在商品名后第一个得到进货价。

3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?

如果是计时反跟踪程序，则可通过观察cli/sti以及mov ah，2ch来注意是否已经到了反跟踪部分，然后通过将cmp 时间1，时间2的部分通过TD直接修改就可以绕开反跟踪。

## 总结

### 任务4.1

认识到了中断的意义，即能够让cpu准确快速切换功能。并且中断向量表的实现也让我想起了数据结构的内容，可以说是让我对目前所学的计算机知识有了一份全新的体会。同时本次任务中间遇到的困难也让人难以忘记，一是没有在dos10号命令前还原中断向量导致程序直接卡死，二是切换堆栈段时一开始没有使用寄存器保存偏移地址而是使用了变量导致只能复制一行元素。这两个很大的坑浪费了我很多时间，但是也让我对汇编的设计等有了更深的认识。

### 任务4.2和4.3

第一次真正将加密方式应用在自己的程序中，体验十分新奇，虽然没能力用汇编语言实现一些经典加密算法。希望能在之后的学习和工作中体验到更高级的加密。同时反汇编破解别人的加密也十分有趣，发现别人的反跟踪程序和想办法跳过它都很有意思。

# WIN32/64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）。

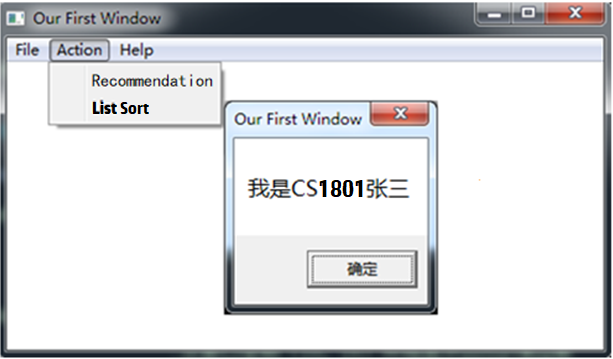


图5.1 菜单示例

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

上述任务中，需要解决的问题的提示：

1. 选择并安装开发环境。汇编答疑QQ群中有一些软件包、操作指南文件可以下载，也可以在网上下载。大家可以选择VS2019、VC、MASM32等软件包完成本次任务。建议将本次任务所选择的开发环境与最后一次实验（任务3.2，与C语言混合编程）的开发环境结合起来考虑，可以减少总体工作量。

2. 对开发包自带（或他人编写好的）的示例程序，进行汇编、连接、运行和调试。观察WIN32/64执行程序代码的特点和执行流程。体会基于窗口的应用程序所包含的四个部分之间的衔接关系。

3. 观察32/64位下调试工具与16位TD的异同。

4. 调试WIN32/64程序与16位段程序的主要差异是什么？

5. 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。

6. 用调试工具观察代码区或数据区时，若所观察的地址范围不是与被调试程序相关的区间，则对应内存中的数据会因为被系统保护了而读不出来（将用“？”或其他符号代替）。请通过修改偏移地址来改变观察的区间，记录此现象。

7. 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。

8. 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。

9. 观察结构变量的推荐度等字段的偏移，体会结构变量优点。

10. 观察简化段的效果。

11. 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。

12.本次编写的程序不要求区分WIN32还是WIN64程序，大家有兴趣的话可以进行探究。

13.在实验方法说明中，要给出List Sort的处理流程。鉴于网络考试平台里画图不方便，需要用“处理步骤描述法”替代流程图，因此，List Sort的流程不用流程图描述，要采用处理步骤描述法描述。下面是一个处理步骤描述法的示例：(小5号字)

计算立方值的程序流程步骤：

（1）提示用户从键盘输入一个数字。

（2）用1号系统功能调用从键盘接收一数字x的ASCII码。

（3）是否是0~9之间的数字？是则转到（4），否则转到（1）。

（4）x的真值 → AL → X。

（5）x的真值 → EBX。

（6）查立方值表（TAB + [2 \* EBX]） → XXX。

（7）程序退出。

## 实验过程

### 实验方法说明

实验环境条件：16G内存；Windows10 ,使用Visual C++6.0搭配MASM32作为开发工具。

根据IDE提示以及课本学习内容,在任务3.1程序的基础上进行重构移植到win32下。项目代码文件包含, 5.asm, menu.rc, menuID.inc等等文件。

### 结构说明

以任务3.1的代码为基础，定义了结构体goods，该结构体包含了物品的名称，折扣等所有信息，并定义了6个实例，如下:

buf goods <'pen', 10, 35, 56, 70, 25, 0>

goods <'book', 9, 12, 30, 25, 5, 0>

goods <'bag', 9, 40, 100, 45, 5, 0>

goods <'note',5,30,50,30,25, 0>

goods <'milk',8,40,56,80,35,0>

goods <'banana',10,44,76,70,25,0>

同时为了在对应框中使用TextOut输出内容还定义了相应属性名的字符串：

msg\_name db 'Name',0

msg\_discount db 'Discount',0

msg\_cost db 'Cost',0

msg\_price db 'Price',0

msg\_quantity db 'Quantity',0

msg\_sold\_num db 'Sold\_num',0

msg\_recommend\_level db 'Recommend\_level', 0

又因为输出的值需要为对应的字符，所以定义了stringbuf变量来保存转换以及函数ConvertString来将数字转换为字符。

Menu.rc文件在VC++上通过直接拖动定义控件生成。分别设置File，Action，help三个主菜单，和clear，exit，recommende，list sort，about几个子菜单,功能分别:clear清空目前显示的内容，exit退出程序，recommend计算推荐度并显示，list sort输出排序好后的商品信息，about显示开发者信息。

关于排序部分:

1. 计算出所有商品推荐度,0->dx，0->bx
2. 从第[dx+bx]个商品开始，比较相邻两商品的推荐度大小,inc bx。
3. 若后者大于前者，将两结构体的所有属性交换.
4. 比较完最后一个商品时若dx不为6，inc dx并重复步骤2.

### 关键代码

冒泡排序:

mov ebx,0

mov dx,0

push dx

loopn:

pop dx

mov ebx,-22

add dx,1

cmp dx,7

JE L3

push dx

loopv:

add ebx,22

add dx,1

mov ax,7

cmp dx,ax

je loopn

mov ax,word ptr buf[ebx].recommend

cmp ax,word ptr buf[ebx+22].recommend

jae loopm

jmp loopv

loopm:

mov ax,word ptr buf[ebx].recommend

mov cx,word ptr buf[ebx+22].recommend

mov word ptr buf[ebx+22].recommend,ax

mov word ptr buf[ebx].recommend,cx

mov ax,word ptr buf[ebx].discount

mov cx,word ptr buf[ebx+22].discount

mov word ptr buf[ebx+22].discount,ax

mov word ptr buf[ebx].discount,cx

mov ax,word ptr buf[ebx].cost

mov cx,word ptr buf[ebx+22].cost

mov word ptr buf[ebx+22].cost,ax

mov word ptr buf[ebx].cost,cx

mov ax,word ptr buf[ebx].price

mov cx,word ptr buf[ebx+22].price

mov word ptr buf[ebx+22].price,ax

mov word ptr buf[ebx].price,cx

mov ax,word ptr buf[ebx].stock

mov cx,word ptr buf[ebx+22].stock

mov word ptr buf[ebx+22].stock,ax

mov word ptr buf[ebx].stock,cx

mov ax,word ptr buf[ebx].sold

mov cx,word ptr buf[ebx+22].sold

mov word ptr buf[ebx+22].sold,ax

mov word ptr buf[ebx].sold,cx

mov ecx,0

loopq:

mov al,byte ptr buf[ebx+ecx+22].itemName

mov ah,0

push ax

mov al,byte ptr buf[ebx+ecx].itemName

mov byte ptr buf[ebx+22+ecx].itemName,al

pop ax

mov byte ptr buf[ebx+ecx].itemName,al

add ecx,1

cmp ecx,9

jne loopq

JMP loopv

转换为字符:

ConvertString proc num:word

pusha

mov ax, num

mov esi, offset stringBuf

mov ebx, 0

clear\_buf:

mov byte ptr[esi+ebx], ' '

inc ebx

cmp ebx, 100

jne clear\_buf

mov bx, 10

mov cx, 0

loopm:

mov dx, 0

div bx

add dl, '0'

push dx

inc cx

cmp ax, 0

jne loopm

loopn:

pop dx

mov byte ptr[esi], dl

inc esi

dec cx

cmp cx, 0

jne loopn

popa

ret

ConvertString endp

### 实验记录与分析

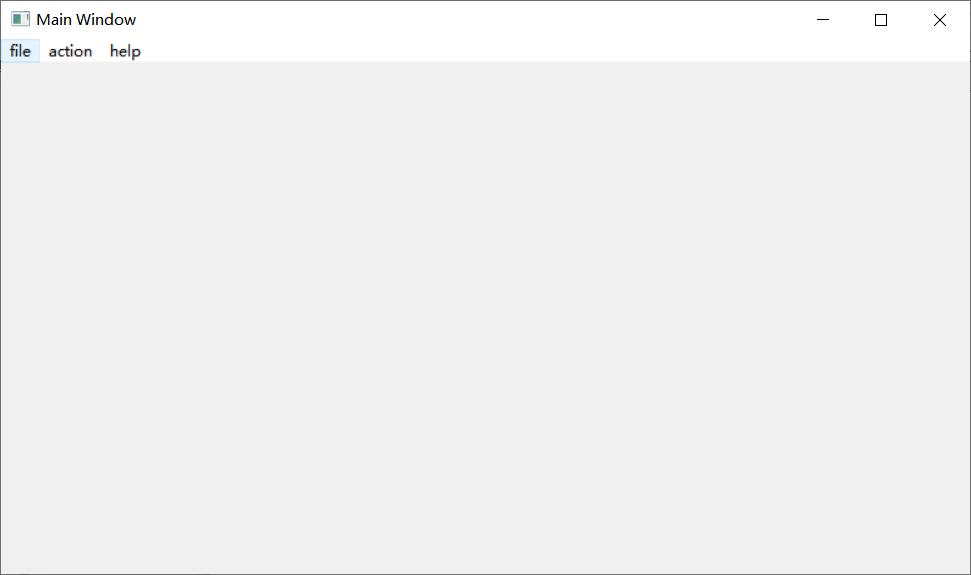


图5-1进入程序

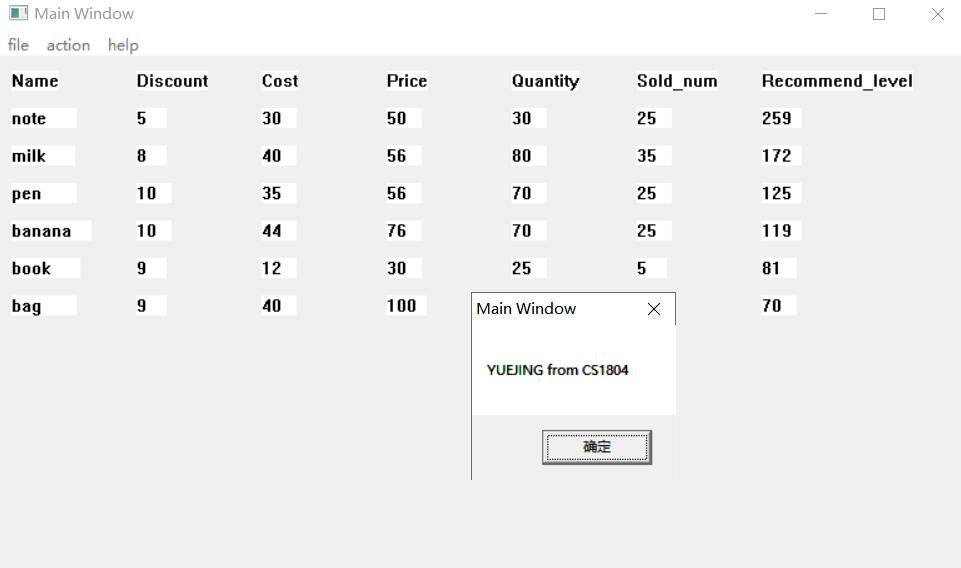


图5-2输出个人信息

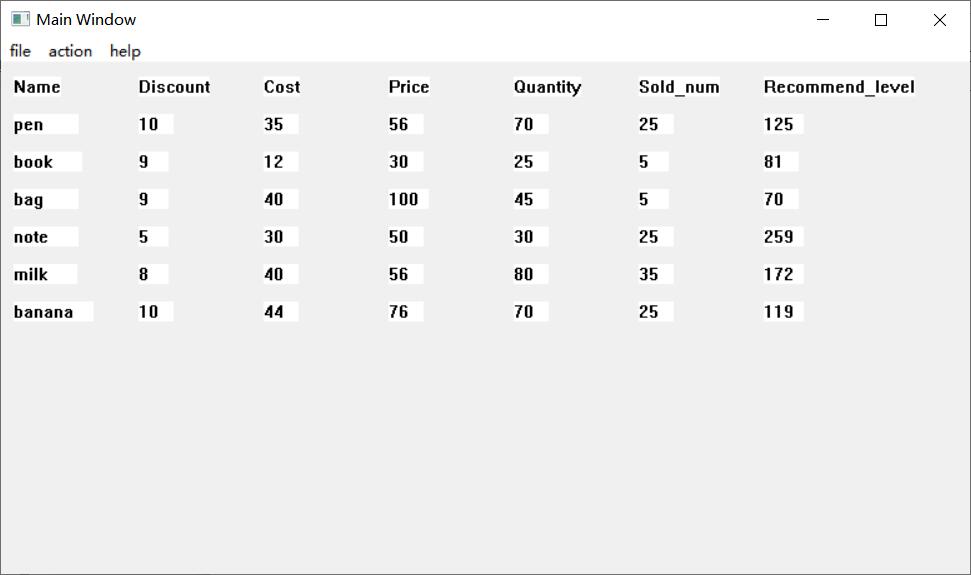


图5-3计算推荐度并输出

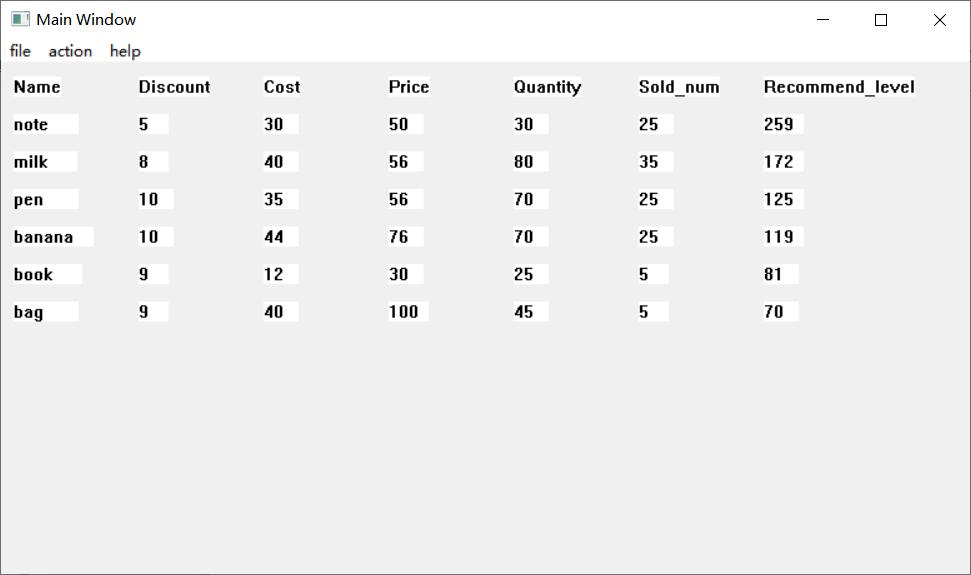


图5-4排序后输出

## 思考题

1.VC下的调试界面

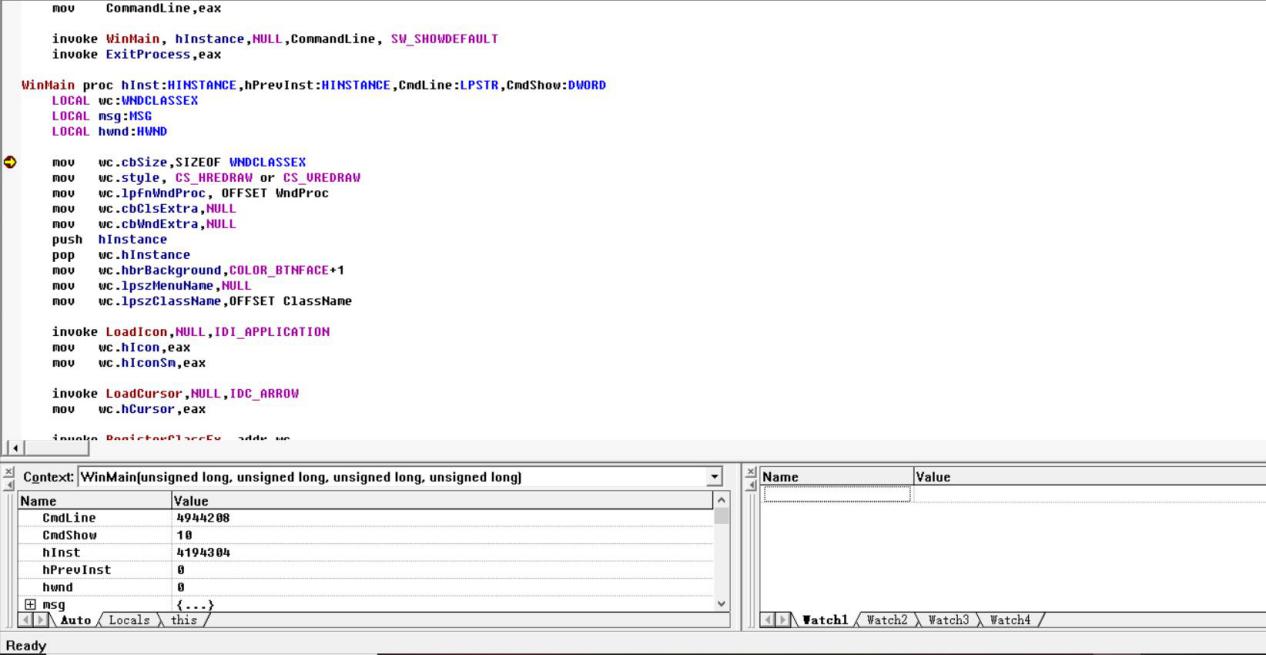


图5-5VC调试界面

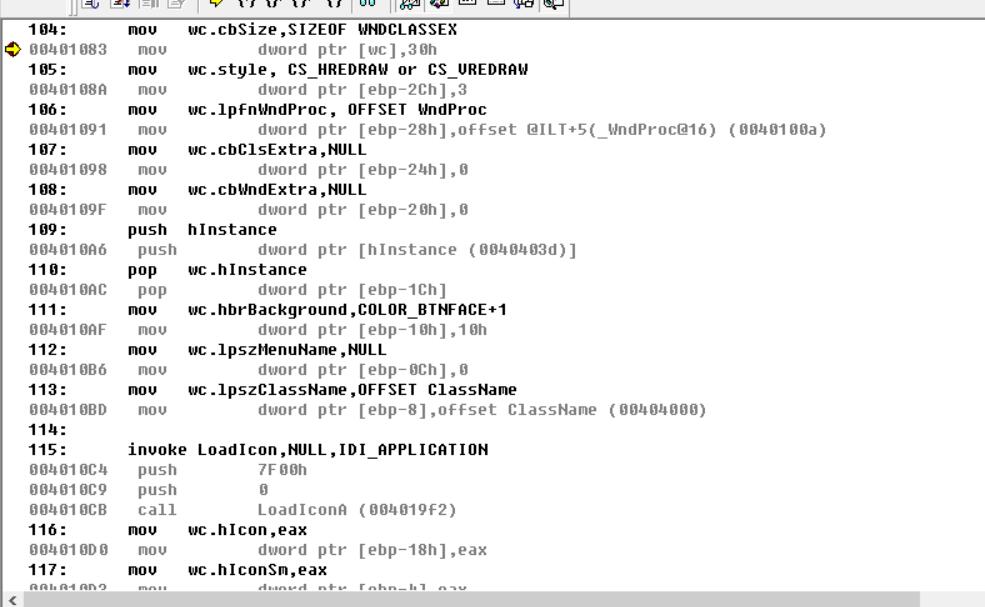


图5-6VC查看反汇编

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

Win32下的输出可使用TextOut函数，该函数比较起使用dos的9号功能可谓是方便了许多倍。同时在win32下字符串也不再需要使用$符号结尾。

## 小结

1. 我在获得任务书时就开始尝试在VS上进行本次任务，但是当时由于设置问题一直不能正确地运行汇编程序，索性就下载了VC配合masm32使用。
2. 本次任务下的win32编程需要使用到windows的gui部分，初上手时感觉比较困难，因为从前没开发过win32程序，但是好在MSDN等文档都十分健全，在gui部分还是比较畅通。
3. 本次实验总的来说因为不是很习惯32位下的很多寄存器等的使用导致部分功能设计得比较粗糙，但是也算是了解到了汇编这门底层语言的上限有多高，虽然操作起来比较繁琐，但确实汇编在操作系统上无所不能，值得之后有机会继续实践。