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： 2018-5-14，14：30-18：00　**实验地点：** 南一楼 804 室

指导教师： 朱虹

专业班级 计算机 201601 班

学　　号： U201614532　**姓　　名：** 吕鹏泽

同组学生：　**报告日期：** 2018 年　5 月 15 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：

成绩评定

实验完成质量得分（70 分）（实验步骤清晰详细深入，实验记录真实完整等）	报告撰写质量得分（30 分）（报告规范、完整、通顺、详实等）	总成绩（100 分）

指导教师签字：

日　　期：

目录

1. 实验目的与要求

（1）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**的平均利润率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　　Average　　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 Average、List 将分别实现计算平均利润率或显示 SHOP1 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。

点菜单项 Average 时，按照实验三的方法计算所有商品的平均利润率。用 TD32 观察计算结果。

点菜单项 List 时，要求能在窗口中列出 SHOP1 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（不要求用中文）。

3. 实验过程

3.1 任务 1

3.1.1 设计思想及存储单元分配

仿照实验 3 的输出所有商品信息和计算利润率，将其改成 win32 编程的风格，如寄存器寻址换成 32 位、函数调用使用 invoke，信息输出使用 textout。

首先修改菜单框架，增加 Action 下的计算利润率子菜单，当选择该菜单时调用计算利润率的函数。增加 About 下的窗口弹框，输出自己的姓名。

输出商品信息：ECX 用于商品的遍历，EBX 调整输出行间距，EDI 调整输出列间距，EDX 用于商品信息的寻址，商品信息转换成 ASCII 后存放在 BUF 中。

计算利润率：同实验 3 任务 1

3.1.2 流程图

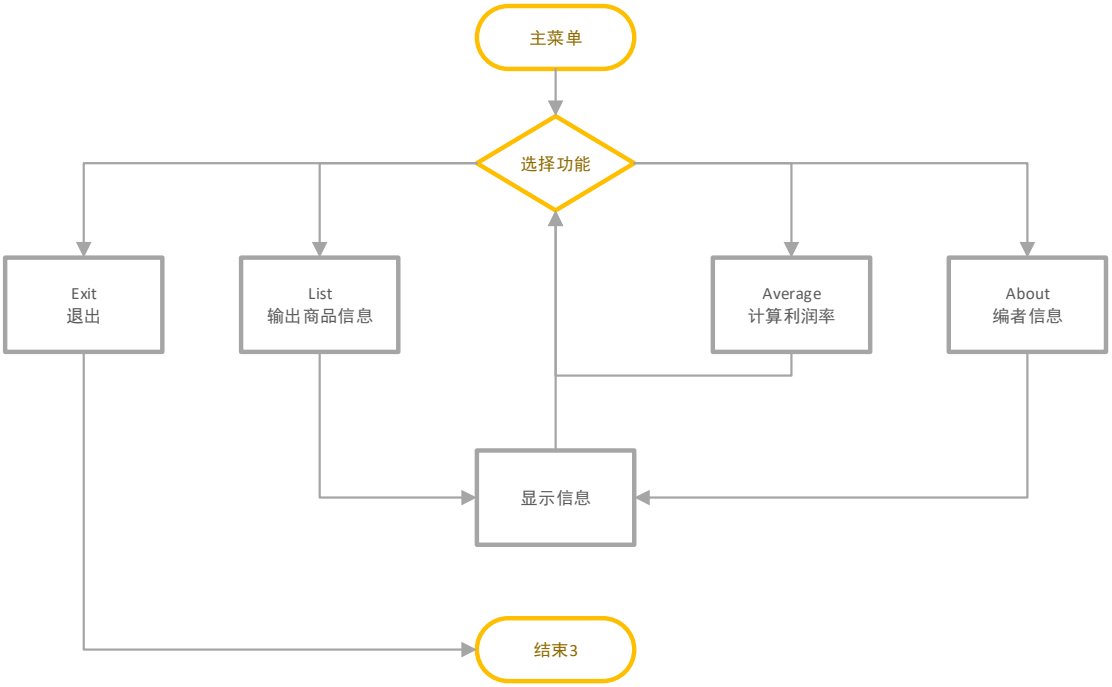

图 3.1 程序主体框架

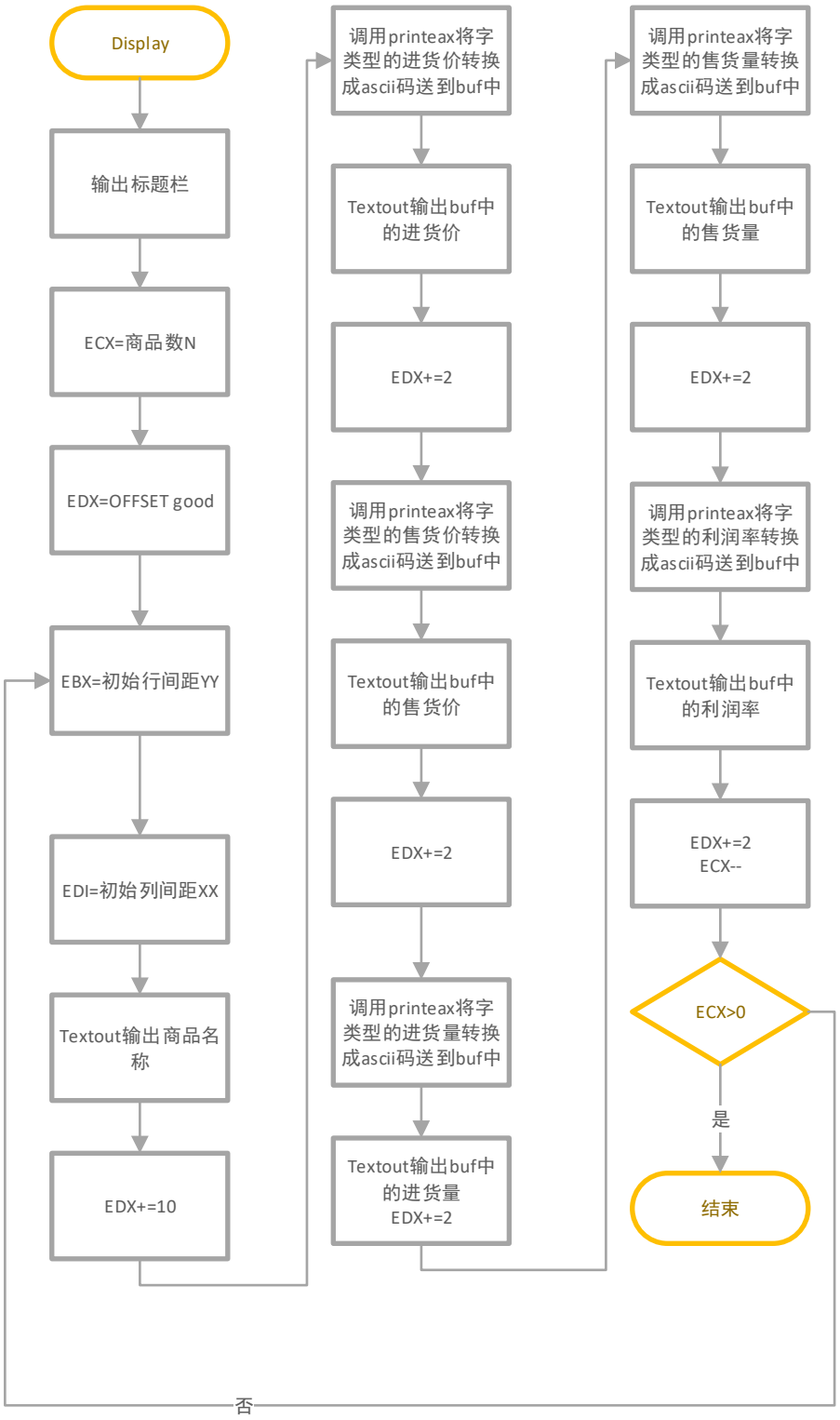

图 3.3 输出商品信息流程图

3.1.3 源程序

3.1.3.1 ASM 文件源码

```
.386
.model      flat,stdcall
option      casemap:none

;MessageBox proto:DWORD,:DWORD,:DWORD,:DWORD
WinMain    proto :DWORD,:DWORD,:DWORD,:DWORD
WndProc    proto :DWORD,:DWORD,:DWORD,:DWORD
Display    proto :DWORD
printeax proto
_AVERAGR_PROFIT PROTO :DWORD,:DWORD
include           menuID.INC

include           D:\Masm32\INCLUDE\windows.inc
include           D:\Masm32\INCLUDE\user32.inc
include           D:\Masm32\INCLUDE\kernel32.inc
include           D:\Masm32\INCLUDE\gdi32.inc
include           D:\Masm32\INCLUDE\shell32.inc

includelib      D:\Masm32\LIB\user32.lib
includelib      D:\Masm32\LIB\kernel32.lib
includelib      D:\Masm32\LIB\gdi32.lib
includelib      D:\Masm32\LIB\shell32.lib

commodity           struct
 commodity_name    db    10 dup(0)
 purchase_price    dw 0
 selling_price       dw 0
 total_num           dw 0
 sell_num            dw 0
 prt                          dw 0
commodity           ends

.data
ClassName        db             'TryWinClass',0
AppName           db              'Commodity information management platform',0
MenuName         db              'MyMenu',0
DlgName             db              'MyDialog',0
AboutMsg          db             'CS1601 LVPENGZE',0
hInstance        dd             0
CommandLine    dd              0
N EQU 5
H DD 0
good    commodity    <'BAG             ',15,32,89,70,0>
          commodity    <'PEN             ',10,25,30,2,0>
          commodity    <'HAT             ',13,25,25,20,0>
          commodity    <'CUP             ',5,15,40,24,0>
          commodity    <'JUICE         ',1,4,45,30,0>

BUF DB ?,?,?,?

msg_name          db             '    Commodity_Name',0 ;14
msg_pur         db             'Purchase_Price',0      ;14
msg_sell          db              'Selling_Price',0              ;13
msg_total_n     db              'Total_Num',0                  ;9
msg_sell_n      db              'Sell_Num',0                   ;8
msg_prt         db             'Prt',0                        ;3
```

```
flag DB 0
.code
Start:          invoke GetModuleHandle,NULL
        mov       hInstance,eax
        invoke GetCommandLine
        mov       CommandLine,eax
        invoke WinMain,hInstance,NULL,CommandLine,SW_SHOWDEFAULT
        invoke ExitProcess,eax
        ;;
WinMain          proc     hInst:DWORD,hPrevInst:DWORD,CmdLine:DWORD,CmdShow:DWORD
        LOCAL    wc:WNDCLASSEX
        LOCAL    msg:MSG
        LOCAL    hWnd:HWND
                 invoke RtlZeroMemory,addr wc,sizeof wc
        mov       wc.cbSize,SIZEOF WNDCLASSEX
        mov       wc.style, CS_HREDRAW or CS_VREDRAW
        mov       wc.lpfnWndProc, offset WndProc
        mov       wc.cbClsExtra,NULL
        mov       wc.cbWndExtra,NULL
        push     hInst
        pop       wc.hInstance
        mov       wc.hbrBackground,COLOR_WINDOW+1
        mov       wc.lpszMenuName, offset MenuName
        mov       wc.lpszClassName,offset ClassName
        invoke LoadIcon,NULL,IDI_APPLICATION
        mov       wc.hIcon,eax
        mov       wc.hIconSm,0
        invoke LoadCursor,NULL,IDC_ARROW
        mov       wc.hCursor,eax
        invoke RegisterClassEx, addr wc
        INVOKE CreateWindowEx,NULL,addr ClassName,addr AppName,\
                        WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,\
                        CW_USEDEFAULT,CW_USEDEFAULT,CW_USEDEFAULT,NULL,NULL,\
                        hInst,NULL
        mov       hWnd,eax
         MOV H,eax
         ;invoke MessageBox,hWnd,addr AboutMsg,addr AppName,0
        INVOKE ShowWindow,hWnd,SW_SHOWNORMAL

        INVOKE UpdateWindow,hWnd
        ;;
MsgLoop:         INVOKE GetMessage,addr msg,NULL,0,0
                 cmp       EAX,0
                 je         ExitLoop
                 INVOKE TranslateMessage,addr msg
                 INVOKE DispatchMessage,addr msg
        jmp       MsgLoop
ExitLoop:       mov       eax,msg.wParam
        ret
WinMain          endp

WndProc          proc     hWnd:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD ;40100Ah
        LOCAL    hdc:HDC
       .IF         uMsg == WM_DESTROY
        invoke PostQuitMessage,NULL
       .ELSEIF uMsg == WM_KEYDOWN
       .IF         wParam == VK_F1
                  ;;your code
       .ENDIF
       .ELSEIF uMsg == WM_COMMAND
       .IF         wParam == IDM_FILE_EXIT
             invoke SendMessage,hWnd,WM_CLOSE,0,0
```

```
        .ELSEIF wParam == IDM_FILE_LIST
              invoke Display,hWnd;004011F4
               MOV flag,1
         .ELSEIF wParam == IDM_FILE_AVG
               invoke _AVERAGR_PROFIT,OFFSET good,5
               MOV flag,0

        .ELSEIF wParam == IDM_HELP_ABOUT
              invoke MessageBox,hWnd,addr AboutMsg,addr AppName,0
        .ENDIF
        .ELSE
                    invoke DefWindowProc,hWnd,uMsg,wParam,lParam
                    ret
         .ENDIF
               xor       eax,eax
         ret
WndProc          endp

Display          proc     hWnd:DWORD
                  XX         equ    10
                  YY         equ    10
         XX_GAP equ    140
         YY_GAP equ    30
                  LOCAL    hdc:HDC
                  invoke GetDC,hWnd
                  mov       hdc,eax
                  invoke TextOut,hdc,XX+0*XX_GAP,YY+0*YY_GAP,offset msg_name,16
                  invoke TextOut,hdc,XX+1*XX_GAP,YY+0*YY_GAP,offset msg_pur,14
                  invoke TextOut,hdc,XX+2*XX_GAP,YY+0*YY_GAP,offset msg_sell,13
                  invoke TextOut,hdc,XX+3*XX_GAP,YY+0*YY_GAP,offset msg_total_n,9
                  invoke TextOut,hdc,XX+4*XX_GAP,YY+0*YY_GAP,offset msg_sell_n,8
                  invoke TextOut,hdc,XX+5*XX_GAP,YY+0*YY_GAP,offset msg_prt,3
                  ;;
                MOV ECX,N                                       ;商品总数
                MOV EBX,YY+YY_GAP                               ;控制输出行间距
                MOV EDX,OFFSET good[0]                          ;遍历所有商品
L1:
                MOV EDI,XX                                      ;控制输出列间距

                PUSHAD
                   invoke TextOut,hdc,EDI,EBX,EDX,10     ;输出商品名称
                POPAD
                ADD EDI,XX_GAP
                ADD EDX,10

                PUSHAD
                MOVSX EAX,WORD PTR [EDX]
                MOV ESI,OFFSET BUF
                invoke printeax
                   invoke TextOut,hdc,EDI,EBX,OFFSET BUF,4;输出商品进价
                POPAD
                ADD EDX,2
                ADD EDI,XX_GAP

                   PUSHAD
                MOVSX EAX,WORD PTR [EDX]
                MOV ESI,OFFSET BUF
                invoke printeax
                   invoke TextOut,hdc,EDI,EBX,OFFSET BUF,4;输出商品售价
                POPAD
                ADD EDX,2
                ADD EDI,XX_GAP
```

```
                PUSHAD
                MOVSX EAX,WORD PTR [EDX]
                MOV ESI,OFFSET BUF
                invoke printeax
                invoke TextOut,hdc,EDI,EBX,OFFSET BUF,4;输出商品进货量
                POPAD
                ADD EDX,2
                ADD EDI,XX_GAP

                PUSHAD
                MOVSX EAX,WORD PTR [EDX]
                MOV ESI,OFFSET BUF
                invoke printeax
                invoke TextOut,hdc,EDI,EBX,OFFSET BUF,4;输出商品售出量
                POPAD
                ADD EDX,2
                ADD EDI,XX_GAP

                PUSHAD
                MOVSX EAX,WORD PTR [EDX]
                MOV ESI,OFFSET BUF
                invoke printeax
                invoke TextOut,hdc,EDI,EBX,OFFSET BUF,4;输出商品利润率
                POPAD
                ADD EDX,2

                ADD EBX,YY_GAP
                DEC ECX
                JNZ L1
                ret
    Display     endp

    printeax PROC
     PUSHAD                          ;保护现场
     MOV EBX,10
     XOR CX,CX                       ;计数器清 0
     CMP EAX,0
     JGE _LOP1
     NEG EAX
     MOV BYTE PTR [ESI],'-'
     INC ESI
    _LOP1:                                  ;(EAX)除以 P，所得商->EAX，余数入栈，CX++，记录余
数个数
     XOR EDX,EDX
     DIV EBX
     PUSH DX
     INC CX
     OR EAX,EAX
     JNZ _LOP1
    _LOP2:                                  ;从栈中弹出一位 P 进制数，并将该数转换成 ASCII 码后输
出
     XOR EBX,EBX
     POP AX
     CMP AL,10
     JB _L1
     ADD AL,7
    _L1:                                    ;输出 P 进制数
     ADD AL,30H
     MOV BYTE PTR [ESI],AL
     INC EBX
     INC ESI
```

```
 LOOP _LOP2
 CMP EBX,4
 JL _P_L2
 JMP _EXIT1
_P_L2:
 MOV BYTE PTR [ESI],' '
 INC ESI
 INC EBX
 CMP EBX,4
 JL _P_L2
_EXIT1:
 POPAD                              ;恢复现场
 RET
printeax ENDP


_AVERAGR_PROFIT PROC GA1:DWORD,N1:DWORD
 PUSHAD
 XOR EBX,EBX
 MOV EBX, GA1
 MOV ESI, 5

LOOP_1:
 MOVSX EAX, WORD PTR [EBX][12] ;售价
 MOVSX EDX, WORD PTR [EBX][16] ;售出数
 IMUL EAX, EDX                              ;销售额

 MOVSX ECX, WORD PTR [EBX][10] ;进价
 MOVSX EDX, WORD PTR [EBX][14] ;进货量
 IMUL ECX, EDX                              ;成本

 SUB EAX, ECX
 IMUL EAX, 100
 CDQ
 IDIV ECX

 MOV WORD PTR [EBX][18], AX

 ADD EBX, 20
 DEC ESI

 JNZ LOOP_1
 POPAD
 RET
_AVERAGR_PROFIT ENDP

                end    Start
```

3.1.3.2 rc 文件源码

```
#define IDM_FILE_EXIT    10001
#define IDM_FILE_LIST    10002
#define IDM_FILE_AVG     10003
#define IDM_HELP_ABOUT 10101

MyMenu    MENU
BEGIN
 POPUP "&File"
        BEGIN
```

```
            MENUITEM "E&xit",IDM_FILE_EXIT
    END

    POPUP "&Action"
    BEGIN
            MENUITEM "A&verage",IDM_FILE_AVG
        MENUITEM "L&ist",IDM_FILE_LIST
    END

    POPUP "&Help"
    BEGIN
        MENUITEM "A&bout",IDM_HELP_ABOUT
    END
    END
```

3.1.3.3 inc 文件源码

```
IDM_FILE_EXITequ 10001
IDM_FILE_LIST equ 10002
IDM_FILE_AVG equ 10003
IDM_HELP_ABOUT   equ 10101
```

3.1.4 实验步骤

1.准备上机环境，编辑、汇编、文件 DEMO

2.选择 About 功能，观察弹出窗口

3.在 TD32 中观察的计算利润率

4.选择 List 功能，观察商品信息的输出

5.比较 TD32 与 TD16 的异同，比较 WIN32 程序与 16 位程序的异同

6.比较源码级调试与非源码级调试

7.修改偏移地址来改变观察的区间

8.编写和处理简单资源脚本，装入菜单，观察效果

9.观察 invoke 调用的入栈顺序

3.1.5 实验记录与分析

实验环境条件：WINDOWS 10 下 TD32 Version5.0

1.点击 about，显示弹框，结果如图 3.4 所示

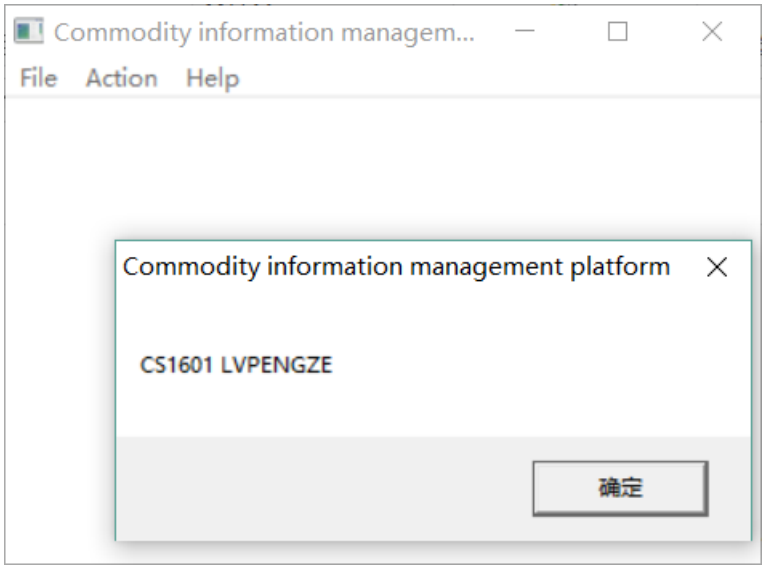

图 3.4 弹框

2.计算利润率前后输出商品信息，结果如图 3.5 所示

图 3.5a 商品信息

图 3.5b 计算利润率后输出商品信息

3.在 TD32 观察计算的利润率，以商品 pen 为例，其利润率结果如图 3.6 所示，为 0043H，转换成 10 进制数为 67，与理论值符合。

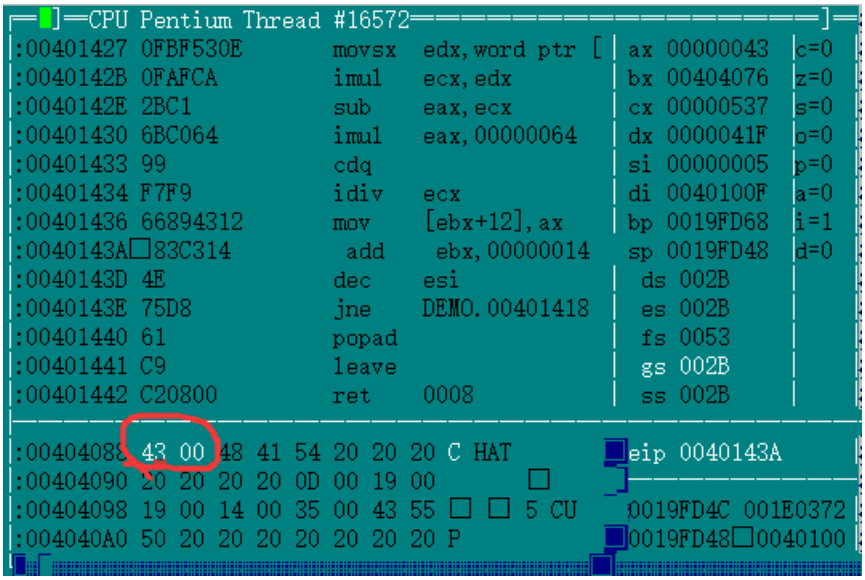

图 3.6 商品 pen 的利润率

4.TD32 与 TD16 的最大的不同在于寄存器变成了 32 位的，数据区与代码区也均为 32 位，可以存放最大 4GB 的数据或代码，此外，对于不属于本程序的数据被保护起来了，因此不可见，显示为？。

5.修改偏移地址观察其他区间，可以发现数据被保护起来了，显示为‘？’

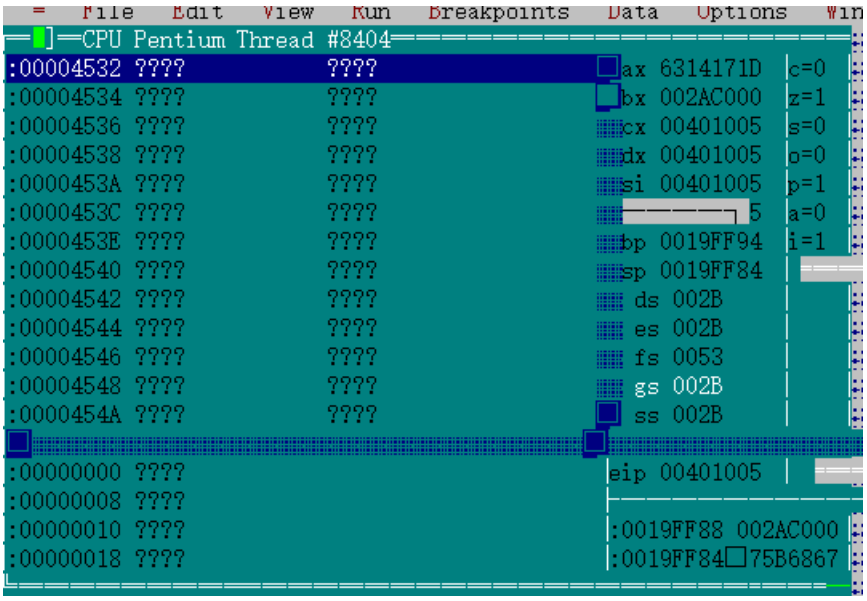

图 3.7 其他区间的数据

6.修改菜单脚本，可以达到修改菜单内容的效果

```
#define IDM_FILE_EXIT  10001
#define IDM_FILE_LIST  10002
#define IDM_FILE_AVG   10003
#define IDM_HELP_ABOUT 10101

MyMenu  MENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "E&xit",IDM_FILE_EXIT
    END

    POPUP "&Action"
    BEGIN
        MENUITEM "A&verage",IDM_FILE_AVG
        MENUITEM "L&ist",IDM_FILE_LIST
    END

    POPUP "&Help"
    BEGIN
        MENUITEM "A&bout",IDM_HELP_ABOUT
    END
END
```

图 3.8 使用 rc 文件显示菜单内容

7.对于 invoke 调用，首先要声明子程序原型，然后通过观察 textout 函数的调用可知参数是按从右到左的顺序入栈的，然后使用 call 调用子程序，当函数中使用参数时会从栈中读取参数。若是无参数的子程序，则直接 call 调用子程序。调用过程如图 3.9 所示

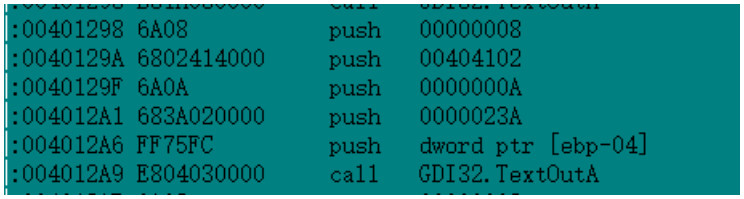

图 3.9 textout 调用

8.dos 中的串输出使用的 9 号调用，参数为输出字符串的首地址，以'$'字符结束，使用寄存器传参。win32 编程使用 textout 输出，参数有 5 个，分别为上下文句柄、起始 x 坐标、起始 y 坐标、字符串地址、串长度，使用的是堆栈传参。

4. 总结与体会

通过本次实验，我初步体会到了 win32 编程的特性，与 dos 中 16 位汇编相比，win32 编程可以支持更大的程序与更大的数据，最高可用 4GB 的内存，同时，它可以进行图形化的界面编写，可视化程度更高，界面与用户操作性更高，在编程风格上也更加类似于 C 语言等高级语言。

在进行这次实验时，我使用了从汇编网站上下载的菜单模板，通过修改.rc 文件来修改菜单栏，但是在修改后却出现了连接失败的问题，于是我重新复查文件，发现我只修改了 menu.rc 的内容，却没有在 menuID.inc 中添加新增菜单栏的句柄号。第二个错误是 about 功能 messagebox 弹出窗口错误，在运行了 about 选项之后，主窗口被取消选中且新窗口并没有如期弹出，只有在按下 alt 键后窗口才会出现，通过与同学的讨论与网上查询，发现是弹出的窗口被主窗口遮挡了，进一步检查程序发现是窗口重绘.ELSEIF uMsg == WM_PAINT 语句的问题，删除之后就好了。第三个错误是在调用 textout 时出现程序崩溃的问题，通过 TD32 的单步调试，发现是调用完 textout 后 esi，ecx 等寄存器的值被改变了，因此在调用前使用 pushad 进行保护问题就解决了。

win32 编程和 16 位编程在编写上面有一些差别，首先 32 位段的简化断定义方便了程序编写，且 invoke 函数调用更类似与 C 语言，编写更方便。此外，由于 32 位段的长度达到了 4GB，因此 win32 编程可以存放大量的数据和代码，可以编写更大功能更复杂的程序。

5. 参考文献

[1]许向阳，《80X86 汇编语言程序设计上机指南》“第七章 MASM32 环境”、“第八章 一个文本编辑器”。

[2] 汇编语言教学网站-》资料下载-》案例-》win32 程序、编译和连接

[3] 汇编语言教学网站-》资料下载-》书籍-》Win32 汇编程序的源码级调试其中的操作说明，给出了几种编译和链接生成执行程序的方法。

[4] MSDN（Microsoft Developer Network）， 有关 Windows API 的帮助。