

**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： U201814604**

**姓 名： 黄俊淇**

**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： 南宁市八桂绿城**

**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：

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 - 4 -](#_Toc41645355)

[0.1 课程目标 - 4 -](#_Toc41645356)

[0.2 成绩构成 - 4 -](#_Toc41645357)

[0.3 实验任务的总体描述 - 4 -](#_Toc41645358)

[1 编程基础 1](#_Toc41645359)

[1.1 实验目的与要求 1](#_Toc41645360)

[1.2 实验内容 1](#_Toc41645361)

[1.3 任务1.1实验过程 3](#_Toc41645362)

[1.3.1 实验方法说明 3](#_Toc41645363)

[1.3.2 实验记录与分析 5](#_Toc41645364)

[1.4 任务1.2实验过程 8](#_Toc41645365)

[1.4.1 实验方法说明 8](#_Toc41645366)

[1.4.2 实验记录与分析 8](#_Toc41645367)

[1.5 任务1.3实验过程 9](#_Toc41645368)

[1.5.1 实验方法说明 9](#_Toc41645369)

[1.5.2 实验记录与分析 10](#_Toc41645370)

[1.6 任务1.4实验过程 11](#_Toc41645371)

[1.6.1 设计思想及存储单元分配 11](#_Toc41645372)

[1.6.2 流程图 13](#_Toc41645373)

[1.6.3 源程序 16](#_Toc41645374)

[1.6.4 实验步骤 26](#_Toc41645375)

[1.6.5 实验记录与分析 26](#_Toc41645376)

[1.7 总结 29](#_Toc41645377)

[2 程序优化 31](#_Toc41645378)

[2.1 实验目的 31](#_Toc41645379)

[2.2 实验内容 31](#_Toc41645380)

[2.3 任务2.1实验过程 32](#_Toc41645381)

[2.3.1 设计思想与存储单元分配 32](#_Toc41645382)

[2.3.2 流程图 32](#_Toc41645383)

[2.3.3 源程序 33](#_Toc41645384)

[2.3.4 实验步骤 35](#_Toc41645385)

[2.3.5 实验记录与分析 35](#_Toc41645386)

[2.4 任务2.2实验过程 37](#_Toc41645387)

[2.4.1 实验方法说明 37](#_Toc41645388)

[2.4.2 实验记录与分析 37](#_Toc41645389)

[2.5 总结 38](#_Toc41645390)

[3 模块化程序设计 40](#_Toc41645391)

[3.1 实验目的 40](#_Toc41645392)

[3.2 实验内容 40](#_Toc41645393)

[3.3 任务3.1实验过程 41](#_Toc41645394)

[3.3.1 设计思想与存储单元分配 41](#_Toc41645395)

[3.3.2 模块结构 42](#_Toc41645396)

[3.3.3 源程序 42](#_Toc41645397)

[3.3.4 实验步骤 46](#_Toc41645398)

[3.3.5 实验记录与分析 46](#_Toc41645399)

[3.4 任务3.2实验过程 48](#_Toc41645400)

[3.4.1 实验方法说明 48](#_Toc41645401)

[3.4.2 实验记录与分析 49](#_Toc41645402)

[3.5 小结 52](#_Toc41645403)

[4 中断与反跟踪 53](#_Toc41645404)

[4.1 实验目的 53](#_Toc41645405)

[4.2 实验内容 53](#_Toc41645406)

[4.3 任务4.1实验过程 54](#_Toc41645407)

[4.3.1 设计思想与存储单元分配 54](#_Toc41645408)

[4.3.2 流程图 54](#_Toc41645409)

[4.3.3 源程序 55](#_Toc41645410)

[4.3.4 实验步骤 58](#_Toc41645411)

[4.3.5 实验记录与分析 58](#_Toc41645412)

[4.4 任务4.2实验过程 59](#_Toc41645413)

[4.4.1 设计思想与存储单元分配 59](#_Toc41645414)

[4.4.2 流程图 59](#_Toc41645415)

[4.4.3 源程序 60](#_Toc41645416)

[4.4.4 实验步骤 61](#_Toc41645417)

[4.4.5 实验记录与分析 61](#_Toc41645418)

[4.5 任务4.3实验过程 62](#_Toc41645419)

[4.5.1 实验方法说明 62](#_Toc41645420)

[4.5.2 实验记录与分析 62](#_Toc41645421)

[4.6 总结 63](#_Toc41645422)

[5 WIN32/64编程 65](#_Toc41645423)

[5.1 实验目的 65](#_Toc41645424)

[5.2 实验内容 65](#_Toc41645425)

[5.3 任务5.1实验过程 66](#_Toc41645426)

[5.3.1 实验方法说明 66](#_Toc41645427)

[5.3.2 实验记录与分析 68](#_Toc41645428)

[5.4 总结 69](#_Toc41645429)

[参考文献 70](#_Toc41645430)

# 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。

# 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。

**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。

**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 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。

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, +0110011B；MOV AL, +1011010B；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。

3. 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, -0101001B；MOV AL, -1011101B；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。

4. 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, +1100101B；MOV AL, -1011101B；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。

以下是实验结果预测。

1. 如图表1.1

表1.1第一问

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | AH | SF | OF | CF | ZF |
| 求和 | 8D | 1 | 1 | 0 | 0 |
| 求差 | D9 | 1 | 0 | 1 | 0 |

1. 如图表1.2

表1.2第二问

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | AH | SF | OF | CF | ZF |
| 求和 | 7A | 0 | 1 | 1 | 0 |
| 求差 | 34 | 0 | 0 | 0 | 0 |

1. 如图表1.3

表1.3第三问

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | AH | SF | OF | CF | ZF |
| 求和 | 8 | 0 | 0 | 1 | 0 |
| 求差 | C2 | 1 | 1 | 1 | 0 |

### 实验记录与分析

(1)mov ah,0110011B，如图1.1所示

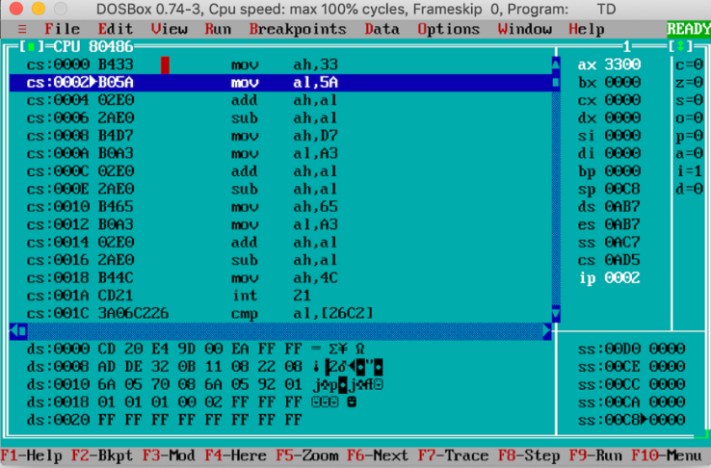


图1.1mov指令

(2)mov al,1011010B，如图1.2所示

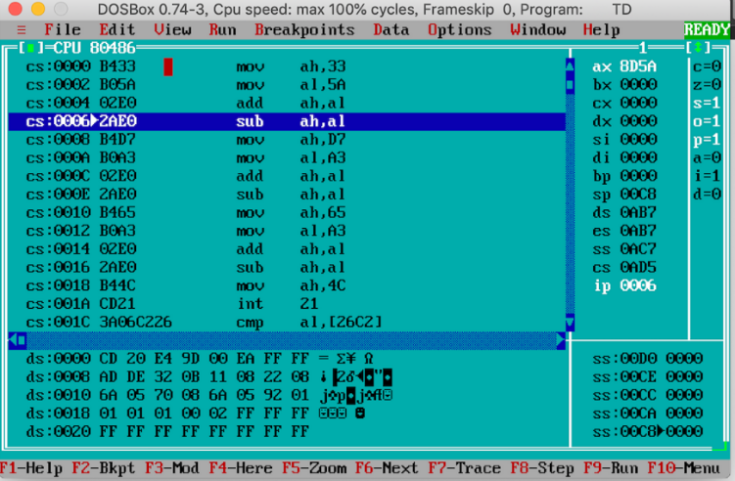


图1.2mov指令

(3)add ah,al，如图1.3所示

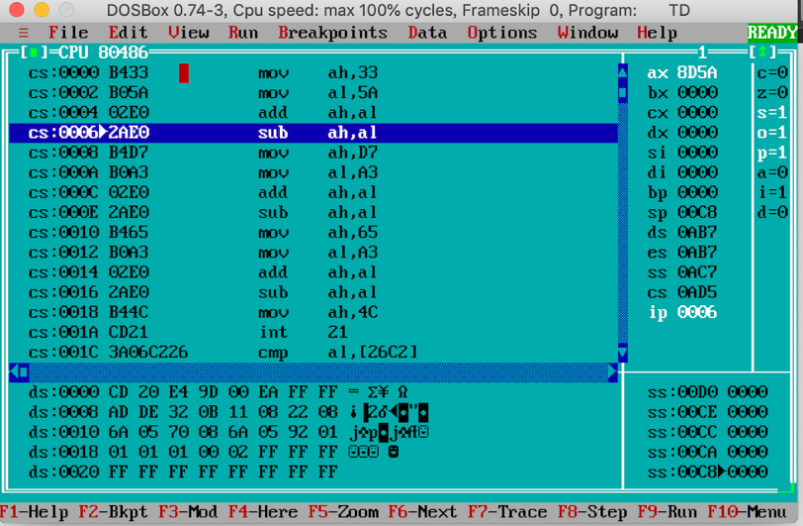


图1.3add指令

(4)sub ah,al，如图1.4所示

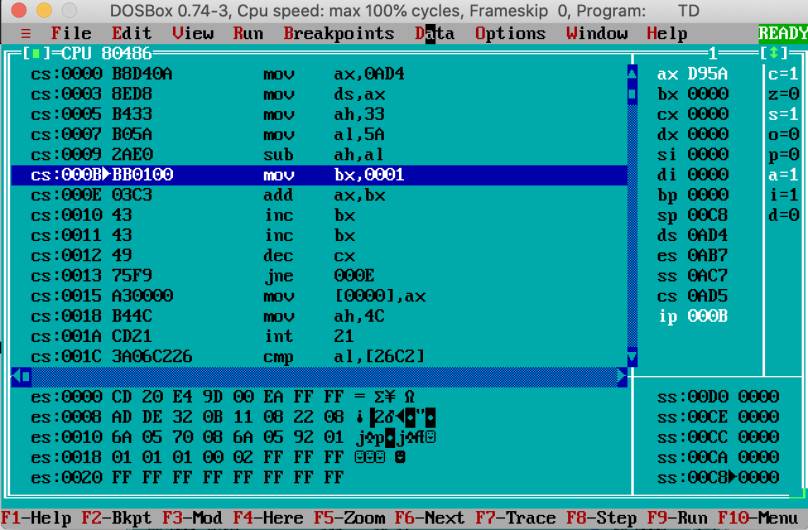


图1.4sub指令

(5)类似的，对-0101001B，-1011101B求和，如图1.5所示

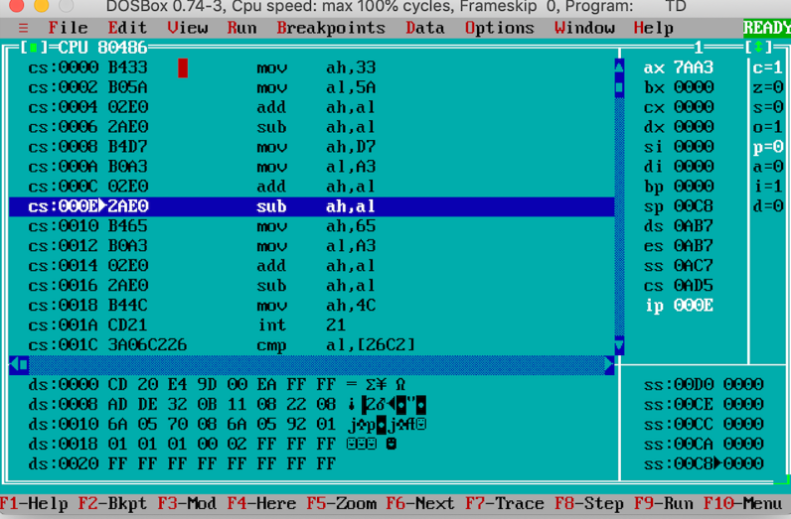


图1.5求和

(6) 对-0101001B，-1011101B求差，如图1.6所示

电脑萤幕画面

描述已自动生成

图1.6求差

(7)对1100101B，-1011101B求和，如图1.7所示

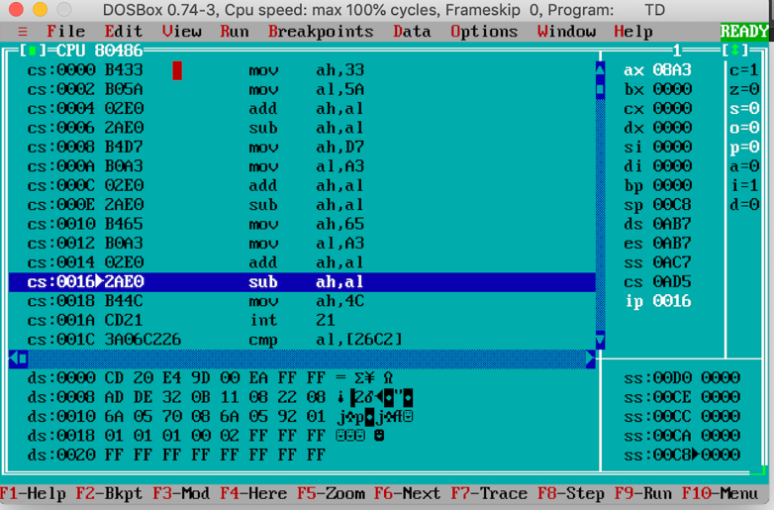


图1.7求和

(8) 对1100101B，-1011101B求差，如图1.8所示

电脑萤幕画面

描述已自动生成

图1.8求差

（9）如表1.4所示

表1.4预测与实验对比

|  |  |  |
| --- | --- | --- |
| 第（1）问 | 预测 | 实验 |
| 求和 | 8D | 8D |
| 求差 | D9 | D9 |
| 第（2）问 | 预测 | 实验 |
| 求和 | 7A | 7A |
| 求差 | 34 | 34 |
| 第（3）问 | 预测 | 实验 |
| 求和 | 8 | 8 |
| 求差 | C2 | C2 |

综上，实验所得与预测一致。

## 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。

以下是对实验结果的预测。

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

### 实验记录与分析

(1)mov cx，10之前的内容，如图1.9所示

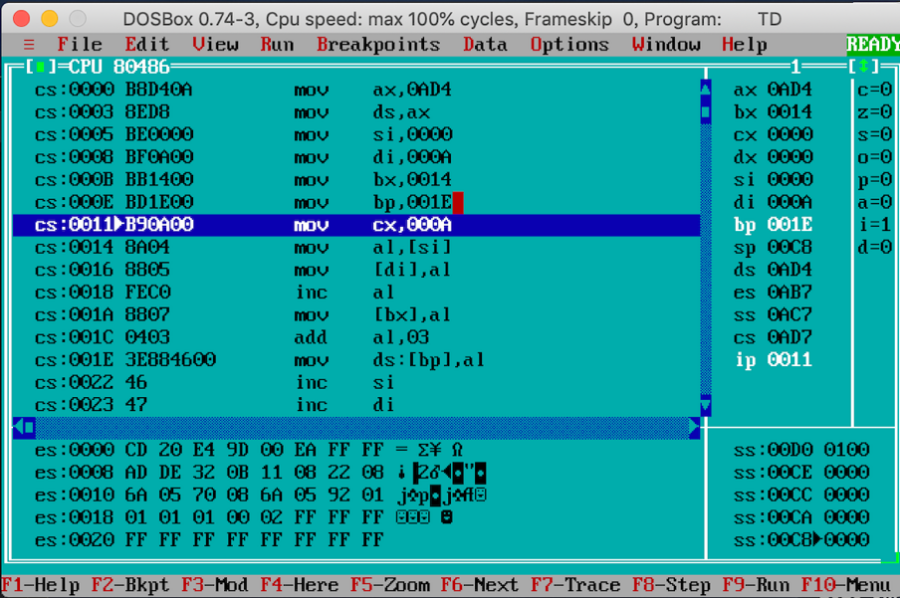


图1.9 mov cx，10之前的内容

(2)int 21H之前的内容，如图10所示

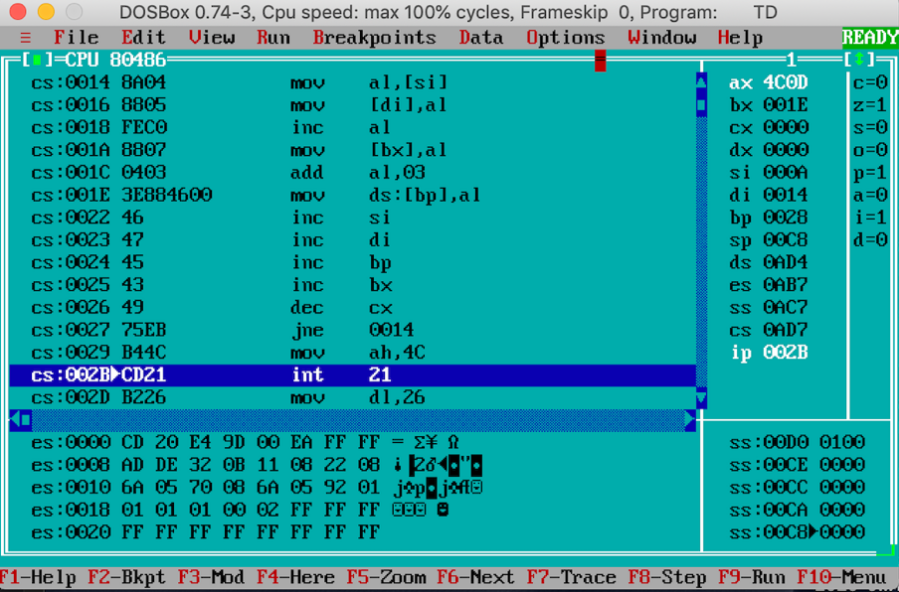


图1.10 int 21H之前的内容

(3)40个字节内容如图1.11所示

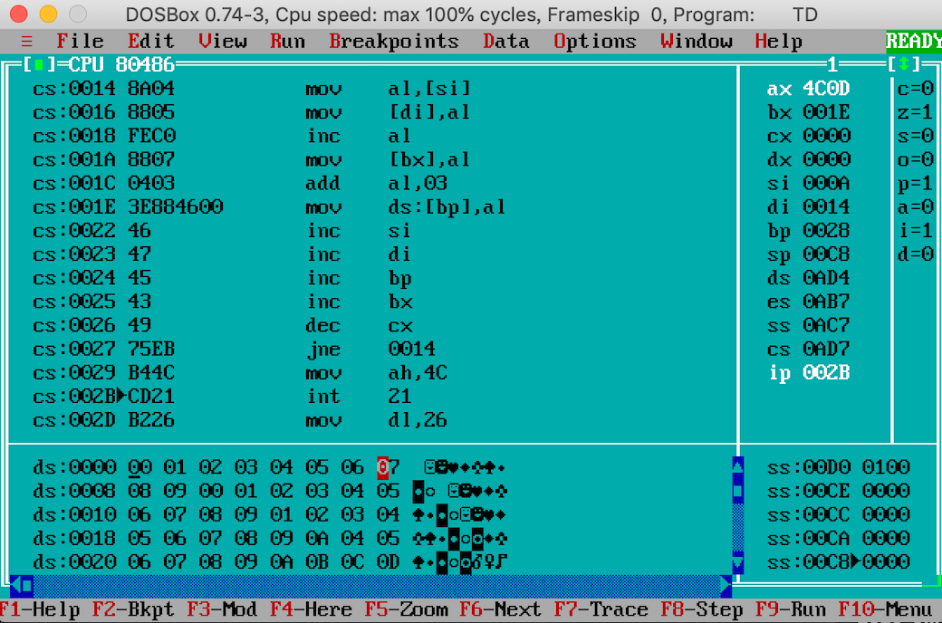


图1.11 40个字节内容

与设想一致

## 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。

给出部分代码：

LOOPA: MOV AL,BUF1[ESI]

MOV BUF2[ESI],AL

INC AL

MOV BUF3[ESI],AL

ADD AL,3

MOV BUF4[ESI],AL

INC ESI

DEC ECX

JNZ LOOPA

MOV AH,4CH

INT 21H

以下是对戒烟结果的预测

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

### 实验记录与分析

(1)16位寄存器，如图1.12所示

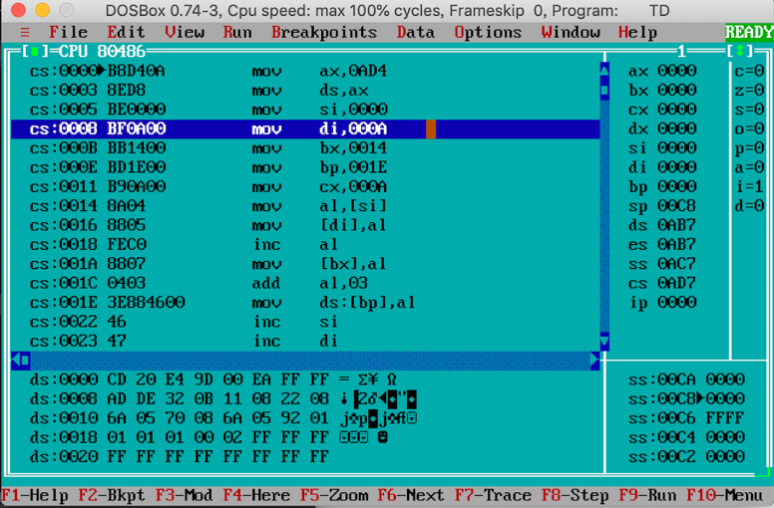


图1.12 16位寄存器

(2)32位寄存器，如图1.13所示

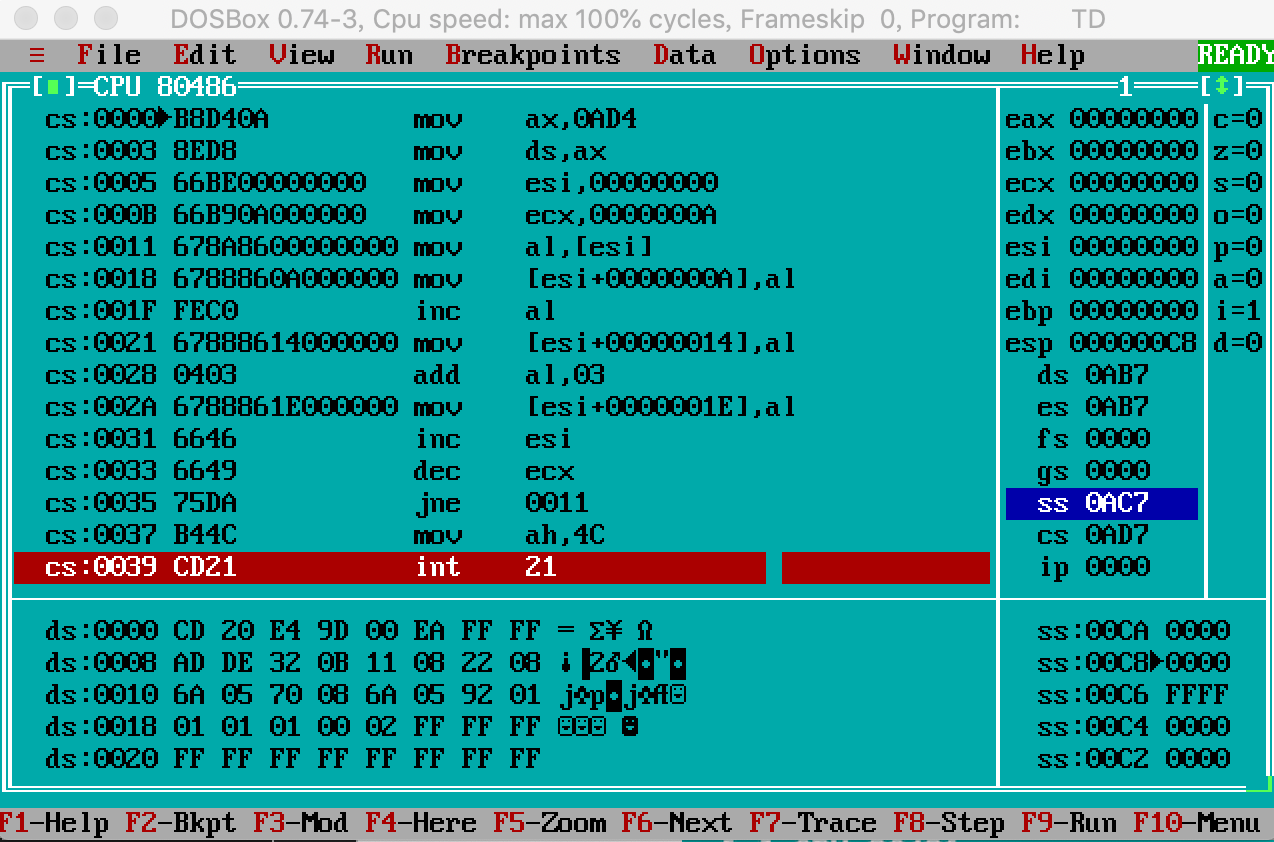


图1.13 32位寄存器

(3)40个字节内容如图1.14所示

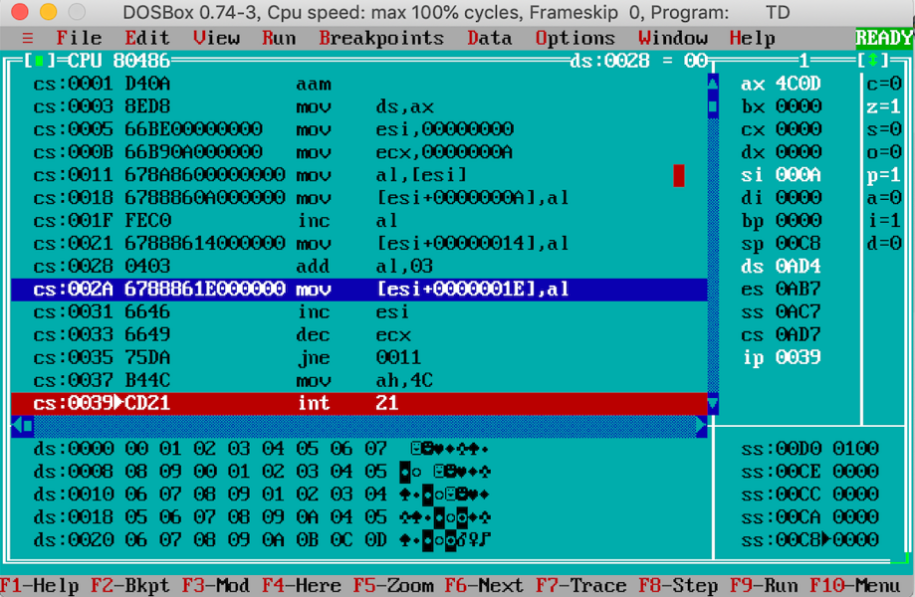


图1.14 40个字节内容

与设想一致

（4）通过观察可以看到，从不同字节开始反汇编，指令结果也不一样

## 任务1.4实验过程

### 设计思想及存储单元分配

1、数据段定义

DATA SEGMENT USE16

BNAME DB 'HUANG JUNQI',0

BPASS DB '12345',0

AUTH DB 0

GOOD DW 0,'$'

N EQU 30

SNAME DB 'SHOP',0,'$'

GA1 DB 'PEN',7 DUP(0),10

DW 35, 56 ,70 ,25 ,?

GA2 DB 'BOOK',6 DUP(0),9

DW 12,30,25,25,?

in\_name DB 20,20 DUP(0)

in\_pwd DB 10,10 DUP(0)

option1 DB 2 dup(0),'$'

menu\_hint1 DB '1.sign in','$'

menu\_hint2 DB '2.serch goods','$'

menu\_hint3 DB '3.order','$'

menu\_hint4 DB '4.caulate','$'

menu\_hint5 DB '5.sort','$'

menu\_hint6 DB '6.revise','$'

menu\_hint7 DB '7.change experiment ','$'

menu\_hint8 DB '8.show','$'

menu\_hint9 DB '9.quit','$'

number\_hint DB 'Please choose a number to function','$'

username\_hint DB 'Please enter your username', '$'

password\_hint DB 'Please enter your password', '$'

success\_hint DB 'Auth success', '$'

query\_hint DB 'Please enter the item name you want to query','$'

fail\_hint DB '404','$'

in\_item DB 80,0,80 dup(0)

TEMP DB 0,0,0,0,'$'

2、主系统界面

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。

3、功能一

（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：当碰到数字0，结束，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。

4、功能二

（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。

5、功能三

（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。

6、功能四

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。

7、功能八

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。

8、功能九

退出系统。

### 流程图

1. 主系统界面，如图1.15所示

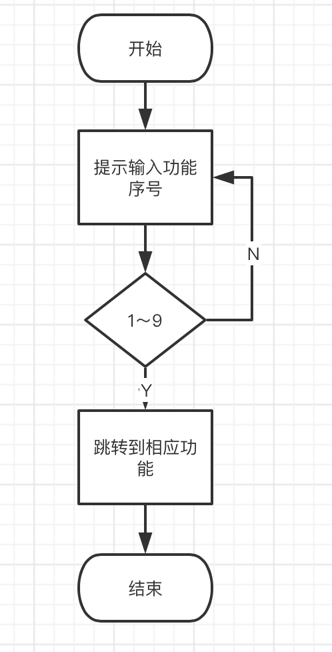


图1.15主系统界面

1. 功能一，比较用户名如图1.16所示，比较密码如图1.17所示

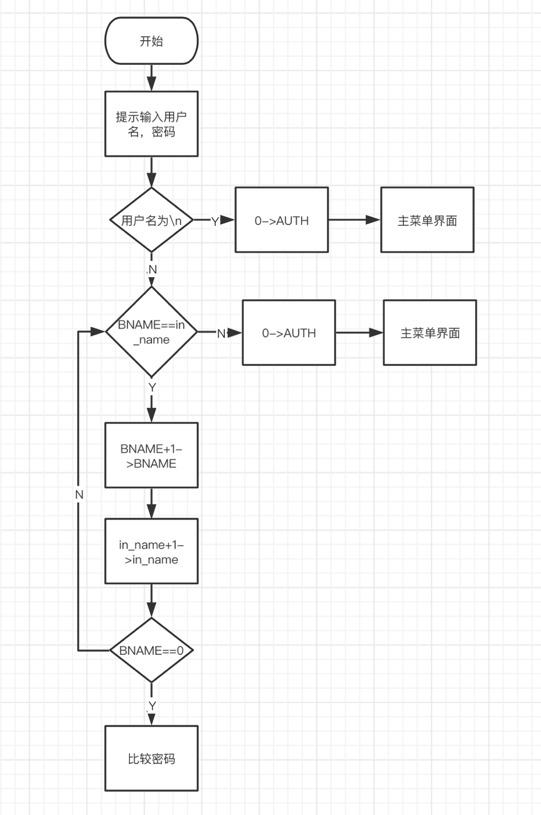


图1.16比较用户名

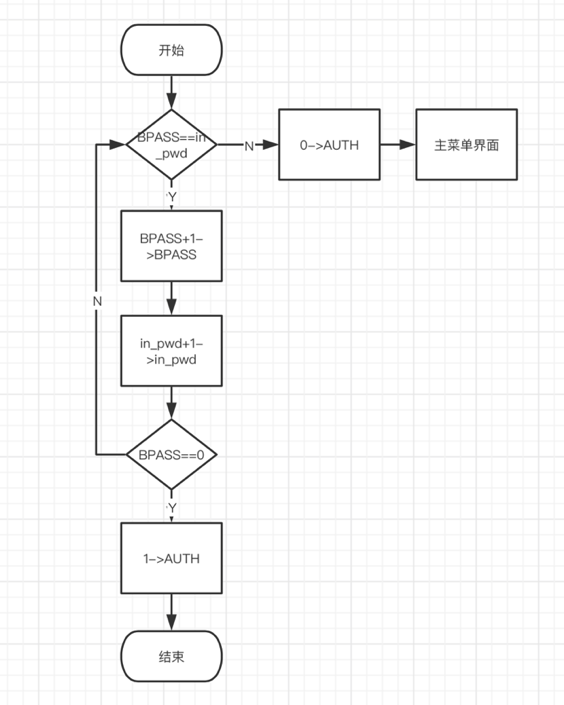


图1.17比较密码

1. 功能二，如图1.18所示

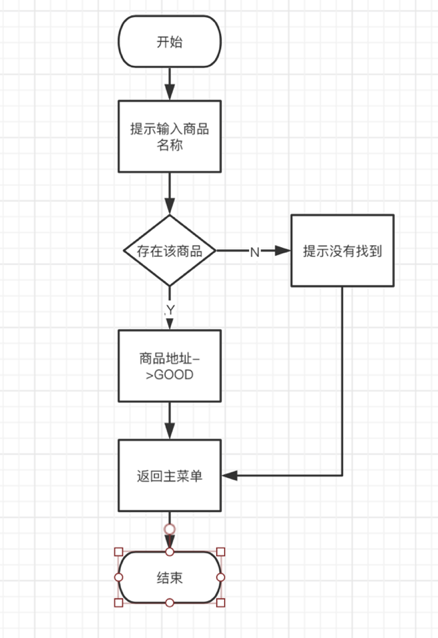


图1.18功能二

1. 功能三，如图1.19所示

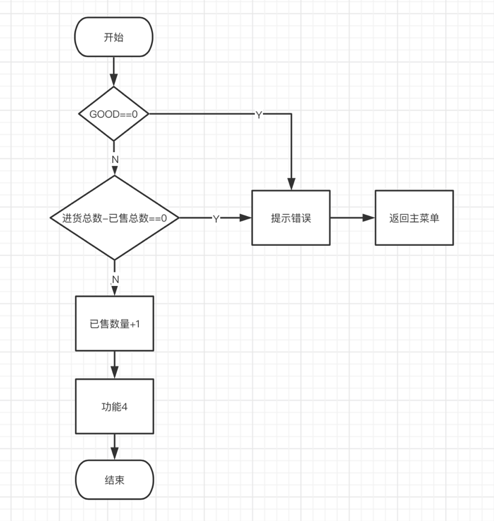


图1.19功能三

1. 功能四，如图1.20所示

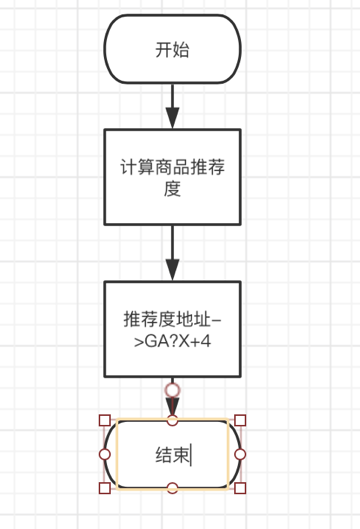


图1.20功能四

### 源程序

.386

STACK SEGMENT USE16 STACK

DB 500 DUP(0)

STACK ENDS

DATA SEGMENT USE16

BNAME DB 'HUANG JUNQI',0

BPASS DB '12345',0

AUTH DB 0

GOOD DW 0,'$'

N EQU 30

SNAME DB 'SHOP',0,'$'

GA1 DB 'PEN',7 DUP(0),10

DW 35, 56 ,70 ,25 ,?

GA2 DB 'BOOK',6 DUP(0),9

DW 12,30,25,25,?

IN\_NAME DB 20,20 DUP(0)

IN\_PWD DB 10,10 DUP(0)

OPTION1 DB 2 DUP(0),'$'

VIEWER\_HINT1 DB 'HUANG JUNQI','$'

VIEWER\_HINT2 DB 'CUSTOMER','$'

GOOD\_HINT1 DB 'PEN','$'

GOOD\_HINT2 DB 'BOOK','$'

MENU\_HINT1 DB '1.SIGN IN','$'

MENU\_HINT2 DB '2.SERCH GOODS','$'

MENU\_HINT3 DB '3.ORDER','$'

MENU\_HINT4 DB '4.CAULATE','$'

MENU\_HINT5 DB '5.SORT','$'

MENU\_HINT6 DB '6.REVISE','$'

MENU\_HINT7 DB '7.CHANGE EXPERIMENT ','$'

MENU\_HINT8 DB '8.SHOW','$'

MENU\_HINT9 DB '9.QUIT','$'

NUMBER\_HINT DB 'PLEASE CHOOSE A NUMBER TO FUNCTION','$'

USERNAME\_HINT DB 'PLEASE ENTER YOUR USERNAME', '$'

PASSWORD\_HINT DB 'PLEASE ENTER YOUR PASSWORD', '$'

SUCCESS\_HINT DB 'AUTH SUCCESS', '$'

QUERY\_HINT DB 'PLEASE ENTER THE ITEM NAME YOU WANT TO QUERY','$'

FAIL\_HINT DB '404','$'

IN\_ITEM DB 80,0,80 DUP(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

VIEW\_ID:CALL NEWLINE

MOV AL,AUTH

CMP AL,0

JZ N1

JMP N2

N1:CALL NEWLINE

LEA DX,VIEWER\_HINT2

MOV AH,9

INT 21H

JMP GOOD\_ID

N2:CALL NEWLINE

LEA DX,VIEWER\_HINT1

MOV AH,9

INT 21H

JMP GOOD\_ID

GOOD\_ID:

CALL NEWLINE

MOV AX,GOOD

CMP AX,0

JZ MENU

CMP AX,1

JZ GO1

CMP AX,2

JZ GO2

GO1:CALL NEWLINE

LEA DX,GOOD\_HINT1

MOV AH,9

INT 21H

JMP MENU

GO2:CALL NEWLINE

LEA DX,GOOD\_HINT2

MOV AH,9

INT 21H

JMP MENU

MENU:CALL NEWLINE

LEA DX,number\_hint

MOV AH,9

INT 21H

call NEWLINE

LEA DX,menu\_hint1

MOV AH,9

INT 21H

call NEWLINE

LEA DX,menu\_hint2

MOV AH,9

INT 21H

call NEWLINE

LEA DX,menu\_hint3

MOV AH,9

INT 21H

call NEWLINE

LEA DX,menu\_hint4

MOV AH,9

INT 21H

call NEWLINE

LEA DX,menu\_hint5

MOV AH,9

INT 21H

call NEWLINE

LEA DX,menu\_hint6

MOV AH,9

INT 21H

call NEWLINE

LEA DX,menu\_hint7

MOV AH,9

INT 21H

call NEWLINE

LEA DX,menu\_hint8

MOV AH,9

INT 21H

call NEWLINE

LEA DX,menu\_hint9

MOV AH,9

INT 21H

call NEWLINE

mov ah,1

int 21h

cmp al,'1'

jz FUNC1

cmp al,'2'

jz func2

cmp al,'3'

jz func3

cmp al,'4'

jz func4

cmp al,'8'

jz func8

cmp al,'9'

jz func9

jmp AUTH\_FAIL

FUNC1: call NEWLINE

LEA DX,SNAME

MOV AH,9

int 21H

CALL NEWLINE

LEA DX,username\_hint

MOV AH,9

int 21H

CALL NEWLINE

LEA DX,in\_name

MOV AH,10

INT 21H

LEA DX,password\_hint

MOV AH,9

INT 21H

CALL NEWLINE

LEA DX,in\_pwd

MOV AH,10

INT 21H

CALL NEWLINE

MOV SI,offset in\_name

CMP BYTE PTR 2[SI],13

JZ VIEW\_ID

JMP CHECK\_USERNAME

AUTH\_FAIL:CALL NEWLINE

MOV AUTH,0

LEA DX,fail\_hint

MOV AH,9

INT 21H

CALL NEWLINE

JMP VIEW\_ID

AUTH\_SUCCESS:CALL NEWLINE

MOV AUTH,1

LEA DX,success\_hint

MOV AH,9

INT 21H

JMP VIEW\_ID

CHECK\_USERNAME:

MOV SI,offset in\_name

MOV DI,offset BNAME

MOV BX,0

JMP LOOPA

LOOPA:

MOV AH,BYTE PTR 2[BX+SI]

MOV AL,BYTE PTR [BX+DI]

CMP AL,0

JZ CHECK\_PASSWORD

CMP AH,AL

JNZ AUTH\_FAIL

INC BX

JMP LOOPA

CHECK\_PASSWORD:

MOV SI,offset in\_pwd

MOV DI,offset BPASS

MOV BX,0

JMP LOOPB

LOOPB:

MOV BYTE PTR AH,2[BX+SI]

MOV BYTE PTR AL,[BX+DI]

CMP AL,0

JZ AUTH\_SUCCESS

CMP AH,AL

JNZ AUTH\_FAIL

INC BX

JMP LOOPB

FUNC2: call NEWLINE

LEA DX, query\_hint

MOV AH,9

INT 21H

CALL NEWLINE

LEA DX,in\_item

MOV AH,10

INT 21H

JMP SERCH1

SERCH1:

MOV SI,offset in\_item

MOV DI,offset GA1

MOV BX,0

JMP LOOPC

LOOPC: MOV AH,BYTE PTR 2[BX+SI]

MOV AL,BYTE PTR [BX+DI]

CMP AH,AL

JNZ SERCH2

INC BX

MOV AH,BYTE PTR 2[BX+SI]

CMP AH,'#'

JNZ LOOPC

MOV GOOD,1

CALL NEWLINE

LEA DX,success\_hint

MOV AH,9

INT 21H

JMP VIEW\_ID

SERCH2:

MOV SI,offset in\_item

MOV DI,offset GA2

MOV BX,0

JMP x

x: MOV AH,BYTE PTR 2[BX+SI]

MOV AL,BYTE PTR [BX+DI]

CMP AH,AL

JNZ AUTH\_FAIL

INC BX

MOV AH,BYTE PTR 2[BX+SI]

CMP AH,'#'

JNZ x

MOV GOOD,2

LEA DX,success\_hint

MOV AH,9

INT 21H

JMP VIEW\_ID

FUNC3:

MOV AX,GOOD

CMP AX,0

JZ AUTH\_FAIL

CMP AX,1

JZ G1

CMP AX,2

JZ G2

G1:

MOV BX,OFFSET GA1

JMP CALCULATE

G2:

MOV BX,OFFSET GA2

JMP CALCULATE

CALCULATE:

ADD BX,10

MOV AH,5[BX]

MOV AL,7[BX]

SUB AH,AL

CMP AH,0

JZ AUTH\_FAIL

MOV AX,1

add 7[bx],AX

MOV AX, WORD PTR 3[BX]

MOV DL, BYTE PTR [BX]

MOV DH, 0

MUL DX

MOV CX, 10

DIV CX

mov cx,ax

MOV AX, 1[BX]

MOV DX, 128

MUL DX

MOV DX, 0

DIV CX

MOV CX, AX

MOV AX, WORD PTR 7[BX]

MOV DX, 64

MUL DX

MOV DX, 0

DIV WORD PTR 5[BX]

ADD AX, CX

MOV WORD PTR 9[BX],AX

CALL NEWLINE

mov dl,byte ptr 9[bx]

MOV AH,2

INT 21H

call newline

JMP VIEW\_ID

FUNC4: MOV BX,offset GA1

ADD BX,10

MOV AX, 3[BX]

MOV DL, BYTE PTR [BX]

MOV DH, 0

MUL DX

MOV CX, 10

DIV CX

MOV CX, AX

MOV AX, 1[BX]

MOV DX, 128

MUL DX

MOV DX, 0

DIV CX

MOV CX, AX

MOV AX, WORD PTR 7[BX]

MOV DX, 64

MUL DX

MOV DX, 0

DIV WORD PTR 5[BX]

ADD AX, CX

MOV 9[BX],AX

CALL NEWLINE

mov dl,byte ptr 9[bx]

MOV AH,2

INT 21H

call newline

MOV BX,offset GA2

ADD BX,10

MOV AX, 3[BX]

MOV DL, BYTE PTR [BX]

MOV DH, 0

MUL DX

MOV CX, 10

DIV CX

MOV CX, AX

MOV AX, 1[BX]

MOV DX, 128

MUL DX

MOV DX, 0

DIV CX

MOV CX, AX

MOV AX, WORD PTR 7[BX]

MOV DX, 64

MUL DX

MOV DX, 0

DIV WORD PTR 5[BX]

ADD AX, CX

MOV 9[BX],AX

CALL NEWLINE

mov dl,byte ptr 9[bx]

MOV AH,2

INT 21H

call newline

JMP VIEW\_ID

FUNC8: call NEWLINE

MOV AX,CS

MOV SI,OFFSET TEMP+3

MOV CX,4

LOOP2:

MOV DH,AL

SHR AX,4

AND DH,0FH

ADD DH,30H

CMP DH,':'

JA NUM1

JB NUM2

NUM1:

ADD DH,7H

MOV [SI],DH

DEC SI

LOOP LOOP2

NUM2:

MOV [SI],DH

DEC SI

LOOP LOOP2

OPUT: MOV DX,OFFSET TEMP

MOV AH,9

INT 21H

call NEWLINE

jmp VIEW\_ID

FUNC9:

MOV AH,4CH

INT 21H

NEWLINE proc

mov dl, 13

mov ah, 2h

int 21h

mov dl, 10

mov ah, 2h

int 21h

ret

NEWLINE endp

CODE ENDS

END START

### 实验步骤

1、编写代码，实现四个有关功能，保存为.asm文件。

2、使用DOSBOX编译生成.exe文件。

3、在TD中运行，使用F8和GOTO通过设置断点和单步调试来观察程序是否正确运行。

### 实验记录与分析

1. 主菜单测试如图1.21a，b，c所示，由于窗口大小，后续不再显示当前浏览者及浏览商品，第一行显示用户，第二行显示商品，接着是菜单。

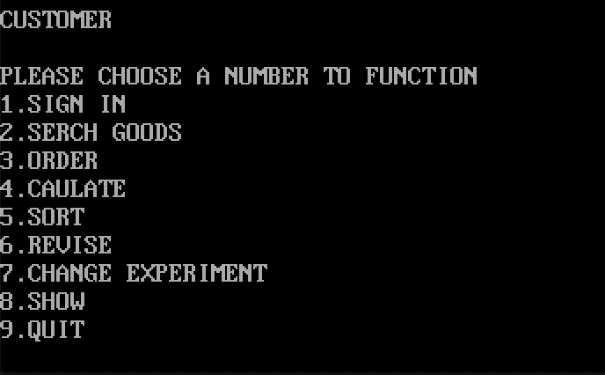


图1.21a主菜单测试(customer)

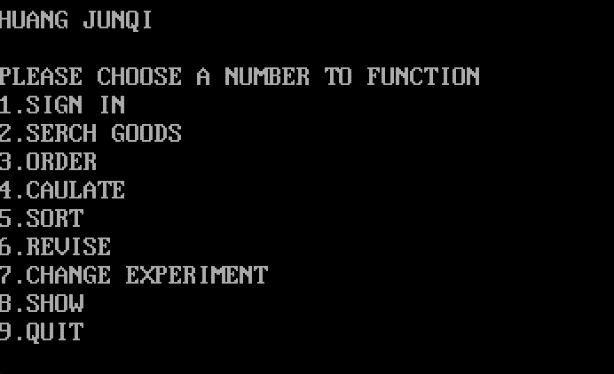


图1.21b主菜单测试（user）

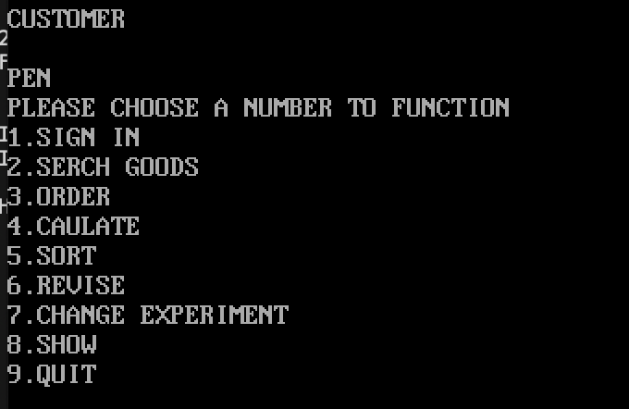


图1.21c主菜单测试（显示当前商品）

1. 功能一测试如图1.22，1.23，1.24，1.25，1.26所示

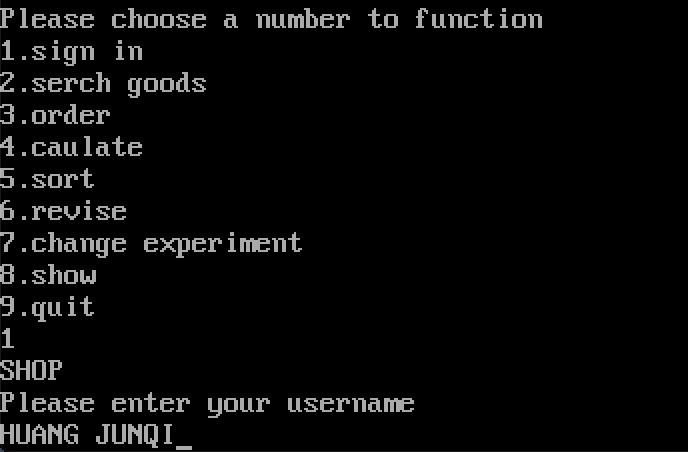


图1.22用户名输入

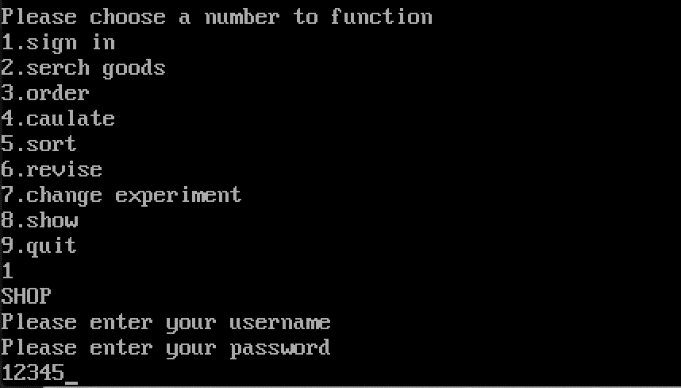


图1.23密码输入

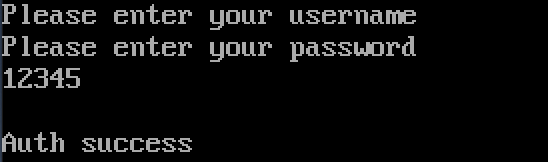


图1.24 登陆成功

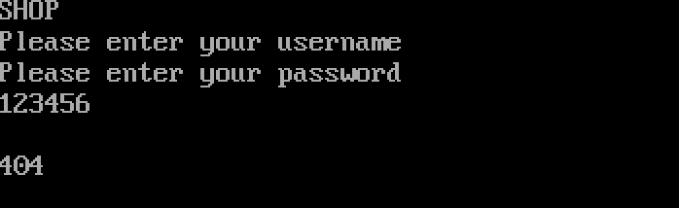


图1.25输入错误

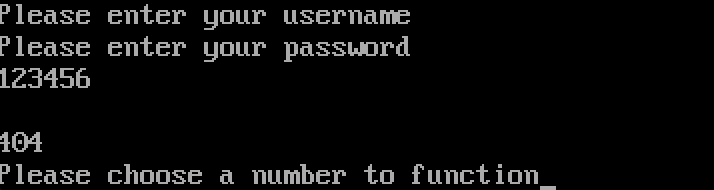


图1.26登陆失败

1. 查找商品如图1.27，1.28所示，输入商品名以“#”结束，因为无法控制循环次数，如果只选定前三次循环对比，那么pen和pencil都无法区分，因此在商品名结束时以“#”结尾。

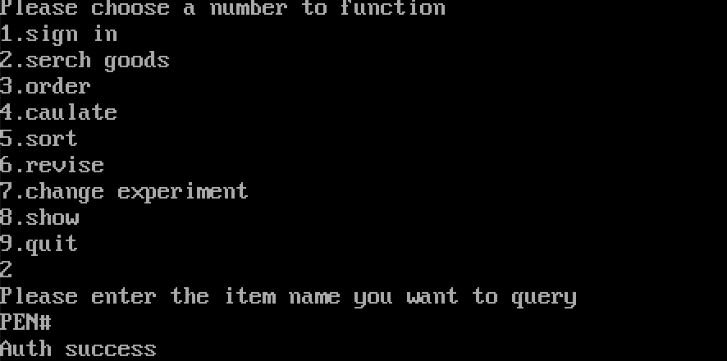


图1.27查找商品成功

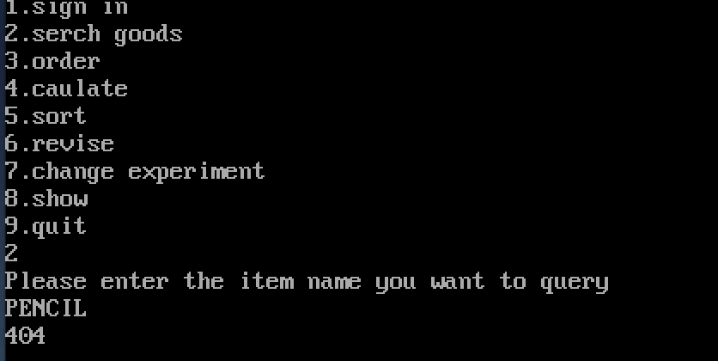


图1.28商品不存在

1. 功能三测试，如图1.29所示，g的十进制为103

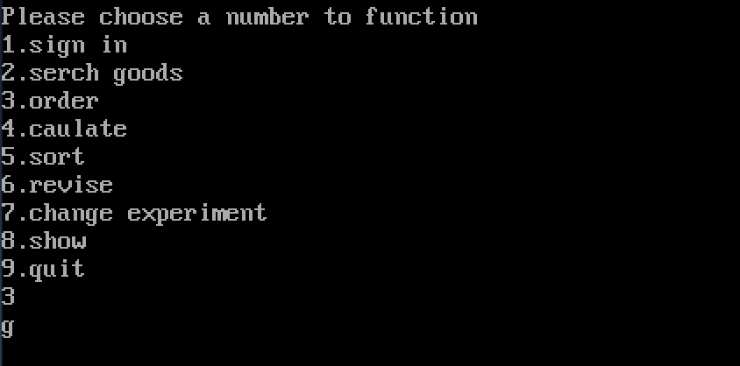


图1.29已售数量加一后的推荐度

商品无效如图1.30所示

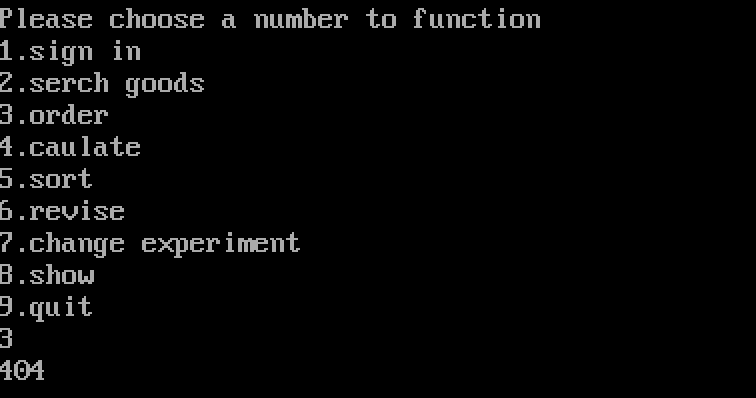


图1.30 商品无效

1. 功能四测试，如图1.31所示，f的十进制为102，x的十进制为120

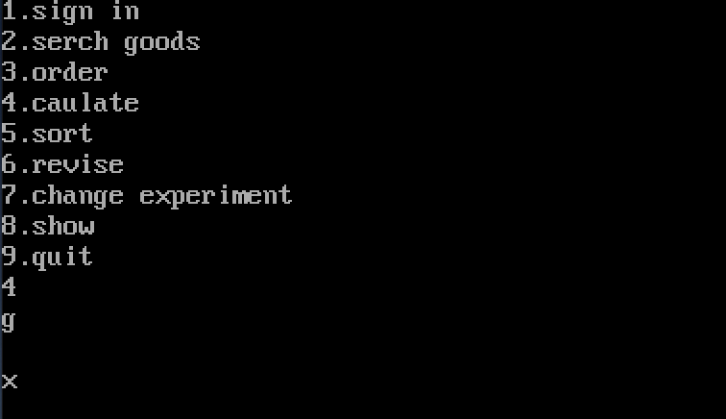


图1.31功能四测试

1. 功能八测试，如图1.32所示，cs首地址为01DB

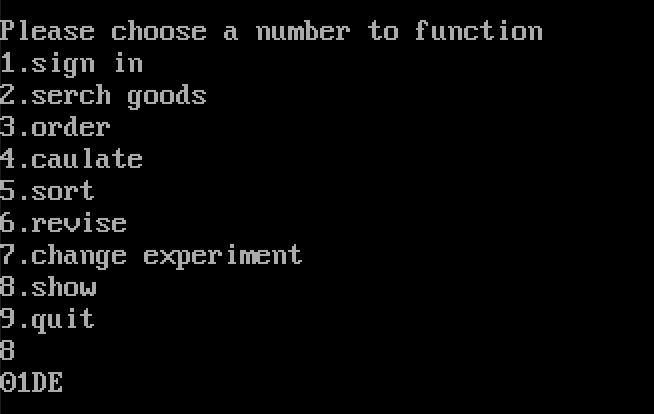


图1.32功能八测试

1. 功能9测试，如图1.33所示

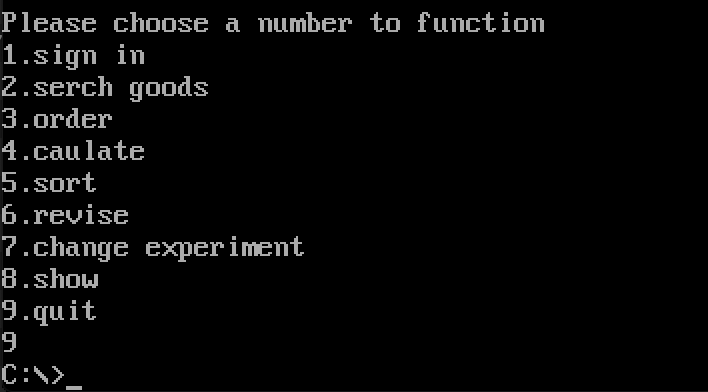


图1.33 功能就测试

## 总结

1. 在代码区输入指令：点击要输入的那一行，从键盘上直接输入即可，不需要右键使用goto。
2. 若鼠标无法使用，常用一些快捷键：

tab——切换段

alt+f10——鼠标右键

alt+f——file

alt+x——退出td

1. 在任务3中可以看到，采用变址寻址会比直接寻址代码量更小，使用的寄存器更少。
2. 变量分配缓冲区时的格式如

Buf db 80，0，80 dup（0）

因此向缓冲区输入内容是从第三个字节开始。

（5）9号功能调用时，若字符串结尾没有$符号，则会将后面内存里的所有内容输出。

（6）若编写子程序，子程序结尾应有：子程序名 endp

# 程序优化

## 实验目的

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。

**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。

**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

### 设计思想与存储单元分配

存储单元分配与任务1.4基本相同，额外添加了两个存储单元：

1、GA3 DB 'BAG',7 DUP(0),9

DW 10,20,20000,0,?

2、m EQU 1000

设计思想：在任务1.4基础上加入循环。在循环之前加入计时代码，循环结束后结束计时，将时间显示在屏幕上，返回主菜单。

### 流程图

流程图如图2.1所示

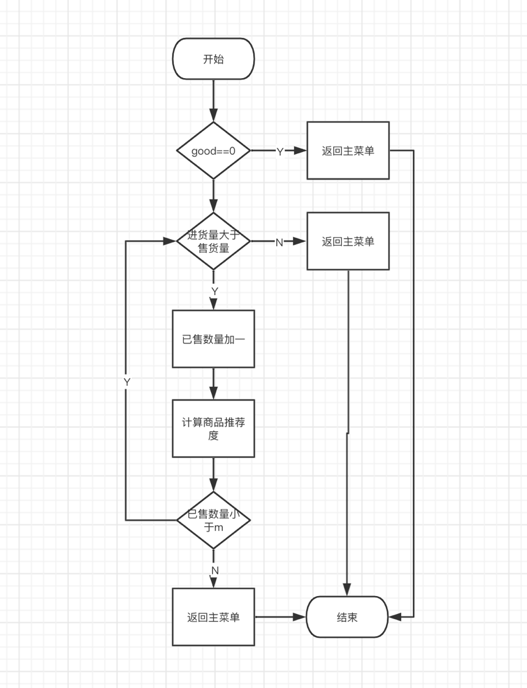


图2.1任务2.1流程图

### 源程序

功能3代码

FUNC3:

MOV AX,GOOD

CMP AX,0

JZ AUTH\_FAIL

mov ax,0

call TIMER；开始计时

mov si,offset GAN

mov bx,0

mov cx,27

jmp nc

nc:

mov ax,124

mov word ptr 19[si+bx],ax

add bx,20

dec cx

cmp cx,0

jnz nc

jmp CALCULATE

CALCULATE:

mov di,offset GA1

ADD di,10

MOV ah,5[di]

MOV al,7[di]

SUB ah,al

CMP ah,0

JZ AUTH\_FAIL

MOV AX, WORD PTR 3[di]

MOV DL, BYTE PTR [di]

MOV DH, 0

MUL DX

MOV CX, 10

DIV CX

mov cx,ax

MOV AX, 1[di]

MOV DX, 128

MUL DX

MOV DX, 0

DIV CX

MOV CX, AX

MOV AX, WORD PTR 7[di]

MOV DX, 64

MUL DX

MOV DX, 0

DIV WORD PTR 5[di]

ADD AX, CX

MOV WORD PTR 9[di],AX

mov di,offset GA2

ADD di,10

MOV ah,5[di]

MOV al,7[di]

SUB ah,al

CMP ah,0

JZ AUTH\_FAIL

MOV AX, WORD PTR 3[di]

MOV DL, BYTE PTR [di]

MOV DH, 0

MUL DX

MOV CX, 10

DIV CX

mov cx,ax

MOV AX, 1[di]

MOV DX, 128

MUL DX

MOV DX, 0

DIV CX

MOV CX, AX

MOV AX, WORD PTR 7[di]

MOV DX, 64

MUL DX

MOV DX, 0

DIV WORD PTR 5[di]

ADD AX, CX

MOV WORD PTR 9[di],AX

mov bx,offset GA3

ADD BX,10

MOV cx,5[BX]

MOV dx,7[BX]

SUB cx,dx

CMP cx,0

JZ AUTH\_FAIL

mov ax,1

add 7[bx],ax

MOV AX, WORD PTR 3[BX]

MOV DL, BYTE PTR [BX]

MOV DH, 0

MUL DX

MOV CX, 10

DIV CX

mov cx,ax

MOV AX, 1[BX]

MOV DX, 128

MUL DX

MOV DX, 0

DIV CX

MOV CX, AX

MOV AX, WORD PTR 7[BX]

MOV DX, 64

MUL DX

MOV DX, 0

DIV WORD PTR 5[BX]

ADD AX, CX

MOV WORD PTR 9[BX],AX

cmp word ptr 7[bx],m

jl CALCULATE

call NEWLINE

mov ax,1

call TIMER；计时结束

JMP VIEW\_ID

### 实验步骤

（1）输入BAG#找到该商品

（2）使用功能三计时

（3）将所用时间显示在屏幕上

### 实验记录与分析

1. CPU Speed：192 circles
2. 执行功能2后，可以在主菜单看到当前浏览商品，如图2.2所示

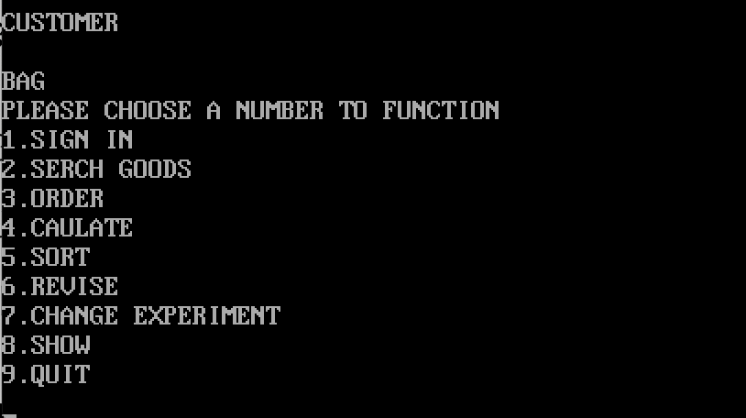


图2.2主菜单

1. m=1000，如图2.3所示

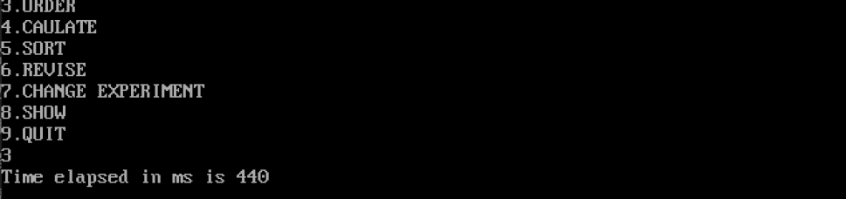


图2.3m=1000执行时间

1. m=1020，如图2.4所示

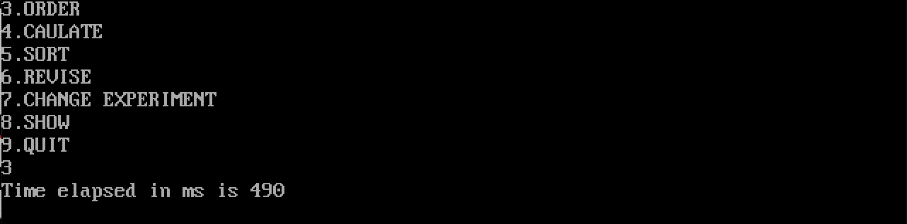


图2.4 m=1020执行时间

1. 通过变更cpu speed 来模拟不同硬件测试，cpu speed：308，579
2. 这里用m=1000来测试，如图2.5a，b所示

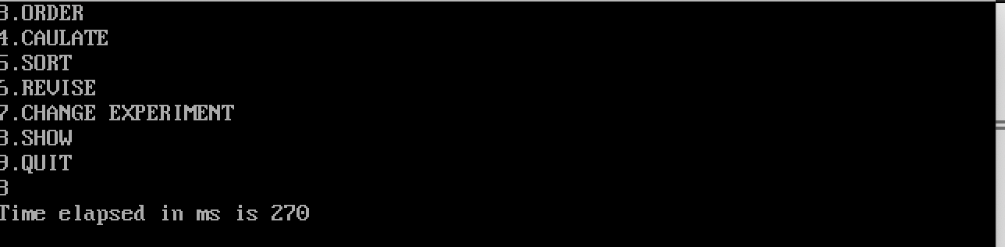


图2.5a cpu308 ，m=1000执行时间

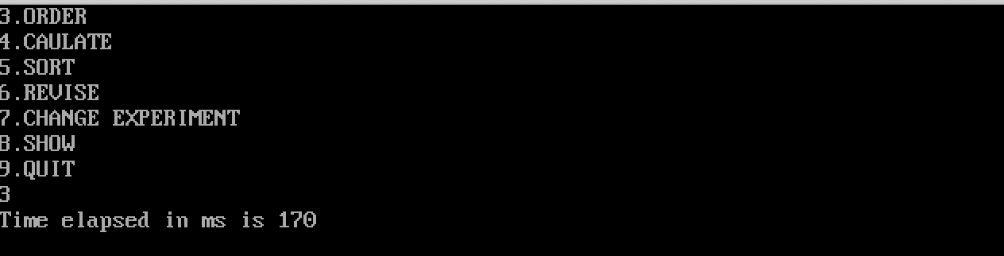


图2.5b cpu579，m=1000执行时间

1. 通过分析，m=1000执行时间440ms，m=1020执行时间490ms，恰好2倍。而提高cpu speed在同样测试条件下，执行时间会降低。

## 任务2.2实验过程

### 实验方法说明

（1）任务2.1中的每次循环都重新计算了除bag以外的所有商品，而其他商品的已售数量不会变化，也就是推荐度不会变化，因此，可以将计算其他商品的指令移到循环外，只计算一次别的商品的推荐度，从而增加效率。

（2）将乘法指令改为对应的移位指令，从而提高效率。

部分代码比较：

优化前：mov dx，128

Mul dx

Mov dx，0

优化后：sal ax，7

Mov dx，0

### 实验记录与分析

1. CPU Speed：192 circles
2. 执行功能2后，可以在主菜单看到当前浏览商品，如图2.6所示

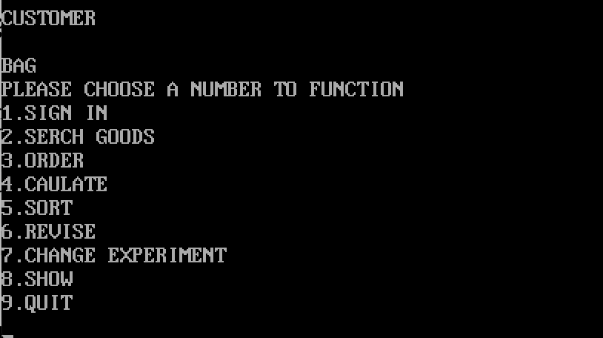


图2.6主菜单

1. m=1000，采用2.4.1（1）如图2.7所示

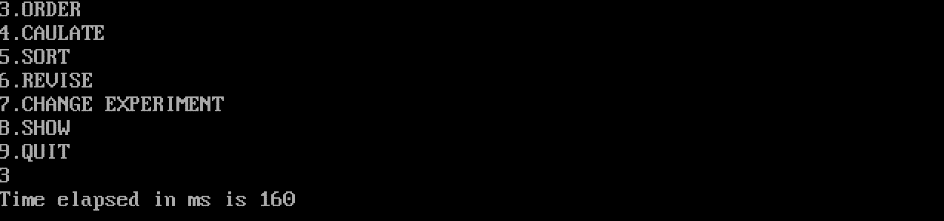


图2.7 采用2.4.1（1）优化后m=1000执行时间

1. m=1020，采用2.4.1（1）如图2.8所示

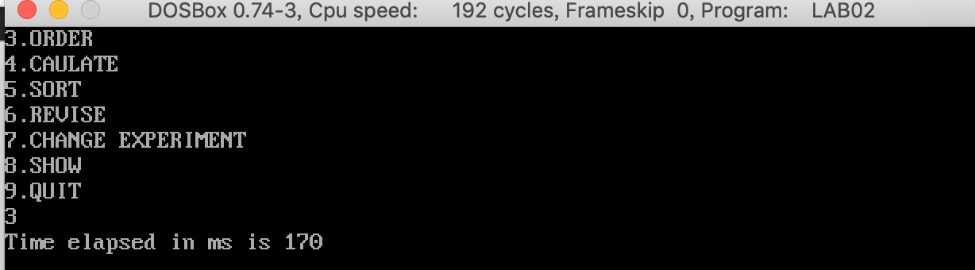


图2.8 采用2.4.1（1）优化后m=1020执行时间

1. 通过计算，采用2.4.1（1）优化后，m=1000效率提升63.64%，m=1020效率提升65.31%，满足任务要求。
2. m=1000，采用2.4.1（2），如图2.9所示

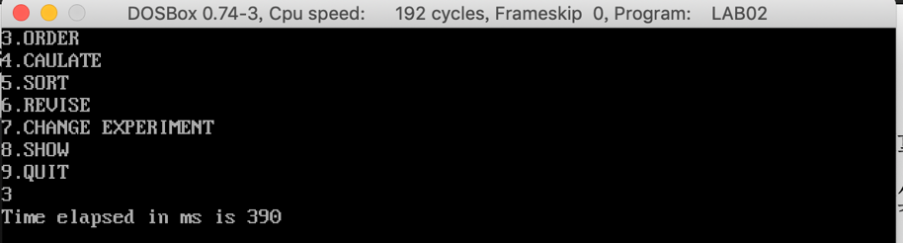


图2.9 采用2.4.1（2）优化后m=1000执行时间

1. m=1020，采用2.4.1（2），如图2.10所示

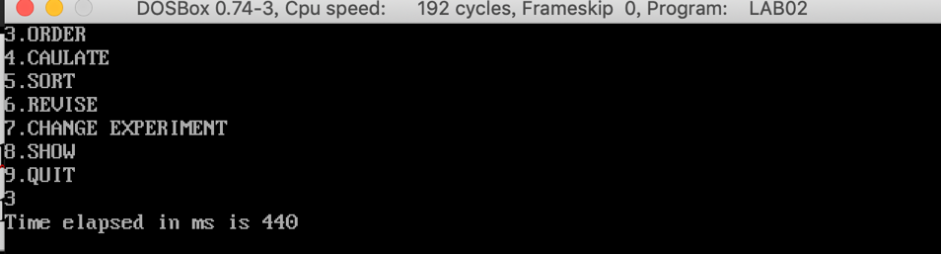


图2.10 采用2.4.1（2）优化后m=1020执行时间

（8）通过计算，采用2.4.1（2）优化后，m=1000效率提高11.36%，m=1020提高10.20%。同样满足任务要求，但与用2.4.1（1）的优化方法相比，差距甚远。

## 总结

1. m=1000，如果在循环内增加一条指令，对程序而言就增加了1000条，大大降低了效率，我优化时减少了两千次重复计算没有已售数量变化的商品，效率翻了一倍不止，可见降低循环内的指令可以提高效率。
2. 优化可以从提高单个指令的效率，也可以减少循环内指令的个数。但在本实验中，可以看到两种优化方法的区别，因此在本实验中，采用减少循环内指令的2.4.1（1）方法比提高单个指令效率的2.4.1（2）方法要好很多。
3. 对于乘除法，尤其是与2的n次幂的常数的乘法可以替换成移位指令。
4. 在td里面可以看到，如果调用子程序，可以使源程序更加简短，节省存储空间，可读性也更高。

# 模块化程序设计

## 实验目的

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** （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）。

任务3.2：在C语言程序中调用汇编语言实现的函数

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

### 设计思想与存储单元分配

1. 主模块与前面任务相同
2. 子模块增加了如下数据：

SUCCESS\_HINT1 DB 'AUTH SUCCESS', '$'

FAIL\_HINT1 DB 'ACCESS DENIED','$'

INPUT\_HINT1 DB 'PLEASE INPUT DISCOUNT','$'

INPUT\_HINT2 DB 'PLEASE INPUT PURCHASING COST','$'

INPUT\_HINT3 DB 'PLEASE INPUT SELLING PRICE','$'

INPUT\_HINT4 DB 'PLEASE INPUT PURCHASING NUMBER','$'

GA11 DB 'PEN',6 DUP(0),'$',10

DW 35, 56 ,70,25 ,?

GA21 DB 'BOOK',5 DUP(0),'$',9

DW 12,30,25,5,?

GA31 DB 'BAG',6 DUP(0),'$',9

DW 10,20,1000,0,?

BUF1 DB 10,0,10 DUP(0)

BUF2 DB 10,0,10 DUP(0)

BUF3 DB 10,0,10 DUP(0)

BUF4 DB 10,0,10 DUP(0)

NUM DW ?

BUF1,BUF2,BUF3,BUF4用来存放新的折扣，进货价，销售价，进货总数。 NUM用来存放每次字符转换成数字的数值。

（3）这次采用模块化的设计，将func6单独作为一个子模块，通过装配实现本次实验功能。

### 模块结构

1. 主模块的引入引出：

PUBLIC NEWLINE

PUBLIC ADDRESS\_TEMP

EXTRN FUNC6:FAR

主模块给func6提供了输出换行的子程序以及存放商品地址的变量，并引入子模块的func6。

1. 子模块的引入引出：

PUBLIC FUNC6

EXTRN NEWLINE:FAR

EXTRN ADDRESS\_TEMP:WORD

子模块给主模块提供了func6的子程序，引入主模块的输出换行的子程序以及存放商品地址的变量。

### 源程序

功能6的源代码如下：

FUNC6 PROC FAR

MOV DI,ADDRESS\_TEMP

DISCOUNT1:

LEA DX,INPUT\_HINT1

MOV AH,9

INT 21H

CALL NEWLINE

LEA DX,BUF1

MOV AH,10

INT 21H

CALL NEWLINE

STOI:

MOV DX,0

MOV BX,10

MOV SI,2

MOV NUM,0

MOV AX,0

LOP:

MOV AL,BUF1[SI]

CMP AL,0DH

JE COST2

SUB AL,30H

CMP AL,0

JL DISCOUNT1

CMP AL,9

JG DISCOUNT1

CMP NUM,0

JE DO\_DEAL

PUSH AX

MOV AX,NUM

MUL BX

MOV NUM,AX

POP AX

DO\_DEAL:

ADD NUM,AX

MOV AX,0

INC SI

JMP LOP

COST2:

MOV AX,NUM

MOV 10[DI],AX

LEA DX,INPUT\_HINT2

MOV AH,9

INT 21H

CALL NEWLINE

LEA DX,BUF2

MOV AH,10

INT 21H

CALL NEWLINE

STOI2:

MOV DX,0

MOV BX,10

MOV SI,2

MOV NUM,0

MOV AX,0

LOP22:

MOV AL,BUF2[SI]

CMP AL,0DH

JZ PRICE3

SUB AL,30H

CMP AL,0

JL COST2

CMP AL,9

JG COST2

CMP NUM,0

JE DO\_DEAL2

PUSH AX

MOV AX,NUM

MUL BX

MOV NUM,AX

POP AX

DO\_DEAL2:

ADD NUM,AX

MOV AX,0

INC SI

JMP LOP22

PRICE3:

MOV AX,NUM

MOV 11[DI],WORD PTR AX

LEA DX,INPUT\_HINT3

MOV AH,9

INT 21H

CALL NEWLINE

LEA DX,BUF3

MOV AH,10

INT 21H

CALL NEWLINE

STOI3:

MOV DX,0

MOV BX,10

MOV SI,2

MOV NUM,0

MOV AX,0

LOP3:

MOV AL,BUF3[SI]

CMP AL,0DH

JZ NUMBER4

SUB AL,30H

CMP AL,0

JL PRICE3

CMP AL,9

JG PRICE3

CMP NUM,0

JE DO\_DEAL3

PUSH AX

MOV AX,NUM

MUL BX

MOV NUM,AX

POP AX

DO\_DEAL3:

ADD NUM,AX

MOV AX,0

INC SI

JMP LOP3

NUMBER4:

MOV AX,NUM

MOV 13[DI],WORD PTR AX

LEA DX,INPUT\_HINT4

MOV AH,9

INT 21H

CALL NEWLINE

LEA DX,BUF4

MOV AH,10

INT 21H

CALL NEWLINE

STOI4:

MOV DX,0

MOV BX,10

MOV SI,2

MOV NUM,0

MOV AX,0

LOP4:

MOV AL,BUF4[SI]

CMP AL,0DH

JZ FINAL

SUB AL,30H

CMP AL,0

JL NUMBER4

CMP AL,9

JG NUMBER4

CMP NUM,0

JE DO\_DEAL4

PUSH AX

MOV AX,NUM

MUL BX

MOV NUM,AX

POP AX

DO\_DEAL4:

ADD NUM,AX

MOV AX,0

INC SI

JMP LOP4

FINAL:

MOV AX,NUM

MOV 15[DI],WORD PTR AX

RET

FUNC6 ENDP

### 实验步骤

1. 编译lab03.asm与func6.asm
2. Link lab03.obj func6.obj
3. 运行Lab03.exe

### 实验记录与分析

1. 登陆成功，开始功能测试如图3.1所示

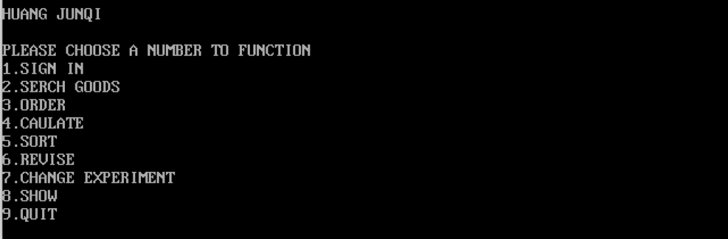


图3.1登陆成功界面

1. 功能二正常测试，以pen为例，如图3.2所示

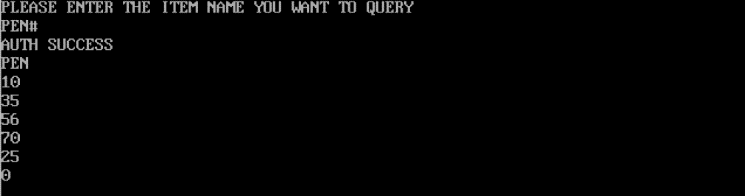


图3.2输出pen的商品信息

1. 功能六正常测试，如图3.3a，b，c所示

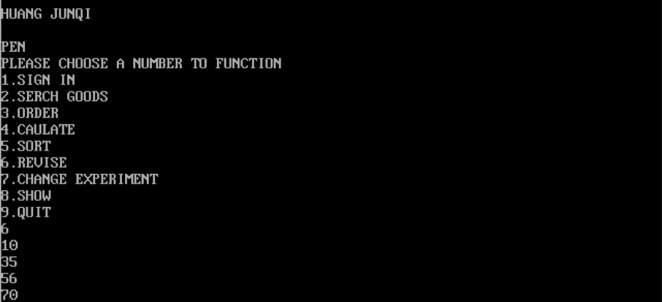


图3.3a正常输出原数值

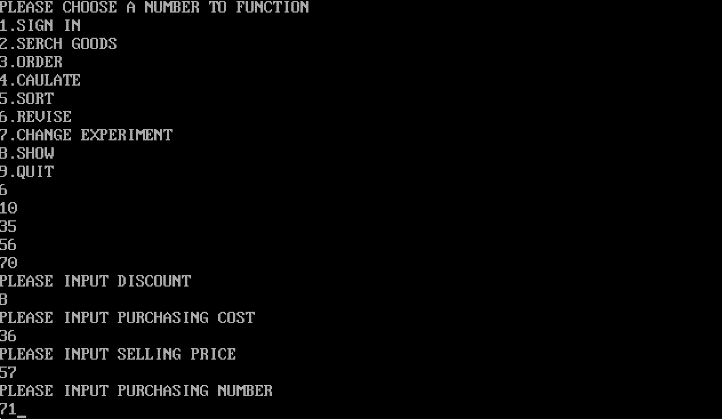


图3.3b输入要修改的数值

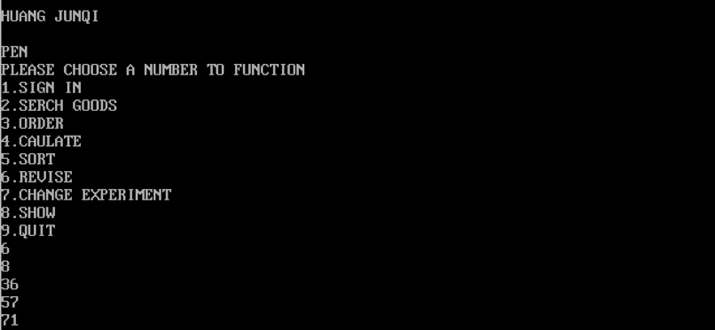


图3.3c修改成功后再执行功能六

1. 功能六异常测试，若不是老板或商品不存在，如图3.4所示

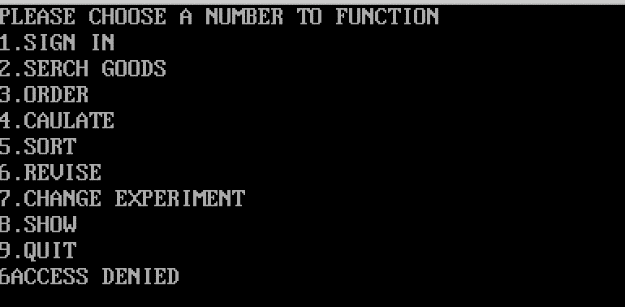


图3.4未登陆或商品不存在

1. 功能六异常测试，输入非法字符，如图3.5所示



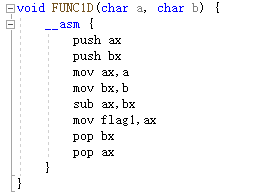
图3.5输入非法字符

1. EXTRN,PUBLIC语句要放在.386前面，否则程序会读取非法地址 。
2. 若只有RET指令，只会返回原点而不清除堆栈段中的地址。
3. TD中执行trace即可进入到子程序中观察。
4. 宏是在执行过程中进行复制相应的指令，而子程序则是编译过程中就完成了。

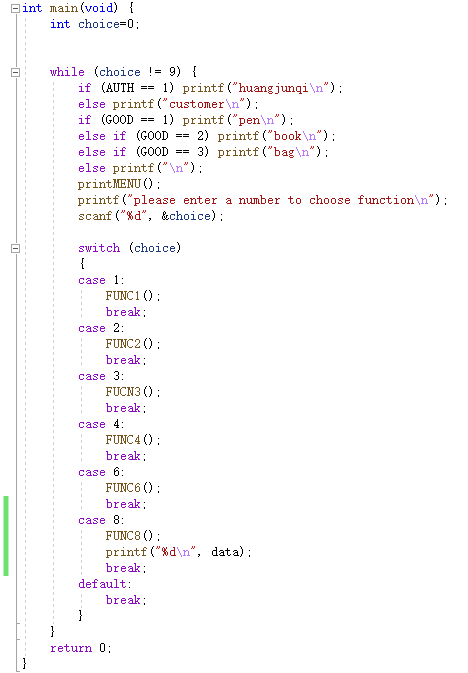
## 任务3.2实验过程

### 实验方法说明

1. 基于之前任务的功能，用C语言实现菜单功能。
2. 对功能一，二进行整合，将比较字符的功能进行统一，并用汇编语言实现。
3. 对功能三，四进行整合，计算推荐度的功能用汇编语言实现。
4. 采用内联汇编的方法，并用全局变量传递参数。
5. 在汇编程序中保护相应寄存器，使其不受破坏，影响程序功能的正确性。
6. 给出一个内联汇编的部分代码：



1. 给出菜单功能的代码：



### 实验记录与分析

1. 实验环境：DOSBox0.74-2；borland c++集成环境。
2. 编译后执行，如图3.6所示。

手机屏幕的截图

描述已自动生成

图3.6菜单功能测试

1. 测试功能一，如图3.7所示。

手机屏幕的截图

描述已自动生成

图3.7 功能一测试

1. 测试功能二，如图3.8所示。

手机屏幕的截图

描述已自动生成

图3.8 功能二测试

1. 测试功能三，如图3.9所示。

手机屏幕截图

描述已自动生成

图3.9功能三测试

1. 测试功能四，如图3.10所示。

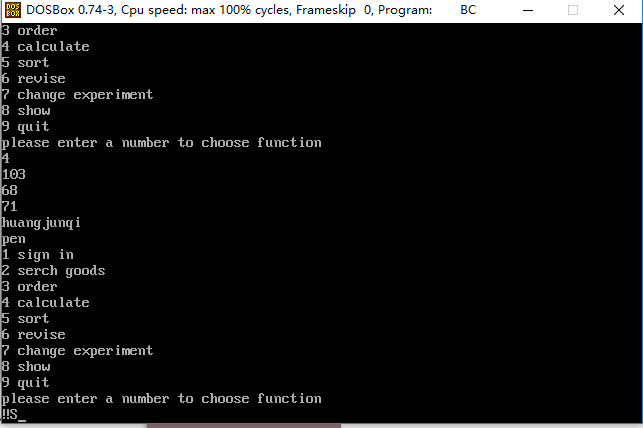


图3.10功能四测试

1. 测试功能六，如图3.11，3.12所示。

手机屏幕截图

描述已自动生成

图3.11执行功能六

手机屏幕截图

描述已自动生成

图3.12验证功能六

1. 执行功能八，如图3.13所示

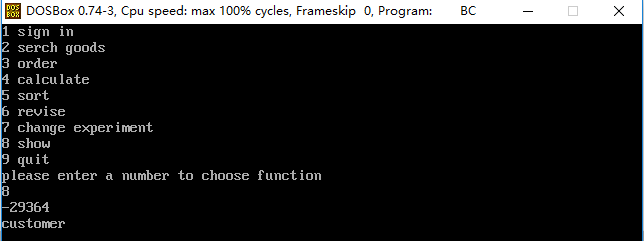


图3.13功能八测试

（9） 通过bcc可以看到，对于调用函数中的参数，是将其地址压栈，我们可以采用：

Push bp

Mov bp,sp

因为调用了函数，已经压栈了cs和ip，所以可以通过[bp+6]取到参数。

（10）“call near ptr \_printf”，这句就是printf在汇编语言的表示。

## 小结

1. 使用宏可以大大减少相同段落的代码，增强可读性。
2. 任务3.2可以说是整个汇编实验里最难的一个了，因为这个c和汇编混合编程的资料比较分散，而且使用的集成环境是几十年前的软件，现在查不到太多关于这个bc++3.1的资料，只能一个一个错误的去试。中间比较困难的就是怎么去传递一个参数，我选择了用全局变量，可以比较方便的在c程序和汇编程序中传递数据。此外，在实现多模块的编程时，代码段名，函数名，变量名前都要添加下划线，我利用bcc命令后查看c文件，里面的函数，变量都是下划线开头的。
3. 通过这次实验，我更好地理解了为什么c语言入门时要教char和int等变量类型占多少个字节，在混合编程中，这尤为重要，因为这会决定寄存器的使用。
4. 在c程序对应的汇编程序中我看到了大量保护寄存器的语句，说明如果使用寄存器时不保护相应的寄存器，很有可能就会影响c程序的正确性。

# 中断与反跟踪

## 实验目的

(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）。

**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。

**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。

## 任务4.1实验过程

### 设计思想与存储单元分配

在任务3.1的基础上，添加了下列数据：

UNINS DB 0

COUNT DB 18

HOUR DB ?,?,':','$'

MIN DB ?,?':'

SEC DB ?,?

INPUT\_SEC DB 20,0

OLD\_INT DW ?,?

TO\_BAK DB 1

通过接管8号中断完成任务。

### 流程图

流程图如图4.1所示

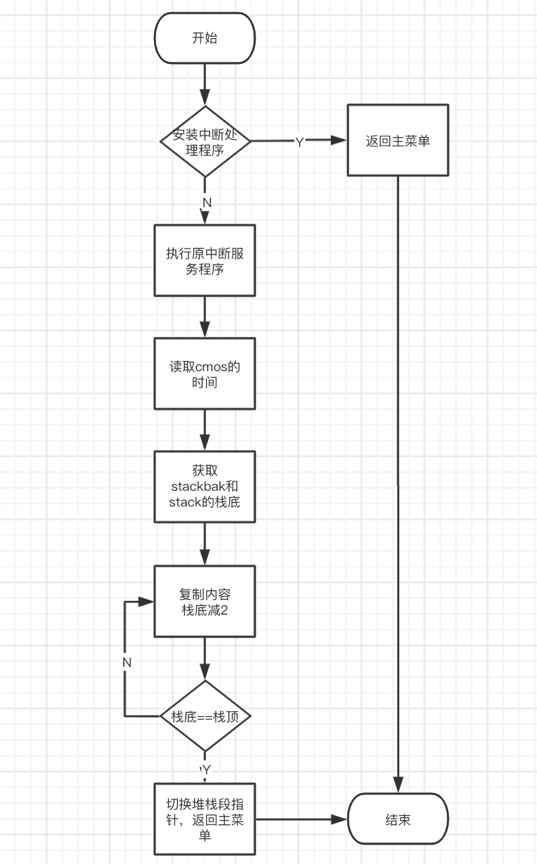


图4.1 任务4.1流程图

### 源程序

NEW08H PROC FAR

PUSHA

PUSH DS

PUSH ES

CALL GET\_TIME

PUSHF

CALL DWORD PTR OLD\_INT

DEC COUNT

JZ DISP

IRET

DISP: MOV COUNT,18

STI

MOV AX,WORD PTR SEC

MOV BX,WORD PTR INPUT\_SEC

JMP C\_STACK

POP ES

POP DS

POPA

IRET

C\_STACK:

MOV AH,TO\_BAK

MOV AL,1

CMP AH,AL

JZ C\_STACK1

JMP C\_STACK2

C\_STACK1:

MOV AL,0

MOV TO\_BAK,AL

MOV AX,STACKBAK

MOV ES,AX

MOV SI,OFFSET STACKBAK

MOV DI,OFFSET STACK

MOV AX,SP

LOOP81:

CMP DI,AX

JZ N\_FINAL1

MOV BX,SS:[DI]

MOV ES:[SI],BX

SUB DI,2

SUB SI,2

JMP LOOP81

N\_FINAL1:

MOV BX,SS:[DI]

MOV ES:[SI],BX

MOV AX,STACKBAK

MOV SS,AX

MOV SP,SI

POP ES

POP DS

POPA

;PUSHF

;CALL DWORD PTR OLD\_INT

;STI

IRET

C\_STACK2:

MOV AL,1

MOV TO\_BAK,AL

MOV AX,STACK

MOV ES,AX

MOV SI,OFFSET STACK

MOV DI,OFFSET STACKBAK

MOV AX,SP

LOOP82:

CMP DI,AX

JZ N\_FINAL2

MOV BX,SS:[DI]

MOV ES:[SI],BX

SUB DI,2

SUB SI,2

JMP LOOP82

N\_FINAL2:

MOV BX,SS:[DI]

MOV ES:[SI],BX

MOV AX,STACK

MOV SS,AX

MOV SP,SI

POP ES

POP DS

POPA

IRET

NEW08H ENDPGET\_TIME PROC

MOV AL,4

OUT 70H,AL

JMP $+2

IN AL,71H

MOV AH,AL

AND AL,0FH

SHR AH,4

ADD AX,3030H

XCHG AH,AL

MOV WORD PTR HOUR,AX

MOV AL,2

OUT 70H,AL

JMP $+2

IN AL,71H

MOV AH,AL

AND AL,0FH

SHR AH,4

ADD AX,3030H

XCHG AH,AL

MOV WORD PTR MIN,AX

MOV AL,0

OUT 70H,AL

JMP $+2

IN AL,71H

MOV AH,AL

AND AL,0FH

SHR AH,4

ADD AX,3030H

XCHG AH,AL

MOV WORD PTR SEC,AX

RET

GET\_TIME ENDP

### 实验步骤

1. 编译，链接，执行exe
2. 进入TD,观察堆栈段中内容是否成功转移
3. 使用功能八，观察段首址是否已经切换

### 实验记录与分析

1. 执行功能8，显示当前堆栈段，如图4.2所示

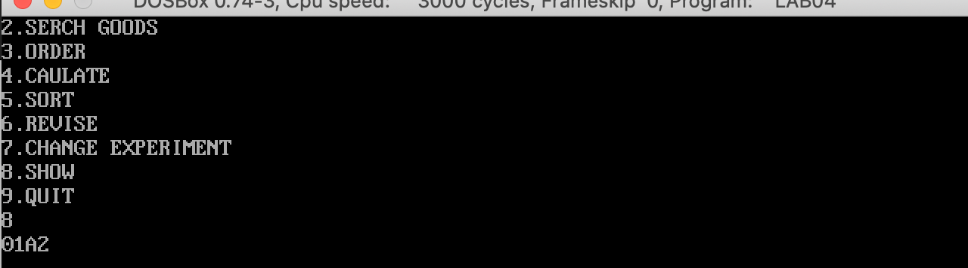


图4.2交换前堆栈段

1. 执行功能7，交换堆栈段，如图4.3所示

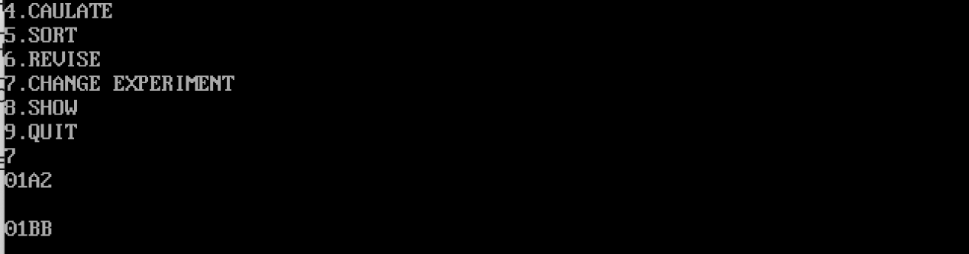


图4.3 交换堆栈段

1. 再执行一次功能8验证是否交换成功，如图4.4所示

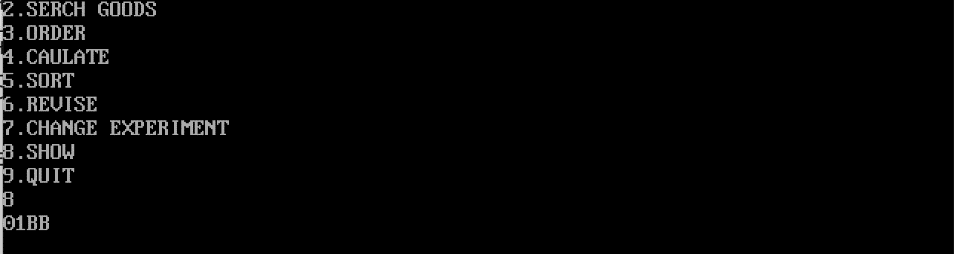


图4.4 交换后堆栈段

## 任务4.2实验过程

### 设计思想与存储单元分配

在任务4.1的基础上，作出了以下改变：

BPASS DB '1' XOR 'H','2' XOR 'H','3' XOR 'H','4' XOR 'H','5' XOR 'H',0

GA1 DB 'PEN',6 DUP(0),'$',10

DW 35 XOR 'H', 56 ,70,25 ,?

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

DW 12 XOR 'H',30,25,5,?

GA3 DB 'BAG',6 DUP(0),'$',9

DW 10 XOR 'H',20,1000,0,?

F1 DW FUNC1

F2 DW FUNC2

F3 DW FUNC3

F4 DW FUNC4

F6 DW FUNC6\_PRE

F7 DW FUNC7

F8 DW FUNC8

F9 DW FUNC9

采用间接转移和计时两种方法进行反跟踪，对数据加密是与‘H’进行异或。

### 流程图

流程图如图4.5所示

地图的截图

描述已自动生成

图4.5 4.4.2流程图

### 源程序

FUNC1D: MOV BX,F1

CALL BX

JMP VIEW\_ID

CHECK\_PASSWORD:

CLI;计时反跟踪开始

MOV AH,2CH

INT 21H

PUSH DX

MOV SI,OFFSET IN\_PWD

MOV DI,OFFSET BPASS

MOV BX,0

MOV AH,2CH;获取第二次秒

INT 21H

STI

CMP DX,[ESP]

POP DX

JZ LOOPB

JMP FUNC9

LOOPB:

MOV BYTE PTR AH,2[BX+SI]

MOV BYTE PTR AL,[BX+DI]

CMP AL,0

JZ AUTH\_SUCCESSD

XOR AH,'H'

CMP AH,AL

JNZ AUTH\_FAILD

INC BX

JMP LOOPB

### 实验步骤

1. 编译，链接执行exe
2. 输入密码进行验证
3. 进入TD观察数据段中存放的数据

### 实验记录与分析

1. 观察存放密码的内存，如图4.6所示，密码本应存放‘1，2，3，4，5’，但加密之后不会在内存中直接显示

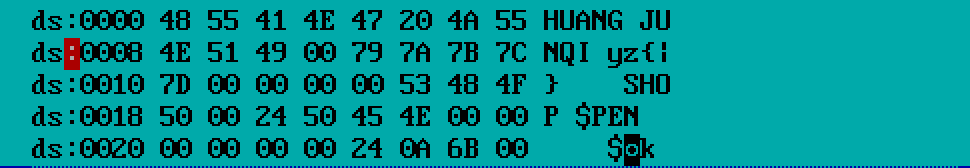


图4.6密码加密后的内存

1. 观察存放进货价的内存，同理，进货价也进行了加密，如图4.7所示



图4.7进货价加密

1. 观察间接转移指令，不会直接显示功能入口地址，如图4.8所示

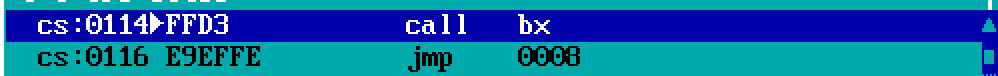


图4.8间接转移

1. 观察计时反跟踪，如图4.9a，b所示，如果是TD逐步执行会有时间差，因此会直接结束程序。

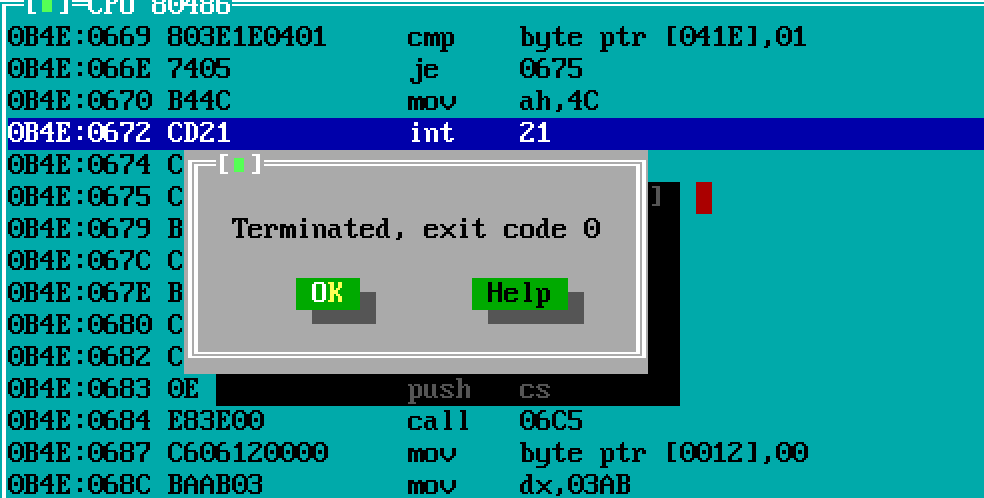


图4.9a TD中单步执行

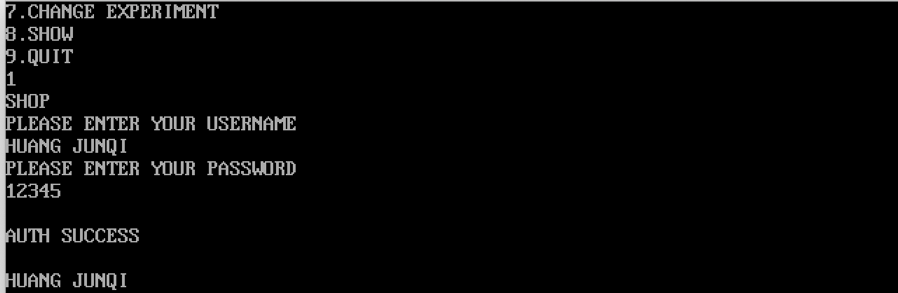


图4.9b 直接运行

## 任务4.3实验过程

### 实验方法说明

对同组同学的程序暴力破解，进入TD观察源程序加密方法，找到对应数据进行解密。

同组同学：岳靖，U201814606

### 实验记录与分析

1. 首先观察数据段，如果是商品附近应有商品名，所以有商品名的附近数据应该是进货价，而不是以$结尾的数据就应该是是密码所在了。如图4.10所示。这个地方同学用了$来混淆，我根据实验任务书给的数据段定义想到店名会在密码附近，所以刚好找到了我需要的加密后的密码。

图片包含 钟表, 物体, 街道, 大

描述已自动生成

图4.10 密码所在地

1. 接着就是一行一行看代码，遇到有可能是反跟踪程序的地方，利用TD修改IP使原本的指令不再生效，即可跳过了反跟踪，如图4.11所示。这个同学用了几种不同的加密方式，不过还是能算得过来，因此，直接一个个试就试出了他的加密方式（x-26h）\*3，再根据找到的密码即可算出他的密码是123456。

电脑屏幕的照片

描述已自动生成

图4.11加密方式

1. 验证加密的密码，如图4.12a，b所示

图片包含 游戏机

描述已自动生成

图4.12a 解密失败

手机屏幕的截图

描述已自动生成

图4.12b 解密成功

## 总结

1. 中断向量表的段首址是0，在设置中断向量表时需要保护DS。
2. Call和jmp都可以进入原中断程序，但相比而言，前者更加灵活。
3. 8h中断自带CLI，因此在后面的操作中需要加上STI来开中断，不然系统会卡死，且无法输入。
4. 中断编号\*4即为中断入口地址。
5. 在本实验中，经测试，若不使用原中断程序，则无法完成新功能。
6. 跟踪破解就是找到相应的加密方式和加密后的数据，而采用中断反跟踪可以防止使用断点绕过反跟踪，再加上计时可以很大程度抑制了反跟踪。
7. 如果明文存放数据，只要一行行看数据段，看到有可能的密码加以尝试即可破解。
8. 反跟踪代码一般有CLI,STI的字样。可通过修改IP直接跳过。

# WIN32/64编程

## 实验目的

（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）。

社交网络的手机截图

描述已自动生成

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）。

手机屏幕截图

描述已自动生成

图5.2 商品信息显示示意图

## 任务5.1实验过程

### 实验方法说明

ListSort处理流程：

1. 计算所有商品推荐度。
2. 将所有推荐度存放在一个变量temp里。
3. 对这个变量的数据冒泡排序。
4. 用排序好的第一个推荐度数据与每一个商品推荐度进行比较，相同的输出该商品所有信息，temp+1，直到输出所有商品信息。
5. 程序退出
6. 部分代码展示：

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\_level  
cmp ax,word ptr buf[ebx+22].recommend\_level  
  
    jae loopm  
jmp loopv  
loopm:  
     mov ax,word ptr buf[ebx].recommend\_level  
 mov cx,word ptr buf[ebx+22].recommend\_level  
 mov word ptr buf[ebx+22].recommend\_level,ax  
 mov word ptr buf[ebx].recommend\_level,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].quantity  
 mov cx,word ptr buf[ebx+22].quantity  
 mov word ptr buf[ebx+22].quantity,ax  
 mov word ptr buf[ebx].quantity,cx   
 mov ax,word ptr buf[ebx].sold\_num  
 mov cx,word ptr buf[ebx+22].sold\_num  
 mov word ptr buf[ebx+22].sold\_num,ax  
 mov word ptr buf[ebx].sold\_num,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

### 实验记录与分析

1. 使用软件VC++6.0。
2. 测试About，如图5.1所示。

手机屏幕截图

描述已自动生成

图5.1 About

1. 测试Recommendation，如图5.2所示。

手机屏幕截图

描述已自动生成

图5.2 Recommendation

1. 测试ListSort，如图5.3所示。

手机屏幕截图

描述已自动生成

图5.3 ListSort

1. 在VC++里很容易知道窗口的ID，如图5.4所示



图5.4窗口ID

那么要把窗口的内容加载到main里，通过以下语句：

invoke LoadMenu ,hInst , 600

invoke SetMenu ,hwnd,eax

invoke ShowWindow, hwnd,SW\_SHOWNORMAL

invoke UpdateWindow, hwnd

1. 编辑窗口内容在VC++里是可视化操作，不需要知道代码，如图5.5所示

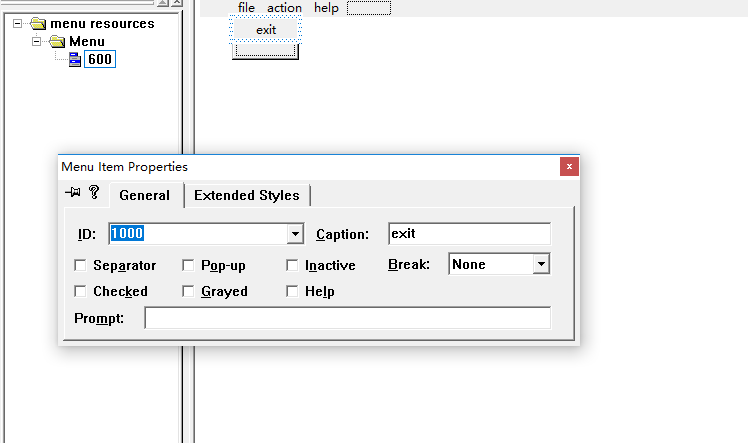


图5.5可视化编辑窗口内容

1. 需要注意的是，在wndproc函数里，比较输入是否为点击窗口某个内容时，最好直接引用ID号，否则就需要在rc文件里define，如图5.6所示

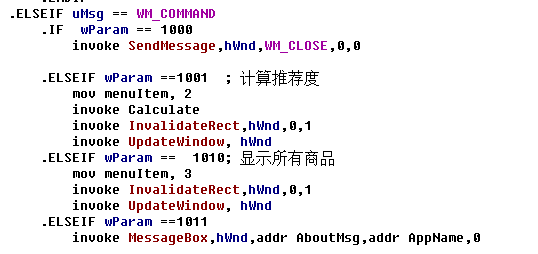


图5.6 ID比较

在这里，1000对应的就是exit操作，1001,1010,1011对应的就是其他三个操作。

1. 在WIN32中，dos中的1，2号，9，10号功能不能再使用，而采取了类似于C的textout函数，相比于dos中的功能，更加方便，而且，字符串的结尾不再采用$结尾，而是直接以0结尾。
2. 对于按键的消息是WM\_COMMAND，如图5.7所示。

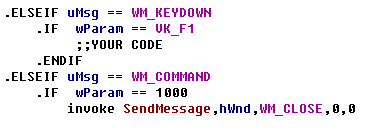


图5.7 按键消息

## 总结

1. VS2019采用插件asmdube，对于windows.inc会有警告，因此我使用了VC++6.0，在创建好winapp的文件后，大框架的代码，比如wndmain函数就已经给好了，非常方便。
2. 通过这次实验的编程，更好的理解了基于窗口的应用程序是怎么工作的，wndmain分发消息，wndproc处理消息后，根据usg的值确定用户的操作，比如点击，按键，销毁……再根据wparam的值可以确定用户对菜单哪个按钮进行了操作，从而调用相应的函数。而函数，与我们之前的任务中所写的代码基本无差。

# 参考文献

[1] 王元珍 曹忠升 韩宗芬.《80x86汇编语言程序设计》华中科技大学出版社