

汇编语言程序设计实验报告

目录

1	实验目的与要求.....	1
2	实验内容.....	1
3	实验过程.....	3
3.1	任务 1.....	3
3.1.1	实验步骤.....	3
3.1.2	实验记录与分析.....	4
3.2	任务 2.....	9
3.2.1	实现新功能的源程序.....	9
3.2.2	实验步骤.....	10
3.2.3	实验记录与分析.....	11
3.2	任务 3.....	14
3.3.1	源程序.....	14
3.3.2	实验步骤.....	14
3.3.3	实验记录与分析.....	15
3.4	任务 4.....	19
3.4.1	源程序.....	19
3.4.2	实验步骤.....	21
3.4.3	实验记录与分析.....	21
3.5	任务 5.....	25
3.5.1	设计思想及存储单元分配.....	25
3.5.2	流程图.....	27
3.5.3	源程序.....	29
3.5.4	实验步骤.....	34
3.5.5	实验记录与分析.....	35
4	总结与体会.....	42
	参考文献.....	44

汇编语言程序设计实验报告

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。

- (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。

(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。

汇编语言程序设计实验报告

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 XUEHA0 开头的存储区中，要求学号的存放以字符方式存放。

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
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

汇编语言程序设计实验报告

- (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)。

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. 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, +0110011B; MOV AL, +1011010B; 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。
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, 10H; MOV AL, -5H; SUB AH, AL; 观察标志位特点；
输入 MOV AH, 0FFH; MOV AL, -5H; SUB AH, AL; 观察标志位特点。
- 实验预测如表 3-1-1, 3-1-2 所示

表 3-1-1 任务 1 预测结果 1

Task	AH	SF	OF	CF	ZF
1add	8DH	1	1	0	0
1sub	0D9H	1	0	1	0

汇编语言程序设计实验报告

2add	7AH	0	1	1	0
2sub	34H	0	0	0	0
3add	08H	0	0	1	0
3sub	0C2H	1	1	1	0

表 3-1-2 任务 1 预测结果 2

Task	AH	SF	OF	CF	ZF
1	15H	0	0	1	0
2	04H	0	0	0	0

3.1.2 实验记录与分析

1. 实验环境条件：P3 1GHz，256M 内存；WINDOWS 10 下 DOSBox0.73；TD.EXE 5.0。
2. 输入指令 MOV AH, +0110011B; MOV AL, +1011010B; ADD AH, AL。执行三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。

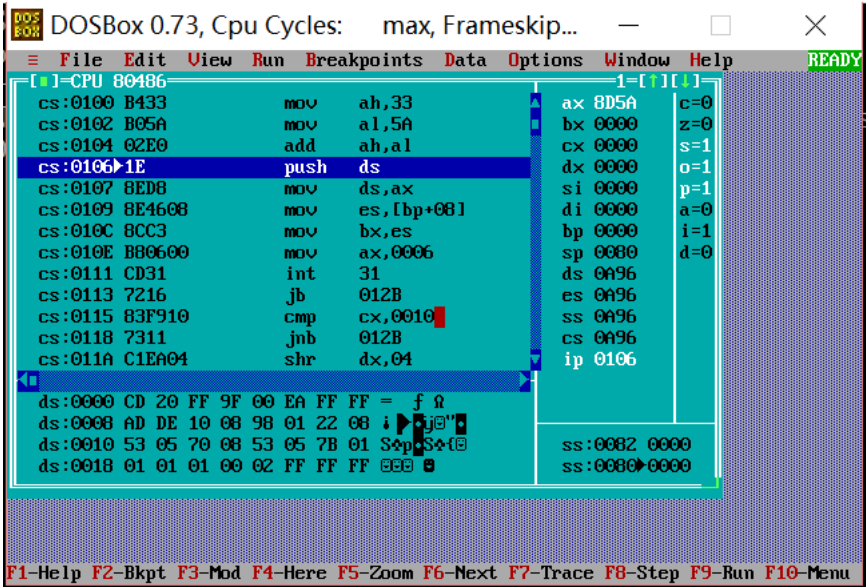


图 3.1.1 执行完测试语句后的状态

3. 输入指令 MOV AH, +0110011B; MOV AL, +1011010B; SUB AH, AL。执行三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（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。

汇编语言程序设计实验报告

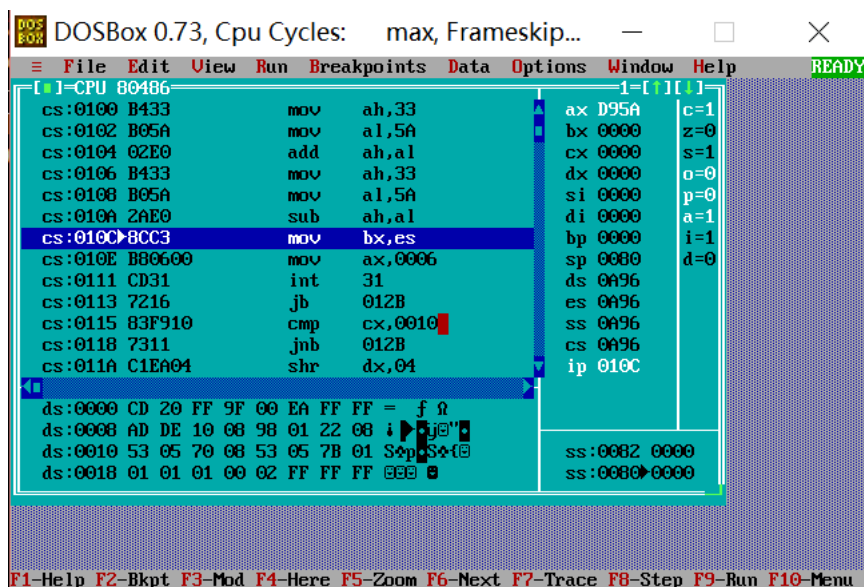


图 3. 1. 2 执行完测试语句后的状态

4. 输入指令 MOV AH, -0101001B; MOV AL, -1011101B; ADD AH, AL。执行三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 (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。

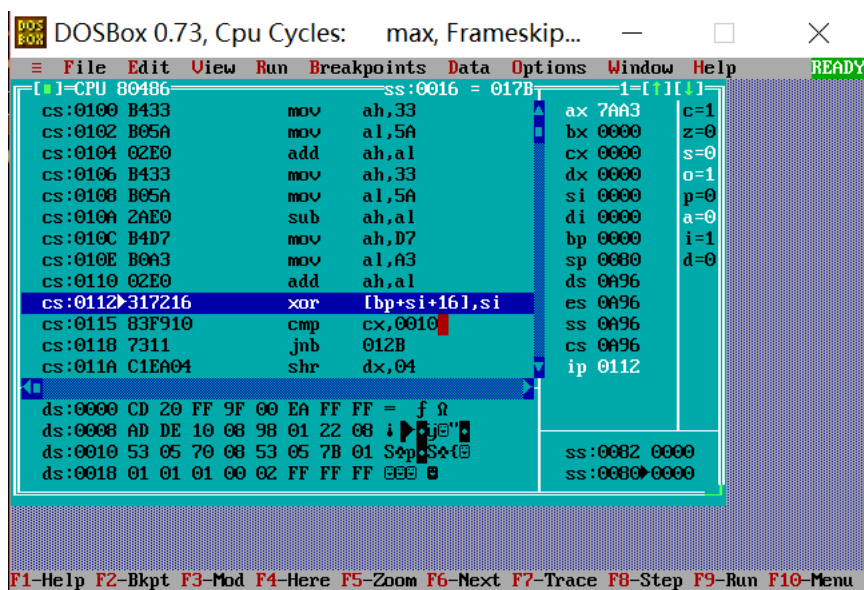


图 3. 1. 3 执行完测试语句后的状态

5. 输入指令 MOV AH, -0101001B; MOV AL, -1011101B; SUB AH, AL。执行三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 (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。

汇编语言程序设计实验报告

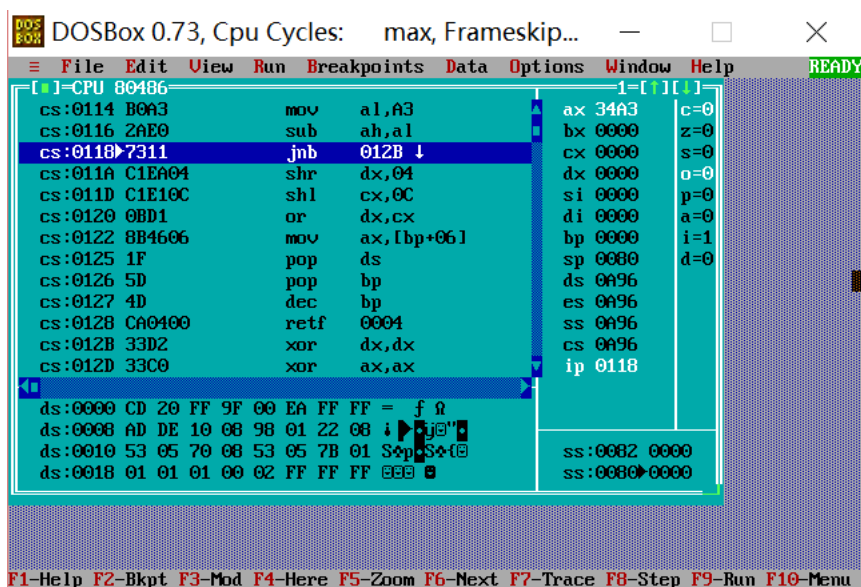


图 3. 1. 4 执行完测试语句后的状态

6. 输入指令 MOV AH, +1100101B; MOV AL, -1011101B; ADD AH, AL。执行三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 (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。

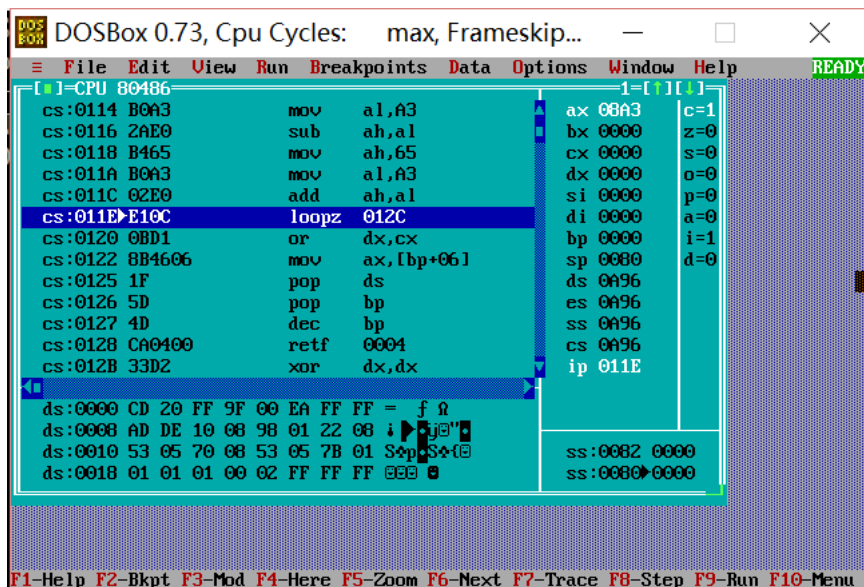


图 3. 1. 5 执行完测试语句后的状态

7. 输入指令 MOV AH, +1100101B; MOV AL, -1011101B; SUB AH, AL。执行三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。

汇编语言程序设计实验报告

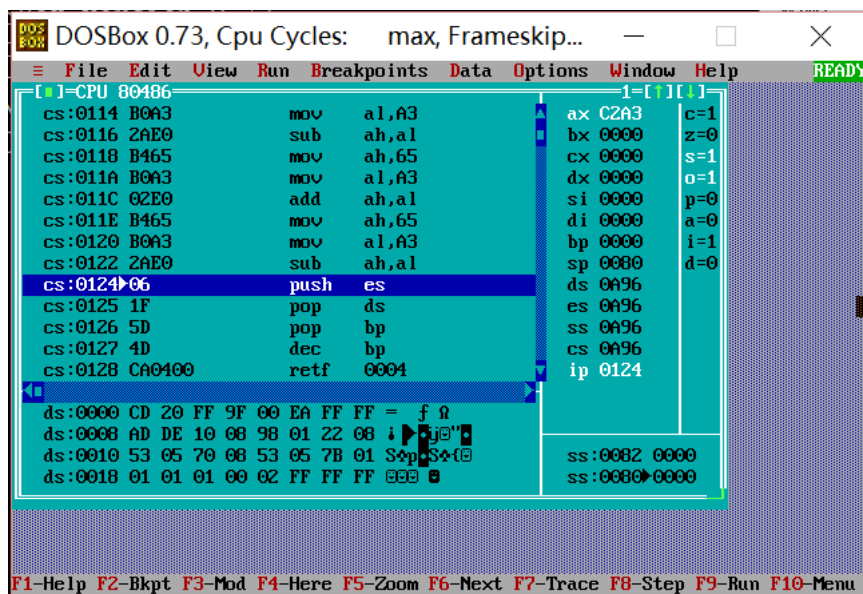


图 3.1.6 执行完测试语句后的状态

8. 输入指令 MOV AH,10H; MOV AL,-5H; SUB AH,AL。执行三条指令后的结果如图 3.1.7 所示。可以看出，计算结果在 AX 的高字节中（15H）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。

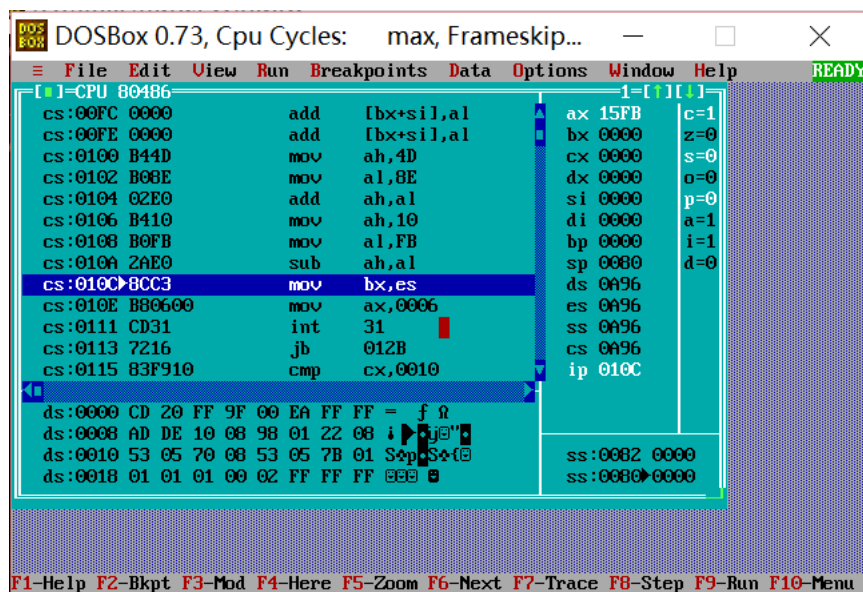


图 3.1.7 执行完测试语句后的状态

9. 输入指令 MOV AH,0FFH; MOV AL,-5H; SUB AH,AL。执行三条指令后的结果如图 3.1.8 所示。可以看出，计算结果在 AX 的高字节中（04H）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。

汇编语言程序设计实验报告

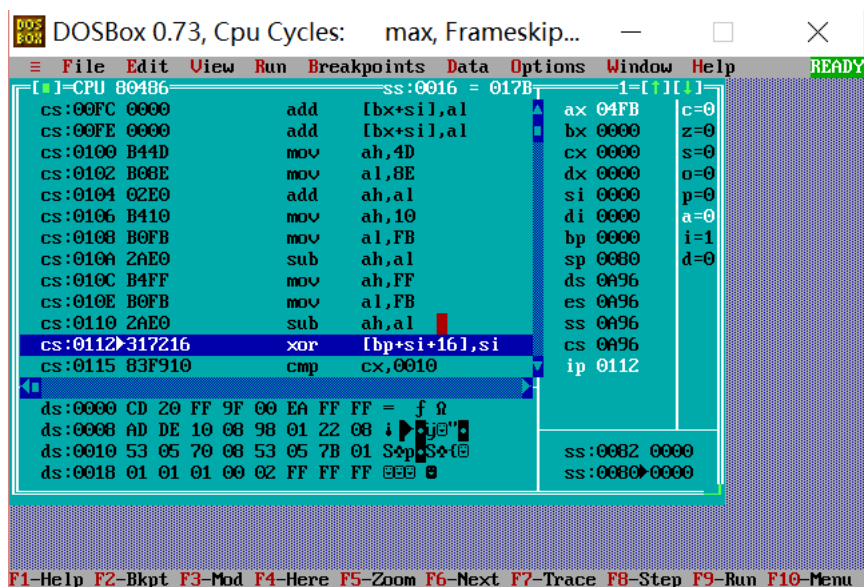


图 3.1.8 执行完测试语句后的状态

10. 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？

若将 A、B 视为有符号数，且 $A > B$ ，则可以通过标志位 $SF=0$ （一直是正值）， $CF=0$ （无进位且无意义）， $OF=0$ ， $ZF=0$ （结果不为 0）。

若将 A、B 视为无符号数，且 $A > B$ ，则标志位中 $CF=0$ （无借位）， $ZF=0$ ；如果 A 的最高位为 0，则计算后 $OF=0$ ， $SF=0$ ；如果 A 最高位为 1，在计算后最高位变为 0，则 $OF=1$ ， $SF=0$ ；如果 A 最高位在计算后仍为 1，则 $OF=0$ ， $SF=1$ 。（SF 的值无意义，OF 的值无意义）

11. 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？

使光标位于代码区，直接通过键盘输入指令，然后通过单步执行 F7 或 F8，执行指令。

（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？

在代码区输入若干条指令，只需将 CS: IP 设置到第一条指令即可，具体方法为右键点击指令，选择 new cs: ip。

（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？

要注意后面加上 h。

（4）输入指令 MOV AH, -128 并执行 查看 AH 寄存器的内容。

由下图可知，ah 内容为 80h。

汇编语言程序设计实验报告

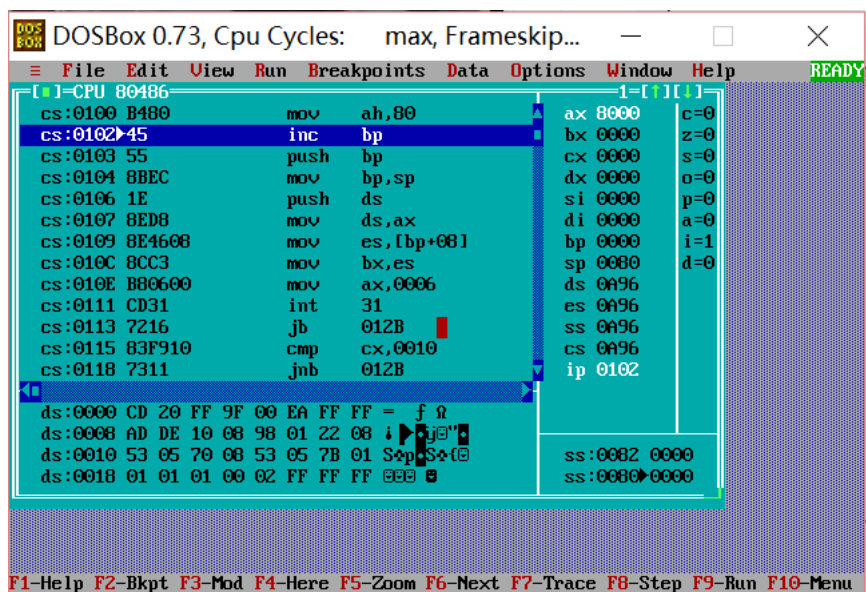


图 3.1.9 执行指定语句

(5) 任务 1 中的(1)~(4)

见实验步骤。

(6) 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？

在界面右边会显示寄存器区，右键点击会出现切换成 32 位寄存器的选项，也会出现修改寄存器值的选项。

(7) 执行一条指令后，如何查看标志寄存器的值？

界面右边会有显示标志寄存器的值。

(8) 经过 6，7 后，总结加减法对标志寄存器的影响？

加减法会影响所有标志寄存器的值，当有溢出时，OF 标志位会变为 1；当有进位或者借位时，CF 标志位会变为 1。最主要影响的标志位就是这两个。

3.2 任务 2

3.2.1 实现新功能的源程序

```
.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)
    buf5 db 0ah,0dh,'Press any key to begin!$'
data ends
code segment use16
    assume cs:code,ds:data,ss:stack
start:
```

汇编语言程序设计实验报告

```
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,buf5
mov ah,9
int 21h
mov ah,1
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
```

3.2.2 实验步骤

1. 准备上机实验环境。
2. 使用编辑程序 EDIT.EXE 录入源程序,存盘文件名为 code2.asm。使用 MASM 6.0 汇编源文件。即 MASM code2; 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。
3. 使用连接程序 LINK.EXE 将汇编生成的 code2.OBJ 文件连接成执行文件。即 LINK code2; 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 code2.EXE 文件。
4. 打开 td 进行单行调试,记录执行到 mov cx, 10 和 int 21h 之前的 bx, bp, si, di 各是多少。
5. 预计数据段开始 40 个字节的内容,然后运行程序并记录,观察程序运行结果是否与设想的一致。
6. 在 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。记录过程。

表 2-1 任务 2 预测结果 1

Task	BX	BP	SI	DI
Mov cx, 10 前	0014h	001eh	0000h	000ah
Int 21h 前	001eh	0028h	000ah	0014h

汇编语言程序设计实验报告

表 2-2 任务 2 预测结果 2

前 40 个字节内容
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

3.2.3 实验记录与分析

- 1. 实验环境条件：P3 1GHz，256M 内存；WINDOWS 10 下 DOSBox0.73；TD.EXE 5.0。
- 2. 使用 masm 编译汇编源程序，使用 link 连接生成 .exe 文件，进入 td 调试程序，在 mov cx, 10 处设置断点，运行程序，结果如图 3.2.1 所示，可以看出，寄存器的值（bx=0014h, bp=001eh, si=0000h, di=000ah）与事前预期的是一致的。

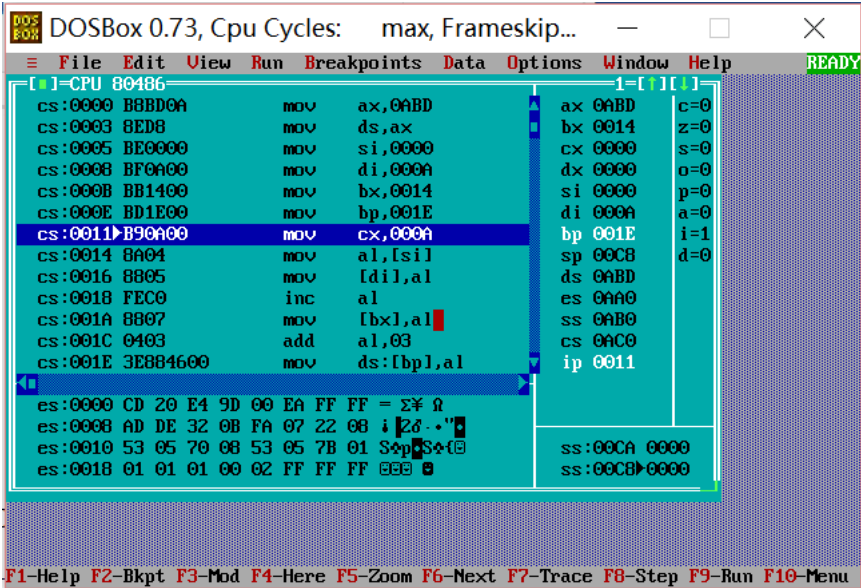


图 3.2.1 执行完测试语句后的状态

- 3. 在 int 21h 处设置断点，运行程序，结果如图 3.2.2 所示，可以看出，寄存器的值（bx=001eh, bp=0028h, si=000ah, di=0014h）与事前预期的是一致的。

汇编语言程序设计实验报告

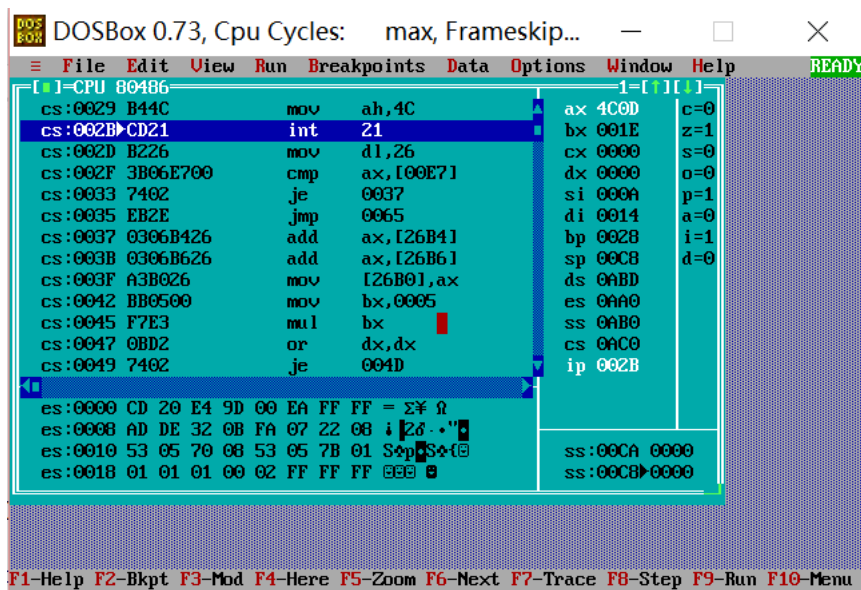


图 3.2.2 执行完测试语句后的状态

4. 在 int 21h 处设置断点，运行程序，数据段结果如图 3.2.3 所示，可以看出，数据段前 40 个字节的值与事前预期的是一致的。

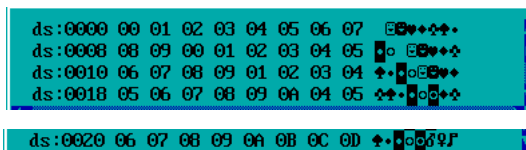


图 3.2.3 执行完测试语句后的状态

5. 编辑源程序，使其实现任务要求功能，在 mov al, [si] 处设置断点，使程序执行到此处，结果如图 3.2.4 所示，在屏幕上显示 Press any key to begin!，此时从键盘任意输入一个键，返回 td 调试界面，cs: ip 指向下一条语句，如图 3.2.5 所示。

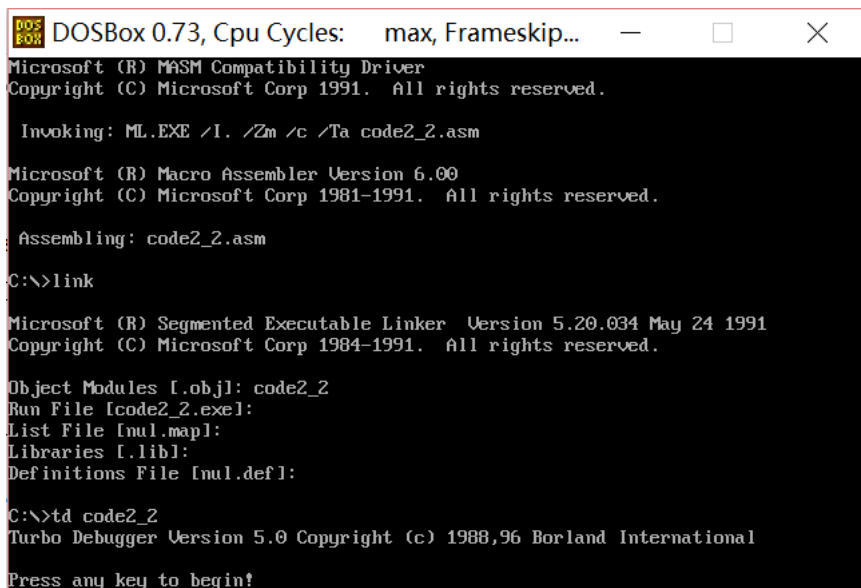


图 3.2.4 执行完测试语句后的状态

汇编语言程序设计实验报告

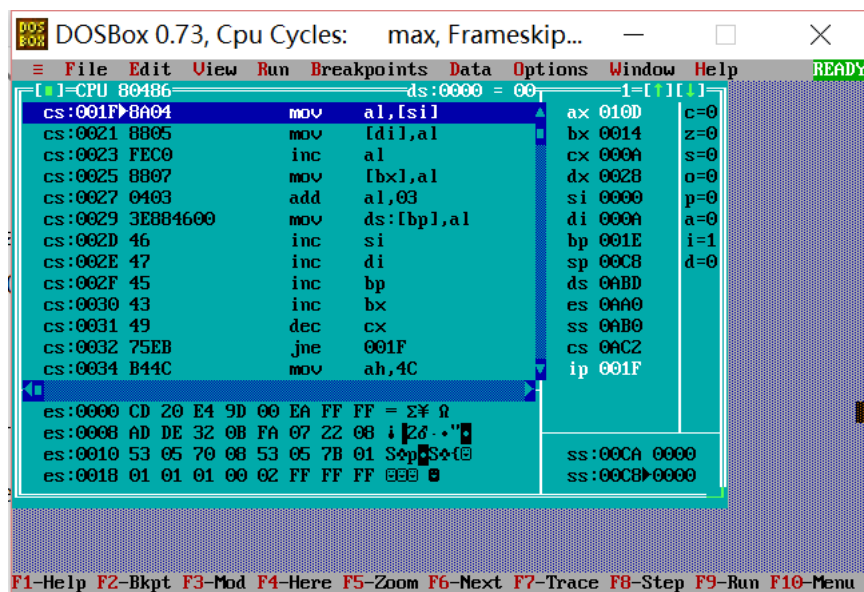


图 3.2.5 键盘输入任意键后的状态

6. 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 下输入 td+可执行程序名字，代码区在左上。机器码与汇编指令间的联系卸载实验步骤和总结中了。

(2) 如何设置断点并运行到断点？

将光标设置到要设置断点的语句处，按下 F2 即可设置断点，之后按下 F9 即可运行到断点处。

(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）

只需按下 F4：Go to cursor 即可。

(4) 如何单步执行一条指令？（多种方法）

按下 F7 或者 F8 可以断不执行一条指令。

(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？

至少有三种方法：(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。可以用这个方法查看其他段。

(6) 修改某个指定的存储单元的值，如任务 2 中的 BUF2。

可以在数据区直接修改。

(7) 如何查看堆栈？

堆栈区在右下角。

(8) 如何汇编一个汇编源程序并链接产生可执行代码？

Masm 用于汇编一个汇编源程序，link 用于连接产生可执行代码。

(9) 如何读懂源程序在汇编过程中产生的错误？

可以通过 Google 查询。

(10) 查看 BUF2 等变量在 TD 中的表示形式并总结

在实验记录中出现。

(11) 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。

汇编语言程序设计实验报告

在总结中已经提及。

(12) 单步执行循环体 2 遍，查看数据段的变化。

在实验过程中已经执行。

3.2 任务 3

3.3.1 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,[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
```

3.3.2 实验步骤

1. 准备上机实验环境。

2. 使用编辑程序 EDIT.EXE 录入源程序，存盘文件名为 code3.asm。使用 MASM 6.0 汇编源文件。即 MASM code3；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

汇编语言程序设计实验报告

- 错为止。
3. 使用连接程序 LINK.EXE 将汇编生成的 code3.OBJ 文件连接成执行文件。即 LINK code3; 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 code3.EXE 文件。
4. 打开 TD 进行单行调试, 预计数据段开始 40 个字节的内容, 然后运行程序并记录, 观察程序运行结果是否与设想的一致。
5. 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。
6. 观察连续存放的二进制串在反汇编成汇编语句时, 从不同字节位置开始反汇编的结果, 并分析。

表 3-1 任务 3 预测结果

前 40 个字节内容
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

3.3.3 实验记录与分析

- 1.实验环境条件: P3 1GHz, 256M 内存; WINDOWS 10 下 DOSBox0.73; TD.EXE 5.0。
- 2.使用 masm 编译汇编源程序, 使用 link 连接生成.exe 文件, 进入 td 调试程序, 在 int 21h 处设置断点, 运行程序, 数据段结果如图 3.3.1 所示, 可以看出, 数据段前 40 个字节的值与事前预期的是一致的。

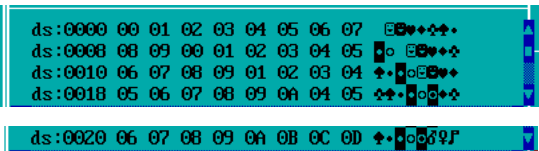
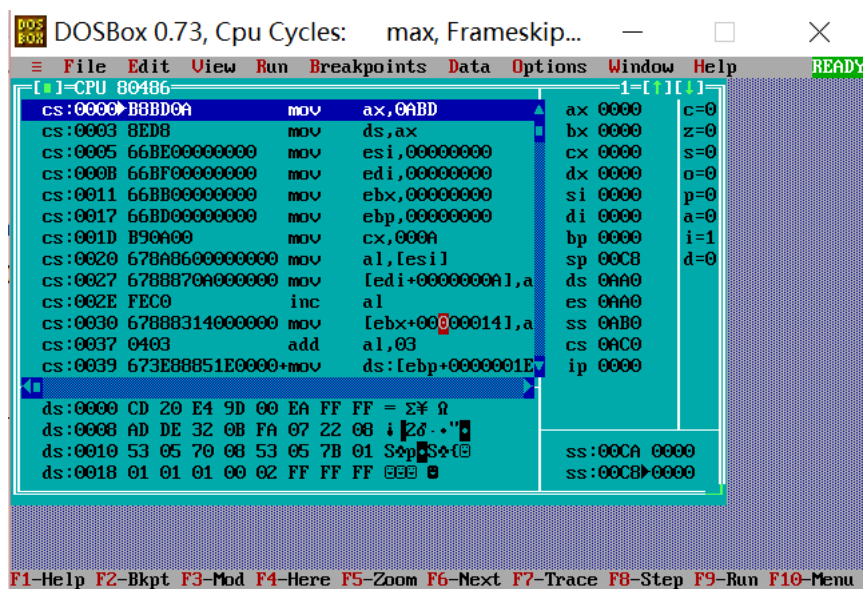


图 3.3.1 执行完测试语句后的状态

- 3.将 TD 中提供的反汇编语句与自己编写的源程序语句进行对照, 有如下发现:
- (1) 操作指令如 mov, add, inc 等以及寄存器名称都与源程序语句相同。
 - (2) 机器指令都是 16 进制数, 不同指令位数不同。
 - (3) 省略了 mov ax,data 语句。
 - (4) 反汇编语句将立即数转换成相应的 16 进制数, 位数则与寄存器位数对应或者自己指定
 - (5) 反汇编语句将变址寻址转换成[R+idata], 其中 R 为寄存器名称, idata 为常量。如 mov al,[esi]+buf1, 直接将 buf1 的偏移地址表示出来。
 - (6) 将 lopa 转换成偏移地址, 如 jnz lopa 在反汇编语句中为 jne 0020。
 - (7) 反汇编语句中只有代码段显示出来了。

汇编语言程序设计实验报告



DOSBox 0.73, Cpu Cycles: max, Frameskip...

File Edit View Run Breakpoints Data Options Window Help

1-CPU 80486

Address	Disassembly	Comment
cs:0000 B8BD0A	mov ax,0ABD	
cs:0003 8ED8	mov ds,ax	
cs:0005 66BE00000000	mov esi,00000000	
cs:000B 66BF00000000	mov edi,00000000	
cs:0011 66BB00000000	mov ebx,00000000	
cs:0017 66BD00000000	mov ebp,00000000	
cs:001D B90A00	mov cx,000A	
cs:0020 678A8600000000	mov al,[esi]	
cs:0027 6788870A000000	mov [edi+0000000A],a	
cs:002E FEC0	inc al	
cs:0030 67888314000000	mov [ebx+00000014],a	
cs:0037 0403	add al,03	
cs:0039 673E88851E0000+mov	ds:[ebp+0000001E]	

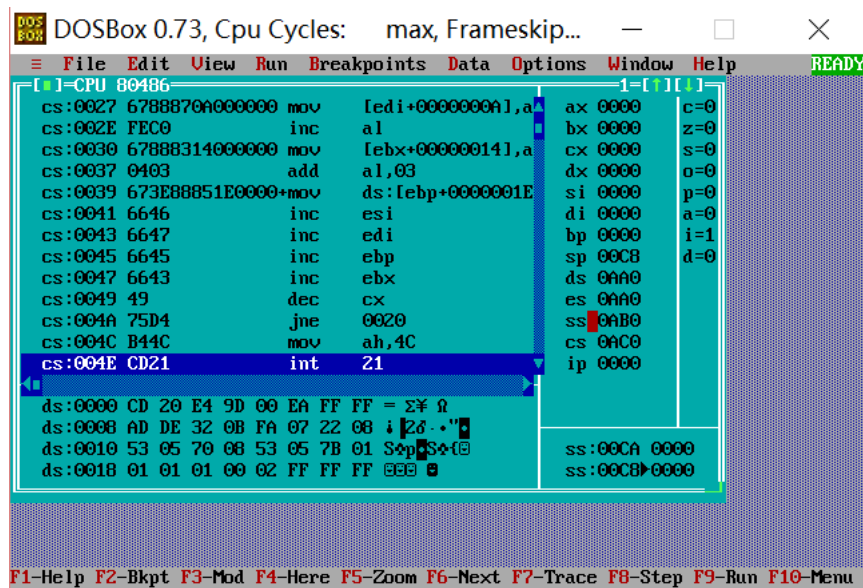
ax 0000 c=0
bx 0000 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 00CB d=0
ds 0AA0
es 0AA0
ss 0AB0
cs 0AC0
ip 0000

ds:0000 CD 20 E4 9D 00 EA FF FF = Σ% Ω
ds:0008 AD DE 32 0B FA 07 22 08 i 2d . . %
ds:0010 53 05 70 08 53 05 7B 01 Sep %S %t %
ds:0018 01 01 01 00 02 FF FF FF 88B %

ss:00CA 0000
ss:00CB 0000

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

图 3.3.2 反汇编语句



DOSBox 0.73, Cpu Cycles: max, Frameskip...

File Edit View Run Breakpoints Data Options Window Help

1-CPU 80486

Address	Disassembly	Comment
cs:0027 6788870A000000	mov [edi+0000000A],a	
cs:002E FEC0	inc al	
cs:0030 67888314000000	mov [ebx+00000014],a	
cs:0037 0403	add al,03	
cs:0039 673E88851E0000+mov	ds:[ebp+0000001E]	
cs:0041 6646	inc esi	
cs:0043 6647	inc edi	
cs:0045 6645	inc ebp	
cs:0047 6643	inc ebx	
cs:0049 49	dec cx	
cs:004A 75D4	jne 0020	
cs:004C B44C	mov ah,4C	
cs:004E CD21	int 21	

ax 0000 c=0
bx 0000 z=0
cx 0000 s=0
dx 0000 o=0
si 0000 p=0
di 0000 a=0
bp 0000 i=1
sp 00CB d=0
ds 0AA0
es 0AA0
ss 0AB0
cs 0AC0
ip 0000

ds:0000 CD 20 E4 9D 00 EA FF FF = Σ% Ω
ds:0008 AD DE 32 0B FA 07 22 08 i 2d . . %
ds:0010 53 05 70 08 53 05 7B 01 Sep %S %t %
ds:0018 01 01 01 00 02 FF FF FF 88B %

ss:00CA 0000
ss:00CB 0000

F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu

图 3.3.3 反汇编语句

汇编语言程序设计实验报告

```
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
```

图 3.3.4 源程序

4.在任务2中将TD中提供的反汇编语句与自己编写的源程序语句进行对照,有如下发现:

(1)反汇编语句将 `lea dx, buf5` 语句转换成 `mov dx, 0028`.这也是将 `buf5` 直接变成偏移地址,同时将 `lea` 指令变成了 `mov` 指令。

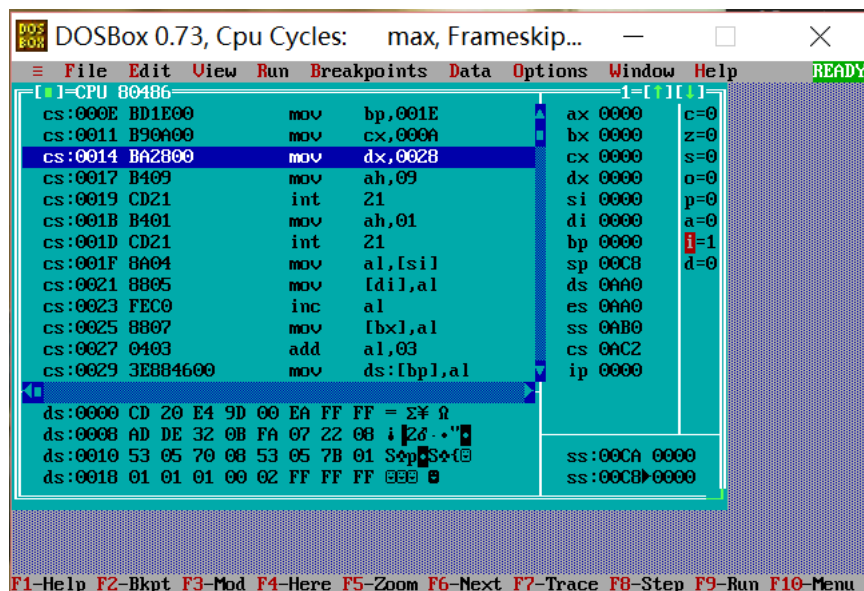


图 3.3.5 反汇编语句

汇编语言程序设计实验报告

```
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,buf5
    mov ah,9
    int 21h
    mov ah,1
    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
```

图 3.3.6 源程序

5.观察连续存放的二进制串在反汇编成汇编语句时，从不同字节位置开始反汇编的结果如下图所示。

从不同字节位置开始反汇编，其实是设置了 cs: ip 的值，汇编程序通过 cs: ip 的位置来控制代码段的运行，图中 ip 值变为 0020h。

汇编语言程序设计实验报告

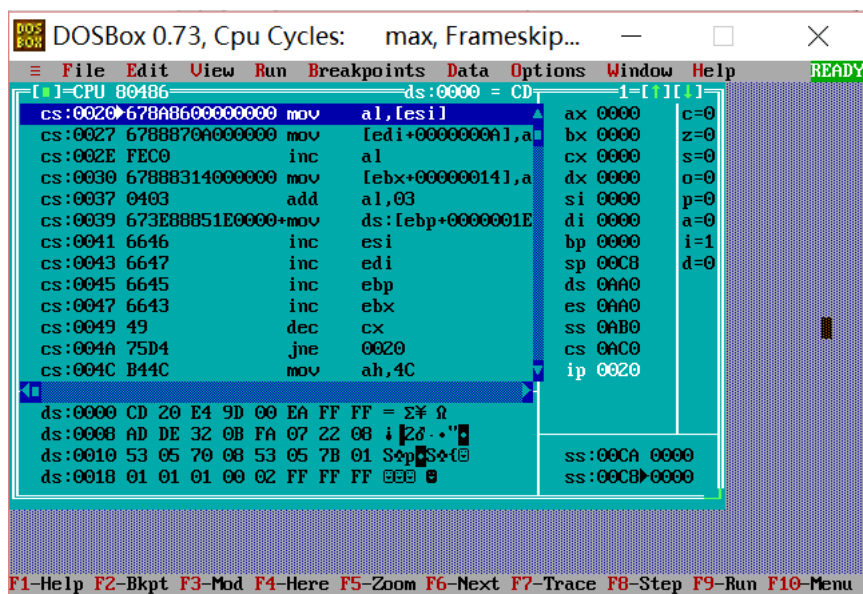


图 3.3.7 设置新的 cs: ip

6. 思考题已经在任务 2 中写完了。

3.4 任务 4

3.4.1 源程序

```
.386
stack segment use16 stack
    db 200 dup(0)
stack ends
data segment use16
    id db '5393'
    xuehao db 4 dup(0)
data ends
code segment use16
    assume cs:code, ds:data, ss:stack

start: mov ax, data
        mov ds, ax
        mov si, offset id
        mov di, offset xuehao
        mov cx, 4
s:      mov al, [si]                ;寄存器间接寻址
        mov [di], al              ;寄存器间接寻址
        inc si
        inc di
        loop s
        mov ah, 4ch
        int 21h
code    ends
        end start
```

汇编语言程序设计实验报告

```
.386
stack segment usel6 stack
    db 200 dup(0)
stack ends
data segment usel6
    id db '5393'
    xuehao db 4 dup(0)
data ends
code segment usel6
    assume cs:code, ds:data, ss:stack

start: mov ax, data
        mov ds, ax
        mov si, 0
        mov cx, 4
s:      mov al, id[si]           ;变址寻址
        mov xuehao[si], al     ;变址寻址
        inc si
        loop s
        mov ah, 4ch
        int 21h
code ends
end start
```

```
.386
stack segment usel6 stack
    db 200 dup(0)
stack ends
data segment usel6
    id db '5393'
    xuehao db 4 dup(0)
data ends
code segment usel6
    assume cs:code, ds:data, ss:stack
```

```
start: mov ax, data
        mov ds, ax
        lea bx, id
        lea bp, xuehao
        mov si, 0
        mov cx, 4
s:      mov al, [bx][si]        ;基址加变址寻址
        mov ds:[bp][si], al   ;基址加变址寻址
        add si, 1
        loop s
        mov ah, 4ch
        int 21h
code ends
end start
```

```
.386
stack segment usel6 stack
    db 200 dup(0)
stack ends
data segment usel6
```

汇编语言程序设计实验报告

```
xuehao db 4 dup(0)
data    ends
code    segment usel6
        assume cs:code,ds:data,ss:stack

start:  mov ax,data
        mov ds,ax
        mov si,offset xuehao
        mov byte ptr [si],35h      ;立即寻址
        inc si
        mov byte ptr [si],33h      ;立即寻址
        inc si
        mov byte ptr [si],39h      ;立即寻址
        inc si
        mov byte ptr [si],33h      ;立即寻址
        inc si
        mov ah,4ch
        int 21h
code    ends
        end start
```

3.4.2 实验步骤

1. 准备上机实验环境。
2. 使用编辑程序 EDIT.EXE 录入源程序，存盘文件名为 code4_1.asm, code4_2.asm, code4_3.asm, code4_4.asm 使用 MASM 6.0 汇编源文件。即 MASM code4_1 等；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。
3. 使用连接程序 LINK.EXE 将汇编生成的 code4_1.OBJ 等文件连接成执行文件。即 LINK code4_1 等；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 code4_1.EXE 等文件。
4. 使用 TD.EXE 观察 code4_1, code4_2, code4_3, code4_4 的执行情况，重点观察反汇编语句中的寻址方式的表达以及数据段里的变化情况。

3.4.3 实验记录与分析

1. 实验环境条件：P3 1GHz，256M 内存；WINDOWS 10 下 DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0；LINK.EXE 5.2；TD.EXE 5.0。
2. 使用 masm 编译汇编源程序，使用 link 连接生成.exe 文件，进入 td 调试程序，单步执行程序，数据段初始状态如图 3.4.1 所示，可以看出，数据段前 4 个字节为 id 所储存的学号；在执行到 mov ah, 4ch 时，数据段第 5-8 个字节变为 35, 33, 39, 33 即学号，如图 3.4.2 所示，完成了将学号后四位存储到以 xuehao 为开头的存储区的功能。

汇编语言程序设计实验报告

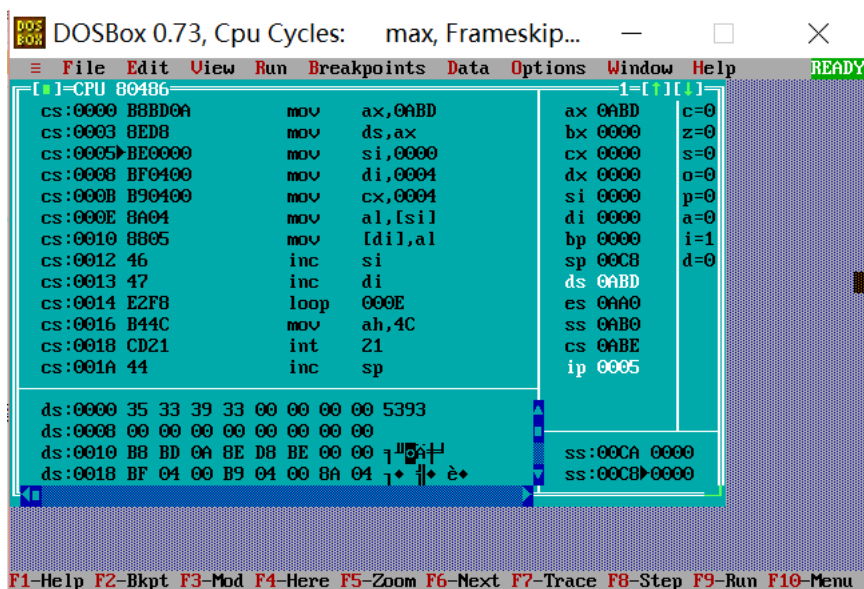


图 3. 4. 1 执行完测试语句后的状态

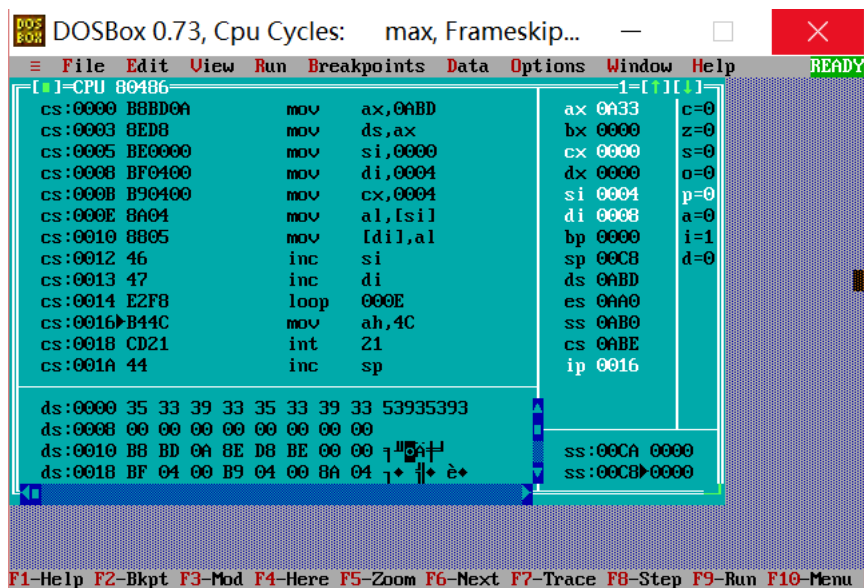


图 3. 4. 2 执行完测试语句后的状态

3.使用 masm 编译汇编源程序，使用 link 连接生成.exe 文件，进入 td 调试程序，单步执行程序，数据段初始状态如图 3.4.3 所示，可以看出，数据段前 4 个字节为 id 所储存的学号；在执行到 mov ah, 4ch 时，数据段第 5-8 个字节变为 35, 33, 39, 33 即学号，如图 3.4.4 所示，完成了将学号后四位存储到以 xuehao 为开头的存储区的功能。

汇编语言程序设计实验报告

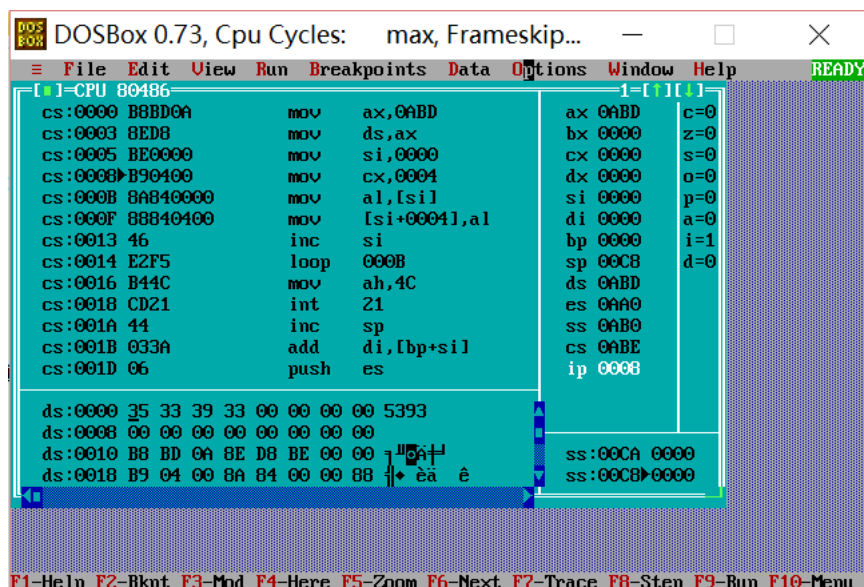


图 3. 4. 3 执行完测试语句后的状态

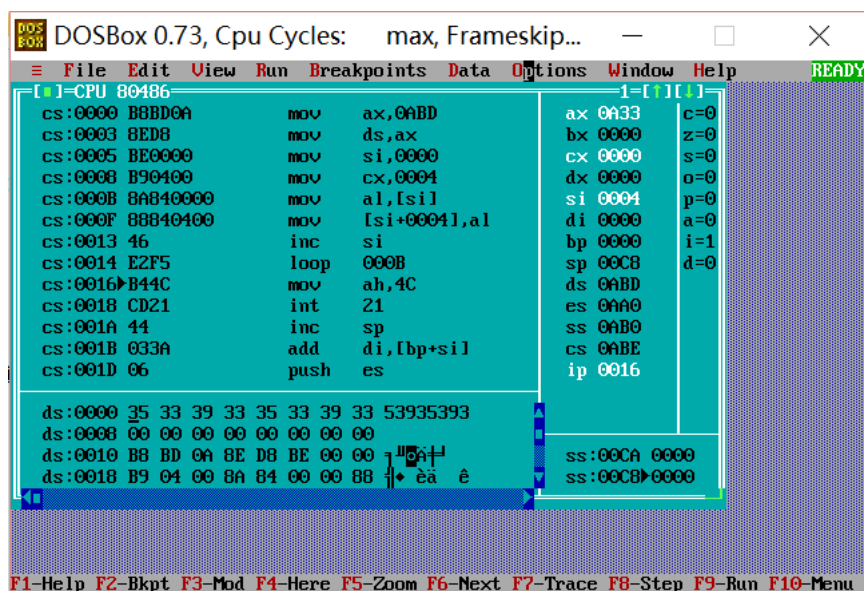


图 3. 4. 4 执行完测试语句后的状态

4.使用 masm 编译汇编源程序，使用 link 连接生成.exe 文件，进入 td 调试程序，单步执行程序，数据段初始状态如图 3.4.5 所示，可以看出，数据段前 4 个字节为 id 所储存的学号；在执行到 mov ah, 4ch 时，数据段第 5-8 个字节变为 35, 33, 39, 33 即学号，如图 3.4.6 所示，完成了将学号后四位存储到以 xuehao 为开头的存储区的功能。

汇编语言程序设计实验报告

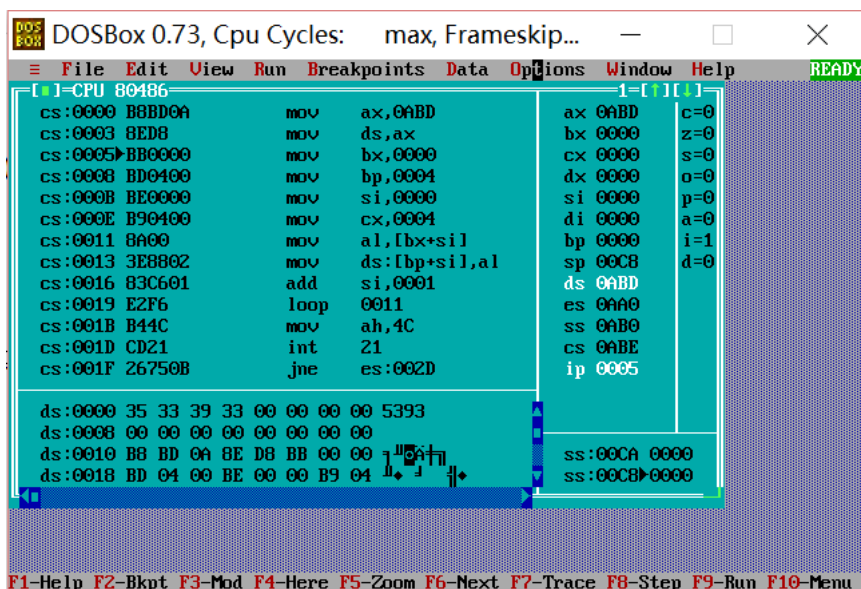


图 3.4.5 执行完测试语句后的状态

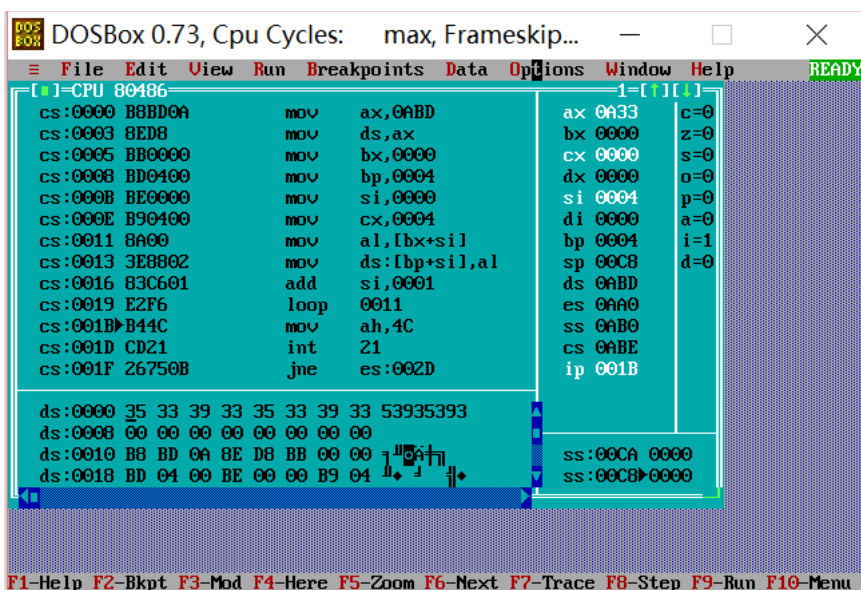


图 3.4.6 执行完测试语句后的状态

5.使用 masm 编译汇编源程序，使用 link 连接生成.exe 文件，进入 td 调试程序，单步执行程序，数据段初始状态如图 3.4.7 所示，可以看出，数据段前 4 个字节为 0；在执行到 mov ah, 4ch 时，数据段前 4 个字节变为 35, 33, 39, 33 即学号，如图 3.4.8 所示，完成了将学号后四位存储到以 xuehao 为开头的存储区的功能。

汇编语言程序设计实验报告

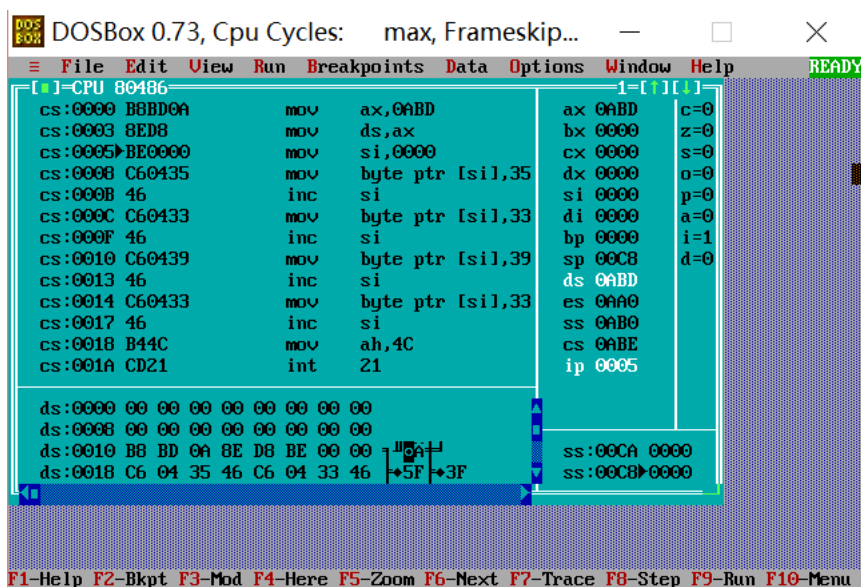


图 3. 4. 7 执行完测试语句后的状态

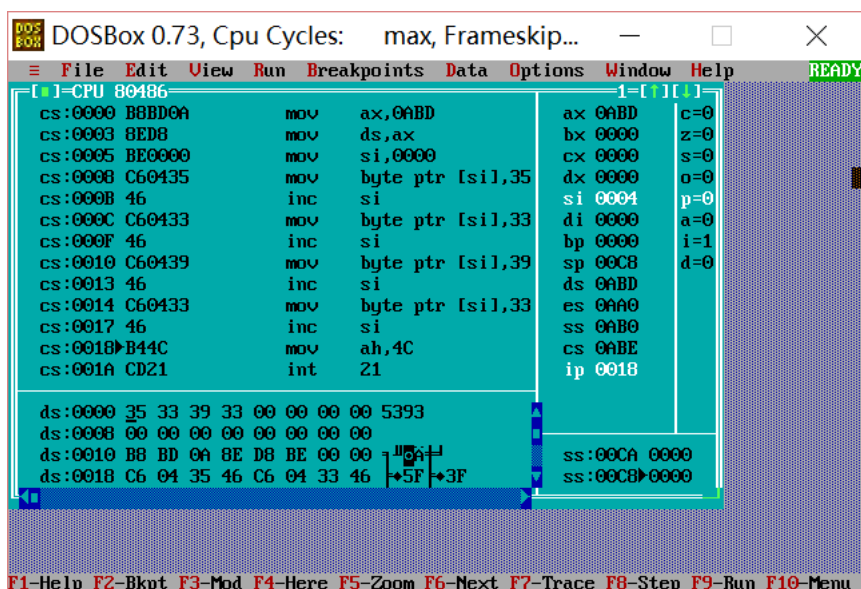


图 3. 4. 8 执行完测试语句后的状态

3.5 任务 5

3.5.1 设计思想及存储单元分配

提示用户输入姓名和密码使用 9 号 dos 系统功能调用，输出相应的字符串。并使用 10 号 dos 调用，分别输入姓名，密码。根据输入的姓名情况判定该执行哪些功能。

登录信息认证主要就是字符串的匹配，使用循环程序结构，比较姓名是否正确，若正确则比较密码，两者任有其一不对则提示登录失败。若登录成功，则可以有特殊权限。

计算指定商品的利润率，主要就是先匹配商品名，这也是字符串的比较。找到商品名后，则根据公式，利用乘除指令，算出利润率，同理在 shop2 里也这样操作，两次计算结果取平均值。

根据计算出的利润率，和指定值进行比较，输出对应的等级。

汇编语言程序设计实验报告

1. 存储单元分配

Tip1-5: 对应的提示语句存储区。

In_name: 输入姓名的存储区。

In_pwd: 输入密码的存储区。

In_itemname: 输入商品的名称存储区。

itemfind: 判断商品是否找到的标志存储区。

Auth: 判断登录状态标志存储区。

Apr: 利润率的存储区。

Mname: 老板姓名存储区。

Mpass: 老板密码存储区。

N: 商品数量的存储区。

S1: 网店 1 名称存储区。

Ga1: 网店 1 商品 1 存储区, 分别存储姓名, 进货价, 销售价, 进货数量, 销售数量, 利润率

Ga2: 商品 2 信息存储区。

Gan: 商品 n 信息存储区。

S2: 网店 2 名称存储区。

Gb1: 网店 2 商品 1 存储区, 分别存储姓名, 进货价, 销售价, 进货数量, 销售数量, 利润率

Gb2: 商品 2 信息存储区。

Gbn: 商品 n 信息存储区。

Stack: 分配栈空间。

2. 寄存器分配

Dx 存放字符串偏移首地址

Bx 存放内存单元格的内容

Dx, di 分别存放输入字符串与程序定义字符串, 用于比较登录信息是否正确
在功能 3 的乘除功能中

Ax 放一个乘数, bx 放一个乘数, 结果低位放在 ax 中, 高位放在 dx 中

Al 放一个乘数, bl 放一个乘数, 结果放在 ax 中

B1 放除数, 被除数默认在 ax 中, 结果商在 al 中

Bx 放除数, 被除数放在 ax, dx 中, 结果商放在 ax 中

汇编语言程序设计实验报告

3.5.2 流程图

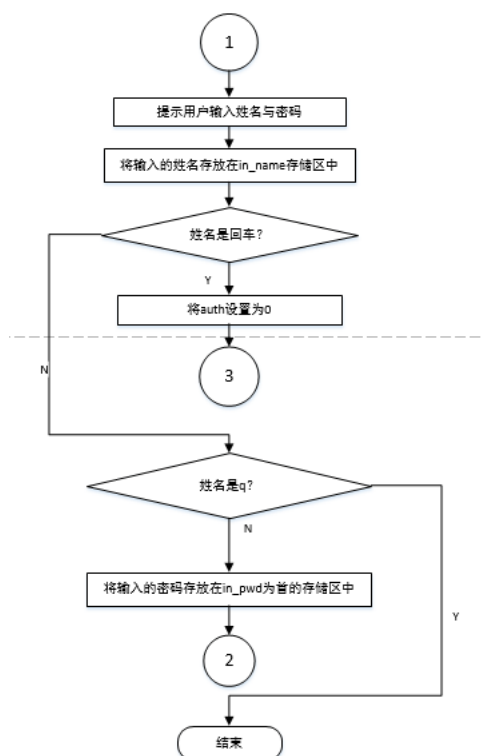
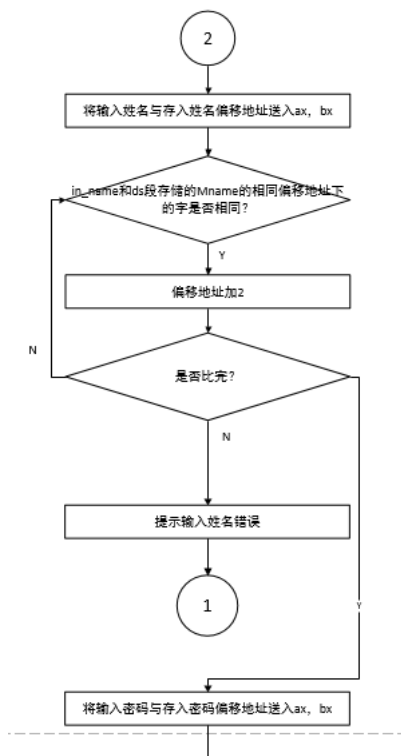


图 3.5.1 功能 1 流程图



汇编语言程序设计实验报告

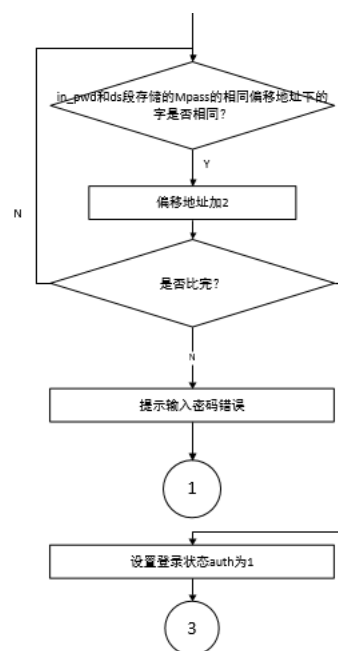


图 3.5.2 功能 2 流程图

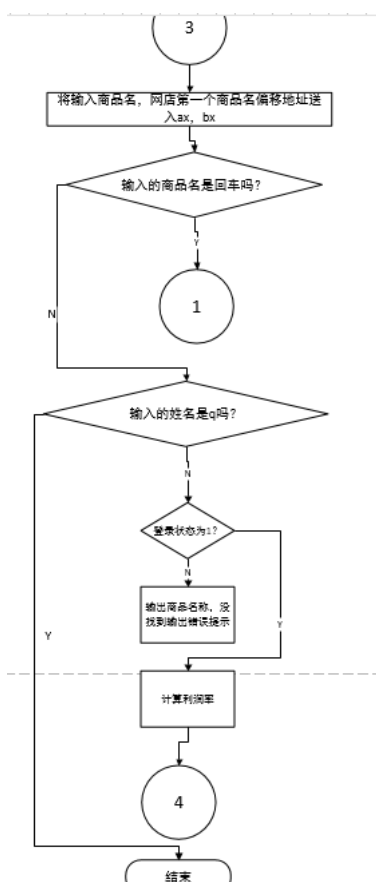


图 3.5.3 功能 3 流程图

汇编语言程序设计实验报告

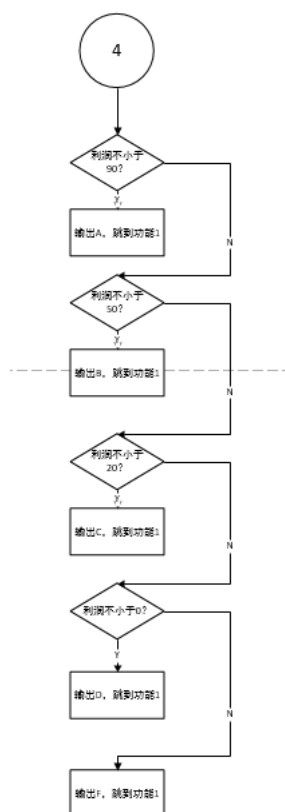


图 3.5.4 功能 4 流程图

3.5.3 源程序

```
. 386
assume cs:code,ds:data,ss:stack

data segment use16
    tip1 db 0ah,0dh,'Please input your name: ','$'
    tip2 db 0ah,0dh,'Please input your code: ','$'
    tip3 db 0ah,0dh,'Failed to log in!','$'
    tip4 db 0ah,0dh,'Please input the item name: ','$'
    tip5 db 0ah,0dh,'The profit ranks: ','$'
    tip6 db 0ah,0dh,'The name of the item is: ','$'
    tip7 db 0ah,0dh,'Cannot find the item!','$'
    in_name db 10
            db 0
            db 10 dup(0)
    in_pwd db 6
            db 0
            db 6 dup(0)
    in_itemname db 10 ;input itemname
                db 0
                db 10 dup(0)
    itemfind db 0 ;flag about whether found item or not
    auth db ? ;flag about the access
    apr1 dw ?
    apr2 dw ?
```

汇编语言程序设计实验报告

```
apr      dw ?                      ;profit

Mname    db 'chao fan',0,0        ;manager name
Mpass    db 'test',0,0           ;password
n        equ 30

s1        db 'shop1',0            ;information about shop1
ga1       db 'pen',7 dup(0)
          dw 35,56,70,25,?
ga2       db 'book',6 dup(0)
          dw 12,30,25,5,?
gan       db n-2 dup('temp-value',15,0,20,0,30,0,2,0,?,?)

s2        db 'shop2',0            ;information about shop2
gb1       db 'pen',7 dup(0)
          dw 35,50,30,24,?
gb2       db 'book',6 dup(0)
          dw 12,28,20,15,?
gbn       db n-2 dup('temp-value',15,0,20,0,30,0,2,0,?,?)
data      ends

stack     segment use16 stack
          db 200 dup(0)
stack     ends

code      segment use16
f1:       mov byte ptr itemfind,0
          mov byte ptr auth,0 ;reset auth
          mov ax,data
          mov ds,ax
          lea dx,tip1
          mov ah,9
          int 21h
          lea dx,in_name ;input name
          mov ah,0ah
          int 21h
          lea dx,tip2
          mov ah,9
          int 21h
          mov bx,offset in_name+2
          mov bx,[bx]
          mov bh,0
          mov ax,0dh ;to see whether name is 0dh or not
          cmp bx,ax
          je s ;jump to s
          mov bl,in_name+1
          mov bh,0
          mov byte ptr in_name+2[bx],0
          mov ax,'q'
          mov bx,offset in_name+2
          mov bx,[bx]
          mov bh,0
          cmp ax,bx ;to see whether name is 'q' or not
          je exit
          jmp short ok
s:        mov bx,offset auth
```

汇编语言程序设计实验报告

```
    mov bx,0
    jmp far ptr f3
ok:   lea dx,in_pwd ;input password
    mov ah,0ah
    int 21h
    mov bl,in_pwd+1
    mov bh,0
    mov byte ptr in_pwd+2[bx],0 ;put the next bit to 0
    jmp short f2
exit: mov ah,4ch
    int 21h

f2:   mov ax,offset in_name+2
    mov bx,offset Mname
    mov cx,5
cmpname:
    push bx
    mov bx,ax
    mov dx,[bx]
    pop bx
    mov di,[bx]
    cmp dx,di ;compare by word
    jne incorrect
    add ax,2
    add bx,2
    loop cmpname
    mov ax,offset in_pwd+2
    mov bx,offset Mpass
    mov cx,3
cmppass:
    push bx
    mov bx,ax
    mov dx,[bx]
    pop bx
    mov di,[bx]
    cmp dx,di ;compare by word
    jne incorrect
    add ax,2
    add bx,2
    loop cmppass
    mov byte ptr auth,1
    jmp short f3
incorrect:
    lea dx,tip3
    mov ah,9
    int 21h
    jmp far ptr f1

f3:   lea dx,tip4
    mov ah,9
    int 21h
    lea dx,in_itemname                ;输入
    mov ah,0ah
    int 21h
    mov bl,in_itemname+1              ;后一位置0
```

汇编语言程序设计实验报告

```
    mov bh, 0
    mov byte ptr in_itemname+2[bx], 0
    mov bx, offset in_itemname+2          ;检测商品名是否是回车
    mov bx, [bx]
    mov bh, 0
    mov ax, 0dh
    cmp bx, ax
    je f1
    mov ax, offset in_itemname+2          ;在shop1里查找商品
    mov bx, offset gal
    mov cx, 30 ;循环30次
cmpitem:
    push cx
    push bx
    mov cx, 5
cmpitemname:
    push bx
    mov bx, ax
    mov dx, [bx]
    pop bx
    mov di, [bx]
    cmp dx, di ;按字比较
    jne incorrectitem
    add ax, 2
    add bx, 2
    loop cmpitemname
    mov byte ptr itemfind, 1 ;如果找到, 状态量置1
    jmp short ok3
incorrectitem:
    pop bx
    add bx, 20 ;下一商品偏移地址
    pop cx
    loop cmpitem
ok3:  mov bx, offset itemfind ;是否找到
    mov bx, [bx]
    mov bh, 0
    cmp bx, 0
    je f3_5
    mov bx, offset auth
    mov bx, [bx]
    mov bh, 0
    cmp bx, 0
    je f3_4

f3_3: pop si ;取出商品偏移首地址
    mov al, 12[si]
    mov bl, 16[si]
    imul bl
    push ax ;销售价乘以销售数量
    mov al, 10[si]
    mov bl, 14[si]
    imul bl ;进货价乘以进货数量
    pop di
    push ax
    sub di, ax
    mov ax, di
    mov bx, 100
```

汇编语言程序设计实验报告

```
    imul bx
    pop di
    idiv di          ;利润计算，商放在ax里
    mov word ptr apr1, ax

    mov ax, offset in_itemname+2      ;在shop2里查找商品
    mov bx, offset gbl
    mov cx, 30
cmpitem1:
    push cx
    push bx
    mov cx, 5
cmpitemname1:
    push bx
    mov bx, ax
    mov dx, [bx]
    pop bx
    mov di, [bx]
    cmp dx, di ;按字比较
    jne incorrectitem1
    add ax, 2
    add bx, 2
    loop cmpitemname1
    jmp short ok33
incorrectitem1:
    pop bx
    add bx, 20 ;转到下一商品偏移首地址
    pop cx
    loop cmpitem1

ok33:  pop si ;取出商品偏移首地址
       mov al, 12[si]
       mov bl, 16[si]
       imul bl ;销售价乘以销售数量
       push ax
       mov al, 10[si]
       mov bl, 14[si]
       imul bl ;进货价乘以进货数量
       pop di
       push ax
       sub di, ax
       mov ax, di
       mov bx, 100
       imul bx
       pop di
       idiv di ;利润计算，商放在ax里
       mov bx, offset apr1
       mov bx, [bx]
       add ax, bx
       mov bl, 2
       idiv bl ;总利润，结果放在al里
       mov ah, 0
       mov word ptr apr, ax
       jmp short f4
```

```
f3_5:  lea dx, tip7
```

汇编语言程序设计实验报告

```
mov ah,9
int 21h
jmp far ptr f3
f3_4: mov bl,in_itemname+1           ;后一位设置结束标志
      mov bh,0
      mov byte ptr in_itemname+2[bx], '$'
      lea dx,tip6
      mov ah,9
      int 21h
      lea dx,in_itemname+2
      mov ah,9
      int 21h
      jmp far ptr f1

f4:   push ax
      lea dx,tip5
      mov ah,9
      int 21h
      pop ax
      cmp al,90 ;不小于90
      jnl f4a
      cmp al,50 ;不小于50
      jnl f4b
      cmp al,20 ;不小于20
      jnl f4c
      cmp al,0 ;不小于0
      jnl f4d
      jmp short f4f
f4a:  mov dl,41h
      mov ah,2
      int 21h
      jmp far ptr f1
f4b:  mov dl,42h
      mov ah,2
      int 21h
      jmp far ptr f1
f4c:  mov dl,43h
      mov ah,2
      int 21h
      jmp far ptr f1
f4d:  mov dl,44h
      mov ah,2
      int 21h
      jmp far ptr f1
f4f:  mov dl,46h
      mov ah,2
      int 21h
      jmp far ptr f1
code ends
end f1
```

3.5.4 实验步骤

1. 准备上机实验环境。

汇 编 语 言 程 序 设 计 实 验 报 告

2. 使用编辑程序 EDIT.EXE 录入源程序，存盘文件名为 code5.ASM。使用 MASM 6.0 汇编源文件。即 MASM code5；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。

3. 使用连接程序 LINK.EXE 将汇编生成的 code5.OBJ 文件连接成执行文件。即 LINK code5，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 code5.EXE 文件。

4. 测试程序各个功能的正确性。

5. 完成思考题。

3.5.5 实验记录与分析

1. 实验环境条件：P3 1GHz, 256M 内存；WINDOWS 10 下 DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0；LINK.EXE 5.2；TD.EXE 5.0。

2. 汇编源程序时，汇编程序没有报错，进入连接。

3. 连接过程没有发生异常。

4. 程序中的逻辑错误如下：（由于当时没有截图，现只在下面说明情况）

（1）一开始程序执行到最后跳到功能 1 时，没有将 auth 的值重置，导致后面的查询操作出现错误。

（2）一开始没有将缓冲区中的回车置 0，导致后面的登录判断出现了问题。

（3）一开始没有做有符号数的乘除运算，导致计算机直接把数据当成无符号数处理，这直接导致了最后的结果出错，与其他数比较时存在严重错误。最后使用了有符号数的操作，imul, idiv 解决了错误问题。

（4）最后利润率计算结果商存储在 al 里，将其与个利润率标准对比的时候，我将 ah 置为 0，再将 ax 与立即数进行比较，我发现这样的操作是有问题的，因为 cmp 比较有符号数的时候，是根据减法后，标志寄存器 sf, of 的值来进行逻辑判断的。而扩展之后，sf, of 的变化不是原生变化，导致 cmp 指令不能正确判断两者的大小，导致最后的比较结果出错。解决方案直接拿 al 与立即数比较，这样结果正确。

5. 测试程序的正确性：（由于发生错误没有截图，错误在上面描述中已经给出）

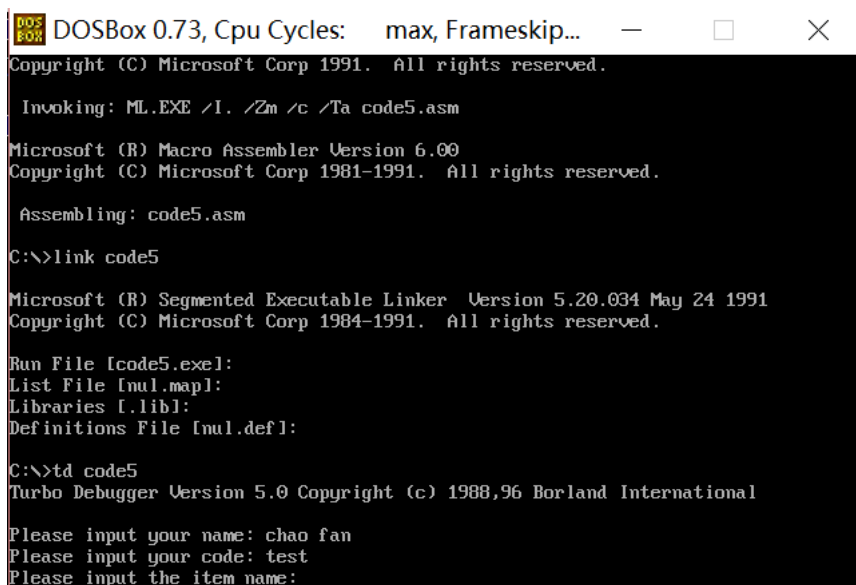
（1）输入姓名时，若输入超过了缓冲区定义的字符数量，会“卡住”，输入不进去字符，这时候只要按 backspace 退几个字符，再按下回车就好。

（2）提示输入姓名，密码：

分别测试输入正确，输入错误，输入 q 的情况，输入回车的情况

测试结果如下图所示，都符合预期

汇编语言程序设计实验报告



```
DOSBox 0.73, Cpu Cycles: max, Frameskip...
Copyright (C) Microsoft Corp 1991. All rights reserved.

Invoking: ML.EXE /I. /Zm /c /Ta code5.asm

Microsoft (R) Macro Assembler Version 6.00
Copyright (C) Microsoft Corp 1981-1991. All rights reserved.

Assembling: code5.asm

C:\>link code5

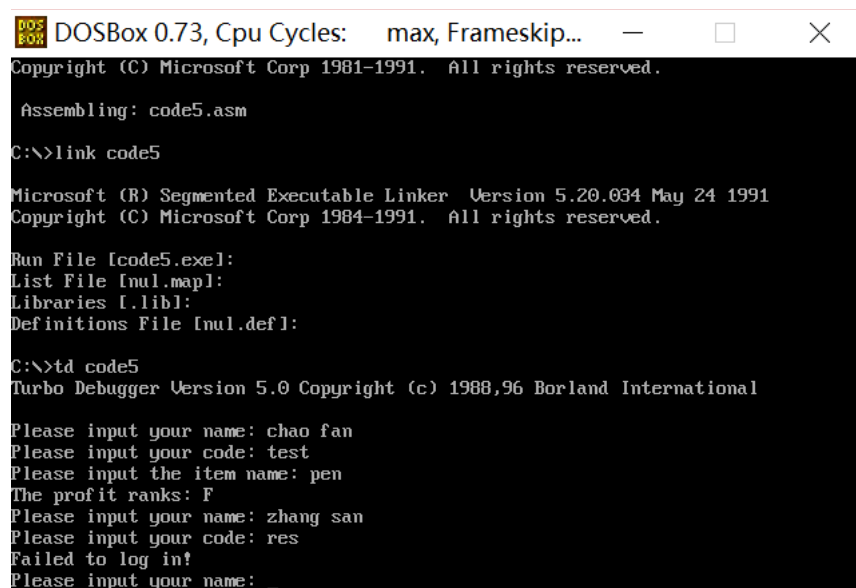
Microsoft (R) Segmented Executable Linker Version 5.20.034 May 24 1991
Copyright (C) Microsoft Corp 1984-1991. All rights reserved.

Run File [code5.exe]:
List File [nul.map]:
Libraries [libl]:
Definitions File [nul.def]:

C:\>td code5
Turbo Debugger Version 5.0 Copyright (c) 1988,96 Borland International

Please input your name: chao fan
Please input your code: test
Please input the item name: _
```

图 3.5.5 提示输入姓名，密码



```
DOSBox 0.73, Cpu Cycles: max, Frameskip...
Copyright (C) Microsoft Corp 1981-1991. All rights reserved.

Assembling: code5.asm

C:\>link code5

Microsoft (R) Segmented Executable Linker Version 5.20.034 May 24 1991
Copyright (C) Microsoft Corp 1984-1991. All rights reserved.

Run File [code5.exe]:
List File [nul.map]:
Libraries [libl]:
Definitions File [nul.def]:

C:\>td code5
Turbo Debugger Version 5.0 Copyright (c) 1988,96 Borland International

Please input your name: chao fan
Please input your code: test
Please input the item name: pen
The profit ranks: F
Please input your name: zhang san
Please input your code: res
Failed to log in!
Please input your name: _
```

图 3.5.6 提示输入姓名，密码，输入错误后提示

汇编语言程序设计实验报告

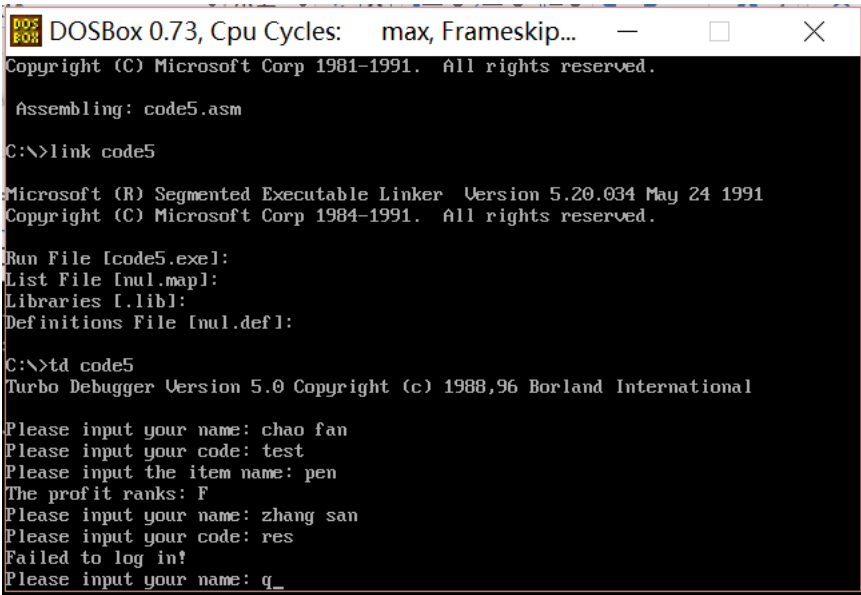


图 3.5.7 提示输入姓名，密码，输入 q

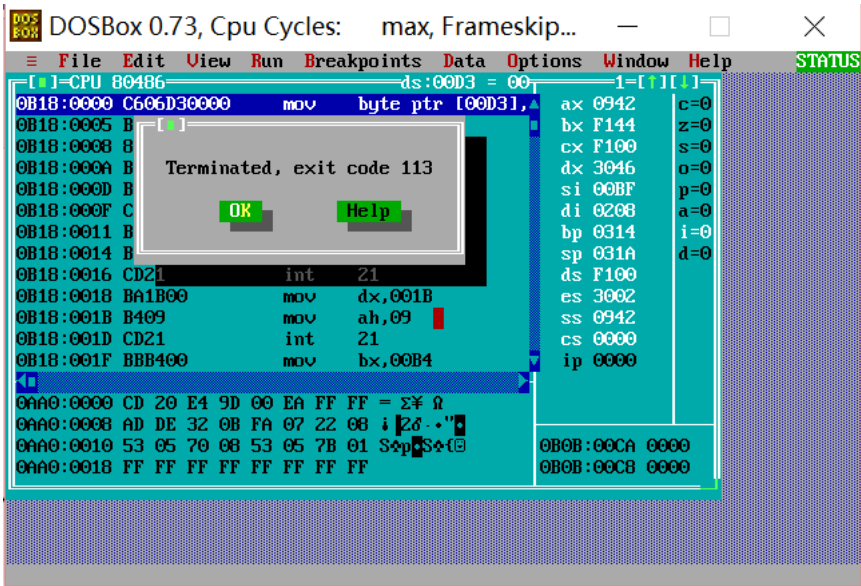
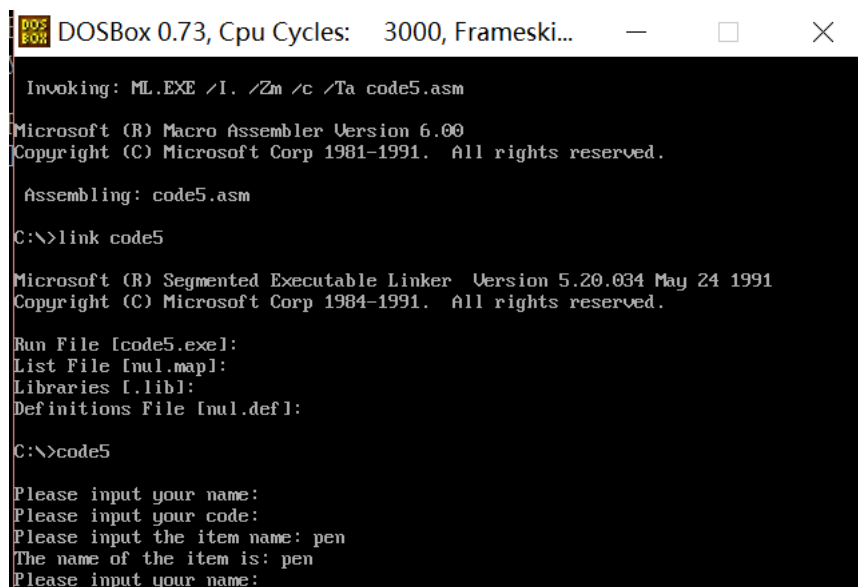


图 3.5.8 退出

汇编语言程序设计实验报告

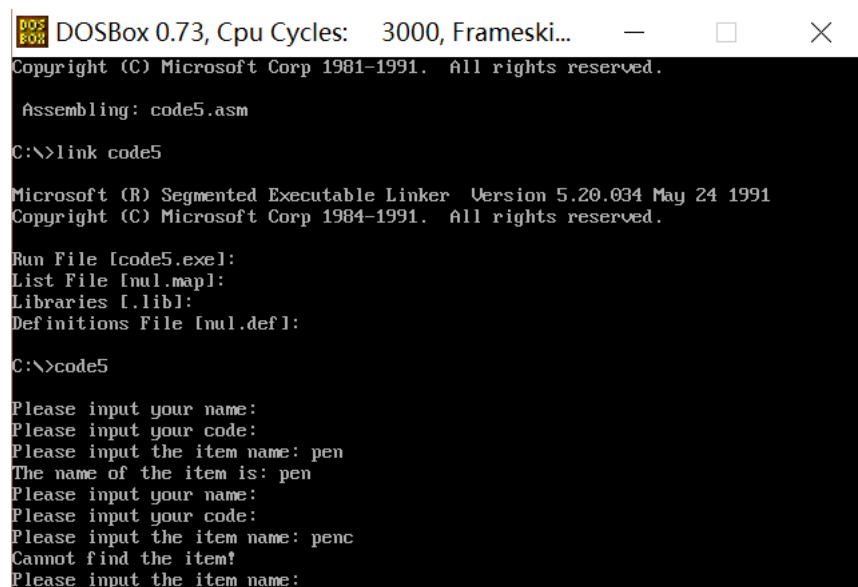


```
DOSBox 0.73, Cpu Cycles: 3000, Frameski...  
Invoking: ML.EXE /I. /Zm /c /Ta code5.asm  
Microsoft (R) Macro Assembler Version 6.00  
Copyright (C) Microsoft Corp 1981-1991. All rights reserved.  
  
Assembling: code5.asm  
C:\>link code5  
  
Microsoft (R) Segmented Executable Linker Version 5.20.034 May 24 1991  
Copyright (C) Microsoft Corp 1984-1991. All rights reserved.  
  
Run File [code5.exe]:  
List File [nul.map]:  
Libraries [libl]:  
Definitions File [nul.def]:  
  
C:\>code5  
  
Please input your name:  
Please input your code:  
Please input the item name: pen  
The name of the item is: pen  
Please input your name: _
```

图 3.5.9 输入回车，功能正常

(3) 在输入的商品名称不存在时，程序输出错误提示

Cannot find the item!



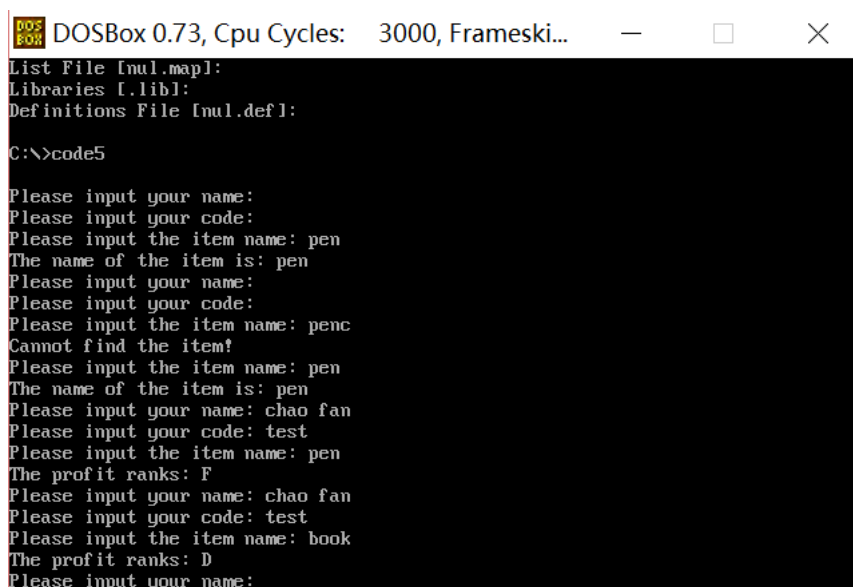
```
DOSBox 0.73, Cpu Cycles: 3000, Frameski...  
Copyright (C) Microsoft Corp 1981-1991. All rights reserved.  
  
Assembling: code5.asm  
C:\>link code5  
  
Microsoft (R) Segmented Executable Linker Version 5.20.034 May 24 1991  
Copyright (C) Microsoft Corp 1984-1991. All rights reserved.  
  
Run File [code5.exe]:  
List File [nul.map]:  
Libraries [libl]:  
Definitions File [nul.def]:  
  
C:\>code5  
  
Please input your name:  
Please input your code:  
Please input the item name: pen  
The name of the item is: pen  
Please input your name:  
Please input your code:  
Please input the item name: penc  
Cannot find the item!  
Please input the item name:
```

图 3.5.10 输入不存在的商品名称

(4) 在登录状态下，查看商品的利润等级

在事前计算中，pen 的利润等级为 F，利润率为负。Book 的利润等级为 D，利润为 12%。测试结果如下图，并成功回到功能 1：

汇编语言程序设计实验报告



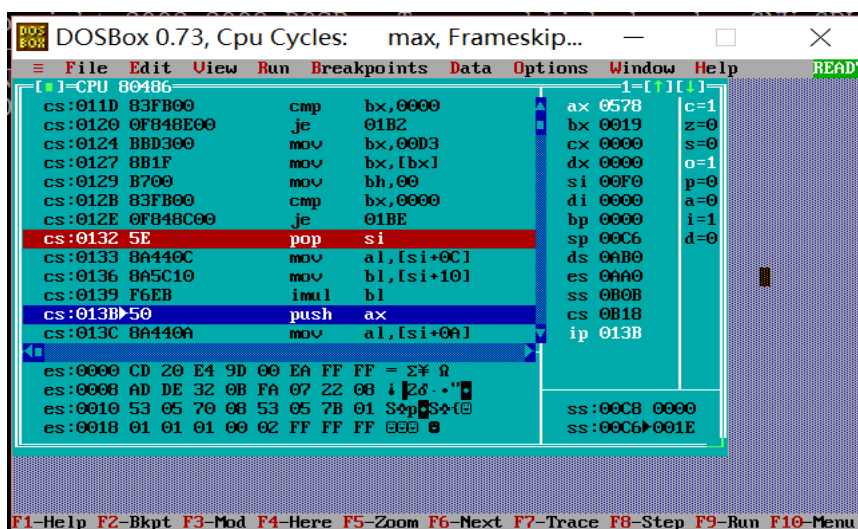
```
DOSBox 0.73, Cpu Cycles: 3000, Frameski...
List File [nul.map]:
Libraries [.lib]:
Definitions File [nul.def]:
C:\>code5

Please input your name:
Please input your code:
Please input the item name: pen
The name of the item is: pen
Please input your name:
Please input your code:
Please input the item name: penc
Cannot find the item!
Please input the item name: pen
The name of the item is: pen
Please input your name: chao fan
Please input your code: test
Please input the item name: pen
The profit ranks: F
Please input your name: chao fan
Please input your code: test
Please input the item name: book
The profit ranks: D
Please input your name: _
```

图 3.5.11 查看商品利润率等级

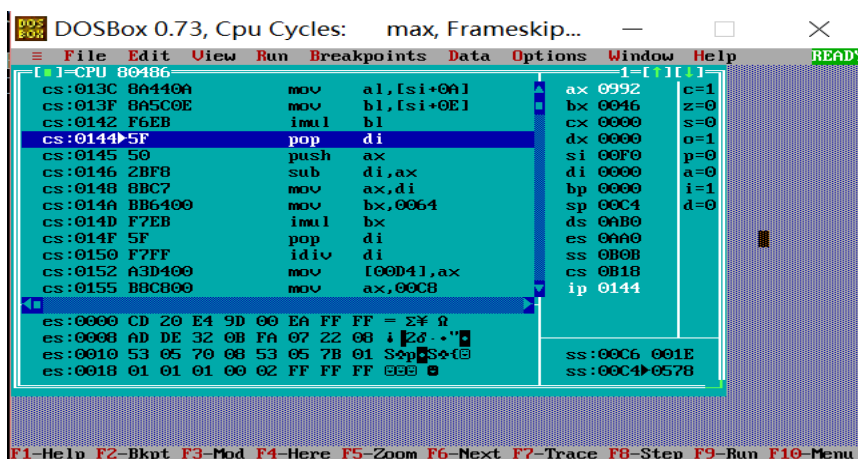
6. 思考题

(1) 单步调试计算过程, 根据事前计算结果, 各步执行结果均正确



```
DOSBox 0.73, Cpu Cycles: max, Frameskip...
File Edit View Run Breakpoints Data Options Window Help
[1]-CPU 80486
cs:011D 83FB00 cmp bx,0000 ax 0578 c=1
cs:0120 0F848E00 je 01B2 bx 0019 z=0
cs:0124 BBD300 mov bx,00D3 cx 0000 s=0
cs:0127 8B1F mov bx,[bx] dx 0000 o=1
cs:0129 B700 mov bh,00 si 00F0 p=0
cs:012B 83FB00 cmp bx,0000 di 0000 a=0
cs:012E 0F848C00 je 01BE bp 0000 i=1
cs:0132 5E pop si sp 00C6 d=0
cs:0133 8A440C mov al,[si+0C] ds 0AB0
cs:0136 8A5C10 mov bl,[si+10] es 0AA0
cs:0139 F6EB imul bl ss 0B0B
cs:013B 50 push ax cs 0B18
cs:013C 8A440A mov al,[si+0A] ip 013B
es:0000 CD 20 E4 9D 00 EA FF FF = 2F 0
es:0008 AD DE 32 0B FA 07 22 08 i 2d .''
es:0010 53 05 70 08 53 05 7B 01 S ap S a (
es:0018 01 01 01 00 02 FF FF FF 000 0
ss:00C8 0000
ss:00C6 001E
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```

图 3.5.12 计算过程



```
DOSBox 0.73, Cpu Cycles: max, Frameskip...
File Edit View Run Breakpoints Data Options Window Help
[1]-CPU 80486
cs:013C 8A440A mov al,[si+0A] ax 0992 c=1
cs:013F 8A5C0E mov bl,[si+0E] bx 0016 z=0
cs:0142 F6EB imul bl cx 0000 s=0
cs:0144 5F pop di dx 0000 o=1
cs:0145 50 push ax si 00F0 p=0
cs:0146 2BFB sub di,ax di 0000 a=0
cs:0148 B8C7 mov ax,di bp 0000 i=1
cs:014A BB6400 mov bx,0064 sp 00C4 d=0
cs:014D F7EB imul bx ds 0AB0
cs:014F 5F pop di es 0AA0
cs:0150 F7FF idiv di ss 0B0B
cs:0152 A3D400 mov [00D4],ax cs 0B18
cs:0155 B8C800 mov ax,00C8 ip 0144
es:0000 CD 20 E4 9D 00 EA FF FF = 2F 0
es:0008 AD DE 32 0B FA 07 22 08 i 2d .''
es:0010 53 05 70 08 53 05 7B 01 S ap S a (
es:0018 01 01 01 00 02 FF FF FF 000 0
ss:00C6 001E
ss:00C4 0578
F1-Help F2-Bkpt F3-Mod F4-Here F5-Zoom F6-Next F7-Trace F8-Step F9-Run F10-Menu
```


汇编语言程序设计实验报告

图 3.5.13 计算过程

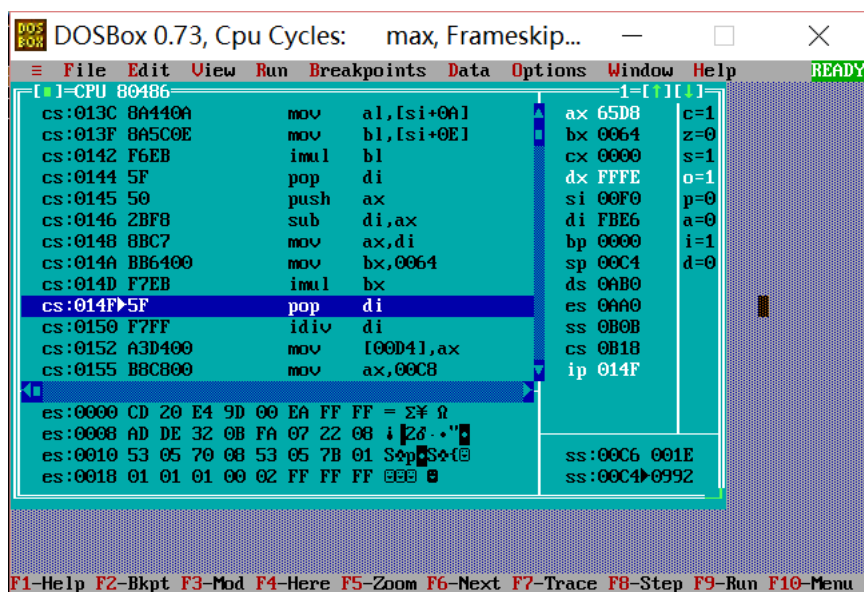


图 3.5.14 计算过程

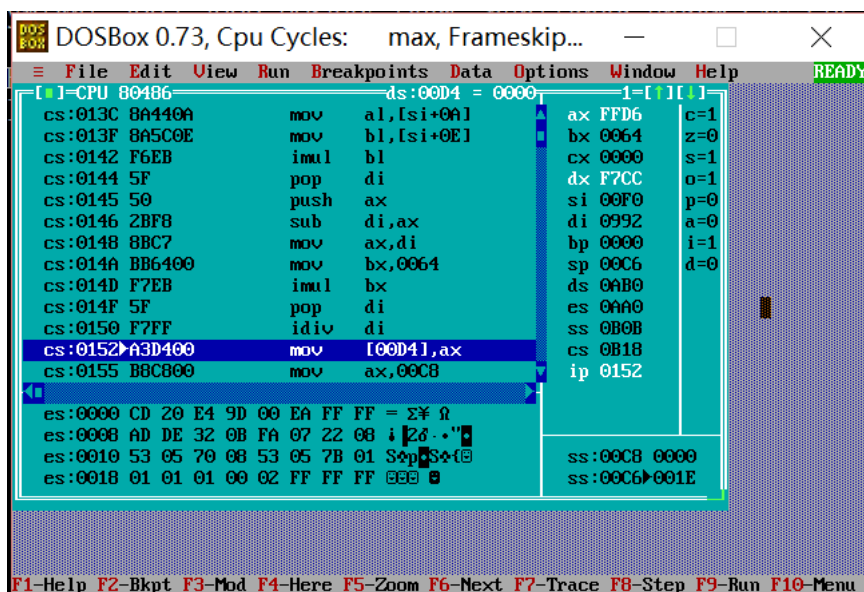


图 3.5.15 计算过程

(2) 对输入姓名合法性进行检查，下面是思考解答：

对是否存在大小写字母之外的字符，这个可以根据 ascii 码来判断，结合 cmp 功能，逐个字符判断即可。

(3) 对 S1/S2 中预先定义的商品信息进行合法性检查，判断是否符合实际（如价格是负数，已售数量大于进货总数等），不符合则提示并退出程序。

这个可以将内存单元中的数值送入寄存器，再结合 cmp 功能进行判断即可。

(4) 除了显示平均利润率的等级外，还将利润率同时显示

这个在后面的任务中将会有涉及。

(5) 采用数据结构来定义 S1/S2 的存储区是否会更好？

汇编语言程序设计实验报告

这样会使寻址更加方便，增加了代码的可读性。

(6) 如何让 9 号功能调用显示的信息放在自己希望的位置？

可以利用换行加回车使信息放在自己希望的位置。

(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？

程序会不停输出，直到遇见一个\$。

(8) 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？

结果不会显示你想要显示的信息。

(9) 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？

会禁止键盘继续输入，这是只能输入 Backspace。

(10) 匹配姓名时，如何提高匹配速度？

可以先匹配名字长度，再匹配是否一致。

(11) 循环或转移时，是否有多种指令的组合方式实现？

比如说 loop 和条件转移指令。

(12) 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。偏移地址会以立即数的方式转换成机器码。

(13) 是否可以简化一下计算公式。

可以通过人工化简的方式，先不用减去进货价乘以总数，这个在最后减去 100 即可。

(14) 结果溢出的情况。

如果是除法溢出，则会引发异常。

汇编语言程序设计实验报告

4 总结与体会

在本次实验中，首先是熟悉了 dosbox 操作环境，学会了 masm, link, td 等工具的使用。其中 td 的掌握需要一定的时间，在本次实验中，我初步掌握了 td 的基本使用方法，通过 td，我知道了数，指令，寻址方式在计算机内是如何表示的，了解了汇编指令与机器指令之间的关系，也了解了汇编源程序与反汇编之间的区别。

通过任务 1 的实验，我知道了 TD 不仅可以调入已有的程序进行调试，而且能随时输入和测试单一一条指令是否正确，执行效果如何，这对未来的学习过程是有极大的帮助的。另外，通过观察，计算机内的加减运算，无论是否有符号数，对应的标志位都是设定好了的，如何使用这些标志，完全由我们选择的指令来决定，这就要求我们编写程序时要理解好题目，选择合适的指令，不是我们随便输入了一个数，系统就会自动识别的。

通过任务 2 的实验，我学会了在 TD 中使用 goto 跳到自己指定的代码段，数据段或栈段。还知道了要输出一串字符串，要在字符串末尾加上\$符号，后来知道了这是一个终止符号，其次 0ah, 0dh 代表了回车和换行。还学会了 lea 语句的使用，这是将变量的偏移地址赋给寄存器的指令，mov ah, 9 int 21h 则是显示字符串的功能调用。mov ah, 1 int 21h 则是等待用户输入任意字符然后进入下一条指令的功能调用。当然，调用这些功能时，要注意保存好 ax 寄存器里的内容，保存内容可以通过栈实现。

通过任务 3 的实验，我了解了机器指令和汇编指令间的关系，知道了反汇编语句与源程序的区别（具体请见任务 3），重点是知道了计算机通过 cs: ip 的值来控制代码的运行，这非常重要，也解释了为什么不能用简单的 mov 指令更改 cs, ip 的值。如果我们随意去更改 cs: ip 的值，会导致程序运行错误，发生意想不到的后果。

通过任务 4 的实验，我主要是对几种寻址方式有了更深的了解，在以后的学习中会比较这几种寻址方式，并考虑该如何使用它们。比如，使用二维数组就可以使用基址加变址寻址方式来实现。这些寻址方式十分重要，希望在以后的学习中能够多加运用。

本次上机不仅提高了编程水平，熟悉了工具的使用，而且加深了对一些知识的理解。主要的经验教训如下：

首先，更加感受到实验前准备的意义。例如：上机前先熟悉一遍具体操作，各种功能的使用，写好源程序，在实验中就能表现得更好。

其次，要注意在编写程序时可能会有一部分书写错误，如伪指令输入错误，寄存器名称不小心输入错误，这些都会导致程序功能不能正常运行，尤其是寄存器名称不小心输入错误，有时这会浪费我们很长时间去修正这个错误。

最后，TD 的功能远不止我们这次上机所用到的几种基本功能，TD 更多的功能都要求我们在以后的实验中去摸索。还有更多的疑问没有在这次实验中体现出来，如如何在数据段中准确寻找到指定元素的位置，在有子程序的时候，有两种调试方法它们有什么区别，为什么有些汇编指令通过反汇编显现出来的语句不一样等。

第二次的任务 5 实验，其难度较上一次实验有了明显的提升，其实主要的难点不在于编写程序，而在于编写完程序后的调试中。代码中的逻辑错误是我们要花很长时间才能去修改正确的。这要求我们平时对汇编语言有更加深入的了解，才能对这些逻辑错误有很好的理解。比如这次程序中

汇编语言程序设计实验报告

出现的问题，一开始程序执行到最后跳到功能 1 时，没有将 `auth` 的值重置，导致后面的查询操作出现错误。这属于很常见的错误，只是一个重置操作而已，但是你忽视了的话就会导致很大的问题。又比如一开始没有将缓冲区中的回车置 0，导致后面的登录判断出现了问题。这在使用循环确认的时候很容易出现错误，一旦将回车输入进去，就会导致你输入正确了，但程序将你误判。又比如一开始没有做有符号数的乘除运算，导致计算机直接把数据当成无符号数处理，这直接导致了最后的结果出错，与其他数比较时存在严重错误。最后使用了有符号数的操作，`imul`，`idiv` 解决了错误问题。这完全是由于当时概念没有仔细看，又比如后面的 `jnl`，`jnb` 之间的区别，一个是有符号的，一个是无符号的，在平时的学习过程中要区分清楚。

在这次编写了一个较完整的程序后，我学会了如何正确编写汇编语言，也了解到汇编的逻辑错误是要花时间去纠正的，希望在这次程序过后，能增长经验，为今后编写更复杂的程序做准备。

汇 编 语 言 程 序 设 计 实 验 报 告

参考文献

- [1] 王元珍、韩宗芬、曹忠升.《80X86 汇编语言程序设计》. 华中科技大学出版社:2005 年 04 月