

**汇编语言与逆向技术**

**课程实验报告**

**实验三：bubble\_sort**

****

学 院 网络空间安全

专 业 信息安全

学 号 2212998

姓 名 胡博浩

班 级­­­­ 2022级信息安全

1. **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
2. **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. **实验步骤**

1. bubble\_sort.asm源代码及注释。

.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

.data

buffer DWORD 10 DUP(0);用于存储实际的数字数组

input BYTE 128 DUP(0); 用于保存用户输入的字符串

output BYTE 128 DUP(0);用于输出最后的字符串

.code

start:

invoke StdIn, ADDR input, sizeof input; 读取用户输入的整数序列，以空格分隔

call trans1;将字符串转换为数字数组

call sort;冒泡排序

call trans2;将数字数组转换为字符串

invoke StdOut, ADDR output;输出结果

invoke ExitProcess, 0

trans1 PROC

;全部清零

xor esi,esi;记录字符串input的位置

xor eax,eax

xor ebx,ebx;临时保存转换后的数字

xor ecx,ecx;记录外层循环次数，即转换了几个数字

; 扫描输入，直到遇到空格，结束符或者达到最大整数个数

ParseInput:

mov al, byte PTR[input + esi];先读取一个字符

inc esi

cmp al, ' ' ; 检查是否是空格

je Yes

cmp al,0; 检查是否是字符串结束符

je Yes

sub al, '0' ; 将字符转为数字

imul ebx,10

add ebx,eax

jmp ParseInput

Yes:

;是空格或者结束符，表明一个数输入结束，进行下一个数的转换

mov dword PTR[buffer+ecx\*4],ebx;将数字传给数字数组

inc ecx; 移到下一个整数的起始位置

xor ebx,ebx;清零，保证下次循环正常

cmp ecx,10;判断有没有输完10个数字

jne ParseInput

ret

trans1 ENDP

sort PROC

mov ecx, 9;外层循环次数

mov ebx, 0

mov esi,0

outer:

mov esi, 0;将其清零，相当于C++中的int i=0

inner:

mov eax, dword PTR[buffer + esi \* 4]

mov edx, dword PTR[buffer + esi \* 4 + 4]

cmp eax, edx;判断是否交换数据

jbe No

; 交换两个整数

mov dword PTR[buffer + esi \* 4], edx

mov dword PTR[buffer + esi \* 4 + 4], eax

No:

inc esi

cmp esi,ecx;若内层循环次数和外层循环次数相等，则结束内层循环

jne inner

dec ecx

cmp ecx,0;判断外层循环是否结束

jne outer

ret

sort ENDP

trans2 PROC

xor edi,edi

xor ecx,ecx;记录output的位置（相当于数组下标）

xor esi,esi;记录外层循环次数

xor edx,edx;作为余数

mov ebx,10;设置10为除数

;外层循环10次，分别转换一个数字

;里面两个循环，负责出栈和入栈,利用栈先进后出的性质

Big:

mov eax,dword PTR[buffer+esi\*4]

L1:

xor edx,edx;将其清零

div ebx

inc edi;记录压栈次数

add edx,'0'

push edx;将余数压入栈

cmp eax,0

jne L1;当商为0，则结束

L2:

dec edi

pop eax;出栈

mov byte PTR[output+ecx],al

inc ecx

cmp edi,0;当edi为0，说明出栈结束

jne L2

mov byte PTR[output+ecx], ' ';给每个字符添加空格

inc ecx

xor edi,edi;清零保证下次循环正常

inc esi

cmp esi,10;判断有没有到10个数

jne Big

ret

trans2 ENDP

END start

该汇编程序共有三个过程（PROC）：trans1、sort和trans2。

trans1 过程用于将用户输入的字符串转换为数字数组。它通过扫描输入字符串，遇到空格或字符串结束符时，将已构建的整数存储到数字数组中。数字数组的最大容量为10个整数。

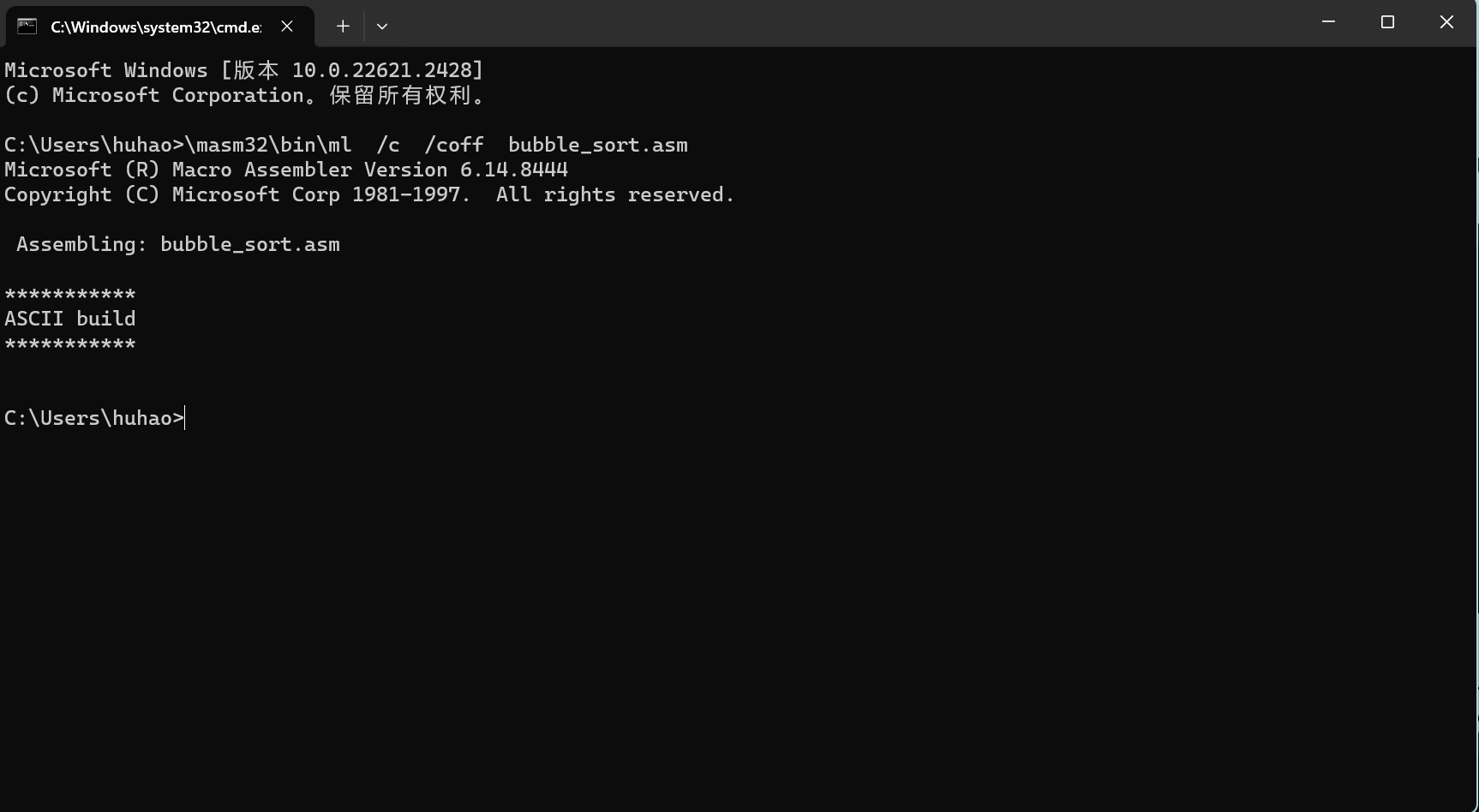
sort 过程使用冒泡排序算法对数字数组进行排序。它通过比较相邻的两个整数，并根据需要交换它们的位置，继续扫描整个数组直到排序完成。

trans2 过程将排序后的数字数组转换为字符串。它遍历数字数组中的每个整数，并使用栈来逆序获取整数的每一位数字，并将其转换为字符。然后，将字符存储到输出字符串中，并在每个整数之间添加一个空格。

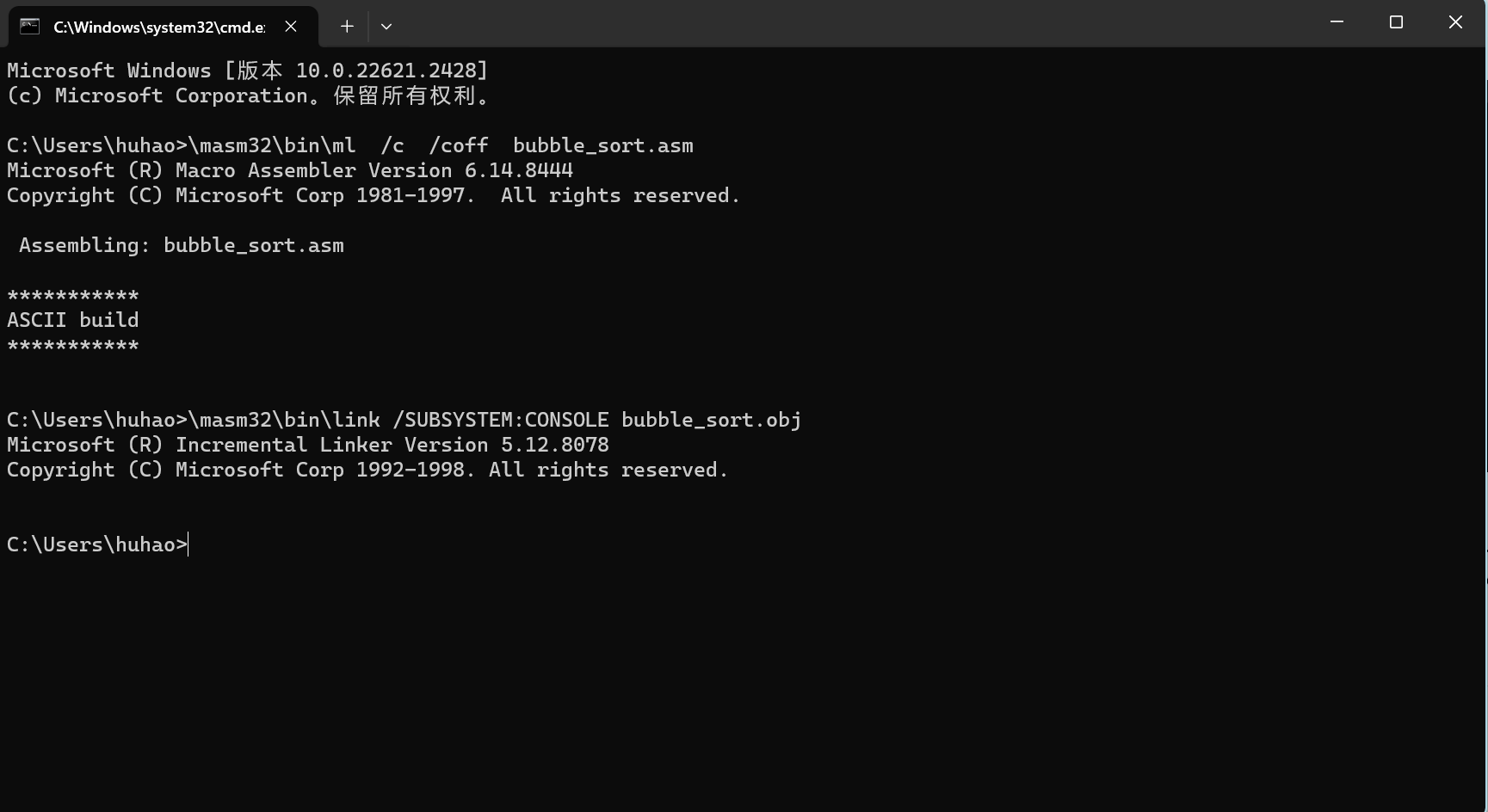
整个程序的核心逻辑是：先将用户输入的字符串转换为数字数组，然后对数字数组进行排序，最后将排序后的数组转换回字符串进行输出。

2. bubble\_sort.asm源代码的编译和链接过程说明。

1）使用ml将bubble\_sort.asm文件汇编到dec2hex.obj目标文件，编译命令：“\masm32\bin\ml /c /coff bubble\_sort.asm”

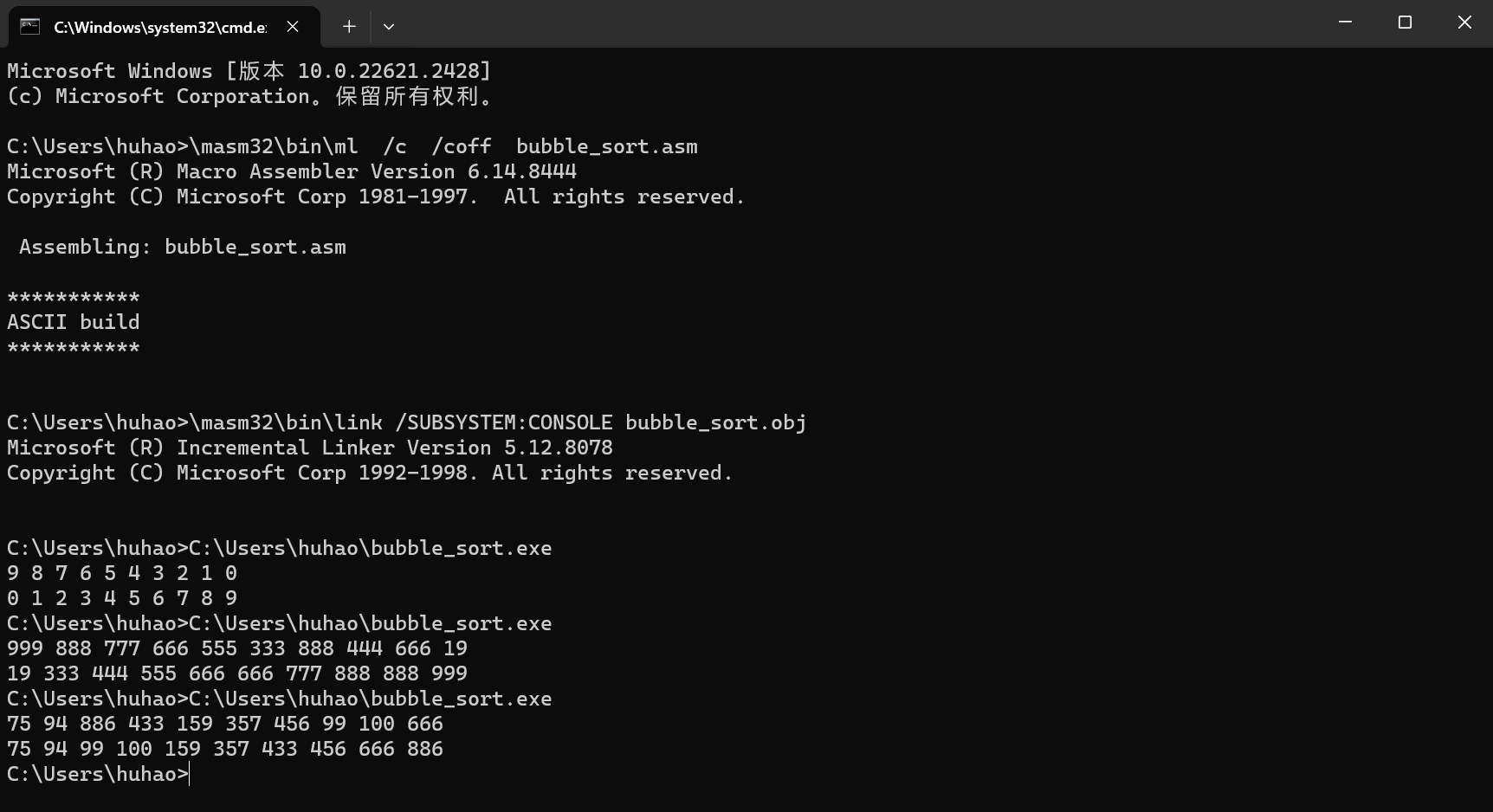


2）使用link将目标文件dec2hex.obj链接成dec2hex.exe可执行文件，链接命令：“\masm32\bin\link /SUBSYSTEM:CONSOLE bubble\_sort.obj”



3. bubble\_sort.exe的测试说明。

执行生成的 bubble\_sort.exe文件。输入三个不同的案例，程序能够正确输出3组排序好的数，说明程序正确，实验三成功。



1. **汇编语言数组操作知识点的总结。**

在汇编里使用数组，有两种方式，分别为基址变址寻址方式和相对基址变址寻址方式。

1. 基址变址寻址方式

基址变址寻址方式是汇编语言中常用的一种访问内存的方法。它涉及两个寄存器：基址寄存器（base）和变址寄存器（index）。基址寄存器 (base)通常用于存储数组的起始地址。变址寄存器 (index)通常用于存储数组中元素的偏移量。格式为[base + index]。例如，mov eax, [ebx + esi]

2. 相对基址变址寻址方式

相对基址变址寻址方式在基址变址寻址方式的基础上增加了一个可选的偏移量（displacement）。

常见的两种格式为：

[base + index + displacement]

displacement[base + index]

示例：

.data

table dword 10h, 20h, 30h, 40h

.code

mov ebx, 2 ; 基址寄存器 ebx 存储了数组 table 的起始地址

mov esi, 1 ; 变址寄存器 esi 存储了偏移量，表示我们想要访问 table[1]

mov eax, [ebx + esi + 2] ; 将 table[1] 的值（20h）加载到 eax 中，偏移量增加了2

在这个例子中，mov eax, [ebx + esi + 2] 中的 +2 是一个额外的偏移量，它会被添加到基址寄存器和变址寄存器的和中，用于计算最终的内存地址。

使用基址变址寻址方式和相对基址变址寻址方式时，确保寄存器中的值和数组的大小以及索引等是正确匹配的。注意避免越界访问数组，以防止产生未定义行为或错误。否则将直接更改内存中其他地址的内容。例如，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al、ah、bl、bh等，若为16位则需要使用ax、bx等。