

# 北京理工大学

## 汇编语言与接口技术

### 实验报告

Lab Report

学    院：	计算机学院
专    业：	计算机科学与技术
学生姓名：	卜梦煜
学    号：	1120192419
指导教师：	张全新

2022 年 6 月 8 日

## 原创性声明

本人郑重声明：所呈交的毕业设计（论文），是本人在指导老师的指导下独立进行研究所取得的成果。除文中已经注明引用的内容外，本文不包含任何其他个人或集体已经发表或撰写过的研究成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。

特此申明。

本人签名：

日期：

年

月

日

## 关于使用授权的声明

本人完全了解北京理工大学有关保管、使用毕业设计（论文）的规定，其中包括：①学校有权保管、并向有关部门送交本毕业设计（论文）的原件与复印件；②学校可以采用影印、缩印或其它复制手段复制并保存本毕业设计（论文）；③学校可允许本毕业设计（论文）被查阅或借阅；④学校可以学术交流为目的，复制赠送和交换本毕业设计（论文）；⑤学校可以公布本毕业设计（论文）的全部或部分内容。

本人签名：

日期：

年

月

日

指导老师签名：

日期：

年

月

日

## 实验报告

### 摘 要

本文是计算机学院 2019 级汇编语言与接口技术个人实验的实验报告，实验内容是“大数乘法”、“Windows 风格的计算器”、“Windows 风格的文本比对”。实验主要利用 Visual Studio 2022 集成开发环境和 MASM32 工具包进行开发，采用纯汇编的代码风格实现。实验结果表明，三个实验要求的功能全部实现。

**关键词：**大数乘法；计算器；文本比对

## Lab Report

### Abstract

This paper is the experimental report of the 2019 personal experiment of assembly language and interface technology in the school of computer science. The experimental contents are "large number multiplication", "windows style calculator" and "windows style text comparison". The experiment is mainly developed by using Visual Studio 2022 integrated development environment and masm32 toolkit, and is implemented in the style of pure assembly code. The experimental results show that all the functions required by the three experiments are realized.

**Key Words: Large number multiplication; Calculator; Text comparison**

## 目 录

摘 要 .....	I
Abstract .....	II
第 1 章 实验基本信息 .....	1
1.1 实验名称 .....	1
1.2 实验目的 .....	1
1.3 实验内容 .....	1
1.4 实验环境 .....	1
第 2 章 大数乘法 .....	1
2.1 代码设计思路 .....	1
2.1.1 数据读入.....	1
2.1.2 数据预处理.....	2
2.1.3 模拟乘法.....	3
2.1.4 结果输出.....	5
第 3 章 汇编计算器 .....	5
3.1 Windows 界面设计.....	5
3.2 消息响应设计 .....	7
3.2.1 AddString 方法.....	8
3.2.2 Clear 方法.....	9
3.2.3 BackSpace 方法.....	9
3.2.4 Calculate 方法.....	10
3.3 控制逻辑.....	11
第 4 章 文本比对 .....	11
4.1 Windows 界面设计.....	11
4.2 文本对比 .....	11
4.3 结果输出 .....	13
第 5 章 实验结果与分析 .....	14
5.1 大数乘法 .....	14
5.2 汇编计算器 .....	15
5.3 文本比对 .....	18
第 6 章 心得体会 .....	20
结 论 .....	21

## 第 1 章 实验基本信息

### 1.1 实验名称

大数乘法、Windows风格的计算器、Windows风格的文本比对

### 1.2 实验目的

(1) 掌握汇编语言常用指令、控制结构、模块通信等基础知识，能够设计与开发具备简单功能的汇编程序。

(2) 熟悉Windows提供的系统API，能够开发简单的具有Windows界面风格的汇编应用程序。

### 1.3 实验内容

(1) 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。

(2) 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。

(3) 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。

### 1.4 实验环境

Visual Studio 2022、masm32

## 第 2 章 大数乘法

### 2.1 代码设计思路

#### 2.1.1 数据读入

由于题目要求是支持超过100位的大数乘法，因此无法使用dword存储，需要将键盘输入的数字转化为字符串，按位做读入、计算、输出等操作。代码如图所示。

```
; 输入数字A、B，存入数组
invoke printf, offset inputMsg
invoke scanf, addr inputFmt, addr numCharA
invoke scanf, addr inputFmt, addr numCharB
```

图 2-1 数据读入

### 2.1.2 数据预处理

对读入的字符串，在运算前需要进行预处理，包括长度计算、逆序处理、以及将字符串每一位转化为对应数字，避免出现进位难以处理、运算代码复杂等问题。

长度计算调用C语言的strlen函数，对于负数只记录数字部分长度。代码如图所示。

```
getLength proc far c numChar: ptr byte, len: ptr dword

; 数组长度写入eax
mov esi, numChar
movzx eax, byte ptr [esi]
.if eax == 2DH
    xor sign, 1
    invoke strlen, numChar
    dec eax
.else
    invoke strlen, numChar
.endif

; eax写入len地址
mov ecx, len
mov [ecx], eax

ret

getLength endp
```

图 2-2 计算长度

逆序处理与数字转化使用栈结构，先将数字从高位到低位依次入栈，再依次出栈并转化为对应数字，存入数字数组中，循环次数为数组长度。代码如图所示。

```
reverseStrToInt proc far c numChar: ptr byte, numInt: ptr dword, len: dword

    mov esi, numChar
    movzx eax, byte ptr [esi]

    ; 负数从数字开始
    .if eax == 2DH
        inc esi
    .endif

    ; 设置循环次数
    mov ecx, len

    ; 数字逐个压入栈
11:
    movzx eax, byte ptr[esi]
    sub eax, 30H
    push eax
    inc esi
    loop 11

    ; 重置循环次数
    mov ecx, len
    mov esi, numInt
12:
    pop eax
    mov dword ptr[esi], eax
    add esi, 4
    loop 12

    ret

reverseStrToInt endp
```

图 2-3 数据预处理

### 2.1.3 模拟乘法

模拟乘法过程即为手动进行乘法运算的过程，被乘数第*i*位与乘数第*j*位相乘，结果存入第*i+j*位。从低位向高位模拟进位操作，完成后更新结果位数。模拟乘法部分核心代码如下。



## 汇编语言与接口技术实验报告

---

```
; 外层循环变量ebx, 遍历A每一位
mov ebx, -1
loopX:
    inc ebx
    cmp ebx, lengthA
    jnb endLoopX

; 内层循环变量ecx, 遍历B每一位
xor ecx, ecx
loopY:
    ; A[ebx] * B[ecx]
    mov eax, dword ptr numIntA[4 * ebx]
    mul numIntB[4 * ecx]    ; 结果存在edx:eax中

    ; 当前结果应保存在哪一位
    mov esi, ebx
    add esi, ecx

    ; 保存结果
    add numIntResult[4 * esi], eax

    inc ecx
    cmp ecx, lengthB
    jnb loopX
    jmp loopY
```

图 2-4 模拟乘法计算

```
; 进位
endLoopX:
    ; 不会超过lengthA+lengthB位, lengthResult=lengthA+lengthB
    mov ecx, lengthA
    add ecx, lengthB
    mov esi, offset lengthResult
    mov [esi], ecx

    xor ebx, ebx
    forward:
        cmp ebx, ecx
        jnb endForward

        xor edx, edx
        mov eax, numIntResult[4 * ebx]
        div base
        ; 商eax, 余数edx
        add numIntResult[4 * ebx + 4], eax
        mov numIntResult[4 * ebx], edx

        inc ebx
        jmp forward
```

图 2-5 模拟乘法进位

## 2.1.4 结果输出

由于计算结果是逆序的数字，结果输出时应先转化成正序的字符串数组才能输出，数字数组与字符串数组的转化与前面类似，利用了栈结构。结果输出用到了C语言的printf函数，按照运算结果是否为负数分类输出。代码如下所示。

```
; 打印结果
.if sign == 1
    invoke printf, offset outputMsgNegative, addr negativeSign, addr numCharResult
.else
    invoke printf, offset outputMsgPositive, addr numCharResult
.endif
```

图 2-6 打印结果

## 第 3 章 汇编计算器

### 3.1 Windows 界面设计

绘制界面调用了Windows32提供的API。首先需要初始化一个主窗口WinMain，包括初始化窗口参数、注册窗口、创建窗口、显示窗口、更新窗口、窗口消息处理。之后对编写窗口消息的响应方法WndProc，对各种消息进行响应，执行不同动作。对Windows32提供的部分核心API及数据类型的查询信息如图所示。

windowsAPI	类型	功能
GetModuleHandle	函数	获取程序句柄
GetCommandLine	函数	获取程序的命令行参数然后以参数的形式传递给WinMain函数
RegisterClassEx	函数	注册窗体
CreateWindowEx	函数	创建窗体
HINSTANCE	变量	程序句柄
LPSTR	变量	运行时参数，命令行内容
WNDCLASSEX	变量	窗口类，描述、编辑、播报窗口
MSG	变量	获取和设置消息的id号
HWND	变量	检索窗口的窗口句柄

**int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nShowCmd)**

- hinstance - 该参数为Windows 为应用程序生成的实例句柄。实例是用来跟踪资源的指针
- hprevinstance 当前已不再使用该参数，之前用来跟踪应用程序的前一个实例，即程序的父亲的程序实例
- lpcmdline - 一个以NULL结尾的字符串，类似于标准main(int argc, char\*\* argv)中的命令行参数，但没有单独的argc来指示命令行参数数量
- ncmdshow—— 最后一个参数是启动期间传递给应用程序的一个整数，指出如何打开程序的窗口。事实上，基本没用，但ncmdshow的值一般会用于ShowWindows()中使用，用来设置新建子窗口的形式

# 汇编语言与接口技术实验报告

图 3-1 Windows 界面 API 理解 1

**LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)**

窗口过程函数决定了当一个窗口从外界接收到不同的信息时，所采取的不同反应，即主要用于处理发送给窗口的信息

- hWnd是要处理窗口的句柄
- message是消息ID，代表了不同的消息类型
  - WM\_DESTROY：窗体摧毁消息
  - WM\_KEYDOWN：键盘消息
  - WM\_PAINT：客户区重绘消息 BeginPaint
- wParam和lParam代表了消息的附加信息，附加信息会随着消息类型的不同而不同
  - wParam的值为按下按键的虚拟键码
  - lParam则存储按键的相关状态信息
  - 当鼠标消息发出时，wParam值为鼠标按键的信息，而lParam则存储鼠标的坐标，高字节代表y坐标，低字节代表x坐标，g\_y = HIWORD(lParam), g\_x = LOWORD(lParam)

图 3-2 Windows 界面 API 理解 2

基于以上资料，设计主窗口代码如下图所示。

```
WinMain proc hInstance:HINSTANCE, hPrevInstance:HINSTANCE, lpCmdLine:LPSTR, nCmdShow:DWORD
LOCAL   wcex:WNDCLASSEX
LOCAL   msg:MSG

mov     wcex.cbSize, sizeof WNDCLASSEX
mov     wcex.style, CS_HREDRAW or CS_VREDRAW
mov     wcex.lpfnWndProc, offset WndProc
mov     wcex.cbClsExtra, 0
mov     wcex.cbWndExtra, 0
push    hInstance
pop     wcex.hInstance
invoke  LoadIcon, hInstance, IDI_APPLICATION
mov     wcex.hIcon, eax
mov     wcex.hIconSm, eax
invoke  LoadCursor, NULL, IDC_ARROW
mov     wcex.hCursor, eax
mov     wcex.hbrBackground, COLOR_WINDOW+1
mov     wcex.lpszMenuName, NULL
mov     wcex.lpszClassName, offset szWindowClass

invoke  RegisterClassEx, addr wcex           ;注册窗口
invoke  CreateWindowEx, NULL, \             ;创建窗口
        addr szWindowClass, \
        addr szWindowClass, \
        WS_CAPTION or WS_SYSMENU or WS_MINIMIZEBOX, \
        CW_USEDEFAULT, \
        CW_USEDEFAULT, \
        242, \
        420, \
        NULL, \
        NULL, \
        hInstance, \
        NULL
mov     hWnd, eax

invoke  ShowWindow, hWnd, nCmdShow           ;显示窗口
invoke  UpdateWindow, hWnd                  ;更新窗口
.while TRUE                                 ;处理消息
    invoke GetMessage, addr msg, NULL, 0, 0 ;获取消息
    .if eax == 0
        .break
    .endif
    invoke TranslateMessage, addr msg        ;转换消息
    invoke DispatchMessage, addr msg        ;分发消息
.endif

mov     eax, msg.wParam
ret
WinMain endp
```

图 3-3 主界面函数 WinMain

基于以上资料，编写窗口消息处理函数。需要处理的消息类型包括WM\_DESTROY、WM\_PAINT、WM\_CREATE、WM\_COMMAND，分别对应窗口关闭、窗口绘制、窗口元素创建、窗口消息响应。

设计的计算器界面参考Windows系统自带的计算器，包括运算符部分和显示部分，如图所示。



图 3-4 计算器界面

### 3.2 消息响应设计

对用户发出的窗口消息需按类响应，具体方法为，在创建窗口元素后获取窗口元素的句柄并保存，并将句柄与对应的处理函数绑定。当用户点击时，绑定消息处理函数的按钮会触发响应的方法，从而实现与用户的交互。部分消息绑定设计如图所示。

按钮设计：

## 汇编语言与接口技术实验报告

```
invoke CreateWindowEx, NULL, addr szStatic, addr szInitExpr, WS_VISIBLE or WS_CHILD or WS_BORDER or ES_MULTILINE or ES_READONLY, \
5, 5, 215, 40, hWin, NULL, NULL, NULL
mov     hwndStatic, eax
invoke CreateWindowEx, NULL, addr szStatic, addr szInitResult, WS_VISIBLE or WS_CHILD or WS_BORDER or ES_MULTILINE or ES_READONLY, \
5, 55, 215, 40, hWin, NULL, NULL, NULL
mov     hwndResult, eax
mov     esi, 0
invoke CreateWindowEx, NULL, addr szButton, addr szDoubleZero, WS_TABSTOP or WS_VISIBLE or WS_CHILD or BS_DEFPUSHBUTTON, \
5, 335, 50, 40, hWin, NULL, NULL, NULL
mov     hwndButton[esi*4], eax
inc     esi
invoke CreateWindowEx, NULL, addr szButton, addr szZero, WS_TABSTOP or WS_VISIBLE or WS_CHILD or BS_DEFPUSHBUTTON, \
60, 335, 50, 40, hWin, NULL, NULL, NULL
mov     hwndButton[esi*4], eax
inc     esi
invoke CreateWindowEx, NULL, addr szButton, addr szPoint, WS_TABSTOP or WS_VISIBLE or WS_CHILD or BS_DEFPUSHBUTTON, \
115, 335, 50, 40, hWin, NULL, NULL, NULL
mov     hwndButton[esi*4], eax
inc     esi
invoke CreateWindowEx, NULL, addr szButton, addr szEqual, WS_TABSTOP or WS_VISIBLE or WS_CHILD or BS_DEFPUSHBUTTON, \
170, 335, 50, 40, hWin, NULL, NULL, NULL
mov     hwndButton[esi*4], eax
inc     esi
```

图 3-5 按钮设计

绑定消息处理函数：

```
mov     eax, lParam
.if eax == hwndButton[0*4]           ;00
    invoke AddString, addr szDoubleZero
.elseif eax == hwndButton[1*4]       ;0
    invoke AddString, addr szZero
.elseif eax == hwndButton[2*4]       ;.
    invoke AddString, addr szPoint
.elseif eax == hwndButton[3*4]       ;=
    invoke Calculate
```

图 3-6 绑定消息函数

消息处理函数分四类：处理数字及非等号运算符的AddString方法，只需要将按钮对应的字符添加到表达式字符串和运算式字符串中储存起来，并将表达式显示到界面文本框中；处理等号的Calculate方法，对运算式进行计算，并将结果保存；处理CE的方法，将当前表达式、运算式及显示的表达式清空；处理BS的方法，将当前表达式回退一个运算符。这里为方便计算处理，将sin、cos、tan在运算式中映射成#、\$、&符号。

### 3.2.1 AddString 方法

AddString方法在用户点击数字及非等号运算符时调用，对点击的按钮，将对应按键值添加到表达式字符串中储存，将三角函数的按键值转化为对应助记符存储在运算式字符串中，并将当前完成表达式输出到界面中。方法中调用了C语言的strcpy、strcat函数。方法设计如图所示。

```
AddString proc string:ptr byte
    invoke strcat, addr expr, string          ;当前输入的表达式
    invoke SetWindowTextA, hwndStatic, addr expr
    invoke SetWindowTextA, hwndResult, addr resultStr

    mov     ebx, string
    movzx   eax, byte ptr[ebx]

    .if eax == "s"                          ;当前按下的按键值，将sin、cos、tan替换为对应助记符
        invoke strcpy, addr strTemp, addr szSinCac1
    .elseif eax == "c"
        invoke strcpy, addr strTemp, addr szCosCac1
    .elseif eax == "t"
        invoke strcpy, addr strTemp, addr szTanCac1
    .else
        invoke strcpy, addr strTemp, string
    .endif

    invoke strcat, addr exprToCalc, addr strTemp

    ret
AddString endp
```

图 3-7 AddString 方法

### 3.2.2 Clear 方法

Clear方法响应CE按钮，用于清空，包括表达式清空、运算式清空，并实时显示在用户界面上。方法设计如图所示。

```
Clear proc
    invoke memset, addr expr, 0, sizeof expr
    invoke memset, addr exprToCalc, 0, sizeof exprToCalc
    invoke SetWindowTextA, hwndStatic, addr expr
    invoke SetWindowTextA, hwndResult, addr resultStr
    ret
Clear endp
```

图 3-8 Clear 方法

### 3.2.3 BackSpace 方法

BackSpace方法响应BS按钮，用于回退一个运算符。对运算式，只需删除末尾运算符即可。对表达式需判断类型，对非三角函数，只需删除末尾字符即可，对三角函数，需删除三个字符。方法设计如图所示。

```

BackSpace proc
    invoke  strlen, addr exprToCalc
    mov     ebx, eax
    invoke  strlen, addr expr

    .if ebx == 0
        ret
    .endif

    dec     ebx
    dec     eax
    mov     exprToCalc[ebx], 0
    mov     expr[eax], 0
    .if exprToCalc[ebx] == "#"
        mov     expr[eax-1], 0
        mov     expr[eax-2], 0
    .elseif exprToCalc[ebx] == "$"
        mov     expr[eax-1], 0
        mov     expr[eax-2], 0
    .elseif exprToCalc[ebx] == "&"
        mov     expr[eax-1], 0
        mov     expr[eax-2], 0
    .endif

    invoke  SetWindowTextA, hwndStatic, addr expr
    invoke  SetWindowTextA, hwndResult, addr resultStr
    ret
BackSpace endp

```

图 3-9 BackSpace 方法

## 3.2.4 Calculate 方法

Calculate方法响应等号按钮，用于进行算术运算。对运算式，每次取出一个字符。若为数字或小数点则循环取出至数字位，小数点前的数字每次按“乘10加数字”的形式储存，小数点后的数字按“除10加数字”的形式储存。若为运算符则判断是否为三角运算或负数表示，是则压入符号栈，否则判断是括号还是+\*/运算，根据当前符号优先级与符号栈栈顶符号优先级判断当前符号的操作，若当前符号优先级高则压入符号栈，若当前符号优先级低则取出栈顶符号开始运算，直到栈顶符号优先级不高于当前符号优先级。

三角函数的运算采用fsin、fcos、ftan默认的弧度制。

计算中需对表达式错误的情况做处理，包括除负数、不合理的运算逻辑、不合理的数字等等。

### 3.3 控制逻辑

该应用程序控制逻辑如下。用户点击非等号运算符按钮时，当前表达式实时显示在窗口上；用户点击CE按钮时，表达式清空；用户点击BS按钮时，表达式回退一个运算符；用户点击等号时，计算表达式并将答案显示在界面上。

## 第4章 文本比对

### 4.1 Windows 界面设计

文本比对程序的界面设计与计算器相同，包括编写WinMain窗口函数和WndProc消息处理函数。因此设计逻辑不再赘述。

设计的界面包括文件路径输入部分和文本比对按钮，如图所示。



图 4-1 文本比对界面

### 4.2 文本对比

对“文本比对”按钮绑定对应方法。首先读出文件路径，调用C语言的fopen函数打开文件并保存文件指针。代码设计如图所示。



```
; 取出输入的文件路径
invoke GetDlgItemText, hWnd, 2, offset filePath1, 1000
invoke GetDlgItemText, hWnd, 3, offset filePath2, 1000

; 打开文件，错误处理，存储文件指针
invoke fopen, offset filePath1, offset mode
.if eax == 0
    invoke MessageBoxA, hWnd, offset fileNotExist, offset alert, MB_OK
    ret
.endif
mov file1, eax

invoke fopen, offset filePath2, offset mode
.if eax == 0
    invoke MessageBoxA, hWnd, offset fileNotExist, offset alert, MB_OK
    ret
.endif
mov file2, eax
```

图 4-2 打开文件

每次比对一行文本内容。开始前先调用C语言memset函数将临时变量buffer清空，然后调用C语言的fgets函数按行读入两个文本的同一行，并调用strcmp函数对比文本内容是否一致，若不一致则记录不一致位置的行号，并调用strcat函数将不一致行保存起来。循环至两个文本都读到文件尾停止。代码设计如图所示。

11:

```
; buffer清空
invoke memset, addr buffer1, 0, 1000
invoke memset, addr buffer2, 0, 1000

; 文本读入第line行, fgets结果返回eax, 若为0则读到文件尾
mov ebx, line
inc ebx
mov line, ebx

invoke fgets, offset buffer1, 1000, file1
push eax
invoke fgets, offset buffer2, 1000, file2
push eax

; 文本比对
invoke strcmp, offset buffer1, offset buffer2

; 文本不同需记录文本不同的行号
.if eax != 0
    mov match, 0
    invoke strcat, offset notMatchMsg, offset notMatchLine
    invoke _itoa, line, offset lineToStr, 10
    invoke strcat, offset notMatchMsg, offset lineToStr
.endif

; 有一个文件未到文件尾则继续比对
pop eax
cmp eax, 0
jnz 11
pop eax
cmp eax, 0
jnz 11
```

图 4-3 逐行文本比对

### 4.3 结果输出

若不存在文本不同的情况则输出“文本内容相同”的提示，否则输出文本内容不同的行号。文本内容不同的信息已经储存在字符串变量中，可直接打印。代码设计如图所示。

```
.if match == 1
    invoke MessageBoxA, hWnd, offset matched, offset note, MB_OK
.else
    invoke MessageBoxA, hWnd, offset notMatchMsg, offset note, MB_OK
.endif
```

图 4-4 输出结果

## 第 5 章 实验结果与分析

### 5.1 大数乘法

根据实验要求，大数乘法的测试点主要包括正负数的大数乘法能否正确计算。  
两个正数相乘，运算结果正确，结果如图。

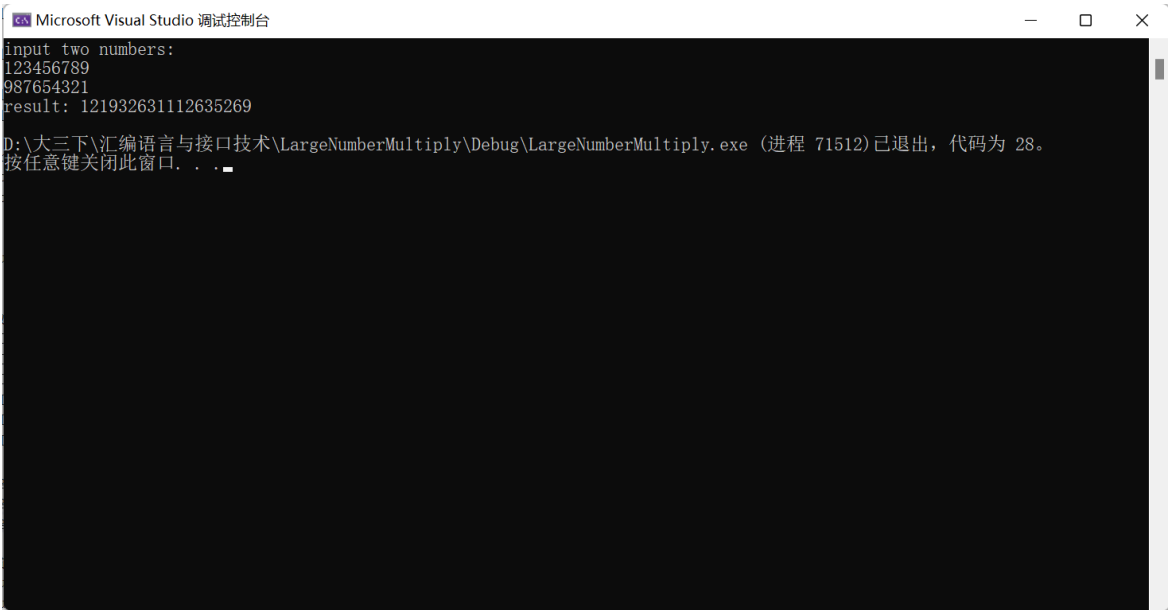


图 5-1 大数乘法测试 1

正数负数相乘，运算结果正确，结果如图。

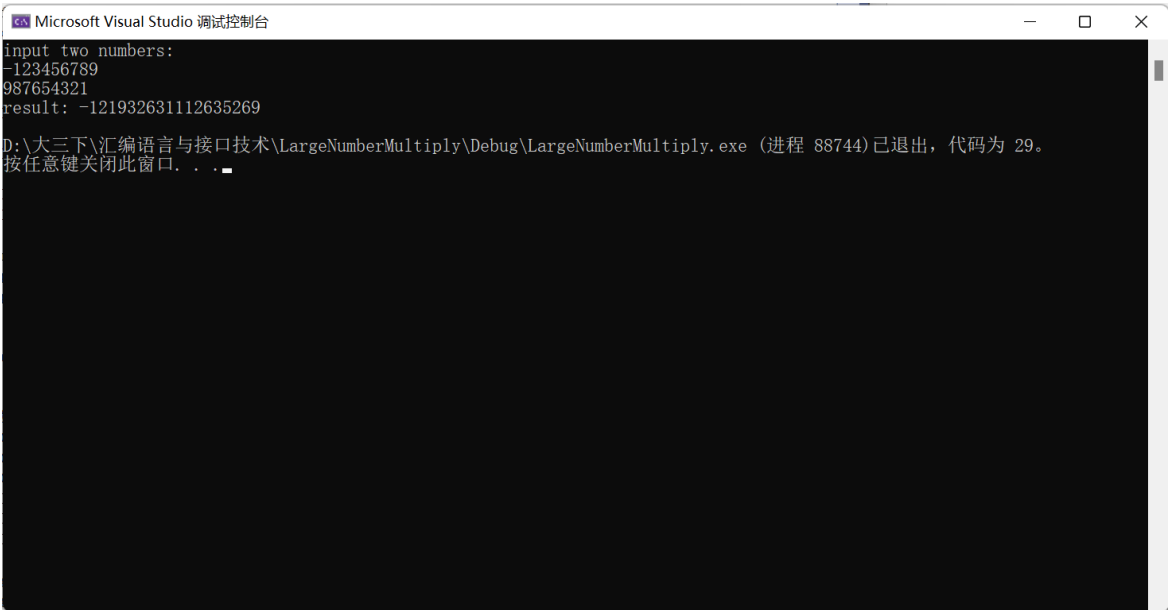
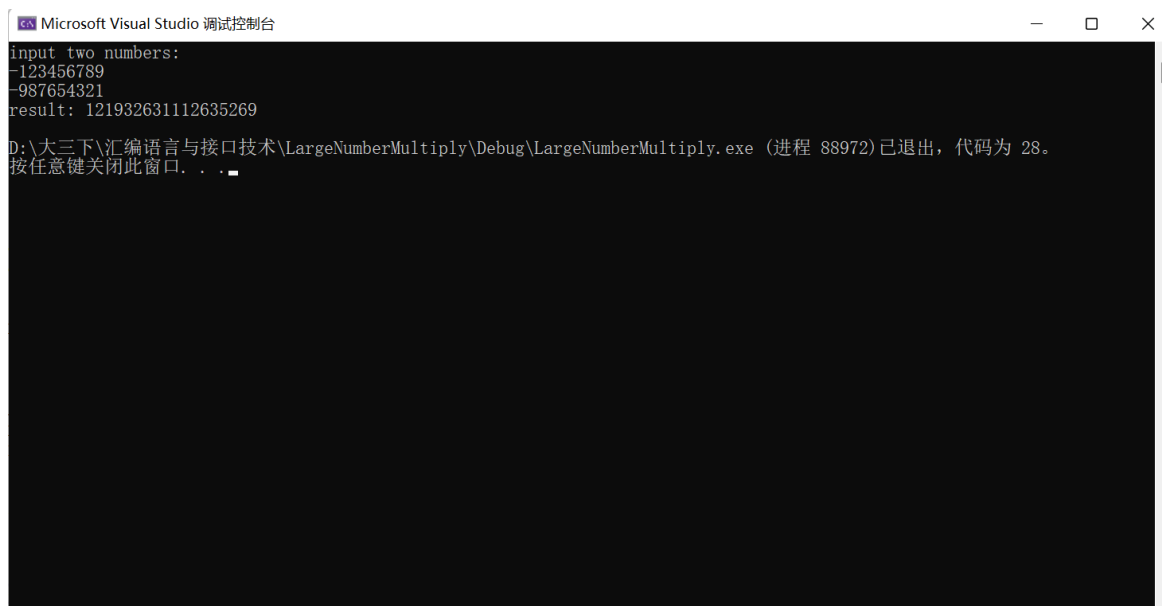


图 5-2 大数乘法测试 2

两个负数相乘，运算结果正确，结果如图。



```
Microsoft Visual Studio 调试控制台
input two numbers:
-123456789
-987654321
result: 121932631112635269
D:\大三下\汇编语言与接口技术\LargeNumberMultiply\Debug\LargeNumberMultiply.exe (进程 88972) 已退出，代码为 28。
按任意键关闭此窗口。 . . .
```

图 5-3 大数乘法测试 3

测试结果表明，实验要求中的功能已全部实现。

## 5.2 汇编计算器

根据实验要求，汇编计算器的测试点主要包括操作是否符合逻辑、能否正确计算表达式、四种功能模块是否正常。

构造多种复合运算的表达式，对正确的表达式运算结果正确，对错误的表达式能提示错误信息。结果如图。



图 5-4 计算器正确表达式测试 1



图 5-5 计算器正确表达式测试 2



图 5-6 计算器错误表达式测试 1



图 5-7 计算器正确表达式测试 1

测试结果表明，实验要求中的功能已全部实现。

5.3 文本比对

根据实验要求，文本比对的测试点主要包括打开文件、文本相同和文本不同时的输出。

输入不存在的文件路径，能正确提示，结果如图。



图 5-8 文本比对测试 1

两文本内容相同时，能给出正确的提示，结果如图。

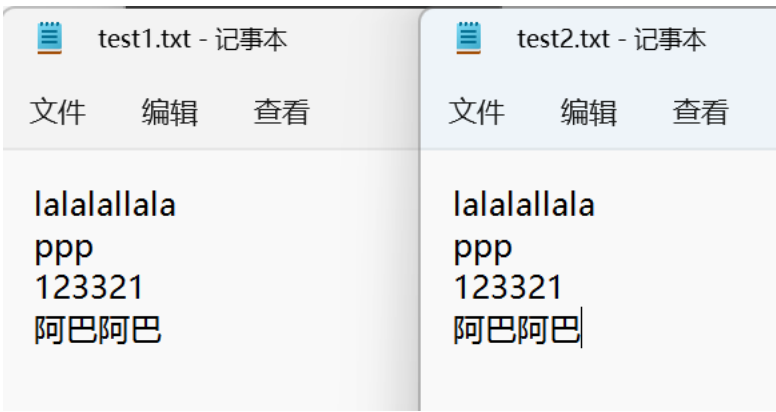


图 5-9 文本比对测试 2 文本

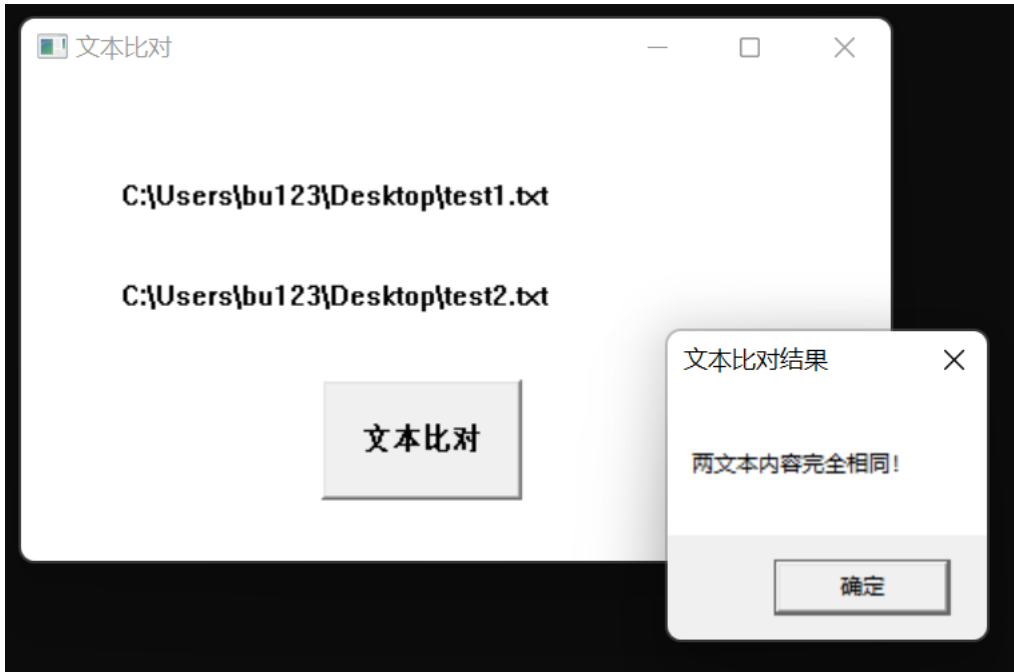


图 5-10 文本比对测试 2 结果

两文本内容不同时，能给出不同的行号，结果如图。



图 5-11 文本比对测试 3 文本



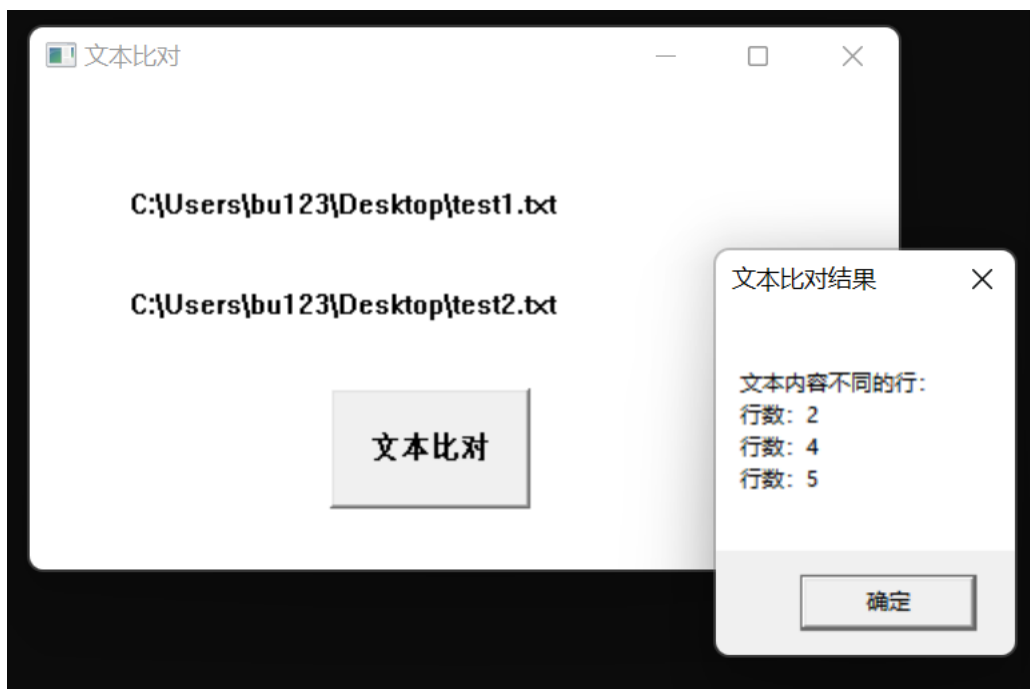


图 5-12 文本比对测试 3 结果

测试结果表明，实验要求中的功能已全部实现。

## 第 6 章 心得体会

本次实验是我第一次使用汇编语言编写程序。编程过程可以说是非常坎坷，因为对汇编指令不够熟悉，以及第一次编写汇编，对汇编程序结构也不够熟悉，导致刚开始编写时难以和书本上的知识联系起来，一直出错。后来在与同学交流和网上查阅资料后，我对汇编指令、汇编函数编写、参数传递机制、汇编调用C语言、汇编调用Windows提供的界面API等等有了较为深入的理解。本次实验我的心得体会主要有以下几点。

(1) 对汇编语言基础知识有了更深入的理解，包括对汇编指令调用、浮点汇编指令、汇编函数编写、函数参数传递、汇编调用C语言函数等等。

(2) 掌握汇编语言环境配置方法，能够编写简单的汇编程序与Windows风格的应用程序。

### 结 论

根据实验结果，本次实验中三个试验任务均顺利完成，要求的功能全部实现。