32位汇编

16位汇编与32位汇编

16位

- 实模式
- 只有一个特权级
- 只有1MB的物理存储空间，分段（64K逻辑段）
- 以中断调用的方法进行系统功能调用
- 特定的寻址方式只能使用特定的寄存器，比如基址变址、相对基址变址只能使用`bp、bx、si、di`

32位

- 保护模式
- 两个特权级，**R0**和**R3**
- 直接使用32位地址寻址一个不分段，达4G的空间
- 只有代码段（代码区）和数据段（数据区）
- 无须和段寄存器打交道
- 利用API
- 所有通用寄存器均可用来寻址
- 新增寻址方式：比例因子寻址，比如`mov ax, [esi + edi * 2 + 0x100]`
    - `edi`即为比例因子且只能取值**1、2、4、8**

指令集

32位汇编拥有不止一个指令集，比如：`.386`/`.386p`，`.486`/`.486p`，`.586`/`.586p`（p后缀是特权指令集）

内存模式、调用约定和大小写敏感

一般使用`.model flat, stdcall`的平坦模式，默认调用约定为标准调用约定

`option casemap: none`采用大小写敏感

代码段和数据段

- `.data`：数据段，直接放在可执行文件中
- `.data?`：未初始化的数据段，不直接放在可执行文件中而在运行期分配
- `.const`：只读数据段
- `.code`：代码段

使用API

1. 包含库

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

2. 调用API
   - 传统

     ```
     push ...
     push ...
     call 函数
     ```

   - `invoke`伪指令

     ```
     invoke 函数, 参数1, 参数2
     ```

定义函数

- `stdcall`

```
; stdcall调用约定
函数名 proc uses ecx ebx 参数名1:类型, 参数名2:类型   ; 类型名可以填写windows已定义的类型, uses填写需要使用的寄存器, 自动保存和恢复

; 局部变量
local @value:DWORD       ; 变量名:类型
local @ary[10]:BYTE
local @wc:WNDCLASS

; 局部变量的使用
mov @value, eax
mov @ary[2], eax
mov @wc.style, CS_HREDRAW or CS_VREDRAW

; ...

xor eax, eax     ; 默认使用eax返回
ret ; 内部自动平栈
函数名 endp
```

- `cdecl`

```
; cdecl调用约定
函数名 proc C 参数名1:类型, 参数名2:类型

; ...

xor eax, eax    ; 默认使用eax返回
ret ; 外部自动平栈
函数名 endp
```

其他伪指令

给编译器使用的指令，方便编写，编译器将其转化为等价的汇编指令

`addr`：取变量地址

`sizeof`：取变量大小

`.if .elseif .else .endif`：条件判断

`.break`：跳出当前循环

`.continue`：跳过后续，执行下次循环

`.while`：等价于`while`循环

`.repeat .until`：等价于`do-while`循环

例子

```
; ***************************
; *
; *      创建窗口程序
; *
; ***************************


.386    ; 386指令集
.model flat, stdcall ; 内存模式，平坦模式，默认调用约定是标准调用约定
option casemap:none ; 标识符大小写敏感

; 包含头文件
include windows.inc
include user32.inc
include kernel32.inc

; 包含库
includelib user32.lib
includelib kernel32.lib

; 初始化数据段
.data
  g_hInst HINSTANCE NULL
```

```
; 只读数据段
.const
    g_szTextShow db '你好,32位汇编!', 0
    g_szCaption  db '标题', 0
    g_szWindowClass db 'CR34Class', 0

; 未初始化的数据段
.data?
    g_aryTest db 100 dup(?)
    g_aryTest0 db 10000h dup(?)


; 代码区
.code
; ************************** 窗口过程 **************************
WndProc proc hWnd:HWND,message:UINT,wParam:WPARAM,lParam:LPARAM
  .if message == WM_DESTROY
      invoke PostQuitMessage, 0     ; 投递退出消息
  .endif

  ; 调用默认窗口过程
  invoke DefWindowProc,hWnd, message, wParam, lParam

  ret
WndProc endp


; ************************** 注册窗口类 **************************
MyRegisterClass proc hInstance:HINSTANCE
    local @wcex:WNDCLASSEXA
    ; 初始化窗口类数据成员
    mov @wcex.cbSize        , sizeof WNDCLASSEX
    mov @wcex.style         , CS_HREDRAW or CS_VREDRAW;
    mov @wcex.lpfnWndProc   , offset WndProc
    mov @wcex.cbClsExtra    , 0
    mov @wcex.cbWndExtra    , 0
    mov eax, hInstance
    mov @wcex.hInstance     , eax;
    mov @wcex.hIcon         , NULL
    mov @wcex.hCursor       , NULL
    mov @wcex.hbrBackground , COLOR_WINDOW+1;
    mov @wcex.lpszMenuName  , NULL
    mov @wcex.lpszClassName , offset g_szWindowClass;
    mov @wcex.hIconSm       , NULL;

    ; 注册窗口类
    invoke RegisterClassExA, addr @wcex
    ret
MyRegisterClass endp

InitInstance proc hInstance:HINSTANCE, nCmdShow:UINT
  local @hWnd:HWND

  mov eax, hInstance
  mov g_hInst, eax
  ; 创建窗口
  invoke CreateWindowExA,\
```

```
        NULL, \
        offset g_szWindowClass, \
        offset g_szCaption, \
        WS_OVERLAPPEDWINDOW,\
        CW_USEDEFAULT, \
        0, \
        CW_USEDEFAULT, \
        0, \
        NULL, \
        NULL, \
        hInstance, \
        NULL

  mov @hWnd, eax
  .if !@hWnd
    mov eax, FALSE  ; 创建失败
    ret
  .endif

  invoke ShowWindow, @hWnd, nCmdShow    ; 显示窗口
  invoke UpdateWindow, @hWnd    ; 更新窗口

  mov eax, TRUE
  ret
InitInstance endp

; ************************** main ***************************
WinMain proc C uses ecx ebx edx esi edi  hInstance:HINSTANCE, \
            hPrevInstance:HINSTANCE,\
            pCmdLine:LPSTR, \
            nCmdShow:UINT
    local @dwVal:DWORD
    local @wVal:WORD
    local @ary[10]:BYTE
    local @wc:WNDCLASS
    local @msg:MSG

    mov @dwVal, eax
    mov @wVal, ax
    mov @ary[2], al

  ; 注册窗口类
  invoke MyRegisterClass, hInstance
  .if eax == 0
    ret
  .endif

  ; 创建、显示更新窗口
  invoke InitInstance, hInstance, nCmdShow
  .if !eax
    ret
  .endif


  ; 主消息循环:
  .while TRUE
    invoke GetMessage, addr @msg, NULL, 0, 0
    .if eax == 0
```

```
      .break
      ;.continue
    .endif

    ; 派发消息
    invoke DispatchMessage, addr @msg
  .endw

  xor eax, eax
  ret
WinMain endp


; ************************** 入口点 ****************************
START:
  invoke GetModuleHandleA, NULL
  invoke WinMain, eax, NULL, NULL, SW_SHOW
  invoke ExitProcess, 0

end START ;设置代码结束位置，设置代码入口
```