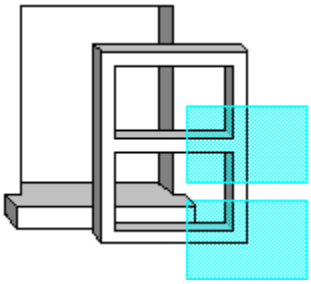
Windows Disassembler 1.7



Introduction

Windows Disassembler disassembles Windows 3.x programs and Dynamic Link Libraries. It displays the assembly language source code of a given executable within a window and allows one to create the assembly language source code files if desired. Version 1.7 disassembles all 486 instructions, assuming that the instructions are intended for 16-bit mode operation.

Commands

Questions, comments, and/or suggestions can be sent to Eric Grass on the Internet via **s876795@umslvma.umsl.edu** or **s876795@slvaxa.umsl.edu**

The Display Window

The display window displays the assembly language code of a segment from the program which you have opened, excluding certain directives (i.e., **TITLE**, **.MODEL**, **.486**, **PROC**, etc.). The code that initially appears in the window when a file is opened is the first segment within the file. Windows Disassembler displays one segment at a time within the window. The View | Segment command must be used to go to another segment. To scroll the text in the window, use the Up Arrow, Down Arrow, Page Up, and Page Down keys, or the scroll bar. To see the address offsets of each instruction, select View | Address Offsets from the main menu. To jump to a specific address, select View | Go To from the main menu and enter the address in hexadecimal format.

The View | Far Call Names command toggles between displaying far function call names and the actual relocation values in far CALL instructions (for example, 0000H:0FFFFH).

All labels have the form of either LxxxxH or DxxxxH, where xxxx is a 4-digit hexadecimal number equal to the offset of the location being referenced. Labels with an 'L' prefix denote locations within the immediate code segment, and labels with a 'D' prefix denote locations within a data segment. Labels within a code segment can either be procedure labels, jump/loop labels, or data labels within the code segment. Assembler directives, while generated for source code text files, are not shown in the display window.

Strings are detected in data segments and translated by Windows Disassembler whenever five or more visible characters occur within a data segment.

At the top of the window is the following information:

Segment number The chronological number of the current segment. (This number is based on the order in which segments are placed in the executable file)

Segment offset A hexadecimal value equal to the segment's offset relative to the beginning of the file

Segment size A hexadecimal value equal to the number of bytes in the segment.

Bytes in the program can be changed from data declarations to instructions and vice versa with the Set Bytes command. The offset of each instruction can be displayed with the Address Offsets command. Only one segment can be displayed at a time. The **Segments** command allows you to choose which segment you want displayed. To create an assembly language file, use the Save Text As command. The **Far Call Names** command toggles between showing the Windows API function call names and the actual relocation bytes.

Creating and Assembling Assembly Language Files

You can create an assembly language file for the program that you've opened by using the Save Text As command. Note, however, that in some instances the source code created by *Windows Disassembler* cannot be reassembled without some editing. Also, you will need the resource files (which can be extracted using Borland's *Resource Workshop*), and a definition file (which can be created with the aid of a program file header utility such as *TDUMP* or *EXEHDR*). You will then need to make the following alterations:

1. *Windows Disassembler* can distinguish **SMALL** programs only. All other types of programs are assumed to have medium models. Change the model type in the **.MODEL** directive to the correct type if it is neither small or medium.
2. Make the data segment accessible to all modules via include files.

The new file will contain tabs. To display the file in the way in which it was intended to be displayed, the user should set his or her editor's tab stop option to 8 spaces.

Windows Disassembler will create **TITLE**, .**CODE** segmentname, .**DATA** segmentname, .**MODEL LARGE**, **.486**, and **EXTRN winAPIfunc:FAR** directives. **PROC** and **ENDP** directives are also created for all exported and far procedures. In the case of non-exported functions, these procedure directives will all have the following form:

```
Functionn    PROC  FAR PUBLIC
             (code)
             RETF
Functionn    ENDP
```

where n is the ordinal number (a decimal integer value) of the procedure in the entry table of the program's executable file header. For exported functions, the name of the function is explicitly written as it is listed in the resident and non-resident names tables in the program's header. For calls to functions in fixed memory segments, a comment is written beside the call. For example,

```
CALL   FAR PTR Proc0AD0HSeg5      ; (Fixed Memory Location)
```

For far calls to procedures within the program in a different segment, **EXTERNDEF**'s are generated. Near procedures are written in the following form:

```
ProcXXXXSegN      PROC  FAR PUBLIC
                  (code)
                  RET
ProcXXXXSegN      ENDP
```

where **XXXX** is a four-digit hexidecimal value equal to the offset of the procedure in the segment and **N** is the decimal number of the segment the procedure is in.

Windows Disassembler generates segment names for segment directives of the form **.CODE SEGn**, where n is the segment number. This name is produced in order to distinguish between segments, and can be deleted or changed. (If the segments are in separate files then the name isn't needed.) If there are exactly 2 segments in a program, *Windows Disassembler* treats the program as having a small model, otherwise it assumes the program has a medium memory model. If the program has a compact or large model, then the **MODEL** directive must be changed to reflect the actual memory model. *Windows Dissassembler* 1.7b translates functions belonging to **commdlg.dll** and **shell.dll**. It also generates information for unknown function calls in the form **Module modulename Ordinal n**. The user can look up the names of these function names using an executable-file header utility on the given dynamic link library. (In other words, one can use the relocation table names and offsets provided by an .exe file header utility to determine the function/variable names in the source code.)

Finally, **EXTRN**'s (or **EXTERNDEF**'s) must be supplied for any far variables used by the program

not already supplied by *Windows Disassembler* (typically the far variable **__winflags** is used by Windows programs, for example).

File Commands

Open Opens a Windows executable file or dynamic link library. The default extension is **.exe**. Windows Disassembler processes one file at a time. If a file is opened while another one is already open, the old file will be automatically closed. When opened, the file's assembly language code appears on the screen, provided that the file has a DOS executable file header, a new executable file header, and at least one segment. Otherwise, a dialog box will inform the user that the file does not meet a particular specification.

Save Text As Creates an assembly language source code file. Three options are available for generating (a) file(s).

OPTIONS:

If the **Create Separate Files For Each Segment** option is checked each segment of the source code is put into a separate file. Each file's name will have an integer appended to the base name of the file equal to the segment's number. That is, each segment's file name will be of the form **yournameN.ext**, where **yourname.ext** is the name the user specifies in the dialog box, and **N** is an integer corresponding to the segment's number and which is appended to the base-name of the file (if necessary, this base name will be truncated to perform the appending). For example, if the user specifies **\work\myprog.asm** as the file name, *Windows Disassembler* will generate files named **\work\myprog1.asm**, **\work\myprog2.asm**, **\work\myprog3.asm**, etc..

If the **Create One File Containing All Segments** option is checked one file will be created containing the whole program. In this case *Windows Disassembler* uses the file name exactly as specified. Select this option only if the program is relatively small, i.e., has only two or three segments. The default file name extension is **.asm**.

If **Save Current Segment Only** is checked, a file is created containing only the segment that is currently being displayed in the window. In this case *Windows Disassembler* uses the file name exactly as specified. The default file name extension is **.asm**. This option preserves the byte settings made for the segment.

NOTES: If a file having the specified name already exists, it will automatically be overwritten. *Saving files erases all byte settings* (see Set Bytes) made in the current segment unless you select the **Save Current Segment Only** option.

Edit Command

Controls the way *Windows Disassembler* translates a (set of) byte(s). If **Byte Data** is selected, all bytes within the specified range are translated as byte declarations. If **Instruction** is selected, all bytes within the given range are translated as machine instructions. The **Labeled Instruction** option allows you to give labels to all instructions within the specified range (note that byte declarations are automatically given labels). Hexadecimal numbers are required for the range values.This command is necessary for programs which have data declarations in their code segments. Note that all modifications which the user has made to a segment will be lost when exiting that segment. The user can save that segment using the **Save Current Segment Only** option as a text file first before quitting to save the changes. However, when the user leaves the segment, there is no way to restore the byte settings except by specifying them over again. **Selecting the Create Separate Files For Each Segment** option will result in the the modifications/settings being erased (lost) before the file is created, hence the user must use the **Save Current Segment Only** option.

View Commands

Go To Goes to a specified address in the current segment (requires a hexadecimal value).

Address Offsets Shows the offset of each instruction in the far left-hand column.

Far Call Names Toggles between showing the Windows function call names and the actual relocation bytes.

Segment Goes to the segment that you specify. Note that all byte format changes which you have made will be lost if you change segments. You can save your byte settings in an assembly language file before you change segments but you cannot retrieve the changes again (except by resetting the bytes manually).

Font Allows the selection of the desired font in which the text in the window is to be written.

Keys

PAGE UP KEY	Scrolls the window up one page.
PAGE DOWN KEY	Scrolls the window down one page.
UP ARROW KEY	Scrolls the window up one line.
DOWN ARROW KEY	Scrolls the window down one line.

Specifications

Files
Works on Windows 3.x executables and dynamic link libraries only.
Instruction Set
Translates all instructions within the 486 instruction set. It assumes that all code is executed in 16-bit mode (since Windows 3.1 uses 16-bit mode only).
Operating System and Hardware
Requires at least DOS 4.0, Windows 3.1, and a 286 or above IBM compatible computer. Installation of SMARTDRV (which comes with Windows) is recommended.