

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**实验时段： 2018/04/02下午**

**指导教师： 左 琼**

**专业班级： CS1603**

**学 号： U201614577**

**姓 名： 龙 际 全**

实验报告成绩

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 实验 序号 | 一 | 二 | 三 | 四 | 五 | 六 | 总评 |
| 成绩 |  |  |  |  |  |  |  |

指导教师签字：

                     日 期：

**计算机科学与技术学院**

**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： 2018.04.02 实验地点：南一楼804\_3实验室79试验台**

**指导教师： 左琼**

**专业班级：计科 201603班**

**学 号： U201614577 姓 名： 龙际全**

**同组学生： 无 报告日期： 2018年 04 月 02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：

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](#_Toc510646222)

[2 实验内容 1](#_Toc510646223)

[3 实验过程 3](#_Toc510646224)

[3.1任务1 3](#_Toc510646225)

[3.1.1源程序 3](#_Toc510646226)

[3.1.2实验步骤 4](#_Toc510646227)

[3.1.3实验记录与分析 4](#_Toc510646228)

[3.2任务2 10](#_Toc510646229)

[3.2.1源程序 10](#_Toc510646230)

[3.2.2实验步骤 11](#_Toc510646231)

[3.2.3实验记录与分析 11](#_Toc510646232)

[3.3任务3 16](#_Toc510646233)

[3.3.1源程序 16](#_Toc510646234)

[3.3.2实验步骤 16](#_Toc510646235)

[3.3.3实验记录与分析 16](#_Toc510646236)

[3.4任务4 19](#_Toc510646237)

[3.4.1源程序 19](#_Toc510646238)

[3.4.2实验步骤 20](#_Toc510646239)

[3.4.3实验记录与分析 20](#_Toc510646240)

[3.5任务5 23](#_Toc510646241)

[3.5.1源程序 23](#_Toc510646242)

[3.5.2实验步骤 29](#_Toc510646243)

[3.5.3实验记录与分析 29](#_Toc510646244)

[4 总结与体会 33](#_Toc510646245)

[5 注 34](#_Toc510646246)

[参考文献 35](#_Toc510646247)

# 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。

# 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。

**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 XUEHAO开头的存储区中，要求学号的存放以字符方式存放。

要求：在报告中给出完整的程序；给出运行效果截图；（不需要画流程图）；在程序注释中，明确指出访问存储单元时，用的是什么寻址方式。

**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（必须是自己名字的拼音）

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

（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到该商品，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）。

**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”。

**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（注意，“%”是不要出现在计算式子和指令语句中的）。

（2）使用转移指令回到“功能一（1）”处（提示并输入姓名和密码）。

# 实验过程

## 3.1任务1

### 3.1.1源程序

第一组：

MOV AH,00110011B  
MOV AL,01011010B  
ADD AH,AL

SUB AH,AL

第一组：

MOV AH,-0101001B  
MOV AL,-1011101B  
ADD AH,AL

SUB AH,ALJ

第一组：

MOV AH,1100101B  
MOV AL,-1011101B  
ADD AH,AL

SUB AH,AL

### 3.1.2实验步骤

1. 准备上机实验环境。

2. 在TD的代码窗口中输入上述三条代码；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。

### 3.1.3实验记录与分析

1. 实验环境条件：AMD A8-8600P 1.6GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。

第一组：

(1)输入第一条指令MOV AH,00110011B，。单步执行后，AX=3300H,IP=0102H，运行截图如下

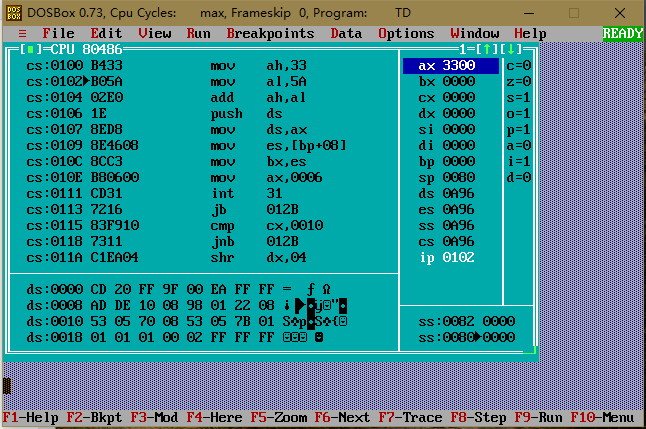


图3.1.1 AH赋值

(2)输入第二条指令MOV AL,01011010B，。单步执行后，AX=335AH,IP=0104H，运行截图如下：

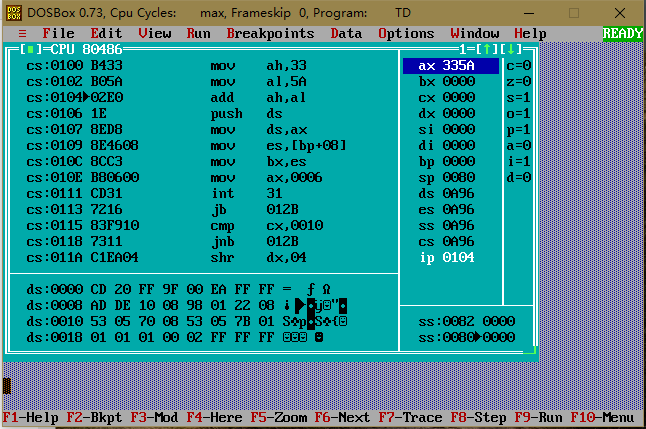


图3.1.2 AL赋值

(3)输入第三条指令ADD AH,AL，。单步执行后，AX=8D5AH，IP=0106H,SF=1,OF=1,ZF=0,CF=0,运行截图如下：

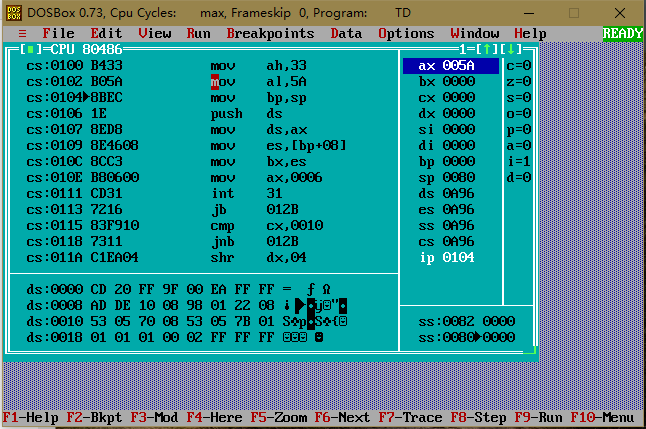


图3.1.3 AH与AL相加

(4)输入第四条指令SUB AH,AL，单步执行后，AX=D95AH，IP=0108H,SF=1,OF=0,ZF=0,CF=1,运行截图如下

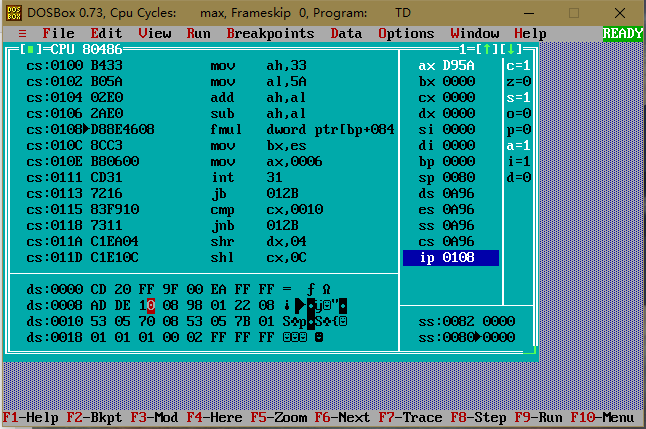


图3.1.4 AH与AL相减

第二组：

(1)输入第一条指令MOV AH,-0101001B，。单步执行后，AX=D700H,IP=0102H，运行截图如下

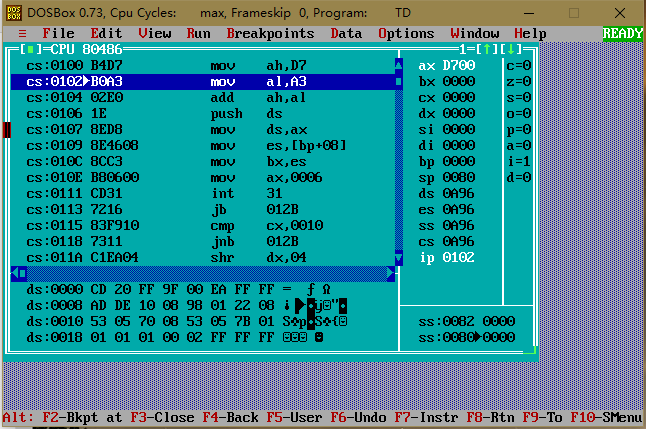


图3.1.5 AH赋值

(2)输入第二条指令MOV AL,-1011101B，。单步执行后，AX=D7A3H,IP=0104H，运行截图如下:

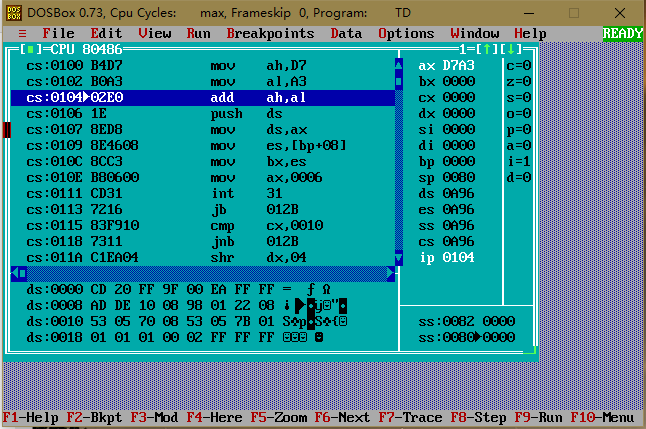


图3.1.6 AL赋值

(3)输入第三条指令ADD AH,AL，。单步执行后，AX=7AA3H，IP=0106H,SF=0,OF=1,ZF=0,CF=1,运行截图如下

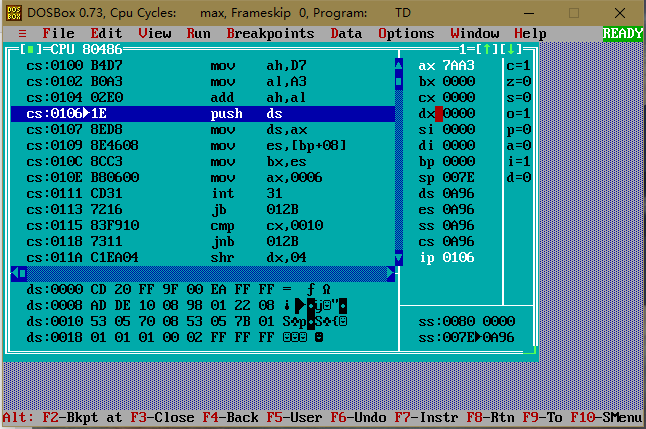


图3.1.7 AH与AL相加

(4)输入第四条指令SUB AH,AL，单步执行后，AX=34A3H，IP=0108H,SF=0,OF=0,ZF=0,CF=0,运行截图如下

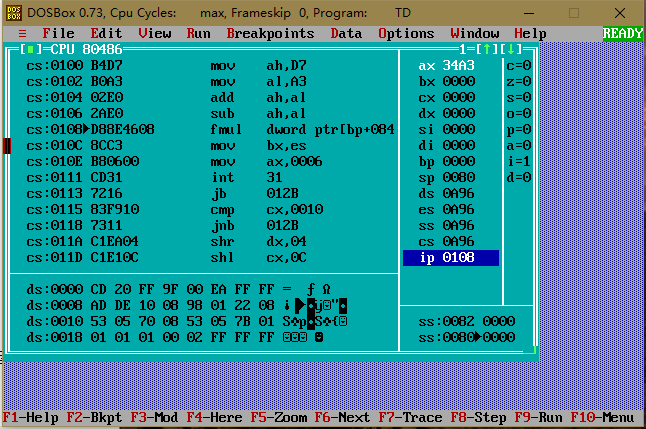


图3.1.7 AH与AL相减

第三组：

(1)输入第一条指令MOV AH,1100101B，。单步执行后，AX=6500H,IP=0102H，运行截图如下

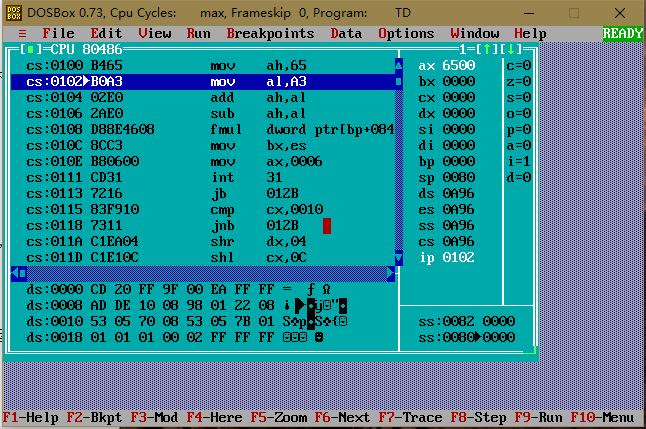


图3.1.9 AH赋值

(2)输入第二条指令MOV AL,-1011101B，。单步执行后，AX=65A3H,IP=0104H，运行截图如下:

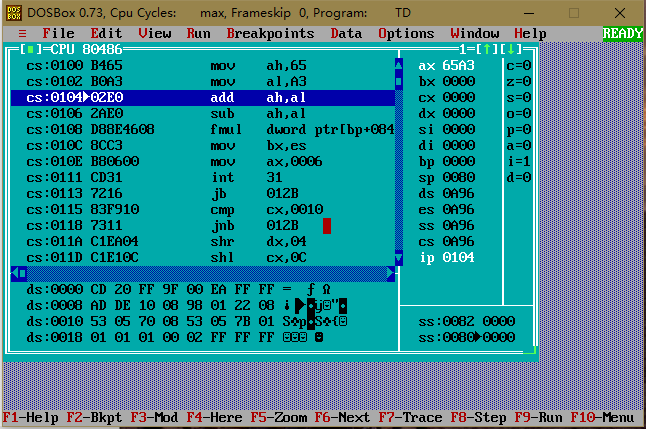


图3.1.10 AL赋值

(3)输入第三条指令ADD AH,AL，。单步执行后，AX=08A3H，IP=0106H,SF=0,OF=0,ZF=0,CF=1,运行截图如下

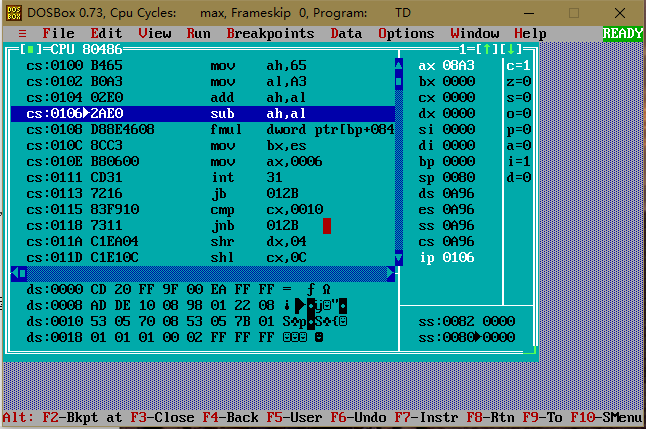


图3.1.11 AH与AL相加

(4)输入第四条指令SUB AH,AL，单步执行后，AX=C2A3H，IP=0108H,SF=1,OF=1,ZF=0,CF=1,运行截图如下

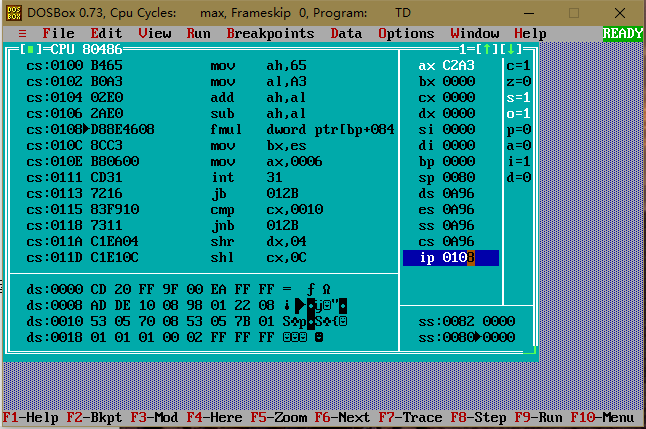


图3.1.11 AH与AL相减

## 3.2任务2

### 3.2.1源程序

;---------------------------------------------------------

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

;----------------------------------------------------------

STACK SEGMENT STACK

DB 200 DUP(0)

STACK ENDS

;-------------------------------------------------------

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

LOOPA: 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 LOOPA

MOV AH, 4CH

INT 21H

CODE ENDS

END START

### 3.2.2实验步骤

* 1. 准备上机环境，编辑、汇编、连接文件TASK2。
  2. 使用TD.EXE观察TASK2的执行情况。即 TD TASK2.EXE回车

（1）记录初始(BX), (BP),(SI),(DI)各是多少。

（2）记录执行到“MOV CX，10”之前时，(BX), (BP),(SI),(DI)各是多少。

（3）记录执行到“INT 21H”之前时，(BX), (BP),(SI),(DI)各是多少。

（4）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。

### 3.2.3实验记录与分析

1. 实验环境条件：AMD A8-8600P 1.6GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。

2. 编译连接正常，如图3.2.1所示

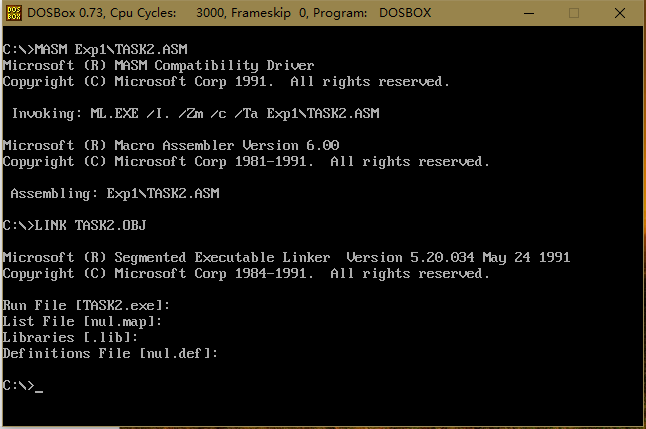


图3.2.1 正常编译连接

* 1. 输入TD TASK2，进入调试界面，(BX), (BP),(SI),(DI)初始值如图3.2.2所示

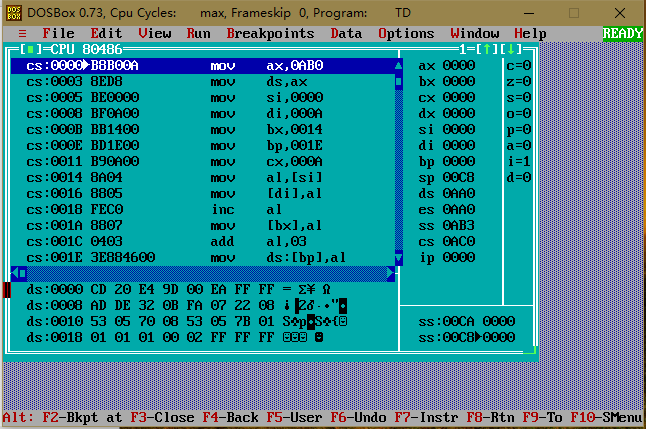


图3.2.2 (BX), (BP),(SI),(DI)初始值

* 1. 执行到“MOV CX，10”之前时，(BX), (BP),(SI),(DI)值如图3.2.3所示

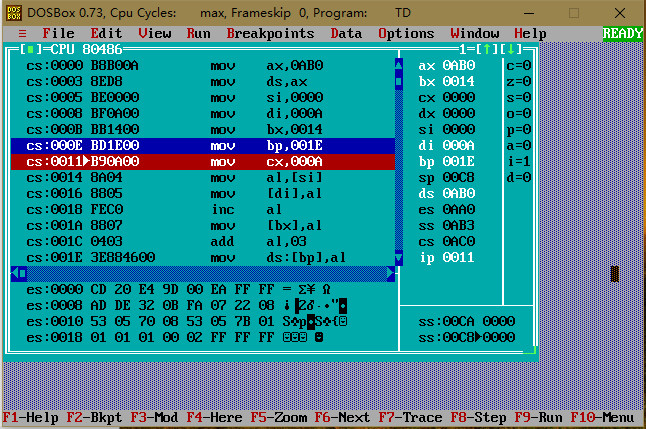


图3.2.3 执行到“MOV CX，10”之前时，(BX), (BP),(SI),(DI)值

* 1. 执行到“INT 21H”之前时，(BX), (BP),(SI),(DI)值如图3.2.4所示

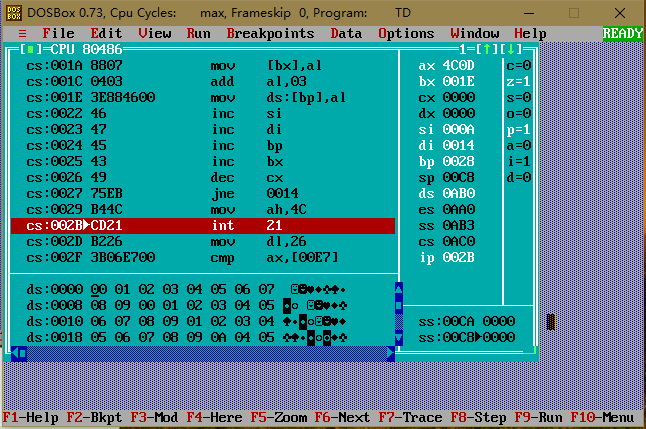


图3.2.4 执行到“INT 21H”之前时，(BX), (BP),(SI),(DI)值及执行到退出之前数据段开始40个字节的内容

* 1. 程序执行到退出之前数据段开始40个字节的内容，预测为

BUF1:0,1,2,3,4,5,6,7,8,9 BUF2:0,1,2,3,4,5,6,7,8,9

BUF3:1,2,3,4,5,6,7,8,9,A BUF4:4,5,6,7,8,9,A,B,C,D

实际结果如图3.2.4及图3.2.5所示

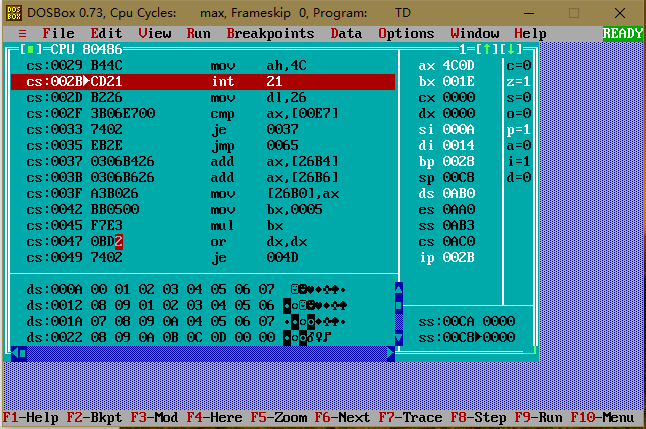


图3.2.5 执行到退出之前数据段开始40个字节的内容

由此可知，程序运行结果与设想的一致。

* 1. 要使程序具备功能三，需要DOS的1号调用和9号调用，修改后的源程序如下：

;---------------------------------------------------------

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)

STOP\_MESSAGE DB 'Press any key to begin!$',0AH,0DH

DATA ENDS

;----------------------------------------------------------

STACK SEGMENT STACK

DB 200 DUP(0)

STACK ENDS

;-------------------------------------------------------

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,STOP\_MESSAGE

MOV AH,9;DOS9号调用

INT 21H

MOV AH,1;DOS1号调用

INT 21H

LOOPA: 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 LOOPA

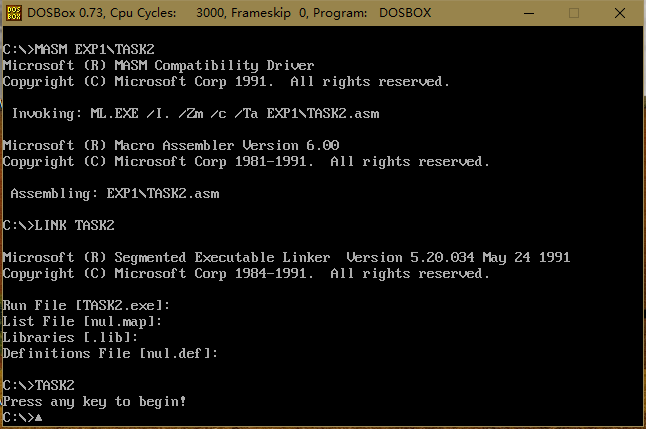
MOV AH, 4CH

INT 21H

CODE ENDS

END START

* 1. 修改后的程序运行截图如下：



图修改后的程序运行截图

可以看到屏幕中输出了字符串’Press any key to begin!’。

## 3.3任务3

### 3.3.1源程序

;------------------------------------------------------------

.386

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

;------------------------------------------------------------

STACK SEGMENT USE32 STACK

DB 200 DUP(0)

STACK ENDS

;------------------------------------------------------------

CODE SEGMENT USE16

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

START: MOV AX,DATA

MOV DS,AX

MOV ESI,0

MOV ECX,10

LOOPA: MOV AL,BUF1[ESI]

MOV BUF2[ESI],AL

INC AL

MOV BUF3[ESI],AL

ADD AL,3

MOV BUF4[ESI],AL

INC ESI

DEC ECX

JNZ LOOPA

MOV AH,4CH

INT 21H

CODE ENDS

END START

### 3.3.2实验步骤

1. 准备上机环境，编辑、汇编、连接文件TASK3。
2. 使用TD.EXE观察TASK3的执行情况。
3. 设置断点到程序退出前
4. 观察数据段开始40个字节的内容
5. 在程序运行的过程中，注意指令在内存中的存放方式以及源程序代码与反汇编代码的区别。
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，结果怎样？

### 3.3.3实验记录与分析

1. 实验环境条件：AMD A8-8600P 1.6GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。
2. 编译连接正常，如图3.3.1所示

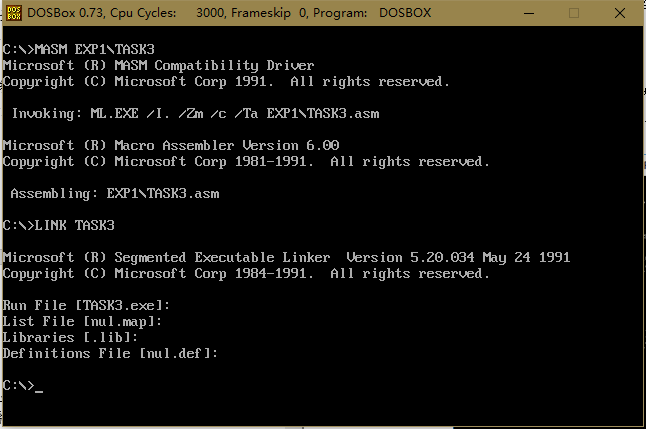


图3.3.1 正常编译连接

1. 输入TD TASK3，进入调试器，设置断点，运行到程序退出前，观察到数据段开始40个字节的内容与预想一样，如图3.3.2所示

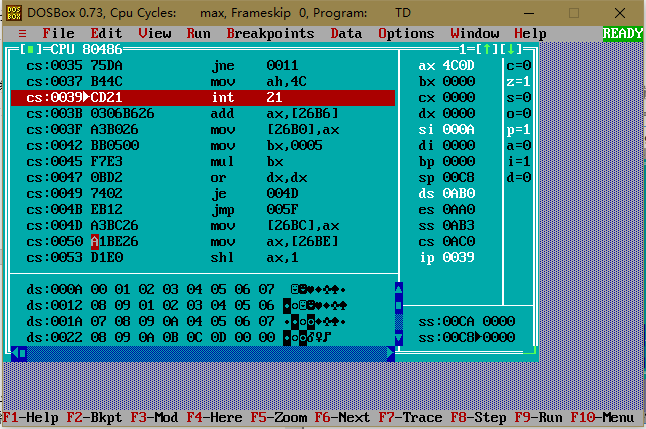


图3.3.2 运行到程序退出之前

1. 在程序运行过程中，由于任务3采用了32位寄存器，且采用变址寻址方式，因此变址寻址中的立即数变成8位。并且反汇编语言与源代码相比，将实际地址进行运算以16进制显示出来。并且发现反汇编语言、汇编语言与机器代码是一一对应关系。如图3.3.3，图3.3.4，图3.3.5所示:

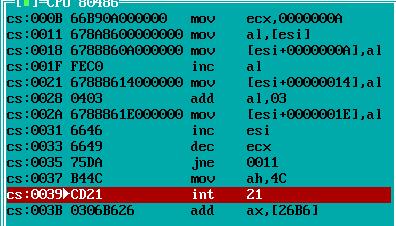


图3.3.3 32位寄存器

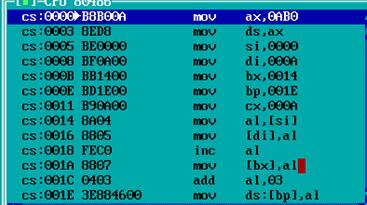


图3.3.4 16位寄存器

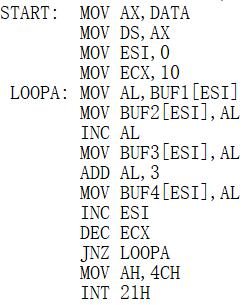


图3.3.5 汇编源代码

1. 重新运行程序，观察CS段存储内容，发现机器语言与CS段存储字节相对应，如图3.3.6所示

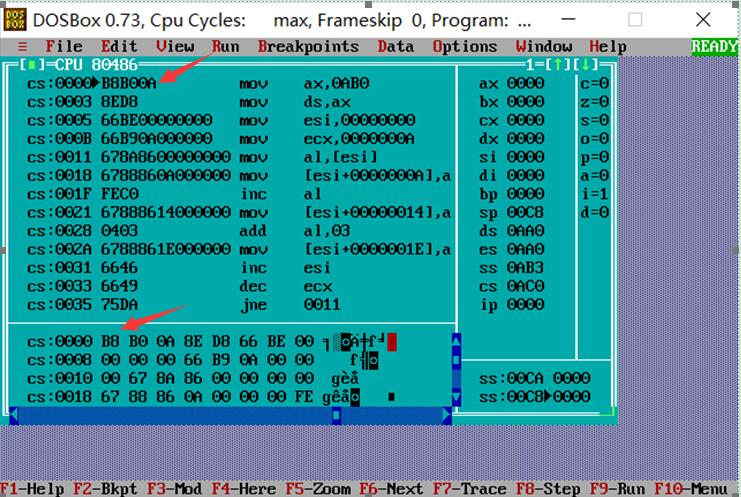


图3.3.6 CS段机器语言与存储字节

1. 修改IP值为0005，运行单步运行程序，如图3.3.7所示，发现程序跳过0004之前的语句，直接执行IP所指向的语句，由此可以看出IP/EIP指明的指令起始位置十分重要，稍有不对或者被误更改，程序的执行顺序都会出乱子。

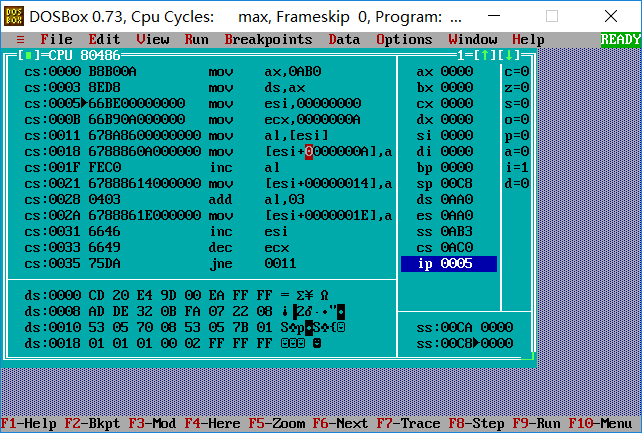


图3.3.7 修改IP值后程序状态

## 3.4任务4

### 3.4.1源程序

;-----------------------------------------------------------------

DATA SEGMENT USE16

ORIGIN\_BUF DB '4','5','7','7'

XUEHAO DB 4 DUP(0)

DATA ENDS

;-----------------------------------------------------------------

STACK SEGMENT STACK

DB 200 DUP(0)

STACK ENDS

;------------------------------------------------------------------

CODE SEGMENT USE16

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

START:MOV AX,DATA

MOV DS,AX

MOV SI,OFFSET ORIGIN\_BUF

MOV DI,OFFSET XUEHAO+1

MOV AL,ORIGIN\_BUF

MOV XUEHAO,AL ;直接寻址

INC SI

MOV AL,[SI]

MOV [DI],AL ;间接寻址

INC SI

MOV AL,ORIGIN\_BUF[SI]

MOV XUEHAO[SI],AL ;变址寻址

INC SI

LEA BX,ORIGIN\_BUF

LEA BP,XUEHAO

MOV AL,[BX][SI]

MOV DS:[BP][SI],AL ;基址加变址寻址

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. 使用记事本编辑录入源程序，存盘文件名为TASK4.ASM。使用MASM 6.0汇编源文件。即masm TASK4.ASM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TASK4.OBJ文件连接成可执行文件。即link TASK4.OBJo；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TASK4.EXE文件。

4. 使用调试器TD.EXE对生成的文件进行调试。

### 3.4.3实验记录与分析

1. 实验环境条件：AMD A8-8600P 1.6GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。
2. 编译连接正常，如图3.1.1所示

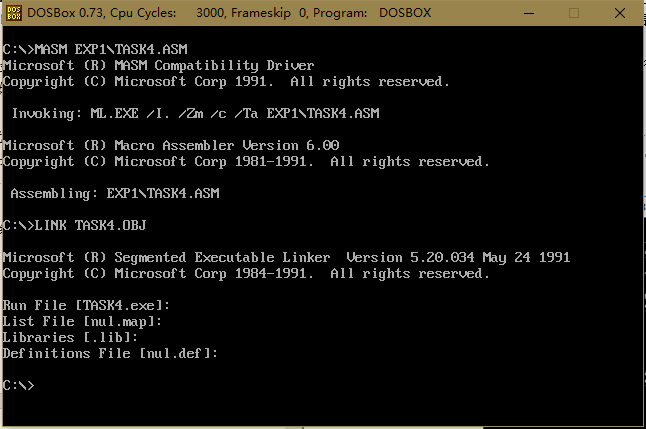


图3.1.1 正确编译连接

1. 输入TD TASK4，进入调试界面，调试两步后，观察DS段,如图3.1.2所示

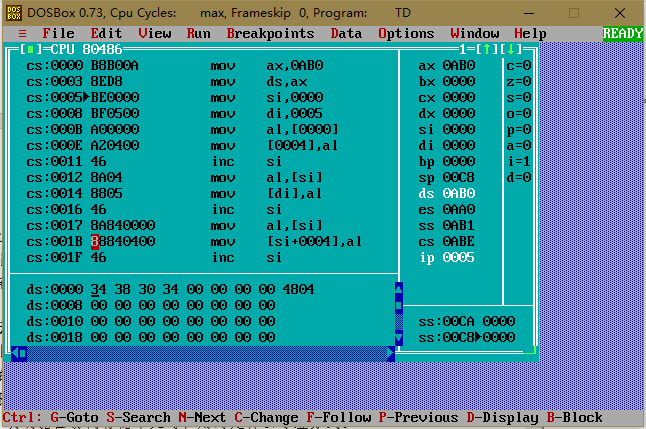


图3.1.2 调试两步后，DS段存储情况

1. 继续调试，将第一个数字以寄存器寻址的方式存储，如图3.1.3所示

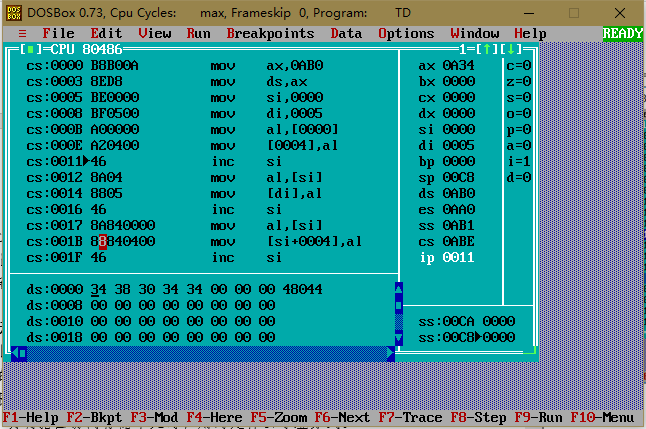


图3.1.3 以寄存器寻址方式存储

1. 继续调试，将第二个数字以寄存器间接寻址的方式存储，如图3.1.4所示

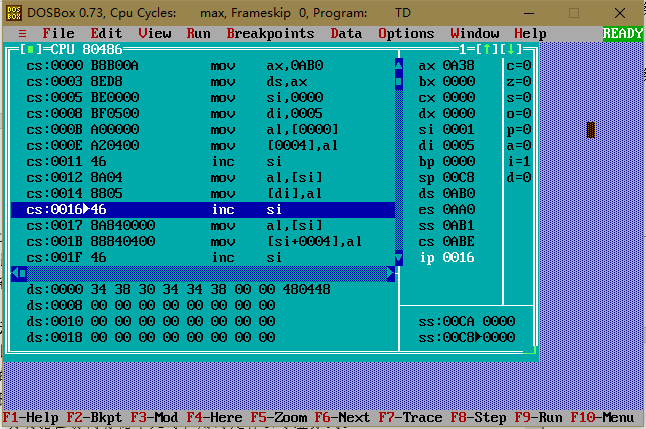


图3.1.4 以寄存器间接寻址方式存储

1. 继续调试，将第三个数字以变址寻址的方式存储，如图3.1.5所示

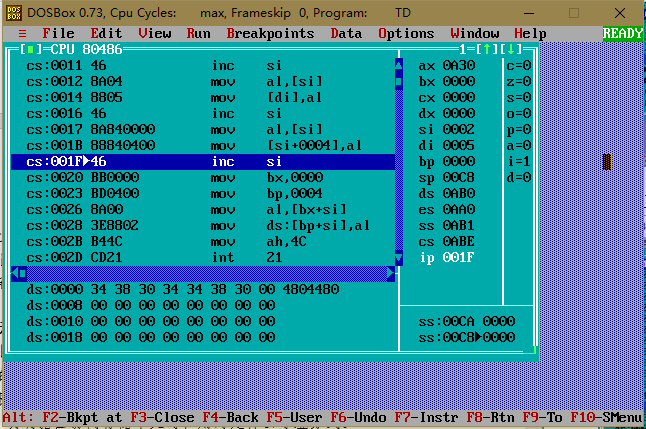


图3.1.5 以变址寻址方式存储

继续调试，将第四个数字以基址加变址的方式进行存储，如图3.1.6所示

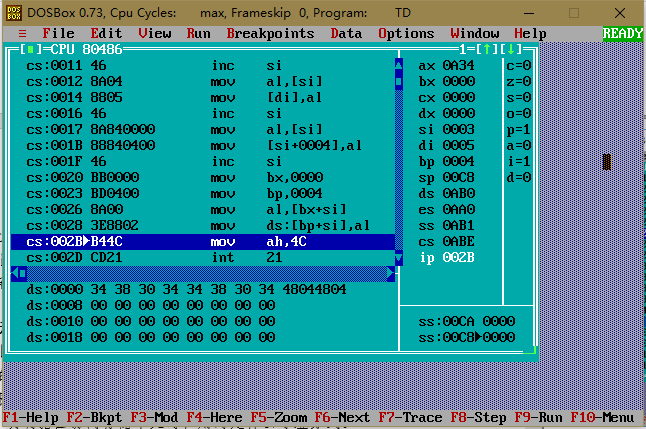


图3.1.5 以基址加变址方式存储

## 3.5任务5

### 3.5.1源程序

.386

;--------------------------------------------------

DATA SEGMENT USE16

in\_name DB 11

DB ?

DB 11 DUP(0)

in\_pwd DB 7

DB ?

DB 7 DUP(0)

goods\_name DB 11

DB ?

DB 11 DUP(0)

PR1 DW 0

PR2 DW 0

APR DW 0

AUTH DB 0

COST1 DW 0;成本

PROFIT1 DW 0;利润

COST2 DW 0;成本

PROFIT2 DW 0;利润

BNAME DB 'LONG JQ',3 DUP(0);老板姓名

BPASS DB 'NOPASS';密码

N EQU 30

SHOP1 DB 'SHOP1',0;网店名称，0结束

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,?,?)

SHOP2 DB 'SHOP2',0;网店名称，0结束

GB2 DB 'BOOK',6 DUP(0)

DW 12,28,20,15,?

GB1 DB 'PEN',7 DUP(0)

DW 35,50,30,24,?

GBN DB N-2 DUP('TEMP-VALUE',15,0,20,0,30,0,2,0,?,?)

INPUT\_NAME\_MSG DB 0AH,0DH,'please input name(input q/Q to exit):$'

INPUT\_PASS\_MSG DB 0AH,0DH,'please input password:$'

LOGIN\_FAILED\_MSG DB 0AH,0DH,'Login failed! Please check the name or the password!$'

END\_MSG DB 0AH,0DH,'end of program, press any key to continue...$'

INPUT\_GOODS\_NAME\_MSG DB 0AH,0DH,'please input the name of the goods:$'

INPUT\_GOODS\_NAME\_AGAIN DB 0AH,0DH,'goods name input error,please input again:$'

GRADE\_A\_MSG DB 0AH,0DH,'A$'

GRADE\_B\_MSG DB 0AH,0DH,'B$'

GRADE\_C\_MSG DB 0AH,0DH,'C$'

GRADE\_D\_MSG DB 0AH,0DH,'D$'

GRADE\_F\_MSG DB 0AH,0DH,'F$'

DATA ENDS

;---------------------------------------------------

STACK SEGMENT STACK

DB 200 DUP(0)

STACK ENDS

;---------------------------------------------------

CODE SEGMENT USE16

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

START:MOV AX,DATA

MOV DS,AX

FUNCION1:MOV SI,0;计数器

MOV CX,10;计数器

SET\_ALL\_ZERO:MOV AL,0

MOV in\_name[SI+2],AL

MOV in\_pwd[SI+2],AL

INC SI

DEC CX

CMP SI,6

JNE SET\_ALL\_ZERO

SET\_NAME\_ALL\_ZERO:MOV AL,0

MOV in\_name[SI+2],AL

INC SI

DEC CX

JNZ SET\_NAME\_ALL\_ZERO

INPUT\_NAME:LEA DX,INPUT\_NAME\_MSG

MOV AH,9

INT 21H

LEA DX,in\_name

MOV AH,10

INT 21H

MOV SI, WORD PTR in\_name[1]

AND SI,00FFH

MOV in\_name[SI+2],0

MOV AL,0

CMP AL,in\_name[3]

JE AUTH\_FAIL

INPUT\_PWD:LEA DX,INPUT\_PASS\_MSG

MOV AH,9

INT 21H

LEA DX,in\_pwd

MOV AH,10

INT 21H

MOV SI, WORD PTR in\_pwd[1]

AND SI,00FFH

MOV in\_pwd[SI+2],0

MOV SI,0;计数器

MOV CX,10;计数器

FUNCTION2\_PWD:MOV AL,in\_name[SI+2]

CMP AL,BNAME[SI]

JNE PRINT\_LOGIN\_FAILED

MOV AL,in\_pwd[SI+2]

CMP AL,BPASS[SI]

JNE PRINT\_LOGIN\_FAILED

INC SI

DEC CX

CMP SI,6

JNE FUNCTION2\_PWD

FUNCTION2\_NAME:MOV AL,in\_name[SI+2]

CMP AL,BNAME[SI]

JNE PRINT\_LOGIN\_FAILED

INC SI

DEC CX

JNZ FUNCTION2\_NAME

AUTH\_SUCCESS:MOV BH,1

MOV BYTE PTR AUTH,BH

JMP FUNCTION3

AUTH\_FAIL:MOV AL,'q'

CMP AL,in\_name[2]

JE EXIT

MOV AL,'Q'

CMP AL,in\_name[2]

JE EXIT

MOV AL,0

CMP AL,in\_name[2]

JNE INPUT\_PWD

MOV BH,0

MOV BYTE PTR AUTH,BH

JMP FUNCTION3

PRINT\_LOGIN\_FAILED:LEA DX,LOGIN\_FAILED\_MSG

MOV AH,9

INT 21H

JMP FUNCION1

FUNCTION3:MOV SI,0

MOV CX,10

SET\_GOODSNAME\_ALL\_ZERO:MOV AL,0

MOV goods\_name[SI+2],AL

INC SI

DEC CX

JNZ SET\_GOODSNAME\_ALL\_ZERO

LEA DX,INPUT\_GOODS\_NAME\_MSG

MOV AH,9

INT 21H

LEA DX,goods\_name

MOV AH,10

INT 21H

MOV SI, WORD PTR goods\_name[1]

AND SI,00FFH

MOV goods\_name[SI+2],0

MOV BX,0

MOV AL,0

CMP AL,goods\_name[2];判断是否只输入了回车

JNE IS\_GOODS

JMP FUNCION1

IS\_PEN:MOV AL,goods\_name[SI+2]

CMP AL,GA1[SI]

JNE IS\_GOODS

INC SI

DEC CX

JNZ IS\_PEN

MOV BYTE PTR goods\_name[SI+3],'$'

MOV AX,WORD PTR GA1[10];进货价

IMUL GA1[14];AX现为cost1

MOV COST1,AX

MOV AX, WORD PTR GA1[12];销售价

IMUL GA1[16];AX现为profit1+COST1

SUB AX,COST1;AX现为profit1

MOV PROFIT1,AX

MOV AX,WORD PTR GB1[10];进货价

IMUL GB1[14];AX现为cost2

MOV COST2,AX

MOV AX,WORD PTR GB1[12];销售价

IMUL GB1[16];AX现为profit2+cost2

SUB AX,COST2

MOV PROFIT2,AX

CMP AUTH,0

JE FUNCTION3\_4

JMP FUNCTION3\_3

IS\_BOOK:MOV AL,goods\_name[SI+2]

CMP AL,GA2[SI]

JNE IS\_GOODS

INC SI

DEC CX

JNZ IS\_BOOK

MOV BYTE PTR goods\_name[SI+3],'$'

MOV AX,WORD PTR GA2[10];进货价

IMUL GA2[14];AX现为cost1

MOV COST1,AX

MOV AX,WORD PTR GA2[12];销售价

IMUL GA2[16]

SUB AX,COST1;AX现为profit1

MOV PROFIT1,AX

MOV AX,WORD PTR GB2[10];进货价

IMUL GB2[14];AX现为cost2

MOV COST2,AX

MOV AX,WORD PTR GB2[12];销售价

IMUL GB2[16]

SUB AX,COST2;AX现为profit2

MOV PROFIT2,AX

CMP AUTH,0

JE FUNCTION3\_4

JMP FUNCTION3\_3

IS\_N:MOV AL,goods\_name[SI+2]

CMP AL,GAN[SI]

JNE IS\_GOODS

INC SI

DEC CX

JNZ IS\_PEN

MOV BYTE PTR goods\_name[SI+3],'$'

MOV AX,WORD PTR GAN[10];进货价

IMUL GAN[14];AX现为cost1

MOV COST1,AX

MOV AX,WORD PTR GAN[12];销售价

IMUL GAN[16]

SUB AX,COST1;AX现为profit1

MOV PROFIT1,AX

MOV AX,WORD PTR GB1[10];进货价

IMUL GB1[14];AX现为cost2

MOV COST2,AX

MOV AX,WORD PTR GB1[12];销售价

IMUL GB1[16]

SUB AX,COST2;AX现为profit2

MOV PROFIT2,AX

CMP AUTH,0

JE FUNCTION3\_4

JMP FUNCTION3\_3

IS\_GOODS:MOV SI,0;计数器

MOV CX,10;计数器

INC BX

CMP BX,1

JE IS\_PEN

CMP BX,2

JE IS\_BOOK

CMP BX,3

JE IS\_N

CMP BX,4

JE FUNCTION3\_1

FUNCTION3\_1:LEA DX,INPUT\_GOODS\_NAME\_AGAIN

MOV AH,9

INT 21H

JMP FUNCTION3

FUNCTION3\_3:MOV AX,PROFIT1

IMUL AX,WORD PTR 10

MOV DX,0

IDIV COST1

MOV AH,0

MOV PR1,AX

MOV AX,PROFIT2

IMUL AX,WORD PTR 10

MOV DX,0

IDIV COST2

MOV AH,0

MOV PR2,AX

ADD AX,PR2

MOV APR,AX

CMP APR,18

JNL GRADE\_A

CMP APR,10

JNL GRADE\_B

CMP APR,4

JNL GRADE\_C

CMP APR,0

JNL GRADE\_D

JMP GRADE\_F

GRADE\_A:LEA DX,GRADE\_A\_MSG

MOV AH,9

INT 21H

JMP FUNCION1

GRADE\_B:LEA DX,GRADE\_B\_MSG

MOV AH,9

INT 21H

JMP FUNCION1

GRADE\_C:LEA DX,GRADE\_C\_MSG

MOV AH,9

INT 21H

JMP FUNCION1

GRADE\_D:LEA DX,GRADE\_D\_MSG

MOV AH,9

INT 21H

JMP FUNCION1

GRADE\_F:LEA DX,GRADE\_F\_MSG

MOV AH,9

INT 21H

JMP FUNCION1

FUNCTION3\_4:MOV DL,0AH

MOV AH,2

INT 21H

MOV DL,0DH

MOV AH,2

INT 21H

LEA DX,goods\_name[2]

MOV AH,9

INT 21H

JMP FUNCION1

EXIT:LEA DX,END\_MSG

MOV AH,9

INT 21H

MOV AH,1

INT 21H

MOV AH,4CH

INT 21H

CODE ENDS

END START

### 3.5.2实验步骤

1. 准备上机实验环境。

2. 使用记事本编辑录入源程序，存盘文件名为TASK5.ASM。使用MASM 6.0汇编源文件。即masm TASK5.ASM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TASK5.OBJ文件连接成可执行文件。即link TASK5.OBJ；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TASK5.EXE文件。

4. 使用调试器TD.EXE对生成的文件进行调试。

### 3.5.3实验记录与分析

1. 实验环境条件：AMD A8-8600P 1.6GHz，8G内存；WINDOWS 10下DOSBox0.73；EDIT.EXE 2.0；MASM.EXE 6.0； LINK.EXE 5.2; TD.EXE 5.0。
2. 编译连接正常，如图3.5.1所示

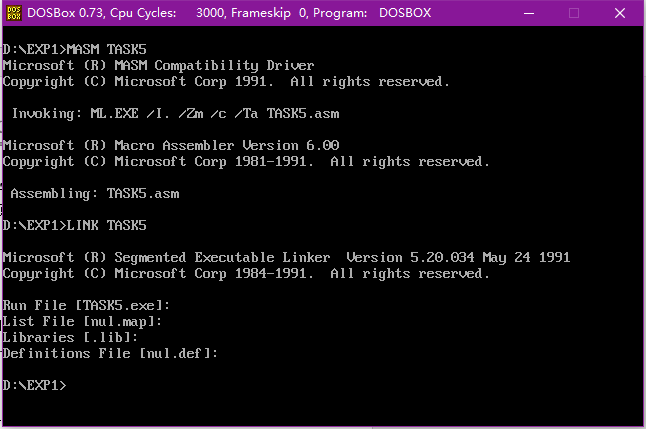


图3.5.1 正确编译连接

1. 输入q/Q，程序退出，如下图3.5.2所示：

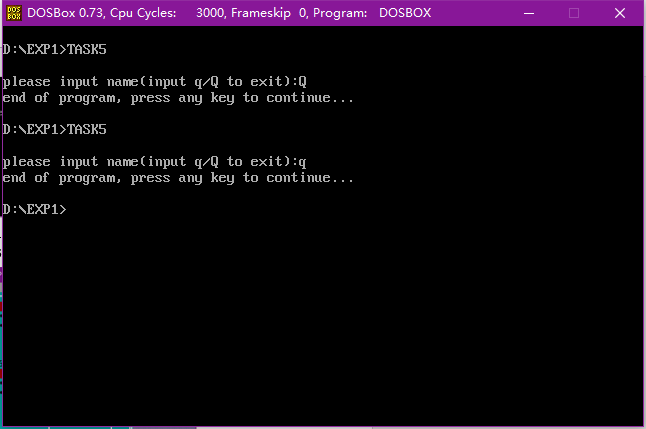


图3.5.2 程序退出

1. 输入不匹配name和password，提示登录失败，如下图3.5.3所示：

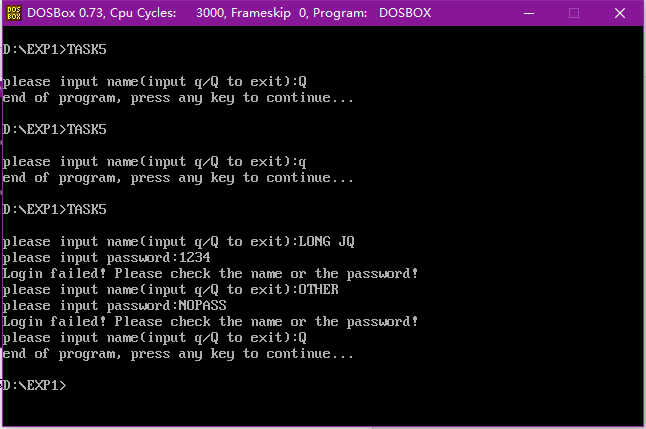


图3.5.3 Login Failed

1. 直接输入回车进入游客系统，只显示商品名称不提示利润率，如下图3.5.4：

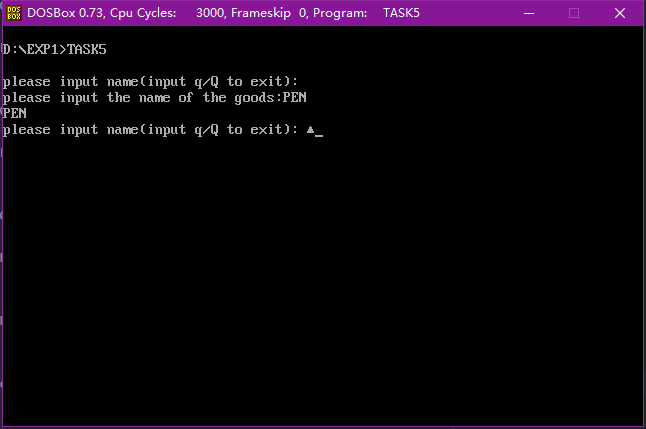


图3.5.4 进入游客系统

1. 输入正确的name和password，进入系统，如下图3.5.5：

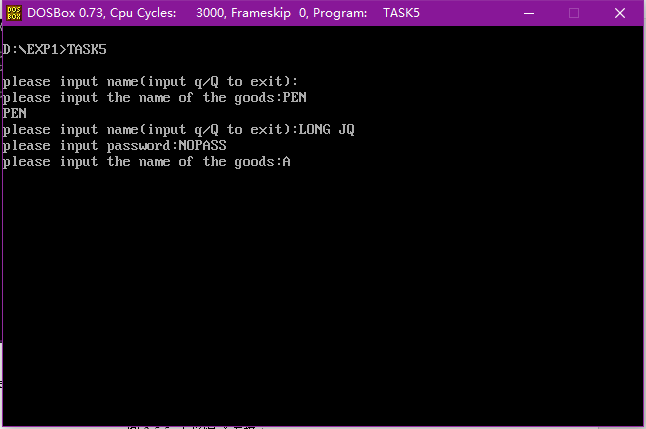


图3.5.5 正确输入name和password

1. 输入商品名称PEN，输出利润率等级，如下图3.5.6：

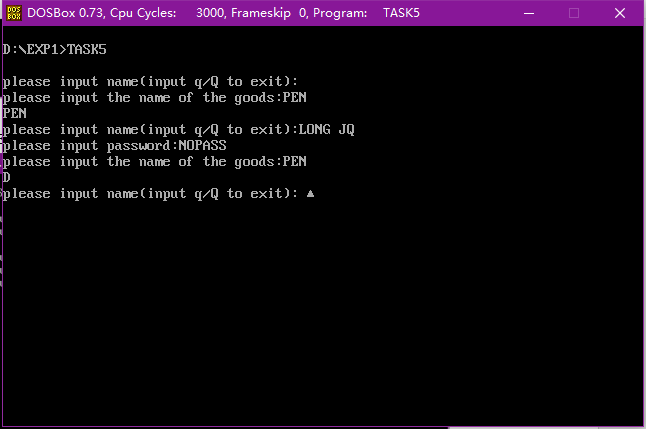


图3.5.6 输出PEN的利润率等级

1. 输入商品名称BOOK，输出利润率等级，如下图3.5.7：

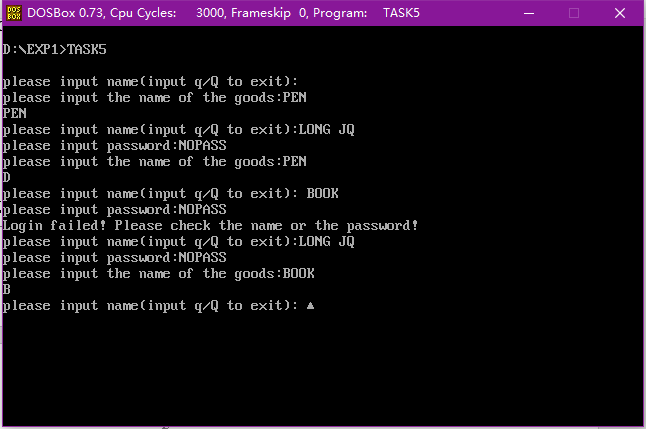


图3.5.7 输出book的利润率等级

1. 输入不正确的商品名称，输出错误提示，如下图3.5.8：

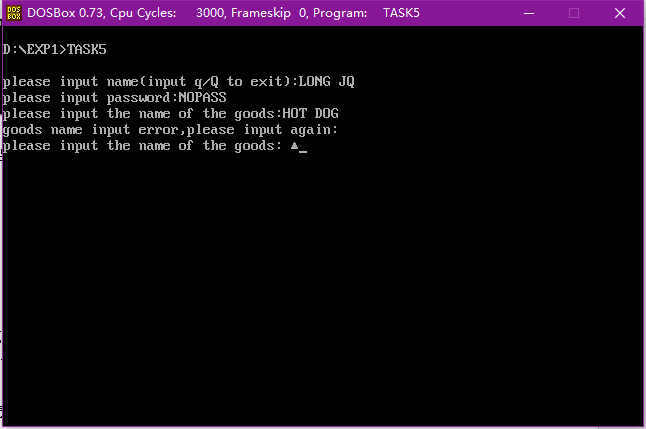


图3.5.8 输出错误提示

# 总结与体会

这次上机实验在拖拖沓沓中迷茫地完成了，首先，非常感谢助教和老师耐心的教导。由于首次接触汇编语言，上机的时候才只会基本的DOS命令和MASM的使用，对于调试器TD的认知基本为零，所以我的第一次上机实验，一边在吐槽TD使用上的不友好，同时也在不断地去百度怎么使用TD，然而百度上说的怎么会有老师和助教说得好，百度的说法参差不齐，没有一个标准的答案，也因此浪费了大量时间。

经过老师和助教手把手教学后，终于对调试器TD有了大概的认识，也掌握了一些基本的技巧，终于不再迷茫了，这个时候才把理论课上朱虹老师说的那些知识点串联起来，六种寻址方式啊，数据段的定义啊，标志寄存器啊，这些东西就一一浮上脑海，然后发现捣鼓了TD一番之后，发现之前对TD的吐槽确实是有点对他不公平，这个TD调试器的功能很强大，用着用着竟有似曾相识的感觉，转头一想TD在功能集成方面竟和Emacs惊人的相似，我吐槽TD他不友好，确实是我使用方法不对，算是冤枉了他。

在做任务五的时候，我深深地体会到计算机资源的可贵性和重要性，我们在做字符串处理时，要非常精确的判断边界条件，在进行算术运算时，要尽可能的避免溢出。站在巨人的肩膀上更加能体会到巨人的伟大，当我们以后用高级语言进行软件开发时，我想每每想到那些被汇编语言支配的恐惧，我们应该都会对我们的前辈怀着一份敬畏之心，他们一层一层地建起了操作系统、编译系统和软件系统。我想如果我两年后还能有幸继续接受教育进入科研领域，我觉得我一定会保持极高的热忱。

俗话说得好，一个良好的开头意味着成功的一半，我们已经掌握了学习汇编语言的工具，这是一个良好的开始，愿我们以后在学习汇编的道路上一马平川。

# 注

有个别实验代码较多较长，不适于直接在word中直接操作。因此本人将本次实验所有代码及二进制可执行文件上传至GitHub，具体网址为<https://github.com/DragonDriver/AssemblyExp/tree/master/Exp1>。

# 参考文献

[1] 王元珍 曹忠升 韩宗芬.《80x86汇编语言程序设计》“第二章 寻址方式”.华中科技大学出版社