

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实验**

**专业班级：计算机科学与技术1706班**

**学 号： U201713321**

**姓 名： 袁杰威**

**指导教师： 金良海**

**实验时段： 2020年3月17日~3月25日**

**实验地点： 南一楼804室**

**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：

报告日期：2020.5.10

实验报告成绩评定：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 |
| 实验完成质量（70%），报告撰写质量（30%），每次满分20分。 |  |  |  |  |  |
| 合计（100分） |  | | | | |

备注：实验完成质量从实验目的达成程度，设计方案、实验方法步骤、实验记录与结果分析论述清楚等方面评价；报告撰写质量从撰写规范、完整、通顺、详实等方面评价。

指导教师签字：

                    日期：

**目录**

[课程总体说明 - 2 -](#_Toc35182379)

[0.1 课程目标 - 2 -](#_Toc35182380)

[0.2 成绩构成 - 2 -](#_Toc35182381)

[0.3 实验任务的总体描述 - 2 -](#_Toc35182382)

[1 编程基础 1](#_Toc35182383)

[1.1 实验目的与要求 1](#_Toc35182384)

[1.2 实验内容 1](#_Toc35182385)

[1.3 任务1.1实验过程 1](#_Toc35182386)

[1.3.1 实验方法说明 1](#_Toc35182387)

[1.3.2 实验记录与分析 1](#_Toc35182388)

[1.4 任务1.2的实验过程 2](#_Toc35182389)

[1.4.1 设计思想及存储单元分配 2](#_Toc35182390)

[1.4.2 流程图 2](#_Toc35182391)

[1.4.3 源程序 3](#_Toc35182392)

[1.4.4 实验步骤 4](#_Toc35182393)

[1.4.5 实验记录与分析 4](#_Toc35182394)

[1.5 小结 5](#_Toc35182395)

[1.5.1 主要收获 6](#_Toc35182396)

[1.5.2 主要看法 6](#_Toc35182397)

[2 程序优化 7](#_Toc35182398)

[2.1 实验目的与要求 7](#_Toc35182399)

[2.2 实验内容 7](#_Toc35182400)

[2.3 任务2.1实验过程 7](#_Toc35182401)

[2.3.1 实验方法说明 7](#_Toc35182402)

[2.3.2 实验记录与分析 7](#_Toc35182403)

[2.4 任务2.2实验过程 7](#_Toc35182404)

[2.4.1 实验方法说明 7](#_Toc35182405)

[2.4.2 实验记录与分析 7](#_Toc35182406)

[2.5 小结 7](#_Toc35182407)

[2.5.1 主要收获 7](#_Toc35182408)

[2.5.2 主要看法 7](#_Toc35182409)

[3 模块化程序设计 7](#_Toc35182410)

[3.1 实验目的与要求 7](#_Toc35182411)

[3.2 实验内容 7](#_Toc35182412)

[。。。。。。。。 7](#_Toc35182413)

[4 中断与反跟踪 8](#_Toc35182414)

[4.1 实验目的与要求 8](#_Toc35182415)

[4.2 实验内容 8](#_Toc35182416)

[。。。。。。。。 8](#_Toc35182417)

[5 WIN32程序设计 8](#_Toc35182418)

[5.1 实验目的与要求 8](#_Toc35182419)

[5.2 实验内容 8](#_Toc35182420)

[。。。。。。。。 8](#_Toc35182421)

[参考文献 9](#_Toc35182422)

# 课程总体说明

## 0.1 课程目标

下表是本课程的目标及与支撑的毕业要求指标点之间的关系。请大家关注下表中最后一列“实验中的注意事项”的内容，以便更有针对性的满足课程目标的要求。

|  |  |  |
| --- | --- | --- |
| 课程目标 | 支撑的毕业要求指标点 | 实验中的注意事项 |
| 掌握汇编语言程序设计的全周期、全流程的基本方法与技术，通过程序调试、数据记录和分析，了解影响设计目标和技术方案的多种因素。 | 3.1掌握与计算机复杂工程问题有关的工程设计和软硬件产品开发全周期、全流程的基本设计/开发方法和技术，了解影响设计目标和技术方案的多种因素。 | 不能只写代码完成功能，还要有设计、调试、记录、分析等部分的内容。 |
| 掌握编写、调试汇编语言程序的基本方法与技术，能根据实验任务要求,设计出较充分利用了汇编语言优势的软件功能部件或软件系统。 | 3.2能为计算机复杂工程问题解决方案设计满足特定需求的软/硬件模块。 | 要思考与运用汇编语言的优势编写某些程序。 |
| 熟悉支持汇编语言开发、调试以及软件反汇编的主流工具的功能、特点与局限性及使用方法。 | 5.1了解计算机专业常用的现代仪器、信息技术工具、工程工具和模拟软件的使用原理和方法，并理解其局限性。 | 熟悉实验中使用的工具，把对工具的看法记录在案。 |

## 0.2 成绩构成

实验课程综合成绩由实验过程成绩和实验报告成绩二部分构成。**实验过程成绩**：30%。主要考察各实验完成过程中的情况，希望大家做到预习准备充分，操作认真熟练，在规定的时间内完成实验任务，结果正确，积极发现和提出问题，交流讨论时描述问题准确、清晰。实验报告成绩：70%。主要考核报告体现的实验完成质量(含问题的分析、设计思想与程序、针对问题的实验方法与步骤、实验记录、实验结果分析等方面)和报告格式规范等撰写质量方面的内容。

## 0.3 实验任务的总体描述

本课程安排了8次4学时的课内实验课时，将实现一个具有一定复杂程度的系统。对该系统的相关要求被划分成了**5个主题**：1）搭建原型系统；2）在原型系统基础上探索程序指令级别的优化；3）通过模块化调整与优化原型系统的程序结构；4）通过中断、内存数据和地址操纵、跟踪与反跟踪、加密等措施增强系统安全性；5）程序在不同平台上的移植。

针对这5个主题，对应地布置了5次实验。**实验1（编程基础）**安排8个课内学时熟悉汇编语言程序设计的基本方法、技术与工具，设计实现指定原型系统的主要功能。针对原型系统的搭建，实验报告中要有全周期、全流程的描述。**实验2（程序优化）**安排4个课内学时探索如何通过选择不同的指令及组合关系来优化程序的性能或代码长度。**实验3（模块化程序设计）**安排8个课内学时，利用子程序、模块化程序设计方法、与C语言混合编程等，调整与优化程序结构。**实验4（中断与反跟踪）**安排8个课内学时，通过利用中断机制、内存数据和地址操纵技术、跟踪与反跟踪技巧、加密等措施增强系统安全性。**实验5（WIN32程序设计）**安排4个课内学时，熟悉在不同操作系统平台上移植实现已有系统功能的基本方法。每次实验的侧重面有所不同，但都会涉及到课程目标的三个方面，因此，需要大家在实验过程中以及实验报告中有所注意和体现。

**本次课程所涉及的原型系统是一个网店商品信息管理系统。下面描述该系统的基本需求，后续每次实验都是以这个基本需求为背景而展开的。**

有一个老板在网上开了1个网店SHOP，网店里有n种商品销售。每种商品的信息包括：商品名称（最长名称9个字节，其后加一个数值0表示名称结束），折扣（字节类型，取值0~10；0表示免费赠送，10表示不打折，1~9为折扣率；实际销售价格=销售价\*折扣/10），进货价(字类型)，销售价（字类型），进货总数（字类型），已售数量（字类型），推荐度【=（进货价/实际销售价格+已售数量/（2\*进货数量））\*128，字类型】。老板管理网店信息时需要输入自己的名字（最长名字9个字节，其后加一个数值0表示结束）和密码（最长密码6个字节，其后加一个数值0表示结束），老板登录后可查看商品的全部信息；顾客（无需登录）可以查看网店中每个商品除了进货价以外的信息，可以对指定商品下单预定。

该系统被执行后，首先显示一个菜单界面，菜单界面信息包括：

当前用户名：（老板名称或顾客）

当前浏览商品名称：（没有时空缺）

请输入数字1…9选择功能：

1.登录/重新登录

2.查找指定商品并显示其信息

3.下订单

4.计算商品推荐度

5.排名

6.修改商品信息

7.迁移商店运行环境

8.显示当前代码段首址

9.退出

当用户输入某一个有效数字后，就进入到指定的功能中执行，执行完之后再回到该菜单界面。如果选择的是退出功能，则程序退出。该菜单中每项菜单的具体功能要求详见每次的实验任务描述。

# 编程基础

## 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1.1**：《80X86汇编语言程序设计》教材中 P31的 1.14题。要求：

(1) 直接在TD中输入指令，完成两个数的求和、求差的功能。求和/差后的结果放在(AH)中。

(2) 请事先指出执行指令后(AH)、标志位 SF、OF、CF、ZF的内容。

(3) 记录上机执行后的结果，与（2）中对应的内容比较。

**任务1.2：**《80X86汇编语言程序设计》教材中 P45的 2.3题。要求：

（1）分别记录执行到“MOV CX，10”和“INT 21H”之前的(BX),(BP), (SI), (DI)各是多少。

（2）记录程序执行到退出之前数据段开始40个字节的内容，指出程序运行结果是否与设想的一致。

**任务1.3：**《80X86汇编语言程序设计》教材中 P45的 2.4题的改写。要求：

(1) 实现的功能不变，但对数据段中变量访问时所用到的变址寄存器采用32位寄存器。

(2) 记录程序执行到退出之前数据段开始40个字节的内容，检查程序运行结果是否与设想的一致。

(3)在TD代码窗口中观察并记录机器指令代码在内存中的存放形式，并与TD中提供的反汇编语句及自己编写的源程序语句进行对照，也与任务1.2做对比。（相似语句记录一条即可，重点理解机器码与汇编语句的对应关系，尤其注意操作数寻址方式的编码形式，比如寄存器间接寻址、变址寻址、32位寄存器与16位寄存器编码的不同、段前缀在代码里是如何表示的等）。

（4）观察连续存放的二进制串在反汇编成汇编语言语句时，从不同字节位置开始反汇编，结果怎样？理解 IP/EIP指明指令起始位置的重要性。

## 任务1.1实验过程

### 实验方法说明

1. 准备上机实验环境，对实验用到的软件进行安装、运行，通过试用初步了解软件的基本功能、操作等。

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,01001101B；MOV AL,-01110010B；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。

1. 预计执行ADD之后：。（AH）=0DBH、SF=1、OF=0、CF=0、ZF=0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。

实验预测如下

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | AH | SF | OF | CF | ZF |
| 1 | 8DH | 1 | 1 | 0 | 0 |
| 2 | 7AH | 0 | 1 | 1 | 0 |
| 3 | 08H | 0 | 0 | 1 | 0 |

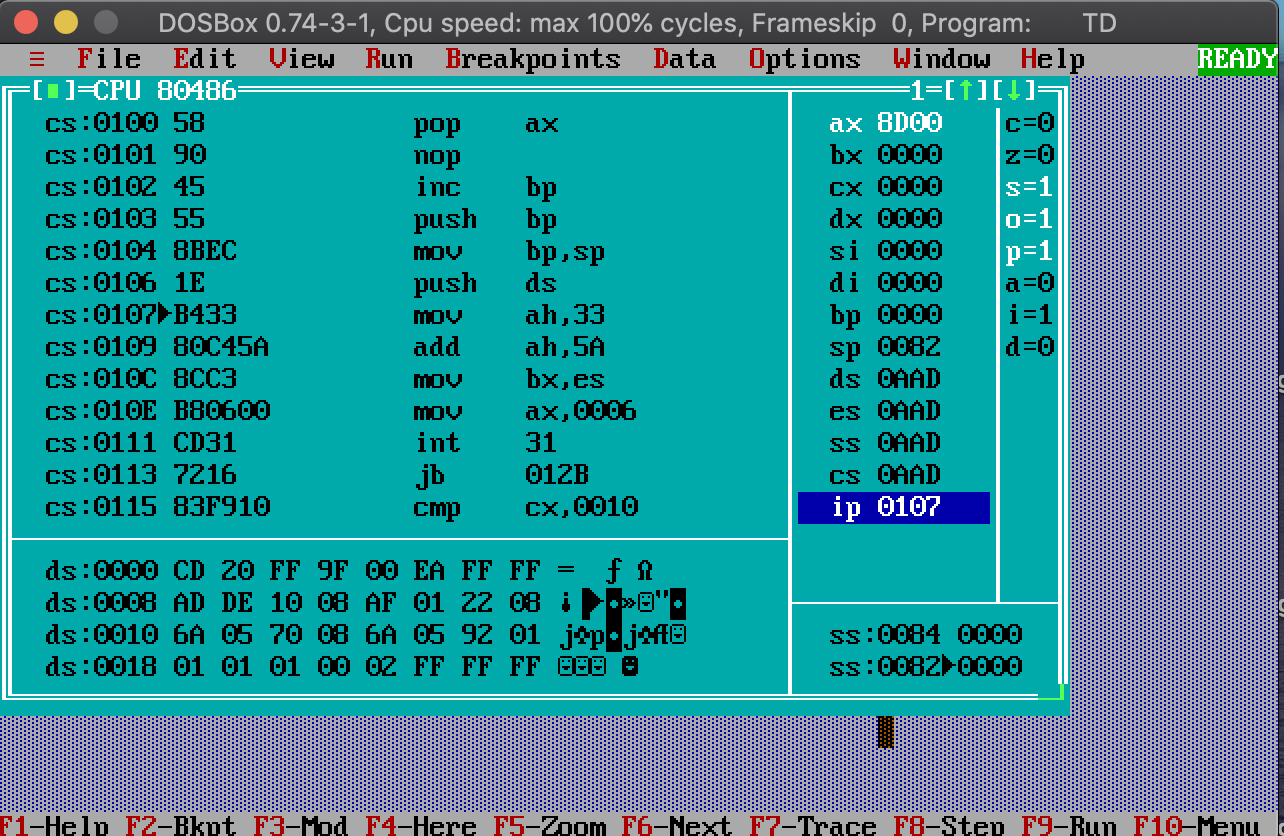
### 实验记录与分析

1. 实验环境条件：intel i5-7360U 2.3GHz，8G内存；MacOS下DOSBox0.74； TD.EXE 5.0。

2. 第（1）小题，x1=+0110011B,x2=+1011010B。

在TD之间输入指令 MOV AH,00110011B; ADD AH,01011010B,F7进行单步调试。

调试结果如图1.1所示。可得ax即AH=8DH,CF=0,ZF=0,SF=1,OF=1。与预测相同。

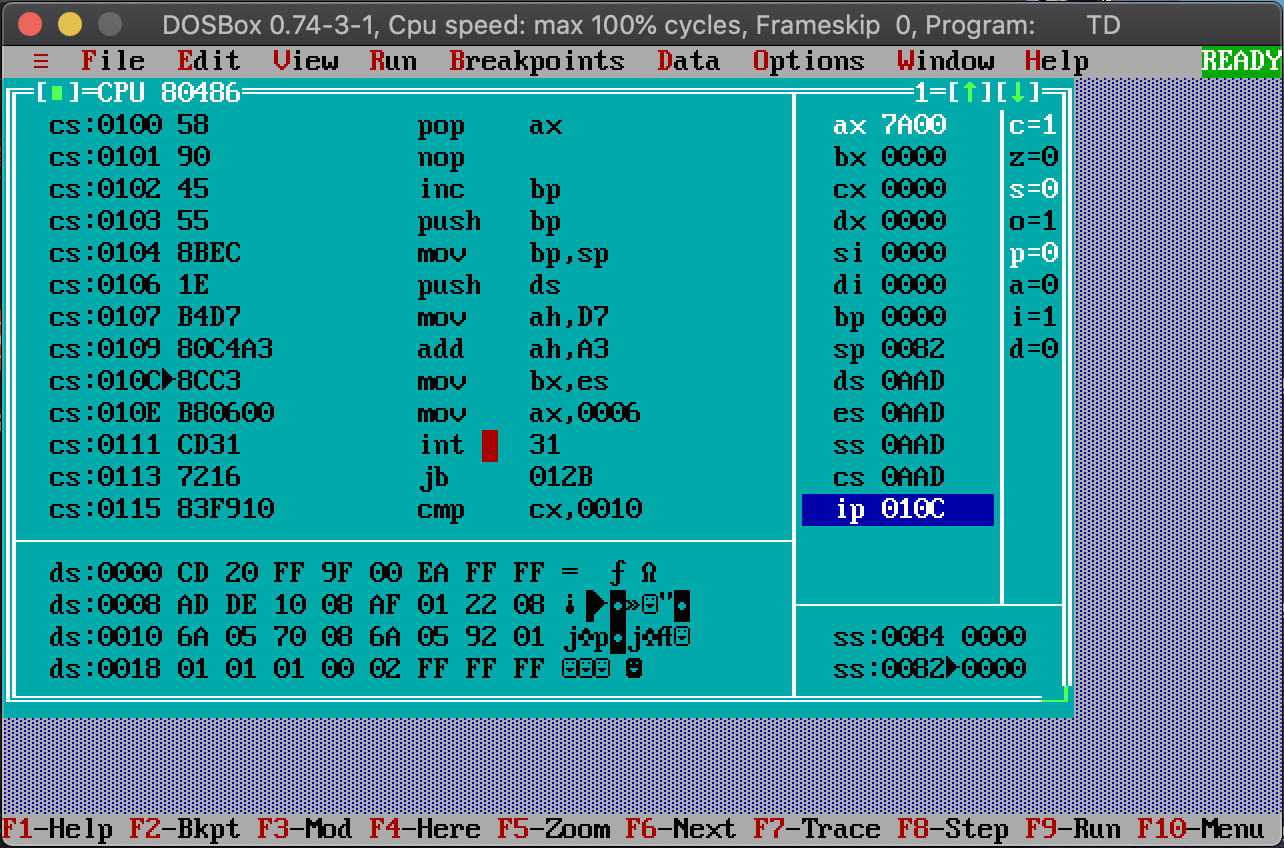


* 1. 任务1第一小题测试

1. 第（2）小题，x1=-0101001B,x2=-1011101B。

在TD之间输入指令 MOV AH,11010111 B; ADD AH,10100011B,F7进行单步调试。

调试结果如图1.1所示。可得ax即AH=7AH,CF=1,ZF=0,SF=0,OF=1。与预测相同。

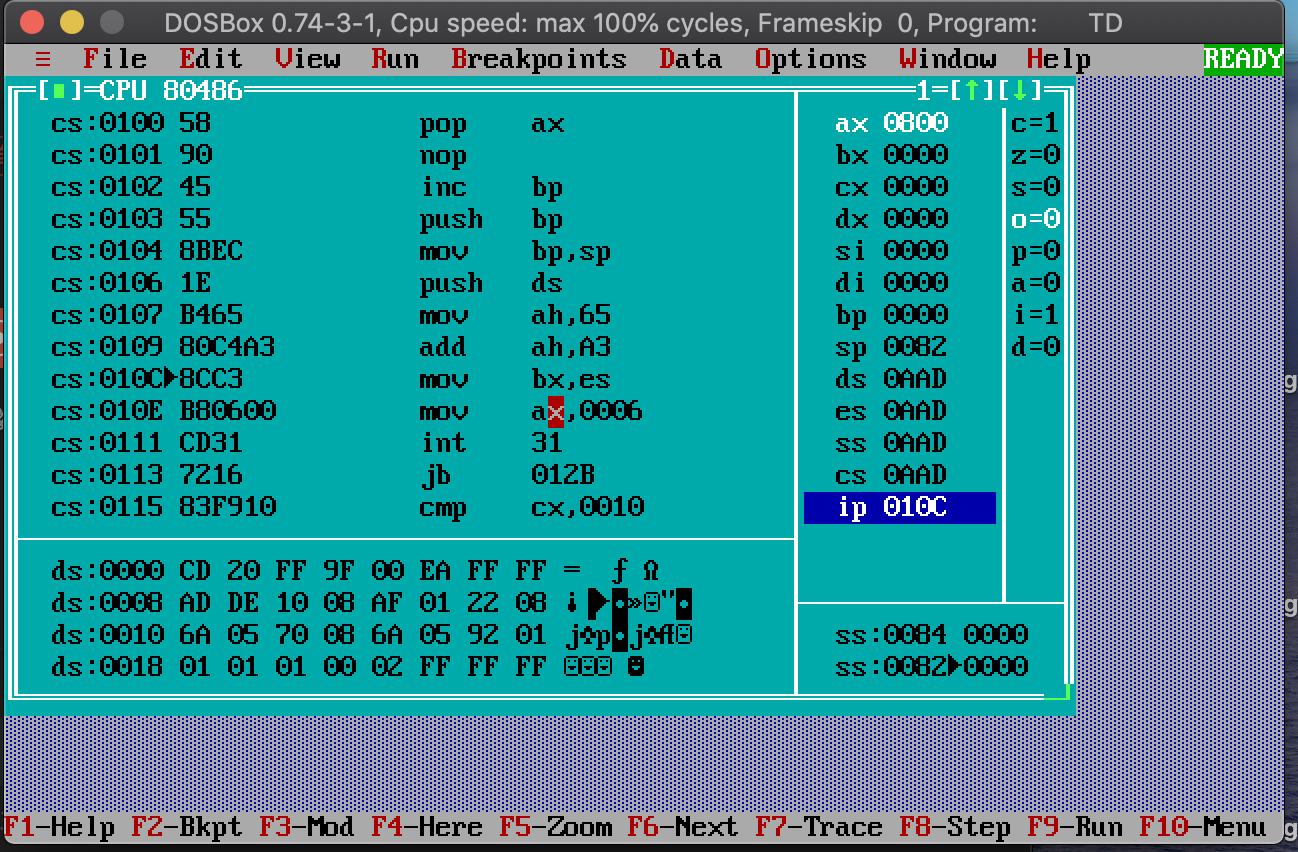


1.2任务1第二小题测试

1. 第（3）小题，x1=+1100101B,x2=-1011101B。

在TD之间输入指令 MOV AH,01100101 B; ADD AH,10100011B,F7进行单步调试。

调试结果如图1.1所示。可得ax即AH=08H,CF=1,ZF=0,SF=0,OF=0。与预测相同。



1.3 任务1第三小题测试

## 任务1.2的实验过程

## 1.4.1 实验方法说明

1. 用记事本将P45 2.3题的代码编写完成，并以.ASM格式保存，命名为45.ASM。
2. 在DOSBox中使用masm汇编源代码，生成45.OBJ文件。
3. 在DOSBox中使用LINK连接45.OBJ，生成可执行文件45.EXE。
4. 运行TD并打开45.EXE文件。

表1.2 任务2预计结果（1）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BX | BP | SI | DI |
| MOV CX 10 | 0014H | 001EH | 0000H | 000AH |
| INT 21H | 001EH | 0028H | 000AH | 0014H |

表1.3 任务2预计结果（2）

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | DS段前40字节内容 | | | | | | | |
| 程序结束前 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
| 08 | 09 | 00 | 01 | 02 | 03 | 04 | 05 |
| 06 | 07 | 08 | 09 | 01 | 02 | 03 | 04 |
| 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D |

### 1.4.2 实验记录与分析

1. 实验环境条件：intel i5-7360U 2.3GHz，8G内存；MacOS下DOSBox0.74； TD.EXE 5.0。
2. 汇编源程序

3. 连接过程没有发生异常。

4. 单步调试，在执行MOV CX 10之前，各寄存器值如1.5图所示，BX=0014H，BP=0014H，BP=001EH，SI=0000H，DI=000AH。符合预测值

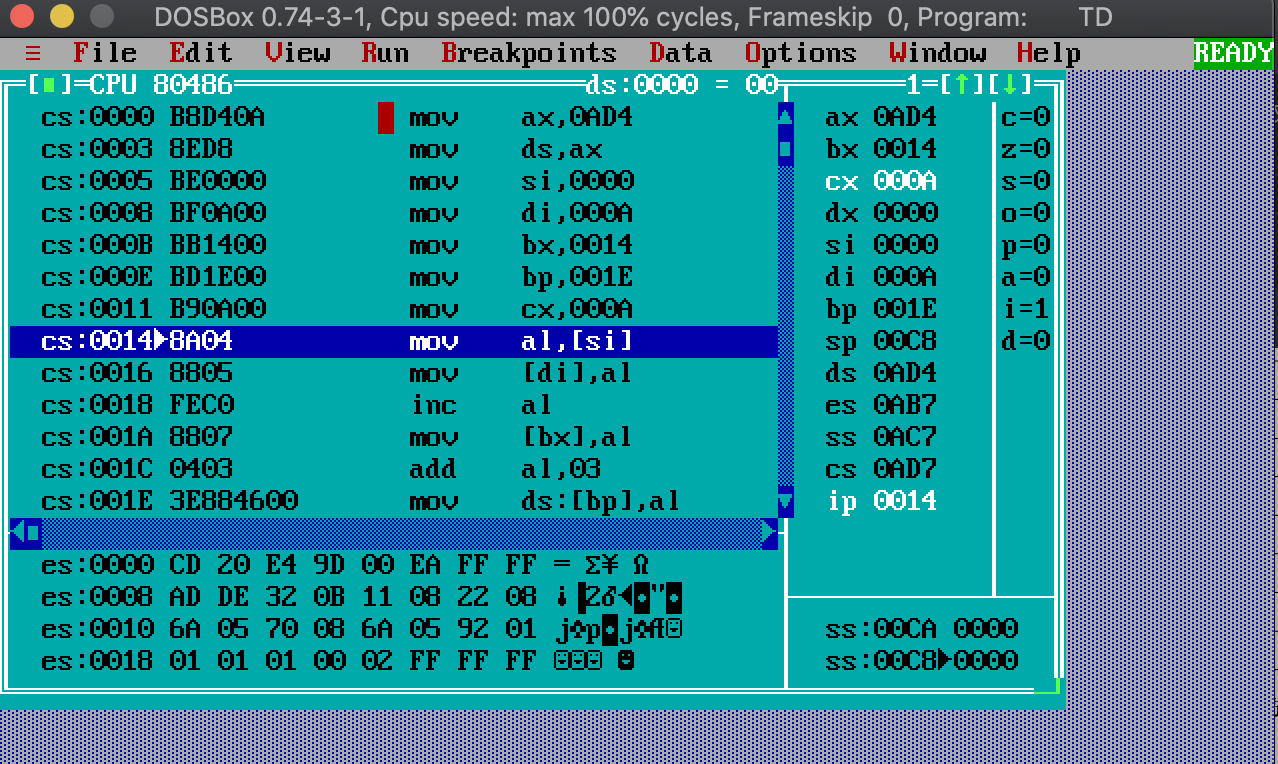


图1.4 执行MOV CX 10前情况

1. 单步调试，在执行INT 21H之前，各值如1.6图所示，BX=001EH,BP=0028H,SI=000AH,DI=0014H。符合预期值。

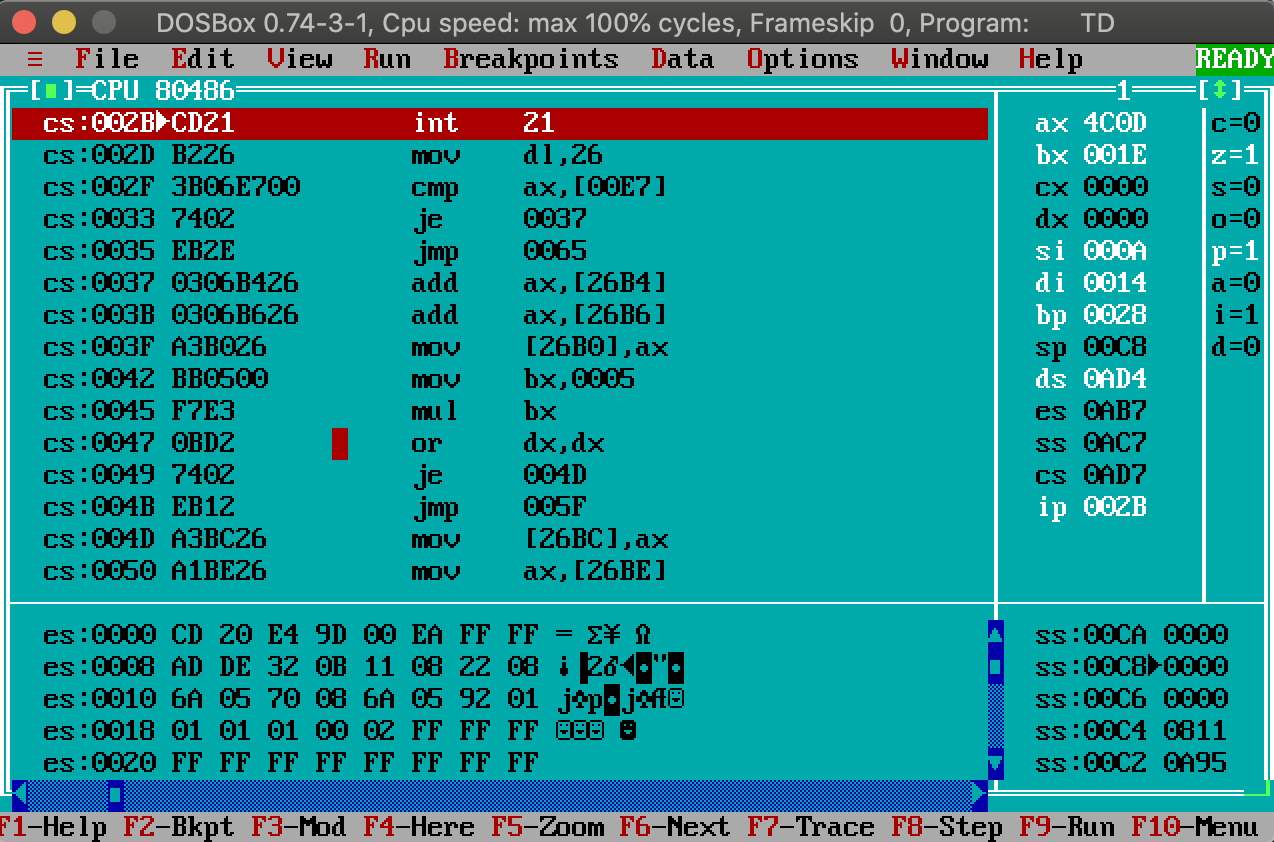


图1.5 执行INT 21H前情况

1. 跳转至DS区。转换至堆栈区后使用goto功能输入 “DS:0”，得到结果如图1.7所示。与预期结果相同。

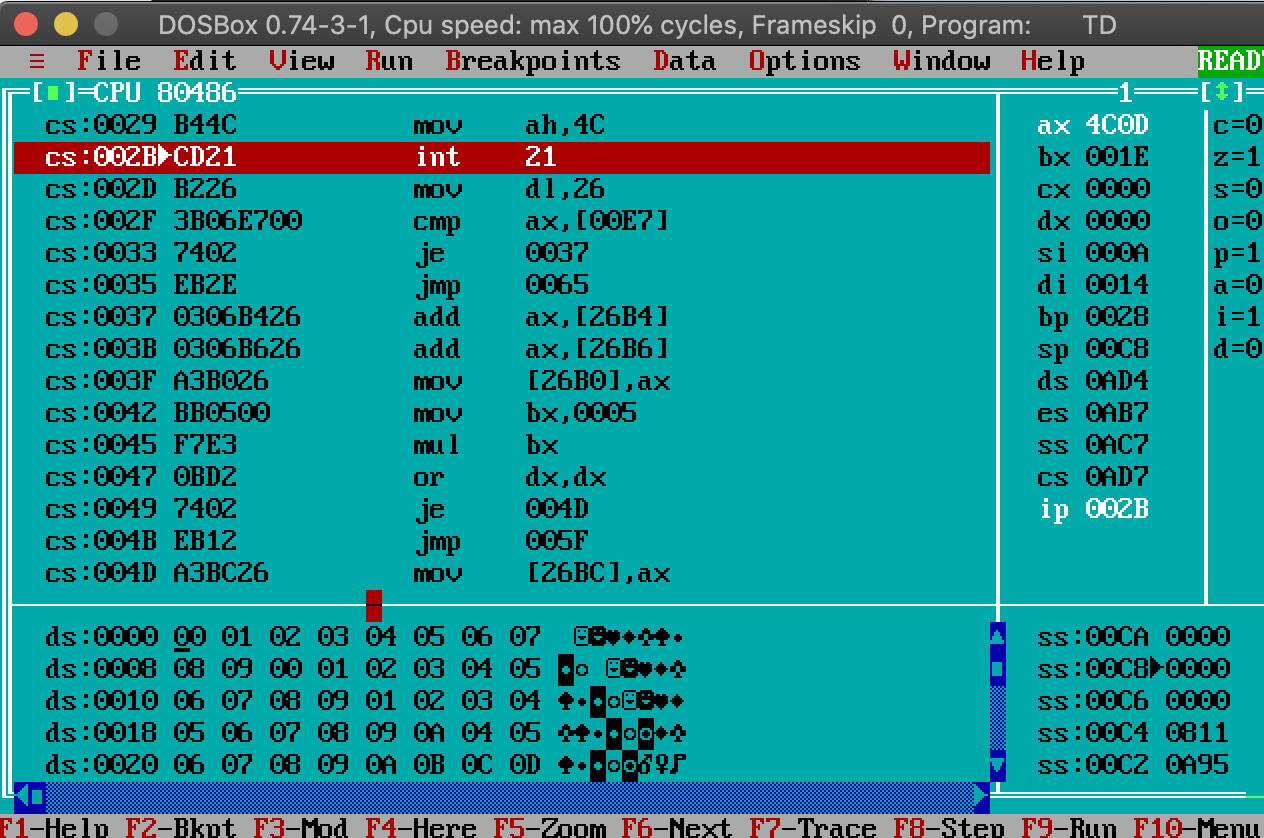


图1.6 DS段数据前40字节

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 SI, OFFSET BUF1

    MOV DI, OFFSET BUF2

    MOV BX, OFFSET BUF3

    MOV BP, OFFSET BUF4

    MOV CX, 10

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

## 任务1.3的实验过程

### 实验步骤

* 1. 修改任务2中的代码,使其满足任务3的需求,命名为expr\_1\_3.asm保存。
  2. 生成对应asm、obj、exe文件操作与使用TD编译操作同任务2。
  3. 实验预测

表1.4任务3预计结果

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | DS段前40字节内容 | | | | | | | |
| 程序结束前 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 |
| 08 | 09 | 00 | 01 | 02 | 03 | 04 | 05 |
| 06 | 07 | 08 | 09 | 01 | 02 | 03 | 04 |
| 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D |

### 实验记录与分析

1. 修改后代码

**.386**

**STACK SEGMENT USE16 STACK**

**DB 200 DUP(0)**

**STACK ENDS**

**DATA SEGMENT USE16**

**BUF1 DB 0,1,2,3,4,5,6,7,8,9**

**BUF2 DB 10 DUP(0)**

**BUF3 DB 10 DUP(0)**

**BUF4 DB 10 DUP(0)**

**DATA ENDS**

**CODE SEGMENT USE16**

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

**START:  MOV      AX,DATA**

**MOV      DS,AX**

**MOV SI, OFFSET BUF1**

**MOV DI, OFFSET BUF2**

**MOV BX, OFFSET BUF3**

**MOV BP, OFFSET BUF4**

**MOV CX, 10**

**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**

1. 实验环境条件：intel i5-7360U 2.3GHz，8G内存；MacOS下DOSBox0.74； TD.EXE 5.0
2. 汇编,连接过程没有发生异常。
3. 对文件进行单步调试，并使用goto跳转至DS段观察前40个字节的内容，最终成功得到结果如图1.8所示，结果与预测相同。

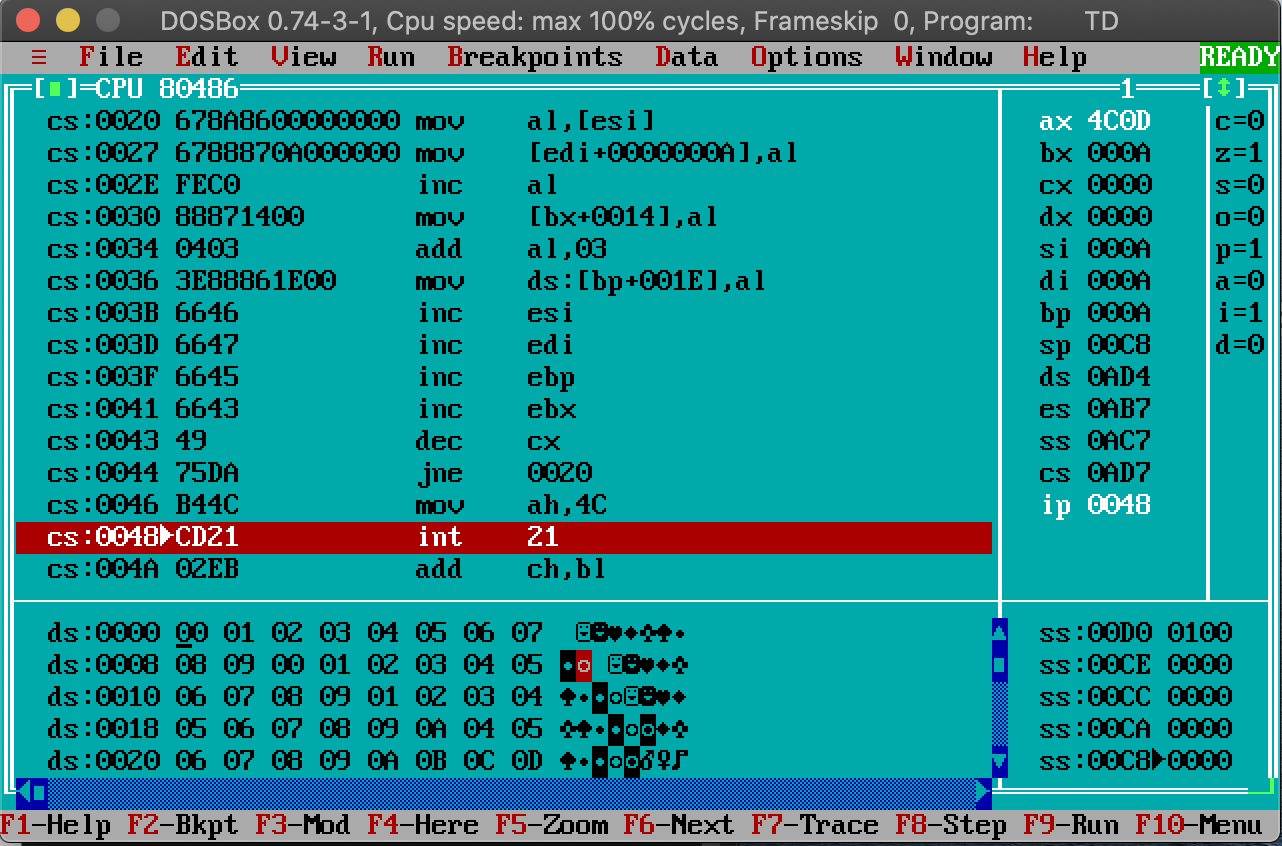


图1.7 任务3DS段前40字节

1. 任务3反汇编指令与原代码指令对比

代码段截图如图1.9所示

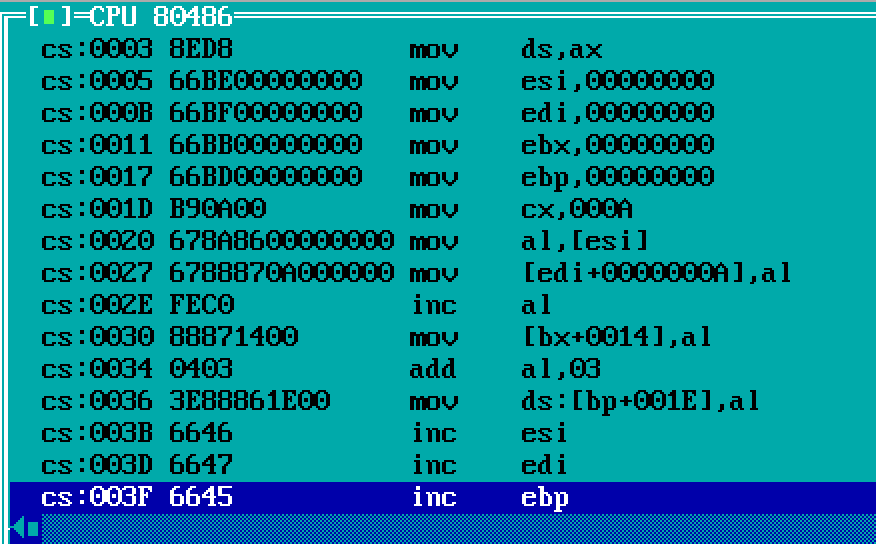
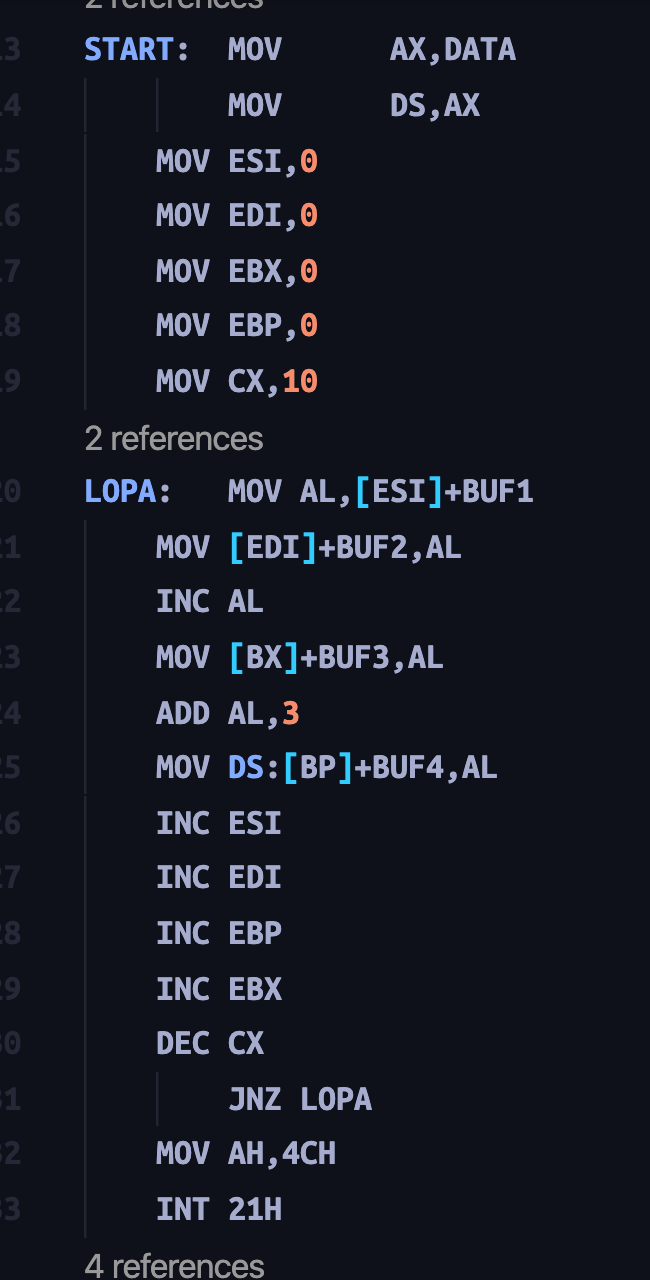
 

图1.8 任务3反汇编代码（左）与源代码（右）对比

（1）由对比可知在源代码中用十进制表示的常数在反汇编后全部转化为了十六进制。如源代码的MOV CX 10在反汇编中变成了MOV CX 000A。

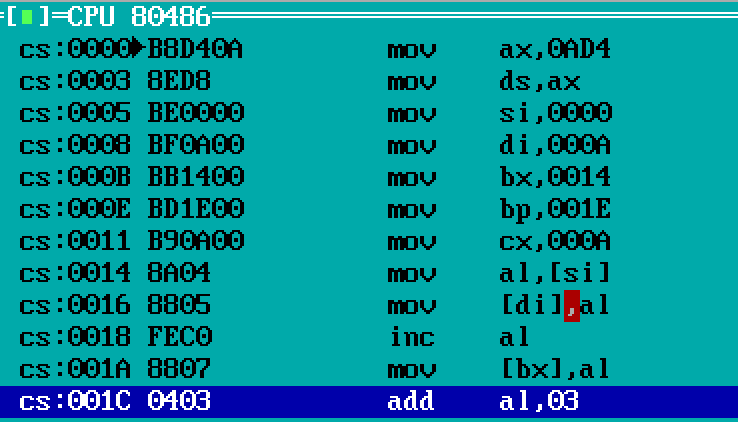
（2）在反汇编中所有的地址表示方法都变成了偏移地址。如MOV DS:[BP]+BUF4,AL在反汇编中变成了MOV DS:[bp+001E],AL

以及JNZ LOPA中LOPA标识也变成了JNE 0020。

（3）除了上述外，大部分表示反汇编任然与源程序表示方法相同。包括选择的寄存器名称，操作指令名称等，都保持不变。

1. 任务2与任务3反汇编代码对比

代码段截图如图1.9所示



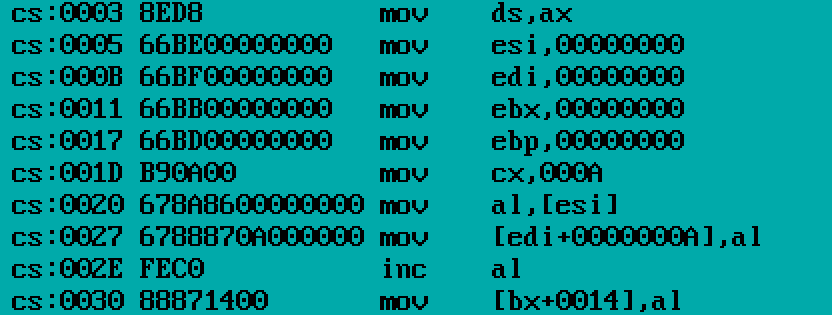


图1.9 任务2反汇编代码（上）任务3反汇编代码（下）

可以清晰对比32位寄存器与16位寄存器在偏移地址上的差别。

1. 尝试从不同位置开始反汇编

最开始仅仅使用了goto语句，并没有将该语句设置为操作句，结果发现寄存器IP并没有改变，且TRACE操作后仍然返回了GOTO之前的语句。

利用goto功能在代码段输入0014地址，并且将其设置为操作句后，观察到IP也随之更改为0014，如图1.10所示。EIP/IP是程序代码段执行的指向，计算机会从EIP/IP所指的位置开始执行代码。

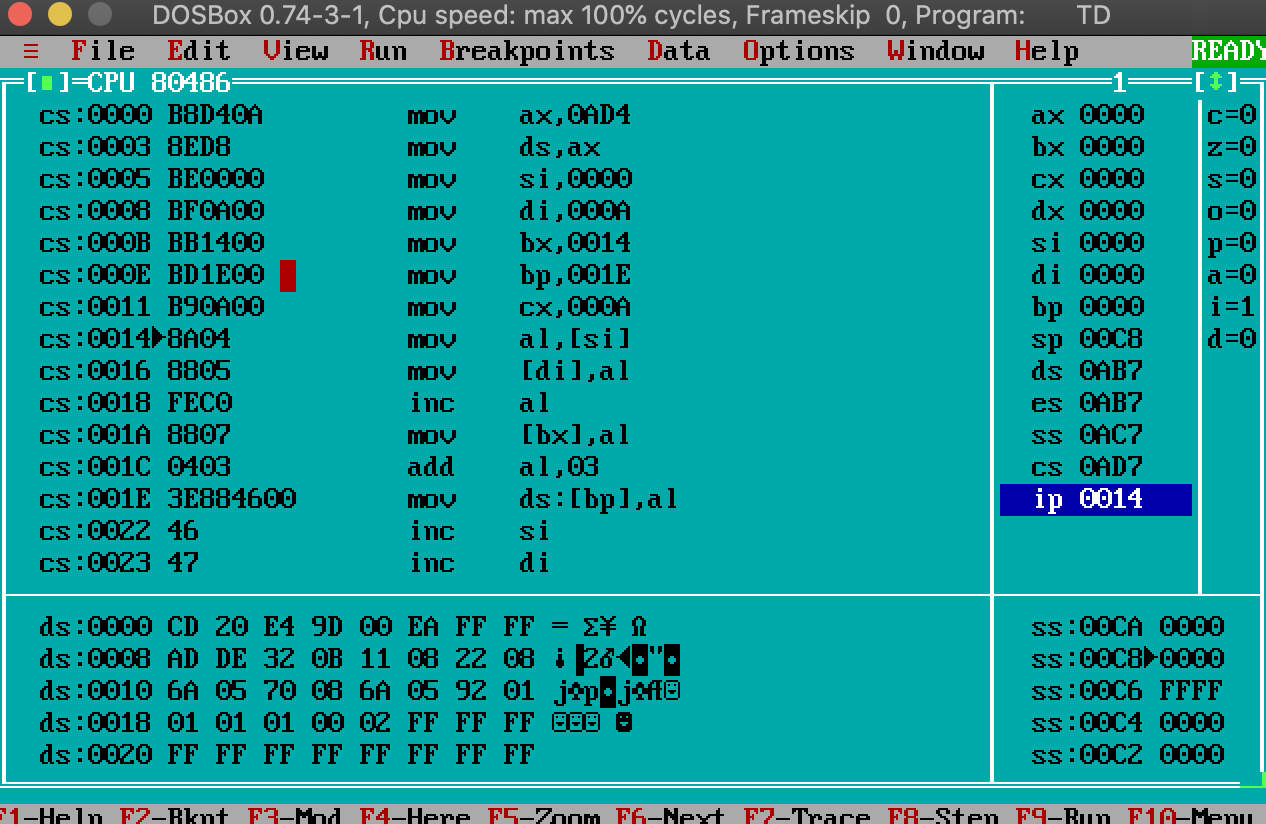


图1.10 goto语句操作

## 任务1.4的实验过程

### 实验步骤

1. 设计登录/重新登录

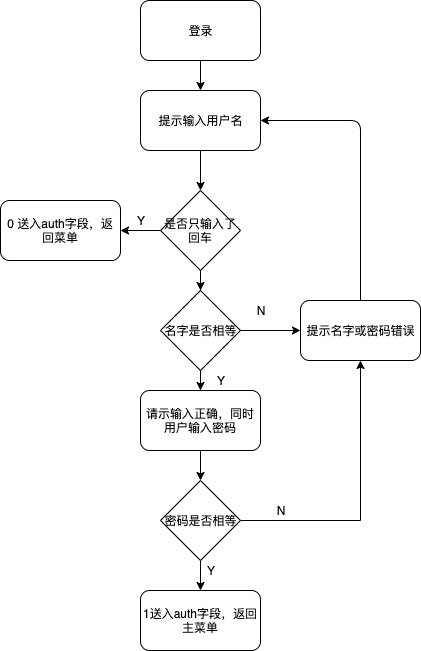


图1.11 登录操作流程图

1. 查找指定商品并显示其信息

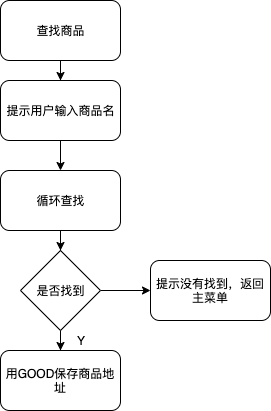


图1.12 登录操作流程图

1. 下订单

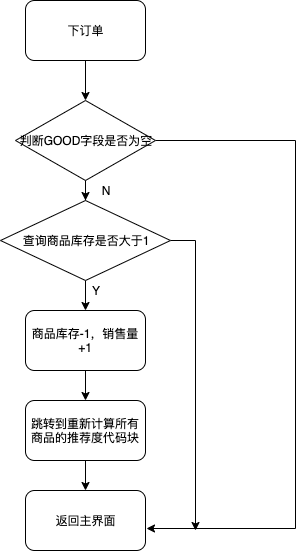


图1.13 登录操作流程图

1. 计算商品推荐度，重点在于找对偏移量

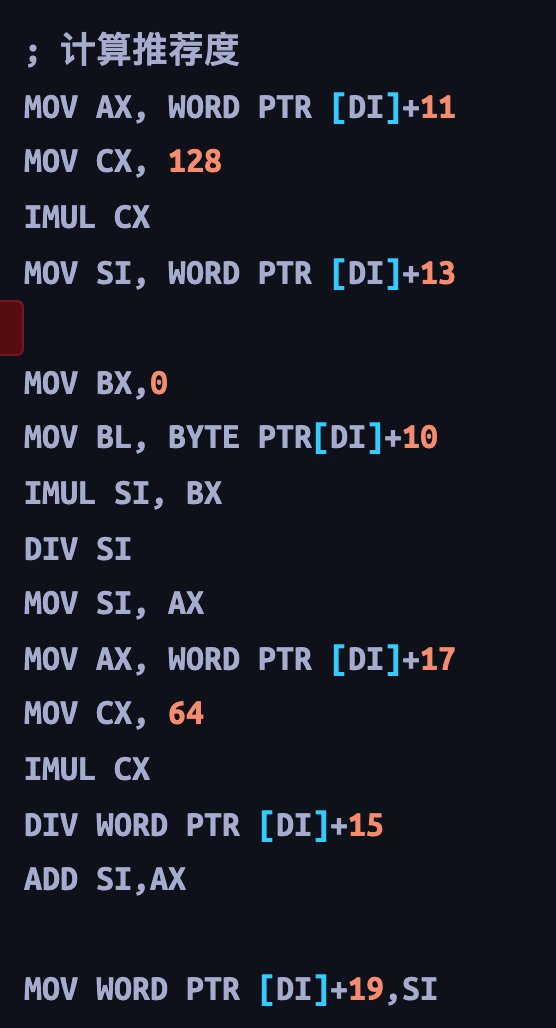


图1.14 计算推荐度代码片段

1. 显示CS的16进制，思路：定义一个buf10:‘0123456789abcdef’,对CS的值取关于16的余数，同时放入栈中，然后一次pop，通过值可以得到buf10中对应的ascii码

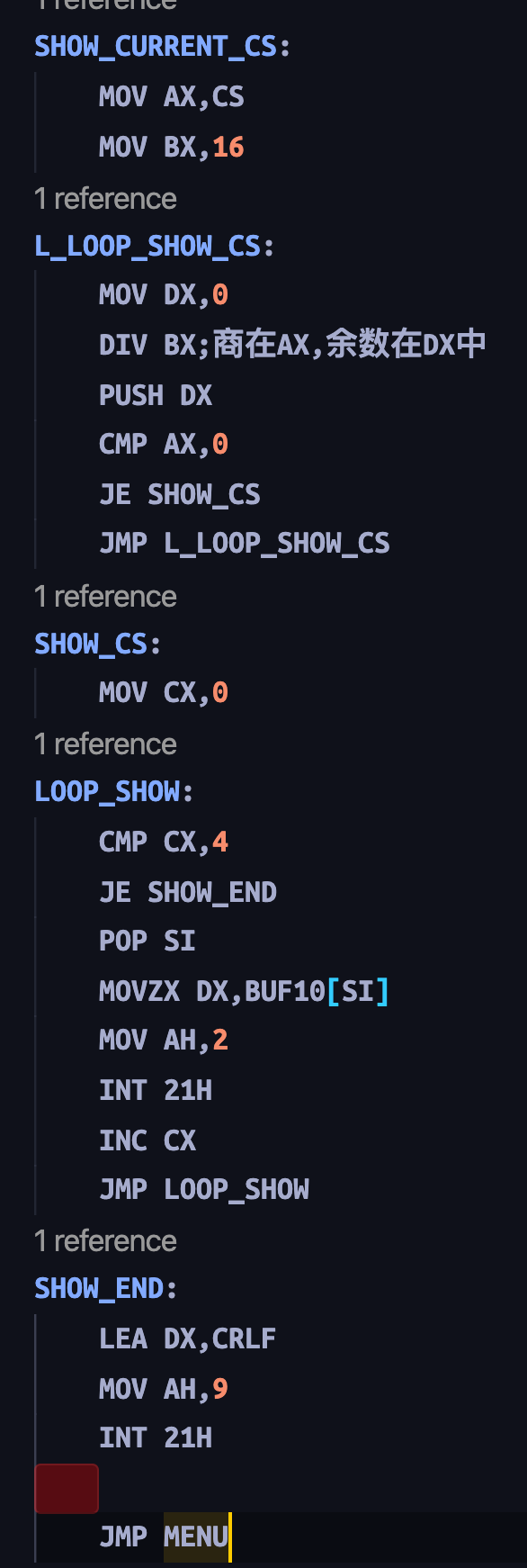


图1.15 输出CS的16进制

1. 实验代码：

**.386**

**STACK SEGMENT USE16 STACK**

**DB 200 DUP(0)**

**STACK ENDS**

**DATA SEGMENT USE16**

**BUF  DB '\*\*\*\*\*THE SHOP IS SHOP\_ONE\*\*\*\*\*$'**

**BUF1 DB 'PLEASE INPUT YOUR NAME $'**

**BUF2 DB 'PLEASE INPUT YOUR PASSWORD $'**

**BUF3 DB 'The name is wrong $'**

**BUF4 DB 'Please input the good you want $'**

**BUF5 DB 'The NAME is right $'**

**BUF6 DB 'The PWD is wrong$'**

**BUF7 DB 'Landed successfully$'**

**BUF8 DB 'Good you want not exist$'**

**BUF9 DB 'Good is not remain$'**

**BUF10 DB '0123456789abcdef'**

**BUF11 DB 'The pwd is right $'**

**BUF12 DB 'Good found!!!$'**

**BUF13 DB 'count !!!$'**

**BUF14 DB 'update success !!!$'**

**AUTH DB ?**

**CRLF DB 0DH,0AH,'$'**

**IN\_NAME DB 11**

**DB ?**

**DB 11 DUP(0)**

**IN\_PWD   DB 7**

**DB ?**

**DB 7 DUP(0)**

**IN\_GOOD DB 11**

**DB ?**

**DB 11 DUP(0)**

**BNAME DB 'yjw',3 DUP(0)**

**COUNT1 = $-BNAME**

**BPASS DB 'test',0,0**

**COUNT2 = $-BPASS**

**N EQU 30**

**S1  DB 'SHOP\_ONE',0**

**GA1 DB 'PEN',7 DUP(0),10**

**DW 35,56,70,25,?**

**GA2 DB 'BOOK',6 DUP(0),9**

**DW 12,30,25,5,?**

**;实际销售价格=销售价\*折扣/10**

**; 进货价(字类型)，**

**; 销售价（字类型），**

**; 进货总数（字类型），**

**;已售数量（字类型）**

**; 推荐度**

**GAN DB N-2 DUP('TEMP-VALUE',8,15,0,20,0,30,0,2,0,?,?)**

**GOOD DW 0**

**DATA ENDS**

**CODE SEGMENT USE16**

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

**START:**

**MOV AX,DATA**

**MOV DS,AX**

**LEA DI,GA1**

**MOV CX,10**

**MENU:**

**; 输出商店名称**

**LEA DX,BUF**

**MOV AH,9**

**INT 21H**

**; 输出空格**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**L\_INPUT:**

**;输入一个字符**

**MOV AH,1**

**INT 21H**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**CMP AL,'1'**

**JE LOGIN**

**CMP AL,'2'**

**JE LIST\_GOOD**

**CMP AL,'3'**

**JE ORDER**

**CMP AL,'4'**

**JE COUNT\_PRODUCE**

**CMP AL,'5'**

**JE RANK**

**CMP AL,'6'**

**JE MODIFY\_GOOD**

**CMP AL,'7'**

**JE CHANGE\_STORE\_RUNTIME**

**CMP AL,'8'**

**JE SHOW\_CURRENT\_CS**

**CMP AL,'9'**

**JE OVER**

**JMP MENU**

**LOGIN:**

**;请输入姓名**

**LEA DX,BUF1**

**MOV AH,9**

**INT 21H**

**LEA DX,IN\_NAME**

**MOV AH,10**

**INT 21H**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**;判断是否仅输入回车，若是，直接跳到**

**CMP IN\_NAME+1,0**

**JE MENU**

**;长度不一致**

**L\_CHECK\_NAME:**

**LEA SI,IN\_NAME**

**LEA DI,BNAME**

**MOV CL,1[SI]**

**CMP CL,3**

**JNE LOGIN\_NAME\_WRONG**

**L\_LOOP\_N:**

**MOV AL,2[SI]**

**MOV BL,[DI]**

**INC SI**

**INC DI**

**CMP AL,BL**

**JNE LOGIN\_NAME\_WRONG**

**LOOP L\_LOOP\_N**

**LEA DX,BUF5**

**MOV AH,9**

**INT 21H**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**L\_IN\_PWD:**

**LEA DX,BUF2;输入密码**

**MOV AH,9**

**INT 21H**

**LEA DX,IN\_PWD**

**MOV AH,10**

**INT 21H**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**;比较密码**

**CHECK\_PWd\_LENGTH:**

**LEA SI,IN\_PWD**

**LEA DI,BPASS**

**MOV CL,1[SI]**

**CMP CL,4**

**JNE LOGIN\_PWD\_WRONG**

**L\_CHECK\_PWD:**

**MOV AL,2[SI]**

**MOV BL,[DI]**

**INC SI**

**INC DI**

**CMP AL,BL**

**JNE LOGIN\_PWD\_WRONG**

**LOOP L\_CHECK\_PWD**

**LEA DX,BUF11**

**MOV AH,9**

**INT 21H**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**JMP L\_AUTH\_1**

**L\_AUTH\_0:**

**;auth设置为0**

**MOV AUTH,0**

**JMP MENU**

**L\_AUTH\_1:**

**;auth设置为1**

**MOV AUTH,1**

**JMP MENU**

**LOGIN\_NAME\_WRONG:**

**LEA DX,BUF3;提示输出错误的信息**

**MOV AH,9**

**INT 21H**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**JMP LOGIN**

**LOGIN\_PWD\_WRONG:**

**LEA DX,BUF6**

**MOV AH,9**

**INT 21H**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**JMP L\_IN\_PWD**

**LIST\_GOOD:**

**;;提示输入商品名称**

**LEA DX,BUF4**

**MOV AH,9**

**INT 21H**

**; 输入商品的名字**

**LEA DX,IN\_GOOD**

**MOV AH,10**

**INT 21H**

**;比较长度**

**CMP IN\_GOOD+1,0**

**JNE L\_CHECK\_GOOD**

**JMP GOOD\_NOT\_FOUND**

**L\_CHECK\_GOOD:**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**MOV DX,0**

**MOV AH,0**

**LL:**

**LEA DI,GA1**

**MOV AX,DX**

**IMUL AX,21**

**ADD DI,AX**

**LEA SI,IN\_GOOD**

**MOV CL,1[SI]**

**L\_LOOP\_G:**

**MOV AL,2[SI]**

**MOV BL,[DI]**

**INC SI**

**INC DI**

**CMP AL,BL**

**JNE L\_NEXT\_GOOD**

**DEC CL**

**CMP CL,0**

**JE  MARK\_GOOD**

**JMP L\_LOOP\_G**

**L\_NEXT\_GOOD:;进入下个商品的位置**

**INC DX**

**CMP DX,30**

**JE GOOD\_NOT\_FOUND**

**JMP LL**

**; 检查是否登陆，如果登陆**

**MARK\_GOOD:**

**LEA DI,GA1**

**IMUL DX,21**

**ADD DI,DX**

**MOV GOOD,DI**

**GOOD\_FOUND:**

**LEA DX,BUF12**

**MOV AH,9**

**INT 21H**

**; 空格**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**JMP MENU**

**GOOD\_NOT\_FOUND:**

**;提示未找到信息**

**LEA DX,BUF8**

**MOV AH,9**

**INT 21H**

**; 空格**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**JMP MENU**

**ORDER:**

**CMP GOOD,0**

**JE MENU**

**MOV DI,GOOD**

**;进货数量-1**

**MOV AX,WORD PTR [DI]+15**

**; 等于0则跳转**

**CMP AX,0**

**JE  GOOD\_NO\_REMAIN**

**SUB AX,1**

**MOV WORD PTR [DI]+15,AX**

**;销量+1**

**MOV AX,WORD PTR [DI]+17**

**ADD AX,1**

**MOV WORD PTR [DI]+17,AX**

**JMP COUNT\_PRODUCE**

**GOOD\_NO\_REMAIN:**

**LEA DX,BUF9 ;提示未找到信息**

**MOV AH,9**

**INT 21H**

**; 空格**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**JMP MENU**

**COUNT\_PRODUCE:**

**MOV DX,0**

**; 计算推荐度**

**COUNT\_LOOP:**

**PUSH DX**

**LEA DI,GA1**

**MOV AX,DX**

**IMUL AX,21**

**ADD DI,AX**

**; 计算推荐度**

**MOV AX, WORD PTR [DI]+11**

**MOV CX, 128**

**IMUL CX**

**MOV SI, WORD PTR [DI]+13**

**MOV BX,0**

**MOV BL, BYTE PTR[DI]+10**

**IMUL SI, BX**

**DIV SI**

**MOV SI, AX**

**MOV AX, WORD PTR [DI]+17**

**MOV CX, 64**

**IMUL CX**

**DIV WORD PTR [DI]+15**

**ADD SI,AX**

**MOV WORD PTR [DI]+19,SI**

**POP DX**

**INC DX**

**CMP DX,30**

**JNE COUNT\_LOOP**

**LEA DX,BUF14**

**MOV AH,9**

**INT 21H**

**; 空格**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**JMP MENU**

**RANK:**

**JMP MENU**

**MODIFY\_GOOD:**

**JMP MENU**

**CHANGE\_STORE\_RUNTIME:**

**JMP MENU**

**SHOW\_CURRENT\_CS:**

**MOV AX,CS**

**MOV BX,16**

**L\_LOOP\_SHOW\_CS:**

**MOV DX,0**

**DIV BX;商在AX,余数在DX中**

**PUSH DX**

**CMP AX,0**

**JE SHOW\_CS**

**JMP L\_LOOP\_SHOW\_CS**

**SHOW\_CS:**

**MOV CX,0**

**LOOP\_SHOW:**

**CMP CX,4**

**JE SHOW\_END**

**POP SI**

**MOVZX DX,BUF10[SI]**

**MOV AH,2**

**INT 21H**

**INC CX**

**JMP LOOP\_SHOW**

**SHOW\_END:**

**LEA DX,CRLF**

**MOV AH,9**

**INT 21H**

**JMP MENU**

**OVER:**

**MOV   AH,4CH**

**MOV   AL, 0;退出码 (如0、0FFH等)**

**INT   21H**

**CODE ENDS**

**END START**

### 实验记录与分析

1. 登录，输入正确的用户名和密码后登录成功，否则登录失败

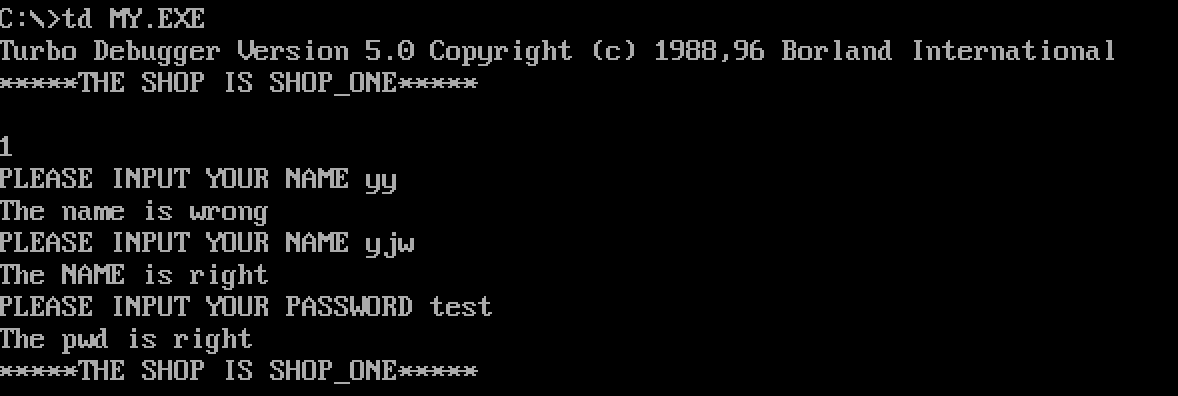


图1.16 登陆验证

1. 查找商品

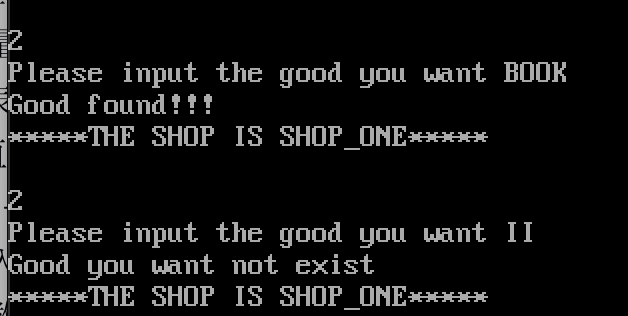


图1.17 登陆验证

1. 下订单，当没有选定商品时，直接返回，下订单后，同时修改推荐度

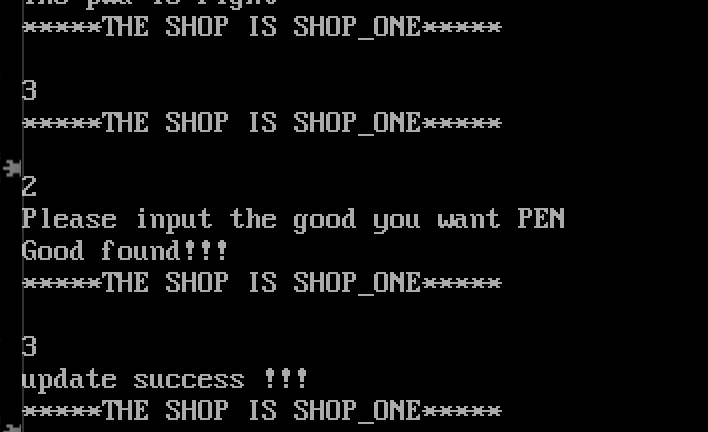


图1.18 登陆验证

1. 输出CS的16进制

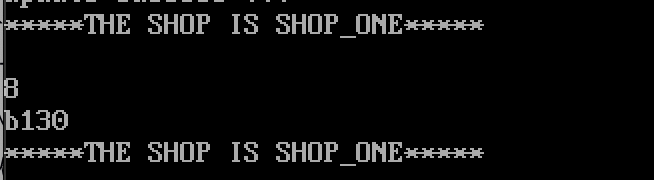


图1.19 16进制输出

## 小结

### 主要收获

通过这个实验，学会了在Doxbox环境下调试汇编程序，通过逐步调试，运行程序，对对应的汇编指令行为和寻址方式有了更直观的理解。在使用寄存器的时候一定要是否会被其他的语句的行为覆盖掉寄存器的值，比如AX,DX在做除法的时候会被覆盖的，在1.4的实验中，一开始总是出现死循环，后面debug发现是做除法的时候覆盖了DX的值

### 主要看法

汇编程序相比高级语言程序，代码逻辑更加细腻，汇编程序实现的功能没有高级程序设计语言这么直观，阅读起来需要花更多的时间，但是汇编程序然我们能够从更加高效的方式去优化程序。

# 程序优化

## 实验目的与要求

## 实验内容

## 任务2.1实验过程

### 实验方法说明

### 实验记录与分析

## 任务2.2实验过程

### 实验方法说明

### 实验记录与分析

## 小结

### 主要收获

### 主要看法

# 模块化程序设计

## 实验目的与要求

## 实验内容

## 。。。。。。。。

# 中断与反跟踪

## 实验目的与要求

## 实验内容

## 。。。。。。。。

# WIN32程序设计

## 实验目的与要求

## 实验内容

## 。。。。。。。。

# 参考文献

[1] 作者.书名.版本(第×版).译者.出版地:出版者,出版年：起页-止页

[序号] URL: 网络地址，如URL:http://www.cbs.dtu.dk/services