汇编语言与逆向技术实验报告

Lab3- Bubble Sort

学号：2112514　姓名：辛浩然　专业：信息安全、法学

一、 实验目的

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。

二、 实验内容

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。

三、 实验步骤

1.使用 StdIn 函数获得用户输入的十进制整数序列，存至 decstr 中；

2.input 过程：将字符串转换成 10 个 word 类型的数据存至 array 中：对读取到的字符，转换成数字，乘 10 并加上下一位数字，然后结果再乘 10 再加下一位数字，……，直至读到’,’为止。这样得到了一个数，存到 array 中。之后进行类似操作。

3.bubble_sort 过程：对 array 中 10 个数据进行冒泡排序：共有 9 轮比较，每轮比较得到一个有序元素，置于无序部分后。每轮比较两两相邻元素，若前者大于后者，则二者互换，直到比较完最后一个无序元素。

4.output 过程：将 array 中 10 个 word 类型的数据按位转换成 ASCII 码存到 res 中：对每个数据，除以 10，余数反序存至 res 中（通过栈实现）；进行上述操作直到数据变为 0。其他数据操作相同。

5.使用 StdOut 函数在 Windows 命令函中输出排好序的十进制整数序列；

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

四、 实验代码及代码分析

```
.386
.model flat, stdcall
option casemap:none
include D:\masm32\include\windows.inc
include D:\masm32\include\kernel32.inc
include D:\masm32\include\masm32.inc
includelib D:\masm32\lib\kernel32.lib
includelib D:\masm32\lib\masm32.lib

.data
          decstr      byte 50 dup(0)
          ;decstr 存放输入的字符串序列
          array       word 10 dup(0)
          ;定义 10 个 word 元素。将 ASCII 字符串转换成 word 数据，存到 array 中。因为每个
数转换成数据以十六进制形式储存时，最多占用 2 字节，所以定义 word 类型
          res         byte 60 dup(0)
          ;将排序后的数字的十进制序列转换成 ASCII 码，用以输出
          num         dword 9
          const10word word 10
          tmp         dword ?
          break       byte ','

.code
main proc
                invoke StdIn,addr decstr,50
                ;输入字符串序列，存至 decstr 中
                call   input
                call   bubble_sort
                call   output
                invoke StdOut,addr res
                ;输出排序好的字符串序列
                invoke ExitProcess,0
main endp

input proc
                mov    eax,0
                mov    ebx,0
                mov    ecx,0
```

```
                ;借助变址寄存器 ecx 访问 decstr 元素
                mov     esi,0
                ;借助变址寄存器 esi 访问 array 元素
        dec2dw:
                mov     al,decstr[ecx]
                ;间接寻址，等同于[ecx+decstr]，ecx 的值与 decstr 的偏移地址相加，得到内存 decstr 的第 ecx 个字符的地址，并复制到 al 中
                sub     al,48
                ;将字符转换成数字
                xchg    ax,bx
                ;交换 ax 和 bx 的值，ax 中是上步的结果，bx 中是该步处理的数字即现在的个位
                mul     const10word
                ;ax 乘 16 位操作数，将结果存至 dx 和 ax 中，事实上，受到数据范围的限制,dx 为 0
                add     ax,bx
                ;将 bx 加至 ax 中，即上步的结果乘 10 后加上个位
                inc     cx
                ;要处理的字符序号加 1
                xchg    ax,bx
                ;交换 ax 和 bx 的值，把该步的结果存到 bx 中
                mov     al,decstr[ecx]
                ;将第 ecx 个字符(即刚处理完的字符的下一个字符)存至 al 中
                cmp     al,','
                ;将字符与','进行隐含的减法操作
                jz      NumEnd
                ;如果结果为 0，二者相等，说明处理完了一个数，跳至 NumEnd
                cmp     al,0
                ;将字符与 0 进行隐含的减法操作
                jnz     dec2dw
                ;如果结果不为 0，二者不相等，说明未处理完，回到 dec2dw，继续处理
                cmp     al,0
                ;将字符与 0 进行隐含的减法操作
                jz      AllEnd
                ;如果结果为 0，二者相等，说明全部处理完，跳至 AllEnd
        NumEnd:
                mov     array[si],bx
                ;array[si]表示 array 中的第 si 个 word 元素，在输入','后，一个数处理完，存到 array 中
                add     si,type array
                ;下个要赋值的是 array 中的下一个元素，所以 si 变成 si+type array
                mov     ebx,0
                mov     eax,0
                ;ebx 与 eax 重新置 0
```

```
                inc     ecx
                ;要处理的字符序号加 1
                jmp     dec2dw
                ;回到 dec2dw，继续处理后面的字符
        AllEnd:  mov     array[si],bx
                ;将处理的最后一个数存在 array 中
                ret
input endp

bubble_sort proc
                mov     ecx,9
                ;外层循环的次数
        Outer:
                mov     tmp,ecx
                mov     ecx,num
                ;修改 ecx 为内层循环的次数，把外层循环的次数先暂存至 tmp 中。第一次内层循环为 9 次，后面逐次递减
                mov     esi,offset array
                ;变址寄存器，esi 为 array 的基地址
        Inner:
                mov     ax,[esi]
                ;从第一个数开始，该数为 word 类型，16 位，将该数赋值给 ax
                cmp     ax,[esi+type array]
                ;将 ax 与下一个数进行隐含的减法操作
                jc      notsort
                ;结果小于 0，CF=1，则跳转 L3
                xchg    ax,[esi+type array]
                mov     [esi],ax
        notsort:
                add     esi,type array
                ;esi 变为第下一个数的地址
                Loop    Inner
                ;内层循环
                dec     num
                ;num 减 1，每次多一个有序数，少循环一次
                mov     ecx,tmp
                ;外层循环的次数
                loop    Outer
                ;外层循环
                ret
bubble_sort endp

output proc
                mov     eax,0
```

```
                mov     bx,0
                mov     edx,0
                mov     ecx,10
                ;循环 10 次
                mov     esi,offset array
                ;变址寄存器，esi 为 array 的基地址
                mov     edi,offset res
                ;变址寄存器，edi 为 res 的基地址
        todec:
                mov     ax,[esi]
                ;esi 地址的数赋给 ax
        pushnum:
                mov     dx,0
                div     const10word
                ;除以 word 类型的 10，除以 16 位操作数，商存在 ax 中，余数存在 dx 中
                add     dx,48
                ;dx 中余数转换成 ASCII 码
                push    dx
                ;入栈
                inc     bx
                ;bx 用来记录进入栈的元素数目
                cmp     ax,0
                ;对 ax 和 0 进行隐含的减法操作
                jnz     pushnum
                ;若 ax＝0，即原数的所有位都已经转成 ASCII 码并压入栈，那么这个数就处理完了；如果没不为 0，没处理完，循环 L10
        popnum:
                pop     dx
                ;栈顶元素赋值给 dx
                mov     [edi],dl
                ;因为元素最多 1 字节，dx 的前 8 位为 0，将 dx 存到 res 数组中即将 dl 存到 res 数组中
                dec     bx
                ;栈中元素数目减一
                add     edi,type res
                ;edi 向后偏移
                cmp     bx,0
                jnz     popnum
                ;如果 bx 不为 0，栈不空，继续出栈
                cmp     ecx,1
                jz      theend
                ;ecx 等于 1 时，是处理完所以元素了，直接跳出循环
                ;下面的操作是在每次每个数把每位存到 res 后，再把一个','存到 res 中，以便于输出好看；最后一个数后不用加','所以最后一次循环时在上面跳出不执行后面的操作
```

```
                    mov     tmp,ecx
                    mov     cl,break
                    mov     [edi],cl
                    ;将','存进去
                    mov     ecx,tmp
                    add     edi,type res
                    ;edi 向后偏移
                    add     esi,type array
                    ;这个数的每位都处理完了，esi 向后偏移
                    loop    todec
        theend:
                    ret
output endp

end main
```

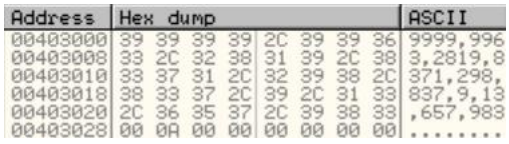
五、 测试过程和截图

1.编译链接与运行

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

 Assembling: bubble.asm

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


D:\汇编与逆向>\masm32\bin\link /SUBSYSTEM:CONSOLE bubble.obj
Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.


D:\汇编与逆向>.\bubble.exe
9999,9013,975,653,1,98,101,76,1,209
1,1,76,98,101,209,653,975,9013,9999
```

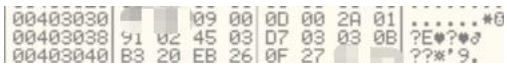
2.一测试用例内存展示

输入的字符串存放在数组 decstr 中：



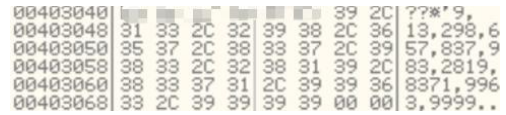
Address	Hex dump	ASCII
00403000	39 39 39 39 2C 39 39 36	9999,996
00403008	33 2C 32 38 31 39 2C 38	3,2819,8
00403010	33 37 31 2C 32 39 38 2C	371,298,
00403018	38 33 37 2C 39 2C 31 33	837,9,13
00403020	2C 36 35 37 2C 39 38 33	,657,983
00403028	00 0A 00 00 00 00 00 00	....

字符串转换成 word 类型整数后存在数组 array 中，并排序好：



00403030	[illegible]		09	00	0D	00	2A	01	......*☺
00403038	91	02	45	03	D7	03	03	0B	?E♥?♥∂
00403040	B3	20	EB	26	0F	27	[illegible]		??*'9.

word 类型的整数再转换成 ASCII 码：



00403040	[illegible]						39	2C	??*'9,
00403048	31	33	2C	32	39	38	2C	36	13,298,6
00403050	35	37	2C	38	33	37	2C	39	57,837,9
00403058	38	33	2C	32	38	31	39	2C	83,2819,
00403060	38	33	37	31	2C	39	39	36	8371,996
00403068	33	2C	39	39	39	39	00	00	3,9999..

六、 汇编语言数组操作

1.数组的定义

(1)可以用数据类型定义数组，如：

```
array dword 10h,20h,30h,40h
```

```
string byte 'Hello', 0
```

(2)定义一个具有一系列相同元素的数组可以用 dup，如：

```
array word 10 dup(0)
```

2.访问数组元素：通过数组首地址和偏移量访问

(1)直接偏移寻址：

变量名称后加偏移值，访问没有显式标号的内存地址。如：

```
mov al,[array+1]
```

(2)寄存器间接寻址：

使用间接操作数，间接操作数可以是任何用方括号括起来的任意 32 位通用寄存器，寄存器里存放着数据的偏移。

```
val byte 10h
```

```
mov esi,offset val
```

```
mov al,[esi]
```

(3)寄存器相对寻址：

使用变址操作数：把常量和寄存器相加以得到一个有效地址。

constant[reg]

[constant+reg]

①把变量名和寄存器结合在一起，变量名是表示变量偏移地址的常量。如：

```
mov esi,0
mov ax,[array+esi]
add esi,type array
add ax,array[esi]
```

②把变址寄存器与常量偏移结合，用变址寄存器存放数组或结构的首地址，用常量标识各个数组元素。如：

```
mov esi, offset array
mov ax, [esi]
add ax, [esi+type array]
```

(4)基址变址寻址：

基址变址操作数把两个寄存器的值相加，得到一个偏移地址。

[base + index]

如：

```
mov ebx,offset array
mov esi,2
mov ax,[ebx+esi]
```

(5)相对基址变址寻址：

相对基址变址操作数把偏移、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。

[base+index+displacement]

displacement[base+index]

displacement 可以是变量的名字或常量表达式。如：

```
mov ax, 0x1000[bx][si]
mov ax, [bx + si + 0x1000]
mov ax, 0x1000[si][bx]
mov ax, 0x1000[bx + si]
```

(6)使用指针

创建指针并用数组的初始地址进行初始化，运行时析取指针的值用以访问数据：

```
arraya dword 10h,20h,30h,40h
arrayb byte 10h,20h,30h,40h
ptra dword offset arraya
pbyte typedef ptr byte
ptrb pbyte arrayb
mov esi,ptra
mov eax,[esi]
mov esi,ptrb
mov bl,[esi]
```