汇 编 语 言 程 序 设 计 实 验 报 告

華中科技大學

课 程 实 验 报 告

课程名称： 汇编语言程序设计实验

实验名称： 实验五　WIN32 编程

实验时间： 2019-5-8，14：00-17：30　**实验地点：** 南一楼 804 室 30 号实验台

指导教师： 曹忠升

专业班级： 计算机科学与技术 ACM1701 班

学　　号： U201714780　**姓　　名：** 刘晨彦

同组学生： 无　**报告日期：** 2019 年 5 月 8 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：2019.05.08

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：

汇 编 语 言 程 序 设 计 实 验 报 告

汇 编 语 言 程 序 设 计 实 验 报 告

目录

汇 编 语 言 程 序 设 计 实 验 报 告

1 实验目的与要求

(1) 熟悉 WIN32 程序的设计和调试方法。

(2) 熟悉宏汇编语言中 INVOKE、结构变量、简化段定义等功能。

(3) 进一步理解机器语言、汇编语言、高级语言之间以及实方式、保护方式之间的一些关系。

2 实验内容

编写一个基于窗口的 WIN32 程序，实现网店商品信息管理程序的推荐度计算及商品信息显示的功能（借鉴实验三的一些做法），具体要求如下描述。

功能一：编写一个基于窗口的 WIN32 程序的菜单框架，具有以下的下拉菜单项：

File　Action　　Help

Exit　Recommendation　About

List

点菜单 File 下的 Exit 选项时结束程序；点菜单 Help 下的选项 About，弹出一个消息框，显示本人信息，类似图 5.1 所示。点菜单 Action 下的选项 Recommendation、List 将分别实现计算推荐度或显示 SHOP 所有商品信息的功能（详见功能二的描述）。

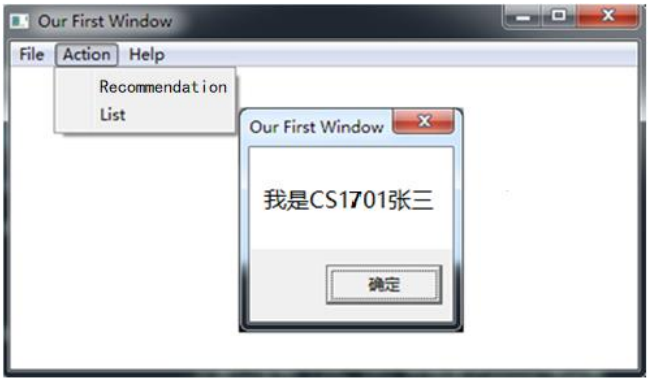

图 5.1 菜单示例

功能二：要求采用结构变量存放商品的相关信息。商品数至少定义 5 种。

（1）点菜单项 Recommendation 时，按照实验三的方法计算所有商品的推荐度。用 TD32 观察计算结果。

（2） 点菜单项 List 时，要求能在窗口中列出 SHOP 的所有商品的信息。具体显示格式自行定义，可以参照图 5.2 的样式。

图 5.2 商品信息显示示意图

3 实验过程

3.1 设计思想及存储单元分配

基于该窗口的应用程序使用了 WIN32 的标准框架。

标准框架可分为：主程序、窗口主程序、窗口消息、处理程序和用户处理程序。在该框架中，操作系统首先执行主程序，待主程序获得与本程序相关的基本信息后，调用窗口主程序创建指定窗口，由该窗口获取操作信息并转入窗口消息处理程序。窗口消息处理程序判断收到信息种类，调用相关功能。

操作栏的实现使用了 RC 文件，在 WinMain 中，需要对菜单进行装载。在窗口过程中，对相应菜单功能进行设计，即可实现相关菜单功能。在实现菜单功能时，需要对子程序进行调用。在本次实验中，调用的子程序有：计算推荐度、显示信息、显示个人信息和退出。

计算推荐度的子程序复用了之前实验中的代码。显示信息的子程序调用了 TEXTOUT 函数，确定每行文本的起始位置后，按照一定的间距将文本输出。

3.2 流程图

主程序流程图如图 5.3 所示。

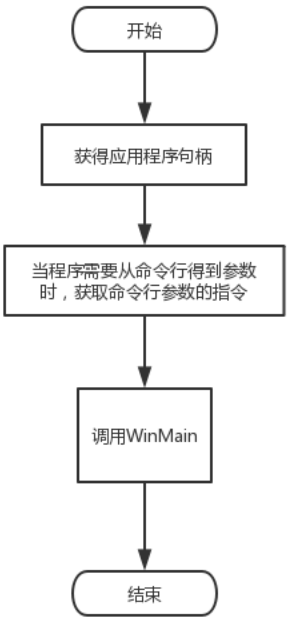

图 5.3 主程序流程图

WinMain 窗口主程序函数流程图如图 5.4 所示。

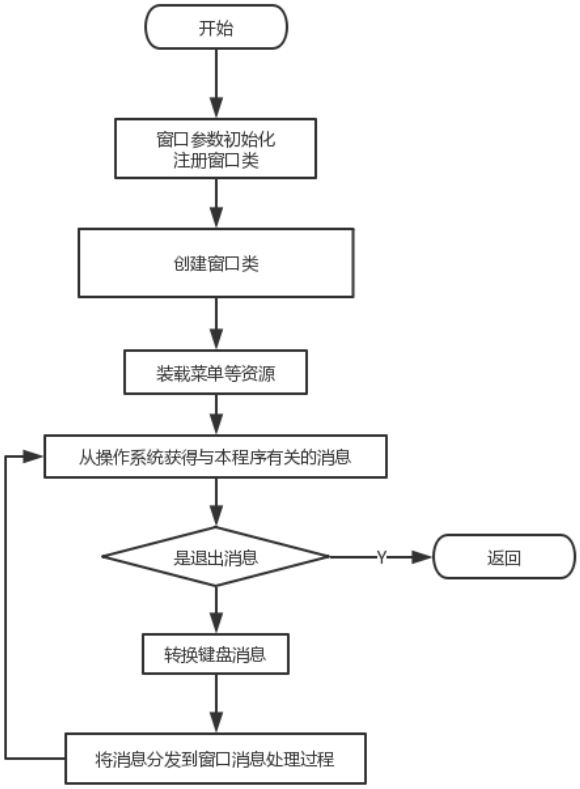

图 5.4 WinMain 窗口主程序流程图

Wndproc 函数流程图如图 5.5 所示。

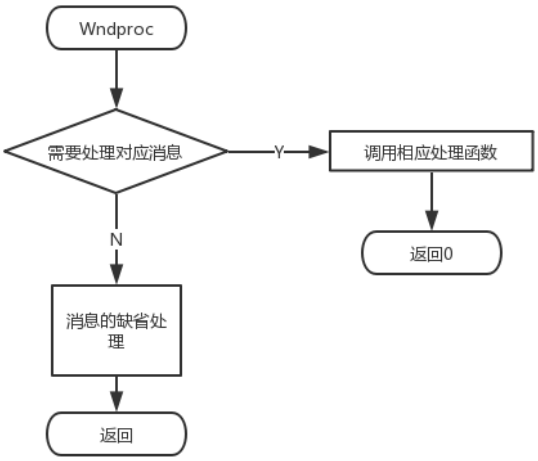

图 5.5　Wndproc 函数流程图

输出商品信息子程序如图 5.6 所示。

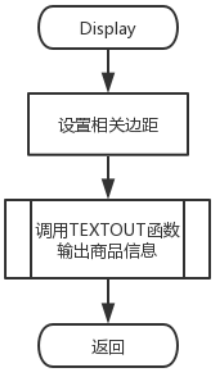

图 5.6 输出商品信息子程序流程图

3.3 源程序

3.3.1　菜单源代码

```
#define IDM_FILE_EXIT  1000
#define IDM_FILE_RECOMMENDATION  1100
#define IDM_HELP_ABOUT 1900
#define IDM_FILE_LIST 1200

600 MENUEX MOVEABLE IMPURE LOADONCALL DISCARDABLE
BEGIN
 POPUP "&File"
      BEGIN
           MENUITEM "E&xit",IDM_FILE_EXIT
 END
```

```
POPUP "&Action"
BEGIN
    MENUITEM "&Recommendation",IDM_FILE_RECOMMENDATION
    MENUITEM "&List",IDM_FILE_LIST
END
POPUP "&Help"
BEGIN
    MENUITEM "&About",IDM_HELP_ABOUT
END
END
```

3.3.2 主程序源代码

```
.386
.model flat, stdcall
option casemap :none   ; case sensitive

include d:\masm32\include\windows.inc
include d:\masm32\include\user32.inc
include d:\masm32\include\kernel32.inc
include d:\masm32\include\gdi32.inc

includelib d:\masm32\lib\user32.lib
includelib d:\masm32\lib\kernel32.lib
includelib d:\masm32\lib\gdi32.lib

szText MACRO Name, Text:VARARG
      LOCAL lbl
      jmp lbl
      Name db Text,0
      lbl:
ENDM

m2m MACRO M1, M2
      push M2
      pop  M1
ENDM

return MACRO arg
      MOV EAX, arg
      ret
ENDM
```

```
;函数声明
WinMain PROTO :DWORD,:DWORD,:DWORD,:DWORD
WndProc PROTO :DWORD,:DWORD,:DWORD,:DWORD
TopXY PROTO   :DWORD,:DWORD
Paint_Proc PROTO :DWORD, hDC:DWORD
profit_rate PROTO :DWORD,:WORD
msg_display PROTO :DWORD,:DWORD,:WORD
Display     PROTO   hWnd:DWORD

_good STRUCT
      goodname    DB    10    DUP(32)
      discount    DB    0
      in_price    DW    0
      out_price   DW    0
      in_num      DW    0
      out_num     DW    0
      pro_rate    DW    0
_good ENDS

.data
szDisplayName db "ONLINE SHOP",0
CommandLine   dd 0
hWnd          dd 0
hInstance     dd 0

N     EQU   6
GOODS1      _good <'BOOK',9,12,30,25,5,0>
            _good <'PEN',10,35,56,70,25,0 >
            _good <'BAG',8,15,20,30,2,0>
          _good <'PENCIL',9,15,56,10,1,0>
          _good <'NOTE',10,15,56,10,1,0>
          _good <'PAPER',8,15,56,10,1,0>

BUF   DB 12 DUP(?)
str1  DB    'List',6 DUP(0)
str2   DB     'Item               Discount            Pur_price           Sell_price
Pur_Number     Sell_Number    Recommendation       '
str3  DB    120 DUP('-')
.code

start:
        INVOKE GetModuleHandle, NULL  ;获得并保存本程序句柄
```

```
        MOV hInstance, EAX

        INVOKE GetCommandLine
        MOV CommandLine, EAX

        INVOKE WinMain,hInstance,NULL,CommandLine,SW_SHOWDEFAULT  ;调用窗口主程序
        INVOKE ExitProcess,EAX  ; 退出本程序，返回 Windows

WinMain proc hInst     :DWORD,  ;winmian 形参定义
            hPrevInst :DWORD,
            CmdLine   :DWORD,
            CmdShow   :DWORD

        ;====================
        ; Put LOCALs on stack
        ;====================

        LOCAL wc   :WNDCLASSEX;创建主窗口时所需要的信息由该结构说明
        LOCAL msg  :MSG;消息结构变量用于存放获取的信息

        LOCAL Wwd  :DWORD
        LOCAL Wht  :DWORD
        LOCAL Wtx  :DWORD
        LOCAL Wty  :DWORD

        MOV wc.cbSize, sizeof WNDCLASSEX;wndclasse 结构类型的字节数
        MOV wc.style, CS_HREDRAW or CS_VREDRAW or CS_BYTEALIGNWINDOW;窗口风格
        MOV wc.lpfnWndProc,    OFFSET WndProc;本窗口过程的入口地址
        MOV wc.cbClsExtra,     NULL;不用自定义数据则不需要 os 预留空间，为 NULL
        MOV wc.cbWndExtra,     NULL;同上
        m2m wc.hInstance,      hInst   ;程序句柄送入 wc.hinstance
        MOV wc.hbrBackground,  COLOR_BTNFACE;窗口背景白色
        MOV wc.lpszMenuName,   NULL;窗口不带菜单
        MOV wc.lpszClassName,  OFFSET szClassName;窗口类名"Template_Class"
          INVOKE LoadIcon,hInst,500    ;装入系统默认图标
        MOV wc.hIcon,          EAX;保存图标句柄
          INVOKE LoadCursor,NULL,IDC_ARROW;装入系统默认光标
        MOV wc.hCursor,        EAX;保存光标句柄
        MOV wc.hIconSm,        0;窗口不带小图标

        INVOKE RegisterClassEx, ADDR wc;注册窗口类

        MOV Wwd, 750
```

```
            MOV Wht, 375

            INVOKE GetSystemMetrics,SM_CXSCREEN
            INVOKE TopXY,Wwd,EAX
            MOV Wtx, EAX

            INVOKE GetSystemMetrics,SM_CYSCREEN
            INVOKE TopXY,Wht,EAX
            MOV Wty, EAX

            szText szClassName,"Template_Class"

            INVOKE  CreateWindowEx,WS_EX_LEFT,  ADDR  szClassName,  ADDR  szDisplayName,
WS_OVERLAPPEDWINDOW, Wtx,Wty,Wwd,Wht, NULL,NULL, hInst,NULL
            MOV   hWnd,EAX

            INVOKE LoadMenu,hInst,600  ; menu ID
            INVOKE SetMenu,hWnd,EAX

            INVOKE ShowWindow,hWnd,SW_SHOWNORMAL
            INVOKE UpdateWindow,hWnd

        StartLoop:
          INVOKE GetMessage,ADDR msg,NULL,0,0
          cmp EAX, 0
          je ExitLoop
          INVOKE TranslateMessage, ADDR msg
          INVOKE DispatchMessage,  ADDR msg
          jmp StartLoop
        ExitLoop:

          return msg.wParam

    WinMain ENDP

    WndProc proc hWin: DWORD, uMsg: DWORD, wParam: DWORD, lParam: DWORD
            LOCAL hDC  :DWORD
            LOCAL Ps   :PAINTSTRUCT


        .if uMsg == WM_COMMAND
        ;======== menu commands ========
            .if wParam == 1000
```

```
            INVOKE SendMessage,hWin,WM_SYSCOMMAND,SC_CLOSE,NULL
        .elseif wParam == 1900
            szText TheMsg,"ACM1701Liu Chenyan"
            INVOKE MessageBox,hWin,ADDR TheMsg,ADDR szDisplayName,MB_OK
      .elseif wParam == 1100
            INVOKE profit_rate,ADDR GOODS1,N
            szText TheaMsg,"Calculation Finished"
              INVOKE MessageBox,hWin,ADDR TheaMsg,ADDR szDisplayName,MB_OK
      .elseif wParam == 1200
            INVOKE msg_display,hWnd,ADDR GOODS1,N
        .endif
    ;====== end menu commands ======

    .elseif uMsg == WM_PAINT
        INVOKE BeginPaint,hWin,ADDR Ps
          MOV hDC, EAX
          INVOKE Paint_Proc,hWin,hDC
        INVOKE EndPaint,hWin,ADDR Ps
        return 0

    .elseif uMsg == WM_CLOSE
        szText TheText,"Please Confirm Exit"
        INVOKE MessageBox,hWin,ADDR TheText,ADDR szDisplayName,MB_YESNO
          .if EAX == IDNO
            return 0
          .endif
    .elseif uMsg == WM_DESTROY
        INVOKE PostQuitMessage,NULL
        return 0
    .endif

    INVOKE DefWindowProc,hWin,uMsg,wParam,lParam

    ret

WndProc ENDP

TopXY proc wDim:DWORD, sDim:DWORD

    shr sDim, 1      ; divide screen dimension by 2
    shr wDim, 1      ; divide window dimension by 2
    MOV EAX, wDim    ; copy window dimension into EAX
    sub sDim, EAX    ; sub half win dimension from half screen dimension
```

```
    return sDim

TopXY ENDP

Paint_Proc proc hWin:DWORD, hDC:DWORD

    LOCAL btn_hi   :DWORD
    LOCAL btn_lo   :DWORD
    LOCAL Rct      :RECT

    INVOKE GetSysColor,COLOR_BTNHIGHLIGHT
    MOV btn_hi, EAX

    INVOKE GetSysColor,COLOR_BTNSHADOW
    MOV btn_lo, EAX

    return 0

Paint_Proc ENDP

profit_rate PROC  lpgoods1: DWORD,num:WORD
            LOCAL TIMES: WORD
            PUSHA
            MOV EBX,0
            MOV ESI,lpgoods1
            MOV AX, num
            MOV TIMES, AX
LOOP1:           MOV AL, [ESI + 10]        ;DISCOUNT IN AX
             MOV AH, 0
          MOV CX, [ESI + 13]      ;SALE PRICE IN CX
          MUL CX                        ;SALE * DISCOUNT IN AX
          MOV CX, 10
          XOR DX, DX
          IDIV CX                       ;ACTUAL SALE PRICE IN AX
          MOV BX, AX                    ;ACTUAL SALE PRICE IN BX
          MOV AX, [ESI + 11]      ;PURCHASE PRICE
          MOV CX, 128
          MUL CX                        ;PURCHASE PRICE * 128 IN AX
          MOV CX, BX                    ;ACTUAL SALE PRICE IN CX
          XOR DX, DX
          IDIV CX                       ;PURCHASE PRICE * 128 / ACTUAL SALE PRICE IN AX
          MOV BX, AX                    ;PURCHASE PRICE * 128 / ACTUAL SALE PRICE IN BX
```

```
            MOV AX, [ESI + 15]      ;NUM OF PURCHASE IN AX
            MOV CX, 2               ;2 IN CX
            MUL CX                      ;2 * NUM OF PURCHASE IN AX
            MOV DI, AX                  ;2 * NUM OF PURCHASE IN DI
            MOV AX, [ESI + 17]      ;NUM OF SALE
            MOV CX, 128
            MUL CX                      ;NUM OF SALE * 128 IN AX
            MOV CX, DI                  ;2 * NUM OF PURCHASE IN CX
            XOR DX,DX
            IDIV CX                     ;NUM OF SALE * 128 / 2 * NUM OF PURCHASE IN AX
            ADD BX, AX
            MOV WORD PTR [ESI + 19], BX
              DEC TIMES
              CMP TIMES, 0
              JE ED
              ADD ESI, 21
              JMP LOOP1
ED:          POPA
              return 0
profit_rate ENDP

msg_display proc  hwnd:DWORD,lp_msg_goods:DWORD,num_goods:WORD
      LOCAL hdc:DWORD
      INVOKE      GetDC,hwnd
      MOV   hdc,EAX
      nY    EQU   40
      nX    EQU   100
      xStart      EQU   20
      yStart      EQU   80

      INVOKE      TextOut,hdc,20,20,addr str1,4
      INVOKE      TextOut,hdc,20,40,addr str2,120
      INVOKE      TextOut,hdc,20,60,addr str3,180

      MOV   EBX,xStart
      MOV   EDI,yStart
      MOV   ESI,lp_msg_goods
      MOV   DX,16
m_lp1:
      INVOKE      TextOut,hdc,EBX,EDI,ESI,10

      ADD   EBX,70
      MOVZX   AX, BYTE PTR [ESI+10]
```

```
        CALL  F2T10
        INVOKE      TextOut,hdc,EBX,EDI,addr BUF,EAX

        ADD EBX,nX
        MOV   AX, [ESI+11]
        CALL  F2T10
        INVOKE      TextOut,hdc,EBX,EDI,addr BUF,EAX

        ADD   EBX,nX
        MOV   AX,[ESI+13]
        CALL  F2T10
        INVOKE      TextOut,hdc,EBX,EDI,addr BUF,EAX

        ADD   EBX,nX
        MOV   AX,[ESI+15]
        CALL  F2T10
        INVOKE      TextOut,hdc,EBX,EDI,addr BUF,EAX

        ADD   EBX,nX
        MOV   AX,[ESI+17]
        CALL  F2T10
        INVOKE      TextOut,hdc,EBX,EDI,addr BUF,EAX

        ADD   EBX,nX
        MOV   AX,[ESI+19]
        CALL  F2T10
        INVOKE      TextOut,hdc,EBX,EDI,addr BUF,EAX

        ADD   ESI,21
        MOV   EBX,xStart
        ADD   EDI,nY
        sub   num_goods,1
        cmp   num_goods,0
        jnz   m_lp1
        return 0
msg_display ENDP

F2T10 PROC  NEAR
              PUSH EBX
              PUSH ESI
              LEA   ESI,BUF
              CMP   DX,32
              JE    B
```

```
            MOVSX EAX,AX
B:          OR    EAX,EAX
            JNS   PLUS
            NEG   EAX
            MOV   BYTE PTR [ESI],'-'
            INC   ESI
PLUS: MOV   EBX,10
            CALL RADIX
            SUB   ESI,OFFSET  BUF
            MOV   EAX,ESI
            POP   ESI
            POP   EBX
            RET
F2T10 ENDP


RADIX PROC
            PUSH ECX
            PUSH EDX
            XOR   ECX,ECX
LOP1: XOR   EDX,EDX
            DIV   EBX
            PUSH DX
            INC   ECX
            OR    EAX,EAX
            JNZ   LOP1
LOP2: POP   AX
            CMP   AL,10
            JB    L1
            ADD   AL,7
L1:         ADD   AL,30H
            MOV   [ESI],AL
            INC   ESI
            LOOP LOP2
            POP   EDX
            POP   ECX
            RET
RADIX ENDP
END START
```

3.4 实验步骤

1.准备上机实验环境。

2.使用VISUAL STUDIO编写程序,要求满足本次实验要求,保存至SHOP.ASM和MENU.RC。使用 MASM32 汇编并连接源文件，观察提示信息，若出错则返回重新编辑源代码，保存后重新汇编，直至不再报错为止。

3.执行程序。按照程序设计要求进行交互，检查是否达到程序设计要求。

4.使用 TD32.EXE 对程序进行调试。观察调试过程与 TD16 的区别。

5.观察 INVOKE 语句翻译成机器码后的特点，记录压栈顺序。

6.比较 DOS、Windows 输出方式，观察 Win32 程序的几种字符串输出方式所用的函数原型。

3.5 实验记录与分析

1.实验环境条件：i7-7700HQ 2.80GHz，8G 内存；WINDOWS 10 下 MASM32。

2.汇编源程序、连接时未发生异常。

3.执行程序。

（1）程序创建窗口类测试，测试情况如图 5.7 所示，测试截图显示功能正常。

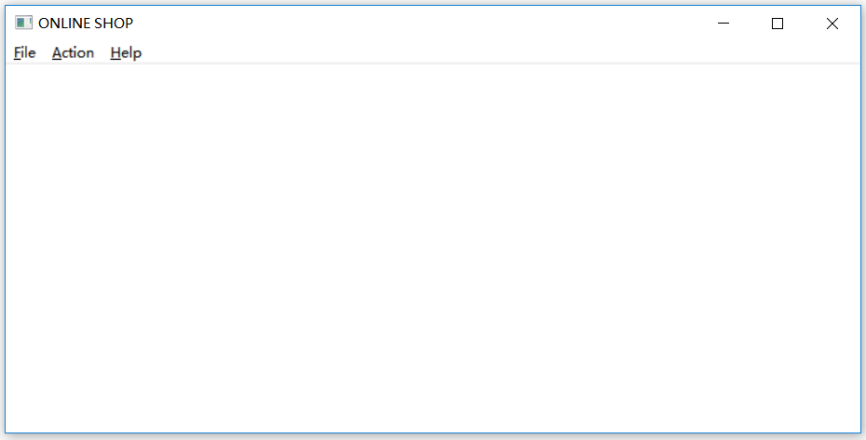

图 5.7 创建窗口类测试截图

（2）菜单框架显示测试,测试情况如图 5.8(a),(b),(c)所示,测试截图显示功能正常。

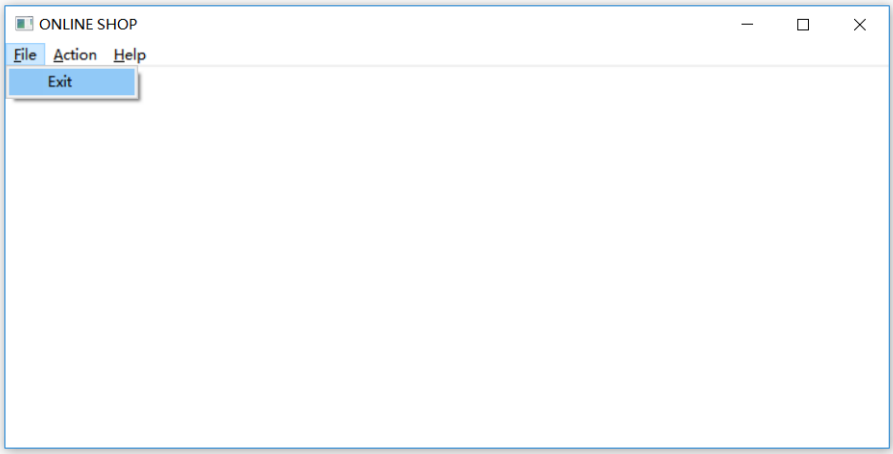

图 5.8(a) 菜单框架测试截图 1

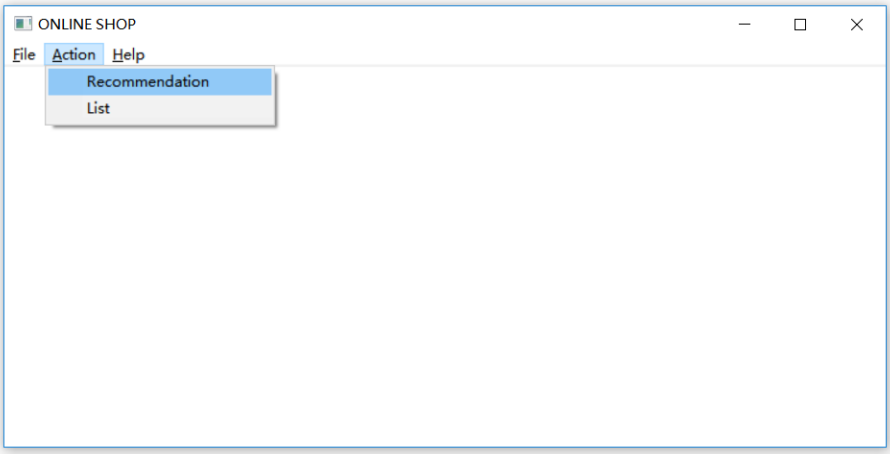

图 5.8(b) 菜单框架测试截图 2

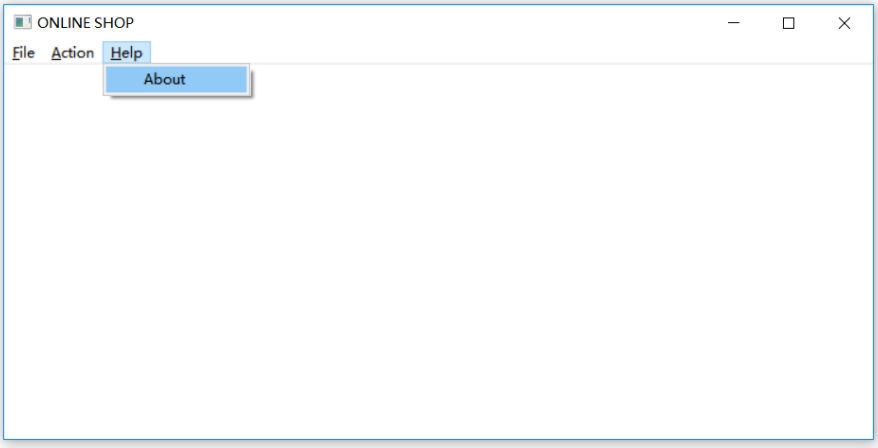

图 5.8(c) 菜单框架测试截图 3

（3）Help-About 功能测试，测试情况如图 5.9 所示，测试结果显示功能正常。

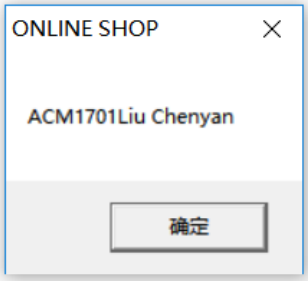

图 5.9 About 功能测试截图

（4）Action-Recommendation 功能测试，测试情况如图 5.10 所示，测试结果显示功能正常。

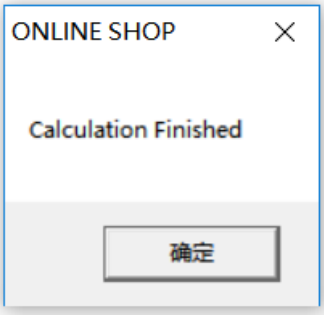

图 5.10 Recommendation 功能测试截图

（5）Action-List 功能测试，测试情况如图 5.11 所示,测试结果显示功能正常。

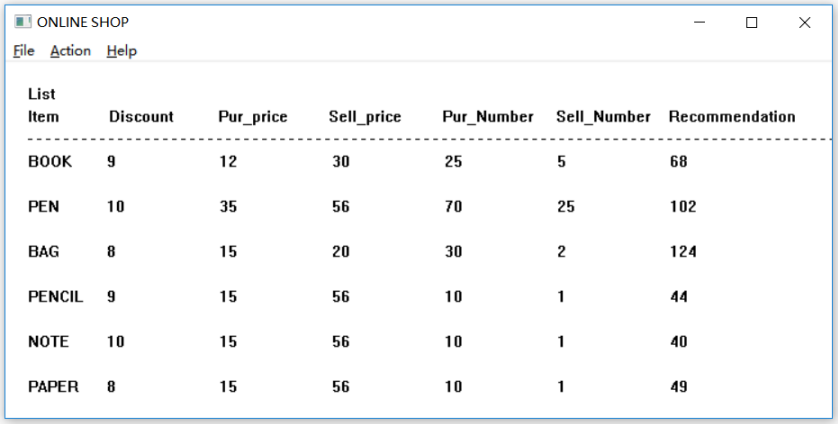

图 5.11 List 功能测试截图

（6）File-Exit 功能测试，测试情况如图 5.12 所示,测试结果显示功能正常。

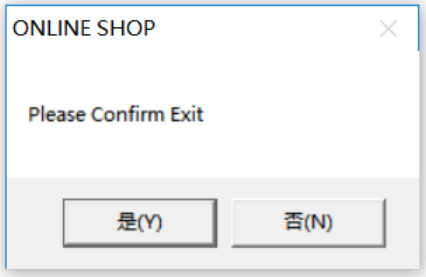

图 5.12 Exit 功能测试截图

4. 使用 TD32 对程序进行调试，观察调试过程与 TD16 的区别。

（1）打开 TD32，如图 5.13 所示，可知 TD32 的代码区、堆栈区、数据区的偏移地址均为 32 位，TD32 的寄存器区中，寄存器均为 32 位。而 TD16 的偏移地址，寄存器一般都默认为 16 位。

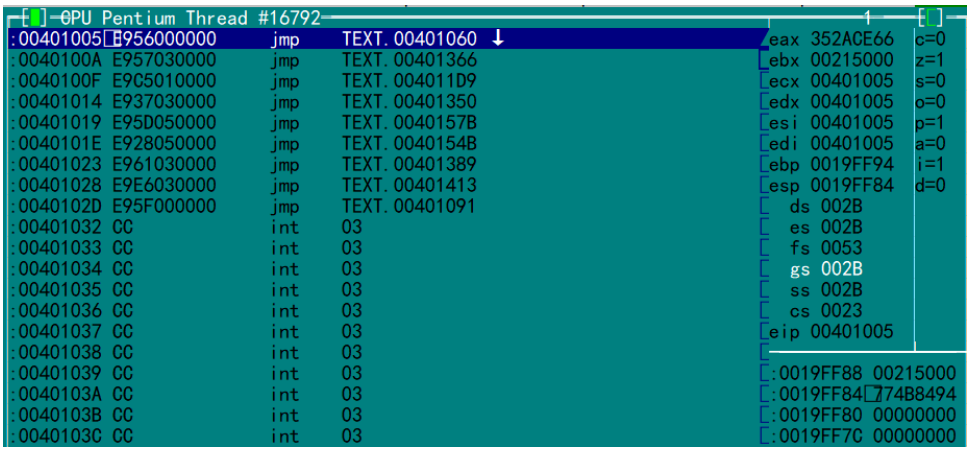

图 5.13 TD32 界面截图

（2）TD32 数据区中无关的数据均用“????”来表示，而 TD16 的数据区显示全部的数据。TD32 数据区如图 5.14 所示。

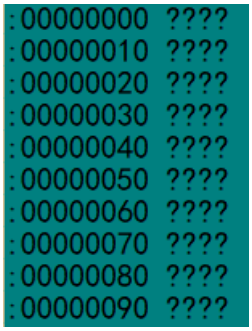

图 5.14 TD32 数据区截图

5. 观察 INVOKE 语句翻译成机器码后的特点，记录压栈顺序。

（1）INVOKE 语句被翻译成反汇编代码后被改为了 CALL 语句，如图 5.15 所示。

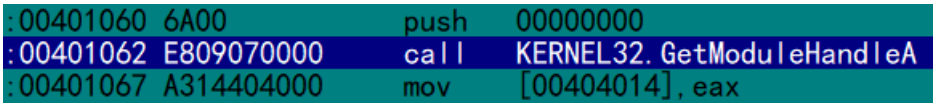

图 5.15 INVOKE 的反汇编代码截图

（2)在进入子程序时,子程序返回的偏移地址(00401062H)被压入栈中,EIP 为0019FF7CH。堆栈情况如图 5.16 所示。

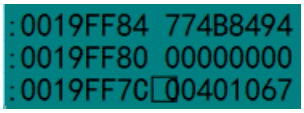

图 5.16 进入子程序后堆栈截图

6. 比较 DOS 和 Windows 输出方式，观察 Win32 程序集中字符串输出方式所用的函数原型。

（1）在 WIN32 程序中，输出使用的是 TEXTOUT 语句，如图 5.17 所示。

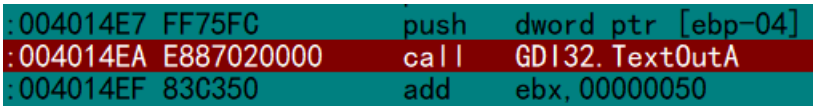

图 5.17 Win32 输出语句截图

TEXTOUT 函数声明如下：

```
BOOL TextOut(
HDC hdc, // 设备描述表句柄
int nXStart, // 字符串的开始位置 x 坐标
int nYStart, // 字符串的开始位置 y 坐标
LPCTSTR lpString, // 字符串
int cbString // 字符串中字符的个数
);
```

（2）在 DOS 中，需要将要输出的字符串首址传入 DX，然后通过 9 号功能调用实现的。

4 总结与体会

本次实验中，我成功编写了一个 WIN32 的窗口程序，对 WIN32 编程有了初步的认识，对 WIN32 编程的特点有了更深的了解。

通过实现 WIN32 窗口程序，我认识到窗口程序的标准框架为：主程序、窗口主程序、窗口消息、处理程序以及用户处理程序。系统首先执行主程序，主程序获得所需消息后调用窗口主程序，创建窗口类。该窗口收到的消息转发至窗口消息处理程序，实现相应功能。

菜单栏需要根据程序功能进行设计，在 WinProc 中对子程序进行调用。

WIN32 编程的一个特点是段的定义被简化，同时可以定义结构体的数据类型。WIN32 编程还支持 INVOKE 语句，可以调用函数的同时传入参数，这是 16 位宏汇编程序中不能做到的。

同时，本次实验首次使用了 TD32，其与 TD16 在偏移地址的表示和数据段信息的表示上有所不同，但还是非常相似的。

参考文献

[1] 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,2007：134-178