Importación de Funciones del API de Windows

Introducción

Windows no permite la ejecución directa de llamadas a nivel de hardware. En su lugar, proporciona APIs que permiten a los programas interactuar con el sistema operativo de manera controlada y segura.

kernel32.lib es una biblioteca fundamental de Windows que contiene funciones esenciales para la gestión de entrada/salida, memoria y procesos.

Explicación de PROTO y el estándar de llamada STDCALL

```
GetStdHandle PROTO STDCALL :DWORD
WriteConsoleA PROTO STDCALL :DWORD, :DWORD, :DWORD, :DWORD, :DWORD
```

Estas líneas definen los prototipos de funciones que se usarán desde kernel32.dll.

- GetStdHandle obtiene un handle de un dispositivo estándar.
- WriteConsoleA imprime texto en la consola.

STDCALL es un estándar de llamada utilizado por Windows donde los parámetros se pasan por la pila y la función llamada es responsable de limpiar la pila.

¿Qué es un Handle en Windows?

Un handle es un identificador único asignado por el sistema operativo para referenciar un recurso como archivos, dispositivos o procesos. En este caso, GetStdHandle devuelve un handle asociado a la salida estándar (stdout), lo que permite a WriteConsoleA escribir en la consola.

Definición de Constantes y Variables

```
STD_OUTPUT_HANDLE equ -11  ; Descriptor para stdout
```

- STD_OUTPUT_HANDLE es un valor predefinido en Windows que representa la salida estándar (stdout).
- equ -11 significa que STD_OUTPUT_HANDLE se sustituirá por -11 en el código.

Definición de Datos

```
.data
mensaje db "Hola, Mundo!", 0
tamanyoMensaje dd 12
```

- mensaje es una cadena de caracteres terminada en NULL.
- tamanyoMensaje almacena el número de caracteres a imprimir (12 en este caso).

Código Principal

Paso 1: Obtener el Handle de la Consola

Importación de Funciones del API de Windows

```
push STD_OUTPUT_HANDLE
call GetStdHandle
mov ebx, eax
```

- push STD_OUTPUT_HANDLE coloca -11 en la pila.
- call GetStdHandle llama a la función y devuelve el handle en EAX.
- mov ebx, eax guarda el handle en EBX.

Paso 2: Preparar los Parámetros para WriteConsoleA

```
mov edx, OFFSET mensaje
push 0
push offset tamanyoMensaje
push 12
push edx
push ebx
```

WriteConsoleA requiere cinco parámetros en este orden:
1. Handle de la consola (ebx)
2. Puntero a la cadena (mensaje)
3. Número de caracteres (12)
4. Puntero a una variable (tamanyoMensaje)
5. Reservado (0 o NULL)

Paso 3: Llamar a WriteConsoleA

```
call WriteConsoleA
```

Esto imprime "Hola, Mundo!" en la consola y almacena el número de caracteres escritos en tamanyoMensaje.

Paso 4: Retornar al Sistema Operativo

```
RET
```

Finaliza la ejecución y devuelve el control al sistema operativo.

Fin del Código

```
main ENDP
END main
```

- main ENDP indica el fin del procedimiento main.
- END main define el punto de entrada del programa.