

**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:

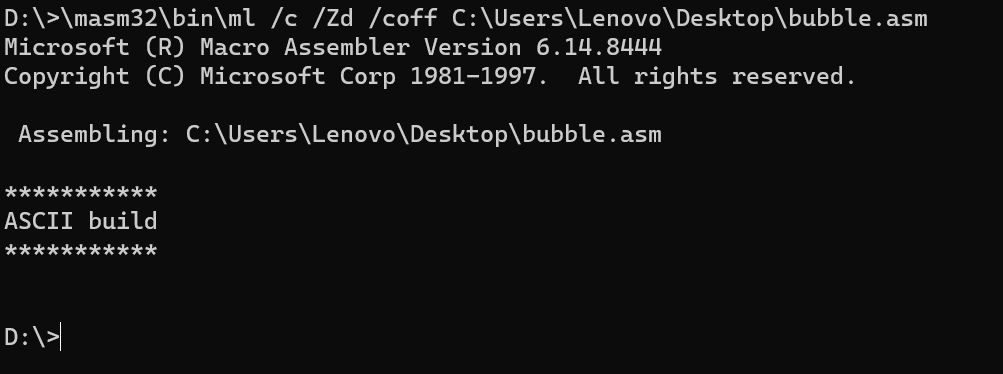
ret

array\_to\_str ENDP

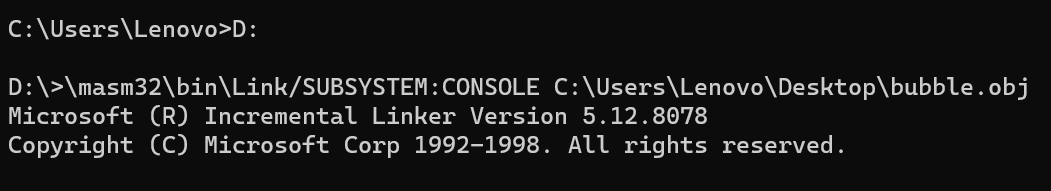
end main;程序结束

**三、测试过程与截图**

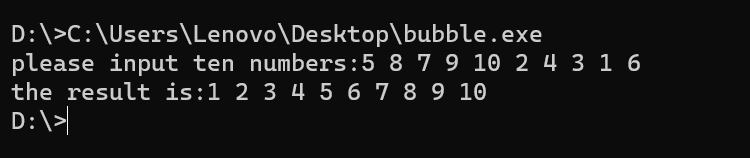
**3.1生成obj文件**



**3.2生成exe文件**



**3.3用exe进行测试**



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。