

**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： 2019-3-20，14：00-17：30 实验地点： 南一楼804室30号实验台**

**指导教师： 曹忠升**

**专业班级：计算机科学与技术ACM1701班**

**学 号： U201714780 姓 名： 刘晨彦**

**同组学生： 无 报告日期： 2019年3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：2019.03.20

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](#_Toc8209224)

[2 实验内容 1](#_Toc8209225)

[3 实验过程 3](#_Toc8209226)

[3.1 任务1 3](#_Toc8209227)

[3.1.1 实验步骤 3](#_Toc8209228)

[3.1.2 实验记录与分析 3](#_Toc8209229)

[3.2 任务2 5](#_Toc8209230)

[3.2.1 实验步骤 5](#_Toc8209231)

[3.2.2 实验记录与分析 5](#_Toc8209232)

[3.3 任务3 6](#_Toc8209233)

[3.3.1 实验步骤 6](#_Toc8209234)

[3.3.2 源代码 7](#_Toc8209235)

[3.3.3 实验记录与分析 7](#_Toc8209236)

[3.4 任务4 9](#_Toc8209237)

[3.4.1 设计思想及存储单元分配 9](#_Toc8209238)

[3.4.2 流程图 11](#_Toc8209239)

[3.4.3 源程序 15](#_Toc8209240)

[3.4.4 实验步骤 19](#_Toc8209241)

[3.4.5 实验记录与分析 19](#_Toc8209242)

[4 总结与体会 25](#_Toc8209243)

[参考文献 27](#_Toc8209244)

# 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。

# 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。

**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。

# 实验过程

## 任务1

### 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输入第一个运算式对应的两个8位数值对应的指令语句MOV AH,01001101B；MOV AL,-01110010B；ADD AH,AL；观察代码区显示的内容与自己输入字符之间的关系；然后确定CS:IP指向的是自己输入的第一条指令的位置，单步执行三次，观察寄存器内容的变化，记录标志寄存器的结果。

预计ADD执行之后（AH）=0DBH SF=1、OF=0、CF=0、ZF=0

重复上述过程，将剩下几个表达式计算完毕，比较结果。

3. 输入MOV AH,+0110011B；MOV AL,+1011010B；ADD AH,AL；观察标志位特点。

预计ADD执行后：(AH)=8DH;SF=1,OF=1,CF=0,ZF=0

输入MOV AH,-0101001B；MOV AL,-1011101B；ADD AH,AL；观察标志位特点。

预计ADD执行后：(AH)=7AH; SF=0,OF=1,CF=1,ZF=0

输入MOV AH,+1100101B；MOV AL,-1011101B；ADD AH,AL；观察标志位特点。

预计ADD执行后：(AH)=08H; SF=0,OF=0,CF=1,ZF=0

实验预计如表1.1所示：

表1.1 任务一预测实验结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | X1 | X2 | AH | SF | OF | CF | ZF |
| （1） | +0110011B | +1011010B | 8DH | 1 | 1 | 0 | 0 |
| （2） | -0101001B | -1011101B | 7AH | 0 | 1 | 1 | 0 |
| （3） | +1100101B | -1011101B | 08H | 0 | 0 | 1 | 0 |

### 实验记录与分析

1. 实验环境条件：i7-7700HQ 2.80GHz，8G内存；WINDOWS 10下DOSBox0.72；TD.EXE 5.0。

2. 输入第一题指令MOV AH,+0110011B；MOV AL,+1011010B；ADD AH,AL后，进行单步调试，执行三条指令后的结果如图1.1所示。

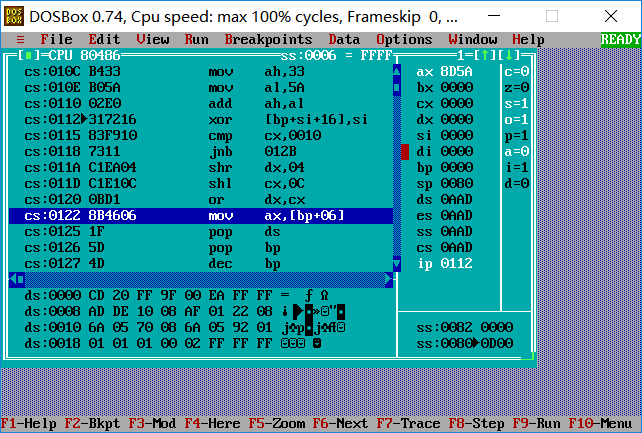


图1.1任务一第（1）小题测试截图

可以看出，计算结果AH = 8DH,与标志位的状态(SF=1,OF=1,CF=0,ZF=0)与事前预期的是一致的。

3. 输入第二题指令MOV AH,-0101001B；MOV AL,-1011101B；ADD AH,AL后，进行单步调试，执行三条指令后的结果如图1.2所示。

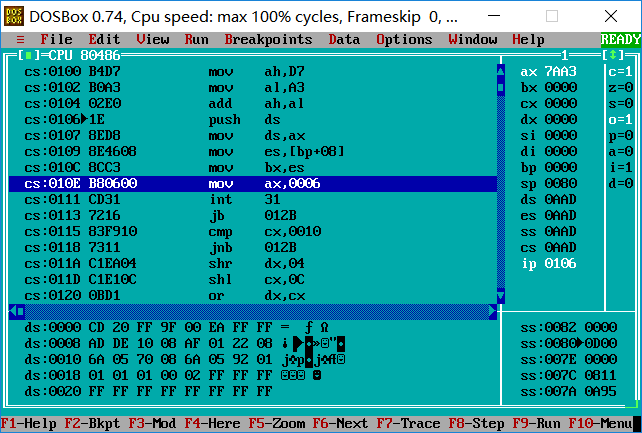


图1.2 任务一第（2）小题测试截图

可以看出，计算结果AH = 7AH,与标志位的状态(SF=0,OF=1,CF=1,ZF=0)与事前预期的是一致的。

4. 输入第三题指令MOV AH,+1100101B；MOV AL,-1011101B；ADD AH,AL后，进行单步调试，执行三条指令后的结果如图1.3所示。

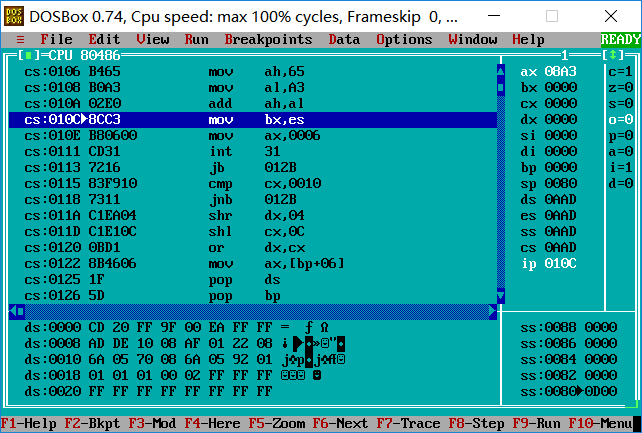


图1.3 任务一第（3）小题测试截图

可以看出，计算结果AH = 08H,与标志位的状态(SF=0,OF=0,CF=1,ZF=0)与事前预期的是一致的。

## 任务2

### 实验步骤

1. 准备上机实验环境。

2. 建立汇编源程序，将题目代码写入文件；汇编源文件，生成目标文件；连接目标文件，建立可执行文件。

3.在td窗口点击FILE菜单中的Open选项,打开文件。找到“MOV CX，10”和“INT 21H”，点击选中并按F2添加断点。

4.按F9运行程序至断点处暂停，记录BX,BP,SI,DI的值。

预计执行到“MOV CX，10”之前，BX=0014H,BP=001EH,SI=0000H,DI=000AH。

预计执行到“INT 21H”之前，BX=001EH,BP=0028H,SI=000AH,DI=0014H。

相关预计如表1.2所示。

表1.2 任务二预测实验结果1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 断点 | BX | BP | SI | DI |
| MOV CX，10 | 0014 | 001E | 0000 | 000A |
| INT 21H | 001E | 0028 | 000A | 0014 |

5.完成断点观察后重新进入，进行单步调试。在代码段右键选择goto，输入DS:0000,转调至DS区。注意查看数据段的变化，预计结束前40个字节的数据段数值如表1.3所示。

表1.3 任务二预测实验结果2

|  |  |
| --- | --- |
| 有效地址 | 前四十个字节的内容 |
| DS0000 | 00 01 02 03 04 05 06 07 |
| DS0008 | 08 09 00 01 02 03 04 05 |
| DS0010 | 06 07 08 09 01 02 03 04 |
| DS0018 | 05 06 07 08 09 0A 04 05 |
| DS0020 | 06 07 08 09 0A 0B 0C 0D |

### 实验记录与分析

1. 实验环境条件：i7-7700HQ 2.80GHz，8G内存；WINDOWS 10下DOSBox0.72；TD.EXE 5.0。

2. 在DOSBOX中打开TD.EXE，通过FILE-OPEN打开23.exe，在“MOV CX，10”和“INT 21H”处添加断点，进行单步调试，观察在“MOV CX，10”之前BX,BP,SI,DI的值，如图1.4所示。结果与预期相符合。

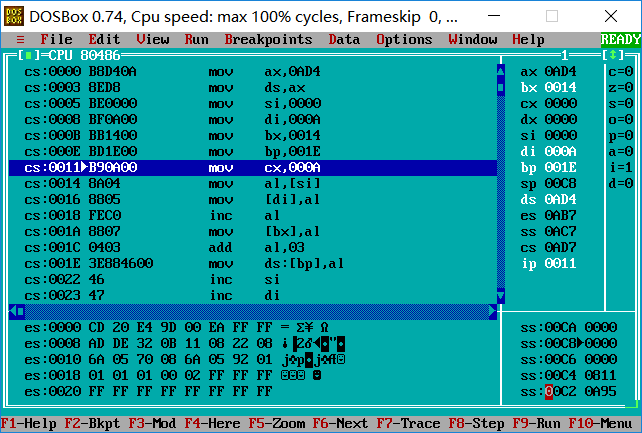


图1.4 “MOV CX，10”断点处的实验测试截图

3. 进行单步调试，观察在“INT 21H”之前BX,BP,SI,DI的值，如图1.5所示。结果与预期相符合。

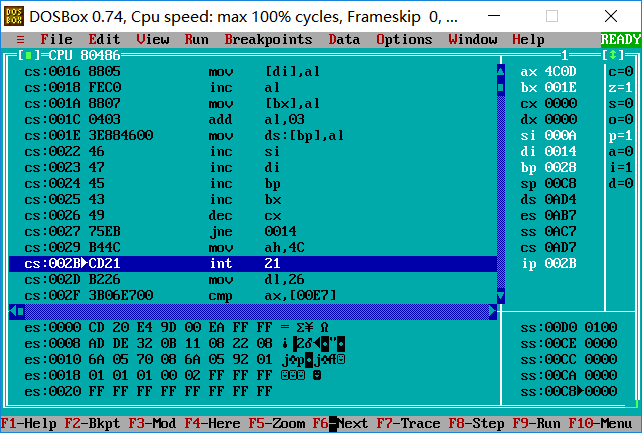


图1.5 “INT 21H”断点处的实验测试截图

4. 在代码段右键选择goto，输入DS:0000,转调至DS区。获得结果如图1.6所示，与预期一致。

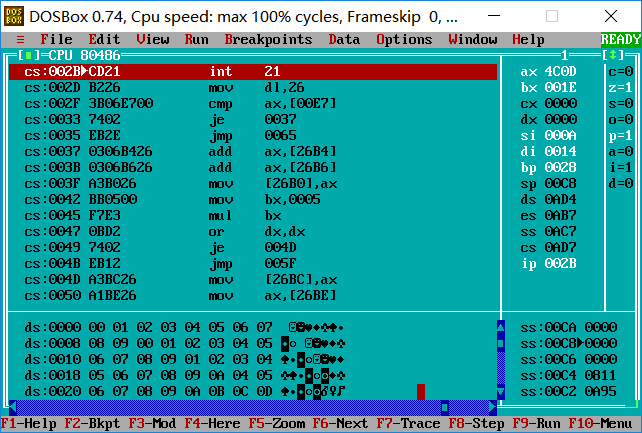


图1.6 结束前40个字节的数据段数值

## 任务3

### 实验步骤

1. 准备上机实验环境。

2. 修改任务2中的代码，使其满足任务3的要求

3. 对修改后的代码建立汇编源程序；汇编源文件，生成目标文件；连接目标文件，建立可执行文件。

4. 在DOSBOX中打开TD.exe，点击FILE-OPEN，选择exe文件，进行单步调试，注意观察数据段和标志符的变化，预计程序执行到退出之前数据段开始40个字节的内容，预计内容如表1.4所示。

表1.4 任务三预测实验结果

|  |  |
| --- | --- |
| 有效地址 | 前四十个字节的内容 |
| DS0000 | 00 01 02 03 04 05 06 07 |
| DS0008 | 08 09 00 01 02 03 04 05 |
| DS0010 | 06 07 08 09 01 02 03 04 |
| DS0018 | 05 06 07 08 09 0A 04 05 |
| DS0020 | 06 07 08 09 0A 0B 0C 0D |

5. 在TD代码窗口中，观察并记录指令代码在内存中的存储形式，将并与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。

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

### 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 EDI, 0

MOV EBX, 0

MOV EBP, 0

MOV CX, 10

LOPA: MOV AL, [ESI]+BUF1

MOV [EDI]+BUF2, AL

INC AL

MOV [EBX]+BUF3,AL

ADD AL, 3

MOV DS:[EBP]+BUF4, AL

INC ESI

INC EDI

INC EBP

INC EBX

DEC CX

JNZ LOPA

MOV AH, 4CH

INT 21H

CODE ENDS

END START

### 实验记录与分析

1. 实验环境条件：i7-7700HQ 2.80GHz，8G内存；WINDOWS 10下DOSBox0.72；TD.EXE 5.0。

2. 打开DOSBOX，使用MASM对源程序23m.asm进行编译，再使用LINK对23m.obj进行连接，生成23m.exe文件。

3. 打开TD.EXE，点击FILE-OPEN，选择23m.exe。

4. 对文件结束位置设置断点，快速查看程序退出前数据段前40个字节的内容，如图1-7所示，结果与预测情况一致。

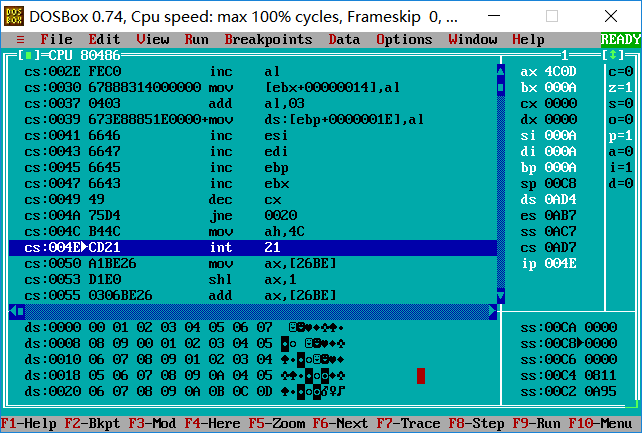


图1-7 任务三结束前40个字节的数据段数值

5.将反汇编代码与自己所写代码进行对比，也与任务二做对比。如图1.8与图1.9，图1.10所示。

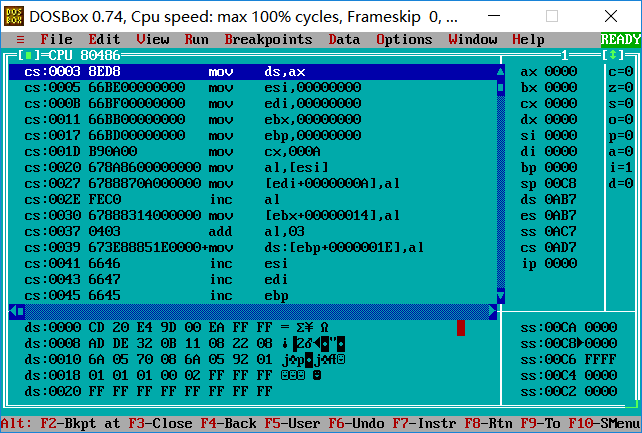


图1.8 任务三部分反汇编代码

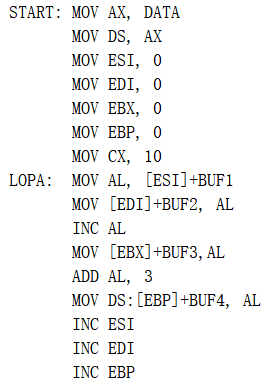


图1.9 部分源代码

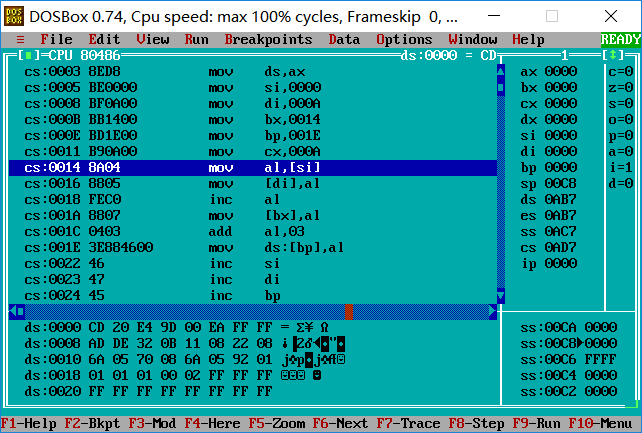


图1.10 任务二部分反汇编代码

对比发现如下：

（1）指令名称在反汇编代码和源代码中是相同的，如”mov”,”inc”等。

（2）在反汇编代码中，变址寻址部分直接将位移量以偏移地址形式显示。如源代码中的：” MOV [EDI]+BUF2, AL”在反汇编代码中为：”mov [edi+0000000A],al”。

（3）在反汇编代码中，START和LOPA都表现为了偏移地址，如源代码中的：” JNZ LOPA”在反汇编代码中为：”jne 0020”。

（4）任务二和任务三的寻址方式不同，使得反汇编代码中的表示也不同，人物二中使用了寄存器间接寻址，在反汇编代码中保持不变，如：”MOV AL,[SI]”在反汇编代码中仍然为：”mov al,[si]”。

（5）在反汇编代码中，常数均转化成十六进制的形式，如源代码中的:”MOV CX,10”在反汇编代码中显示为:”mov cx 000A”

6. 在代码段执行GOTO语句，输入cs:0017，并设置为当前操作语句，此时寄存器ip地址变为0017，如图1.11所示。

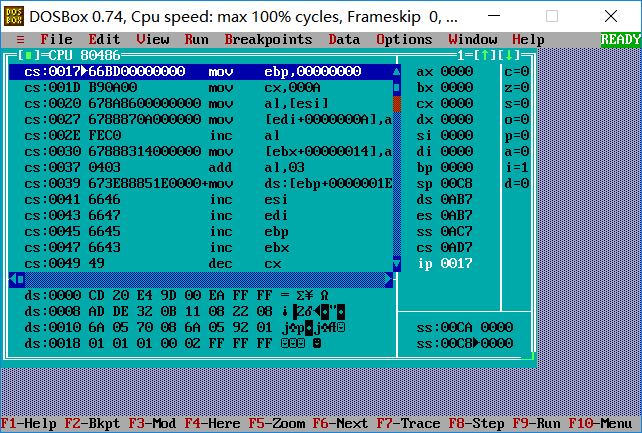


图1.11 执行GOTO之后的状态截图

## 任务4

### 设计思想及存储单元分配

**a)设计思想：**

程序功能一主要是用了9号和10功能调用，完成输出和输入，其中包含了简单的判断语句：当只输入了回车时，程序进入功能三，当只输入了q时，程序退出。

程序功能二使用了两个单层循环结构进行字符串的匹配，若不匹配则输出提示返回功能一，若匹配则进入功能三。

功能三除了使用了9号和10号调用以外，最主要使用了双层循环结构，比较输入的商品字符串和存储在内存中的字符串是否相同。若不同，则重新要求输入商品名。若相同，根据用户登录与否创建分支结构：若登录，则重新输出商品名并回到功能一，未登录则计算商品推荐度并进入功能四。

功能四主要使用了多个分支结构，根据商品的推荐度给出推荐等级。

**b)程序的主要算法：**

串的比较：在比较用户名和密码时，首先判断在输入缓冲区的指定位置是否为0DH，确保输入字符数和正确的字符串长度相等，否则重新输入。然后进入循环，从字符串首开始比较，不同则退出循环要求重新输入，直到相同且计数器为0退出。在比较商品名称时，则是从字符串首开始比较，比较至输入商品字符串结束且均相同时，判断商品字符串的下一位是否为0DH，若不是，则说明输入的字符串是当前比较字符串的子串，依然不相同。

推荐度的计算：为了避免进价除以售价商为0的情况，首先将被除数乘以128以获得正确结果。

显示：为了正确显示输出和输入，在每一次输入和输出之后都确保程序会输出回车换行，否则容易出现输入内容被下一次输出所覆盖的情况。

**c)存储单元分配：**

BNAME：用户名的首地址。

BPASS：密码的首地址。

IN\_NAME:存放输入用户名的首地址。

IN\_PWD：存放输入密码的首地址。

IN\_ITEM:：存放输入商品名称的首地址。

AUTH：字节变量，存放登录状态。

SNAME：字节变量，存放网点名称。

GA1、GA2、GAN：字节变量，存放商品信息。

BUF1、BUF2、BUF3、BUF4:字节变量，存放输出提示语句。

CRLF：字节变量，存放回车换行。

**d)寄存器分配：**

AX：主要用于算术运算

CX，BX：主要用于计数器

SI，DI：临时寄存器，存储计算结果。

### 流程图

代码总体流程图如1.12所示：

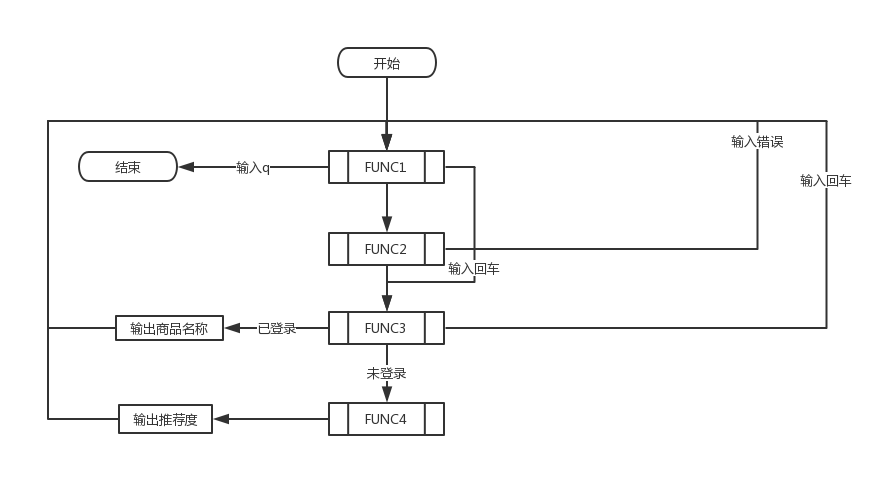


图1.12 程序总体流程图

程序各功能流程图如图1.13，1.14，1.15(a), (b), (c)，1.16所示：

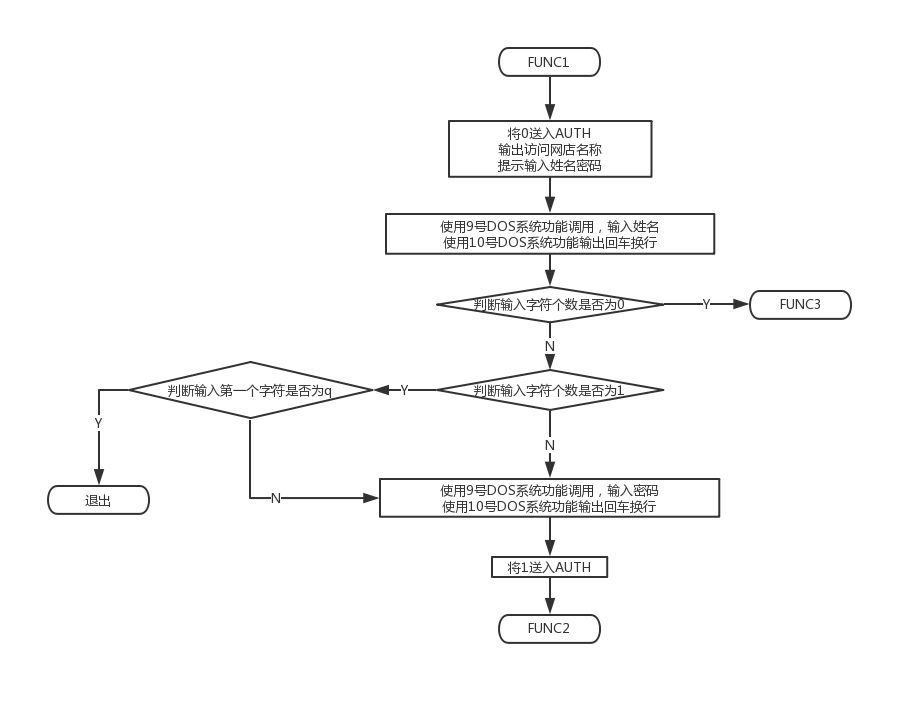


图1.13 程序功能1实现流程图

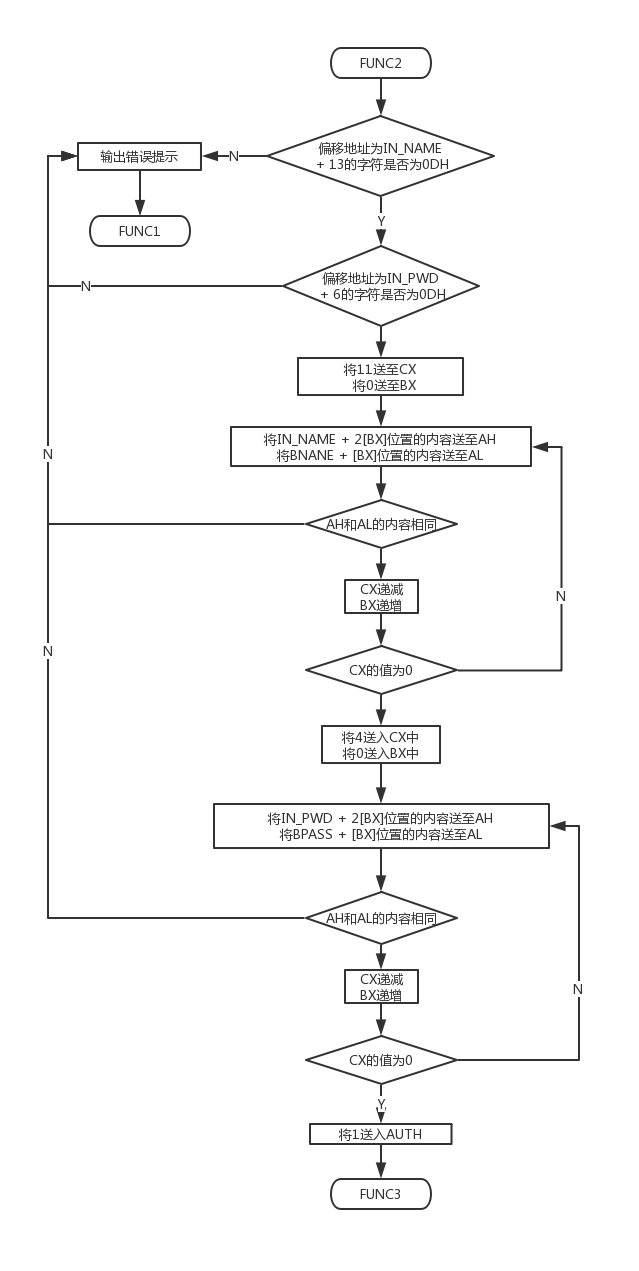


图1.14 程序功能2实现流程图

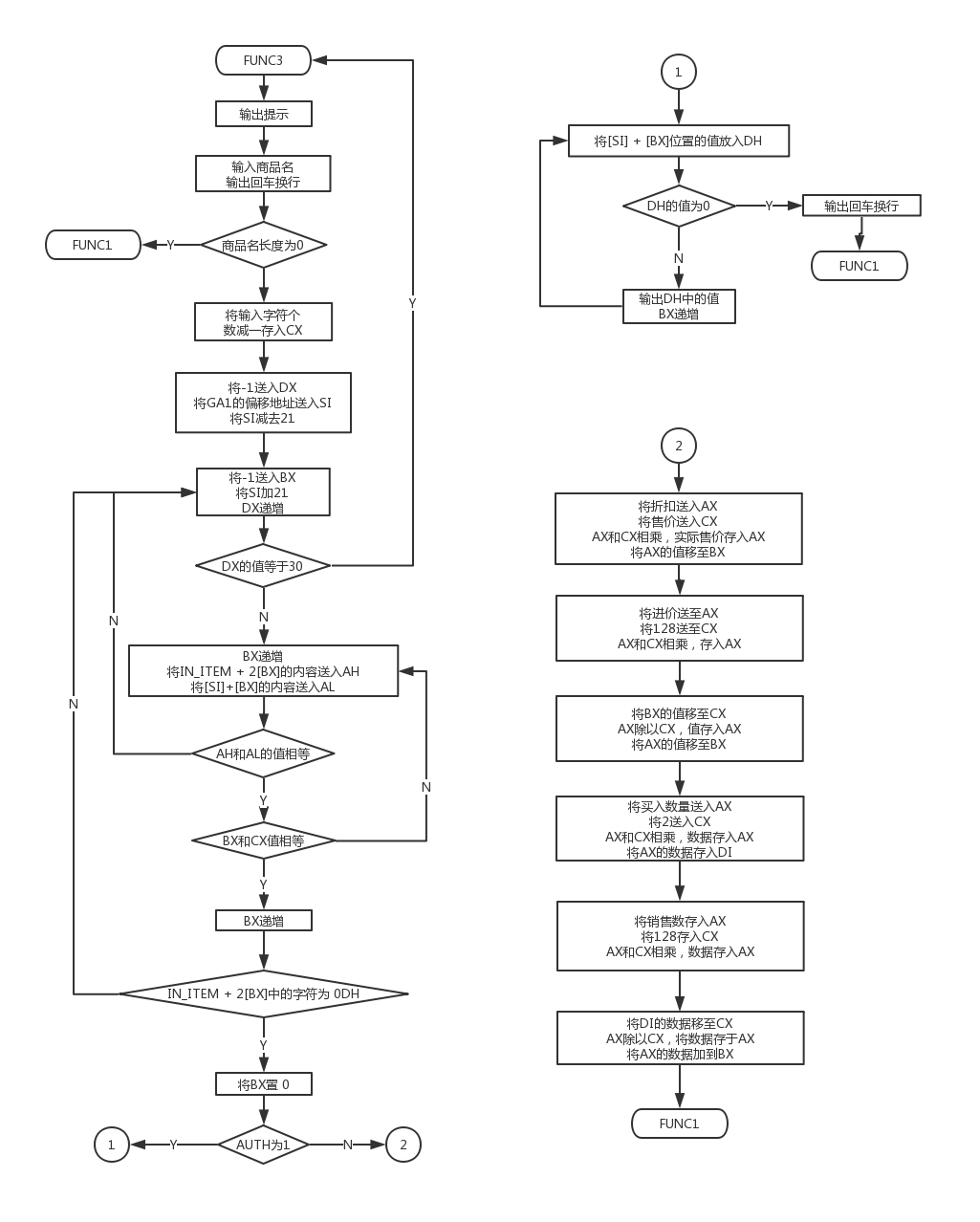


图1.15(a) 程序功能3实现流程图1

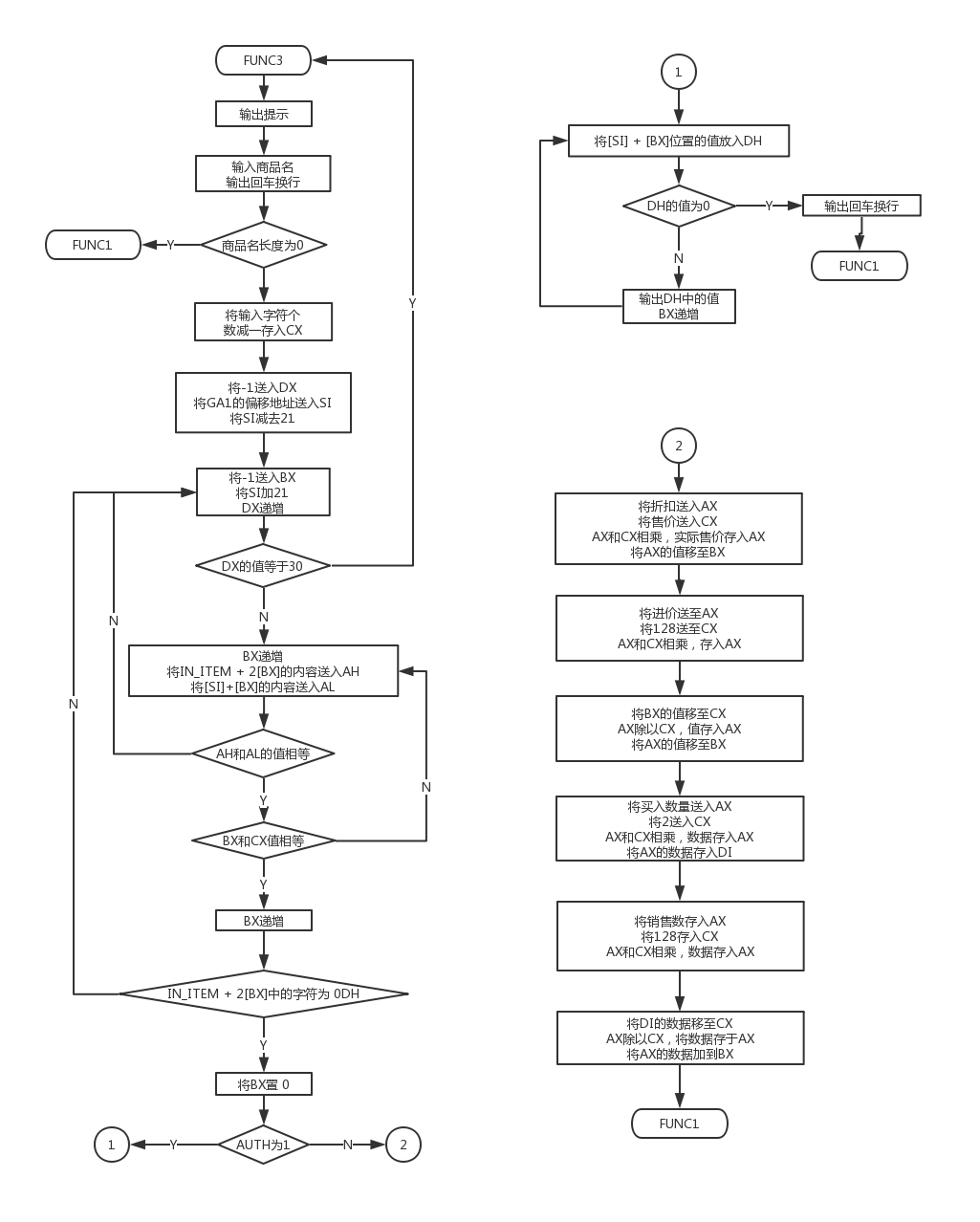


图1.15(b) 程序功能3实现流程图2

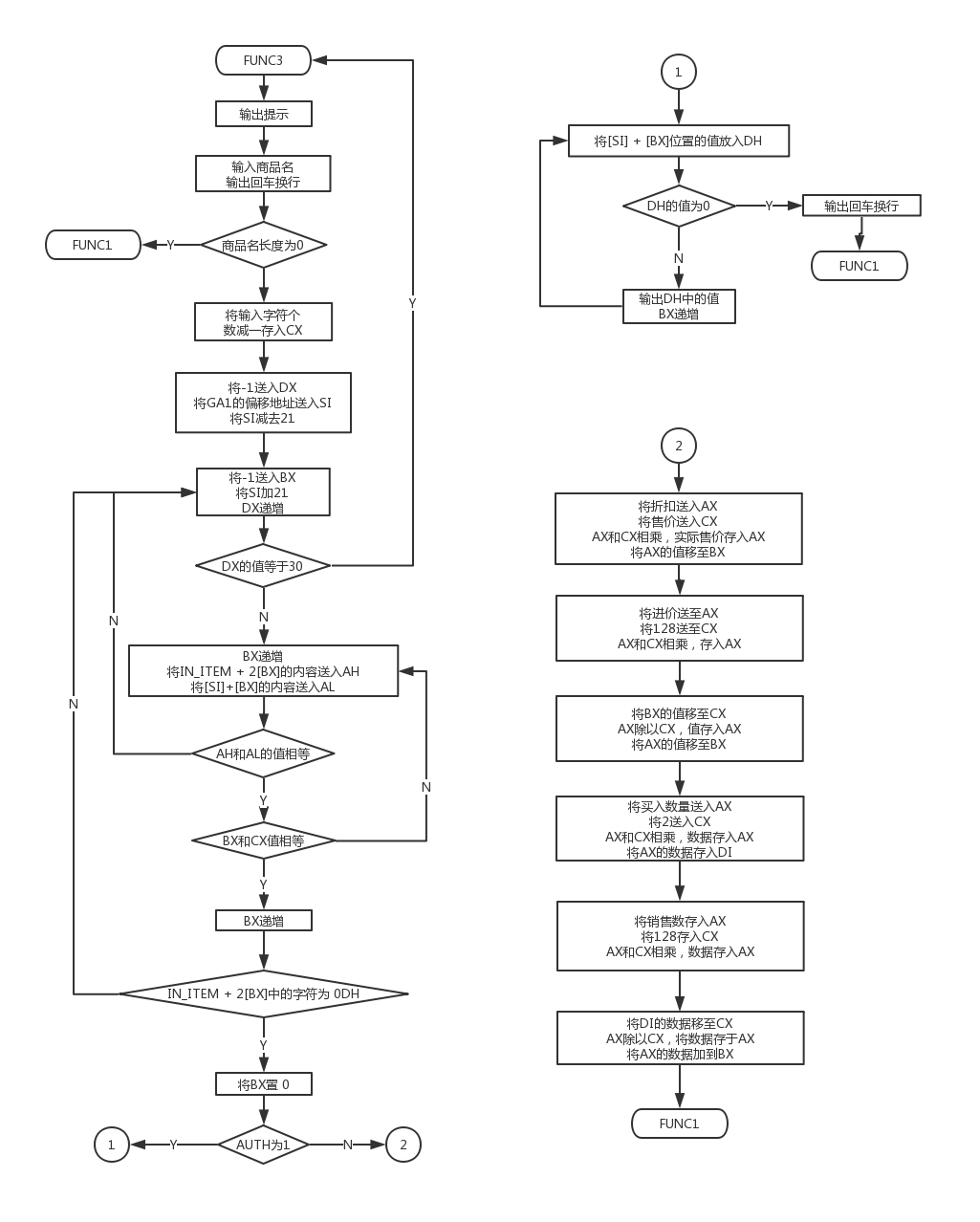


图1.15(c) 程序功能3实现流程图3

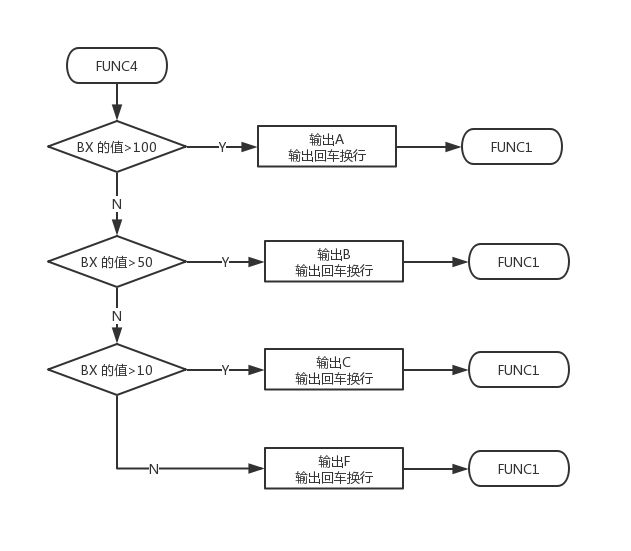


图1.16 程序功能4实现流程图

### 源程序

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

BNAME DB 'liu chenyan', 0 ;owner's name

BPASS DB 'test', 0, 0 ;correct password

IN\_NAME DB 15 ;palce storing name

DB 0

DB 15 DUP(0)

IN\_PWD DB 10 ;place storing password

DB 0

DB 10 DUP(0)

IN\_ITEM DB 20

DB 0

DB 20 DUP(0)

N EQU 30

AUTH DB 0

SNAME DB 'ONLINE SHOP'

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

DW 35, 56, 70, 25, ?

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

DW 12, 30, 25, 5, ?

GAN DB N-2 DUP('Temp-Value', 8, 15, 0, 20, 0, 30, 0, 2, 0, ?, ?)

BUF1 DB 'WELCOME! YOU ARE VISITING ONLINE SHOP', 0AH, 0DH, '$'

BUF2 DB 'PLEASE ENTER YOUR NAME AND PASSWORD:', 0AH, 0DH, '$'

BUF3 DB 'FAIL TO LOG IN!', 0AH, 0DH, '$'

BUF4 DB 'PLEASE ENTER THEN ITEM YOU WOULD LIKE TO INQURE:', 0AH, 0DH, '$'

CRLF DB 0DH, 0AH, '$'

DATA ENDS

CODE SEGMENT USE16

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

START: MOV AX, DATA

MOV DS, AX

FUNC1: MOV AUTH, 0

LEA DX, BUF1 ;print: YOU ARE VISITING ONLINE SHOP

MOV AH, 9

INT 21H

LEA DX, BUF2 ;print: PLEASE ENTER YOUR NAME AND PASSWORD

MOV AH, 9

INT 21H

LEA DX, IN\_NAME ;scanf: name

MOV AH, 10

INT 21H

LEA DX, CRLF ;回车换行

MOV AH, 9

INT 21H

CMP IN\_NAME + 1, 0 ;if entered nothing, goto FUNC3

JE FUNC3

CMP IN\_NAME + 1, 1 ;if entered only one char, goto CODION1 to see more

JNE CODON1

CMP IN\_NAME+2, 'q' ;if the name is 'q', exit

JE EXT

CODON1: LEA DX, IN\_PWD ;scanf: password

MOV AH,10

INT 21H

LEA DX, CRLF ;回车换行

MOV AH, 9

INT 21H

MOV AUTH, 1 ;else let AUTH be 1 and goto FUNC2

JMP FUNC2

FUNC2: CMP IN\_NAME + 13, 0DH

JNE WARN

CMP IN\_PWD + 6, 0DH

JNE WARN

MOV CX, 11

MOV BX, 0

LOP1: MOV AH, IN\_NAME + 2[BX]

MOV AL, BNAME + [BX]

CMP AH, AL

JNE WARN

DEC CX

INC BX

CMP CX, 0

JNE LOP1

MOV CX, 4

MOV BX, 0

LOP2: MOV AH, IN\_PWD + 2[BX]

MOV AL, BPASS + [BX]

CMP AH, AL

JNZ WARN

DEC CX

INC BX

CMP CX, 0

JNE LOP2

MOV AUTH, 1

JMP FUNC3

WARN: LEA DX, BUF3 ;print: FAIL TO LOG IN

MOV AH, 9

INT 21H

JMP FUNC1

FUNC3: LEA DX, BUF4 ;print: PLEASE ENTER THEN ITEM YOU WOULD LIKE TO INQURE

MOV AH, 9

INT 21H

LEA DX, IN\_ITEM ;scan the item user want to inquire

MOV AH, 10 ;scan the item user want to inquire

INT 21H ;scan the item user want to inquire

LEA DX, CRLF ;回车换行

MOV AH, 9 ;回车换行

INT 21H ;回车换行

CMP IN\_ITEM + 1, 0 ;if user entered nothing, goto FUNC1

JE FUNC1 ;if user entered nothing, goto FUNC1

MOV CL, IN\_ITEM + 1

SUB CL, 1

MOV CH, 0

MOV DX, -1 ;DX counts the term of item

MOV SI, OFFSET GA1

SUB SI, 21

LOP3: MOV BX, -1 ;BX counts the term of literal

ADD SI, 21

INC DX

CMP DX, 30

JE FUNC3

LOP4: INC BX

MOV AH, IN\_ITEM + 2[BX]

MOV AL, [SI]+[BX]

CMP AH, AL

JNE LOP3

CMP BX, CX

JNE LOP4

INC BX

CMP IN\_ITEM + 2[BX], 0DH

JNE LOP3

MOV BX, 0

CMP AUTH, 1

JE OPITEM

CMP AUTH, 0

JE RECOM

OPITEM: MOV DH, [SI]+[BX]

CMP DH, 0

JE OPITEM1

MOV DL, DH

MOV AH, 2

INT 21H

INC BX

JMP OPITEM

OPITEM1:LEA DX, CRLF ;回车换行

MOV AH, 9 ;回车换行

INT 21H

JMP FUNC1

RECOM: MOV AL, [SI] + 10 ;DISCOUNT IN AX

MOV AH, 0

MOV CX, [SI] + 13 ;SALE PRICE IN CX

MUL CX ;SALE \* DISCOUNT IN AX

MOV CX, 10

MOV DX, 0

DIV CX ;ACTUAL SALE PRICE IN AX

MOV BX, AX ;ACTUAL SALE PRICE IN BX

MOV AX, [SI] + 11 ;PURCHASE PRICE

MOV CX, 128

MUL CX ;PURCHASE PRICE \* 128 IN AX

MOV CX, BX ;ACTUAL SALE PRICE IN CX

MOV DX, 0

DIV CX ;PURCHASE PRICE \* 128 / ACTUAL SALE PRICE IN AX

MOV BX, AX ;PURCHASE PRICE \* 128 / ACTUAL SALE PRICE IN BX

MOV AX, [SI] + 15 ;NUM OF PURCHASE IN AX

MOV CX, 2 ;2 IN CX

MUL CX ;2 \* NUM OF PURCHASE IN AX

MOV DI, AX ;2 \* NUM OF PURCHASE IN DI

MOV AX, [SI] + 17 ;NUM OF SALE

MOV CX, 128

MUL CX ;NUM OF SALE \* 128 IN AX

MOV CX, DI ;2 \* NUM OF PURCHASE IN CX

MOV DX, 0

DIV CX ;NUM OF SALE \* 128 / 2 \* NUM OF PURCHASE IN AX

ADD BX, AX

JMP FUNC4

FUNC4: CMP BX, 100

JA PUTA

CMP BX, 50

JA PUTB

CMP BX, 10

JA PUTC

CMP BX, 10

JBE PUTF

PUTA: MOV DL, 'A'

MOV AH,2

INT 21H

LEA DX, CRLF ;回车换行

MOV AH, 9 ;回车换行

INT 21H ;回车换行

JMP FUNC1

PUTB: MOV DL, 'B'

MOV AH,2

INT 21H

LEA DX, CRLF ;回车换行

MOV AH, 9 ;回车换行

INT 21H ;回车换行

JMP FUNC1

PUTC: MOV DL, 'C'

MOV AH,2

INT 21H

LEA DX, CRLF ;回车换行

MOV AH, 9 ;回车换行

INT 21H ;回车换行

JMP FUNC1

PUTF: MOV DL, 'F'

MOV AH,2

INT 21H

LEA DX, CRLF ;回车换行

MOV AH, 9 ;回车换行

INT 21H ;回车换行

JMP FUNC1

EXT: MOV AH, 4CH

INT 21H

CODE ENDS

END START

### 实验步骤

1.准备上机实验环境。

2.使用VISUAL STUDIO编辑程序，实现要求功能，保存至SHOP.ASM。使用MASM6.0汇编源文件，观察提示信息，若出错则返回重新编辑SHOP.ASM，保存后重新汇编，直至不再报错为止。

3.使用连接程序LINK.EXE将生成的SHOP.OBJ文件连接成执行文件。

4.执行程序。按照程序设计要求进行交互，检查是否达到程序设计要求。

5. 使用TD.EXE观察SHOP.EXE的执行情况。即 TD SHOP.EXE回车

（1）观察CS、IP、SP、DS、ES、SS的值。

（2）单步执行开始2条指令，观察DATA的实际值，以及DS的改变情况。

（3）观察DS：0开始数据区，找到各变量在数据段中的位置和值。

（4）观察执行9号语句时，源操作数DX的值是否与待输出变量的偏移地址相同。

（5）执行10号语句指令，输入用户名。观察IN\_NAME缓冲区的值的变换。

（6）执行字符串比较段时，观察比较过程中程序如何转跳。

（7）执行乘法与除法操作时，注意观察AX，DX值的变化以及计算后结果传向何处。

6. 将程序重新装入TD中（或将CS：IP重置到MOV AH，9的位置），在执行9号功能调用之前，用TD将数据段中INPUT缓冲区的‘$’（24H）改成其他数值（如00H），再执行9号功能调用，观察现象。

7. 10号功能调用时，输入的字符数超过定义的数量时，观察现象。

### 实验记录与分析

1. 实验环境条件：i7-7700HQ 2.80GHz，8G内存；WINDOWS 10下DOSBox0.72；TD.EXE 5.0。

2. 汇编源程序时未发生异常

3. 连接过程未发生异常

4. 模块功能测试：

（1）功能一测试：

网店名称及输入提示语输出测试，测试结果如图1.17所示，测试结果显示该功能正常。



图 1.17 输出网店名称与提示语句测试截图

未登录模式访问测试，测试结果如图1.18所示，测试结果显示该功能正常。

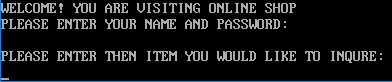


图1.18 未登录模式访问测试截图

退出程序测试，测试结果如图1.19所示，测试结果显示该功能正常。

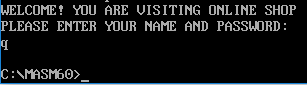


图1.19 退出程序测试截图

登录模式访问测试，测试结果如图1.20所示，测试结果显示该功能正常。

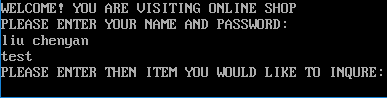


图1.20 登录模式访问测试截图

（2）功能二测试：

比较用户输入姓名正确性测试，测试结果如图1.20和1.21所示，测试结果显示该功能正常。

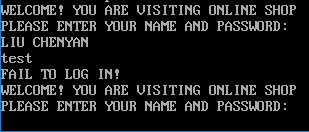


图1.21(a) 错误输入用户名测试截图

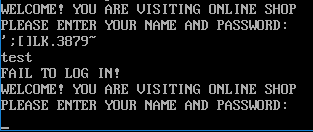


图1.21(b) 错误输入用户名测试截图

比较用户入密码正确性测试，测试结果如图1.20和1.22所示，测试结果显示该功能正常。

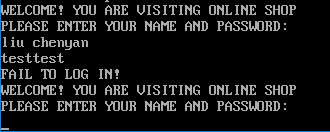


图1.22(a) 错误输入密码测试截图

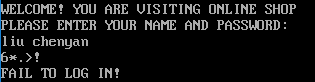


图1.22(b) 错误输入密码测试截图

登陆失败并回到功能一测试，测试结果如图1.21和1.22所示，测试结果显示该功能正常。

登陆成果并进入功能三测试，测试结果如图1.20所示，测试结果显示该功能正常。

（3）功能三测试

输出查询提示测试，测试结果如图1.23所示，测试结果显示该功能正常。



图1.23 输出查询提示测试截图

未能找到输入商品测试，测试结果如图1.24所示，测试结果显示该功能正常。

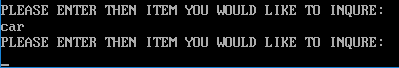


图1.24 未能找到输入商品测试截图

输入回车返回功能一测试，测试结果如图1.25所示，测试结果显示功能正常。

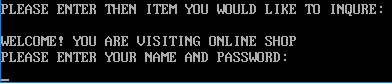


图1.25 输入回车返回功能一测试截图

登陆模式下正确查询测试，测试结果如图1.26所示，测试结果显示功能正常。

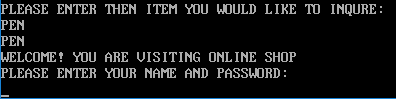


图1.26 登陆模式下正确查询测试截图

未登录模式下正确查询测试，测试结果如图1.27所示，测试结果显示功能正常。

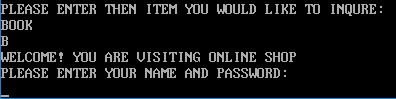


图1.27 未登录模式下正确查询测试截图

（4）功能四测试

推荐度显示测试，测试结果如图1.27所示，测试结果显示功能正常。

5.用TD调入shop.exe后

（1）(CS) = 0B0AH, (IP) = 0000H, (SP) = 00C8H, (DS) = 0AB7H, (ES) = 0AB7H, (SS) = 0AC7H。可从段首址取值的不同得知，代码段、数据段、堆栈段处在不同的位置，且先后顺序为：数据段、堆栈段、代码段。

（2）单步执行第二条指令，此时(DS) = 0AD4H，TD数据区跳至ES:0000H处，若需回到DS:0000H，则需要使用goto指令。发现此时各段的实际次序为：堆栈段、数据段、代码段，与源程序定义的各段的次序相一致。

（3）DS：0000H开始的数据区存放了BNAME变量为首址的用户名，DS：000CH开始的数据区存放了以BPASS变量为首址的密码。DS：0012至DS：0045分别为：IN\_NAME、IN\_PWD、IN\_ITEM的缓冲区，DS：0046开始的数据区存放了网店名称，DS：0051H开始的数据区存储了商品的各种信息。具体存放形式如图

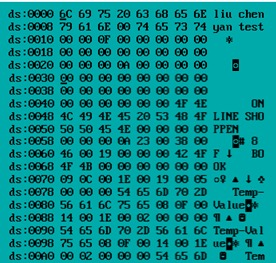


图1.28 具体部分数据存放情况截图

其中，DS：02C7H开始存放了程序运行中所需要的所有提示信息字符串，具体存放形式如图1.29所示。

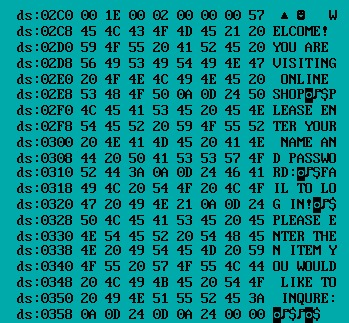


图1.29 字符串具体存放截图

（4）程序执行9号语句时，源操作数DX的值与待输出变量的偏移地址相同，如图1.30所示。

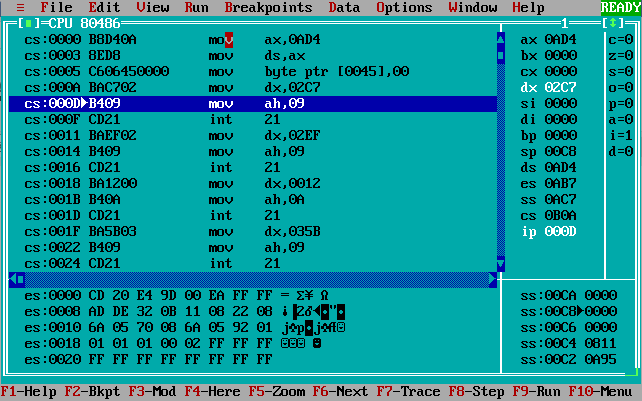


图1.30 DX的值测试截图

（5）执行10号语句指令时，正确输入用户名，执行后找到对应位置观察IN\_NAME中的值的变换。其中IN\_NAME + 1处的值为0BH，为字符串长度，IN\_NAME + 2 开始为具体字符串值。如图1.31所示。



图1.31 字符串存储截图

（6）执行字符串比较时，采用循环结构，程序在CS:0066H至CS:0077H中循环跳转，每次循环CX递减，BX递增，具体如图1.32：

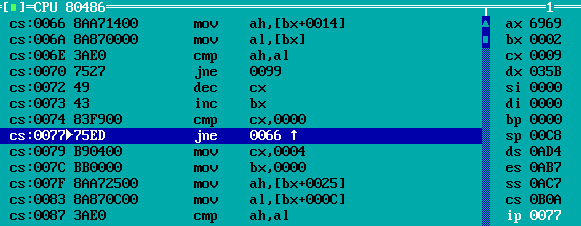


图1.32 循环结构运行截图

（7）执行乘法操作时，AX的值发生了改变而由于高位为零，DX无变化，乘法的结果存入了AX中，如图1.33所示：

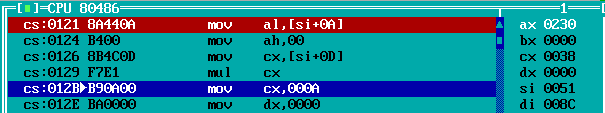


图1.33 乘法运算后寄存器情况截图

执行除法操作时，AX和DX的值都有可能发生变化，除法的商存入AX中，余数存入DX中，如图1.34所示。

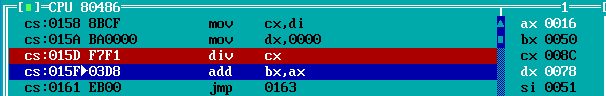


图1.34 除法运算后寄存器情况截图

6.将程序重新装入TD中，在执行9号功能调用前，用TD将数据段中缓冲区DS:02E7的24H改为其他数值，结果如图1.35和1.36所示。可见程序在没有遇见24H的情况下继续输出，直到遇到在下一段字符串末尾的24H后停止输出。

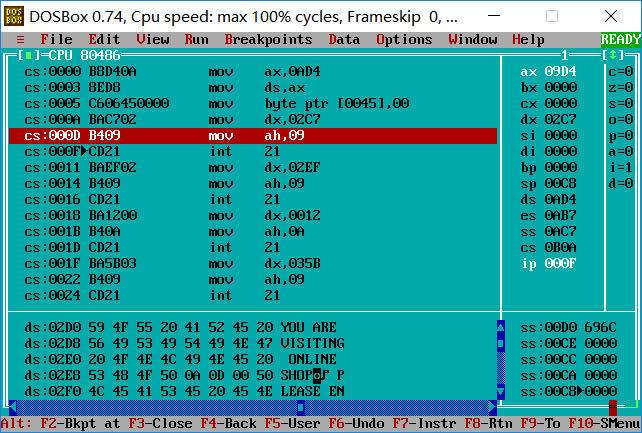


图1.35 修改缓冲区数据截图

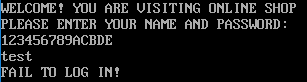


图1.36 程序输出截图

7. 10号功能调用时，输入的字符数超过或等于定义的数量（15）时，由于无法读入回车，程序无法完成输入动作。当输入的字符小于等于定义的数量（15）时，程序才能完成输入动作，继续执行程序。如图1.37所示。



1.37（a） 输入的字符数超过或等于定义的数量时程序截图



1.37(b) 当输入的字符小于等于定义的数量时程序截图

# 总结与体会

本次实验中，实验者熟悉了DOSBOX的使用与操作环境，学会了建立目标源程序、汇编源文件，生成目标文件、连接目标文件与运行调试程序。其中对运行调试工具Turbo Debugger的使用有了基本的了解和熟悉。同时，实验者了解了反汇编的定义，对文件在内存中如何存储有了直观的认识。

实验者首先学会了如何对源代码文件进行汇编（MASM），连接（LINK）与运行，同时学会了将虚拟机挂载在需要的位置。

其次实验者对于Turbo Debugger有了基本的认识：了解TD界面有五个区域，其中代码区能够查看代码的偏移地址、机器指令和目标代码；数据区能够查看偏移地址，数据内容和各个字节所对应的ASCII符号；寄存器区能够查看各寄存器的值，且能够切换16位与32位寄存器；标志寄存器区列出了主要标志位的值；堆栈区则能够查看有效地址与该地址对应的字单元中的内容。在实验中学会熟练使用一些常见的快捷键如：F2：添加断点，F5：放大，F8：单步调试，F9：运行，F10：菜单栏。同时在代码区使用右键GOTO可以转跳到制定的代码段，而在数据区使用GOTO则可以转跳至对应的偏移地址，查看所存储的数据。

在任务一的第二小题中遇到的问题是对于负数的输入存在问题，以-0101001B为例，尝试输入：“10101001B（加符号位）”、“-0101001B（直接输入）”、“11010111B（补码）”，其中直接保留符号输入与补码输入效果相同，存在溢出的情况，结果均为7AH，因为溢出而产生错误。而采用直接加符号位的方式输入，可以在AH获得正确答案的绝对值86H，符号标志SF为1。由于直接加符号位的做法不符合实际应用，所以最终选择直接输入的方式计算结果，即使存在着由溢出而引发的错误。通过这个问题，弄清了标志寄存器中各个主要标志位所代表的意义，能够帮助实验者更好的了解计算的经过。

在任务二中，实验者学会了如何添加断点并使用一些快捷键，在需要快速跳至断点时，可选择F9直接运行至断点，而在观察数据段开始的40个字节内容时，可选择F8单步调试，帮助理解每一行机器指令的作用。同时，学会在数据区使用GOTO，快速跳转至所关心的偏移地址位置。

在实验三中，实验者通过将任务二中的代码改写成满足任务三要求的代码，加深了对汇编语言寻址方式的理解，例如在一开始，没有意识到题目中：“变址寄存器用32位寄存器”的要求，仍旧保留了SI,DI,BX,BP等寄存器，导致汇编失败。将其改为ESI,EDI,EBX,EBP之后汇编成功。在比较反汇编语言与源代码时，加深了对机器码与汇编语句对应关系的理解。在代码区使用GOTO转跳时，理解了IP指明起始位置的重要性，同时学会了在代码区快速转跳。

在实验四中，实验者第一次完整实现了一个汇编程序，尤其是对寄存器的使用，不同寻址方式的理解有了更深的认识，并通过debug与实验报告撰写的过程提高了使用TD的熟练程度。

本次实验中，实验者更加体会到了实验前预习的重要性，上机前准备越充分，上机时的目标就更加明确，解决问题也更有针对性。

同时，通过解决程序中出现的bug，认识了一些报错提示，例如A2008说明指令语法错误。

本次实验，对汇编语言有了一个具体的认识，但是对DOSBOX和TD的使用仍有加强之处，同时自己汇编的能力还有待提高。

# 参考文献

[1] 许向阳.80X86汇编语言程序设计上机指南.武汉:华中科技大学出版社,2007：1-61

[2] 王元珍.曹忠升.韩宗芬. 80X86汇编语言程序.武汉:华中科技大学出版社,2007