南开大学

Bubble

（汇编语言与逆向技术实验 3）

姓名：申宗尚

学号：2213924

专业：信息安全

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．实验环境

Windows 记事本的汇编语言编写环境

MASM32 编译环境

Windows 命令行窗口

三．实验内容

排序算法中，汇编语言的基址变址寻址方式和相对基址变址寻址方式起到了重要的作用。

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。

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：

```
table dword 10h, 20h, 30h, 40h
row_size = ($ - table)
        dword 50h, 60h, 70h, 80h
        dword 90h, 0a0h, 0b0h, 0c0h
mov ebx, row_size
mov esi, 2
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。

冒泡排序算法（Bubble Sort）的过程是从位置 0 和 1 开始比较每对数据的值，

如果两个数据的顺序不对，就进行交换。如果一遍处理完之后，数组没有排好序，就开始下一次循环。在最多完成 n-1 次循环后，数组排序完成。

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。

四．实验代码及注释解析

```
.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
    strPrompt byte "please insert numbers:", 0
    strResult byte "the answer is:", 0
    inputString byte 80 dup(0)
    outputString byte 80 dup(0)
    numbersArray dword 12 dup(0)
    constTen dword 10
.code ;定义代码段

main PROC
    ; 输出提示信息
    invoke StdOut, addr strPrompt
```

```
    ; 输入十个无符号整数
    invoke StdIn, addr inputString, 80
    ; 将输入字符串转换为整数数组
    call str2array
    ; 对整数数组进行冒泡排序
    call bubble
    ; 将排序后的数组转换为字符串
    call array2str
    ; 输出排序结果的提示信息
    invoke StdOut, addr strResult
    ; 输出排序后的字符串
    invoke StdOut, addr outputString
    ; 退出程序
    invoke ExitProcess, 0
main ENDP

; 将输入字符串转换为整数数组
str2array proc
    mov eax, 0
    mov ebx, 0
    mov ecx, 0
    mov esi, 0
L1:
    mov bl, [inputString + esi]
    cmp bl, 20h
    jne L2
    add ecx, 4
    inc esi
    mov bl, [inputString + esi]
L2:
    sub bl, 30h
    mov eax, [numbersArray + ecx]
    mul constTen
    add eax, ebx
    mov [numbersArray + ecx], eax
    inc esi
    cmp [inputString + esi], 0
    jne L1
    ret
str2array endp

; 对整数数组进行冒泡排序
bubble proc
    mov ecx, 10 ; 共十个数
```

```
L3:
    dec ecx ; 十个数，外层循环需 9 次，每次-1
    cmp ecx, 0 ; 判断是否循环结束
    je exit ; 若结束，则退出排序循环
    mov ebx, ecx ; 将外层循环的值赋给 ebx，用来计数内层循环
    mov esi, 0
L4:
    mov eax, [numbersArray + esi] ; 需要用 numbersArray 来访问数组元素！！
    cmp eax, [numbersArray + esi + 4]
    jle L5
    xchg eax, [numbersArray + esi + 4] ; 若不相等，则交换值
    mov [numbersArray + esi], eax
L5:
    dec ebx
    cmp ebx, 0
    je L3
    add esi, 4h
    jmp L4
exit:
    ret
bubble endp

; 将整数数组转换为输出字符串
array2str proc
    mov esi, 0 ; 访问 outputString 每一位
    mov edi, 0 ; 访问 numbersArray 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, [numbersArray + edi]
L6:
    mov edx, 0 ; 存商
    div constTen ; 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
    mov [outputString + 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 [outputString + 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
        add edi, 4
        mov eax, [numbersArray + edi]
        jmp L6
    L9:
        ret
    array2str endp

    end main
```

五．实验程序测试

```
C:\Users\KKkai>\masm32\bin\ml /c /coff C:\Users\KKkai\Desktop\bubble.asm
Microsoft (R) Macro Assembler Version 6.14.8444
Copyright (C) Microsoft Corp 1981-1997.  All rights reserved.

 Assembling: C:\Users\KKkai\Desktop\bubble.asm

************
ASCII build
************
```

```
C:\Users\KKkai>\masm32\bin\link /SUBSYSTEM:CONSOLE bubble.obj
Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
```

如图进行编译、链接

```
C:\Users\28301>C:\Users\28301\bubble.exe
please insert numbers:231234 23 524 3 5436 4364565 44 3245 586 4893
the answer is:3 23 44 524 586 3245 4893 5436 231234 4364565
C:\Users\28301>
```

如图，测试了输入十个整数：

231234 23 524 3 5436 4364565 44 3245 586 4893

输出其经冒泡排序后的升序序列：

3 23 44 524 586 3245 4893 5436 231234 4364565

六、数组知识点总结

（一）基址变址

基址变址（base-index）操作数将两个寄存器的值相加，生成一个偏移地址。这两个寄存器分别被称为基址寄存器（base）和变址寄存器（index）。格式为[base + index]，例如 mov eax, [ebx + esi]。在此示例中，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。

相对基址变址（based-indexed with displacement）操作数结合偏移、基址、变址以及可选的比例因子，形成一个偏移地址。常见的两种格式为：[base + index + displacement] 和 displacement[base + index]，例如：

```
table dword 10h, 20h, 30h, 40h
row_size = ($ - table)
dword 50h, 60h, 70h, 80h
dword 90h, 0a0h, 0b0h, 0c0h

mov ebx, row_size
mov esi, 2
mov eax, table[ebx + esi * 4]
```

在这个例子中，table 是一个二维数组，共 3 行 4 列。ebx 作为基址寄存器，相当于二维数组的行索引，esi 作为变址寄存器，相当于二维数组的列索引。

（二）行主序与列主序

从汇编语言程序员的视角来看，二维数组是一维数组的高阶抽象。对于二维数组在内存中行列的存储，高级语言一般采用下面的两种方法：行主序（row-major order）和列主序（column-major order）。使用行主序存储（最常使用）时，第一行放在内存块的开始，第一行的最后一个元素后接第二行的第一个元素。使用列主序存储时，第一列的元素放在内存块的开始，第一列的最后一个元素后接第二列的第一个元素。

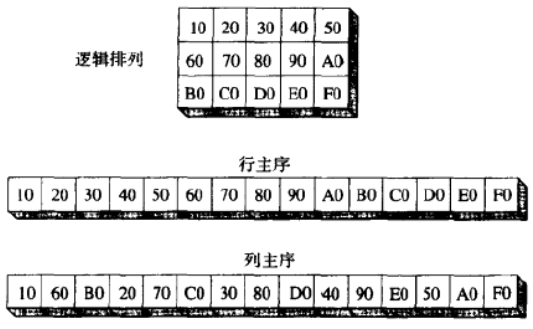

图 9.4　行主序和列主序