目录

汇 编 语 言 程 序 设 计 实 验 报 告

1 实验目的与要求

(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 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。

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

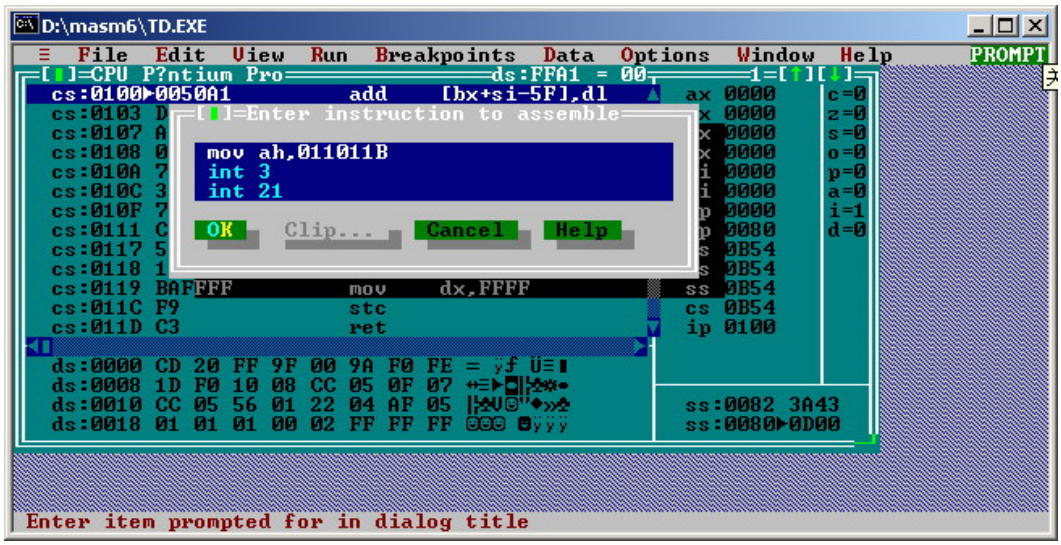

图 2-1 TD 中输入指令示例

汇 编 语 言 程 序 设 计 实 验 报 告

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。

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，但对数据段中变量访问时所用到的变址寄存器采用 32 位寄存器。

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

(3)在 TD 代码窗口中观察并记录机器指令代码在内存中的存放形式，并与 TD 中提供的反汇编语句及自己编写的源程序语句进行对照，也与任务 2 做对比。（相似语句记录一条即可，重点理解机器码与汇编语句的对应关系，尤其注意操作数寻址方式的编码形式，比如寄存器间接寻址、变址寻址、32 位寄存器与 16 位寄存器编码的不同、段前缀在代码里是如何表示的等）。

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

任务 4. 设计实现一个网店商品信息管理的程序。

1、实验背景

有一个老板在网上开了 1 个网店 SHOP，网店里有 n 种商品销售。每种商品的信息包括：商品名称（10 个字节，数据段中定义时，名称不足部分补 0），折扣（字节类型，取值 0~10；0 表示免费赠送，10 表示不打折，1~9 为折扣率；实际销售价格=销售价*折扣/10），进货价(字类型)，销售价（字类型），进货总数（字类型），已售数量（字类型），推荐度【=（进货价/实际销售价格+已售数量/（2*进货数量））*128，字类型】。老板管理网店信息时需要输入自己的名字（10 个字节，数据段中定义时，不足部分补 0）和密码（6 个字节，数据段中定义时，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）可以查看网店中每个商品除了进货价以外的信息。

例如：

```
BNAME  DB   ‘ZHANG SAN’,0  ；老板姓名
BPASS  DB   ‘test’，0，0    ；密码
N     EQU   30
SNAME  DB   ‘SHOP’,0              ；网店名称，用 0 结束
GA1    DB    ‘PEN’， 7 DUP(0) ， 10  ；商品名称及折扣
       DW   35，56，70，25，？       ；  推荐度还未计算
GA2    DB    ‘BOOK’， 6 DUP(0) ， 9  ；  商品名称及折扣
       DW   12，30，25，5，？        ；推荐度还未计算
GAN  DB N-2 DUP( ‘Temp-Value’,8，15，0，20，0，30，0，2，0，？，？）;除了 2 个已经具体定义了的商品信息以外，其他商品信息暂时假定为一样的。
```

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(第一行显示将要访问的网店名称)。

（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：计算指定商品的推荐度。

（1）提示用户输入要查询的商品名称。若未能找到该商品，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）在下一行显示该商品的名称，然后回到功能一（1)。

（4）计算该商品的推荐度， 然后进入功能四。

5、功能四：将功能三计算的推荐度进行等级判断，并显示判断结果。

（1）等级显示方式：若推荐度大于 100，显示“A”；大于 50，显示“B”；大于 10，显示“C”；其他，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

3.1.1 任务 3 源程序

```
.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 EDI, 0
      MOV EBX, 0
      MOV EBP, 0
      MOV CX,10
LOPA:
      MOV AL,BUF1[ESI]
      MOV BUF2[EDI], AL
      INC AL
      MOV BUF3[EBX], AL
      ADD AL, 3
      MOV DS:BUF4[EBP], AL
      INC ESI
      INC EDI
      INC EBP
      INC EBX
      DEC ECX
      JNZ LOPA
      MOV AH, 4CH
      INT 21H
CODE ENDS
      END START
```

3.1.2 关键的实验记录与分析

（1） 任务一

1) 实验步骤

1. 打开 DOSBOX 进入 TD 进行调试。
2. 输入 MOV AH,X1　　ADD AH X2 (SUB AH,X2)　　将 X1 与 X2 两数相加或者相减。
3. 预测相应的 AH,SF,OF,CF,ZF 结果
4. 点击 trace(F7)进行单步调试，观察 AH,SF,OF,CF,ZF 的结果，验证是否与猜测一致

2) 结果预测

加法结果如下：

表 3-1 任务 1 加法结果预测表

	(AH)	(SF)	(OF)	(CF)	(ZF)
X_1 = +0110011B X_2 = +1011010B	8DH	1	1	0	0
X_1 = -0101001B	7AH	0	1	1	0

X_2 = -1011101B					
X_1 = +1100101B X_2 = -1011101B	08H	0	0	1	0

减法结果如下表：

表 3-2 任务 1 减法预测表

	(AH)	(SF)	(OF)	(CF)	(ZF)
X_1 = +0110011B X_2 = +1011010B	0D9H	1	0	1	0
X_1 = -0101001B X_2 = -1011101B	34H	0	0	0	0
X_1 = +1100101B X_2 = -1011101B	0C2H	1	1	1	0

3) **验证截图与分析说明**

a) X_1 = +0110011B，X_2 = +1011010B 时，说明如下：

X_1 补码为 00110011，X_2 补码为 01011010，进行补码加减分别有：

加法时候 AH=8DH，SF=1(最高位为 1)，OF=1(有溢出)，CF=0(无进位)，ZF=0(不为 0)，这与手算结果相符，正确

减法时候 AH=0D9H，SF=1(最高位为 1)，OF=0(无溢出)，CF=1(有借位)，ZF=0(不为 0) ，这与手算结果相符，正确

截图分别如下：

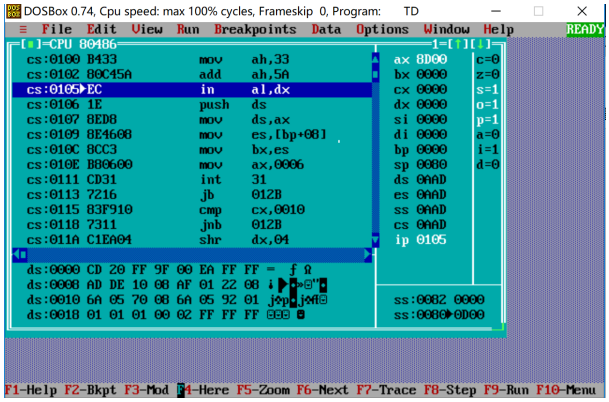

图 3-1 样例一加法结果截图

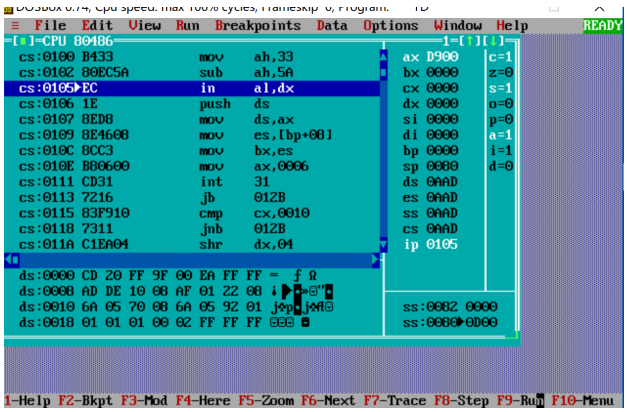

图 3-2 样例一减法结果截图

b) X1 = -0101001B，X2 = -1011101B 时，说明如下：

X_1 补码为 11010111，X_2 补码为 10100011，进行补码加减分别有：

加法时候 AH=7AH, SF=0(最高位为 0)，OF=1(有溢出)，CF=1(有进位)，ZF=0(不为 0)，这与手算结果相符，正确

减法时候 AH=34H, SF=0(最高位为 0)，OF=0(无溢出)，CF=0(无借位)，ZF=0(不为 0)，这与手算结果相符，正确

截图分别如下：

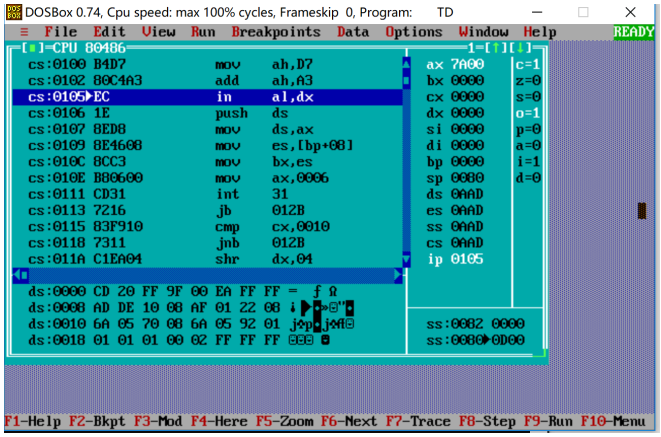

图 3-3 样例二加法结果截图

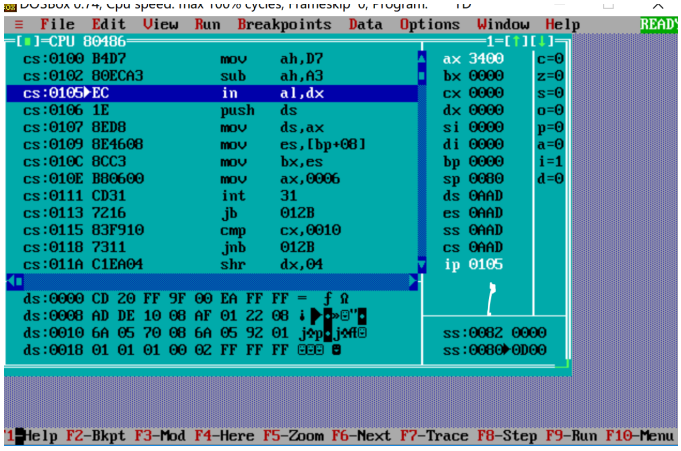

图 3-4 样例二减法结果截图

c) X1 = +1100101B，X2 = -1011101B 时，说明如下：

X_1 补码为 01100101，X_2 补码为 10100011，进行补码加减分别有：

加法时候 AH=08H, SF=0(最高位为 0)，OF=0(无溢出)，CF=1(有进位)，ZF=0(不为 0)，这与手算结果相符，正确。

减法时候 AH=0C2H, SF=1(最高位为 1)，OF=1(有溢出)，CF=1(有借位)，ZF=0(不为 0)，这与手算结果相符，正确。

加法和减法的截图分别如下：

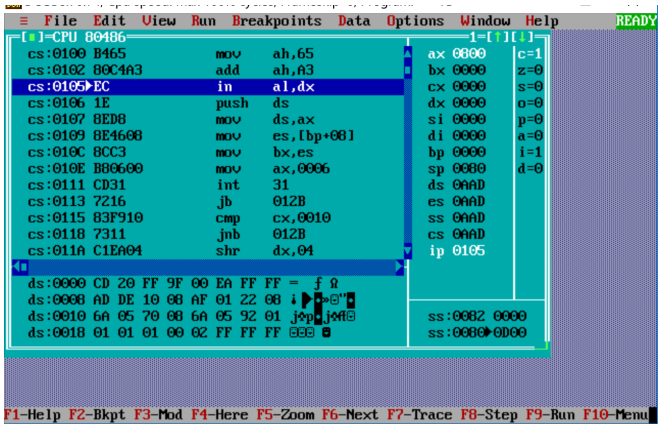

图 3-5 样例三加法结果截图

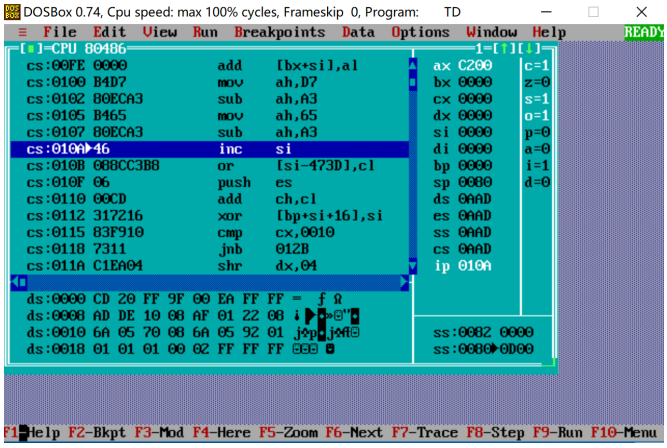

图 3-6 样例三减法结果截图

4) 关于汇编语言二进制数加减的思考与总结

通过以上实验，主要总结出以下几点：

1. 有符号数在机器内均采用补码表示，运算的时候不再管符号而当作将整个数作为无符号数处理，符号为也一起参加运算。

2. 如果做减法运算，判断 CF 是 0 还是 1 需要把 X_1 的补码和 X_2 的补码均看成无符号数，看 X_1 的补码和 X_2 的补码的大小关系，如果 X_1 的补码大小大于 X_2 的补码的大小，则说明没有借位，CF=0，否则则说明更需要借位 CF=1。

3. 若出现两正数相加等于负数或者两负数相加等于正数，则一定出现了溢出情况，OF=1。

（2） 任务二

1) 实验步骤

1. 录入源程序，文件后缀名为.ASM，用 MASM 汇编代码生成.OBJ 文件并 LINK.OBJ 文件生成.EXE 文件。

2. 在 DOSBOX 中用 TD.EXE 调试生成的.EXE 文件，进行单步调试（Ctrl+F7），记录执行到“MOV CX,10”以及“INT 21H”之前的(BX),(BP),(SI),(DI)各是多少。

3. 猜想程序执行到退出之前数据段开始 40 个字节的内容。

4. 记录程序执行到退出之前数据段开始 40 个字节的内容，验证其与猜想是否一致。

2) 步骤 2 的试验记录与有关的分析说明

每一个 BUF 变量长度均为 10 字节，故我们作出预测，从程序开始到运行到 MOV CX, 10 时，分别有(BX)，(BP)，(SI)，(DI)列出表格如下：

表 3-3 预测从程序开始到运行到 MOV CX, 10 表格

语句	(BX)	(BP)	(SI)	(DI)
MOV AX, DATA	0000	0000	0000	0000
MOV DS, AX	0000	0000	0000	0000
MOV SI, OFFSET BUF1	0000	0000	0000	0000
MOV DI, OFFSET BUF2	0000	0000	0000	000A
MOV BX, OFFSET BUF3	0014	0000	0000	000A
MOV BP, OFFSET BUF4	0014	001E	0000	000A
MOV CX, 10	0014	001E	0000	000A

下面给出执行到 MOV CX,10 之前的程序截图，可以看出，该结果与预测结果相符：

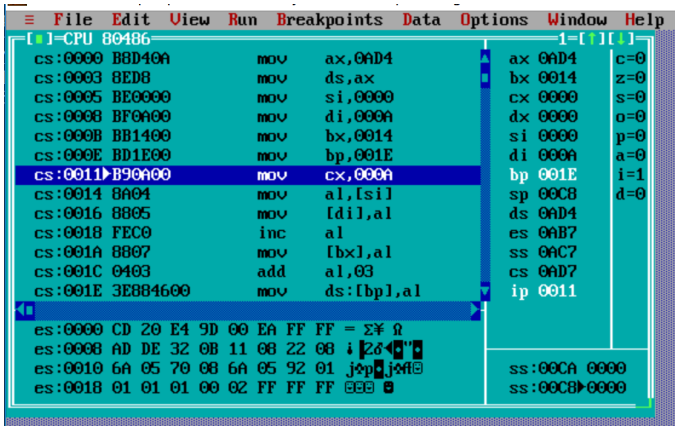

图 3-7 运行到 MOV CX, 10 时的截图

从标志 LOPA 开始，实际上是 10 个循环（CX=10），在每个循环执行一下操作：

把 SI 偏移地址所存的值送进 AL，把 AL 的值送进地址偏移值为[DI]的位置，AL 的值加一。把 AL 的值送入[BX]的位置，然后 AL 的值增加 3，再把 AL 的值送入 DS:[BP]，之后 SI,DI,BP,BX 分别增加 1，CX 减少 1，CX 没减少到 0 的时候继续循环。

这样做的结果是使得 BUF2 中的数复制 BUF1 中的数，BUF3 中的数等于 BUF1 加一，BUF4 中的数等于 BUF1 加 4。

根据这个循环特点，我们给出运行到 INT 21H 之前的结果为(BX)=001E，(BP)=0028，(SI)=000A，(DI)=0014

下面给出执行到 INT 21H 之前的程序截图，可以看出，该结果与预测结果相符：

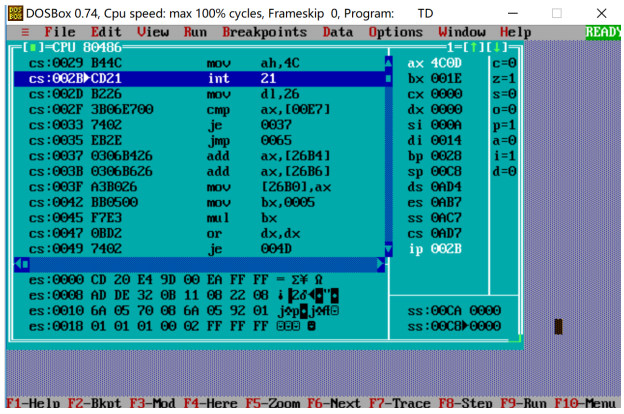

图 3-8 程序运行到最后一步的截图

由上述分析可知，BUF2 中数不变，BUF2 中的数完全复制 BUF1 的数，BUF3 的数为 BUF1 的数加一，BUF4 中的数为 BUF1 加四。我们给出执行到退出之前数据段开始 40 个字节的内容猜测如下：

表 3-4 推出前数据段前 40 个字节内容猜测

00	01	02	03	04	05	06	07
08	09	00	01	02	03	04	05
06	07	08	09	01	02	03	04
05	06	07	08	09	0A	04	05
06	07	08	09	0A	0B	0C	0D

运行到 INT 21H 时右键选择 goto 输入 DS:0，跳转到 DS 段。得到如下截图，由截图可得到，实验得到的结果和预测相符（由于 DS 段只能显示 4 行，而一共有五行，故截图两张，第一张为 1-4 行，第二张为 2-5 行）

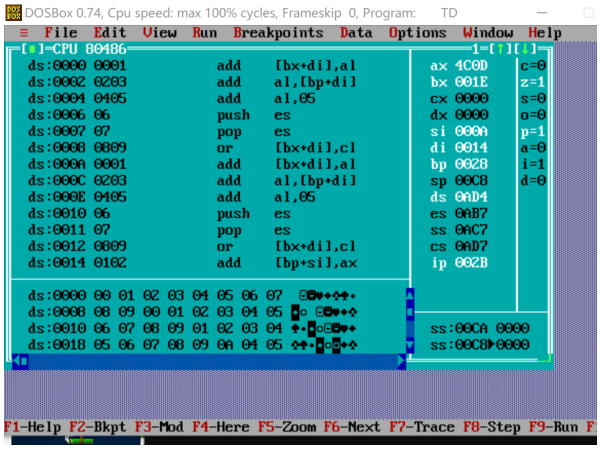

图 3-9 运行完成后数据段前 4 行截图

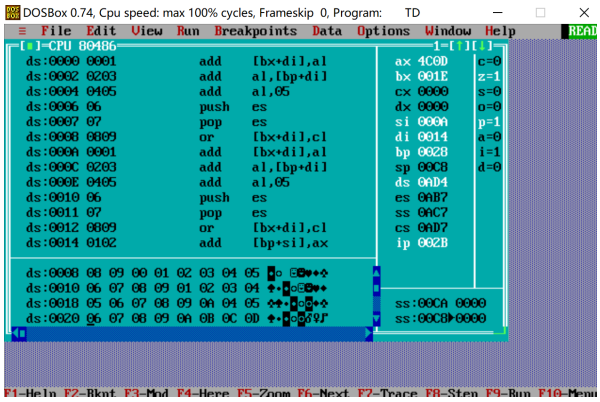

图 3-10 运行完成后数据段 2-5 行截图

3) 实验总结注意事项

1. 在实验调试过程中难免会遇到意外推出的情况，再重新启动 td 调试的时候可以灵活的运用 Here(CntrlF4)跳转到对应语句，观察该语句下的有关结果。

2. 在查看数据段的时候一定记得先用 goto 输入 DS:0 跳转到数据段，我在实验中曾因为没有输入这一句话得到了错误的结果。

（3） 任务 3

1）实验步骤

1. 录入源程序，对数据段中变量访问时所用到的变址寄存器采用 32 位寄存器（把 DI，SI，BP，BX 改为 EDI，ESI，EBP，EBX），将源程序进行汇编（MASM），连接(LINK)，得到.exe 文件。（源程序在 3-1 给出）

2. 给出程序结束之前数据段开始前 40 个字节的内容的预测并且检查程序是否与设想一致。

3. 观察 TD 中机器指令码在内存中的存放形式，将反汇编语句与源程序语句进行对照，也与任务 2 中 16 位的语句对照

4. 观察从不同字节开始反汇编会怎样，解释 IP/EIP 指明指令起始位置的重要性。

2）数据段前 40 个字节内容

a) 内容预测

预测内容与 16 位寄存器的一致，即为上一节的表 3-4，分析方式与得出表 3-4 的方式一致。

b) 程序验证

由下面两张截图可知，运行后的数据段前 40 个数字的结果与设想完全一致，设想得证。可知，使用 32 位寄存器与 16 位寄存器在该题中对于数据的变化没有差别。

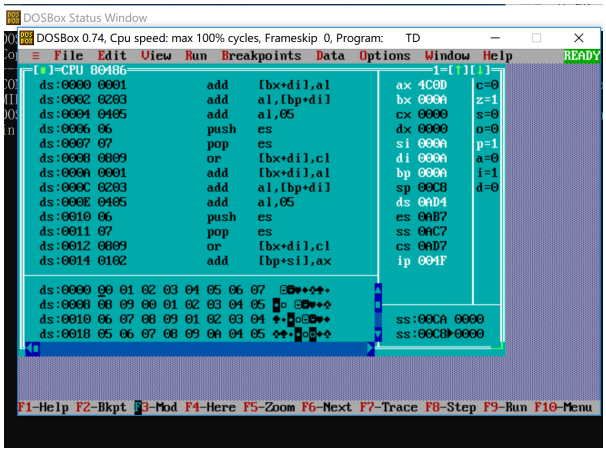

图 3-11 32 位寄存器数据段前四行截图（运行结束前）

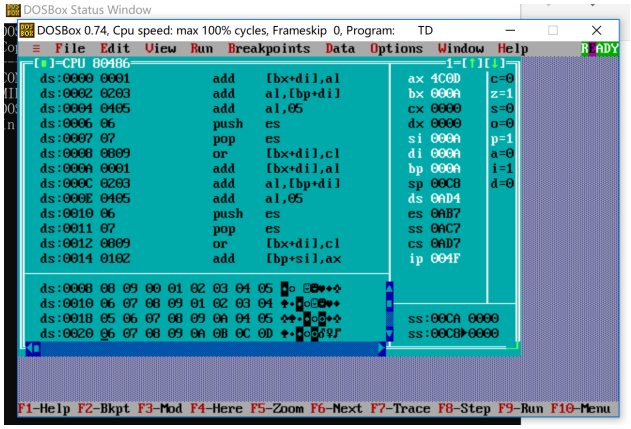

图 3-12 32 位寄存器数据段 2-5 行截图（运行结束前）

3） 16 位与 32 位汇编指令机器码的对比与结果分析

下面给出 LOPA 中循环语句的 16 位和 32 位的汇编指令机器码对比：

表 3-5 16 位与 32 位机器码对比

汇编指令	16 位机器码	32 位机器码
16 位：MOV AL, [SI]	8A04	----
32 位：MOV AL,BUF1[ESI]	----	678A8600000000
16 位：MOV [DI], AL	8805	----
32 位：MOV BUF2[EDI], AL	----	6788870A000000
INC AL	FEC0	FEC0
16 位：MOV [BX], AL	8807	----
32 位：MOV BUF3[EBX], AL	----	67888314000000
ADD AL, 3	0403	0403
16 位：MOV DS:[BP], AL	3E884600	----
32 位：MOV DS:BUF4[EBP], AL	----	673E88851E0000
16 位：INC SI	46	----

32 位：INC ESI	----	6646
16 位：INC DI	47	----
32 位：INC EDI	----	6647
16 位：INC BP	45	----
32 位：INC EBP	----	6645
16 位：INC BX	43	----
32 位：INC EBX	----	6643
16 位：DEC CX	49	----
32 位：DEC ECX	----	6649

结果分析如下：

1. 将 16 位扩展成 32 位之后，一些指令因为扩展需要，发生了变化，比如后面的 INC SI 变为 INC ESI 之后，指令码由 46 变为 6646，对于 INC DI,INC BP,INC BX,DEC CX 都是同理的，都是在 16 位机器码之前加了 66.

2. 然而对于一些没有扩展成 32 位的语句，还是维持原有机器码，例如 INC AL 保持 FEC0 不变

3. 对于例如 MOV BUF2[EDI], AL 这样的语句的机器码特点。其机器码为：6788870A000000，其中由实际意义的为高 8 位，其中高 8 位的最后两位 0A 代表着 BUF2 的偏移地址，即被加上去的地址。前面 6 位代表着 32 位的该操作本来的机器码（没有变址寻址，只有寄存器简介寻址的的情况下）。对于其他的也成立，例如语句 MOV BUF3[EBX], AL 的机器码为 67888314000000，其中 14 代表 BUF3 的偏移地址，前面 6 为代表原来的无偏移的操作码。

5） 从不同字节位置开始反汇编实验分析与 IP/EIP 指明起始位置重要性分析

a） 首先我们按照正常情况，不改变 ip 的起始位置，执行到 MOV CX,10,截图如下：

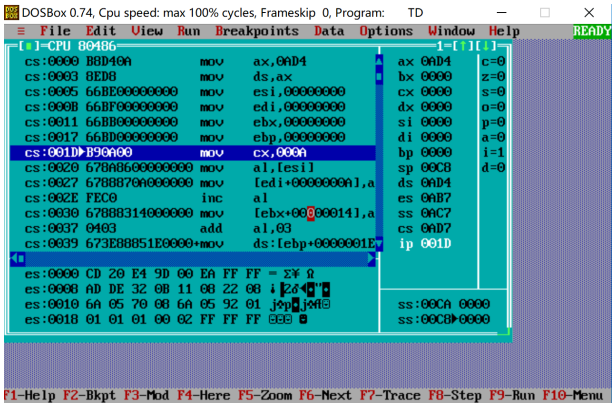

图 3-13 不改变起始 ip 的运行截图

b） 然后我们将起始 IP 设定为 000AH，依然单步调试到 MOV CX,10，运行截图如下：

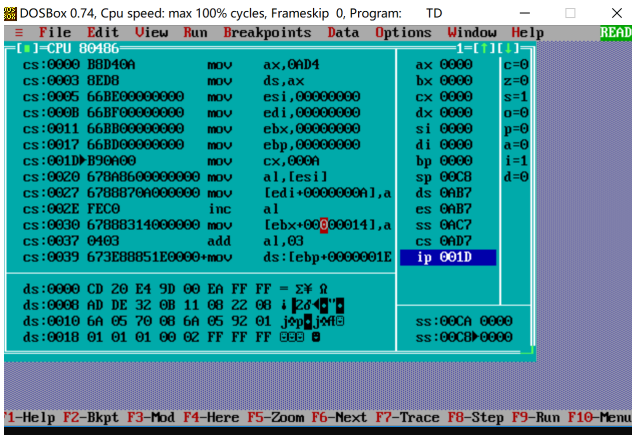

图 3-14 改起始 IP 为 000AH 之后的运行截图

c） 对比两张图我们可以总结起始 IP 错误导致的结果以及可能产生的危害：

AX 寄存器没有获得正确的值（0AD4），并且 AX 的值也没有正常传入 DS 导致数据段起始地址错误。

DS 地址的错误将会影响后续的地址偏移，导致 DS:[EDI]等语句的错误。

综上，若没有从正确的 IP 开始执行调试，会导致整个程序全部出错，从正确的 IP 开始调试程序显得相当重要。

3.2 任务 4

3.2.1 设计思想及存储单元分配

1）设计思想

1. 在数据段中存入老板姓名（BNAME）、密码(BPASS)、商店名称(SNAME)、商品信息等基本内容。

2. 在数据段中定义数据 AUTH（判断是老板还是顾客）、定义三个输入缓冲区，用来接收用户输入的姓名（IN_NAME）、密码(IN_PWD)和商品信息(IN_ITEM)

3. 在数据段中定义其它变量存储，方便推荐度的计算，同时存入需要输出的提示信息（例如提示用户输入姓名、密码、商品名称等）

4. 进入到代码段。完成功能一。验证用户输入姓名。利用循环一个个判断用户输入的是否正确，直到标准姓名遇到 0，则重新输入，输入 q 则跳到 EXIT 退出，输入回车直接跳到查询。

5. 完成功能二。判断输入密码是否正确。利用循环结构逐个判断标准密码与输入密码直到标准密码遇到 0。若成功，将 AUTH 的值设置为 1，输出成功的提示信息并换行，进入功能 3，若输出错误则返回功能一。

6. 完成功能三。判断输入的商品是否存在。利用双重循环结构。外层循环是每个商品，循环数从 N 开始递减，内层循环是每个商品名称判断，若循环到输入回车且标准输入是 0 则找到商品跳出循环，否则，外层循环减一，跳到下一个商品。当外层循环数为 0 时，失败，重新查找。若是找到了，则判断 AUTH 值，若是为 1 则下一行显示该商品名称，若是为 0，则计算商品推荐度，

进入下一个功能。

7. 关于推荐度的计算，将式子分为两个部分，第一部分为(进货价*128/实际销售价格)，实际售价的算法为售价*折扣/10。第二部分为（已售数量*64）/进货数量，最后再将这两部分相加。得到推荐度。

8. 比较得到的推荐度，利用分支结构，若推荐度大于 100 则输出 A，大于 50 输出 B,大于 10 输出 C，其他显示 F，并跳回输入名称的位置（功能一）。

2) 存储单元分配

a) 寄存器

1. CX：主要用来计数，判断循环，有些情况下也用于存放数据
2. DX: 存放缓冲区基地址，存放数据
3. AX：用于存放数据
4. BX：主要用于存放数据
5. DI 和 SI：主要用于存放偏移地址
6. AL、AH、BL、BH：主要用于存放字节型数据，有些情况下也用于存放偏移地址

b) 存储器

1. 变量 BANME 和 BPASS 存入老板姓名和密码
2. GA1,GA2…GAN 存入商品信息
3. AUTH 存入登录状态
4. IN_NAME,IN_PWD,IN_ITEM 分别为姓名、密码、商品名称的输入缓冲区
5. PUTNAME,NOTITEM,PUTPASSWORD,ITEMNAME,LOGFAIL,LOGINREMIND 分别为提示信息。

3.2.2 流程图

由于整个工程比较庞大，我将整个流程分为 4 个模块，分别为姓名输入模块、密码输入模块、查询模块、推荐度计算模块，最后我用一个流程图展示了整个调用关系，流程图如下所示。

1) 姓名输入模块

主要功能为接收用户输入的名字并进行匹配判断，具体如下：

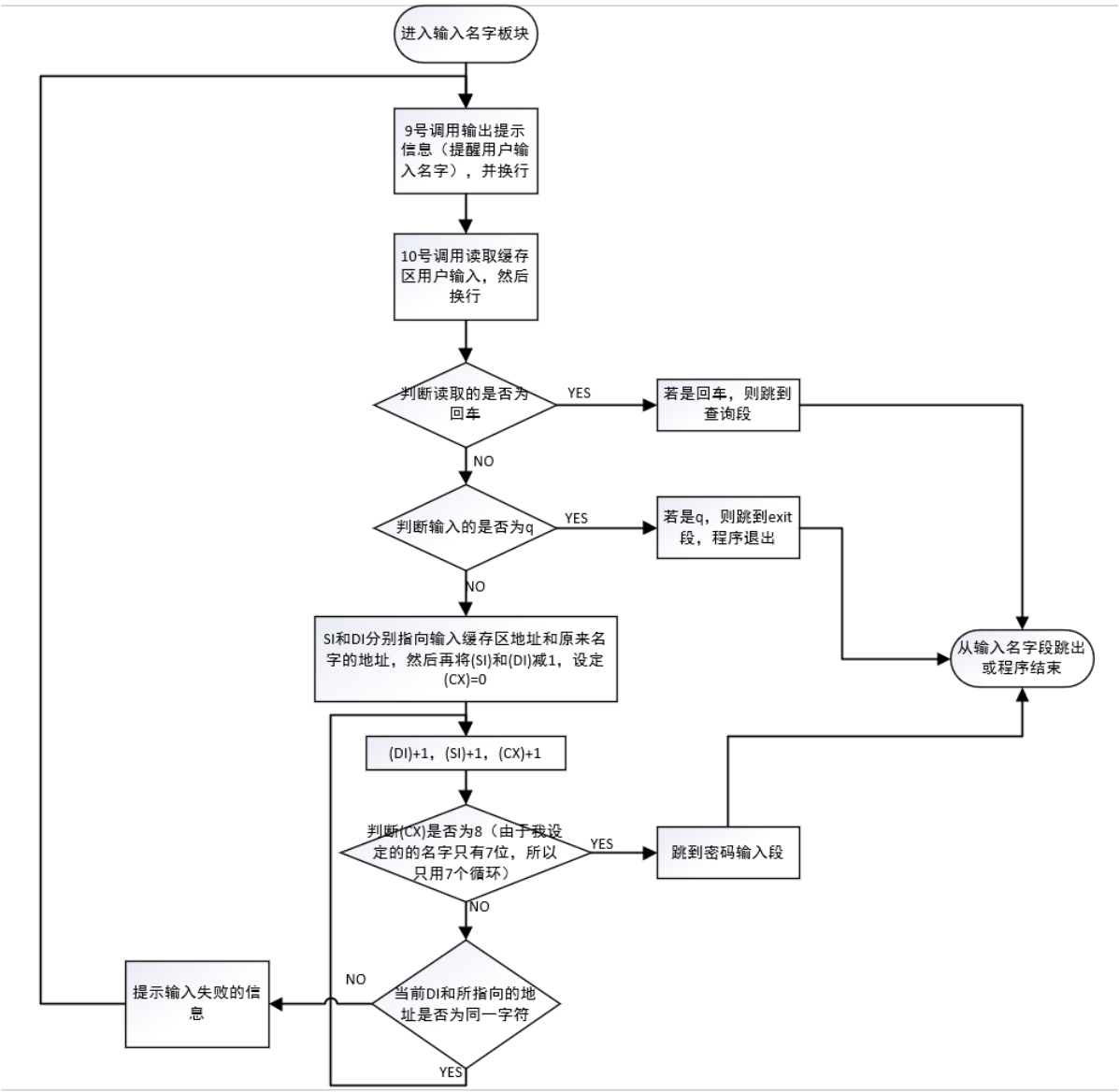

图 3-15 姓名输入模块流程图

2） 密码输入模块

主要功能为接收用户输入的密码并判断，具体如下：

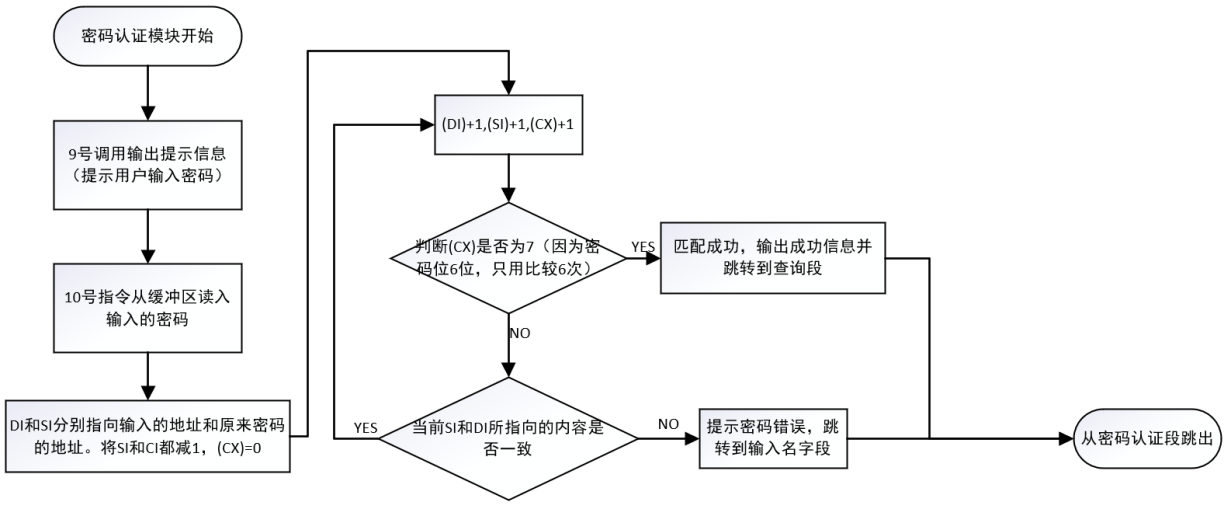

图 3-16 密码输入模块流程图

3） 查询模块

主要功能为接收用户输入的货物名称并查询货物是否存在，具体如下：

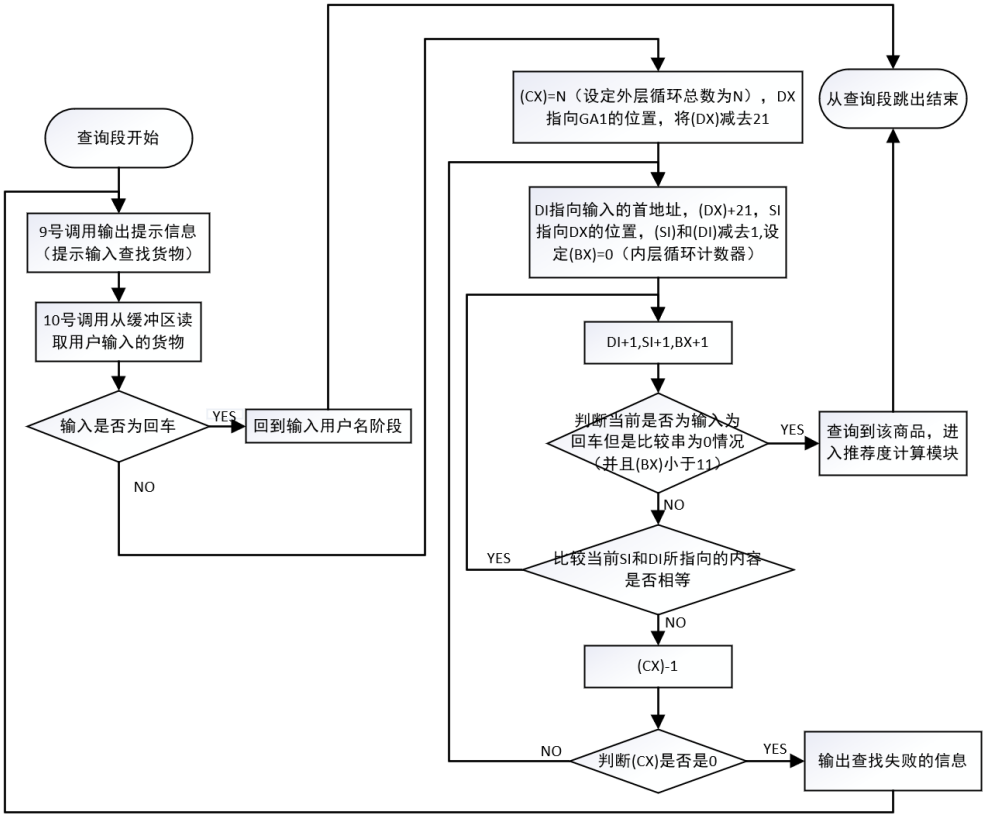

图 3-17 查询模块流程图

4） 推荐度计算模块

主要功能为计算推荐度并输出等级，具体如下：

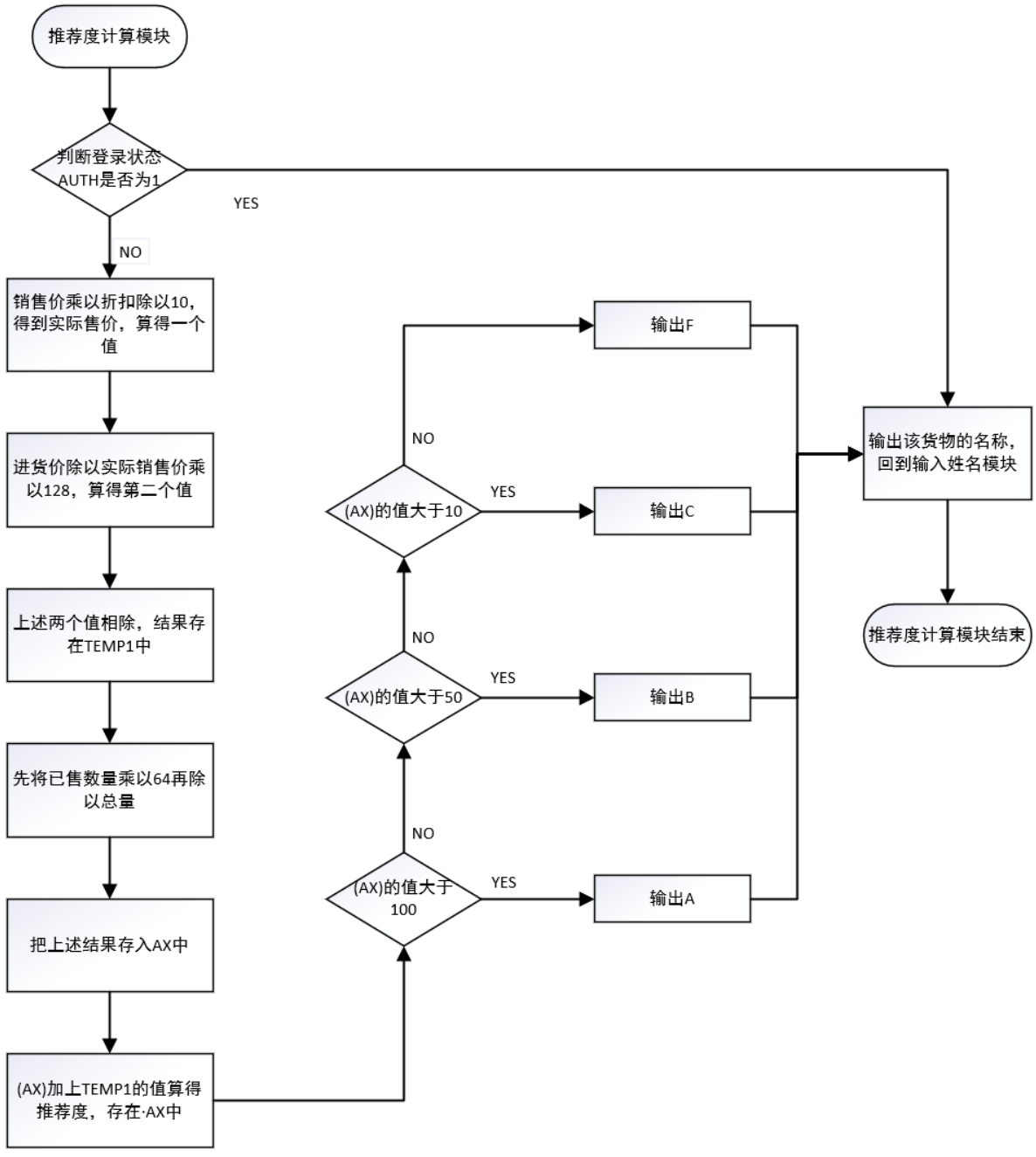

图 3-18 推荐度计算模块流程图

5) 各模块调用关系展示

展示上述四模块的调用关系与整个程序的结构，具体如下：

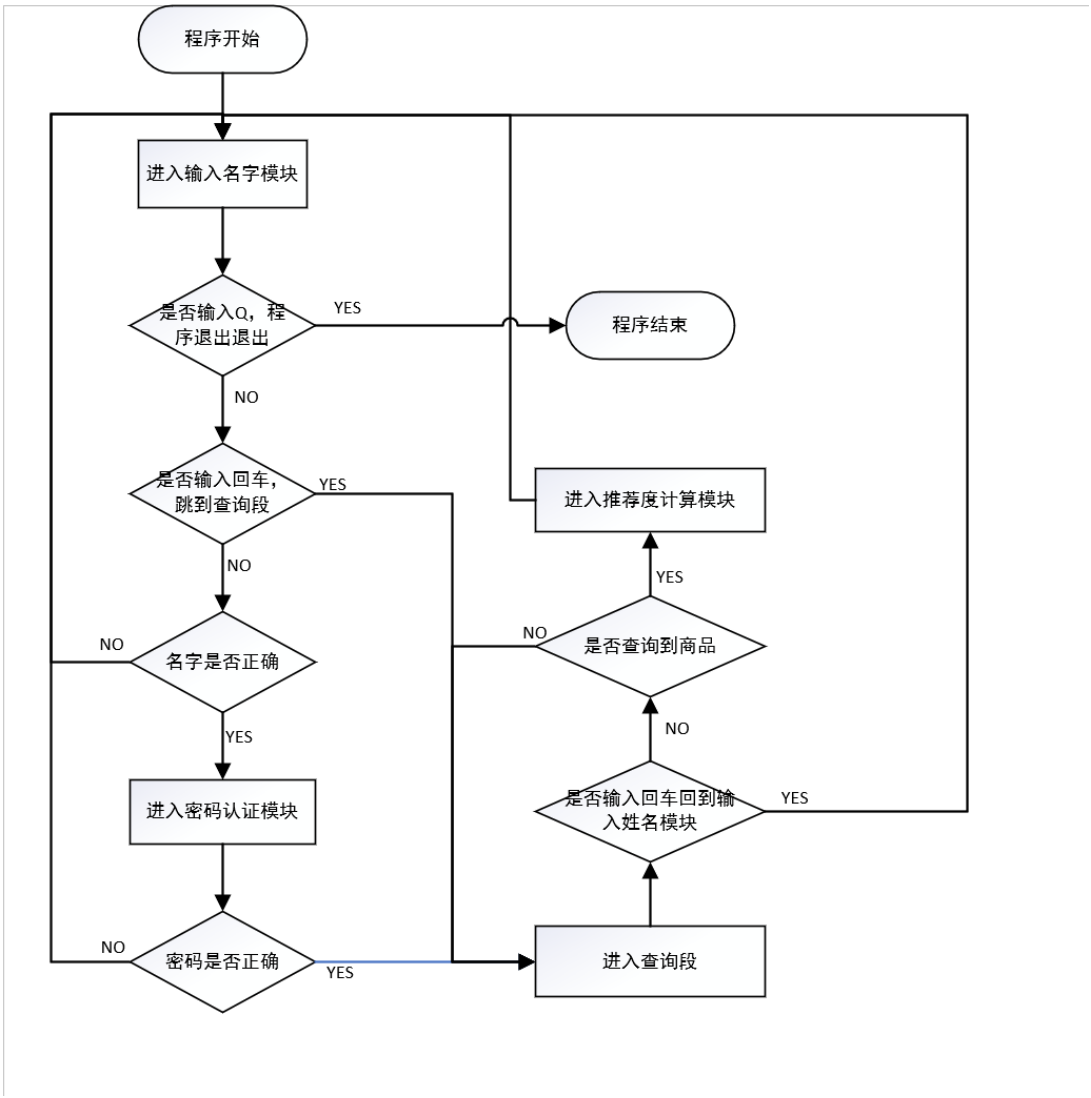

图 3-19 各模块调用关系流程图

3.2.3 源程序

```
.386

STACK SEGMENT USE16 STACK
        DB 200 DUP(0)
STACK ENDS

DATA SEGMENT USE16
    BNAME DB 'WL DING',0,0,0  ;老板姓名（由于我的名字拼音超过了 10 个字节，故名采用缩写）
    BPASS DB '111111' ;密码
    N EQU 30 ;商品数量
    GA1 DB 'PEN',  7 DUP(0), 10 ;商品名称及折扣
                DW 35, 56, 70, 25, ? ;推荐度还未计算
```

```
    GA2 DB 'BOOK', 6 DUP(0), 9 ;商品名称及折扣
                    DW 12, 30, 25, 5, ? ;推荐度还未计算
        GA3 DB 'TEST1',5 DUP(0), 9
                 DW 500,30,25,5,? ;设置比较大的进货量，防溢出测试
        GA4 DB 'TEST2',5 DUP(0), 10
                  DW 0,30,25,0,? ;进货价和已销售量设置为0，看能否正常输出
    GAN DB N-4 DUP('Temp-Value',8)
              DW N-4 DUP(12,30,25,5,?) ;除了几个已经具体定义了的商品信息以外，其他商品信息暂时假
定为一样的

    AUTH DB 0
    TEMP1 DW ?
    IN_NAME DB 12
            DB ?
            DB 12 DUP(0)
    IN_PWD DB 10
            DB ?
            DB 10 DUP(0)
    IN_ITEM DB 12
            DB ?
            DB 12 DUP(0)
    PUTNAME DB 'PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)$'
    NOTITEM DB 'DO NOT FIND THE ITEM$'
    PUTPASSWORD DB 'PLEASE INPUT THE PASSWORD$'
    ITEMNAME DB 'PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)$'
    LOGFAIL DB 'YOUR USER NAME OR PASSWORD IS WRONG!$'
    LOGINREMIND DB 'IDENTIFICATION GOT!$'
    DATA ENDS

CODE SEGMENT USE16
    ASSUME CS:CODE, DS:DATA
START:
    MOV AX,DATA
    MOV DS,AX ;设置数据段寄存器的值

INNAME:
    MOV AUTH,0
    LEA DX,PUTNAME ;提示用户输入名字
    MOV AH,9H
    INT 21H
    MOV DL, 0AH
    MOV AH,2H
    INT 21H

    LEA DX,IN_NAME
    MOV AH,10
    INT 21H
    MOV DL,0AH
    MOV AH,2H
    INT 21H
    LEA BX,OFFSET IN_NAME

    CMP BYTE PTR [BX+2],0DH ;如果是回车，则直接查询
    JE LOOKUP
    CMP BYTE PTR [BX+2],'Q' ;如果是q就退出
    JE EXIT
```

```
    MOV DI, OFFSET IN_NAME ;指向输入的地址
    ADD DI,2
    MOV SI,OFFSET BNAME ;指向原来匹配名字的地址
    SUB SI,1
    SUB DI,1
    MOV CX,0

LOPA:
    ADD DI,1
    ADD SI,1
    ADD CX,1
    CMP CX,8H
    JE PASSWORD
    MOV BL,BYTE PTR [SI] ;在 BL 中存入真实的姓名
    CMP BL,BYTE PTR [DI] ;将输入值有与真实姓名作比较
    JE LOPA ;如果为真，继续执行下一个比较
    JNE INPUTFAIL ;如不为真，则提示失败

    PASSWORD:
    LEA DX,PUTPASSWORD ;提示用户输入密码
    MOV AH,9H
    INT 21H
    MOV DL,0AH
    MOV AH,2H
    INT 21H

    LEA DX,OFFSET IN_PWD
    MOV AH,10
    INT 21H
    MOV DL,0AH
    MOV AH,2H
    INT 21
    LEA BX,OFFSET IN_PWD

    MOV DI,OFFSET IN_PWD ;指向输入的地址
    ADD DI,2
    MOV SI,OFFSET BPASS ;指向原来匹配密码的地址
    SUB SI,1
    SUB DI,1
    MOV CX,0

LOPB:
    ADD DI,1
    ADD SI,1
    ADD CX,1
    CMP CX,7H
    JE SUCCESSFUL ;如果密码匹配成功，则输出成功信息
    MOV BL, BYTE PTR [SI] ;BL 中存入真实密码
    CMP BL,BYTE PTR [DI] ;输入密码和真是密码比较
    JE LOPB ;若为真比较下一位
    JNE INPUTFAIL;若不为真则提示失败

INPUTFAIL:
    MOV AUTH,0
    MOV DL,0AH
    MOV AH,2H
    INT 21H
```

```
    LEA DX, LOGFAIL
    MOV AH,9
    INT 21H
    MOV DL,0AH
    MOV AH,2H
    INT 21H
    JMP INNAME

SUCCESSFUL:
    MOV AUTH,1
    MOV DL,0AH
    MOV AH,2H
    INT 21H
    LEA DX,LOGINREMIND ;若成功则输出成功的提示信息
    MOV AH,9
    INT 21H
    MOV DL,0AH
    MOV AH,2H
    INT 21H
    JMP LOOKUP

LOOKUP:
    MOV DX,OFFSET ITEMNAME ;提示输入要查找的货物
    MOV AH,9H
    INT 21H
    MOV DL,0AH
    MOV AH,2H
    INT 21H
    LEA DX,IN_ITEM ;用户输入要查找的货物
    MOV AH,10
    INT 21H
    MOV BL,IN_ITEM+1
    MOV BH,0
    MOV BYTE PTR IN_ITEM+2[BX],'$' ;在输入串尾部补上'$'
    MOV DL,0AH
    MOV AH,2H
    INT 21H

    MOV SI,OFFSET IN_ITEM
    CMP BYTE PTR [SI+2],0DH ;判断是否为回车，若是则回到输入用户名
    JE INNAME
        MOV CX,N ;设定循环的总数
    MOV DX,OFFSET GA1


NEXT:
    MOV DI,OFFSET IN_ITEM
    MOV BL,BYTE PTR [DI+1]
    ADD BX,1
    ADD DI,2
    MOV SI,DX
    SUB SI,1
    SUB DI,1

LOPC:
    ADD DI,1
    ADD SI,1
```

```
    SUB BL,1
    CMP BL,0
    JE FINDSUC
    MOV AL,BYTE PTR [SI]
    CMP AL,BYTE PTR [DI] ;逐个判断货物名称是否相符
    JE LOPC
    DEC CX
    ADD DX,21
    CMP CX,0
    JNE NEXT ;若货物没有循环完，判断下一个货物
    JE FINDFAIL ;若循环完了还没有找到，则查找失败

FINDFAIL:
    LEA DX,NOTITEM ;如果没有找到，输出没有找到的错误消息
    MOV AH,9H
    INT 21H
    MOV DL,0AH
    MOV AH,2H
    INT 21H
    JMP LOOKUP ;回到 LOOKUP 重新输入寻找

FINDSUC:
    CMP AUTH,1 ;判断登录状态，若是若已经登陆则显示该商品名称，若不是则计算推荐度
    JE PRINT
    MOV SI,DX
    MOV AL,[SI+10] ;折扣
        MOV AH,0
        MOV BX,[SI+13]  ;销售价
        MUL BX ;销售价乘以折扣
        MOV BX,10
        MOV DX,0
        DIV BX ;销售价乘以折扣除以 10，实际售价
        MOV CX,AX
        MOV AX,[SI]+11 ;进货价
        MOV BX,128
        MUL BX ;进货价乘以 128
        MOV DX,0
        DIV CX ;进货价乘以 128 除以销售价
        MOV TEMP1,AX
        MOV AX,[SI]+17 ;已售数量
        MOV BX,64
        MUL BX ;已售数量乘以 64
        MOV BX,[SI]+15  ;进货数量
        MOV DX,0
        DIV BX ;已售数量乘以 64 除以进货数量
    ADD AX,TEMP1 ;两部分推荐度相加，存入 AX
    CMP AX,100
    JGE LEVELA ;大于 100 输出 A
    CMP AX,50
    JGE LEVELB ;大于 50 输出 B
    CMP AX,10
    JGE LEVELC ;大于 10 输出 C
    JMP LEVELF ;其他输出 F

LEVELA:
    MOV DL,41H
    MOV AH,2
```

```
    INT 21H
    MOV DL,0AH
    MOV AH,2
    INT 21H
    JMP INNAME

LEVELB:
    MOV DL,42H
    MOV AH,2
    INT 21H
    MOV DL,0AH
    MOV AH,2
    INT 21H
    JMP INNAME

LEVELC:
    MOV DL,43H
    MOV AH,2
    INT 21H
    MOV DL,0AH
    MOV AH,2
    INT 21H
    JMP INNAME

LEVELF:
    MOV DL,46H
    MOV AH,2
    INT 21H
    MOV DL,0AH
    MOV AH,2
    INT 21H
    JMP INNAME

PRINT:
    LEA DX,IN_ITEM+2
    MOV AH,9
    INT 21H
    MOV DL,0AH
    MOV AH,2
    INT 21H
    JMP INNAME

EXIT:
    MOV AH,4CH
    INT 21H

CODE ENDS

    END START
```

3.2.4 实验步骤

1. 使用编辑程序 EDIT.EXE 录入源程序，存文件名为 shop.asm。
2. 汇编源程序，即 MASM shop.asm。

3. 思考解决出现的报错，可通过查阅手册弄懂问题。

4. 链接源程序，即 LINK shop.obj。

5. 得到 shop.exe 之后测试其功能是否正常，若不正常找出错误

6. 测试程序。观察输入名字、密码能否正常运行，输入货物名称能否正常匹配并正确计算推荐度。在测试中需要进行边缘测试。主要进行如下边缘测试：

1） 输入名字时输入空串，输入纯数字，输入超过标准名字长度，看能否正常报错

2） 输入密码时输入空串，输入纯字母，输入超多标准长度，看能否正常报错

3） 测试货物名称时输入空串，输入纯数字，输入中带有特殊字符（如*，&，空格），看能否正常报错。

4） 测试推荐度时，将某一商品进货量和已售数量设定为 0，看能否正常匹配推荐等级，将进价得很大，看是否会溢出（理论上字类型存储长度为 2 的 16 次方，由于我还有乘以 128 操作，所以最多能容纳的数为 2 的 9 次方，即 512，边缘测试时，可以将一个值设定在接近 512 时，看是否能正常匹配等级，看是否会溢出）

7. 如果在 9 号功能调用时，带显示字符串的结尾没有“$”结束符会怎样？在源程序中去掉‘$’结束符看程序能否正常运行，若不正常，有什么错误。更改程序后验证。

8. 如果在 9 号功能调用前，未对 DS 赋值，结果会如何？更改程序验证。

9. 注意观察转移指令机器码的编码方法，观察对应标号的偏移地址与该编码之间关系。

3.2.5 实验记录与分析

1. 汇编源程序时，汇编程序报了以下几个个错误。截图如下：

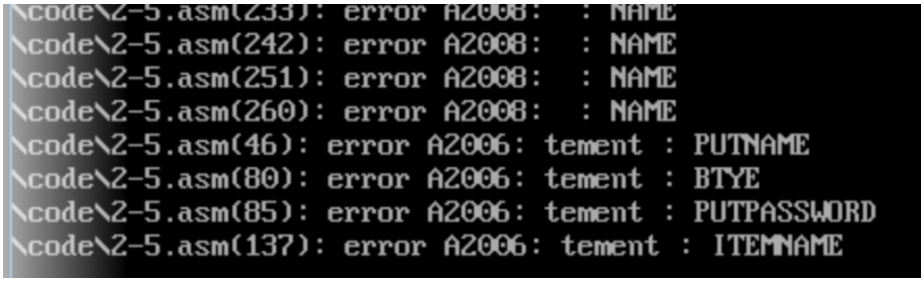

图 3-20 汇编过程中的程序调用

1). 按照提示，进行查阅手册，发现前面几个 NAME 的报错是因为我使用了 NAME 标号，而 NAME 模块定义伪操作，是被系统正用的名字，所以我在编译的时候会出错

2). PUTNAME,PUTPASSWORD,ITEMNAME 这三个是我定义的变量名，他们报错是因为我在前面设定字符串的时候在’…’里面也用了’，导致这些变量名没有正常定义，所以出错。

3). BTYE 的报错是因为我拼写 BYTE 错误。

4). 排除错误后，程序能正常编译通过。

2. 连接过程没有发生异常。

3. 程序功能测试

1） 姓名输入测试

a) 我们先输入正确的用户名，正常跳转到输入密码环节

```
C:\>shop.exe
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
WL DING
PLEASE INPUT THE PASSWORD
```

图 3-21 输入正确用户名

b) 边缘测试，输入空串，出现报错，并重新回到输入名字环节

```
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)

YOUR USER NAME OR PASSWORD IS WRONG!
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
```

图 3-22 输入空串报错

c) 边缘测试，输入纯数字，发现报错，并重新回到输入名字环节

```
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
1111111
YOUR USER NAME OR PASSWORD IS WRONG!
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
```

图 3-23 输入纯数字报错

d) 边缘测试，输入长度大于标准长度，报错，并重新回到输入名字环节

```
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
OVER FLOW
YOUR USER NAME OR PASSWORD IS WRONG!
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
```

图 3-24 输入长度大于标准名字长度报错

2) 密码输入测试

a) 我们先输入正确的密码，输出正确的提示信息并跳转到查询环节

```
C:\>shop.exe
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
WL DING
PLEASE INPUT THE PASSWORD
111111
IDENTIFICATION GOT!
PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)
```

图 3-25 输入正确密码截图

b) 边缘测试，输入空串会报错并跳转到输入姓名环节

```
C:\>shop.exe
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
WL DING
PLEASE INPUT THE PASSWORD

YOUR USER NAME OR PASSWORD IS WRONG!
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
```

图 3-26 输入空串报错

c) 边缘测试，输入纯字母，发现报错并跳转到输入姓名环节

```
C:\>shop.exe
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
WL DING
PLEASE INPUT THE PASSWORD
str pass
YOUR USER NAME OR PASSWORD IS WRONG!
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
```

图 3-27 输入纯字符串操作报错

d) 边缘测试，输入长度超标准密码长度，会报错并跳转到重新输入名字

```
C:\>shop.exe
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
WL DING
PLEASE INPUT THE PASSWORD
12345678
YOUR USER NAME OR PASSWORD IS WRONG!
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
```

图 3-28 输入长度超过标准密码长度报错

3) 货物名称测试

a) 登陆的情况下，输入正确的货物名称，会返回该货物名称

```
C:\>shop.exe
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
WL DING
PLEASE INPUT THE PASSWORD
111111
IDENTIFICATION GOT!
PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)
PEN
PEN
PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)
```

图 3-29 输入正常货物名称截图

b) 输入错误会有错误提示

```
PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)
PENCIL
DO NOT FIND THE ITEM
PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)
```

图 3-29 输入不存在的货物报错

c) 边缘测试，输入空串，会有错误信息提示

```
PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)

DO NOT FIND THE ITEM
PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)
```

图 3-30 在输入货物信息的位置输入空串报错

d) 边缘测试，在输入货物的地方输入纯数字串，会正常报错

```
PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)
11111111
DO NOT FIND THE ITEM
PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)
```

图 3-31 在输入货物的地方输入纯数字串报错

e）边缘测试，在输入货物的地方输入特殊字符，会正常报错

图 3-32 输入特殊字符报错

4） 推荐度测试

a）在未登录的情况下，输入商品名称会出现推荐度，下面展示题目中已经给出的 PEN 和 BOOK 的推荐度截图：

图 3-33 PEN 的推荐度的截图

图 3-34 BOOK 的推荐度的截图

说明：

1. PEN：按照我给出的算法，PEN 推荐度的计算为 35*128/56=80,25*64/70=22(除法只保留整数部分)，80+22=102，所以为 A

2. BOOK:BOOK 推荐度的算法为 30*9/10=27,12*128/27=56,5*64/25=12（除法只保留整数部分)，56+12=68，所以为 B

b） 边缘测试，将进货价和已售数量设定为 0（如数据段我自己添加的样例 GA2，货物名为‘TEST2’)，这时候按照公式手算，推荐度为 0，程序正常输出 F，截图如下

图 3-35 将一些数字置零的边缘测试

c） 如实验步骤中所说，参与计算的数字理论最大值为 512，再大了会溢出，为此设计样例，如数据段中的 GA3,商品名为‘TEST1’，将进货商品数量设置成了 500，接近理论极限。按照手算结果，推荐度应该为 A，截图如下：

图 3-36 数字接近理论界限测试

4. 问题解决：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？

方案：我们将程序中的提示输入姓名字符串末尾的“$”去掉，看程序运行效果，以下是去掉了输入姓名提示之后的代码截图：

```
PUTNAME DB 'PLEASE INPUT THE NAME(INPUT ENTER TO LOOKUP, INPUT Q TO QUIT)'
NOTITEM DB 'DO NOT FIND THE ITEM$'
PUTPASSWORD DB 'PLEASE INPUT THE PASSWORD$'
ITEMNAME DB 'PLEASE INPUT THE ITEM NAME(INPUT ENTER TO INPUT NAME)$'
LOGFAIL DB 'YOUR USER NAME OR PASSWORD IS WRONG!$'
LOGINREMIND DB 'IDENTIFICATION GOT!$'
```

图 3-37 去掉了输入姓名提示之后的代码

进行正常编译链接，运行结果如下图：

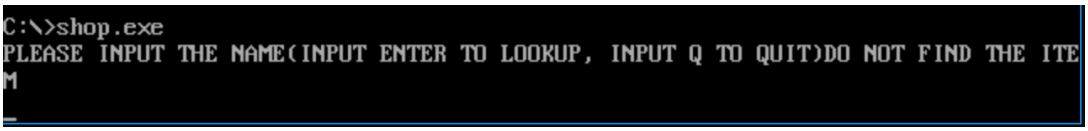

图 3-38 去掉了提示输入的‘$’之后运行截图

可以看出，程序一直输出到了数据段下一个‘$’所在的位置，即其将 NOTITEM 的字符串也输出了，但是对于其他实际功能来说，并没有太大影响，输入姓名能正常跳转到输入密码，如下截图：

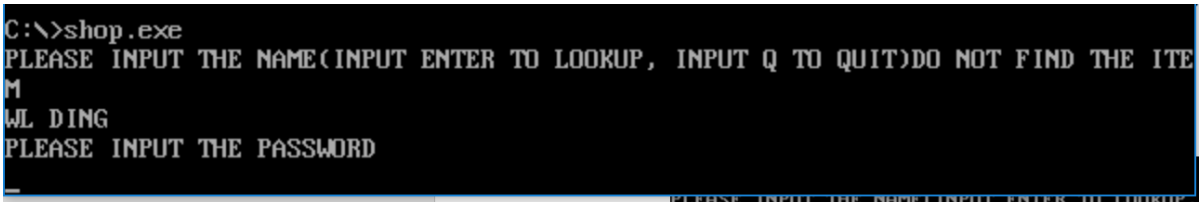

图 3-39 正常跳转到输入密码模块

通过实验，对该问题回答总结如下：

1. 去掉一个‘$’之后会顺着数据段往后面输出，直到遇到下一个‘$’。
2. 除了输出显示有问题之外，对于程序其他功能没有太大影响，能正常进行后续功能。

5. 问题解决：如果在 9 号功能调用前，未对 DS 赋值，结果会如何？更改程序验证。

注释掉对于 DS 赋值的语句之后正常编译连接，运行截图如下：

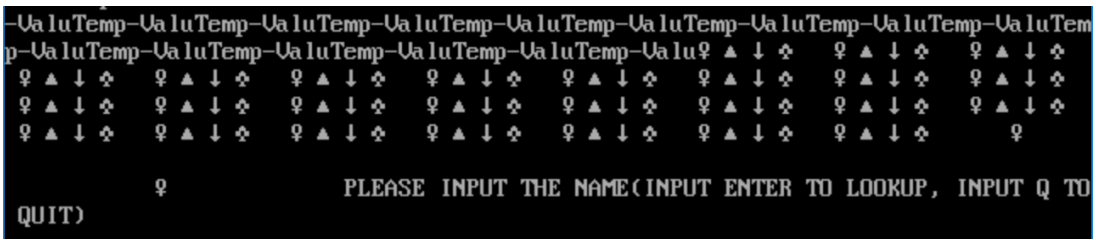

图 3-40 未对 DS 赋值运行截图

会发现程序在正常输出提示信息之前输出了一大段乱码，打开 td 单步调试，了解其中原因。下面两张图，第一张是没对 DS 赋值的调试界面，第二张是对 DS 赋值过后的调试界面。可以发现，没有赋值的时候，ds 是一个随机值，它与 es 相等，所以 ds 没有指向数据段的位置，所以输出的时候会出现一些乱码。通过第二个图可以看出 ds 赋值过后得到真实值，其与第一张图的 ds 不一样。

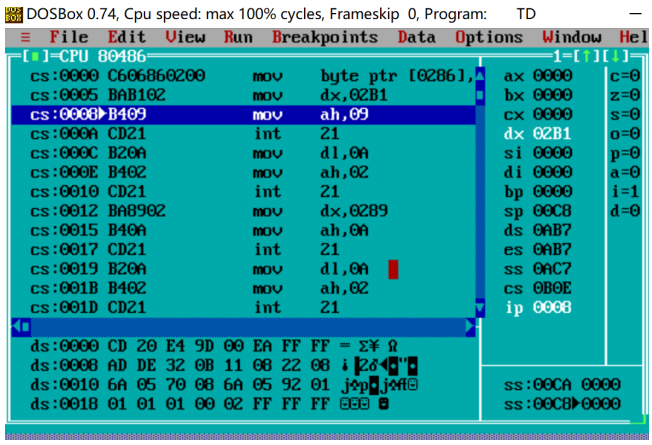

图 3-41 没有对 DS 赋值的调试结果

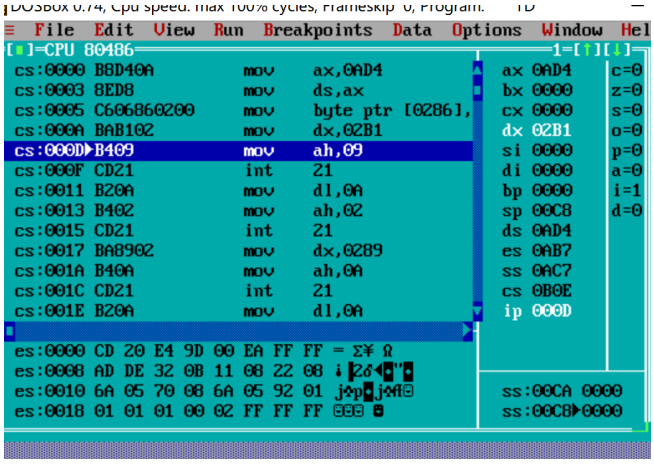

图 3-42 对 DS 赋值之后的调试结果

总结：若代码段未对 DS 赋值，则导致 DS 的值不正确，指向的数据段位置不正确，输出时会出现乱码。

6. 问题解决：注意观察转移指令机器码的编码方法，观察对应标号的偏移地址与该编码之间的关系。

打开 td 调试，我们以姓名输入的循环为例，解释转移指令机器码的编码方法与标号的偏移地址与改编码之间的关系。

下面这段代码有三个转移指令，分别是 JE PASSWORD、JE LOPA 和 JNE INPUTFAIL。

下面分析以 JE LOPA 为例。

```
LOPA:
        ADD DI,1
        ADD SI,1
        ADD CX,1
        CMP CX,8H
        JE PASSWORD
        MOV BL,BYTE PTR [SI] ;在BL中存入真实的姓名
        CMP BL,BYTE PTR [DI] ;将输入值有与真实姓名作比较
        JE LOPA ;如果为真，继续执行下一个比较
        JNE INPUTFAIL ;如不为真，则提示失败
```

图 3-43 一段有转移指令的代码

我们找到了对应的在 td 中的位置如下图：

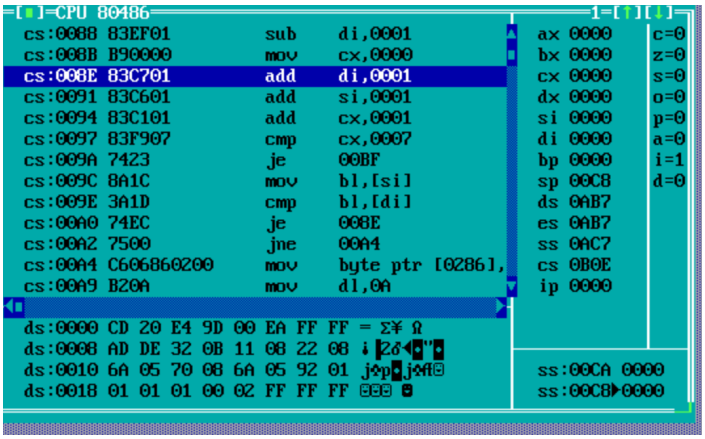

图 3-44 上面代码对应的 td 截图

可以看出 008E 对应 LOPA 标号的偏移首址(即语句 ADD DI,1 的地址),转移指令直接为 je 008E,即从这里直接跳到偏移地址 008E。所以转移指令的特点是直接跳转到所对应的偏移地址。

观察其机器编码，je 008E 的机器编码为 74EC，je 00BF 的机器编码为 7423。所以我们可以总结而出 je 的短转移机器码的特点为 74xx。通过浏览整个程序的机器码，还可以总结出，jne 的端转移机器码特点为 75xx，jmp 的长转移机器码的特点为 EBxx，je 的长转移机器码为 0F84xxxxxxxx,jne 的长转移机器码为 0F85xxxxxxxx。

4 总结与体会

4.1 任务 1~3 的总结与体会

前三个任务让我们熟悉了 TD 的调试功能，也让我们简单地了解了汇编程序汇编，连接以及调试的过程。

任务一让我们对机器码加减法运算有了更加直观深刻的认识，机器中的加减法运算均采用补码运算，运算的时候不再管符号而把整个数作为无符号数处理，符号位也一起参与运算，在这个基础上，我们能判断 SF,CF,ZF,OF 的值。（具体内容在第三节相应部分有总结）

任务二让我们单步调试程序，观察寄存器的变化，然后观察数据段前 40 个数值。一开始我观察的结果不正确，通过询问同学才知道我没有在 goto 中设置 DS:0 去查看，也就是说要从开始查看数据段，一定要设置 DS:0。

任务三改写源程序为变址寻址以及把 16 位寄存器改为 32 位重新编写任务二的程序,通过观察,发现了 16 位程序的编码与 32 位的联系以及 32 位码的独特特点（在第三节相应部分有所总结）。并

且发现改用 32 位寄存器之后，数据段前 40 位依然不变。最后我们从不同字节位置开始反汇编，了解与 IP/EIP 指明起始位置的重要性。我们发现，如果程序的 IP 一开始指的位置不是初始位置，有可能导致程序整个运行错误，就如第 3 节中的例子，从 000AH 开始的 IP，会导致 DS 的值不正确从而影响整个程序。所以 IP 指明起始位置的正确性是很重要的。

4.2 任务 4 的总结与体会

这次是我们第一次用汇编编写一个完整的大程序，让我们对用汇编语言设计程序有了一个清晰完整的认识。我们在这次实验中熟练掌握了字符和字符串的输入输出，转移指令的运用，循环结构和分支结构的设计和运用，运用汇编语言进行简单计算的技巧与注意事项。

在编写程序的过程中遇到了许多错误，最后都通过查阅资料一一解决了。例如我在运用 2 号调用输出时有时候会误将 ASCII 码写入 DX 而不是 DL，有时候忘记加 MOV AH,2，导致了一些输出错误；有时候在输出的时候，直接调用了地址，而忘记了在后面加上‘$’符号再输出，也导致了输出错误，对于这些小问题以后要多加留意。在调试过程中我发现我的推荐度无论怎么样算出来都是 F，后来在同学的帮助下我了解到，在做除法之前，要记得将 DX 清 0，否则会出现错误，按照这个方法调试，最后解决了问题。

这个实验设计有一个小缺憾就是在计算推荐度的时候，其能运算的数最大不能超过 500，否则可能会存在溢出的问题，计算推荐度的时候，由于除法运算只取了整数部分，会对于推荐度计算存在一定的误差，在这一点上还有待提高和改进。

参考文献

[1] 王元珍 曹忠升 韩宗芬 编著.80X86 汇编语言程序设计.武汉:华中科技大学出版社,2005 年 4 月.100 页-187 页

[2] URL: https://blog.csdn.net/pengwill97/article/details/79249631

[3] URL: https://www.cnblogs.com/BoBoRing/p/10240891.html