

**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201802班**

**学 号： U201814734**

**姓 名： 顾时禹**

**指导教师： 班老师**

**实验时段： 2020年3月19日~5月7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： 纸质版再签名

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** 1](#_Toc35376206)

[0.1 课程目标 1](#_Toc35376207)

[0.2 成绩构成 1](#_Toc35376208)

[0.3 实验任务的总体描述 1](#_Toc35376209)

[**1** **编程基础** 3](#_Toc35376210)

[1.1 实验目的与要求 3](#_Toc35376211)

[1.2 实验内容 3](#_Toc35376212)

[1.3 任务1.1实验过程 5](#_Toc35376213)

[1.3.1 实验方法说明 5](#_Toc35376214)

[1.3.2 实验记录与分析 5](#_Toc35376215)

[1.4 任务1.2实验过程 8](#_Toc35376216)

[1.4.1 实验方法说明 8](#_Toc35376217)

[1.4.2 实验记录与分析 8](#_Toc35376218)

[1.5 任务1.3实验过程 11](#_Toc35376219)

[1.5.1 实验方法说明 11](#_Toc35376220)

[1.5.2 实验记录与分析 11](#_Toc35376221)

**课程总体说明**

## 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。 |

## 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。

## 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

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. 熟悉操作系统的常用功能调用，如常用的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；
7. 掌握设计实现一个原型系统的基本方法。

## 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（不需要编写完整程序），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1.4 设计实现一个网店商品信息管理系统。**

该系统的基本需求见“2020实验报告样例”文档中的“实验任务的总体描述”。

根据系统的基本需求，可以制定如下的数据段的定义（供参考）：

BNAME DB ‘ZHANG SAN’,0 ；老板姓名（本实验要求必须是自己名字的拼音）

BPASS DB ‘test’，0，0 ，0 ；密码

AUTH DB 0 ；当前登录状态，0表示顾客状态

GOOD DB/DW … ；当前浏览商品名称或地址（自行确定）

N EQU 30

SNAME DB ‘SHOP’,0 ；网店名称，用0结束

GA1 DB ‘PEN’, 7 DUP(0) ，10 ；商品名称及折扣

DW 35，56，70，25，？ ； 推荐度还未计算

GA2 DB ‘BOOK’, 6 DUP(0) ，9 ； 商品名称及折扣

DW 12，30，25，5，？ ；推荐度还未计算

GAN DB N-2 DUP( ‘TempValue’ ,0,8，15，0，20，0，30，0，2，0，？，？) ;除了2个已经具体定义了的商品信息以外，其他商品信息暂时假定为一样的。

本次实验主要是利用分支、循环程序的结构，实现该系统的基本功能，并能熟悉全周期、全流程地设计实现一个原型系统的基本方法。本次实验要具体实现的功能要求如下：

0.主菜单界面

完整显示“实验任务的总体描述”中给出的界面信息。等待用户输入数字（可使用1号DOS系统功能调用）。对用户输入的字符进行判断，看是否是1~9的数字；是的话就转移到对应功能的程序标号，不是的话就提示错误，回到主菜单界面。

1.登录/重新登录

（1）先后分别提示用户输入姓名和密码（可使用9号DOS系统功能调用）。

（2）分别获取输入的姓名和密码（可使用10号DOS系统功能调用）。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。

（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，回到主菜单界面。

（4）进行身份认证：

（a）使用循环程序结构，比较姓名是否正确。若不正确，则跳到（c）。

（b）若正确，再比较密码是否相同，若相同，跳到（d）。

（c）若名字或密码不对，则提示登录失败，并转到“（3）”的位置。

（d）若名字和密码均正确，则将1送到AUTH变量中，回到主菜单界面。

2.查找指定商品并显示其信息

（1）提示用户输入商品名称。

（2）在商店中寻找是否存在该商品。

（3）若存在，则将商品名称或地址记录到GOOD字段中。商品信息的显示暂时不做。返回到主菜单界面。

（4）若没有找到，提示没有找到，返回到主菜单界面。

3.下订单

（1）判断当前浏览商品是否有效（GOOD不为空），若有效，判断其剩余数量是否为0，不为0则将已售数量加1，重新计算所有商品的推荐度（目前不是用子程序实现的，所以，跳转之前，要把返回地址送到指定变量中），返回主菜单界面。

（2）若无效或剩余数量为0，则提示错误，回到主菜单界面。

4.计算商品推荐度

按照给出的公式计算所有商品的推荐度，返回到指定的位置（JMP 含返回地址的指定变量）。

要求尽量避免溢出。结果只保留整数部分。

5.排名

暂不实现，直接返回主菜单界面。

6.修改商品信息

暂不实现，直接返回主菜单界面。

7.迁移商店运行环境

暂不实现，直接返回主菜单界面。

8.显示当前代码段首址

将当前代码段寄存器CS里面的内容按照16进制的方式显示到屏幕上，返回主菜单界面。

9.退出

退出本系统（可使用4CH号DOS系统功能调用）

## 任务1.1实验过程

### 实验方法说明

1. 准备实验环境，安装DOSBOX，MASM60。学习了解相关软件和指令的基本功能和应用方法。

2. 在TD中输入进行1.4题中运算的代码，观察每条代码执行后AH、BH、SF、OF、CF、ZF中的内容值。与自己预测的值比较。

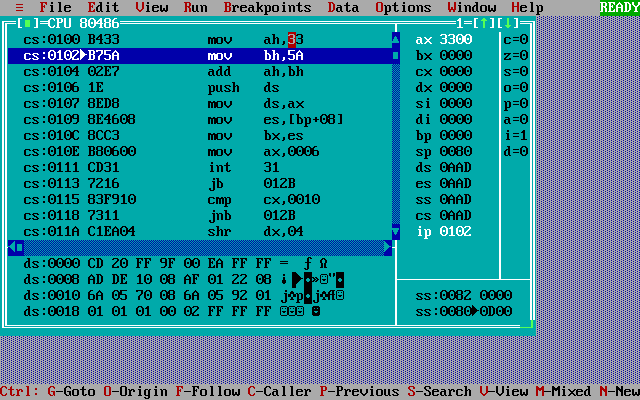
### 实验记录与分析

1. 分别对三组实验结果进行预测。预测结果如表1-3-1

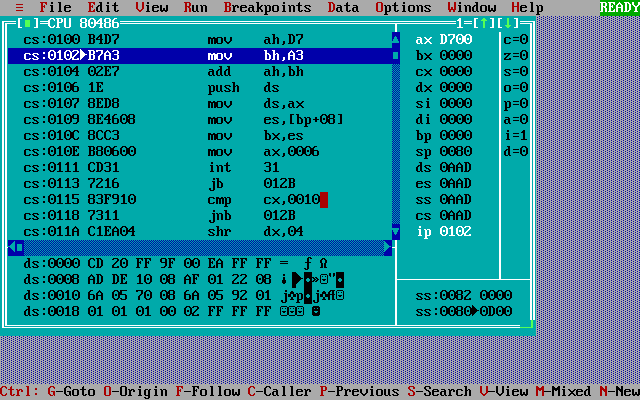
**表1.1 实验预测结果**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 组别 | AH | SF | OF | CF | ZF |
| 1 | 10001101B | 1 | 1 | 0 | 0 |
| 2 | 01111010B | 0 | 1 | 1 | 0 |
| 3 | 00001000B | 0 | 0 | 1 | 0 |

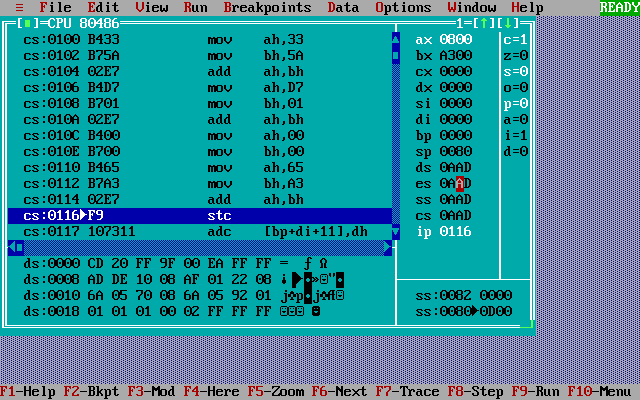
2. 逐条输入语句后运行结果如图1-3-1~1-3-3，和预期结果一致。



**图1.1 第一组运行结果**

****

**图1.2 第二组运行结果**

****

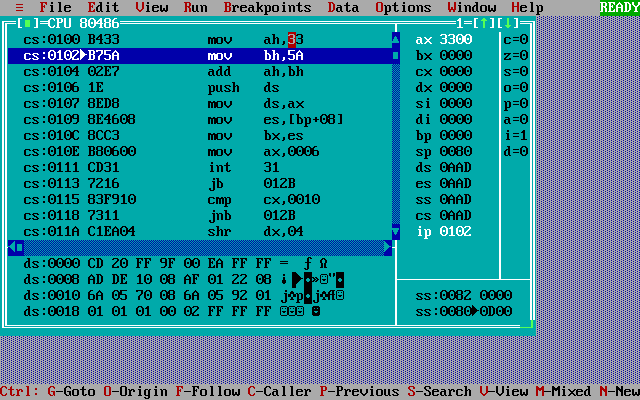
**图1.3 第三组运行结果**

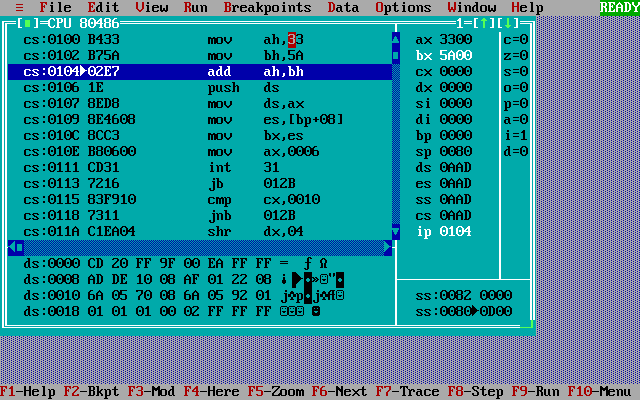
3. 在TD中选择语句后按下任意字母就可以直接输入指令，输入完毕指令后运行如果使用Run菜单会从ip=0000开始运行，但是修改的的指令不一定在0000位置，所以会引起程序崩溃。要使用F7和F8来运行代码。

4. 指令的两个参数之间要用,分割，并不是空格分隔。

5. 可以使用右键菜单点击new CS:IP来指定下一条运行的语句。

6. 在右侧的寄存器窗口可以实时监控每个寄存器的值，如图1-3-4~1-3-5所示。

**图1.4 运行第一条语句后**

**图1.5 运行第二条语句后**

## 任务1.2实验过程

### 实验方法说明

1. 按题2.3编写汇编程序。

2. 在TD中单步调试程序，观察数据段前40个字节的内容，记录并与设想的情况对比，观察是否一致

3. 运行到“MOV CX，10”和“INT 21H”之前，记录(BX),(BP), (SI), (DI)的内容，并观察他们在运行过程中的变化。

### 实验记录与分析

1. 题2.3的汇编源代码如下：

.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

2. 程序的功能为，把BUF1中的内容放到BUF2中，每字节+1后放到BUF3中，每字节+4后放入BUF4中。程序运行完毕DS段前40字节的内容应该为

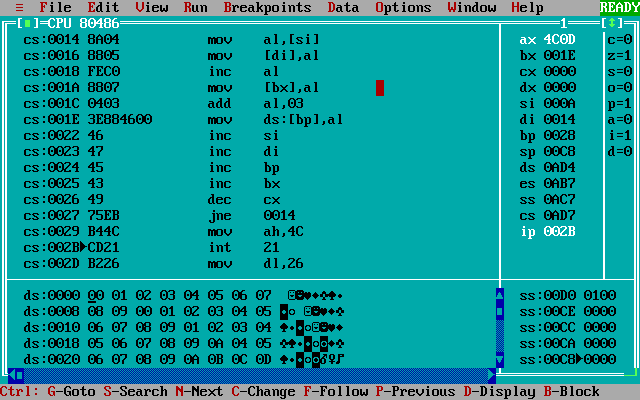
00 01 02 03 04 05 06 07 08 09

00 01 02 03 04 05 06 07 08 09

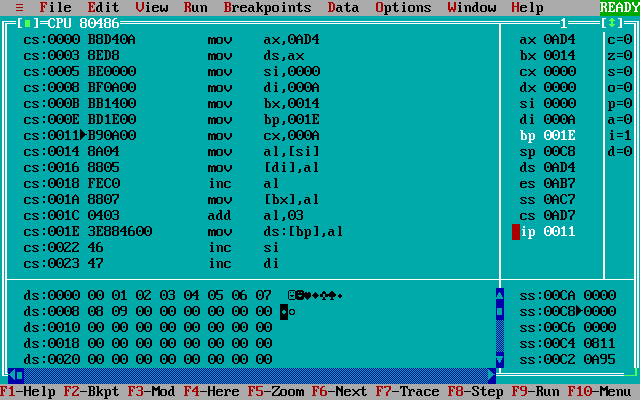
01 02 03 04 05 06 07 08 09 0A

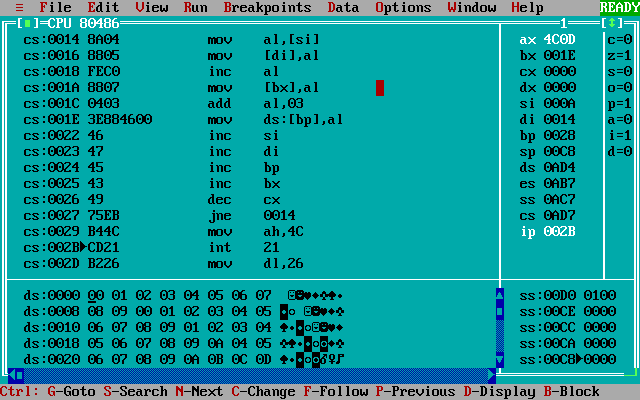
04 05 06 07 08 09 0A 0B 0C 0D

3. 程序运行完毕结果如图1-4-1，和预测的结果一致

**图1.6 程序运行结果**

4. 在MOV CX, 10指令之前，程序将BUF1,2,3,4的首字节在数据段中的偏移地址分别保存到SI,DI,BX,BP中，之后每次循环都将四个寄存器中的值加一，即为指向BUF1.2.3.4中的下一个字节。程序运行结束后四个寄存器的值应该比初始值大10。程序运行时寄存器的值如图1-4-2~1-4-3

**图1.7 MOV CX, 10前寄存器的值**

****

**图1.8 运行到INT 21H前寄存器的值**

## 任务1.3实验过程

### 实验方法说明

1. 修改2.3的程序，使用变址寻址方式访问存储单元，并使用32位存储器。

2. 在TD中单步调试程序，观察数据段前40个字节的内容，记录并与设想的情况对比，观察是否一致

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。重点观察汇编语句代码与机器码的联系。

4. 从不同的字节位置开始反汇编，观察结果的不同。理解EIP/IP的重要性。

### 实验记录与分析

1. 修改后的程序源代码如下：

.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

LOPA: MOV AL, BUF1[ESI]

MOV BUF2[ESI], AL

INC AL

MOV BUF3[ESI], AL

ADD AL, 3

MOV BUF4[ESI], AL

INC ESI

CMP ESI, 10

JNZ LOPA

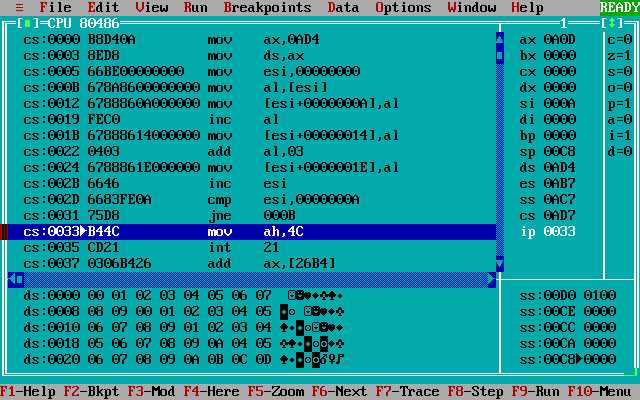
MOV AH, 4CH

INT 21H

CODE ENDS

END START

2. 实验结果和任务1.2并无区别，如图1-5-1所示

****

**图1.9 实验结果**

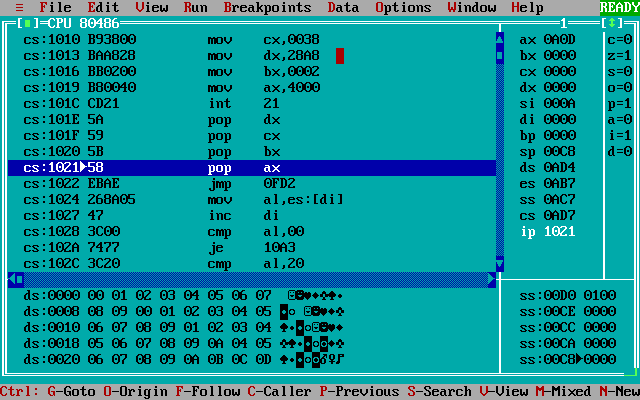
3. 本任务需要改写两个地方，一是使用32位寄存器，二是使用变址寻址。为满足要求，使用ESI存储偏移量，使用BUFX[ESI]来访问内存中的数据。

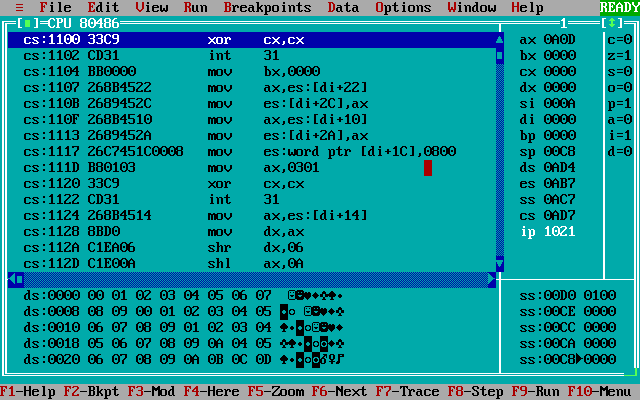
4. 通过对比反汇编代码和源代码可以得到以下结论：

DATA等标识符在汇编阶段不会被翻译为机器代码，而是会被汇编程序替换为一些偏移量，比如DATA被置换为0AD4，BUF1,2,3,4的首地址分别为0000,000A,0014,001E，和上一个任务的源代码比较，可以发现在使用32位寄存器时，机器代码的操作数也被调整为32位。

5. 通过观察生成的机器代码可以发现，同类指令的机器码有一部分共同部分，还有一部分表示了操作数，不同的寄存器也分别用不同的代码来表示

6. 如果从不同的位置开始反汇编，就会执行没有定义过的随机代码，而保证IP指针的正确保证了会执行正确的指令，如图1-4-2~1-4-3所示

**图1.10 从CS:1010位置反汇编**

**图1.11 从CS:1100位置反汇编**

## 任务1.4实验过程

### 设计思想及储存单元分配

（1）主界面

MENUT储存主界面输入命令提示字符串。

主界面部分先调用09号命令输出MENUT的内容。再调用1号命令读取一个字符，判断字符是否为1~9，是就跳转到对应功能，是0就退出程序，其他情况返回主界面。

（2）登录功能

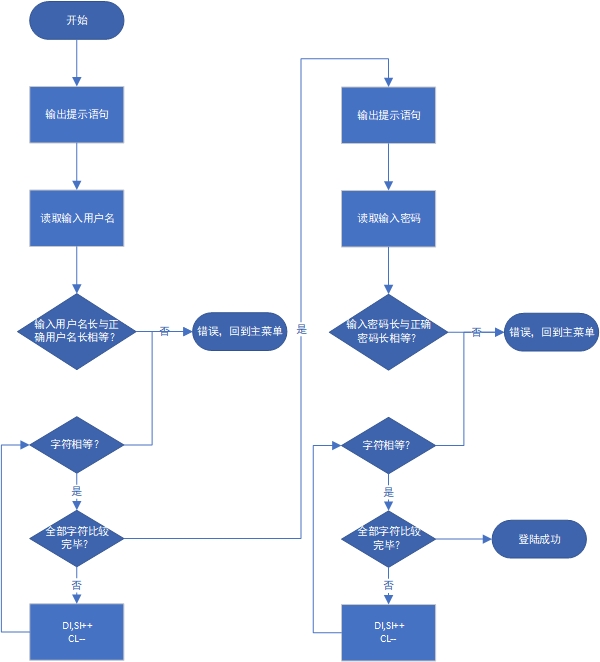
BNAME储存老板用户名，BPASS储存密码，INNAMET储存用户名输入提示信息，INNAME储存输入的用户名，WRONGNAMET储存用户名错误提示信息。INPWDT储存密码属入提示信息，INPWD储存输入的密码。WRONGPWDT储存密码错误提示信息，LOGINST储存登录成功信息。

CL用于储存输入字符串长度和正确用户名、正确密码的长度。DI和SI分别储存当前比较的输入字符和正确字符的地址。

FUNC1标号内为功能初始化部分，CHECK\_USER标号为用户名比较部分，CHECK\_PWD为密码比较部分，WRONG\_NAME，WRONG\_PWD分别为用户名和密码错误时运行的语句块，LOGIN\_S为登录成功时运行的语句。

先调用09号功能输出提示字符串，再调用10号功能读取一个字符串输入，将输入的字符串长度存入CL，与正确字符串的长度比较，不相等则直接跳转到错误时的代码语句，相等则开始比较字符串，DI初始为输入字符串第一个字节地址，SI初始为正确字符串第一个字节地址，如果两个字节不相等也跳转到错误时的代码语句，相等就将DI,SI加一，CL减一，CL为0时跳转到下一步。

登录部分流程图如图1-6-1：



**图1.12 登录部分流程图**

（2）查找商品功能

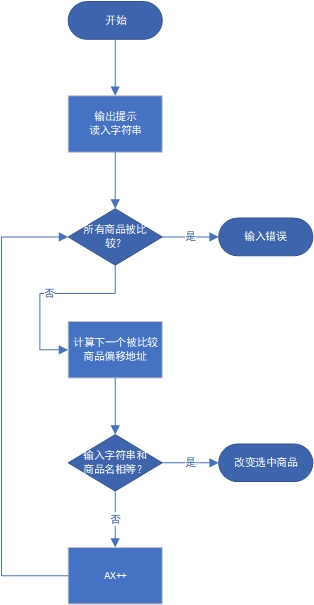
INGOODT储存商品名输入提示信息，WRONGGOODT储存错误商品名提示信息，INGOOD储存输入的商品名。CHANGEGOODT储存商品改变提示信息，GOOD储存当前选中的商品的偏移地址。

AX储存已查询的商品总数，CX用于计算当前比较的商品偏移地址时的中间储存器，CL储存输入字符串长，DI储存输入字符串第一个字节位置，SI储存当前比较的商品名的第一个字节。

FUNC2语句块为功能初始化语句，CHECK\_GOOD部分语句计算下一个比较的商品的偏移地址，CHECK\_NAME比较输入语句和当前比较的商品名是否一样，CHANGE\_GOOD将选中商品改变成输入的商品，WRONG\_GOOD是输入商品名不存在时运行的语句块。

先类似于登录部分输出提示信息和读入输入商品名，使用GA1+21\*AX计算下一个要比较的商品的偏移地址，并将偏移地址置入SI，将输入字符串第一个字符的偏移地址置入DI，输入字符串长置入CL，逐字符比较两字符串，如果有不同的字符，跳转到下一个商品，如果所有商品都被比较完，则跳转到输入错误部分。如果字符全部相同，查看被比较的商品名的下一个字符是不是0，是则将当前被比较的商品的偏移地址保存到GOOD中，不是则比较下一个商品。

查找商品部分的流程图如图1-6-2：

****

**图1.13 查找商品流程图**

（3）卖出商品功能

SELLGOODT储存商品已卖出提示信息，NOGOODT储存商品已卖空提示信息。

BX储存当前选中商品偏移地址，AX储存商品进货数，DX储存商品已售数。

FUNC3为功能初始化部分以及售空部分代码，SELL\_GOOD为成功售出部分代码。

比较当前商品是否还有余货，有则将当前商品售出数加一，输出卖出成功信息并调用功能四，如果没有则输出无货信息。

（4）计算推荐度功能

UPDATEPRIORT储存计算推荐度提示信息。

BP储存计算过的商品数，CX用作计算中间数存储器，SI储存当前商品偏移地址，BX和EBX为除数变量，AX和EAX为被除数变量，DI和EDI 用于存储计算常数，如128、2等

FUNC4为初始化部分，COMPUTE\_PRIOR为计算推荐度部分。

类似于商品查找部分计算下一个计算的商品的偏移地址。为了避免结果为0，先做乘法再做除法。由于售出数，进货价都为16位数，128为8位数，乘积可以为24位数，24位数做被除数尚6可以为24位数，所以除数要用EBX储存，为了保证高位的正确性，被除数和常数也使用32位寄存器。为了保证数据读入的正确性，从存储器中读取数据时使用16位寄存器。

（5）输出CS内容功能

使用BP作为计数器，CL用于循环左移，BX用于保存CS内容，DL为输出字符。

FUNC8为初始化代码，IS\_NUM为判断为数字时执行的代码，SHOW\_OVER输出换行。

现将CS内容移入BX中，在每次对BX内容循环左移4位，将内容储存到DX中，再将DX中内容与00001111做与运算，相当于取BX内容的高四位。然后再判断是数字还是字母，是数字则把DL内容加30H，字母则加37H，然后输出。

### 源程序

.386

STACK SEGMENT USE16 STACK

DB 200 DUP(0)

STACK ENDS

DATA SEGMENT USE16

MENUT DB 'Please Enter a Number(1~9):$'

INNAMET DB 'User Name:$'

WRONGNAMET DB 'Invalid User Name$'

INNAME DB 10

DB ?

DB 10 DUP(0)

INPWDT DB 'Password:$'

WRONGPWDT DB 'Invalid Password$'

INPWD DB 10

DB ?

DB 10 DUP(0)

LOGINST DB 'Login in$'

INGOODT DB 'Good Name:$'

WRONGGOODT DB 'Invalid Good Name$'

INGOOD DB 10

DB ?

DB 10 DUP(0)

CHANGEGOODT DB 'Good Selected$'

NOGOODT DB 'No Goods$'

SELLGOODT DB 'Good Sold$'

NOTSELEVTEDT DB 'No Goods Selected$'

UPDATEPRIOT DB 'Priority Updated$'

BNAME DB 'GU SHIYU', 0 ;老板姓名

BPASS DB '153426', 0 ;密码

AUTH DB 0 ;当前登录状态，0表示顾客状态

GOOD DW 0 ;当前浏览商品地址

N EQU 30

SNAME DB 'SHOP', 0 ;网店名称，用0结束

GA1 DB 'PEN', 7 DUP(0), 10 ;商品名称及折扣

DW 35, 56, 70, 25, ? ;进货价，销售价，进货总数，已售数量，推荐度

GA2 DB 'BOOK', 6 DUP(0), 9 ;商品名称及折扣

DW 12, 30, 25, 5, ? ;推荐度还未计算

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

;其他商品默认值

CRLF DB 0DH, 0AH, '$' ;回车

A DB 'A$' ;调试用

B DB 'B$'

DATA ENDS

CODE SEGMENT USE16

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

START: MOV AX, DATA ;程序开始

MOV DS, AX

MENU: LEA DX, MENUT ;主菜单

MOV AH, 09H

INT 21H

MOV AH, 01H

INT 21H

MOV AH, 09H

LEA DX, CRLF

INT 21H

CMP AL, '1'

JE FUNC1

CMP AL, '2'

JE FUNC2

CMP AL, '3'

JE FUNC3

CMP AL, '4'

JE FUNC4

CMP AL, '8'

JE FUNC8

CMP AL, '9'

JMP OVER

JMP MENU

FUNC1: LEA DX, INNAMET ;功能1

MOV AH, 09H

INT 21H

LEA DX, INNAME

MOV AH, 0AH

INT 21H

LEA DX, CRLF

MOV AH, 09H

INT 21H

LEA DI, INNAME+1

MOV CL, [DI]

CMP CL, 8

JNE WRONG\_NAME

LEA DI, INNAME+2

LEA SI, BNAME

CHECK\_USER: ;逐字节比较用户名

MOV AL, [DI]

CMP AL, [SI]

JNE WRONG\_NAME

INC DI

INC SI

DEC CL

JNZ CHECK\_USER

LEA DX, INPWDT

MOV AH, 09H

INT 21H

LEA DX, INPWD

MOV AH, 0AH

INT 21H

LEA DX, CRLF

MOV AH, 09H

INT 21H

LEA DI, INPWD+1

MOV CL, [DI]

CMP CL, 6

JNE WRONG\_PWD

LEA DI, INPWD+2

LEA SI, BPASS

CHECK\_PWD: ;逐字节比较密码

MOV AL, [DI]

CMP AL, [SI]

JNE WRONG\_PWD

INC DI

INC SI

DEC CL

JNZ CHECK\_PWD

JMP LOGIN\_S

WRONG\_NAME:

LEA DX, WRONGNAMET

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

JMP MENU

WRONG\_PWD:

LEA DX, WRONGPWDT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

JMP MENU

LOGIN\_S: ;登陆成功

MOV AUTH, 1

LEA DX, LOGINST

MOV AH, 09H

INT 21H

MOV AH, 09H

LEA DX, CRLF

INT 21H

JMP MENU

FUNC2: LEA DX, INGOODT ;功能2

MOV AH, 09H

INT 21H

LEA DX, INGOOD

MOV AH, 0AH

INT 21H

LEA DX, CRLF

MOV AH, 09H

INT 21H

MOV AX, -1

CHECK\_GOOD: ;选择下一个比较商品

INC AX

CMP AX, 30

JE WRONG\_GOOD

MOV CX, AX

IMUL CX, 21

ADD CX, OFFSET GA1

MOV SI, CX

LEA DI, INGOOD+1

MOV CL, [DI]

LEA DI, INGOOD+2

CHECK\_NAME: ;比较商品名

MOV BL, [DI]

CMP BL, [SI]

JNE CHECK\_GOOD

INC DI

INC SI

DEC CL

JNZ CHECK\_NAME

MOV BL, [SI]

CMP BL, 0

JE CHANGE\_GOOD

JMP CHECK\_GOOD

WRONG\_GOOD:

LEA DX, WRONGGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

JMP MENU

CHANGE\_GOOD: ;更改选中商品

MOV CX, AX

IMUL CX, 21

ADD CX, OFFSET GA1

MOV GOOD, CX

LEA DX, CHANGEGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

JMP MENU

FUNC3: MOV BX, GOOD ;功能3

CMP BX, 0

JE NO\_SELECTED

MOV AX, 15[BX]

MOV DX, 17[BX]

CMP AX, DX

JNE SELL\_GOOD

LEA DX, NOGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

JMP MENU

SELL\_GOOD:

INC DX

MOV 17[BX], DX

LEA DX, SELLGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

JMP FUNC4

NO\_SELECTED:

LEA DX, NOTSELEVTEDT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

JMP MENU

FUNC4: MOV BP, -1 ;功能4

LEA DX, UPDATEPRIOT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

COMPUTE\_PRIO:

INC BP

CMP BP, 30

JE MENU

LEA SI, GA1

MOV CX, BP

IMUL CX, 21

ADD SI, CX

MOV BX, SI

ADD BX, 13

MOV AX, [BX-2]

MOV BX, [BX]

MOVSX EAX, AX

MOVSX EBX, BX

MOV EDI, 128

IMUL EDI

IDIV EBX

MOV ECX, EAX

MOV BX, SI

ADD BX, 15

MOV AX, [BX+2]

MOV BX, [BX]

MOVSX EAX, AX

MOVSX EBX, BX

IMUL EDI

MOV EDI, 2

IDIV EDI

MOV EDX, 0

IDIV EBX

ADD EAX, ECX

MOV BX, SI

ADD BX, 19

MOV [BX], AX

JMP COMPUTE\_PRIO

FUNC8: MOV BP, 4 ;功能8

MOV BX, CS

GET\_LETTER:

MOV CL, 4;

ROL BX, CL

MOV DL, BL

AND DL, 0FH

CMP DL, 0AH

JB IS\_NUM

ADD DL, 07H

; IS\_NUM: ADD DL, 30H

MOV AH, 02H

INT 21H

DEC BP

JZ SHOW\_OVER

JMP GET\_LETTER

SHOW\_OVER:

LEA DX, CRLF

MOV AH, 09H

INT 21H

JMP MENU

OVER: MOV AH, 4CH

INT 21H

CODE ENDS

END START

### 实验步骤

（1）准备实验环境DOSBOX以及TD

（2）编译连接源程序

（3）输入正确的用户名和密码，验证能否成功登陆

（4）输入错误的用户名和密码，验证能否提示输入错误

（5）使用TD查看AUTH是否被更改

（6）验证功能二是否会更改GOOD

（7）验证功能三能否正常卖出

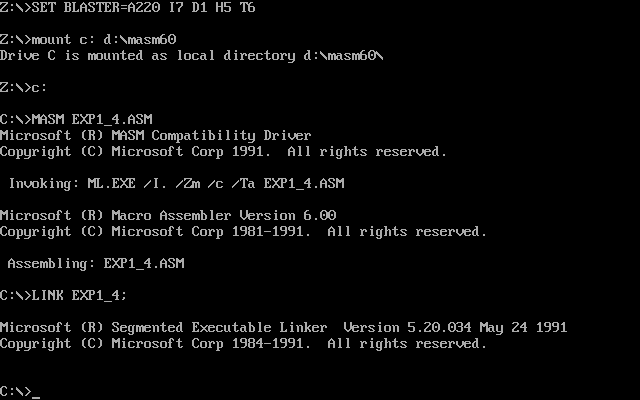
（8）验证功能三在售空时能否正常提示

（9）验证功能四推荐度计算的正确性

（10）验证功能八正确性

### 实验记录与分析

1. 编写源程序，编译，链接源程序。



**图1.14 编译链接源程序**

2. 打开TD调试源程序。

3. 验证主界面显示功能

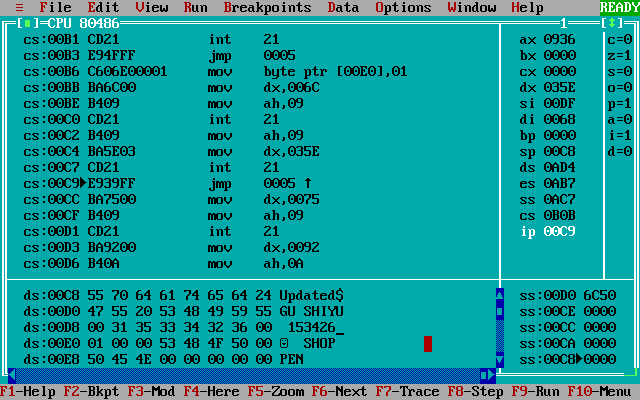


**图1.15 主界面显示**

4. 验证能否成功登陆

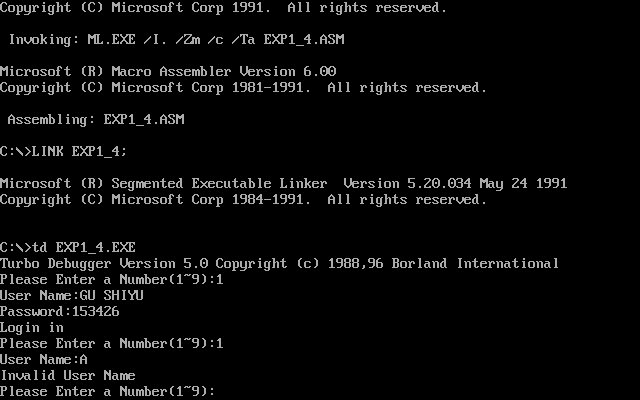


**图1.16 提示登陆成功**



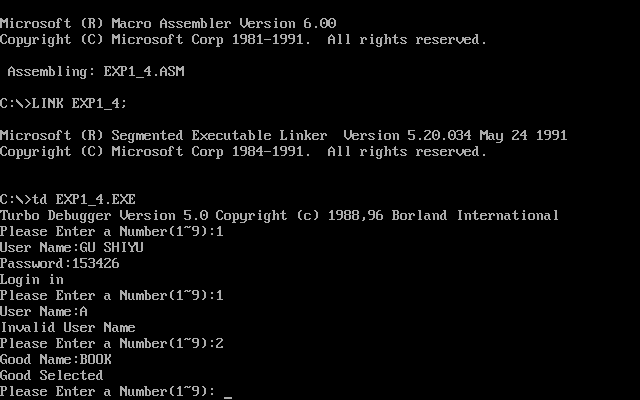
**图1.17 DS:00E0即AUTH值为1**

5. 验证登陆失败情况

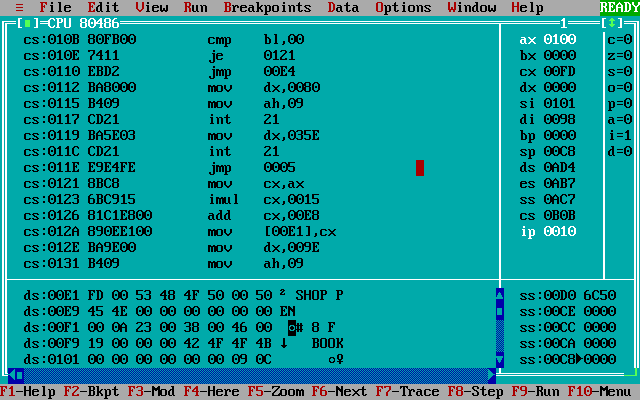


**图1.18 登陆失败**

6. 验证功能二在输入正确时的结果



**图1.19 商品已选择**



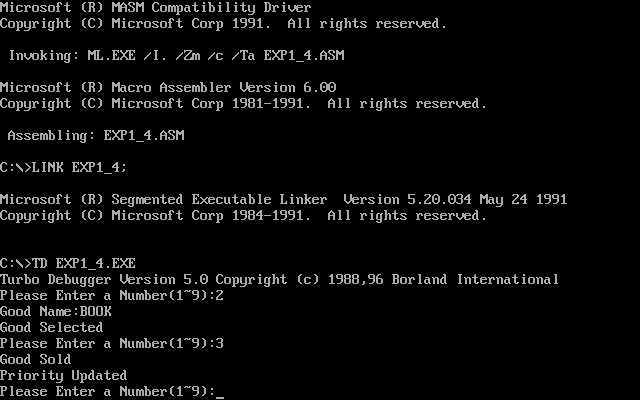
**图1.20 DS:00E1即GOOD位置内容为BOOK商品首地址**

7. 验证商品名无效的结果



**图1.21 商品名无效**

8. 验证功能三有存货的结果



**图1.22 商品卖出**

9. 验证功能三无存货的结果

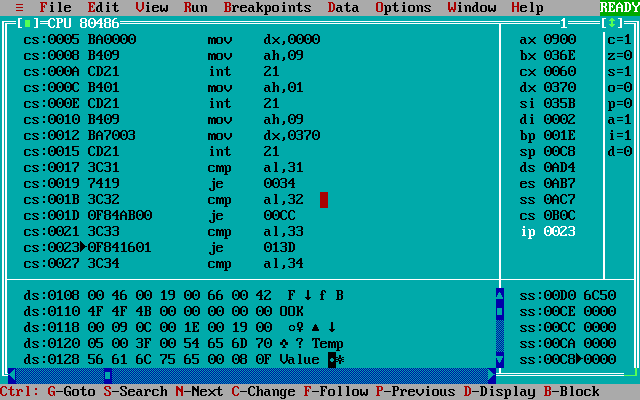


**图1.23 没有存货输出错误**

10. 验证功能四

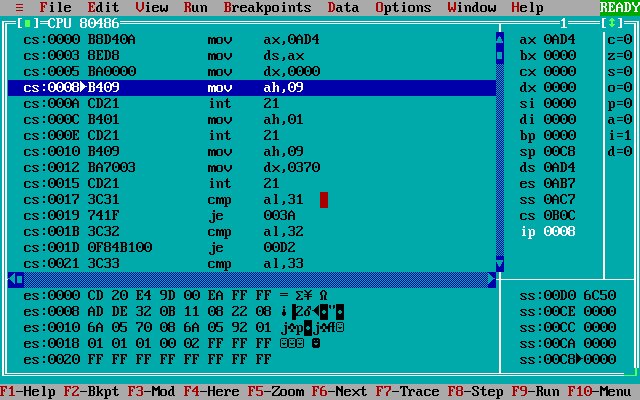


**图1.24 计算完毕**

****

**图1.25 商品1的推荐度为66H（第1行第6列）商品2的推荐度为3FH（第4行第3列）**

11. 验证功能八



**图1.26 CS内容为0B0C**



**图1.27 输出CS内容为0B0C**

11. 注意事项

（1） 系统调用功能9不会自动输出换行，要单独储存CRLF内容即0DH，0AH手动输出。

（2） 使用间接寻址和变址寻址方式时只能使用BX、BP、DI、SI四个寄存器，不能使用其他的寄存器

（3） 可以在反汇编代码中查看对应变量在数据段中定义的位置，来监视变量，也可以使用TD的监视窗口方便的监视变量。

## 小结

### 主要收获

在本次实验中，我主要学习运用了DOSBOX，MASM，TD等软件的使用方法和环境的搭建，基本语句的编写与执行，不同寻址方式的使用，机器码和汇编指令的转换，以及简单的程序设计。

在任务1.1中，主要练习了汇编开发环境的使用，主要是TD 的使用，TD提供了方便的寄存器监视，主存储器监视，代码反汇编等功能，还可以方便的直接修改代码运行，修改内存数据调试，也提供了断点调试，监视窗口等功能，熟练的使用TD可以极大的方便我们调试和编写程序。

在任务1.2中，通过观察程序的运行，我了解了汇编中存储器中的数据是如何被程序使用的，以及数据在存储器和寄存器中交换的过程。在实验中可以观察到，数据段一开始是未定义的默认值，在将我们定义的DATA段基址移动到DS中后，数据段的位置也发生了变化，而数据存储的位置也就是段基址+偏移量。

在任务1.3中，通过反汇编可以看出，变量在被翻译成机器代码时，直接使用偏移量常量替换了代码名，这有助于我们理解变量和标号的本质都是一个地址，同时通过观察和1.2反汇编代码的不同，以及比较类似语句的机器码，我初步归纳了汇编指令转化为机器码的规律。

最后任务1.4是一个简单的商店系统的程序设计，通过这个程序设计，我第一次尝试编写汇编语言程序，从中感受到了与C语言编程的相同点和不同点。相同点是编程的中心思想是类似的，都是注重过程，将要执行的功能一步一步翻译为汇编语句即可，但是相比于C语言，汇编语言几乎没有库函数，输入输出等功能固定执行也依赖于没有学习过的DOS指令，其次汇编语言的寄存器数量有限，需要思考如何分配，避免丢失需要的数据。指令的参数也有很多限制条件，需要记忆他们的规则。所有的内存管理都要手动进行，很容易出现数据越界的情况，这些都需要在编写程序时格外小心。

### 总结反思

本次实验中我存在很多问题，应当在之后的学习中加以改进。

首先是编程环境的使用，DOSBOX本身的兼容性较差，在使用时有很多不便，不如经常会丢失鼠标的输入。一开始我遇到这种问题只能重新开启DOSBOX，非常浪费时间，效率很低，经过这次的实验，我基本上熟悉了TD的各种快捷键和DOSBOX的各种指令，可以脱离鼠标进行调试，下次实验的调试效率应该会有所提高。

其次是自己汇编语言的熟练度还不高，还保留着很多C语言编程的思想，比如1.4的功能2需要遍历每一个商品，在计算商品地址的同时还要维护计数器，还要比较商品名，在寄存器的分配上没有足够的经验，经常发生丢失数据或者访问了其他内存区域的情况。

最后汇编的数字计算更加的底层，用到的命令也更复杂，需要考虑很多溢出的问题，需要以后特别注意。

### 课后问题

|  |  |  |  |
| --- | --- | --- | --- |
| 任务编号 | 序号 | 学生尝试的操作 | 结果 |
| 任务1.1 | 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？ | 将光标移到代码区，直接输入指令，按F8运行 |
| 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？ | 可以通过new ip菜单设置下一条运行语句的位置 |
| 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？ | 在立即数后加上H |
| 4 | 输入指令MOV AH, -128并执行 查看AH寄存器的内容 | D800 |
| 5 | 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？ | 在右侧有寄存器监视窗口，右键寄存器，选择change |
| 6 | 执行一条指令后，如何查看标志寄存器的值？ | 右侧有标志寄存器监视窗口 |
| 7 | 将A、B视为有符号数，运算后得到的结果、所有标志位信息，与将A、B视为无符号数，再次做同样运算后得到的结果、所有标志位信息进行比较，有何不同？ | 有符号数sf位会有变化，无符号数不会影响sf位 |
| 8 | 总结加减法对标志寄存器的影响？ | 会影响AF,OF,PF,SF,ZF,CF标志位 |
| 任务1.2和任务1.3 | 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dos运行td 程序名指令。汇编指令按照一定规则被翻译为机器代码 |
| 2 | 如何设置断点并运行到断点？ | 使用breakpoint菜单的toggle功能 |
| 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） | 使用F4或者run菜单中的run to cursor |
| 4 | 如何单步执行一条指令？（多种方法） | 使用run菜单的trace in或者使用F7 |
| 5 | 在数据区找到数据段的方法（包括找到某个变量在哪里）？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） | 使用goto菜单 |
| 6 | 修改某个指定的存储单元的值，如任务1.2中的BUF2 | 使用goto找到单元位置后直接输入数据 |
| 7 | 如何查看堆栈？ | 右下角有堆栈监视区 |
| 8 | 如何汇编一个汇编源程序并链接产生可执行代码？ | 使用MASM命令和LINK命令 |
| 9 | 如何读懂源程序在汇编过程中产生的错误？ | 在最开始有行号和错误代码提醒，可以在行号附近查找错误 |
| 10 | 使用DOS系统功能调用前应注意保存AX寄存器的值 |  |
| 11 | 查看BUF2等变量在TD中的表示形式并总结（包括在定义的地方、使用的地方等处的表现形式） | 定义的地方在机器代码中不存在，使用的地方被替换为了变量的偏移地址 |
| 12 | 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。 | 反汇编指令中不会包含标号名，会替换为他们的偏移地址 |
| 13 | 单步执行循环体2遍，查看数据段的变化 | 每个变量的前两个字节被改变了 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | 序号 | 学生尝试的操作 | 结果 |
| 任务1.4 | 1 | 如何让9号功能调用显示的信息放在自己希望的位置？ | 在需要的地方使用9号功能 |
| 2 | 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？ | 会打印出乱码 |
| 3 | 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？ | 会打印出乱码 |
| 4 | 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？ | 不允许输入 |
| 5 | 匹配姓名时，如何提高匹配速度？ | 一次比较多个字符 |
| 6 | 循环或转移时，是否有多种指令的组合方式实现？ | 可以使用不同的转移条件 |
| 7 | 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。 | 编码中会包含转移指令的目的地址 |
| 8 | 检查循环程序，是否可能出现死循环的情况。 |  |
| 任务1.4 | 1 | 是否可以简化一下计算公式。 | 加号后面的部分可以改为\*64 |
| 2 | 单步观察计算过程，结果是否正确？ |  |
| 3 | 用TD直接修改数据区的数值，使得程序对应的每个分支都能得到验证。 |  |
| 4 | 选取特殊的值，观察计算结果溢出的情况。 | 溢出可能导致死循环 |
|  |  |  |

# 程序优化

## 实验目的与要求

（1）了解程序计时的方法以及运行环境对程序执行情况的影响。

（2）熟悉汇编语言指令的特点，掌握代码优化的基本方法。

## 实验内容

**任务2.1 观察多重循环对CPU计算能力消耗的影响。**

请通过适当修改任务1.4的程序，完成如下研究：

1.请描述并实现对一段代码的执行时间进行测量的方法。该方法应能观察到程序中的一条指令发生修改时，程序完成同样功能时的执行时间的变化。

2.通过在不同软硬件运行环境下运行同一个程序，观察程序执行时间是否会随之发生变化。

对于任务1.4，现假设在双十一零点时，SHOP网店中的“Bag”商品共有m件，有m个顾客几乎同时下单购买了该商品。请模拟后台处理上述信息的过程并观察执行的时间。

上述场景的后台处理过程，可以理解为在同一台电脑上有m个请求一起排队使用任务1.4的程序。为了观察从第1个顾客开始进入购买至第m个顾客购买完毕之间到底花费了多少时间，我们让任务1.4的“下订单”调整后的代码重复执行m次，通过计算这m次循环执行前和执行后的时间差，来感受其影响。

**调整后的下订单的描述：**

【前提条件】用户查询到了商品“Bag”，在主菜单界面下选择了“3”对应的下订单功能。

**《下订单》**

（1）判断当前浏览商品是否有效（GOOD不为空），若有效，判断其剩余数量是否为0，不为0则将已售数量加1，重新计算所有商品的推荐度。

返回主菜单界面

（2）若无效或剩余数量为0，则提示错误，回到主菜单界面。

通过在不同的计算机硬件、不同的虚拟机环境、以及调试工具TD等下运行该程序，观察环境对程序执行时间的影响。

**任务2.2 对任务2.1中的汇编源程序进行优化。**

优化工作包括代码长度的优化和执行效率的优化，本次优化的重点是执行效率的优化。请通过优化m次循环体内的程序，使程序的执行时间尽可能减少10%以上（注意，在编写任务2.1的程序时，尽量不要考虑代码优化的问题）。

**注意：（1）由于本课程关注的是指令级别的影响，因此，请大家不要只考虑纯算法级别的优化。（2）在优化思想的描述中，首先要对任务2.1中可能需要优化的做法或程序片段进行分析说明，然后提出自己的优化思路。（3）对于优化问题，其实验结果的记录与分析中，必须有相同条件下优化前后的数据对比。**

## 任务2.1实验过程

### 实验方法说明

1. 准备实验环境，DOSBOX，TD等。

2. 修改任务1.4源程序，加入计时功能。

3. 通过改变M和N的值来观察计时功能的效果。

4. 通过改变DOSBOX的CPU时钟周期来观察不同硬件环境对程序运行效率的影响。

### 实验记录与分析

1. 首先在源程序中加入计时功能，更改部分的代码如下：

FUNC3: PUSH 0

MOV BX, GOOD ;功能3

CMP BX, 0

JE NO\_SELECTED

MOV AX, 15[BX]

MOV DX, 17[BX]

CMP AX, DX

JNE SELL\_GOOD

LEA DX, NOGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

TIME: POP CX

CMP CX, M

JE TIME\_E

INC CX

PUSH CX

PUSH 1

JMP FUNC4

TIME\_E: MOV AX, 1

CALL TIMER

JMP MENU

SELL\_GOOD:

MOV AX, 0

CALL TIMER

INC DX

MOV 17[BX], DX

LEA DX, SELLGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

PUSH 1

JMP FUNC4

NO\_SELECTED:

LEA DX, NOTSELEVTEDT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

MOV AX, 1

CALL TIMER

JMP MENU

FUNC4: MOV BP, -1 ;功能4

;LEA DX, UPDATEPRIOT

;MOV AH, 09H

;INT 21H

;LEA DX, CRLF

;INT 21H

COMPUTE\_PRIO:

INC BP

CMP BP, N

JE COMPUTE\_OVER

LEA SI, GA1

MOV CX, BP

IMUL CX, 21

ADD SI, CX

MOV BX, SI

ADD BX, 13

MOV AX, [BX-2]

MOV BX, [BX]

MOVSX EAX, AX

MOVSX EBX, BX

MOV EDI, 128

MUL EDI

DIV EBX

MOV ECX, EAX

MOV BX, SI

ADD BX, 15

MOV AX, [BX+2]

MOV BX, [BX]

MOVSX EAX, AX

MOVSX EBX, BX

IMUL EDI

MOV EDI, 2

DIV EDI

MOV EDX, 0

DIV EBX

ADD EAX, ECX

MOV BX, SI

ADD BX, 19

MOV [BX], AX

JMP COMPUTE\_PRIO

COMPUTE\_OVER:

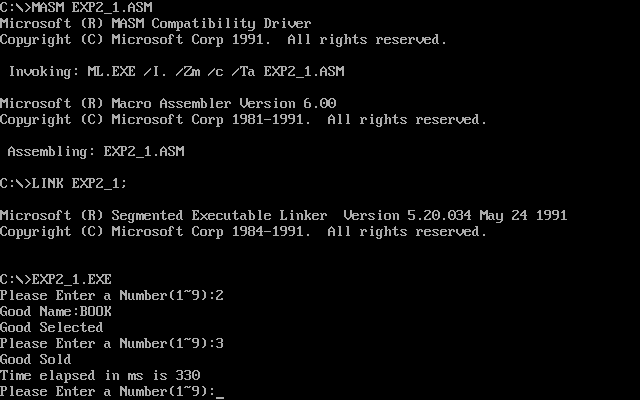
POP AX

CMP AX, 1

JE TIME

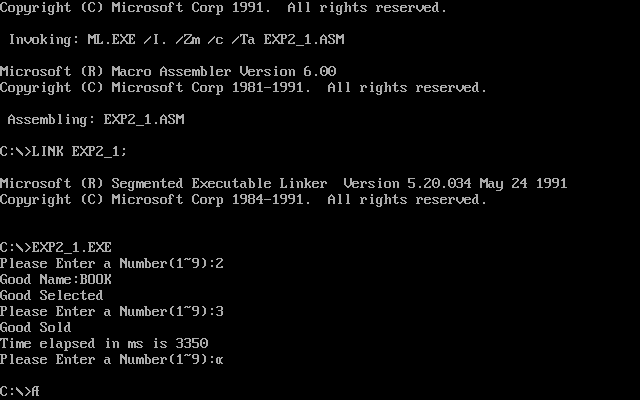
JMP MENU

2. 将商品数N设为30，计算次数设为1000，观察运行时间。



**图2.1 30商品，1000次计算所用时间**

3. 将商品数N设为30， 计算次数设为10000，观察运行时间。

****

**图2.2 30商品， 计算次10000运行时间**

4. 将商品数设为300，计算次数设为10000，观察运行时间



**图2.3 300商品，10000计算次数运行时间**

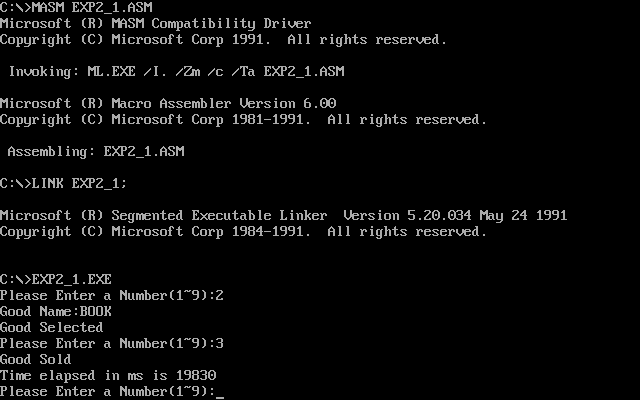
5. 可以看出程序运行所用时间与m\*n的大小基本成正比，也就是与循环体内代码的运行次数成正比。

6. 将CPU时钟频率固定为5000，商品数设为30，运行次数设为10000，观察运行时间。



**图2.4 时钟频率5000运行时间**

7. 将时钟频率设为固定8000，观察运行时间。



**图2.5 时钟频率8000运行时间**

8. 可以看出时钟频率越高，程序运行时间越短。

## 任务2.1实验过程

### 实验方法说明

1. 准备实验环境。

2. 针对循环体内指令条数进行优化，观察优化效果。

3. 使用更快的指令进行优化，观察优化效果。

4. 使用32位寄存器寻址进行优化，观察优化效果。

### 实验记录与分析

1. 将程序商品数设置为30，运行次数设置为10000,。

2. 去除循环体内的冗余语句，观察程序运行时间，优化后的代码如下：

FUNC3: PUSH 0

MOV BX, GOOD ;功能3

CMP BX, 0

JE NO\_SELECTED

MOV DX, 17[BX]

CMP DX, 15[BX]

JNE SELL\_GOOD

LEA DX, NOGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

TIME: POP CX

CMP CX, M

JE TIME\_E

INC CX

PUSH CX

PUSH 1

JMP FUNC4

TIME\_E: MOV AX, 1

CALL TIMER

JMP MENU

SELL\_GOOD:

MOV AX, 0

CALL TIMER

INC DX

MOV 17[BX], DX

LEA DX, SELLGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

PUSH 1

JMP FUNC4

NO\_SELECTED:

LEA DX, NOTSELEVTEDT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

MOV AX, 1

CALL TIMER

JMP MENU

FUNC4: MOV BP, -1 ;功能4

;LEA DX, UPDATEPRIOT

;MOV AH, 09H

;INT 21H

;LEA DX, CRLF

;INT 21H

COMPUTE\_PRIO:

INC BP

CMP BP, N

JE COMPUTE\_OVER

LEA SI, GA1

MOV CX, BP

IMUL CX, 21

ADD SI, CX

MOV BX, SI

MOVSX EAX, WORD PTR 11[BX]

MOVSX EBX, WORD PTR 13[BX]

MOV EDI, 128

MUL EDI

DIV EBX

MOV ECX, EAX

MOV BX, SI

MOVSX EAX, WORD PTR 17[BX]

MOVSX EBX, WORD PTR 15[BX]

MUL EDI

MOV EDI, 2

DIV EDI

MOV EDX, 0

DIV EBX

ADD EAX, ECX

MOV BX, SI

MOV 19[BX], AX

JMP COMPUTE\_PRIO

COMPUTE\_OVER:

POP AX

CMP AX, 1

JE TIME

JMP MENU



**图2.6 指令数量优化运行时间**

优化前的程序运行时间为3350ms，优化后的程序运行时间为2690ms，提升了约19.7%的效率

3. 将循环体中的乘法语句尽可能替换为左移语句，观察运行时间，优化后的代码如下：

FUNC3: PUSH 0

MOV BX, GOOD ;功能3

CMP BX, 0

JE NO\_SELECTED

MOV DX, 17[BX]

CMP DX, 15[BX]

JNE SELL\_GOOD

LEA DX, NOGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

TIME: POP CX

CMP CX, M

JE TIME\_E

INC CX

PUSH CX

PUSH 1

JMP FUNC4

TIME\_E: MOV AX, 1

CALL TIMER

JMP MENU

SELL\_GOOD:

MOV AX, 0

CALL TIMER

INC DX

MOV 17[BX], DX

LEA DX, SELLGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

PUSH 1

JMP FUNC4

NO\_SELECTED:

LEA DX, NOTSELEVTEDT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

MOV AX, 1

CALL TIMER

JMP MENU

FUNC4: MOV BP, -1 ;功能4

;LEA DX, UPDATEPRIOT

;MOV AH, 09H

;INT 21H

;LEA DX, CRLF

;INT 21H

COMPUTE\_PRIO:

INC BP

CMP BP, N

JE COMPUTE\_OVER

LEA SI, GA1

MOV CX, BP

IMUL CX, 21

ADD SI, CX

MOV BX, SI

MOVSX EAX, WORD PTR 11[BX]

MOVSX EBX, WORD PTR 13[BX]

SAL EAX, 7

MOV EDX, 0

DIV EBX

MOV ECX, EAX

MOV BX, SI

MOVSX EAX, WORD PTR 17[BX]

MOVSX EBX, WORD PTR 15[BX]

SAL EAX, 6

MOV EDX, 0

DIV EBX

ADD EAX, ECX

MOV BX, SI

MOV 19[BX], AX

JMP COMPUTE\_PRIO

COMPUTE\_OVER:

POP AX

CMP AX, 1

JE TIME

JMP MENU



优化后运行时间为2420ms，提升了大约10%的效率。

4. 使用32位寄存器，有更加灵活的寻址方式，优化后的代码如下：

FUNC3: PUSH 0

MOV BX, GOOD ;功能3

CMP BX, 0

JE NO\_SELECTED

MOV DX, 17[BX]

CMP DX, 15[BX]

JNE SELL\_GOOD

LEA DX, NOGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

TIME: POP CX

CMP CX, M

JE TIME\_E

INC CX

PUSH CX

PUSH 1

JMP FUNC4

TIME\_E: MOV AX, 1

CALL TIMER

JMP MENU

SELL\_GOOD:

MOV AX, 0

CALL TIMER

INC DX

MOV 17[BX], DX

LEA DX, SELLGOODT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

PUSH 1

JMP FUNC4

NO\_SELECTED:

LEA DX, NOTSELEVTEDT

MOV AH, 09H

INT 21H

LEA DX, CRLF

INT 21H

MOV AX, 1

CALL TIMER

JMP MENU

FUNC4: MOV BP, -1 ;功能4

;LEA DX, UPDATEPRIOT

;MOV AH, 09H

;INT 21H

;LEA DX, CRLF

;INT 21H

COMPUTE\_PRIO:

INC BP

CMP BP, N

JE COMPUTE\_OVER

LEA ESI, GA1

MOVSX ECX, BP

IMUL ECX, 21

ADD ESI, ECX

MOVSX EAX, WORD PTR 11[ESI]

MOVSX EBX, WORD PTR 13[ESI]

SAL EAX, 7

MOV EDX, 0

DIV EBX

MOV ECX, EAX

MOVSX EAX, WORD PTR 17[ESI]

MOVSX EBX, WORD PTR 15[ESI]

SAL EAX, 6

MOV EDX, 0

DIV EBX

ADD EAX, ECX

MOV 19[ESI], AX

JMP COMPUTE\_PRIO

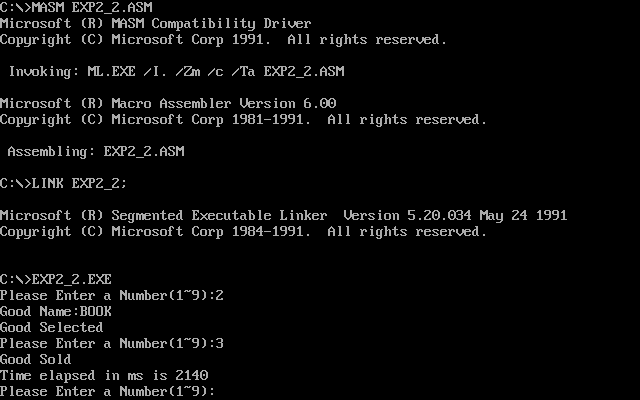
COMPUTE\_OVER:

POP AX

CMP AX, 1

JE TIME

JMP MENU



优化后的运行时间为2140ms，提升了大约11.6%的效率。

5. 可以看出，减少循环体内的语句数量是最直接的优化程序运行效率的方式，使用32位寄存器的根本目的也是通过更佳灵活的寻址方式来减少语句的数量。通过使用更快的语句也可以加快程序运行的效率，但是优化的限制比较多，优化的效果也小于直接减少指令条数，所以一般不要为了使用更快的指令而额外增加指令。

## 实验小结

### 主要收获

本次实验主要学习了计时程序的使用，以及如何在指令级别优化程序。通过本次实验，我意识到了指令级别的优化与算法级别的优化有很多不同，指令级别的优化在思路上倾向于优化到每一条语句，尽可能的使语句的数量减少，这需要我们对指令的使用特别熟悉，并掌握很多优化的技巧，比如使用LEA指令加基址变址寻址来替换一些简单的表达式。还需要我们对指令的性能有着足够的认识，了解哪些指令的运行时间比较快。在这次实验中，我对指令性能有了更加深刻的认识，也学到了一些指令使用的技巧。

### 课后问题

|  |  |  |  |
| --- | --- | --- | --- |
| 任务编号 | 序号 | 学生尝试的操作 | 结果 |
| 任务2.1 | 1 | 改变循环程序的设计和循环次数，对CPU资源消耗的影响有多大？ | 程序设计越复杂，循环次数越多，CPU资源消耗越大 |
| 2 | 内循环体中若有信息显示的代码（比如2号或9号功能调用），程序执行时间会有多大影响？ | 信息显示的代码相比于普通代码消耗的时间要稍大 |
| 3 | m和n的最大值受到哪些限制？当商品种类的数量很多时（比如10000以上），如何处理一个数据段空间不够的问题？ | 受到段最大空间的影响，可以使用代码段来储存变量，也可使用附加段 |
| 任务2.2 | 1 | 汇编语言程序的优化可以从哪些方面进行？总结不同类别的优化措施对效率的影响程度。 | 可以从指令数量，使用更高效的指令来优化程序。指令数量对程序运行效率的影响最为明显，指令运行效率其次 |
| 2 | 哪些指令是需要优化的关键性指令？ | 在循环体中的指令 |
| 3 | 循环体中用子程序调用与不用子程序调用会有多大影响？ | 调用子程序会增加运行时间，因为要保护和恢复现场，还增加了跳转指令 |
| 4 | 利用32位寄存器书写寻址方式具有较多的灵活性。你是否有通过此途径优化指令语句的例子？ | 可以使用LEA加基址变址寻址方式来代替一些简单的算式，还可以节省一两个寄存器，减少一些MOV指令 |