República Bolivariana de Venezuela
Universidad Nacional Experimental Politécnica
"Antonio José de Sucre"
Vice Rectorado Barquisimeto
Departamento de Ingeniería Electrónica

Practica 3
laboratorio de diseño de sistemas de computación

Integrantes:
Gerardo Alfonzo Campos Fonseca
V. 27085179
José Andrés Cortez Teran
V. 26540824

Barquisimeto, Julio del 2021

Índice

Índice de figuras

1. Primera Actividad

1.7

Se creo un único archivo **mensaje.obj**, el cual es un archivo de tipo “objeto”.

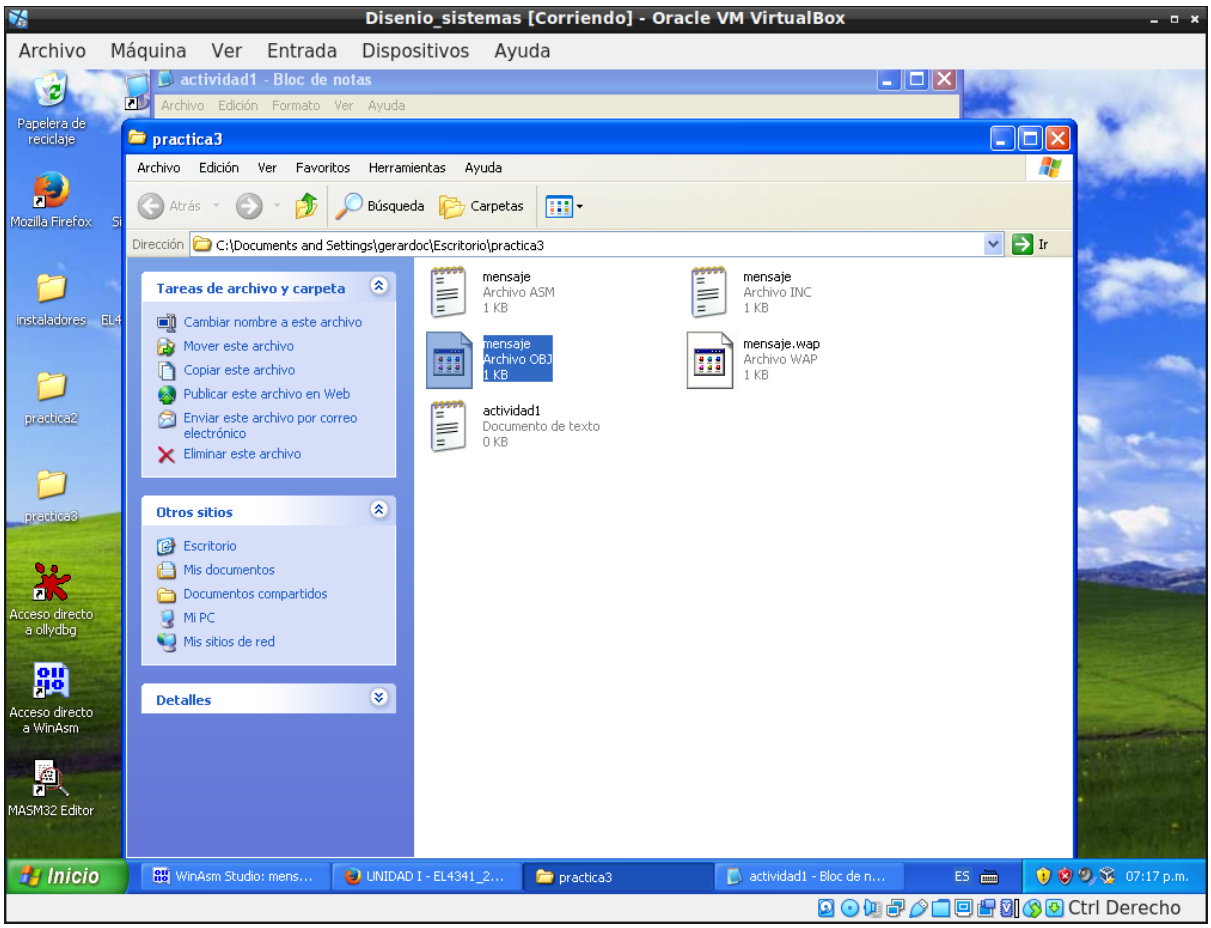

Figura 1: archivos creados al compilar

1.8

Se creo un único archivo, el ejecutable **mensaje.exe**.

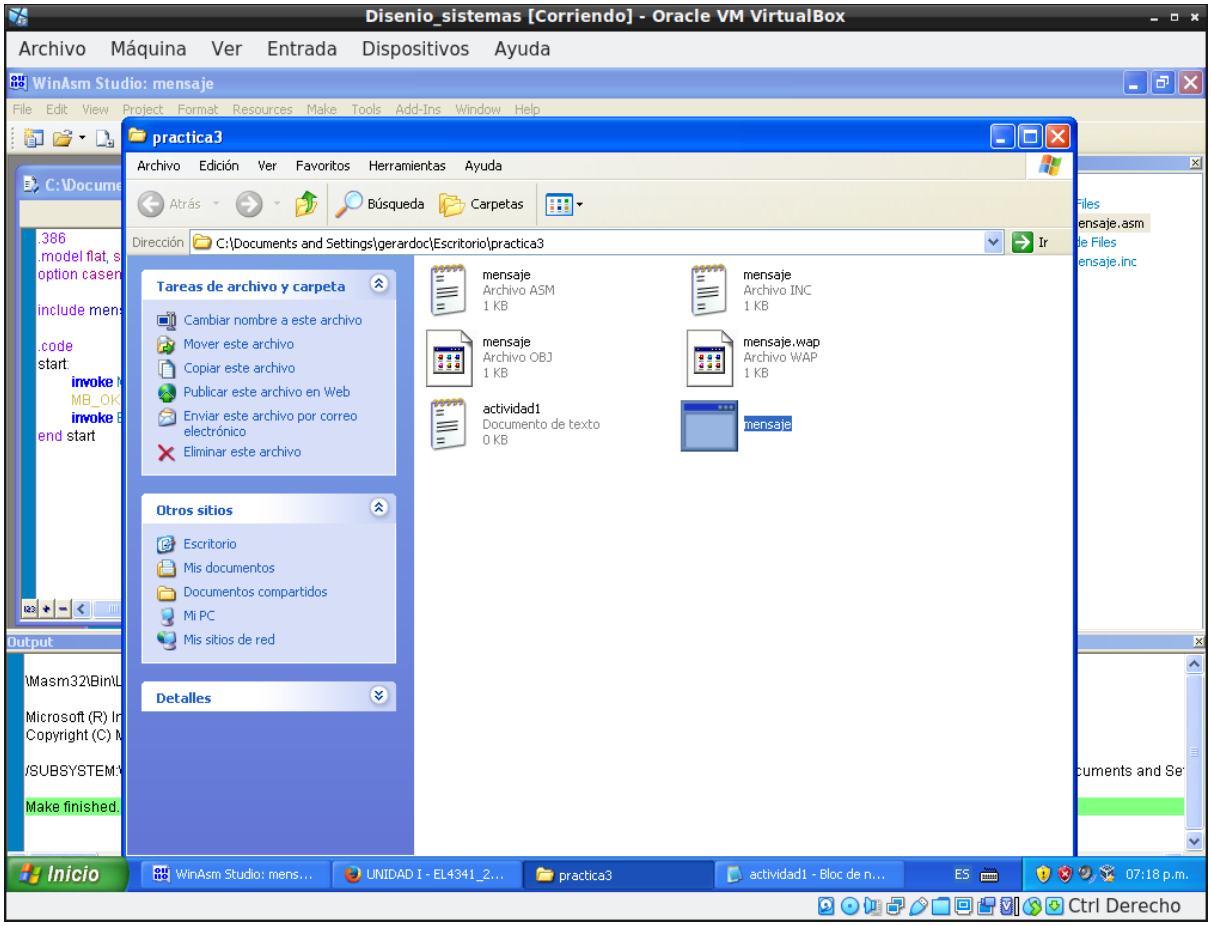

Figura 2: archivos creados al enlazar

1.9

Al ejecutar el programa se abre una ventana de mensajes (ventana emergente), la cual dice: “Este es un programa simplea para Windows” y se cierra al presionar el boton “Aceptar”.

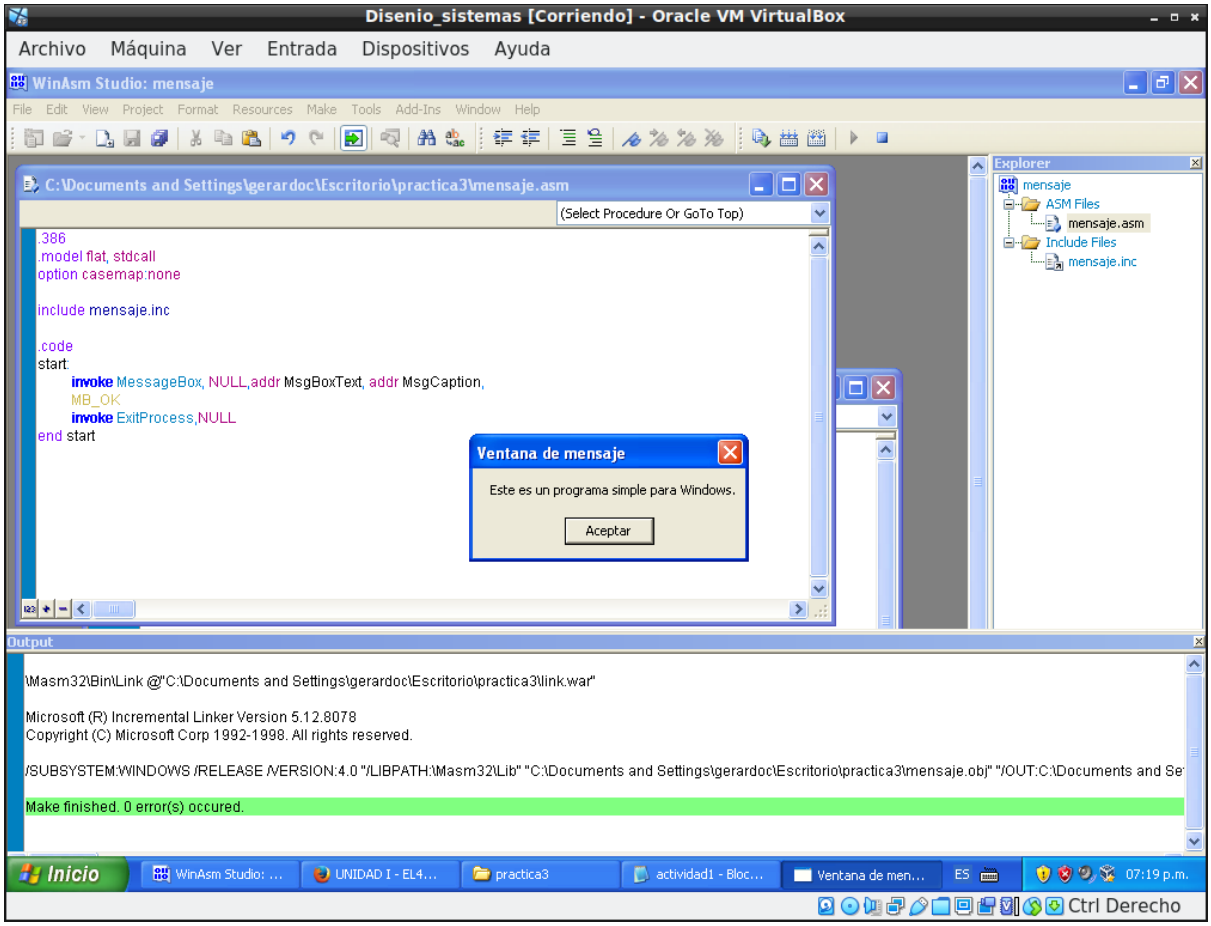

Figura 3: ejecución del programa

2. Segunda Parte

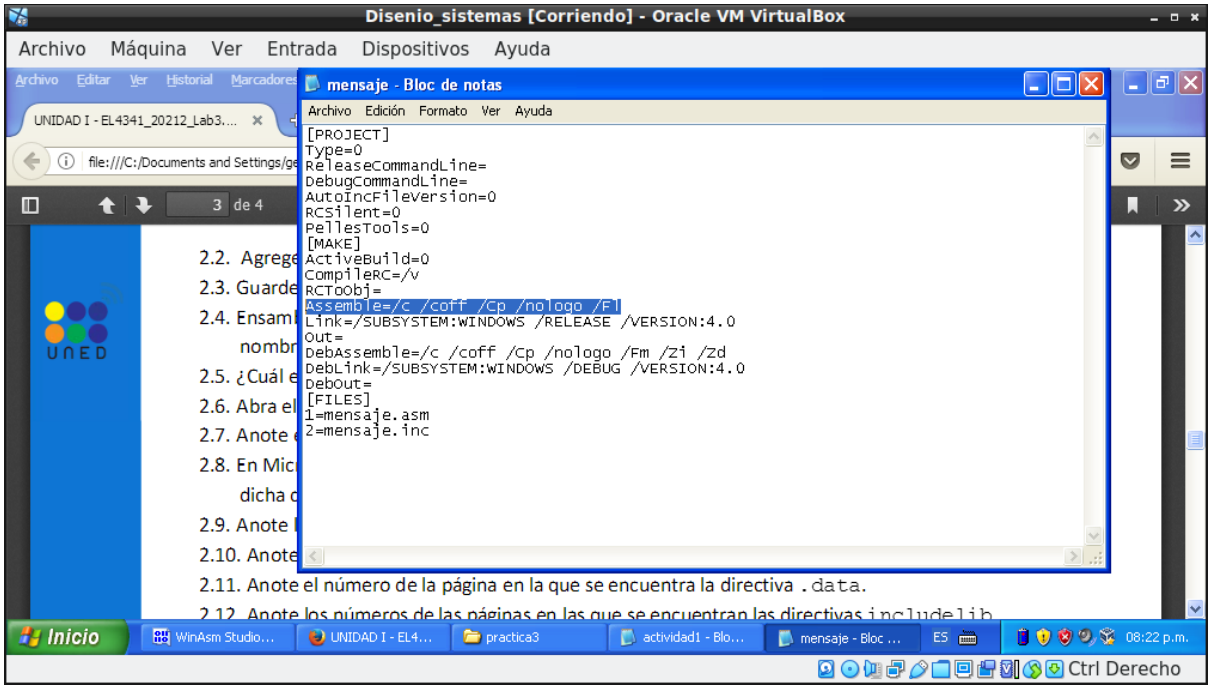

Figura 4: edición del proyecto

2.4

Se usa el comando:

```
\Masm32\Bin\ML /c /coff /Cp /nologo /Fl /I"\Masm32\Include"
"C:\Documents and Settings\gerardoc\Escritorio\practica3\mensaje.asm"
```

Los archivos que se crearon fueron:

- **mensaje.obj** (igual que anteriormente)
- **mensaje.lst**

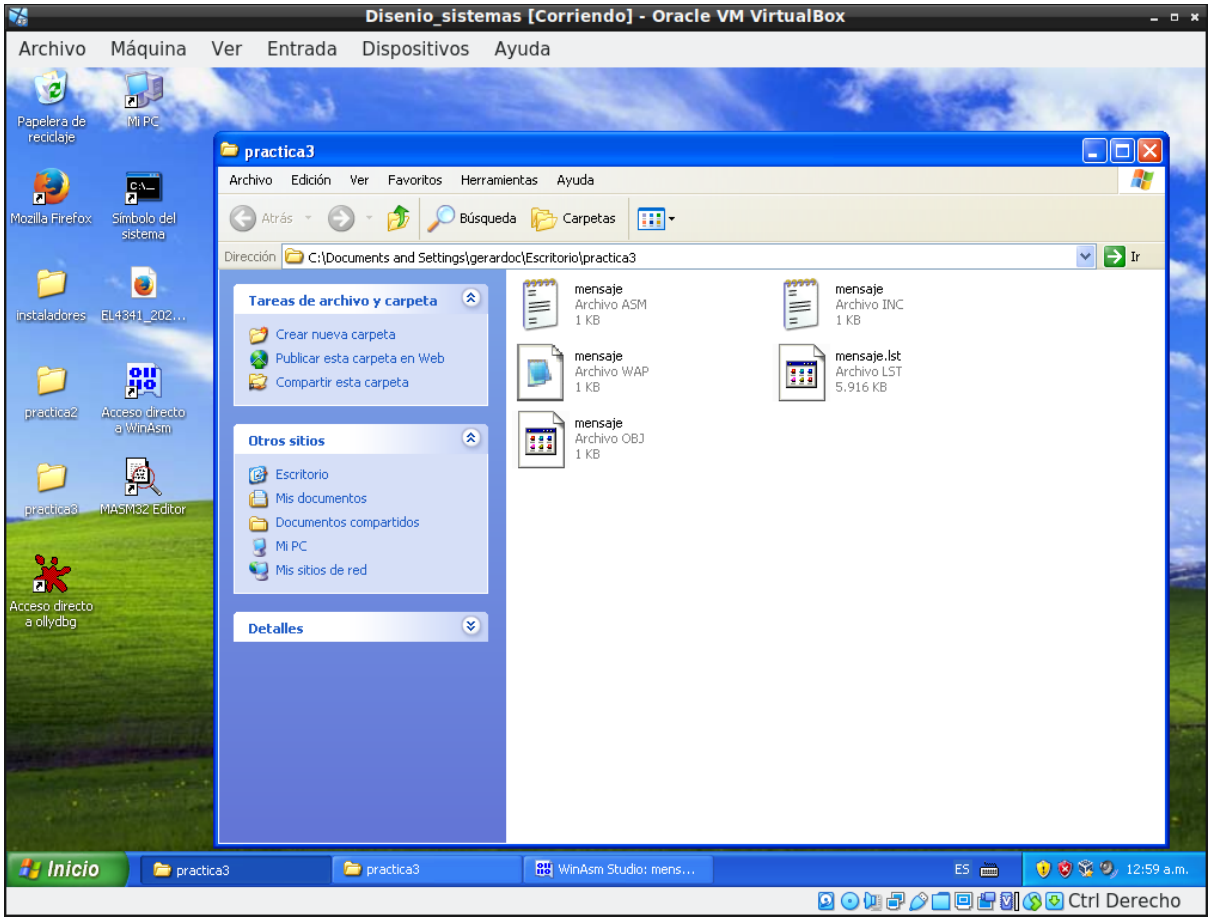

Figura 5: archivos creados al compilar

2.5

La opción **/Fl** se utiliza para generar una lista de código ensamblado (Generates an assembled code listing).

2.7

El archivo ocupa 2752 paginas, Cabe resaltar que se uso **Libreoffice Writer**.

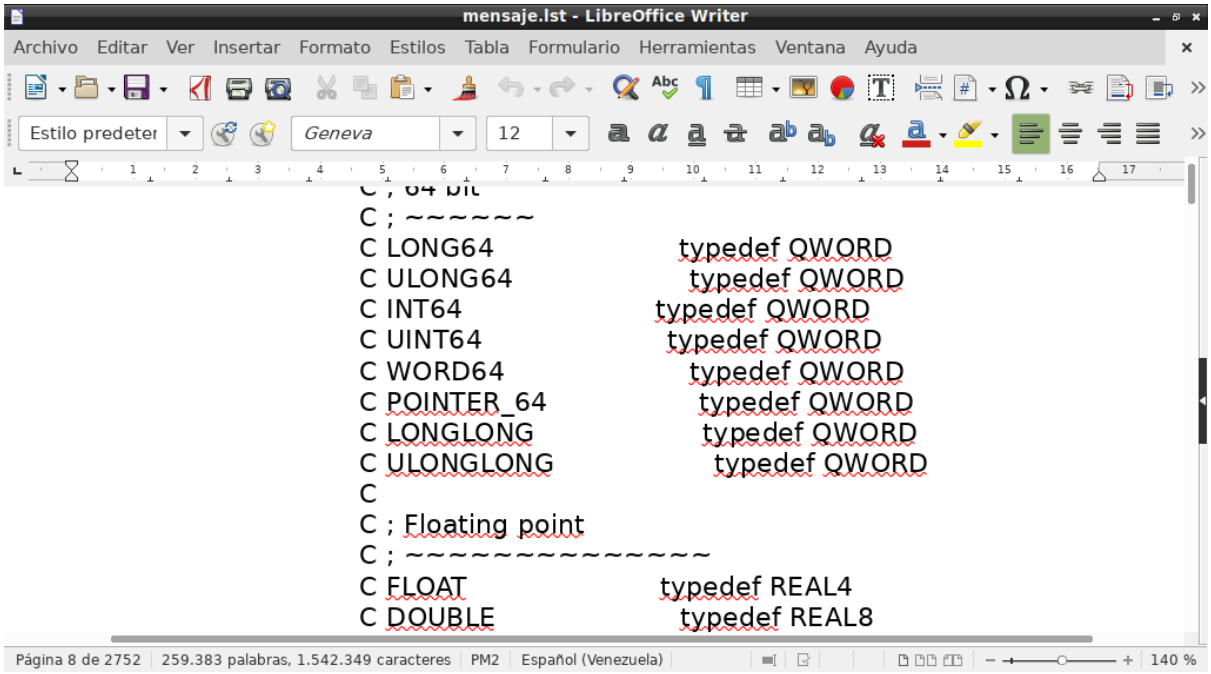

Figura 6: Numero de paginas

2.8

La directiva **end start** se encuentra en la pagina 1638.

2.9

Las dos instrucciones que comienzan con **invoke** se encuentran en la pagina 1638.

2.10

La directiva **.code** se encuentra en la pagina 1638.

2.11

La directiva **.data** se encuentra en la pagina 1638.

2.12

Ambas directivas **includelib** se encuentra en la pagina 1638.

2.13

Las directivas se encuentran en:

- **include windows.inc** en la pagina 1.
- **include kernel32.inc** en la pagina 1531.
- **include user32.inc** en la pagina 1591.

2.14

La directiva include **mensaje.inc** esta en la pagina 1.

2.15

Inserta código fuente, del archivo de código fuente proporcionado por **filename**, en el archivo de código fuente actual durante el ensamblado.

3. Tercera Parte

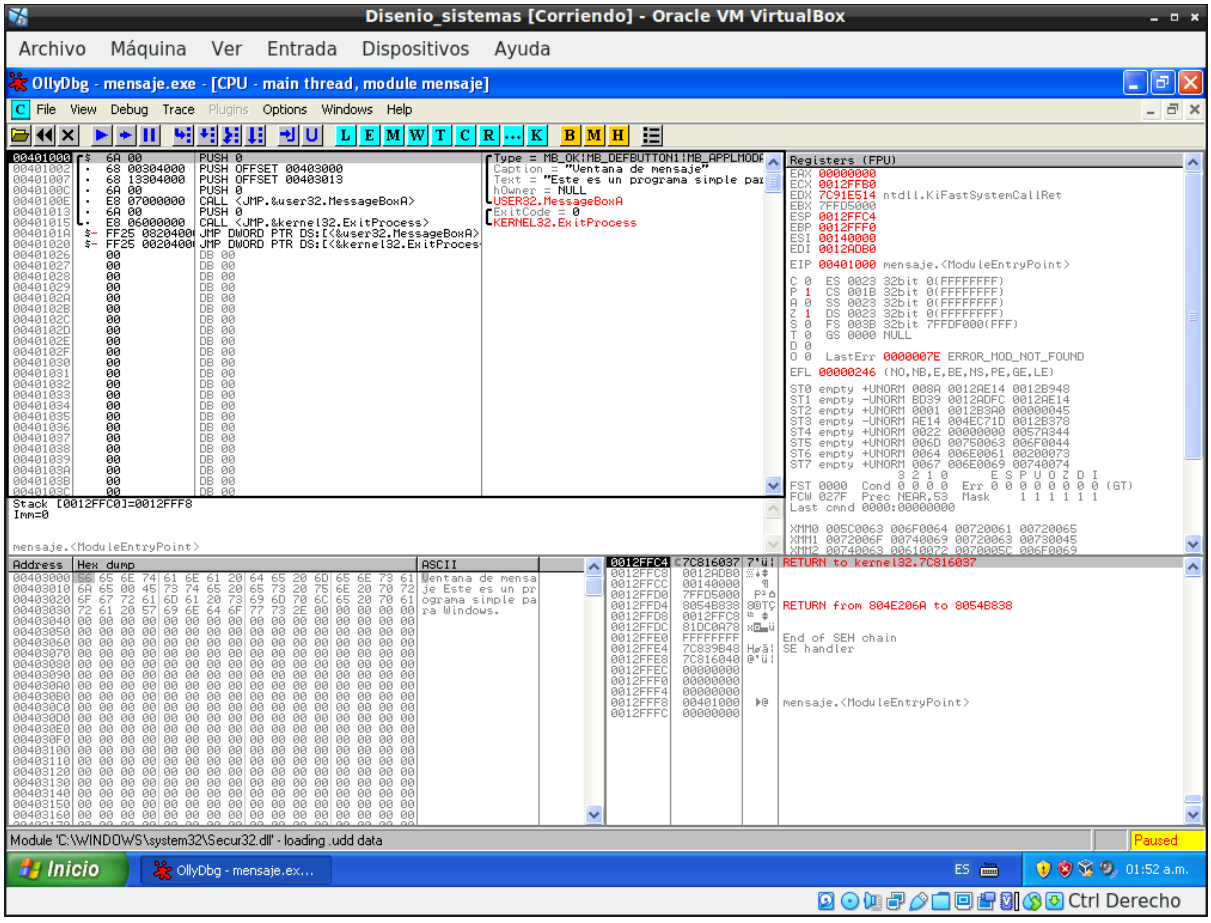

Figura 7: Ejecutable en Olly

3.2

La primera instrucción según **Olly** es `PUSH 0`.

3.3

La dirección de la primera instrucción es 00401000 y le da el nombre de `mensaje.<moduleEntryPoint>`.

3.4

Las lineas dicen:

- **MessageBox:** dice: `Dest = mensaje.0040101A - jumps to user32.MessageBoxA` y se lee la direccion a la que saltara en: `mensaje.<moduleEntryPoint>+0E`.

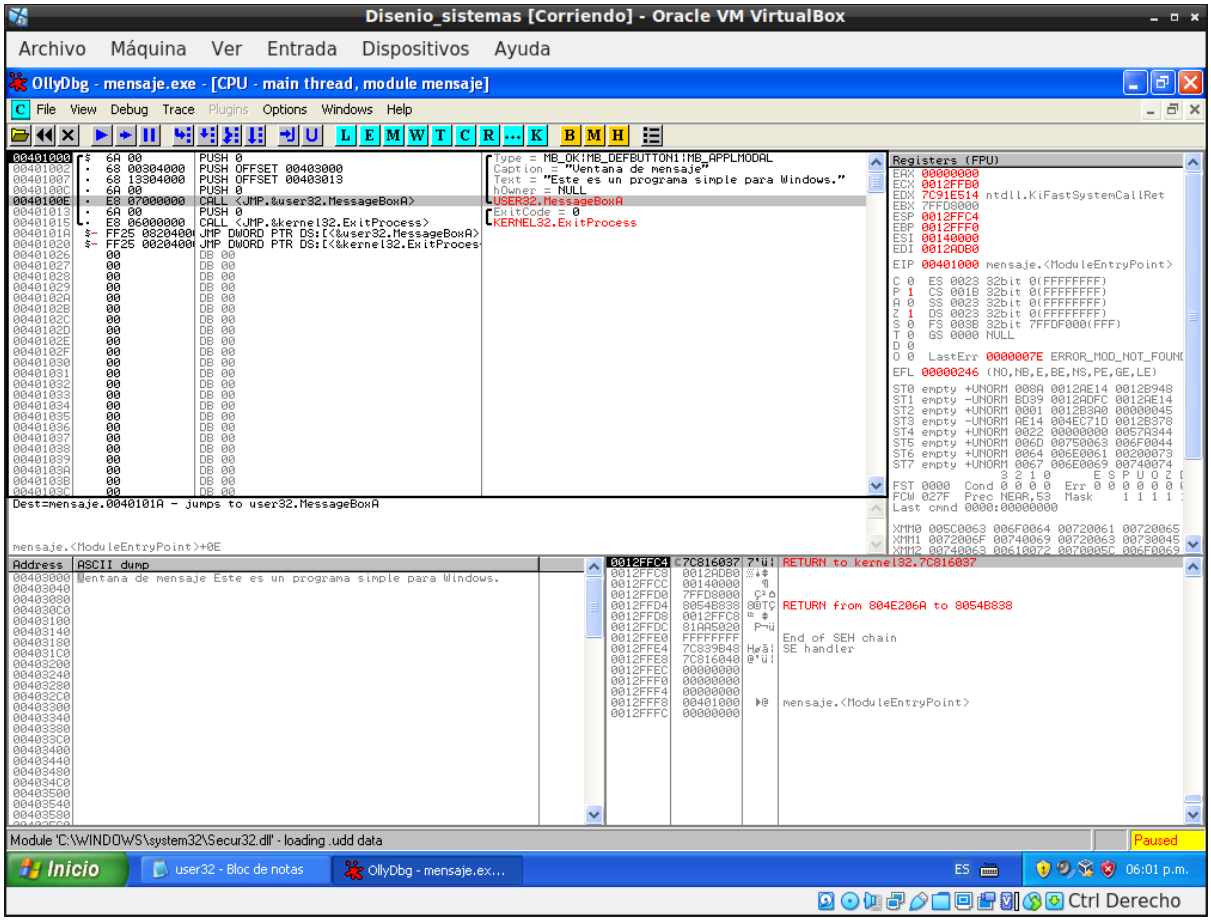

Figura 8: Ejecutable en Olly, MessageBoxA instruction

- MessageBox: dice: `Dest = mensaje.00401020 - jumps to Kernel32.ExitProcess` y se lee la direccion a la que saltara en: `mensaje.<moduleEntryPoint>+15`.

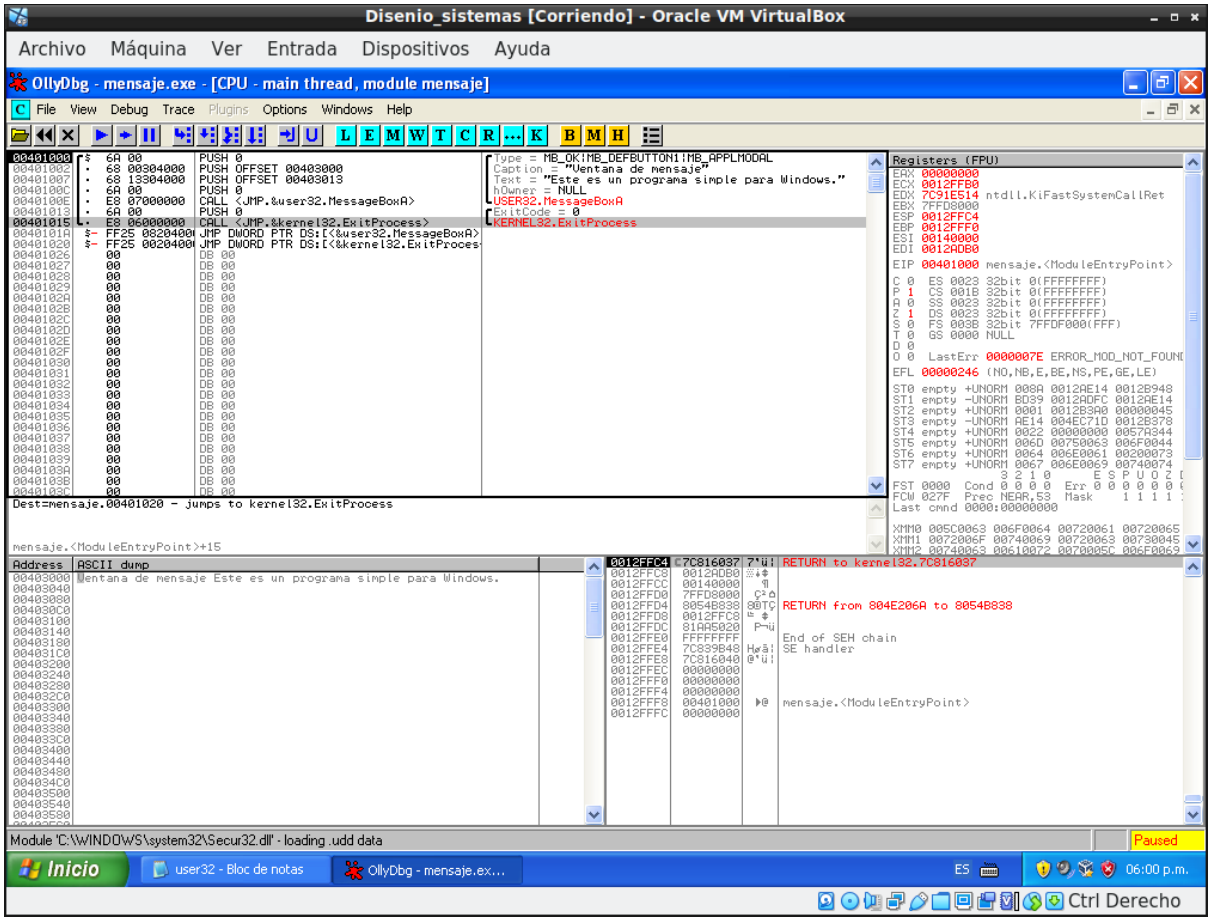

Figura 9: Ejecutable en Olly, ExitProcess instruction

3.5

Según Microsoft, 2021c es una directiva que solo funciona con **MASM** de 32 bits y llama al procedimiento de la dirección proporcionada por la expresión, puede contener argumentos y todo esto es pasado por la pila o en registros, según sean las convenciones de llamada estándar del lenguaje. Se usa de la forma `INVOKE expresion [,argumento, \dots]`.

4. Cuarta Parte

4.1

Debido a que no se encontró el archivo **win32.hlp**, se busco en internet y consiguió **win32.chm** en laurencejackson, 2015. La cual es versión actualizada del sistema de ayudas de Windows, pero de todas formas contiene la misma información.

MessageBox

La función **MessageBox** crea, muestra y opera una caja de mensaje. El **MessageBox** contiene un mensaje definido en la aplicación, un titulo y un numero predefinido de iconos y botones.

```
int MessageBox(
HWND hWnd, // handle of owner window
LPCTSTR lpText, // address of text in message box
LPCTSTR lpCaption, // address of title of message box
UINT uType // style of message box
);
```

ExitProcess

La función ExitProcess termina el proceso y todos sus hilos de ejecución.

```
VOID ExitProcess(
UINT uExitCode // exit code for all threads
);
```

4.2

La linea es: `ExitProcess PROTO STDCALL :DWORD`.

4.3

MessageBoxA aparece en la definición y todo es:

```
MessageBoxA PROTO STDCALL :DWORD,:DWORD,:DWORD,:DWORD
IFNDEF __UNICODE__
  MessageBox equ <MessageBoxA>
```

Post

1.

Una interfaz para programación de aplicaciones, como su nombre lo indica, es una interfaz que facilita la elaboración de aplicaciones. Las aplicaciones de Windows necesitan hacer llamadas al núcleo del sistema operativo. Como estas funcionalidades pueden cambiar entre diferentes versiones del sistema operativo, Microsoft elaboro la **API** de windows. El **API** proporciona una interfaz estándar, que mantiene retrocompatibilidad mientras que permite cambiar su implementación en diferentes versiones del sistema operativo. Ademas, el **API** también ayuda al programar proporcionando funciones que facilitan tareas comunes en la programación de aplicaciones.

2.

Una biblioteca de enlace dinámico es un conjunto de funciones que son cargadas por una aplicación en tiempo de ejecución. Esto lo diferencia de las bibliotecas estáticas que son enlazadas en tiempo de compilación. Esto trae múltiples ventajas y múltiples desventajas. La principales ventajas son que disminuye el tamaño del ejecutable y que la actualización de las librerías no requiere una actualización de las aplicaciones. Las principales desventajas son que tiene una penitencia en tiempo de ejecución y que son vector de ataque para **hackers**, los cuales pueden insertar un **DLL** malicioso en una aplicación con privilegios.

3.

Según el articulo Microsoft, 2017 **Kernel32.dll** posee funciones de bajo nivel para manejar recursos del sistema operativo, como el manejo de memoria y creación de procesos ; **GDI32.dll** posee funciones para el manejo de dispositivos de salida, como puede ser el dibujar en pantalla o el manejo de fuentes tipográficas y **User32.dll** tiene funciones para el manejo de temporizadores, menús y comunicaciones.

4.

El prototipo de una función indica que parámetros toma la función y que tipo de dato retorna. Tienen la siguiente estructura:

```
ValorDeRetorno nombreDeFuncion(TipoDato1 param1, ....){
    // Información adicional del cuerpo de la funcion
}
```

5.

Una archivo que contiene declaraciones, cabeceras, funciones y otros datos relacionados a un librería. Son equivalentes a los archivos **.h** de C.

6.

Contiene la implementación de funciones declaradas en los **.inc**, son la contraparte estatica de los **.dll**. Son archivos binarios que contienen codigo compilado de un lenguaje de alto nivel o código ensamblado.

7.

Ambas son librerías (conjunto de funciones o datos reutilizables), la diferencia esta en su forma de enlazado. En las librerías estáticas el proceso de enlazado ocurre en la etapa de compilación, mientras que en las librerías dinámicas el enlazado ocurre en tiempo de ejecución.

8.

La directiva **INVOKE** es un poderoso reemplazo para las instrucciones **CALL** de intel, ya que deja pasar múltiples argumentos de una forma mas sencilla. Por tanto, se encarga de llamar procedimientos que se encuentren en una dirección dada, definida por una **expresión**, y pasarle los parámetros necesarios. Maneja argumentos de tipo inmediato, nombres de variables, direcciones de memorias y registros. Tiene como ventaja el otorgar facilidad al momento de llamar los procedimientos, ya que se elimina el uso del sufijo `@N` y la misma directiva se encarga de cargar los parámetros en el **stack**. Simplificando el código de esta forma. Un ejemplo:

```
PUSH par1 PUSH par2 PUSH par3 PUSH par4 CALL NAME_PROC@N ; N - Number of bytes sent to the stack
Pasa a ser:
INVOKE NAME - PROC, par4, par3, par2, par1
```

9.

Los archivos.**inc** son archivos de texto que contienen declaraciones, funciones, información y/o constantes las cuales son utilizadas en el código fuente.
Los archivos. **lib** contienen una librería de información usada por un programa en especifico, entre lo cual pueden estar funciones, constantes e incluso objetos y media.
Y dado que la directiva **INVOKE** se encarga de hacer el llamado a procedimientos mediante una expresión-dirección, se suele usar esta para hacer uso de códigos o procedimientos ubicados en estos archivos, permitiendo de esta forma un desarrollo modular o funcional del código fuente.

10.

Esta directiva informa al enlazador que el modulo que se esta trabajando debe ser enlazado con la librería indicada. Su sintaxis es: `INCLUDELIB libraryname`

11.

En el caso de librerías estáticas, el enlazador copia el código de cada una de las librerías, mientras que el caso de las librerías dinámicas el enlazador de todas formas necesita el nombre de las funciones que se van a llamar y como ubicar el **.dll** en el sistema. La diferencia entre un **.dll** y una **.lib** es si el código de la implementación se encuentra en el binario o no.

Referencias Bibliográficas

Aps, B. S. (2010-2021). INC File [fecha de consulta: 19/7/2021]. https://file.org/extension/inc

flylib.com. (2008-2017). INCLUDELIB [fecha de consulta: 19/7/2021]. https://flylib.com/books/en/4.288.1.9/1/

Irvine, K. R. (2002). Assembly Language for Intel Assembly Language for Intel-Based Computers, 4 Computers, 4th Edition Edition [fecha de consulta: 19/7/2021]. https://www.csie.ntu.edu.tw/~acpang/course/asm_2004/slides/chapt_08Solve.pdf

laurencejackson. (2015). Microsoft's Old API Help File Reborn [[fecha de consulta: 19/7/2021]].

Microsoft. (2017). Identifying Functions in DLLs [fecha de consulta: 19/07/2021]. https://docs.microsoft.com/en-us/dotnet/framework/interop/identifying-functions-in-dlls

Microsoft. (2021a). INCLUDE [fecha de consulta: 18/7/2021]. https://docs.microsoft.com/es-es/cpp/assembler/masm/include-masm?view=msvc-160

Microsoft. (2021b). INCLUDELIB [fecha de consulta: 18/7/2021]. https://docs.microsoft.com/en-us/cpp/assembler/masm/includelib-masm?view=msvc-160

Microsoft. (2021c). INVOKE [fecha de consulta: 18/7/2021]. https://docs.microsoft.com/es-es/cpp/assembler/masm/invoke?view=msvc-160

Microsoft. (2021d). ML and ML64 command-line reference [fecha de consulta: 18/7/2021]. https://docs.microsoft.com/en-us/cpp/assembler/masm/ml-and-ml64-command-line-reference?view=msvc-160

Productions, S. (2021). .LIB File Extension [fecha de consulta: 19/7/2021]. https://fileinfo.com/extension/lib