

**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： 2018-3-19，14：00-17：30 实验地点： 南一楼804室**

**指导教师：**

**专业班级：计算机科学与技术201706班**

**学 号： U201714761 姓 名： 胡澳**

**同组学生： 无 报告日期： 2018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：2018.3.20

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 2](#_Toc4758397)

[2 实验内容 2](#_Toc4758398)

[3 实验过程 4](#_Toc4758399)

[3.1 任务1~3 4](#_Toc4758400)

[3.1.1 任务3源代码 4](#_Toc4758401)

[3.1.2 关键的实验记录与分析 5](#_Toc4758402)

[3.2 任务4 9](#_Toc4758403)

[3.2.1 设计思想及存储单元分配 9](#_Toc4758404)

[3.2.2 流程图 9](#_Toc4758405)

[3.2.3 源程序 9](#_Toc4758406)

[3.2.4 实验步骤 18](#_Toc4758407)

[3.2.5 实验记录与分析 18](#_Toc4758408)

[4 总结和体会 19](#_Toc4758409)

# 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。

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（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('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。

提示：字符串比较时，当采用输入串的长度作为循环次数时，若因循环次数减为0而终止循环，则还要去判断网店中定义的字符串的下一个字符是否是结束符0，若是，才能确定找到了（这样做是为了避免输入的字符串仅仅是数据段中所定义字符串的子集的误判情况）。

4、功能三：计算指定商品的推荐度。

(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。

要求尽量避免溢出。

提示：使用循环程序结构，注意寻址方式的灵活使用。结果只保留整数部分。

5、功能四：将功能三计算的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”。

提示：使用分支程序结构，采用2号DOS系统功能调用显示结果。

(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, BUF1[ESI]

MOV BUF2[EDI],AL

INC AL

MOV BUF3[EBX], AL

ADD AL,3

MOV BUF4[EBP],AL

INC ESI

INC EDI

INC EBP

INC EBX

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. 实验一

使用TD求题目中三组数据的和，实验记录截图如图1-1、1-2、1-3所示。

执行指令后，(AH)、SF、OF、CF、ZF的值见表1-1。

表1-1 执行指令后(AH)、SF、OF、CF、ZF的值

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 算式 | (AH) | SF | OF | CF | ZF |
| 0110011B + 1011010B | 8DH | 1 | 1 | 0 | 0 |
| -0101001B + (-1011101B) | 7AH | 0 | 1 | 1 | 0 |
| 1100101B + (-1011101B) | 08H | 0 | 0 | 1 | 0 |

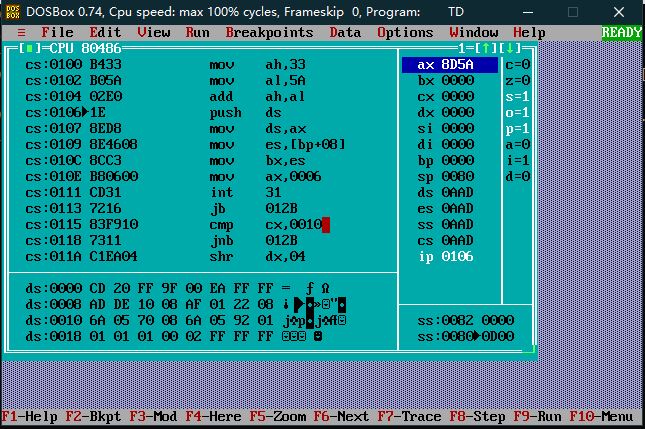


图1-1 0110011B + 1011010B计算结果截图

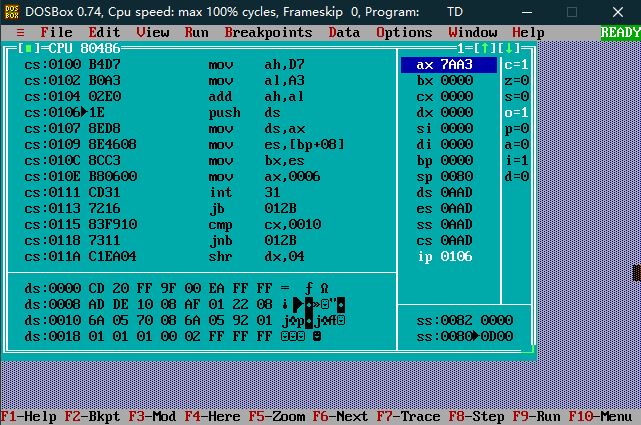


图1-2 -0101001B + (-1011101B)计算结果截图

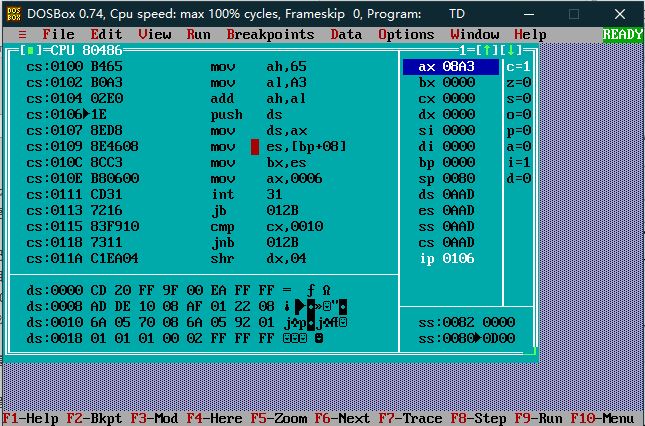


图1-3 1100101B + (-1011101B)计算结果截图

2. 任务二

使用文本编辑器将题2-3的源代码输入到problem2.asm中，在DosBox中使用masm编译、使用link链接得到problem2.exe的可执行文件。使用td打开该文件，首先单步执行，观察执行指令后相应寄存器以及标志位的变化。当执行到“MOV CX, 10”之前时，td界面的截图如图1-4所示。在指令“INT 21H”处添加断点，将程序执行到断点出，此时td界面的截图如图1-5所示，此时内存显示区使用goto功能，使该区域显示从DS:0000开始的内存空间的内容，该内存空间的内容的截图如图1-6所示。当程序执行到这两条指令前的时候，(BX)、(BP)、(SI)、(DI)的值见表1-2。

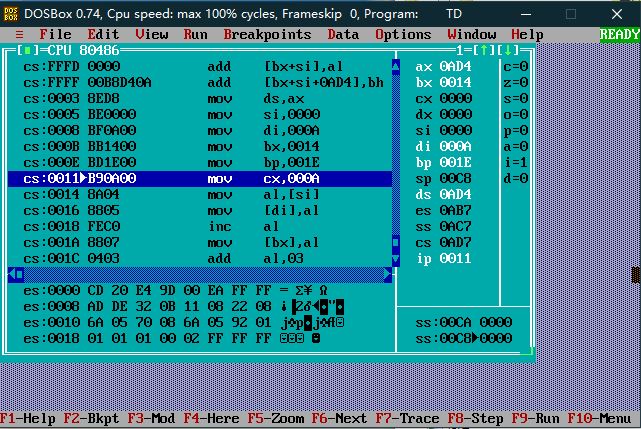


图1-4 程序执行到“MOV CX, 10”前的截图

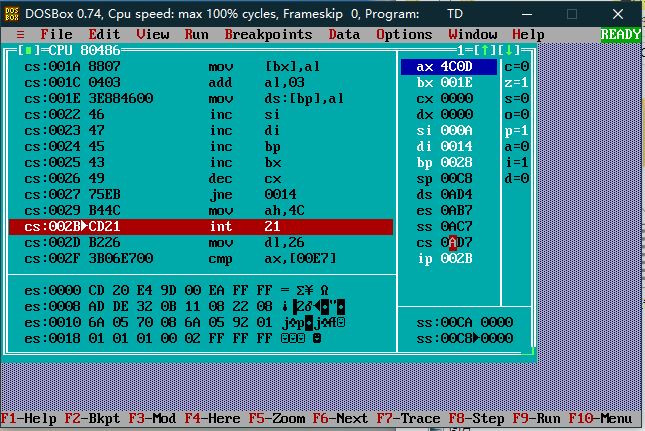


图1-5 程序执行到“INT 21H”前的截图

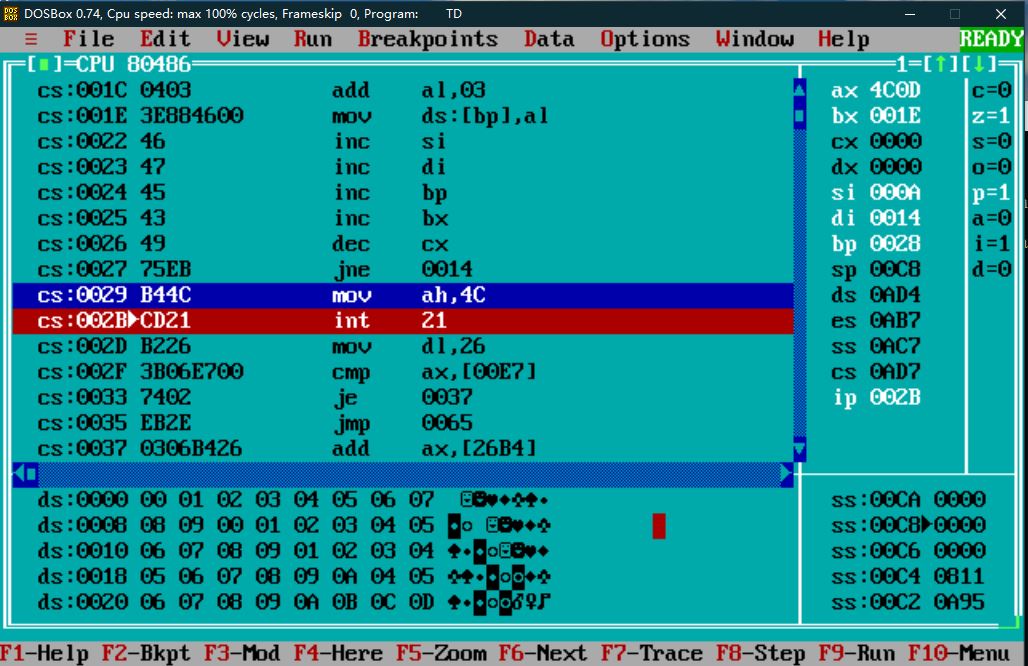


图1-6 程序执行到“INT 21H”前DS段内存的内容

表1-2 程序执行到相应指令前(BX)、(BP)、(SI)、(DI)的值

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令 | BX | BP | SI | DI |
| MOV CX, 10 | 14H | 1EH | 0 | 0AH |
| INT 21H | 1EH | 28H | 0AH | 14H |

3. 任务三

修改后的源程序见3.1.1，将该源程序编译链接后使用td打开调试，在程序结束前加上断点，并将程序执行到该断点的位置，然后将内存显示区域调整为显示DS段的内存内容，此时内存信息的截图如图1-7所示。

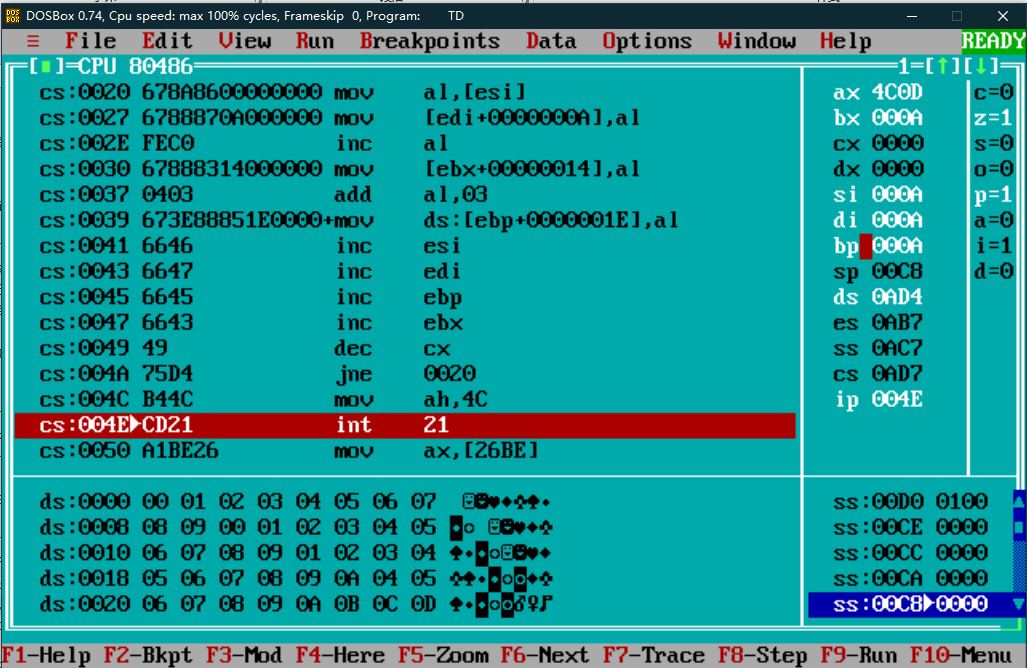


图1-7 修改后的程序运行到结束前DS段的内存内容

将该内存信息与修改前DS段的内存信息、即图1-6的内容相互比较，发现两者内容完全相等，由此可推断，修改前后的程序功能未发生变化。

从不同位置反汇编二进制串会产生不同的汇编指令，由此可知，IP/EIP寄存器指明下一条待执行指令的偏移地址对于程序的执行是至关重要的，否则，计算机将无法正确的解析可执行文件，从而无法正确执行程序。

## 3.2 任务4

### 3.2.1 设计思想及存储单元分配

程序的设计思路如下：调用DOS系统功能，在控制台与用户进行简单的交互。首先程序使用9号功能输出提示信息，并使用10号功能读取用户输入，然后根据输入判断用户是老板还是顾客，并将该信息存储到相应的内存单元中。判断规则为：若用户输入了用户名和密码，则判断输入是否正确，若正确，则为老板，否则，输出登陆错误；若用户未输入数据，则认为用户为顾客。然后程序提示用户输入待查看的商品的名称，程序读取到商品名称后查找该商品，若找到该商品，则根据用户的类型给定相应的输出，若未找到该商品，则输出未找到该商品。

存储单元分配方案如下：所有的数据均存储在数据段中。老板名字以及密码均定义为10个字节的串，并在该串的后面存储一个数字，用于存储名字和密码的长度。商店名称定义为以‘$’结束的串。将用户类型定义为字节类型，1表示为老板，0表示为顾客。商品信息首先为一个10个字节的串，用于保存商品名称，接下来一个字节用于存储商品名称的长度，接下来一个字节为商品的折扣率，最后有5个字类型的数据，分别存储商品的进货价、销售价、进货数量、已售数量和推荐度。另外，还设有两个内存空间用于存储用户输入的用户名和密码。

### 3.2.2 流程图

程序流程图如图1-8所示。

### 3.2.3 源程序

程序的源代码如下：

.386

data segment use16

    bname   DB  'huao', 6 dup(0), 4     ;boss name

    bpass   DB  'huao', 6 dup(0), 4     ;password

    N       EQU 30                      ;number

    shopname    DB  'huao$'             ;shop name

    auth    DB  ?

    ga1     DB  'pen$', 6 dup(0)

            DB  3

            DB  10

            DW  35, 56, 70, 25, ?



图1-8 任务4流程图

    ga2     DB  'book$', 5 dup(0)

            DB  4

            DB  9

            DW  12, 30, 25, 5, ?

    welcomeMsg  DB  'welcome to ', '$'

    welcomeShopMsg  DB  ' shop$'

    loginErrorMsg   DB  'Login Error', 0ah, 0dh, '$'

    outenter    DB  0ah, 0dh, '$'

    info    DB  'input your name and password', 0ah, 0dh, '$'

    in\_name DB  10, ?, 10 dup(0), '$'

    in\_pass DB  6, ?, 6 dup(0), '$'

    checkname   DB  'input commodity you want to check:', '$'

    goodname    DB  10, ?, 10 dup(0)

    notFound    DB  'good not found', 0ah, 0dh, '$'

data ends

stack segment use16 stack

    DB 200 dup(0)

stack ends

code segment use16

    assume DS:data, CS:code, SS:stack

start:

    mov ax, data

    mov ds, ax

    lea dx, welcomeMsg[0]

    mov ah, 9

    int 21h

    lea dx, shopname[0]

    mov ah, 9

    int 21h

    lea dx, welcomeShopMsg[0]

    mov ah, 9

    int 21h

    lea dx, outenter[0]

    mov ah, 9

    int 21h

    lea dx, info[0]

    mov ah, 9

    int 21h

    lea dx, in\_name[0]

    mov ah, 10

    int 21h

    lea dx, outenter[0]

    mov ah, 9

    int 21h

    mov al, in\_name[1]

    cbw

    mov bx, ax

    mov ax, 0

    mov in\_name[bx + 2], al

    mov al, in\_name[1]

    cmp al, 0

    jz user

    cmp al, 1

    jz checkQuit

    jmp next3

checkQuit:

    mov al, in\_name[2]

    cmp al, 71h

    jz quit

next3:

    lea dx, in\_pass[0]

    mov ah, 10

    int 21h

    mov al, in\_pass[1]

    cbw

    mov bx, ax

    mov ax, '$'

    mov in\_pass[bx + 2], al

    lea dx, outenter[0]

    mov ah, 9

    int 21h

    lea bx, in\_name[2]

    mov al, in\_name[1]

    cmp al, bname[10]

    jnz errorLogin

    call compareName

    cmp ch, 1

    jnz errorLogin

    lea bx, in\_pass[2]

    mov al, in\_pass[1]

    cmp al, bpass[10]

    jnz errorLogin

    call comparePass

    cmp ch, 1

    jnz errorLogin

    mov al, 1

    mov auth[0], al

    jmp recommandGood

errorLogin:

    lea dx, loginErrorMsg[0]

    mov ah, 9

    int 21h

    jmp start

user:

    mov al, 0

    mov auth[0], al

    jmp recommandGood

recommandGood:

    lea dx, checkname[0]

    mov ah, 9

    int 21h

    lea dx, goodname[0]

    mov ah, 10

    int 21h

    lea dx, outenter[0]

    mov ah, 9

    int 21h

    mov cl, goodname[1]

    cmp cl, 0

    jz start

    call findgood

    cmp ch, -1

    jz printNotFound

    mov cl, auth[0]

    cmp cl, 1

    jz authOK

    jmp authNotOK

authOK:

    mov dx, bx

    mov ah, 9

    int 21h

    lea dx, outenter[0]

    mov ah, 9

    int 21h

    jmp start

authNotOK:

    mov al, [bx + 12]

    mov cl, 128

    mul cl

    push ax

    mov al, [bx + 14]

    mov cl, [bx + 11]

    mul cl

    mov cl, 10

    div cl

    mov ah, 0

    mov dx, ax

    pop ax

    div dl

    mov ah, 0

    jmp func4

printNotFound:

    lea dx, notFound[0]

    mov ah, 9

    int 21h

    jmp recommandGood

func4:

    cmp ax, 100

    jns printA

    cmp ax, 50

    jns printB

    cmp ax, 10

    jns printC

    jmp printF

printA:

    mov dl, 41H

    mov ah, 2

    int 21h

    jmp returnStart

printB:

    mov dl, 42H

    mov ah, 2

    int 21h

    jmp returnStart

printC:

    mov dl, 43H

    mov ah, 2

    int 21h

    jmp returnStart

printF:

    mov dl, 46H

    mov ah, 2

    int 21h

    jmp returnStart

returnStart:

    lea dx, outenter[0]

    mov ah, 9

    int 21h

    jmp start

Quit:

    mov ah, 4Ch

    int 21h

findgood proc

    mov bl, goodname[1]

    mov bh, 0

    mov si, bx

    lea bx, ga1[0]

    mov al, [bx + 10]

    mov ah, 0

    cmp ax, si

    jnz next1

    call compare

    cmp ch, 1

    jz foundgood

next1:

    lea bx, ga2[0]

    mov al, [bx + 10]

    mov ah, 0

    cmp ax, si

    jnz next2

    call compare

    cmp ch, 1

    jz foundgood

next2:

    mov ch, -1

    ret

foundgood:

    mov ch, bh

    ret

findgood endp

compare proc

comp:

    cmp al, 0

    jz found

    dec al

    mov ah, 0

    mov di, ax

    mov dl, goodname[2 + di]

    sub dl, [bx + di]

    jz comp

    mov ch, 0

    ret

found:

    mov ch, 1

    ret

compare endp

compareName proc

compName:

    cmp al, 0

    jz trueName

    dec al

    mov ah, 0

    mov di, ax

    mov dl, bname[di]

    sub dl, [bx + di]

    jz compName

    mov ch, 0

    ret

trueName:

    mov ch, 1

    ret

compareName endp

comparePass proc

compPass:

    cmp al, 0

    jz truePass

    dec al

    mov ah, 0

    mov di, ax

    mov dl, bpass[di]

    sub dl, [bx + di]

    jz compPass

    mov ch, 0

    ret

truePass:

    mov ch, 1

    ret

comparePass endp

code ends

    end start

### 3.2.4 实验步骤

使用visual studio code进行源代码的编辑，将文件命名为problem4.asm。在dosbox虚拟机中使用masm对其进行编译，然后使用link对其进行链接，得到可执行文件problem4.exe。然后可以使用td对其进行调试，或者直接输入文件名执行该文件。

### 3.2.5 实验记录与分析

1. 读取用户名和密码的操作

该操作有3种可能情况，分别为用户名和密码正确(如图1-9所示)、未输入用户名和密码(如图1-10所示)、输入q退出程序(如图1-11所示)。

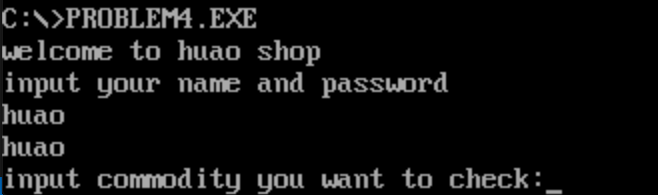


图1-9 用户名和密码输入正确

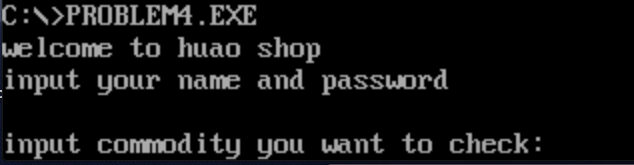


图1-10 未输入用户名和密码

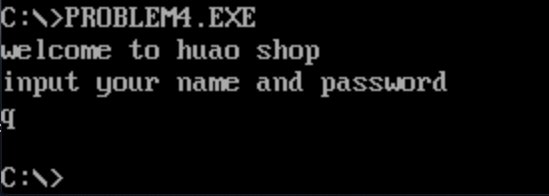


图1-11 输入q，退出程序

2. 输入待查询的商品名称

输入商品名称有3种情况。若不存在该商品，则程序输出“good not found”，并提示用户重新输入待查询的商品名称(如图1-12所示)；若存在该商品，则根据用户的类型执行不同的操作，若用户为老板，则程序输出该商品的名称(如图1-13所示)，否则，程序输出该商品的推荐度(如图1-14所示)，执行完上述操作后，程序重新提醒用户输入用户名和密码。

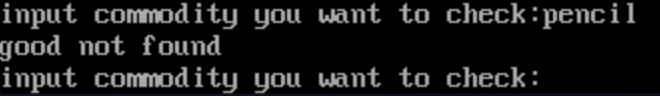


图1-12 未找到待查询商品信息

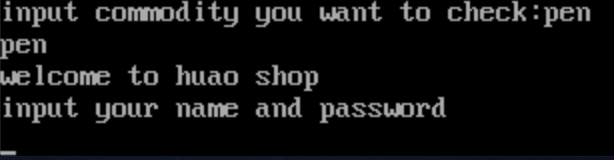


图1-13 老板查询商品

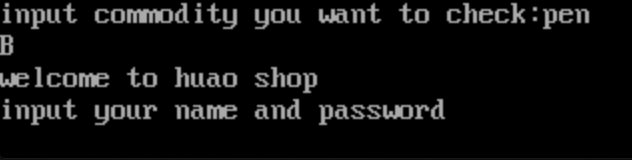


图1-14 顾客查询商品

# 4 总结和体会

本次实验是汇编语言的第一次实验。在本次实验中，我了解了汇编语言程序的生成和执行环境，对汇编语言的使用有了初步的认识。

通过本次实验，我学会了使用虚拟机在DOS操作系统下使用masm、link来编译和链接汇编语言源代码，得到相应的可执行文件，并使用td对生成的可执行文件进行调试。在使用td调试程序的过程中，通过观察程序执行过程中各寄存器以及标志位的变化，并将其与源代码进行对照，我对汇编语言的指令有了更直观的认识，特别是对于标志位的变化，通过直观的观察程序的执行过程，使我更加明确各标志位的含义。

另外，通过使用td的反汇编功能从可执行文件的不同位置开始反汇编，让我对计算机执行程序的过程有了更加深刻的理解。可执行文件在计算机中表现为一个二进制串，只有使用正确的方式对其进行解读才能够正确的执行该程序，本次实验让我认识到IP/EIP寄存器在程序执行过程中的重要意义，也让我对计算机程序的本质有了更深刻的了解。

任务4是一个规模较大的汇编程序，在程序中存在大量的分支程序，因此需要执行很多判断和分支跳转指令，在对这些分支进行设计和编写的过程中，我对汇编程序的执行过程的认识更加的深刻了，另外，本程序中有大量的数据存储在内存中，由于很多指令不能够对多个内存同时进行操作，而寄存器的数量有限，我在编写程序的过程中加强了如何有效的使用少量的寄存器完成相应的操作的能力，同时，在对内存的操作中，我更加深刻的认识到了各种寻址方式的使用方法。由于只有少数几个寄存器可以用于变址寻址和基址加变址寻址，在本次实验中，我使用过错误的寄存器用于寻址操作，导致了一些错误，通过修正这些错误，让我对这些寻址方式中寄存器的使用有了更加深刻的印象。