



南開大學
Nankai University

计算机学院

汇编语言与逆向技术实验报告

Lab3-整数数组的冒泡排序

姓名：杨冰雪

学号：2110508

专业：计算机科学与技术

2023 年 10 月 31 日

目录

| | |
|--------------------|----------|
| 1 实验目的 | 2 |
| 2 实验环境 | 2 |
| 3 实验内容 | 2 |
| 4 实验过程 | 2 |
| 4.1 源代码 | 2 |
| 4.2 编译 | 4 |
| 4.3 链接 | 5 |
| 4.4 运行结果 | 5 |
| 5 实验总结 | 5 |
| 6 实验心得 | 6 |

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 实验环境

MASM32 编译环境

Windows 命令行窗口

3 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。

4 实验过程

4.1 源代码

BubbleSort.asm

```
1 .386
2 .model flat , stdcall
3 option casemap:none
4 include D:\masm32\include\windows.inc
5 include D:\masm32\include\kernel32.inc
6 include D:\masm32\include\masm32.inc
7 includelib D:\masm32\lib\kernel32.lib
8 includelib D:\masm32\lib\masm32.lib
9
10 .data
11 str0 BYTE "please input ten number:",0
12 str1 BYTE "the Bubble Sort result is:",0
13 number_array Dword 10 DUP(0)
14 number_input BYTE 64 DUP(0)
```

```
15 number_output BYTE 64 DUP(0)
16 dec_index DWORD 10
17 count DWORD 10 ;输入十次
18 const10 Dword 10
19 str_sep BYTE ', '
20
21 .code
22 start:
23 ; 输入并将输入字符串数转化为十个十进制数
24 invoke StdOut, addr str0
25 Mov edi, offset number_array
26 ;循环十次输入
27 Input_str:
28     invoke StdIn, addr number_input, 64
29     Mov eax, 0 ;用来存放十进制数
30     Mov esi, 0
31 str2dec:
32     ;将字符串转化为十进制数
33     Mov bh, number_input[esi]
34     Sub bh, 48
35     Movzx ebx, bh
36     Mul dec_index ;eax中存放乘法的结果且edx也会受影响
37     Add eax, ebx
38     inc esi
39     cmp number_input[esi], 0
40     jnz str2dec
41 save_dec:
42     ;将十进制数存储到十进制数组中
43     Mov [edi], eax
44     Add edi, TYPE number_array
45     Dec count
46     cmp count, 0
47     jnz Input_str
48
49 ; 冒泡排序
50 sort:
51     Mov ecx, 9 ;外层循环9次
52 L1:
53     Mov ebx, ecx
54     Mov edi, offset number_array
55 L2:
56     Mov eax, [edi]
57     cmp eax, [edi+4]
58     JLE L3
59     xchg eax, [edi+4] ;交换元素位置
60     xchg eax, [edi]
61 L3:
62     Add edi, TYPE number_array
63     Loop L2
```

```

64     Mov ecx,ebx
65     Loop L1
66 ;将十进制数转化为字符串
67 invoke StdOut,addr str1
68     mov ebp,0
69 dec2str:
70     mov esi,4
71     mov eax,number_array[ebp]
72 save_result:
73     mov edx,0
74     div const10;
75     add edx,48
76     cmp dl,58
77     mov BYTE PTR[number_output+esi],dl;
78     dec esi
79     cmp esi,0
80     jnl save_result
81     INVOKE StdOut, addr number_output
82     INVOKE StdOut, addr str_sep
83     add ebp,4
84     cmp ebp,38
85     jl dec2str
86
87 invoke ExitProcess,0
88 END start

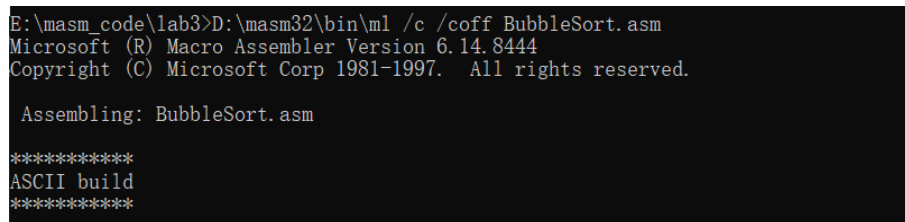
```

- 通过循环调用 10 次输入，因为 StdIn 函数只能输入字符型，所以还需要对每次输入的字符串转化为 DWORD 类型存入到数组中。
- 使用冒泡排序对数组进行排序，通过双重循环的方法实现，外层循环控制循环的趟数，内层循环控制次数；这里通过 Loop 指令来控制循环。
- 通过循环进行输出，因为 StdOut 函数只能输出字符串，所以要把 DWORD 类型的整数转换为字符串，通过把整数循环除以 10，依次储存余数直至商为 0，则表示已转换为字符串，调用输出并且输出分隔符”，”。

4.2 编译

在 Windows 命令行窗口中输入如下命令，对汇编文件进行编译。

```
1 "D:\masm32\bin\ml /c /coff BubbleSort.asm"
```



```

E:\masm_code\lab3>D:\masm32\bin\ml /c /coff BubbleSort.asm
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997. All rights reserved.

Assembling: BubbleSort.asm

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

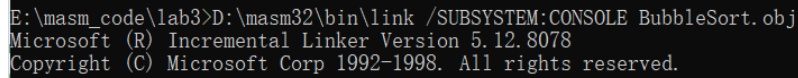
```

图 4.1: 编译

4.3 链接

在 Windows 命令行窗口中输入如下命令，对生成的目标文件进行链接，从而得到可执行文件。

```
1 "D:\masm32\bin\link /SUBSYSTEM:CONSOLE BubbleSort.obj"
```

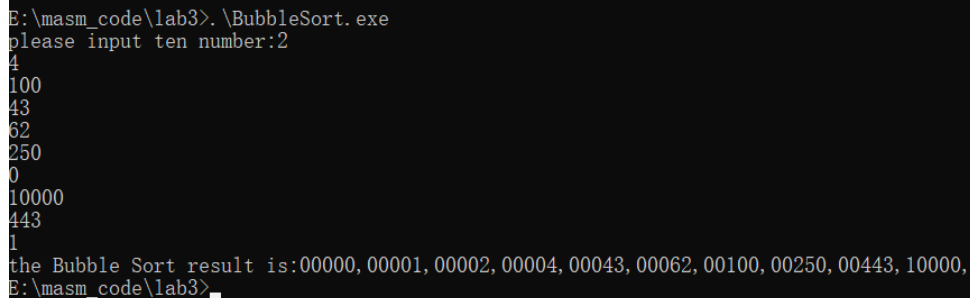


```
E:\masm_code\lab3>D:\masm32\bin\link /SUBSYSTEM:CONSOLE BubbleSort.obj
Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
```

图 4.2: 链接

4.4 运行结果

调用可执行文件，其运行结果如下：



```
E:\masm_code\lab3>. \BubbleSort.exe
please input ten number:2
4
100
43
62
250
0
10000
443
1
the Bubble Sort result is:00000, 00001, 00002, 00004, 00043, 00062, 00100, 00250, 00443, 10000,
E:\masm_code\lab3>_
```

图 4.3: 实验结果

5 实验总结

1. 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 (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]，例子如下：

```
1 table dword 10h, 20h, 30h, 40h
2 row_size = ($ - table)
3 dword 50h, 60h, 70h, 80h
4 dword 90h, 0a0h, 0b0h, 0c0h
5 mov ebx, row_size
6 mov esi, 2
7 mov eax, table[ebx + esi * 4]
```

table 是一个二维数组，共 3 行 4 列。`ebx` 是基址寄存器，相当于二维数组的行索引，`esi` 是变址寄存器，相当于二维数组的列索引。

3. 对于 dword 数组, 可用 [reg] 方式访问例如:

```
1  var1 dword 50h, 60h, 70h, 80h
```

其中 50h 为 var1[0]、60h 为 var1[4]、70h 为 var1[8]、80h 为 var1[12] 地址依次相隔 4 个字节

6 实验心得

通过本次实验, 让我对 masm 汇编有了更加深入的了解, 能更加熟练的定义和操作汇编语言的整数数组; 对于基址变址与相对基址变址等知识有了更深入的了解, 基本掌握排序算法的底层实现细节。