

**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： 2018-3-26，14：00-17：30 实验地点： 南一楼804室90号实验台**

**2018-4-2，14：00-17：30**

**指导教师： 李海波**

**专业班级：校交 201601班**

**学 号： U201612696 姓 名： 陈淏睿**

**同组学生： 无 报告日期： 2018年 4 月 3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：2018.4.3

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](#_Toc515139489)

[2 实验内容 1](#_Toc515139490)

[3 实验过程 3](#_Toc515139492)

[3.1 任务1 3](#_Toc515139493)

[3.1.1 实验步骤 3](#_Toc515139494)

[3.1.2 实验记录与分析 4](#_Toc515139495)

[3.2 任务2 5](#_Toc515139496)

[3.2.1 实验步骤 5](#_Toc515139497)

[3.2.2 源程序 6](#_Toc515139498)

[3.2.3 实验记录与分析 7](#_Toc515139499)

[3.3 任务3 9](#_Toc515139500)

[3.3.1 实验步骤 9](#_Toc515139501)

[3.3.2 源程序 10](#_Toc515139502)

[3.3.3 实验记录与分析 11](#_Toc515139503)

[3.4 任务4 14](#_Toc515139504)

[3.4.1 实验步骤 14](#_Toc515139505)

[3.4.2 源程序 14](#_Toc515139506)

[3.4.3 实验记录与分析 15](#_Toc515139507)

[3.5 任务5 17](#_Toc515139508)

[3.5.1 设计思想及存储单元分配 17](#_Toc515139509)

[3.5.2 流程图 18](#_Toc515139510)

[3.5.3 源程序 20](#_Toc515139511)

[3.5.4 实验步骤 27](#_Toc515139512)

[3.5.5 实验记录与分析 27](#_Toc515139513)

[4 总结与体会 30](#_Toc515139514)

[参考文献 34](#_Toc515139515)

# 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

## 任务1

### 实验步骤

1. 准备上机实验环境。

2. 在TD的代码窗口中的当前光标下输入运算式对应的两个8位数值对应的指令语句MOV AL,（第一个操作数）；ADD AL,（第二个操作数）；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。

(1)中预计ADD执行之后（AH）=8D00H SF=1、OF=1、CF=0、ZF=0

(2)中预计ADD执行之后（AH）=7A00H SF=0、OF=1、CF=1、ZF=0

(3)中预计ADD执行之后（AH）=0800H SF=0、OF=0、CF=1、ZF=0

重复上述过程，将剩下几个表达式计算完毕，比较结果。

3. 输入MOV AH,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。

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU 2.60GHz，2.86G内存；WINDOWS 7下DOSBox0.74； notepad++ 7.55;MASM.EXE 6.0;LINK.EXE 5.2;TD.EXE 5.0。

2. 输入指令MOV AH,+0110011B，ADD AH, +1011010B。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（8D00H）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。第一次接触TD时，我尚不知道如何在执行完语句后退出TD返回DOSbox主界面，因此只能强制退出虚拟机再重新进入。请教同学后得知可以在程序段尾添加结束运行语句MOV AH,4CH和INT 21H，也可直接INT 21H返回DOS界面。

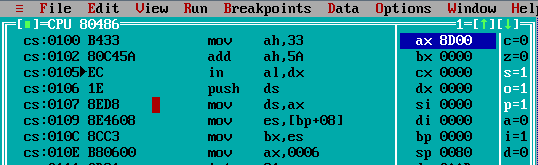


图3.1.1第一个算式执行语句后的状态

3．输入指令MOV AH,-0101001B，ADD AH, -1011101B。执行三条指令后的结果如图3.1.2所示。计算结果在AX的高字节中（7A00H）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。请教同学后，我明白了要返回执行程序的第一条语句时可以修改ip存放的地址为第一条命令的位置，顺带明白了ip对于保证程序有序运行的意义。

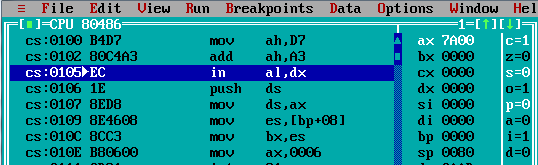


图3.1.2第二个算式执行语句后的状态

4．输入指令MOV AH,+1100101B，ADD AH, -1011101B。执行三条指令后的结果如图3.1.3所示。计算结果在AX的高字节中（0800H）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。

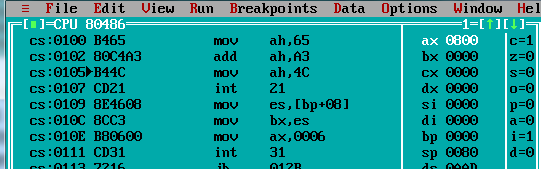


图3.1.3第三个算式执行语句后的状态

5. 输入MOV AH,10H；MOV AL,-5H；SUB AH,AL,发现ax高八位变为ah与al之差而低八位不变，sf=0,原因是运算后ax最高位仍为0.

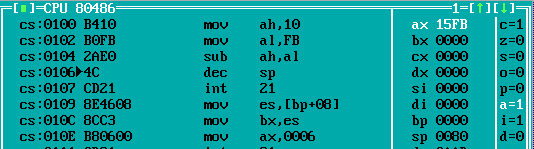


图3.1.4第一个算式执行语句后的状态

6. 输入MOV AH,0FFH；MOV AL,-5H；SUB AH,AL，发现ax高八位变为ah与al之代数和而低八位不变，sf=0,原因是运算后ax最高位由1变为0.

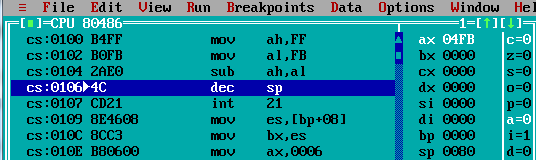


图3.1.5第二个算式执行语句后的状态

## 任务2

### 实验步骤

1. 准备上机实验环境。

2. 使用notepad++录入源程序，存盘文件名为task2.ASM。使用MASM 6.0汇编源文件。即MASM task2；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task2.OBJ文件连接成执行文件。即LINK task2；

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task2.EXE文件。

4. 打开TD.exe,从TD中打开task2.exe。

5．在两目标语句处设置断点，单步调试程序至断点处，观察目标值。即在命令行提示符后输入task2后回车，观察执行现象。

6.用goto语句，输入ds:0，查看数据段前40个字节的内容，预计为：

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

7. 为目标字符串添加存储空间，并在LOPA标号前添加语句：

LEA DX,STR

MOV AH,9

INT 21H

MOV AH,1

INT 21H

汇编，连接，运行，观察实验结果。

### 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)

MSTR DB 'Press any key to begin!$' ;为字符串分配空间

DATA 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,MSTR ;输出字符串

MOV AH,9

INT 21H

MOV AH,1 ;输入任意字符

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

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU 2.60GHz，2.86G内存；WINDOWS 7下DOSBox0.74； notepad++ 7.55;MASM.EXE 6.0;LINK.EXE 5.2;TD.EXE 5.0。

2. 汇编并连接源程序，生成相应的task2.obj与task2.exe。

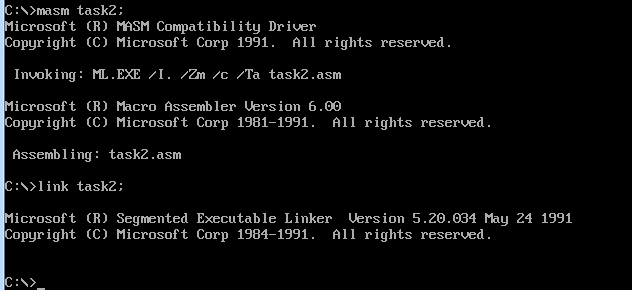


图3.2.1汇编，连接

3. 用TD单步执行程序，在MOV CX，10和INT 21H处设置断点。

4. F7单步执行至MOV CX,10处，结果如图3.2.2所示。可见SI,DI,BX,BP作为四段长为10的空间的头指针，其地址各相差10.

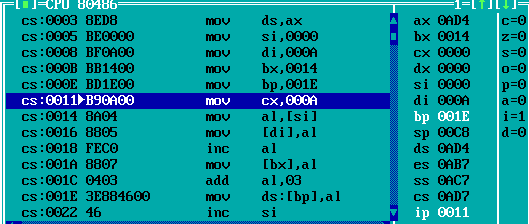


图3.2.2执行至MOV CX,10处的结果

5. 单步执行至INT 21H处，结果如图3.2.3所示。可见SI,DI,BX,BP各加了10，其地址仍各相差10.请教同学后，发现可用F5键扩大窗口，便于截图。

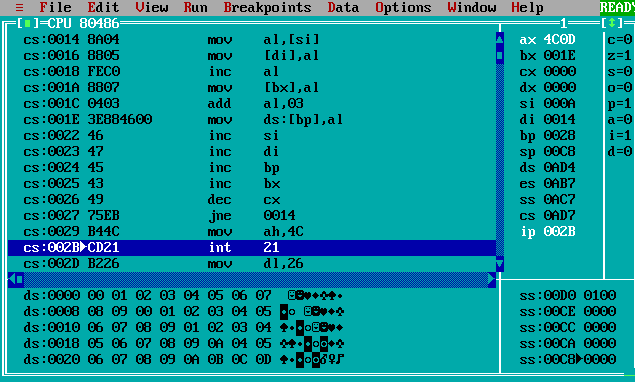


图3.2.3执行至INT 21H处的结果

6. 用goto查看ds:0后数据段头40个字节的内容如图3.2.3所示。BUF1段中各元素被直接复制进BUF2段，加1后被复制进BUF3段，又加3后被复制进BUF4段，因此BUF1，BUF2，BUF3，BUF4各段中相同位置元素关系：BUF1=BUF2=BUF3-1=BUF4-4。

7. 修改BUF2，在数据段中新定义字符串：STR DB 'Press any key to begin!$'。第一次定义时忘记在字符串后加’$’导致输出乱码。在循环前添加语句：

LEA DX,STR

MOV AH,9

INT 21H

MOV AH,1

INT 21H

保存task2.asm,汇编,连接,执行，观察输出情况。



图3.2.4运行情况

## 任务3

### 实验步骤

1. 准备上机实验环境。

2. 在第二题的基础上删除数据段定义时的USE16， 并将所有寄存器寻址相关寄存器由16位改为32位，存盘文件名为task3-32.ASM。使用MASM 6.0汇编源文件。即MASM task3-32.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task3-32.OBJ文件连接成执行文件。即LINK task3-32；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task3-32.EXE文件。

4. 新建task3-bz.asm,在task3-32.asm的代码基础上删除所有的寄存器自增语句，并用计数器为基础，用一尚未使用的32位寄存器设计偏移地址。使用变址寻址方式修改所有MOV语句。

5. 对task3-bz.asm重复过程2-3，生成对应的task3-bz.obj和task3-bz.exe文件。

6. 使用TD.EXE观察task3-bz的执行情况。即 TD task3-bz；回车。单步执行置INT 21H处，用goto语句观察数据段前40个字节的内容。设想其内容与任务二中完全一致。

7. 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。将其机器指令分别截图。

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

### 源程序

task3-32.sam

.386

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

STACK SEGMENT STACK USE16

DB 200 DUP(0)

STACK ENDS

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

DATA SEGMENT

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,OFFSET BUF1

MOV EDI,OFFSET BUF2

MOV EBX,OFFSET BUF3

MOV EBP,OFFSET BUF4

MOV CX,10

LOPA: MOV EAX,[ESI]

MOV [EDI],AL

INC AL

MOV [EBX],AL

ADD AL,3

MOV DS:[EBP],AL

INC ESI

INC EBP

INC EBX

INC EDI

DEC CX

JNZ LOPA

MOV AH,4CH ;exit

INT 21H

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

CODE ENDS

END START

task3-bz.asm

.386

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

STACK SEGMENT STACK USE16

DB 200 DUP(0)

STACK ENDS

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

DATA SEGMENT

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,OFFSET BUF1

MOV EDI,OFFSET BUF2

MOV EBX,OFFSET BUF3

MOV EBP,OFFSET BUF4

MOV ECX,10

LOPA: MOV EDX,10

SUB EDX,ECX ;//修改存储空间统一的偏移地址

MOV AL,[ESI]+EDX ;//设计错误，正确语句应为MOV AL,[ESI+EDX]

MOV [EDI]+EDX,AL ;//设计错误，正确语句应为MOV AL,[EDI+EDX]

INC AL

MOV [EBX]+EDX,AL ;//设计错误，正确语句应为MOV AL,[EBX+EDX]

ADD AL,3

MOV DS:[EBP+EDX],AL ;//设计错误，正确语句应为MOV MOV DS:[EBP+EDX],AL

DEC CX ;计数器自减

JNZ LOPA

MOV AH,4CH ;exit

INT 21H

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

CODE ENDS

END START

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU 2.60GHz，2.86G内存；WINDOWS 7下DOSBox0.74； notepad++ 7.55;MASM.EXE 6.0;LINK.EXE 5.2;TD.EXE 5.0。

2. 新建task3-32.asm,在task2.asm的基础上将删除数据段定义时的USE16， 并将所有寄存器寻址相关寄存器由16位改为32位。

3. 汇编并连接task3-32.asm，发现生成相应的.obj及.exe文件。



图3.3.1汇编并连接task3-32.asm



图3.3.2对应的.obj和.exe文件

4. 汇编task3-bz.asm，发现报错。按照提示，对对应行仔细检查，发现了错误原因。见源程序中“//”后的说明。

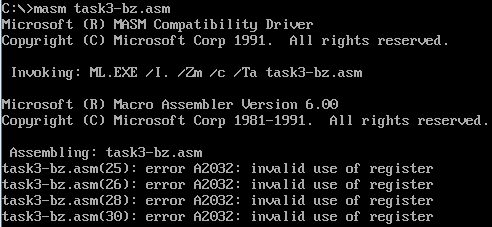


图3.3.3汇编task3-bz.asm报错

5. 修改完毕后，汇编并连接task3-bz.asm，发现生成相应的.obj及.exe文件。

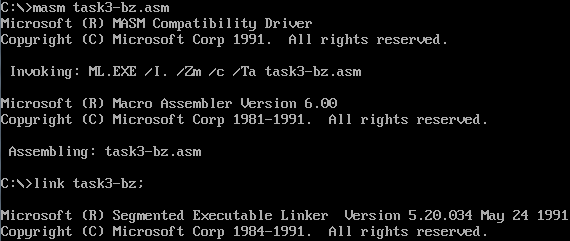


图3.3.4汇编并连接task3-bz.asm

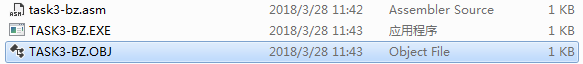
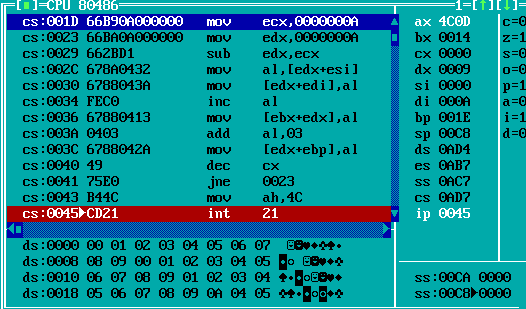


图3.3.5对应的.obj和.exe文件

6. 用TD单步执行至退出程序前，用goto语句访问数据段前40字节的内容，结果如图3.3.6与3.3.7所示。可以发现其内容与任务二中完全相同。



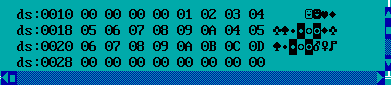


图3.3.6&图3.3.7 ds中前40个字节的内容

7.图3.3.6中左起第一列为代码段cs名称,第二列为地址，第三列即操作指令机器码。

8. 重新开始调试，在一重循环到jnx语句时直接在ip处修改待调用指令的地址为0043。确认后继续操作，发现程序本应在回到地址为0023处继续执行却跳出了循环。再次观察ds，与执行一次循环后得到的结果吻合。可见ip提示下一条预定执行语句的位置，对于保证程序按设计者规定的流程执行具有重要意义。

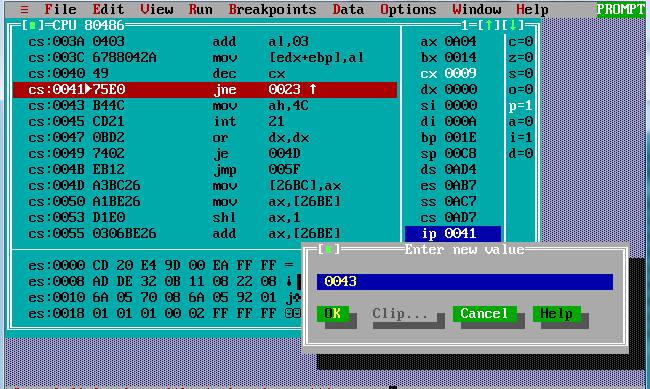
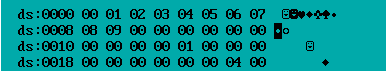


图3.3.8修改下一条语句的地址



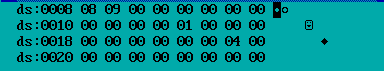


图3.3.9&3.3.10程序结束时数据段中前40字节内容的变化

## 任务4

### 实验步骤

1. 准备上机实验环境。

2. 使用notepad++录入源程序，存盘文件名为task4.ASM。

3. 使用MASM 6.0汇编源文件。即MASM task4；观察提示信息，若出错，则用编辑程序修改错误，盘后重新汇编，直至不再报错为止。

4. 使用连接程序LINK.EXE将汇编生成的task4.OBJ文件连接成执行文件。即LINK task4；

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CUBE.EXE文件。

5. 用TD单步调试该程序，观察执行现象。针对逻辑错误进行修改，直至对于各种输入可得出正确的执行结果。

### 源程序

Task4.asm

.386

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

STACK SEGMENT STACK USE16

DB 20 DUP(0)

STACK ENDS

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

DATA SEGMENT USE16

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 BX,OFFSET XUEHAO

MOV CL,'2'

MOV [BX],CX ;BX用寄存器间接寻址，CX用寄存器寻址

MOV CL,'6'

MOV AL,1

MOV DS:[AX],CX ;//寄存器间接寻址用16为寄存器时只能用BX,BP,SI,DI之一，此处连同上一行一并改为SI

MOV CL,'9'

MOV [BX+2],CX ;变址寻址

MOV CL,'6'

MOV SI,1

MOV [BX+SI+2],CX ;基址加变址寻址

MOV AH,4CH ;exit

INT 21H

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

CODE ENDS

END START

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU 2.60GHz，2.86G内存；WINDOWS 7下DOSBox0.74； notepad++ 7.55;MASM.EXE 6.0;LINK.EXE 5.2;TD.EXE 5.0。

2. 新建task4.asm,为XUEHAO分配空间，分别按四种寻址方式向XUEHAO开头的空间插入数据。

3.在第一次汇编时，发现报错如图3.4.1。



图3.4.1汇编时报错

按照提示，对对应行仔细检查，发现了错误原因。见源程序中“//”后的说明。

4. 汇编并连接，生成相应的.obj和.exe文件。

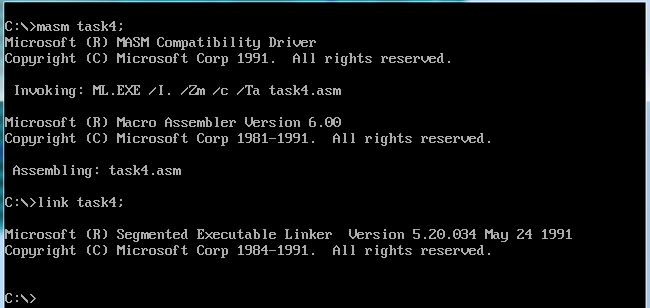


图3.4.2汇编及连接过程

5. 用TD单步执行程序，单步调试至放入第一个字符。

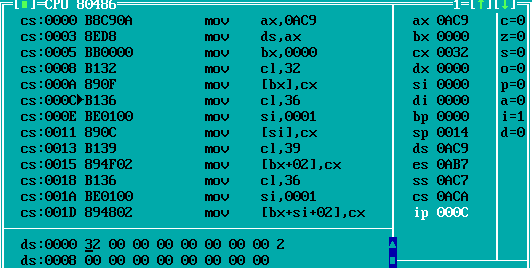


图3.4.3放入第一个字符

6.单步调试至放入第二个字符。

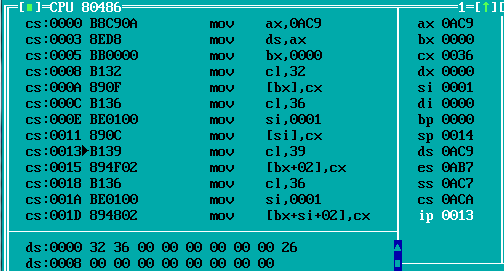


图3.4.4放入第二个字符

7.单步调试至放入第三个字符。

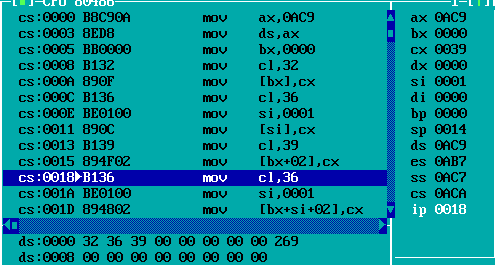


图3.4.5放入第三个字符

8.单步调试至放入第四个字符。四个字符均成功放入，证明程序运行得出了正确结果。

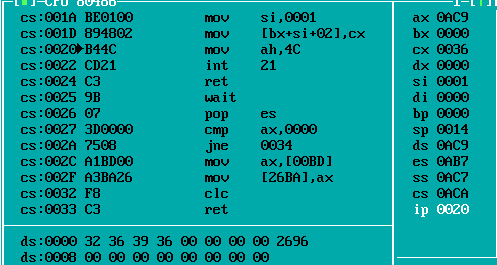


图3.4.6放入第四个字符

## 任务5

### 设计思想及存储单元分配

整个程序为一包含四个模块，且仅提供一种退出方式的循环体，其中模块一（输入登录用户的姓名与密码）与模块二（登录信息认证）联系较为紧密，模块三（计算指定商品的利润率）包含两个二重循环，模块四（利润率进行等级判断）为一分支结构。

1.存储单元分配

BNAME：给定的正确用户名，长12字节。

BPASS：给定的正确密码，长8字节。

S1:商店一名称，长6字节。

GA1：商店一中货物一（PEN）的信息，包含长为10字节的名称字符串及5个字类型数，共长20字节。后续所有标志商品类型的变量的存储内容类型及长度均与GA1相同。

GA2：商店一中货物二（BOOK）的信息。

GAN：商店一中剩余28个货物的占位变量，每个货物所占存储空间的内容及长度均与GA1相同，用DUP语句完成28个空间的分配。

S2:商店二名称，长6字节。

GB1：商店二中货物一（BOOK）的信息。

GB2：商店二中货物二（PEN）的信息。

GBN：商店二中剩余28个货物的占位变量，内容及长度同GAN。

TIP\_1：提示信息一：输入用户名

TIP\_2：提示信息二：输入密码

TIP\_3：提示信息三：输入待查商品名

TIP\_4：提示信息四：重新输入待查商品名

TIP\_5：提示信息五：以游客身份登录

TIP\_6：提示信息六：以店主身份登录

TIP\_7：提示信息七：用户名错误

TIP\_8：提示信息八：密码错误

IN\_NAME：存储输入的用户名，第一个字节存放串空间长度，第二个字节存放实际串长，之后12字节存放输入的用户名字符串。共14字节。

IN\_PWD：存储输入的密码，空间分派方案同IN\_NAME，不同处在于用8字节存密码字符串。共10字节。

IN\_GOOD：存储输入的商品名，空间分派方案同IN\_NAME，不同处在于用10字节存商品字符串。共12字节。

AUTH：1字节，存放登录系统方式

PR1：1字，存放利润一的地址。根据据此地址用寄存器间接寻址即可取回利润一。

2.寄存器分配

使用了所有16位数据寄存器除SP外所有16位指示器变址寄存器。其中三个指示器变址寄存器主要用于寻址，数据寄存器用于数据处理转移及承担系统调用。

### 流程图

图3.2.1是任务2求一个数的立方值的程序流程图。

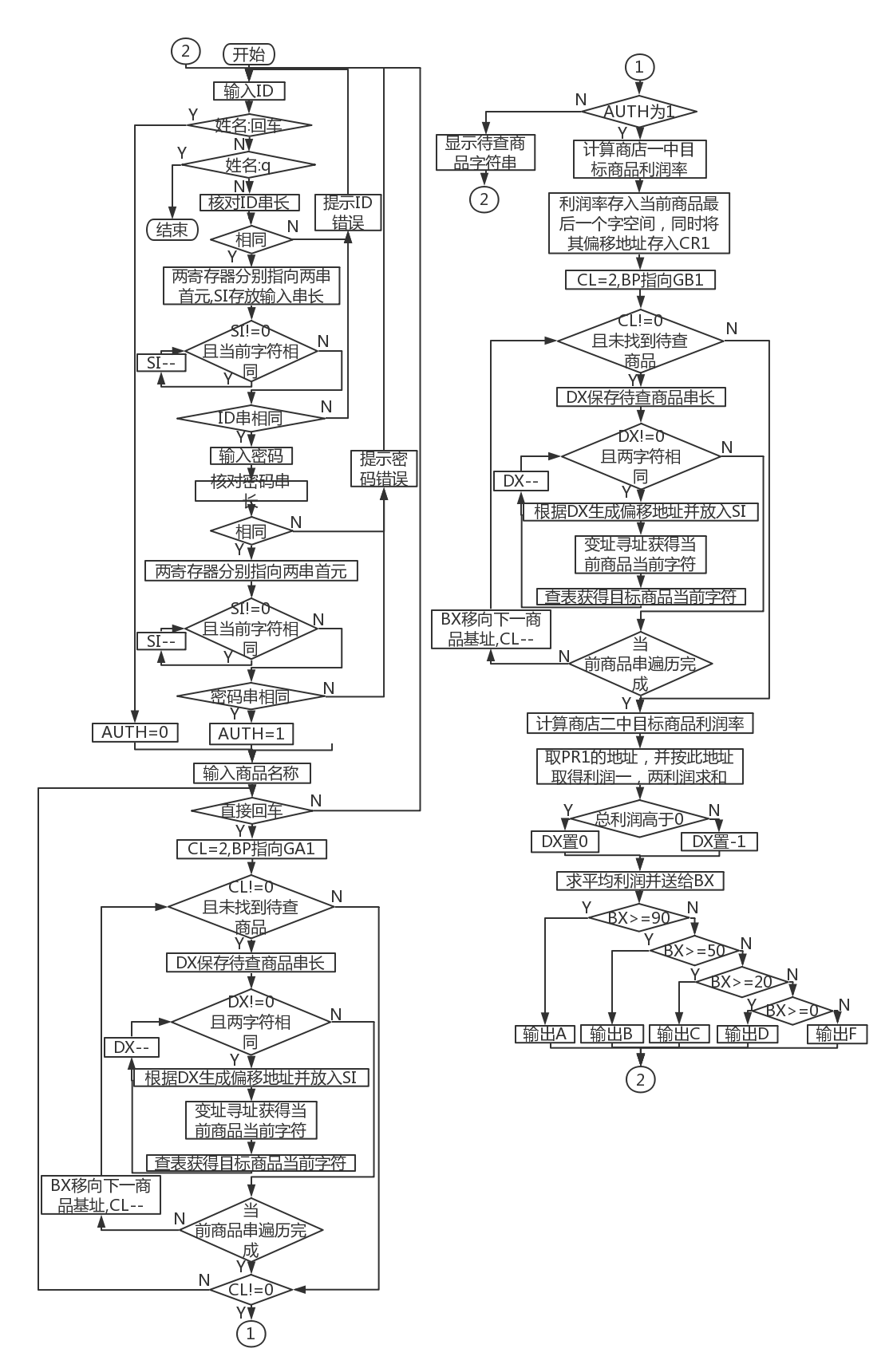


图3.5.1程序总体流程图

### 源程序

.386

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

STACK SEGMENT STACK USE16

DB 20 DUP(0)

STACK ENDS

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

DATA SEGMENT

BNAME DB 'CHEN HAORUI',0 ;老板姓名（必须是自己名字的拼音）

BPASS DB 'PASSWD',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

GBN 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。

TIP\_1 DB 'Enter ID:$'

TIP\_2 DB 'Enter PW:$'

TIP\_3 DB 'Input the commodity you want to look for:$'

TIP\_4 DB 'Input the commodity you want to look for again$'

TIP\_5 DB 'Login as visitor$'

TIP\_6 DB 'Login as manager$'

TIP\_7 DB 'ID incorrect$'

TIP\_8 DB 'password incorrect$'

IN\_NAME DB 12

DB ?

DB 12 DUP(0) ;存储输入的用户名

IN\_PWD DB 8

DB ?

DB 8 DUP(0) ;存储输入的密码

IN\_GOOD DB 10

DB ?

DB 10 DUP(0) ;存储输入的商品名称

AUTH DB ? ;判断登录方式

PR1 DW ? ;商店一中的利润地址

DATA ENDS

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

CODE SEGMENT USE16

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

START:

MOV AX,DATA

MOV DS,AX

STEP1:

LEA DX,TIP\_1 ;显示提示：输入用户名

MOV AH,9

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

LEA DX,IN\_NAME ;输入用户名

MOV AH,10

INT 21H

MOV BL,0 ;单独输入回车符时真实串长为0

LEA DI,IN\_NAME

INC DI ;找到真实串长

CMP BL,[DI]

JE BAR1

JMP BRK8

BAR1:

MOV AUTH,0 ;认证方式:顾客

LEA DX,TIP\_5 ;显示提示：以游客方式登录

MOV AH,9

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

JMP STEP3

BRK8:

MOV BL,'q' ;比较q

INC DI ;字符串本体首地址

CMP BL,[DI]

JE EXIT

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

JMP STEP2 ;核验用户名正确性

BRK7:

LEA DX,TIP\_2 ;显示提示：输入密码

MOV AH,9

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

LEA DX,IN\_PWD ;输入密码

MOV AH,10

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

JMP BRK9 ;核验密码正确性

STEP2:

MOV CL,11 ;正确用户名串长

MOV CH,[IN\_NAME+1] ;输入用户名串长

CMP CH,CL

JNE BAR2

LPA\_1:

LEA DI,IN\_NAME

LEA DI,[DI+2]

LEA DX,BNAME

MOV BX,[DX] ;//寄存器间接寻址用16为寄存器时只能用BX,BP,SI,DI之一，此处改为SI

CMP BX,[DI]

JNE BAR2

INC DX

INC DI

DEC CL

JNE LPA\_1

JMP BRK7

BAR2:

LEA DX,TIP\_7 ;显示提示:用户名错误

MOV AH,9

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

JMP STEP1

BRK9:

MOV CL,6 ;正确密码串长

MOV CH,[IN\_PWD+1] ;输入密码串长

CMP CH,CL

JNE BAR3

LPA\_2:

LEA DI,IN\_PWD

LEA DI,[DI+2]

LEA DX,BPASS

MOV BX,[DX] ;//寄存器间接寻址用16为寄存器时只能用BX,BP,SI,DI之一，此处改为SI

CMP BX,[DI]

JNE BAR3

INC DX

INC DI

DEC CL

JNE LPA\_2

JMP BRK10

BAR3:

LEA DX,TIP\_8 ;显示提示:密码错误

MOV AH,9

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

JMP STEP1

BRK10:

MOV AUTH,1 ;认证方式:店主

LEA DX,TIP\_6 ;显示提示:以店主身份登录

MOV AH,9

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

STEP3:

LEA DX,TIP\_3 ;显示提示:输入待查商品名称

MOV AH,9

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

LEA DX,IN\_GOOD ;输入待查商品名称

MOV AH,10

INT 21H

MOV BL,0 ;单独输入回车符时真实串长为0

LEA DI,IN\_GOOD

INC DI ;找到真实串长

CMP BL,[DI]

JE STEP1

MOV CL,2 ;外循环计数器

LEA BP,GA1

LPB:

LEA BX,IN\_GOOD

INC BX

MOV DX,[BX]

MOV DH,0

LPBA:

LEA BX,IN\_GOOD

INC BX

MOV AX,[BX]

MOV AH,0

SUB AX,DX

MOV SI,AX

MOV BX,DS:[BP+SI]

MOV BH,0

MOV DI,BX

LEA BX,IN\_GOOD

ADD BX,2

MOV AX,SI

XLAT

CMP DI, AX

JNE BRK1 ;当前字符不同时直接退出循环

DEC DX

JNE LPBA ;SI非0时继续下一重循环

BRK1:

CMP DX,0 ;SI不为0说明对输入字符串的遍历未到头，即字符串不同，应继续比较下一字符串；为0则跳出外循环

JNE BRK13

INC SI

MOV BX,DS:[BP+SI]

CMP BX,0

JE BRK2

BRK13:

ADD BP,20 ;BP移向商店一中下一货物

DEC CL

JNE LPB ;商品未查找完

BRK2:

CMP CL,0 ;CL不为0说明已找到商品

JNE BRK3

LEA DX,TIP\_4 ;显示提示:重新输入待查商品名称

MOV AH,9

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

JMP STEP3

BRK3:

CMP AUTH,1

JNE BRK6

MOV BX,DS:[BP+10] ;进货价

;BP目前在目标商品名称字符串头，取数据时应加上相应偏移地址

IMUL BX,DS:[BP+14] ;进货数量

MOV AX,DS:[BP+12] ;销售价

IMUL AX,DS:[BP+16] ;已售数量

SUB AX,BX

MOV CX,100

IMUL CX

IDIV BX

ADD BP,18

MOV DS:[BP],AX ;商店一中的利润按字形式保存

LEA DI,PR1

MOV [DI],BP ;保存利润一的地址

;查找商品在商店二中的位置

MOV CL,2 ;外循环计数器

LEA BP,GB1

LPC:

LEA BX,IN\_GOOD

INC BX

MOV DX,[BX]

MOV DH,0

LPCA:

LEA BX,IN\_GOOD

INC BX

MOV AX,[BX]

MOV AH,0

SUB AX,DX

MOV SI,AX

MOV BX,DS:[BP+SI]

MOV BH,0

MOV DI,BX

LEA BX,IN\_GOOD

ADD BX,2

MOV AX,SI

XLAT

CMP DI, AX

JNE BRK4 ;当前字符不同时直接退出循环

DEC DX

JNE LPCA ;CH非0时继续下一重循环

BRK4:

CMP DX,0 ;CH不为0说明对字符串的遍历未到头，即字符串不同，应继续比较下一字符串；为0则跳出外循环

JNE BRK14

INC SI

MOV BX,DS:[BP+SI]

CMP BX,0

JE BRK5

BRK14:

ADD BP,20 ;DX移向商店一中下一货物

DEC CL

JNE LPC ;商品未查找完

BRK5:

MOV BX,DS:[BP+10] ;进货价

;DX目前在目标商品名称字符串尾，取数据时应加上相应偏移地址

IMUL BX,DS:[BP+14] ;进货数量

MOV AX,DS:[BP+12] ;销售价

IMUL AX,DS:[BP+16] ;已售数量

SUB AX,BX

MOV CX,100

IMUL CX

IDIV BX

MOV DS:[BP+18],AX ;商店二中的利润按字形式保存

MOV DI,PR1 ;从PR1里取利润一的地址

MOV BX,[DI]

ADD AX,BX

CMP AX,0

JL BRK11

MOV DX,0 ;AX为正

JMP BRK12

BRK11:

MOV DX,0FFFFH ;AX为负

BRK12:

MOV CX,2

IDIV CX ;求平均利润

MOV BX,AX

JMP STEP4

BRK6:

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

LEA BP,IN\_GOOD

INC BP

MOV BX,DS:[BP] ;IN\_GOOD有效串长

MOV BH,0

LPD:

LEA BP,IN\_GOOD

INC BP ;IN\_GOOD有效串长

MOV AX,DS:[BP]

MOV AH,0

SUB AX,BX

MOV SI,AX

INC BP

MOV DL,DS:[BP+SI]

MOV AH,2

INT 21H

DEC BX

JNE LPD ;SI非0时继续下一重循环

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

JMP STEP1

STEP4: ;EBX已被占用，存储总利润率

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

CMP BX,90

JGE SW1

CMP BX,50

JGE SW2

CMP BX,20

JGE SW3

CMP BX,0

JGE SW4

JMP SW5

SW1:MOV DL,'A'

JMP BAR4

SW2:MOV DL,'B'

JMP BAR4

SW3:MOV DL,'C'

JMP BAR4

SW4:MOV DL,'D'

JMP BAR4

SW5:MOV DL,'F'

BAR4:

MOV AH,2

INT 21H

MOV DL,0AH ;换行

MOV AH,2

INT 21H

MOV DL,0DH ;回车

MOV AH,2

INT 21H

JMP STEP1

EXIT:

MOV AH,4CH ;exit

INT 21H

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

CODE ENDS

END START

### 实验步骤

1. 准备上机实验环境。

2. 使用notepad++录入源程序，存盘文件名为task5.ASM。

3. 使用MASM 6.0汇编源文件。即MASM task5；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。

4. 使用连接程序LINK.EXE将汇编生成的task5.OBJ文件连接成执行文件。即LINK task5；

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CUBE.EXE文件。

5. 用TD单步调试该程序，观察执行现象。针对逻辑错误进行修改，直至对于各种输入可得出正确的执行结果。

### 实验记录与分析

1. 实验环境条件：Intel® Core™ i5-3230M CPU 2.60GHz，2.86G内存；WINDOWS 7下DOSBox0.74； notepad++ 7.55;MASM.EXE 6.0;LINK.EXE 5.2;TD.EXE 5.0。

2. 汇编源程序时，汇编程序报了3个错误，如图3.2.2所示：

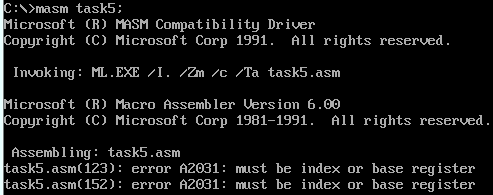


图3.5.2编译器报错

按照提示，对对应行仔细检查，发现了错误原因。见源程序中“//”后的说明。

3. 连接过程没有发生异常。

4. 用TD单步调试task5。在调试功能一时，我一开始认为直接输入回车时字符串首字节存放回车的ASCII码，同时实际串长为1.然而发现直接输入回车后实际串长仍为0，如图3.2.3所示。因此我将判断第一个字符是否为回车改为判断实际串长是否为0，从而保证输入回车时给出相应的结果，即以游客身份登陆。但此处将首字符与回车符比较也可得出同样的正确结果。

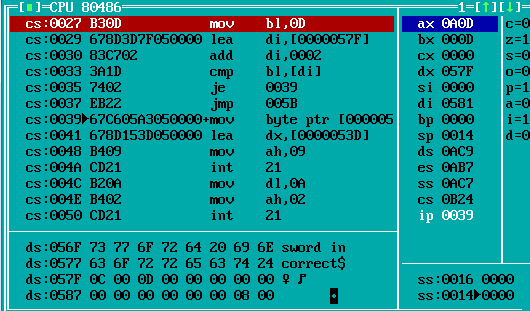


图3.5.3用户名串中直接输入回车符内存情况

5.试运行程序。发现在直接输入空格（以游客身份登录）后出现了如图3.5.4的格式错误，即“输入用户名”字符串被吞掉，“输入商品名”串直接接在了“以游客方式登录”之后。检查代码后，发现在各输出信息之后我并没有严格换行回车，从而导致了上述错误。因此我重点检查了一遍输出字符串相关调用，并同意在后面加上了回车换行。今天上课时，我又看到了老师提供的将回车换行组成一字符串一同输出的技巧，在有大量字符串输出的程序中可在一定程度上减少代码量，计划在今后的实验中应用。

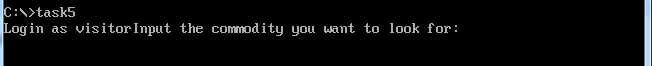


图3.5.4用户名串中直接输入回车符内存情况

6.单步调试功能三时，按图3.5.4方式输入信息。由于之前的部分已经单步调试保证正确性，因此在此处理应在商店一中找到相应商品并得出正确的利润。然而发现数据的明显错误，如图3.5.5所示。翻书后发现漏记了变址寻址中用BP作基址时默认段寄存器为SS的规定，即并未从DS中相应偏移地址处取数据，导致错误。在前面加入DS:标识后取数据恢复正常。

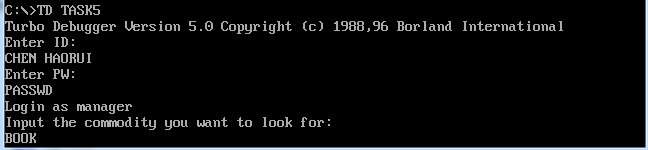


图3.5.5输入信息

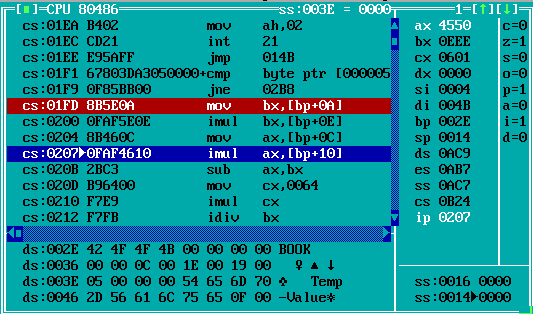


图3.5.6向BX及AX的数据转移明显出错

7.单步调试功能三，查找商品为PEN。在之前的代码均保证正确，对两利润取平均值时，除以2后程序跳转至图3.5.6所示位置。考虑到PEN的两利润之和为负数，再结合对负数做有符号字除法时要用到DX，猜想程序出错的原因是DX并不为0FFFFH。因此在做除法之前应根据总利润修改DX为0（总利润为正）或0FFFFH（总利润为负）。修改后程序运行恢复正常且能得出正确结果。

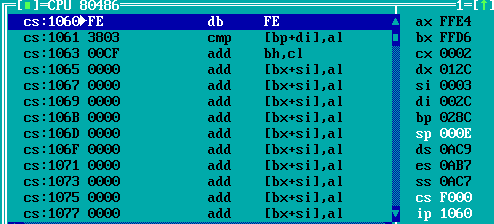


图3.5.7程序跳转至错误位置

8.单步调试功能四，目标商品为PEN时，正确的总利润应为负数，然而程序却在错误的位置跳转，导致输出结果为A。照书检查后发现JNB时无符号数跳转指令，然而此处比较的是有符号数，因此发生错误。将该部分所有JAE均改为JGE后输出结果恢复正确，如图3.5.9所示。

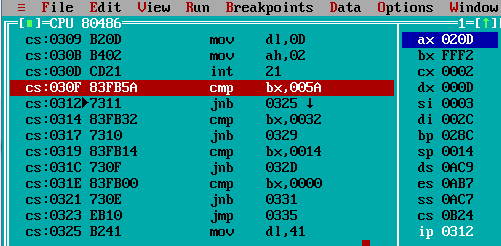


图3.5.8跳转时机错误

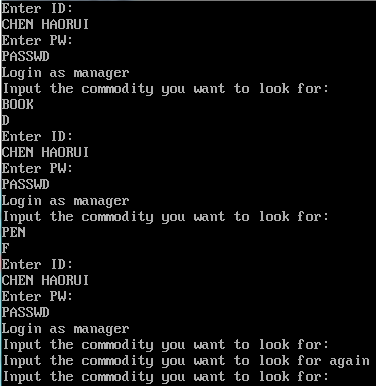


图3.5.9输出正确

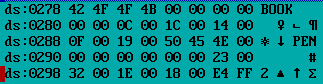


图3.5.10总利润分别保存在ds:028AH与ds:029EH开头的字中

# 总结与体会

通过任务1的实验，我基本了解了TD的单步调试与界面各部分的内容，知道了TD中可以直接在代码段输入汇编语句，ds段字节内容的查看方法与输入即时数可用二，十，十六进制等形式，同时也对各标志位的变化条件更加熟悉。经验是对于直接在TD中写命令的情况，可在代码段后面加上退出程序语句，以在单步调试完成时方便地退出TD并返回DOSbox主界面。

通过任务2的实验，我熟悉了汇编源程序从编译，连接并单步调试的整个过程以及运行汇编源程序的命令的简单方式。对于输入输出命令的使用也更为熟悉。由于在进行汇编时会根据行数显示编译错误信息，因此记事本并不算方便的代码编辑界面，可选择notepad++等有行数参考的工具编辑代码以在汇编报错时更为方便地定位错误。

任务3中，我理解了数据在数据段中的存放形式仅与其被定义的类型有关，不随段空间位长的改变而改变。通过观察汇编代码与机器指令的一一对应，我了解了其存储方式。同时我也领悟了IP对指令的指示对于程序运行中保持逻辑正确的重要性。

任务4时我可以说真的很多次碰上了“听完课以为会了，动起手来还是不会”和“错也不知道怎么错的，对也不知道怎么对的”一样的情况。完成此任务使我对四种寻址方式，相应的限制条件（尤其是16位下）以及一些基本语法变的更为熟悉。通过调试并观察各寄存器的值，我对一条语句中对某个寄存器到底在取地址还是取值，何时该取地址，何时该取值有了更深入的辨识能力。

任务5正如老师所说是个难关。这是我尝试编写的第一个汇编语言程序。在编程时需要考虑的因素有很多，第一点就是程序段间的跳转。简单的遍历还好说，循环结构中依靠判断条件的退出以及二重循环的跳转位置着实费了我一番心力。此外对于寄存器的使用也遇到了困难。在商店二中用二重循环查找目标商品时，除SP与IP外我已使用了已学的全部16位寄存器，但此时仍需一寄存器保存利润一在存储器中的位置，因此我开始第一次考虑寄存器向存储器传递数据的问题。但尽管本次依靠临时在寄存器中保存数据解决了问题，但频繁地做数据交互在增加代码量的同时也会增加程序循行的时间，这其中的平衡还需要在今后慢慢寻找。

源程序定义的段的次序，与调到内存中段的次序是有对应关系的。但数据段开始没有被赋值到实际的数据段首址中，需要执行了我们的相关赋值语句后才变得实际有效。另外，各种寻址方式在内存中确实不如源程序直观了，但可以看到变量的偏移值都被很好的计算出来了，我们可以通过源程序中变量与内存中对应位置的信息获得该变量的偏移地址，方便我们在数据显示区输入该地址，查看对应的变量内容。

思考题：

任务1

1.在选中的代码行直接输入新指令，F7单步执行。

2.在ip中输入第一条语句在CS中的偏移地址，回车。

3.数字后加H，表明是16进制数

4.如图4.1，AH为D8，AL仍为0.



图4.1 AL与AH的值

6.在寄存器窗口查看各寄存器的值；右键选择Registers 32-bit切换YES/NO；直接选中目标寄存器，输入新值。

8.计算结果最高位为1时，SF为1；加法运算结果为负/减法运算结果与减数相同时OF为1；加法从最高位进位/减法从最高位借位时CF为1

任务2&3

1.TD顶部菜单File中Open选项打开文件；最大的窗口即代码区

2.设置断点：选中语句，F2或顶部菜单Run中Toggle；单步运行至断点：F7或F8；直接运行至断点：顶部菜单Run中Run To Cursor

3.选中语句，F4

7.右键选择goto，输入ss:0

8.汇编：masm （程序名）；连接：link（程序名）；

9.碰见没见过的报错时百度/问老师/问同学，见得多了以后形成经验

11.数据以16进制形式，以字节为单位在DS中保存，先定义的放在低位，后定义的放在高位

12.表示为机器码，内容为指令与寄存器中保存的数据的组合

任务5

一、

2.持续输出之后的字符，直至碰到‘$’，如图4.2所示。

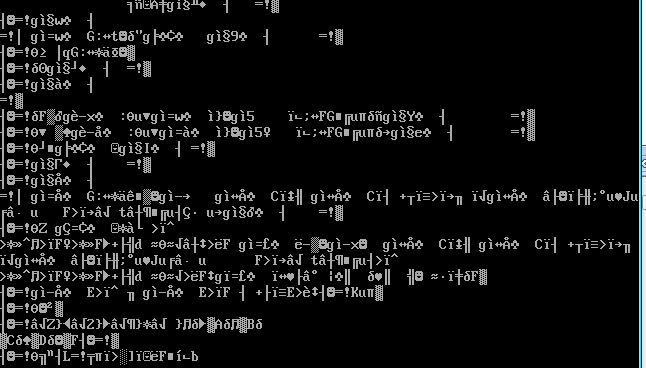


图4.2输出乱码

3.上图为task5中“Enter ID”串基址未赋给DX的情况，图4.4表明为开始调试时DX的初值为0。即不会从预想的位置开始输出字符串，而是从DX在9号调用前的位置开始输出字符串，直至遇到‘$’，如图4.3所示。

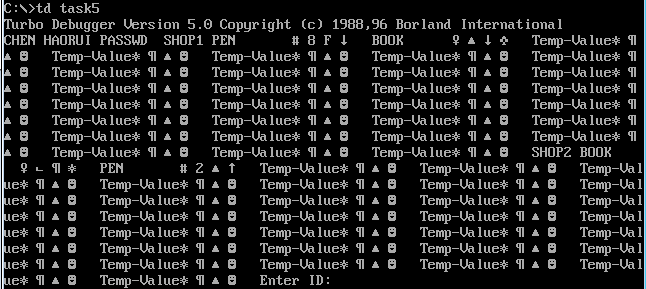


图4.3 输出乱码

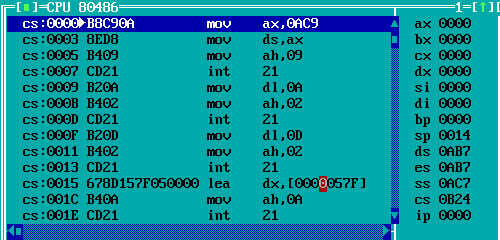


图4.4 DX的值

4.停止输入字符，也停止进行下一步，直至字符串长度小于给定长度的情况下再按回车。

5.先比较给定姓名串与输入串的长度，若不同则直接报错返回。

8.用于判定的变量在循环时不向目标值收敛则出现死循环。

# 参考文献

[1]许向阳，《80X86汇编语言程序设计上机指南》