Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza baja

**INVESTIGACIONES**

**UNIDAD VI**

Integrantes: **Martinez Reyes Fernando**

**Gonzalez Saldívar Luis Roberto**

Profesor: **Dr. García Ruiz Alejandro Humberto**

Asignatura: **Programación de Microprocesadores**

**8v**o. Semestre – Grupo “**I**”

2022-3

**ÍNDICE**

[1.- ¿Qué es la programación hibrida? 3](#_Toc121244234)

[2.- ¿Qué es asm inline? 3](#_Toc121244235)

[3.- Cuales son las aplicaciones de una DLL 4](#_Toc121244236)

[4.- Cuando conviene el uso de programación hibrida 5](#_Toc121244237)

[5.- DLL import 5](#_Toc121244238)

[6.- Int Ptr 6](#_Toc121244239)

# 1.- ¿Qué es la programación hibrida?

La programación hibrida es utilizada en los casos en donde el código en ensamblador dificulta la estructuración del programa. La programación híbrida proporciona un mecanismo por medio del cual podemos aprovechar las ventajas del lenguaje ensamblador y los lenguajes de alto nivel, todo esto con el fin escribir programas más rápidos y eficientes.

Al trabajar con un lenguaje de alto nivel, en ocasiones nos encontramos con el problema de que necesitamos que haga determinada función o trabajo pero desafortunadamente ésta solo existe en otro lenguaje que no es el que necesitamos utilizar, o simplemente, no encontramos esa función en ningún lenguaje de alto nivel.

**Ventajas de la Programación Híbrida**

* Para mejorar la escalabilidad
* Cuando muchas tareas producen desbalanceo
* Aplicaciones que combinan paralelismo de grano grueso y fino
* Reducción del tiempo de desarrollo de código
* Cuando el número de procesos MPI es fijo
* En caso de mezcla de paralelismo funcional y de datos

En este momento el lenguaje ensamblador constituye una herramienta no solo eficaz, sino simple para producir un parche para el compilador de nuestro lenguaje preferido. Tal vez el mayor problema con el que nos enfrentemos sea el de cómo conectar ambos programas (el de alto y el de bajo niveles) y cómo pasar variables de un programa al otro. Para conseguir nuestro objetivo se utilizan pseudo-operadores, es decir, instrucciones que aparecen en el código fuente del ensamblador pero que no generan ninguna instrucción de máquina, pero proporcionan directivas para que el ensamblador pueda operar con datos, ramificaciones condicionales, generación de listados y con macros durante el proceso de ensamble.

# 2.- ¿Qué es asm inline?

Un ensamblador en línea es una característica de algunos compiladores que permite que el código de bajo nivel escrito en lenguaje ensamblador se incruste dentro de un programa, entre el código que de otro modo se ha compilado a partir de un lenguaje de nivel superior como C o Ada.

La incrustación de código en lenguaje ensamblador generalmente se realiza por una de estas razones:

* **Optimización:** los programadores pueden usar código de lenguaje ensamblador para implementar las partes más sensibles al rendimiento de los algoritmos de su programa , código que puede ser más eficiente que el que podría generar el compilador.
* **Acceso a instrucciones específicas del procesador:** la mayoría de los procesadores ofrecen instrucciones especiales, como comparar e intercambiar y probar y establecer instrucciones que pueden usarse para construir semáforos u otras primitivas de sincronización y bloqueo. Casi todos los procesadores modernos tienen estas instrucciones u otras similares, ya que son necesarias para implementar la multitarea . Se encuentran ejemplos de instrucciones especializadas en los conjuntos de instrucciones SPARC VIS , Intel MMX y SSE y Motorola Altivec .
* Acceso a convenciones de llamadas especiales que aún no son compatibles con el compilador.
* **Llamadas al sistema e interrupciones:** los lenguajes de alto nivel rara vez tienen una función directa para realizar llamadas al sistema arbitrarias, por lo que se usa código ensamblador. Las interrupciones directas se proporcionan incluso con menos frecuencia.
* Emitir directivas especiales para el enlazador o ensamblador, por ejemplo para cambiar seccionamientos, macros, o para hacer alias de símbolos.

# 3.- Cuales son las aplicaciones de una DLL

DLL (Dinamic Link Library). Una DLL no es más que una porción de programa que realiza diversas funciones, que está alojada en un directorio del sistema y que se ejecuta a petición de una aplicación. Dicha aplicación la enlaza dinámicamente, es decir, ejecuta y llama a las funciones que la dll aloja mientras se ejecuta para obtener los datos o el comportamiento que la aplicación desea. Las ventajas que tienen las DLL son claras:

Reducen el tamaño de los archivos ejecutables: Ya que la mayor parte de las funciones se realizan dentro de la dll, la aplicación un mero ejecutante de las mismas y mostrador de los resultados obtenidos, lo que minimiza el tamaño del programa.

Pueden estar compartidas entre varias aplicaciones: Por ejemplo las Microsoft Fundation Classes MFC se encargan de gestionar la parte gráfica de Windows. Muchas aplicaciones hacen uso de dichas dll.

Facilitan la gestión y aprovechamiento de la memoria del sistema: Al ser el sistema operativo el que realiza la carga, es capaz de gestionarlas de una manera más óptima. Además, al estar compartidas no es necesario tener más que una copia al tiempo cargada en la memoria, lo que redunda en mejoras de aprovechamiento y rendimiento.

Flexibilidad frente a cambios: No es necesario rehacer todo el programa para solucionar errores u optimizar el rendimiento. Símplemente modificando la dll se obtiene el resultado pretendido.

# 4.- Cuando conviene el uso de programación hibrida

Al contrario que para las aplicaciones nativas, que se desarrollan en el lenguaje nativo de cada dispositivo móvil, el desarrollo de las aplicaciones híbridas se consigue con tecnologías web (HTML5, CSS3 y JAVA) y un lenguaje diferente para cada sistema operativo.

**Ventajas**

* Una buena parte de su desarrollo sirve para todas las plataformas, Android, Windows Phone, iOS, etc.
* Son accesibles desde un dispositivo móvil por medio de un icono en el escritorio.
* El desarrollo de aplicaciones híbridas es más sencillo, lleva menos tiempo y supone un menor coste.
* Gozan de mayor mantenibilidad. Las actualizaciones de la aplicación solo dependen de un desarrollo con pequeños cambios para cada plataforma

**Inconveniencias**

* Se requiere su instalación en el dispositivo, ya que, aunque su diseño es simulado para parecer una aplicación nativa como lo son las cámaras, los sensores, etc., hay que descargarlas previamente.
* Si se trata de una aplicación muy compleja, es posible que la velocidad y la fluidez se vean mermadas.
* No aprovechan al 100 % las características del dispositivo, especialmente cuando se trata de la funcionalidad de videojuegos, la de 3D, etc.
* Consumen bastante espacio en el dispositivo del usuario.

# 5.- DLL import

DllImport Attribute es una etiqueta declarativa utilizada en C # para marcar un método de clase como definido en una biblioteca externa de enlace dinámico (DLL) en lugar de en cualquier ensamblado .NET.

El atributo DllImport se usa en tiempo de ejecución para llamar a una función exportada en una DLL externa implementada con código no administrado que se ejecuta fuera del control de Common Language Runtime (CLR). También se usa para especificar otra información necesaria al llamar a la función, como la convención de llamada, el tipo de conversión de los parámetros pasados ​​entre el código administrado y el no administrado, etc. Por lo tanto, se utiliza para interoperar con el código en componentes no administrados y heredados que residen en Windows DLL y está escrito en C o C ++.

DllImport es útil en situaciones en las que la funcionalidad de una aplicación administrada, cuyo código es ejecutado por CLR, solo se puede ampliar accediendo y reutilizando la funcionalidad de la interfaz de programación de aplicaciones (API) Win32 y otros códigos no administrados definidos por el usuario que se implementan en DLL. Esto puede ahorrar esfuerzo de desarrollo y hacer uso del código existente robusto y bien probado sin reinventar la rueda en una aplicación administrada. Por lo general, se necesita al usar archivos DLL que no se pueden reescribir pero que tienen la capacidad de modificarse

# 6.- Int Ptr

Representa un entero con signo donde el ancho de bits es el mismo que un puntero.

El IntPtr tipo está diseñado para ser un entero cuyo tamaño es el mismo que un puntero. Es decir, se espera que una instancia de este tipo sea de 32 bits en un proceso de 32 bits y 64 bits en un proceso de 64 bits.

El IntPtr tipo se puede usar en idiomas que admiten punteros y como medio común de hacer referencia a datos entre idiomas que sí y no admiten punteros.

IntPtr Los objetos también se pueden usar para contener identificadores. Por ejemplo, las instancias de IntPtr se usan ampliamente en la System.IO.FileStream clase para contener identificadores de archivo.

El uso IntPtr de como puntero o un identificador es propenso a errores y no es seguro. Es simplemente un tipo entero que se puede usar como formato de intercambio para punteros y identificadores debido a que tiene el mismo tamaño. Fuera de requisitos de intercambio específicos, como para pasar datos a un lenguaje que no admite punteros, se debe usar un puntero con tipo correcto para representar punteros y SafeHandle se debe usar para representar identificadores.