**汇编语言与逆向技术实验报告**

**实验3 整数数组的冒泡排序**

**学号：2113662 姓名：张丛 专业：信息安全**

1. **实验内容**
   1. 熟悉汇编语言的整数数组；
   2. 熟悉基址变址操作数、相对基址变址操作数；
   3. 掌握排序算法的底层实现细节
2. **实验步骤**

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. **实验设计报告**

使用StdIn输入10个元素并储存，然后：

1.比较相邻的元素。如果第一个比第二个大，就交换他们两个。

2.对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。

3.针对所有的元素重复以上的步骤，除了最后一个。

4.持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

排序后StdOut输出。

1. **代码和注释**

.386

.model flat,stdcall；平坦模式、调用约定

option casemap :none；大小写敏感

include \masm32\include\windows.inc

include \masm32\include\kernel32.inc

include \masm32\include\masm32.inc；函数的常量和声明

includelib \masm32\lib\kernel32.lib

includelib \masm32\lib\masm32.lib；链接库

.data

decnum DWORD 0000h,0000h,0000h,0000h,0000h,0000h,0000h,0000h,0000h,0000h

index DWORD 0,1,2,3,4,5,6,7,8,9

decstr byte 50 DUP(0), 0；用于储存输入的10个数（字符）

const10 DWORD 10；常数10

count DWORD 0

res byte 5 DUP(?), 0

count1 dword 0

count2 dword 0

i dword 0

spac byte " ",0；空格

.code

dec2dw PROC

mov esi, 0

mov ebx, 0

mov ecx,0

L1:

mov ecx, count

mov bl, [decstr+esi];取数字字符对应ASCII码

sub bl, 30h ;将ASCII码变为二进制码

mov eax, [decnum+ecx]

mul const10 ;10进制数提高位权值

mov [decnum+ecx], eax

add [decnum+ecx], ebx

L2:

inc esi

mov bl, [decstr+esi] ;[ carname +num]寻址操作

cmp bl,20h

je L3

cmp bl, 0 ;ASCII码 0 值

jnz L1

L3:

cmp bl, 0 ;ASCII码 0 值

je L4

add count,4

inc esi

jmp L1

L4:

ret

dec2dw ENDP

bubble\_sort PROC

L0:

mov ecx,9；外层循环，9次

L1:

mov count,ecx

mov ecx,9；内层循环，9次

sub ecx,i

mov esi,0

mov edx,0

mov ebx,0

L2:

mov esi, [decnum+ebx]；每次（外层循环）从头开始

mov edx,ebx

add edx,4

mov edi,[decnum+edx]

cmp esi,edi；比较相邻两个

ja L3；前者大于后者则交换

jmp L4

L3:

mov eax,[decnum+ebx]

xchg eax,[decnum+edx]

mov [decnum+ebx],eax

mov eax,[index+ebx]

xchg eax,[index+edx]

mov [index+ebx],eax

jmp L4

L4:

add ebx,4

LOOP L2；内层循环

mov ecx,count

inc i

LOOP L1；外层循环

ret

bubble\_sort ENDP

output PROC

mov i,0

L0:

mov esi, 0

mov eax,0

mov ebx, 0

mov ecx,0

mov count,0

mov edx,0

L1:

mov ecx,i

mov eax,count

cmp eax,[index+ecx\*4]

je L

L2:

inc esi

mov bl, [decstr+esi];

cmp bl,20h

je L3

jmp L2

L3:

inc count

jmp L1

L:

cmp bl,20h

jne L4

inc esi

L4:

mov bl, [decstr+esi];

cmp bl,20h

je L6

cmp bl,0

je L6

L5:

mov [res+edx],bl

inc esi

inc edx

jmp L4

L6:

mov [res+edx],0

invoke StdOut, addr res

invoke StdOut,addr spac

L7:

inc i

cmp i,10

je L8；相等则输出完毕

jmp L0

L8:

ret

output ENDP

start:

invoke StdIn, addr decstr, 50；输入

call dec2dw；调用转化字符的函数

call bubble\_sort；调用冒泡排序的函数

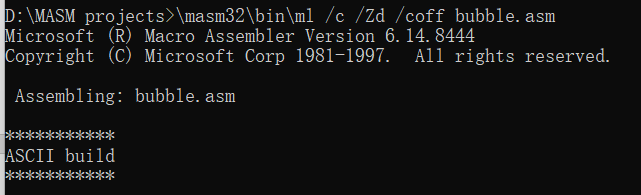
call output；调用输出的函数

invoke ExitProcess, 0

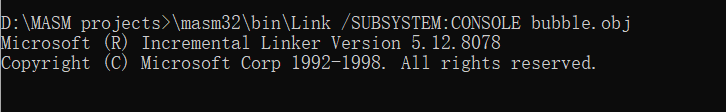
END start

1. **实验截图**

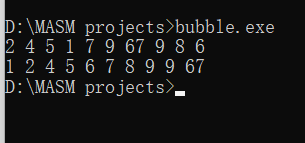
**汇编：**



**链接：**



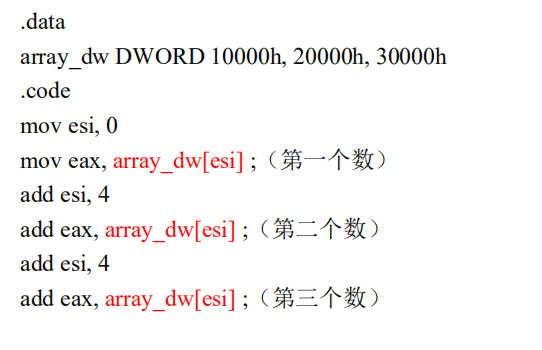
**测试：**



1. **数组操作知识点总结**

变址操作数（indexed operand）把常量和寄存器相加以得到一个有效地址。

任何32位通用寄存器都可以作为变址寄存器

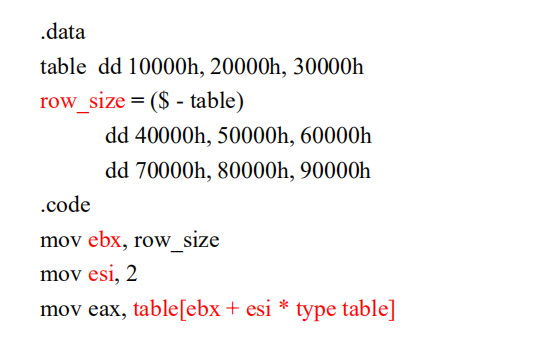


（1）基址变址寻址方式

基址变址（base-index）操作数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]。在例子中，ebx是基址寄存器，esi是变址寄存器。基址寄存器和变址寄存器可以使用任意的32位通用寄存器。

（2）相对基址变址寻址方式

相对基址变址（based-indexed with displacement）操作数把偏移、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]



1. **实验心得**

**本次实验熟悉了基址变址寻访和相对基址变址寻访，复习了冒泡排序的思想。此外，对10和数字的输入、储存、转化，以及排序后的输出也进行了考查，难度较大。思路清晰，但代码工作量比较大，容易出错。**

**冒泡排序中，需要运用循环的嵌套，需要掌握。**