

Practical Malware Analysis

Ch 5: IDA Pro



Last modified
2-6-16

IDA Pro Versions

- Full-featured pay version
- Old free version
 - Both support x86
 - Pay version supports x64 and other processors, such as cell phone processors
- Both have code signatures for common library code in FLIRT (Fast Library identification and Recognition Technology)

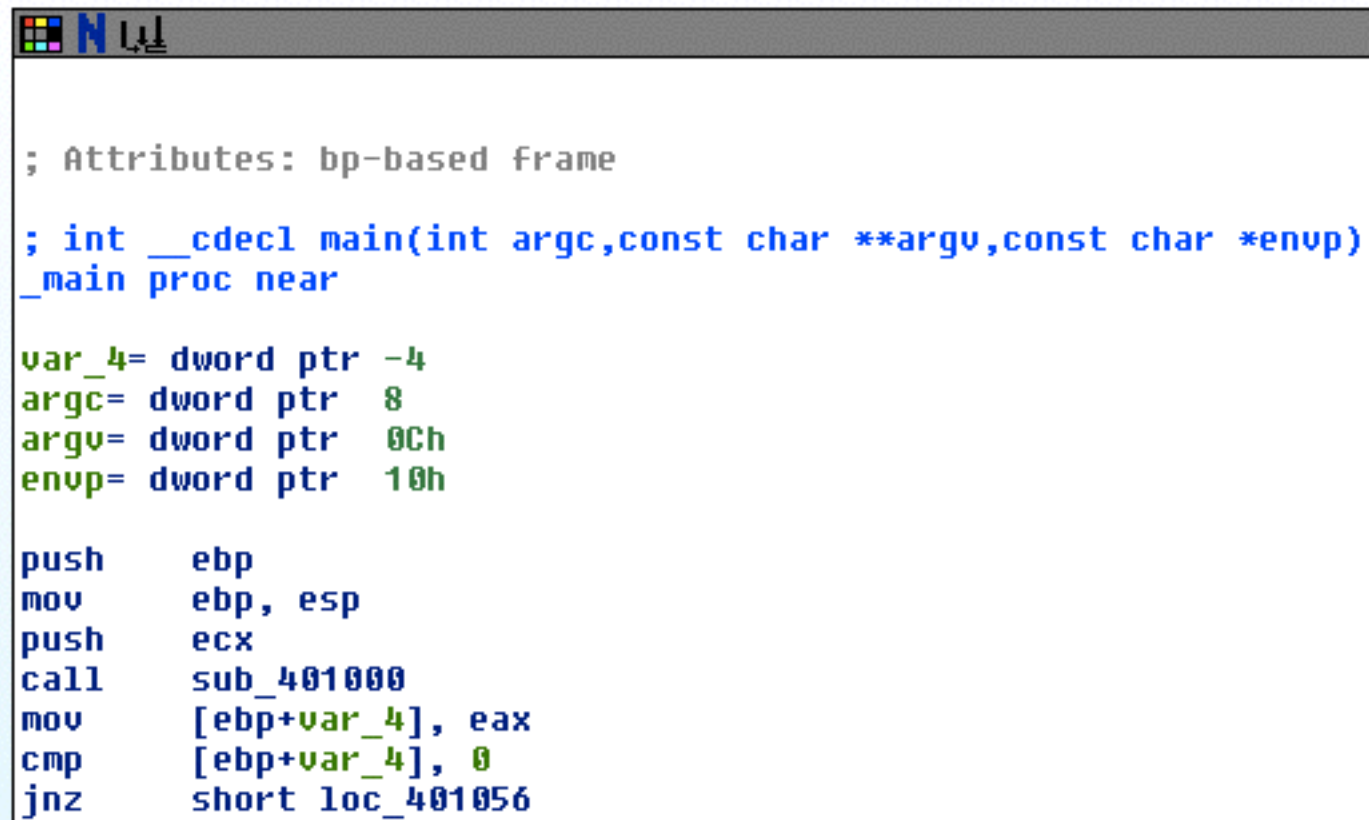
Graph and Text Mode

- Spacebar switches mode



```
IDA View-A
.text:00401040
.text:00401040 ; Attributes: bp-based frame
.text:00401040
.text:00401040 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:00401040 _nain proc near ; CODE XREF: start+AF1p
.text:00401040
.text:00401040 var_4 = dword ptr -4
.text:00401040 argc = dword ptr  8
.text:00401040 argv = dword ptr  0Ch
.text:00401040 envp = dword ptr  10h
.text:00401040
* .text:00401040 push    ebp
* .text:00401041 mov     ebp, esp
* .text:00401043 push    ecx
```

Default Graph Mode Display



The image shows a screenshot of a debugger window, likely Visual Studio, displaying assembly code for a C program. The window has a title bar with a color palette icon, the letter 'N', and some Arabic text. The code is as follows:

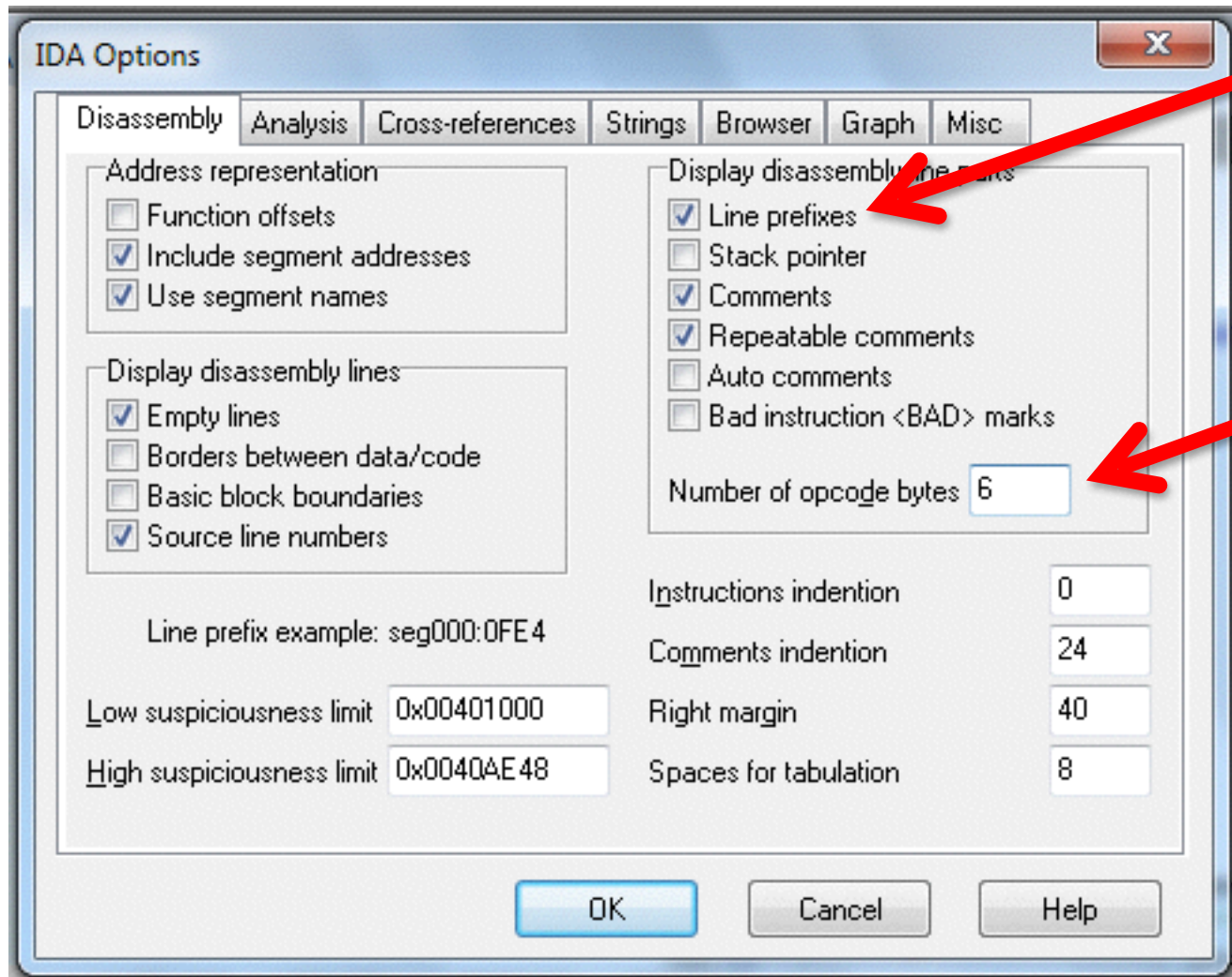
```
; Attributes: bp-based frame

; int __cdecl main(int argc,const char **argv,const char *envp)
_main proc near

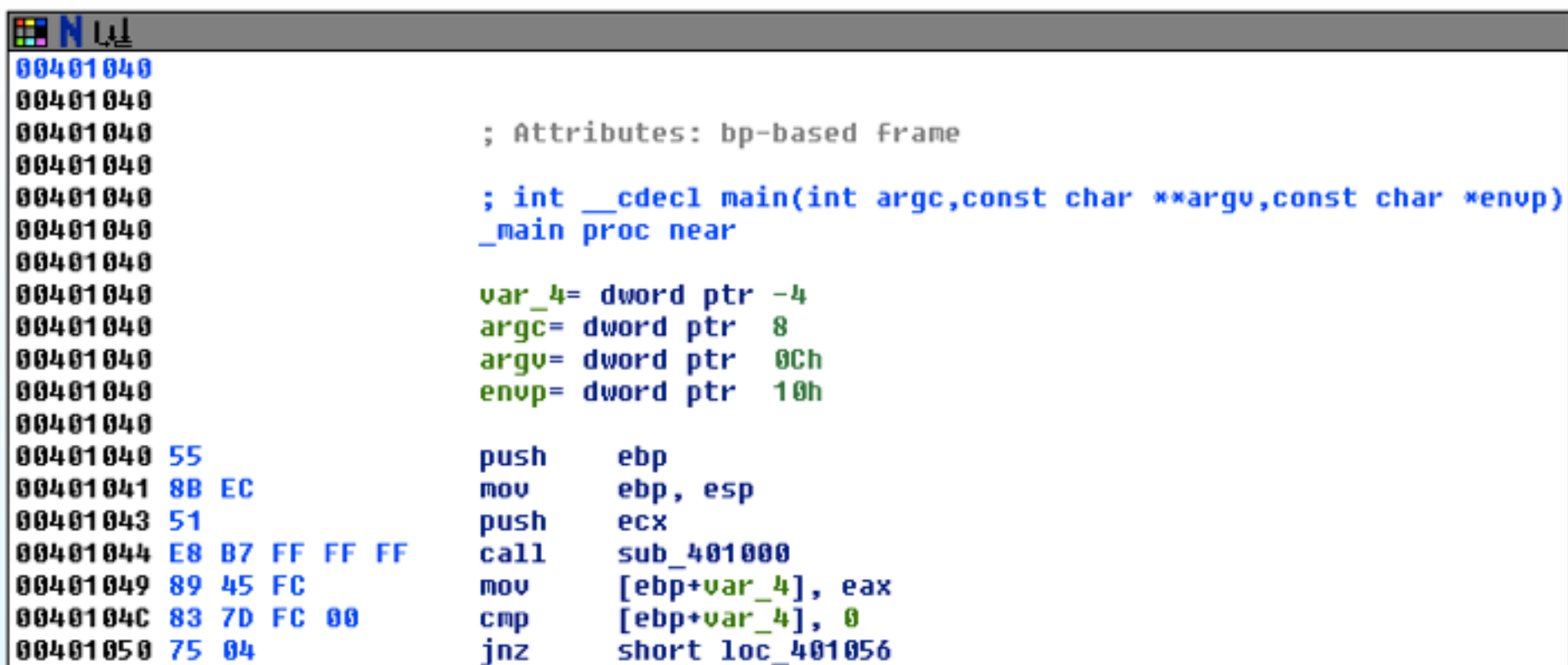
var_4= dword ptr -4
argc= dword ptr  8
argv= dword ptr  0Ch
envp= dword ptr  10h

push    ebp
mov     ebp, esp
push    ecx
call    sub_401000
mov     [ebp+var_4], eax
cmp     [ebp+var_4], 0
jnz     short loc_401056
```

Options, General



Better Graph Mode View

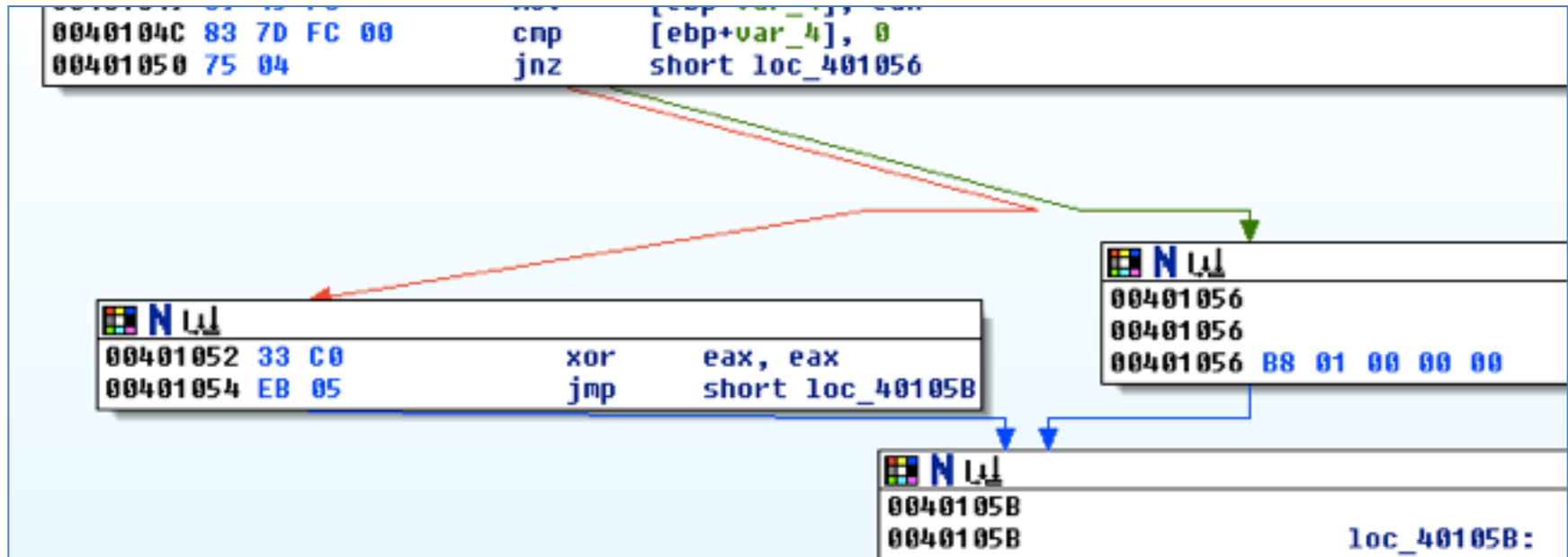


```
00401040
00401040
00401040      ; Attributes: bp-based frame
00401040
00401040      ; int __cdecl main(int argc,const char **argv,const char *envp)
00401040      _main proc near
00401040
00401040      var_4= dword ptr -4
00401040      argc= dword ptr 8
00401040      argv= dword ptr 0Ch
00401040      envp= dword ptr 10h
00401040
00401040 55          push     ebp
00401041 8B EC       mov      ebp, esp
00401043 51          push     ecx
00401044 E8 B7 FF FF call     sub_401000
00401049 89 45 FC       mov      [ebp+var_4], eax
0040104C 83 7D FC 00     cmp      [ebp+var_4], 0
00401050 75 04          jnz      short loc_401056
```

Arrows

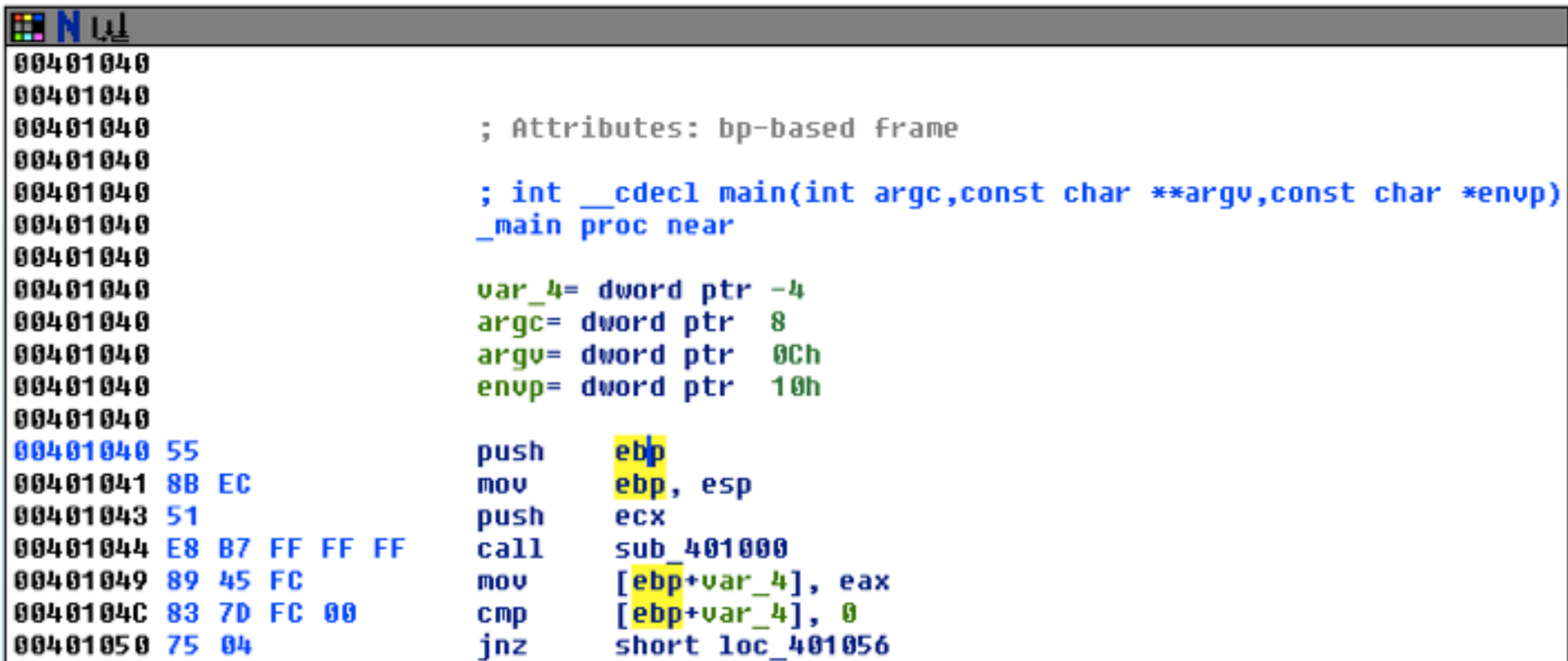
- Colors
 - Red Conditional jump not taken
 - Green Conditional jump taken
 - Blue Unconditional jump
- Direction
 - Up Loop

Arrow Color Example



Highlighting

- Highlighting text in graph mode highlights every instance of that text



The screenshot shows a debugger window with a list of memory addresses on the left and assembly instructions on the right. The text 'ebp' and '[ebp+var_4]' are highlighted in yellow across multiple lines of code.

```
00401040
00401040
00401040      ; Attributes: bp-based frame
00401040
00401040      ; int __cdecl main(int argc,const char **argv,const char *envp)
00401040      _main proc near
00401040
00401040      var_4= dword ptr -4
00401040      argc= dword ptr  8
00401040      argv= dword ptr  0Ch
00401040      envp= dword ptr  10h
00401040
00401040 55          push     ebp
00401041 8B EC      mov      ebp, esp
00401043 51          push     ecx
00401044 E8 B7 FF FF FF  call     sub_401000
00401049 89 45 FC      mov      [ebp+var_4], eax
0040104C 83 7D FC 00    cmp      [ebp+var_4], 0
00401050 75 04        jnz      short loc_401056
```

Text Mode

Arrows

Solid = Unconditional

Dashed = Conditional

Up = Loop

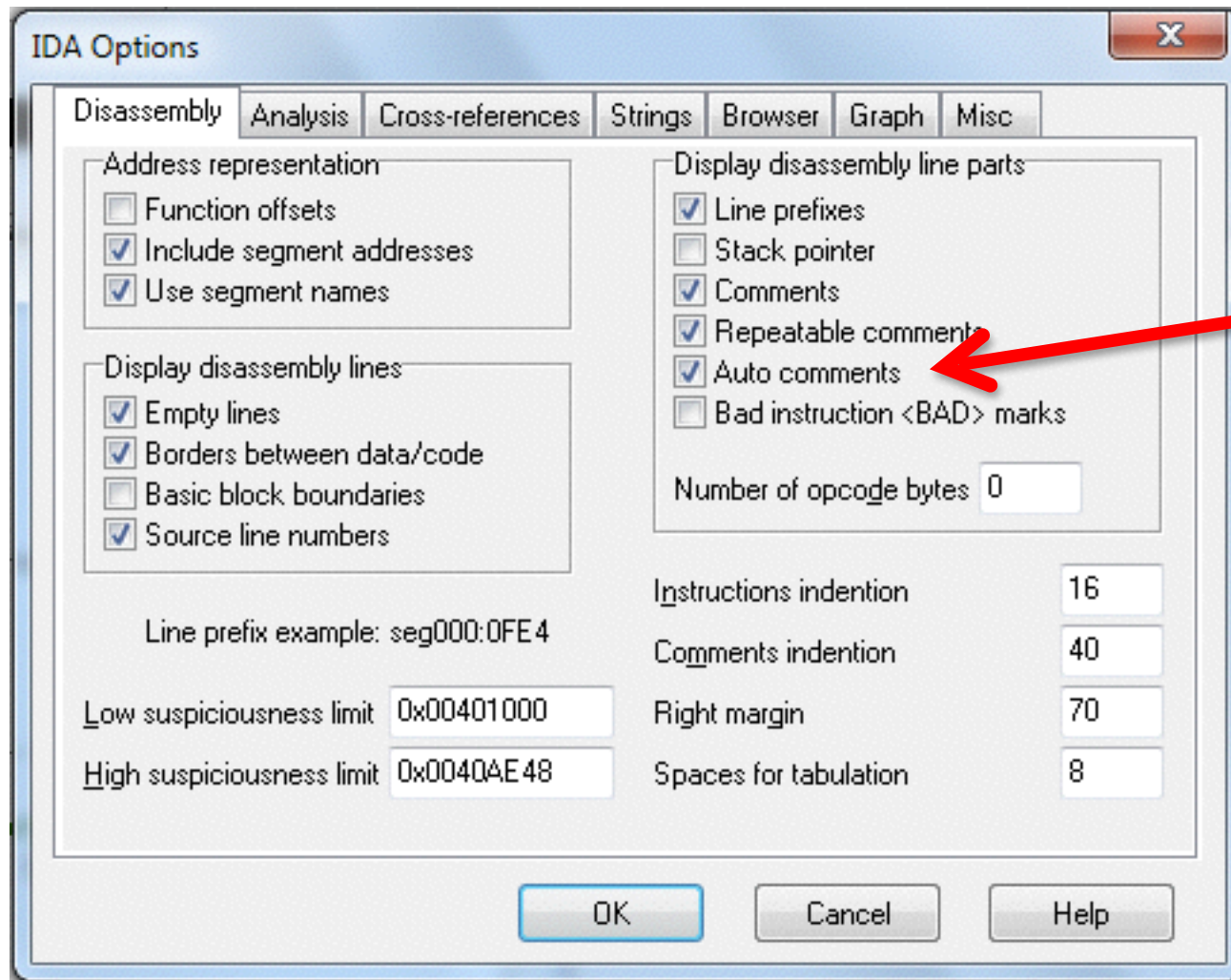
Section

Address

Comment
Generated by
IDA Pro

```
.text:00401015      jz      short loc_40102B
.text:00401017      push     offset aSuccessInterne ; "Success: Internet Connection\n"
.text:0040101C      call    sub_40105F
.text:00401021      add     esp, 4
.text:00401024      mov     eax, 1
.text:00401029      jmp     short loc_40103A
.text:0040102B      ; -----
.text:0040102B      loc_40102B: ; CODE XREF: sub_401000+151j
.text:0040102B      push     offset aError1_1NoInte ; "Error 1.1: No Internet\n"
.text:00401030      call    sub_40105F
.text:00401035      add     esp, 4
.text:00401038      xor     eax, eax
.text:0040103A      loc_40103A: ; CODE XREF: sub_401000+291j
.text:0040103A      mov     esp, ebp
.text:0040103C      pop     ebp
```

Options, General



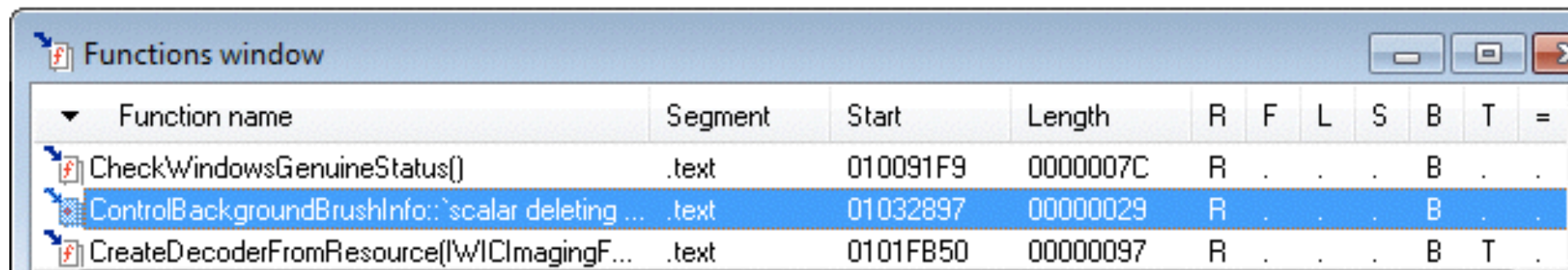
Adds Comments to Each Instruction

```
.text:00401015      jz      short loc_40102B ; Jump if Zero (ZF=1)
.text:00401017      push     offset aSuccessInterne ; "Success: Internet Connection\n"
.text:0040101C      call    sub_40105F          ; Call Procedure
.text:00401021      add     esp, 4              ; Add
.text:00401024      mov     eax, 1
.text:00401029      jmp     short loc_40103A ; Jump
.text:0040102B      ; -----
.text:0040102B      loc_40102B:                ; CODE XREF: sub_401000+15↑j
.text:0040102B      push     offset aError1_1NoInte ; "Error 1.1: No Internet\n"
.text:00401030      call    sub_40105F          ; Call Procedure
.text:00401035      add     esp, 4              ; Add
.text:00401038      |      xor     eax, eax      ; Logical Exclusive OR
.text:0040103A      loc_40103A:                ; CODE XREF: sub_401000+29↑j
.text:0040103A      mov     esp, ebp
.text:0040103C      pop     ebp
```

Useful Windows for Analysis

Functions

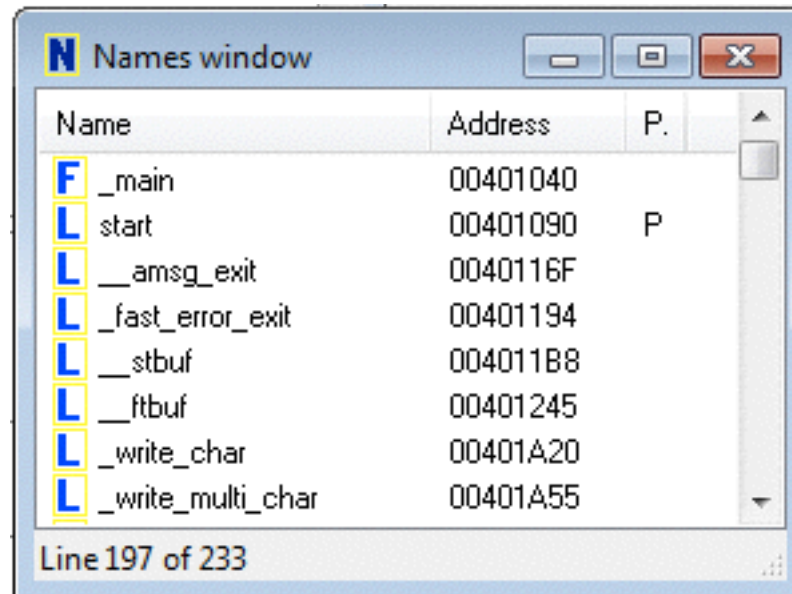
- Shows each function, length, and flags
 - L = Library functions
- Sortable
 - Large functions usually more important



Function name	Segment	Start	Length	R	F	L	S	B	T	=
CheckWindowsGenuineStatus()	.text	010091F9	0000007C	R	.	.	.	B	.	.
ControlBackgroundBrushInfo::`scalar deletingtext	01032897	00000029	R	.	.	.	B	.	.
CreateDecoderFromResource(IWICImagingF...	.text	0101FB50	00000097	R	.	.	.	B	T	.

Names Window

- Every address with a name
 - Functions, named code, named data, strings

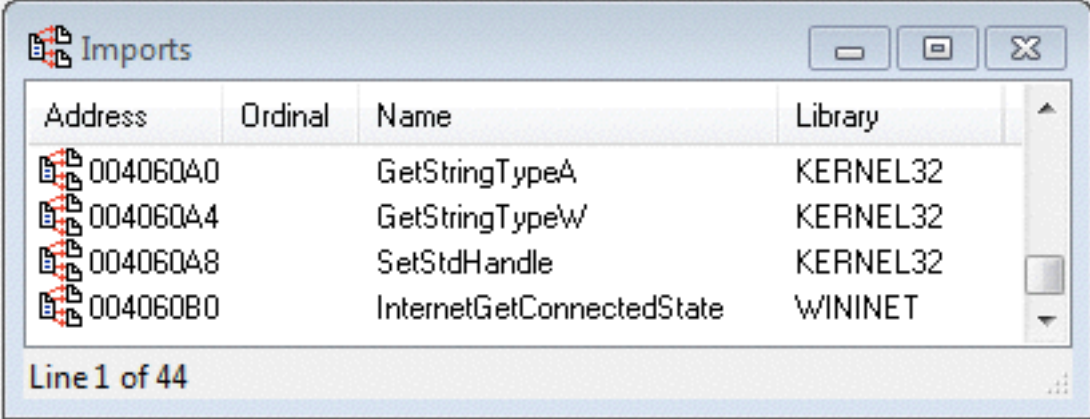


Strings

Strings window

Address	Length	Type	String
"..." .rdata:0...	0000000F	C	GetStringTypeW
"..." .rdata:0...	0000000D	C	SetStdHandle
"..." .rdata:0...	0000000C	C	CloseHandle
"..." .rdata:0...	0000000D	C	KERNEL32.dll
"..." .data:00...	00000018	C	Error 1.1: No Internet\n
"..." .data:00...	0000001E	C	Success: Internet Connection\n

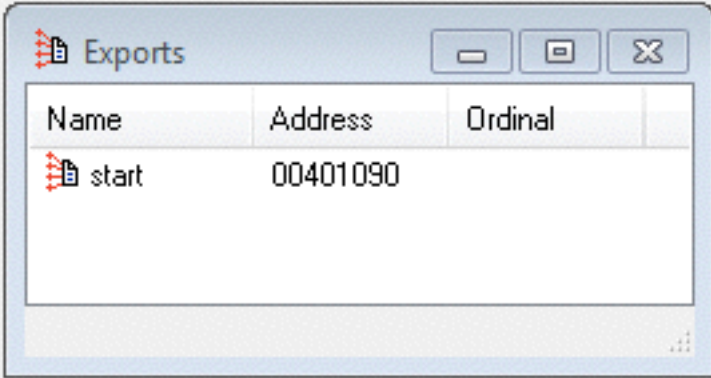
Imports & Exports



The Imports window displays a table of imported functions. Each row includes a red cross icon, a memory address, an ordinal, a function name, and the library name. The status bar at the bottom indicates 'Line 1 of 44'.

Address	Ordinal	Name	Library
004060A0		GetStringTypeA	KERNEL32
004060A4		GetStringTypeW	KERNEL32
004060A8		SetStdHandle	KERNEL32
004060B0		InternetGetConnectedState	WININET

Line 1 of 44

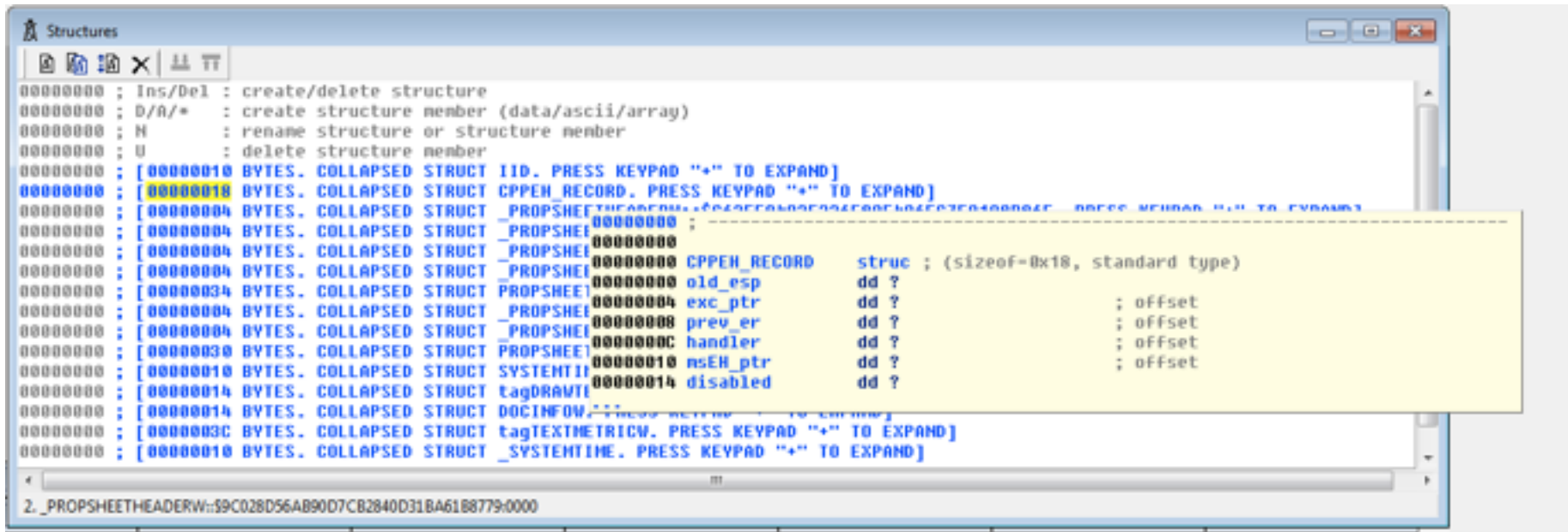


The Exports window displays a table of exported functions. It includes a red cross icon, the function name, the address, and the ordinal. The status bar at the bottom right shows a small icon.

Name	Address	Ordinal
start	00401090	

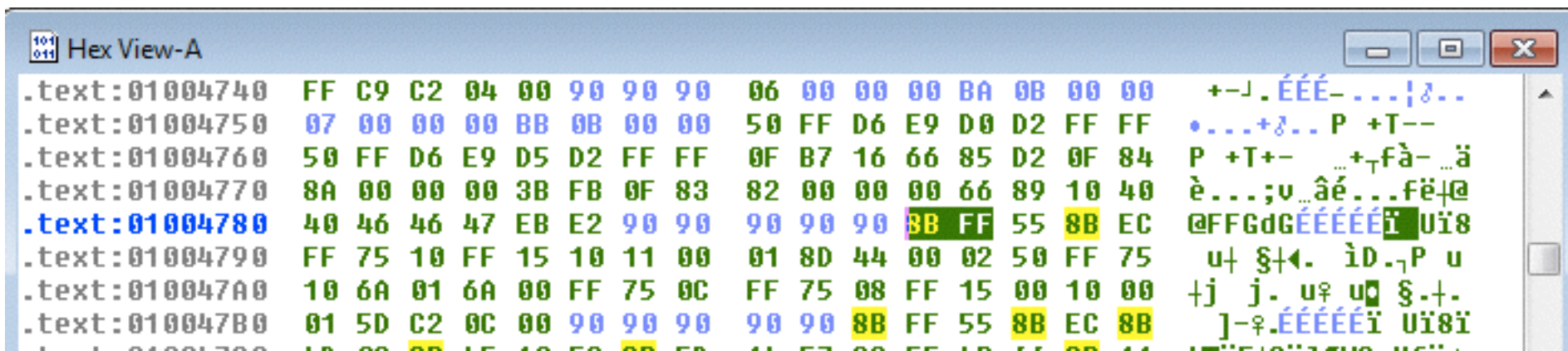
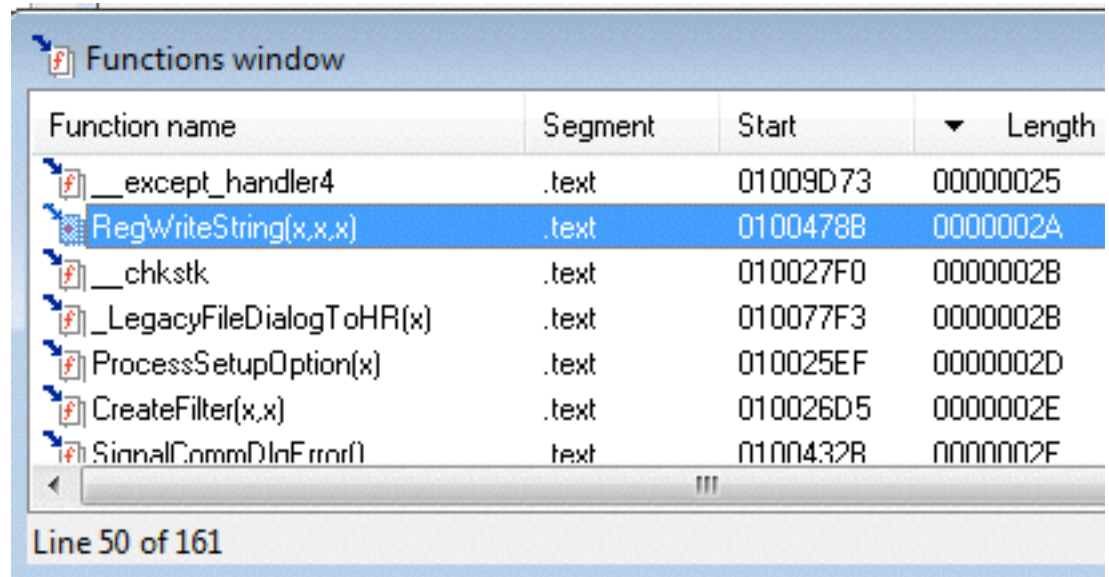
Structures

- All active data structures
 - Hover to see yellow pop-up window



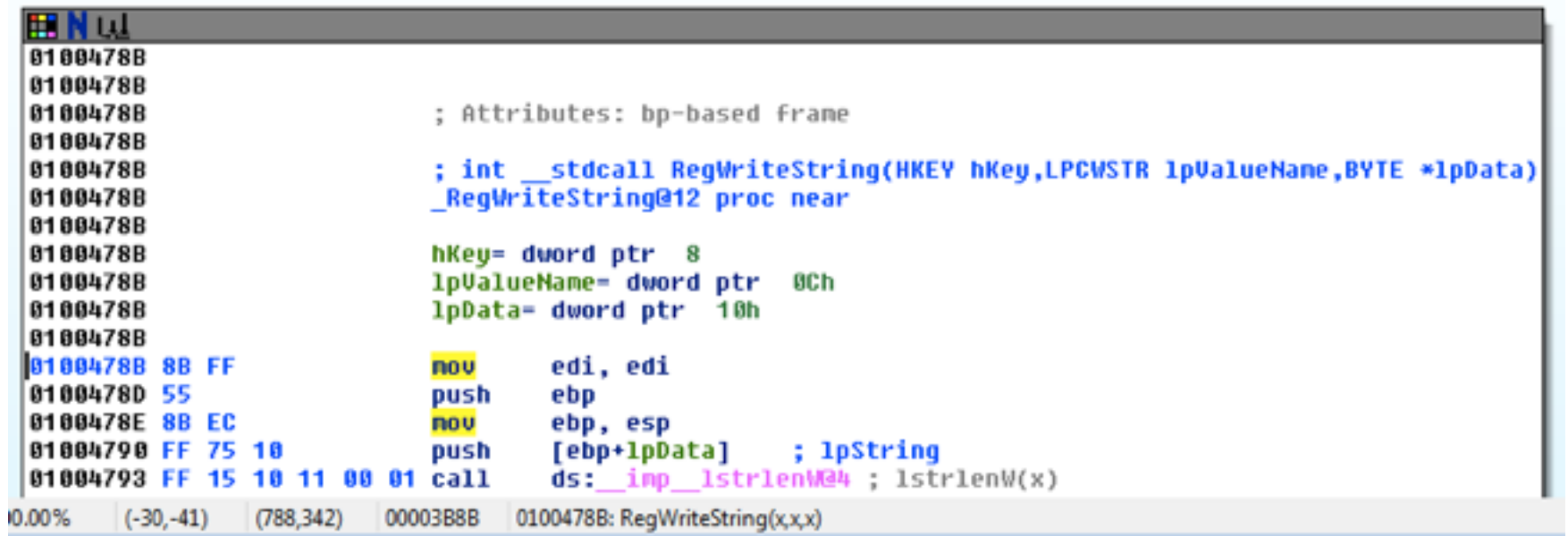
Cross-Reference

- Double-click function
- Jump to code in other views



Function Call

- Parameters pushed onto stack
- CALL to start function



The screenshot shows a debugger window with the following assembly code and comments:

```
0100478B ; Attributes: bp-based frame
0100478B ; int __stdcall RegWriteString(HKEY hKey,LPCWSTR lpValueName, BYTE *lpData)
0100478B _RegWriteString@12 proc near
0100478B hKey= dword ptr 8
0100478B lpValueName= dword ptr 0Ch
0100478B lpData= dword ptr 10h
0100478B
0100478B 8B FF      mov     edi, edi
0100478D 55        push    ebp
0100478E 8B EC      mov     ebp, esp
01004790 FF 75 10   push    [ebp+lpData] ; lpString
01004793 FF 15 10 11 00 01 call    ds:__imp__lstrlenW@4 ; lstrlenW(x)
```

The status bar at the bottom shows: 0.00% (-30,-41) (788,342) 00003B8B 0100478B: RegWriteString(x,x,x)

Returning to the Default View

- Windows, Reset Desktop
- Windows, Save Desktop
 - To save a new view