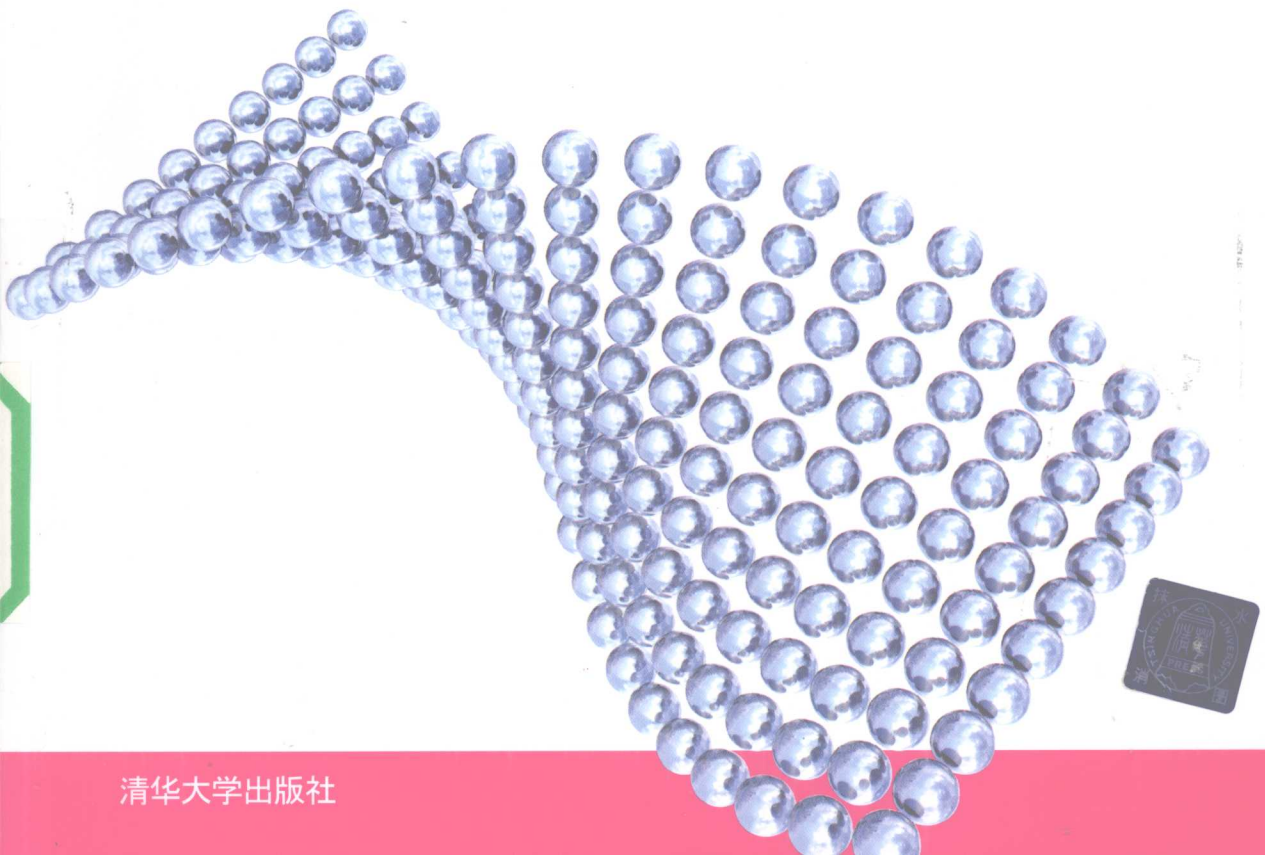
Windows 汇编语言

程序案例解析

戴水贵 童爱红 俞海英 冯小明 编著



清华大学出版社

Windows 汇编语言程序案例解析

戴水贵 童爱红 俞海英 冯小明 编著

清华大学出版社
北京

内 容 简 介

本书结合完整的程序实例讲解 Windows API 函数的使用方法，程序中加有很多汉字注释，这样读起来更舒服，更能使读者有整体概念，并且学了就会用。书中每个程序都给出运行结果，这样有利于理解程序。学完本书后，会了解 Windows 操作系统中的一些奥秘。

本书内容包括 Windows 汇编语言程序设计基础知识，文件管理，直接访问硬盘，内存管理和命令行参数，Windows 图形界面编程，时钟中断编程，图形操作，窗口和键盘输入，动态链接库程序。

本书对已学过 DOS 汇编语言，并想过渡到 Windows 汇编语言的读者来说是一本好书。

图书在版编目（CIP）数据

Windows 汇编语言程序案例解析 / 戴水贵等编著. —北京：清华大学出版社，2009.6
ISBN 978-7-302-19934-2

Ⅰ. W…　Ⅱ. 戴…　Ⅲ. 汇编语言-程序设计　Ⅳ. TP313

中国版本图书馆 CIP 数据核字（2009）第 058587 号

责任编辑：闫红梅
责任校对：焦丽丽
责任印制：杨　艳
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　　总　　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京市清华园胶印厂
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：27.5　**字　数**：670 千字
版　　次：2009 年 6 月第 1 版　　**印　　次**：2009 年 6 月第1 次印刷
印　　数：1～3000
定　　价：39.00 元

前　言

一提起汇编语言，人们的感觉就是麻烦加难学。其实，Windows 环境下的汇编语言与高级语言已非常接近，且省去了高级语言中的条条框框，编译、链接及程序排错更方便。汇编语言更接近操作系统，在汇编语言中使用系统函数更方便，学习 Windows 汇编语言更能了解操作系统的运行细节，而且用 Windows 汇编语言同样可以开发大型应用软件。

本书的读者对象是学过 DOS 汇编语言的人。本书不再讲解汇编语言基本语法（仅讲解一些高级语法），而是给出一个个完整的程序例子，让用户在轻松的环境下掌握书中的内容。Windows 提供的 API 函数非常多，本书不求多，只求懂一些常用方法。

一个完整的程序实例胜过一大篇文字说明，正因为如此，作者将调试通过的程序汇编成册供大家共享。为了结合程序理解 Windows API 函数的使用方法，程序中加有很多汉字注释，这种在程序中加注释的方法，读起来更舒服，更能使读者有整体概念，使读者学了就会用。书中每个程序都给出运行结果，这样有利于读者理解程序。

如何从 DOS 环境下的汇编语言编程转到 Windows 环境下的汇编语言编程，是一个急待解决的问题。目前，这方面的书比较少。Windows 汇编语言的书不太好写，原因是 Windows 图形界面下的程序都比较长。本书尽量选择一些小程序。

由于 Windows 操作系统提供的 API 函数很多，用户在阅读程序时要不断翻阅或查找很多资料，这显得不太方便。为此，本书在程序中使用更多的注释。通过一个个小而完整的程序，边读边上机调试，以加深对各类 API 函数的理解。

为了便于初学者循序渐进地学习，书中程序从小到大，注释从多到少（因为程序中有很多东西是可以重复引用的）。特别提醒初学者，在学习图形界面编程时，不要被第一个长程序吓倒，因为程序中的大部分内容是图形界面程序的构架，是可以重复使用的。

Windows 环境下有控制台编程和图形界面编程之分，控制台编程是面向过程的编程，与 DOS 环境下的编程类似。图形界面编程是面向对象的编程，初学者有一定的难度。本书从控制台编程开始，因为它与 DOS 环境下的汇编语言编程相似，容易让初学者入门。

对已学过 DOS 汇编语言，并想过渡到 Windows 汇编语言的读者来说，本书是一本好书。

书中第 1 章为 Windows 汇编语言程序设计基础知识，第 2 章为文件管理，第 3 章为直接访问硬盘，第 4 章为内存管理和命令行参数，第 5 章为 Windows 图形界面编程，第 6 章为时钟中断编程，第 7 章为图形操作，第 8 章为窗口和键盘输入，第 9 章为动态链接库程序。

书中所有程序都能在清华大学出版社的网站（www.tup.com.cn）上找到，下载后每个文件夹下都有编译链接批命令文件 mlexe.bat 和编译链接所需要的所有资源。读者计算机只

要装有 VC 6.0，并配好编译链接环境变量（具体配置方法见书中有关章节），就可将书中源代码编译链接成可执行程序（下载资料中有可执行程序）。

为了方便读者，下载资料中的 MASM32 文件夹下包含书中所有的头文件（.inc）和库文件（.lib），用户只要将其复制到 C:\就可以使用。

编　者

2008 年 12 月

目　录

第 1 章　Windows 汇编语言程序设计基础

Windows 汇编语言程序分为控制台编程和图形界面编程两种，控制台编程相对简单一些。为了由浅入深，本书从控制台编程开始讲解。

读者总希望用最快的速度掌握书中的概貌，为此从一个最简单的程序开始。一些汇编语言语法也结合程序进行讲解，有些指令和语法用注解的方法说明。

1.1　第一个完整的 Windows 汇编语言程序

Windows 汇编语言程序有自己的编程规范，它的编程规范比 Visual C 要简单得多，调试也很方便。更重要的是系统把重要的东西都呈现给读者，使读者更能掌握其中的本质。

用一条一条的汇编语言指令很难写出大程序，Windows 汇编语言程序也是调用系统提供的 API 来写程序。因而，用 Windows 汇编语言同样可写出大程序。以下是一个最简单的 Windows 程序。

```
;程序功能：显示一个信息框。
;ex1.asm(e:\masm\base)        ;程序名
;编译链接方法：
;ml /c /coff  ex1.asm
;link /subsystem:console ex1.obj
    .386                    ;指明指令集
    .model flat,stdcall ;程序工作模式，flat为Windows程序使用的模式(代码和数据
                            ;使用同一个4GB段),stdcall为API调用时右边的参数先入栈
    option casemap:none ;指明大小写敏感

include windows.inc
include user32.inc
includelib user32.lib
include kernel32.inc
includelib kernel32.lib

        .data               ;数据段
szCaption   db  '抬头串',0
szText      db  'Hello! ',0

        .code               ;代码段
start:
```

```
invoke MessageBox,          ;显示信息框
    NULL,                   ;父窗口句柄
    offset szText,          ;正文串的地址
    offset szCaption,       ;抬头串的地址
    MB_OK                   ;按钮

invoke ExitProcess,         ;终止一个进程
    NULL                    ;退出代码

end start                   ;指明程序入口点
```

图 1-1

程序运行结果见图 1-1。

说明：程序调用了两个 Windows 提供的 API。invoke 是汇编语言中的伪指令，该指令的使用方法见 1.4 节。

1.2 编译、链接和运行

1.2.1 创建编译链接环境

（1）安装 MASM615 调试工具。

（2）建立一个 VAR.BAT 文件，内容如下。

```
@echo off
rem 请根据 Masm32 软件包的安装目录修改下面的 Masm32Dir 环境变量!
set Masm32Dir=c:\Masm32
set include=%Masm32Dir%\Include;
c:\Program Files\Microsoft Visual Studio\VC98\Include;(本行应接在上行后)
Program Files\Microsoft Visual Studio\VC98\MFC\Include;(本行应接在上行后)
%include%(本行应接在上行后)
set lib=%Masm32Dir%\lib;%lib%
set w2k=%Masm32Dir%\Incluse\w2k;%Include\w2k%
set path=%Masm32Dir%\Bin;%Masm32Dir%\Include;%Masm32Dir%\Include\w2k;
%Masm32Dir%\lib;%Masm32Dir%;%PATH%(本行应接在上行后)
set Masm32Dir=
echo on
```

编译链接程序前，需要切换到命令提示符方式，并运行该文件（设置好环境），然后方可进行编译链接。

1.2.2 编译链接和运行

以下以编译链接 ex1.asm 为例：

（1）编译。

```
ML /Zi /c /Fl /coff ex1.asm
```

ML 参数说明（注意参数大小写）：

```
/Zi -- 加符号调试信息
/c -- 连接前的编译
/Fl -- Fl[file]产生列表文件
/coff -- 产生COFF格式目标文件
```

编译的更多参数说明，可用命令 ML /?查阅。

（2）链接。

```
LINK /SUBSYSTEM:console ex1.obj
```

其中 console 指明是控制台编程，如果是 Windows 窗口编程，则将 console 改为 Windows。

（3）运行。

在 Windows 下双击 ex1.exe 或在 DOS 命令提示符下键入 ex1 回车。

1.2.3　建立编译链接批命令文件

可以把编译链接过程写成批命令文件，以减少键盘输入量。例如：

```
MLEXE.BAT
ML /Zi /c /Fl /coff %1.asm
LINK /subsystem:console %1.obj
del %1.obj
dir %1.*
```

如果要编译链接 ex1.asm，则只需输入：

```
MLEXE ex1    回车
```

1.3　将 Windows 汇编语言程序反汇编后的程序原形

将可执行程序用 IDA 反汇编工具反汇编后，程序的代码部分可直接使用，程序的其他部分稍作修改后，即可再编译链接成可执行程序。具体修改部分见程序尾的说明。

```
;iex1.asm,本程序为ex1.exe反汇编后的程序。
;iex1.asm(e:\masm\base)
;编译链接方法:
;ml /c /coff iex1.asm
;link /subsystem:Windows iex1.obj
    .386
    .model flat,stdcall
```

```
    option casemap:none
    .data
include windows.inc
include user32.inc
includelib user32.lib
include kernel32.inc
includelib kernel32.lib

Caption     db '抬头串',0
Text        db 'Hello!',0
    .code
        public start
start       proc near
        push    0               ;uType
        push    offset Caption  ;"抬头串"
        push    offset Text     ;"Hello!"
        push    0               ;hWnd
        call    MessageBoxA
        push    0               ;uExitCode
        call    ExitProcess
start       endp
        end start
```

说明：API 调用时右边的参数先入栈。用反汇编工具 IAD 反汇编后，保留代码段不变，将数据段中的数据搬入代码段，将其余部分删除，再加入包含文件和程序中的前 4 条指令，即可再编译链接成可执行程序。

将可执行文件反汇编成汇编语言程序，经过适当修改后，再编译链接成可执行文件，这是十分有意义的。

1.4 invoke 伪指令的使用格式、变量及数据段 data 和 data? 的区别

1.4.1 invoke 伪指令的使用格式

invoke 伪指令的使用格式为：

```
invoke 函数名[,参数1][,参数2]...
```

参数的个数不定，可以没有，也可以有多个。如果 invoke 与某个函数的参数个数不匹配（少或多），则编译时报错。如果参数个数少，则报错“error A2137:too few arguments to INVOKE”；如果参数个数多，则报错“error A2137:too many arguments to INVOKE”。

1.4.2　变量

1. 变量的命名规则

变量由大写字母 A，B，…，Z，小写字母 a，b，…，z，数字 0，1，2，…，9，下划线，符号@、$和?组成，且变量的第一个符号不能是数字。变量的长度不能超过 240 个字符，不能使用指令名关键字，在同一个作用域内不能重名。应该养成良好的命名习惯，如表 1-1 所示。

表　1-1

缩　写	含　义	缩　写	含　义
sz	表示以 0 结尾的字符串（ASCIIZ）	lp	表示指针 long point
h	表示句柄 handle	lpsz	表示指向 ASCIIZ 的指针
b	表示字节 byte	f	表示浮点数 float
w	表示字 word	st	表示结构体 struct
dw	表示双字 double word		

例如：

```
hWin          表示窗口句柄
lpArray       表示指向数组的指针
szString      以0结尾的字符串
stWndClass    WNDCLASS结构
bNumber       以字节定义的数
dwNumber      以双字定义的数
wNumber       以字定义的数
```

2. 全局变量

全局变量的作用域为整个程序。在 .data 和 .data? 段内定义的变量为全局变量。全局变量的定义格式为：

```
变量名 类型 初始值
变量名 类型 重复数量 dup(初始值)
```

例如：

```
count dw 0
array db 10 dup(0)
```

3. 局部变量

局部变量的作用域为一个程序内。局部变量的定义格式为：

```
local 变量名1[重复数量][: 类型],变量名2[重复数量][: 类型]...
```

局部变量要放在子程序的开始位置，并且没有初始值。例如：

```
.386
```

```
        .model flat, stdcall
        option casemap :none
include     windows.inc
include     kernel32.inc
includelib  kernel32.lib
    .code
SubProc proc,x:byte,y:byte

    local a:byte            ;定义局部变量
    local b:byte            ;定义局部变量

    mov al,x
    mov a,al
    mov al,y
    mov b,al
    ret
SubProc endp
main    proc
    invoke SubProc,1,2      ;调用子程序（右边参数先入栈）
    invoke ExitProcess,0    ;退出进程
main    endp
    end main
```

4. 局部变量在栈中的位置

将以上程序用 IDA 反汇编后的程序如下：

```
sub_401000  proc near

var_2       = byte ptr -2
var_1       = byte ptr -1
arg_0       = byte ptr  8
arg_4       = byte ptr  0Ch

        push    ebp
        mov ebp, esp
        add esp, 0FFFFFFFCh     ;(-4)的补码=0FFFFFFFCh
        mov al, [ebp+arg_0]     ;x=1
        mov [ebp+var_1], al
        mov al, [ebp+arg_4]     ;y=2
        mov [ebp+var_2], al
        leave
        retn    8               ;将入口参数（x,y)退栈
sub_401000  endp
        public start
start   proc near
        push    2
        push    1
        call    sub_401000
```

```
        push    0
        call    $+5
        jmp ds:ExitProcess
start   endp
```

说明：参数在栈中的位置见图 1-2。

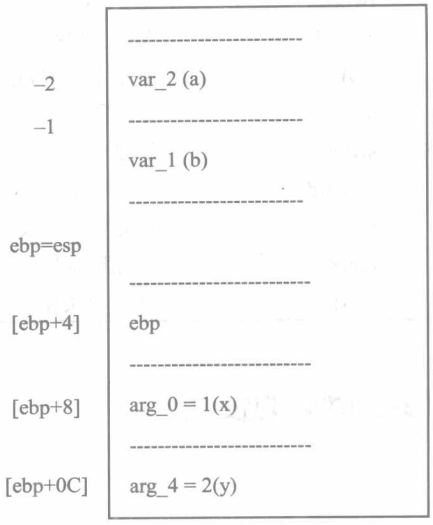


图 1-2　栈区示意图

1.4.3　数据段 data 和 data?的区别

程序的一般结构为：

```
.data
定义变量并初始化（有初始值）
.data?
定义变量（变量的初始值为'? '）
.const
定义常量
.code
```

定义在 .data? 段中的变量的初始值只能是'?'。定义在 .data? 段中的变量不占用磁盘空间，即不增加 .exe 文件的大小。这是用 .data? 段的优点。例如，在 .data? 中定义变量：

```
.data?
sum dw  ?
array 10 db dup(?)
```

在实际应用中，上述变量的初始值为 0。

汇编语言中的类型如表 1-2 所示。

表 1-2

缩　写	全名写法	名称和字节数
db	byte	字节（1 字节）
dw	word	字（2 字节）
dd	dword	双字（4 字节）
df	fword	三字（6 字节）
dq	qword	四字（8 字节）
dt	tbyte	10 字节 BCD 码
	sbyte	有符号字节（1 字节）
	sword	有符号字（2 字节）
	sdword	有符号双字（4 字节）
	real4	单精度实型数（4 字节）
	real8	双精度实型数（8 字节）
	real10	10 字节实型数（10 字节）

1.4.4 高级语法 while-endw 的使用

while-endw 的格式为：

```
.while(条件)
  循环体（条件满足时执行）
.endw
```

```
;ex2.asm(e:\masm\base) 高级语法while-endw的使用示例。
;程序功能：用while循环给字节数组赋值并计算。
    .386
    .model flat, stdcall
    option casemap :none
include    windows.inc
include    kernel32.inc
includelib kernel32.lib
include    user32.inc
includelib user32.lib

   .data
a       db  10 dup(0)          ;定义字节数组
buffer  db  10 dup(0)
CapMsg  db  '输出',0
szFmt   db  '结果是:%d',0
i       db  0
sum     db  0

   .code
```