指令系统作业

2018 信息安全 2 班-20184347-王浩

**2. 交换数组元素对

编写循环程序，用变址寻址交换数组中的数值对，每对中包含偶数个元素。即，元素 i 与元素 i+1 交换，元素 i+2 与元素 i+3 交换，以此类推。

```
; 2. 交换数组元素对（变址寻址）
.386
.model flat,stdcall
.stack 4096

include kernel32.inc
includelib kernel32.lib

.data
list    DWORD 1, 2, 3, 4, 5, 6, 7, 8

.code
main PROC
    mov     eax, 0                      ; 偶数位变址寄存器
    mov     ebx, 1                      ; 奇数位变址寄存器
    mov     ecx, (LENGTHOF list/2)      ; 只用循环数组长度的一半
L1:
    mov     edx, list[eax*TYPE list]
    xchg    edx, list[ebx*TYPE list]    ; 交换奇偶位置的元素
    mov     list[eax*TYPE list], edx
    add     eax, 2                      ; 修改变地址
    add     ebx, 2                      ; 修改变地址
    loop    L1
    INVOKE ExitProcess,0
main ENDP
END main
```

执行前：

Address	Hex dump	ASCII
00403000	01 00 00 00 02 00 00 00 03 00 00 00 04 00 00 00	□ □ □ □
00403010	05 00 00 00 06 00 00 00 07 00 00 00 08 00 00 00	□ □ □ □

执行后：

Address	Hex dump	ASCII
00403000	02 00 00 00 01 00 00 00 04 00 00 00 03 00 00 00	□ □ □ □
00403010	06 00 00 00 05 00 00 00 08 00 00 00 07 00 00 00	□ □ □ □

可以看到奇数位和偶数位已经发生了交换。

**3. 数组元素间隔之和

编写循环程序，用变址寻址计算连续数组元素的间隔总和。数组元素为双字，按非递减次序排列。比如，数组为 {0，2，5，9，10}，则元素间隔为 2、3、4 和 1，那么间隔之和等于 10。

```
; 3. 元素间隔之和（变址寻址）
.386
.model flat,stdcall
.stack 4096

include kernel32.inc
includelib kernel32.lib

.data
list    DWORD 0, 2, 5, 9, 10
sum DWORD 0

.code
main PROC
    mov     esi, 0                          ; 变址寄存器
    mov     ecx, (LENGTHOF list-1)          ; 循环次数
L1:
    mov     eax, list[esi*SIZE list]
    inc     esi
    mov     ebx, list[esi*SIZE list]
    sub     ebx, eax                        ; 求间隔
    add     sum, ebx
    loop    L1

    mov eax, sum
    INVOKE ExitProcess,0
main ENDP
END main
```

执行结果如图：

Address	Hex dump												ASCII	
00403000	00	00	00	00	02	00	00	00	05	00	00	00	09	00	00	00	□	□
00403010	0A	00	00	00	0A	00	00	00	00	00	00	00	00	00	00	00		

[0, 2, 5, 9, 10] 间隔之和为 10 计算正确。

**5. 斐波那契数列

编写循环程序，计算斐波那契（Fibonacci）数列前七个数值之和，算式如下：

$Fib(1)=1$，$Fib(2)=1$，$Fib(n)=Fib(n-1)+Fib(n-2)$

```
; 斐波那契数列前七个数之和
.386
.model flat,stdcall
.stack 4096

include kernel32.inc
includelib kernel32.lib
.data
sum     DWORD 2         ; 前两个斐波那契数之和

.code
main PROC
    mov     eax, 1          ; Fib(1)
    mov     ebx, 1          ; Fib(2)
    mov     ecx, 5          ; 循环 5 次求到第七个斐波那契数
L1:
    add     eax, ebx        ; 下一个斐波那契数
    add     sum, eax
    xchg    eax, ebx
    loop    L1
    INVOKE ExitProcess,0
main ENDP
END main
```

执行结果如图：

Address	Hex dump	ASCII
00403000	21 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	!

斐波那契数列前七项之和为 1+1+2+3+5+8+13=33=21h 计算正确。

***8. 数组元素移位

编写循环程序，用变址寻址把一个32位整数数组中的元素向前（向右）循环移动一个位置，数组最后一个元素的值移动到第一个位置上。比如，数组[10，20，30，40]移位后转换为[40，10，20，30]。

```
; 8. 数组元素移位（变址寻址）
.386
.model flat,stdcall
.stack 4096

include kernel32.inc
includelib kernel32.lib

.data
list    DWORD 10, 20, 30, 40

.code
main PROC
    mov     esi, LENGTHOF list-1
    mov     eax, list[esi*SIZE list]
    mov     edx, eax                    ; 备份最后一个元素
    mov     ecx, LENGTHOF list-1        ; 循环次数
L1:
    dec     esi
    mov     eax, list[esi*SIZE list]
    mov     list[ecx*SIZE list], eax    ; 前一个元素移到后一个位置
    loop    L1
    mov     eax, edx
    mov     list, eax                   ; 最后一个元素放在第一个位置
    INVOKE ExitProcess,0
main ENDP
END main
```

执行前：

Address	Hex dump	ASCII
00403000	0A 00 00 00 14 00 00 00 1E 00 00 00 28 00 00 00	▯ (

执行后：

Address	Hex dump	ASCII
00403000	28 00 00 00 0A 00 00 00 14 00 00 00 1E 00 00 00	(▯

可以看到已经循环右移了一个单元。

8. 编写指令序列，把三个内存字节左移一位，使用数据如下：

```
wordArray WORD 810Dh, 0C064h,93ABh
```

```
; 6 个字节左移一位
.386
.model flat,stdcall
.stack 4096

include kernel32.inc
includelib kernel32.lib
.data
wordArray WORD 810Dh, 0C064h, 93ABh

.code
main PROC
    mov     esi, 0
    shl     BYTE PTR wordArray[esi], 1    ; 第一个字节移位不需要管溢出
    mov     ecx, SIZEOF wordArray-1       ; 循环次数 5
L1:
    mov     ebx, esi
    inc     esi
    shl     BYTE PTR wordArray[esi], 1
    lahf                                  ; 加载标志位
    and     ah, 01h                       ; 掩码提取出 CF 标志
    add     BYTE PTR wordArray[ebx], ah   ; 将 CF 标志加到前一个字节的末尾
    loop L1
    INVOKE ExitProcess,0
main ENDP
END main
```

执行前：

Address	Hex dump	ASCII
00403000	0D 81 64 C0 AB 93 00 00 00 00 00 00 00 00	dÀ«“

执行后：

Address	Hex dump	ASCII
00403000	1B 02 C9 81 57 26 00 00 00 00 00 00 00 00	▯▯É W&

解释：

要把内存中的六个字节整体左移一位除了需要考虑按字节分别进行移位以外，如果后一个字节移出的最高位为 1 还需要将其补到前一个字节的最低位（第一个字节除外），因此需要利用到进位标识位 CF 进行判断。

执行前 0D 81 64 C0 AB 93 二进制为 00001101 10000001 01100100 11000000 10101011 10010011，执行后 1B 02 C9 81 57 26 二进制为 00011011 00000010 11001001 10000001 01010111 00100110，对比前后可以看到实现了 6 个字节整体的左移一位。

11. 使用 32 位无符号操作数，用汇编语言实现下述 C++ 表达式：

```
val1 = (val2 * val3) / (val4 - 3)
```

```
; 11. 汇编语言实现 C++表达式 val1 = (val2 * val3) / (val4 - 3)
.386
.model flat,stdcall
.stack 4096

include kernel32.inc
includelib kernel32.lib

.data
val1    DWORD 1
val2    DWORD 2
val3    DWORD 3
val4    DWORD 4

.code
main PROC
    mov     eax, val2
    mul     val3
    mov     ebx, val4
    sub     ebx, 3
    div     ebx
    mov     val1, eax
    INVOKE ExitProcess,0
main ENDP
END main
```

执行结果如图：

Address	Hex dump												ASCII			
00403000	06	00	00	00	02	00	00	00	03	00	00	00	04	00	00	00	□	□	□	□

可以看到当 val2=2, val3=3, val4=4 时 val1 = (val2 * val3) / (val4 - 3)=6 计算正确。