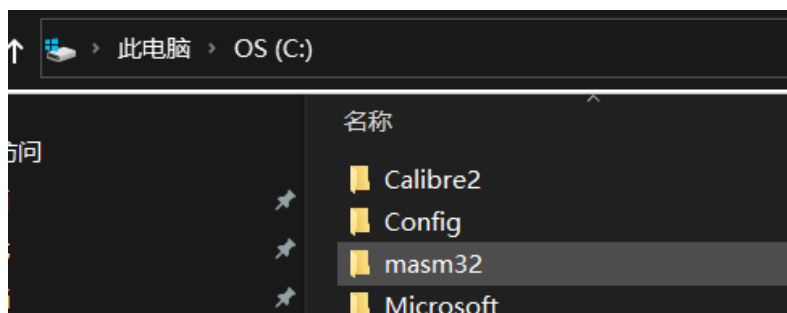


《汇编语言》实验报告 06

班级	2022 秋	实验日期	2022.12.11	实验成绩	
姓名	黄勳	学号	22920212204392		
实验名称	汇编语言第六次实验				
实验目的、要求	<p>1) 汇编指令综合应用</p> <p>2) 熟悉 32 位 Intel 汇编指令</p> <p>3) 学习使用汇编语言编写冒泡排序算法</p> <p>ps:</p> <p>4) 在本次实验中我还实现了输入字符串，并将输入串数据转化成数组保存，并进行排序，最后将数据输出</p>				
实验内容、步骤及结果	<p>1、请使用 32 位的 Intel x86 的指令，编写计算冒泡排序算法的程序（从小到大排序、从大到小排序）；并在 32 位的 Intel x86 汇编语言环境下运行通过。</p> <p>数据原顺序：7,5,3,2,6,9,1,8,4,0</p> <p>从小到大和从大到小都需要输出，可以放在两个程序里分别运行。</p> <p>编写过程：</p> <p>1) 实验环境设置</p> <p>解压实验发的编译器压缩包，运行 install.exe 文件，安装在 C 盘。</p> <div></div> <p>安装结束后，在 C 盘目录下会产生一个 masm32 的文件夹</p>				



正确配置系统环境变量即可在其他目录进行编译连接操作。

2) 开始编写代码，先声明汇编与链接库

```
1  .386 ;允许汇编80386处理器的非特权指令，禁用其后处理器引入的汇编指令
2  .model flat, stdcall ;.model用于初始化程序的内存模式
3  option casemap :none ;大小写敏感
4  include C:\masm32\include\windows.inc
5  include C:\masm32\include\masm32.inc
6  include C:\masm32\include\kernel32.inc
7  includelib C:\masm32\lib\kernel32.lib
8  includelib C:\masm32\lib\masm32.lib ;函数的常量和链接库声明
9  .stack 4096 ;栈空间声明
```

3) 声明数据段

```
10  .data
11      str1 byte "Please input unsigned int number: ",0 ;提示输入数组(以空格分
12      str2 byte "The result of bubble sort is: ",0 ;提示输出数组(以空格分割)
13      inputstr byte 80 dup(0) ;存储输入的字符串
14      pArray dword 15 dup(0) ;存储字符串转化成的各个数字
15      outputstr byte 80 dup(0) ;存储输出的字符串
16      const10 dword 10 ;常数10
17      count dword 1 ;数组长度
18      count2 dword 0 ;用于保存ecx
19  .code
```

4) main 过程编写

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。

```

21  main proc ;main过程
22      invoke StdOut,addr str1 ;输出提示
23      invoke StdIn,addr inputstr,60 ;输入数组
24      call strtonum ;输入字符串转化为数组
25      call BubbleSort ;数组冒泡排序
26      call numtostr ;排序后的数组重新转化为可输出的字符串
27      invoke StdOut,addr str2 ;输出提示
28      invoke StdOut,addr outputstr ;输出排序后数组
29      invoke ExitProcess,0
30  main endp

```

5) 处理数据

```

32  strtonum proc uses esi eax ebx ecx ;输入字符串转化为数组
33      mov esi,0
34      mov ebx,0
35      mov ecx,0
36  L1:
37      mov bl,[inputstr+esi] ;以esi为变址寄存器读取输入字符串
38      cmp bl,32 ;判断是否读取到空格(ASCII码值为32)
39      jne L2
40      inc count
41      add ecx,4 ;是空格则count+1,同时ecx+4
42      inc esi
43      mov bl,[inputstr+esi] ;略过空格
44  L2:
45      sub bl,48 ;字符转化为数字
46      mov eax,[pArray+ecx] ;读取数字
47      mul const10 ;eax值乘10
48      add eax,ebx ;加上刚读取的数字
49      mov [pArray+ecx],eax ;存回
50      inc esi
51      cmp [inputstr+esi],0 ;读取到串尾0停止
52      jne L1
53      ret
54  strtonum endp

```

6) 编写冒泡排序

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。

```
56  BubbleSort proc uses eax ecx esi ;数组冒泡排序
57      mov ecx,count
58      dec ecx ;令ecx等于数组长度-1
59  L3:
60      push ecx ;入栈,保存外层ecx
61      mov esi,0
62  L4:
63      mov eax,[pArray+esi]
64      cmp eax,[pArray+esi+4] ;比较数
65      jbe L5
66      xchg eax,[pArray+esi+4]
67      mov [pArray+esi],eax ;若前一个数比后一个大则交换
68  L5:
69      add esi,4
70      loop L4 ;内层循环
71      pop ecx ;出栈,恢复外层ecx
72      loop L3 ;外层循环
73      ret
74  BubbleSort endp
```

7) 排序后的数组重新转化为可输出的字符串

```

76  numtostr proc ;排序后的数组重新转化为可输出的字符串
77      mov ecx,count
78      dec ecx
79      mov esi,0
80  L10:
81      add esi,4
82      loop L10 ;使esi=4*(count-1),即偏移为最后一个数字
83      mov edi,0
84      mov edx,0
85      mov ecx,count
86  L6:
87      mov count2,ecx ;保存ecx
88      mov eax,[pArray+esi] ;获取数字
89      sub esi,4 ;移动到前一个数字
90  L7:
91      div const10 ;除法指令,被除数为edx:eax,eax存商,edx存余数
92      mov ecx,edx
93      mov edx,0
94      add ecx,48 ;数字转化为字符
95      push ecx ;字符入栈
96      inc edi
97      cmp eax,0
98      jne L7
99      push 32 ;商为0则入栈一个空格
100     inc edi
101     mov ecx,count2 ;恢复ecx
102     loop L6
103
104     mov ecx,edi ;循环次数为输出字符串长度
105     mov esi,0
106  L8:
107     pop eax ;循环出栈
108     mov [outputstr+esi],al ;出栈字符保存到输出字符串中
109     inc esi
110     loop L8
111     mov [outputstr+esi],0 ;补充串尾0
112     ret
113 numtostr endp

```

8) 根据编写好的从小到大排序的 asm 代码,将冒泡排序函数中起比较作用的 jbe 改成 jae,即可得到从大到小排序的代码

9) 编译链接

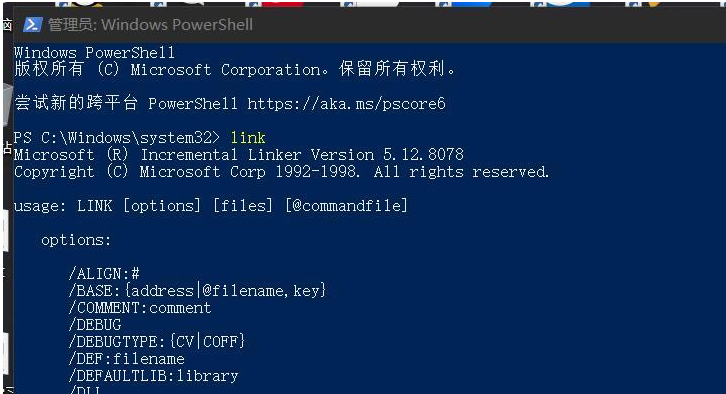
我在此使用 Windows Powershell 控制台,cd 到本实验目录,使用 ml 和 link 程序将源代码编译、链接成可执行文件 bubble_sort_xxx.exe。完成后即可使用。

使用的命令:

```
ml.exe -c -coff .\bubble_sort_asc.asm
```

```
link -subsystem:console .\bubble_sort_asc.obj
```

另一个程序与此类似

	
总结	<p>这一次实验我对汇编语言指令有了更深的理解，并且这一次实验的实践操作颇丰，在练习编码的过程中我加深了 32 位指令的编写的操作的熟练度，我对每一个指令的用途和用法有了更深的认识；通过一步步地解决问题，我的实践能力提高了，这让我受益匪浅；具体遇到问题的解决方案我在上文做了更详细的总结，在此就不多赘述；在未来我还要探索汇编语言的更多应用方面，寻找更多问题，并在发现问题的过程中继续提高我对汇编语言的掌握能力，这是一次颇有意义的实验！</p> <p>算法及其实现方式总结：</p> <p>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。</p> <p>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 <code>mov eax, [ebx + esi]</code>。在例子中，<code>ebx</code> 是基址寄存器，<code>esi</code> 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。</p> <p>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：</p> <pre>table dword 10h, 20h, 30h, 40h row_size = (\$ - table) dword 50h, 60h, 70h, 80h</pre>

	<pre> dword 90h, 0a0h, 0b0h, 0c0h mov ebx, row_size mov esi, 2 mov eax, table[ebx + esi * 4]</pre> <p>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。</p> <p>知识点总结：</p> <ol style="list-style-type: none">1、交换指令 xchg。xchg 指令交换两个操作数的内容，但不能直接交换两个内存的内容，可用于数组内的交换。2、循环指令 loop。Loop 以 ecx 为循环计数器进行循环，可用于遍历数组3、判断指令 cmp，条件跳转指令，无条件跳转。 Cmp 指令通过修改 cpu 的标志位达到比较的目的，通常和 je,jne,ja,jb,jg,jl 等条件跳转配合使用，以及无条件跳转 jmp 指令。4、间接寻址—变址操作数、基址变址操作数。 形如[<code>eax + array1</code>]的操作数便称为变址操作数，最常用于遍历数据。 形如[<code>ebx+esi</code>]的操作数称为基址变址操作数，可用于访问二维数组5、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。6、寄存器 esi 和 edi 是常用的变址寄存器。他们类似于指针，对字符操作非常有用。7、在处理 dword 类型的数组时，偏移量是以 4 为单位，而不是 1，因为一个带符号双字节占用 4 个字节的内存空间8、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
--	---