北京理工大學

汇编语言与接口技术个人实验报告

汇编语言与接口技术个人实验报告

Personal Assignment Report on Assembly Language and Microcomputer Interface

学　　院: 徐特立学院

专　　业: 计算机科学与技术

学生姓名: 关晓宇

学　　号: 1120190699

2022 年 6 月 6 日

汇编语言与接口技术个人实验报告

摘　要

本实验报告分别讲解了三次个人上机作业的程序功能、代码实现思路、程序重难点以及解决方案。一，大数相乘，此实验的编码难点在于逆序存放数组并模拟乘法竖式进行机器计算。二，结合 Windows 界面编程和浮点数编程实现完善的计算器功能，此实验的难点在于熟练掌握浮点数运算指令，按照运算符优先级编码实现计算器运算功能。三，使用 Windows 界面风格完成两个文件的按行比对，此实验的难点在于使用相关函数创建 Windows 窗口界面并完成消息处理。

关键词：x86 汇编语言，浮点数计算，大数相乘，Windows 界面汇编

Abstract

This report explains the functions, algorithm structure, program difficulties and solutions of three personal assignments. First, multiplying two large numbers, the program difficulty of this assignment is to store the array in reverse order and simulate the multiplication vertical for machine calculation. Second, combine Windows interface programming and floating-point number programming to achieve perfect calculator functions, the difficulty of this assignment is to master the floating-point number operation instructions in .80386 assembly language as well as implement the calculator operation function according to the operators' priority. Third, use the Windows interface style to complete the line-by-line comparison of two files, the difficulty of this assignment is to use the relevant libraries to create a Windows interface and complete the message processing.

Keywords：x86 assembly language, floating-point number calculations, multiplication of large numbers, Windows interface

目 录

摘 要 I
第 1 章 大数相乘 1
1.1 实验目的 1
1.2 实验环境 1
1.3 代码思路 1
1.3.1 str2int 函数 1
1.3.2 int2str 函数 2
1.3.3 BigIntMultiply 函数 3
1.4 运行效果 4
1.5 实验心得 5
第 2 章 文件比较 6
2.1 实验目的 6
2.2 实验环境 6
2.3 代码思路 6
2.3.1 注册并打开 Windows 窗口 6
2.3.2 消息处理 8
2.3.3 文件选择窗口 9
2.3.4 文件比较 10
2.4 运行效果 11
2.5 实验心得 15
第 3 章 计算器 16
3.1 实验目的 16
3.2 实验环境 16
3.3 代码思路 16
3.3.1 计算机窗口 UI 设计 16
3.3.2 按键处理 18
3.4 运行效果 18
3.5 实验心得 20

第 1 章 大数相乘

1.1 实验目的

大数相乘。要求实现两个十进制大整数的相乘（100 位以上），输出乘法运算的结果。此实验的编码难点在于逆序存放数组并模拟乘法竖式进行机器计算。

1.2 实验环境

编译器及开发工具：Visual Studio 2019，MASM32

处理器：Intel(R) Core(TM) i5-8265U CPU（80386 机器）

1.3 代码思路

本实验采用控制台编程，在文件数据段前声明了 C 函数 scanf、printf 来进行控制台输入输出。基本思想是将乘数 A、B 读入字符数组（dword 150 dup(0)），然后分别检查乘数 A、B 是否为负数，并进行符号处理以确定结果的正负号。然后利用堆栈循环将无符号的乘数 A、B 分别倒序存储在 dword 数组中，通过模拟竖式的大数相乘将无符号的结果存储在另一个 dword 数组中，最后再次利用堆栈循环将每个结果数字加'0' 变为字符并以正常序存储在字符数组中输出。

1.3.1 str2int 函数

函数声明如下：

```
str2int proc  C CharNumber:ptr byte,  IntNumber:ptr dword,  len:dword
```

其中，CharNumber 是一个乘数的字符数组指针（不带负号），len 是该乘数的长度，该函数的作用是将字符数组中的每个 byte（'0'-'9'）转为 dword（0-9），并以倒序存储在 IntNumber 数组中。

具体的循环操作用到了堆栈：CharNumber 中的内容依次入栈，高位先入栈，然后依次出栈，低位先出栈，实现反转功能。出栈时 byte（'0'-'9'）减去'0'得 dword（0-9），由此将 byte 型数组转化为 dword 型数组。具体实现细节以及需要注意的细节如下：

```
  MOV   ecx,  len          ;字符串长度为循环的次数，ecx控制循环次数
  MOV   esi,  CharNumber   ;esi是位置指针

LOOP_IN:                 ;循环每次拿取字符数组中一个字节的字符0-9转数字0-9并入栈
```

```
  MOVZX eax,  byte ptr[esi] ;在指针处取一个字节的字符，扩展成32位，高位填0
  SUB   eax,  30H           ;30H是字符0的Ascii码
  PUSH  eax
  INC   esi
  LOOP  LOOP_IN

  MOV   ecx,  len           ;字符数组：高位先入栈 int数组：低位先出栈，放在高位
  MOV   esi,  IntNumber

LOOP_OUT:
  POP   eax
  MOV   dword ptr[esi], eax
  ADD   esi, 4              ;int双字，四个字节，因此此处ESI+4
  LOOP  LOOP_OUT
  ret
```

1.3.2 int2str 函数

本函数与 str2int 函数思想相似，因此不再赘述。具体实现细节以及需要注意的细节如下：

```
  MOV   ecx,  lengthResult
  MOV   esi,  0

LOOP_IN:                ;循环将resultint数组中数字0-9+‘0’转字符0-9入栈
  MOV   eax,  dword ptr resultInt[4 * esi]
  ADD   eax,  30H       ;30H是字符0的Ascii码
  PUSH  eax
  INC   esi
  LOOP  LOOP_IN

  MOV   ecx,  lengthResult
  MOV   esi,  0

LOOP_OUT:             ;循环出栈，将eax的最低字节al存在resultchar中
  POP   eax
  MOV   byte ptr resultChar[esi], al
  INC   esi
  LOOP  LOOP_OUT
  ret
```

1.3.3 BigIntMultiply 函数

本函数功能为模拟竖式进行大数相乘。首先，不考虑进位，使用双层循环进行每一位的乘法运算，结果数组第 i+j 位的值等于所有大数 A 的第 i 位和大数 B 的第 j 位的乘法结果之和。用 C 语言的形式即：

```
for (int i = 0; i < lenA; i++) {
    for (int j = 0; j < lenB; j++) {
        result[i + j] += numA[i] * numB[j];
    }
}
```

具体汇编形式实现细节如下，需要注意的细节已在注释中标明：

```
  MOV   ebx, -1
  ALOOP:
    INC   ebx
    CMP   ebx, lengthA
    JNB   ENDLOOPMUL        ;如果循环次数ebx大于等于A长度结束循环
    XOR   ecx, ecx
  BLOOP:
    MOV   eax, dword ptr numIntA[4 * ebx]
    MUL   numIntB[4 * ecx]    ;eax存储两个数相乘的结果
    ;可以用eax是因为此处最大的结果也就是9*9=81，不会超过8个字节，结果不会溢出到
    edx:eax
    MOV   esi, ecx
    ADD   esi, ebx        ;esi存储两个乘数数组下标之和
    ADD   resultInt[4 * esi], eax ;把两个位相乘的结果加到result的相应位上
    INC   ecx
    CMP   ecx, lengthB
    JNB   ALOOP         ;循环次数ecx大于等于B长度时跳出这位循环，进行下一位循环
    循环
    JMP   BLOOP
  ENDLOOPMUL:
    MOV   ecx, lengthA
    ADD   ecx, lengthB
    INC   ecx
    MOV   [lengthResult], ecx   ;结果长度最大为lengthA + lengthB + 1
```

然后作进位处理，核心思路用 C 语言的形式即：

```
for(int i=0;i<lengthResult;i++)
  {
```

```
    result[i+1] += result[i]/10;
    result[i] = result[i] % 10;
}
```

具体汇编形式实现细节如下，需要注意的细节已在注释中标明：

```
XOR   ebx, ebx         ;计算进位
CARRYLOOP:
  CMP   ebx, ecx
  JNB   ENDLOOPCARRY      ;循环次数ebx大于等于结果长度跳出循环
  MOV   eax, resultInt[4 * ebx]
  XOR   edx, edx
  DIV   divu
  ADD   resultInt[4 * ebx + 4], eax
  MOV   resultInt[4 * ebx], edx
  INC   ebx
  JMP   CARRYLOOP
ENDLOOPCARRY:
  MOV   ecx, lengthResult   ;除去结果数组首位（目前的末位）多余的0
MOVEZEROLOOP:
  CMP   dword ptr resultInt[4 * ecx], 0
  JNZ   ENDMUL          ;result的末位不为0
  DEC   ecx           ;每检测到一个0，实际长度减一
  JMP   MOVEZEROLOOP
ENDMUL:
  INC   ecx           ;实际长度：最大下标+1
  MOV   [lengthResult], ecx
```

1.4 运行效果

能够进行正确计算以及负号处理：

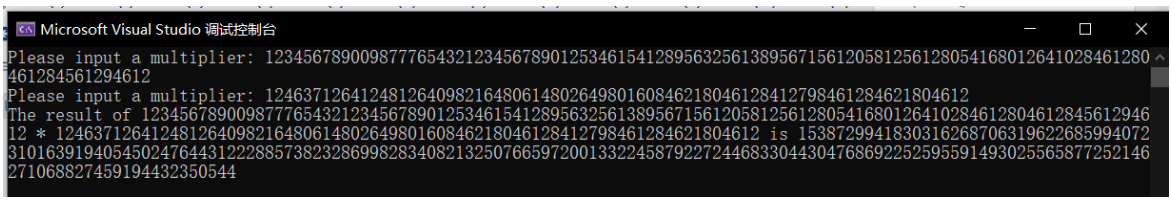

图 1-1 大数相乘运行效果

真，大数相乘：

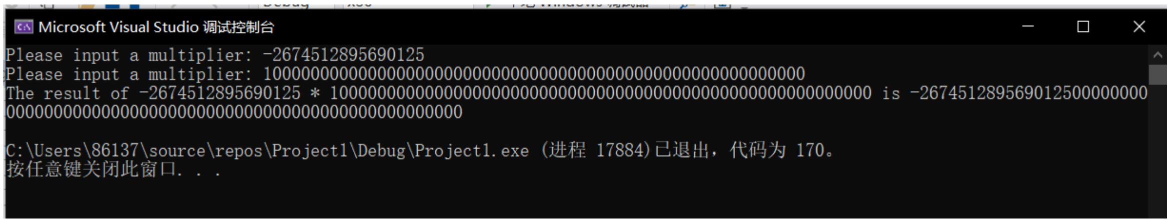

图 1-2 大数相乘运行效果，负号处理

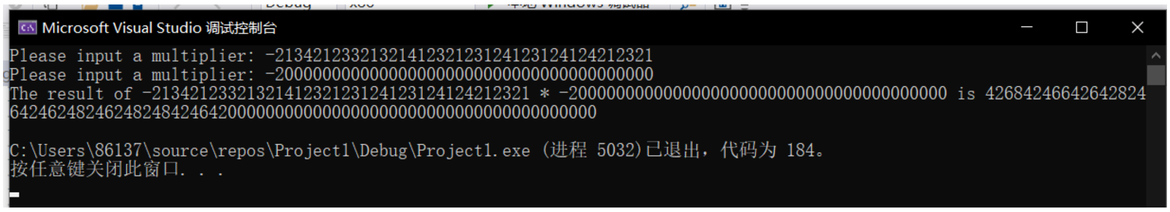

图 1-3 大数相乘运行效果

1.5 实验心得

本实验难度适中，使用的知识都是先前掌握的比较好的知识，先前使用 C 语言构建过相似的代码，因此算法思路也比较清晰。在编写汇编代码的时候能显著感受到汇编语言代码能力的提升，最大的两个收获是学会了需要注意寄存器访存细节以及函数传参调用形式。如果想进一步提升代码鲁棒性，应考虑对非数字字符（如逗号，问号，英语字符等符号）输入进行检测修正。

第 2 章 文件比较

2.1 实验目的

Windows 界面风格实现两个文本文件内容的比对。若两文件内容一样，输出相应提示；若两文件不一样，输出对应的行号。

本实验需要采用 Windows 界面风格编程，即需要学习使用 user32，gdi32，Comdlg32，kernel32 等相关库函数以注册、显示一个 Windows 窗口并完成窗口上相关的消息处理函数，即打开两个文件并进行按行比较。由于此实验与计算器实验都需要采用 Windows 界面风格编程，本人先完成了本实验以深入了解 Windows 界面风格编程，再此之上完成代码难度更高的计算器实验。

2.2 实验环境

编译器及开发工具：Visual Studio 2019，MASM32

处理器：Intel(R) Core(TM) i5-8265U CPU（80386 机器）

2.3 代码思路

本实验重点在于 Windows 界面风格编程，本部分内容将结合实际代码进行分析。

2.3.1 注册并打开 Windows 窗口

Windows 窗口使用 WNDCLASSEX 窗口类进行注册，WNDCLASSEX 结构体各成员定义了一个 Windows 窗口的主要属性，如光标、背景颜色、进程句柄、消息处理函数等，我们只需要定义成员属性并调用内置 RegisterClassEX 函数就可以完成窗口注册。本部分详细代码如下：

```
LOCAL _windowsClass:  WNDCLASSEX        ;窗口类 定义主要属性 如图标 光标 背景色
invoke  GetModuleHandle,NULL           ;获取当前exe句柄，存于eax
MOV   _processHandle,EAX

invoke  RtlZeroMemory,addr _windowsClass,sizeof _windowsClass  ;初始化结构体
invoke  LoadCursor,0,IDC_ARROW
MOV   _windowsClass.hCursor,EAX
PUSH  _processHandle
```

```
POP    _windowsClass.hInstance
MOV    _windowsClass.cbSize,sizeof WNDCLASSEX
MOV    _windowsClass.style,CS_HREDRAW or CS_VREDRAW
MOV    _windowsClass.lpfnWndProc,offset  _ProcWinMain  ;本句指定了该窗口的消息处理函数为_ProcWinMain，内置的消息传递函数为SengMessage(hWnd,uMsg,wParam,lParam)
MOV    _windowsClass.hbrBackground,COLOR_WINDOW+1
MOV    _windowsClass.lpszClassName,offset  _appClassName ;给窗口类一个名字

invoke  RegisterClassEx,addr  _windowsClass ;注册窗口类
```

注册好一个 Windows 窗口后，需要调用 CreateWindowEx 函数创建窗口并获得其句柄。CreateWindowEx 函数的第二个传参也可以是字符串'button'，从而为某窗口创建一个按钮。该函数的返回值为窗口句柄，存于 EAX 中。随后可以使用 ShowWindow 和 UpdateWindow 函数将窗口显示在桌面上。本部分具体代码如下：

```
invoke  CreateWindowEx,WS_EX_CLIENTEDGE,\   ;新建窗口，句柄放eax
    offset _appClassName,offset AppCaption,\ ;注册类  窗口名称/button（按钮）
    WS_OVERLAPPEDWINDOW,100,100,600,180,\ ;四个数字含义为窗口左上角xy坐标+宽+长
    NULL,NULL,_processHandle,NULL
MOV    _windowHandle,EAX
invoke  ShowWindow,_windowHandle,SW_SHOWNORMAL  ;显示窗口
invoke  UpdateWindow,_windowHandle        ;刷新窗口
```

窗口创建成功后，需要循环获取窗口中各种消息事件如鼠标点击、键盘输入等事件并进行处理。

Windows 窗口是用 MSG 结构体来传递消息的，调用 GetMessage 函数可以获取消息队列中的第一个消息，如果是退出消息则存于 EAX 的返回值为 0。

消息类型及相关参数存于 MSG 结构体中，可以用 DispatchMessage 函数找到该窗口的消息处理函数进行消息处理，本实验中为 _ProcWinMain 函数。消息处理循环的代码如下：

```
LOCAL _windowsMsg:  MSG             ;窗口类传递消息的结构体
APP_PROCESS_MESSAGE:                ;循环消息队列处理消息
invoke  GetMessage,addr _windowsMsg,NULL,0,0  ;从消息队列中取消息
CMP    EAX,0                ;退出消息则EAX=0
JE     APP_END
invoke  TranslateMessage,addr _windowsMsg   ;把基于键盘扫描码的按键信息转换成
```

```
    对应的ASCII码，如果消息不是通过键盘输入的这步无效
  invoke  DispatchMessage,addr _windowsMsg    ;找到该窗口程序的窗口过程，处理消
    息
  JMP   APP_PROCESS_MESSAGE
APP_END:
  ret
```

2.3.2 消息处理

消息处理函数的默认传参形式为 SendMessage(hWnd,uMsg,wParam,lParam)，hWnd 为窗口句柄，uMsg 为消息类型，如 WM_PAINT（绘制窗口）、WM_CLOSE（关闭窗口）、WM_COMMAND（点击事件）等。wParam 和 lParam 为可变参数，含义由消息类型决定。如在 WM_COMMAND 消息类型中，wParam 参数为按钮句柄。本函数主要结构如下，根据不同的消息类型做出不同函数回应。

```
  _ProcWinMain proc uses ebx edi esi,hWnd:HWND,uMsg:UINT,wParam:WPARAM,lParam:
    LPARAM
  LOCAL PaintInfo:  PAINTSTRUCT
LOCAL hDc:    HDC
.if   uMsg == WM_PAINT            ;uMsg消息类型
  invoke  BeginPaint,hWnd,addr PaintInfo    ;绘制窗口
  MOV   hDc,EAX
  MOV   screenHandle,EAX
  invoke  TextOut,hDc,250,15,addr TextNotChosen,5 ;在固定坐标处打印字符串，长度
    为5
  invoke  TextOut,hDc,250,55,addr TextNotChosen,5
  invoke  EndPaint,hWnd,addr PaintInfo
.elseif uMsg == WM_CLOSE              ;关闭窗口
  invoke  DestroyWindow,_windowHandle
  invoke  PostQuitMessage,NULL
.elseif uMsg == WM_CREATE           ;创建窗口，给3个按钮赋句柄
  invoke CreateWindowEx,NULL,offset Button,offset Button1Text,\
    WS_CHILD or WS_VISIBLE,10,10,200,30,\
    hWnd,1,_processHandle,NULL        ;1表示该按钮的句柄是1
  {...}
.elseif uMsg == WM_COMMAND            ;点击事件。若点击按钮，wParam是按钮句柄
  .if wParam == 1
    invoke  _OpenFile,1         ;自己编写的函数，功能是打开一个窗口选择文件
    {...}
  .elseif wParam == 2
```

```
    invoke  _OpenFile,2
    {...}
  .elseif wParam == 3
    invoke  CompareFile          ;自己编写的函数，功能是逐行比较两个文件并返回结果
    {...}
  .endif
.else                          ;其他消息类型，用默认方式处理
  invoke DefWindowProc,hWnd,uMsg,wParam,lParam
  ret
.endif
XOR   EAX,EAX
ret
_ProcWinMain endp
```

2.3.3 文件选择窗口

本函数主要是利用 commdlg.h 提供的 GetOpenFileName 系统函数选择一个本机上的文件，此函数的传参只有一个，即一个类型为 OPENFILENAME 的结构体，此结构体记录了要打开的文件名、打开文件的父进程句柄、文件打开方式等各种信息。本函数对此结构体进行初始化后，填充相关信息，通过调用函数的传参（1 或 2）判断此时用户在选择文件 1 还是文件 2，并将文件路径存在文件 1 路径名变量或文件 2 路径名变量中。若选择文件成功，弹出 MessageBox 提示用户已选择的文件名，并将文件路径打印到窗口中。本函数部分代码如下：

```
_OpenFile proc  flag:dword
LOCAL filehandle:OPENFILENAME          ;要打开的文件名

invoke  RtlZeroMemory,addr filehandle,sizeof filehandle

MOV   filehandle.lStructSize,sizeof filehandle;初始化文件句柄
PUSH  _windowHandle
POP   filehandle.hwndOwner
MOV   filehandle.lpstrFilter,offset MyFilter
MOV   filehandle.nMaxFile,MAX_PATH
MOV   filehandle.Flags,OFN_FILEMUSTEXIST OR OFN_PATHMUSTEXIST

MOV   EAX,flag                 ;通过flag标记打开的是文件1或2
CMP   EAX,1
```

```
  JNE   FILE_NOT_1
  MOV   filehandle.lpstrFile,offset FileName1
  JMP   MARK_FILE_DONE
  {...}

MARK_FILE_DONE:
  invoke  GetOpenFileName,addr filehandle     ;打开一个窗口选择文件
  {...}
  invoke  MessageBox,_windowHandle,offset FileName1,\
      offset FileChosenHint,MB_OK
  {...}
  ret
_OpenFile endp
```

2.3.4 文件比较

本部分使用 CreateFile 函数先打开两个文件，然后分别读取一行内容，调用 lstrcmp 函数比较他们是否相同，不同则记录行数。在此过程中，维护两个指针指示两个文件分别读到了第几行，若一个文件读完了（指针指向文件末尾），而另一个文件还没有读完，则可以直接记录不同行行数并跳出循环。

由于 ReadFile 函数只通过指定字符数量读取文件内容，无法一次性读取文件中一行。所以，编写了一个新函数 ReadOneLine 来读取文件中的一行。具体思路是，调用 ReadFile 函数一次只读一个字符，当读取字符为普通字符时，存入返回数组，当读取字符为换行符或为空字符时，退出循环。

部分重要的框架代码展示如下：

```
  CompareFile proc
  LOCAL filehandle1:  HANDLE          ;文件句柄
  LOCAL filehandle2:  HANDLE
  LOCAL ptr1:     DWORD         ;文件指针
  LOCAL ptr2:     DWORD
  LOCAL _line:      DWORD         ;行数
  LOCAL differenceline[1000]: BYTE

  MOV   _line,0                 ;打开两个文件并获得文件句柄
  MOV   DiffNum,0
  invoke  CreateFile,offset FileName1,GENERIC_READ, FILE_SHARE_READ, NULL,
    OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, NULL
```

```
  MOV   filehandle1,EAX
  {...}

READ_LINE_LOOP:
  INC   _line
  invoke  RtlZeroMemory,offset  Buffer1,sizeof  Buffer1
  invoke  ReadOneLine,filehandle1,offset  Buffer1 ;写了一个按行读取函数
  MOV   ptr1,EAX                ;长度存于EAX
  {...}
  ;文件1 2如果不为空，就用strcmp比较
  invoke  lstrcmp,offset  Buffer1,offset Buffer2
  CMP   EAX,0
  JE    READ_LINE_START
  JMP   DIFFERENCE_LINE_EXIST

COMPARE_END:
  invoke  CloseHandle,filehandle1         ;关闭两个文件句柄
  invoke  CloseHandle,filehandle2
  ret

DIFFERENCE_LINE_EXIST:                ;两行不同，打印行号
  invoke  sprintf,addr  differenceline,offset DifferenceHint,_line
  invoke  lstrcat,addr  Difference,addr differenceline
  INC   DiffNum
  JMP   READ_LINE_START
CompareFile endp
```

文件比较函数运行完毕后，在消息处理函数 _ProcWinMain 中判断两个文件是否存在不同行，并调用 MessageBox 进行相应的提示。

2.4 运行效果

图 2-1 窗口创建成功后, 文件比较窗口 UI

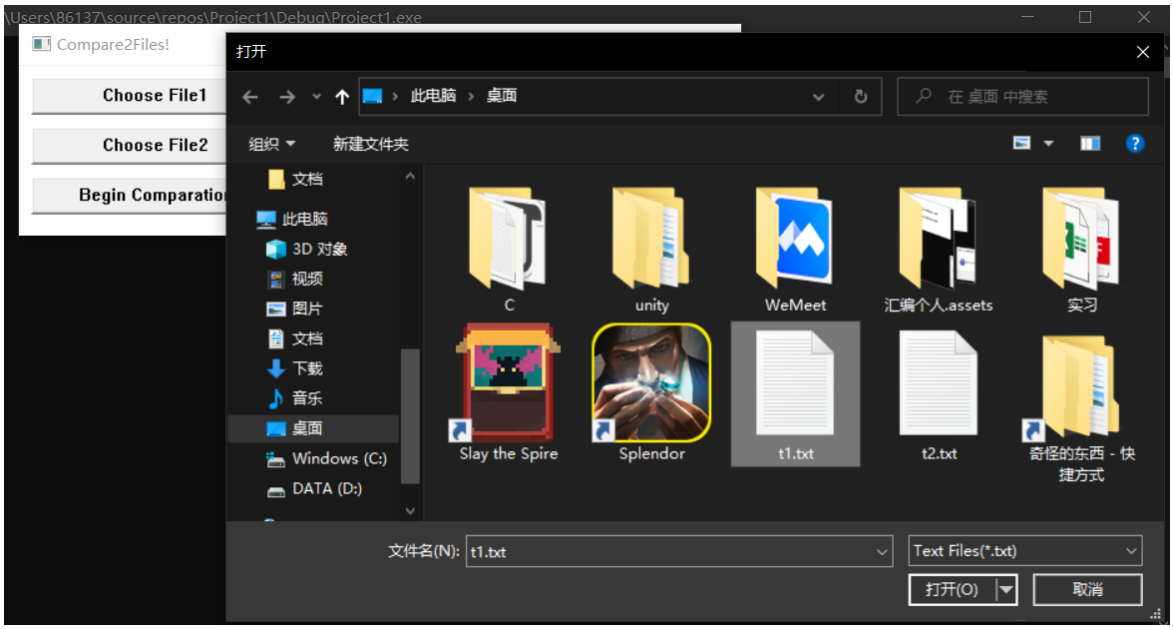

图 2-2 点击 Choose File1 按钮打开文件选择窗口

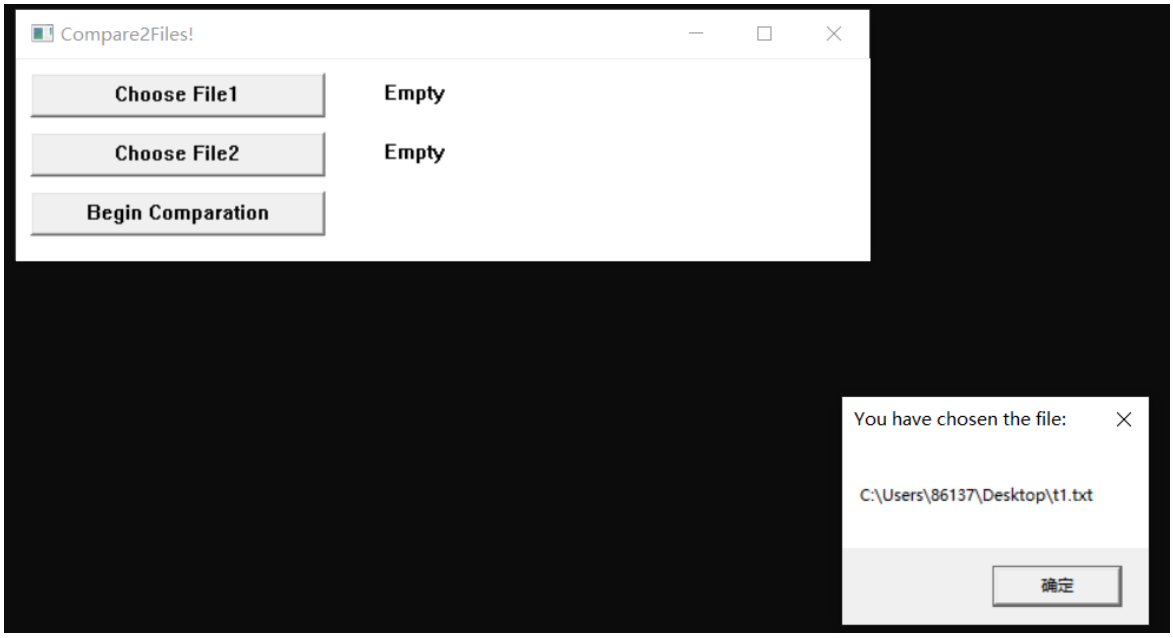

图 2-3 选择文件后，弹出消息框提示

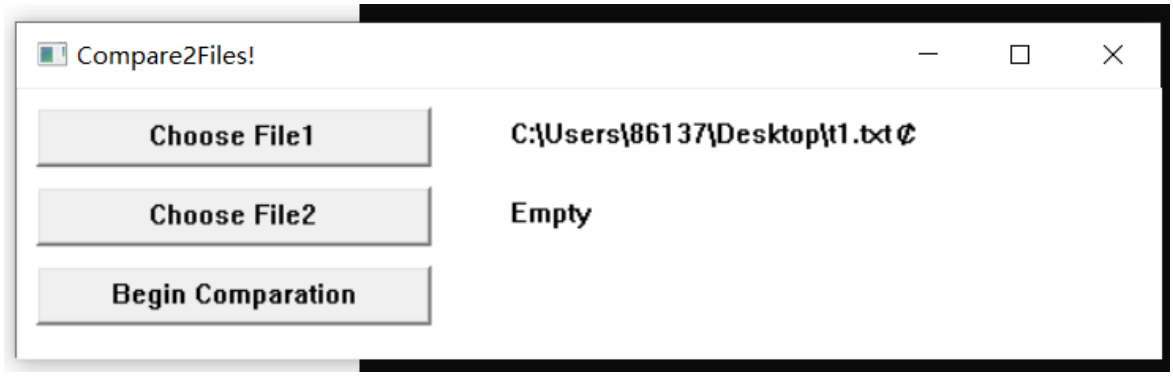

图 2-4 选择文件后，窗口文字显示

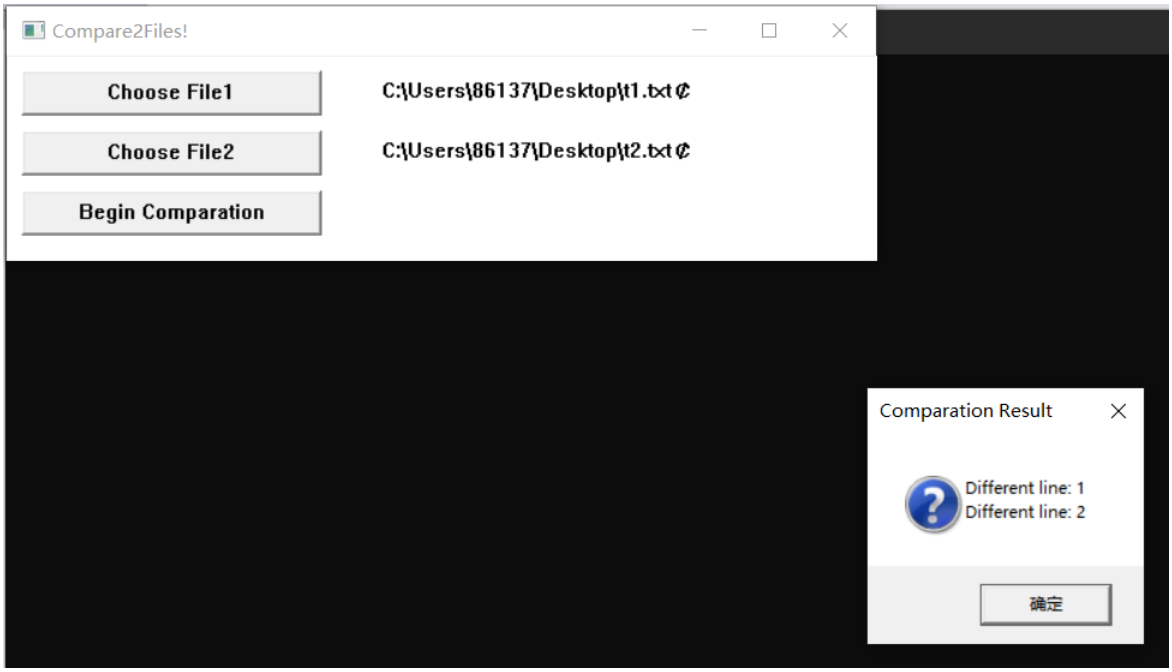

图 2-5 选择好两个文件后点击 Begin Comparation 按钮进行比较，弹出消息框提示文件不同行数

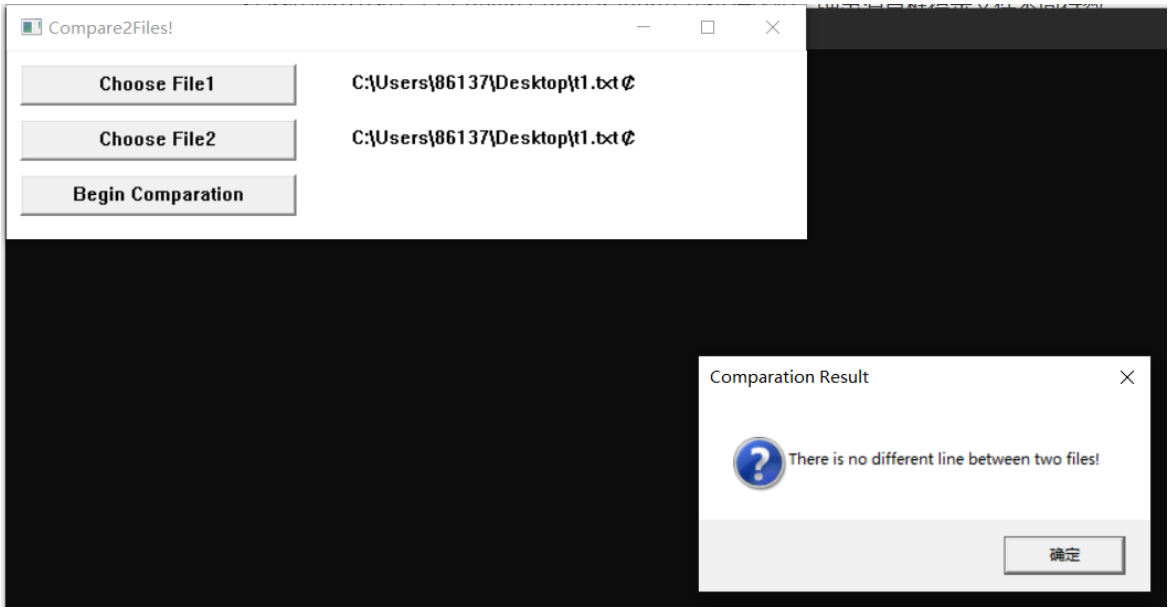

图 2-6 选择同一个文件或相同内容的文件进行比较，弹出消息框提示文件内容相同

2.5 实验心得

本实验主要难点还是在于相关库函数的不熟悉，在掌握之后其实编码还是比较简单的。本实验说是实现了文件的按行比较，其实由于消息显示设置采用了MessageBox 的形式，如果不一样的行太多后面的会显示不出来。而且这里的文件过滤 filter 限制了对.doc，.txt 这种文本格式的文档内容比对，即并不是什么文件都可以比较，并且由于文件编码格式的问题，如果 word 里有表格会出问题，且没有做报错信息提示。总之，做出一个 Windows 窗口并实现相应的功能还是让人很有成就感的。

第 3 章 计算器

3.1 实验目的

结合 Windows 界面编程和浮点数编程，实现完善的计算器功能，支持浮点运算和三角函数等功能。由于在文件比对实验中已经深入学习了 Windows 界面编程，此实验的难点在于熟练掌握浮点数运算指令，并按照运算符优先级编码实现计算器运算功能。

3.2 实验环境

编译器及开发工具：Visual Studio 2019，MASM32

处理器：Intel(R) Core(TM) i5-8265U CPU（80386 机器）

3.3 代码思路

3.3.1 计算机窗口 UI 设计

本部分的基本编码流程与第二章相似，即注册窗口-显示窗口-窗口消息循环处理，此处不再赘述。在消息处理函数中对不同类型的消息进行分类处理。由于此界面的按钮较多，本实验采用了界面常量的宏定义形式为每个按钮、文本框控件设置句柄，增强代码可读性，示例如下：

```
;界面常量
handleButton0    EQU    0
handleButton1    EQU    1
{...}
handleButtonEqu    EQU    10
handleButtonPlus   EQU    11
handleButtonSub    EQU    12
{...}
handleButtonSin    EQU    18
handleButtonCos    EQU    19
handleButtonTan    EQU    20
handleButtonLeft   EQU    21
handleButtonDot    EQU    22
handleTextExpression   EQU    23
handleTextResult     EQU    24
```

```
16
17 _ProcWinMain proc uses ebx edi esi,hWnd:HWND,uMsg:UINT,wParam:WPARAM,lParam:
       LPARAM
18 {...}
19   .elseif uMsg == WM_CREATE            ;创建窗口，按钮赋句柄
20     invoke CreateWindowEx,NULL,offset Button,offset button1Text,\
21       WS_CHILD or WS_VISIBLE,20,200,60,60,\
22       hWnd,handleButton1,_processHandle,NULL
23     {...}
24   .elseif uMsg == WM_COMMAND             ;点击事件 如果点击按钮，wParam是按钮句柄

25     .if   ((wParam >= handleButton0) && (wParam<= handleButton9))
26       invoke  pressNum,hWnd,wParam
27     .elseif ((wParam >= handleButtonEqu) && (wParam<= handleButtonRight))
28       invoke  pressOp,hWnd,wParam
29 {...}
30 _ProcWinMain endp
```

计算器界面上除了各种功能按键，还有两个文本框。上方的文本框是表达式文本框，显示目前计算的表达式。下方的文本框显示计算结果。界面展示如图 3-1。

图 3-1 计算器界面 UI

3.3.2 按键处理

本计算器的实现思路是，不管当前用户是否输入了浮点数，在程序内部都以浮点数的形式进行存储、计算。用户按下每个按钮的消息处理思路如下：

用户按下数字键 0-9 时，将当前域的数字长度 +1，利用浮点栈更新当前数字值并更新表达式文本框内容，但不作具体计算。

用户按下加减乘除号、等于号、右括号时，利用符号栈、当前数字栈、符号栈指针、当前数字栈指针进行相应的浮点数运算，并相应地改变栈中的内容和界面文本框中的内容。

用户按下左括号时，将左括号压入符号栈，并相应地更新表达式文本框内容，但不作具体计算。

用户按下 AC 键时，初始化各种全局变量、符号栈、数字栈，并将两个文本框的内容置为空。

用户按下 Sin、Cos、Tan 这三个三角函数键时，默认是对当前数字栈顶的第一个数字进行相应的三角函数操作，在浮点数栈中使用 fsin、fcos 分别进行对应计算后，将结果更新到界面文本框中。

用户按下 +/-键时，默认是将当前数字栈顶的第一个数字取反，程序将浮点数栈 s[0] 取反以及全局变量 isNeg 符号位取反，将结果更新到界面文本框中。

用户按下小数点键时，由于程序内部本身存储的就是浮点数，只需要更新当前数字的阶并相应地将结果更新到界面表达式文本框中即可。

详细的代码实现细节在源码中已做好注释，由于很多操作都是重复性较高、难度不大、比较繁琐，此处就不再赘述。浮点数编程主要使用到了 fld、fadd、fsub、fmul、fdiv 等运算指令，以及 fst、fstp 这两个指令将浮点栈顶的内容复制到一个内存变量中。更新界面文本框主要使用了 strcat 函数以及 SetDlgItemText 函数将对应的内容添加到字符串末尾，并根据文本框句柄更新相应文本框控件。

3.4 运行效果

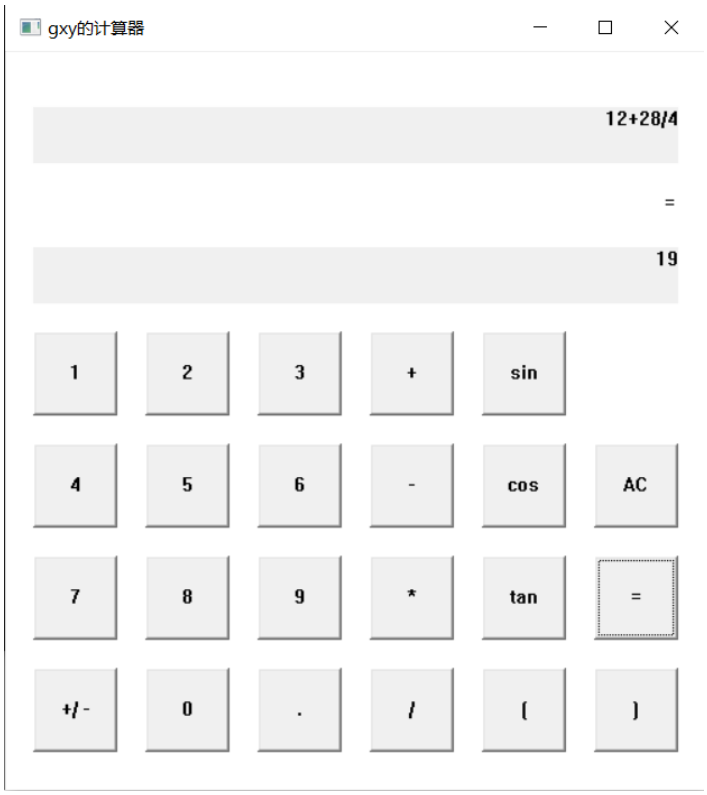

图 3-2 计算器运行效果

图 3-3 计算器运行效果

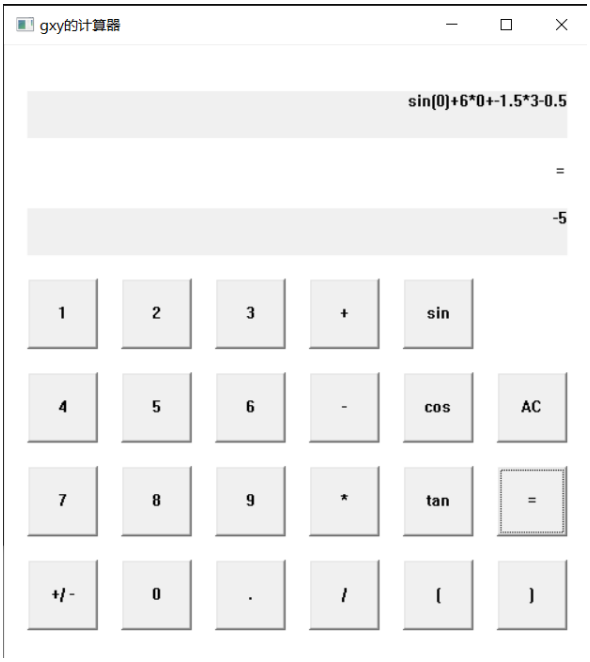

图 3-4 计算器运行效果

3.5 实验心得

这个实验真的做了好久，最后其实还有挺多 bug 的，比如 0 作分母没有报错，结果会溢出成 0.0e46 等等等等。很多计算操作都是重复性高、简单但琐碎的工作，代码难度看似不算大，但是一不留神就容易出错。例如让我印象深刻的，当前没有输入浮点数的情况下在用户按下数字键 0-9 时，程序要利用浮点栈将其转换为浮点数这里由于不熟悉浮点栈的工作机制 debug 了很久。总之最后完成了一个简易计算器成就感非常非常高。本次实验整体上来说收获颇丰。