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

Practica 2
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

Configuración del "Path"(adicional)

Cabe resaltar que esta parte de la actividad fue realizada para llamar con mas facilidad los programas desde la terminal, dado que para poder LLamar un programa desde la misma se tiene que indicar la ubicación de los archivos los cuales se quieren ejecutar si este no se encuentra en la "variable de entorno" "path" de windows. Para resolver esto se puede realizar lo siguiente:

- Referirse a un "path" explicito, se envía este, ya sea absoluto (desde disco local:... en este caso C:) o relativo (desde donde se "encuentra" la terminal actualmente).
- Configuración de la variable de entorno "path".

El primer método no requiere explicación, solo se debe hacer referencia explicita a la ubicación de los archivos que se quiere llamar cada vez que se requieran (compilador o enlazador respectivamente); El segundo método requiere de la configuración de las variables de estado una sola vez y luego permite referirse al compilador o enlazador de forma directa, sin necesidad de especificar el "path", para esto se hace:

Se selecciona inicio -> panel de control -> sistema:

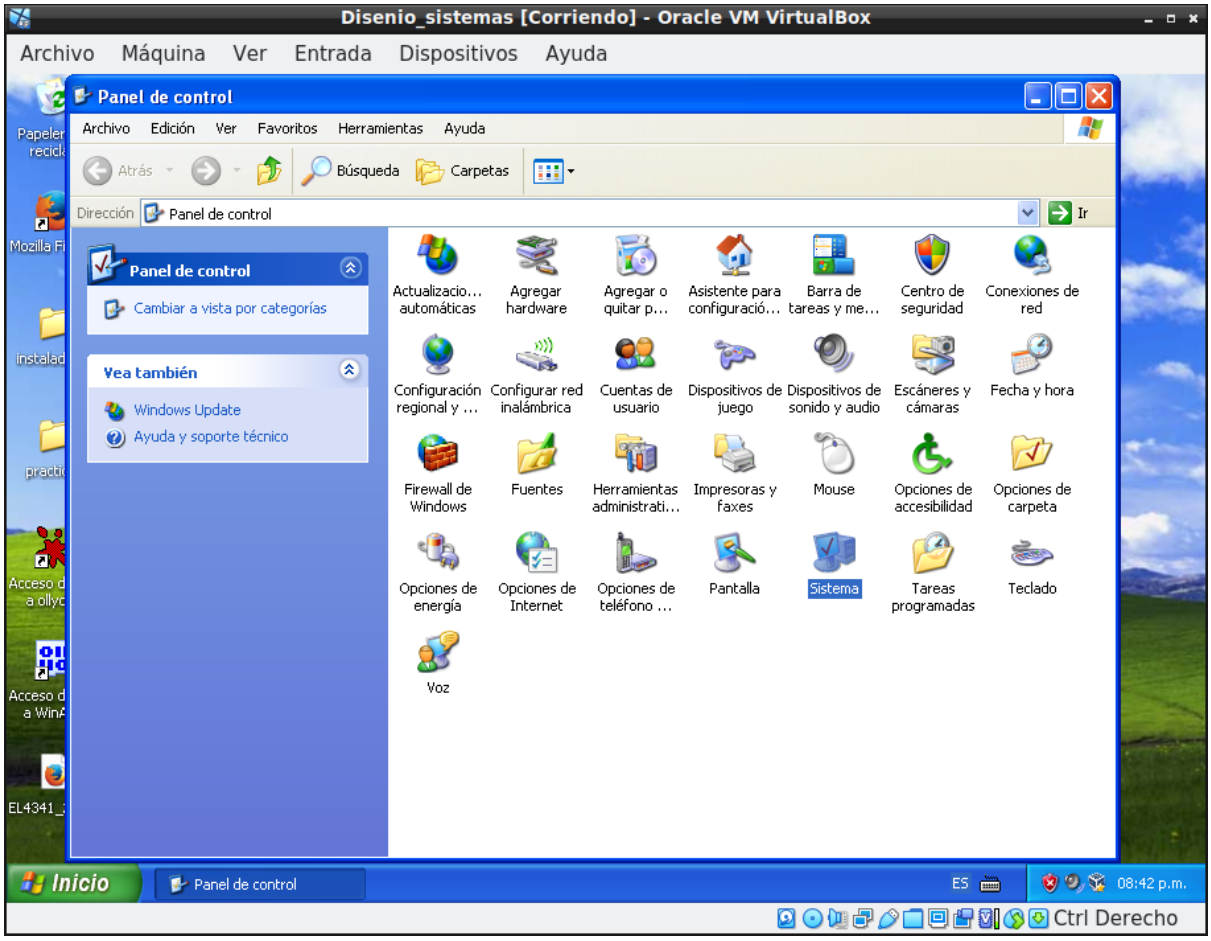

Figura 1: apertura de ventana sistema

Posteriormente se selecciona, de la ventana emergente de sistema, la opción de variables de entorno y

modificar “path”; se debe agregar “;” para indicar un nuevo “path”, y la dirección explicita de la carpeta. en este caso en particular sería: $C:\masm32\bin$

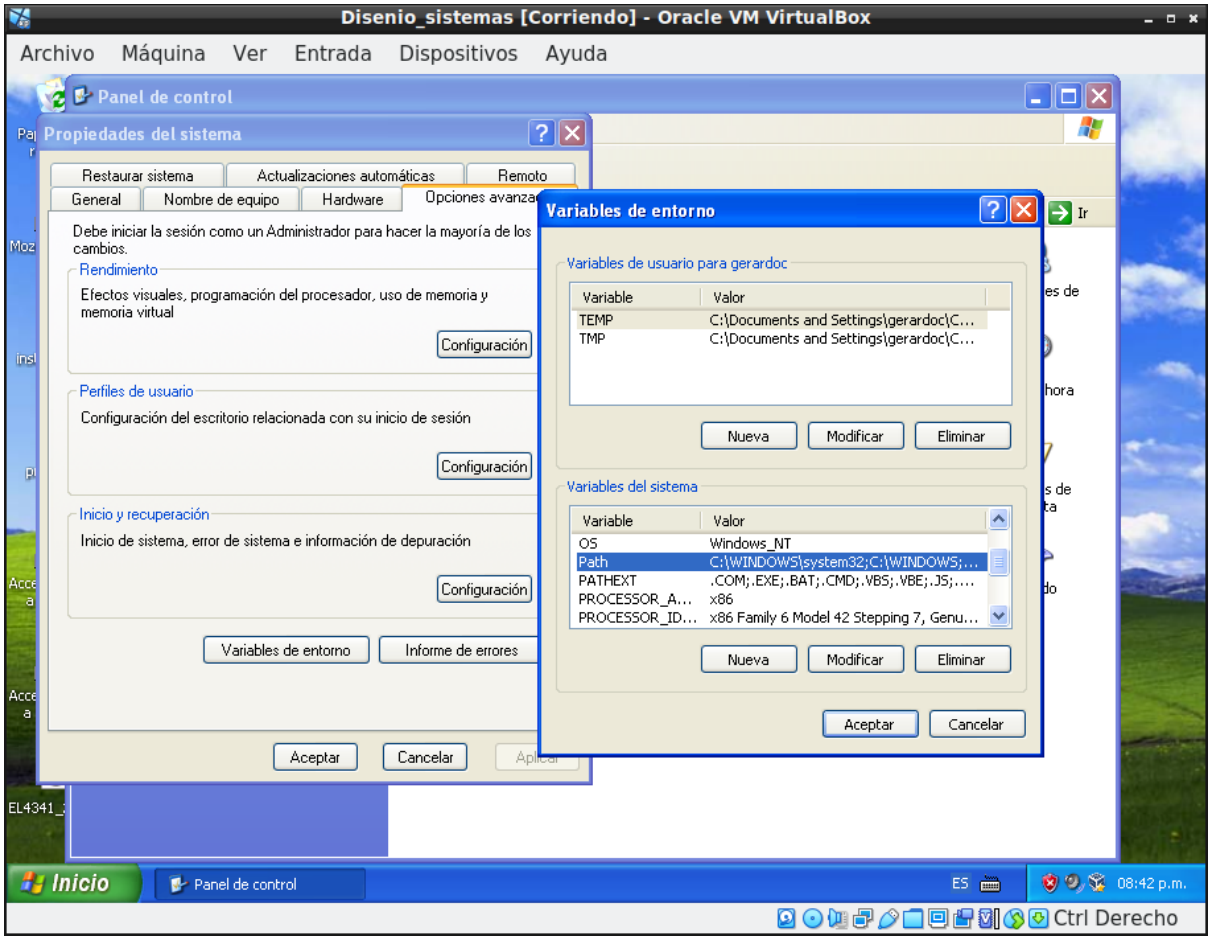

Figura 2: agregar la variable al path

1. Primera parte

1.3 Ensamble el programa usando la opción Make>Assemble.

1.3.1 para compilar desde la terminal desde el directorio C (se especifican los “path”) se usa el comando:

$\Masm32\Bin\ML\ /c\ /I$“$\Masm32\Include$”

“$C:\DocumentsandSettings\gerardoc\Escritorio\practica2\actividad1\act1.asm$”

mas como se incluyo este a las variables de entorno y “path” general de computadora y ademas se uso el comando cd nueva direccion para moverse entre las carpetas hasta el lugar del archivo act1.asm, se uso el comando: “$ml\cact1.asm$ ”

1.3.2 Para este caso particular las únicas opciones (o switches utilizados son:

- /c el cual le indica a WinASM que solamente debe ensamblar el archivo y no enlazarlo.
- /I el cual el programa utiliza automáticamente para ser capaz de incluir el archivo de código fuente ".asm" que se va a ensamblar.

1.3.3 La ventana de salida del WinASM indica la ubicación del ensamblador utilizado, en este caso el "path" es el siguiente "$\backslash Masm32\backslash Bin\backslash ML$". Seguidamente se muestran los switches utilizados, la inclusión del archivo de código fuente.
Luego se visualiza la versión del ensamblador utilizado, para este caso el Microsoft (R) Macro Assembler Version 6.14.8444 junto con un mensaje de copyrigth.
Y finalmente indica el resultado del ensamblado, y los errores si existieron.

1.3.2 El ensamblador produce un archivo objeto, con formato ".obj".

1.4 Enlace el programa usando la opción Make>Link. Note qué archivo es producido por el enlazador.

1.4.1 Con las configuraciones mencionadas anteriormente("path" fijado y en la carpeta del archivo, el comando que se utiliza es: Link16 act1.obj, cabe resaltar que da opciones para:

- cambiar el nombre del ejecutable. Da una opción ("act1.exe" en este caso) por defecto si no se agrega nada.
- incluir otros archivos. "nul.map" por defecto.
- Seleccionar el "path" de las librerías, de nuevo se uso por defecto.
- un archivo de definiciones, "nul.def" por defecto.

1.4.2 Ningún switch es utilizado en este apartado

1.4.3 Adicional a mostrar los archivos que se ejecutan para realizar el "linking", los cuales están en las siguientes direcciones, "$\backslash Masm32\backslash Bin\backslash Link16$" y "$C:\backslash WinAsm\backslash link.war$"
Se muestra la versión del enlazador utilizado (Microsoft (R) Segmented Executable Linker Version 5.60.339 Dec 5 1994) y el mensaje del copyrigth asociado.begingroup.
El archivo ".obj" que se va a enlazar, en este caso "act1.obj"
La finalización o no del proceso y los errores existentes (o ninguno)

1.4.4 El archivo ejecutable del programa “act1.exe”

1.5Ejecute el programa usando la opción Make>Execute.

Al ejecutar el archivo, se abre una ventana de “símbolo del sistema” ubicada en “path” del archivo ejecutable, en la cual se muestra “Hello, World!” y al presionar una tecla se cierra

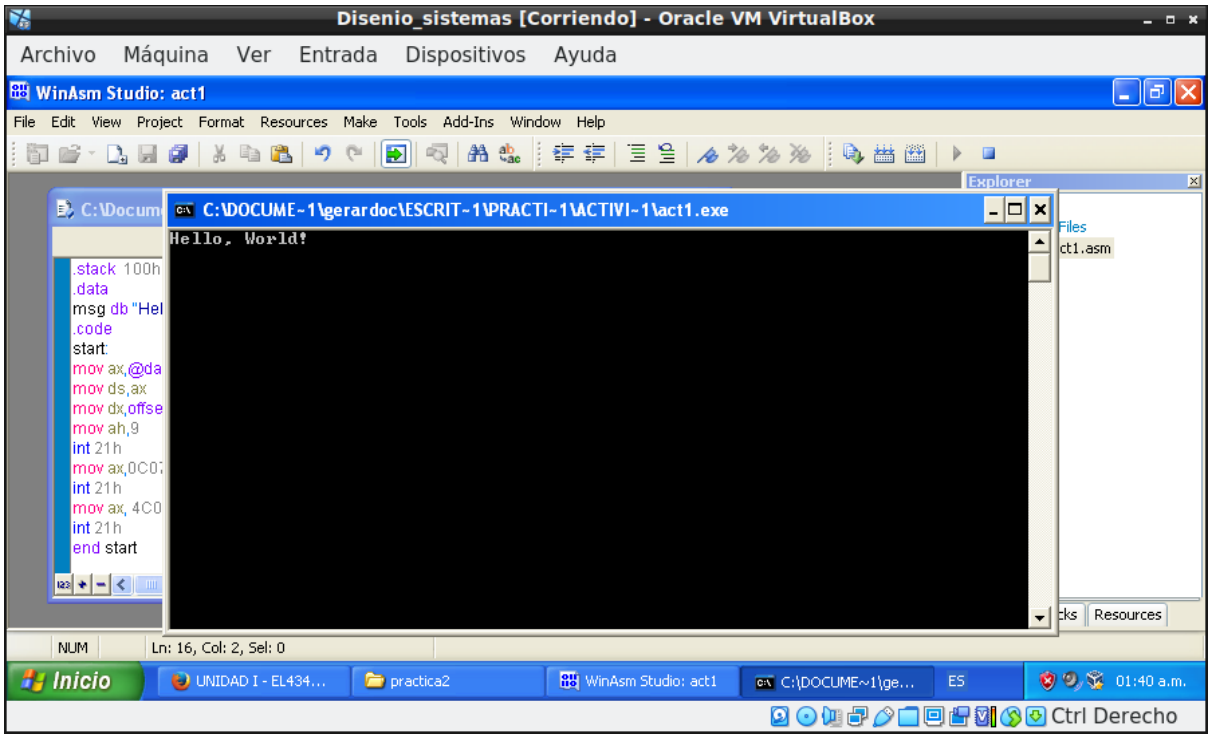

Figura 3: programa1 desde WinAsm

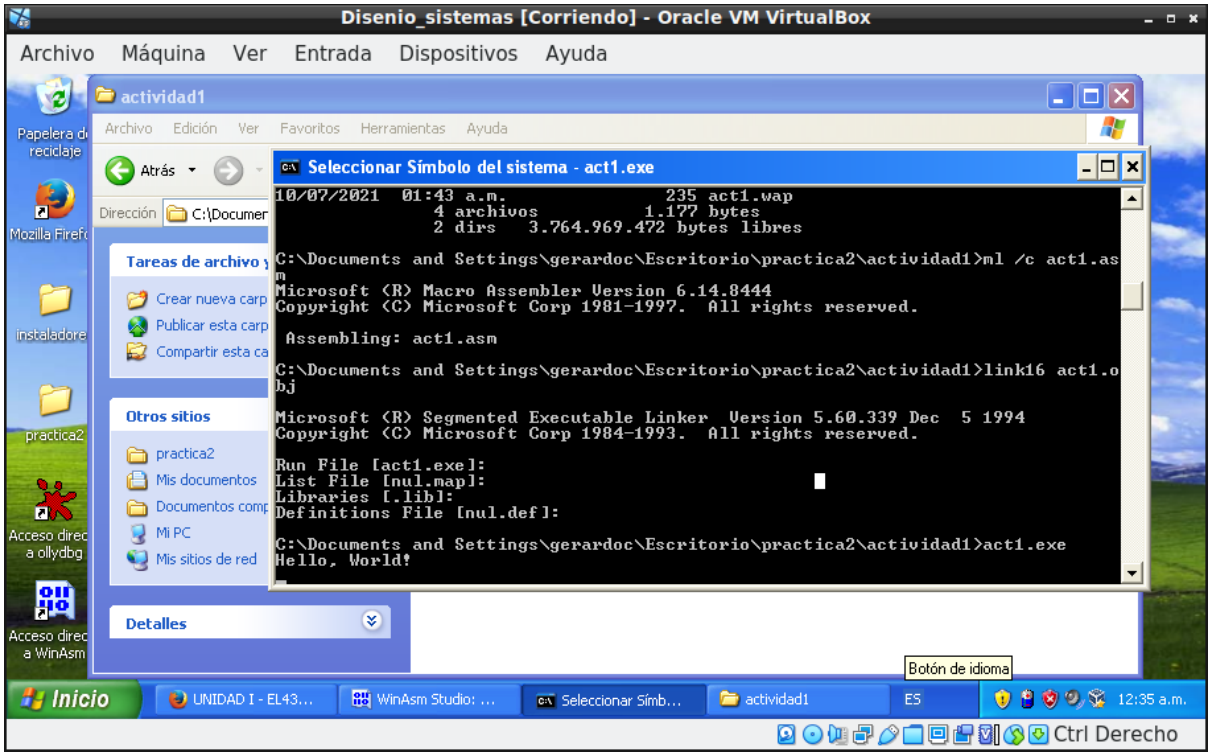

Figura 4: programa1 desde terminal

1.6 Ejecute el programa por medio de OllyDbg.

Al introducir el archivo .exe a Olly, inmediatamente aparece un mensaje que advierte que el archivo probablemente no es un ejecutable portátil de 32 bits. Al abrir se muestran las ventanas regulares del Olly, donde se muestra la memoria, instrucciones y valores actuales de los registros, cabe resaltar que es bastante dificil de interpretar. Adicionalmente cuando se procede a correr el programa, se observa que el sistema se detiene automaticamente en la salida de cada llamada, a subsistema o fin definitivo, y por ende hay que precionar varias veces el boton de continuar para ver el mensaje en pantalla.

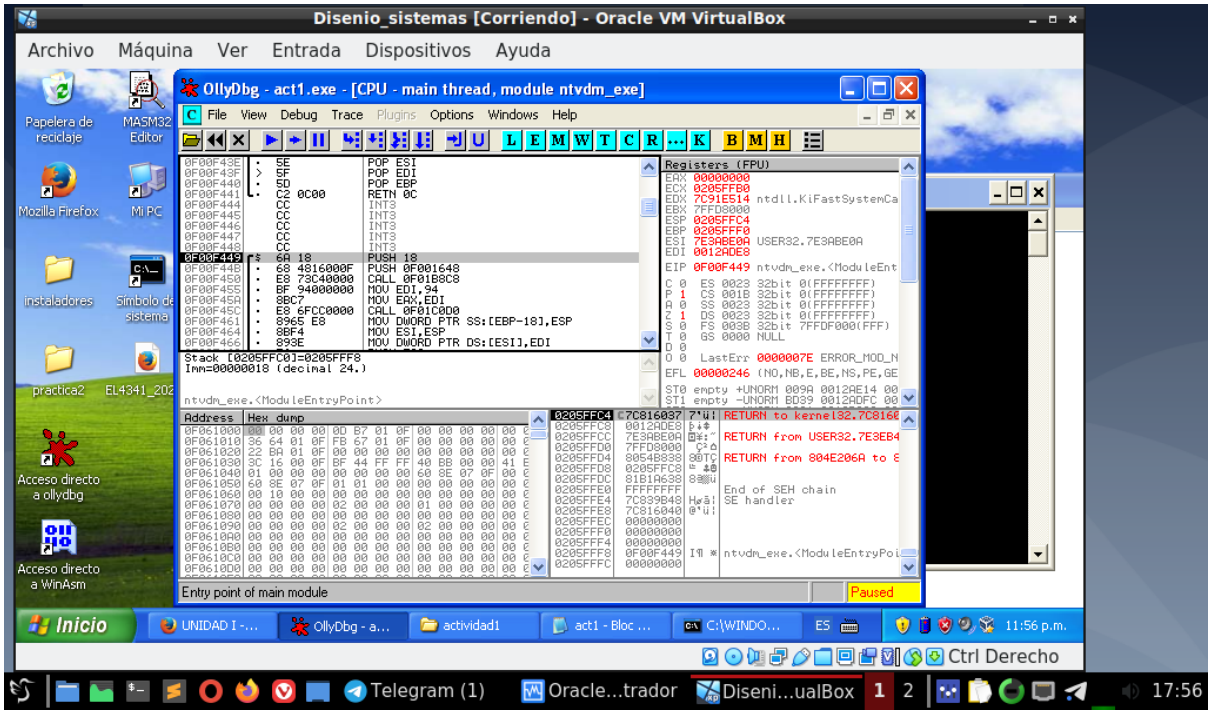

Figura 5: programa1 desde Olly inicio

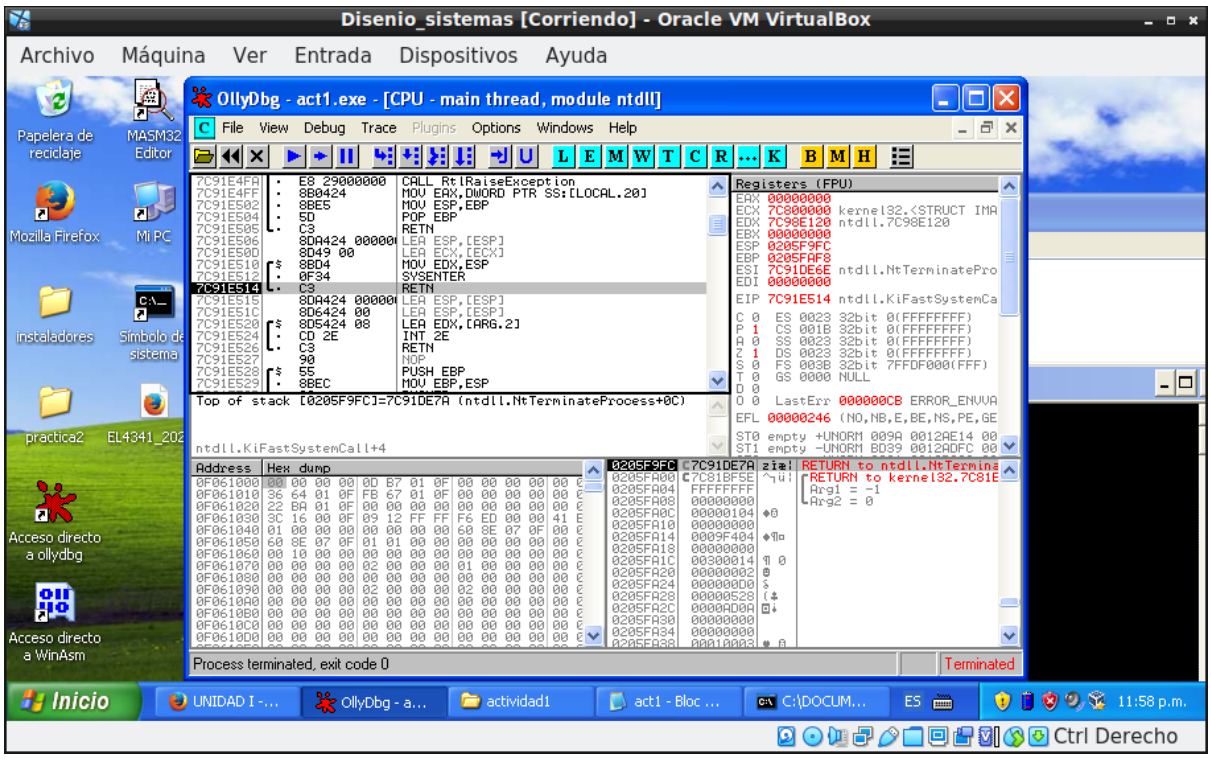

Figura 6: programa1 desde Olly fin

2. Segunda parte

2.3 Ensamble el programa usando la opción Make>Assemble.

2.3.1 Similarmente para compilar desde la terminal, en el directorio del proyecto se usa el siguiente comando:
$ml\ /c$ "$act2.asm$"

2.3.2 Para este caso la única opción (o switch utilizado es):

- /c el cual le indica a MASM que solamente se debe ensamblar el archivo y no enlazarlo.

2.3.3 Similar al caso anterior, se muestra que se ensamblo correctamente el archivo, mostrando su nombre.

2.3.4 Se produce un unico archivo, act2.obj.

2.4 Enlace el programa uando la opción Make>Link

2.4.1 Para enlazar el programa desde la terminal, se usa el siguiente comando $link16\ act2.obj$, , , , ,. Las comas se agregan para evitar que el programa solicite los archivos adiccionales.

2.4.2 No se utiliza ningun opcion en la linea de comandos.

2.4.3 Ocurre un error, debido a que el programa usa win32 y otras librerias de 32 bits no compatibles con ms-dos.

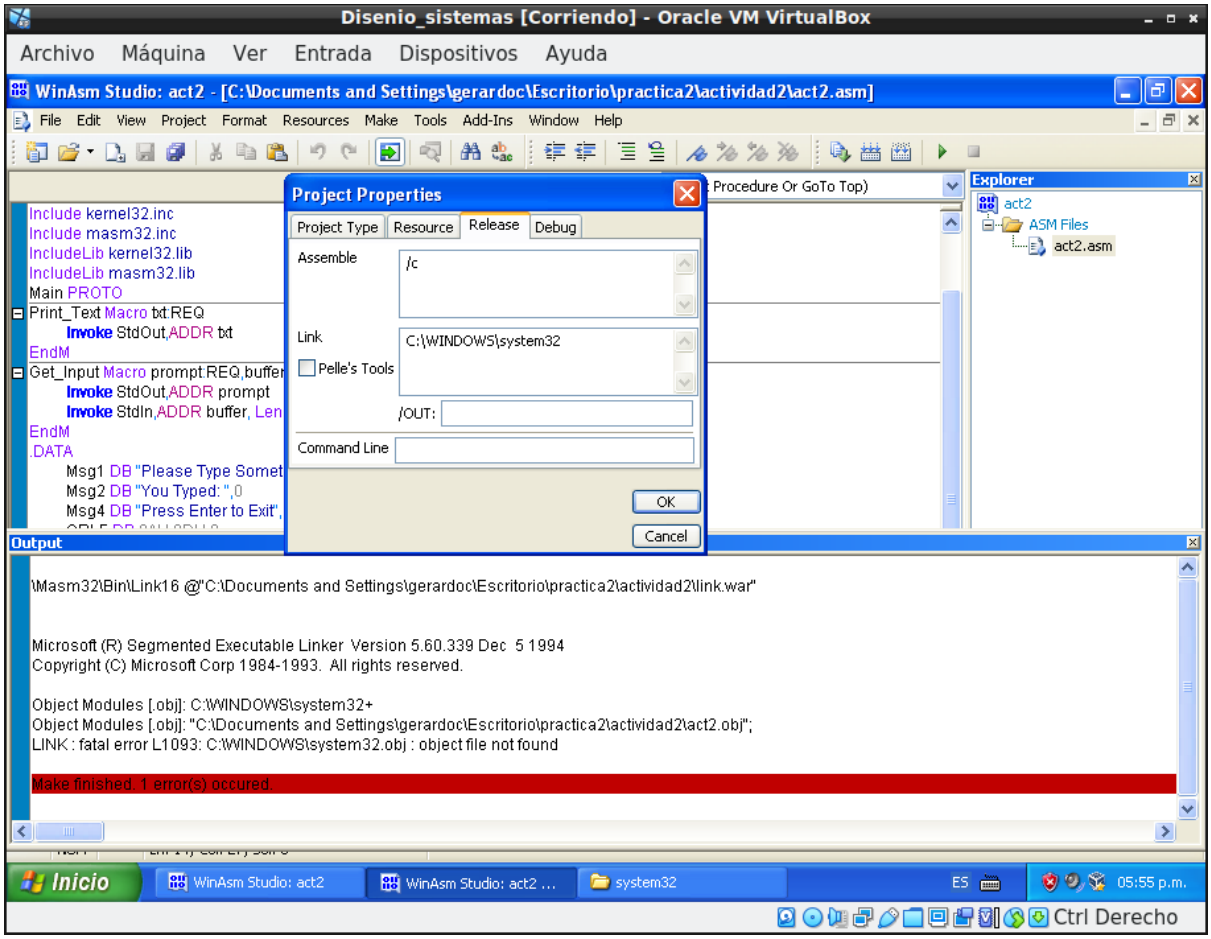

Figura 7: error programa 2

2.4.4 Ninguna, el programa no se pudo enlazar.

2.5 Como no se creo ningun programa, ejecutar no produce ninguna acción.

2.6 No existe ningun programa para depurar.

3. Tercera parte

3.3 Ensamble el programa usando la opción Make>Assemble.

3.3.1 Para compilar desde la terminal, en el directorio del proyecto se usa el siguiente comando:

$ml\ /c/coff\ \text{“}act1.asm\text{”}$.

3.3.2 Las opciones usadas (o switches utilizados) son:

- /c El cual le indica a MASM que solamente se debe ensamblar el archivo y no enlazarlo.
- /coff Esta Opcion se utiliza para indicar que queremos que el archivo ensamblado sea de tipo coff y no de tipo OMF (usado en 16 bits)

3.3.3 Produce un error de ensamblaje, debido a que a el programa no es compatible con la opcion coff. Sin esta opcion no se puede enlazar como aplicacion de consola y esto ocurre porque el programa es de ms-dos y no es compatible con windows NT.

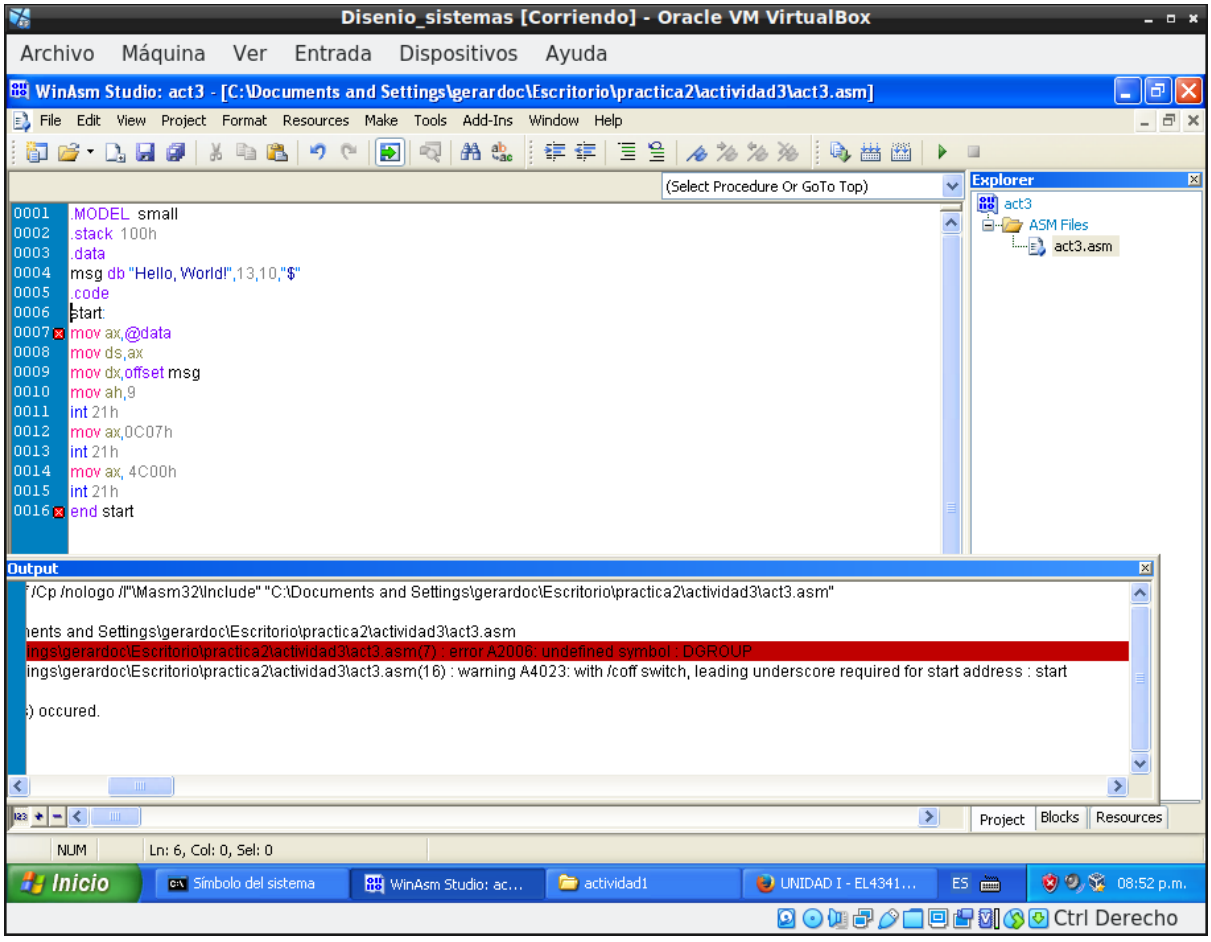

Figura 8: error ensamblaje programa 1

3.3.4 No se produce ningun archivo

3.4 Enlace el programa uando la opción Make>Link

3.4.1 Para enlazar el programa desde la terminal, se usaria el siguiente comando $link\ act1.obj\ /SUBSTEM:CONSOLE\ \ldots$

3.4.2 No se utiliza ningun opcion en la linea de comandos. Sin embargo, no se pudo ensamblar el programa por lo tanto no se puede enlazar

3.4.3 No existe arhcivo para enlazar

3.4.4 Ninguna, el programa no se pudo enlazar.

3.5 Como no se creo ningun programa, ejecutar no produce ninguna acción.

3.6 No existe ningun programa para depurar.

4. Cuarta parte

4.3 Ensamble el programa usando la opción Make>Assemble.

4.3.1 se usa el siguiente comando desde la terminal:
$ml\ /c\ /coff\ /Cp\ /nologo\ /I$“$\backslash Masm32\backslash Include$” $act4.asm$
mas, WinAsm usa:
$\backslash Masm32\backslash Bin\backslash ML\ /c\ /coff\ /Cp\ /nologo\ /I$“$\backslash Masm32\backslash Include$”
“$C:\backslash DocumentsandSettings\backslash gerardoc\backslash Escritorio\backslash practica2\backslash actividad4\backslash act4.asm$”

4.3.2 los switches usados son:

- /c para indicar que solamente se va a ensamblar.
- /coff para generar un archivo objeto de formato .coff .
- /Cp para preservar las mayusculas/minusculas de todos los identificadores de usuario, case sensitive.
- /nologo para suprimir los mensajes de ensamblado exitoso.

4.3.3 La salida es:

```
Assembling:
C:\Documents and Settings\gerardoc\Escritorio\practica2\actividad4\act4.asm
```

```
***********
ASCII build
***********


Make finished. 0 error(s) occured.
```

la cual, como se observa no da informacion adicional a la explicada anteriormente

4.3.4 El archivo generado es act4.obj, un archivo de tipo objeto

4.4 Enlace el programa usando la opción Make>Link.

4.4.1 El comando usado fue:

```
Link /SUBSYSTEM:CONSOLE /RELEASE /VERSION:4.0 "/LIBPATH:\Masm32\Lib"
"C:\Documents and Settings\gerardoc\Escritorio\practica2\actividad4\act4.obj"
"/OUT:C:\Documents and Settings\gerardoc\Escritorio\practica2\actividad4\act4.exe"
```

4.4.2 Los switches utilizados fueron:

- /SUBSYSTEM:CONSOLE: Para indicarle al Sistema operativo que debe correr el .exe con la consola
- /RELEASE: Para fijar el 'checksum' de la cabecera del .exe
- /VERSION:4.0: Para asignarle una versión

4.4.3 La salida es:

```
\Masm32\Bin\Link
@"C:\Documents and Settings\gerardoc\Escritorio\practica2\actividad4\link.war"

Microsoft (R) Incremental Linker Version 5.12.8078
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.
```

```
/SUBSYSTEM:CONSOLE /RELEASE /VERSION:4.0 "/LIBPATH:\Masm32\Lib"
"C:\Documents and Settings\gerardoc\Escritorio\practica2\actividad4\act4.obj"
"/OUT:C:\Documents and Settings\gerardoc\Escritorio\practica2\actividad4\act4.exe"

Make finished. 0 error(s) occured.
```

que no es mas que el comando, mensaje de copyright y finalizacion exitosa

4.4.4 Genera el archivo ejecutable act4.exe

4.5 Al ejecutar el programa, se abre una ventana de consola de Windows, la cual muestra “Please type something: “. El programa en este punto espera una entrada de texto del usuario, al ser introducida y presionada la tecla enter, el programa muestra “You typed: “y a continuación se muestra el texto escrito en el paso anterior. Luego de esto, la consola permanece mostrando esos mensajes hasta que se presione enter para cerrar la ventana.

desde WinAsm:

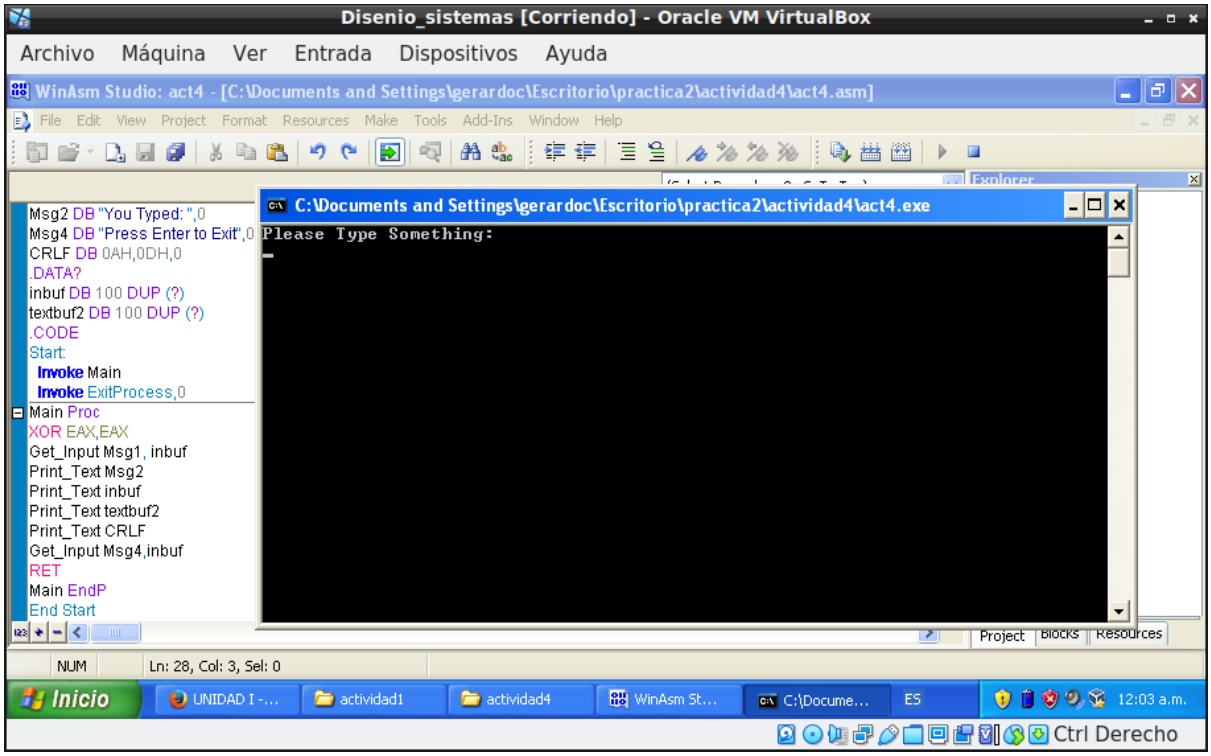

Figura 9: parte 1: programa 4 desde WinAsm

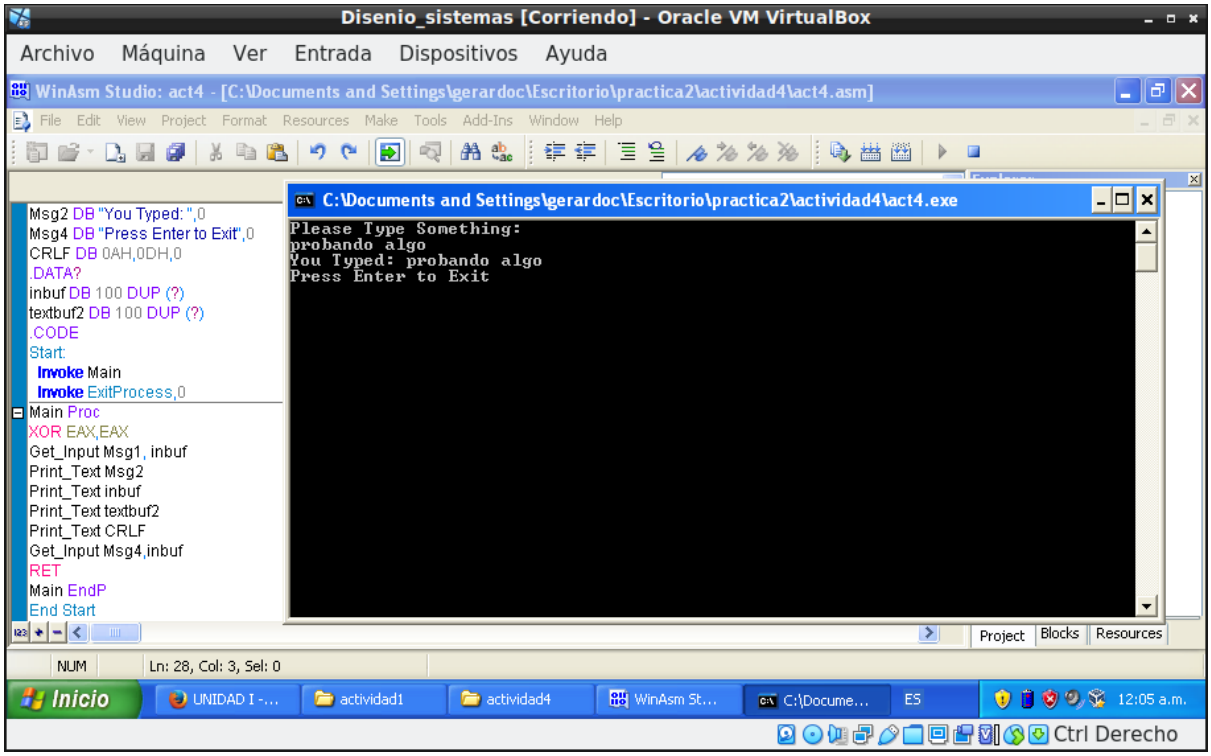

Figura 10: parte 2: programa 4 desde WinAsm

desde la terminal:

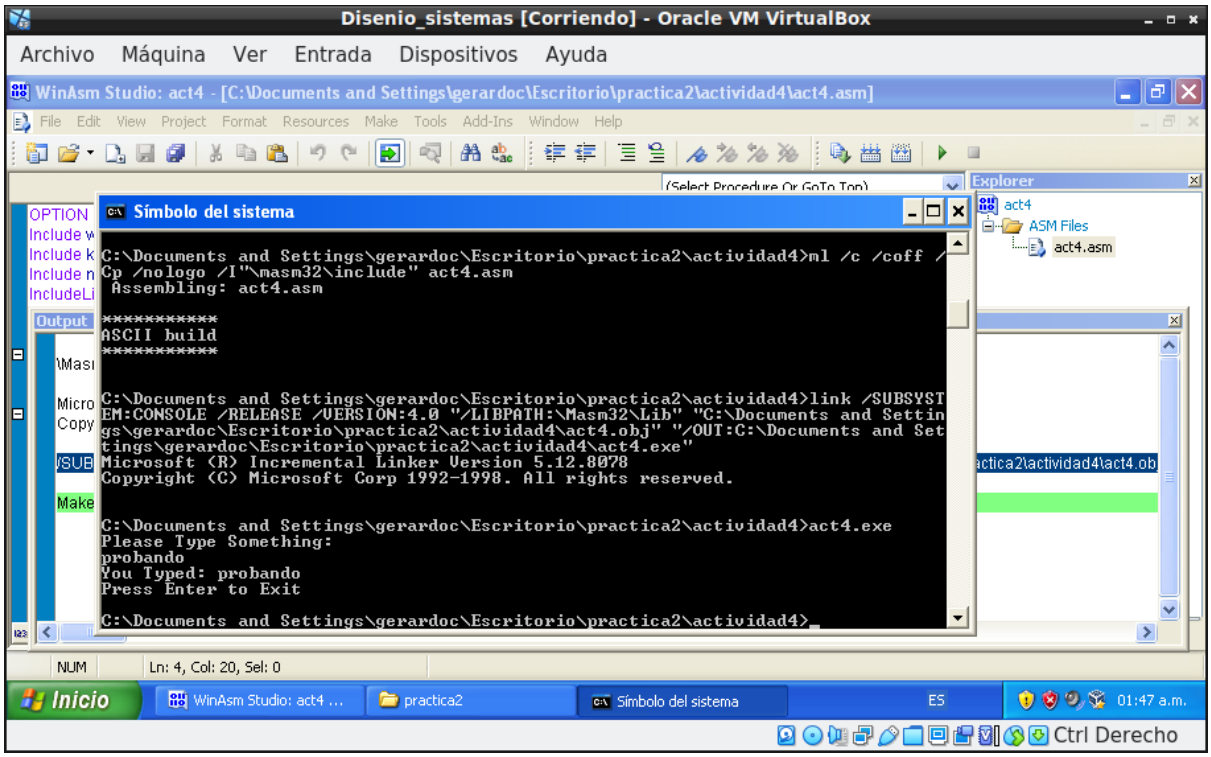

Figura 11: programa 4 desde la terminal

4.6 Este programa es un poco mas entendible, y si se ejecuta sin breakpoints solo se

detiene para esperar las entradas de usuario.

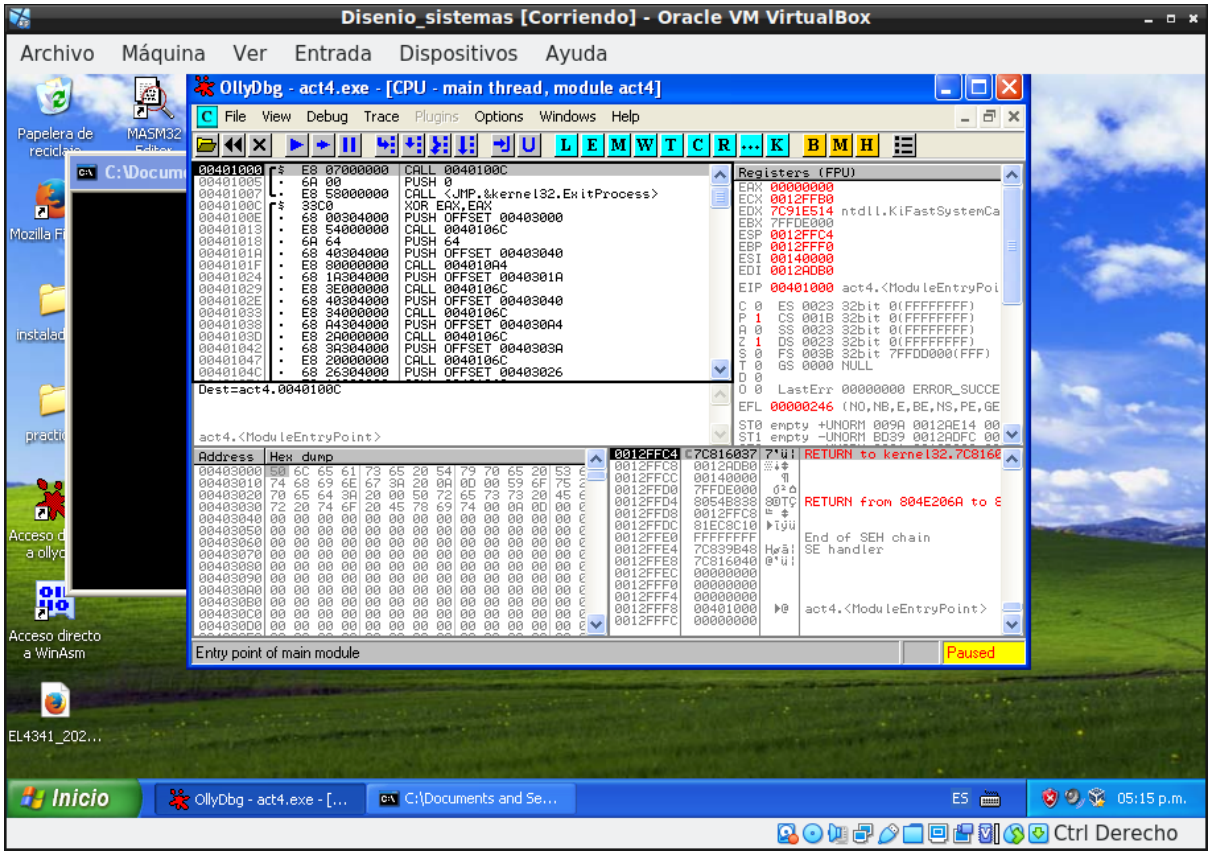

Figura 12: programa 4 desde Olly

5. Conclusiones

1) Ambas aplicaciones funcionan mediante una interfaz de texto, ambas tienen el concepto de standard input y standard ouput, y ambas aplicaciones intereactuan con el sistema operativo mediante APIs de bajo nivel. Sin embargo, las aplicaciones de ms-dos se comunican con el sistema operativo mediante interrupciones, mientras que las de consola cuenta con el API win32; Las aplicaciones de ms-dos son de 16 bits, mientras que las de consola son de 32 bits y la diferencia mas importante es que las aplicaciones de ms-dos son una emulacion sobre una maquina virtual, debido a que a partir de windows xp se usa el nucleo Windows NT, por lo tanto ambos tipos de aplicaciones funcionan con nucleos diferentes.

2) Las aplicaciones de consola pueden parecer una reliquia del pasado, sin embargo son una forma eficiente de comunicarse con aplicaciones para usuarios mas avanzados con el uso de un ordenador y han estado ganando popularidad en los ultimos años. En el caso particular de compilar programas de ensamblador, la terminal es una forma eficiente

de compilar proyectos cuando se conocen todas las opciones del ensamblador, ademas de permitir automatizar la construccion de proyectos con el uso de programas de batch.