

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验时段： 2018年3月-2018年5月**

**指导教师： 周英飙**

**专业班级： 计卓1601班**

**学 号： U201612922**

**姓 名： 刘子煜**

实验报告成绩：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实验 序号 | 一 | 二 | 三 | 四 | 五 | 总评 |
| 成绩 |  |  |  |  |  |  |

备注：总评是实验报告的最终成绩，是五次成绩的平均值，它将与平时实验过程中的表现一起构成本实验课程的最终成绩。

指导教师签字：

                     日期：

**计算机科学与技术学院**



**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验一 编程基础 （一）**

**实验时间： 2018-3-29，8：00-11：50 实验地点： 南一楼808室38号实验台**

**指导教师： 周英彪**

**专业班级：计算机卓越工程师201601班**

**学 号： U201612922 姓 名： 刘子煜**

**同组学生： 无 报告日期： 2018年 3 月 29日**

**原创性声明**

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

特此声明！

学生签名：

日期：2018.3.29

成绩评定

|  |  |  |
| --- | --- | --- |
| 实验完成质量得分（70分）（实验步骤清晰详细深入，实验记录真实完整等） | 报告撰写质量得分（30分）（报告规范、完整、通顺、详实等） | 总成绩（100分） |
|  |  |  |

指导教师签字：

                     日期：

**目录**

[1 实验目的与要求 1](#_Toc476742634)

[2 实验内容 1](#_Toc476742635)

[3 实验过程 2](#_Toc476742636)

[3.1 任务1 2](#_Toc476742637)

[3.1.1 设计思想及存储单元分配 2](#_Toc476742638)

[3.1.2 实验步骤 2](#_Toc476742638)

[3.1.3 实验记录与分析 2](#_Toc476742639)

[3.2 任务2 5](#_Toc476742640)

[3.2.1 设计思想及存储单元分配 5](#_Toc476742641)

[3.2.2 源代码 5](#_Toc476742642)

[3.2.3 实验步骤 6](#_Toc476742644)

[3.2.4 实验记录与分析 7](#_Toc476742645)

[3.3 任务3 9](#_Toc476742640)

[3.3.1 设计思想及存储单元分配 9](#_Toc476742641)

[3.3.2 源代码 9](#_Toc476742642)

[3.3.3 实验步骤 1](#_Toc476742644)0

[3.3.4 实验记录与分析 1](#_Toc476742645)0

[4 总结与体会 1](#_Toc476742646)2

[参考文献 1](#_Toc476742647)3

# 实验目的与要求

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

(1)掌握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。

# 实验内容

任务1：《80X86汇编语言程序设计》教材中 P31的 1.14题。

要求：

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

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

(3) 记录上机执行后的结果，与（2）中对应的内容比较。

(4)求差运算中，若将A、B视为有符号数，且A>B, 标志位有何特点？若将A、B视为无符号数，且A>B, 标志位又有何特点？

任务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。

(3)在标号LOPA前加上一段程序，实现新的功能：先显示提示信息“Press any key to begin!”, 然后，在按了一个键之后继续执行LOPA处的程序。

任务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，对数据段中变量访问时所用到的寻址方式中的寄存器改成32位寄存器。 (2)内存单元中数据的访问采用变址寻址方式。

(3)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。

(4)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形式）。

(5)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。

# 实验过程

## 任务1

## 3.1.1设计思想及存储单元分配

寄存器分配：AH和AL，运算后的结果存储在AH中

### 3.1.2实验步骤

1.准备上机实验环境。

2.在TD的代码窗口中的当前光标下分别输入三个运算式对应的两个8位数值对应的指令语句 （1）MOV AH,+0110011B；MOV AL,+1011010B；ADD AH,AL；

（2）MOV AH,+0110011B；MOV AL,+1011010B；SUB AH,AL；

（3）MOV AH,-0101001B；MOV AL,-1011101B；ADD AH,AL；

（4）MOV AH,-0101001B；MOV AL,-1011101B；SUB AH,AL；

（5）MOV AH,+1100101B；MOV AL,-1011101B；ADD AH,AL；

（6）MOV AH,+1100101B；MOV AL,-1011101B；SUB AH,AL；

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

预计

(1)ADD执行之后（AX）= 8D5A SF = 1、OF = 1、CF = 0、ZF = 0

(2)SUB执行之后 (AX) = D95A SF = 1、OF = 0、CF = 1、ZF = 0

(3)ADD执行之后（AX）= 7AA3 SF = 0、OF = 1、CF = 1、ZF = 0

(4)SUB执行之后 (AX) = 34A3 SF = 0、OF = 0、CF = 0、ZF = 0

(5)ADD执行之后（AX）= 08A3 SF = 0、OF = 0、CF = 1、ZF = 0

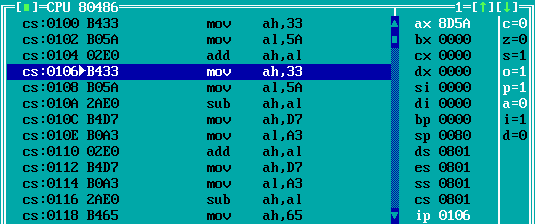
(6)SUB执行之后 (AX) = C2A3 SF = 1、OF = 1、CF = 1、ZF = 0

### 3.1.3实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

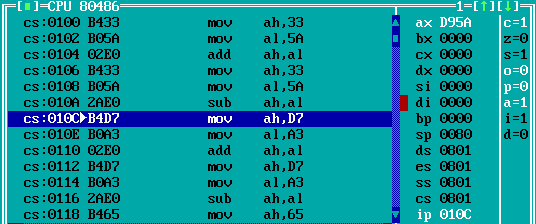
2. 输入第二步中所有指令，并进行单步操作后，得到结果如图3.1.1所示。可以看出，计算结果在AX的高字节中与标志位的状态与事前预期的是一致的。

（1）MOV AH,+0110011B；MOV AL,-1011010B；ADD AH,AL；（求和）



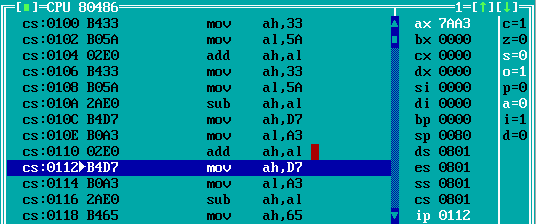
分析结果：标志寄存器区c=0，说明没有进位，z=0，说明不为0，o=1，说明产生溢出，s=1，说明结果为负数。

（2）MOV AH,+0110011B；MOV AL,-1011010B；SUB AH,AL；



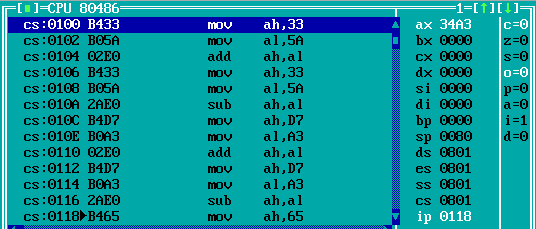
分析结果：标志寄存器区c=1，说明出现进位，z=0，说明不为0，o=0，说明没有溢出，s=1，说明结果为负数。

（3）MOV AH,-0101001B；MOV AL,-1011101B；ADD AH,AL；



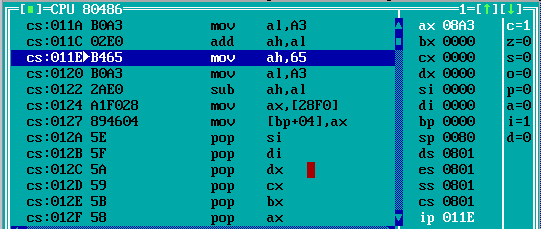
分析结果：标志寄存器区c=1，说明出现进位，z=0，说明不为0，o=1，说明出现溢出，s=0，说明结果为正数。

（4）MOV AH,-0101001B；MOV AL,-1011101B；SUB AH,AL；



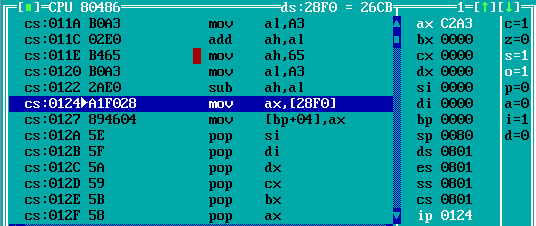
分析结果：标志寄存器区c=0，说明没有进位，z=0，说明不为0，o=0，说明没有溢出，s=0，说明结果为正数。

（5）MOV AH,+1100101B；MOV AL,-1011101B；ADD AH,AL；



分析结果：标志寄存器区c=1，说明出现进位，z=0，说明不为0，o=0，说明没有溢出，s=0，说明结果为正数。

（6）MOV AH,+1100101B；MOV AL,-1011101B；SUB AH,AL；



分析结果：标志寄存器区c=1，说明出现进位，z=0，说明不为0，o=1，说明出现溢出，s=1，说明结果为负数。

图3.1.1 执行完每一组测试语句后的状态

## 任务2

### 3.2.1设计思想及存储单元分配

### AL初值为0，每一次循环使AL+1，再赋予数据段中不同位置以数值，CX用来空着循环的次数。

### 3.2.2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 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

### 3.2.3 实验步骤

1.准备上机实验环境。

2.使用记事本录入源程序，存盘文件名为TEST2.ASM。使用MASM 6.0汇编源文件。即MASM TEST2；观察提示信息，若出错，则用编辑程序修改错误，存盘后重新汇编，直至不再报错为止。

3.使用连接程序LINK.EXE将汇编生成的TEST2.OBJ文件连接成执行文件，即LINK TEST2；若连接时报错，则依照错误信息修改源程序。之后重新汇编和连接，直至不再报错并生成TEST2.EXE文件。

4.使用TD.EXE观察TEST2的执行情况。即 TD TEST2.EXE回车

（1）观察BX,BP,SI,DI各是多少。

（2）在“MOV CX，10”处设置断点。

（3）点击“RUN”让程序运行至断点处，观察BX, BP,SI,DI的值，并记录，取消断点。

（4）在“INT 21H”处设置断点。

（5）点击“RUN”让程序运行至断点处，观察BX, BP,SI,DI的值，并记录，取消断点。

（6）观察DS数据段开始40个字节的内容，并记录，并与预计进行比较。

预计如下：

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

5.在DATA定义段加入一段程序：

hello DB 'Press any key to begin! $'

6.在标号LOPA前加上一段程序：

LEA DX,hello

MOV AH, 9

int 21h

MOV AH, 7

int 21h

7.存储为TEST3.ASM，汇编，连接文件TEST3，生成TEST3.EXE。

8.使用TD.EXE观察TEST3的执行情况。即 TD TEST3.EXE回车

（1）一步步执行直到弹出窗口。

（2）按下任意键后，可继续执行到结束或直接退出。

### 3.2.4 实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70；MASM.EXE； LINK.EXE; TD.EXE。

2. 编译过程没有发生异常。

3. 连接过程没有发生异常。

4.用TD调入TEST2.EXE后

（1）TD刚运行时，BX,BP,SI,DI的值为0000,0000,0000,0000，如下图所示。

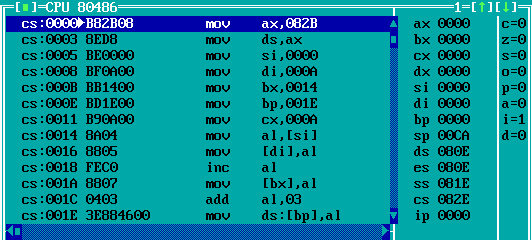


图3.2.1 TD刚运行时的状态

（2）在“MOV CX,10”处设置断点并运行到断点时，BX,BP,SI,DI的值为0014,001E,0000,000A，如下图所示。

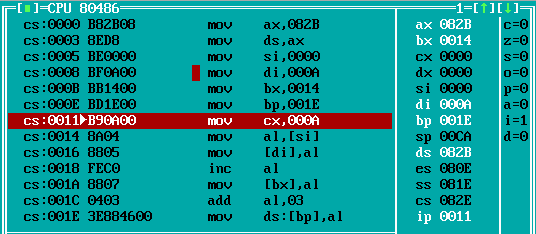


图3.2.2运行至“MOV CX,10”时的状态

（3）在“INT 21H”处设置断点并运行到断点时，BX,BP,SI,DI的值为001E,0028,000A,0014，如下图所示。

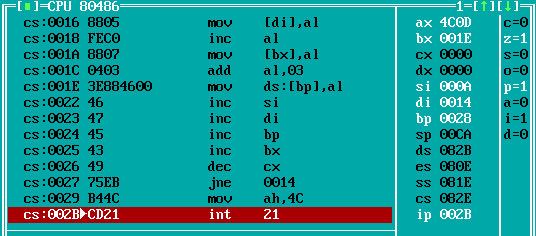


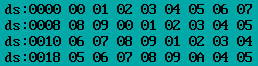
图3.2.3运行至“INT 21H”时的状态

（4）在程序运行中数据段中第1+n位，第10+n位，第20+n位，第30+n位在第n次循中，值发生改变，分别为：0+n，0+n，1+n，4+n。

运行结束前数据段中开始40位的值为：

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

结果与预测相符，运行结果如下图所示：



9RUQ@A(G(M(Z9E[8KI]X}20

图3.2.5程序结束前数据段前40位的值

5.在DATA中定义要输出的内容：'Press any key to begin! $'（$是字符串结束标志），在代码段中添加“LEA DX,hello；MOV AH, 9；int 21h；MOV AH, 7；int 21h”，其中第一条意为将hello的偏移地址存入DX，第二条意为显示DS:DX处的字符串，第四条用于无回显的键盘输入。

6.在重新编辑源程序后，重新编译，连接没有出现异常。

7.用TD调入TEST3.EXE后

（1）运行完“MOV AH,7；int 21h”后弹出提示“Press any key to begin!”如下图所示：

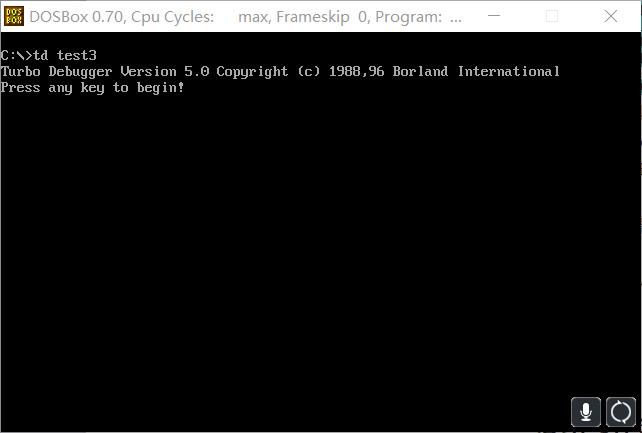


图3.2.6修改程序后弹出的提示

（2）输入任意键后，回到程序，如下图所示：

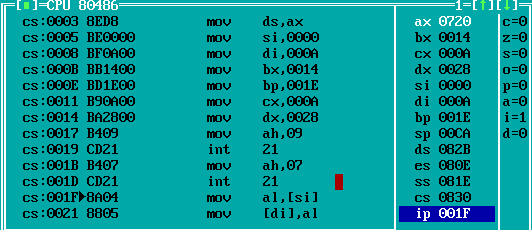


图3.2.7按下任意键后继续回到程序

## 任务3

### 3.3.1设计思想及存储单元分配

### AL初值为0，每一次循环使AL+1，用BUF[ESI]进行数据的记录，CX用来空着循环的次数。

**3.3.2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 AX, DATA

MOV DS, AX

MOV ESI, 0

MOV CX, 10

LOPA: MOV AL, BUF1[ESI]

MOV BUF2[ESI], AL

INC AL

MOV BUF3[ESI], AL

ADD AL, 3

MOV BUF4[ESI], AL

INC ESI

DEC CX

JNZ LOPA

MOV AH, 4CH

INT 21H

CODE ENDS

END START

### 3.3.3 实验步骤

### 1.准备上机环境，编辑、汇编、连接文件TEST4。

2.使用TD.EXE观察TEST4的执行情况。即 TD TEST4.EXE回车

设置断点使程序运行至最后，观察DS数据段开始40个字节的内容，并记录，并与预计进行比 较。

预计如下：

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

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

4.理解 IP/EIP指明指令起始位置的重要性。

### 3.3.4 实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70；MASM.EXE； LINK.EXE; TD.EXE。

2.在程序二的基础上使用32位寄存器ESI，并对在内存单元中数据的访问时采用变址寻址方式BUF[ESI]

3. 汇编源程序时，汇编程序报了6个错误。它们是：

TEST4.asm(14): error A2006: undefined symbol : ESI

TEST4.asm(16): error A2006: undefined symbol : ESI

TEST4.asm(17): error A2006: undefined symbol : ESI

TEST4.asm(19): error A2006: undefined symbol : ESI

TEST4.asm(21): error A2006: undefined symbol : ESI

TEST4.asm(22): error A2006: undefined symbol : ESI

经过分析发现出现错误的原因是：源程序的开头没有输入.386，导致在源程序中对32位寄存器的调用出现问题。

4. 连接过程没有发生异常。

5.用TD调入TEST4.EXE后，将程序运行至最后得到如下图所示的结果

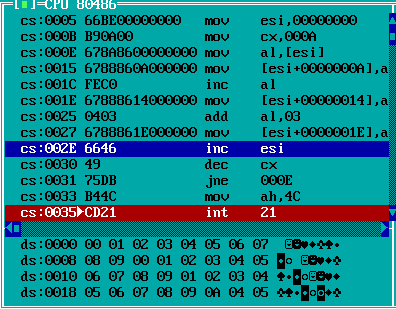
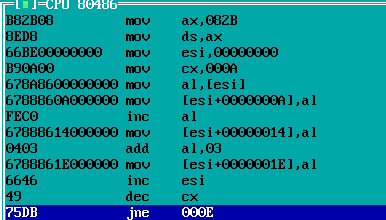
)K1{O@@%MRE`LQEDZTBJM5G

图3.3.1将TEST4运行到结束前数据段开始40个字节的内容

6.如下图所示，从左到右是每条指令对应的机器码以及TD中所提供的反汇编语句



HKFMY0D@F1GBJPCUK08H{58

图3.3.2 TEST4中的机器码及对应的反汇编语句

7.如下图所示是源程序语句

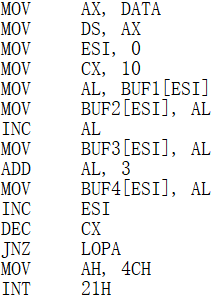


图3.3.3 TEST4中的源程序语句

8.TD程序在反汇编时，未显示伪指令，并将数值拓展成符合该指令中寄存器位数的数，且BUF的值直接显示为偏移量。

9.[IP](https://www.baidu.com/s?wd=IP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9rycsrHIBPv7WnHIbmhnL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjbzPWcYrHR)是指令寄存器，存放当前指令的下一条指令的地址。[CPU](https://www.baidu.com/s?wd=CPU&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9rycsrHIBPv7WnHIbmhnL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjbzPWcYrHR)该执行哪条指令就是通过[IP](https://www.baidu.com/s?wd=IP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9rycsrHIBPv7WnHIbmhnL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjbzPWcYrHR)来指示的，E[IP](https://www.baidu.com/s?wd=IP&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9rycsrHIBPv7WnHIbmhnL0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPjbzPWcYrHR)是32位机的指令寄存器。

# 总结与体会

通过任务1的实验，让我知道了TD不仅可以调入现成的执行程序进行调试，而且能随时输入和测试单一一条指令是否正确，执行效果如何，这将方便未来的学习过程。另外，通过观察，计算机内的加减运算，无论是否有符号数，对应的标志位都是设定好了的，如何使用这些标志，完全由程序员选择的指令来决定，这就要求我们编写程序时要理解好题目的语义，选择合适的指令语句，而不是我写了个负数交给汇编程序，系统就会自动选择有符号指令的。

任务2的实验表明，源程序定义的段的次序，与调到内存中段的次序是有对应关系的。但数据段开始没有被赋值到实际的数据段首址中，需要执行了我们的相关赋值语句后才变得实际有效。另外，各种寻址方式在内存中确实不如源程序直观了，但可以看到变量的偏移值都被很好的计算出来了，我们可以通过源程序中变量与内存中对应位置的信息获得该变量的偏移地址，方便我们在数据显示区输入该地址，查看对应的变量内容。

同时任务2也让我学会了如何在程序中显示出字符串，也弄懂了这些调用语句具体代表的含义。

任务3是对任务2的改写，将任务2中的寄存器变为32位， 使用变址寻址方式，并通过在TD与任务2中反编译码和机器码的对比，进一步理解了不同寻址方式在编译过程中的差异，同时通过这一系列实验也明白了IP/EIP在程序中的用途和重要性。

本次上机不仅提高了编程水平，熟悉了工具的使用，而且加深了对一些知识的理解。主要的经验教训如下：

首先，更加感受到实验前准备的意义。例如：上机前准备越充分（如先编好源程序，制定好准备做的一些步骤），上机的时候目的越明确，可以解决较多的问题。

其次，录入程序时要注意一些细节，比如中文分号、字母O等问题，虽然汇编程序指出其所在行有错，但很难发现具体是哪个符号错了，耽误了不少时间。TD在程序细节的观察、动态修改方面有很大的作用，要主动用TD的调试功能帮助自己发现、理解与解决问题。

最后，由于操作不够熟练，时间比较紧张等原因，还有些问题需要以后进一步解决，如堆栈中数据变化的原因、各个段在内存中存放的关系、是否可跟踪到INT 21H中去、多次调入程序时初始的段值是否相同等等。

# 参考文献

[1] 作者：王元珍 曹忠升 韩宗芬，书名：80X86汇编语言程序设计，出版者：华中科技大学出版社。

[2] URL:https://wenku.baidu.com/view/137903edc77da26924c5b014.html



**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验一 编程基础 （二）**

**实验时间： 2018-4-12，8：00-11：50 实验地点： 南一楼804室**

**指导教师： 周英飚**

**专业班级：计算机卓越工程师班1601班**

**学 号： U201612922 姓 名： 刘子煜**

**同组学生： 无 报告日期： 2018年 4 月 13日**

**原创性声明**

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

特此声明！

学生签名：

日期：2018.4.13

成绩评定

|  |  |  |
| --- | --- | --- |
| 实验完成质量得分（70分）（实验步骤清晰详细深入，实验记录真实完整等） | 报告撰写质量得分（30分）（报告规范、完整、通顺、详实等） | 总成绩（100分） |
|  |  |  |

指导教师签字：

                    日期：

**目录**

[1 实验目的与要求 1](#_Toc476742634)

[2 实验内容 1](#_Toc476742635)

[3 实验过程 3](#_Toc476742636)

[3.1 设计思想及存储单元分配 3](#_Toc476742641)

[3.2 流程图 4](#_Toc476742641)

[3.3 源程序 4](#_Toc476742642)

[3.4 实验步骤 1](#_Toc476742643)1

[3.5 实验记录与分析 1](#_Toc476742644)2

[4 总结与体会 1](#_Toc476742646)6

[参考文献 1](#_Toc476742647)7

# 1 实验目的与要求

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

1. 掌握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。

# 2 实验内容

任务. 设计实现一个网店商品信息查询的程序。

1、实验背景

有一个老板在网上开了2个网店SHOP1,SHOP2；每个网店有n种商品销售，不同网店之间销售的商品种类相同，但数量和销售价格可以不同。每种商品的信息包括：商品名称（10个字节，名称不足部分补0），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），利润率（%）【=（销售价\*已售数量-进货价\*进货总数）\*100/（进货价\*进货总数），字类型】。老板管理网店信息时需要输入自己的名字（10个字节，不足部分补0）和密码（6个字节，不足部分补0），登录后可查看商品的全部信息；顾客（无需登录）可以查看所有网店中每个商品除了进货价、利润率以外的信息。

例如：

BNAME DB ‘ZHANG SAN’,0 ;老板姓名

BPASS DB ‘test’，0，0 ；密码

N EQU 30

S1 DB ‘SHOP1’,0 ;网店名称，用0结束

GA1 DB ‘PEN’, 7 DUP(0) ; 商品名称

DW 35，56，70，25，？ ；利润率还未计算

GA2 DB ‘BOOK’, 6 DUP(0) ; 商品名称

DW 12，30，25，5，？ ；利润率还未计算

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

S2 DB ‘SHOP2’,0 ;网店名称，用0结束

GB1 DB ‘BOOK’, 6 DUP(0) ; 商品名称

DW 12，28，20，15，？ ；利润率还未计算

GB2 DB ‘PEN’, 7 DUP(0) ; 商品名称

DW 35，50，30，24，？ ；利润率还未计算

……

2、功能一：提示并输入登录用户的姓名与密码

（1）使用9号DOS系统功能调用，先后分别提示用户输入姓名和密码。

（2）使用10号DOS系统功能调用，分别输入姓名和密码。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，进入功能二的处理。

（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，跳过功能二，进入功能三；若在输入姓名时仅仅输入字符q，则程序退出。

3、功能二：登录信息认证

（1）使用循环程序结构，比较姓名是否正确。若不正确，则跳到（3）。

（2）若正确，再比较密码是否相同，若不同，跳到（3）。

（3）若名字或密码不对，则提示登录失败，并回到“功能一（1）”的位置，提示并重新输入姓名与密码。

（4）若名字和密码均正确，则将1送到AUTH变量中，进到功能三。

**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）。

4、功能三：计算指定商品的利润率。

（1）提示用户输入要查询的商品名称。若未能在第一个网店中找到该商品，重新提示输入商品名称。若只输入回车，则回到功能一（1）。

（2）判断登录状态，若是已经登录的状态，转到（3）。否则，转到（4）。

（3）首先计算第一个网店该商品的利润率PR1，然后在第二个网店中寻找到该商品，也计算其利润率PR2。最后求出该商品的平均利润率APR=(PR1+PR2)/2。进入功能四。

（4）若是未登录状态，则只在下一行显示该商品的名称，然后回到功能一（1）。

**要求尽量避免溢出**。

**提示：**使用循环程序结构，注意寻址方式的灵活使用。结果只保留整数部分。

5、功能四：将功能三计算的平均利润率进行等级判断，并显示判断结果。

（1）等级显示方式：若平均利润率大于等于90%，显示“A”；大于等于50%，显示“B”；大于等于20%，显示“C”；大于等于0%，显示“D”；小于0%，显示“F”。

**提示：**使用分支程序结构，采用2号DOS系统功能调用显示结果（注意，“%”是不要出现在计算式子和指令语句中的）。

（2）使用转移指令回到“功能一（1）”处（提示并输入姓名和密码）。

# 3 实验过程

### 3.1设计思想及存储单元分配

将商店信息，账户信息，商品信息，利润率以及提示信息等放入数据段中，在进行登录，查询等操作时将信息调入寄存器中进行比较，计算，输出显示等操作。

1.存储单元分配

LIN：用户名输入提示。

LIP：密码输入提示。

FIN：查询输入提示。

in\_name：输入用户名储存。

in\_pwd：输入密码储存。

PR1：商店1利润率。

PR2：商店2利润率。

APR：平均利润率。

AUTH：用户身份。

MISN：用户名错误提示。

MISP：密码错误提示。

BNAME：商店老板用户名。

BPASS：商店老板密码。

COUNTN：用户名长度。

COUNTP：密码长度。

N：商店数量。

GN：输入的要查询的商品的名称。

S1：商店1名称。

GA1：商店1的商品1的信息。

GA2：商店1的商品2的信息。

S2：商店2名称。

GB1：商店2的商品1的信息。

GB2：商店2的商品2的信息。

2.寄存器分配

CX：计数器。

AX、BX、DX、SI、DI、EAX、EBX、ECX、EDX：临时寄存器。

### 3.2流程图

图3.2是网店信息查询系统的程序流程图。

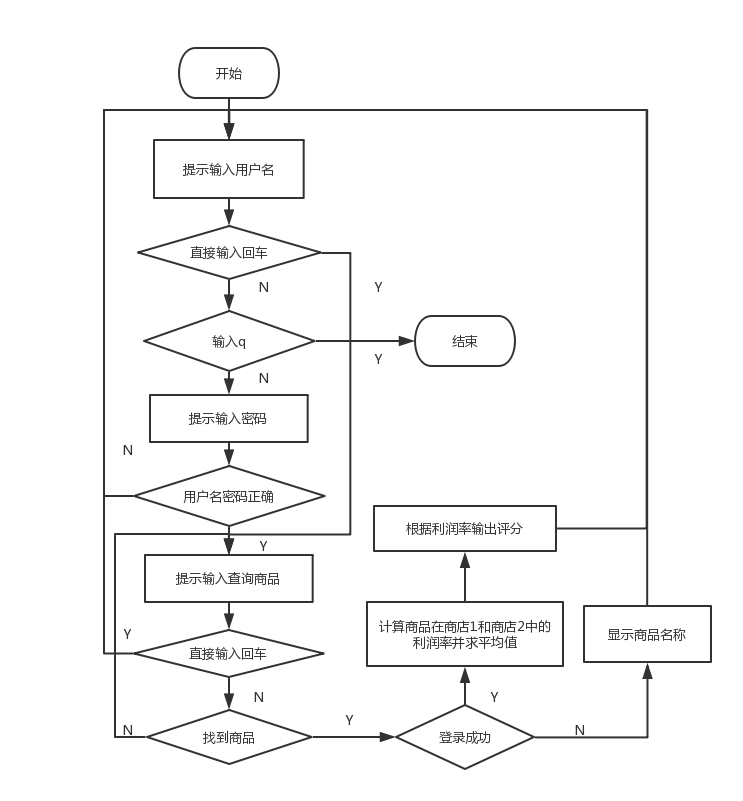


图3.2 网店信息查询系统的程序流程图

### 3.3源程序

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

LIN DB 'Please enter your username:$'

LIP DB 'Please enter your password:$'

FIN DB 'Please enter what you want:$'

in\_name DB 10,0,10 DUP(0)

in\_pwd DB 6,0,6 DUP(0)

PR1 DD 0

PR2 DD 0

APR DD 0

RSA DB 'A$'

RSB DB 'B$'

RSC DB 'C$'

RSD DB 'D$'

RSF DB 'F$'

QUIT DB 'p$'

KONG DB 10

AUTH DB 0

MISN DB 'Username error!$'

MISP DB 'Password error!$'

BNAME DB 'LIU ZIYU',0,0

BPASS DB 'TEST',0,0

COUNTN EQU 8

COUNTP EQU 4

N DD 2

GN DB 10,0,10 DUP(0)

S1 DB 'SHOP1',0

GA1 DB 'PEN',7 DUP(0)

DW 35,56,70,25,?

GA2 DB 'BOOK',6 DUP(0)

DW 12,30,25,5,?

S2 DB 'SHOP',0

GB1 DB 'BOOK',6 DUP(0)

DW 12,28,20,15,?

GB2 DB 'PEN',7 DUP(0)

DW 35,50,30,24,?

DATA ENDS

CODE SEGMENT USE16

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

START: MOV AX, DATA

MOV DS, AX

RP: LEA DX, LIN

MOV AH, 9

INT 21H

LEA DX, in\_name

MOV AH, 10

INT 21H

CMP in\_name+2, 13

JE COS

CMP in\_name+2, 'q'

JE EXIT

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

LEA DX, LIP

MOV AH, 9

INT 21H

LEA DX, in\_pwd

MOV AH, 10

INT 21H

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

MOV AL, in\_name+1

CMP AL, COUNTN

JNE UNMS

MOV CX, COUNTN

LEA SI, in\_name+2

LEA DI, BNAME

MOV CX, COUNTN

RCM: MOV AL, [SI]

CMP AL, [DI]

JNE UNMS

INC SI

INC DI

DEC CX

JE EQ1

JMP RCM

EQ1: MOV AL, in\_pwd+1

CMP AL, COUNTP

JNE PWMS

LEA SI, in\_pwd+2

LEA DI, BPASS

MOV CX, COUNTP

RCM2: MOV AL, [SI]

CMP AL, [DI]

JNE PWMS

INC SI

INC DI

DEC CX

JE EQ2

JMP RCM2

EQ2: MOV AL, 1

MOV AUTH, AL

JMP FUNC3

COS: MOV AL, 0

MOV AUTH, AL

JMP FUNC3

FUNC3: LEA DX, FIN

MOV AH, 9

INT 21H

LEA DX, GN

MOV AH, 10

INT 21H

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

MOV BX, 0

CMP GN+2, 13

JE RP

MOV CX, 0

MOV CL, [GN+1]

COMP: LEA SI, GN+2

LEA DI, [GA1+BX]

RCM3: MOV AL, [SI]

CMP AL, [DI]

JNE ALL

INC SI

INC DI

DEC CX

JE CPL

JMP RCM3

BCP: CMP AUTH, 0

JE NIN

MOV AX, WORD PTR [GA1+BX+10]

CWDE

MOV ECX, EAX

MOV AX, WORD PTR [GA1+BX+14]

CWDE

IMUL ECX, EAX

MOV AX, WORD PTR [GA1+BX+12]

CWDE

MOV EDX, EAX

MOV AX, WORD PTR [GA1+BX+16]

CWDE

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR1, EAX

MOV BX, 0

MOV CX, 0

MOV CL, [GN+1]

COMP2: LEA SI, GN+2

LEA DI, [GB1+BX]

RCM4: MOV AL, [SI]

CMP AL, [DI]

JNE ALL2

INC SI

INC DI

DEC CX

JE CPL2

JMP RCM4

BCP2: MOV AX, WORD PTR [GB1+BX+10]

CWDE

MOV ECX, EAX

MOV AX, WORD PTR [GB1+BX+14]

CWDE

IMUL ECX, EAX

MOV AX, WORD PTR [GB1+BX+12]

CWDE

MOV EDX, EAX

MOV AX, WORD PTR [GB1+BX+16]

CWDE

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR2, EAX

MOV EDX, PR1

ADD EAX, EDX

CDQ

MOV EBX, 2

IDIV EBX

MOV APR, EAX

JMP FUNC4

CPL: CMP BYTE PTR [DI+1], 0

JE BCP

JMP ALL

CPL2: CMP BYTE PTR [DI+1], 0

JE BCP2

JMP ALL2

NIN: MOV CL, [GN+1]

LEA SI, GN+2

PRN: MOV DX, [SI]

MOV AH, 2

INT 21H

INC SI

DEC CL

JNE PRN

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

JMP RP

ALL: CMP BX, 40

JE FUNC3

ADD BX, 20

JMP COMP

ALL2: ADD BX,20

JMP COMP2

UNMS: LEA DX, MISN

MOV AH, 9

INT 21H

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

JMP RP

PWMS: LEA DX, MISP

MOV AH, 9

INT 21H

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

JMP RP

FUNC4: CMP AX, 90

JGE PA

CMP AX, 50

JGE PB

CMP AX, 20

JGE PC

CMP AX, 0

JGE PD

JMP PF

PA: LEA DX, RSA

MOV AH, 9

INT 21H

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

JMP RP

PB: LEA DX, RSB

MOV AH, 9

INT 21H

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

JMP RP

PC: LEA DX, RSC

MOV AH, 9

INT 21H

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

JMP RP

PD: LEA DX, RSD

MOV AH, 9

INT 21H

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

JMP RP

PF: LEA DX, RSF

MOV AH, 9

INT 21H

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

JMP RP

EXIT: INT 21H

CODE ENDS

END START

### 3.4实验步骤

1. 准备上机实验环境、编辑、汇编、连接文件shop。

2. 执行该程序。即在命令行提示符后输入shop后回车，观察执行现象。

3. 输入正确用户名密码，商品名，观察执行现象。

4. 输入错误用户名密码，观察执行现象。

5. 输入错误商品名，观察执行现象。

6. 用户名输入回车并输入正确的商品名，观察执行现象。

7. 用户名输入q，观察执行现象。

8．商品名输入回车，观察执行现象。

### 3.5实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

2. 汇编过程没有发生异常

3. 连接过程没有发生异常。

4. 执行之后提示输入用户名，如图3.5.1所示：

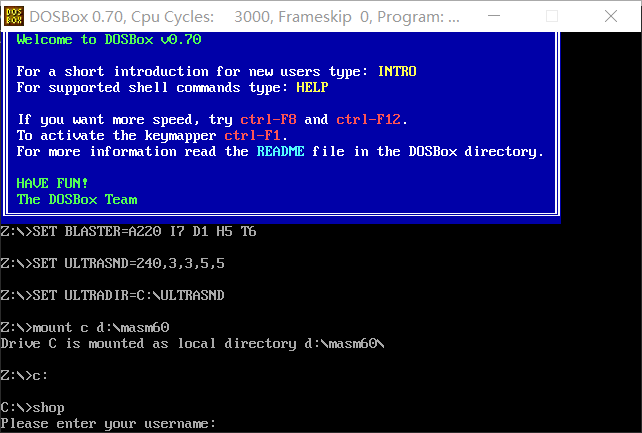


图3.5.1 执行程序后

5. 输入正确的用户名密码商品名后，计算该商品的平均利润率并评级，并返回再次输入用户名，如图3.5.2所示：

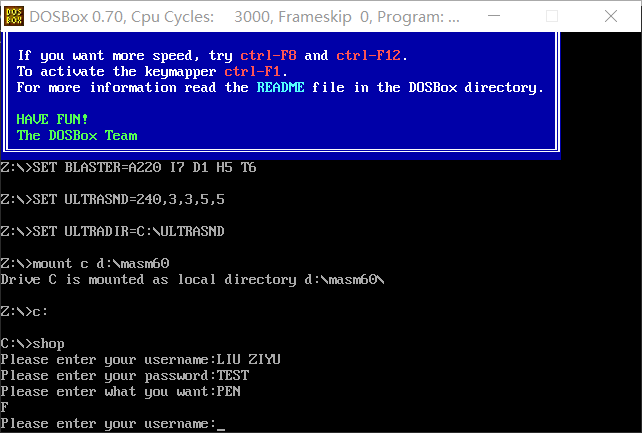


图3.5.2 老板输入正确的商品名

1. 输入错误的用户名密码后，再次提醒输入用户名密码，如图3.5.3所示：

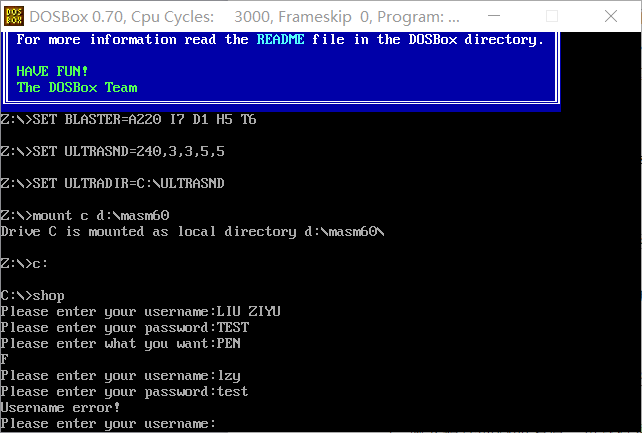


图3.5.3 输入错误用户名或密码

1. 输入错误的商品名后，再次提醒输入商品名，如图3.5.4所示：

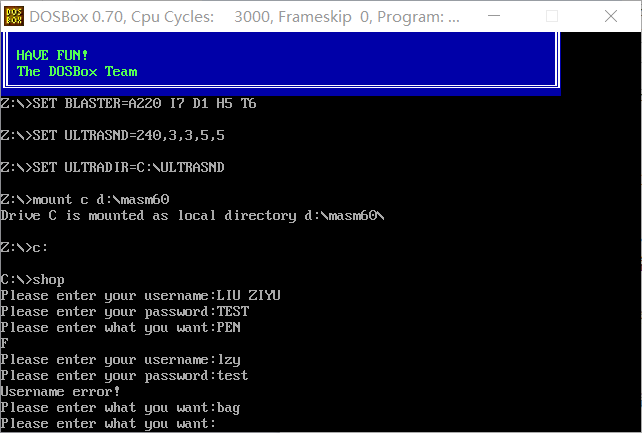


图3.5.4 输入错误的商品名

1. 用户名输入回车并输入正确的商品名后，会在下一行显示所输入的商品的名称，并返回再次输入用户名，如图3.5.5所示：

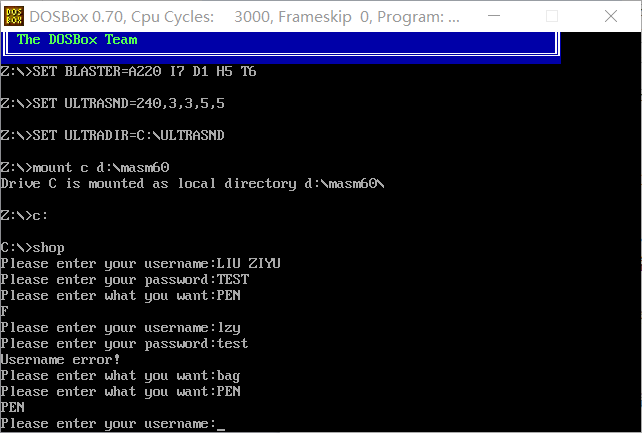


图3.5.5 顾客输入商品名

1. 用户名输入q后，直接退出程序，如图3.5.6所示：

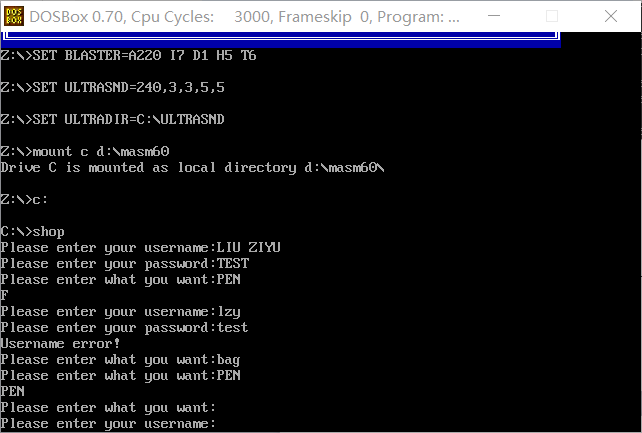


图3.5.6 输入q退出程序

1. 商品名输入回车，返回再次输入用户名，如图3.5.7所示：

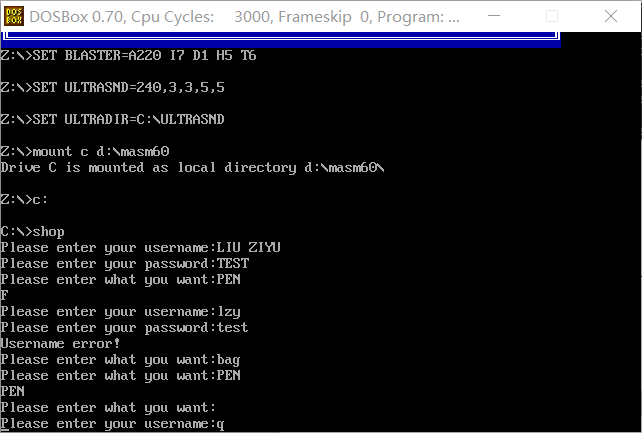


图3.5.7 商品名输入回车

# 4 总结与体会

这次任务是第一次自己独立完成一个功能较为复杂和完整地汇编程序，通过这一次实验让我更好地理解了各种寻址方式的具体表达形式，也让我熟悉了循环，比较，跳转等基础功能的实现，同时也让我对计算溢出问题有了更深的理解，也掌握了一些消除这个问题的方法，同时这次实验中也出现了许多问题，例如指令中两个操作数的类型不确定或不相同等问题，再比如在寄存器使用方面，有时会粗心地在同一块中重复使用一个寄存器导致程序出现错误，这些都是需要在以后的实验中注意和改进的。

本次上机不仅提高了编程水平，熟悉了工具的使用，而且加深了对一些知识的理解。主要的经验教训如下：

这次实验的课前准备不够充足，导致在上课时没能继续将程序完成。同时，这次实验是第一次真正自己写汇编程序，在写的过程中出现了很多基础性的错误，导致debug花费了很多时间，而且很慢才能找出问题的所在，TD调试程序使用的也不够熟练，在找到出问题的语句上浪费了很多时间，以后的实验前我会更加充分地理解题目并争取完成实验内容，课外我也会自行进行练习，让自己更加熟悉编程和调试的环境。

# 参考文献

[1] 作者：王元珍 曹忠升 韩宗芬，书名：80X86汇编语言程序设计，出版者：华中科技大学出版社。



**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验二 程序执行时间与代码长度优化**

**实验时间： 2018-4-19，8：00-11：50 实验地点： 南一楼804室**

**指导教师： 周英飚**

**专业班级：计算机卓越工程师班1601班**

**学 号： U201612922 姓 名： 刘子煜**

**同组学生： 无 报告日期： 2018年 4 月 20日**

**原创性声明**

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

特此声明！

学生签名：

日期：2018.4.20

成绩评定

|  |  |  |
| --- | --- | --- |
| 实验完成质量得分（70分）（实验步骤清晰详细深入，实验记录真实完整等） | 报告撰写质量得分（30分）（报告规范、完整、通顺、详实等） | 总成绩（100分） |
|  |  |  |

指导教师签字：

                      日期：

**目录**

[1 实验目的与要求 1](#_Toc476742634)

[2 实验内容 1](#_Toc476742635)

[3 实验过程 2](#_Toc476742636)

[3.1 任务一 2](#_Toc476742637)

[3.1.1 设计思想及存储单元分配 2](#_Toc476742641)

[3.1.2 流程图 3](#_Toc476742641)

[3.1.3 源程序 4](#_Toc476742642)

[3.1.4 实验步骤](#_Toc476742643) 6

[3.1.5 实验记录与分析](#_Toc476742644) 7

[3.2 任务二 8](#_Toc476742637)

[3.2.1 设计思想及存储单元分配 8](#_Toc476742641)

[3.2.2 流程图 8](#_Toc476742641)

[3.2.3 源程序 8](#_Toc476742642)

[3.2.4 实验步骤](#_Toc476742643) 9

[3.2.5 实验记录与分析](#_Toc476742644) 10

[4 总结与体会 1](#_Toc476742646)1

[参考文献 1](#_Toc476742647)2

# 1 实验目的与要求

1. 了解程序计时的方法以及运行环境对程序执行情况的影响。
2. 熟悉汇编语言指令的特点，掌握代码优化的基本方法。

# 2 实验内容

**任务1. 观察多重循环对CPU计算能力消耗的影响**

应用场景介绍：以实验一任务四的背景为基础，只要顾客买走了网店中的一件商品，老板就需要重新获得全部商品的平均利润率。现假设在双十一零点时，SHOP1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实验一任务四的程序。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，我们让实验一任务四的功能三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。功能三之外的其他功能不纳入到这m次循环体内（但可以保留不变）。

**调整后的功能三的描述：**

（1）在SHOP1中找到“Bag”商品，判断已售数量是否大于等于进货总数，若是，则回到功能一（1），否则将已售数量加1。

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

请按照上述设想修改实验一任务四的程序，并将m和n值尽量取大（比如大于1000，具体数值依据实验效果来改变，逐步增加到比较明显的程度，比如秒级的时间间隔），以得到较明显的效果。

**提示:** （1）在进入调整后的功能三之前增加m次循环的初始化工作，在调整后的功能三结束之后增加m次循环的条件判断和转移语句。

（2）学校汇编教学网站的软件下载中提供了显示当前时间“秒和百分秒”的子程序。若在m次循环前调用一下该子程序，m次循环执行完之后再调用一下该子程序，就能在屏幕上观察并感受到执行循环前后的时间差（时间差值需要自行手工计算，当然，你也可以选用网站上另一个计时程序，它是可以帮你计算好差值的）。注意，由于虚拟机环境下CPU会被分时调度，故该时间差值会因计算机运行环境与状态的不同而不同。

**任务2. 对任务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。减少的越多，评价越高！

**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，尽量采用32位寄存器寻址，能有更多的机会提高指令执行效率。

# 3 实验过程

### 3.1任务一：观察多重循环对CPU计算能力消耗的影响

### 3.1.1设计思想及存储单元分配

将商店信息，账户信息，商品信息，利润率以及提示信息等放入数据段中，在进行登录，查询等操作时将信息调入寄存器中进行比较，计算，输出显示等操作（在此只记录与上次实验不同的部分）。

1.存储单元分配

M：顾客数量。

LOC：当前商品在商店1中的位置。

WLOC：顾客所购买的商品在商店1中的位置。

2.寄存器分配

CX：计数器。

BX：记录偏移地址。

AX、DX、SI、DI、EAX、EBX、ECX、EDX：临时寄存器。

### 3.1.2流程图

图3.2是网店信息查询系统的程序流程图。

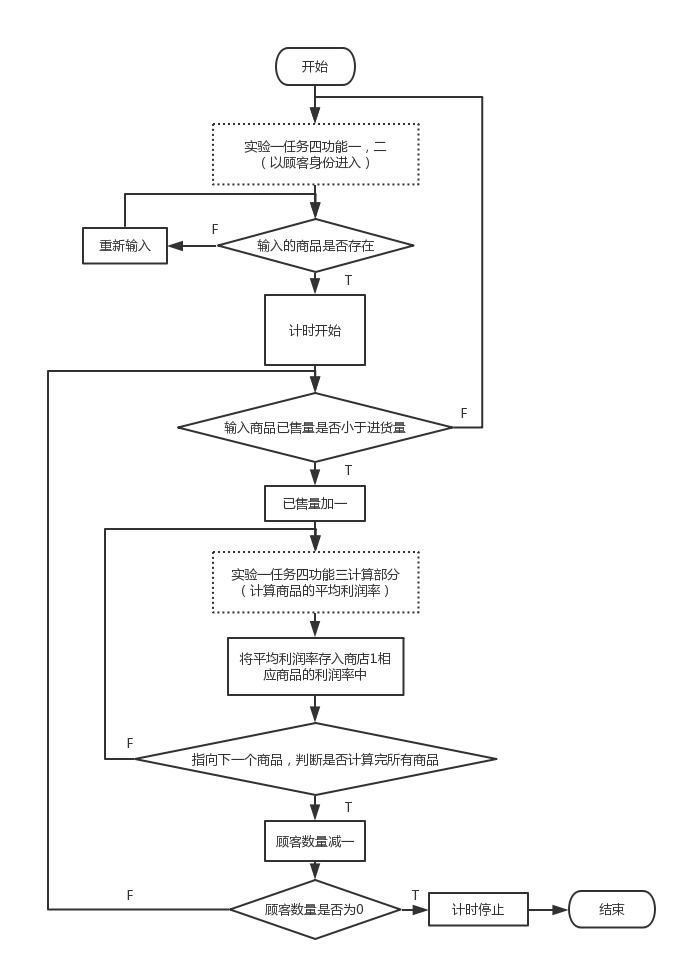


图3.2 网店顾客购买利润率刷新流程图

### 3.1.3源程序

┇

N EQU 30 ；此行之前为实验一任务四数据段相关代码

M DW 2000 ；这三行为本次新增代码

LOC DW 0

WLOC DW 0

GN DB 10,0,10 DUP(0) ；此行之后为实验一任务四数据段相关代码

┇

JMP RCM3 ；此行之前为实验一任务四功能三相关代码

BCP: MOV WLOC, BX ；这四行为本次新增或修改代码

CMP AUTH, 0

JE TST

BAP: MOV AX, WORD PTR [GA1+BX+10]

CWDE ；此行之后为实验一任务四功能三相关代码

┇

MOV PR1, EAX ；此行之前为实验一任务四功能三相关代码

MOV LOC, BX ；这一行为本次新增或修改代码

MOV BX, 0 ；此行之后为实验一任务四功能三相关代码

┇

JMP RCM4 ；此行之前为实验一任务四功能三相关代码

BCP2: MOV EAX, 0 ；这八行为本次新增或修改代码

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [GB1+EBX+10]

MOV CX, WORD PTR [GB1+EBX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [GB1+EBX+12]

MOV DX, WORD PTR [GB1+EBX+16]

IMUL EAX, EDX ；此行之后为实验一任务四功能三相关代码

┇

MOV APR, EAX ；此行之前为实验一任务四功能三相关代码

MOV BX, LOC ；这十二行为本次新增或修改代码

MOV WORD PTR [GA1+BX+18], AX

JMP CAL

ALR: MOV BX, WLOC

MOV AX, M

DEC AX

MOV M, AX

CMP AX, 0

JNE NIN

MOV AX, 1

CALL TIMER

JMP EXIT

CPL: CMP BYTE PTR [DI+1], 0 ；此行之后为实验一任务四功能三相关代码

┇

JMP ALL2 ；此行之前为实验一任务四功能三相关代码

TST: MOV AX, 0 ；这十九行为本次新增或修改代码

CALL TIMER

NIN: MOV CL, [GN+1]

LEA SI, GN+2

NIN: MOV AX, WORD PTR [GA1+BX+14]

MOV CX, WORD PTR [GA1+BX+16]

CMP AX, CX

JNG RP

MOV BX, 0

ALL: ADD BX, 20

CMP BX, 600

JE FUNC3

JMP COMP

ALL2: ADD BX,20

JMP COMP2

CAL: ADD BX, 20

CMP BX, 600

JE ALR

JMP BAP

UNMS: LEA DX, MISN ；此行之前为实验一任务四功能三，四相关代码

┇

添加计时子程序在代码段中：

TIMER PROC

PUSH DX

PUSH CX

PUSH BX

MOV BX, AX

MOV AH, 2CH

INT 21H ;CH=hour(0-23),CL=minute(0-59),DH=second(0-59),DL=centisecond(0-100)

MOV AL, DH

MOV AH, 0

IMUL AX,AX,1000

MOV DH, 0

IMUL DX,DX,10

ADD AX, DX

CMP BX, 0

JNZ \_T1

MOV CS:\_TS, AX

\_T0: POP BX

POP CX

POP DX

RET

\_T1: SUB AX, CS:\_TS

JNC \_T2

ADD AX, 60000

\_T2: MOV CX, 0

MOV BX, 10

\_T3: MOV DX, 0

DIV BX

PUSH DX

INC CX

CMP AX, 0

JNZ \_T3

MOV BX, 0

\_T4: POP AX

ADD AL, '0'

MOV CS:\_TMSG[BX], AL

INC BX

LOOP \_T4

PUSH DS

MOV CS:\_TMSG[BX+0], 0AH

MOV CS:\_TMSG[BX+1], 0DH

MOV CS:\_TMSG[BX+2], '$'

LEA DX, \_TS+2

PUSH CS

POP DS

MOV AH, 9

INT 21H

POP DS

JMP \_T0

\_TS DW ?

DB 'Time elapsed in ms is '

\_TMSG DB 12 DUP(0)

TIMER ENDP

### 3.1.4实验步骤

1. 准备上机实验环境、编辑、汇编、连接文件shopx。

2. 执行该程序。即在命令行提示符后输入shopx后回车。

3. 若购买数量大于库存数，观察执行现象

4. 若购买数量小于库存数，观察执行现象

### 3.1.5实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

2. 汇编过程没有发生异常

3. 连接过程没有发生异常。

4. 执行一段时间后，提示重新输入用户名，即购买数量超过库存数量时中断，如图3.1.5.1所示：

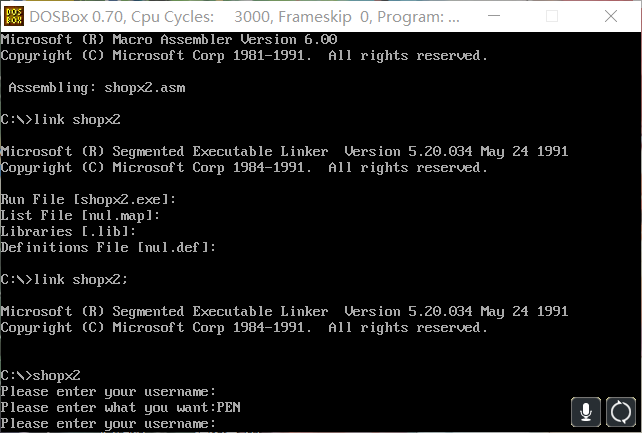


图3.1.5.1 当购买数量超过库存

5. 当库存足够时，显示执行程序所花费的时间，如图3.1.5.2所示：

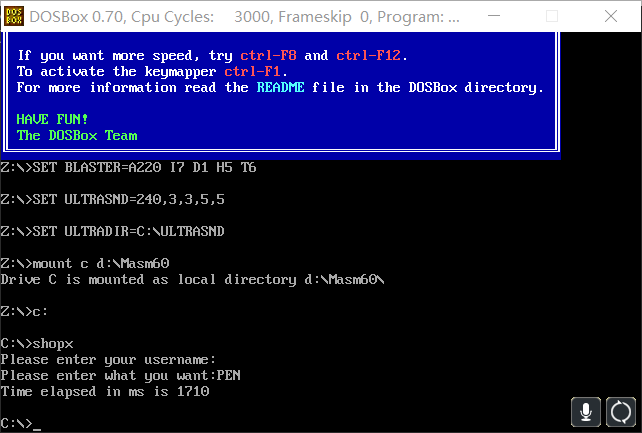


图3.1.5.2 当库存足够

**任务2. 对任务1中的汇编源程序进行优化**

### 3.2.1设计思想及存储单元分配

1.存储单元分配

LOCN:商品偏移地址的偏移地址

BLOC:商店2物品名称偏移地址

### 3.2.2流程图

与任务一相同。

### 3.2.3源程序

┇

JE TST ；此行之前为实验二任务一功能三相关代码

MOV EAX, 0 ；这八行为本次新增或修改代码

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [GA1+BX+10]

MOV CX, WORD PTR [GA1+BX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [GA1+BX+12]

MOV DX, WORD PTR [GA1+BX+16]

IMUL EAX, EDX ；此行之前为实验二任务一功能三相关代码

┇

MOV ECX, 0 ；此行之前为实验二任务一功能三相关代码

CMP M, 2000 ；这几行为本次新增或修改代码

JNE SCMP

COMP2: MOV CL, [GN+1]

LEA DI, GN+2

LEA SI, [GB1+EBX]

RCM4: MOV AL, [DI]

CMP AL, [SI]

JNE ALL2

INC DI

INC SI

DEC ECX

JE CPL2

JMP RCM4

SAV: MOV AX, LOCN

MOV BX, [BLOC+AX]

ADD AX, 2

MOV LOCN, AX

BCP2: MOV EAX, 0

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [GB1+EBX+10]

MOV CX, WORD PTR [GB1+EBX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [GB1+EBX+12]

MOV DX, WORD PTR [GB1+EBX+16]

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR2, EAX

MOV EDX, PR1

ADD EAX, EDX

SAR EAX, 1

MOV APR, EAX

MOV BX, LOC

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

JMP CAL

ALR: MOV AX, 0

MOV LOCN, AX

MOV BX, WLOC ；此行之后为实验二任务一功能三相关代码

┇

ADD EAX, EDX ；此行之前为实验二任务一功能三相关代码

SAR EAX, 1 ；这一行为本次新增或修改代码

MOV APR, EAX ；此行之后为实验二任务一功能三相关代码

┇

CPL2: CMP BYTE PTR [DI+1], 0 ；此行之前为实验二任务一功能三相关代码

JE SAV ；这几行为本次新增或修改代码

JMP ALL2

SCMP: MOV AX, LOCN

MOV BX, [BLOC+AX]

ADD AX, 2

MOV LOCN, AX

JMP BCP2

TST: MOV AX, 0 ；此行之后为实验二任务一功能三相关代码

┇

### 3.2.4实验步骤

1. 准备上机实验环境、编辑、汇编、连接文件shopx2。

2. 执行该程序。即在命令行提示符后输入shopx2后回车。

3. 选择与任务一中相同的商品进行操作，观察运行时间。

### 3.2.5实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

2. 汇编过程没有发生异常

3. 连接过程没有发生异常。

4. 以顾客身份进入并输入商品PEN，输出的运行时间如图3.2.5.1所示：

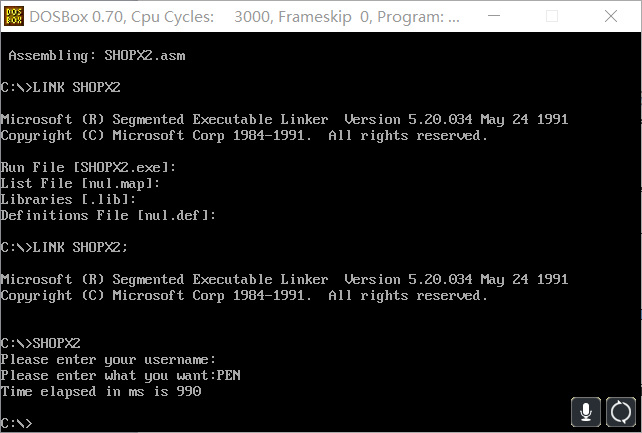


图3.2.5.2 经过优化后运行与任务一相同的命令时所消耗的时间

# 4 总结与体会

这次任务是在上一次任务的基础上添加新的功能，并对自己的程序进行优化，虽然新添加的功能较为简单，但想要对自己的程序进行优化减少其运行时间则要考虑很多方面。

任务一中有两种获取程序执行时间以展示优化效果的方法，一是分别记录开始前和结束后的时间，另一种则是直接展现运行的时间，本程序使用第二种方法，在对循环的控制中，我发现多重循环和过多循环的次数会大量占用CPU的资源，也会使程序执行较为缓慢。通过在不同的环境中执行同一程序，我发现CPU的效率会在很大程度上影响该程序的运行速度，性能越好的CPU执行程序越快。

任务二中通过对程序的优化调试，我发现汇编语言程序的优化可以从以下几个方面进行：

1. 更换效率更高的寻址方式
2. 使用效率更高的计算方法
3. 减少复杂耗时的语句，以耗时少的语句替代
4. 减少多重循环
5. 减少冗余重复的语句

在本次试验中（2）-（4）的优化效果较为明显，（1）带来的优化效果较小，程序中没有发现重复语句（5）的效果没有进行验证。在任务二中我进行了以下几点优化：

1. 将除法指令更换为位移指令（效率提升%5）
2. 更改了防止数据运算溢出的方式（效率提升%5）
3. 将第一次循环找到的商品的偏移地址进行了保存，使程序在以后的循环中不再需要查找商品的位置，减少了一层循环，以牺牲空间的代价提高了时间效率（效率提升30%）

本任务二程序相对于任务程序优化大概40%，很大程度上提高了程序的运行效率。

通过本次实验我清楚地明白了大量循环及多次循环对CPU资源的占用以及对程序运行速度的减缓，同时也发现了许多对程序进行优化的方法，在以后的编程过程中我会尽量写出较为优化的程序，并在功能实现后运行所学到的方法对自己的程序进行检查优化，提高自己程序的效率。

# 参考文献

[1] 作者：王元珍 曹忠升 韩宗芬，书名：80X86汇编语言程序设计，出版者：华中科技大学出版社。



**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验三 模块化程序设计**

**实验时间：2018-4-26， 2018-5-3，8：00-11：50 实验地点： 南一楼804室**

**指导教师： 周英飚**

**专业班级：计算机卓越工程师班1601班**

**学 号： U201612922 姓 名： 刘子煜**

**同组学生： 张佳兴 报告日期： 2018年 4 月 26日 2018年 5 月 3日**

**原创性声明**

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

特此声明！

学生签名：

日期：2018.4.26，2018.5.3

成绩评定

|  |  |  |
| --- | --- | --- |
| 实验完成质量得分（70分）（实验步骤清晰详细深入，实验记录真实完整等） | 报告撰写质量得分（30分）（报告规范、完整、通顺、详实等） | 总成绩（100分） |
|  |  |  |

指导教师签字：

                     日期：

**目录**

[1 实验目的与要求 1](#_Toc476742634)

[2 实验内容 1](#_Toc476742635)

[3 实验过程 4](#_Toc476742636)

[3.1 任务一 4](#_Toc476742637)

[3.1.1 设计思想及存储单元分配 4](#_Toc476742641)

[3.1.2 流程图 9](#_Toc476742641)

[3.1.3 源程序 1](#_Toc476742642)2

[3.1.4 实验步骤](#_Toc476742643) 37

[3.1.5 实验记录与分析](#_Toc476742644) 37

[3.2 任务二 4](#_Toc476742637)1

[3.2.1 设计思想及存储单元分配 4](#_Toc476742641)1

[3.2.2 流程图 4](#_Toc476742641)1

[3.2.3 源程序 4](#_Toc476742642)1

[3.2.4 实验步骤](#_Toc476742643) 60

[3.2.5 实验记录与分析](#_Toc476742644) 60

[4 总结与体会 6](#_Toc476742646)2

[参考文献 6](#_Toc476742647)4

# 1 实验目的与要求

1. 掌握子程序设计的方法与技巧，熟悉子程序的参数传递方法和调用原理；
2. 掌握宏指令、模块化程序的设计方法;
3. 掌握较大规模程序的合作开发与调试方法；
4. 掌握汇编语言程序与C语言程序混合编程的方法；
5. 熟悉C编译器的基本优化方法;
6. 了解C语言编译器的命名方法，主、子程序之间参数传递的机制。

# 2 实验内容

**任务1 宏与子程序设计**

进一步修改与增强实验一任务四的**网店商品信息管理程序**的功能，主要调整功能三。

**1.调整后的功能三的描述**

（1）首先显示一个功能菜单（格式自行定义。若是未登录状态，只显示菜单“1”和“6”）:

1=查询商品信息，2=修改商品信息，3=计算平均利润率，

4=计算利润率排名，5=输出全部商品信息，6=程序退出。

输入1-6的数字进入对应的功能。

（2）查询商品信息

提示用户输入要查询的商品名称。若未能在第一个网店中找到该商品，重新提示输入商品名称。若只输入回车，则回到功能三（1）。

找到该商品之后，按照：“SHOP1，商品名称，销售价，进货总数，已售数量”顺序显示该商品的信息，同时还要将“SHOP2”中该商品的信息也显示出来。显示之后回到功能三（1）。

（3）修改商品信息

提示用户输入要修改信息的商品名称（先指定网店名称）。[若把接下来的处理步骤写成子程序，则网店名称和商品名称（或其偏移地址）就是子程序的入口参数，是否找到、是否是回车或者修改成功的信息是出口参数]。若未能在指定网店中找到该商品，重新提示输入网店名称和商品名称。若只输入回车，则回到功能三（1）。

找到该商品之后，按照：进货价，销售价，进货总数的次序，逐一先显示原来的数值，然后输入新的数值（若输入有错，则重新对该项信息进行显示与修改。若直接回车，则不修改该项信息）。

如：进货价：25》24 //符号“》”仅作为分隔符，也可以选择其他分隔符号

销售价：46》5A6 //输入了非法数值，下一行重新显示和输入

销售价：46》56

进货总数：30》 //直接回车时，对这项信息不做修改

当对三项信息都处理完毕后，回到功能三（1）。

（4）计算平均利润率

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

（5）计算利润率排名

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

（6）输出全部商品信息

将SHOP1和SHOP2中的所有商品信息显示到屏幕上，包括平均利润率和排名（替代了商品原有的利润率字段）。具体的显示格式自行定义（可以分网店显示，也可以按照商品排名显示，等等，显示方式可以作为子程序的入口参数）。回到功能三（1）。

**2.其他要求**

（1）**两人一组**，一人负责包括菜单显示、程序退出在内的主程序，以及菜单中的功能（1）和（2）；另一人负责菜单中的功能（3）、（4）和（5）。各自汇编自己的模块，设计测试方法，测试通过；然后把自己的模块交给对方，各自把对方的程序整合到自己的程序中，连接生成一个程序，再进行整体调试。

**注意，**在每个模块的开始，注明编写者的名字以及同组同学的名字。整合到一起时，要注意删掉自己测试时额外加的代码，若有重复的模块（如：两个人都会使用进制转换子程序，各自模块中可能都有相同的进制转换程序），也需要去掉重复的部分。

**建议分组方法：**按照学号顺序依次两人一组，若班级人数为奇数，则最后三人一组（其中两人的分工是相同的，第三人只需要选择其中一个同学的模块与自己模块整合即可）。

（2）排名的基本要求是按照平均利润率从高到低计算名次，也可以考虑按照指定字段（比如已售数量等）排名。相同平均利润率时排名相同，下一个相邻平均利润率的名次应该是排名在前的所有商品种类“和”的下一个数值。

（3）将9号和10号DOS系统功能调用定义成宏指令并调用。功能（1）-（5）应尽量采用子程序方式实现。需要借鉴书上的进制转换程序：十进制转二进制的子程序F10T2和二进制转十进制的子程序F2T10。

**上述任务1中，值得思考或者写到操作步骤中的问题提示：**

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。
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。
3. 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。
4. 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。
5. 在编程中使用不同的子程序参数传递方法来编写子程序。
6. 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？
7. 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。
8. 如何使菜单等显示信息显示得更漂亮一点？
9. EXTRN说明语句放在.386之前或者之后有什么区别？
10. 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？如果不能带入，是否可以通过加段前缀的方法来解决？
11. 如何利用宏功能使汇编语言的程序变得更加直观易读？

**任务2：在C语言程序中调用 汇编语言实现的函数**

对于任务1的程序进行改造，主控程序、以及输入输出较多的某一个功能（如功能（1）、（2）、（5）中的某一个）用C语言实现，其他功能用独立的汇编语言子程序的方式实现； 在C语言程序中调用汇编语言子程序。

**要求与提示：**

1. 在不同的C语言开发环境中实现与汇编语言程序的混合编程，其操作方法有可能是不同的。请大家选择自己熟悉的C语言开发环境并查找相关的资料完成本实验（建议用BC31，其功能与操作方法相对比较简单）。
2. 在实验报告中，比较详细的给出你的开发环境及其实现方法。
3. 观察C语言编译器中对各种符号的命名规则（指编译器内部可以识别的命名规则，比如，符号名前面是否加下划线“\_”，等），主、子程序之间参数传递的机制，通过堆栈传递参数后堆栈空间回收的方法（要设计一个有多个参数需要传递的C函数）。
4. 对混合编程形成的执行程序，用调试工具观察由C语言形成的程序代码与由汇编语言形成的程序代码之间的相互关系，包括段、偏移的值，汇编指令访问C的变量时是如何翻译的，等。
5. 请尝试在C语言源程序中不合理地嵌入汇编语言的指令语句，达到破坏C语言程序的正确性的目的。比如，在连续的几条C语言语句中间加入一条修改AX寄存器（或DS等其他寄存器）的汇编指令语句，而AX的内容在此处本不该被修改，这样就可观察到破坏C语言程序正确性的效果（该项实验表明：在C语言程序中，若不考虑上下语句翻译成怎样的机器码而随意嵌入汇编指令语句时，有可能存在出错的风险）。
6. 观察C编译器的优化策略对代码的影响。通过实际观察与分析，记录本实验中汇编语言程序的效率会优于C语言程序的实例（至少给出一处的观察结果）。
7. 通过调试混合编程的程序，体会与纯粹汇编语言编写的程序的调试过程的差异。
8. 通过本次实验，希望大家明白：不同的编程语言是可以协同解决一个问题的，而且可以利用不同语言的特点来更好地解决问题；利用汇编语言的知识，能够更好地理解高级语言的内部处理原理与策略，为编写更好的C语言程序、用好C编译器提供支持。

# 3 实验过程

### 3.1任务一：宏与子程序设计

### 3.1.1设计思想及存储单元分配

将商店信息，账户信息，商品信息，利润率，提示信息以及临时需要存储的变量等放入数据段中，在进行登录，查询，计算，排序，修改等操作时将信息调入寄存器中进行相应的操作。

1.宏定义

M9：9号功能调用

M2: 2号功能调用

M10： 10号功能调用

NEWLINE： 换行

2.存储单元分配

语句提示：

INPUT\_NAME\_MSG DB 0AH,0DH,'Please input your name:$'

INPUT\_PASSWORD\_MSG DB 0AH,0DH,'Please input your passwords:$'

LOGIN\_FAIL\_MSG DB 0AH,0DH,'Login failed,Try Again:$'

GOODNAME\_MSG DB 0AH,0DH,'Please Input The Good Name:$'

GOODNAME\_MSG2 DB 0AH,0DH,'There is no this good in shop1.Please input again:$'

END\_MSG DB 0AH,0DH,'Press anykey to exit:$'

SHOWGOODNAME DB 0AH,0DH,'GoodName:$'

BUYPRICE DB 0AH,0DH,'BuyPrice:$'

PRICE DB 0AH,0DH,'Price:$'

TOTAL DB 0AH,0DH,'Total:$'

SOLD DB 0AH,0DH,'Sold:$'

PROFITRATE DB 0AH,0DH,'ProfitRate:$'

PEN DB 0AH,0DH,'PEN$'

BOOK DB 0AH,0DH,'BOOK$'

SHOP1 DB 0AH,0DH,'SHOP1$'

SHOP2 DB 0AH,0DH,'SHOP2$'

BUYMASSGE DB 0AH,0DH,'Please Input The Number You Want To Buy:$'

SOLDMASSGE DB 0AH,0DH,'OVERSOLD$'

JHJ DB 'BUYING PRICE:$'

XSJ DB 'SALES PRICE:$'

JHL DB 'RETAIL PURCHASES:$'

XSL DB 'SALES VOLUME:$'

SHO DB 'Please enter the shop you choose:$'

FIN DB 'Please enter what you want:$'

商店相关信息：

S1 DB 'SHOP1',0

GA1 DB 'PEN',7 DUP(0)

DW 35,56,22,25,?

GA2 DB 'BOOK',6 DUP(0)

DW 12,30,33,5,?

GA3 DB 'BAG',7 DUP(0)

DW 35,56,41,25,?

GA4 DB 'SI',8 DUP(0)

DW 35,56,42,25,?

GA5 DB 'WU',8 DUP(0)

DW 35,56,43,25,?

GA6 DB 'LIU',7 DUP(0)

DW 35,56,44,25,?

GA7 DB 'QI',8 DUP(0)

DW 35,56,45,25,?

GA8 DB 'BA',8 DUP(0)

DW 35,56,46,25,?

GA9 DB 'JIU',7 DUP(0)

DW 35,56,47,25,?

GA10 DB 'SHI',7 DUP(0)

DW 35,56,48,25,?

GA11 DB 'SHIYI',5 DUP(0)

DW 35,56,49,25,?

GA12 DB 'SHIER',5 DUP(0)

DW 35,56,50,25,?

GA13 DB 'SHISAN',4 DUP(0)

DW 35,56,51,25,?

GA14 DB 'SHISI',5 DUP(0)

DW 35,56,52,25,?

GA15 DB 'SHIWU',5 DUP(0)

DW 35,56,53,25,?

GA16 DB 'SHILIU',4 DUP(0)

DW 35,56,54,25,?

GA17 DB 'SHIQI',5 DUP(0)

DW 35,56,55,25,?

GA18 DB 'SHIBA',5 DUP(0)

DW 35,56,56,25,?

GA19 DB 'SHIJIU',4 DUP(0)

DW 35,56,57,25,?

GA20 DB 'ERSHI',5 DUP(0)

DW 35,56,58,25,?

GA21 DB 'ERYI',6 DUP(0)

DW 35,56,59,25,?

GA22 DB 'ERER',6 DUP(0)

DW 35,56,60,25,?

GA23 DB 'ERSAN',5 DUP(0)

DW 35,56,61,25,?

GA24 DB 'ERSI',6 DUP(0)

DW 35,56,62,25,?

GA25 DB 'ERWU',6 DUP(0)

DW 35,56,63,25,?

GA26 DB 'ERLIU',5 DUP(0)

DW 35,56,64,25,?

GA27 DB 'ERQI',6 DUP(0)

DW 35,56,65,25,?

GA28 DB 'ERBA',6 DUP(0)

DW 35,56,66,25,?

GA29 DB 'ERJIU',5 DUP(0)

DW 35,56,67,25,?

GA30 DB 'SANLING',3 DUP(0)

DW 35,56,68,25,?

S2 DB 'SHOP2', 0

GB1 DB 'PEN', 7 DUP(0)

DW 35, 50, 26, 24, ?

GB2 DB 'BOOK', 6 DUP(0)

DW 12, 28, 27, 15, ?

GB3 DB 'BAG',7 DUP(0)

DW 35,50,28,24,?

GB4 DB 'SI',8 DUP(0)

DW 35,56,29,25,?

GB5 DB 'WU',8 DUP(0)

DW 35,56,30,25,?

GB6 DB 'LIU',7 DUP(0)

DW 35,56,10000,25,?

GB7 DB 'QI',8 DUP(0)

DW 35,56,31,25,?

GB8 DB 'BA',8 DUP(0)

DW 35,56,32,25,?

GB9 DB 'JIU',7 DUP(0)

DW 35,56,33,25,?

GB10 DB 'SHI',7 DUP(0)

DW 35,56,34,25,?

GB11 DB 'SHIYI',5 DUP(0)

DW 35,56,35,25,?

GB12 DB 'SHIER',5 DUP(0)

DW 35,56,36,25,?

GB13 DB 'SHISAN',4 DUP(0)

DW 35,56,37,25,?

GB14 DB 'SHISI',5 DUP(0)

DW 35,56,38,25,?

GB15 DB 'SHIWU',5 DUP(0)

DW 35,56,39,25,?

GB16 DB 'SHILIU',4 DUP(0)

DW 35,56,40,25,?

GB17 DB 'SHIQI',5 DUP(0)

DW 35,56,41,25,?

GB18 DB 'SHIBA',5 DUP(0)

DW 35,56,42,25,?

GB19 DB 'SHIJIU',4 DUP(0)

DW 35,56,43,25,?

GB20 DB 'ERSHI',5 DUP(0)

DW 35,56,44,25,?

GB21 DB 'ERYI',6 DUP(0)

DW 35,56,45,25,?

GB22 DB 'ERER',6 DUP(0)

DW 35,56,46,25,?

GB23 DB 'ERSAN',5 DUP(0)

DW 35,56,47,25,?

GB24 DB 'ERSI',6 DUP(0)

DW 35,56,48,25,?

GB25 DB 'ERWU',6 DUP(0)

DW 35,56,49,25,?

GB26 DB 'ERLIU',5 DUP(0)

DW 35,56,50,25,?

GB27 DB 'ERQI',6 DUP(0)

DW 35,56,51,25,?

GB28 DB 'ERBA',6 DUP(0)

DW 35,56,52,25,?

GB29 DB 'ERJIU',5 DUP(0)

DW 35,56,53,25,?

GB30 DB 'SANLING',3 DUP(0)

DW 35,56,54,25,?

BNAME DB 'zhang',5 DUP(0);老板姓名

BPASSWORD DB '49056',5 DUP(0);老板密码

COUNTN EQU 8

COUNTP EQU 4

N EQU 30

临时存储：

BUFF DB 12 DUP(?)

AUTH DB 0

COST1 DD 0

COST2 DD 0

PROFIT1 DD 0

PROFIT2 DD 0

PR1 DD 0

PR2 DD 0

APR DD 0

APRN DW 30 DUP(0)

ADR DW 30 DUP(0)

DIP DB 6,0,6 DUP(0)

OUTNUM DW 10 DUP(0)

JGF DB '>'

SN DB 6,0,6 DUP(0)

GN DB 10,0,10 DUP(0)

REALNUM DW 0

INPUT DW 0

JISHU DW 0

CHS DW 10

LOC DW 0

LOCB DW 0

LOCR DW 0

COUNT DW 0

M DW 9000

M1 DW 0

IP1 DW 0

IP2 DW 0

INNAME DB 10

DB 0

DB 10 DUP(0)

INPASSWORD DB 10

DB 10

DB 10 DUP(0)

GETGOODNAME DB 10

DB 0

DB 10 DUP(0)

NUMBER DW 10

DW 0

DW 10 DUP(0)

BUF DW 0

BUF1 DW 0

3.寄存器分配

CX：计数器。

BX：记录偏移地址。

AX、DX、SI、DI、EAX、EBX、ECX、EDX：临时寄存器。

### 3.1.2流程图

（仅包含本人编写的功能三，四，五的流程图）

图3.1.2.1是功能三商品信息修改的程序流程图。

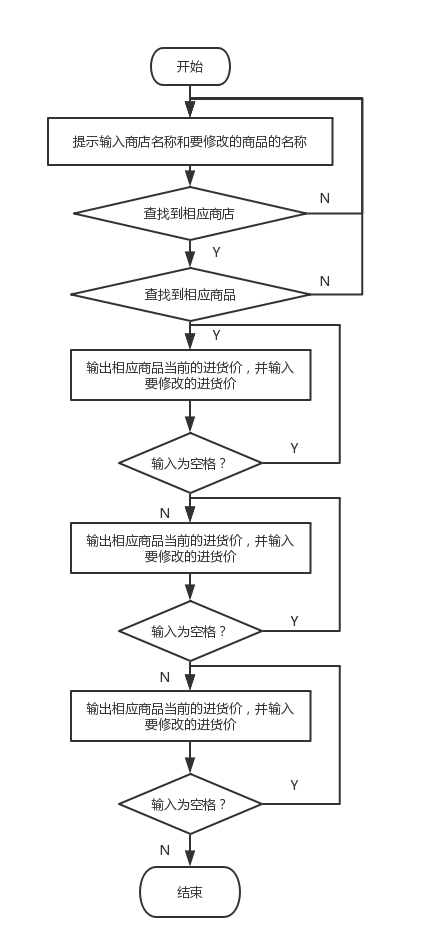


图3.1.2.1 功能三商品信息修改流程图

图3.1.2.2是功能四商品平均利润率的程序流程图。

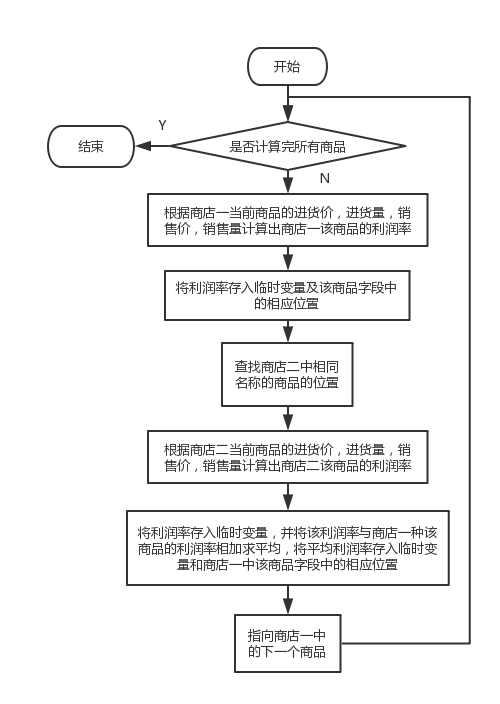


图3.1.2.2 功能四商品平均利润率流程图

图3.1.2.3是功能五商品利润率排名的程序流程图。

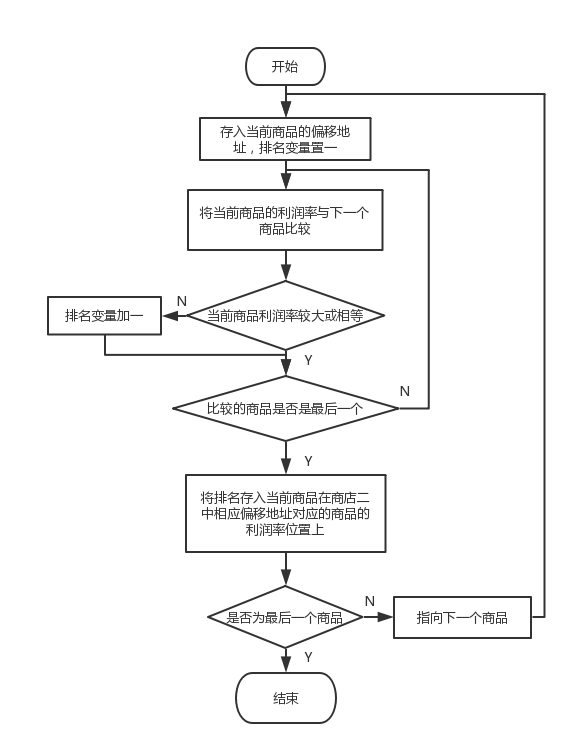


图3.1.2.3 功能五商品利润率排名流程图

### 3.1.3源程序

;编写者：张佳兴，同组：刘子煜

;菜单，功能一，功能五

EXTRN FUN2:FAR,FUN3:FAR,FUN4:FAR

PUBLIC RDAIX,PRINTAX

.386

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;宏定义

M9 MACRO A

LEA DX,A

MOV AH,9

INT 21H

ENDM ;9号功能调用

M2 MACRO A

MOV DL,A

MOV AH,2

INT 21H

ENDM ;2号功能调用

M10 MACRO A

LEA DX,A

MOV AH,10

INT 21H

ENDM ;10号功能调用

NEWLINE MACRO

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

ENDM ;换行

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DATAS SEGMENT USE16

INPUT\_NAME\_MSG DB 0AH,0DH,'Please input your name:$'

INPUT\_PASSWORD\_MSG DB 0AH,0DH,'Please input your passwords:$'

LOGIN\_FAIL\_MSG DB 0AH,0DH,'Login failed,Try Again:$'

GOODNAME\_MSG DB 0AH,0DH,'Please Input The Good Name:$'

GOODNAME\_MSG2 DB 0AH,0DH,'There is no this good in shop1.Please input again:$'

END\_MSG DB 0AH,0DH,'Press anykey to exit:$'

SHOWGOODNAME DB 0AH,0DH,'GoodName:$'

BUYPRICE DB 0AH,0DH,'BuyPrice:$'

PRICE DB 0AH,0DH,'Price:$'

TOTAL DB 0AH,0DH,'Total:$'

SOLD DB 0AH,0DH,'Sold:$'

PROFITRATE DB 0AH,0DH,'ProfitRate:$'

PEN DB 0AH,0DH,'PEN$'

BOOK DB 0AH,0DH,'BOOK$'

SHOP1 DB 0AH,0DH,'SHOP1$'

SHOP2 DB 0AH,0DH,'SHOP2$'

BUYMASSGE DB 0AH,0DH,'Please Input The Number You Want To Buy:$'

SOLDMASSGE DB 0AH,0DH,'OVERSOLD$'

JHJ DB 'BUYING PRICE:$'

XSJ DB 'SALES PRICE:$'

JHL DB 'RETAIL PURCHASES:$'

XSL DB 'SALES VOLUME:$'

SHO DB 'Please enter the shop you choose:$'

FIN DB 'Please enter what you want:$'

S1 DB 'SHOP1',0

GA1 DB 'PEN',7 DUP(0)

DW 35,56,22,25,?

GA2 DB 'BOOK',6 DUP(0)

DW 12,30,33,5,?

GA3 DB 'BAG',7 DUP(0)

DW 35,56,41,25,?

GA4 DB 'SI',8 DUP(0)

DW 35,56,42,25,?

GA5 DB 'WU',8 DUP(0)

DW 35,56,43,25,?

GA6 DB 'LIU',7 DUP(0)

DW 35,56,44,25,?

GA7 DB 'QI',8 DUP(0)

DW 35,56,45,25,?

GA8 DB 'BA',8 DUP(0)

DW 35,56,46,25,?

GA9 DB 'JIU',7 DUP(0)

DW 35,56,47,25,?

GA10 DB 'SHI',7 DUP(0)

DW 35,56,48,25,?

GA11 DB 'SHIYI',5 DUP(0)

DW 35,56,49,25,?

GA12 DB 'SHIER',5 DUP(0)

DW 35,56,50,25,?

GA13 DB 'SHISAN',4 DUP(0)

DW 35,56,51,25,?

GA14 DB 'SHISI',5 DUP(0)

DW 35,56,52,25,?

GA15 DB 'SHIWU',5 DUP(0)

DW 35,56,53,25,?

GA16 DB 'SHILIU',4 DUP(0)

DW 35,56,54,25,?

GA17 DB 'SHIQI',5 DUP(0)

DW 35,56,55,25,?

GA18 DB 'SHIBA',5 DUP(0)

DW 35,56,56,25,?

GA19 DB 'SHIJIU',4 DUP(0)

DW 35,56,57,25,?

GA20 DB 'ERSHI',5 DUP(0)

DW 35,56,58,25,?

GA21 DB 'ERYI',6 DUP(0)

DW 35,56,59,25,?

GA22 DB 'ERER',6 DUP(0)

DW 35,56,60,25,?

GA23 DB 'ERSAN',5 DUP(0)

DW 35,56,61,25,?

GA24 DB 'ERSI',6 DUP(0)

DW 35,56,62,25,?

GA25 DB 'ERWU',6 DUP(0)

DW 35,56,63,25,?

GA26 DB 'ERLIU',5 DUP(0)

DW 35,56,64,25,?

GA27 DB 'ERQI',6 DUP(0)

DW 35,56,65,25,?

GA28 DB 'ERBA',6 DUP(0)

DW 35,56,66,25,?

GA29 DB 'ERJIU',5 DUP(0)

DW 35,56,67,25,?

GA30 DB 'SANLING',3 DUP(0)

DW 35,56,68,25,?

S2 DB 'SHOP2', 0

GB1 DB 'PEN', 7 DUP(0)

DW 35, 50, 26, 24, ?

GB2 DB 'BOOK', 6 DUP(0)

DW 12, 28, 27, 15, ?

GB3 DB 'BAG',7 DUP(0)

DW 35,50,28,24,?

GB4 DB 'SI',8 DUP(0)

DW 35,56,29,25,?

GB5 DB 'WU',8 DUP(0)

DW 35,56,30,25,?

GB6 DB 'LIU',7 DUP(0)

DW 35,56,10000,25,?

GB7 DB 'QI',8 DUP(0)

DW 35,56,31,25,?

GB8 DB 'BA',8 DUP(0)

DW 35,56,32,25,?

GB9 DB 'JIU',7 DUP(0)

DW 35,56,33,25,?

GB10 DB 'SHI',7 DUP(0)

DW 35,56,34,25,?

GB11 DB 'SHIYI',5 DUP(0)

DW 35,56,35,25,?

GB12 DB 'SHIER',5 DUP(0)

DW 35,56,36,25,?

GB13 DB 'SHISAN',4 DUP(0)

DW 35,56,37,25,?

GB14 DB 'SHISI',5 DUP(0)

DW 35,56,38,25,?

GB15 DB 'SHIWU',5 DUP(0)

DW 35,56,39,25,?

GB16 DB 'SHILIU',4 DUP(0)

DW 35,56,40,25,?

GB17 DB 'SHIQI',5 DUP(0)

DW 35,56,41,25,?

GB18 DB 'SHIBA',5 DUP(0)

DW 35,56,42,25,?

GB19 DB 'SHIJIU',4 DUP(0)

DW 35,56,43,25,?

GB20 DB 'ERSHI',5 DUP(0)

DW 35,56,44,25,?

GB21 DB 'ERYI',6 DUP(0)

DW 35,56,45,25,?

GB22 DB 'ERER',6 DUP(0)

DW 35,56,46,25,?

GB23 DB 'ERSAN',5 DUP(0)

DW 35,56,47,25,?

GB24 DB 'ERSI',6 DUP(0)

DW 35,56,48,25,?

GB25 DB 'ERWU',6 DUP(0)

DW 35,56,49,25,?

GB26 DB 'ERLIU',5 DUP(0)

DW 35,56,50,25,?

GB27 DB 'ERQI',6 DUP(0)

DW 35,56,51,25,?

GB28 DB 'ERBA',6 DUP(0)

DW 35,56,52,25,?

GB29 DB 'ERJIU',5 DUP(0)

DW 35,56,53,25,?

GB30 DB 'SANLING',3 DUP(0)

DW 35,56,54,25,?

COUNTN EQU 8

COUNTP EQU 4

N EQU 30

BUFF DB 12 DUP(?)

AUTH DB 0

COST1 DD 0

COST2 DD 0

PROFIT1 DD 0

PROFIT2 DD 0

PR1 DD 0

PR2 DD 0

APR DD 0

APRN DW 30 DUP(0)

ADR DW 30 DUP(0)

DIP DB 6,0,6 DUP(0)

OUTNUM DW 10 DUP(0)

JGF DB '>'

SN DB 6,0,6 DUP(0)

GN DB 10,0,10 DUP(0)

REALNUM DW 0

INPUT DW 0

JISHU DW 0

CHS DW 10

LOC DW 0

LOCB DW 0

LOCR DW 0

COUNT DW 0

M DW 9000

M1 DW 0

IP1 DW 0

IP2 DW 0

INNAME DB 10

DB 0

DB 10 DUP(0)

INPASSWORD DB 10

DB 10

DB 10 DUP(0)

GETGOODNAME DB 10

DB 0

DB 10 DUP(0)

NUMBER DW 10

DW 0

DW 10 DUP(0)

BNAME DB 'zhang',5 DUP(0);老板姓名

BPASSWORD DB '49056',5 DUP(0);老板密码

BUF DW 0

BUF1 DW 0

CAIDAN1 DB 0AH,0DH,'1=SEARCH PRODUCT INFORMATION$'

CAIDAN2 DB 0AH,0DH,'2=MODIFY PRODUCT INFORMATION$'

CAIDAN3 DB 0AH,0DH,'3=CALCULATE AVERAGE PROFIT$'

CAIDAN4 DB 0AH,0DH,'4=CALCULATE PROFIT RATE RANKING$'

CAIDAN5 DB 0AH,0DH,'5=EXPORT ALL PRODUCT INFORMATION$'

CAIDAN6 DB 0AH,0DH,'6=PROGRAME EXIT$'

CAIDAN7 DB 0AH,0DH,'You can just input between 1 to 6!$'

CAIDAN8 DB 0AH,0DH,'Please input your choice:$'

CAIDAN9 DB 0AH,0DH,'You can just input between 1 or 6!$'

DATAS ENDS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

STACKS SEGMENT USE16 STACK 'STACKS'

DB 200 DUP(0)

STACKS ENDS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CODES SEGMENT USE16 PUBLIC 'CODES'

ASSUME CS:CODES,DS:DATAS,SS:STACKS

START:

MOV AX,DATAS

MOV DS,AX

FUNCTION1:;输入用户名和密码

MOV SI,2;计数器

MOV CX,10;计数器

LEA DX,INPUT\_NAME\_MSG;输入姓名提示

MOV AH,9

INT 21H

LEA DX,INNAME;输入姓名

MOV AH,10

INT 21H

LEA DX,INPUT\_PASSWORD\_MSG;输入密码提示

MOV AH,9

INT 21H

LEA DX,INPASSWORD;输入密码

MOV AH,10

INT 21H

MOV AL,0DH;未输入用户名时跳转至用户模式

CMP AL,INNAME[SI]

JE FUNCTION1\_1

MOV AL,71H;用户名为q时退出

CMP AL,INNAME[SI]

JE Q1

MOV SI,0

MOV CL,INNAME[SI+1]

JMP FUNCTION2

Q1: MOV AL,0DH;用户名为q时退出

CMP AL,INNAME[SI+1]

JE EXIT

FUNCTION2:;用户名和密码检查

MOV AL,INNAME[SI+2]

MOV BL,BNAME[SI]

CMP AL,BL

JNE FUNCTION1;用户名与老板用户名不符时返回

MOV AL,INPASSWORD[SI+2]

CMP AL,BPASSWORD[SI]

JNE START;密码与老板密码不符时返回

INC SI

DEC CL

JNZ FUNCTION2

MOV AUTH,1

JMP BEG

FUNCTION1\_1:

MOV BH,0

MOV BYTE PTR AUTH, BH

JMP BEG0

;;;;;;;;;

BEG0: M9 CAIDAN1

M9 CAIDAN6

M9 CAIDAN8

MOV AH,1

INT 21H

CMP AL,'1'

JE L1

CMP AL,'6'

JE EXIt

M9 CAIDAN9

JMP BEG0

;;;;;;;;;;;;;;

BEG: M9 CAIDAN1

M9 CAIDAN2

M9 CAIDAN3

M9 CAIDAN4

M9 CAIDAN5

M9 CAIDAN6

M9 CAIDAN8

MOV AH,1

INT 21H

CMP AL,'1'

JE L1

CMP AL,'2'

JE L2

CMP AL,'3'

JE L3

CMP AL,'4'

JE L4

CMP AL,'5'

JE L5

CMP AL,'6'

JE EXIt

M9 CAIDAN7

JMP BEG

L1: CALL FUN1

JMP BEG

L2: CALL FUN2

JMP BEG

L3: CALL FUN3

JMP BEG

L4: CALL FUN4

JMP BEG

L5: CALL FUN5

JMP BEG

;;;;;;;;;;;;;;;;;;;;;;;;;;;

FUN1 PROC FAR

FUNCTION3:

M9 GOODNAME\_MSG

M10 GETGOODNAME

MOV SI,2;输入为回车时返回

MOV AL,0DH

MOV BL,GETGOODNAME[SI]

CMP AL,BL

JE BEG

MOV DX,0

MOV SI,0

MOV CL,GETGOODNAME[SI+1]

FUNCTION3\_0\_1:

MOV DI,0

IMUL SI,DX,20

CMP SI,580

JE FUNCTION3

INC DX

FUNCTION3\_1:

MOV AL,GETGOODNAME[DI+2];输入名称与商品名称比较

MOV BL,GA1[SI]

CMP AL,BL

JNE FUNCTION3\_0\_1;不符合跳转到该商店的下一商品

INC SI

INC DI

DEC CL

JNZ FUNCTION3\_1

MOV CL,GETGOODNAME[1]

DEC CL

MOV CH,0

SUB SI,CX

JMP INFO;符合跳转到该商品的信息

INFO:

DEC DX

MOV JISHU,DX

IMUL SI,DX,20

M9 SHOP1

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHJ

MOV AX,WORD PTR GA1[SI+10]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSJ

MOV AX,WORD PTR GA1[SI+12]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHL

MOV AX,WORD PTR GA1[SI+14]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSL

MOV AX,WORD PTR GA1[SI+16]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 SHOP2

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHJ

MOV AX,WORD PTR GB1[SI+10]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSJ

MOV AX,WORD PTR GB1[SI+12]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHL

MOV AX,WORD PTR GB1[SI+14]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSL

MOV AX,WORD PTR GB1[SI+16]

MOV DX,16

CALL PRINTAX

RET

FUN1 ENDP

;;;;;;;;;;;;;;;;;;;;;;;;

PRINTAX PROC FAR

PUSH EBX

PUSH SI

LEA SI,BUFF

CMP DX,32

JE L

MOVSX EAX,AX

L: OR EAX,EAX

JNS PLUS

NEG EAX

MOV BYTE PTR [SI],'-'

INC SI

PLUS: MOV EBX,10

CALL RDAIX

MOV BYTE PTR [SI],'$'

LEA DX,BUFF

MOV AH,9

INT 21H

POP SI

POP EBX

RET

PRINTAX ENDP

RDAIX PROC FAR

PUSH CX

PUSH EDX

XOR CX,CX

LOP1: XOR EDX,EDX

DIV EBX

PUSH DX

INC CX

OR EAX,EAX

JNZ LOP1

LOP2: POP AX

CMP AL,10

JB LL1

ADD AL,7

LL1: ADD AL,30H

MOV [SI],AL

INC SI

LOOP LOP2

POP EDX

POP CX

RET

RDAIX ENDP

FUN5 PROC FAR

F1:

MOV DI,0

LL:

IMUL SI,DI,20

MOV BYTE PTR GA1[SI+9],'$'

MOV BYTE PTR GB1[SI+9],'$'

M9 GA1[SI]

MOV DL,32

MOV AH,2

INT 21H

MOV AX,WORD PTR GA1[SI+18]

MOV DX,16

CALL PRINTAX

MOV DL,32

MOV AH,2

INT 21H

MOV AX,WORD PTR GB1[SI+18]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

INC DI

CMP DI,30

JNE LL

RET

FUN5 ENDP

EXIT :

MOV AH,4CH

INT 21H

CODES ENDS

END START

;编写者：刘子煜，同组：张佳兴

;功能二，功能三，功能四

EXTRN RDAIX:FAR,PRINTAX:FAR

PUBLIC FUN2,FUN3,FUN4

.386

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;宏定义

M9 MACRO A

LEA DX,A

MOV AH,9

INT 21H

ENDM ;9号功能调用

M2 MACRO A

MOV DL,A

MOV AH,2

INT 21H

ENDM ;2号功能调用

M10 MACRO A

LEA DX,A

MOV AH,10

INT 21H

ENDM ;10号功能调用

NEWLINE MACRO

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

ENDM ;换行

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DATAS SEGMENT USE16

INPUT\_NAME\_MSG DB 0AH,0DH,'Please input your name:$'

INPUT\_PASSWORD\_MSG DB 0AH,0DH,'Please input your passwords:$'

LOGIN\_FAIL\_MSG DB 0AH,0DH,'Login failed,Try Again:$'

GOODNAME\_MSG DB 0AH,0DH,'Please Input The Good Name:$'

GOODNAME\_MSG2 DB 0AH,0DH,'There is no this good in shop1.Please input again:$'

END\_MSG DB 0AH,0DH,'Press anykey to exit:$'

SHOWGOODNAME DB 0AH,0DH,'GoodName:$'

BUYPRICE DB 0AH,0DH,'BuyPrice:$'

PRICE DB 0AH,0DH,'Price:$'

TOTAL DB 0AH,0DH,'Total:$'

SOLD DB 0AH,0DH,'Sold:$'

PROFITRATE DB 0AH,0DH,'ProfitRate:$'

PEN DB 0AH,0DH,'PEN$'

BOOK DB 0AH,0DH,'BOOK$'

SHOP1 DB 0AH,0DH,'SHOP1$'

SHOP2 DB 0AH,0DH,'SHOP2$'

BUYMASSGE DB 0AH,0DH,'Please Input The Number You Want To Buy:$'

SOLDMASSGE DB 0AH,0DH,'OVERSOLD$'

JHJ DB 'BUYING PRICE:$'

XSJ DB 'SALES PRICE:$'

JHL DB 'RETAIL PURCHASES:$'

XSL DB 'SALES VOLUME:$'

SHO DB 'Please enter the shop you choose:$'

FIN DB 'Please enter what you want:$'

S1 DB 'SHOP1',0

GA1 DB 'PEN',7 DUP(0)

DW 35,56,22,25,?

GA2 DB 'BOOK',6 DUP(0)

DW 12,30,33,5,?

GA3 DB 'BAG',7 DUP(0)

DW 35,56,41,25,?

GA4 DB 'SI',8 DUP(0)

DW 35,56,42,25,?

GA5 DB 'WU',8 DUP(0)

DW 35,56,43,25,?

GA6 DB 'LIU',7 DUP(0)

DW 35,56,44,25,?

GA7 DB 'QI',8 DUP(0)

DW 35,56,45,25,?

GA8 DB 'BA',8 DUP(0)

DW 35,56,46,25,?

GA9 DB 'JIU',7 DUP(0)

DW 35,56,47,25,?

GA10 DB 'SHI',7 DUP(0)

DW 35,56,48,25,?

GA11 DB 'SHIYI',5 DUP(0)

DW 35,56,49,25,?

GA12 DB 'SHIER',5 DUP(0)

DW 35,56,50,25,?

GA13 DB 'SHISAN',4 DUP(0)

DW 35,56,51,25,?

GA14 DB 'SHISI',5 DUP(0)

DW 35,56,52,25,?

GA15 DB 'SHIWU',5 DUP(0)

DW 35,56,53,25,?

GA16 DB 'SHILIU',4 DUP(0)

DW 35,56,54,25,?

GA17 DB 'SHIQI',5 DUP(0)

DW 35,56,55,25,?

GA18 DB 'SHIBA',5 DUP(0)

DW 35,56,56,25,?

GA19 DB 'SHIJIU',4 DUP(0)

DW 35,56,57,25,?

GA20 DB 'ERSHI',5 DUP(0)

DW 35,56,58,25,?

GA21 DB 'ERYI',6 DUP(0)

DW 35,56,59,25,?

GA22 DB 'ERER',6 DUP(0)

DW 35,56,60,25,?

GA23 DB 'ERSAN',5 DUP(0)

DW 35,56,61,25,?

GA24 DB 'ERSI',6 DUP(0)

DW 35,56,62,25,?

GA25 DB 'ERWU',6 DUP(0)

DW 35,56,63,25,?

GA26 DB 'ERLIU',5 DUP(0)

DW 35,56,64,25,?

GA27 DB 'ERQI',6 DUP(0)

DW 35,56,65,25,?

GA28 DB 'ERBA',6 DUP(0)

DW 35,56,66,25,?

GA29 DB 'ERJIU',5 DUP(0)

DW 35,56,67,25,?

GA30 DB 'SANLING',3 DUP(0)

DW 35,56,68,25,?

S2 DB 'SHOP2', 0

GB1 DB 'PEN', 7 DUP(0)

DW 35, 50, 26, 24, ?

GB2 DB 'BOOK', 6 DUP(0)

DW 12, 28, 27, 15, ?

GB3 DB 'BAG',7 DUP(0)

DW 35,50,28,24,?

GB4 DB 'SI',8 DUP(0)

DW 35,56,29,25,?

GB5 DB 'WU',8 DUP(0)

DW 35,56,30,25,?

GB6 DB 'LIU',7 DUP(0)

DW 35,56,10000,25,?

GB7 DB 'QI',8 DUP(0)

DW 35,56,31,25,?

GB8 DB 'BA',8 DUP(0)

DW 35,56,32,25,?

GB9 DB 'JIU',7 DUP(0)

DW 35,56,33,25,?

GB10 DB 'SHI',7 DUP(0)

DW 35,56,34,25,?

GB11 DB 'SHIYI',5 DUP(0)

DW 35,56,35,25,?

GB12 DB 'SHIER',5 DUP(0)

DW 35,56,36,25,?

GB13 DB 'SHISAN',4 DUP(0)

DW 35,56,37,25,?

GB14 DB 'SHISI',5 DUP(0)

DW 35,56,38,25,?

GB15 DB 'SHIWU',5 DUP(0)

DW 35,56,39,25,?

GB16 DB 'SHILIU',4 DUP(0)

DW 35,56,40,25,?

GB17 DB 'SHIQI',5 DUP(0)

DW 35,56,41,25,?

GB18 DB 'SHIBA',5 DUP(0)

DW 35,56,42,25,?

GB19 DB 'SHIJIU',4 DUP(0)

DW 35,56,43,25,?

GB20 DB 'ERSHI',5 DUP(0)

DW 35,56,44,25,?

GB21 DB 'ERYI',6 DUP(0)

DW 35,56,45,25,?

GB22 DB 'ERER',6 DUP(0)

DW 35,56,46,25,?

GB23 DB 'ERSAN',5 DUP(0)

DW 35,56,47,25,?

GB24 DB 'ERSI',6 DUP(0)

DW 35,56,48,25,?

GB25 DB 'ERWU',6 DUP(0)

DW 35,56,49,25,?

GB26 DB 'ERLIU',5 DUP(0)

DW 35,56,50,25,?

GB27 DB 'ERQI',6 DUP(0)

DW 35,56,51,25,?

GB28 DB 'ERBA',6 DUP(0)

DW 35,56,52,25,?

GB29 DB 'ERJIU',5 DUP(0)

DW 35,56,53,25,?

GB30 DB 'SANLING',3 DUP(0)

DW 35,56,54,25,?

COUNTN EQU 8

COUNTP EQU 4

N EQU 30

BUFF DB 12 DUP(?)

AUTH DB 0

COST1 DD 0

COST2 DD 0

PROFIT1 DD 0

PROFIT2 DD 0

PR1 DD 0

PR2 DD 0

APR DD 0

APRN DW 30 DUP(0)

ADR DW 30 DUP(0)

DIP DB 6,0,6 DUP(0)

OUTNUM DW 10 DUP(0)

JGF DB '>'

SN DB 6,0,6 DUP(0)

GN DB 10,0,10 DUP(0)

REALNUM DW 0

INPUT DW 0

JISHU DW 0

CHS DW 10

LOC DW 0

LOCB DW 0

LOCR DW 0

COUNT DW 0

M DW 9000

M1 DW 0

RANK DW 0

IP1 DW 0

IP2 DW 0

INNAME DB 10

DB 0

DB 10 DUP(0)

INPASSWORD DB 10

DB 10

DB 10 DUP(0)

GETGOODNAME DB 10

DB 0

DB 10 DUP(0)

NUMBER DW 10

DW 0

DW 10 DUP(0)

BNAME DB 'zhang',5 DUP(0);老板姓名

BPASSWORD DB '49056',5 DUP(0);老板密码

BUF DW 0

BUF1 DW 0

CAIDAN1 DB 0AH,0DH,'1=SEARCH PRODUCT INFORMATION$'

CAIDAN2 DB 0AH,0DH,'2=MODIFY PRODUCT INFORMATION$'

CAIDAN3 DB 0AH,0DH,'3=CALCULATE AVERAGE PROFIT$'

CAIDAN4 DB 0AH,0DH,'4=CALCULATE PROFIT RATE RANKING$'

CAIDAN5 DB 0AH,0DH,'5=EXPORT ALL PRODUCT INFORMATION$'

CAIDAN6 DB 0AH,0DH,'6=PROGRAME EXIT$'

CAIDAN7 DB 0AH,0DH,'You can just input between 1 to 6!$'

CAIDAN8 DB 0AH,0DH,'Please input your choice:$'

CAIDAN9 DB 0AH,0DH,'You can just input between 1 or 6!$'

DATAS ENDS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

STACKS SEGMENT USE16 STACK 'STACKS'

DB 200 DUP(0)

STACKS ENDS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CODES SEGMENT USE16 PUBLIC 'CODES'

ASSUME CS:CODES,DS:DATAS,SS:STACKS

START:

MOV AX,DATAS

MOV DS,AX

FUN2 PROC FAR

FST2: LEA DX, SHO

MOV AH, 9

INT 21H

LEA DX, SN

MOV AH, 10

INT 21H

NEWLINE

LEA DX, FIN

MOV AH, 9

INT 21H

LEA DX, GN

MOV AH, 10

INT 21H

NEWLINE

CMP SN+2, 13

JE BACK2

MOV BX, 0

MOV CX, 0

MOV CL, [SN+1]

SCOMP: LEA SI, SN+2

LEA DI, [S1+BX]

CSC: MOV AL, [SI]

CMP AL, [DI]

JNE SALL

INC SI

INC DI

DEC CX

JE SCPL

JMP CSC

GNL: MOV CX, 0

MOV CL, [GN+1]

ADD BX, 6

GCOMP: LEA SI, GN+2

LEA DI, [S1+BX]

CGC: MOV AL, [SI]

CMP AL, [DI]

JNE GALL

INC SI

INC DI

DEC CX

JE GCPL

JMP CGC

MJJ: MOV LOC, BX

LEA DX, JHJ

MOV AH, 9

INT 21H

MOV AX, WORD PTR [S1+BX+10]

MOV DX, 16

CALL PRINTAX

PJG: MOV DL, [JGF]

MOV AH, 2

INT 21H

LEA DX, DIP

MOV AH, 10

INT 21H

NEWLINE

MOV BX, 0

LEA DI, DIP+2

CTS: MOV AX, 0

MOV CX, 0

MOV CL, [DIP+1]

DEC CX

SUB CX, BX

MOV AL, [DI+BX]

CMP AL, '0'

JL MJJ

CMP AL, '9'

JG MJJ

TTS: SUB AX, 30H

CMP CX, 0

JE TAD

IMUL AX, 10

DEC CX

JNE TTS

TAD: MOV DX, COUNT

ADD DX, AX

MOV COUNT, DX

INC BX

CMP BL, BYTE PTR [DIP+1]

JNE CTS

LJJ: MOV BX, LOC

MOV AX, COUNT

MOV WORD PTR [S1+BX+10], AX

MOV COUNT, 0

MXJ: MOV LOC, BX

LEA DX, XSJ

MOV AH, 9

INT 21H

MOV AX, WORD PTR [S1+BX+12]

MOV DX, 16

CALL PRINTAX

MOV DL, [JGF]

MOV AH, 2

INT 21H

LEA DX, DIP

MOV AH, 10

INT 21H

NEWLINE

MOV BX, 0

LEA DI, DIP+2

CTS2: MOV AX, 0

MOV CX, 0

MOV CL, [DIP+1]

DEC CX

SUB CX, BX

MOV AL, [DI+BX]

CMP AL, '0'

JL MXJ

CMP AL, '9'

JG MXJ

TTS2: SUB AX, 30H

CMP CX, 0

JE TAD2

IMUL AX, 10

DEC CX

JNE TTS2

TAD2: MOV DX, COUNT

ADD DX, AX

MOV COUNT, DX

INC BX

CMP BL, BYTE PTR [DIP+1]

JNE CTS2

LXJ: MOV BX, LOC

MOV AX, COUNT

MOV WORD PTR [S1+BX+12], AX

MOV COUNT, 0

MJL: MOV LOC, BX

LEA DX, JHL

MOV AH, 9

INT 21H

MOV AX, WORD PTR [S1+BX+14]

MOV DX, 16

CALL PRINTAX

MOV DL, [JGF]

MOV AH, 2

INT 21H

LEA DX, DIP

MOV AH, 10

INT 21H

NEWLINE

MOV BX, 0

LEA DI, DIP+2

CTS3: MOV AX, 0

MOV CX, 0

MOV CL, [DIP+1]

DEC CX

SUB CX, BX

MOV AL, [DI+BX]

CMP AL, '0'

JL MJL

CMP AL, '9'

JG MJL

TTS3: SUB AX, 30H

CMP CX, 0

JE TAD3

IMUL AX, 10

DEC CX

JNE TTS3

TAD3: MOV DX, COUNT

ADD DX, AX

MOV COUNT, DX

INC BX

CMP BL, BYTE PTR [DIP+1]

JNE CTS3

LJL: MOV BX, LOC

MOV AX, COUNT

MOV WORD PTR [S1+BX+14], AX

MOV COUNT, 0

JMP BACK2

SALL: CMP BX, 1210

JE FST2

ADD BX, 605

JMP SCOMP

SCPL: CMP BYTE PTR [DI], 0

JE GNL

JMP SALL

GALL: CMP BX, 600

JE FST2

ADD BX, 20

JMP GCOMP

GCPL: CMP BYTE PTR [DI], 0

JE MJJ

JMP GALL

BACK2: RET

FUN2 ENDP

FUN3 PROC FAR

MOV BX, 0

BAP: MOV EAX, 0

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [GA1+BX+10]

MOV CX, WORD PTR [GA1+BX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [GA1+BX+12]

MOV DX, WORD PTR [GA1+BX+16]

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR1, EAX

MOV LOC, BX

MOV CX, 0

MOV BX, LOC

LEA DI, [GA1+BX]

MOV BX, 0

LEA SI, [GB1+BX]

COMP\_3: MOV CL, 10

LEA SI, [GB1+BX]

MOV LOCB, BX

MOV BX, LOC

LEA DI, [GA1+BX]

MOV BX, LOCB

RCM: MOV AL, [DI]

CMP AL, [SI]

JNE ALL

INC DI

INC SI

DEC CX

JE CCOP

JMP RCM

CCOP: MOV EAX, 0

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [GB1+BX+10]

MOV CX, WORD PTR [GB1+BX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [GB1+BX+12]

MOV DX, WORD PTR [GB1+BX+16]

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR2, EAX

MOV EDX, PR1

ADD EAX, EDX

SAR EAX, 1

MOV APR, EAX

MOV BX, LOC

MOV WORD PTR [GA1+BX+18], AX

ADD BX, 20

CMP BX, 600

JE BACK3

JMP BAP

ALL: ADD BX,20

JMP COMP\_3

BACK3: RET

FUN3 ENDP

FUN4 PROC FAR

MOV CX,-1

LOOPX:

INC CX

IMUL BX,CX,20

MOV AX,WORD PTR GA1+18[BX]

PUSH CX

PUSH BX

MOV CX,-1

MOV RANK,1

LOOPY:

INC CX

IMUL BX,CX,20

CMP AX,WORD PTR GA1+18[BX]

JL LARANK

L3P:

CMP CX,N-1

JGE L352

JMP LOOPY

L352:

POP BX

POP CX

JMP L351

LARANK:

INC RANK

JMP L3P

L351:

MOV AX,RANK

MOV WORD PTR GB1+18[BX],AX

CMP CX,N-1

JGE LLL

JMP LOOPX

LLL: RET

FUN4 ENDP

EXIT :

MOV AH,4CH

INT 21H

CODES ENDS

END START

### 3.1.4实验步骤

1. 准备上机实验环境、编辑、汇编、连接文件3,4。

2. 执行该程序。即在命令行提示符后输入3后回车，观察执行结果。（以店主身份进入）

3. 执行功能一，观察执行输出

4. 执行功能二，调用功能一观察执行输出

5. 执行功能三，调用功能五观察执行输出

6. 执行功能四，调用功能五观察执行输出

### 3.1.5实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

2. 汇编过程没有发生异常

3. 连接过程没有发生异常。

4. 输入用户名和密码，以店主身份进入。

1. 执行功能一，根据提示输入要查询的商品的名称，输出找到的商品的信息，执行结果如图3.1.5.1所示：

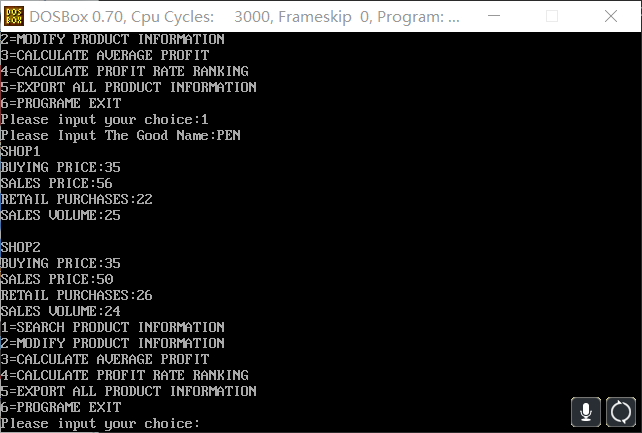
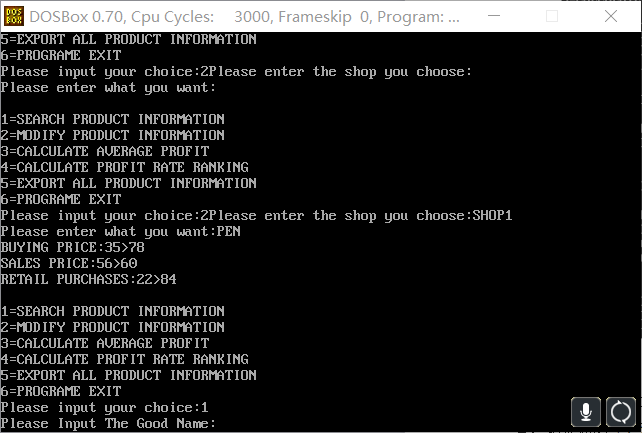


图3.1.5.1 功能一

1. 执行功能二，输入想要修改的商店，商品以及想要修改的内容，修改后执行功能一进行查看，结果如图3.1.5.2所示：



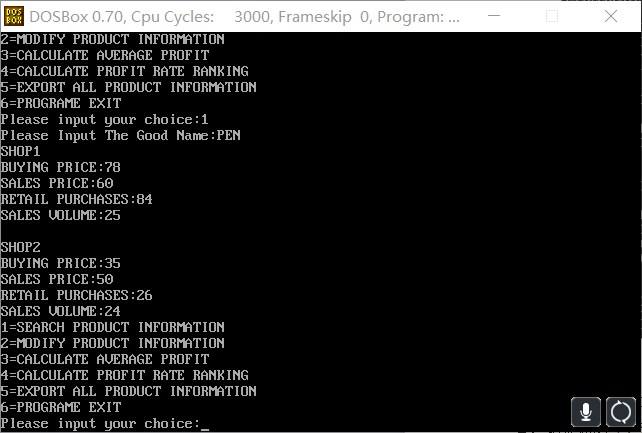
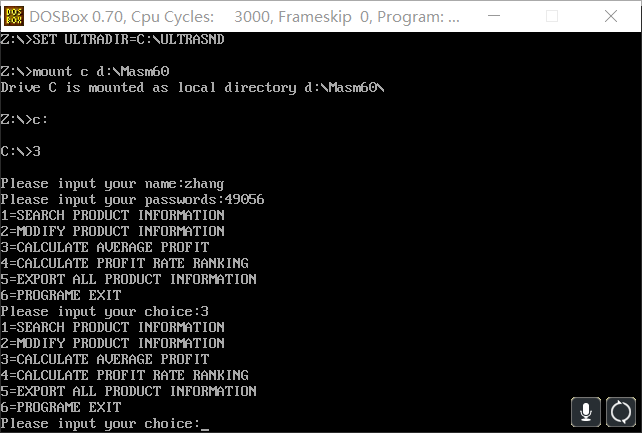


图3.1.5.2 功能二

1. 执行功能三，经过计算和修改后返回菜单，调用功能五进行查看，如图3.1.5.3所示：



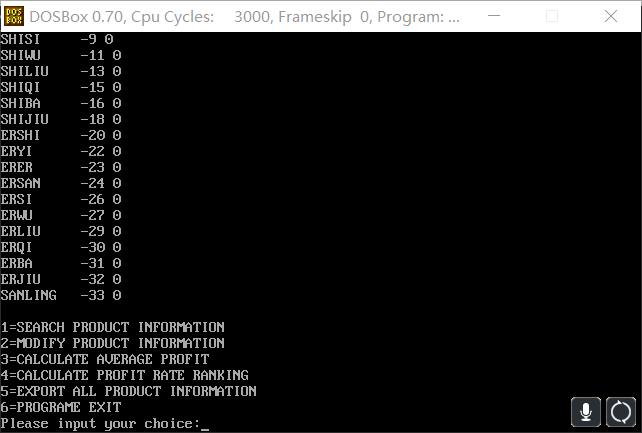
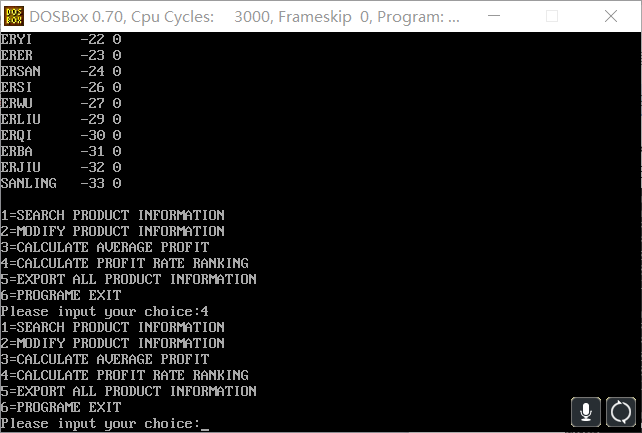


图3.1.5.3 功能三（商品过多展示不全）

1. 执行功能四，经过排序和保存后返回菜单（排序子程序仅适用于两个商店商品顺序相同时的情况），调用功能五进行查看，如图3.1.5.4所示：



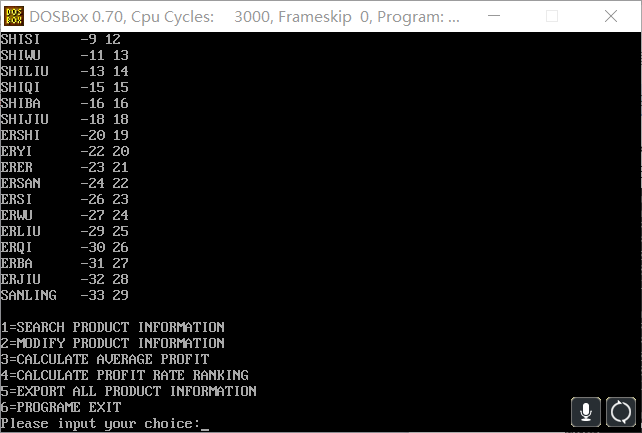


图3.1.5.4 功能四（商品过多展示不全）

1. 思考题：
2. 想要进入子程序内部，可以通过Trace into（快捷键F7），或者可以在子程序内部设置断点，让程序运行至断点处。在进入子程序之前，程序会将调用子程序的下一条代码的ip地址压入堆栈中，在子程序结束RET时，将其弹出，使程序从调用子程序处继续运行。

H$J%3~19UOVG%H]Z%$PX56J

FV3@@NPEIALK%20HZYM6O_S

FAR类型RET的机械码是CB，而NEAR类型的是C3。在调用FAR型子程序时，会将代码段的地址放入堆栈中，以便在进入其他代码段后能返回时能准确地返回原来的主程序。

1. 将多个模块合并成一个模块，能使程序的整体性更强，需要的功能介绍，输入输出介绍更简单，调用关系图更简单，但会可能会使程序更难读懂，模块之间有需要公用的重复部分；而如果将一个模块拆分成多个模块，能使每个模块的功能更加明确，简洁，易懂，且可供多个模块调用，通用性强，减少代码量，但是需要的功能，输入输出介绍会更加复杂，调用关系图更为复杂，修改调试时更为麻烦。我们在子程序和模块化程序设计时应该将需要共用的部分设计成子程序，而完整性强，不同功能的部分分别设计成不同的子程序，在子程序前添加功能，输入输出介绍能够让使用者在调用时更加方便，在调试，测试时也会更加顺畅，同时也能增加程序可读性，而调用关系图可以完整地反映整个程序的整体结构，让使用者对程序的整体构架逻辑的认识更为清晰。
2. 用PUBLIC定义的不同模块中的同名段相连接形成段，其连接次序由连接命令指定，例如实验中连接指令为link 3+4，则4.asm中的数据段连接在3.asm数据段之后。
3. 观察模块间的参数的传递方法，包括公共符号的定义和外部符号的引用，若符号名不一致或类型不一致会有什么现象发生？若符号名不一致或类型不一致在编译会报错
4. 宏调用在运行过程中直接将宏调用指令用宏定义中的内容替代，而子程序调用时运用CALL语句进入子程序内部进行操作执行。宏本质上依然是代码段，在预处理指令时被转化为对应的源代码。
5. EXTRN说明语句放在.386之前时可以正常调用执行所有的功能，但将EXTRN说明语句放在调用.386指令集之后，运行前两个功能时正常，但运行到第三个时出现虚拟WIN32系统报错，然后程序崩溃，猜测原因可能是关于32位寄存器的调用。
6. 本程序中的宏设计：

M9 MACRO A

LEA DX,A

MOV AH,9

INT 21H

ENDM ;9号功能调用

M2 MACRO A

MOV DL,A

MOV AH,2

INT 21H

ENDM ;2号功能调用

M10 MACRO A

LEA DX,A

MOV AH,10

INT 21H

ENDM ;10号功能调用

NEWLINE MACRO

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

ENDM ;换行

**任务2. 在C语言程序中调用 汇编语言实现的函数**

### 3.2.1设计思想及存储单元分配

为方便C语言中对汇编程序数据段中的变量进行调用，将两个商店中第一个商品的名称进行修改：

\_GA1:商店1中第一个商品的名称

\_GB1:商店2中第二个商品的名称

### 3.2.2流程图

子程序流程图与任务一相同。

### 3.2.3源程序

menu.c：（包含菜单功能和用C语言实现的功能五）

#include <stdio.h>

#include <stdlib.h>

extern void FUN1(void);

extern void FUN2(void);

extern void FUN3(void);

extern void FUN4(void);

extern void FUN5(void);

extern char GA1;

extern char GB1;

void function5(void)

{

int i = 0;

for(i = 0;i < 600; i = i + 20)

{

printf("%s\t", &GA1+i);

printf("%d\t", (short) (\*(&GA1+18+i)));

printf("%d\n", (short) (\*(&GB1+18+i)));

}

}

int main(void)

{

int i;

while (1)

{

printf("-------- menu --------\n");

printf(" 1.query the good\n");

printf(" 2.modify the good\n");

printf(" 3.compute the avrp\n");

printf(" 4.sort\n");

printf(" 5.outall\n");

printf(" 6.quit\n");

printf("-------- menu --------\n");

printf("choose sth 1~6:\n");

scanf("%d", &i);

switch (i)

{

case 1:

FUN1();

break;

case 2:

FUN2();

break;

case 3:

FUN3();

break;

case 4:

FUN4();

break;

case 5:

function5();

break;

default:

break;

}

if (i == 6)

break;

}

return 0;

}

test.asm（去除掉菜单功能，将所有子程序放在一个asm文件中）：

.386

PUBLIC \_FUN1,\_FUN2,\_FUN3,\_FUN4,\_FUN5

PUBLIC \_GA1,\_GB1

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;宏定义

M9 MACRO A

LEA DX,A

MOV AH,9

INT 21H

ENDM ;9号功能调用

M2 MACRO A

MOV DL,A

MOV AH,2

INT 21H

ENDM ;2号功能调用

M10 MACRO A

LEA DX,A

MOV AH,10

INT 21H

ENDM ;10号功能调用

NEWLINE MACRO

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

ENDM ;换行

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

\_DATA SEGMENT WORD PUBLIC USE16 'DATA'

INPUT\_NAME\_MSG DB 0AH,0DH,'Please input your name:$'

INPUT\_PASSWORD\_MSG DB 0AH,0DH,'Please input your passwords:$'

LOGIN\_FAIL\_MSG DB 0AH,0DH,'Login failed,Try Again:$'

GOODNAME\_MSG DB 0AH,0DH,'Please Input The Good Name:$'

GOODNAME\_MSG2 DB 0AH,0DH,'There is no this good in shop1.Please input again:$'

END\_MSG DB 0AH,0DH,'Press anykey to exit:$'

SHOWGOODNAME DB 0AH,0DH,'GoodName:$'

BUYPRICE DB 0AH,0DH,'BuyPrice:$'

PRICE DB 0AH,0DH,'Price:$'

TOTAL DB 0AH,0DH,'Total:$'

SOLD DB 0AH,0DH,'Sold:$'

PROFITRATE DB 0AH,0DH,'ProfitRate:$'

PEN DB 0AH,0DH,'PEN$'

BOOK DB 0AH,0DH,'BOOK$'

SHOP1 DB 0AH,0DH,'SHOP1$'

SHOP2 DB 0AH,0DH,'SHOP2$'

BUYMASSGE DB 0AH,0DH,'Please Input The Number You Want To Buy:$'

SOLDMASSGE DB 0AH,0DH,'OVERSOLD$'

JHJ DB 'BUYING PRICE:$'

XSJ DB 'SALES PRICE:$'

JHL DB 'RETAIL PURCHASES:$'

XSL DB 'SALES VOLUME:$'

SHO DB 'Please enter the shop you choose:$'

FIN DB 'Please enter what you want:$'

S1 DB 'SHOP1',0

\_GA1 DB 'PEN',7 DUP(0)

DW 35,56,22,25,?

GA2 DB 'BOOK',6 DUP(0)

DW 12,30,33,5,?

GA3 DB 'BAG',7 DUP(0)

DW 35,56,41,25,?

GA4 DB 'SI',8 DUP(0)

DW 35,56,42,25,?

GA5 DB 'WU',8 DUP(0)

DW 35,56,43,25,?

GA6 DB 'LIU',7 DUP(0)

DW 35,56,44,25,?

GA7 DB 'QI',8 DUP(0)

DW 35,56,45,25,?

GA8 DB 'BA',8 DUP(0)

DW 35,56,46,25,?

GA9 DB 'JIU',7 DUP(0)

DW 35,56,47,25,?

GA10 DB 'SHI',7 DUP(0)

DW 35,56,48,25,?

GA11 DB 'SHIYI',5 DUP(0)

DW 35,56,49,25,?

GA12 DB 'SHIER',5 DUP(0)

DW 35,56,50,25,?

GA13 DB 'SHISAN',4 DUP(0)

DW 35,56,51,25,?

GA14 DB 'SHISI',5 DUP(0)

DW 35,56,52,25,?

GA15 DB 'SHIWU',5 DUP(0)

DW 35,56,53,25,?

GA16 DB 'SHILIU',4 DUP(0)

DW 35,56,54,25,?

GA17 DB 'SHIQI',5 DUP(0)

DW 35,56,55,25,?

GA18 DB 'SHIBA',5 DUP(0)

DW 35,56,56,25,?

GA19 DB 'SHIJIU',4 DUP(0)

DW 35,56,57,25,?

GA20 DB 'ERSHI',5 DUP(0)

DW 35,56,58,25,?

GA21 DB 'ERYI',6 DUP(0)

DW 35,56,59,25,?

GA22 DB 'ERER',6 DUP(0)

DW 35,56,60,25,?

GA23 DB 'ERSAN',5 DUP(0)

DW 35,56,61,25,?

GA24 DB 'ERSI',6 DUP(0)

DW 35,56,62,25,?

GA25 DB 'ERWU',6 DUP(0)

DW 35,56,63,25,?

GA26 DB 'ERLIU',5 DUP(0)

DW 35,56,64,25,?

GA27 DB 'ERQI',6 DUP(0)

DW 35,56,65,25,?

GA28 DB 'ERBA',6 DUP(0)

DW 35,56,66,25,?

GA29 DB 'ERJIU',5 DUP(0)

DW 35,56,67,25,?

GA30 DB 'SANLING',3 DUP(0)

DW 35,56,68,25,?

S2 DB 'SHOP2', 0

\_GB1 DB 'PEN', 7 DUP(0)

DW 35, 50, 26, 24, ?

GB2 DB 'BOOK', 6 DUP(0)

DW 12, 28, 27, 15, ?

GB3 DB 'BAG',7 DUP(0)

DW 35,50,28,24,?

GB4 DB 'SI',8 DUP(0)

DW 35,56,29,25,?

GB5 DB 'WU',8 DUP(0)

DW 35,56,30,25,?

GB6 DB 'LIU',7 DUP(0)

DW 35,56,10000,25,?

GB7 DB 'QI',8 DUP(0)

DW 35,56,31,25,?

GB8 DB 'BA',8 DUP(0)

DW 35,56,32,25,?

GB9 DB 'JIU',7 DUP(0)

DW 35,56,33,25,?

GB10 DB 'SHI',7 DUP(0)

DW 35,56,34,25,?

GB11 DB 'SHIYI',5 DUP(0)

DW 35,56,35,25,?

GB12 DB 'SHIER',5 DUP(0)

DW 35,56,36,25,?

GB13 DB 'SHISAN',4 DUP(0)

DW 35,56,37,25,?

GB14 DB 'SHISI',5 DUP(0)

DW 35,56,38,25,?

GB15 DB 'SHIWU',5 DUP(0)

DW 35,56,39,25,?

GB16 DB 'SHILIU',4 DUP(0)

DW 35,56,40,25,?

GB17 DB 'SHIQI',5 DUP(0)

DW 35,56,41,25,?

GB18 DB 'SHIBA',5 DUP(0)

DW 35,56,42,25,?

GB19 DB 'SHIJIU',4 DUP(0)

DW 35,56,43,25,?

GB20 DB 'ERSHI',5 DUP(0)

DW 35,56,44,25,?

GB21 DB 'ERYI',6 DUP(0)

DW 35,56,45,25,?

GB22 DB 'ERER',6 DUP(0)

DW 35,56,46,25,?

GB23 DB 'ERSAN',5 DUP(0)

DW 35,56,47,25,?

GB24 DB 'ERSI',6 DUP(0)

DW 35,56,48,25,?

GB25 DB 'ERWU',6 DUP(0)

DW 35,56,49,25,?

GB26 DB 'ERLIU',5 DUP(0)

DW 35,56,50,25,?

GB27 DB 'ERQI',6 DUP(0)

DW 35,56,51,25,?

GB28 DB 'ERBA',6 DUP(0)

DW 35,56,52,25,?

GB29 DB 'ERJIU',5 DUP(0)

DW 35,56,53,25,?

GB30 DB 'SANLING',3 DUP(0)

DW 35,56,54,25,?

COUNTN EQU 8

COUNTP EQU 4

N EQU 30

BUFF DB 12 DUP(?)

AUTH DB 0

COST1 DD 0

COST2 DD 0

PROFIT1 DD 0

PROFIT2 DD 0

PR1 DD 0

PR2 DD 0

APR DD 0

APRN DW 30 DUP(0)

ADR DW 30 DUP(0)

DIP DB 6,0,6 DUP(0)

OUTNUM DW 10 DUP(0)

JGF DB '>'

SN DB 6,0,6 DUP(0)

GN DB 10,0,10 DUP(0)

REALNUM DW 0

INPUT DW 0

JISHU DW 0

CHS DW 10

LOC DW 0

LOCB DW 0

LOCR DW 0

COUNT DW 0

M DW 9000

M1 DW 0

RANK DW 1

IP1 DW 0

IP2 DW 0

INNAME DB 10

DB 0

DB 10 DUP(0)

INPASSWORD DB 10

DB 10

DB 10 DUP(0)

GETGOODNAME DB 10

DB 0

DB 10 DUP(0)

NUMBER DW 10

DW 0

DW 10 DUP(0)

BNAME DB 'zhang',5 DUP(0);老板姓名

BPASSWORD DB '49056',5 DUP(0);老板密码

BUF DW 0

BUF1 DW 0

CAIDAN1 DB 0AH,0DH,'1=SEARCH PRODUCT INFORMATION$'

CAIDAN2 DB 0AH,0DH,'2=MODIFY PRODUCT INFORMATION$'

CAIDAN3 DB 0AH,0DH,'3=CALCULATE AVERAGE PROFIT$'

CAIDAN4 DB 0AH,0DH,'4=CALCULATE PROFIT RATE RANKING$'

CAIDAN5 DB 0AH,0DH,'5=EXPORT ALL PRODUCT INFORMATION$'

CAIDAN6 DB 0AH,0DH,'6=PROGRAME EXIT$'

CAIDAN7 DB 0AH,0DH,'You can just input between 1 to 6!$'

CAIDAN8 DB 0AH,0DH,'Please input your choice:$'

CAIDAN9 DB 0AH,0DH,'You can just input between 1 or 6!$'

\_DATA ENDS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

STACKS SEGMENT USE16 STACK 'STACKS'

DB 200 DUP(0)

STACKS ENDS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

\_TEXT SEGMENT BYTE PUBLIC USE16 'CODE'

ASSUME CS:\_TEXT,DS:\_DATA,SS:STACKS

\_FUN1 PROC NEAR

FUNCTION3:

M9 GOODNAME\_MSG

M10 GETGOODNAME

MOV SI,2;输入为回车时返回

MOV AL,0DH

MOV BL,GETGOODNAME[SI]

CMP AL,BL

JE EXIT1

MOV DX,0

MOV SI,0

MOV CL,GETGOODNAME[SI+1]

FUNCTION3\_0\_1:

MOV DI,0

IMUL SI,DX,20

CMP SI,580

JE FUNCTION3

INC DX

FUNCTION3\_1:

MOV AL,GETGOODNAME[DI+2];输入名称与商品名称比较

MOV BL,\_GA1[SI]

CMP AL,BL

JNE FUNCTION3\_0\_1;不符合跳转到该商店的下一商品

INC SI

INC DI

DEC CL

JNZ FUNCTION3\_1

MOV CL,GETGOODNAME[1]

DEC CL

MOV CH,0

SUB SI,CX

JMP INFO;符合跳转到该商品的信息

INFO:

DEC DX

MOV JISHU,DX

IMUL SI,DX,20

M9 SHOP1

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHJ

MOV AX,WORD PTR \_GA1[SI+10]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSJ

MOV AX,WORD PTR \_GA1[SI+12]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHL

MOV AX,WORD PTR \_GA1[SI+14]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSL

MOV AX,WORD PTR \_GA1[SI+16]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 SHOP2

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHJ

MOV AX,WORD PTR \_GB1[SI+10]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSJ

MOV AX,WORD PTR \_GB1[SI+12]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHL

MOV AX,WORD PTR \_GB1[SI+14]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSL

MOV AX,WORD PTR \_GB1[SI+16]

MOV DX,16

CALL PRINTAX

EXIT1:

RET

\_FUN1 ENDP

;;;;;;;;;;;;;;;;;;;;;;;;

\_FUN2 PROC NEAR

FST2: LEA DX, SHO

MOV AH, 9

INT 21H

LEA DX, SN

MOV AH, 10

INT 21H

NEWLINE

LEA DX, FIN

MOV AH, 9

INT 21H

LEA DX, GN

MOV AH, 10

INT 21H

NEWLINE

CMP SN+2, 13

JE BACK2

MOV BX, 0

MOV CX, 0

MOV CL, [SN+1]

SCOMP: LEA SI, SN+2

LEA DI, [S1+BX]

CSC: MOV AL, [SI]

CMP AL, [DI]

JNE SALL

INC SI

INC DI

DEC CX

JE SCPL

JMP CSC

GNL: MOV CX, 0

MOV CL, [GN+1]

ADD BX, 6

GCOMP: LEA SI, GN+2

LEA DI, [S1+BX]

CGC: MOV AL, [SI]

CMP AL, [DI]

JNE GALL

INC SI

INC DI

DEC CX

JE GCPL

JMP CGC

MJJ: MOV LOC, BX

LEA DX, JHJ

MOV AH, 9

INT 21H

MOV AX, WORD PTR [S1+BX+10]

MOV DX, 16

CALL PRINTAX

PJG: MOV DL, [JGF]

MOV AH, 2

INT 21H

LEA DX, DIP

MOV AH, 10

INT 21H

NEWLINE

MOV BX, 0

LEA DI, DIP+2

CTS: MOV AX, 0

MOV CX, 0

MOV CL, [DIP+1]

DEC CX

SUB CX, BX

MOV AL, [DI+BX]

CMP AL, '0'

JL MJJ

CMP AL, '9'

JG MJJ

TTS: SUB AX, 30H

CMP CX, 0

JE TAD

IMUL AX, 10

DEC CX

JNE TTS

TAD: MOV DX, COUNT

ADD DX, AX

MOV COUNT, DX

INC BX

CMP BL, BYTE PTR [DIP+1]

JNE CTS

LJJ: MOV BX, LOC

MOV AX, COUNT

MOV WORD PTR [S1+BX+10], AX

MOV COUNT, 0

MXJ: MOV LOC, BX

LEA DX, XSJ

MOV AH, 9

INT 21H

MOV AX, WORD PTR [S1+BX+12]

MOV DX, 16

CALL PRINTAX

MOV DL, [JGF]

MOV AH, 2

INT 21H

LEA DX, DIP

MOV AH, 10

INT 21H

NEWLINE

MOV BX, 0

LEA DI, DIP+2

CTS2: MOV AX, 0

MOV CX, 0

MOV CL, [DIP+1]

DEC CX

SUB CX, BX

MOV AL, [DI+BX]

CMP AL, '0'

JL MXJ

CMP AL, '9'

JG MXJ

TTS2: SUB AX, 30H

CMP CX, 0

JE TAD2

IMUL AX, 10

DEC CX

JNE TTS2

TAD2: MOV DX, COUNT

ADD DX, AX

MOV COUNT, DX

INC BX

CMP BL, BYTE PTR [DIP+1]

JNE CTS2

LXJ: MOV BX, LOC

MOV AX, COUNT

MOV WORD PTR [S1+BX+12], AX

MOV COUNT, 0

MJL: MOV LOC, BX

LEA DX, JHL

MOV AH, 9

INT 21H

MOV AX, WORD PTR [S1+BX+14]

MOV DX, 16

CALL PRINTAX

MOV DL, [JGF]

MOV AH, 2

INT 21H

LEA DX, DIP

MOV AH, 10

INT 21H

NEWLINE

MOV BX, 0

LEA DI, DIP+2

CTS3: MOV AX, 0

MOV CX, 0

MOV CL, [DIP+1]

DEC CX

SUB CX, BX

MOV AL, [DI+BX]

CMP AL, '0'

JL MJL

CMP AL, '9'

JG MJL

TTS3: SUB AX, 30H

CMP CX, 0

JE TAD3

IMUL AX, 10

DEC CX

JNE TTS3

TAD3: MOV DX, COUNT

ADD DX, AX

MOV COUNT, DX

INC BX

CMP BL, BYTE PTR [DIP+1]

JNE CTS3

LJL: MOV BX, LOC

MOV AX, COUNT

MOV WORD PTR [S1+BX+14], AX

MOV COUNT, 0

JMP BACK2

SALL: CMP BX, 1212

JE FST2

ADD BX, 606

JMP SCOMP

SCPL: CMP BYTE PTR [DI], 0

JE GNL

JMP SALL

GALL: CMP BX, 606

JE FST2

CMP BX, 1212

JE FST2

ADD BX, 20

JMP GCOMP

GCPL: CMP BYTE PTR [DI], 0

JE MJJ

JMP GALL

BACK2: RET

\_FUN2 ENDP

;;;;;;;;;;;;;;;;;

\_FUN3 PROC NEAR

MOV BX, 0

BAP: MOV EAX, 0

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [\_GA1+BX+10]

MOV CX, WORD PTR [\_GA1+BX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [\_GA1+BX+12]

MOV DX, WORD PTR [\_GA1+BX+16]

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR1, EAX

MOV LOC, BX

MOV CX, 0

MOV BX, LOC

LEA DI, [\_GA1+BX]

MOV BX, 0

LEA SI, [\_GB1+BX]

COMP\_3: MOV CL, 10

LEA SI, [\_GB1+BX]

MOV LOCB, BX

MOV BX, LOC

LEA DI, [\_GA1+BX]

MOV BX, LOCB

RCM: MOV AL, [DI]

CMP AL, [SI]

JNE ALL

INC DI

INC SI

DEC CX

JE CCOP

JMP RCM

CCOP: MOV EAX, 0

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [\_GB1+BX+10]

MOV CX, WORD PTR [\_GB1+BX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [\_GB1+BX+12]

MOV DX, WORD PTR [\_GB1+BX+16]

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR2, EAX

MOV EDX, PR1

ADD EAX, EDX

SAR EAX, 1

MOV APR, EAX

MOV BX, LOC

MOV WORD PTR [\_GA1+BX+18], AX

ADD BX, 20

CMP BX, 600

JE BACK3

JMP BAP

ALL: ADD BX,20

JMP COMP\_3

BACK3: RET

\_FUN3 ENDP

;;;;;;;;;;;;;;;;;

\_FUN4 PROC NEAR

MOV CX,-1

LOOPX:

INC CX

IMUL BX,CX,20

MOV AX,WORD PTR \_GA1+18[BX]

PUSH CX

PUSH BX

MOV CX,-1

MOV RANK,1

LOOPY:

INC CX

IMUL BX,CX,20

CMP AX,WORD PTR \_GA1+18[BX]

JL LARANK

L3P:

CMP CX,N-1

JGE L352

JMP LOOPY

L352:

POP BX

POP CX

JMP L351

LARANK:

INC RANK

JMP L3P

L351:

MOV AX,RANK

MOV WORD PTR \_GB1+18[BX],AX

CMP CX,N-1

JGE LLL

JMP LOOPX

LLL: RET

\_FUN4 ENDP

;;;;;;;;;;;;;;;;;

\_FUN5 PROC NEAR

F1:

MOV DI,0

LL:

IMUL SI,DI,20

MOV BYTE PTR \_GA1[SI+9],'$'

MOV BYTE PTR \_GB1[SI+9],'$'

M9 \_GA1[SI]

MOV DL,32

MOV AH,2

INT 21H

MOV AX,WORD PTR \_GA1[SI+18]

MOV DX,16

CALL PRINTAX

MOV DL,32

MOV AH,2

INT 21H

MOV AX,WORD PTR \_GB1[SI+18]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

INC DI

CMP DI,30

JNE LL

RET

\_FUN5 ENDP

;;;;;;;;;;;;;;;;;

PRINTAX PROC NEAR

PUSH EBX

PUSH SI

LEA SI,BUFF

CMP DX,32

JE L

MOVSX EAX,AX

L: OR EAX,EAX

JNS PLUS

NEG EAX

MOV BYTE PTR [SI],'-'

INC SI

PLUS: MOV EBX,10

CALL RDAIX

MOV BYTE PTR [SI],'$'

LEA DX,BUFF

MOV AH,9

INT 21H

POP SI

POP EBX

RET

PRINTAX ENDP

RDAIX PROC NEAR

PUSH CX

PUSH EDX

XOR CX,CX

LOP1: XOR EDX,EDX

DIV EBX

PUSH DX

INC CX

OR EAX,EAX

JNZ LOP1

LOP2: POP AX

CMP AL,10

JB LL1

ADD AL,7

LL1: ADD AL,30H

MOV [SI],AL

INC SI

LOOP LOP2

POP EDX

POP CX

RET

RDAIX ENDP

\_TEXT ENDS

END

### 3.2.4实验步骤

1. 打开BC，新建项目test.prj，在项目中添加准备好地menu.c和test.asm

2. 对程序进行编译连接，选择build all

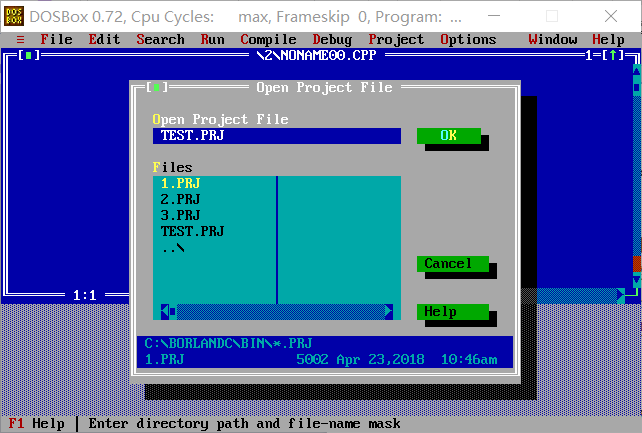
3. 成功后，选择run运行程序。

4. 执行各个功能，观察能否正常运行

### 3.2.5实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

2. 新建项目test.prj，在项目中添加准备好地menu.c和test.asm，如图3.2.5.1所示：



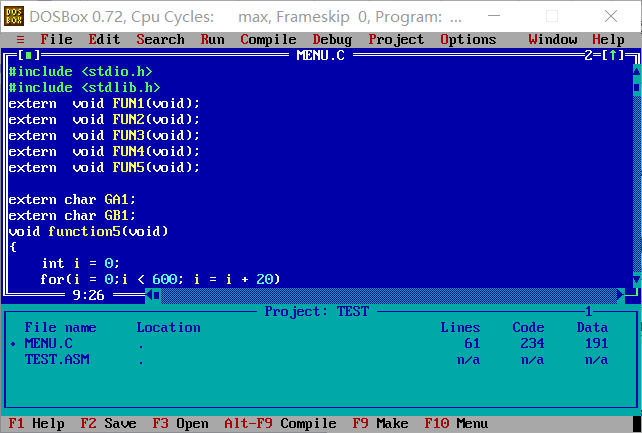


图3.2.5.1 新建项目并添加文件

1. 编译，结果如图3.2.5.2所示：

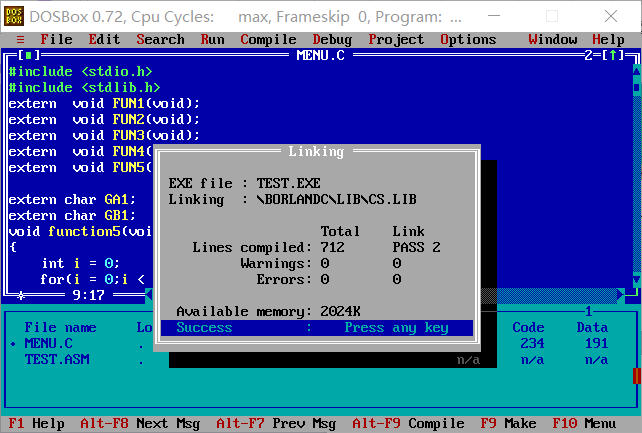


图3.2.5.2 编译成功

1. 调用的子程序代码逻辑功能及运行结果与任务一，不在此展示，在此只展示在C语言环境下实现的功能五，如图3.2.5.3所示：

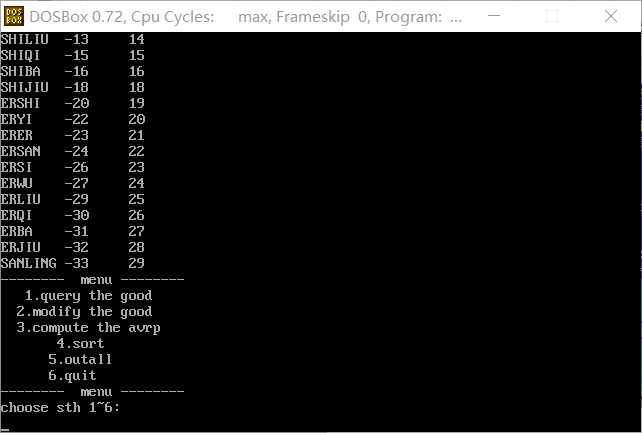


图3.2.5.3 由C语言实现的功能五

1. 思考题：

C语言对汇编中子程序或者变量名称的识别机制要求在子程序和变量名称前面加上‘\_’。C语言中调用汇编中子程序时，传递的参数被压入栈中，然后调用子程序，子程序中的参数返回C时，出口固定为AX，过堆栈传递参数后可以通过根据参数的个数来调整sp的指向从而回收使用的堆栈空间。

# 4 总结与体会

这次任务是在原来商店任务的基础上添加了许多新的功能，要求运用子程序进行模块化的编程，并用C语言进行与汇编的混合编程，新添的功能有些较为复杂，在第一周分组进行完成，而第二周的C语言汇编混合编程虽然并不复杂，但是需要对原有程序进行修改，对连接调试等过程进行探索。

任务一中，由于对题目理解的错误，我负责了功能二，三，四，由我的队友负责菜单和功能一，二，在这三个功能中，功能三是之前做过的，而功能二，四是从未在汇编中实现过的，虽然在C语言中很常用，但是用汇编的逻辑去编写这两个功能着实花了不少功夫，而在功能的实现中我所使用的寻址方式与我使用的寄存器不符导致在很多地方出现了bug，这也使我调试bug上花费了很大功夫，这导致我们在课上大部分时间都用来debug，在此我要感谢我的同组的同学独自调试好了两个程序的连接，让我在调试完程序后能够立刻与他的程序相连接，这次实验的经历让我重新认识到汇编程序在编写上的限制和即使要实现一个不难的逻辑也要花费大量时间与精力。

任务二是我第一次亲身感受到汇编语言与高级语言的关系，通过对自己汇编程序进行规定要求的修改，我成功地在C语言程序中调用了汇编中的子程序，也成功调用了汇编数据段中的数据，完成了用C语言实现任务一中功能五的要求。这次的任务虽然不难，但是由与版本或是其他的问题，如何成功地将C语言程序和汇编程序连接起来让我花费很多时间去探究，虽然最后成功实现了功能，但对其要求，规定，格式等还有很多不清楚的地方。

通过本次实验我再次提高了自己对于汇编程序的编写熟练度和经验，在以后的实验中我会吸取教训不再犯与这次相同的低级错误从而导致自己差点没能完成任务，同时这次实验也让我真正认识到了汇编语言在我们以后的实验，编程，工作中将会发挥不小的作用，虽然其版本可能不再适用，但是它其中所包含的逻辑，它其中最底层对计算机的调用的功能，形式必定会对我们有所帮助。

# 参考文献

[1] 作者：王元珍 曹忠升 韩宗芬，书名：80X86汇编语言程序设计，出版者：华中科技大学出版社。



**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验四 中断与反跟踪**

**实验时间：2018-5-10， 2018-5-17，8：00-11：50 实验地点： 南一楼804室**

**指导教师： 周英飚**

**专业班级：计算机卓越工程师班1601班**

**学 号： U201612922 姓 名： 刘子煜**

**同组学生： 无 报告日期： 2018年 5 月 10日 2018年 5 月 17日**

**原创性声明**

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

特此声明！

学生签名：

日期：2018.5.10，2018.5.17

成绩评定

|  |  |  |
| --- | --- | --- |
| 实验完成质量得分（70分）（实验步骤清晰详细深入，实验记录真实完整等） | 报告撰写质量得分（30分）（报告规范、完整、通顺、详实等） | 总成绩（100分） |
|  |  |  |

指导教师签字：

                     日期：

**目录**

[1 实验目的与要求 1](#_Toc476742634)

[2 实验内容 1](#_Toc476742635)

[3 实验过程 3](#_Toc476742636)

[3.1 任务一 3](#_Toc476742637)

[3.1.1 设计思想及存储单元分配 3](#_Toc476742641)

[3.1.2 流程图 4](#_Toc476742641)

[3.1.3 源程序 4](#_Toc476742642)

[3.1.4 实验步骤](#_Toc476742643) 5

[3.1.5 实验记录与分析](#_Toc476742644) 5

[3.2 任务二 7](#_Toc476742637)

[3.2.1 设计思想及存储单元分配](#_Toc476742641) 7

[3.2.2 流程图](#_Toc476742641) 7

[3.2.3 源程序](#_Toc476742642) 9

[3.2.4 实验步骤](#_Toc476742643) 10

[3.2.5 实验记录与分析](#_Toc476742644) 10

[3.3 任务三 1](#_Toc476742637)2

[3.3.1 设计思想及存储单元分配 1](#_Toc476742641)2

[3.3.2 流程图 1](#_Toc476742641)3

[3.3.3 源程序 1](#_Toc476742642)3

[3.3.4 实验步骤](#_Toc476742643) 15

[3.3.5 实验记录与分析](#_Toc476742644) 15

[3.4 任务四 16](#_Toc476742637)

[3.4.1 设计思想及存储单元分配 1](#_Toc476742641)6

[3.4.2 流程图 1](#_Toc476742641)7

[3.4.3 源程序 1](#_Toc476742642)7

[3.4.4 实验步骤](#_Toc476742643) 40

[3.4.5 实验记录与分析](#_Toc476742644) 40

[3.5 任务五 4](#_Toc476742637)4

[3.5.1 设计思想及存储单元分配 4](#_Toc476742641)4

[3.5.2 流程图 4](#_Toc476742641)4

[3.5.3 源程序 4](#_Toc476742642)4

[3.5.4 实验步骤](#_Toc476742643) 45

[3.5.5 实验记录与分析](#_Toc476742644) 45

[4 总结与体会 4](#_Toc476742646)8

[参考文献 4](#_Toc476742647)9

# 1 实验目的与要求

(1) 掌握中断矢量表的概念；

(2） 熟悉I/O访问，BIOS功能调用方法；

(3) 掌握实方式下中断处理程序的编制与调试方法；

(4) 熟悉跟踪与反跟踪的技术；

(5) 提升对计算机系统的理解与分析能力。

# 2 实验内容

**任务1：用三种方式获取中断类型码1H 、10H对应的中断处理程序的入口地址。**

要求：首先要进入虚拟机状态，然后

（1） 直接运行调试工具（TD.EXE），观察中断矢量表中的信息。

（2） 编写程序，用 DOS系统功能调用方式获取，观察功能调用相应的出口参数与“（1）”看到的结果是否相同 （使用TD观看出口参数即可）。

（3） 编写程序，直接读取相应内存单元，观察读到的数据与“（1）”看到的结果是否相同 （使用TD观看程序的执行结果即可）。

**任务2：编写一个接管键盘中断的中断服务程序并驻留内存，要求在程序返回DOS操作系统后，输入键盘上的小写字母时都变成了大写字母。**

要求：

（1）在 DOS虚拟机或DOS窗口下执行程序，中断服务程序驻留内存。

（2）在DOS命令行下键入小写字母，屏幕显示为大写，键入大写时不变。执行TD，在代码区输入指令“mov AX,0”，看是否都变成了大写。

（3）选作：另外编写一个中断服务程序的卸载程序，将键盘中断服务程序恢复到原来的状态（只需要还原中断矢量表的信息，先前驻留的程序可以不退出内存）。

**任务3：读取CMOS内指定单元的信息，按照16进制形式显示在屏幕上。**

要求：

在数据段定义一个待读取的CMOS内部单元的地址编号。再使用IN/OUT指令，读取CMOS内的指定单元的信息。

（2） 将读取的信息用16进制的形式显示在屏幕上。若是时间信息，可以人工判断一下是否与操作系统显示的时间一致。

上述任务中，值得在实验步骤或总结中解决的问题提示：

|  |  |  |
| --- | --- | --- |
| 任务编号 | 序号 | 需要尝试的操作或思考的问题 |
| 任务1 | 1 | 打开TD之后，如何在数据区切换到中断向量表所在内存区域 |
| 2 | 如何计算某个中断入口在中断向量表内的偏移地址？ |
| 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读取获取中断入口地址？可以在TD中录入指令语句或编写完整程序来尝试。 |
| 5 | 用TD把中断矢量表里的中断矢量的值随意改成其他值会有什么现象发生？（比如修改21H,1H,3H的中断矢量） |
| 任务2 | 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键盘中断时使键盘操作失灵，可以先用其他方法调试该中断服务程序，调试好后再安装成接管键盘中断的状态，请给出这种“其他”调试方法的描述并具体实施一下。 |
| 3 | 编写的中断驻留程序执行后能否正常返回到DOS？DOS是否还能正常工作？如果重复驻留多次，会有什么现象？ |
| 4 | 同时打开另外一个虚拟DOS窗口，键盘大小写是否被替代？ |
| 5 | 如何确定自己编写的中断处理程序已被系统调用？（除了小写字母已经被替换成大写的途径之外） |
| 6 | 选作的要求(3)应该如何实现，如何找到保存的原中断入口地址？ 如何保证不会错误恢复？（比如，你的程序还没有驻留，但运行了恢复程序） |
| 任务3 | 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？ |

**任务4：数据加密与反跟踪**

在实验三任务1的**网店商品信息管理程序**的基础上，增加输入用户名和密码时，最大错误次数的限制，即，当输入错误次数达到三次时，直接按照未登录状态进入后续功能。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。

可以采用计时、中断矢量表检查、堆栈检查、间接寻址等方式中的一种或多种方式反跟踪（建议采用两种反跟踪方法，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5：跟踪与数据解密**

解密同组同学的加密程序，获取各个商品的进货价。

**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。

上述任务4和5中，需要在实验步骤或总结中解决的问题提示：

1. 若密码是用明文存放在数据段中的，如何更快地获取密码？
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？（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）
3. 如何对密码实现快速的暴力破解？（描述一下实现的具体思路即可）
4. 如何综合利用静态反汇编和动态反汇编的信息破解程序？（在体会里总结一下即可）
5. 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? （在体会里具体描述）
6. 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）
7. 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？

# 3 实验过程

### 3.1任务一：用三种方式获取中断类型码1H 、10H对应的中断处理程序的入口地址。

### 3.1.1设计思想及存储单元分配

对于1H，10H对应的中断处理程序的入口地址，可以跳转地址查看，或是调用系统35H号功能进行查询，或是拿寄存器存储该地址在TD中查看。

寄存器分配：

AX，AL，AH临时寄存器。

### 3.1.2流程图

图3.1.2.1是使用三种方法查看中断程序入口地址的程序流程图。

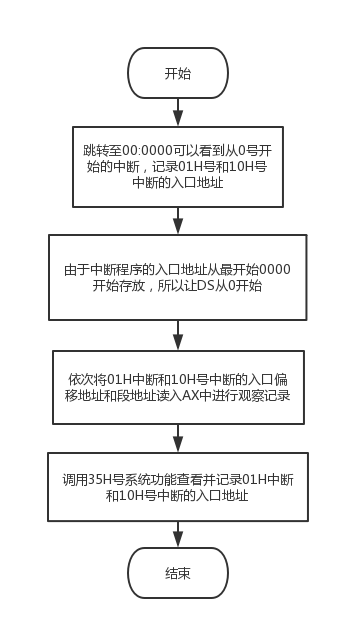


图3.1.2.1 使用三种方法查看中断程序入口地址流程图

### 3.1.3源程序

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

BUF DB 20 DUP(0)

DATA ENDS

CODE SEGMENT USE16

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

START: XOR AX, AX

MOV DS, AX

MOV AX, DS:[01H\*4]

MOV AX, DS:[01H\*4+2]

MOV AX, DS:[10H\*4]

MOV AX, DS:[10H\*4+2]

MOV AL, 01H

MOV AH, 35H

INT 21H

MOV AL, 10H

MOV AH, 35H

INT 21H

CODE ENDS

END START

### 3.1.4实验步骤

1. 准备上机实验环境、编辑、汇编、连接。

2. 使用td调试该程序。即在命令行提示符后输入td zd后回车，观察执行结果。

3. 在数据区右键Goto至00:0000，观察第01H号以及第10H号中断的入口地址并记录。

4. 按F8单步执行zd中的指令，观察MOV AX……指令执行后，AX中的值并记录。

5. 执行35H号系统功能调用后，观察BX中的值并记录。

6. 观察这三种不同方式得出的中断程序的入口地址的值是否相同。

### 3.1.5实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

2. 汇编过程没有发生异常

3. 连接过程没有发生异常。

4. 在数据区右键Goto至00:0000，第01H号以及第10H号中断的入口地址，如下图所示：

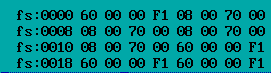




图3.1.5.1 第一种方法获取入口地址

这里得出：01H号中断入口地址为0080 0070

10H号中断入口地址为02C0 F100

1. 单步执行程序，观察到的AX的值如下图所示：

~{11S%PI}3U`A{3CWN8@1KD

)ZG)L366AW}H])JFL1`GUHE

RNC(%7ZC80D_(ZQ%%HL[TYE

XYD9PBYJ5RQ_7[AP(E{W)J1

图3.1.5.2 第二种方法获取入口地址

这里得出：01H号中断入口地址为0B1A 055C

10H号中断入口地址为02C0 F100

1. 继续执行，调用35H号系统功能可以得到如下图的结果：

_GVF]A}6F~[W2`C8TIOPE~Y

9H$ECDQV653EJ8TZN)Y`0YP

图3.1.5.3 第三种方法获取入口地址

这里得出：01H号中断入口地址为0B1A

10H号中断入口地址为02C0

1. 通过上述的实验结果，我们可以看出对用10H号中断，三种方法得到的结果都相同，而对于01H号中断，方法二与三得到结果相同，而与方法一不同。
2. 思考题：
3. 在数据区点击右键选择Goto 输入 00:0000，即切换到中断向量表所在内存区域。
4. 每个中断入口站四个字节，第n位的入口地址第一位偏移地址为n\*4。
5. 运用35H号系统功能调用。
6. 直接推算其偏移地址进行操作。
7. 直接修改会在单步执行后被系统改回原来的值。

**任务2. 编写一个接管键盘中断的中断服务程序并驻留内存，要求在程序返回DOS操作系统后，输入键盘上的小写字母时都变成了大写字母。**

### 3.2.1设计思想及存储单元分配

### 将6号中断负责键盘的部分用写好的全都显示大写的程序接管覆盖，将旧的的入口地址保存在代码段中。

1.存储单元分配

OLD\_INT：存储旧的中断程序的入口地址。

2.寄存器分配

AX、DX、AL、AH：临时寄存器。

### 3.2.2流程图

1.更改中断的程序流程图

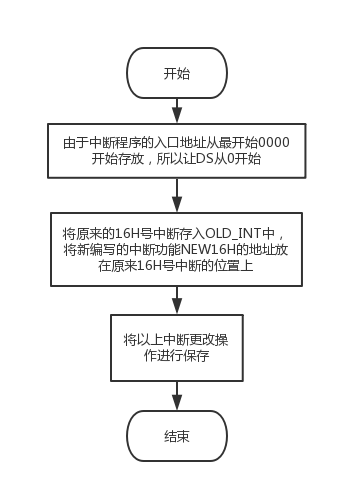


图3.2.2.1 更改为新的中断的程序流程图

2.新中断所实现的功能流程图：

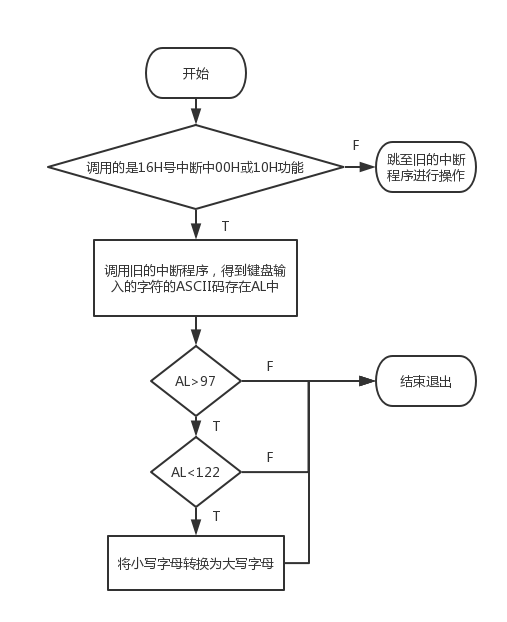


图3.2.2.2 新中断程序功能流程图

3.还原中断的流程图：

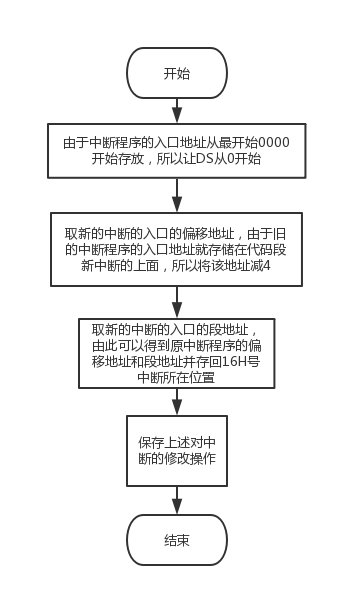


图3.2.2.3 还原旧中断流程图

### 3.2.3源程序

修改中断：

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

CODE SEGMENT USE16

ASSUME CS: CODE, SS:STACK

OLD\_INT DW ?, ?

NEW16H: PUSHF

CMP AL, 00H

JE TRS

CMP AL, 10H

JE TRS

JMP DWORD PTR OLD\_INT

TRS: CALL DWORD PTR OLD\_INT

CMP AL, 97

JNL XXP

JMP QUIT

XXP: CMP AL, 122

JNG XX

JMP QUIT

XX: SUB AL, 32

JMP QUIT

QUIT: IRET

START: XOR AX, AX

MOV DS, AX

MOV AX, DS:[16H\*4]

MOV OLD\_INT, AX

MOV AX, DS:[16H\*4+2]

MOV OLD\_INT+2, AX

CLI

MOV WORD PTR DS:[16H\*4], OFFSET NEW16H

MOV DS:[16H\*4+2], CS

STI

MOV DX, OFFSET START+15

SHR DX, 4

ADD DX, 10H

MOV AL, 0

MOV AH, 31H

INT 21H

CODE ENDS

END START

复原中断：

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

CODE SEGMENT USE16

ASSUME CS: CODE, SS:STACK

START: XOR AX, AX

MOV DS, AX

CLI

MOV BX, WORD PTR DS:[16H\*4]

SUB BX, 4

MOV SS, WORD PTR DS:[16H\*4+2]

MOV AX, SS:[BX]

MOV WORD PTR DS:[16H\*4], AX

MOV AX, SS:[BX+2]

MOV WORD PTR DS:[16H\*4+2], AX

STI

MOV DX, OFFSET START+15

SHR DX, 4

ADD DX, 10H

MOV AL, 0

MOV AH, 31H

INT 21H

CODE ENDS

END START

### 3.2.4实验步骤

1. 准备上机实验环境、编辑、汇编、连接。

2. 直接执行zd2程序。即在命令行提示符后输入zd2后回车。

3. 使用键盘随意输入字母，观察字母的大小写

4. 运行td，在代码区输入指令“mov AX,0”，观察其大小写。

5. 直接执行unist程序。即在命令行提示符后输入zd2后回车。

6. 使用键盘随意输入字母，观察字母的大小写

### 3.2.5实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

2. 汇编过程没有发生异常

3. 连接过程没有发生异常。

1. 运行zd2后，输入全部变为大写，如下图所示：

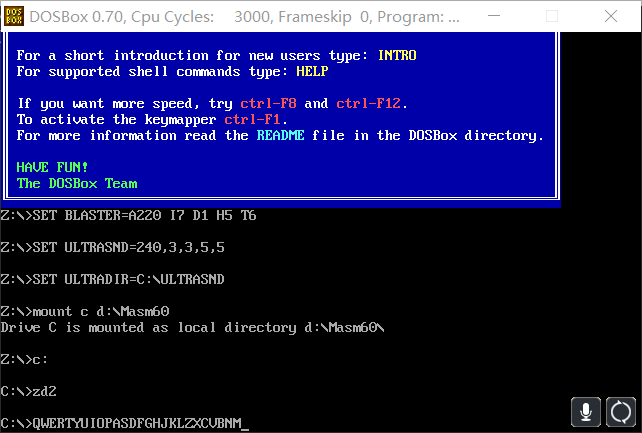


图3.2.5.1 zd2实现功能展示

1. 运行td，输入的“mov AX,0”指令没有变成大写
2. 运行unist后，输入变回其相对应的输入，如下图所示：

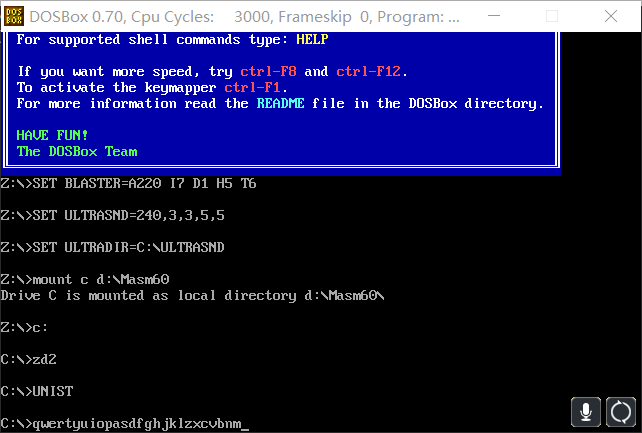


图3.2.5.2 unist实现功能展示

1. 思考题：

（1）使用CALL可以进行调用执行完后返回程序，而JMP则是跳转，进入后不再返回

（2）

（3）可以正常返回到DOS，DOS在执行其他程序时会出现键盘失灵，驻留多次并未出现特殊现象。

（4）另一个DOS窗口没有被替代。

（5）进入TD，直接查看被替代的中断处的入口地址是否发生变化，或是调用复原程序，发现复原程序能够有效地还原中断。

（6）通过将旧的中断程序的入口地址放在代码段新的中断修改的上面，通过对偏移地址的修改可以迅速找到保存的中断入口地址。在恢复之前检测原中断是否被替代。

**任务3. 读取CMOS内指定单元的信息，按照16进制形式显示在屏幕上。**

### 3.3.1设计思想及存储单元分配

### 将6号中断负责键盘的部分用写好的全都显示大写的程序接管覆盖，将旧的的入口地址保存在代码段中。

1.存储单元分配

HOUR：存储当前时间的小时

MIN：存储当前时间的分钟

SEC：存储当前时间的秒

HOUR\_AD：存储时对应的地址

MIN\_AD：存储分对应的地址

SEC\_AD：存储秒对应的地址

2.寄存器分配

AX、BX、DX、AL、AH：临时寄存器。

### 3.3.2流程图

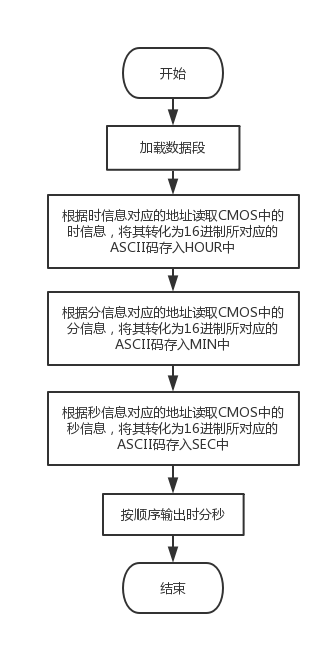


图3.3.2.1 读取CMOS中时间信息以16进制显示流程图

### 3.3.3源程序

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

HOUR DB ?, ?, ':'

MIN DB ?, ?, ':'

SEC DB ?, ?, '$'

HOUR\_AD DB 04H

MIN\_AD DB 02H

SEC\_AD DB 00H

TRJZ DB 16

DATA ENDS

CODE SEGMENT USE16

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

START: MOV AX, DATA

MOV DS, AX

MOV AL, HOUR\_AD

OUT 70H, AL

JMP $+2

IN AL, 71H

MOV AH, AL

AND AL, 0FH

SHR AH, 4

CALL TTST

MOV WORD PTR HOUR, AX

MOV AL, MIN\_AD

OUT 70H, AL

JMP $+2

IN AL, 71H

MOV AH, AL

AND AL, 0FH

SHR AH, 4

CALL TTST

MOV WORD PTR MIN, AX

MOV AL, SEC\_AD

OUT 70H, AL

JMP $+2

IN AL, 71H

MOV AH, AL

AND AL, 0FH

SHR AH, 4

CALL TTST

MOV WORD PTR SEC, AX

LEA DX, HOUR

MOV AH, 9

INT 21H

MOV AH, 4CH

INT 21H

TTST PROC

PUSH BX

MOV BX, 0

MOV BL, AH

IMUL BX, 10

MOV AH, BL

ADD AL, AH

CMP AL, 0

JE DG

MOV AH, 0

IDIV TRJZ

CMP AH, 0AH

JL DG

JMP WOD

DG: ADD AX, 3030H

JMP QUIT

WOD: ADD AX, 3730H

QUIT: POP BX

RET

TTST ENDP

CODE ENDS

END START

### 3.3.4实验步骤

1. 准备上机实验环境、编辑、汇编、连接。

2. 打开time.asm文件查看数据段中定义的待读取的CMOS内部单元的地址编号。

3. 再查看代码中如何使用IN/OUT指令，读取CMOS内的指定单元的信息。

4. 运行time，即输入time 回车，观察输出结果。

5. 与当前时间进行比较。

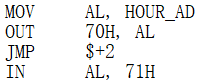
### 3.3.5实验记录与分析

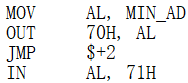
1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。
2. 汇编过程没有发生异常
3. 连接过程没有发生异常。
4. 在数据段中定义待读取的时，分，秒信息的地址编号，如下图所示：

8@DK)$SF6{N~I(T)8]$O8G1

图3.3.5.1 地址编号在数据段中的定义

1. 使用IN/OUT指令，读取CMOS内的指定单元的信息，具体程序如下图所示：





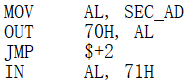


图3.3.5.2 使用IN/OUT读取信息的具体程序

1. 执行，具体程序如下图所示：

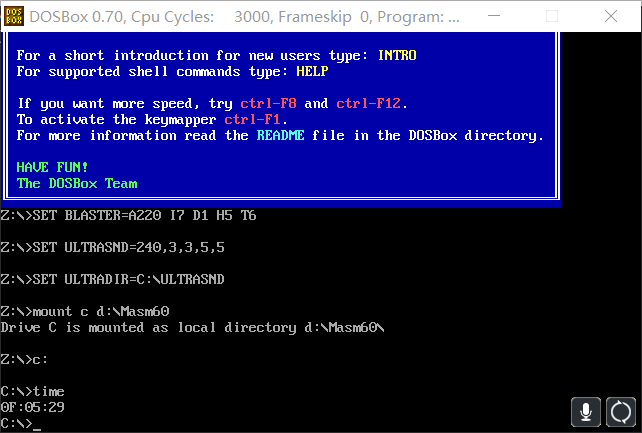


图3.3.5.3 运行程序得到的当前系统时间

经比较与当前时间相符。

1. 思考题
2. 直接在代码区输入IN/OUT相关的指令并执行。
3. 压缩BCD码的每一位用4位二进制表示，一个字节表示两位[十进制数](https://www.baidu.com/s?wd=%E5%8D%81%E8%BF%9B%E5%88%B6%E6%95%B0&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)。例如10010110表示[十进制数](https://www.baidu.com/s?wd=%E5%8D%81%E8%BF%9B%E5%88%B6%E6%95%B0&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)96

**任务4. 数据加密与反跟踪。**

### 3.4.1设计思想及存储单元分配

加密进货价，用户名和密码，主要在登录界面设置反跟踪程序

存储单元分配（只记录与实验三相比有所变化的存储单元）：

BNAME DB 'l' XOR 'D','z' XOR 'O', 'y'XOR 'T', 7 DUP(0);老板姓名

BPASSWORD DB 4 XOR 'r'

DB ('D' - 1BH) \* 2 + 5AH

DB ('O' - 1BH) \* 2 + 5AH

DB ('T' - 1BH) \* 2 + 5AH

DB ('A' - 1BH) \* 2 + 5AH

DB 78H, 1FH;老板密码

GA1-30，GB1-30的进货价进行加密，与‘A’进行异或操作

### 3.4.2流程图

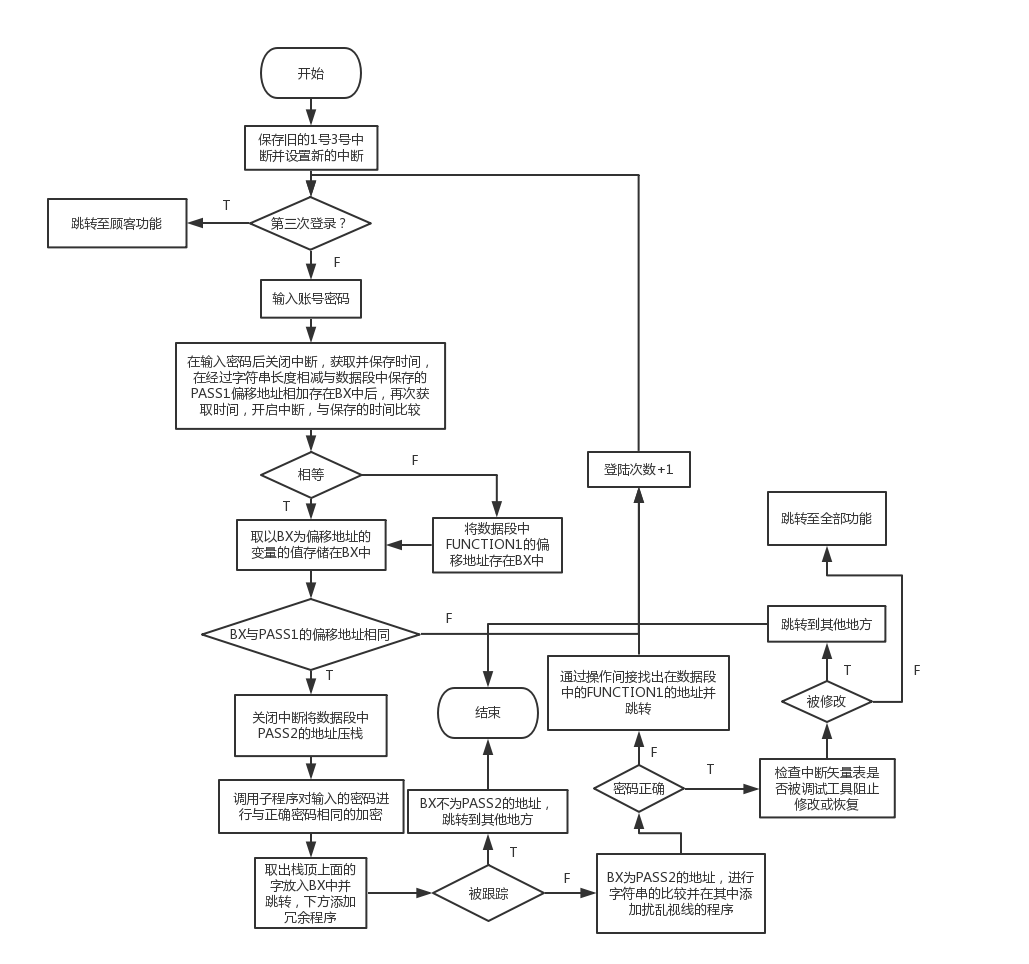


图3.4.2.1 添加了反跟踪后登录功能的流程图

### 3.4.3源程序

.386

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;宏定义

M9 MACRO A

LEA DX,A

MOV AH,9

INT 21H

ENDM ;9号功能调用

M2 MACRO A

MOV DL,A

MOV AH,2

INT 21H

ENDM ;2号功能调用

M10 MACRO A

LEA DX,A

MOV AH,10

INT 21H

ENDM ;10号功能调用

NEWLINE MACRO

MOV DL, 13

MOV AH, 2

INT 21H

MOV DL, 10

MOV AH, 2

INT 21H

ENDM ;换行

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

DATAS SEGMENT USE16

INPUT\_NAME\_MSG DB 0AH,0DH,'Please input your name:$'

INPUT\_PASSWORD\_MSG DB 0AH,0DH,'Please input your passwords:$'

LOGIN\_FAIL\_MSG DB 0AH,0DH,'Login failed,Try Again:$'

GOODNAME\_MSG DB 0AH,0DH,'Please Input The Good Name:$'

GOODNAME\_MSG2 DB 0AH,0DH,'There is no this good in shop1.Please input again:$'

END\_MSG DB 0AH,0DH,'Press anykey to exit:$'

SHOWGOODNAME DB 0AH,0DH,'GoodName:$'

BUYPRICE DB 0AH,0DH,'BuyPrice:$'

PRICE DB 0AH,0DH,'Price:$'

TOTAL DB 0AH,0DH,'Total:$'

SOLD DB 0AH,0DH,'Sold:$'

PROFITRATE DB 0AH,0DH,'ProfitRate:$'

PEN DB 0AH,0DH,'PEN$'

BOOK DB 0AH,0DH,'BOOK$'

SHOP1 DB 0AH,0DH,'SHOP1$'

SHOP2 DB 0AH,0DH,'SHOP2$'

BUYMASSGE DB 0AH,0DH,'Please Input The Number You Want To Buy:$'

SOLDMASSGE DB 0AH,0DH,'OVERSOLD$'

JHJ DB 'BUYING PRICE:$'

XSJ DB 'SALES PRICE:$'

JHL DB 'RETAIL PURCHASES:$'

XSL DB 'SALES VOLUME:$'

SHO DB 'Please enter the shop you choose:$'

FIN DB 'Please enter what you want:$'

S1 DB 'SHOP1',0

GA1 DB 'PEN',7 DUP(0)

DW 35 XOR 'A',56,22,25,?

GA2 DB 'BOOK',6 DUP(0)

DW 12 XOR 'A',30,33,5,?

GA3 DB 'BAG',7 DUP(0)

DW 35 XOR 'A',56,41,25,?

GA4 DB 'SI',8 DUP(0)

DW 35 XOR 'A',56,42,25,?

GA5 DB 'WU',8 DUP(0)

DW 35 XOR 'A',56,43,25,?

GA6 DB 'LIU',7 DUP(0)

DW 35 XOR 'A',56,44,25,?

GA7 DB 'QI',8 DUP(0)

DW 35 XOR 'A',56,45,25,?

GA8 DB 'BA',8 DUP(0)

DW 35 XOR 'A',56,46,25,?

GA9 DB 'JIU',7 DUP(0)

DW 35 XOR 'A',56,47,25,?

GA10 DB 'SHI',7 DUP(0)

DW 35 XOR 'A',56,48,25,?

GA11 DB 'SHIYI',5 DUP(0)

DW 35 XOR 'A',56,49,25,?

GA12 DB 'SHIER',5 DUP(0)

DW 35 XOR 'A',56,50,25,?

GA13 DB 'SHISAN',4 DUP(0)

DW 35 XOR 'A',56,51,25,?

GA14 DB 'SHISI',5 DUP(0)

DW 35 XOR 'A',56,52,25,?

GA15 DB 'SHIWU',5 DUP(0)

DW 35 XOR 'A',56,53,25,?

GA16 DB 'SHILIU',4 DUP(0)

DW 35 XOR 'A',56,54,25,?

GA17 DB 'SHIQI',5 DUP(0)

DW 35 XOR 'A',56,55,25,?

GA18 DB 'SHIBA',5 DUP(0)

DW 35 XOR 'A',56,56,25,?

GA19 DB 'SHIJIU',4 DUP(0)

DW 35 XOR 'A',56,57,25,?

GA20 DB 'ERSHI',5 DUP(0)

DW 35 XOR 'A',56,58,25,?

GA21 DB 'ERYI',6 DUP(0)

DW 35 XOR 'A',56,59,25,?

GA22 DB 'ERER',6 DUP(0)

DW 35 XOR 'A',56,60,25,?

GA23 DB 'ERSAN',5 DUP(0)

DW 35 XOR 'A',56,61,25,?

GA24 DB 'ERSI',6 DUP(0)

DW 35 XOR 'A',56,62,25,?

GA25 DB 'ERWU',6 DUP(0)

DW 35 XOR 'A',56,63,25,?

GA26 DB 'ERLIU',5 DUP(0)

DW 35 XOR 'A',56,64,25,?

GA27 DB 'ERQI',6 DUP(0)

DW 35 XOR 'A',56,65,25,?

GA28 DB 'ERBA',6 DUP(0)

DW 35 XOR 'A',56,66,25,?

GA29 DB 'ERJIU',5 DUP(0)

DW 35 XOR 'A',56,67,25,?

GA30 DB 'SANLING',3 DUP(0)

DW 35 XOR 'A',56,68,25,?

S2 DB 'SHOP2', 0

GB1 DB 'PEN', 7 DUP(0)

DW 35 XOR 'A', 50, 26, 24, ?

GB2 DB 'BOOK', 6 DUP(0)

DW 12 XOR 'A', 28, 27, 15, ?

GB3 DB 'BAG',7 DUP(0)

DW 35 XOR 'A',50,28,24,?

GB4 DB 'SI',8 DUP(0)

DW 35 XOR 'A',56,29,25,?

GB5 DB 'WU',8 DUP(0)

DW 35 XOR 'A',56,30,25,?

GB6 DB 'LIU',7 DUP(0)

DW 35 XOR 'A',56,10000,25,?

GB7 DB 'QI',8 DUP(0)

DW 35 XOR 'A',56,31,25,?

GB8 DB 'BA',8 DUP(0)

DW 35 XOR 'A',56,32,25,?

GB9 DB 'JIU',7 DUP(0)

DW 35 XOR 'A',56,33,25,?

GB10 DB 'SHI',7 DUP(0)

DW 35 XOR 'A',56,34,25,?

GB11 DB 'SHIYI',5 DUP(0)

DW 35 XOR 'A',56,35,25,?

GB12 DB 'SHIER',5 DUP(0)

DW 35 XOR 'A',56,36,25,?

GB13 DB 'SHISAN',4 DUP(0)

DW 35 XOR 'A',56,37,25,?

GB14 DB 'SHISI',5 DUP(0)

DW 35 XOR 'A',56,38,25,?

GB15 DB 'SHIWU',5 DUP(0)

DW 35 XOR 'A',56,39,25,?

GB16 DB 'SHILIU',4 DUP(0)

DW 35 XOR 'A',56,40,25,?

GB17 DB 'SHIQI',5 DUP(0)

DW 35 XOR 'A',56,41,25,?

GB18 DB 'SHIBA',5 DUP(0)

DW 35 XOR 'A',56,42,25,?

GB19 DB 'SHIJIU',4 DUP(0)

DW 35 XOR 'A',56,43,25,?

GB20 DB 'ERSHI',5 DUP(0)

DW 35 XOR 'A',56,44,25,?

GB21 DB 'ERYI',6 DUP(0)

DW 35 XOR 'A',56,45,25,?

GB22 DB 'ERER',6 DUP(0)

DW 35 XOR 'A',56,46,25,?

GB23 DB 'ERSAN',5 DUP(0)

DW 35 XOR 'A',56,47,25,?

GB24 DB 'ERSI',6 DUP(0)

DW 35 XOR 'A',56,48,25,?

GB25 DB 'ERWU',6 DUP(0)

DW 35 XOR 'A',56,49,25,?

GB26 DB 'ERLIU',5 DUP(0)

DW 35 XOR 'A',56,50,25,?

GB27 DB 'ERQI',6 DUP(0)

DW 35 XOR 'A',56,51,25,?

GB28 DB 'ERBA',6 DUP(0)

DW 35 XOR 'A',56,52,25,?

GB29 DB 'ERJIU',5 DUP(0)

DW 35 XOR 'A',56,53,25,?

GB30 DB 'SANLING',3 DUP(0)

DW 35 XOR 'A',56,54,25,?

COUNTN EQU 8

COUNTP EQU 4

N EQU 30

BUFF DB 12 DUP(?)

AUTH DB 0

COST1 DD 0

COST2 DD 0

PROFIT1 DD 0

PROFIT2 DD 0

PR1 DD 0

PR2 DD 0

APR DD 0

APRN DW 30 DUP(0)

ADR DW 30 DUP(0)

DIP DB 6,0,6 DUP(0)

OUTNUM DW 10 DUP(0)

JGF DB '>'

SN DB 6,0,6 DUP(0)

GN DB 10,0,10 DUP(0)

REALNUM DW 0

INPUT DW 0

JISHU DW 0

CHS DW 10

LOC DW 0

LOCB DW 0

LOCR DW 0

COUNT DW 0

M DW 9000

M1 DW 0

LOGCOUNT DW 0

RANK DW 0

IP1 DW 0

IP2 DW 0

INNAME DB 10

DB 0

DB 10 DUP(0)

INPASSWORD DB 7

DB ?

DB 7 DUP(0)

GETGOODNAME DB 10

DB 0

DB 10 DUP(0)

NUMBER DW 10

DW 0

DW 10 DUP(0)

BNAME DB 'l' XOR 'D','z' XOR 'O', 'y'XOR 'T', 7 DUP(0);老板姓名

BPASSWORD DB 4 XOR 'r'

DB ('D' - 1BH) \* 2 + 5AH

DB ('O' - 1BH) \* 2 + 5AH

DB ('T' - 1BH) \* 2 + 5AH

DB ('A' - 1BH) \* 2 + 5AH

DB 78H, 1FH;老板密码

BUF DW 0

BUF1 DW 0

P1 DW PASS1 ;地址表

E1 DW FUNCTION1

P2 DW PASS2

P0 DW FUNCTION1\_1

CAIDAN1 DB 0AH,0DH,'1=SEARCH PRODUCT INFORMATION$'

CAIDAN2 DB 0AH,0DH,'2=MODIFY PRODUCT INFORMATION$'

CAIDAN3 DB 0AH,0DH,'3=CALCULATE AVERAGE PROFIT$'

CAIDAN4 DB 0AH,0DH,'4=CALCULATE PROFIT RATE RANKING$'

CAIDAN5 DB 0AH,0DH,'5=EXPORT ALL PRODUCT INFORMATION$'

CAIDAN6 DB 0AH,0DH,'6=PROGRAME EXIT$'

CAIDAN7 DB 0AH,0DH,'You can just input between 1 to 6!$'

CAIDAN8 DB 0AH,0DH,'Please input your choice:$'

CAIDAN9 DB 0AH,0DH,'You can just input between 1 or 6!$'

OLDINT1 DW 0, 0

OLDINT3 DW 0, 0

DATAS ENDS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

STACKS SEGMENT USE16 STACK 'STACKS'

DB 200 DUP(0)

STACKS ENDS

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CODES SEGMENT USE16 PUBLIC 'CODES'

ASSUME CS:CODES,DS:DATAS,SS:STACKS

START:

MOV AX, DATAS

MOV DS, AX

XOR AX, AX

MOV ES, AX

MOV AX, ES:[1\*4]

MOV OLDINT1, AX

MOV AX, ES:[1\*4+2]

MOV OLDINT1+2, AX

MOV AX, ES:[3\*4]

MOV OLDINT3, AX

MOV AX, ES:[3\*4+2]

MOV OLDINT3+2, AX

CLI

MOV AX, OFFSET NEWINT

MOV ES:[1\*4], AX

MOV ES:[1\*4+2], CS

MOV ES:[3\*4], AX

MOV ES:[3\*4+2], CS

STI

FUNCTION1:;输入用户名和密码

MOV CX, LOGCOUNT

CMP CX, 3

JZ FUNCTION1\_1

MOV SI,2;计数器

INC CX

MOV LOGCOUNT, CX

LEA DX,INPUT\_NAME\_MSG;输入姓名提示

MOV AH,9

INT 21H

LEA DX,INNAME;输入姓名

MOV AH,10

INT 21H

LEA DX,INPUT\_PASSWORD\_MSG;输入密码提示

MOV AH,9

INT 21H

LEA DX,INPASSWORD;输入密码

MOV AH,10

INT 21H

MOV AL,0DH;未输入用户名时跳转至用户模式

CMP AL,INNAME[SI]

JE FUNCTION1\_1

MOV AL,71H;用户名为q时退出

CMP AL,INNAME[SI]

JE Q1

MOV SI,0

JMP FUNCTION2

Q1: MOV AL,0DH;用户名为q时退出

CMP AL,INNAME[SI+1]

JE EXIT

FUNCTION2:;用户名和密码检查

CLI

MOV AH, 2CH

INT 21H

PUSH DX

MOV CL, INPASSWORD+1

XOR CL, 'r'

SUB CL, BPASSWORD

MOVSX BX, CL

ADD BX, OFFSET P1

MOV AH, 2CH

INT 21H

STI

CMP DX, [ESP]

POP DX

JZ OK1

MOV BX, OFFSET E1

OK1: MOV BX, [BX]

CMP BX, PASS1

JZ OK2

JMP E1

OK2: JMP BX

DB 'How to go'

PASS1: MOVZX CX, INPASSWORD+1

CLI

PUSH P2

MOV SI, 0

MOV DL, 2

POP AX

MOV BX, [ESP-2]

STI

CALL JIAMI

JMP BX

DB 'I Do Not Know!'

PASS2: MOVZX BX, BPASSWORD+2

MOV AL, INPASSWORD+2[SI]

ADD BX, 3BH

SUB BX, AX

ADD BX, CX

CMP BL, INPASSWORD+3

CMP AL, BPASSWORD+1[SI]

JNZ ERR2

INC SI

LOOP PASS2

JMP PASS3

ERR2: MOV EBX, OFFSET P1

MOV EDX, 1

JMP WORD PTR [EBX+EDX\*2]

DB 'YES, Get it!'

PASS3: MOV BX, ES:[1\*4]

INC BX

JMP BX

DB 'Now, you see.'

FUNCTION1\_1:

MOV BH,0

MOV BYTE PTR AUTH, BH

JMP BEG0

;;;;;;;;;

BEG0: M9 CAIDAN1

MOV AL, INPASSWORD[5]

MUL DX

MOV BX, AX

M9 CAIDAN6

MOV AX, CX

ADD CX, BX

SUB BX ,AX

MOV DX, CX

ADD DX, BX

M9 CAIDAN8

MOV AH,1

INT 21H

CMP AL,'1'

JE L1

CMP AL,'6'

JE EXIt

M9 CAIDAN9

JMP BEG0

;;;;;;;;;;;;;;

BEG: DB 'No, not that!'

MOVZX DX, BPASSWORD[SI]

MOVZX AX, INPASSWORD[5]

MUL DX

MOV BX, AX

M9 CAIDAN1

XOR AX, BX

M9 CAIDAN2

SUB BX, CX

M9 CAIDAN3

ADD EAX, EBX

M9 CAIDAN4

SUB AX, CX

M9 CAIDAN5

ADD AX, BX

M9 CAIDAN6

SUB BX, DX

M9 CAIDAN8

MOV AH,1

INT 21H

CMP AL,'1'

JE L1

CMP AL,'2'

JE L2

CMP AL,'3'

JE L3

CMP AL,'4'

JE L4

CMP AL,'5'

JE L5

CMP AL,'6'

JE EXIt

M9 CAIDAN7

JMP BEG

L1: CALL FUN1

JMP BEG

L2: CALL FUN2

JMP BEG

L3: CALL FUN3

JMP BEG

L4: CALL FUN4

JMP BEG

L5: CALL FUN5

JMP BEG

;;;;;;;;;;;;;;;;;;;;;;;;;;;

FUN1 PROC

FUNCTION3:

M9 GOODNAME\_MSG

M10 GETGOODNAME

MOV SI,2;输入为回车时返回

MOV AL,0DH

MOV BL,GETGOODNAME[SI]

CMP AL,BL

JE BEG

MOV DX,0

MOV SI,0

MOV CL,GETGOODNAME[SI+1]

FUNCTION3\_0\_1:

MOV DI,0

IMUL SI,DX,20

CMP SI,580

JE FUNCTION3

INC DX

FUNCTION3\_1:

MOV AL,GETGOODNAME[DI+2];输入名称与商品名称比较

MOV BL,GA1[SI]

CMP AL,BL

JNE FUNCTION3\_0\_1;不符合跳转到该商店的下一商品

INC SI

INC DI

DEC CL

JNZ FUNCTION3\_1

MOV CL,GETGOODNAME[1]

DEC CL

MOV CH,0

SUB SI,CX

JMP INFO;符合跳转到该商品的信息

INFO:

DEC DX

MOV JISHU,DX

IMUL SI,DX,20

M9 SHOP1

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHJ

MOV AX,WORD PTR GA1[SI+10]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSJ

MOV AX,WORD PTR GA1[SI+12]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHL

MOV AX,WORD PTR GA1[SI+14]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSL

MOV AX,WORD PTR GA1[SI+16]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 SHOP2

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHJ

MOV AX,WORD PTR GB1[SI+10]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSJ

MOV AX,WORD PTR GB1[SI+12]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 JHL

MOV AX,WORD PTR GB1[SI+14]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

M9 XSL

MOV AX,WORD PTR GB1[SI+16]

MOV DX,16

CALL PRINTAX

RET

FUN1 ENDP

;;;;;;;;;;;;;;;;;;;;;;;;

FUN2 PROC

FST2: LEA DX, SHO

MOV AH, 9

INT 21H

LEA DX, SN

MOV AH, 10

INT 21H

NEWLINE

LEA DX, FIN

MOV AH, 9

INT 21H

LEA DX, GN

MOV AH, 10

INT 21H

NEWLINE

CMP SN+2, 13

JE BACK2

MOV BX, 0

MOV CX, 0

MOV CL, [SN+1]

SCOMP: LEA SI, SN+2

LEA DI, [S1+BX]

CSC: MOV AL, [SI]

CMP AL, [DI]

JNE SALL

INC SI

INC DI

DEC CX

JE SCPL

JMP CSC

GNL: MOV CX, 0

MOV CL, [GN+1]

ADD BX, 6

GCOMP: LEA SI, GN+2

LEA DI, [S1+BX]

CGC: MOV AL, [SI]

CMP AL, [DI]

JNE GALL

INC SI

INC DI

DEC CX

JE GCPL

JMP CGC

MJJ: MOV LOC, BX

LEA DX, JHJ

MOV AH, 9

INT 21H

MOV AX, WORD PTR [S1+BX+10]

XOR AX, 'A'

MOV DX, 16

CALL PRINTAX

PJG: MOV DL, [JGF]

MOV AH, 2

INT 21H

LEA DX, DIP

MOV AH, 10

INT 21H

NEWLINE

MOV BX, 0

LEA DI, DIP+2

CTS: MOV AX, 0

MOV CX, 0

MOV CL, [DIP+1]

DEC CX

SUB CX, BX

MOV AL, [DI+BX]

CMP AL, '0'

JL MJJ

CMP AL, '9'

JG MJJ

TTS: SUB AX, 30H

CMP CX, 0

JE TAD

IMUL AX, 10

DEC CX

JNE TTS

TAD: MOV DX, COUNT

ADD DX, AX

MOV COUNT, DX

INC BX

CMP BL, BYTE PTR [DIP+1]

JNE CTS

LJJ: MOV BX, LOC

MOV AX, COUNT

XOR AX, 'A'

MOV WORD PTR [S1+BX+10], AX

MOV COUNT, 0

MXJ: MOV LOC, BX

LEA DX, XSJ

MOV AH, 9

INT 21H

MOV AX, WORD PTR [S1+BX+12]

MOV DX, 16

CALL PRINTAX

MOV DL, [JGF]

MOV AH, 2

INT 21H

LEA DX, DIP

MOV AH, 10

INT 21H

NEWLINE

MOV BX, 0

LEA DI, DIP+2

CTS2: MOV AX, 0

MOV CX, 0

MOV CL, [DIP+1]

DEC CX

SUB CX, BX

MOV AL, [DI+BX]

CMP AL, '0'

JL MXJ

CMP AL, '9'

JG MXJ

TTS2: SUB AX, 30H

CMP CX, 0

JE TAD2

IMUL AX, 10

DEC CX

JNE TTS2

TAD2: MOV DX, COUNT

ADD DX, AX

MOV COUNT, DX

INC BX

CMP BL, BYTE PTR [DIP+1]

JNE CTS2

LXJ: MOV BX, LOC

MOV AX, COUNT

MOV WORD PTR [S1+BX+12], AX

MOV COUNT, 0

MJL: MOV LOC, BX

LEA DX, JHL

MOV AH, 9

INT 21H

MOV AX, WORD PTR [S1+BX+14]

MOV DX, 16

CALL PRINTAX

MOV DL, [JGF]

MOV AH, 2

INT 21H

LEA DX, DIP

MOV AH, 10

INT 21H

NEWLINE

MOV BX, 0

LEA DI, DIP+2

CTS3: MOV AX, 0

MOV CX, 0

MOV CL, [DIP+1]

DEC CX

SUB CX, BX

MOV AL, [DI+BX]

CMP AL, '0'

JL MJL

CMP AL, '9'

JG MJL

TTS3: SUB AX, 30H

CMP CX, 0

JE TAD3

IMUL AX, 10

DEC CX

JNE TTS3

TAD3: MOV DX, COUNT

ADD DX, AX

MOV COUNT, DX

INC BX

CMP BL, BYTE PTR [DIP+1]

JNE CTS3

LJL: MOV BX, LOC

MOV AX, COUNT

MOV WORD PTR [S1+BX+14], AX

MOV COUNT, 0

JMP BACK2

SALL: CMP BX, 1210

JE FST2

ADD BX, 605

JMP SCOMP

SCPL: CMP BYTE PTR [DI], 0

JE GNL

JMP SALL

GALL: CMP BX, 600

JE FST2

ADD BX, 20

JMP GCOMP

GCPL: CMP BYTE PTR [DI], 0

JE MJJ

JMP GALL

BACK2: RET

FUN2 ENDP

FUN3 PROC

MOV BX, 0

BAP: MOV EAX, 0

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [GA1+BX+10]

XOR AX, 'A'

MOV CX, WORD PTR [GA1+BX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [GA1+BX+12]

MOV DX, WORD PTR [GA1+BX+16]

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR1, EAX

MOV LOC, BX

MOV CX, 0

MOV BX, LOC

LEA DI, [GA1+BX]

MOV BX, 0

LEA SI, [GB1+BX]

COMP\_3: MOV CL, 10

LEA SI, [GB1+BX]

MOV LOCB, BX

MOV BX, LOC

LEA DI, [GA1+BX]

MOV BX, LOCB

RCM: MOV AL, [DI]

CMP AL, [SI]

JNE ALL

INC DI

INC SI

DEC CX

JE CCOP

JMP RCM

CCOP: MOV EAX, 0

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [GB1+BX+10]

MOV CX, WORD PTR [GB1+BX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [GB1+BX+12]

MOV DX, WORD PTR [GB1+BX+16]

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR2, EAX

MOV EDX, PR1

ADD EAX, EDX

SAR EAX, 1

MOV APR, EAX

MOV BX, LOC

MOV WORD PTR [GA1+BX+18], AX

ADD BX, 20

CMP BX, 600

JE BACK3

JMP BAP

ALL: ADD BX,20

JMP COMP\_3

BACK3: RET

FUN3 ENDP

FUN4 PROC

MOV CX,-1

LOOPX:

INC CX

IMUL BX,CX,20

MOV AX,WORD PTR GA1+18[BX]

PUSH CX

PUSH BX

MOV CX,-1

MOV RANK,1

LOOPY:

INC CX

IMUL BX,CX,20

CMP AX,WORD PTR GA1+18[BX]

JL LARANK

L3P:

CMP CX,N-1

JGE L352

JMP LOOPY

L352:

POP BX

POP CX

JMP L351

LARANK:

INC RANK

JMP L3P

L351:

MOV AX,RANK

MOV WORD PTR GB1+18[BX],AX

CMP CX,N-1

JGE LLL

JMP LOOPX

LLL: RET

FUN4 ENDP

PRINTAX PROC

PUSH EBX

PUSH SI

LEA SI,BUFF

CMP DX,32

JE L

MOVSX EAX,AX

L: OR EAX,EAX

JNS PLUS

NEG EAX

MOV BYTE PTR [SI],'-'

INC SI

PLUS: MOV EBX,10

CALL RDAIX

MOV BYTE PTR [SI],'$'

LEA DX,BUFF

MOV AH,9

INT 21H

POP SI

POP EBX

RET

PRINTAX ENDP

JIAMI PROC

PUSH SI

PUSH DX

PUSH CX

MOV SI, 0

MOV DL, 2

MOVZX CX, INPASSWORD+1

AG: MOVZX AX, INPASSWORD+2[SI]

SUB AX, 1BH

MUL DL

ADD AX, 5AH

MOV INPASSWORD+2[SI], AL

INC SI

LOOP AG

POP CX

POP DX

POP SI

RET

JIAMI ENDP

RDAIX PROC

PUSH CX

PUSH EDX

XOR CX,CX

LOP1: XOR EDX,EDX

DIV EBX

PUSH DX

INC CX

OR EAX,EAX

JNZ LOP1

LOP2: POP AX

CMP AL,10

JB LL1

ADD AL,7

LL1: ADD AL,30H

MOV [SI],AL

INC SI

LOOP LOP2

POP EDX

POP CX

RET

RDAIX ENDP

FUN5 PROCF1:

MOV DI,0

LL:

IMUL SI,DI,20

MOV BYTE PTR GA1[SI+9],'$'

MOV BYTE PTR GB1[SI+9],'$'

M9 GA1[SI]

MOV DL,32

MOV AH,2

INT 21H

MOV AX,WORD PTR GA1[SI+18]

MOV DX,16

CALL PRINTAX

MOV DL,32

MOV AH,2

INT 21H

MOV AX,WORD PTR GB1[SI+18]

MOV DX,16

CALL PRINTAX

MOV DL,0AH

MOV AH,2

INT 21H

INC DI

CMP DI,30

JNE LL

RET

FUN5 ENDP

NEWINT: IRET

TESTINT: JMP BEG

EXIT : CLI

MOV AX, OLDINT1

MOV ES:[1\*4], AX

MOV AX, OLDINT1+2

MOV ES:[1\*4+2], AX

MOV AX, OLDINT3

MOV ES:[3\*4], AX

MOV AX, OLDINT3+2

MOV ES:[3\*4+2], AX

STI

MOV AH,4CH

INT 21H

CODES ENDS

END START

### 3.4.4实验步骤

1. 准备上机实验环境、编辑、汇编、连接。

2. 执行4-1程序，输入三次错误的用户名密码，观察执行结果。

3. 执行4-1程序，输入正确的用户名密码，观察执行结果。

4. 通过TD单步调试追踪4-1，观察各个反跟踪程序的执行效果。

### 3.4.5实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

2. 汇编过程没有发生异常

3. 连接过程没有发生异常。

4. 三次输入登录错误，进去顾客查询界面，如下图所示：

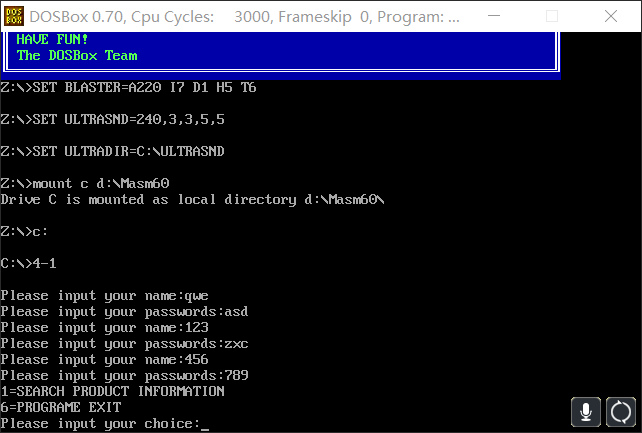


图3.4.5.1 顾客查询界面

1. 登录成功，显示完整的功能界面，如下图所示：

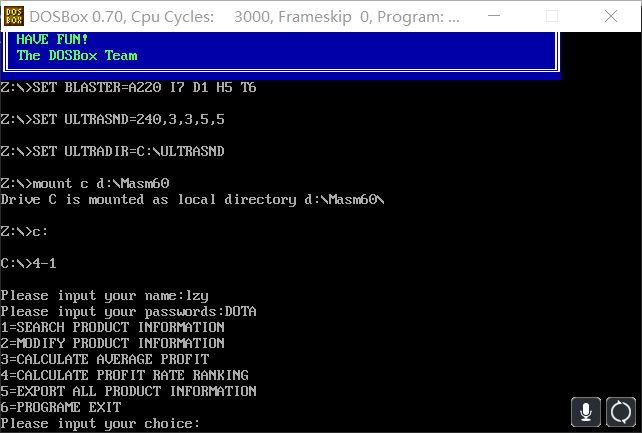


图3.4.5.2 完整的功能界面

1. 用TD单步调试该程序，在修改中断部分TD会卡死，如下图所示：

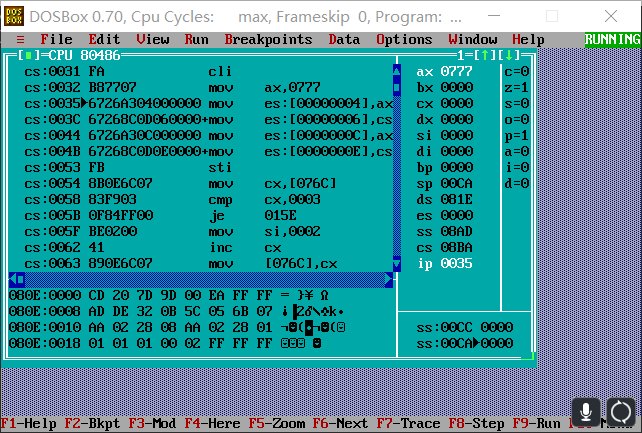


图3.4.5.3 修改中断部分

1. 跳过中断程序，在时间记录反跟踪的作用下，BX与PASS1的偏移地址不同，使程序跳转回用户名密码输入之前，同时通过记录偏移地址进行间接跳转的方式，使程序更加难以理解，如下图所示：

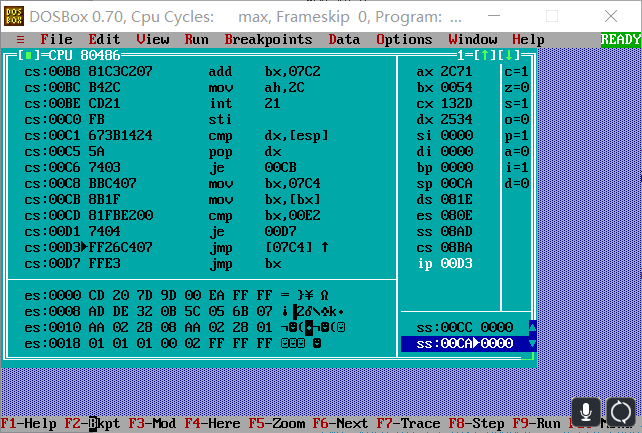


图3.4.5.4 计时跳转反跟踪

1. 在程序中添加了冗余信息后，在TD中反汇编出了几条与程序无关的语句，扰乱视听，如下图所示：

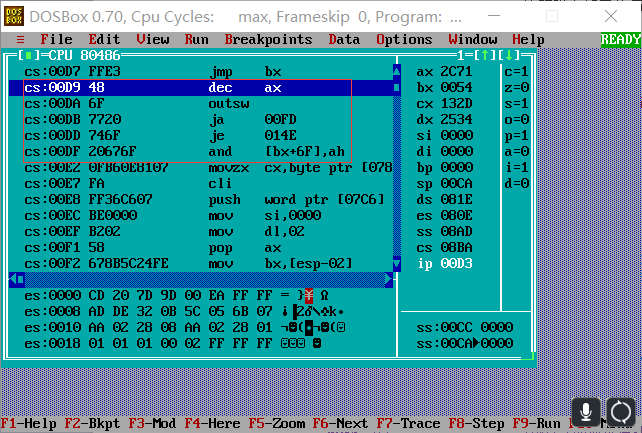
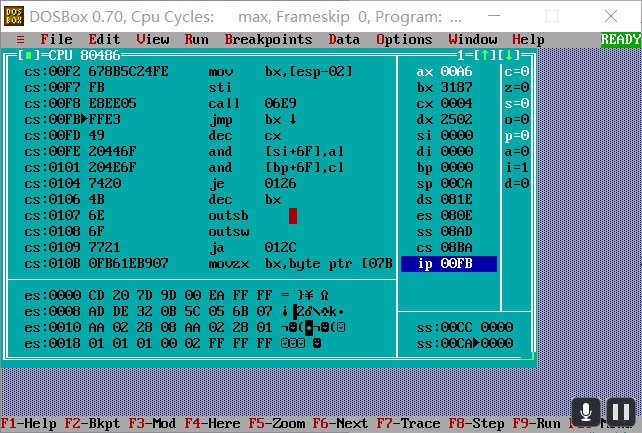


图3.4.5.5 冗余信息添加

1. 在堆栈反跟踪作用下，程序跳转到未知的位置，使程序无法正常运行，如下图所示：



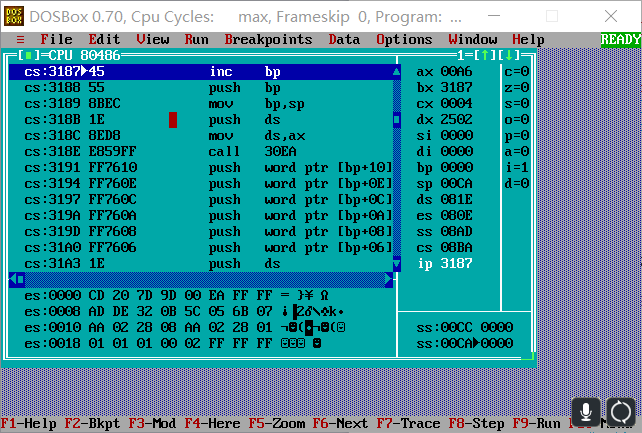
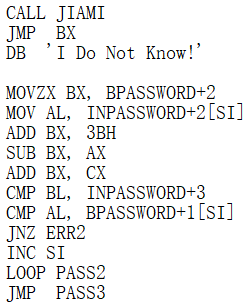


图3.4.5.6 堆栈反跟踪

1. 本程序将对输入密码的加密操作放在了子程序中，并放堆栈反跟踪的跳转之前，在后面比较阶段加入了混淆视听的无关操作，让该程序的加密算法更加难以被发现，如下图所示：



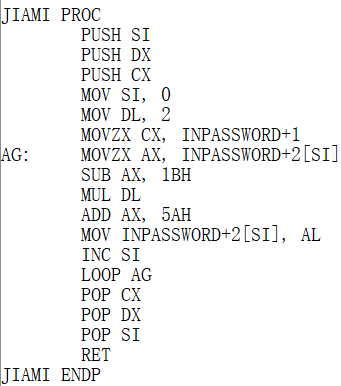


图3.4.5.7 加密算法反跟踪

### 3.5任务5：跟踪与数据解密

### 3.5.1设计思想及存储单元分配

无

### 3.5.2流程图

无

### 3.5.3源程序

无

### 3.5.4实验步骤

1. 使用TD调试同组同学的加密程序。

2. 单步执行每一条语句，找出其中反跟踪的操作（使程序崩溃，或无法继续向下执行，或跳转到未知的地方等等）进行跳过，最终找出其账号密码的加密方法以及进货价的加密方法。

### 3.5.5实验记录与分析

1. 实验环境条件：i7-6700 2.6GHz，16G内存；WINDOWS 10下DOSBox 0.70； TD.EXE 。

2. 解密过程

（1）运行td进入调试界面

（2）反汇编代码中，载入数据段操作后由明显的中断修改代码





图3.5.5.1 中断反跟踪

1. 通过修改ip跳过中断修改部分



图3.5.5.2 修改IP跳转

1. 运行以下程序，随意输入用户名和密码

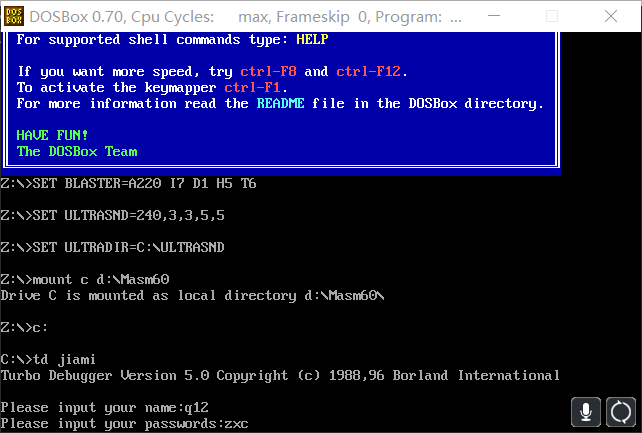
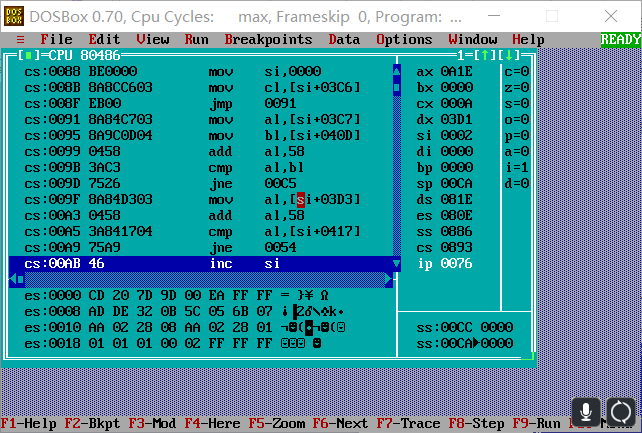


图3.5.5.3 登录

1. 返回到td，在经过与特殊情况的比较后，可以通过输入信息存储的地址以及循环比较看出在进行账号密码的比较：



B7JII_7_J84(X503K_{}BDQ

图3.5.5.4 账号密码加密算法

1. 通过对输入数据的加密操作，可以看出密码的加密方法：+58H，同时观察与之比较的数据的地址，可以找到账号密码所存储的位置，跳转至该位置：

1G_YF6$PS(92TMG%]2}]1EY

[B[[HXTMEM)@2(5X$S83R71

M(]91]_I70UD[PD6(WC[NJC

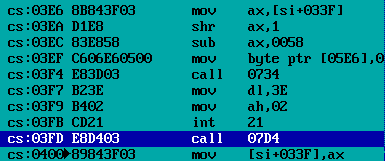
图3.5.5.5 数据段中的账号密码

1. 对于这些数据进行解密操作，可以得出账号密码：XULEI，XULEI，重新调试该程序，输入解密的账号密码进行验证，发现正确
2. 验证通过后，执行如下代码会发生无意义的跳转，仔细观察后可以看出该代码为中断对应的反跟踪操作，同样用修改ip的方式跳过：

T8EWQZ60VG$NDG8LU{4IB56

图3.5.5.6 修改IP跳过

1. 之后选择需要对进价解密的功能2，观察输出解密后进价之前的操作，可以看出进价的加密方式及地址：



MZ40HBQJMGC0N7K}D9C(_}7

图3.5.5.7 进货价的加密算法及数据段中的进货价

通过其解密方法：SHR AX，1 ，SUB AX,0058，可以看出其加密方式为：加58H，左移一位。通过地址找到的加密过后的进货价E8H在通过解密后为：28，经验证正确，如下图所示：

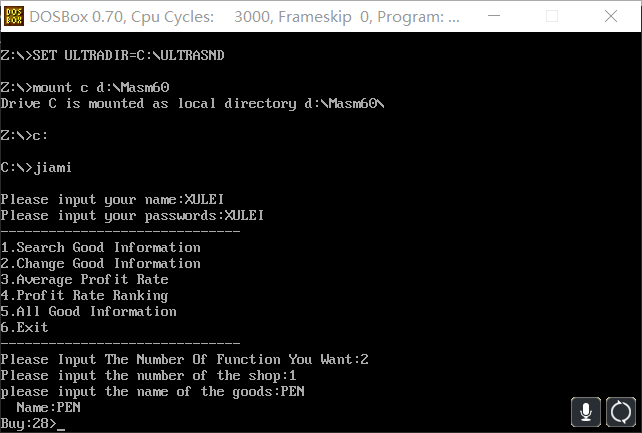


图3.5.5.8 进货价验证

3 .思考题：

1. 运行td在数据段中寻找像是密码的数据，进行尝试暴力破解。
2. 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。
3. 对于密码以明文方式存放在待破解程序中的程序，可以利用文件编辑工具直接打开该待破解程序的文件并查看里面的内容，找到疑似密码的字符串，再通过实际测试，最终确定真实的密码。对于仅仅需要输入密码且没有出错次数限制的程序，可以采用暴力破解密码的方法来获取密码，也即编写一个程序，自动调用与执行待破解的程序，并按照枚举方式自动给待破解程序输入可能的密码，直至密码正确为止。
4. 通过静态反汇编确认部分无需注意的非反跟踪程序以及当前代码所执行的程序的功能，通过动态跟踪调试观察不能确认的代码。
5. 如上述实验记录中的，明显的中断修改程序（即有一定反跟踪程序设计经验，可以一眼看出的反跟踪程序），在单步执行某段代码后跳转至未知的地方或是陷入循环等等非正常运行情况，且与程序实现功能并无关系时，就需要观察刚刚执行的代码是否是反跟踪的代码，应对反跟踪代码可以选择直接跳过不执行的方式，即修改ip指定程序下一步运行的位置，从而避过反跟踪程序。
6. 汇编语言在进行调试时，会对中断，堆栈等很多数据产生影响，改变其值，而在运行时并不会，且调试时产生的反汇编代码，而不是源代码很可能被冗余刻意添加的无用代码所影响，也会使程序更加难以理解，加大了代码破译的难度，而在C语言中我并不知道通过外部反编译会对程序带来什么直接的影响，或是并不会带来什么影响，所以不知道如何用C语言实现反跟踪。

# 4 总结与体会

这次任务是在于对中断程序的理解和修改，CMOS内信息的读取，加密反跟踪与跟踪解密，与之前设计的程序有所不同，不需要太多逻辑上的思考，主要是熟悉中断及CMOS的相关操作，以及初步了解加密，反跟踪，跟踪解密的相关方法及步骤。

任务一，二中，是对于中断程序的相关操作，之前从没进行过相关的操作，上课讲解这方面内容的时候，我也只是一知半解，而通过这次实验以及对书的进一步解读，我对中断程序，中断矢量表等有了进一步或者说整体的理解，同时通过书上的实例我也是对修改中断的程序在功能及含义上都有了一定的了解。

任务三是对CMOS内部信息的操作，虽然对于相关知识掌握地并不好，但是通过书上的实例也基本掌握了通过IN/OUT从CMOS内部读取信息的技巧，并成功完成了实验，花费时间更多的是对读取出的数据的处理和进制转换，这一部分我一直没能很好地掌握，也因此出现了不少bug。

任务四，五则是之前也从来没有接触过的数据的加密反跟踪以及跟踪解密，之前学习的C语言并没有对应的反编译程序或软件，也因此从未接触过加解密方面的内容，对汇编特殊的加解密操作也让我进一步了解了调试软件在调试过程中对内存中数据的修改，通过实验我也基本掌握了相关方面的知识与技巧。

本次实验的内容很多都是我从未接触过的内容，我再次提高了自己对于汇编程序的编写熟练度和经验，以及对汇编和调试反汇编的进一步理解，此次实验的内容虽然陌生但是在我们以后的学习工作过程中大有用处，我在课下的练习学习过程中也会重视相关的内容。

# 参考文献

[1] 作者：王元珍 曹忠升 韩宗芬，书名：80X86汇编语言程序设计，出版者：华中科技大学出版社。



**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验名称： 实验五 WIN32编程**

**实验时间： 2018-5-24，8：00-11：50 实验地点： 南一楼807室**

**指导教师： 周英飚**

**专业班级：计算机卓越工程师班1601班**

**学 号： U201612922 姓 名： 刘子煜**

**同组学生： 无 报告日期： 2018年 5 月 24日**

**原创性声明**

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

特此声明！

学生签名：

日期：2018.5.24

成绩评定

|  |  |  |
| --- | --- | --- |
| 实验完成质量得分（70分）（实验步骤清晰详细深入，实验记录真实完整等） | 报告撰写质量得分（30分）（报告规范、完整、通顺、详实等） | 总成绩（100分） |
|  |  |  |

指导教师签字：

                     日期：

**目录**

[1 实验目的与要求 1](#_Toc476742634)

[2 实验内容 1](#_Toc476742635)

[3 实验过程 2](#_Toc476742636)

[3.1 设计思想及存储单元分配 2](#_Toc476742641)

[3.2 流程图 3](#_Toc476742641)

[3.3 源程序 5](#_Toc476742642)

[3.4 实验步骤 1](#_Toc476742643)5

[3.5 实验记录与分析 1](#_Toc476742644)5

[4 总结与体会 2](#_Toc476742646)1

[参考文献 2](#_Toc476742647)2

# 1 实验目的与要求

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

（1）熟悉WIN32程序的设计和调试方法；

（2）熟悉宏汇编语言中INVOKE、结构变量、简化段定义等功能；

（3）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。

# 2 实验内容

编写一个基于窗口的WIN32程序，实现网店商品信息管理程序的平均利润率计算及商品信息显示的功能（借鉴实验三的一些做法），具体要求如下描述。

功能一：编写一个基于窗口的WIN32程序的菜单框架，具有以下的下拉菜单项：

File Action Help

Exit Average About

List

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Average、List将分别实现计算平均利润率或显示SHOP1所有商品信息的功能（详见功能二的描述）。



图5.1 菜单示例

功能二：要求采用结构变量存放商品的相关信息。商品数至少定义5种。

1. 点菜单项Average时，按照实验三的方法计算所有商品的平均利润率。用TD32观察计算结果。
2. 点菜单项List时，要求能在窗口中列出SHOP1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

# 3 实验过程

### 3.1设计思想及存储单元分配

将商品信息以及提示信息等放入数据段中，在窗口中创建计算，列出信息，退出和弹出相关信息窗口的子菜单选项，编写相关的计算，信息输出转换程序，在选择相关选项时进行调用并配合API函数进行输出。

1.结构体设计

goods struct

Goodname ：商品名称

Buyprice ：商品进价

price ：商品售价

total ：商品总量

sold ：商品销量

profitrate ：利润率

goods ends

2.存储单元分配

OldCommand：记录窗口接受到的指令信息。

ClassName：注册类名称。

AppName：窗口名称。

ComputeTip：计算提示字符串。

MenuName：菜单名称。

AboutMsg：相关信息字符串。

hInstance：应用程序当前实例的句柄。

CommandLine：指向应用程序命令行的字符串的[指针](https://baike.baidu.com/item/%E6%8C%87%E9%92%88)。

buf：商品信息。

S1：商店名称。

FENGE：分隔符。

SHOPGOODNAME：商品名称提示字符串。

BUYPRICE：进价提示字符串。

PRICE：售价提示字符串。

TOTAL：总量提示字符串。

SOLD：销售了字符串。

PROFITRATE：利润率字符串。

p\_buyprice：转换后的进价字符串，第一位为长度

p\_price：转换后的售价字符串，第一位为长度

p\_total：转换后的总量字符串，第一位为长度

p\_sold：转换后的销售量字符串，第一位为长度

p\_profitrate：转换后的利润率字符串，第一位为长度

2.寄存器分配

CX：计数器。

AX、AL、BX、DX、SI、DI、EAX、EBX、ECX、EDX：临时寄存器。

### 3.2流程图

(求平均利润率子程序流程图与实验三中求平均利润率基本相同，寄存器及本程序特有LOCAL变量使用有所变化，但逻辑流程相同，在此不再展示）

1.图3.2.1是窗口程序相应流程图。

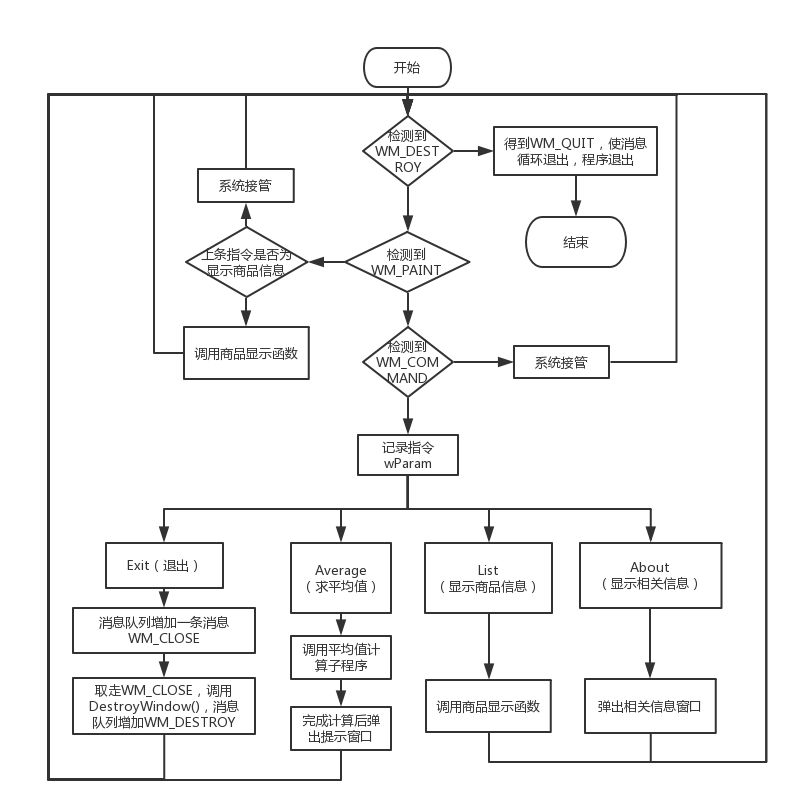


图3.2.1 窗口程序相应流程图

2.图3.2.2是商品显示函数相应流程图。（其中转换函数除在所给地址的变量的第一位记录长度以外，与前面实验所使用的转换函数逻辑流程相同）

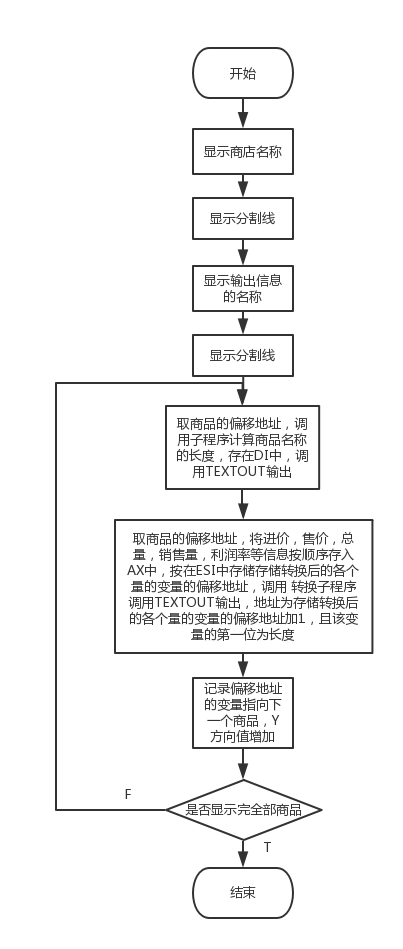


图3.2.2 商品显示函数流程图

### 3.3源程序

demo.asm:

.386

.model flat,stdcall

option casemap:none

WinMain proto :DWORD,:DWORD,:DWORD,:DWORD

WndProc proto :DWORD,:DWORD,:DWORD,:DWORD

Display proto :DWORD

include menuID.INC

include windows.inc

include user32.inc

include kernel32.inc

include gdi32.inc

include shell32.inc

includelib user32.lib

includelib kernel32.lib

includelib gdi32.lib

includelib shell32.lib

goods struct

goodname db 10 dup(0)

buyprice dw 0

price dw 0

total dw 0

sold dw 0

profitrate dw 0

goods ends

.data

OldCommand dd 0

ClassName db 'TryWinClass',0

AppName db 'Our First Window',0

ComputeTip db '计算完成',0

MenuName db 'MyMenu',0

DlgName db 'MyDialog',0

AboutMsg db '我是刘子煜',0

hInstance dd 0

CommandLine dd 0

buf goods <>

goods <'PEN',35,56,30,25,?>

goods <'BOOK',12,30,38,35,?>

goods <'BAG',35,56,41,35,?>

goods <'SI',35,56,42,35,?>

goods <'WU',35,56,43,35,?>

goods <'LIU',35,56,44,38,?>

goods <'QI',35,56,45,36,?>

goods <'BA',35,56,46,39,?>

goods <'JIU',35,56,47,40,?>

goods <'SHI',35,56,48,45,?>

goods <'PEN',35,50,26,24,?>

goods <'BOOK',12,28,27,25,?>

goods <'BAG',35,50,100,24,?>

goods <'SI',35,56,29,25,?>

goods <'WU',35,56,30,25,?>

goods <'LIU',35,56,36,29,?>

goods <'QI',35,56,31,25,?>

goods <'BA',35,56,32,28,?>

goods <'JIU',35,56,33,27,?>

goods <'SHI',35,56,34,30,?>

goods 3 dup(<>)

S1 DB 'SHOP1',0

FENGE DB '--------------------------------------------------------------------------------------------',0

SHOPGOODNAME DB 'GoodName', 0

BUYPRICE DB 'BuyPrice', 0

p\_buyprice DB 10 DUP(0)

PRICE DB 'Price', 0

p\_price DB 10 DUP(0)

TOTAL DB 'Total', 0

p\_total DB 10 DUP(0)

SOLD DB 'Sold', 0

p\_sold DB 10 DUP(0)

PROFITRATE DB 'ProfitRate', 0

p\_profitrate DB 10 DUP(0)

.code

Start: invoke GetModuleHandle,NULL

mov hInstance,eax

invoke GetCommandLine

mov CommandLine,eax

invoke WinMain,hInstance,NULL,CommandLine,SW\_SHOWDEFAULT

invoke ExitProcess,eax

;;

WinMain proc hInst:DWORD,

hPrevInst:DWORD,

CmdLine:DWORD,

CmdShow:DWORD

LOCAL wc:WNDCLASSEX

LOCAL msg:MSG

LOCAL hWnd:HWND

invoke RtlZeroMemory,addr wc,sizeof wc

mov wc.cbSize,SIZEOF WNDCLASSEX

mov wc.style, CS\_HREDRAW or CS\_VREDRAW

mov wc.lpfnWndProc, offset WndProc

mov wc.cbClsExtra,NULL

mov wc.cbWndExtra,NULL

push hInst

pop wc.hInstance

mov wc.hbrBackground,COLOR\_WINDOW+1

mov wc.lpszMenuName, offset MenuName

mov wc.lpszClassName,offset ClassName

invoke LoadIcon,NULL,IDI\_APPLICATION

mov wc.hIcon,eax

mov wc.hIconSm,0

invoke LoadCursor,NULL,IDC\_ARROW

mov wc.hCursor,eax

invoke RegisterClassEx, addr wc

INVOKE CreateWindowEx,NULL,addr ClassName,addr AppName,\

WS\_OVERLAPPEDWINDOW,CW\_USEDEFAULT,\

CW\_USEDEFAULT,CW\_USEDEFAULT,CW\_USEDEFAULT,NULL,NULL,\

hInst,NULL

mov hWnd,eax

INVOKE ShowWindow,hWnd,SW\_SHOWNORMAL

INVOKE UpdateWindow,hWnd

;;

MsgLoop: INVOKE GetMessage,addr msg,NULL,0,0

cmp EAX,0

je ExitLoop

INVOKE TranslateMessage,addr msg

INVOKE DispatchMessage,addr msg

jmp MsgLoop

ExitLoop: mov eax,msg.wParam

ret

WinMain endp

WndProc proc hWnd:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD

LOCAL hdc:HDC

mov eax, OldCommand

.IF uMsg == WM\_DESTROY

invoke PostQuitMessage,NULL

.ELSEIF uMsg == WM\_KEYDOWN

.IF wParam == VK\_F1

;;your code

.ENDIF

.ELSEIF uMsg == WM\_COMMAND

.IF wParam == IDM\_FILE\_EXIT

invoke SendMessage,hWnd,WM\_CLOSE,0,0

.ELSEIF wParam == IDM\_ACTION\_LIST

mov eax, wParam

mov OldCommand, eax

invoke Display,hWnd

.ELSEIF wParam == IDM\_ACTION\_AVERAGE

CALL Computa

invoke MessageBox,hWnd,addr ComputeTip,addr AppName,MB\_OK

.ELSEIF wParam == IDM\_HELP\_ABOUT

invoke MessageBox,hWnd,addr AboutMsg,addr AppName,MB\_OK

.ENDIF

.ELSEIF uMsg == WM\_PAINT

mov eax, OldCommand

.IF eax == IDM\_ACTION\_LIST

invoke Display,hWnd

.ENDIF

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

.ELSE

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

ret

.ENDIF

xor eax,eax

ret

WndProc endp

Display proc hWnd:DWORD

XX equ 10

YY equ 10

XX\_GAP equ 150

YY\_GAP equ 30

FG\_GAP equ 50

LOCAL hdc:HDC

LOCAL numb: DWORD

LOCAL adr: DWORD

invoke GetDC,hWnd

mov hdc,eax

mov eax, 0

mov numb, 130

lea ecx, buf[1\*20]

mov adr, ecx

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+0\*YY\_GAP,offset S1,5

invoke TextOut,hdc,XX+0\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+1\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+2\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+3\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+4\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+5\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+6\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+7\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+8\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+9\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+10\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+11\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+12\*FG\_GAP,YY+1\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+0\*XX\_GAP,YY+2\*YY\_GAP,offset SHOPGOODNAME,8

invoke TextOut,hdc,XX+1\*XX\_GAP,YY+2\*YY\_GAP,offset BUYPRICE,8

invoke TextOut,hdc,XX+2\*XX\_GAP,YY+2\*YY\_GAP,offset PRICE,5

invoke TextOut,hdc,XX+3\*XX\_GAP,YY+2\*YY\_GAP,offset TOTAL,5

invoke TextOut,hdc,XX+4\*XX\_GAP,YY+2\*YY\_GAP,offset SOLD,4

invoke TextOut,hdc,XX+5\*XX\_GAP,YY+2\*YY\_GAP,offset PROFITRATE, 10

invoke TextOut,hdc,XX+0\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+1\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+2\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+3\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+4\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+5\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+6\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+7\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+8\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+9\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+10\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+11\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

invoke TextOut,hdc,XX+12\*FG\_GAP,YY+3\*YY\_GAP,offset FENGE,50

GOP:

MOV ECX, adr

CALL Leng

invoke TextOut,hdc,XX+0\*XX\_GAP,numb,adr,DI

MOV ECX, adr

MOV AX, [ECX+10]

MOV ESI, OFFSET p\_buyprice

CALL Tentoasc

invoke TextOut,hdc,XX+1\*XX\_GAP,numb,offset p\_buyprice+1,[p\_buyprice]

MOV ECX, adr

MOV AX, [ECX+12]

MOV ESI, OFFSET p\_price

CALL Tentoasc

invoke TextOut,hdc,XX+2\*XX\_GAP,numb,offset p\_price+1,[p\_price]

MOV ECX, adr

MOV AX, [ECX+14]

MOV ESI, OFFSET p\_total

CALL Tentoasc

invoke TextOut,hdc,XX+3\*XX\_GAP,numb,offset p\_total+1,[p\_total]

MOV ECX, adr

MOV AX, [ECX+16]

MOV ESI, OFFSET p\_sold

CALL Tentoasc

invoke TextOut,hdc,XX+4\*XX\_GAP,numb,offset p\_sold+1,[p\_sold]

MOV ECX, adr

MOV AX, [ECX+18]

MOV ESI, OFFSET p\_profitrate

CALL Tentoasc

invoke TextOut,hdc,XX+5\*XX\_GAP,numb,offset p\_profitrate+1,[p\_profitrate]

add numb, 30

add adr, 20

cmp numb, 430

jnz GOP

ret

Display endp

Leng proc

MOV DI, 0

CONT: MOV AX, [ECX]

CMP AX, 0

JE EXIT\_L

INC ECX

INC DI

JMP CONT

EXIT\_L: ret

Leng endp

Tentoasc proc

PUSH CX

PUSH DX

PUSH BX

MOV CX, 0

MOV BX, 10

OR AX , AX

JNS LOP1

NEG AX

MOV DI, 1

INC CX

LOP1: MOV DX, 0

DIV BX

PUSH DX

INC CX

OR AX, AX

JNZ LOP1

MOV [ESI], CX

INC ESI

CMP DI, 1

JNZ LOP2

DEC CX

MOV BYTE PTR [ESI], '-'

INC ESI

LOP2: POP AX

CMP AL, 10

JB L1

ADD AL, 7

L1: ADD AL, 30H

MOV [ESI], AL

INC ESI

DEC CX

JNZ LOP2

POP BX

POP DX

POP CX

ret

Tentoasc endp

Computa proc

N equ 10

LOCAL PR1 : DWORD

LOCAL adr\_a : DWORD

LOCAL adr\_b : DWORD

LOCAL count : DWORD

PUSH EAX

PUSH EBX

PUSH ECX

PUSH EDX

LEA eax, buf[1\*20]

MOV adr\_a, eax

MOV BX, 0

MOV count, 0

BAP: mov ebx, adr\_a

LEA eax, buf[11\*20]

MOV adr\_b, eax

MOV EAX, 0

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [ebx+10]

MOV CX, WORD PTR [ebx+14]

IMUL ECX, EAX

MOV AX, WORD PTR [ebx+12]

MOV DX, WORD PTR [ebx+16]

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV PR1, EAX

MOV CX, 0

COMP\_3: MOV CL, 10

MOV EBX, adr\_b

MOV EAX, EBX

MOV EBX, adr\_a

MOV EDX, EBX

RCM: MOV DI, [EAX]

MOV SI, [EDX]

CMP DI, SI

JNE CPLT

INC EAX

INC EDX

DEC CX

JE CCOP

JMP RCM

CCOP: MOV EBX, adr\_b

MOV EAX, 0

MOV ECX, 0

MOV EDX, 0

MOV AX, WORD PTR [EBX+10]

MOV CX, WORD PTR [EBX+14]

IMUL ECX, EAX

MOV AX, WORD PTR [EBX+12]

MOV DX, WORD PTR [EBX+16]

IMUL EAX, EDX

SUB EAX, ECX

IMUL EAX, 100

CDQ

IDIV ECX

MOV EDX, PR1

ADD EAX, EDX

SAR EAX, 1

MOV EBX, adr\_a

MOV WORD PTR [EBX+18], AX

ADD adr\_a, 20

ADD count, 1

CMP count, N

JE BACK3

JMP BAP

CPLT: ADD adr\_b,20

JMP COMP\_3

BACK3: POP EDX

POP ECX

POP EBX

POP EAX

RET

Computa endp

end Start

menu.rc:

#define IDM\_FILE\_EXIT 10001

#define IDM\_ACTION\_AVERAGE 10101

#define IDM\_ACTION\_LIST 10102

#define IDM\_HELP\_ABOUT 10201

MyMenu MENU

BEGIN

POPUP "&File"

BEGIN

MENUITEM "E&xit",IDM\_FILE\_EXIT

END

POPUP "&Action"

BEGIN

MENUITEM "&Average",IDM\_ACTION\_AVERAGE

MENUITEM "&List",IDM\_ACTION\_LIST

END

POPUP "&Help"

BEGIN

MENUITEM "&About",IDM\_HELP\_ABOUT

END

END

menuID.inc:

IDM\_FILE\_EXIT equ 10001

IDM\_ACTION\_AVERAGE equ 10101

IDM\_ACTION\_LIST equ 10102

IDM\_HELP\_ABOUT equ 10201

### 3.4实验步骤

准备上机实验环境、编辑、汇编、连接文件demo， menu。

1. 修改菜单及其选项
2. 修改程序使其在移动窗口时，显示信息不消失且窗口可以顺利弹出

3. 修改弹出窗口信息

4. 修改显示函数，使其显示出数据段中新增的商品的信息

5. 添加平均利润率计算，计算两家商店商品的平均利润率

6. 准备上机实验环境、编辑、汇编、连接文件demo， menu。

7. 检测各个功能

### 3.5实验记录与分析

1. 修改menu.rc 和menuID.inc文件如下图：

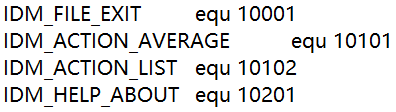
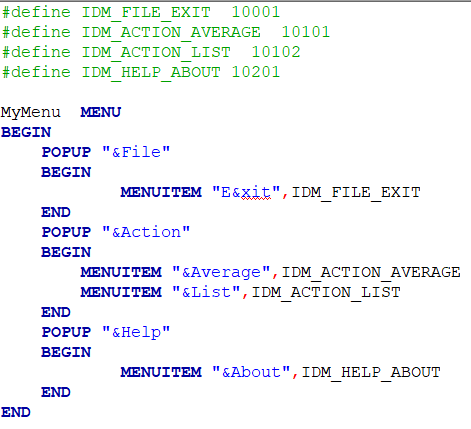


图3.5.1 修改菜单及其选项

并在demo中窗口响应的COMMAND响应中修改不同的情况：

Z]6]PXM%G$XM1I5T[(VBB}H

A_BGVB33P}GI4(0FWRC}3XY

JLBK3%HMSD)3X@@HMPAFH9C

SZD%Q}CROOSCBM4JH[{S)2M

图3.5.2 修改菜单选项对应的功能

1. 在uMsg == WM\_COMMAND时保存指令：

mov eax, wParam

mov OldCommand, eax

WndProc运行时取回：mov eax, OldCommand

并在uMsg == WM\_PAINT时添加如下程序：

.IF eax == IDM\_ACTION\_LIST

invoke Display,hWnd

.ENDIF

invoke DefWindowProc,hWnd,uMsg,wParam,lParam

1. 修改弹出窗口只需修改弹出窗口中的信息，即将数据段中的AboutMsg做如下修改：

AboutMsg db '我是刘子煜',0

1. 第一行输出商店名称，第二行输出分割线，第三行输出商品信息的提示，第四行输出分割线，从第五行开始运用循环输出商品的信息，在调用TEXTOUT进行输出前，将数据放入AX中，将用来存放转换后数据的变量的地址放入ESI中，调用转换子程序Tentoasc，转换后的数据的长度及内容放入了该变量，此时调用TEXT进行输出：（adr为该商品开头的地址）

MOV ECX, adr

MOV AX, [ECX+10]

MOV ESI, OFFSET p\_buyprice

CALL Tentoasc

invoke TextOut,hdc,XX+1\*XX\_GAP,numb,offset p\_buyprice+1,[p\_buyprice]

1. 计算平均利润率的程序根据实验三中平均利润率的计算并经过一定的修改使其符合环境。

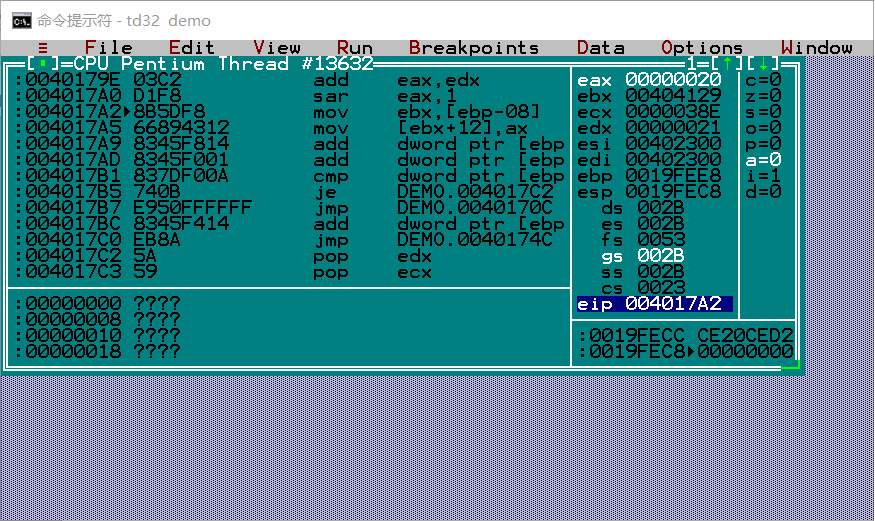


图3.5.3 TD32中的计算结果

1. 编译过程没有发生异常。
2. 连接过程没有发生异常。
3. 功能测试，结果如下图所示：

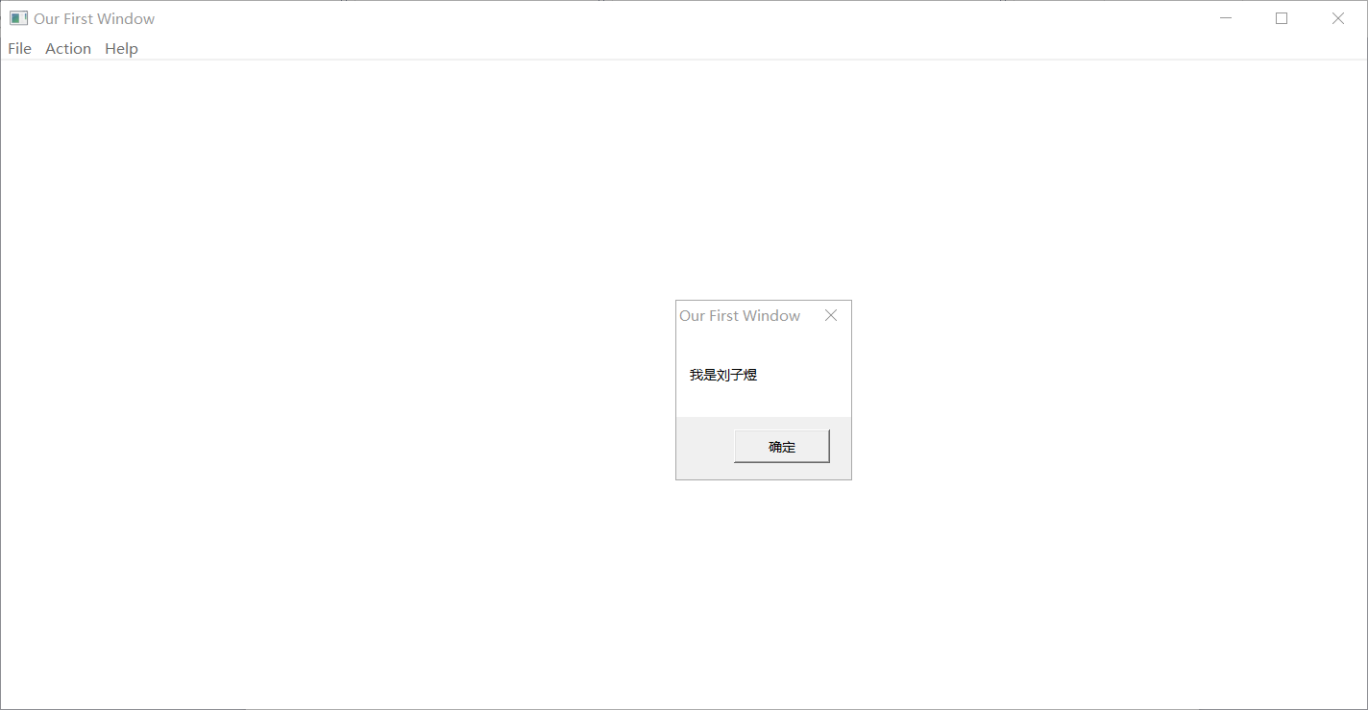
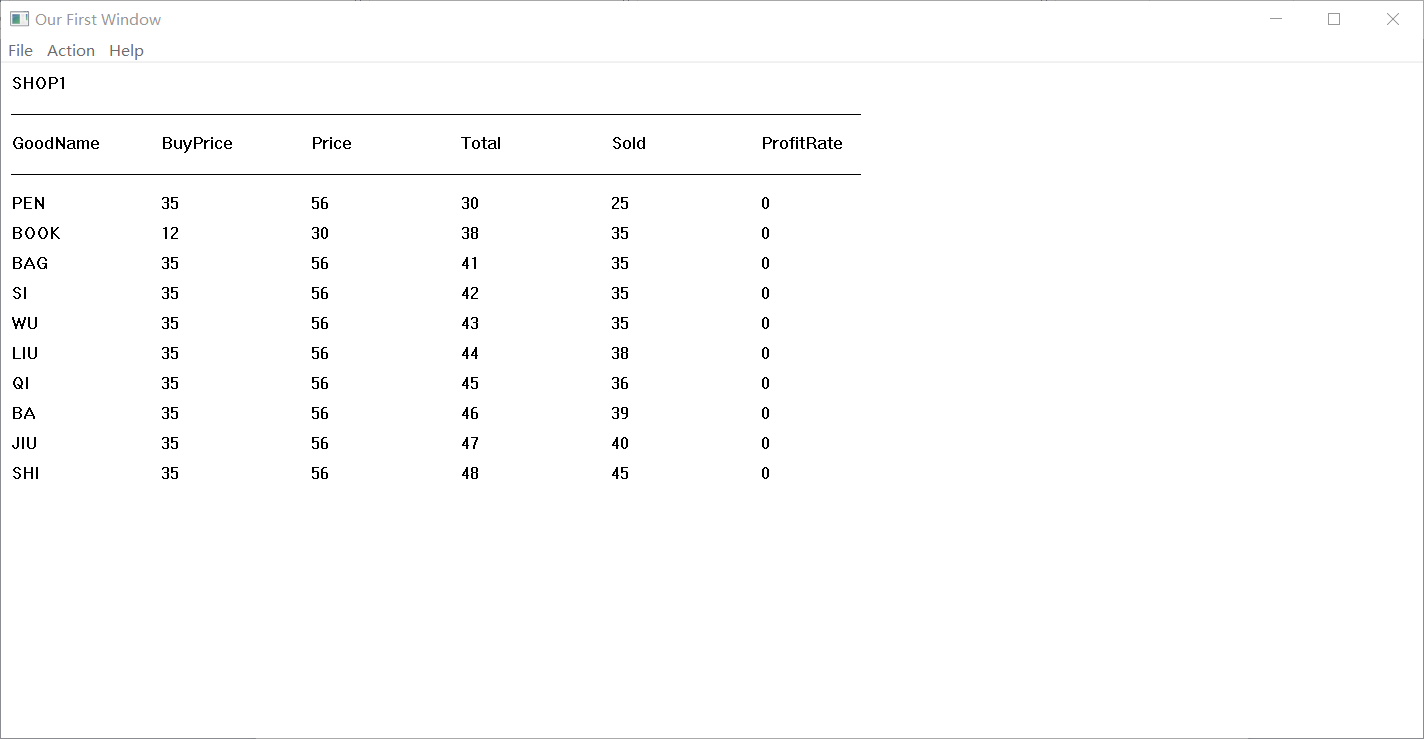


图3.5.4 About功能

图3.5.5 List功能

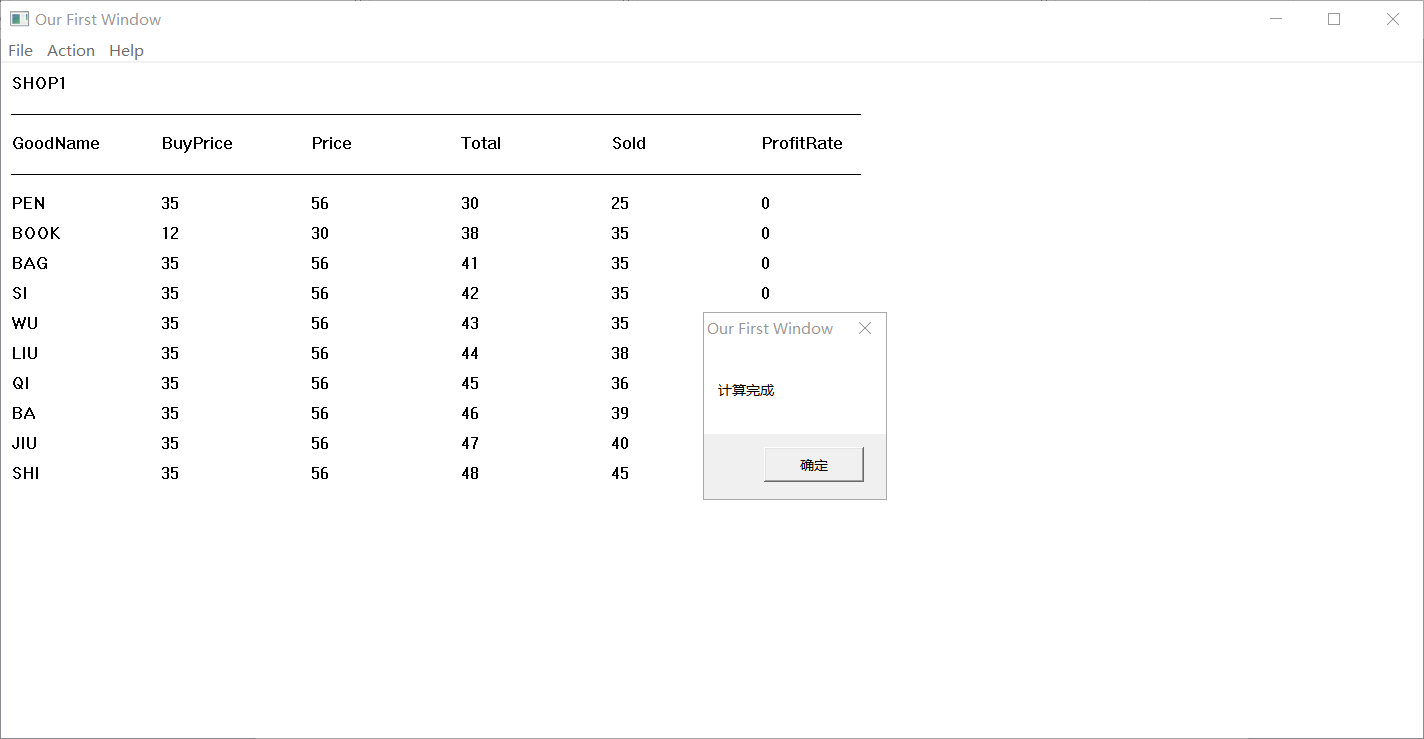


图3.5.6 Average功能完成提示

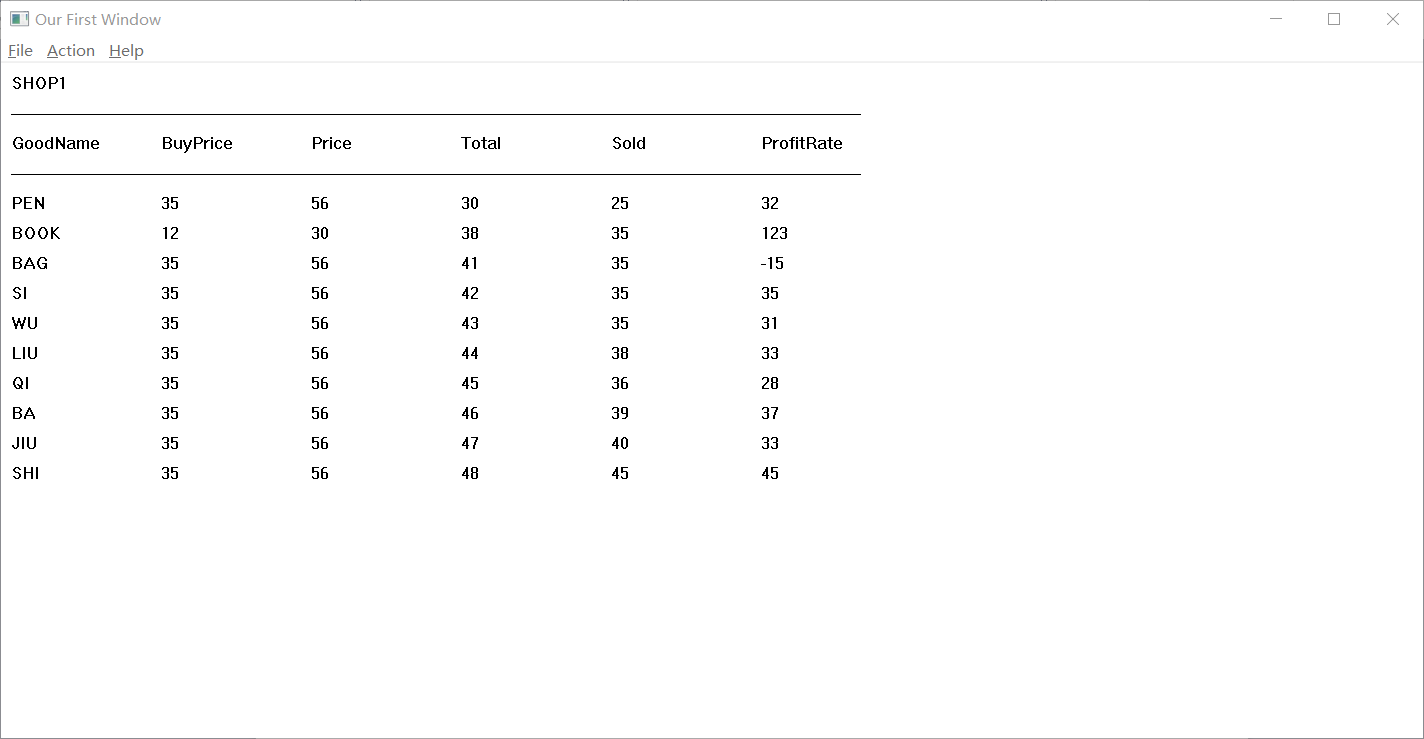


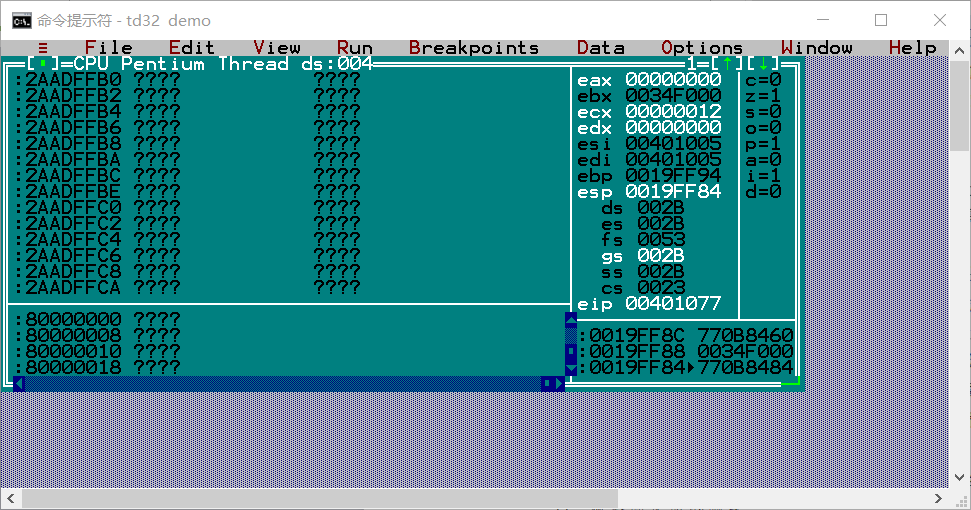
图3.5.7 Average功能结果验证

1. 思考题：
2. 基于窗口的应用程序所包含的四个部分之间的衔接关系：编译汇编源文件x\_1.asm和资源文件x\_2.rc，产生x\_1.obj和x\_2.res，连接x\_1.obj和x\_2.res，形成demo.exe文件。程序执行流程为顺序执行。
3. TD32与16位TD的异同：

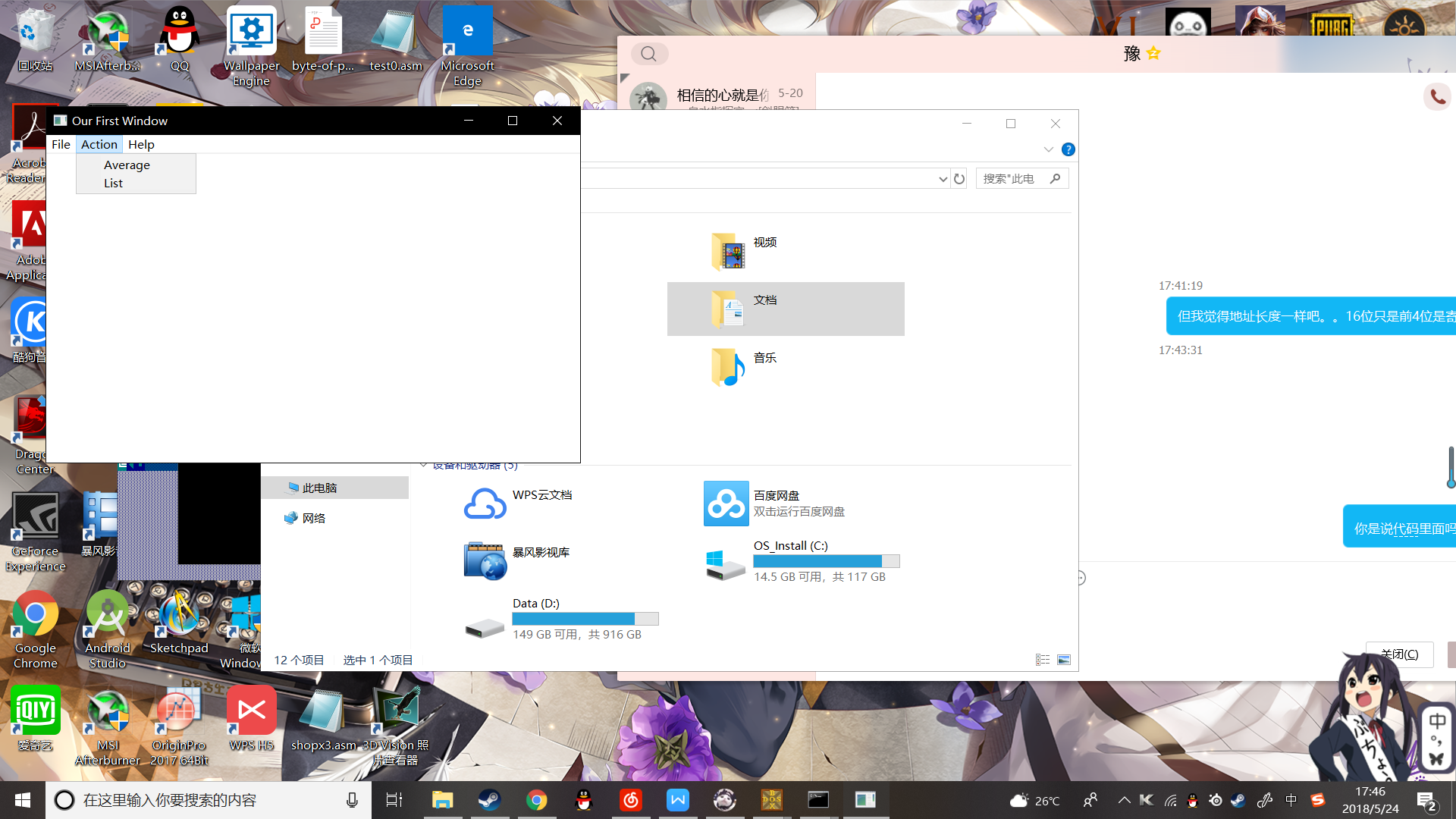
相同点：分区相同，每个群中所显示的内容相同，即都为数据区，代码区等；操作方式基本相同

不同点：16位TD中代码区，数据区和堆栈区中地址前相应的段寄存器名，而TD32中并没有；16TD中寄存器区中使用的是AX，BX，CX，DX，SI，DI，BP，SP，而TD32中使用的是EAX，EBX，ECX，EDX，ESI，EDI，EBP，ESP。

1. 用TD32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：



1. 编写和处理简单资源脚本，装入菜单，观察效果，即对menu.rc的改写，代码如上图3.5.1所示，装载后的执行结果如下图所示：



1. 每个菜单项或按键等操作所对应的消息信息：

当选择菜单项，发出消息：WM\_COMMAND

根据菜单项的不同，wParam有所不同根据定义：

Exit：IDM\_FILE\_EXIT

Average： IDM\_ACTION\_LIST

List： IDM\_ACTION\_AVERAGE

About： IDM\_HELP\_ABOUT

当接受到WM\_CLOSE，则会发出消息：WM\_DESTROY

当按下键盘上的见，则会发出消息：WM\_KEYDOWN

当有窗口移动，缩放等以及弹出窗口时，则会发出消息：WM\_PAINT

1. 输出函数TextOut的原型：设备描述表句柄，字符串的开始位置x坐标，字符串的开始位置y坐标，字符串开头地址，字符串中字符的个数
2. 使用结构变量在固定位数时，不需要根据前面字符串的长度更改后面填充的位数，且使用结构变量进行初始化时方便，清晰，每个变量之间偏移量固定。
3. Invoke语句翻译成机器码后，反汇编出的CALL指令的机器码都为E8\_ \_ 080000，压栈顺序为入口参数顺序从后向前依次压栈

# 4 总结与体会

这次任务是第一次完成win32程序的编写，通过这一次实验让我认识到了它与之前汇编环境的不同，同时这也是第一次汇编环境下完成各种窗口的操作，虽然由于时间紧迫没能了解所有相关函数的原型，以及示例程序中出现的各个特定变量的功能，但是也掌握了一部分函数的基本使用方法。

在刚开始进行程序编写试运行的时候，程序总是会崩溃，主要是没有掌握该环境下语法的要求，使用了错误的语法，虽然能编译通过但是在执行过程中会出现问题，比如使用包含变量的变址寻址时，同时寄存器间接寻址时不能使用SI，DI，使用会在编译过程中报错……这些具体的语法要求也是我们应该记住，在以后使用过程中需要注意的。同时，在开始时还不太懂一些函数的使用方法以及逻辑结构，在调整一些函数以完成功能的时候，遇到了很大的问题，后来逐渐懂了各个函数的含义及逻辑结构问题也就迎刃而解了。调试方法上，只尝试使用了TD32进行调试，未尝试使用源码级调试，以后有时间也会在课下进行尝试。

总而言之，这次实验的这次实验过程还算顺利，虽然遇到了很多问题，但还是在自己的查询和不断地尝试下，基本将它们完全解决了，但经过一次实验还说不上是熟悉了Win32的环境，只能说是初步了解了一下，在课下我也会更多地去进行相关的实验保证自己能够基本掌握其环境和大多数功能的使用方法。

# 参考文献

[1] 作者：王元珍 曹忠升 韩宗芬，书名：80X86汇编语言程序设计，出版者：华中科技大学出版社。