南開大學

汇编语言与逆向技术课程实验报告

实验三：Bubble Sort

学　　院　网络空间安全学院
专　　业　信息安全
学　　号　2313546
姓　　名　蒋枘言
班　　级　信息安全班

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 操作系统，MASM32 编译环境。

三、实验原理

冒泡排序是通过重复地走访要排序的数列，一次比较两个相邻的元素，如果它们的顺序错误（即升序时前一个元素大于后一个元素，或降序时前一个元素小于后一个元素），则交换这两个元素的位置。这个过程会重复进行，直到数列中没有再需要交换的元素，也就是该数列已经排序完成。

四、实验过程

1.编辑：用编辑软件（记事本）形成源程序 bubble_sort.asm。

●代码解析

```
.386
.model flat, stdcall
option casemap:none
include \masm32\include\windows.inc
include \masm32\include\kernel32.inc
include \masm32\include\user32.inc
include \masm32\include\masm32.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\user32.lib
includelib \masm32\lib\masm32.lib

.data
numarr DWORD 10 DUP(?)
; 创建一个大小为 10 的 DWORD 数组 numarr
temp01 DWORD ?
temp02 DWORD ?
; temp01 和 temp02 用于移出寄存器 ecx 的值，以防 ecx 的值被篡改导致循环出现 bug
output01 BYTE "Please input 10 integers(0~10000):",0Dh,0Ah,0
output02 BYTE 0Dh,0Ah,"The sorted numbers are:",0Dh,0Ah,0
output03 BYTE " ",0

.code
main PROC
; 以下进行输入操作
    invoke StdOut, addr output01
    mov ecx, 10
    mov esi, 0
L1:
    mov temp01, ecx
    ; 移出寄存器 ecx 的值，以防 ecx 的值被篡改
    ; 但是我觉得不会被改啊啊！！
    ; 我也不知道为什么不这样做就会出现 bug
    invoke StdIn, addr [numarr+esi*4], 4
    ; 输入 10 个 DWORD 数据
    inc esi
    ; esi 寄存器作为数组索引
    mov ecx, temp01
    loop L1
```

```
; 以下进行冒泡排序操作
    mov ecx, 10
    dec ecx
    ; 设置循环次数，不要忘了是 10 减 1
LN1:
    push ecx
    ; 将 ecx 的当前值压入堆栈，以便在内层循环结束后恢复
    mov esi, 0
    ; 初始化索引寄存器 esi 为 0，用于数组 numarr 的索引
    dec ecx
    ; ecx 自减 1，用于内层循环的控制，因为内层循环需要比较到倒数第二个元素
    mov temp02, ecx
LN2:
    mov eax, [numarr+esi*4]
    inc esi
    ; esi 加 1，准备加载下一个元素，之后不要忘了减 1
    mov ebx, [numarr+esi*4]
    dec esi
    cmp eax, ebx
    jle LN3
    ; 比较 eax 和 ebx 的值，如果 eax 大于 ebx，则跳转到 LN3，否则继续执行交换操作
    inc esi
    xchg eax, [numarr+esi*4]
    dec esi
    mov [numarr+esi*4], eax
    ; 以上几行为交换操作
LN3:
    inc esi
    cmp esi, temp02
    ; 比较 esi 和 temp02 的值，如果 esi 小于 temp02，则继续内层循环
    jl LN2
    pop ecx
    ; 恢复 ecx 的值，准备下一次外层循环
    loop LN1

; 以下进行输出
    invoke StdOut, addr output02
    mov ecx, 10
    mov esi, 0
L2:
    mov temp01, ecx
    invoke StdOut, addr [numarr+esi*4]
    invoke StdOut, addr output03
    inc esi
    mov ecx, temp01
    loop L2
main ENDP
END main
```

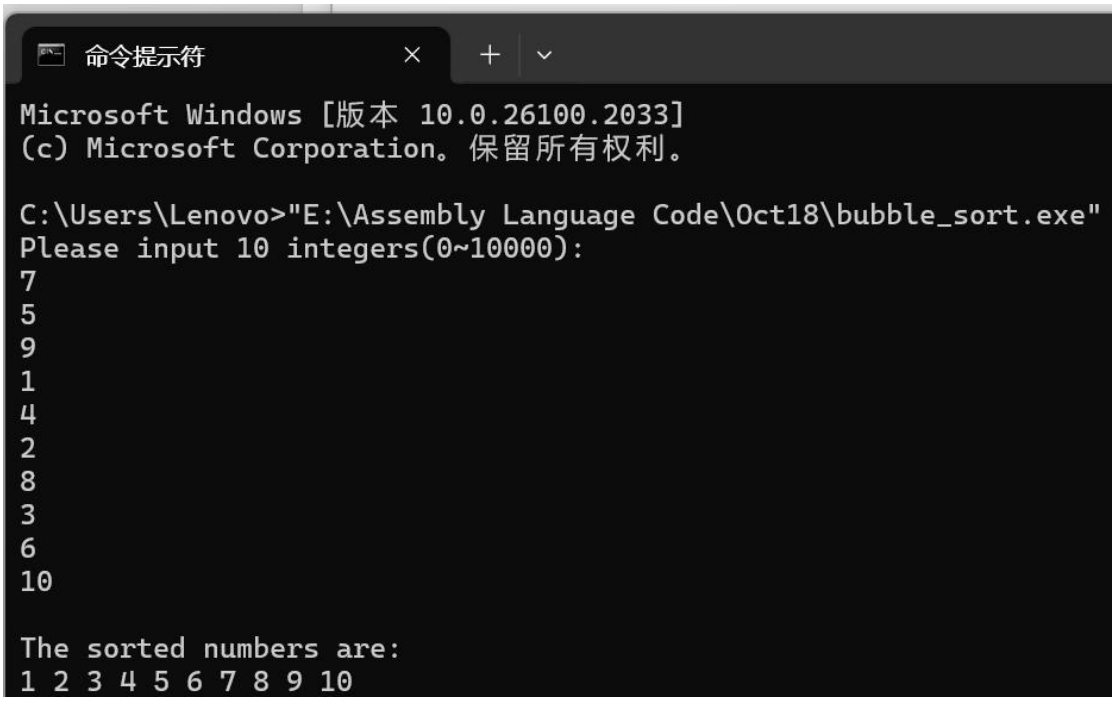
2.编译：用汇编程序（\masm32\bin\ml.exe）对源程序进行汇编，形成目标文件（.obj），格式如下：

“\masm32\bin\ml /c /Zd /coff bubble_sort.asm”

3.链接：用连接程序（\masm32\bin\link.exe）对目标程序进行连接，形成可执行文件（.exe），格式如下：

“\masm32\bin\Link /SUBSYSTEM:CONSOLE bubble_sort.obj”

4.执行：结果如下。



三、汇编语言数组操作知识点的总结

Date.

2024.10.1

1.整数常量 [{+|-}]数字[基数]　十进制 d　二进制 b　十六进制 h　八进制 o

☆以字母开头的十六进制常量前面必须加1个0，以区分标识符。

FF3h ✗　0FF3h ✓

2.整数表达式：包含整数值和算术运算符

( )
* / MOD
+ -
（高→低 优先级）

3.实数常量 [{+|-}]整数.[整数][E[{+|-}]整数]
（十进制）　小数点必须有　指数

4.字符常量 单引号或双引号括起来 如'A' "B"

☆汇编器将其转为ASCII码。

5.字符串常量 单引号或双引号括起来的一串字符 如'ABC' "abc"

引号可以嵌套。

6.保留字

①指令助记符：MOV ADD　④运算符
②伪指令 INCLUDE PROC　⑤预定义符号 @data
③属性 BYTE WORD

2.定义字符串：以0结尾，每个字符占1个字节

str BYTE "HELLO",0　换行 str BYTE "Hello World",0Dh,0Ah,0

或 str BYTE 'H','E','L','L','O',0

3.（无符号 WORD DWORD
有符号 SWORD SDWORD

4.DUP伪指令：为字符串或数组分配内存空间

BYTE 20 DUP(0)

7.标识符：用来标识变量、常量、过程或代码标号。

☆包含1-247个字符。

☆大小写不敏感。

☆第一个字符必须是字母、下划线、@、?或$。

☆不能与保留字相同。

8.指令：汇编语言中的指令是一条汇编语句。

汇编过程：汇编器把汇编指令翻译成机器指令。

格式 [标号:] 指令助记符 操作数 [;注释]

mov add sub mul jmp call

操作数：寄存器、内存、常量、I/O端口

注释：单行注释 分号开始；块注释 COMMENT !

9.标号：充当指令或数据位置标记的标识符

①数据标号：标识了变量的地址

count DWORD 100

相对.data数据段在内存起始位置的偏移

②代码标号：以冒号(:)结尾

10.NOP指令：空指令，占用1个字节内存

用于后继指令对齐。（IA-32处理器从偶数双字地址加载代码和数据更快）

11.伪指令：内嵌在源代码中，由汇编器识别并执行，在运行时不执行。

2024.10.3

1.定义数据 [变量名] 数据定义伪指令 初始值[,初始值…]

如：my_var DWORD 0,1,2,3

var DWORD ?

伪指令	含义
BYTE	8bit DB
WORD	16bit DW
DWORD	32bit DD

2024.10.19

1.MOV指令　不能为CS,EIP和IP

mov reg,reg　（目的操作数 源操作数 ⇒尺寸必须一致）
mov reg,mem
mov reg,imm
mov mem,reg
mov mem,imm
mov mem,mem

mov 段寄存器,立即数

2.movzx指令：零扩展传送，适用于无符号整数

BYTE 4 DUP ("Hello") → 20字节

4个"Hello" 1个"Hello"5字节

5、符号常量：不占用存储空间。

6、等号伪指令

COUNT = 500 优点：易于维护

mov eax, COUNT

7、计算数组和字符串大小

8、返回当前语句地址偏移值

List BYTE 10,20 } 次次紧跟

ListSize = ($ - list)

8、EQU伪指令：将符号名与整数表达式或任意文本联系起来

name EQU { expression / symbol / <text>

MyCons EQU 100 ;定义MyCons为常量100

;不会分配实际内存空间

press_key EQU <"Press any key to continue", 0>

EQU不允许重定义。"="允许。

9、TEXTEQU伪指令：与EQU相似，也可创建文本宏

TEXTEQU可以重定义。

message EQU <"Hello"> 等价于 message TEXTEQU <"Hello">

message2 TEXTEQU <message>

TEXTEQU还可定义整数常量表达式，"%"符号后跟表达式

size TEXTEQU %(1024*8)

被定义为8192

6、inc指令：操作数加1 inc reg

dec指令：操作数减1 inc mem

7、ADD指令：尺寸相同的操作数相加，结果储存在目的操作数

SUB指令：将源操作数从目的操作数中减掉。

add reg,reg

add reg,mem

add mem,reg

add mem,imm

add reg,imm

add mem,mem

sub与add相同

8、NEG指令：将数字转换成补码求相反数

neg reg/mem

例：.data

var DWORD 1000h 二进制数 0000 0000 0000 0000 0001 0000 0000 0000

.code

neg var

取反 1111 1111 1111 1111 1110 1111 1111 1111

加1 1111 1111 1111 1111 1111 0000 0000 0000

即0FFFFF000h

var为0FFFFF000h

计算var1-var2

move eax,var1

sub eax,var2

等价

neg var2

mov eax,var1

add eax,var2

2024.10.20

1、OFFSET操作符：返回数据标号的偏移地址。

偏移量或于为32位(64字节)(实地址模式下为16位)

代表标号距离数据段开始的距离。

若CS为0，则等同于虚拟地址

2024.10.25

1、JMP指令

L1:

...

jmp L1 } 无限循环

其他：JZ 为0则跳转 JNZ 不为0则跳转

2、LOOP指令：ecx为寄存器，执行完LOOP自动减1，接着JNZ

mov ecx,5

L1:

inc ax

loop L1 } 循环5次

LOOP循环嵌套：移出当前ecx值并重新赋值

mov ecx,100

L1:

mov count,ecx

mov ecx,10

L2:

...

loop L2

mov ecx,count

loop L1

例：将字符串src复制到dst

.data

src BYTE "Hello World",0Dh,0Ah,0

dst BYTE SIZEOF src DUP(0),0

.code

mov ecx, SIZEOF src ;循环次数

mov esi, 0 ;字符索引

L1:mov al, BYTE PTR src[esi]

mov BYTE PTR dst[esi],al

inc esi

loop L1

movzx 大,小 高位补0

3、movsx指令：符号扩展传送，用于有符号整数

movsx 大,小 小的最高位循环填充大的高位

例：(1).data

var1 BYTE 10h

.code 实际二进制：0001 0000

movzx eax,var1 2最高位(符号位)为0(负)

movsx ebx,var1

高位补0

eax为00000010h

ebx为00000010h 用0扩充

(2).data

var1 BYTE DAOh

.code 实际二进制：1010 0000

movzx eax,var1 2最高位(符号位)为1(负)

movsx ebx,var1

高位补0

eax为0000 00A0h

ebx为FFFF FFA0h 用F扩充

F开头，用0补全

movsx与movzx目的操作数都是寄存器。

4、LAHF指令：将EFLAGS寄存器的低字节复制到AH寄存器。

包括SF、ZF、AF、PF、CF

SAHF指令：复制AH的值至EFLAGS寄存器低字节。

5、XCHG指令：交换两个操作数

XCHG reg,reg

XCHG reg,mem

XCHG mem,reg

XCHG mem,mem

2、ALIGN伪指令：将变量的位置按字节、字、双字或段边界对齐。

格式：align 边界值

(可为1,2,4或16)

CPU处理偶数地址比奇数地址快！

3、PTR操作符：重载操作数声明的默认尺寸

.data

myDouble DWORD 12345678h

.code

mov ax,WORD PTR myDouble ;mov ax,myDouble不被允许

ax为5678h

例：.data

var1 DWORD 12345678h

.code

movzx eax,BYTE PTR var1

eax为00000078h

1字节 BYTE

78 56 34 12

地址值小→大

var1

.data

var1 WORD 1234h

var2 WORD 5678h

.code

mov eax,DWORD PTR var1

eax为 56781234h

34 12 78 56

地址值小→大

var1 var2

Intel处理器使用小尾顺序存储，即变量的最低有效字节储存在地址值最小的储存单元。

4、TYPE操作符：返回变量字节数

LENGTHOF操作符：计算数组中元素数目(元素出现在同一行的值定义)

SIZEOF操作符：返回TYPE和LENGTH的乘积

5、LABEL伪指令：赋予标号尺寸属性(实际未分配任何空间)

例 .data

var是var1的别名 var1 WORD 1234h

var2 WORD 5678h

.code

mov eax,var

eax为 56781234h

地址值低→高

34 12 78 56

DWORD