汇编语言与逆向技术课程实验报告

实验三：Bubble

学　　院 网络空间安全学院
专　　业 信息安全
学　　号 2211044
姓　　名 陆皓喆
班　　级 信息安全

一、实验内容与设计

1.1 实验内容

本次实验要求编写汇编程序 bubble_sort.asm，功能是将 Windows 命令行输入的 10 个 1 万以内的十进制无符号整数，进行排序，然后输出在 Windows 命令行中。10 个无符号整数之间用逗号","或者空格" "分割。

使用 StdIn 函数获得用户输入的十进制整数序列。StdIn 函数的定义在\masm32\include\masm32.inc，库文件是\masm32\lib\masm32.lib。StdIn 函数的定义“StdIn PROTO :DWORD,:DWORD”，有两个参数，第一个是内存存储空间的起始地址，第二个是内存存储空间的大小。

使用 StdOut 函数在 Windows 命令函中输出排好序的十进制整数序列。StdOut 函数的定义在\masm32\include\masm32.inc，库文件是\masm32\lib\masm32.lib。StdOut 函数的定义“StdOut PROTO :DWORD”，只有一个参数，是内存存储空间的起始地址。

使用 ml 和 link 程序将源代码编译、链接成可执行文件 bubble_sort.exe。

1.2 实验设计

本实验的主要思路分为三个部分，在代码实现中我使用了三个过程来分别实现。第一个过程是 str_to_array，表示从字符转化到数字；第二个过程是核心函数——冒泡排序的函数 bubble，能够实现对数组的排序；第三个过程是 array_to_str，能够实现从数字转化到字符。

二、源代码与注释

```
.386
.model flat, stdcall
option casemap :none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\masm32.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\masm32.lib
.stack 4096
.data;定义数据段
        str1 byte "please input ten numbers:",0;输出“please input ten numbers:”
        str2 byte "the result is:",0;输出“the result is:”
        istr byte 80 dup(0);定义 byte 型变量 istr
        ostr byte 80 dup(0);定义 byte 型变量 ostr
        array dword 12 dup(0);定义数组 array，初值均为 0
       const10 dword 10;定义常数 const10
.code;定义代码段
main PROC;主函数的代码
    invoke StdOut,addr str1;输出上面的提示语
    invoke StdIn,addr istr,80;输入 10 个数字
    call str_to_array;调用过程 str_to_array
    call bubble;调用冒泡排序的过程
    call array_to_str;调用过程 array_to_str
    invoke StdOut,addr str2;输出最后的提示语
    invoke StdOut,addr ostr
    invoke ExitProcess,0;程序结束
main ENDP
str_to_array PROC;字符串转数字
```

```
    MOV eax,0
    MOV ebx,0
    MOV ecx,0
    MOV esi,0;将四个寄存器的值均赋值为 0
L1:
    MOV bl,[istr+esi];将 array 数组中的第 esi 个值赋给 bl（bl 是 ebx 的后八位）
    CMP bl,20h;判断是否遇到空格，如果遇到了空格说明已经完成了一个数字的转录
    JNE L2;跳转到 L2 即可
    ADD ecx,4;否则就跳转到数组下一个内存空间进行存储
    INC esi;esi 执行加 1 操作跳转到下一个数字
    MOV bl,[istr+esi];将 istr 的后移 esi 的位置赋值给 bl
L2:
    SUB bl,30h;字符串转数字,并且存在 ebx 中
    MOV eax,[array+ecx];array+ecx 表示最后一位的值，存到 eax 中
    MUL const10;将 eax 乘上 10
    ADD eax,ebx;将 eax 与 ebx 的值加在一起，得到了比原来多一位的数字
    MOV [array+ecx],eax;将计算出来的 eax 赋值给 array 数组的对应位置
    INC esi;esi 寄存器执行加一操作
    CMP [istr+esi],0;比较 istr 的后移 esi 位与 0 的关系，若不等于 0 就跳转到 L1 继续循环
    JNE L1;如上，若不等于 0 就继续跳转到 L1 继续执行
    ret
str_to_array ENDP;字符串转数字代码段结束

bubble PROC;冒泡排序代码段开始
    MOV ecx,10;一共 10 个数，需要进行 10 轮循环
L3:
    DEC ecx;每次循环都需要减掉 1，循环 10 次
```

CMP ecx,0;判断 ecx 是否等于 0，即循环是否结束

JE exit;若结束则直接退出循环

MOV ebx,ecx;将外层循环的值赋值给 ebx，用来对内部的循环计数

MOV esi,0;给 esi 寄存器清零

L4:

MOV eax,[array+esi];将 array 数组第 esi 个数字赋值给 eax

CMP eax,[array+esi+4];将 eax 的值与第 esi+1 个数字进行比较

JLE L5;如果比前面的数字大，就执行 L5，如果小的话就执行下面两句话

XCHG eax,[array+esi+4];将 eax 寄存器中的值（当前的数字）与后一位的值进行交换

MOV [array+esi],eax;将交换之后的 eax 的值赋值给目前的[array+esi]，第 esi 个元素

L5:

DEC ebx;ebx 计数器进行减一操作，进行计数

CMP ebx,0;将 ebx 的值与 0 进行比较，判断循环是否结束

JE L3;如果结束了，那么就跳转到 L3

ADD esi,4h;把 esi 的内存位置加 4h，移动到后面一位

JMP L4;跳转到 L4 语句，继续进行交换操作，直到 10 次减完（ebx 为 0）结束

exit:

ret

bubble ENDP;冒泡排序代码段结束

array_to_str PROC

MOV esi,0;访问 ostr 每一位

MOV edi,0;访问 array 每一位

MOV ecx,10;计数外层循环(共 10 个数)

MOV ebx,0;计数出栈次数

MOV eax,[array+edi];给 eax 寄存器分别赋值

L6:

```
        MOV edx,0;存商
        DIV const10;商存在 eax，余数存在 edx
        ADD edx,30h
        PUSH edx;入栈（先算的末位，先进后出）
        INC ebx;计数栈中有几个字符，决定出栈次数
        CMP eax,0;若商为 0，则该数已取完
        JNE L6
L7:
```

POP eax;这一段主要实现的是将 ebx 寄存器中的值全部压入 esi 寄存器中，再进行加入空格的操作，若 ebx 的值为 0 的话，说明已经全部出栈了，直接跳出循环即可。

```
        MOV [ostr+esi],al
        INC esi
        DEC ebx
        CMP ebx,0;ebx 为 0 则说明当前整数的各位已经全部出栈，跳出循环
        JNE L7
        MOV [ostr+esi],20h;加一个空格
        INC esi
L8:;判断是否终止
        DEC ecx
        CMP ecx,0;ecx 为 0 则说明十个整数全被弹出，结束过程
        JE L9;如果没有全部弹出，则继续进行 L6，再入栈。
        ADD edi,4
        MOV eax,[array+edi]
        JMP L6
L9:
        ret
array_to_str ENDP
end main;程序结束
```

三、测试过程与截图

3. 1 生成 obj 文件

```
D:\>\masm32\bin\ml /c /Zd /coff C:\Users\Lenovo\Desktop\bubble.asm
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997.  All rights reserved.

 Assembling: C:\Users\Lenovo\Desktop\bubble.asm

***********
ASCII build
***********


D:\>
```

3.2 生成 exe 文件

```
C:\Users\Lenovo>D:

D:\>\masm32\bin\Link/SUBSYSTEM:CONSOLE C:\Users\Lenovo\Desktop\bubble.obj
Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
```

3. 3 用 exe 进行测试

```
D:\>C:\Users\Lenovo\Desktop\bubble.exe
please input ten numbers:5 8 7 9 10 2 4 3 1 6
the result is:1 2 3 4 5 6 7 8 9 10
D:\>
```

如图所示，测试成功!!

四、汇编语言数组操作知识点的总结

1.对数组进行访问时，需要通过**[基地址操作数+变址操作数]**进行访问。

2.数组进行寻址有两种方法：第一种是基址变址寻址方式，即把两个寄存器的值相加，得到一个偏移地址。两个寄存器分别称为基址寄存器（**base**）和变址寄存器（**index**）。格式为 **[base＋index]**，例如 **mov eax, [ebx＋esi]**；第二种是相对基址变址寻址方式，即把偏移、基址、变址以及可选的比例因子组合起来，产生一个偏移地址。常见的两种格式为：**[base + index + displacement]** 和 **displacement[base + index]**。

3.对数组当前位进行 mov 操作时，要判断当前[ ]为几位操作数，若为 8 位时则需要使用 32 位寄存器的后八位，如 **eax** 的 **al**、**ebx** 的 **bl** 部分，不然会出现错误。