華中科技大學

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

实验时间： 2018-3-26，14：00-17：30，2018-4-02，14：00-17：30

实验地点： 南一楼 804 室 8 号实验台

指导教师： 朱虹

专业班级 计算机 201601 班

学　　号： U201614532　　**姓　　名：** 吕鹏泽

同组学生： 无　　**报告日期：** 2018 年　3 月 26 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：

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

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。

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；

理解数、符号、寻址方式等在计算机内的表现形式；

理解指令执行与标志位改变之间的关系；

熟悉常用的 DOS 功能调用；

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；

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 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。

任务 4. 内存单元的访问。

以四种不同的内存寻址方式，将自己学号的后四位依次存储到 以 XUEHAO 开头的存储区中，要求学号的存放以字符方式存放。

要求：在报告中给出完整的程序；给出运行效果截图；（不需要画流程图）；在程序注释中，明确指出访问存储单元时，用的是什么寻址方式。

任务 5. 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（必须是自己名字的拼音）
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。

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。

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”。

（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 任务 1

3.1.1 实验步骤

1. 准备上机实验环境。

2. 直接在 DOSBOX 命令行中键入 TD 回车。

3. 直接在 TD 中的代码显示区置的任意位置键入：

MOV AH,+0110011B

ADD AH,+1011010B

然后执行这两条程序指令，观察执行后的 SF、OF、CF、ZF 的值。

4. 重复步骤 3，实现剩下两组计算操作。

5. 将 ADD 指令换成 SUB 指令后重复上述步骤进行减法运算。

6. 预计执行加法运算之后三组数据分别为：

（1）（AH）=8DH　　SF=1,OF=1,CF=0,ZF=0

（2）（AH）=7AH　　SF=0,OF=1,CF=1,ZF=0

（3）（AH）=08H　　SF=0,OF=0,CF=1,ZF=0

7. 预计执行减法运算之后三组数据分别为：

（1）（AH）=0D9H　CF=1,ZF=0,SF=1,OF=0

（2）（AH）=34H　　CF=0,ZF=0,SF=0,OF=0

（3）（AH）=0C2H　CF=1,ZF=0,SF=1,OF=1

3.1.2 实验记录与分析

1. 实验环境条件：WINDOWS 10 下 DOSBox0.72； TD.EXE 5.0

2. 输入第一个运算式对应的指令 MOV AH,+0110011B；ADD AH,+1011010B;TD 调试过程中直接将二进制码翻译成了十六进制的数,其中高 8 位存在高地址字节中，低 8 位存在低地址字节中。一开始，我忘记设置代码执行的起始位置，导致了错误的运行结果，修改后执行两条指令后的结果如图 3.1.1 所示。可以看出，计算结果在 AX 的高字节中（8DH）与标志位的状态(CF=0,ZF=0,SF=1,OF=1)与事前预期的是一致的。

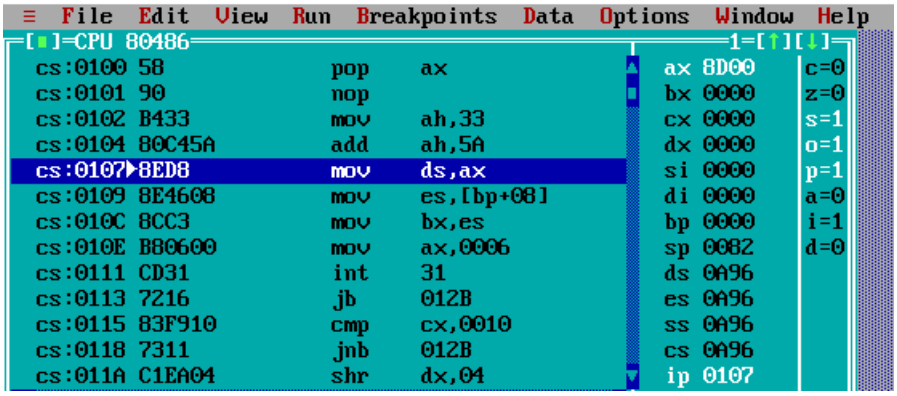


图 3.1.1 执行完测试语句 1 后的状态

3. 输入第二个运算式对应的指令 MOV AH,-0101001B；ADD AH,-1011101B;。执行两条指令后的结果如图 3.1.2 所示。可以看出，计算结果在 AX 的高字节中（7AH）与标志位的状态(CF=1,ZF=0,SF=0,OF=1)与事前预期的是一致的。

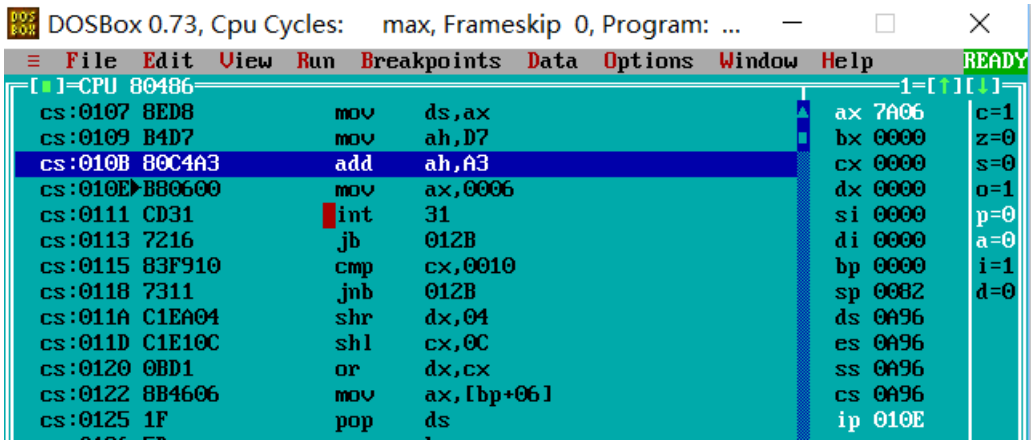


图 3.1.2 执行完测试语句 2 后的状态

4. 输入第三个运算式对应的指令 MOV AH,+1100101B；ADD AH,-1011101B;。执行两条指令后的结果如图 3.1.3 所示。可以看出，计算结果在 AX 的高字节中（08H）与标志位的状态(CF=1,ZF=0,SF=0,OF=0)与事前预期的是一致的。

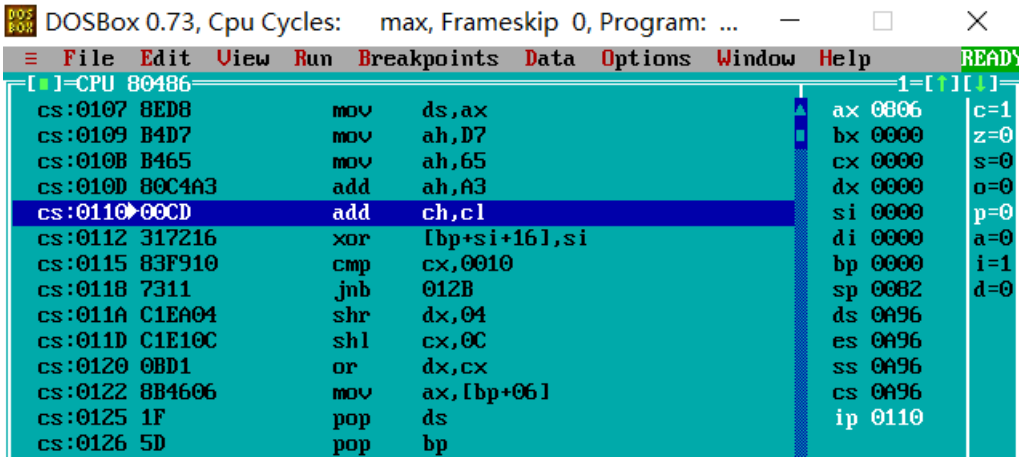


图 3.1.3 执行完测试语句 3 后的状态

5. 输入第一个运算式对应的指令 MOV AH,+0110011B;SUB AH,+1011010B;。执行两条指令后的结果如图 3.1.4 所示。可以看出，计算结果在 AX 的高字节中（0D9H）与标志位的状态(CF=1,ZF=0,SF=1,OF=0)与事前预期的是一致的。

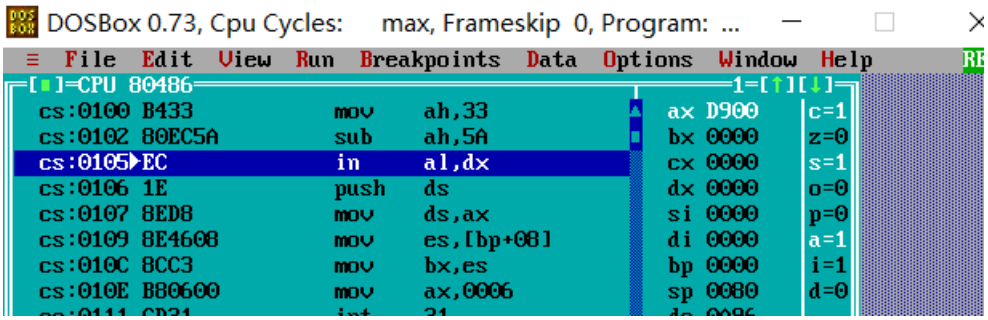


图 3.1.4 执行完测试语句 3 后的状态

6. 输入第二个运算式对应的指令 MOV AH,-0101001B;SUB AH,-1011101B;。执行两条指令后的结果如图 3.1.5 所示。可以看出，计算结果在 AX 的高字节中（34H）与标志位的状态(CF=0,ZF=0,SF=0,OF=0)与事前预期的是一致的。

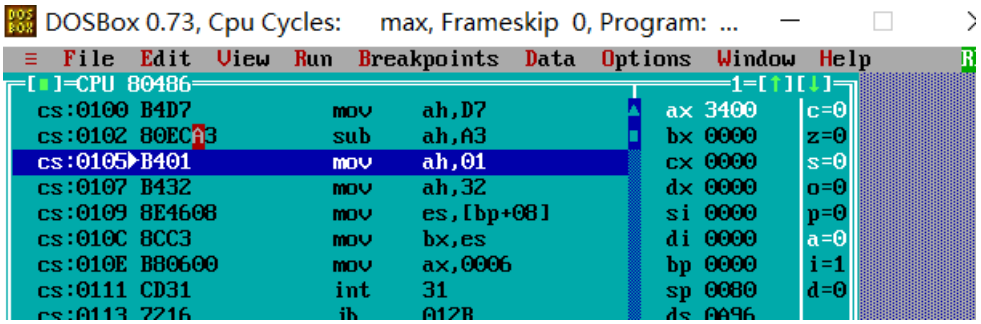


图 3.1.5 执行完测试语句 3 后的状态

7. 输入第三个运算式对应的指令 MOV AH,+1100101B;SUB AH,-1011101B;。执行两条指令后的结果如图 3.1.6 所示。可以看出，计算结果在 AX 的高字节中（0C2H）与标志位的状态(CF=1,ZF=0,SF=1,OF=1)与事前预期的是一致的。

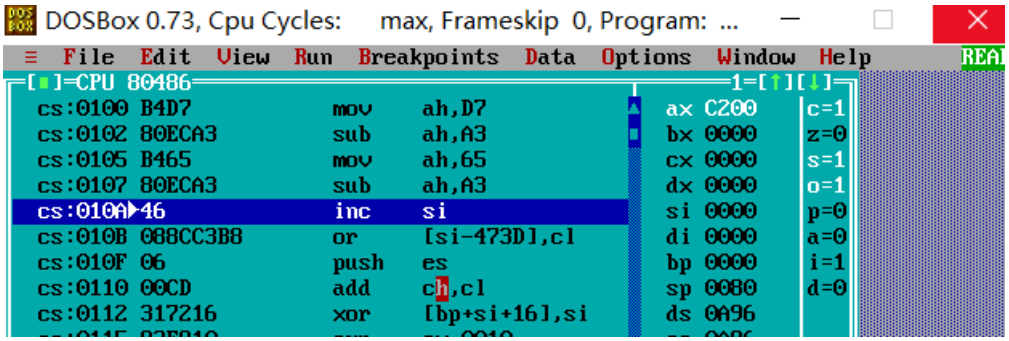


图 3.1.6 执行完测试语句 3 后的状态

8.通过运算可知，系统在进行加减法运算时将无符号数和有符号数均视为有符号数，运用补码进行运算，它们的区别是有符号数将运算结果的最高位视为符号位，而无符号数将运

算结果的最高位视为数值位。与加法运算不同的是，在进行减法运算 A-B 时，将$[A]_{补}$[B]$_{补}$视为无符号数，如果$[A]_{补}$>=$[B]_{补}$,则置 CF=0，否则置 CF=1。

3.1.3 思考题

1. 打开 TD 之后，如何在代码区输入一条指令，并且执行这条指令？

使用鼠标或键盘选中修改的内容，然后直接在键盘中输入修改后的代码，Ctrl+N 快捷点设置执行位置，F8 执行。

2. 如何在代码区输入若干条指令后，再从输入的第一条指令开始执行？

光标选中希望执行的位置后使用 Ctrl+N 快捷点设置执行位置为当前位置

3. 在输入一条指令中的数据时，若以 16 进制输入，需要注意什么问题？

若以字母开头加前缀 0，结尾要加 H

4. 执行一条指令后，如何查看寄存器的值（含 32 位寄存器）？如何修改寄存器的值？

TD 右侧可查看寄存器的值；直接用光标选中修改的位置后在键盘输入即可

5. 执行一条指令后，如何查看标志寄存器的值？

在 TD 右侧的标志区查看

6. 经过 6，7 后，总结加减法对标志寄存器的影响？

执行完指令后，若结果为正，SF=0，否则 SF=1。若运算结果为 0，ZF=1，否则 ZF=0。两正数相加得负数或两负数相加得正数 OF=1，否则 OF=0。加法运算有进位或借位 CF=1，否则 CF=0。减法运算 A-B 时，将$[A]_{补}$[B]$_{补}$视为无符号数，如果$[A]_{补}$>=$[B]_{补}$,则置 CF=0，否则置 CF=1。

3.2 任务 2

3.2.1 实验步骤

1. 使用 VS2017 录入源程序，存盘文件名为 TEST1.ASM。

2. 使用 MASM 汇编源文件,即 MASM TEST1；

3. 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。

4. 使用连接程序 LINK.EXE 将汇编生成 TEST1.OBJ 文件连接成执行文件。 即 LINK TEST1.OBJ；

5. 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 TEST1.EXE 文件。

6. 使用 TD.EXE 观察 TEST1 的执行情况。即 TD TEST1.EXE 回车

（1）开始时观察寄存器 BX、BP、SI、DI 和数据段开始的前 40 字节的内容。

（2）执行至“MOV CX，10”，观察寄存器 BX、BP、SI、DI 中的内容。

（3）执行至“INT 21H”，观察寄存器 BX、BP、SI、DI 中的内容，并观察数据段开始的前 40 字节的内容。

7. 源代码 DATA 段中加入待打印的字符

```
BUF DB 0AH,0DH,'Press any key to begin!$'
CRTF DB 50
     DB 0
     DB 50 DUP(0)
```

8.在 CODE 段中 LOPA 前加入打印字符代码：

```
LEA DX,BUF
MOV AH,09H
INT 21H
LEA DX,CRTF
MOV AH,01H
INT 21H
```

9.观察运行结果

10. 程序将 buf1 每个字节复制到 buf2，buf1 每个字节+1 复制到 buf3，buf1 每个字节+4 复制到 buf4 理论结果如下：

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

3.2.2 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)

BUF DB 0AH,0DH,'Press any key to begin!$'
CRTF DB 50
     DB 0
     DB 50 DUP(0)

CHAR DB ?
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

LEA DX,BUF
MOV AH,09H
INT 21H
LEA DX,CRTF
MOV AH,01H
INT 21H

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 实验记录与分析

1.用 TD 打开 test1.exe 文件后，寄存器的初始值如图 3.2.1 所示：

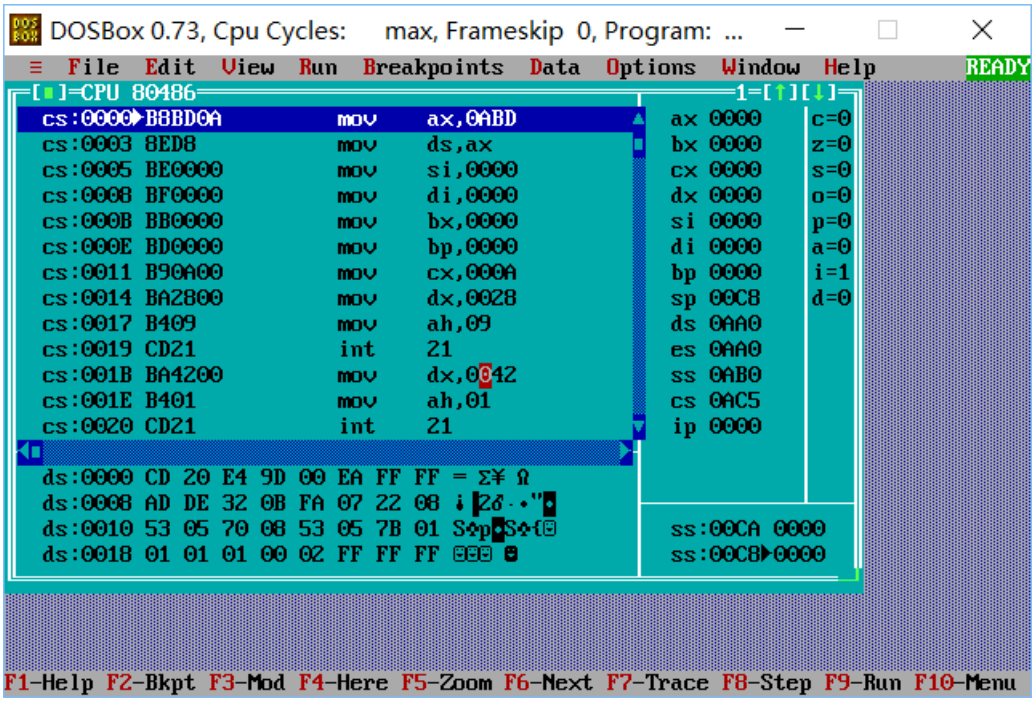


图 3.2.1 进入 TD 后各寄存器初始值

2.单步执行到语句“MOV　CX，10”，BX=BP=SI=DI=0000H,各寄存器的值如图 3.2.2 所

示，此时，数据段各个变量的偏移地址被赋值给了相应的寄存器。

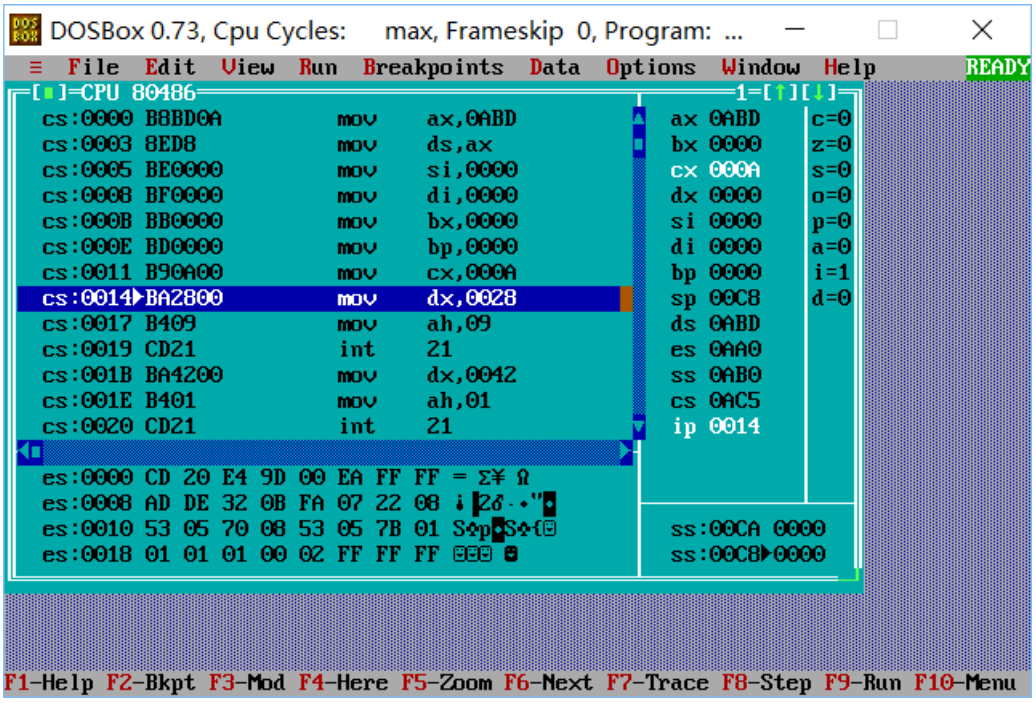


图 3.2.2 执行到第一条指定语句各寄存器初始值

3.继续单步执行，可以观察到数据段的值在被循环赋值，其中 BUFF1 每个字节赋值到 BUFF2，BUFF1 每个字节加 1 赋值到 BUFF3，BUFF1 每个字节加 4 赋值到 BUFF4.最后各变量内存单元值如图 3.2.3 所示，可以看到运行结果是符合理论值的。

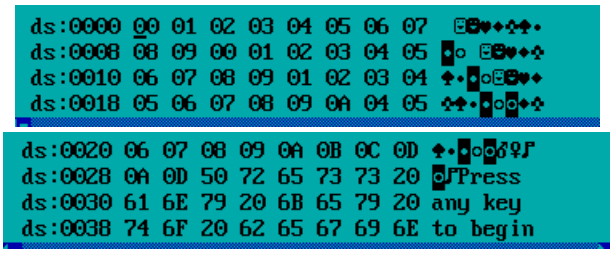


图 3.2.3 程序运行结束后数据域前 40 字节内容

5. 当执行到 INT 21H 时各寄存器的值如图 3.2.4 所示，BX=000AH,BP=000AH,SI=000AH,DI=00AH。

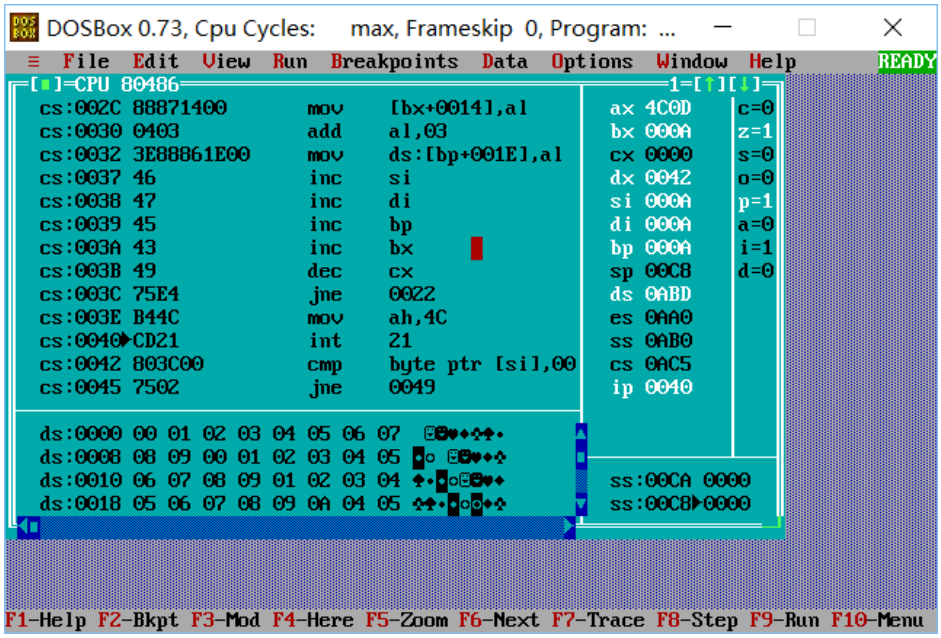


图 3.2.4 执行到第二条指定语句各寄存器初始值

6.添加代码后运行 test1.exe 文件结果如图 3.2.5 所示。首先命令行输出字符串，接下来输入任意字符后（这里输入'e'）进行下一步操作

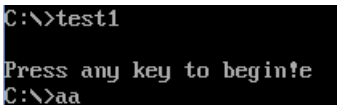


图 3.2.5 修改后的字符串输出

3.2.4 思考题

1. 如何将一可执行程序调入 TD，并查看代码区？理解机器码与汇编指令之间的对应关系？

在 DOSBOX 中使用指令 TD xxxx.exe 将程序调入 TD，在代码区使用 goto 指令可查看指定位置的代码

2. 如何设置断点并运行到断点？

光标选中，F2 在光标所在位置设置断点，F9 运行程序，在断点处会自动停下

3. 如何使程序运行到光标的当前点？（假设活动光标在代码区，指向某一条指令）

快捷键 F4（Go to cursor）

4. 如何单步执行一条指令？（多种方法）

快捷键 F7 或 F8

5. 在数据区找到数据段的方法？思考：是否可以用这一方法查看代码段甚至整个程序?

(1)goto，DS：偏移地址，(2)goto，直接输入：段寄存器的值：偏移地址；(3)直接在数据区用

光标移动查找。可以

6. 修改某个指定的存储单元的值，如任务 2 中的 BUF2

找到 BUF2 的存储位置，直接在 TD 数据域中 BUF2 存储位置使用键盘输入新的数据，回车确定

7. 如何查看堆栈？

选中 View 菜单，选择 Stack 查看堆栈

8. 如何汇编一个汇编源程序并链接产生可执行代码？

使用指令 ML xxxx.ASM 或指令 MASM xxxx.ASM 和 LINK xxxx.ASM;

9. 如何读懂源程序在汇编过程中产生的错误？

根据产生的错误代码在课本的附录中查找错误原因

10. 查看 BUF2 等变量在 TD 中的表示形式并总结

以 16 进制数表示，以字节分隔

12 查看寄存器间接寻址、变址寻址的汇编源程序经汇编、链接后在 TD 中的表示形式？总结源程序的指令和 TD 中的指令差异。

见任务 3 分析.

3.3 任务 3

3.3.1 实验步骤

1. 使用文本文档录入源程序，存盘文件名为 TEST2.ASM。
2. 使用 MASM 汇编源文件。即 MASM TEST2；
3. 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。
4. 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 LINKTEST2.OBJ；
5. 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。
6. 执行该程序。
7. 使用 TD.EXE 观察 TEST2 的执行情况。即 TD TEST2.EXE 回车。
8. 单步执行代码，观察数据段前 40 字节的内容变化

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;//ESI 作为变址寄存器，起始值为 0
 MOV ECX,10;//控制循环
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 实验记录与分析

1.使用 TD 调试程序，将寄存器改成 32 位

2.运行到对应指令时与任务 2 中的正确结果进行比对，在执行到“MOV　CX，10”时结果如图所示

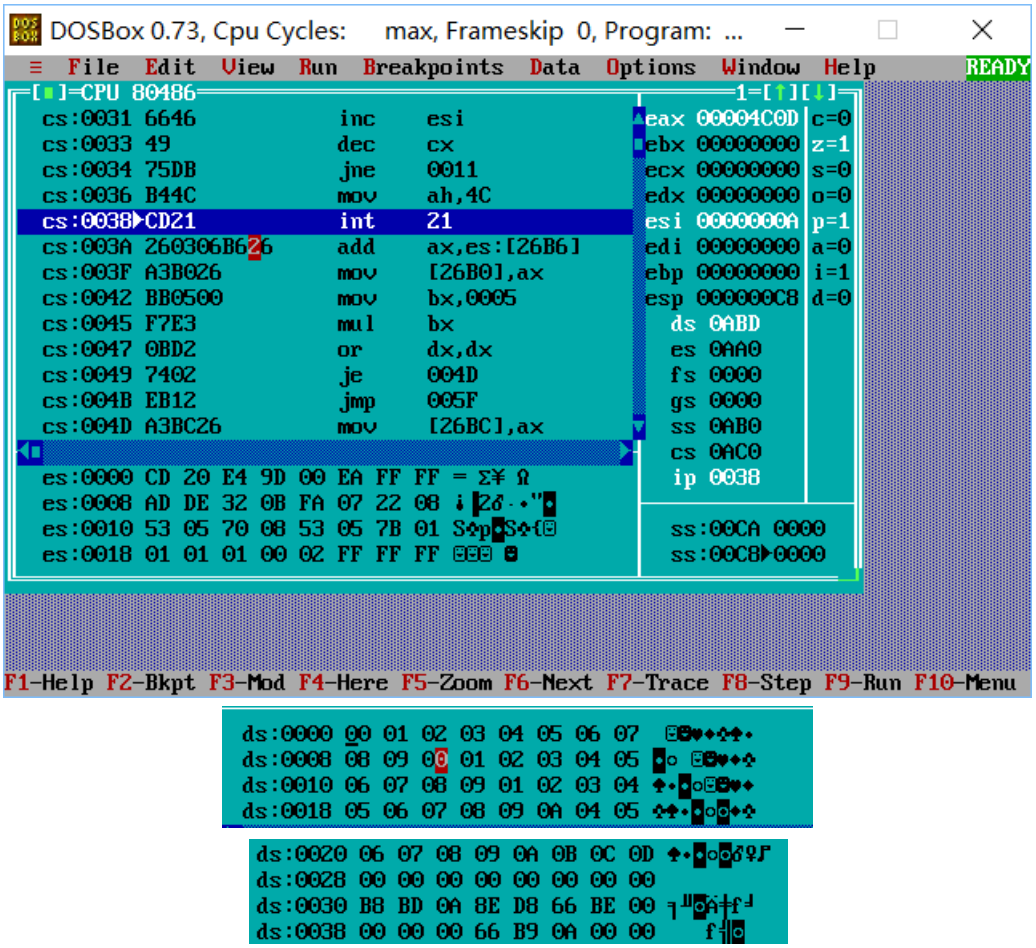


图 3.3.1 运行结束后数据域前 40 字节内容

3.通过观察 TD 的代码窗口可以发现，在计算机中的机器指令代码以二进制码的形式存在，且每一条汇编语句对应相应的机器指令，以指令+操作数的格式，且对应的数在机器码中是低位在前，高位在后。观察反汇编过程，可知 TD 反汇编生成的代码与自己写的有些不同，将 10 进制数转换为 16 进制数，且省略了一些语句。比如语句 MOV ECX,10 变成了如图 3.3.2 所示代码，将 10 进制数 10 转换为了 16 进制数 A

图 3.3.2 数的进制转换

再比如语句 MOV BUF3[ESI],AL 变成了如图 3.3.3 所示代码，它将 BUF3 的偏移地址直接计算出来并替换掉 BUFF3

图 3.3.3 代码的转换

而在任务二执行相同功能的语句的反汇编代码如图 3.3.4 所示，同编写的一样。

图 3.3.4 代码的转换

4.从其他不同字节位置反汇编得不到正确的结果，甚至 TD 会崩溃而异常退出，可见 IP/EIP 指向起始位置是重要的而且是必要的。

3.4 任务 4

3.4.1 实验步骤

1. 使用 VS2017 录入源程序，存盘文件名为 TEST3.ASM。

2. 使用 MASM 汇编源文件,即 MASM TEST3；

3. 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。

4. 使用连接程序 LINK.EXE 将汇编生成 TEST3.OBJ 文件连接成执行文件。 即 LINK TEST3.OBJ；

5. 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 TEST3.EXE 文件。

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

3.4.2 源代码

```
.386
STACK SEGMENT USE16 STACK
DB 200 DUP(0)
STACK ENDS
 DATA SEGMENT USE16
 XUEHAO DB 4 DUP(0);声明变量

 DATA ENDS
CODE SEGMENT USE16
ASSUME CS:CODE,DS:DATA,SS:STACK
START:
 MOV AX,DATA
 MOV DS,AX
 MOV AH,'4'
 MOV BX,OFFSET XUEHAO;
 MOV [BX],AH                    ;第一位字符'4'采用寄存器间接寻址方式

 MOV AH,'5'
 MOV SI,1                       ;SI 作为变址寄存器，初值为 1
 MOV XUEHAO[SI],AH              ;第二位字符'5'采用变址寻址方式

 MOV AH,'3'
 LEA BX,XUEHAO                  ;XUEHAO 的偏移地址送到 BX 寄存器中,作为基址寄存器
 MOV SI,2                       ;SI 作为变址寄存器，初值为 2
 MOV [BX][SI],AH                ;第三位字符'3'采用基址加变址寻址方式

 MOV AH,'2'
```

```
MOV DS:[3H],AH                    ;第四位字符采用直接寻址方式

MOV AH,0                          ;程序结束
CODE ENDS
END START
```

3.4.3 实验记录与分析

如图 3.4.1，通过观察 DATA 域可知，程序已成功将’4532’移入到 XUEHAO 变量中。

图 3.4.1 程序执行后 DATA 域的值

3.5 任务 5

3.5.1 设计思想及存储单元分配

设计思想:

1.提示并输入用户的账号和密码

2.利用循环判断账号类型和账号是否和密码匹配

3.提示用户输入查询商品的名称，并根据其登陆状态给出相应的利润率等商品信息

4.根据计算出的利润率给出商品等级

5.使用转移指令回到步骤 1

存储单元分配：

BNAME:管理员账号，DB，10 字节

BPASS:登陆密码，DB，6 字节

AUTH:标记登陆状态，DB，1 字节

S1:网店 1 名称，DB，6 字节

GA1:商品 1，DB，商品名称，10 字节；DW，商品信息，10 字节

GA2:商品 2，DB，商品名称，10 字节；DW，商品信息，10 字节

GAN:商品 N

S2:同 S1….

MSG1-MSG4:输入提示字符串

CRLF:回车换行串

SIGH:DB,标记利润率正负，正置 1，负置 0

in_name:输入姓名缓冲区，DB，11 字节

in_pwd:输入密码缓冲区，DB，7 字节

in_goods:输入商品名称缓冲区，11 字节

寄存器分配：

CX：控制循环；

BX：基址寄存器；

BP：存放缓冲区基地址；

AL：临时读取数据域的值；

AX，DX，SI：临时寄存器；

3.5.2 流程图

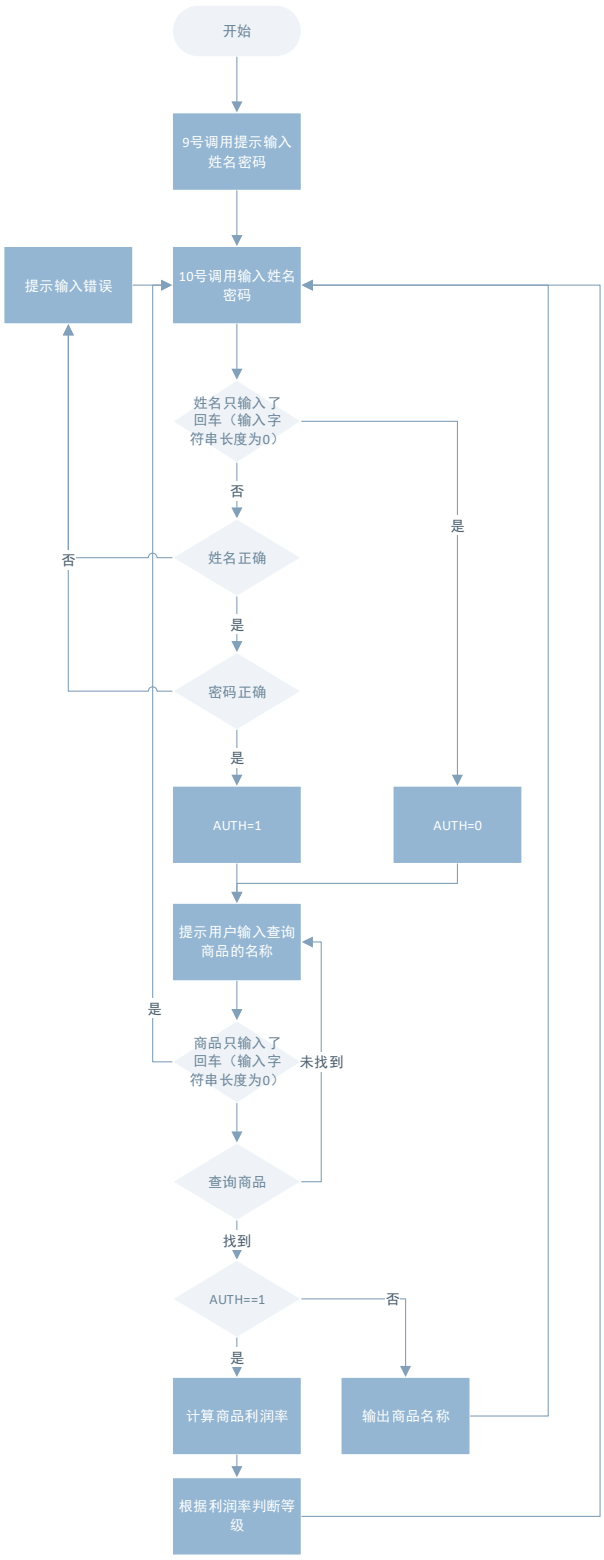
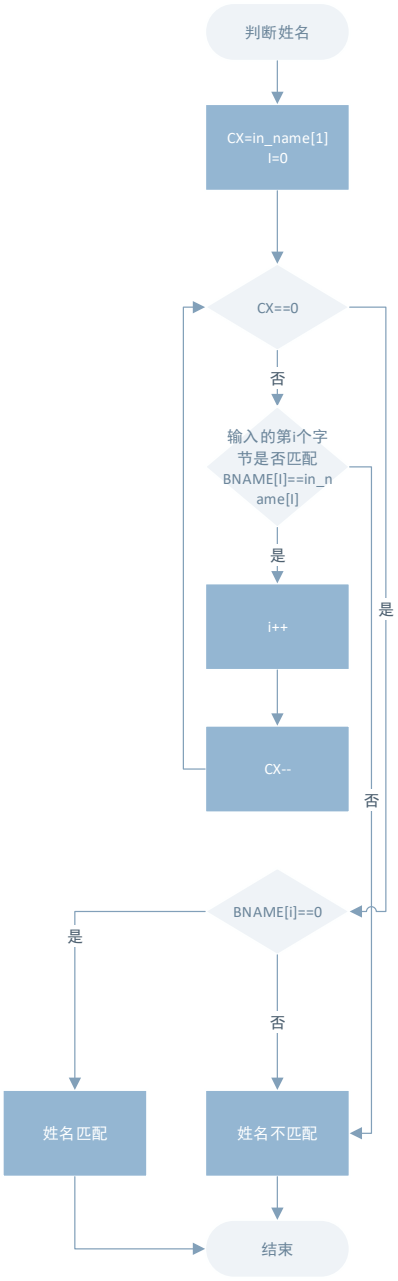


图 3.5.1 程序总体流程图



3.5.2 字符串匹配流程图

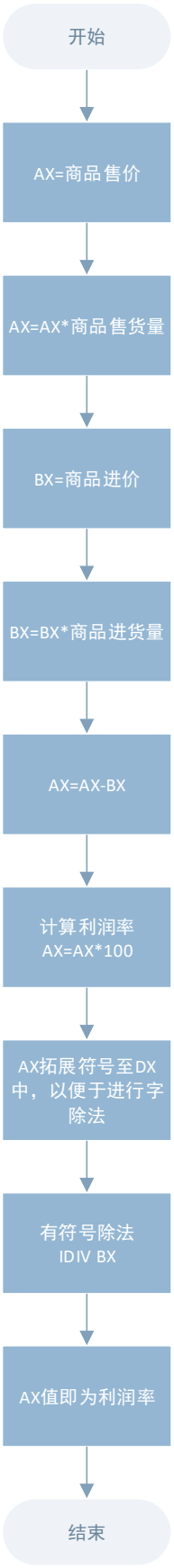

开始
AX=商品售价
AX=AX*商品售货量
BX=商品进价
BX=BX*商品进货量
AX=AX-BX
计算利润率
AX=AX*100
AX拓展符号至DX中，以便于进行字除法
有符号除法
IDIV BX
AX值即为利润率
结束

图 3.5.3 利润率计算流程图

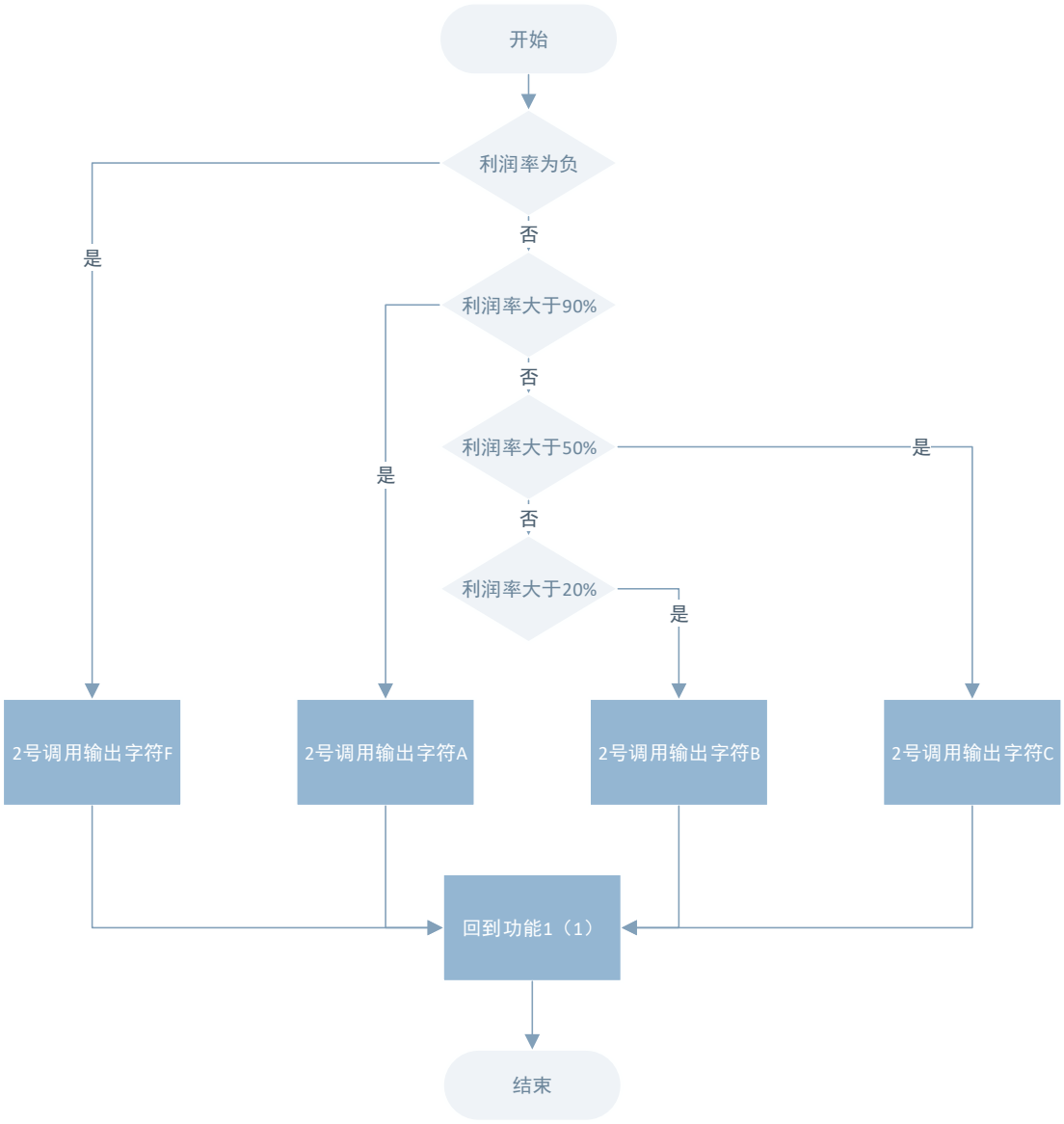


图 3.5.4 利润率等级判断

3.5.3 源程序

```
.386
STACK SEGMENT USE16 STACK
DB 200 DUP(0)
STACK ENDS
DATA SEGMENT USE16
BNAME DB 'LVPENGZE',0,0                ;管理员姓名
BPASS DB 'test',0,0                    ;密码
AUTH DB ?                              ;标记登陆状态
N EQU 30
S1 DB 'SHOP1',0                        ;网店 1 名称，用 0 结束
GA1 DB 'PEN',7 DUP(0)                  ;商品 1
```

```
 DW 35,56,70,25,?                ;进货价、销售价、进货总数、已售数量、利润率
GA2 DB 'BOOK',6 DUP(0)           ;商品 2
 DW 12,30,25,5,?
GAN DB N-2 DUP('Temp-Value',15,0,20,0,30,0,2,0,?,?);其他商品

S2 DB 'SHOP1',0                  ;网店 2 名称，用 0 结束,商品类型同网店 1
GB1 DB 'PEN',7 DUP(0)            ;商品 1
 DW 35,56,70,25,?                ;进货价、销售价、进货总数、已售数量、利润率
GB2 DB 'BOOK',6 DUP(0)           ;商品 2
 DW 12,30,25,5,?
GBN DB N-2 DUP('Temp-Value',15,0,20,0,30,0,2,0,?,?);其他商品

PR1 DW 0                         ;商店 1 利润率
PR2 DW 0                         ;商店 2 利润率
APR DW 0                         ;平均利润率
SIGN DB 0                        ;APR 大于 0 置 1，否则置 0
MSG1 DB 0AH,0DH,'Input your account:',0DH,0AH,'$'
MSG2 DB 'Input your password:',0DH,0AH,'$'
MSG3 DB 'WRONG ACCOUNT',0DH,0AH,'$'
MSG4 DB 'Enter the name of the item:',0DH,0AH,'$'

CRLF DB 0DH,0AH,'$'              ;回车换行
in_name   DB 11                  ;姓名缓冲区
          DB ?
          DB 11 DUP(0)
in_pwd      DB 7                       ;密码缓冲区
          DB ?
          DB 7 DUP(0)
in_goods DB 11                   ;商品名缓冲区
        DB ?
        DB 11 DUP(0)


DATA ENDS
CODE SEGMENT USE16
ASSUME CS:CODE,DS:DATA,SS:STACK
START:
 MOV AX,DATA
 MOV DS,AX
NEXT:                                  ;功能 1，提示登陆
 LEA DX,MSG1
 MOV AH,9
 INT 21H                         ;9 号调用，提示用户输入姓名

 LEA DX,in_name
 MOV AH,10
 INT 21H                         ;10 号调用，用户输入姓名
 CALL PRINTCRLF

 LEA DX,MSG2
 MOV AH,9
 INT 21H                         ;9 号调用，提示用户输入密码

 LEA DX,in_pwd
 MOV AH,10
 INT 21H                         ;10 号调用，用户输入密码
 CALL PRINTCRLF

 MOV BL,in_name[2]
 CMP BL,'q'
 JE    EXIT                      ;输入'q'，退出程序
```

```
 MOV AH,1
 MOV SIGN,AH
 MOV BL,in_name[1]
 CMP BL,0
 JE UNLOGIN                    ;输入回车,转未登录
 JMP LOGIN                     ;否则转登录

UNLOGIN:
 MOV AH,0
 MOV AUTH,AH                   ;标记登陆状态
 JMP PROFIT                    ;转利润率计算

LOGIN:                              ;功能 2，判断登陆
 MOV CX,0
 MOV SI,0                   ;循环条件初始化
LOOP1:                              ;循环比较姓名
 MOV BL,in_name[SI+2]
 CMP BL,BNAME[SI]
 JNE ACCOUNTWRONG           ;姓名不匹配转 ACCOUNTWRONG

 INC SI
 INC CX
 MOV AL,in_name[1]
 CBW                           ;将字节转换成字
 CMP CX,AX
 JNE LOOP1                     ;循环未结束

 MOV BL,0
 CMP BL,BNAME[SI]
 JNZ ACCOUNTWRONG           ;输入的字符串是定义字符串的子集

 MOV AL,in_pwd[1]           ;循环比较密码
 CBW
 MOV CX,AX
 MOV SI,0                   ;循环条件初始化
LOOP2:
 MOV BL,in_pwd[SI+2]
 CMP BL,BPASS[SI]
 JNE ACCOUNTWRONG           ;密码不匹配转 ACCOUNTWRONG
 INC SI
 DEC CX
 JNZ LOOP2
 MOV BL,0
 CMP BL,BPASS[SI]
 JNZ ACCOUNTWRONG           ;输入为子集，转 ACCOUNTWRONG
 JMP  SUCCESS               ;账号密码都相同转登陆成功

ACCOUNTWRONG:                       ;登陆失败
 LEA DX,MSG3
 MOV AH,9
 INT 21H                    ;9 号调用，提示登录失败
 JMP NEXT                   ;回到初始状态

SUCCESS:                    ;登陆成功
 MOV AH,1
 MOV AUTH,AH
 JMP PROFIT

PROFIT:                             ;功能 3，计算利润率
 LEA DX,MSG4
```

```
 MOV AH,9
 INT 21H                          ;9 号调用，提示输入商品名称

 LEA DX,in_goods
 MOV AH,10
 INT 21H                          ;10 号调用，用户输入商品名称
 CALL PRINTCRLF

 MOV BL,in_goods[1]               ;00B7H
 CMP BL,0
 JE NEXT                          ;输入回车,回到初始状态

 MOV CX,0                         ;CX 控制第一层循环
 MOV BX,0                         ;BX 基址寄存器
 MOV SI,0                         ;SI 变址寄存器
LOOP3:                                  ;第一层循环，顺序查询 N 件商品
 MOV SI,0
 MOV AL,in_goods[1]
 CBW
 MOV DX,AX                        ;DX 控制第二层循环
 LOOP4:                           ;第二层循环，比较商品名称
       MOV AH,GA1[BX][SI]
       CMP AH,in_goods[SI+2]
       JNE F1                     ;名称不同

       INC SI                     ;变址++
       DEC DX
       CMP DX,0
       JE JUDGE                   ;找到商品,转 JUDGE
       JMP LOOP4
F1:
 ADD BX,20
 INC CX
 CMP CX,N
 JNE LOOP3

 JMP PROFIT                       ;未找到商品，重新输入商品名称
JUDGE:
 CMP SI,10                        ;商品名称恰好 10 个字符
 JE F2

 MOV AH,GA1[BX][SI]
 CMP AH,0
 JNE F1                           ;商品名称为子集,属于未找到

F2:    MOV AH,AUTH
 CMP AH,1
 JE L1                            ;登陆状态转 L1
 JMP L2                           ;未登录状态转 L2
L1:
 MOV ECX,EBX                      ;
 MOV EAX,0
     LEA BP,GA1[BX][10]           ;S1 中商品信息基址
 MOV AX,WORD PTR DS:[BP+2]        ;销售价
 IMUL AX,WORD PTR DS:[BP+6]       ;AX=销售价*已售数量

 MOV EBX,0
 MOV BX,WORD PTR DS:[BP]   ;进货价
 IMUL BX,WORD PTR DS:[BP+4]       ;BX=进货价*进货总数

 SUB AX,BX                        ;AX=利润
```

```
 CWDE
 IMUL EAX,100
 MOV EDX,EAX
 SHR EDX,16                          ;将 EAX 的高 16 位移动到 DX 中
 IDIV BX                             ;AX=利润率
 MOV PR1,AX
 MOV WORD PTR DS:[BP],AX             ;商店 1 利润率

 MOV EBX,ECX
 MOV EAX,0
     LEA BP,GB1[BX][10]              ;S1 中商品信息基址
 MOV AX,WORD PTR DS:[BP+2]           ;销售价
 IMUL AX,WORD PTR DS:[BP+6]          ;AX=销售价*已售数量

 MOV EBX,0
 MOV BX,WORD PTR DS:[BP]   ;进货价
 IMUL BX,WORD PTR DS:[BP+4]          ;BX=进货价*进货总数

 SUB AX,BX                           ;AX=利润
 CWDE
 IMUL EAX,100
 MOV EDX,EAX
 SHR EDX,16                          ;将 EAX 的高 16 位移动到 DX 中
 IDIV BX                             ;AX=利润率
 MOV PR2,AX
 MOV WORD PTR DS:[BP],AX             ;商店 1 利润率

 MOV AX,PR1
 ADD AX,PR2
 SAR AX,1                      ;计算总利润率
 MOV APR,AX
 CMP AX,0
 JL SET0
 JMP LEVEL
SET0:                                ;利润率小于 0
 MOV AH,0
 MOV SIGN,AH
 JMP LEVEL

L2:
     MOV AL,in_goods[1]
     CBW
     MOV SI,AX
 MOV AH,'$'
     MOV in_goods[SI+2],AH

 LEA DX,in_goods[2]
 MOV AH,9
 INT 21H                             ;9 号调用，输出商品名称

 CALL PRINTCRLF

 JMP NEXT                            ;回到初始状态
LEVEL:                                    ;功能 4，商品等级判断

 MOV AH,SIGN
 CMP AH,0                      ;利润率为负，输出一个负号，并求 APR 的绝对值
 JNE _J
 mov ax, APR ; 赋值
 cwde
     cdq ; 扩展符号位
```

```
            ;求补对正数无效
            ;求补对负数等于是取绝对值
        xor eax, edx ;  取反
        sub eax, edx ;  加 1
        mov APR, ax ;  出参
 MOV AH,2
 MOV DL,'-'
 INT 21H
_J:

 MOV AX,APR
 CALL PRINTAX
 MOV AH,2
 MOV DL,'%'
 INT 21H
 CALL PRINTCRLF
 MOV AX,APR
 MOV BH,0
 CMP BH,SIGN
 JE _F                          ;利润率为负
 CMP AX,90
 JGE _A                             ;利润率大于 90%
 CMP AX,50
 JGE _B                             ;利润率大于 50%
 CMP AX,20
 JGE _C                             ;利润率大于 20%

_A:
 MOV AH,2
 MOV DL,'A'
 INT 21H
 CALL PRINTCRLF
 JMP NEXT
_B:
 MOV AH,2
 MOV DL,'B'
 INT 21H
 CALL PRINTCRLF
 JMP NEXT
_C:
 MOV AH,2
 MOV DL,'C'
 INT 21H
 CALL PRINTCRLF
 JMP NEXT
_D:
 MOV AH,2
 MOV DL,'D'
 INT 21H
 CALL PRINTCRLF
 JMP NEXT
_F:
 MOV AH,2
 MOV DL,'F'
 INT 21H
 CALL PRINTCRLF
 JMP NEXT

EXIT:
 MOV AH,4CH
      INT 21H                           ;退出程序
```

```
;-------------------------------以 10 进制输出 AX 中的无符号整数
PRINTAX PROC
        MOV    BX, 10         ;按 10 进制输出.
        OR     AX, AX
        JZ     _0_
LOOP_P:
        XOR    DX, DX
        DIV    BX
        MOV    CX, AX         ;商.
        OR     CX, DX
        JZ     _E_            ;若商与余数都为 0 则结束递归.
        PUSH DX               ;保存 DX 中的余数.
        CALL LOOP_P           ;递归.
        POP    DX             ;恢复余数.
        ADD    DL, '0'        ;变成 ASCII 码.
        JMP    _1_
_0_:    MOV    DL, '0'        ;是 0 则直接输出.
_1_:    MOV    AH, 2
        INT    21H
_E_:    RET
PRINTAX ENDP
;-------------------------------

;-------------------------------输出回车换行
PRINTCRLF PROC
 LEA DX,CRLF
 MOV AH,9
 INT 21H
 RET
PRINTCRLF ENDP
;-------------------------------
CODE ENDS
END START
```

3.5.4 实验步骤

1. 使用 notepad++录入源程序，存盘文件名为 SHOP.ASM。

2. 使用 MASM 汇编源文件,即 MASM SHOP.ASM；

3. 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。

4. 使用连接程序 LINK.EXE 将汇编生成 SHOP.OBJ 文件连接成执行文件。 即 LINK SHOP.OBJ；

5. 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 SHOP.EXE 文件。

6.运行程序，观察运行结果是否与理论一致，若不一致修改程序重新编译，若一致执行步骤 7

7.修改程序让 9 号调用信息显示在自己希望的位置

8.观察 9 号调用待显示字符串结尾没有’$’字符会怎样，即删去数据段中字符串的’$’字

符

9. 10 号功能调用时，输入的字符数超过定义的数量时，观察它是如何处理的

10.使用 TD 单步执行观察程序是否一致

11.选取特殊的值，观察结果溢出的情况

3.5.5 实验记录与分析

1.如图运行程序后，会提示用户输入姓名

图 3.5.5 进入程序界面

2.如图 3.5.6 当以回车作为输入时，系统会直接进入到功能 3 并提示用户输入商品名称，以商品‘PEN‘作为输入时，系统会输出商品名称并回到功能 1（1）。如图 3.5.7，当输入的商品名不正确时，需要重新输入，直到输入正确为止

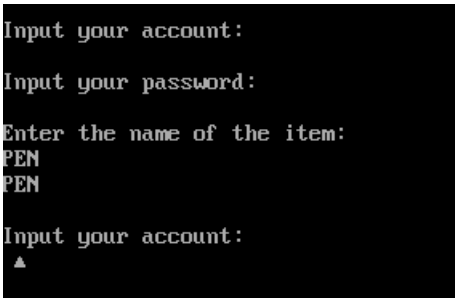


图 3.5.6 顾客演示

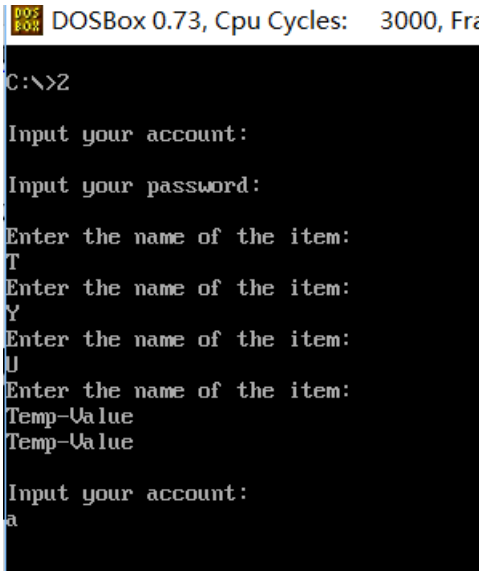


图 3.5.7 商品名输入演示

3.当以老板身份进行登陆时，只有输入正确的姓名和密码后才会进入功能 3，如图 3.5.8，输入了正确的姓名和错误的密码；图 3.5.9，错误的姓名和正确的密码；图 3.5.10，正确的姓名和密码；

图 3.5.8 姓名√密码×

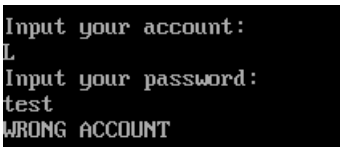


图 3.5.9 姓名×密码√

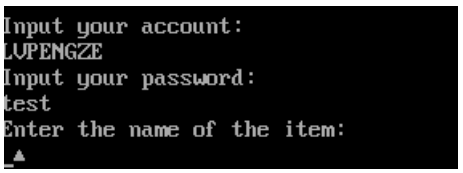


图 3.5.10 姓名√密码√

若是在姓名栏输入了'q',则系统会退出，结果如图 3.5.11 所示

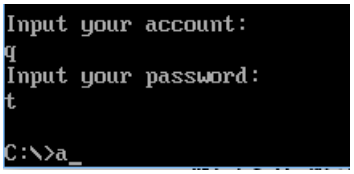


图 3.5.11 程序退出演示

4.以老板身份进入功能 3 后，输入商品名称会给出商品利润率和商品评级，已知商品利润率的理论计算结果如表 3.5.1 所示

表 3.5.1 商品信息及利润

	进货价	销售价	进货数	销售数	进货总价	销售总价	利润	利润率	评级
PEN	35	56	70	25	2450	1400	-1050	-42%	F
BOOK	12	30	25	5	300	150	-150	-50%	F
Temp-Value	15	20	30	2	450	40	-410	-91%	F

如图 3.5.12，图 3.5.13，图 3.5.14 分别为三种商品的计算结果，可知是符合理论结果的

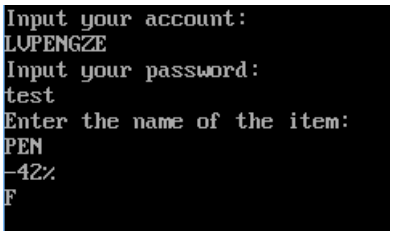


图 3.5.12 PEN 的利润率

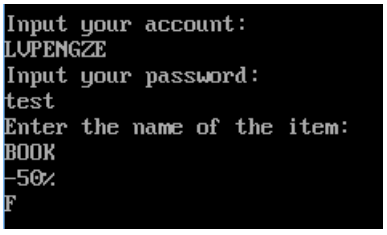


图 3.5.13 BOOK 的利润率

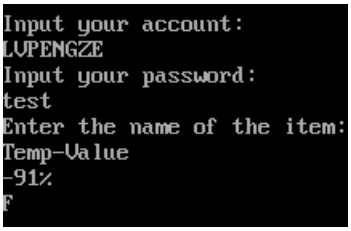


图 3.5.14 Temp-Value 的利润率

5.修改销售量后以测试所有的等级判断，修改后的理论结果如表 3.5.2 所示

表 3.5.2 修改后理论结果

	进货价	销售价	进货数	销售数	进货总价	销售总价	利润	利润率	评级
PEN	35	56	70	70	2450	3920	1470	60%	B
BOOK	12	30	25	25	300	750	450	150%	A
Temp-Value	15	20	30	30	450	600	150	33%	C
PEN	35	56	70	50	2450	2800	350	14%	D

根据测试结果和理论数据比较可知，程序运算正确

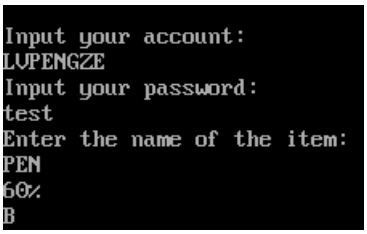


图 3.5.15 样例 1 测试结果

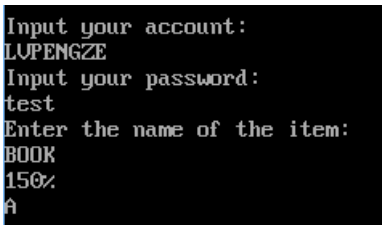


图 3.5.16 样例 2 测试结果

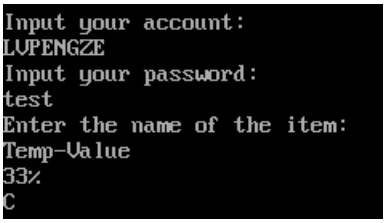


图 3.5.17 样例 3 测试结果

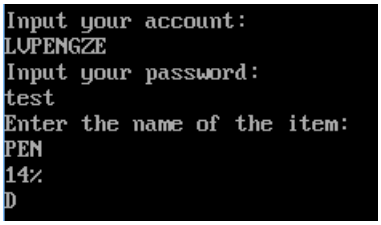


图 3.5.18 样例 4 测试解雇

3.5.6 思考题

1. 如何让 9 号功能调用显示的信息放在自己希望的位置？

可以提前输出一定数量的空格，调整输出位置

2. 如果在 9 号功能调用时，带显示字符串的结尾没有“$”结束符会怎样？

系统会一直输出，知道恰好遇到’$’字符停止

3. 如果在 9 号功能调用前，未对 DS 赋值，或者未对 DX 给予正确的值，结果会如何？

执行一定步骤之后系统会异常退出

4. 10 号功能调用时，输入的字符数超过定义的数量时，它是如何处理的？

当达到指定数量时无法继续输入

5. 匹配姓名时，如何提高匹配速度？

可以先比较字符串长度，若不一致直接判断不相同

4. 总结与体会

通过任务 1，我掌握了编译汇编源程序的方法和汇编调试程序 TD 的用法，学会了在 TD 中修改指令并执行然后观察运行结果，使我对计算机的底层实现有了更加深刻的认识。令我最深刻的是 TD 强大的功能，通过 TD 我们可以直接修改程序，通过单步调试观察程序执行结果我们可以发现错误并改正，学会使用 TD 对我们以后的汇编编程有巨大的帮助。通过任务 2 和 3，我明白了使用 TD 查看数据，在任务 4 中，我对汇编的几种寻址方式有了更加深刻的理解。当然，在这次实验中我也发现了自己的一些问题：

由于是第一次进行实验，时间较赶，对基础知识的把握理解不够，课堂上积累问题较多，导致预习没有作好，在实际上机过程中效率低下，尤其是花了许多时间在软件的配置上，在今后的实验中，我一定要充分地做好预习工作，记录预习过程中遇到的问题，这样在上机时就能有目的有计划地向老师请教，提高上机效率。

在编写程序过程中，要注意一些细节问题，尤其是注意符号的问题，不要输入成中文符号，此外，在定义字符串时一定要记得用'$'作为结束符。

任务 5 难度较高，程序写的比较长，在 C 语言一行代码就能写完的功能在汇编中往往要写很多行，无形之中增加了难度。但一旦上起手来，发现也没有那么困难，从头到尾都是对寻址方式和字符串比较的运动，只要理解了寻址方式和比较，任务 5 也就迎刃而解了。通过任务 5，我明白了在写程序前一定要做好规划设计，尤其是寄存器的分配，否则很容易就绕迷自己。

总的来说，通过这两次的上机实验，我初步理解了汇编程序的书写方法，对计算机底层实现有了初步的认识，也让我发现了自己对实验准备不够充分的缺点，在接下来的几次实验里，我一定要在课前好好准备实验，在上机实验时有目的的实验，提高自己的效率。

5. 参考文献

[1] 王元珍,曹忠升,韩宗芬.80X6 汇编语言程序设计.版本(第 1 版).武汉.华中科技大学出版社,2005 年 4 月.

[2]URL:https://zhidao.baidu.com/question/274923703.html