

**课 程 实 验 报 告**

**课程名称： 汇编语言程序设计实践**

**专业班级：**

**学 号：**

**姓 名：**

**指导教师：**

**实验时段： 2021年3月29日~5月21日**

**实验地点： 南一楼803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：2021.6.7

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](#_Toc74164023)

[0.1 课程目标 1](#_Toc74164024)

[0.2 成绩构成 1](#_Toc74164025)

[0.3 实验任务的总体描述 2](#_Toc74164026)

[1 编程基础 3](#_Toc74164027)

[1.1实验目的与要求 3](#_Toc74164028)

[1.2 实验内容 3](#_Toc74164029)

[1.3 任务1.1实验过程 3](#_Toc74164030)

[1.3.1 实验方法说明 3](#_Toc74164031)

[1.3.2 实验记录与分析 3](#_Toc74164032)

[1.4 任务1.2实验过程 5](#_Toc74164033)

[1.4.1 实验方法说明 5](#_Toc74164034)

[1.4.2 实验记录与分析 5](#_Toc74164035)

[1.5 任务1.3实验过程 7](#_Toc74164036)

[1.5.1 设计思想及存储空间分配 7](#_Toc74164037)

[1.5.2 流程图 7](#_Toc74164038)

[1.5.3 源程序 8](#_Toc74164039)

[1.5.4 实验步骤 12](#_Toc74164040)

[1.5.5 实验记录与分析 13](#_Toc74164041)

[1.6 小结 14](#_Toc74164042)

[1.6.1 主要收获 14](#_Toc74164043)

[1.6.2 主要看法 14](#_Toc74164044)

[2 程序优化 16](#_Toc74164045)

[2.1 实验目的与要求 16](#_Toc74164046)

[2.2 实验内容 16](#_Toc74164047)

[2.3 任务2.1实验过程 16](#_Toc74164048)

[2.1.1 实验方法说明 16](#_Toc74164049)

[2.1.2 实验记录与分析 17](#_Toc74164050)

[2.4 小结 18](#_Toc74164051)

[2.4.1 主要收获 18](#_Toc74164052)

[2.4.2 主要看法 19](#_Toc74164053)

[3 模块化程序设计 20](#_Toc74164054)

[3.1 实验目的与要求 20](#_Toc74164055)

[3.2 实验内容 20](#_Toc74164056)

[3.3 任务3.1实验过程 20](#_Toc74164057)

[3.3.1 设计思想及存储空间分配 20](#_Toc74164058)

[3.3.2 流程图 21](#_Toc74164059)

[3.3.3 源程序 21](#_Toc74164060)

[3.3.4 实验步骤 22](#_Toc74164061)

[3.3.5 实验记录与分析 22](#_Toc74164062)

[3.4 任务3.2实验过程 24](#_Toc74164063)

[3.4.1 实验方法说明 24](#_Toc74164064)

[3.4.2 试验记录与分析 24](#_Toc74164065)

[3.5 小结 25](#_Toc74164066)

[3.5.1 主要收获 25](#_Toc74164067)

[3.5.2 主要看法 26](#_Toc74164068)

[4 中断与反跟踪 27](#_Toc74164069)

[4.1 实验目的与要求 27](#_Toc74164070)

[4.2 实验内容 27](#_Toc74164071)

[4.3 任务4.1实验过程 27](#_Toc74164072)

[4.3.1 设计思想 27](#_Toc74164073)

[4.3.2 流程图 28](#_Toc74164074)

[4.3.3 源程序 28](#_Toc74164075)

[4.3.4 实验步骤 29](#_Toc74164076)

[4.3.5 实验记录与分析 29](#_Toc74164077)

[4.4 任务4.2实验过程 30](#_Toc74164078)

[4.4.1 实验方法说明 30](#_Toc74164079)

[4.4.2 实验记录与分析 31](#_Toc74164080)

[4.5 任务4.3实验过程、 31](#_Toc74164081)

[4.5.1 实验方法说明 31](#_Toc74164082)

[4.5.2 实验记录与分析 31](#_Toc74164083)

[4.6 小结 32](#_Toc74164084)

[4.6.1 主要收获 32](#_Toc74164085)

[4.6.2 主要看法 33](#_Toc74164086)

[5. 16/32/64位编程比较 34](#_Toc74164087)

[5.1 实验目的与要求 34](#_Toc74164088)

[5.2 实验内容 34](#_Toc74164089)

[5.3 任务5.1实验过程 34](#_Toc74164090)

[5.4.1 实验方法说明 34](#_Toc74164091)

[5.4.2 实验记录与分析 34](#_Toc74164092)

[5.4 任务5.2实验过程 35](#_Toc74164093)

[5.4.1 实验方法说明 35](#_Toc74164094)

[5.4.2 ·实验记录与分析 35](#_Toc74164095)

[5.5 任务5.3实验过程 35](#_Toc74164096)

[5.5.1 实验方法说明 35](#_Toc74164097)

[5.5.2 实验记录与分析 35](#_Toc74164098)

[5.6 小结 36](#_Toc74164099)

[5.6.1 主要收获 36](#_Toc74164100)

[5.6.2 主要看法 36](#_Toc74164101)

# 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）通过模块化方法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实现。

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（16/32/64位编程比较）**安排4个课内学时，熟悉在不同软硬件平台上移植实现指定功能的基本方法。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。

# 1 编程基础

## 实验目的与要求

（1）熟练掌握程序开发平台(VS2019) 的基本用法，包括程序的编译、链接和调试；掌握DOSBox下16位汇编语言程序开发工具的基本用法；

（2）熟悉编程的基础知识，包括数据在计算机内的表现形式、寻址方式、常用指令等；

（3）熟悉程序运行的基本原理；

（4）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；

（5）加深对转移指令及一些常用的汇编指令的理解；

（6）掌握设计实现一个原型系统的基本方法。

## 1.2 实验内容

**任务1.1：**利用VS2019的反汇编等功能了解C语言和汇编语言的相关关系。

**任务1.2：**利用VS2019进行直接进行汇编语言调试和运行；在DOSBOX下改造、运行和观察上述程序。

## 1.3 任务1.1实验过程

### 1.3.1 实验方法说明

1. 利用VS2019创建和编译给定C语言代码。

2. 在反汇编窗口下观察C语言与汇编语言的对应关系。

3. 在监视窗口和内存窗口中观察变量的值在内存中的对应形式。

4. 观察有符号数和无符号数的加减运算过程，注意标志寄存器的设置和变化。

### 1.3.2 实验记录与分析

1. 当勾选“显示符号名”时，对变量的调用会使用[变量名]表示；而不勾选时则会直接使用地址。

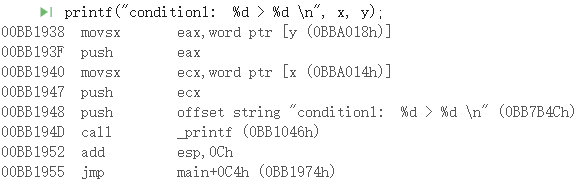


图1.1 勾选“显示符号名”

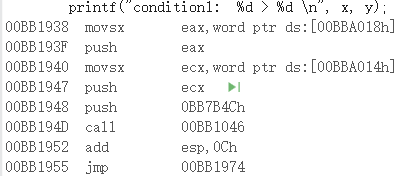


图1.2 不勾选“显示符号名”

2. 整形数在内存中保存为补码，占用四个字节，且采取小端存储的方式。



图1.3 x和y在内存中的存储形式

数组在内存中占用一片连续的存储空间



图1.4 str[10] = "The end!"的存储形式



图1.5 a[5] = { 1, 2, 3, 4, 5 }的存储形式

3. 对于有符号数的运算，相应汇编代码使用了符号拓展传送指令；而无符号数的运算使用的则是无符号拓展传送指令。

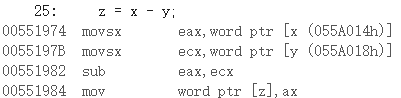


图1.6 有符号数的减法运算

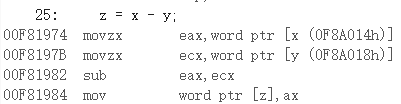


图1.7 无符号数的减法运算

4. 执行比较语句和加法语句时标志寄存器的设置如图1.8、1.9所示。区别在于，比较语句中辅助进位标志为0，而加法语句中为1，且加法语句中符号标志可能会发生改变。



图1.8 比较语句中标志寄存器设置



图1.9 加法语句中标志寄存器设置

## 1.4 任务1.2实验过程

### 1.4.1 实验方法说明

1. 在VS2019下编译运行给定汇编语言代码，观察反汇编窗口，寄存器窗口、监视窗口、内存窗口等。观察并分析存储规律。

2. 在DOSBOX下改造该汇编程序，体验16位段程序的编译、链接和调试过程。

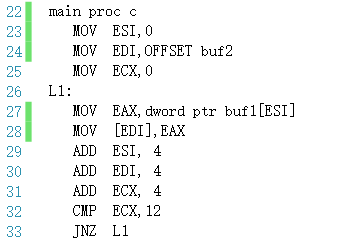
### 1.4.2 实验记录与分析

1. 数据段在内存中占用一片连续的存储空间，且不同变量间无分隔。



图1.10 数据段在内存中的存储

2. 将buf1的寻址方式改为变址寻址，如图1.11所示



图表 1.11 修改后的代码段

3. DOSBOX下，将源代码转化为可执行文件需要经过编译、链接的过程，运行后的输出会直接显示在命令行界面中。如图1.12、1.13、1.14所示。



图 1.12 DOSBOX下的编译过程

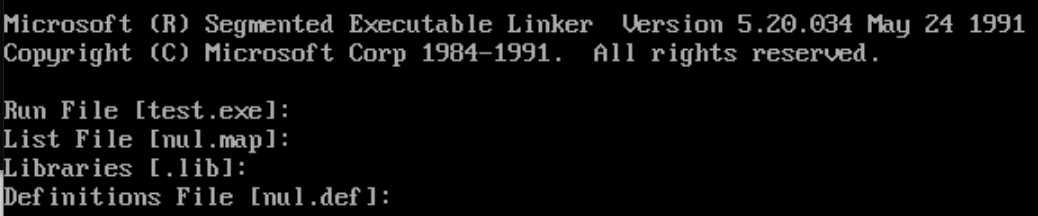


图 1.13 DOSBOX下的链接过程

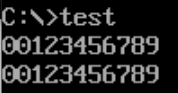


图 1.14 DOSBOX下的运行结果

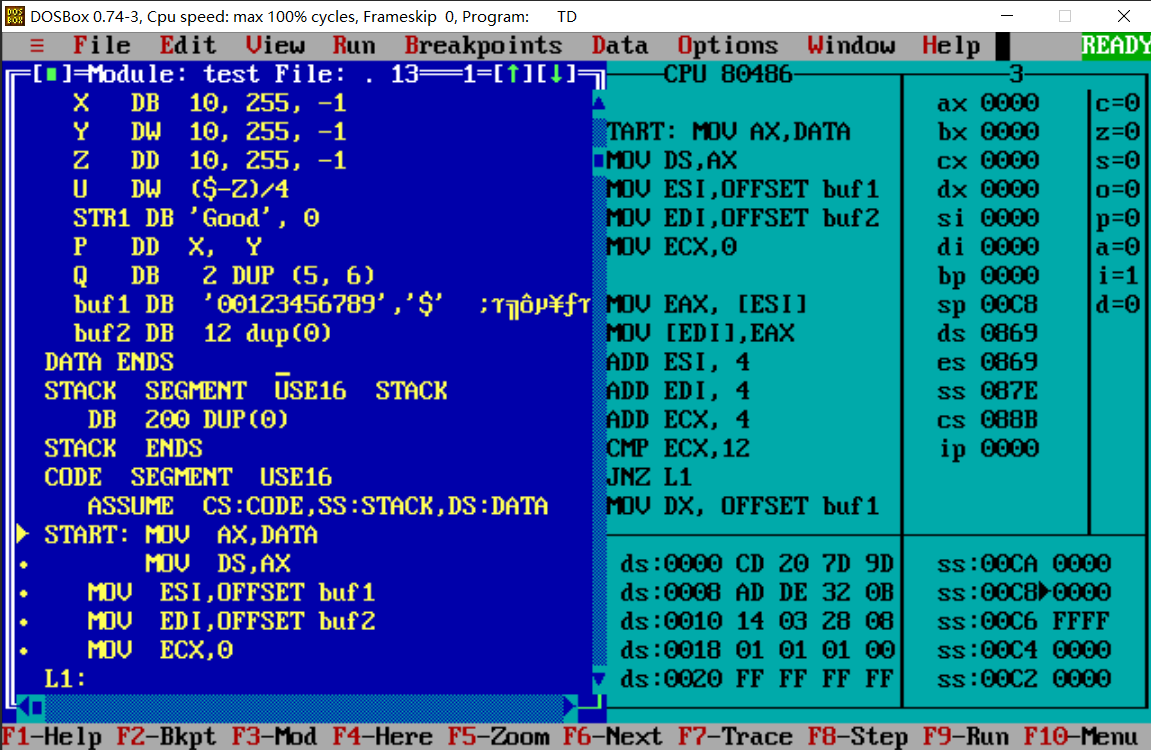


图 1.15 TD.exe下的调试窗口

## 1.5 任务1.3实验过程

### 1.5.1 设计思想及存储空间分配

1. 对于用户名和密码的输入，可以采用逐字节对比的方法。菜单的构建可利用循环的结构；对应功能部分则采用相应的计算来实现。

2. 存储单元分配

字节变量BNAME、BPASS用于存储正确的用户名和密码，INAME、IPASS用于保存从键盘输入的用户名和密码。

字节变量STRn用于存储可能用到的提示信息。

变量GAn用于存储商品相关信息。

### 1.5.2 流程图

程序主要部分结构和过程较为简单，故省略。难点为利润率计算部分，其流程图如图1.12所示。



图 1.16 利润率计算部分的流程图

### 1.5.3 源程序

.686P

.model flat, stdcall

 ExitProcess PROTO STDCALL :DWORD

 printf proto c:ptr sbyte,:vararg

 scanf proto c:ptr sbyte,:vararg

 strcmp proto c:ptr sbyte,:vararg

 search proto

 includelib  kernel32.lib

 includelib  libcmt.lib

 includelib  legacy\_stdio\_definitions.lib

 .DATA

BNAME  DB 'WANGJIASHUN',0

INAME   DB 20 DUP(0)

BPASS  DB 'U201914983',0

IPASS   DB 20 DUP(0)

N  EQU  30

choice  DW ?

num DW ?

lpfmt DB '%s',0

numtype DB '%hd',0

chartype DB '%c',0

STR1 DB '用户名: ',0

STR2 DB '密码: ',0

STR3 DB '用户名不存在',0

STR4 DB '密码错误',0

STR5 DB '登录成功',0AH,0

STR6 DB '请选择要执行的操作',0AH,0

STR7 DB '1.查找商品  2.出货  3.补货  4.计算利润率',0AH,0

STR8 DB '商品名称：%s  进货价：%hd  销售价：%hd  进货数量：%hd  已售数量：%hd  利润率：%hd%%',0AH,0

STR9 DB '未找到目标商品',0AH,0

STR10 DB '请输入商品名: ',0

STR11 DB '请输入出货量: ',0

STR12 DB '操作失败，出货量应小于剩余数量',0AH,0

STR13 DB '操作成功',0AH,0

STR14 DB '请输入补货量: ',0

GA1  DB  'PEN',7 DUP(0)

    DW  15,20,70,25,?

GA2  DB  'PENCIL',4 DUP(0)

    DW  2,3,100,50,?

GA3  DB  'BOOK',6 DUP(0)

    DW  30,40,25,5,?

GA4  DB  'RULER',5 DUP(0)

    DW  3,4,200,150,?

GAN  DB  N-4 DUP('TempValue',0,15,0,20,0,30,0,2,0,?,?)

IGA  DB  10 DUP(0)

.stack 200

.code

main proc c

   invoke printf,offset STR1

   invoke scanf,offset lpfmt,offset INAME

   invoke strcmp,offset BNAME,offset INAME

   cmp eax,0

   jnz INPUT\_ERROR1

   invoke printf,offset STR2

   invoke scanf,offset lpfmt,offset IPASS

   invoke strcmp,offset BPASS,offset IPASS

   cmp eax,0

   jnz INPUT\_ERROR2

   invoke printf,offset STR5

MAIN\_MENU:

   invoke printf,offset STR6

   invoke printf,offset STR7                 ;账号密码输入与菜单界面

   invoke scanf,offset numtype,offset choice

   cmp choice,1

   jz CHOICE1

   cmp choice,2

   jz CHOICE2

   cmp choice,3

   jz CHOICE3

   cmp choice,4

   jz CHOICE4

CHOICE1:

   invoke printf,offset STR10

   invoke scanf,offset lpfmt,offset IGA

   call search

   cmp eax,0

   jz SEARCH\_ERROR

   invoke printf,offset STR8,eax,word ptr [eax+10],word ptr[eax+12],WORD PTR [EAX+14],WORD PTR [EAX+16],WORD PTR[EAX+18]

   jmp MAIN\_MENU

SEARCH\_ERROR:

   invoke printf,offset STR9

   jmp MAIN\_MENU

CHOICE2:

   invoke printf,offset STR10

   invoke scanf,offset lpfmt,offset IGA

   call search

   cmp eax,0

   jz SEARCH\_ERROR

   mov esi,eax

   invoke printf,offset STR11

   invoke scanf,offset numtype,offset num

   mov bx,[esi+16]

   add bx,num

   cmp bx,[esi+14]

   jns OK1

   invoke printf,offset STR12

   jmp MAIN\_MENU

OK1:

   mov [esi+16], bx

   invoke printf,offset STR13

   jmp MAIN\_MENU

CHOICE3:

   invoke printf,offset STR10

   invoke scanf,offset lpfmt,offset IGA

   call search

   cmp eax,0

   jz SEARCH\_ERROR

   mov esi,eax

   invoke printf,offset STR14

   invoke scanf,offset numtype,offset num

   mov bx,[esi+14]

   add bx,num

   mov [esi+14],bx

   invoke printf,offset STR13

   jmp MAIN\_MENU

CHOICE4:

   mov esi,offset GA1

CAL\_LOOP:

   movsx eax,word ptr [esi+12]

   movsx ebx,word ptr [esi+16]

   imul eax,ebx

   movsx ecx,word ptr [esi+10]

   movsx edx,word ptr [esi+14]

   imul ecx,edx

   sub eax,ecx

   imul eax,100

   CDQ

   idiv ecx

   mov [esi+18],ax

   add esi,20

   cmp esi,offset GA1+580

   js CAL\_LOOP

   invoke printf,offset STR13

   jmp MAIN\_MENU

INPUT\_ERROR1:

   invoke printf,offset STR3

   jmp PROGRAM\_EXIT

INPUT\_ERROR2:

   invoke printf,offset STR4

   jmp PROGRAM\_EXIT

PROGRAM\_EXIT:

   invoke ExitProcess,0

main endp

search proc

   mov esi,offset GA1

SEARCH\_LOOP:

   invoke strcmp,offset IGA,esi

   cmp eax,0

   jz SEARCH\_SUCCESS

   cmp esi,OFFSET GA1+580

   jns SEARCH\_FAIL

   add esi,20

   jmp SEARCH\_LOOP

SEARCH\_SUCCESS:

   mov eax,esi

   ret

SEARCH\_FAIL:

   mov eax,0

   ret

search endp

end

### 1.5.4 实验步骤

1. 准备上机实验环境。

2. 在Visual Studio2019（以下简称VS）中创建项目，新建源文件main.asm，在VS中编辑和调试运行，观察调试信息，遇到错误时进行修改，直到编译通过。

3. 先进行菜单部分的编辑调试，实现对选项输入的正确跳转效果。

4. 进行查找和输出部分的编辑调试，达到能够正确在商品中进行查找指定商品名并输出信息的效果。其中查找操作在其他功能中都有涉及，故将其独立写为一个子程序，以供调用。

5. 进行出货部分的编辑调试，要能够正确地对指定商品比较剩余货物与待出货物数量关系，并正确地改变已出货数量。

6. 进行进货部分的编辑调试，达到正确将指定商品进货量及进行修改的目标。

7. 进行利润率计算部分的编辑调试，要能够正确地对利用给定公式，用循环的方式求求解每种商品的利润率，并修改相应数值。同时注意对正负数值的处理是否正确。

### 1.5.5 实验记录与分析

1. 实验环境为AMD Ryzen 5 2500U with Radeon Vega Mobile Gfx 2.00 GHz，8G内存（5.9G可用）；Windows10下Visual Studio2019 Community，版本16.7.5

2. 菜单实现后，在调试查找输出部分时，程序直接跳过查找内容输入部分，检查后发现是菜单输入变量choice类型（字类型）与类型说明符（%d）不符。

3. 查找完毕的输出部分，编译器报错A2114 ：INVOKE argument type mismatch : argument : 0。是由于[eax+10]前未加word ptr。

4. 在利润率计算部分，语句idiv ecx执行时出现Integer overflow错误，查阅资料得知，执行此指令前可先使用CDQ指令将EAX的第31位复制到EDX的所有位上，从而防止当EAX内值为负时，除法运算导致被除数数值改变。

5. 上述调试完毕后，运行并测试程序是否满足要求。



图 1.17 登录报错功能正常

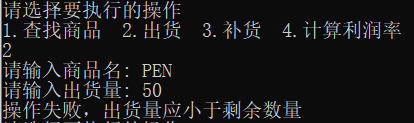


图 1.18 出货报错正常

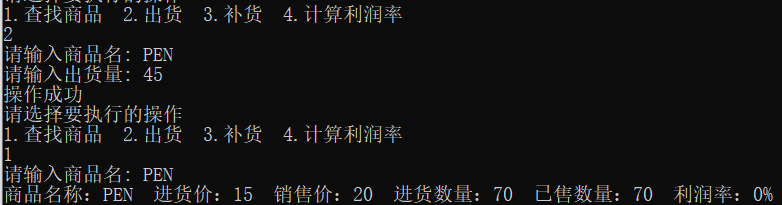


图 1.19 出货功能正常

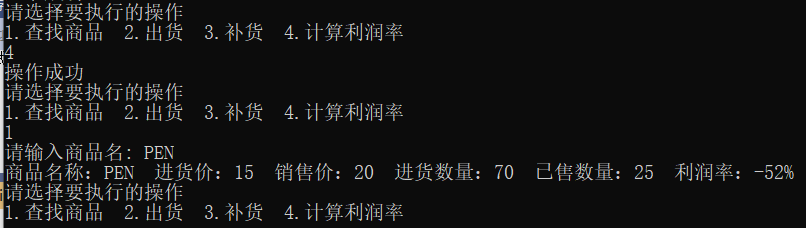


图 1.20 利润率计算功能正常

如图1.17，1.18，1.19，1.20所示，主要功能实现状况良好（部分功能测试截图略去），基本达成实验要求。

## 1.6 小结

### 1.6.1 主要收获

本次实验的主要内容是认识和了解汇编程序，学习汇编程序的编译、调试，以及利用学到的汇编指令实现分支、循环等结构

在任务1.1中，将我们熟悉的C语言进行反汇编、并打开内存窗口、寄存器窗口，让我了解到C指令与汇编指令间的对应关系，一条高级语言的语句可能对应多条汇编指令。内存窗口展示了变量在内存中的存储形式，也认识到C语言计算中整形的类型提升现象在汇编语言中的反映。

在任务1.2中，在VS中编写和调试汇编程序，使我熟悉在VS中对汇编程序的操作。在反汇编窗口，可见大多数语句与反汇编得到的语句相同切且是一一对应的关系。在对内存的观察中可以看到，数据段的所有数据在内存中占用一大片连续的存储空间，且不同变量之间无分隔。在DOSBOX的操作中，我体验到在DOS界面下对汇编程序进行编译、链接形成EXE文件的过程。同时，在用TD.exe对程序进行调试的过程也让我感受到工具更新对开发的巨大帮助。

在任务1.3中，通过编写带有菜单的程序的过程中，让我了解到利用跳转指令实现循环结构和分支结构的方法，也体会到高级语言在此方面的优越性。此外，在程序编写过程中，多次出现的寻址问题也深化了我对汇编程序中寻址方式的认识。在利润率的计算过程中，除法是比较特别的一个——被除数在计算中会被扩展，DX和AX或EDX与EAX会共同构成一个被除数。在这一前提下，CDQ指令的使用便是保证计算结果正确的关键因素。另外，在汇编程序中，一个指令后不能接两个变量参数，这点也是值得注意的。

汇编程序中的一个重要部分是对寄存器的使用。在实验过程中，常常出现调用函数后某些寄存器的值发生变化的现象。这是由于调用了C库函数，而这些函数通常会改变EAX，ECX，EDX的值。因此在调用C库函数时应避开实验这些寄存器。

### 1.6.2 主要看法

本次实验中主要采用的是VS2019这一IDE，其好处在于简化了编译调试的过程，同时操作方便，界面也更为美观。但是其在标志寄存器窗口中使用的名称与教材有所差异，给使用造成了一定的不便。

而在1.2中使用的DOSBOS配合相应工具，可以模拟DOS的环境，便于体验16位环境下的相关操作，但是命令行的操作方式并不够方便。TD.exe的调试虽然已有类似于VS的窗口界面和鼠标的操作方式，但是其操作仍与windows下的日常操作逻辑不同，使用起来较为不适应。

在实验中，对于同样的操作，C语言下的语句更加简单和贴近自然语言，而汇编语句则可能需要很多语句和跳转指令才能实现，体现出高级语言对编程的简化作用。

# 2 程序优化

## 2.1 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）深刻理解CPU执行指令的过程，不同特点的编程技巧和指令序列组合对程序长度及执行效率的影响，掌握代码优化的基本方法。

## 2.2 实验内容

**任务2.1**：在任务1.3程序基础上添加“5. 按照利润率由高到低输出商品信息”的功能。添加计时函数并循环运行功能4、5，观察运行时间，并对程序进行适当优化

## 2.3 任务2.1实验过程

### 2.1.1 实验方法说明

（1）在VS2019下对任务1.3程序添加按利润率排序的功能，并将排序结果进行输出。具体操作为：

① 建立一个查找表，保存各个商品信息在内存中的地址，在程序开始时初始化。

② 在菜单选项处增加选项5，并指向相应的程序段。

③ 在选项5对应程序段中，利用冒泡排序法得到正确的顺序，其中对于数据的交换在查找表中交换地址即可。

④ 排序完毕后进行输出，得到按利润率由高到低顺序的排列。

（2）将给定的计时函数加入到程序中，并在合适位置进行调用。

（3）改变程序结构，循环运行功能4、5到一定次数，同时调用计时函数，得到计时结果。具体操作为：

① 在功能4程序段开始处初始化寄存器edi，开始计时。

② 功能4结束后不回主菜单，而是直接跳转至功能5

③ 功能5结束前将edi的值加一，并与目标循环次数比较，达到目标时结束计时并回到主菜单；否则跳转至功能4计时之后的语句。

（4）对原有程序进行适当优化，以期减少运行时间。

### 2.1.2 实验记录与分析

为增大记录到的时间间隔，功能4，5循环次数设为10 000 000次。

1. 编写完毕的功能5运行截图如图2.1所示，其中各数据项定义如图2.2所示，TempValue项被略去。

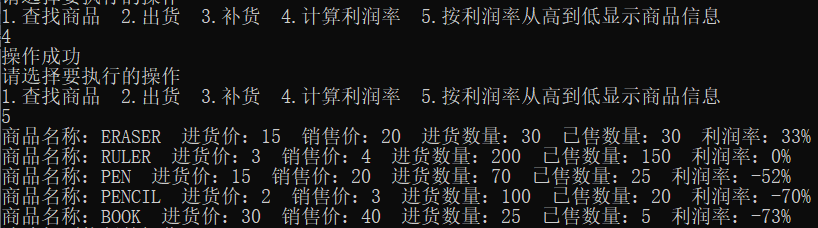


图 2.1 功能5的运行结果

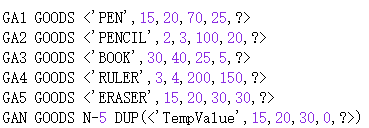


图 2.2 待排序数据的定义

2. 改变程序结构，循环运行功能4，5的时间大约为10000ms，如图2.3所示

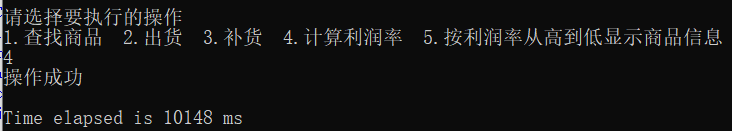


图 2.3优化前的执行时间

3. 修改排序方法，将原本循环N^2的排序方法改为循环N(N-1)次的标准冒泡排序法，运行时间约为2000ms，如图2.4所示

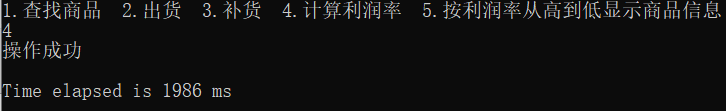


图 2.4 第一次优化后的运行结果

4. 因未能发现更好优化方案，故尝试添加无效语句观察时间变化。在功能5的内循环中，将一条mov指令复制两次，观察到运行时间明显延长，从原本的约2000ms变为约2100ms。

由此可见，双循环的内循环中的少数多余语句可能对程序运行时间造成很大影响。

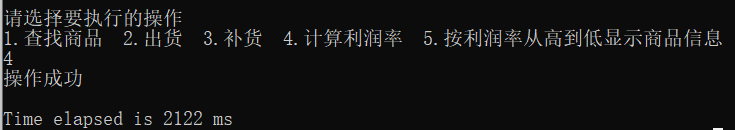


图 2.5 “负优化”后的运行结果

5. 在不同环境下运行此程序，观察循环过程时间。

在非debug环境下运行，观察到程序运行时间变为约2050ms，如图2.6所示

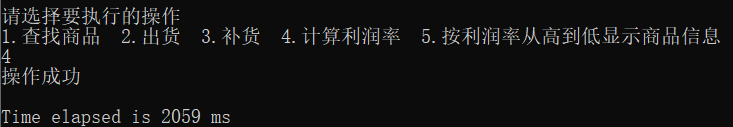


图 2.6 非debug环境下的运行结果

在Intel core i5 9400F环境下运行，程序运行时间约为1100ms，如图2.7所示

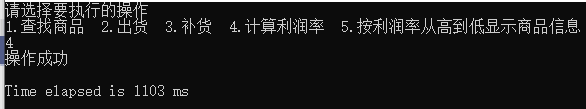


图 2.6 在其他计算机环境下的运行结果

## 2.4 小结

### 2.4.1 主要收获

本次实验的主要内容是程序优化，任务要求中提到不只考虑纯算法方面的优化，但遗憾的是我并没有成功在语句方面实现优化。但是通过“负优化”也让我了解到对语句的精简，尤其是在多重循环中的语句精简的重要性，即多余的语句在大量的循环中可能会严重拖慢程序运行的时间。

另外，在实验中，我也明显地感觉到寄存器有些不够用。在后续程序设计中，我也将想办法解决这一问题，以免在程序运行中出现寄存器冲突导致的难以发现的错误。

程序的运行时间受环境影响很大，且即使在同一环境下，重复运行的时间也有所不同。对于本次的汇编实验，在调试与非调试下运行时间相差并不明显，这可能是汇编程序的优势。

在本次实验中，对商品信息的排序并没有采用直接在地址中交换商品信息的方法，而是建立查找表，在查找表中交换对应地址的方式达成目的，这样虽然占用更多存储空间，但是提高了运行速度，是用空间换时间的方法。

### 2.4.2 主要看法

对于汇编程序，其运行效率较高，而代码优化则是进一步发挥汇编程序优化的重要方式。在实际操作中，常常只需要调换部分语句的顺序，便可达到精简语句和提高效率的作用。同时，由于循环会将语句执行多次，因此尽量减少循环的语句数量也是优化代码的重要手段。

对于排序功能的实现，有多种排序算法可选。但是对汇编程序而言，直接插入排序等方法较为复杂。因此根据不同语言特性和实际问题选择合适的算法能够大大提高程序编写的效率。

# 3 模块化程序设计

## 3.1 实验目的与要求

（1）掌握子程序设计的方法与技巧，熟悉子程序的参数传递方法和调用原理；

（2）掌握宏指令、模块化程序的设计方法;

（3）掌握汇编语言程序与C语言程序混合编程的方法；

（4）掌握较大规模程序的开发与调试方法；理解模块之间的信息传递与组装的基本方法；

（5）完成指定功能的程序设计与调试。

## 3.2 实验内容

**任务3.1：**

（1）改变程序结构，将程序的功能写为函数的形式。

（2）将字符串比较模块写成一个函数。、

（3）将程序写为多个源程序的形式。

（4）定义带参数的宏指令，编写一个带有参数的函数。

**任务3.2**

（1）将用户登录与菜单显示部分用C函数实现。

（2）用C语言实现“6. 添加新商品”功能。

## 3.3 任务3.1实验过程

### 3.3.1 设计思想及存储空间分配

（1）字符串比较函数，采用逐字节比较的方式，当各字节都相同时，返回0；否则当读取到第一个不同的单词时，比较两字符串的大小关系，大于返回1，小于返回-1。

（2）将程序分为两个源程序main.asm和func.asm，其中main.asm为main函数部分，func.asm为函数部分，包含定义的main函数以外所有函数的函数体。

### 3.3.2 流程图



图 3.1 函数间的调用关系

### 3.3.3 源程序、

…

MAIN\_MENU:

invoke printf,offset STR6

invoke printf,offset STR7 ;账号密码输入与菜单界面

invoke scanf,offset numtype,offset choice ;numtype应与choice相匹配（此处为%hd），否则会出现意想不到的错误

cmp choice,1

jz C1

cmp choice,2

jz C2

cmp choice,3

jz C3

cmp choice,4

jz C4

cmp choice,5

jz C5

invoke printf,offset STR15

jmp MAIN\_MENU

C1:

call CHOICE1

jmp MAIN\_MENU

C2:

call CHOICE2

jmp MAIN\_MENU

C3:

call CHOICE3

jmp MAIN\_MENU

C4:

call CHOICE4

jmp MAIN\_MENU

C5:

call CHOICE5

jmp MAIN\_MENU

INPUT\_ERROR1:

invoke printf,offset STR3

jmp PROGRAM\_EXIT

INPUT\_ERROR2:

invoke printf,offset STR4

jmp PROGRAM\_EXIT

PROGRAM\_EXIT:

invoke ExitProcess,0

…

### 3.3.4 实验步骤

（1）改变程序结构，将代码拆分到两个源文件中。将菜单中各个选项对应代码段改写为子程序，进行编译运行和调试。

（2）编写strcmp函数，实现比较两个字符串的功能。

（3）将一组常用指令改写为带参数的宏定义形式，并在程序中进行替换。

（4）在上述任务完成的基础上进行探究任务。

### 3.3.5 实验记录与分析

（1）子程序和主程序之间，主程序通过寄存器、全局变量或传递形参来向子程序传递信息；子程序通过保存在寄存器中的值向主程序中传递信息。

（2）程序开始运行时，堆栈中存储内容如图3.2所示（0x008FFBC4-0x008FFC08）

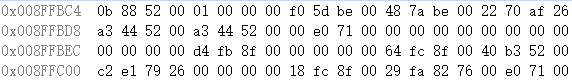


图 3.2 程序开始时堆栈的内容

（3）在执行call语句时，call语句后的语句的地址进栈，同时esp地址-4，如图3.3，3.4所示。

执行ret语句时，栈顶地址出战，出栈数据存入EIP中，即pop EIP，同时语句跳转至EIP对应语句，即之前地址进栈的语句。如图3.5所示



图 3.3 执行call语句后的堆栈栈顶

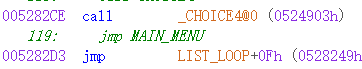


图 3.4 栈顶对应地址的语句



图 3.5 执行ret语句后EIP的值变为栈顶元素值

（4）若在执行ret前将栈顶元素修改，则执行ret语句时，程序仍会跳转至栈顶元素地址处，但如果栈顶元素的值不正确时会出现问题。例如，将栈顶元素值+5，则会跳转至call语句的后两条语句处，如图3.6所示、

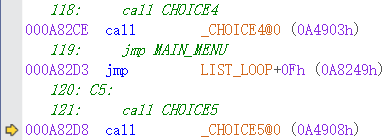


图 3.6 栈顶元素值+5后的跳转结果

（5）invoke伪指令包含将传入的参数进栈以及call语句，如图3.7所示。

此外，函数体的开头会增加语句push ebp和语句mov ebp,esp，而ret语句前增加leave语句，如图3.8所示。

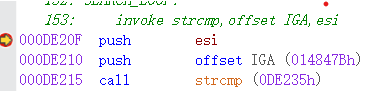


图 3.7 invoke语句对应的反汇编语句、



图 3.8 ret语句反汇编结果

（6）子程序中局部变量的存储空间在堆栈中，可利用lea指令获取局部变量的地址。

（7）对一个NEAR类型的子程序强制使用FAR调用，会产生报错，如图3.9所示。

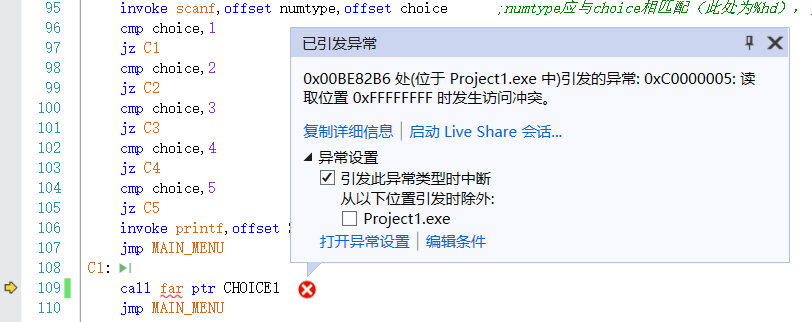


图 3.9 VS2019报错

（8）宏指令的调用，是直接将调用处语句替换为宏体内的语句，宏参数替换为传入的参数，如图3.10所示。

而子程序是程序跳转至子程序处运行。

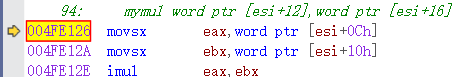


图 3.10 一条宏指令的反汇编语句

## 3.4 任务3.2实验过程

### 3.4.1 实验方法说明

（1）新建源文件cfunc.c保存任务3.2中新增的C函数。

（2）用C语言编写原main函数中的用户登录部分和菜单打印功能，并在原main函数中调用这两个模块。main函数其余部分保持不变。

（3）用C语言编写新增加功能的函数，并在原主菜单处新增对应的调用。

### 3.4.2 试验记录与分析

（1） 在C语言模块中调用汇编语言中定义的字符串时，声明语句应将其写成字符数组的形式，否则调用时会报错。如图3.11所示。



图 3.11 声明错误时的报错情况

（2）汇编程序中访问C中定义的变量时，未产生相应汇编指令（可能在预处理过程中完成），而是在调用时直接使用其地址。C程序中访问汇编程序中的变量时也是如此。

（3）由于在C程序中并无调用结构体的定义，因此应现在C程序中定义待引用结构体，随后用extern语句引用相应结构数组，并在”[]”中写明结构体个数。

（4）地址类型转换，即将该地址按照转换后的字节长度接受赋值。如在语句char a[10]; \*(int\*)a = 123;执行时，会用a[0]~a[3]这四个字节的空间存储数值123.

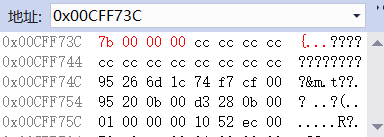


图 3.12 a[10]的前四个字节的值发生变化

（5）C语言函数调用语句与汇编语言函数调用相同，均为call语句。进入函数前，会将除eax，ecx，edx之外的寄存器入栈，return语句执行后恢复。执行return语句时，会将return后的值存入eax中，若无返回值则无上述过程。参数的传递同样是通过栈传递。

## 3.5 小结

### 3.5.1 主要收获

本次实验的主要内容为汇编语言的模块化编程方法，包括多源文件、多子程序以及汇编语言和C语言的混合编程。当程序达到一定规模时，将所有代码写在同一个源文件、同一个子程序中将会使程序难以阅读。同时，当相同的步骤需要反复进行时，会极大增大代码长度。因此，将程序分解为多个子程序、多个源文件时很有必要的。

在C语言编程中，我们很少关心调用函数时参数的传递和语句的跳转，但在汇编语言中，这些功能都需要我们自己去完成。为了防止函数调用时改变寄存器中不应被改变的值，我们需要对其进行保护，即利用堆栈储存或尽量避免使用这些寄存器。另外，由于函数的跳转等都涉及到堆栈的使用，我们也应该尽量避免对寄存器esp、ebp的使用。

对于部分结构，如嵌套的循环、分支等，用汇编语言编写会较为复杂，因此我们可以用C语言来代替这些部分。而汇编语言由于更加贴近机器语言，在设计得当的情况下执行效率又会优于C语言。因此在合适的情况下对二者进行结合有助于提高程序的编写效率和运行效率。

### 3.5.2 主要看法

对于大型程序的开发，模块化编程是很重要的方法。模块化不仅可以提高代码的可阅读性、可维护性，也方便了程序调试，使程序逻辑更加清晰。这个“模块”，可以是多文件、可以是多子程序、甚至可以是多种语言的混合。合理地使用多语言混编地方法，可以提高程序开发效率和运行效率。但是要注意不同模块间信息的传递，以及调用其他模块时寄存器和堆栈的保护，以免出现逻辑上的错误。

# 4 中断与反跟踪

## 4.1 实验目的与要求

（1）通过观察与验证，理解中断矢量表的概念；

（2）熟悉I/O访问，BIOS功能调用方法；

（3）掌握实方式下中断处理程序的编制与调试方法；

（4）进一步熟悉内存的一些基本操纵技术；

（5）熟悉跟踪与反跟踪的技术，熟悉动态与静态反汇编工具，深刻理解汇编语言的特殊能力；

（6）完成指定功能的程序设计与调试，提升对计算机系统的理解与分析能力。

## 4.2 实验内容

**任务4.1 ：**

（1）利用中断，在DOSBOX下实现在指定位置显示时分秒的功能。

（2）在上述程序的基础上实现程序的驻留功能。

（3）在上述程序基础上实现避免程序重复安装的功能。

**任务4.2：**

在任务3.1的基础上将老板的密码加密后存储在数据段中。

**任务4.3**

尝试对其他同学的程序进行跟踪和数据解密

## 4.3 任务4.1实验过程

### 4.3.1 设计思想

（1）在主程序中设置接管系统08H中断程序，保存原08H中断程序地址，设置程序驻留。

（2） 在新的08H中断程序中，利用IO指令读取CMOS芯片中的时间信息并输出到屏幕上。

### 4.3.2 流程图



图 4.1 任务4.1的流程图

### 4.3.3 源程序

…

str0    db 'main process finished',0AH,0

 str1    db 'fail:program exists',0AH,0

 installed dw 1

…

begin:

        push cs

        pop ds

        mov ax,3508H                 ;获取原08H的中断矢量

        int 21H

        mov di,bx

        sub di,2

        cmp word ptr [di],1

        jz  fail

        mov old\_int,bx               ;保存中断矢量

        mov old\_int+2,es

        mov dx,offset NEW08H

        mov ax,2508H                 ;设置新的08H中断矢量

        int 21H

        jmp done

fail:

        mov si,offset str1

        jmp print

done:

        mov si,offset STR0

print:

        mov dl,[si]

        cmp dl,0

        jz  exit

        mov ah,02H

        int 21h

        inc si

        jmp print

exit:

        mov ah,31h

        mov dx,offset begin+15

        mov cl,4

        shr dh,cl

        add dh,10H

        mov al,0

        int 21h

### 4.3.4 实验步骤

1. 参考资料，编写接管后的中断程序和时间获取程序。

2. 编写主程序，实现判断是否重复安装和实现内存驻留的功能。

3. 在DOSBOX下编译和运行，并完成观察中断矢量表等任务。

### 4.3.5 实验记录与分析

1. 编写和调试完毕后，重复运行程序会报错，

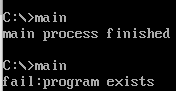


图 4.2 重复安装时的报错信息

2. 在TD下观察到的中断矢量表如图4.3所示。当第二次运行时，可观察到原中断是量表被接管处被修改。

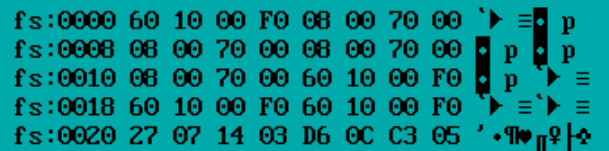


图 4.3 接管前的中断矢量表

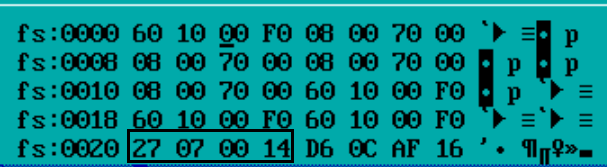


图 4.4 接管后，08中断矢量被修改

3. 在int语句处使用快捷键Alt+F7即可进入中断管理程序。

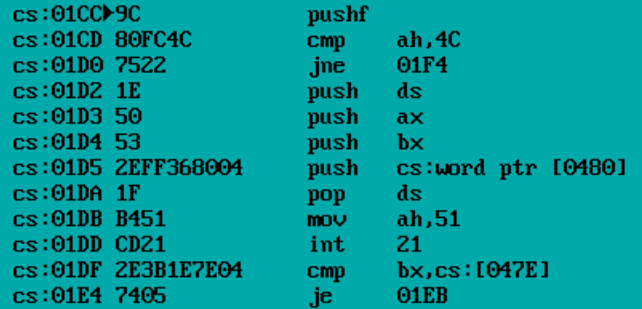


图 4.5 中断管理程序内容

## 4.4 任务4.2实验过程

### 4.4.1 实验方法说明

1. 利用某种算法，实现老板密码等信息的加密处理。

2. 采用计时、动态修改执行代码等方式实现反跟踪。

### 4.4.2 实验记录与分析

1. 在数据段定义数据的过程中，使用“异或”对进货价、登录密码进行加密。在使用密码时，先将输入的密码按照相同方式变换后与数据段中内容进行比较；在使用进货价时，先对其及逆行相同的变换后代入利润率计算中。

2. 在部分函数中插入计时程序，防止单步执行跟踪。

3. 动态修改函数语句，将数据段中保存的部分语句的机器码复制到代码段中。

## 4.5 任务4.3实验过程、

### 4.5.1 实验方法说明

利用多种方式尝试破解其他同学的加密程序。

### 4.5.2 实验记录与分析

1. 利用W32asm反编译程序，并用串式数据参考功能读取数据内容，找到用户名和密码的位置。

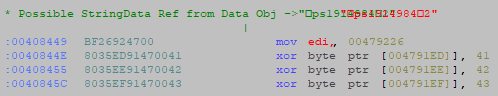


图 4.6 代码中密码操作所在位置

2. 根据反汇编结果中的三个异或操作，对应数据段中相应位置可以计算出正确的密码。

从数据段中得到三次异或操作结果为14H，70H，73H，再次进行异或操作得到解密后的字符为’U’,’2’,’0’。

3. 利用前一步获得的密码登录系统，直接调用系统中的信息输出函数即可得到各个商品的进货价等信息，如图4.7所示。

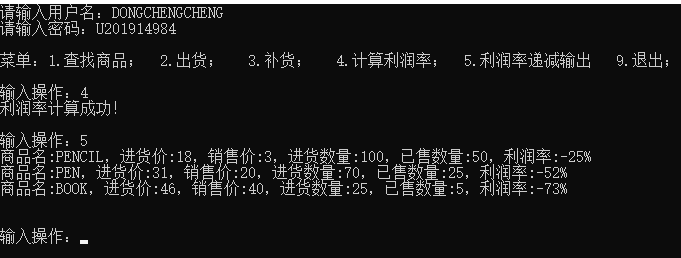


图 4.7 获取到的进货价等信息

4. 反编译得到的程序，除了自己编写的源代码（经编译器处理）外，还包含大量库函数等语句，在一些位置还有大量的int 03语句，如图4.8所示。

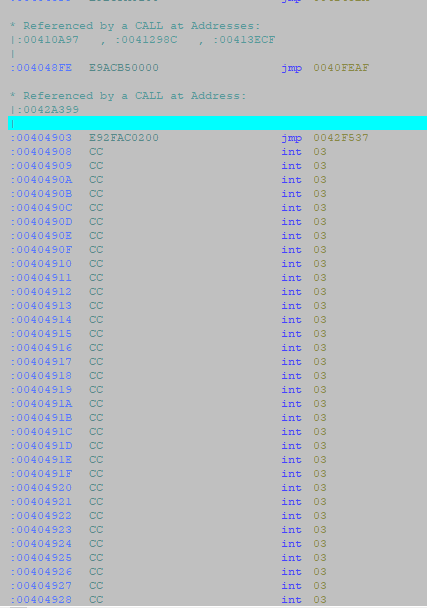


图 4.8 从00404903开始的大量int 03代码

## 4.6 小结

### 4.6.1 主要收获

本次实验的内容为中断与反跟踪，所涉及到的内容相比C语言更加接近操作系统甚至计算机硬件。在中断程序设计中，我们利用对中断程序和中断矢量表的修改，实现了对系统底层功能的修改和操纵；在系统时间的获取中，我们利用IO指令，实现从CMOS中读取当前时间，并保存到寄存器中的功能。这些任务的完成，使我了解到，程序的“中断”并不是简单的、字面意义上的中断，而是一种暂时停止程序的运行，并根据相应设置执行其他更加高级指令的方法。

加密与解密也是本次实验的重点。本实验中，除了利用异或等运算对明文信息进行加密外，还利用了动态修改代码的方式，即在程序运行过程中，将一部分数据段内容复制到代码段中（利用VirtualProtect函数解除保护），并将被复制内容的开头设置为下一条指令的机器码，从而增加代码迷惑性。

解密过程使用了W32asm软件。在W32asm中，软件可以分析出程序中定义的变量，并跳转到调用变量的位置，为密码的解密提供了方便。另外，软件也支持数据段等的分析，也方便了对加密信息的对照和破解。

### 4.6.2 主要看法

中断是操作系统相对底层的功能。利用中断可以实现多种诸如从键盘中读取字符等底层命令。合理利用中断也能实现许多难以直接实现、以及在高级语言中通过标准库才能实现的指令。通过修改中断程序和中断矢量表，可以实现类似子程序的功能，但是实现的功能比欧通子程序更加接近操作系统。

程序的加密和解密，一直是软件设计和使用中绕不开的话题。通过合理的方法进行加密，可以大大增加不怀好意者对程序的攻击和破坏的难度，从而提高系统的安全性、阻止信息的泄露，也可以保护开发者的劳动成果不被窃取。而解密则是为了从现有的软件中破解出想得到的信息，绕开商业软件许可等目的。任何软件都会被破解，考验的则是加密者、解密者的功力，而优秀的反编译软件则是解密者的强大工具。利用W32asm或其他更高级工具，可以将编译得到的可执行文件反编译为汇编指令，从而方便解密者进行跟踪和破解。IDA等工具甚至可以将程序解析为多个模块，并展示其中的相互关系，降低了分析的难度。

中断和反跟踪，是汇编语言相对高级语言的优势。通过调用系统级指令，可以更为高效地实现某些功能，而不必调用标准库函数，提高程序运行效率。由于中断矢量表的修改可以影响到其他程序，因此合理地利用中断也可以实现不同程序间的交互，以及判断驻留后的程序是否重复运行。对于反跟踪，汇编语言可以通过复制数据段内容到代码段中，实现动态修改程序代码，从而扰乱跟踪者的注意力。这类功能均为高级语言不具备或很难实现的功能。对于跟踪破解，由于反编译软件直接获得的是汇编指令，因此利用汇编语言进行分析效率会更高，也会更准确。

总之，汇编语言与高级语言各有特点，高级语言优势在于容易理解和容易编写，而汇编语言的优势在于贴近底层、执行效率高，便于加密与跟踪。

# 5. 16/32/64位编程比较

## 5.1 实验目的与要求

（1）了解16/32/64位环境下程序设计的不同特点及配套的开发工具；

（2）通过完成指定的程序设计，观察并理解汇编语言在不同环境下的基本特点。

## 5.2 实验内容

**任务5.1：**以任务3.1的程序为基础，编写基于窗口的win32程序。

**任务5.2：**在VS2019下调试一个64位程序并观察其与32位程序的不同之处。

**任务5.3：**查阅相关资料，了解armv8系列汇编语言编程资料，并与80X86系列进行对比。

## 5.3 任务5.1实验过程

### 5.4.1 实验方法说明

1. 在给定示例基础上进行修改，实现显示所要求窗口的功能。

2. 由于窗口中只能输出字符串，因此还需编写一个程序实现从数字到字符串的转换。

3. 修改about窗口中输出内容，使之能够显示姓名与学号。

4. 修改Display函数，使之能够按照格式正确显示各个商品的信息。

### 5.4.2 实验记录与分析

1. 在VS2019中，在可视化界面下对menu.rc进行修改，会无法成功编译运行，故采用文本编辑器对其进行修改，同时给与相应的编号。

2. win32窗口无法直接输出数字，需要将其中的数字转换成ASCII字符串再调用相应函数输出。对于负数，先在转换后的串前加负号，再将其转成正数，最后转换成ASCII码字符串。

3. 转换完毕后，根据示例中所给格式，对于每个商品修改输出时的行信息，得到最终效果如图5.1所示。

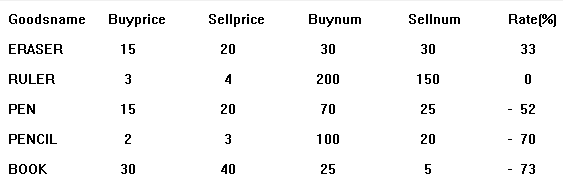


图 5.1 输出各个商品信息结果

## 5.4 任务5.2实验过程

### 5.4.1 实验方法说明

1. 在VS2019下创建项目，设置平台为X64，设置程序入口点。

2. 运行程序，观察并比较其与X86平台下的差异。

### 5.4.2 ·实验记录与分析

1. X64程序开头无处理器选择伪指令和存储模型说明伪指令。

2. 在声明函数原型的时候可以不声明参数表。因此可以不用proto进行声明。

3. 程序的入口默认为mainCRTStartup，也可根据需要手动设定。

4. 编译器不再支持invoke伪指令，只能使用call指令，并由调用者传递参数。

5. 增加了R8~R15共8个64位通用寄存器，原有的8个寄存器被扩展到64位，即共有16个64位通用寄存器。

## 5.5 任务5.3实验过程

### 5.5.1 实验方法说明

阅读相关文档，比较armv8指令与x86指令的差异

### 5.5.2 实验记录与分析

1. ARMV8提供了X0~X30共31个64位通用寄存器，其低32位为W0~W30，是32位寄存器。执行写Wn寄存器时，会将Xn的高32位清零。

2. 在ARMV8中，用AREA 定义一个段，并说明所定义段的相关属性；CODE 用以指明为代码段； ENTRY 标识程序的入口点 ；END 为程序结束。

3. 在指令中，立即数前加#。

4. 指令后接3个操作数，第一个操作数保存后两个操作数执行后的结果。

5. 指令的机器码均为32位，对应CPU的位数，而X86中指令机器码长度不固定。

6. 子程序通过寄存器进行参数传递，返回值存储在R0中。

7. ARMv8汇编与C语言的混合编程需遵守ATPCS规则，在汇编语言中需用IMPORT伪指令声明C语言函数；在C语言中插入ARMv8汇编代码段与插入X86汇编代码段方法相同。

## 5.6 小结

### 5.6.1 主要收获

本次实验的主要内容为Win32窗口程序编写、32位/64位编程比较和ARM/X86平台汇编差异的比较。

在Win32窗口程序编写中，较难的部分是Win32窗口的编辑和调用，由于实验提供了示例代码，只需在其基础上进行修改即可。而另一方面，Win32窗口的信息输出并不支持数字的直接输出，需要将其转化为ASCII字符串才能正确输出，这一部分是这次实验的另一个难点。但汇编语言中的除法结果既有商，也有余数，这对结果的处理产生了较大方便。

X64程序相对X86程序变化较小，主要在于代码开头部分的简化、寄存器的增加和扩展以及各种指令的64位升级，使从32位到64位的过渡更加容易，也保证了兼容性。

ARMv8汇编语言和X86/X64差异较大。除运算指令大致相同外，大量指令的用法和功能都有所变化。在寄存器使用方面，ARMv8提供了更多的寄存器。但是与X86汇编不同，对64位寄存器的低32位进行操作时，ARMv8会将相应的高32位清零。另外，ARMv8和X86的显著差别在于，X86语句对应机器指令长度不固定，而ARMv8汇编指令的机器指令长度均为定值，这也是RISC和CISC两种模式的重要区别。

### 5.6.2 主要看法

在VS2019中，对窗口菜单的编写提供了可视化页面，有助于简化菜单的编辑。但是在实际使用中，效果并不尽如人意，这可能是我对相关原理不够了解的问题。这也告诉我，在使用一个工具时，一定要尽量理解使用功能的作用。

X86到X64的变化并不是彻底的改变，而是在原有基础上进行的增补和优化，这样做降低了开发者的学习成本、提高了CPU对软件的兼容性。但是，对兼容性的保证也导致许多开发者保持32位编程而不转向64位，一定程度上降低了软件运行效率。同时，CPU对过老指令集的兼容也会影响CPU的能效。相比之下，ARM平台一直在放弃对较老指令集的兼容，从而换来更高的能效比，大有赶超X86/64平台的趋势，这也是CPU升级发展需要做出的取舍。

ARMv8指令与X86/X64指令相差较大，因此在X86/X64环境下可以运行的程序在ARMv8下基本无法正常运行。这也告诉我们，软件的开发一定要选择正确的平台和指令集，错误的选择可能会使前期努力作废。指令集的不通用，为多平台开发维护造成了较大困难，希望未来能有更多更好的解决方案。

ARMv8指令属于RISC，即精简指令集，特点主要为指令长度均为定长。这样做以一定程度上降低语句执行的效率为代价，带来了容易编写、低能耗等优势，也是未来发展的方向。

# 参考文献

[1]许向阳. x86汇编语言程序设计. 武汉：华中科技大学出版社，2020（第19章，第3章、第4章、第5章）

[2]许向阳. 80X86汇编语言程序设计上机指南. 武汉：华中科技大学出版社， 2007

[3]王元珍，曹忠升，韩宗芬. 80X86 汇编语言程序设计. 武汉：华中科技大学出版社，2005

[4]汇编语言课程组. 《汇编语言程序设计实践》任务书与指南，2021