南開大學

汇编语言与逆向技术课程实验报告

实验三： 整数数组的冒泡排序

学　　院　网络空间安全学院
专　　业　信息安全、法学双学位
学　　号　2212000
姓　　名　宋奕纬
班　　级　1061

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、实验原理

1、 实验环境：

MASM32 编译环境、

Windows 命令行窗口、

Windows 记事本的汇编语言编写环境。

2、 实验思路

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
```

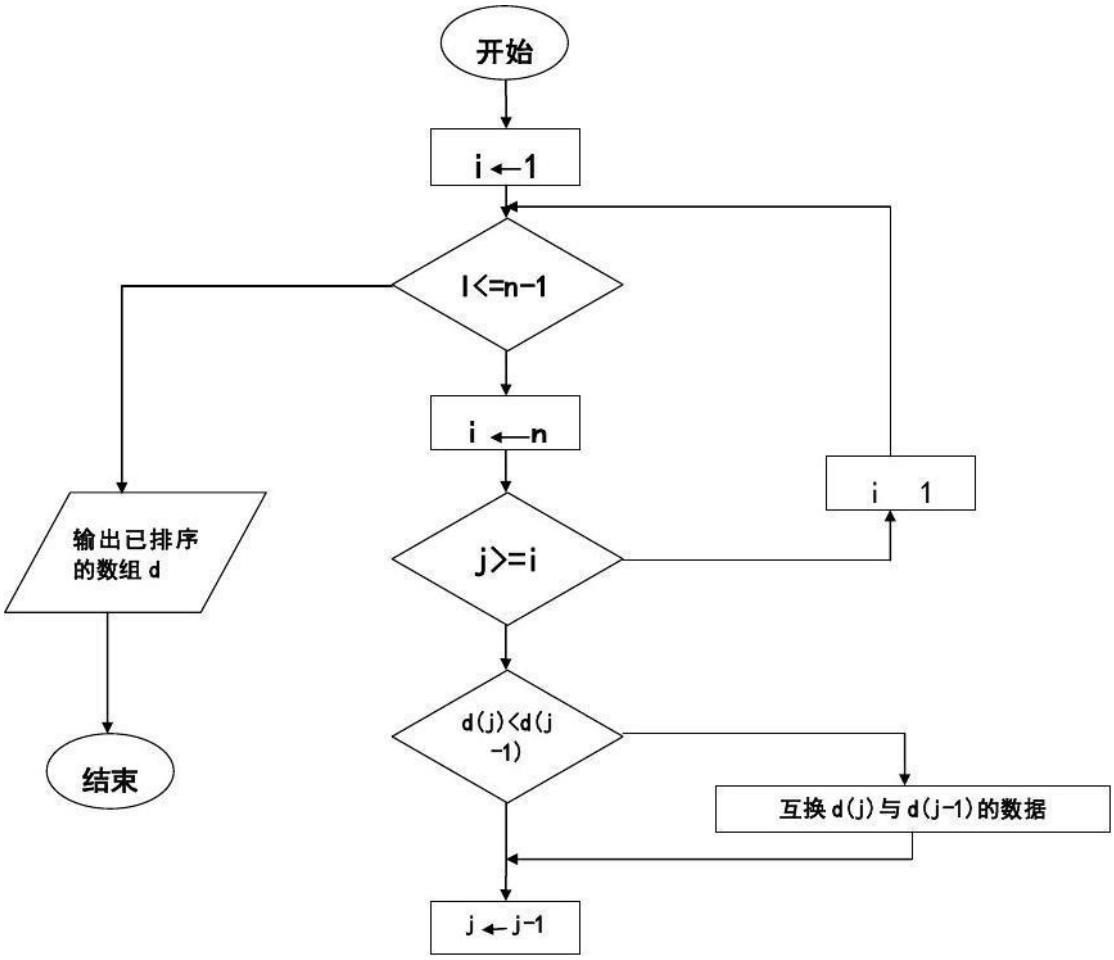
mov ebx, row_size

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。



3、 自己的一些摸索：由于 StdIn 只能读取字符串，不会读取整数，且无法直接放到数组中，需要将字符串先改成数字的形式（类似上一个实验中的将字符串转化为数字的过程），再存入数组中。故考虑采用别的输入输出方

式。在课下学到了在汇编中调用 C 函式进行输入输出（ctr_scanf 可以直接读取数字，加上字符串格式'%d'，0 即可进行多重数字的输入与读取）。

三、实验过程

1、指定处理器、指令集，指定内存模式，引入头文件，链接静态库文件。

```
.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
includelib \masm32\lib\msvcrt.lib            ;此处导入了C函式标准库
include \masm32\include\msvcrt.inc
```

2、定义数据段，进行相关数据的定义、声名和初始化。

```
.data
InputMsg db "Input 10 number(separated by space):",0 ;提示用户输入的消息字符串
OutMsg db "Number :",0                               ;提示输出排序的消息字符串

IntFormat db "%d",0
IntFormat1 db "%d",20h,0
count dword 10
UserInData  DWORD 10 DUP(0)
```

3、主代码部分书写（冒泡排序过程的定义）

```
BubbleSort PROC                 ; 定义冒泡排序过程
    mov ecx, count              ; 将未排序元素的数量存入ECX寄存器
    dec ecx                     ; 减小ECX寄存器的值，数组下标从0开始
L1:                             ; 外层循环开始
    push ecx                    ; 保存ECX寄存器的值
    mov esi, OFFSET UserInData  ; 将UserInData的地址存入ESI寄存器
L2:                             ; 内层循环开始
    mov eax, [esi]              ; 将当前元素的值存入EAX寄存器
    cmp [esi+4], eax            ; 比较当前元素和下一个元素的值
    jge L3                      ; 如果下一个元素大于等于当前元素，则跳
```

```
                                    转到L3
    mov ebx, [esi+4]            ; 将下一个元素的值存入ebx寄存器
    mov [esi+4], eax            ; 将当前元值存入下一个元素的位置
    mov [esi], ebx              ; 将下一个元素值存入当前元素的位置
L3:
    add esi, 4                  ; 更新esi寄存器的值,指向下一个元素
    loop L2                     ; 循环执行内层循环
    pop ecx                     ; 恢复ECX寄存器的值
    loop L1                     ; 循环执行外层循环
    ret                         ; 返回
BubbleSort ENDP                 ; 冒泡排序过程结束
```

4、运行部分

```
start:                                  ; 程序入口
    ;输出提示信息，获取用户输入的十进制数序
    invoke StdOut, ADDR InputMsg        ; 调用StdOut函数输出提示信息
    lea edi, UserInData                 ; 将UserInData的地址存入EDI寄存器
L5:
    invoke crt_scanf, addr IntFormat, edi    ; 调用crt_scanf函数获取用户
                                               输入的十进制数字
    mov eax,UserInData                       ; 将UserInData的地址存入eax
                                               寄存器
    add edi, 4                               ; 更新EDI寄存器的值，指向下
                                               一个元素
    dec count                                ; 减小未排序元素的计数
    jnz L5                                   ; 如果未排序元素的计数不为
                                               0，则继续循环

    ; 冒泡排序
    mov count, 10                            ; 将未排序元素的数量重
                                               置为10
    call BubbleSort                          ; 调用冒泡排序过程
    invoke StdOut, ADDR OutMsg               ; 输出提示信息
    mov count, 10                            ; 将未排序元素的数量重置为
                                                 10
    lea edi, UserInData                      ; 将UserInData的地址存入edi
                                               寄存器
L6:
    mov eax, [edi]                           ; 将当前元素值存入eax寄存器
    invoke crt_printf,addr IntFormat1, eax   ; 调用crt_printf函数输
                                               出当前元素的值
    add edi, 4                               ; 更新EDI寄存器的值，指向下
                                               一个元素
    dec count                                ; 减小未排序元素的计数
```

```
    jnz L6                                  ; 如果未排序元素的计数不为
                                              0，则继续循环

    invoke ExitProcess, 0                   ; 退出程序
END start                                   ; 程序结束
```

5、编译、链接、测试

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

 Assembling: Bubble.asm

D:\>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:\>Bubble.exe
Input 10 number(separated by space):1 3 5 77 890 324 676 234 9999 54
Number :1 3 5 54 77 234 324 676 890 9999
D:\>
```

成功实现从小到大的冒泡排序。

四、实验结论及心得体会

1、 心得体会：更清晰地看到了底层的细节，对计算机运行的底层逻辑有了更深入的理解，对高级语言的实现有了基础的认识。感受到了汇编语言高效率、自由的魅力。

2、 收获：对数组、寻址都有了更深入的理解与体会，同时自己寻找相关方法，对在汇编中调用 C 函式等技巧进行了学习，提升了自主学习能力。

3、 数组知识点总结：

1）dup操作符用于声明大型数组，包括需要初始化的数组和不要初始化的数组
2）寄存器esi和edi被称为源索引寄存器和目的索引寄存器。类似于指针。
3）在处理sdword类型的数组时，请注意偏移量是以4为单位（因为一个带符号双字节占用4个字节的内存空间）
4）带有offset操作符的mov和lea指令能够获取变量的地址，前者是静态的（编译时），后者是动态的（运行时）
5）把寄存器edi、esi和edi加上方括号，获取到的不是寄存器的内容，

而是他们指向的内存位置的内容
6）操作符lengthof用于计算数组的元素个数，操作符sizeof用于计算数组占用字节空间