Universidad Nacional Experimental Politécnica

“Antonio José de Sucre”

Vicerrectorado Barquisimeto

Departamento de Ingeniería Electrónica

Lab. de Diseño de Sist. De Computación: Práctica 2

Autor:

Alejandro Aguilar

Expediente: 20142-0001

Semestre: 2019-2

Barquisimeto, noviembre de 2019

Primera parte

1.3.1

\Masm32\Bin\ML /c /I"\Masm32\Include" "C:\WinAsm\Practica 2a.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 de hecho no se escoge, sino que el programa lo 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 "\Masm32\Bin\ML". Seguidamente se muestran los switches utilizados, se muestra la inclusión del archivo de código fuente, cuya ubicación es "\Masm32\Include" "C:\WinAsm\Practica 2a.asm"

Luego se visualiza la versión del ensamblador utilizado, para este caso el Microsoft (R) Macro Assembler Version 6.14.8444, y por último, indica la finalización exitosa (o no) del ensamblado, con los errores existentes (o ninguno)

1.3.4

El ensamblador produce un archivo con formato '.obj'

1.4.1

\Masm32\Bin\Link16 @"C:\WinAsm\link.war"

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,

"\Masm32\Bin\Link16" y "C:\WinAsm\link.war",

- Se muestra la versión del enlazador utilizado (Microsoft (R) Segmented Executable Linker Version 5.60.339 Dec 5 1994)
- El archivo 'obj' que se va a enlazar, en este caso "Practica 2a.objj"
- La finalización o no del proceso y los errores existentes (o ninguno)

1.4.4

El archivo ejecutable del programa “Practica1.exe”

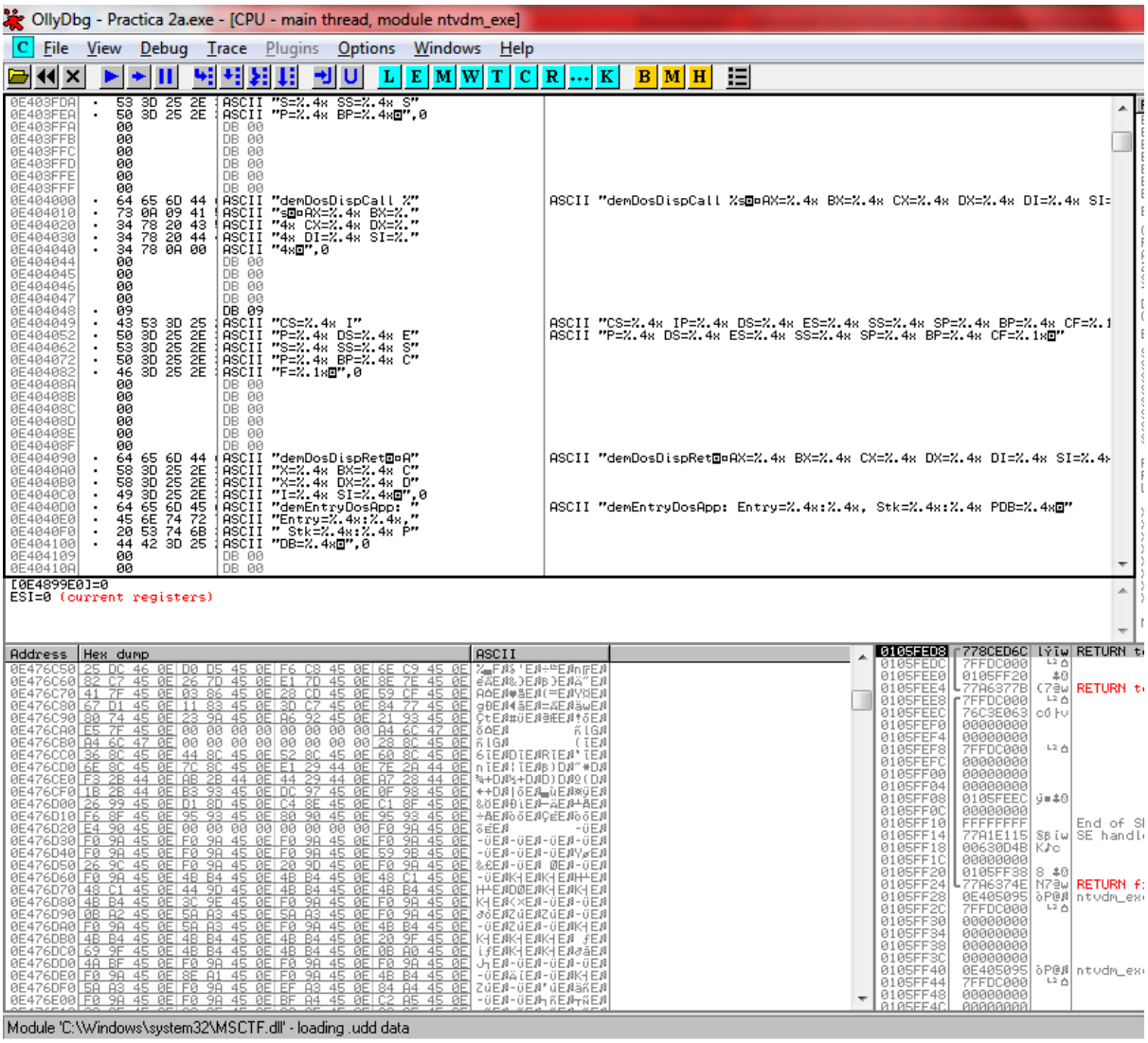
1.5

Al ejecutar el archivo, se abre una ventana parecida a la de “cmd” de Windows, en la cual se muestra “Hello, World!”. Al presionar cualquier tecla la misma se cierra

1.6

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 finalmente se muestran las ventanas regulares del Olly, donde se muestra la memoria, instrucciones y valores actuales de los registros, sin embargo, ninguna de estas 3 cosas es fácilmente legible o interpretable.



Segunda parte

2.3.1

\Masm32\Bin\ML /c /I"\Masm32\Include" "C:\WinAsm\Practica 2b.asm"

2.3.2

Las opciones utilizadas para ensamblar son las mismas que en el apartado anterior, “/c” y “/I”, para permitir solamente el ensamblado y para incluir el archivo de código fuente, respectivamente.

2.3.3

La información en la ventana de salida es similar al apartado anterior, sin embargo, con este código aparece por primera vez en la salida lo siguiente

****

ASCII build

****

2.3.4

El archivo producido por el ensamblador de igual forma es un archivo con extensión “.obj”

2.4.1

\Masm32\Bin\ML /c /I"\Masm32\Include" "C:\WinAsm\Practica 2b.asm"

2.4.2

De igual forma no se utilizó ningún switch.

2.4.3

La ventana de salida completa muestra lo siguiente:

\Masm32\Bin\Link16 @"C:\WinAsm\link.war"

Microsoft (R) Segmented Executable Linker Version 5.60.339 Dec 5 1994

Object Modules [.obj]: C:\WinAsm\Practica 2b.objj

LINK: fatal error L1104: \masm32\lib\kernel32.lib : not valid library

Make finished. 1 error(s) occured.

2.4.4

No se genera un archivo ejecutable, por no permitir enlazar los archivos

2.5

No se puede ejecutar el archivo porque no existe archivo .exe

2.6

No se puede visualizar en ollydbg porque no se tiene archivo ejecutable

Tercera parte

3.3.1

\Masm32\Bin\ML /c /coff /Cp /nologo /I"\Masm32\Include" "C:\WinAsm\Practica 2c.asm"

3.3.2

Los switches utilizados son

- "/c" para indicar que solamente se va a ensamblar
- "/coff" para generar un archivo de objeto de formaro .coff en lugar de .obj
- "/Cp" para preservar las mayusculas/minusculas de todos los identificadores de usuario
- "/nologo" para suprimir los mensajes de ensamblado exitoso

3.3.3

La salida completa al intentar ensamblar este codigo es:

\Masm32\Bin\ML /c /coff /Cp /nologo /I"\Masm32\Include" "C:\WinAsm\Practica 2c.asm"

Assembling: C:\WinAsm\Practica 2c.asm

C:\WinAsm\Practica 2c.asm(7) : error A2006: undefined symbol : DGROUP

C:\WinAsm\Practica 2c.asm(16) : warning A4023: with /coff switch, leading underscore required for start address : start

Make finished. 2 error(s) occured.

3.3.4

El archivo generado es "Practica3.obj"

En primer lugar se da la dirección del ensamblador, se visualizan también los switches utilizados y se incluye el codigo fuente a ensamblar.

Luego en este caso tenemos 2 errores, el segundo, indica que utilizando el switch “/coff” es necesario un “_” para la dirección de inicio “start”, así que al cambiar en el código fuente “start” por “_start” se resuelve este error.

3.4.1

\Masm32\Bin\Link @"C:\WinAsm\link.war"

3.4.2

Los switches utilizados son los siguientes

- /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

3.4.3

Lo siguiente, es la salida completa:

\Masm32\Bin\Link @"C:\WinAsm\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:\WinAsm\Practica 2c.obj" "/OUT:C:\WinAsm\Practica 2c.exe"

LINK : fatal error LNK1181: cannot open input file "C:\WinAsm\Practica 2c.obj"

Make finished. 1 error(s) occurred

Se aprecia que hay un error al no poder abrir el archivo de entrada, que realmente ya explicamos

3.4.4

Ningún archivo porque el código fuente no se pudo ni siquiera ensamblar, en otras palabras, no se pudo generar un archivo Practica2c.obj el cual enlazar

3.5

El programa no se ejecuta, simplemente sale un mensaje emergente que indica que la ejecución falló

3.6

Este programa no tiene archivo ejecutable, por lo tanto, no se puede debuggear en ollydbg

Cuarta parte

4.3.1

\Masm32\Bin\ML /c /coff /Cp /nologo /I"\Masm32\Include" "C:\WinAsm\Practica 2d.asm"

4.3.2

Los switches utilizados son

- "/c" para indicar que solamente se va a ensamblar
- "/coff" para generar un archivo de objeto de formaro .coff en lugar de .obj
- "/Cp" para preservar las mayusculas/minusculas de todos los identificadores de usuario
- "/nologo" para suprimir los mensajes de ensamblado exitoso

4.3.3

\Masm32\Bin\ML /c /coff /Cp /nologo /I"\Masm32\Include" "C:\WinAsm\Practica 2d.asm"

Assembling: C:\WinAsm\Practica 2d.asm

****

ASCII build

****

Make finished. 0 error(s) occured.

Como tal, no se muestra ninguna información relevante adicional a la explicada anteriormente

4.3.4

El archivo generado por el ensamblador es el archivo objeto, "Practica4d.obj"

4.4.1

\Masm32\Bin\Link @"C:\WinAsm\link.war"

4.4.2

Los switches utilizados son los siguientes

- /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

\Masm32\Bin\Link @"C:\WinAsm\link.war"

Microsoft (R) Incremental Linker Version 5.12.8078

/SUBSYSTEM:CONSOLE /RELEASE /VERSION:4.0 "/LIBPATH:\Masm32\Lib" "C:\WinAsm\Practica 2d.obj" "/OUT:C:\WinAsm\Practica 2d.exe"

Make finished. 0 error(s) occured.

4.4.4

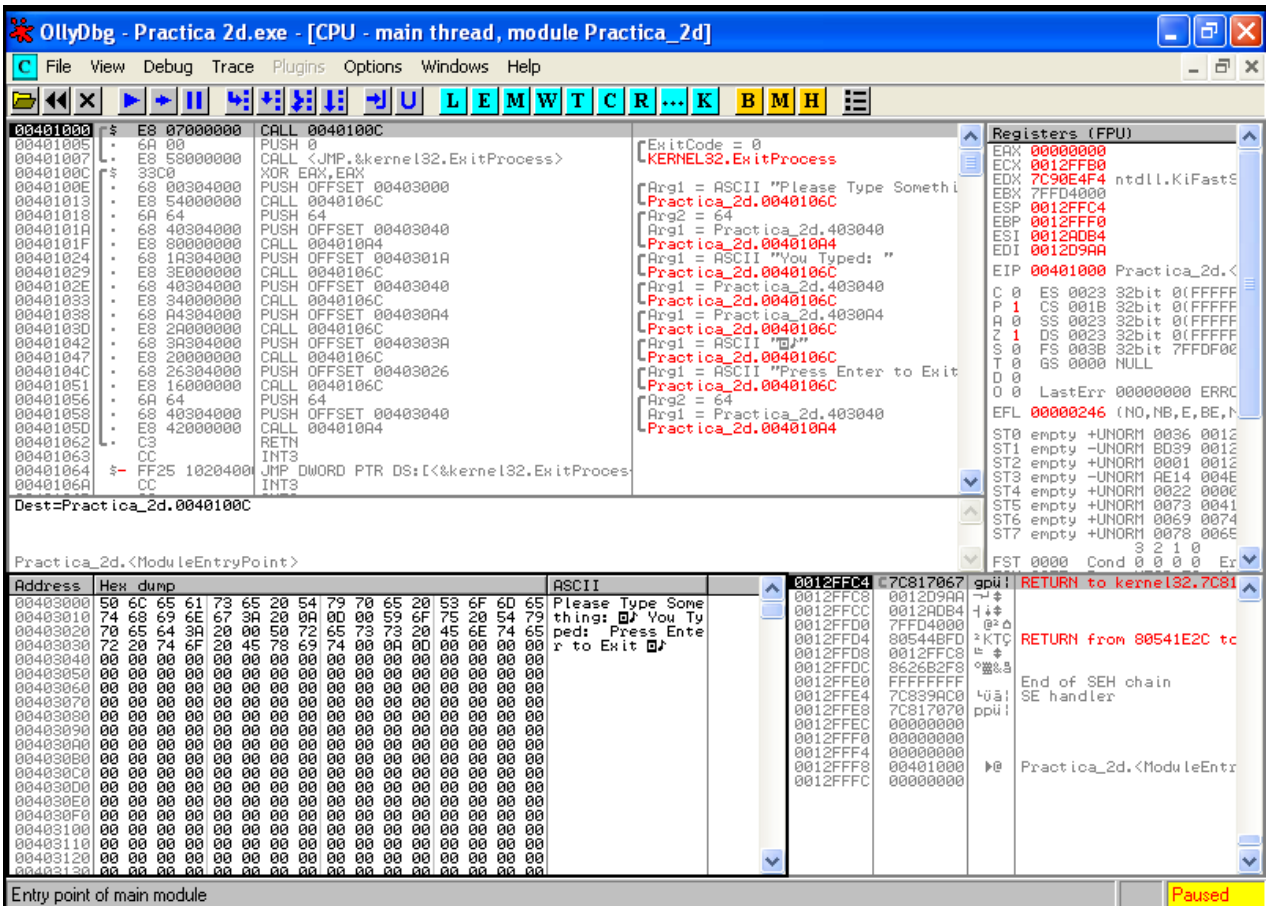
El linker genera el archivo ejecutable del programa, “Practica2d.exe”

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.

4.6

En este programa si se puede entender más claramente la serie de eventos que suceden. Se tiene una visión más clara de la memoria, de las instrucciones y de los registros.



Conclusiones:

En el primer apartado, no se tienen inconvenientes al ensamblar, enlazar, ni ejecutar el archivo debido a que el tipo de proyecto utilizado (DOS Project) es compatible con la totalidad de las instrucciones y registros utilizados.

DOS es a menudo confundido con las aplicaciones de consola de Windows, pero realmente una aplicación de consola de Windows es una forma especial de una aplicación de Windows. Windows de 32 bits puede correr programas DOS (16 bits) utilizando una tecnología llamada NT Virtual DOS Machine (NTVDM) (no incluido en versiones de 64 bits de Windows).

Por lo tanto, con el primer apartado, no había ningún inconveniente de compatibilidad porque el código es de 16 bits, y el tipo de proyecto es compatible con 16 bits.

En el segundo apartado, se tiene un código claramente de 32 bits, que se está trabajando bajo un tipo de proyecto DOS, de 16 bits, el cual intenta enlazar, con el enlazador de 16 bits, al archivo objeto creado con librerías de 32 bits, claramente esta operación no será exitosa, por eso se aprecia el error al momento de hacer el “linking” y no se genera el archivo ejecutable.

Hay que resaltar que para evitar un problema recurrente de WinASM hubo que hacerle una pequeña modificación al código fuente, direccionando de manera más especifica las librerías necesarias, quedando el código así:

…

Include \masm32\include\windows.inc
Include \masm32\include\kernel32.inc
Include \masm32\include\masm32.inc
IncludeLib \masm32\lib\kernel32.lib
IncludeLib \masm32\lib\masm32.lib

…

En el tercer apartado, ahora se trabaja bajo un tipo de proyecto de aplicación de consola, las cuales corren en 32 bits, y estamos intentando ensamblar y enlazar un código fuente de 16 bits en el. El problema más específicamente según informaciones del lenguaje, es la referencia que se hace con @data, a la dirección del segmento de memoria utilizado. Siendo "DGROUP" un concepto de 16 bits, es necesario un enlazador de 16 bits (y en este tipo de proyecto se usa el de 32 bits) para completar exitosamente el ensamblaje y enlazamiento, sin embargo, se aprecia como por este inconveniente, no se puede generar el archivo .obj ni mucho menos el .exe.

Para el cuarto y último apartado tenemos un código de 32 bits, trabajando con un tipo de proyecto funcional a 32 bits. En este caso se ve como no existe problema alguno al momento de ensamblar ni enlazar, porque no existen problemas de compatibilidad

Como anexo, a continuación se muestran las diferentes opciones (o switches) al momento de realizar la operación de "assemble" sobre un proyecto provistas en el manual de WinASM

Option	Action
/AT	Enables tiny-memory-model support. Enables error messages for code constructs that violate the requirements for .com format files. Note that this is not equivalent to the .MODEL TINY directive.
/Bl filename	Selects an alternate linker.
/c	Assembles only. Does not link.
/Cp	Preserves case of all user identifiers.
/Cu	Maps all identifiers to uppercase (default).
/Cx	Preserves case in public and extern symbols.
/coff	Generate COFF format object file
/Dsymbol[[=value]]	Defines a text macro with the given name. If value is missing, it is blank. Multiple tokens separated by spaces must be enclosed in quotation marks.
/EP	Generates a preprocessed source listing (sent to STDOUT). See /Sf.
/Fhexnum	Sets stack size to hexnum bytes (this is the same as /link /STACK:number). The value must be expressed in hexadecimal notation. There must be a space between /F and hexnum.
/Fefilename	Names the executable file.
/Fl[[filename]]	Generates an assembled code listing. See /Sf.
/Fm[[filename]]	Creates a linker map file.
/Fofilename	Names an object file.
/FPi	Generates emulator fix-ups for floating-point arithmetic (mixed language only).
/Fr[[filename]]	Generates a source browser .SBR file.
/FR[[filename]]	Generates an extended form of a source browser .SBR file.
/Gc	Specifies use of FORTRAN- or Pascal-style function calling and naming conventions. Same as OPTION LANGUAGE:PASCAL.
/Gd	Specifies use of C-style function calling and naming conventions. Same as OPTION

	LANGUAGE:C.
/Gz	Specifies use Stdcall calls
/H number	Restricts external names to number significant characters. The default is 31 characters.
/help	Calls QuickHelp for help on ML.
/I pathname	Sets path for include file. A maximum of 10 /I options is allowed.
/nologo	Suppresses messages for successful assembly.
/Sa	Turns on listing of all available information.
/Sc	Adds instruction timings to listing file.
/Sf	Adds first-pass listing to listing file.
/Sg	Turns on listing of assembly-generated code.
/Sl width	Sets the line width of source listing in characters per line. Range is 60 to 255 or 0. Default is 0. Same as PAGE width.
/Sn	Turns off symbol table when producing a listing.
/Sp length	Sets the page length of source listing in lines per page. Range is 10 to 255 or 0. Default is 0. Same as PAGE length.
/Ss text	Specifies text for source listing. Same as SUBTITLE text.
/St text	Specifies title for source listing. Same as TITLE text.
/Sx	Turns on false conditionals in listing.
/Ta filename	Assembles source file whose name does not end with the .asm extension.
/w	Same as /W0.
/Wlevel	Sets the warning level, where level = 0, 1, 2, or 3.
/WX	Returns an error code if warnings are generated.
/X	Ignore INCLUDE environment path
/Zd	Generates line-number information in object file.
/Zf	Makes all symbols public.
/Zi	Generates CodeView information in object file.
/Zm	Enables M510 option for maximum compatibility with MASM 5.1.
/Zp[[alignment]]	Packs structures on the specified byte boundary. The alignment can be 1, 2, or 4.
/Zs	Performs a syntax check only.
/?	Displays a summary of ML command-line syntax.

A continuación, se muestran todas las posibles opciones (o switches) y las acciones que estas ejecutan, provistas por el manual de WinASM

Option	Action
/ALIGN:number	Specifies the alignment of each section
/BASE:{address | @filename,key}	Sets a base address for the program
/COMMENT:["]comment["]	Inserts a comment string into header
/DEBUG	Creates debugging information
/DEBUGTYPE:CV /DEBUGTYPE:COFF /DEBUGTYPE:BOTH	Creates particular formats of debugging information
/DEF:filename	Passes a module-definition (.DEF) file to the linker
/DEFAULTLIB:library	Searches specified library when resolving external references
/DELAY	Controls the delayed loading of DLLs
/DELAYLOAD	Causes the delayed loading of the specified DLL
/DLL	Builds a DLL
/DRIVER[:UPONLY]	Creates a Windows NT kernel mode driver
/ENTRY:function	Sets the starting address
/EXETYPE:DYNAMIC	Builds a virtual device driver
/EXPORT	Exports a function

/FIXED[:NO]	Creates a program that can be loaded only at its preferred base address
/FORCE[:{MULTIPLE|UNRESOLVED}]	Forces link to complete in spite of unresolved or multiply defined symbols
/GPSIZE:#	Specifies the size of communal variables for MIPS and Alpha platforms
/HEAP:reserve[,commit]	Sets the size of the heap in bytes
/IMPLIB:filename	Overrides the default import library name
/INCLUDE:symbol	Forces symbol references
/INCREMENTAL:{YES|NO}	Controls incremental linking
/LARGEADDRESSAWARE	Tells the compiler that the application supports addresses larger than two gigabytes.
/LIBPATH:path	Allows the user to override the environmental library path
/LINK50COMPAT	Generates import libraries in Visual C++ Version 5.0 format
/MACHINE:{IX86|ALPHA|ARM|MIPS| MIPSR41XX|PPC|SH3|SH4}	Specifies the target platform
/MAP	Creates a map file
/MAPINFO:{EXPORTS|FIXUPS|LINES}	Includes the specified information in the map file
/MERGE:from=to	Combines sections
/NODEFAULTLIB[:library]	Ignores all (or specified) default libraries when resolving external references
/NOENTRY	Creates a resource-only DLL
/NOLOGO	Suppresses startup banner
/OPT:{REF|NOREF|ICF[,iterations]|NOICF}	Controls LINK optimizations
/ORDER:@filename	Places COMDATs into the image in a predetermined order
/OUT:filename	Specifies the output file name
/PDB:filename	Creates a program database (.PDB) file
/PDBTYPE:{con[solidate]|sept[ypes]}	Specifies where to store the Program Database (PDB) debug type information.
/PROFILE	Enables profiling (creates a mapfile)
/RELEASE	Sets the checksum in the .EXE header
/SECTION:name,attributes	Overrides the attributes of a section
/STACK:reserve[,commit]	Sets the size of the stack in bytes
/STUB:filename	Attaches an MS-DOS stub program to a Win32 program
/SUBSYSTEM:{CONSOLE|WINDOWS| NATIVE|POSIX|WINDOWSCE} [,major[.minor]]	Tells the operating system how to run the .EXE file
/SWAPRUN:{NET|CD}	Tells the operating system to copy the linker output to a swap file before running it
/VERBOSE[:LIB]	Prints linker progress messages
/VERSION:major[.minor]	Assigns a version number
/VXD	Creates a virtual device driver (VxD)
/WARN[:level]	Specifies warning level
/WS:AGGRESSIVE	Aggressively trim process memory

Algunas referencias utilizadas:

http://moisesrbb.tripod.com/unidad2.htm

https://stackoverflow.com/questions/46829110/alternative-to-the-mov-ax-data-mov-ds-ax-instruction

https://pentiumevolution.blogspot.com/

https://es.wikipedia.org/wiki/Intel_Pentium

http://www.asmcommunity.net/forums/topic/?id=29587

http://masm32.com/board/index.php?topic=1853.0

https://stackoverflow.com/questions/8106893/error-a2006-undefined-symbol-dgroup

https://en.wikipedia.org/wiki/Windows_Console