日期：2017.5.3

目录

汇 编 语 言 程 序 设 计 实 验 报 告

1 实验目的与要求

本次实验的主要目的与要求有下面 5 点，所有的任务都会围绕这 5 点进行，希望大家事后检查自己是否达到这些目的与要求。

（1） 掌熟悉 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

任务：编写一个基于窗口的 WIN32 程序，实现学生成绩表信息的平均值计算及显示功能（借鉴前面实验中的一些做法），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值或显示所有成绩的功能（详见功能二的描述）。

功能二：每个学生的相关信息包括：姓名（结尾含 1 个以上的数值 0，共占 10 个字节），语文成绩（1 个字节），数学成绩（1 个字节），英语成绩（一个字节），平均成绩（1 个字节），等级（1 个字节）。要求采用结构变量存放学生的相关信息。学生人数至少 5 人。姓名和各科成绩直接在数据段中给定，不必运行时输入。**成绩表中最后一个学生必须使用自己的姓名**。

（1） 点菜单项 Average 时，计算平均成绩并给出等级（等级的定义见实验一，但这里不用单独显示等级）。平均成绩的计算仍按照实验一的公式进行。平均成绩和等级保存到上述结构变量的相应字段中。用 TD32 观察计算结果。

（2） 点菜单项 List 时，要求能在窗口中列出所有学生信息，包括姓名、各科成绩、平均成绩、等级等。如图 5.2 所示。平均成绩尚未计算时，平均成绩及等级显示为空白。

3 实验过程

3.1 任务

3.1.1 设计思想

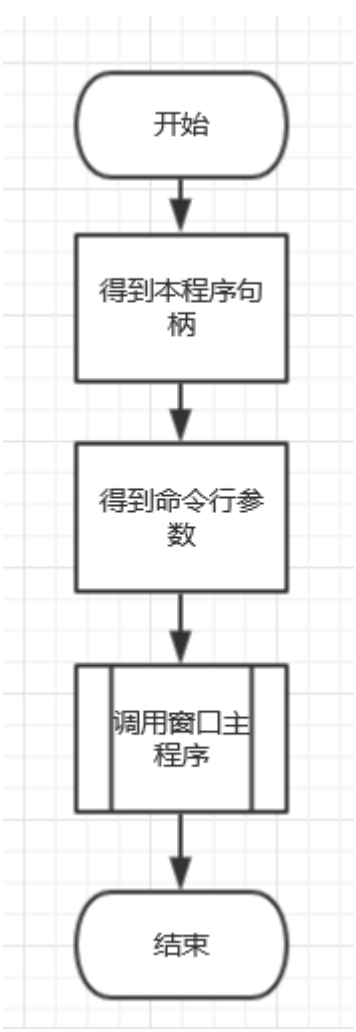
基于窗口的应用程序一般可以划分为四个部分：主程序，窗口程序，窗口消息处理程序，用户处理程序。操作系统首先执行“主程序”，“主程序”将该窗口收到的信息通过操作系统转发到“窗口消息处理程序”，“窗口消息处理程序”判断收到的消息种类，决定应该使用“用户处理程序”中的哪个来完成相应功能。

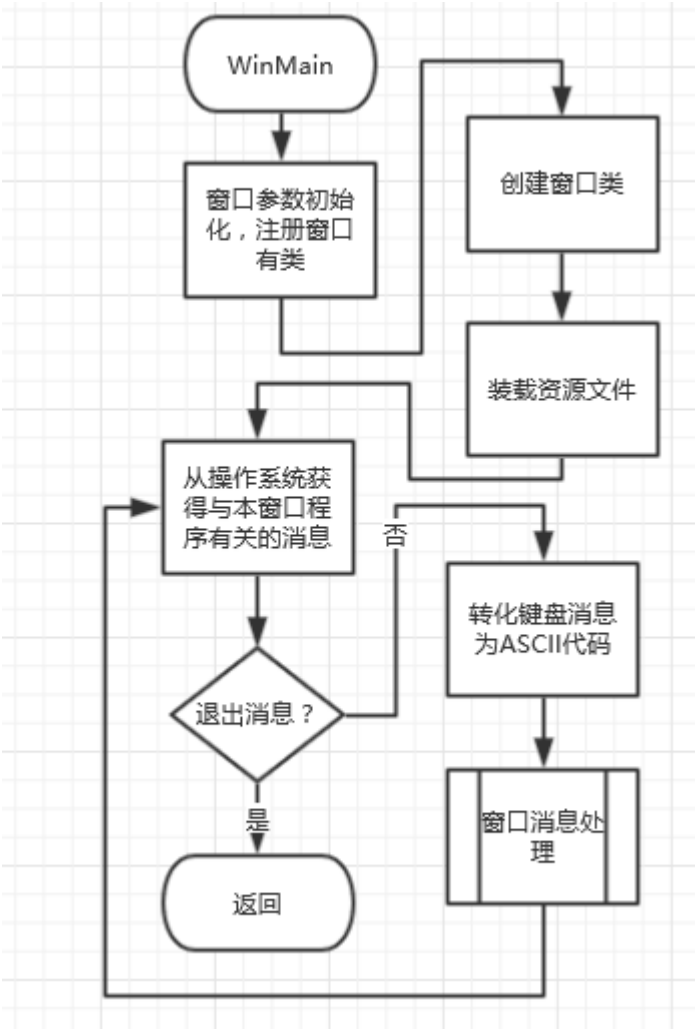
首先我们在主程序中通过调用 windows API ： RegisterClassEx; CreateWindowEx 来创建主窗口，这里可以设置很多参数来创建合适的窗口；大小，风格等。然后进入窗口消息处理程序

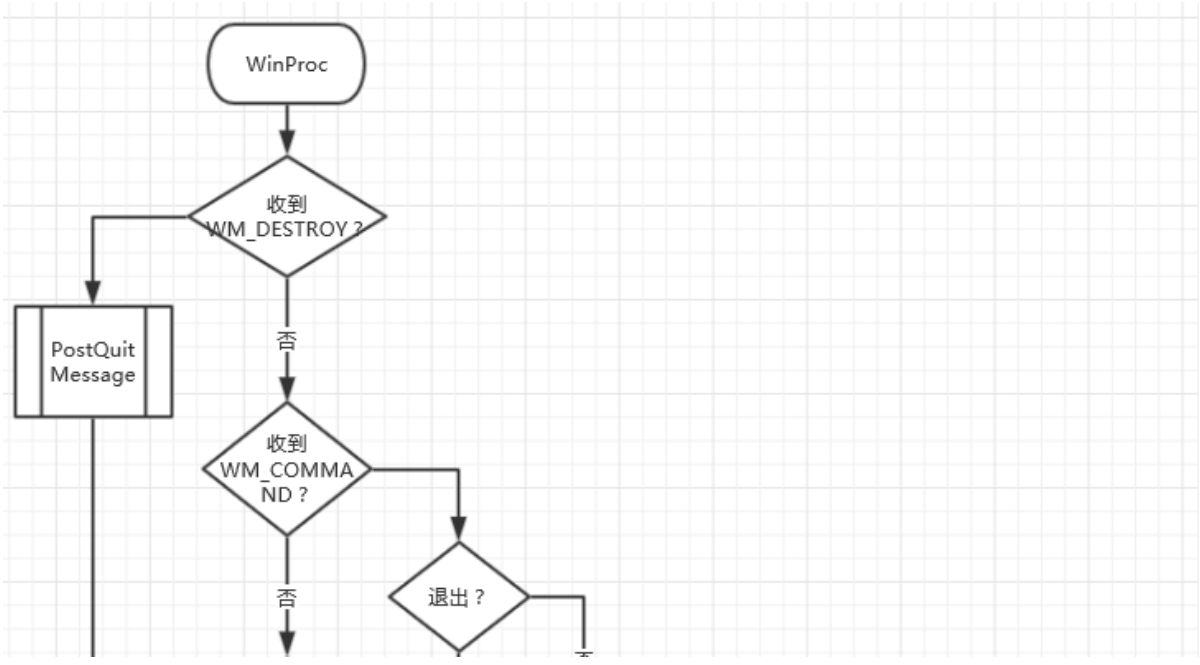
窗口消息处理程序首先要处理一些必要的信息：重绘操作，destory 信息，以及未匹配的默认信息，其他的信息为自己触发的信息。分别进行用户函数的处理。

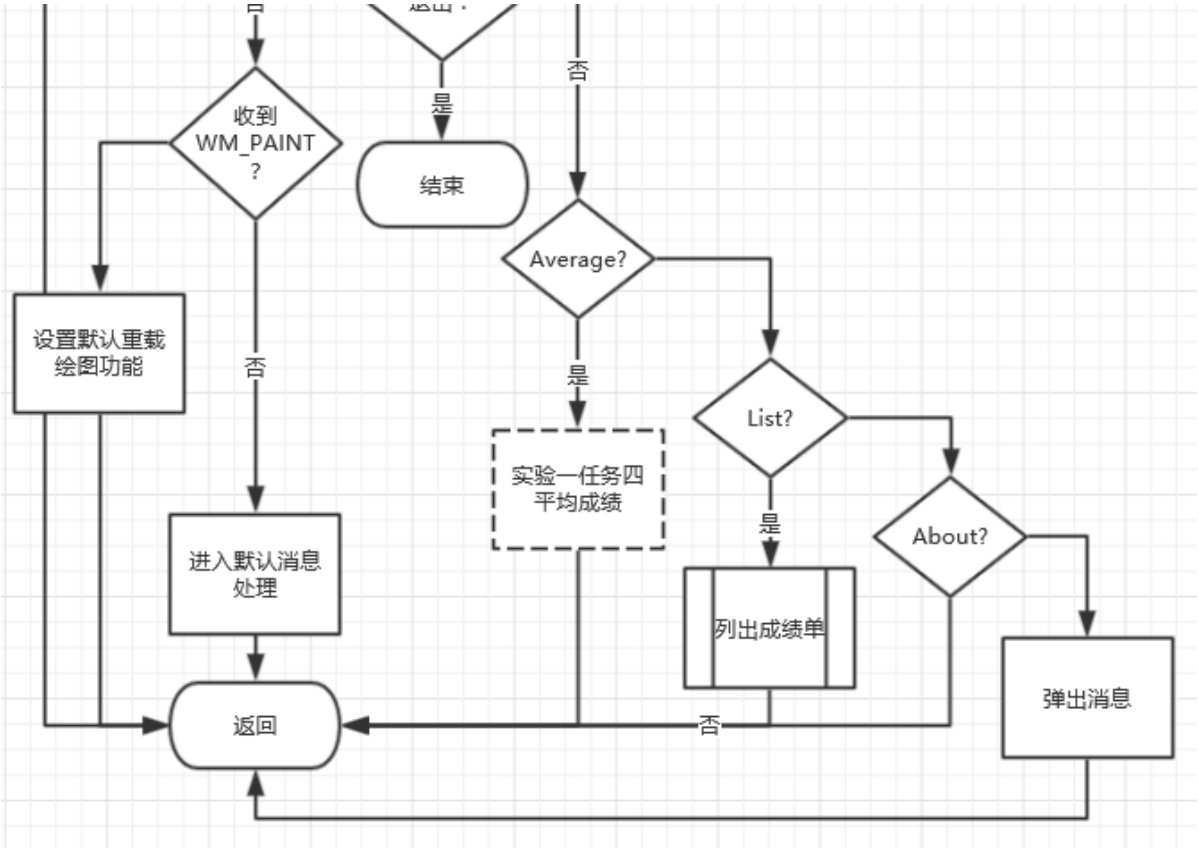
用户信息有四个，退出和弹出子窗口都是直接使用 win API 来做，设置平均值的函数也是去使用实验一的代码，然后输出成绩单也就是简单的输出，没什么好说的。

3.1.2 流程图


开始
得到本程序句柄
得到命令行参数
调用窗口主程序
结束


WinMain
窗口参数初始化，注册窗口有类
创建窗口类
装载资源文件
从操作系统获得与本窗口程序有关的消息
否
转化键盘消息为ASCII代码
退出消息？
是
窗口消息处理
返回


WinProc
收到WM_DESTROY？
PostQuit Message
否
收到WM_COMMAND？
否
退出？



3.1.3 源代码

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

WinMain    proto :DWORD,:DWORD,:DWORD,:DWORD
WndProc    proto :DWORD,:DWORD,:DWORD,:DWORD
Paint_Proc PROTO :DWORD, hDC:DWORD
Display    proto :DWORD

include         menuID.INC

include         windows.inc
include         user32.inc
include         kernel32.inc
include         gdi32.inc
include         shell32.inc
```

```
includelib      user32.lib
includelib      kernel32.lib
includelib      gdi32.lib
includelib      shell32.lib

return MACRO arg
            mov eax, arg
            ret
          ENDM



student          struct
             myname      db    10 dup(0)
             chinese    db    0
             math         db    0
             english    db    0
             average    db    0
             grade       db    0
student          ends

.data
ClassName       db           'TryWinClass',0
AppName          db           'Our First Window',0
MenuName         db           'MyMenu',0
DlgName             db           'MyDialog',0
AboutMsg         db           'I am ACM1501 Xin Jie',0
hInstance        dd           0
CommandLine    dd           0
buf              student    <>
             student    <'Jin',96,98,100,0,'A'>
             student    3 dup(<>)
msg_name         db           'name',0
msg_chinese    db           'chinese',0
msg_math         db           'math',0
msg_english    db           'english',0
```

```
msg_average    db          'average',0
msg_grade      db          'grade',0
chinese        db          2,0,0,0, '96'
math           db          2,0,0,0, '98'
english        db          3,0,0,0, '100'
average        db          2,0,0,0, '98'

.code

Start:          invoke GetModuleHandle,NULL ; 参数为 0(NULL) 表明获得此程序句柄
          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   ;变量名：类型
          LOCAL   msg:MSG
          LOCAL   hWnd:HWND
          local   mywidth:DWORD
          local   myheight:DWORD
                  invoke RtlZeroMemory,addr wc,sizeof wc
          mov       wc.cbSize,SIZEOF WNDCLASSEX ;这个结构体大小
          mov       wc.style, CS_HREDRAW or CS_VREDRAW ;窗口风格
          mov       wc.lpfnWndProc, offset WndProc    ;窗口消息处理的指针
          mov       wc.cbClsExtra,NULL ;
          mov       wc.cbWndExtra,NULL
          push      hInst
          pop       wc.hInstance ;处理的窗口的句柄
          mov       wc.hbrBackground,COLOR_WINDOW+1 ;背景色
          mov       wc.lpszMenuName, offset MenuName ;资源文件中描述菜单的资源文
```

```
件名字符串的指针
        mov     wc.lpszClassName,offset ClassName ;  指向类名称的指针
        invoke LoadIcon,NULL,IDI_APPLICATION
        mov     wc.hIcon,eax; hIcon  窗口图标资源
        mov     wc.hIconSm,0
        invoke LoadCursor,NULL,IDC_ARROW
        mov     wc.hCursor,eax ; hCursor  光标资源
        invoke RegisterClassEx, addr wc ;注册窗口
       mov myheight, 500
       mov mywidth, 800
        INVOKE CreateWindowEx,NULL,addr ClassName,addr AppName,\
                    WS_OVERLAPPEDWINDOW,CW_USEDEFAULT,\
                    CW_USEDEFAULT,mywidth,myheight,NULL,NULL,\
                    hInst,NULL ;创建窗口
        mov     hWnd,eax ;执行成功返回窗口句柄
        INVOKE ShowWindow,hWnd,SW_SHOWNORMAL
        INVOKE UpdateWindow,hWnd
        ;;
MsgLoop:        INVOKE GetMessage,addr msg,NULL,0,0 ;msg,  句柄,消息范围
                cmp     EAX,0 ; 0  是返回的退出消息
                je      ExitLoop
                INVOKE TranslateMessage,addr msg ;将键盘输入转化为对应 ACSCII 码
                INVOKE DispatchMessage,addr msg ;  转移到  窗口消息处理的指针所在
函数
        jmp     MsgLoop
ExitLoop:       mov     eax,msg.wParam
        ret
WinMain         endp

WndProc                                                                 proc
hWnd:DWORD,uMsg:DWORD,wParam:DWORD,lParam:DWORD
        LOCAL   hdc:DWORD
       LOCAL Ps     :PAINTSTRUCT
     .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 ;接受消息的窗口，要发送的消息，消息类别，消息类别
            .ELSEIF wParam == IDM_ACTION_AVERAGE
                 call SET_AVERANGE_GRADE
            .ELSEIF wParam == IDM_ACTION_LIST
                 invoke Display,hWnd
            .ELSEIF wParam == IDM_HELP_ABOUT
                 invoke MessageBoxA,hWnd,addr AboutMsg,addr AppName,0 ;所属窗口，显示消息，标题，类型
            .ENDIF
        .ELSEIF uMsg == WM_PAINT
            invoke BeginPaint,hWnd,ADDR Ps
            mov hdc, eax
            ;invoke Paint_Proc,hWnd,hdc
            invoke EndPaint,hWnd,ADDR Ps
            return 0
              ;;redraw window again
        .ELSE
                    invoke DefWindowProc,hWnd,uMsg,wParam,lParam
                    ret
        .ENDIF
             xor       eax,eax
             ret
WndProc          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

    ; ------------------------------------------------------

    ; ------------------------------------------------------
    ; The following code draws the border around the client area
    ; ------------------------------------------------------
    invoke GetClientRect,hWin,ADDR Rct

    add Rct.left,      1
    add Rct.top,       1
    sub Rct.right,     1
    sub Rct.bottom, 1



    add Rct.left,      4
    add Rct.top,       4
    sub Rct.right,     4
    sub Rct.bottom, 4


    return 0

Paint_Proc endp
SET_AVERANGE_GRADE:
     PUSH ESI
     PUSH EAX
     PUSH EBX
     PUSH ECX
     PUSH EDX
```

```
  MOV SI, 0
  MOV CX, 5
MATH:
  MOV AX, 0
  MOV BX, 0
  MOV DX, 0
  MOV AL, buf[15].chinese
  MOV AH, 2
  MUL AH ;AX IS CHINESE * 2 <= 200 16 IS ENOUGH
  MOV BL, buf[15].math;MATH GRADE

  MOV BH, 0
  ADD BX, AX
  MOV AL, buf[15].english ;ENGLISH

  MOV AH, 0
  MOV DL, 2
  DIV DL
  ADD BX, AX
  MOV AX, 2
  MUL BX
  MOV BL, 7
  DIV BL

  MOV buf[15].average, AL
  ADD SI, 15
  LOOP MATH

  POP EDX
  POP ECX
  POP EBX
  POP EAX
  POP ESI
  RET
```

```
Display         proc     hWnd:DWORD
                 XX        equ   10
                 YY        equ   10
            XX_GAP equ    100
            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,4
                 invoke TextOut,hdc,XX+1*XX_GAP,YY+0*YY_GAP,offset msg_chinese,7
                 invoke TextOut,hdc,XX+2*XX_GAP,YY+0*YY_GAP,offset msg_math,4
                 invoke TextOut,hdc,XX+3*XX_GAP,YY+0*YY_GAP,offset msg_english,7
                 invoke                  TextOut,hdc,XX+4*XX_GAP,YY+0*YY_GAP,offset
msg_average,7
                 invoke TextOut,hdc,XX+5*XX_GAP,YY+0*YY_GAP,offset msg_grade,5
                 ;;
               push eax
               xor ax, ax
               mov al, buf[0].chinese
               mov bl, 10
               div bl ; al is /      ah is %

               pop eax
                 invoke                  TextOut,hdc,XX+0*XX_GAP,YY+1*YY_GAP,offset
buf[1*15].myname,3
                 invoke                  TextOut,hdc,XX+1*XX_GAP,YY+1*YY_GAP,offset
chinese+4,chinese
                 invoke     TextOut,hdc,XX+2*XX_GAP,YY+1*YY_GAP,offset      math+4,
math
                 invoke                  TextOut,hdc,XX+3*XX_GAP,YY+1*YY_GAP,offset
english+4,english
                 invoke                  TextOut,hdc,XX+4*XX_GAP,YY+1*YY_GAP,offset
average+4,average
                 invoke                  TextOut,hdc,XX+5*XX_GAP,YY+1*YY_GAP,offset
buf[1*15].grade,1
```

```
                ret
Display         endp

                end    Start
```

3.1.4 实验步骤

1. 配置环境
   i. 操作系统 ： windows 10
   ii. 设置环境变量
2. 安装 MAXSM32 软件包，观察结构目录
3. 编写和处理简单资源脚本，装入菜单，观察效
4. 试对\masm32\EXAMPLE1\3DFRAMES\下的例子，进行汇编、连接、运行和调试（TD32.EXE）。观察 WIN32 执行程序代码的特点和执行流程。体会基于窗口的应用程序所包含的四个部分之间的衔接关系。
5. 观察 TD32 与 16 位 TD 的异同。
6. 用 TD32 观察代码区或数据区时，若所观察的地址范围不是与被调试程序相关的区间，则对应内存中的数据会因为被系统保护了而读不出来（将用？代替），请通过修改偏移地址来改变观察的区间，记录此现象。
7. 观察收到的消息，记录每个菜单项或按键等操作所对应的消息信息。
8. 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函数的原型。
9. 观察结构变量的平均成绩等字段的偏移，体会结构变量优点。
10. 观察简化段的效果。
11. 观察 Invoke 语句翻译成机器码后的特点，记录参数压栈顺序。

3.1.5 实验记录与分析

1. 简单运行 bat 文件设置自己的环境变量
2. 观察 masm 目录结构：

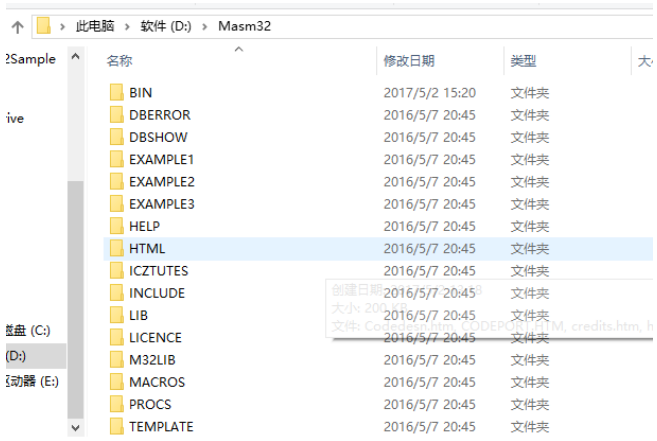


图 1- 1 目录结构

与课本对照，我们需要的链接文件在 lib include 下可以找到，我们的编译器，连接器在 bin 目录下，还有几个例子文件夹

3. 很简单得进行修改即可，按照原来文件的格式修改就好。

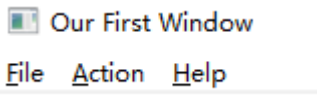


图 1- 2 程序菜单

4. 试对\masm32\EXAMPLE1\3DFRAMES\下的例子，进行汇编、连接、运行和调试（TD32.EXE）。观察 WIN32 执行程序代码的特点和执行流程。体会基于窗口的应用程序所包含的四个部分之间的衔接关系

   Win32 代码进行汇编链接运行后，我发现基于窗口的应用程序在调用主窗口之后你的程序信息其实是在 windows 与自己的程序之间传递的，你已经不是全部的信息都在你的程序中掌握的了，很大一部分是 windows 掌握的。Windows 通过对句柄（地址）的统筹来掌控这些信息。

5. 观察 TD32 与 16 位 TD 的异同。

   TD32 与 16 位 TD 我觉得不同的地方不是很多。基本在预料之内。首先由于你的打开环境变成了 windows，已经不像在 dosbox 下一个简单的环境了，你要让 windows 环境可以较为合理得打开 TD。这里面涉及到自己的计算机语言选项，自己命令行风格。其次打开后，在界面上没什么区别，但是寄存器显示变成了 32 位寄存器，并且多增加了以前没有见过的寄存器。

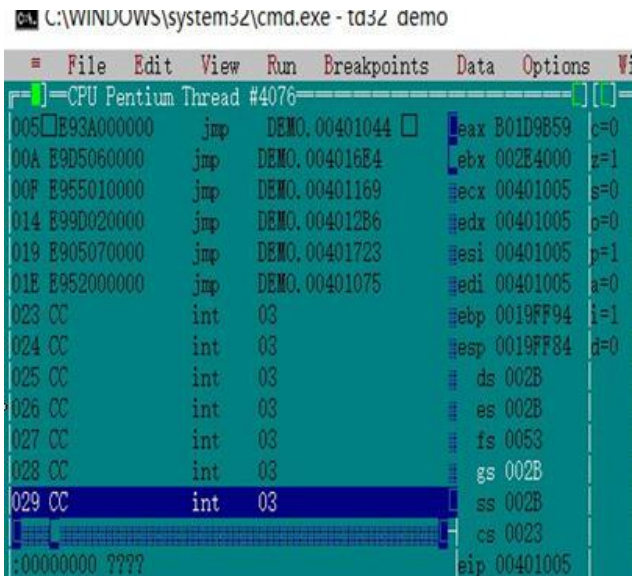


图 1- 3

6. 用 TD32 观察代码区或数据区时，若所观察的地址范围不是与被调试程序相关的区间，则对应内存中的数据会因为被系统保护了而读不出来（将用？代替），请通过修改偏移地址来改变观察的区间，记录此现象。

7. 观察收到的消息，记录每个菜单项或按键等操作所对应的消息信息。

可以使用热键，先点击 Alt 键，窗口跳转到菜单选择，分别使用 F,A,H 可以选择对应的主菜单，继续输入所选主菜单下的子菜单，可选择相应的功能函数。

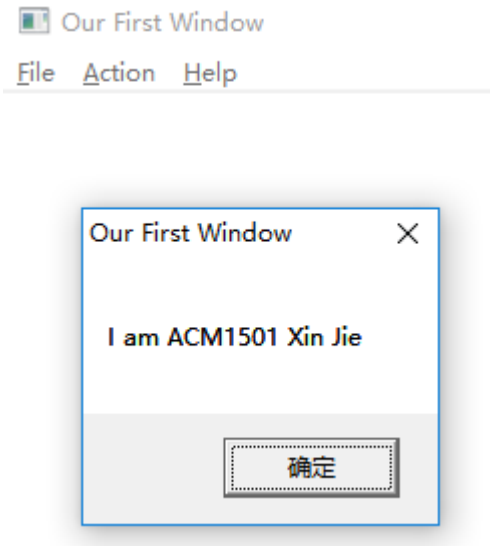


图 1- 4

8. 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函数的原型。

DOS 下我们的输出是通用使用 int 的系统功能调用来实现的，而在 windows 下我们却没有这样的系统功能调用，但是他提供了 API 接口让我们可以比较轻松得访问到他的内置功能。我们将需要调用它的程序的地址传入 TestMessage 中然后就可以使用。 这相当于自己将程序的执行转移回了 windows 与 int 中断本质上类似（个人理解）。

9. 观察结构变量的平均成绩等字段的偏移，体会结构变量优点。

   观察结构变量的平均成绩等字段的偏移，如语句 MOV AL,buf[SI].chinese 将 SI/(SIZEOF(BUF))个结构中语文成绩赋值给 AL,结构化变量的引用方式的确较为简单便捷。

10. 观察简化段的效果。

    简化段的格式为 .CODE 创建一个代码段，段名为可选项，不给出段名时则采用默认段名，对于多个代码段的模型，则应为每一个代码段指定段名。

4 总结与体会

这次试验我觉得需要自己仔细注意的地方主要有两点，第一点是我们在 32 位的模拟环境下进行的此次试验，与前几次的实验肯定有很多不相同的地方，第二点是对于调用 API 的熟悉。

首先是对于第一点的理解，其实我的理解十分不深入也不知道正确与否，32 未下的寻址方式肯定是 32 位的，我们在对内存的访问中，记录地址时候肯定只能用 32 位寄存器，然后 push 和 pop 我们最好也使用 32 位的。对于几种简化段的定义，平坦模式之类，自己还没有弄得很明白。

然后是 windows 编程。Windows 编程中 windows 对于窗口的组织：我在 windows 的任务管理器中发现自己程序的 exe 下面还可以看到窗口的程序执行，所以说我觉得 windows 对于每个窗口都是将控制权控制在自己手中而不是你写的程序手中，这不是简单的函数调用，你将控制权也给了 windows，我不清楚自己的理解对不对，但是我觉得应该是这样子。相当于是 windows 开了多个进程。

接下来是一些编程的体会，编程中遇到了许多问题，其中的一个是我弹出窗口时的速度极其缓慢，我在每个地方进过测试后发现他是在一个窗口弹出后接下来弹出的窗口就会非常缓慢。最后经过询问别人，查找资料，明白了那是因为 windows 要进行窗口重绘而你没有把重绘信息给写好，所以自己写好之后就解决了问题。

参考文献

[1] 王爽.汇编语言.第三版.清华大学出版社,2003 01- 310
[2] 曹忠升 80X86 汇编语言程序设计 华中科技大学出版社
