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： 2019-3-20，8：00-11：50　**实验地点：** 南一楼 803 室

指导教师： 李专

专业班级： CSIE1701 班

学　　号： U201715264　**姓　　名：** 邹子一

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：2019.3.20

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) 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。

任务 2. 《80X86 汇编语言程序设计》教材中 P45 的 2.3 题。

任务 3. 《80X86 汇编语言程序设计》教材中 P45 的 2.4 题的改写。

任务 4. 设计实现一个网店商品信息管理的程序。

本次实验中需要学生尝试体会与解决的操作及问题可分为三类：

1. TD 的操作（直接在 TD 内输入指令，设置断点、单步执行程序、运行到断点、运行到指定的指令、查看寄存器的内容、查看标志寄存器的某个标志位的值、查看指定的存储单元的值、查看整个程序在内存中的存放方式）；

2. 将一个汇编源程序生成一个可执行文件的步骤，包括如何读懂汇编源程序在汇编过程中产生的错误信息；编辑工具的使用（使用 Edit， notepad 等需要注意的问题）；

3. 操作数寻址方式的改变，对目标码产生的影响； DOS 系统功能调用中应注意的问题。

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 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 SI
         DEC CX
         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 1

计算$[x1]_{补}+[x2]_{补}$，将计算结果存储在 AH 中，并记录 SF、OF、ZF、CF 的状态。

1) X1=+0110011B，X2=+1011010B

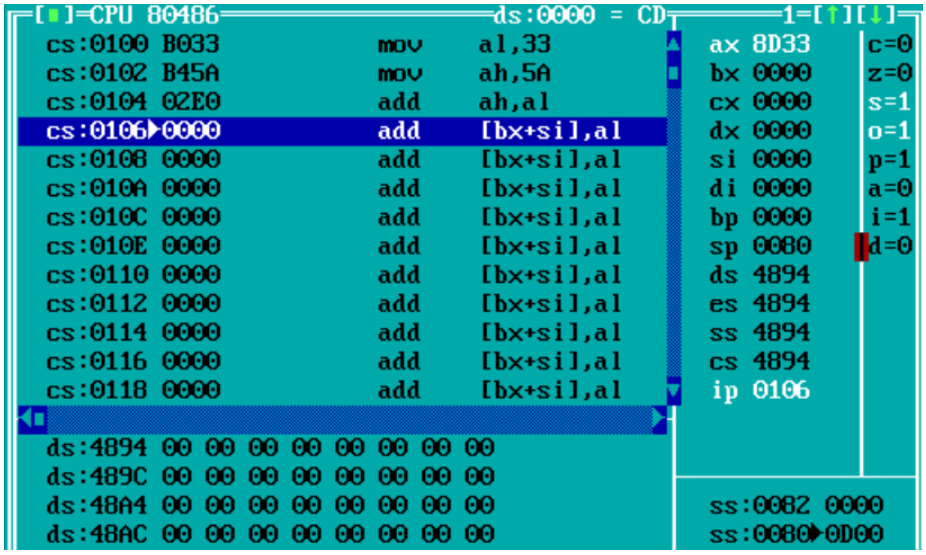


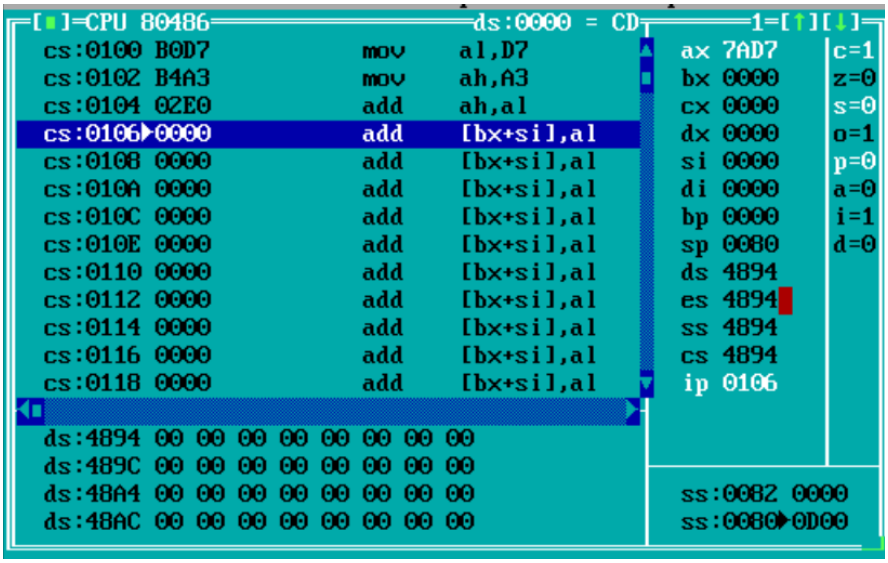
图 1-1 计算结果截图

由图可见，相加之后，S=1，O=1，发生了溢出，并且结果为负数。

2) X1=-0101001B，X2=-1011101B

由图可见，相加之后，C=1，O=1，发生了溢出，并且存在进位。

图 1-2 计算结果截图



3) X1=+1100101B，X2=-1011101B

由图可见，相加之后，C=1，发生了进位。

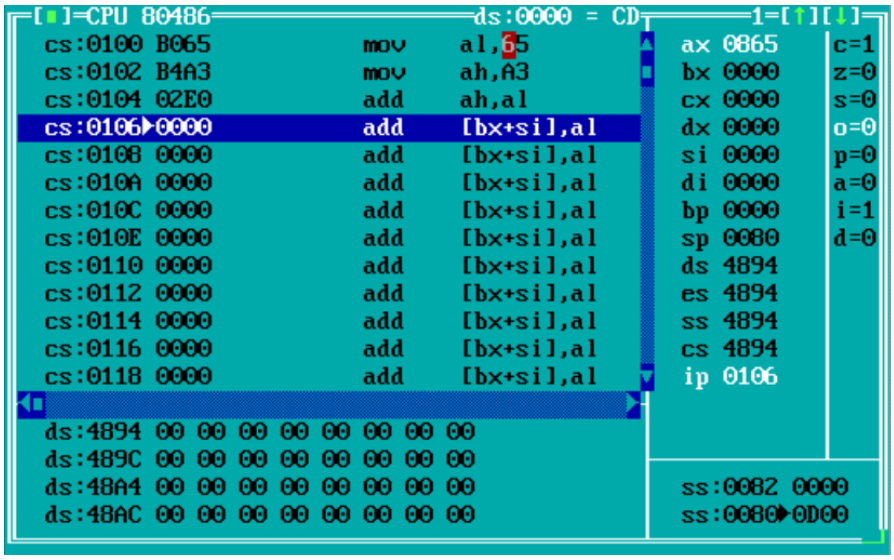


图 1-3 计算结果截图

有符号数与无符号数运行结果的比较分析

对于本题而言，AL、AH 均为 8 位寄存器，若参与运算的数字符号相同，相加之后第 7 位发生进位，使第 8 位（符号位）改变，若超出了有符号数的范围，即为溢出，否则是发生了进位。若将数字视作有符号数，则数字只会发生溢出；若将数字视作无符号数，则数字只会发生进位。

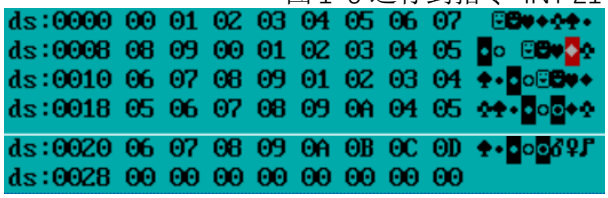
任务 2

在本任务中，在代码指定位置加上断点，并分析运行到断点时寄存器、内存的状态。

图 1-4 运行到指令 MOV CX，10 时寄存器状态截图

由图可见，SI=0,DI=10,BX=20,BP=30.

图 1-5 运行到指令 INT 21 时寄存器状态截图



由图可见，SI=10,DI=20,BX=30,BP=40.

图 1-6 运行到指令 INT 21 时 DS 段内存状态截图

由图可见，内存中原本的 30 个 0 被改写成了 0-9、1-10、4-13.

单步执行一条指令的多种方法

方法 1：在 TD 中，使用 F7（Trace）或 F8（Step）进行单步跟踪调试。
方法 2：通过使用断点与 F9（Run）进行单步跟踪调试。
方法 3：通过改变 IP 寄存器或使用 Goto 进行单步执行指令。

任务 3

该任务将书本上 2-4 的题目改写为使用 32 位寄存器。具体源代码参见 3.1.1。

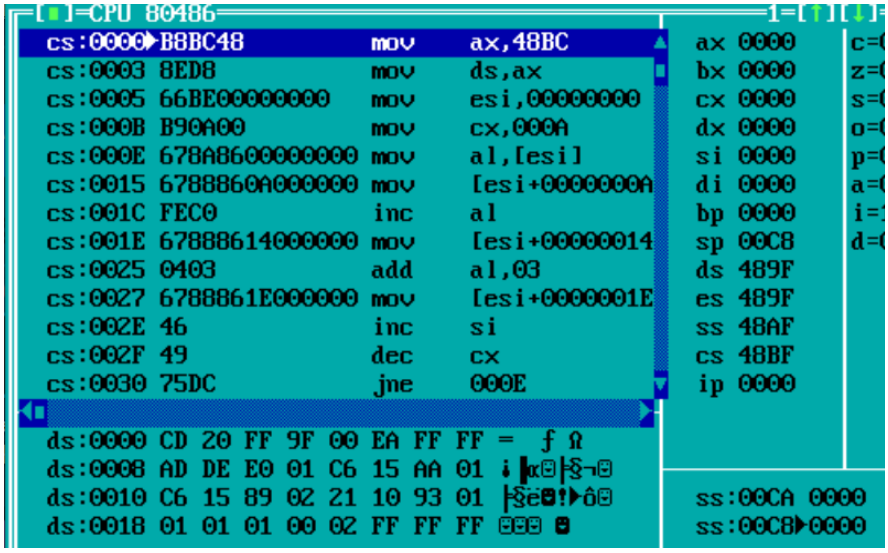


图 1-7 使用 32 位寄存器时，反汇编界面截图

由图可见，此时寄存器中数值的存储方式默认为 32 位。

图 1-8 切换为 32 位寄存器后寄存器内数值存储方式

二进制反汇编成汇编指令

TD 通过对二进制指令进行反汇编，形成可读的指令。

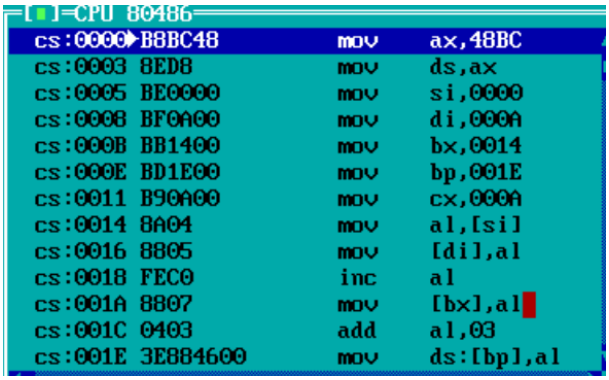


图 1-9 部分指令反汇编结果

```
13 START:  MOV AX,DATA
14         MOV DS,AX
15         MOV SI,OFFSET BUF1
16         MOV DI,OFFSET BUF2
17         MOV BX,OFFSET BUF3
18         MOV BP,OFFSET BUF4
19         MOV CX,10
20 LOPA:   MOV AL,[SI]
21         MOV [DI],AL
22         INC AL
23         MOV [BX],AL
24         ADD AL,3
```

```
es 489F
fs 0000
gs 0000
ss 48AF
cs 48BF
ip 000E
```

图 1-10 对应源代码截图

对比两图，可见：汇编指令的二进制码每一行长度不相等，可能为 2、4、8、10…；汇编指令中的地址常量，被编译器翻译成了具体的地址；10 进制数、带符号数等，都被统一转换成了补码。若更改 ip 寄存器，将其指向一条命令的中段执行，会发生意想不到的结果。反之，在源文件中，使用伪指令 `DB 08EH,0D8H`，代替 mov ds，ax 这一行，在 TD 中显示的反汇编指令也是 mov ds，ax。由此可见，编译器将汇编指令翻译为二进制程序，而 TD 通过反汇编解读二进制文件的内存，并通过一一映射关系将其重新解读为汇编指令。

3.2 任务 4

3.2.1 设计思想及存储单元分配

本程序将主要的数据存储在 Data 段，并通过寻址访问内存。通过建立子程序，将比较字符串、输出字符串的部分模块化，更具有可读性。

Data 段具体分配如下：

BNAME、BPASS 用于记录账户及密码，N 为商品总数量，SNAME 为网店名称，

GA1、GA2、GAN 分别为定义的商品，in_name 和 in_pwd 用于记录用户登录时所输入的用户名和密码,BUF 作为缓冲区使用，AUTH 用于标志用户是否登录，REAL 和 TEMP 用于临时存放数据,STR1-STR6 用作显示的字符串。

3.2.2 流程图

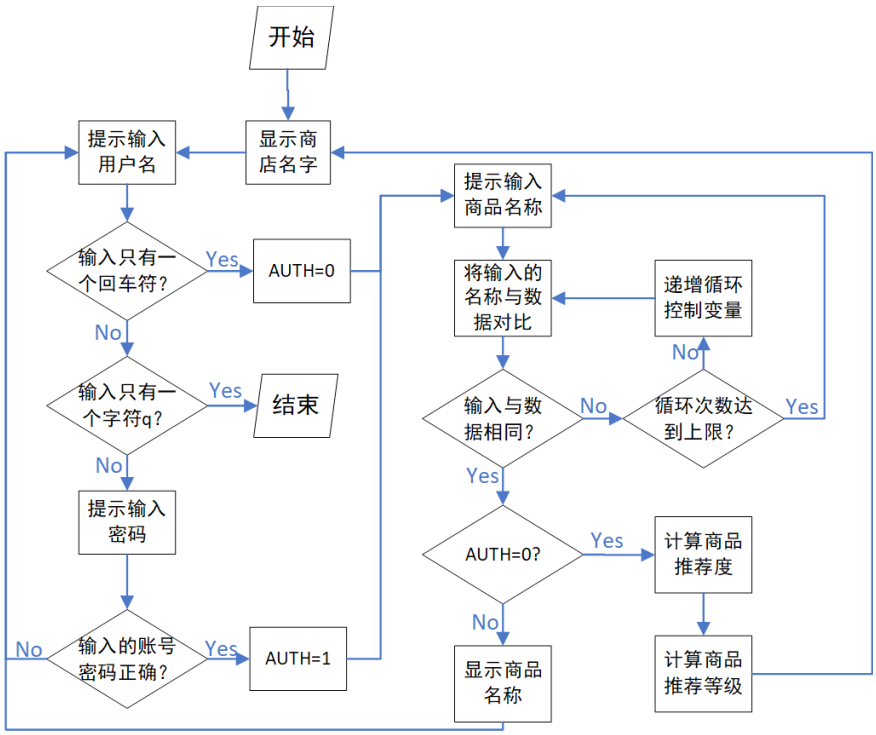


图 1-11 任务 4 主要流程图

3.2.3 源程序

```
.386
DATA    SEGMENT USE16
BNAME   DB  'zzy',7 DUP(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
SNAME   DB 'SHOP',0               ; 网店名称,用0结束
GA1     DB 'PEN', 7 DUP(0),10   ; 商品名称及折扣
        DW 35,56,70,25,?          ; 推荐度还未计算
GA2     DB 'BOOK', 6 DUP(0) ,9 ; 商品名称及折扣
        DW 12,30,25,5,?           ; 推荐度还未计算
GAN     DB N-2 DUP( 'TempValue',0,8,15,0,20,0,30,0,2,0,?,?)
BUF     DB 30,?,30 DUP('$')
in_name DB 10,?,10 DUP(0)
in_pwd  DB 6,?,6 DUP(0)
AUTH    DB ?
STR1    DB 30 DUP('$')
STR2    DB 'The name of the netshop you wanna visit is:$'
STR3    DB 'Please input your name:',10,'$'
STR4    DB 'Please input your password:',10,'$'
STR5    DB 'Wrong input.',10,'$'
STR6    DB 'Please input the name of the good you wanna query:',10,'$'
DATA    ENDS
STACK   SEGMENT USE16 STACK
        DB 200 DUP(0)
STACK   ENDS
```

```
CODE    SEGMENT USE16
        ASSUME  DS:DATA,SS:STACK,CS:CODE
; 子程序:COMP
; 用途:比较两个字符串
; 参数:DATA 段中元素和 INPUT 的元素
; 传参形式:栈
COMP    PROC
        pop cx
        pop bp
        pop bx
        push cx
        mov al,ds:[bp]
        mov ah,ds:[bx]
        jmp CONT
INCR:   inc bp
        inc bx
        mov al,ds:[bp]
        mov ah,ds:[bx]
        cmp al,0dH
        je  PRERET
CONT:   cmp al,ah
        je INCR
        jne ERROR
ERROR:  mov ax,0
        ret
PRERET: mov al,1
        cmp ah,0
        je  RETURN
        jne ERROR
RETURN: mov ax,1
        ret
COMP    ENDP
; 子程序: SHOWPTR
; 用于显示字符串
; 参数:字符串地址
; 传递参数形式:栈
SHOWSTR PROC
        pop cx
        pop bx
        push cx
        mov si,0
MAKE:   mov al,ds:[bx]
        cmp al,0
        je PRINT
        mov STR1[si],al
        inc si
        inc bx
        jmp MAKE
PRINT:  mov STR1[si],'$'
        lea dx,offset STR1
        mov ah,9
```

```
        int 21H
        mov dl,10
        mov ah,2
        int 21H                    ; 换行
        ret
SHOWSTR ENDP
START:  mov     ax,DATA
        mov     ds,ax
;以下为功能1
LOGIN:  lea dx,offset STR2         ; 输出商店信息
        mov ah,9
        int 21h
        mov bx,GA1-SNAME
        sub bx,1
        mov SNAME[bx],'$'
        lea dx,offset SNAME
        mov ah,9
        int 21H
        mov dl,10
        mov ah,2
        int 21H                    ; 换行
        lea dx,offset STR3
        mov ah,9
        int 21H                    ; 要求输入名字
        lea dx,offset in_name
        mov ah,10
        int 21H                    ; 将名字存入in_name
        cmp in_name[2],0dH         ; 判断用户名是否为空
        je  SETAUTH
        cmp in_name[1],1           ; 判断用户名是否为q
        je PREEXIT
        jmp CLOGIN
SETAUTH:mov AUTH,0
        jmp CALC
CLOGIN: mov dl,10
        mov ah,2
        int 21H                    ; 换行
        lea dx,offset STR4
        mov ah,9
        int 21H                    ; 要求输入密码
        lea dx,offset in_pwd
        mov ah,10
        int 21H                    ; 将密码存入in_pwd
        mov dl,10
        mov ah,2
        int 21H                    ; 换行
;以下为功能2
COMPARE:mov bx,offset BNAME
        push bx
        mov bx,offset in_name
        add bx,2
```

```
        push bx
        call COMP
        cmp ax,1
        jne ERRINFO
        mov bx,offset BPASS
        push bx
        mov bx,offset in_pwd
        add bx,2
        push bx
        call COMP
        cmp ax,1
        jne ERRINFO
        mov AUTH,1
        jmp CALC
ERRINFO:lea dx,STR5
        mov ah,9
        int 21H
        jmp LOGIN
;以下为功能3
CALC:    lea dx,offset STR6
        mov ah,9
        int 21h
        lea dx,offset BUF
        mov ah,10
        int 21H                         ; 将商品名称存入buf
        cmp BUF[2],0dH
        je LOGIN
        mov cx,0
FIND:
        push cx
        mov bx,offset GA1
        mov ax,cx
        imul ax,21
        add bx,ax
        push bx
        mov bx,offset BUF
        add bx,2
        push bx
        call COMP
        pop cx
        cmp ax,1
        je  CALC2
        inc cx
        cmp cx,N
        je  ERRNAME
        jmp FIND
ERRNAME:mov dl,10
        mov ah,2
        int 21H
        lea dx,STR5
        mov ah,9
```

```
        int 21H
        jmp CALC
CALC2:   cmp AUTH,0
        je CALC4
CALC3:   mov dl,10
        mov ah,2
        int 21H                    ; 换行
        imul cx,21
        mov ax,cx
        add ax,offset GA1
        push ax
        call SHOWSTR
        jmp LOGIN
CALC4:
        mov ax,cx
        mov cx,21
        mul cx
        mov bx,offset GA1
        add bx,ax
        add bx,10
        mov al,0[bx];折扣
        jz CALC5
        mov dx,3[bx];实际售价
        mul dx
        mov cx,ax;cx=折扣*实际售价
        mov ax,1[bx]
        mov dx,1280
        mul dx;ax=1280*进货价
        div cx;ax=rec1
        mov 9[bx],ax
        mov ax,7[bx]
        mov dx,128
        mul dx
        mov cx,5[bx]
        div cx
        add ax,9[bx];ax=rec2
        mov 9[bx],ax
        jmp REC
CALC5:
        mov ax,256
        mov 9[bx],ax
REC:
        mov dl,10
        mov ah,2
        int 21H
        mov ax,200
        cmp 9[bx],ax
        jbe  RECB
        mov dl,'A'
        mov ah,2
        int 21H
```

```
        jmp PRELOGIN
RECB:
        mov ax,150
        cmp 9[bx],ax
        jbe RECC
        mov dl,'B'
        mov ah,2
        int 21H
        jmp PRELOGIN
RECC:
        mov ax,100
        cmp 9[bx],ax
        jbe RECD
        mov dl,'C'
        mov ah,2
        int 21H
        jmp PRELOGIN
RECD:
        mov ax,50
        cmp 9[bx],ax
        jbe RECE
        mov dl,'D'
        mov ah,2
        int 21H
        jmp PRELOGIN
RECE:
        mov dl,'E'
        mov ah,2
        int 21H
        jmp PRELOGIN
PRELOGIN:
        mov dl,10
        mov ah,2
        int 21h
        jmp LOGIN
PREEXIT:cmp in_name[2],'q'
        je EXIT
        jmp CLOGIN
EXIT:    mov AH,4CH
        int 21H
CODE    ENDS
END START
```

3.2.4 实验步骤

1.编写源程序，通过MASM编译，并在DOSBox中运行。

2.检验功能的正确性,遇到与预期不相符的输出与死循环，先对代码的跳转逻辑进行观察，再通过TD进行断点设置与逐步执行调试，通过观察内存与寄存器，分析错误的原因。

3.修改代码，重新编译、运行。

在功能性方面，重点检查的程序功能如下：

（1）输入用户名时，输入q能否正常退出

（2）输入店主的用户名时，如果店主名字是输入的用户名的字串，是否能正常判断出错误。

（3）不输入姓名和密码登陆时，如果查询商店中没有的商品，能否输出未找到该商品的提示，并要求顾客再次输入想要查询的商品

（4）输入店主的用户名/密码时，如果输入了错误的用户名/密码，能否提示输入错误并重新输入。

（5）以店主的身份登陆时，能否正确输出查询商品的名字。

（6）第一次以店主身份登录，第二次以访客身份登录，是否会对内存中的数据产生干扰，导致第二次运行结果不正确。

3.2.5 实验记录与分析

1.实验环境条件：DOSBox 0.74.

2.在编写源程序过程中，发现的主要语法错误如下：

（1）AX 寄存器不能用做变址寻址的寄存器

（2）LEA 指令的操作数不能是常量

（3）CMP 指令的操作时不能是常量

3.在实验过程中，遇到的主要逻辑错误如下：

（1）输入商品名称后，预期其会在下一行重新输出商品名字，但没有。

解决方案：通过查询资料，得知此处需要加入一个换行符，才能正确输出。因此通过调用 DOS 系统 10 号功能输出一个换行符，再输出商品名称，成功解决该问题。

（2）输入错误的用户名/密码时，会重复报错，占满整个屏幕。

解决方案：通过观察源代码，发现报错部分结束后没有设置跳转回重新输入，导致进入死循环。因此通过加入到重新输入用户名的跳转，成功解决该问题。

（3）当折扣为 0 时，程序卡住，无法继续执行。

解决方案：通过观察源代码，并用 TD 反汇编调试，发现发生了错误“Divided by zero”，分析后发现当折扣为 0 时，会发生 DIV 0 的错误。因为 0 不能为被除数，使得程序崩溃。因此通过加入对折扣是否为 0 的判断，成功解决该问题。

（4）对商品的推荐度计算有误。

解决方案：通过 TD 反汇编发现，用于存储数据区基址的寄存器 BX 因为意外操作被清空了，导致无法访问正确地址。因此通过调整使用的寄存器，成功解决该问题。

（5）对商品推荐度的计算仍然有误。

解决方案：通过观察发现，对前一步用于存储循环控制变量的寄存器 CX，乘的系数是 20，但 DATA 段每个商品所占内存是 21，导致发生偏差。通过修改为 MUL 21，成功解决该问题。

（5）对商店中的默认商品 BOOK 和 PEN，若先查询 PEN，能在下一行正确输出 PEN，但如果先查询 BOOK，再查询 PEN，会输出 PENK。

解决方案：通过 TD 反汇编发现，循环控制变量错误的多加了 1，导致对'$'符号放置的位置发生了错位。通过修改循环逻辑，成功解决该问题。

（6）在使用 DX 作为被除数时，同样会发生“Devided by zero”错误。

解决方案：通过研究 DIV 的运算逻辑得知，DX 会被用来存储余数，中间的运算过程导致了如果 DX 作为被除数，会使得运算过程中产生 DIV 0，使得程序崩溃。通过修改寄存器，成功解决该问题。

（7）运算时可能会发生溢出。

解决方案：通过分析 MUL、DIV 的运算逻辑，发现 MUL 做 16 位乘法时，会自动将 AX

扩充到 DX；而 DIV 是将 DX、AX 整体作为被除数。通过使用连续的 MUL、DIV，解决了溢出问题。

（8）将折扣 mov 进 AX 寄存器时，读取的数值与预期不同。

解决方案：`mov ax,0[bx]`指令，编译器默认的 0[bx]的类型为 WORD，而实际上折扣在内存中存放的形式是 BYTE，类型不符，导致读取结果有误。修改为 `mov al,0[bx]`，并提前将 ah 置为 0 后，成功解决该问题。

（9）调用子程序后，无法正确回到主程序中的位置。

解决方案：通过观察 TD 反汇编结果，发现调用子程序后，栈顶元素发生了变化；此时如果手动操作栈，会使得返回主程序时，IP 寄存器中的值错误。因此，在进入子程序后，先使用 `pop cx` 将栈顶的值保存，然后将栈中需要的元素取出后，使用 `push cx` 将之前在栈顶的值重新放回栈顶，成功解决该问题。

4 总结与体会

4.1 任务 1-3 的相关总结与体会

通过本次实验，我初步了解了 DOSBox 的使用方法、汇编语言环境的搭建，能使用 TD 等工具进行简单的 Debug 等操作，对计算机内底层命令的执行方式、cpu 的工作方式、内存中数据的存储方式都有了更深入的了解，为对汇编语言的深入学习与计算机的深入学习奠定了基础。

4.2 任务 4 的相关总结与体会

通过本次实验，我掌握了对 DOS 系统功能 2、9、10、12 的调用，初步理解了分支、循环、子程序等简单的程序设计方法，初步掌握了 MUL、DIV 等指令的使用方式，对溢出的避免方式，子程序与主程序间值的传递方式，更深入的了解了各种寻址方式的使用技巧内存中数据的存储方式，8 位、16 位、32 位的运算与存储区别等知识。

5 参考文献

[1]《80X86 汇编语言程序设计》，王元珍，曹忠升，韩宗芳编著，华中科技大学出版社，2005 年 4 月第一版