Practical Malware Analysis

Ch 5: IDA Pro

About

IDA - The Interactive Disassembler

Freeware Version 5.0

(c) 2010 Hex-Rays SA

Welcome to the freeware edition of IDA Pro 5.0.
This version is fully functional but does not offer all the bells and whistles of the commercial versions of IDA Pro.

Try the commercial version of IDA Pro today!

http://www.hex-rays.com

□ Do not display IDA 6.x info next time

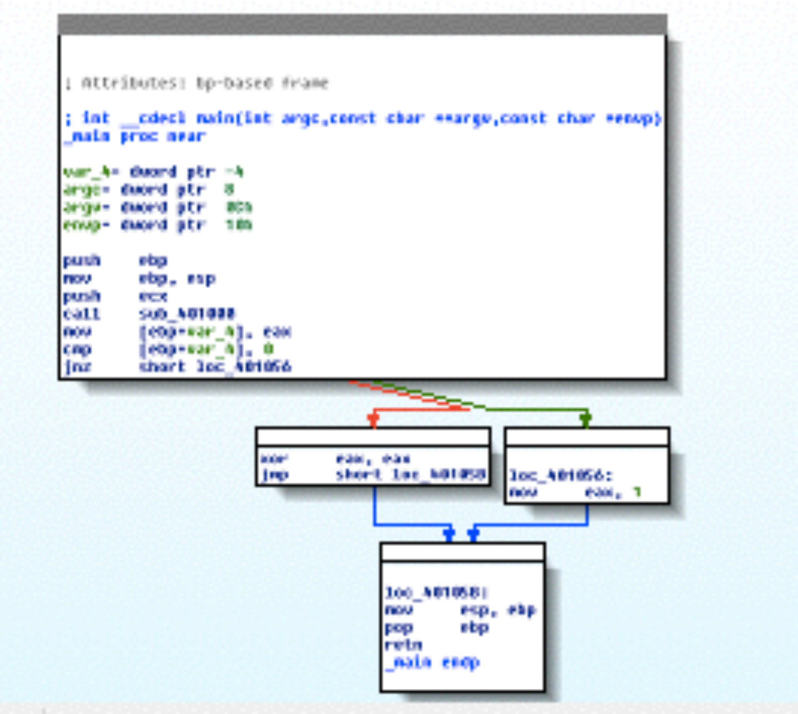
Last modified
2-6-16

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

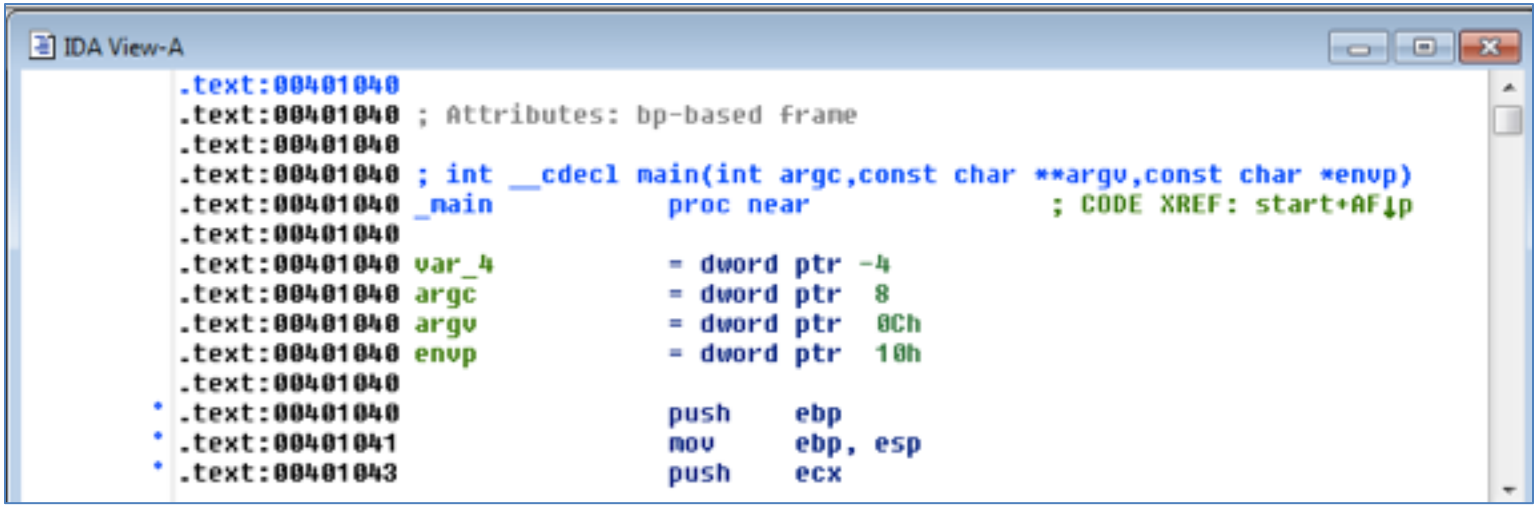


```
; Attributes: bp-based frame

; 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
```

```
xor     eax, eax
jmp     short loc_401058
```

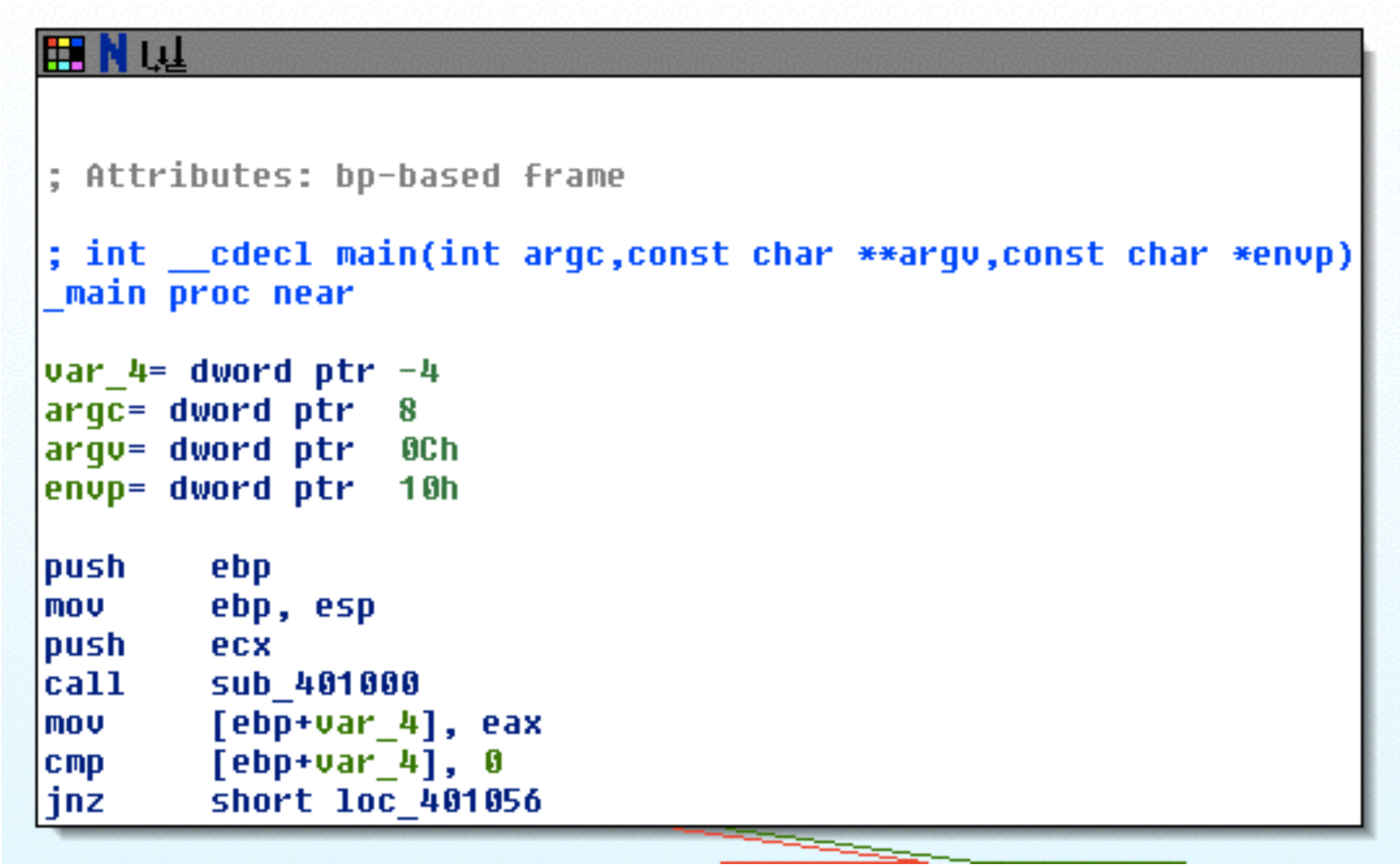
```
loc_401056:
mov     eax, 1
```

```
loc_401058:
mov     esp, ebp
pop     ebp
retn
_main endp
```



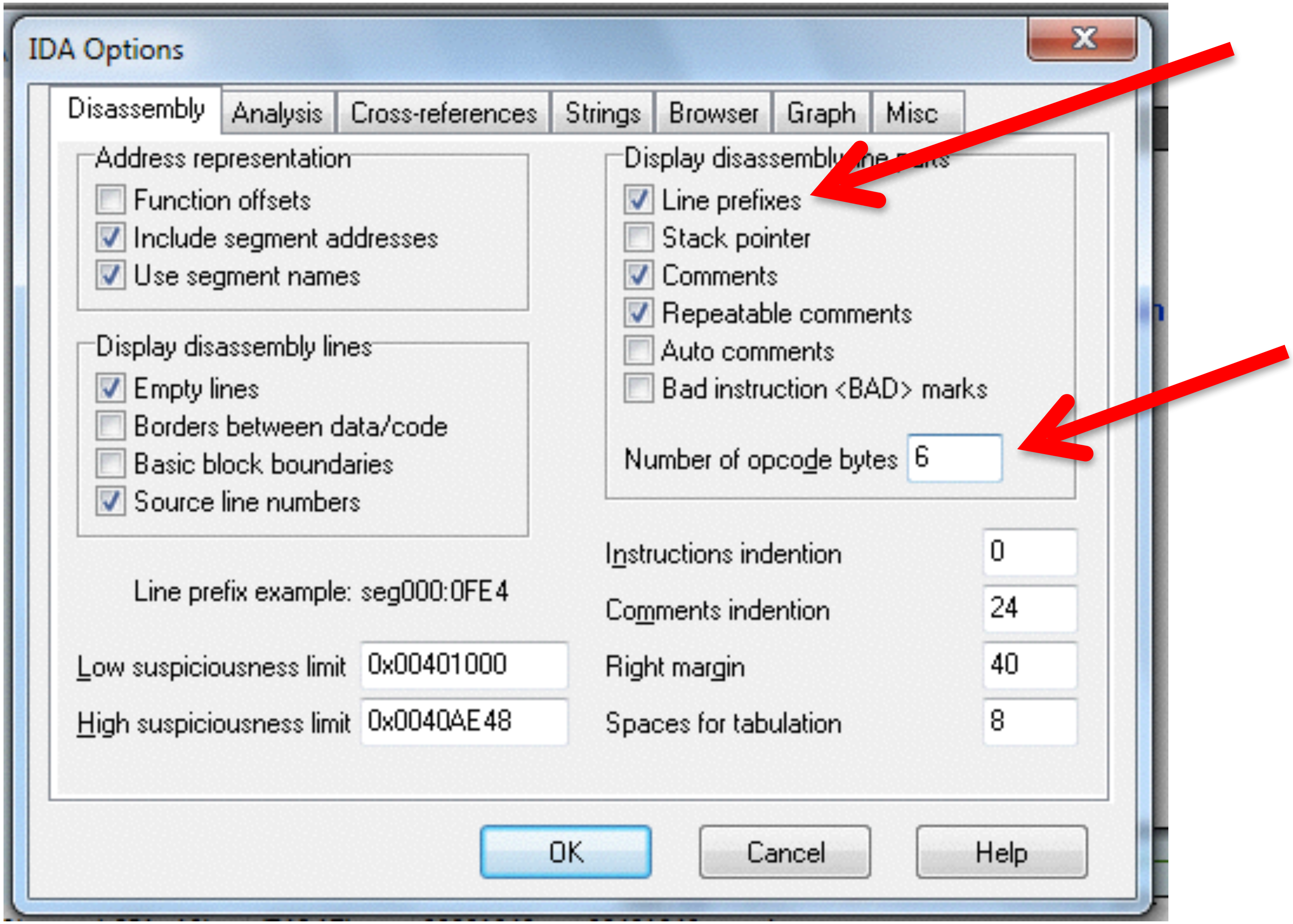
```
.text:00401040
.text:00401040 ; Attributes: bp-based frame
.text:00401040
.text:00401040 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:00401040 _main           proc near               ; CODE XREF: start+AF↓p
.text:00401040
.text:00401040 var_4           = dword ptr -4
.text:00401040 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



```
; 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



Address representation

- □ Function offsets
- ☑ Include segment addresses
- ☑ Use segment names

Display disassembly lines

- ☑ Empty lines
- □ Borders between data/code
- □ Basic block boundaries
- ☑ Source line numbers

Line prefix example: seg000:0FE4

Low suspiciousness limit: 0x00401000

High suspiciousness limit: 0x0040AE48

Display disassembly line parts

- ☑ Line prefixes
- □ Stack pointer
- ☑ Comments
- ☑ Repeatable comments
- □ Auto comments
- □ Bad instruction <BAD> marks

Number of opcode bytes: 6

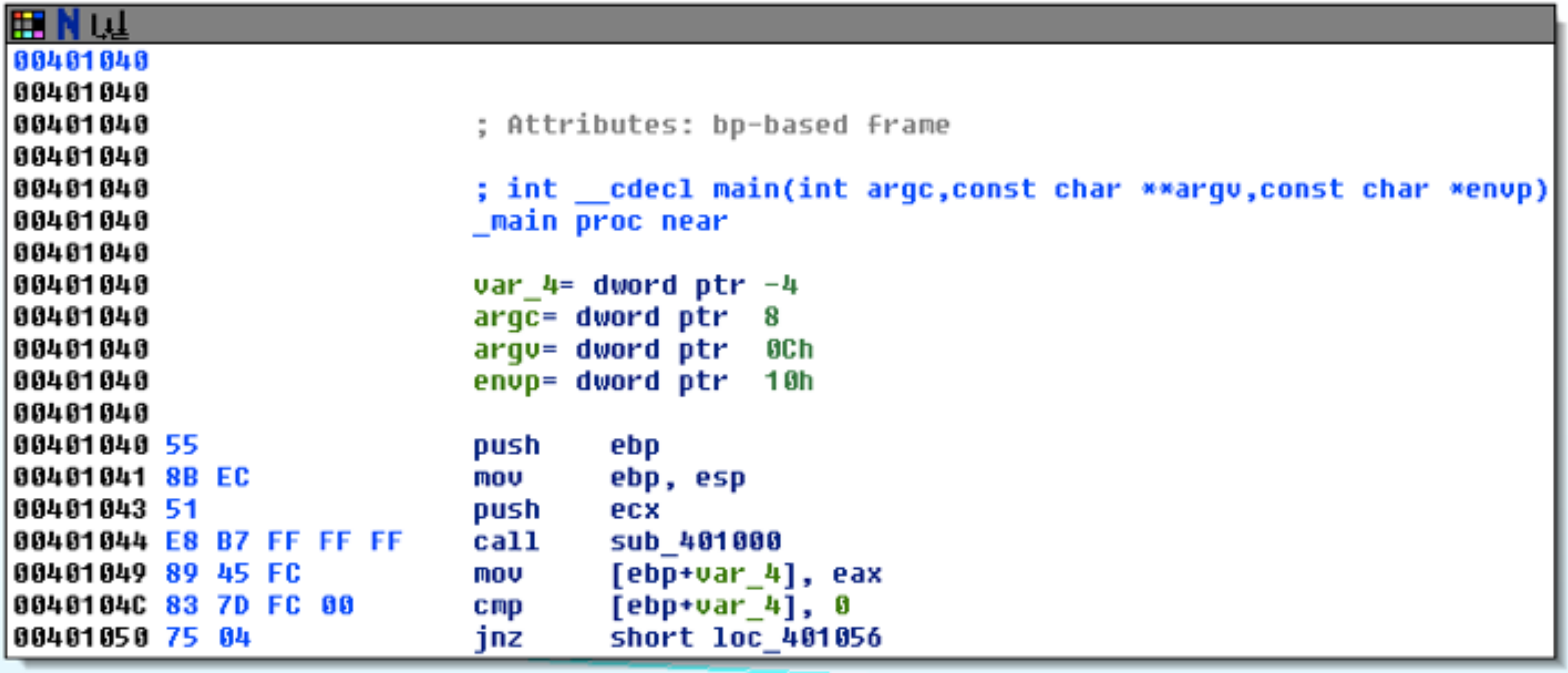
Instructions indention: 0

Comments indention: 24

Right margin: 40

Spaces for tabulation: 8

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 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

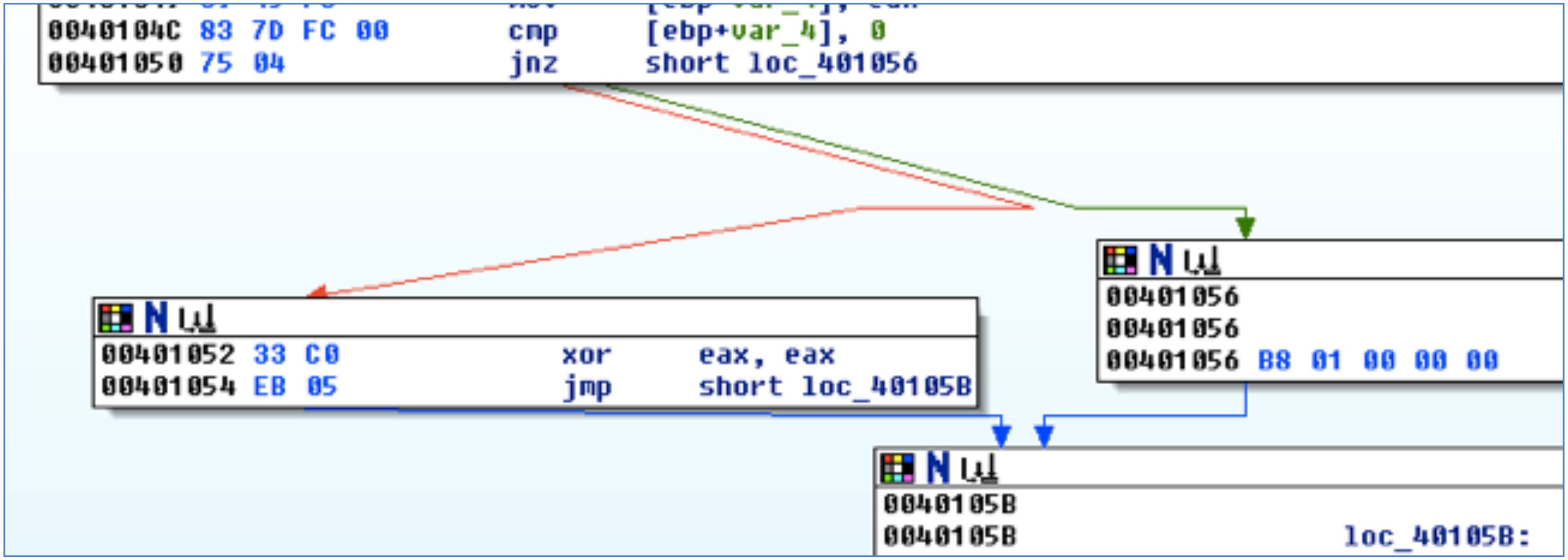
- Colors

– Red	Conditional jump not taken
– Green	Conditional jump taken
– Blue	Unconditional jump

- Direction

– Up	Loop

Arrow Color Example



```
0040104C 83 7D FC 00         cmp     [ebp+var_4], 0
00401050 75 04               jnz     short loc_401056
```

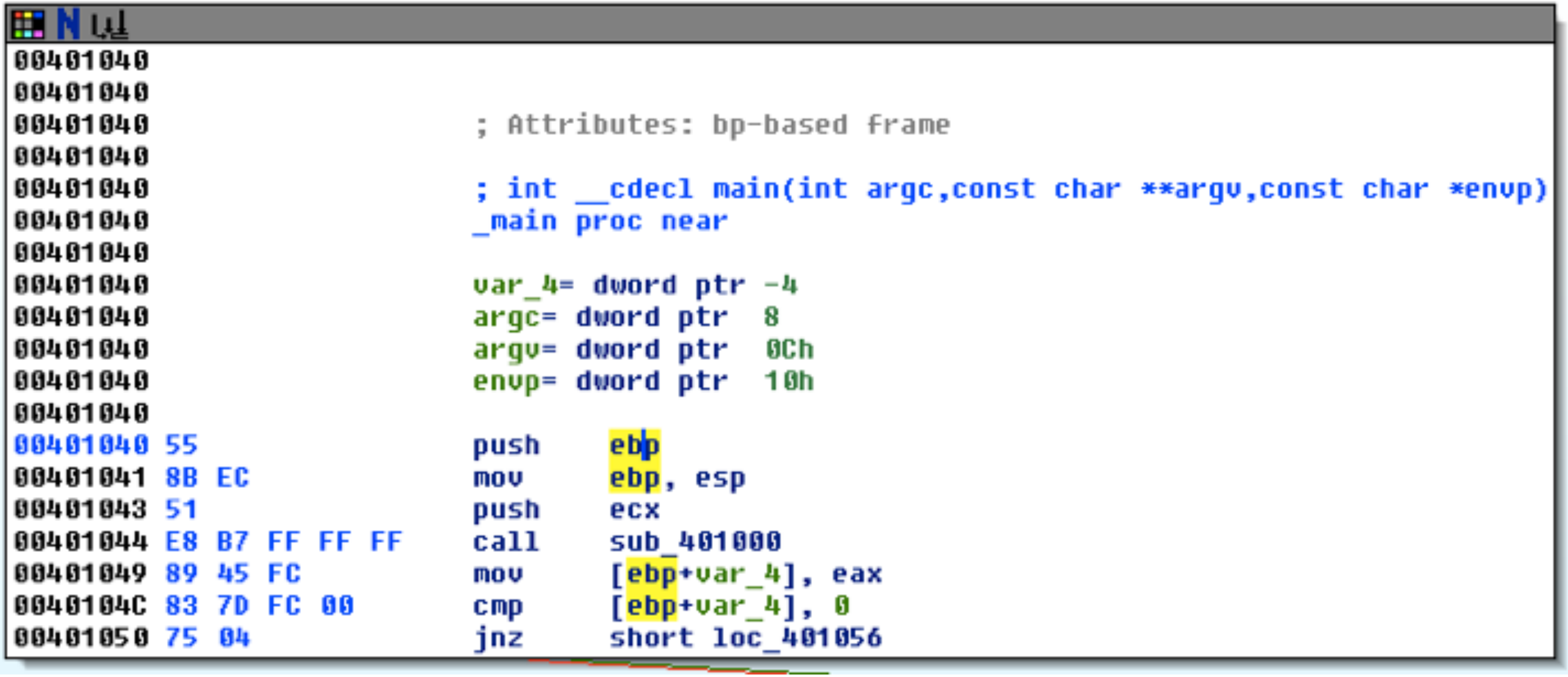
```
00401052 33 C0               xor     eax, eax
00401054 EB 05               jmp     short loc_40105B
```

```
00401056
00401056
00401056 B8 01 00 00 00
```

```
0040105B
0040105B                     loc_40105B:
```

Highlighting

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



```
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
```

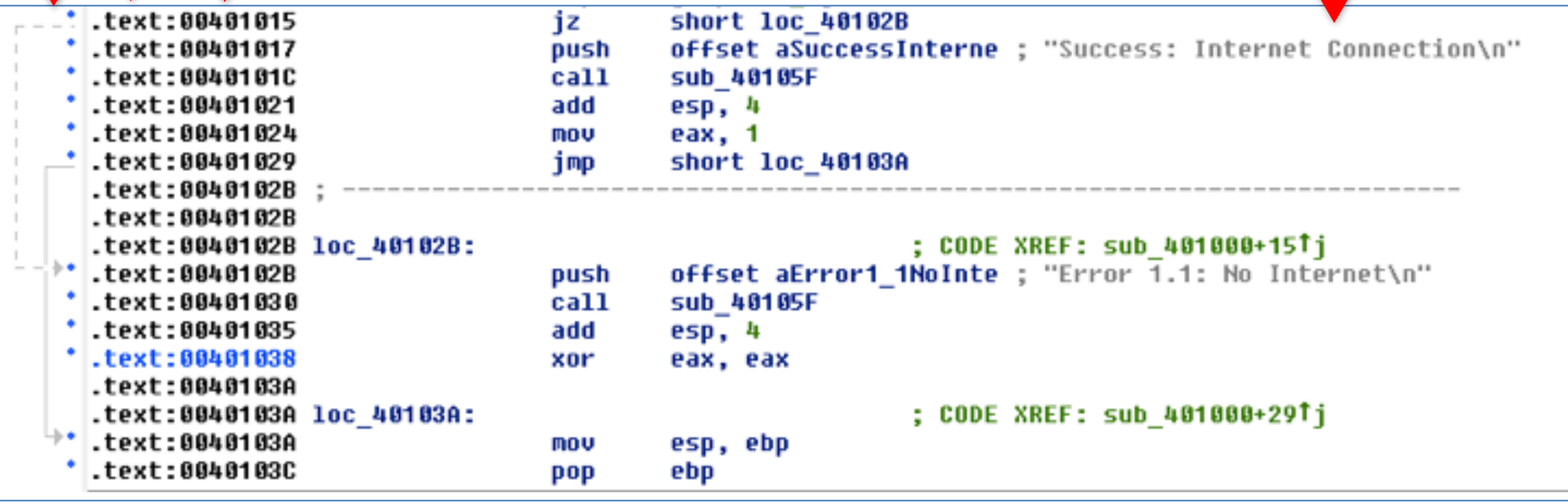
Text Mode

Arrows
Solid = Unconditional
Dashed = Conditional
Up = Loop

Section

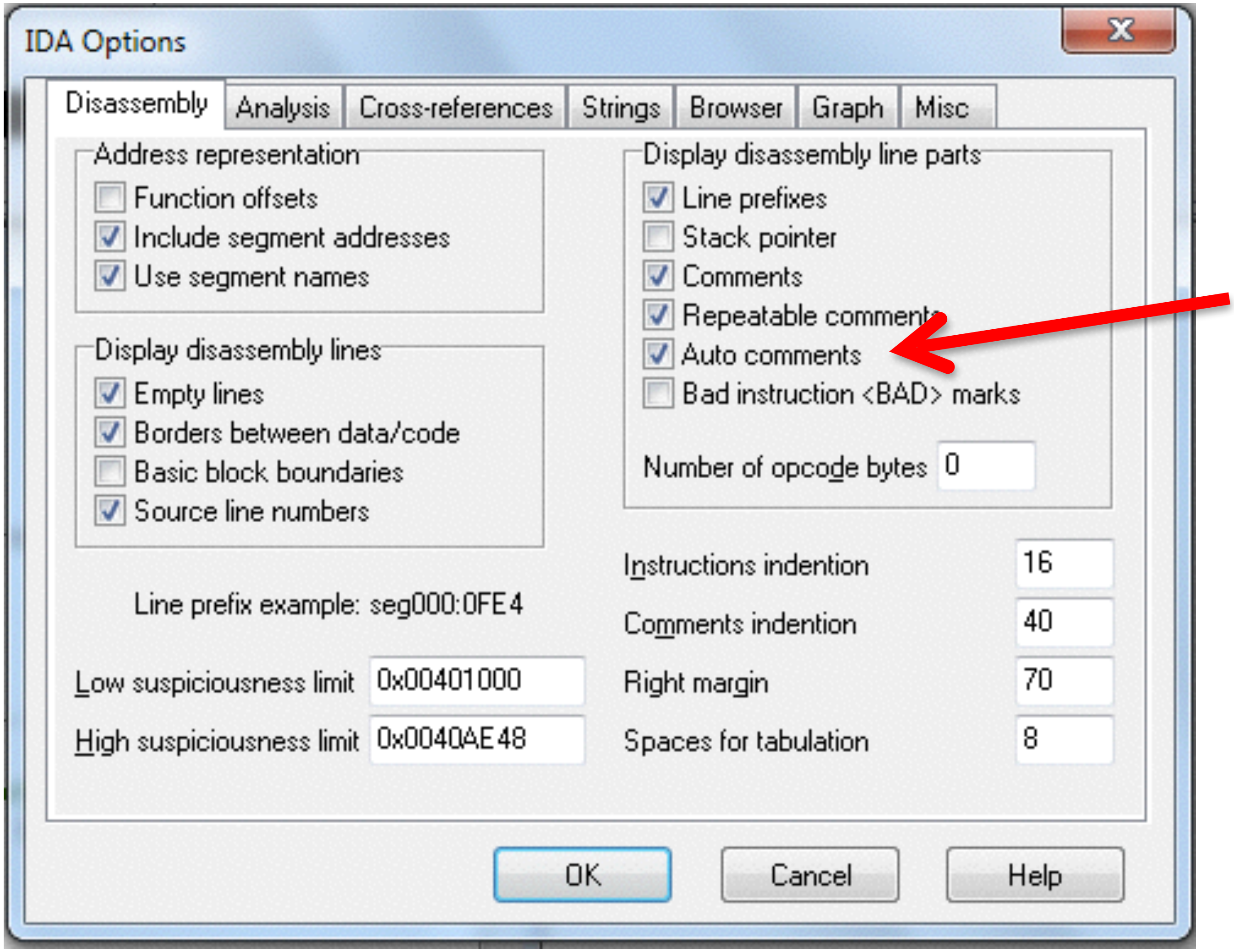
Address

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
.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
.text:00401035                 add     esp, 4
.text:00401038                 xor     eax, eax
.text:0040103A
.text:0040103A loc_40103A:                             ; CODE XREF: sub_401000+29↑j
.text:0040103A                 mov     esp, ebp
.text:0040103C                 pop     ebp
```

Options, General



IDA Options

Address representation

- ☐ Function offsets
- ☑ Include segment addresses
- ☑ Use segment names

Display disassembly lines

- ☑ Empty lines
- ☑ Borders between data/code
- ☐ Basic block boundaries
- ☑ Source line numbers

Line prefix example: seg000:0FE4

Low suspiciousness limit: 0x00401000

High suspiciousness limit: 0x0040AE48

Display disassembly line parts

- ☑ Line prefixes
- ☐ Stack pointer
- ☑ Comments
- ☑ Repeatable comments
- ☑ Auto comments
- ☐ Bad instruction <BAD> marks

Number of opcode bytes: 0

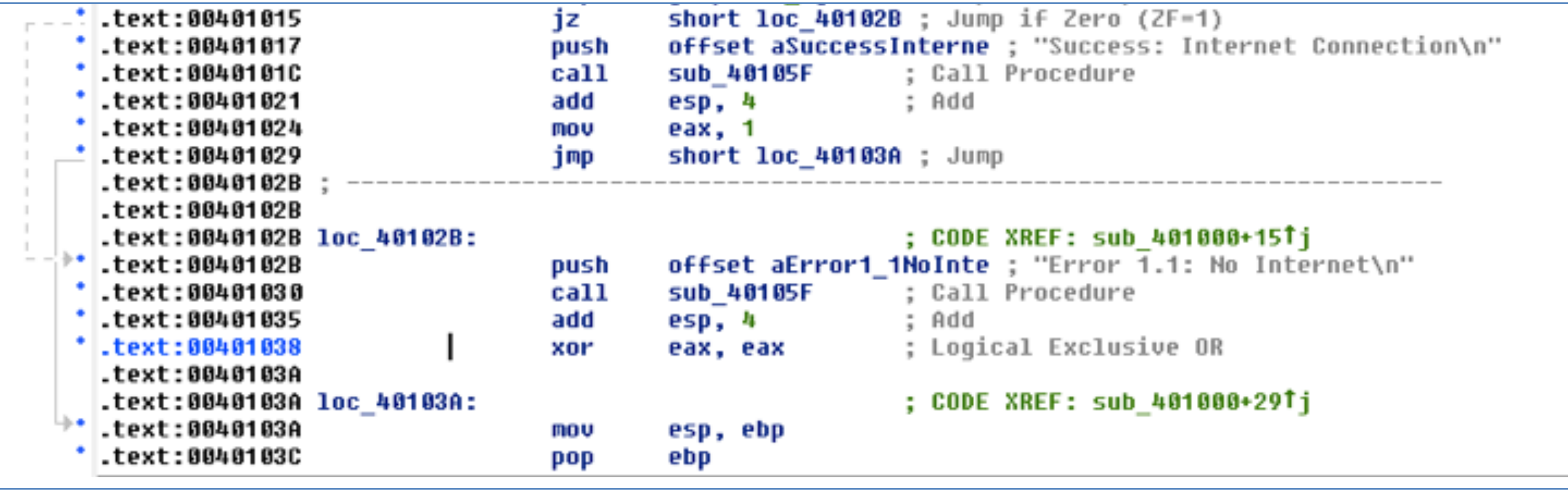
Instructions indention: 16

Comments indention: 40

Right margin: 70

Spaces for tabulation: 8

Adds Comments to Each Instruction

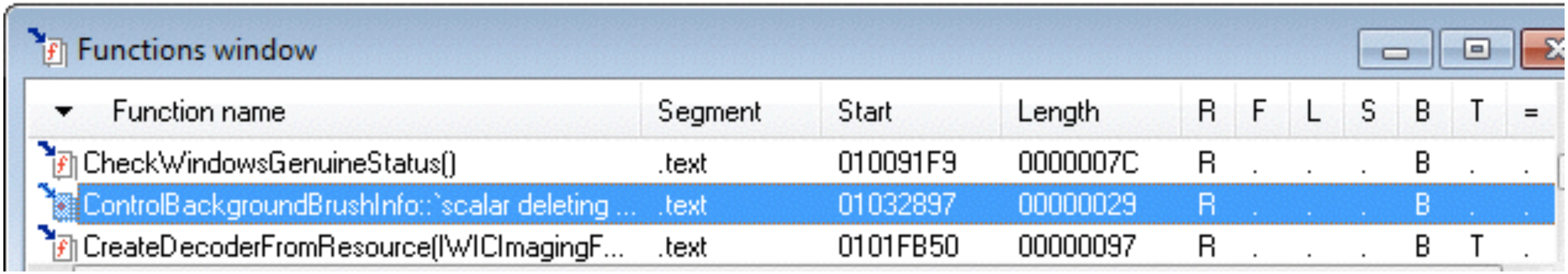


```
.text:00401015                 jz      short loc_40102B ; Jump if Zero (ZF=1)
.text:00401017                 push    offset aSuccessInterne ; "Success: Internet Connection\n"
.text:0040101C                 call    sub_40105F      ; Call Procedure
.text:00401021                 add     esp, 4          ; Add
.text:00401024                 mov     eax, 1
.text:00401029                 jmp     short loc_40103A ; Jump
.text:0040102B ; ---------------------------------------------------------------------------
.text:0040102B
.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
.text:0040103A loc_40103A:                             ; CODE XREF: sub_401000+29↑j
.text:0040103A                 mov     esp, ebp
.text:0040103C                 pop     ebp
```

Useful Windows for Analysis

Functions

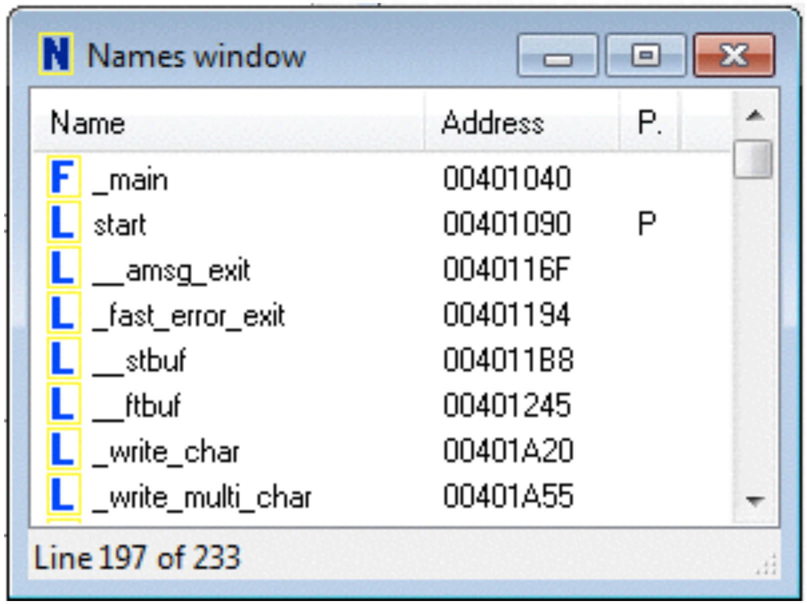
- Shows each function, length, and flags
  - L = Library functions
- Sortable
  - Large functions usually more important



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

Names Window

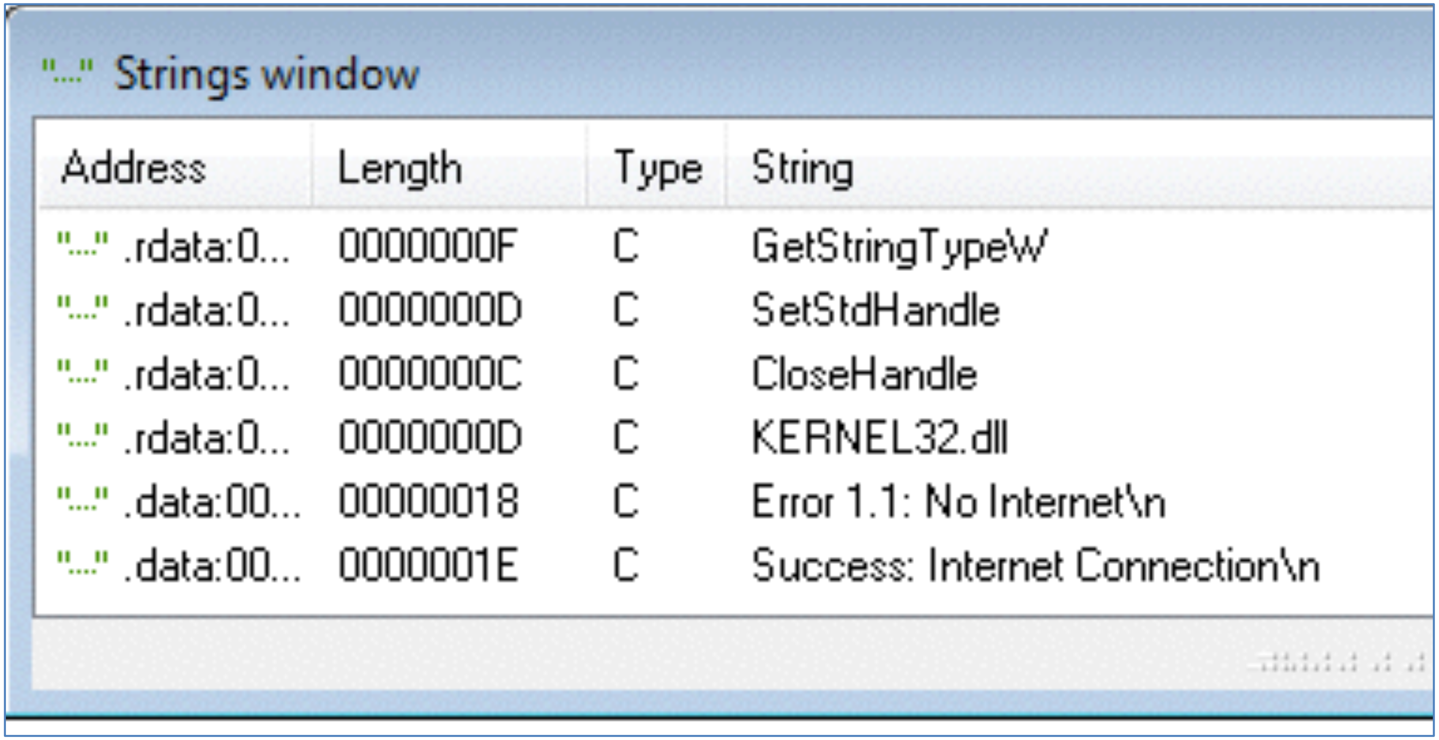
- Every address with a name
  - Functions, named code, named data, strings



Name	Address	P.
F _main	00401040	
L start	00401090	P
L __amsg_exit	0040116F	
L _fast_error_exit	00401194	
L __stbuf	004011B8	
L __ftbuf	00401245	
L _write_char	00401A20	
L _write_multi_char	00401A55	

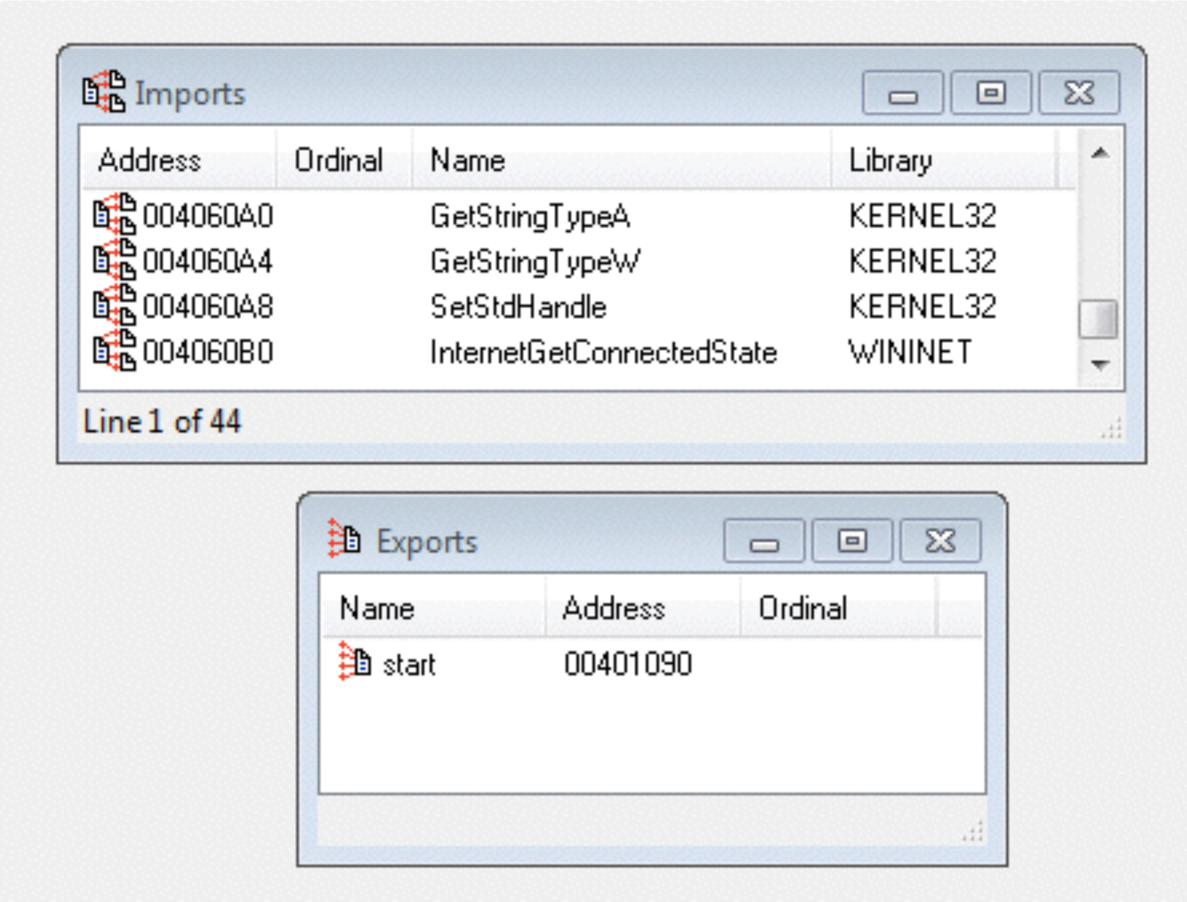
Line 197 of 233

Strings



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



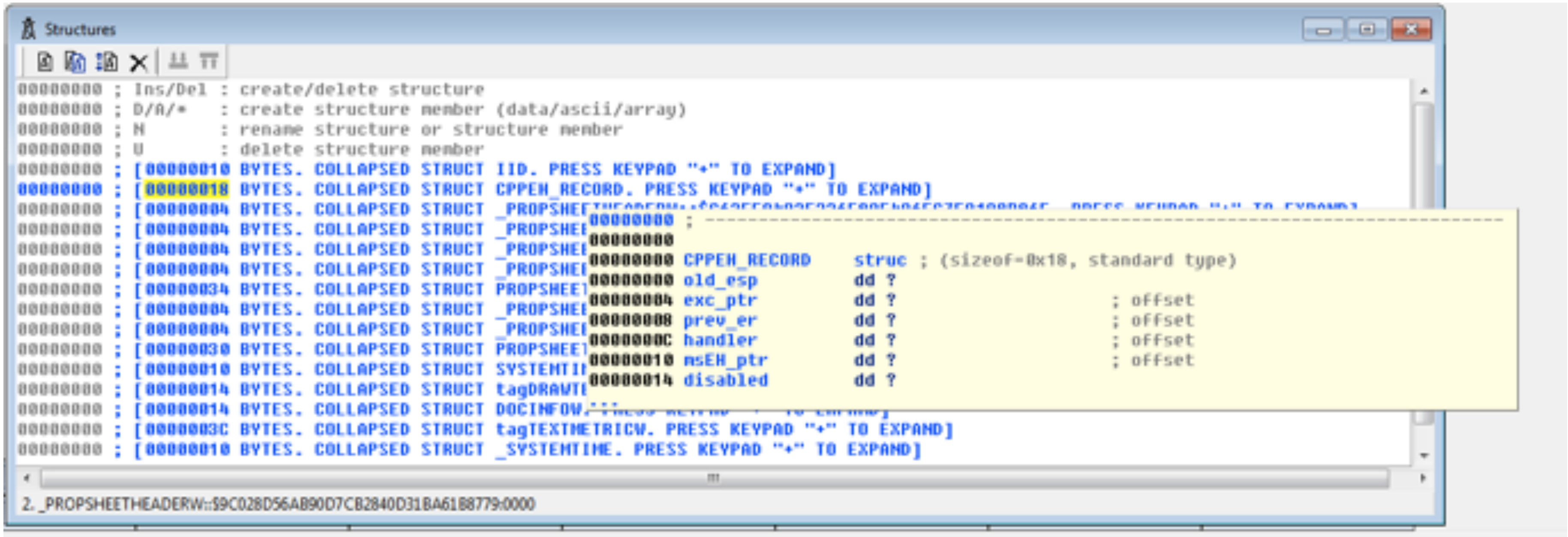
Address	Ordinal	Name	Library
004060A0		GetStringTypeA	KERNEL32
004060A4		GetStringTypeW	KERNEL32
004060A8		SetStdHandle	KERNEL32
004060B0		InternetGetConnectedState	WININET

Line 1 of 44

Name	Address	Ordinal
start	00401090	

Structures

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



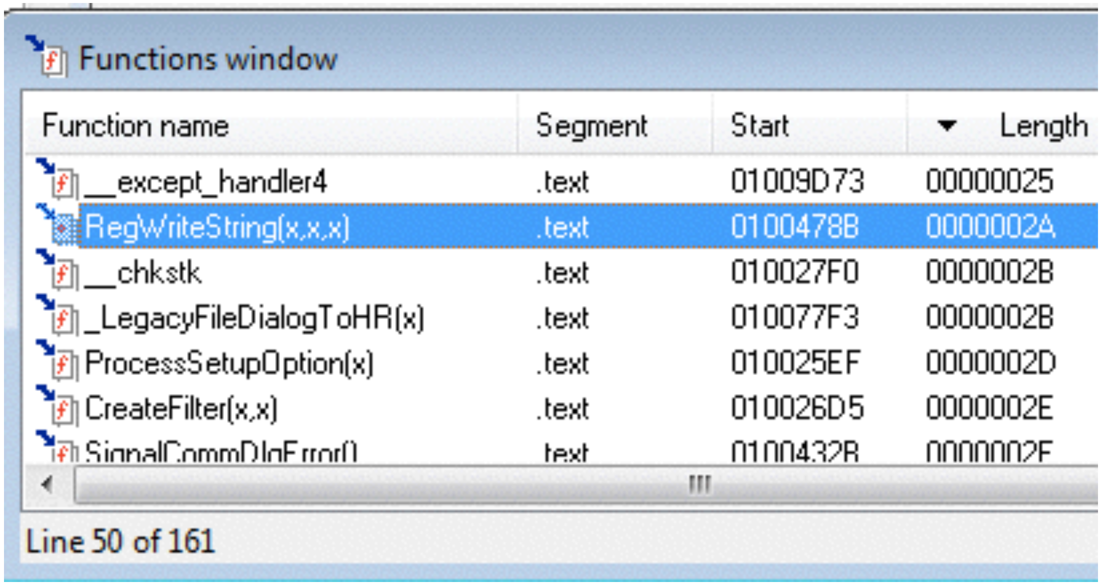
```
00000000 ; Ins/Del  : create/delete structure
00000000 ; D/A/*    : create structure member (data/ascii/array)
00000000 ; N        : rename structure or structure member
00000000 ; U        : delete structure member
00000000 ; [00000010 BYTES. COLLAPSED STRUCT IID. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000018 BYTES. COLLAPSED STRUCT CPPEH_RECORD. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEE
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEE
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEE
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEE
00000000 ; [00000034 BYTES. COLLAPSED STRUCT PROPSHEET
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEE
00000000 ; [00000004 BYTES. COLLAPSED STRUCT _PROPSHEE
00000000 ; [00000030 BYTES. COLLAPSED STRUCT PROPSHEET
00000000 ; [00000010 BYTES. COLLAPSED STRUCT SYSTEMTI
00000000 ; [00000014 BYTES. COLLAPSED STRUCT tagDRAWTE
00000000 ; [00000014 BYTES. COLLAPSED STRUCT DOCINFOW.
00000000 ; [0000003C BYTES. COLLAPSED STRUCT tagTEXTMETRICW. PRESS KEYPAD "+" TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT _SYSTEMTIME. PRESS KEYPAD "+" TO EXPAND]
```

```
00000000 ; ---------------------------------------------------------------
00000000
00000000 CPPEH_RECORD    struc ; (sizeof=0x18, standard type)
00000000 old_esp         dd ?
00000004 exc_ptr         dd ?                        ; offset
00000008 prev_er         dd ?                        ; offset
0000000C handler         dd ?                        ; offset
00000010 msEH_ptr        dd ?                        ; offset
00000014 disabled        dd ?
```

2. _PROPSHEETHEADERW::$9C028D56AB90D7CB2840D31BA61B8779:0000

Cross-Reference

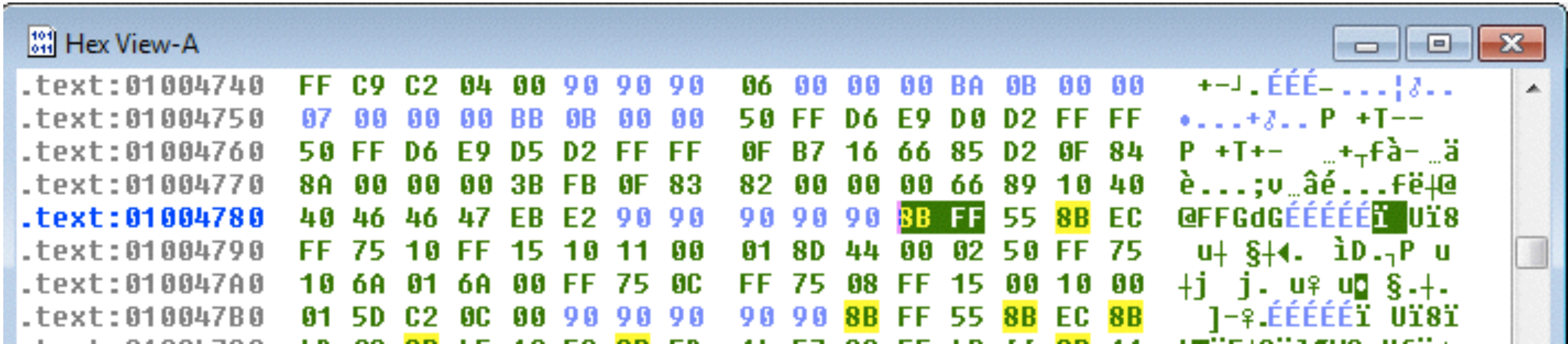
- Double-click function
- Jump to code in other views



Functions window

Function name	Segment	Start	Length
__except_handler4	.text	01009D73	00000025
RegWriteString(x,x,x)	.text	0100478B	0000002A
__chkstk	.text	010027F0	0000002B
_LegacyFileDialogToHR(x)	.text	010077F3	0000002B
ProcessSetupOption(x)	.text	010025EF	0000002D
CreateFilter(x,x)	.text	010026D5	0000002E
SignalCommDlgError()	.text	0100432B	0000002F

Line 50 of 161

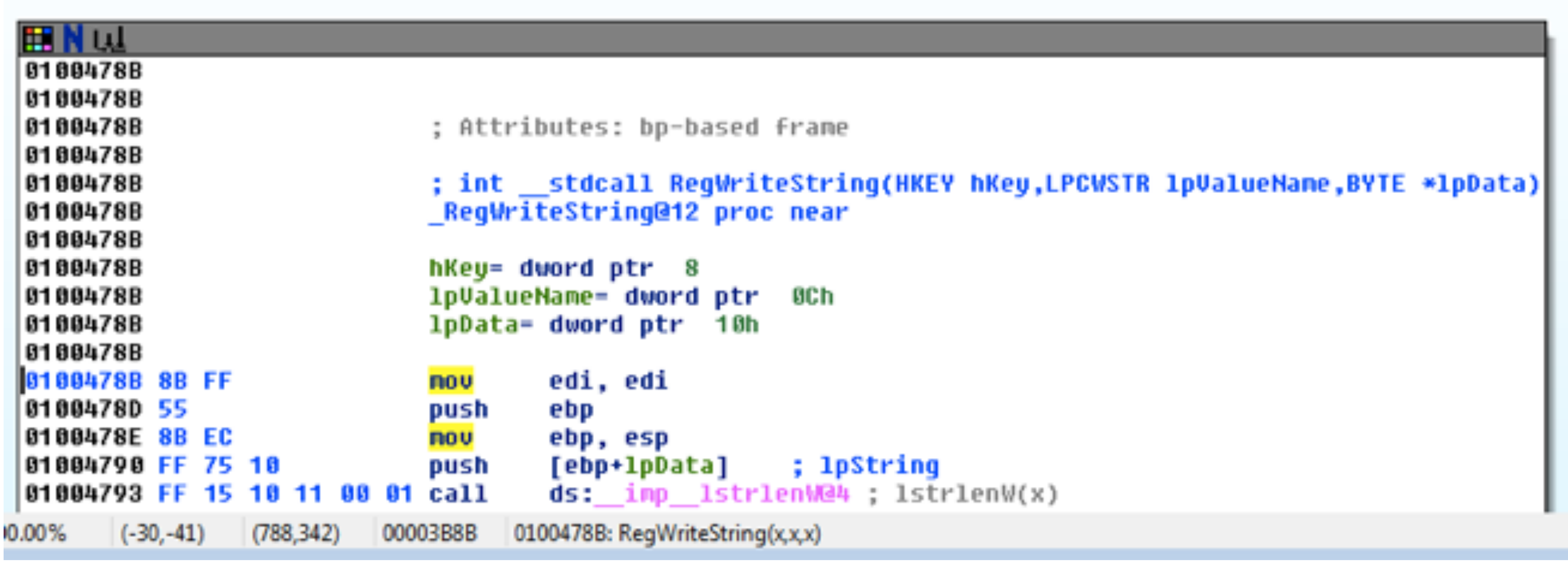


Hex View-A

```
.text:01004740  FF C9 C2 04 00 90 90 90  06 00 00 00 BA 0B 00 00   +-┘.ÉÉÉ-...¦♂..
.text:01004750  07 00 00 00 BB 0B 00 00  50 FF D6 E9 D0 D2 FF FF   •...+♂..P +T--
.text:01004760  50 FF D6 E9 D5 D2 FF FF  0F B7 16 66 85 D2 0F 84   P +T+-  …+┬fà-…ä
.text:01004770  8A 00 00 00 3B FB 0F 83  82 00 00 00 66 89 10 40   è...;v…âé...fë╪@
.text:01004780  40 46 46 47 EB E2 90 90  90 90 90 8B FF 55 8B EC   @FFGdGÉÉÉÉÉï Uï8
.text:01004790  FF 75 10 FF 15 10 11 00  01 8D 44 00 02 50 FF 75    u╪ §╪◄. ìD.┐P u
.text:010047A0  10 6A 01 6A 00 FF 75 0C  FF 75 08 FF 15 00 10 00   ╪j j. u♀ u◘ §.╪.
.text:010047B0  01 5D C2 0C 00 90 90 90  90 90 8B FF 55 8B EC 8B    ]-♀.ÉÉÉÉÉï Uï8ï
```

Function Call

- Parameters pushed onto stack
- CALL to start function



```
0100478B
0100478B
0100478B                           ; Attributes: bp-based frame
0100478B
0100478B                           ; int __stdcall RegWriteString(HKEY hKey,LPCWSTR lpValueName,BYTE *lpData)
0100478B                           _RegWriteString@12 proc near
0100478B
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)
```

```
0.00%    (-30,-41)    (788,342)    00003B8B    0100478B: RegWriteString(x,x,x)
```

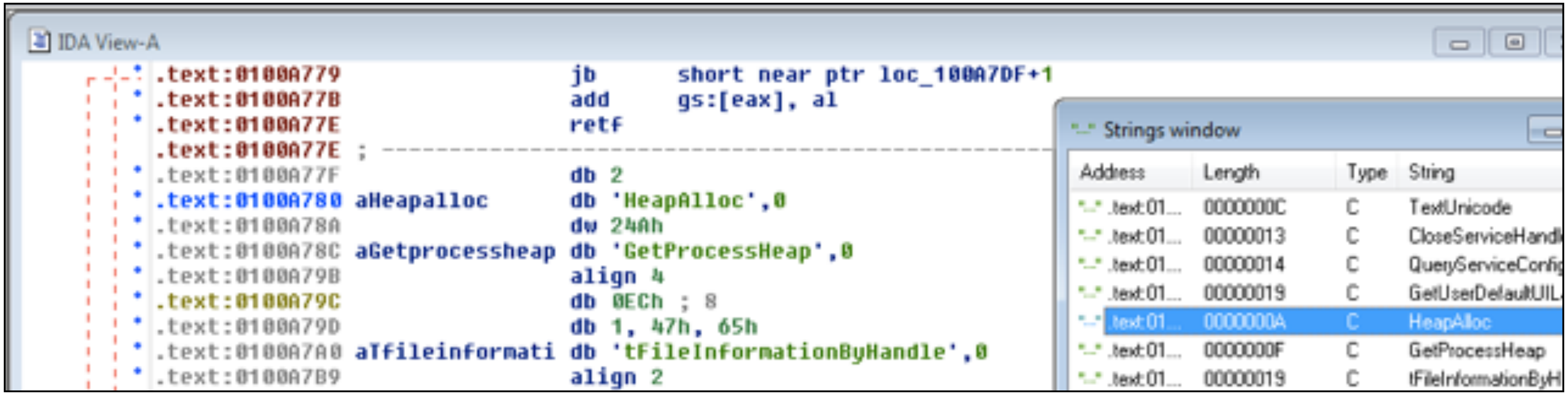
Returning to the Default View

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

Navigating IDA Pro

Imports or Strings

- Double-click any entry to display it in the disassembly window

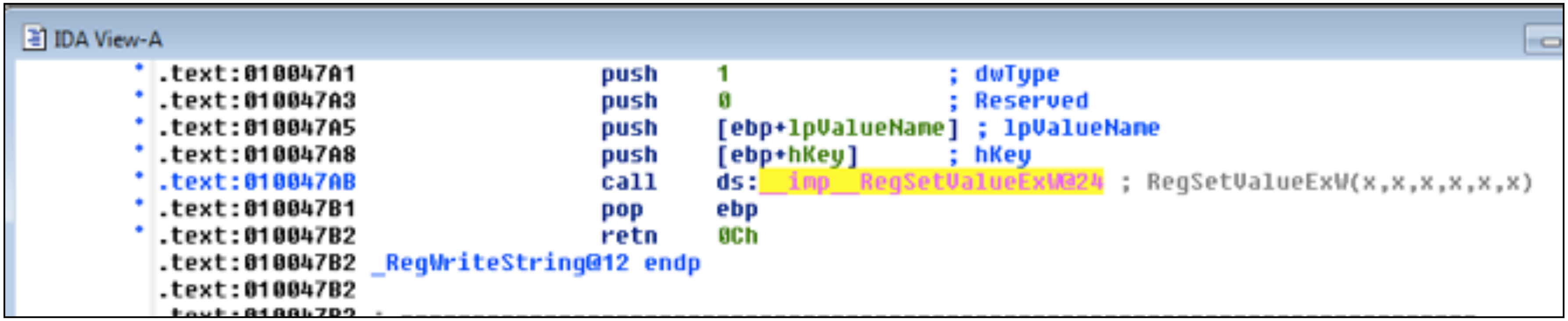


```
.text:0100A779                   jb      short near ptr loc_100A7DF+1
.text:0100A77B                   add     gs:[eax], al
.text:0100A77E                   retf
.text:0100A77E ; ---------------------------------------------
.text:0100A77F                   db 2
.text:0100A780 aHeapalloc        db 'HeapAlloc',0
.text:0100A78A                   dw 24Ah
.text:0100A78C aGetprocessheap db 'GetProcessHeap',0
.text:0100A79B                   align 4
.text:0100A79C                   db 0ECh ; 8
.text:0100A79D                   db 1, 47h, 65h
.text:0100A7A0 aTfileinformati db 'tFileInformationByHandle',0
.text:0100A7B9                   align 2
```

Address	Length	Type	String
.text:01...	0000000C	C	TextUnicode
.text:01...	00000013	C	CloseServiceHandl
.text:01...	00000014	C	QueryServiceConfig
.text:01...	00000019	C	GetUserDefaultUIL
.text:01...	0000000A	C	HeapAlloc
.text:01...	0000000F	C	GetProcessHeap
.text:01...	00000019	C	tFileInformationByH

Using Links

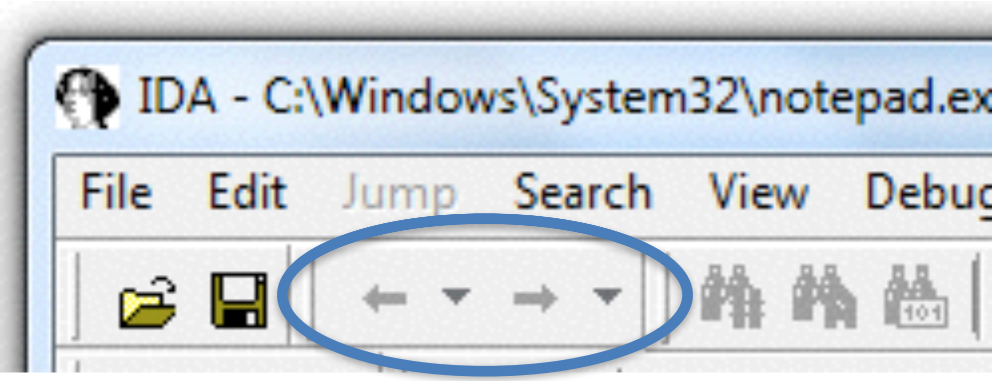
- Double-click any address in the disassembly window to display that location



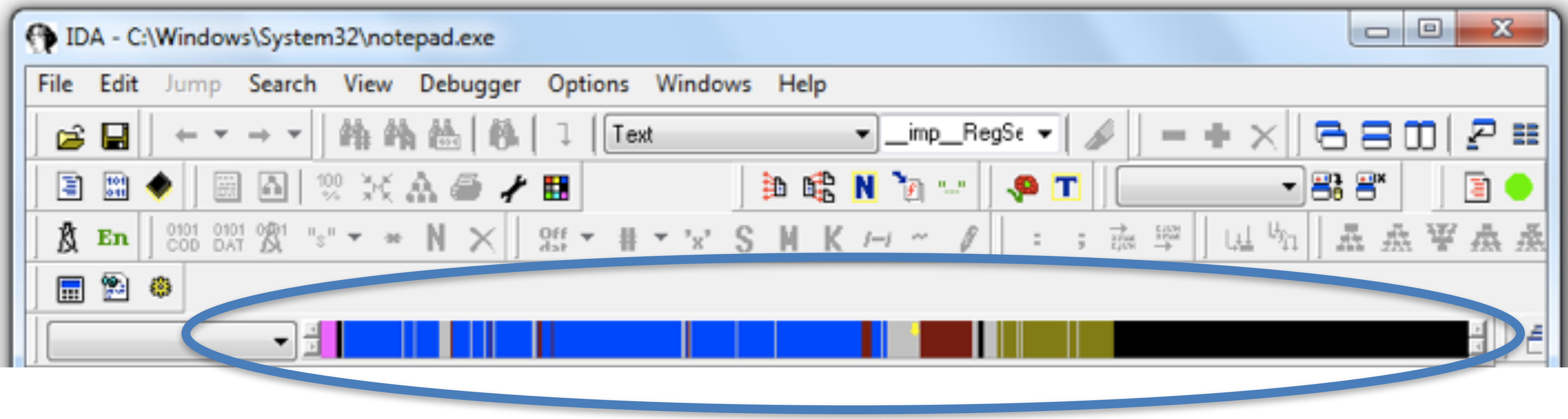
```
.text:010047A1                 push    1               ; dwType
.text:010047A3                 push    0               ; Reserved
.text:010047A5                 push    [ebp+lpValueName] ; lpValueName
.text:010047A8                 push    [ebp+hKey]      ; hKey
.text:010047AB                 call    ds:__imp__RegSetValueExW@24 ; RegSetValueExW(x,x,x,x,x,x)
.text:010047B1                 pop     ebp
.text:010047B2                 retn    0Ch
.text:010047B2 _RegWriteString@12 endp
.text:010047B2
.text:010047B3 ; ---------------------------------------------------------------------------
```

History

- Forward and Back buttons work like a Web browser



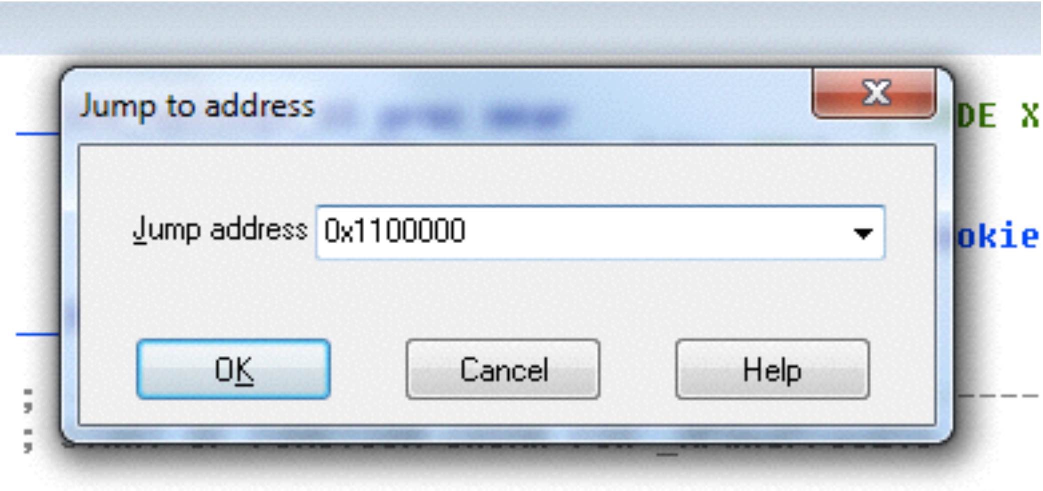
Navigation Band



- **Light blue:** Library code
- **Red:** Compiler-generated code
- **Dark blue:** User-written code – **Analyze this**

Jump to Location

- Press **G**
- Can jump to address or named location

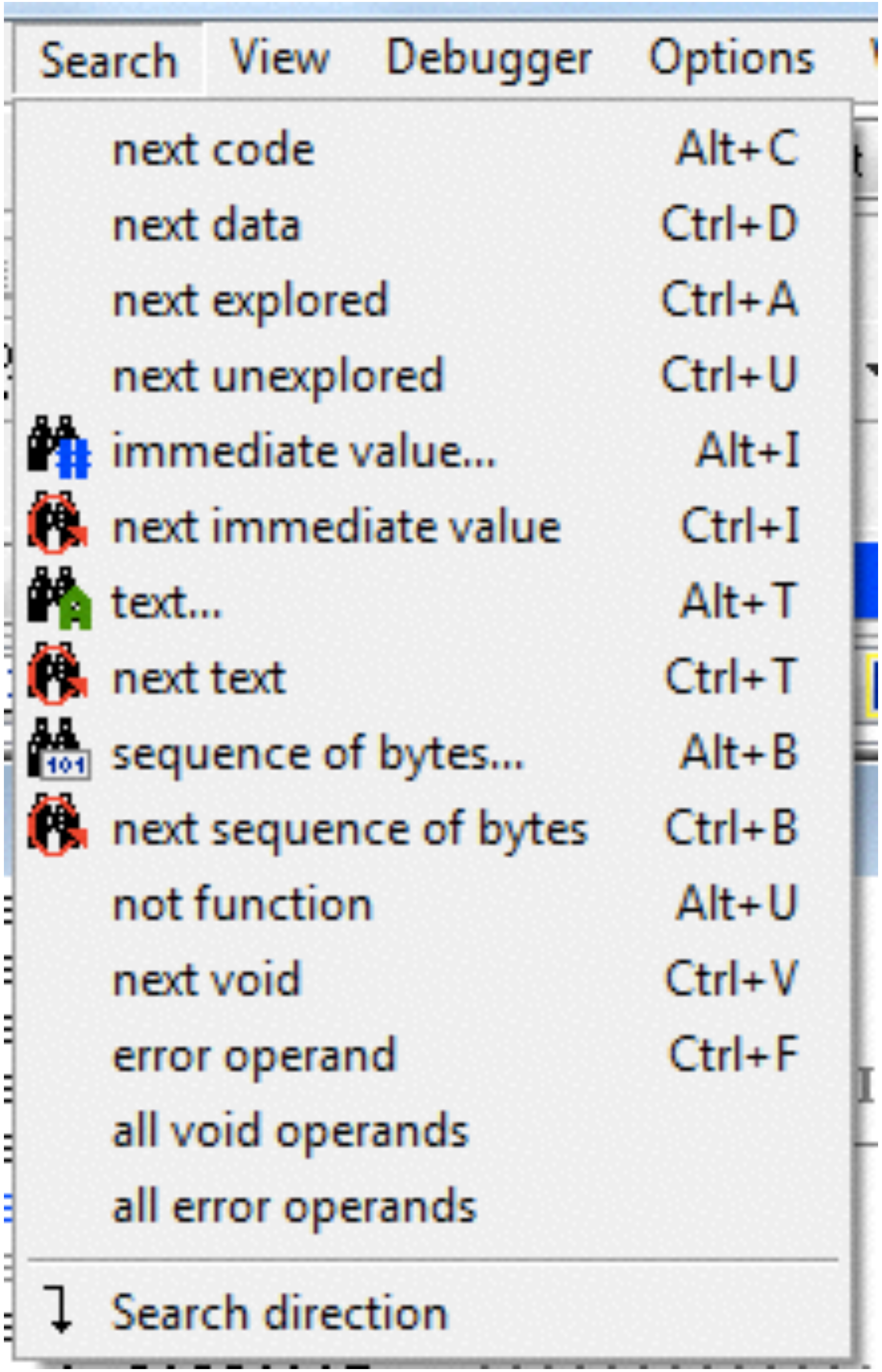


Jump to address

Jump address 0x1100000

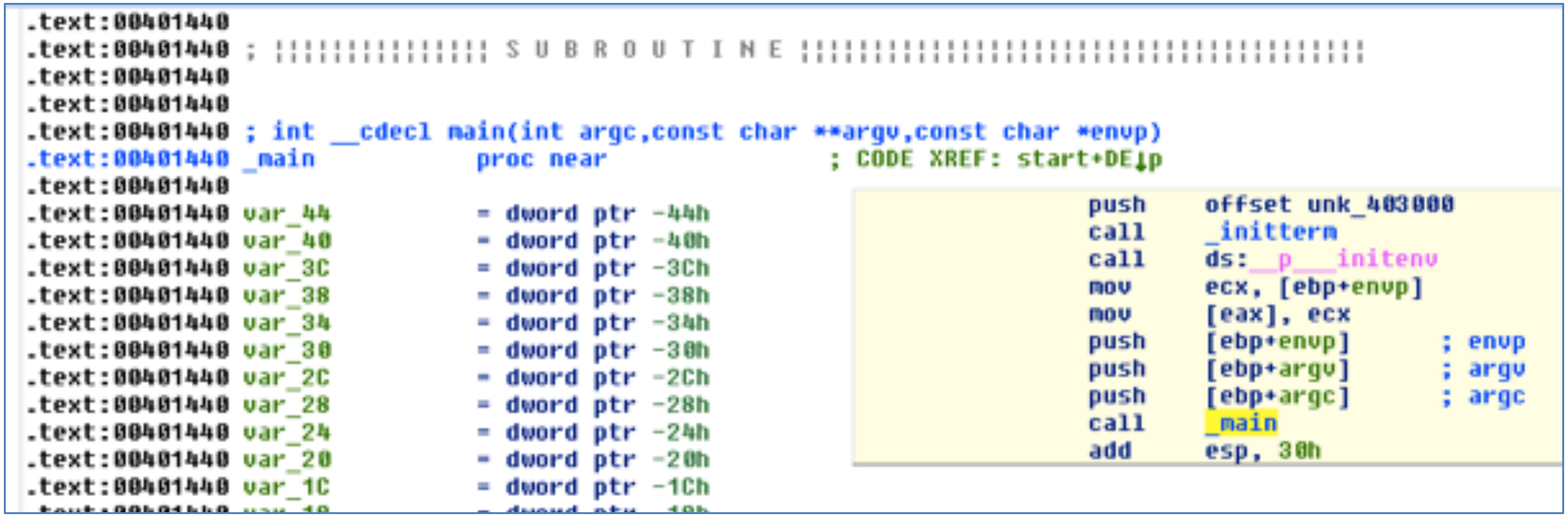
Searching

- Many options
- Search, Text is handy



Using Cross-References

Code Cross-References



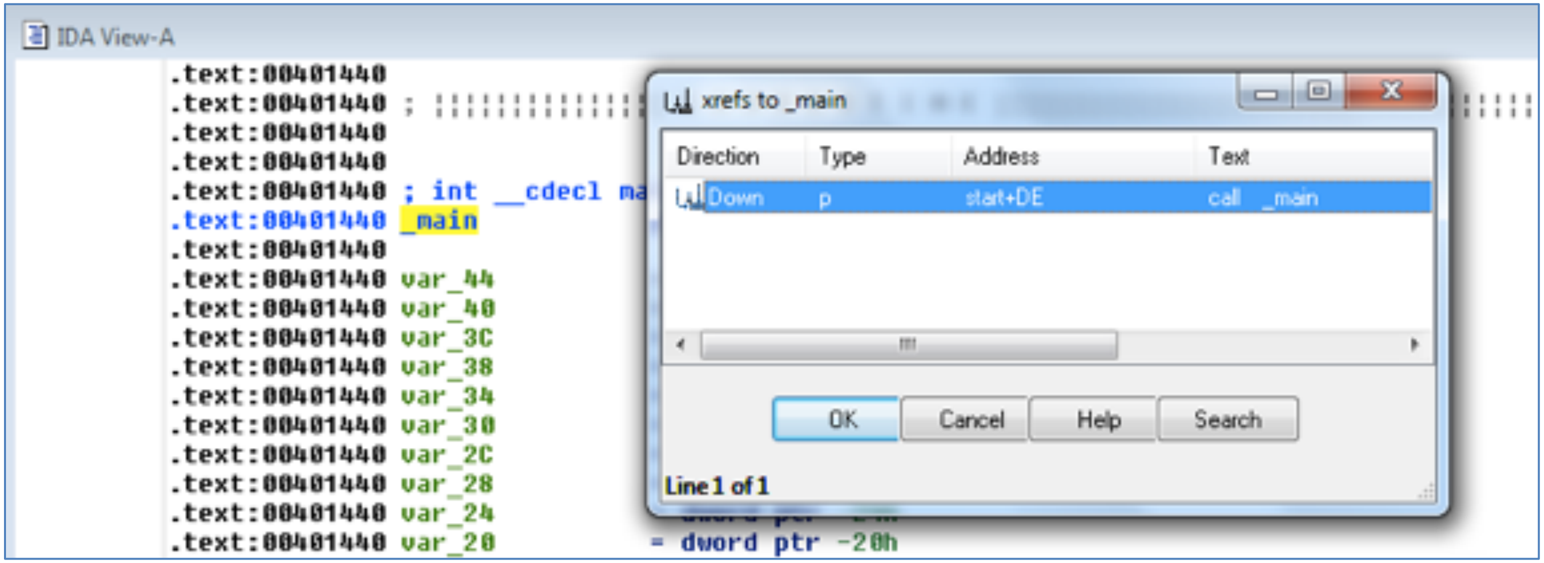
```
.text:00401440
.text:00401440 ; ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦ S U B R O U T I N E ¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦¦
.text:00401440
.text:00401440
.text:00401440 ; int __cdecl main(int argc,const char **argv,const char *envp)
.text:00401440 _main           proc near               ; CODE XREF: start+DE↓p
.text:00401440
.text:00401440 var_44          = dword ptr -44h
.text:00401440 var_40          = dword ptr -40h
.text:00401440 var_3C          = dword ptr -3Ch
.text:00401440 var_38          = dword ptr -38h
.text:00401440 var_34          = dword ptr -34h
.text:00401440 var_30          = dword ptr -30h
.text:00401440 var_2C          = dword ptr -2Ch
.text:00401440 var_28          = dword ptr -28h
.text:00401440 var_24          = dword ptr -24h
.text:00401440 var_20          = dword ptr -20h
.text:00401440 var_1C          = dword ptr -1Ch
```

```
                push    offset unk_403000
                call    _initterm
                call    ds:__p___initenv
                mov     ecx, [ebp+envp]
                mov     [eax], ecx
                push    [ebp+envp]      ; envp
                push    [ebp+argv]      ; argv
                push    [ebp+argc]      ; argc
                call    _main
                add     esp, 30h
```

- XREF comment shows where this function is called
- But it only shows a couple of cross-references by default

To See All Cross-References

- Click function name and press X



```
.text:00401440
.text:00401440 ; ::::::::::::::
.text:00401440
.text:00401440
.text:00401440 ; int __cdecl ma
.text:00401440 _main
.text:00401440
.text:00401440 var_44
.text:00401440 var_40
.text:00401440 var_3C
.text:00401440 var_38
.text:00401440 var_34
.text:00401440 var_30
.text:00401440 var_2C
.text:00401440 var_28
.text:00401440 var_24
.text:00401440 var_20                = dword ptr -20h
```

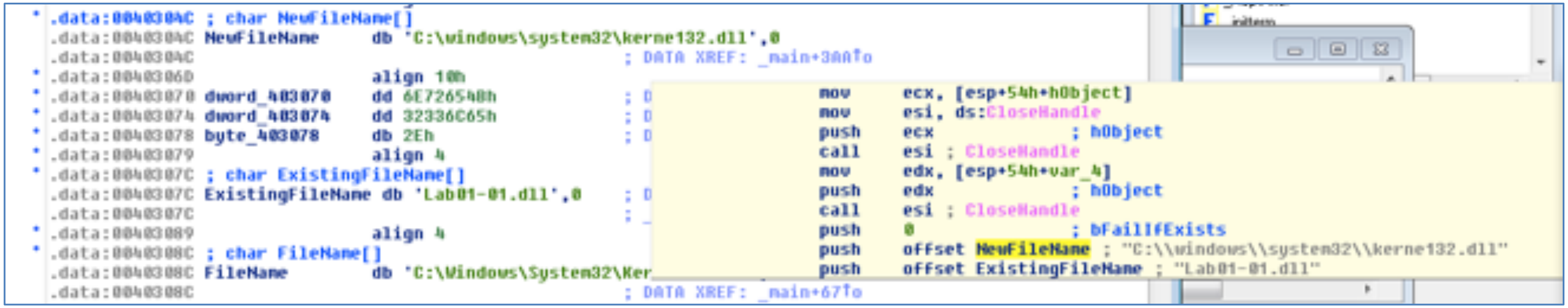
xrefs to _main

Direction	Type	Address	Text
Down	p	start+DE	call _main

Line 1 of 1

Data Cross-References

- Demo:
  - Start with strings
  - Double-click an interesting string
  - Hover over DATA XREF to see where that string is used
  - X shows all references



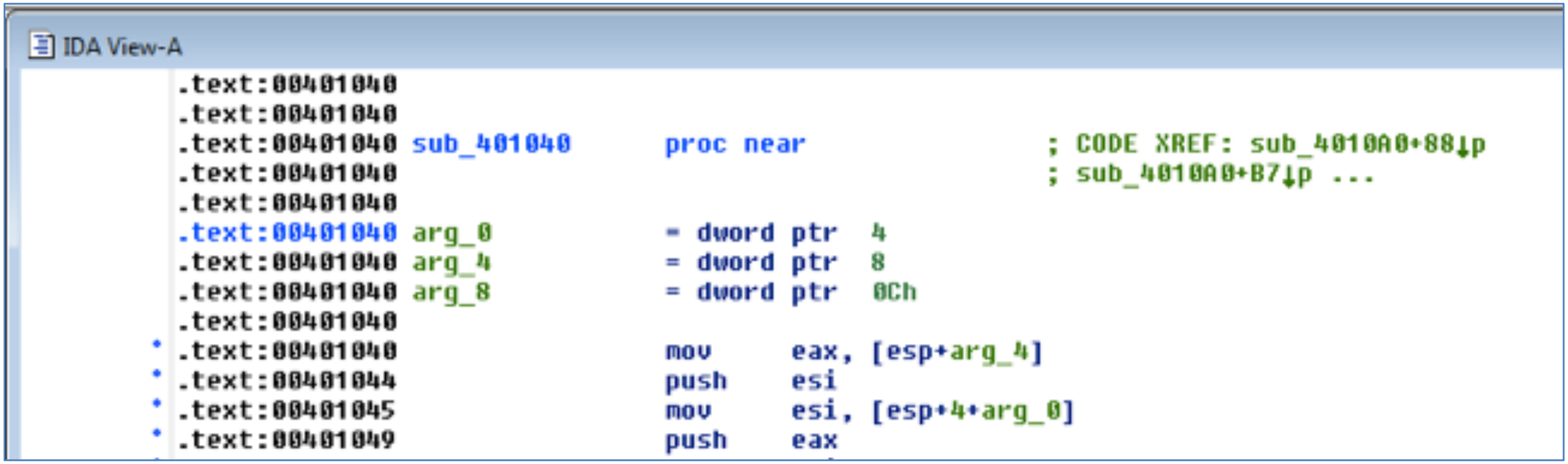
```
.data:0040304C ; char NewFileName[]
.data:0040304C NewFileName     db 'C:\windows\system32\kerne132.dll',0
.data:0040304C                                         ; DATA XREF: _main+3AA↑o
.data:0040306D                 align 10h
.data:00403070 dword_403070    dd 6E72654Bh            ; D
.data:00403074 dword_403074    dd 32336C65h            ; D
.data:00403078 byte_403078     db 2Eh                  ; D
.data:00403079                 align 4
.data:0040307C ; char ExistingFileName[]
.data:0040307C ExistingFileName db 'Lab01-01.dll',0    ; D
.data:0040307C                                         ; _
.data:00403089                 align 4
.data:0040308C ; char FileName[]
.data:0040308C FileName        db 'C:\Windows\System32\Ker
.data:0040308C                                         ; DATA XREF: _main+67↑o
```

```
                mov     ecx, [esp+54h+hObject]
                mov     esi, ds:CloseHandle
                push    ecx             ; hObject
                call    esi ; CloseHandle
                mov     edx, [esp+54h+var_4]
                push    edx             ; hObject
                call    esi ; CloseHandle
                push    0               ; bFailIfExists
                push    offset NewFileName ; "C:\\windows\\system32\\kerne132.dll"
                push    offset ExistingFileName ; "Lab01-01.dll"
```

Analyzing Functions

Function and Argument Recognition

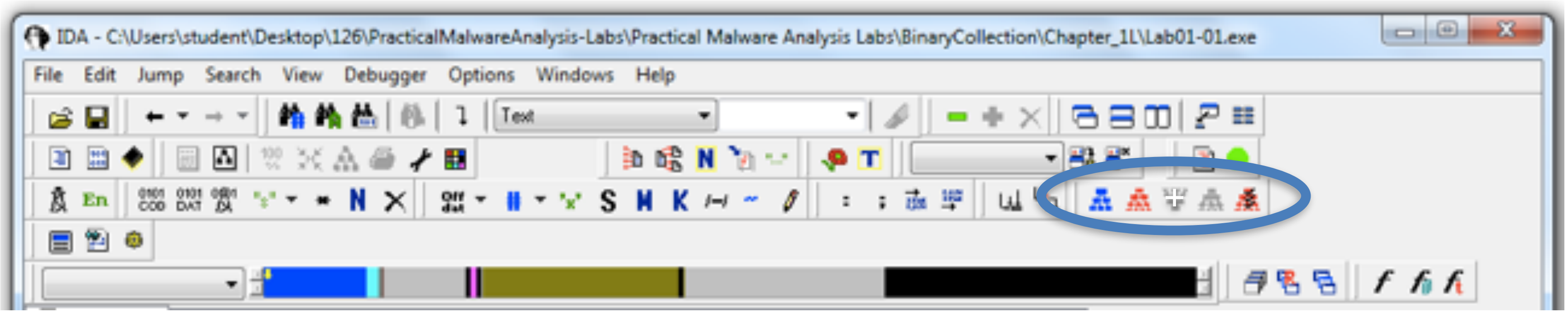
- IDA Pro identifies a function, names it, and also names the local variables
- It's not always correct

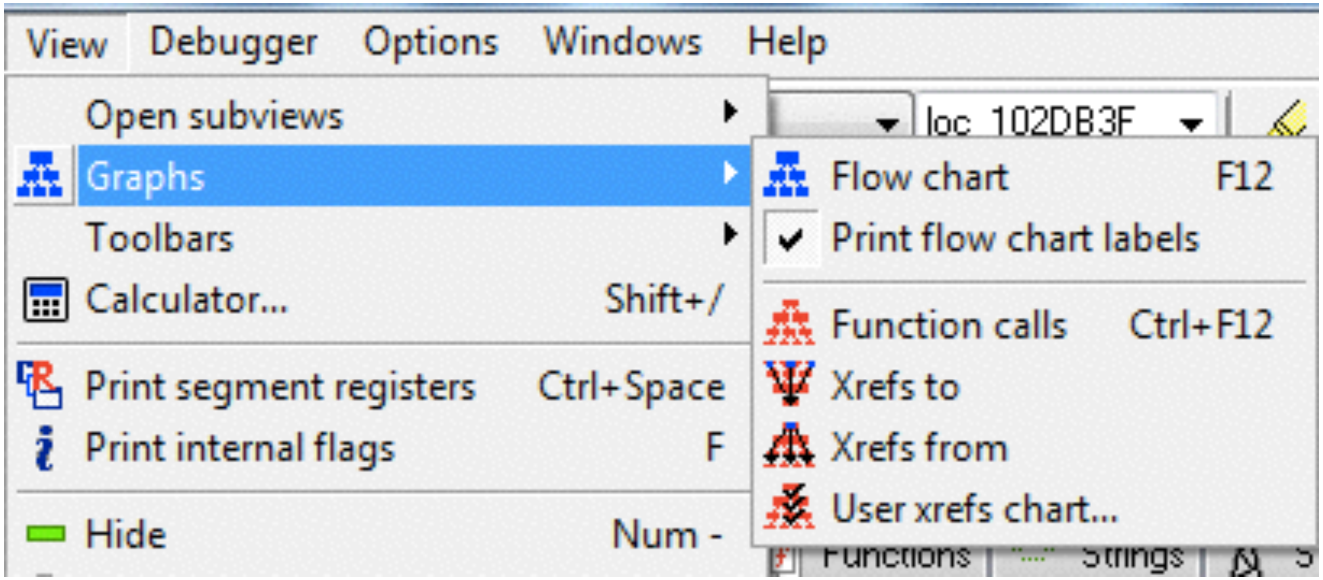


```
.text:00401040
.text:00401040
.text:00401040 sub_401040      proc near               ; CODE XREF: sub_4010A0+88↓p
.text:00401040                                         ; sub_4010A0+B7↓p ...
.text:00401040
.text:00401040 arg_0           = dword ptr  4
.text:00401040 arg_4           = dword ptr  8
.text:00401040 arg_8           = dword ptr  0Ch
.text:00401040
.text:00401040                 mov     eax, [esp+arg_4]
.text:00401044                 push    esi
.text:00401045                 mov     esi, [esp+4+arg_0]
.text:00401049                 push    eax
```

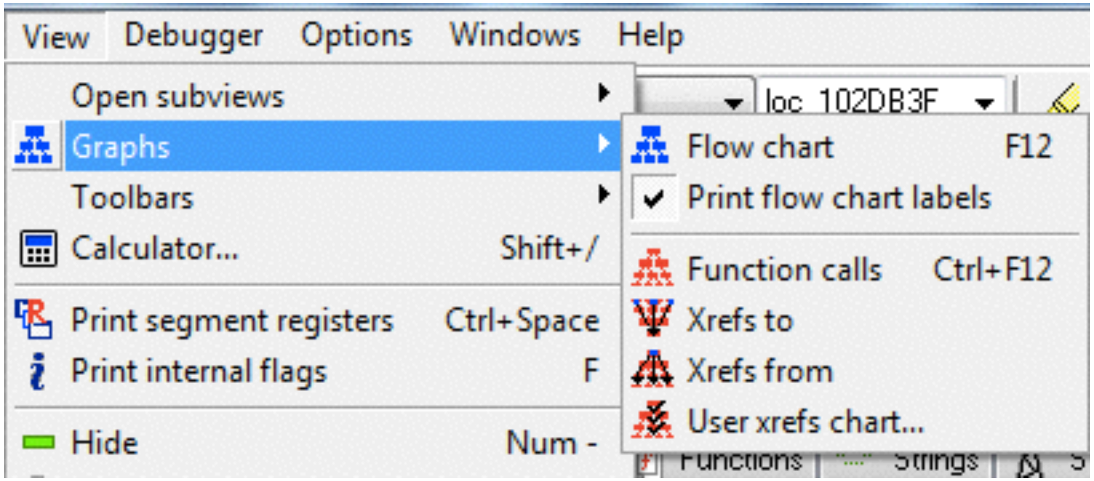
Using Graphing Options

Graphing Options


IDA - C:\Users\student\Desktop\126\PracticalMalwareAnalysis-Labs\Practical Malware Analysis Labs\BinaryCollection\Chapter_1L\Lab01-01.exe
File
Edit
Jump
Search
View
Debugger
Options
Windows
Help
Text

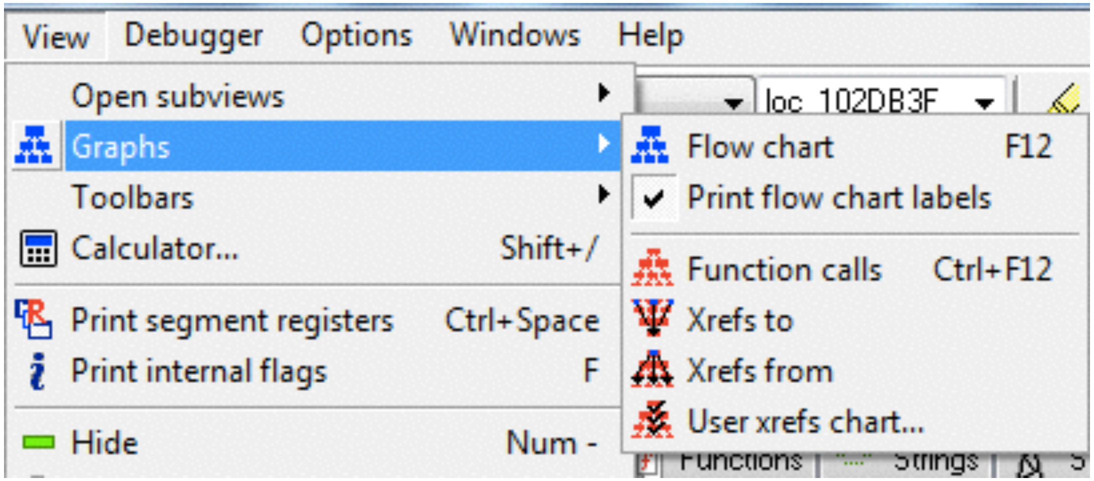


Graphing Options



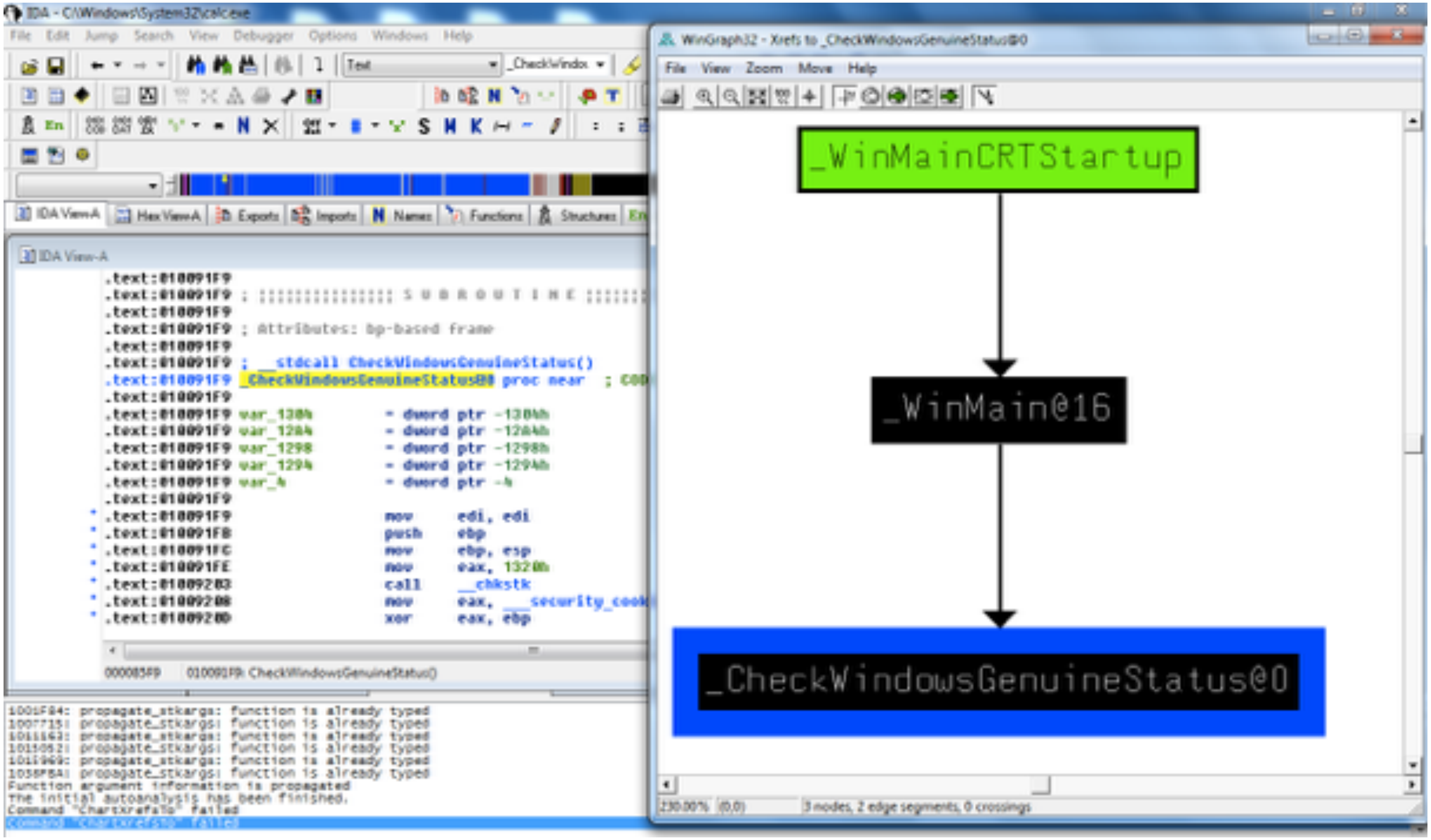
- These are "Legacy Graphs" and cannot be manipulated with IDA
- The first two seem obsolete
  - **Flow chart**
    - Create flow chart of current function
  - **Function calls**
    - Graph function calls for entire program

Graphing Options



- **Xrefs to**
  - Graphs XREFs to get to selected XREF
  - Can show all the paths that get to a function

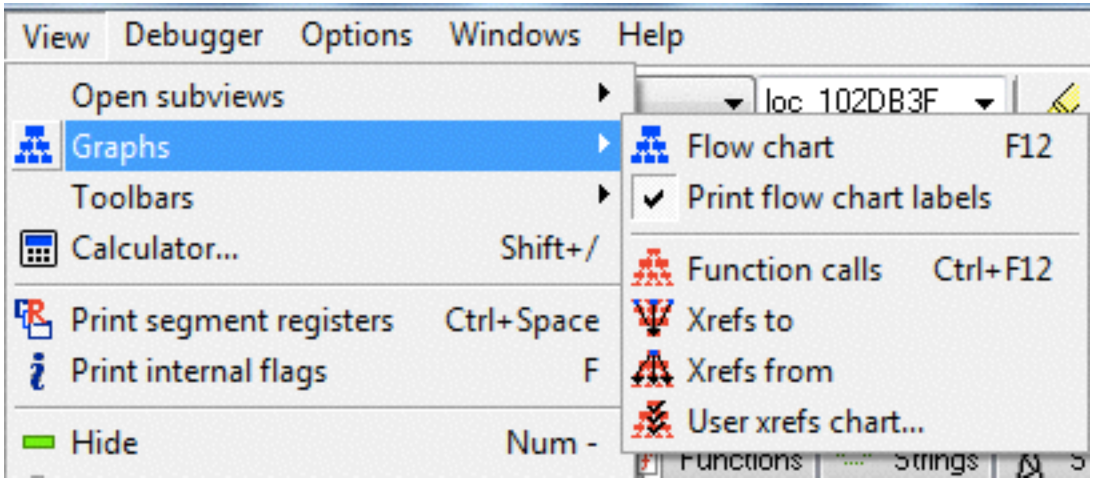
Windows Genuine Status in Calc.exe


IDA - C:\Windows\System32\calc.exe
File Edit Jump Search View Debugger Options Windows Help
IDA View-A
Hex View-A
Exports
Imports
Names
Functions
Structures
WinGraph32 - Xrefs to _CheckWindowsGenuineStatus@0
File View Zoom Move Help
_WinMainCRTStartup
_WinMain@16
_CheckWindowsGenuineStatus@0
230.00% (0,0) 3 nodes, 2 edge segments, 0 crossings

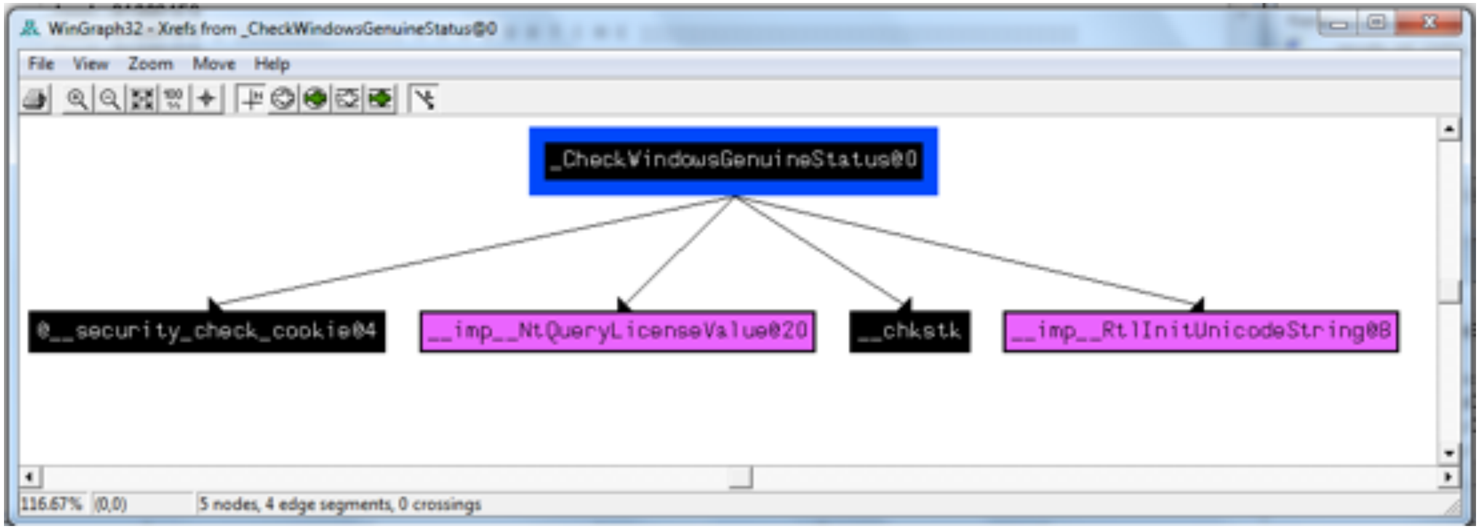
```
.text:010091F9
.text:010091F9 ; ::::::::::::::: S U B R O U T I N E :::::::
.text:010091F9
.text:010091F9 ; Attributes: bp-based frame
.text:010091F9
.text:010091F9 ; __stdcall CheckWindowsGenuineStatus()
.text:010091F9 _CheckWindowsGenuineStatus@0 proc near   ; COD
.text:010091F9
.text:010091F9 var_1304       = dword ptr -1304h
.text:010091F9 var_12A4       = dword ptr -12A4h
.text:010091F9 var_1298       = dword ptr -1298h
.text:010091F9 var_1294       = dword ptr -1294h
.text:010091F9 var_4          = dword ptr -4
.text:010091F9
.text:010091F9                mov     edi, edi
.text:010091FB                push    ebp
.text:010091FC                mov     ebp, esp
.text:010091FE                mov     eax, 1320h
.text:01009203                call    __chkstk
.text:01009208                mov     eax, ___security_cook
.text:0100920D                xor     eax, ebp

000085F9    010091F9: CheckWindowsGenuineStatus()
```

```
1001F84: propagate_stkargs: function is already typed
1007715: propagate_stkargs: function is already typed
1011163: propagate_stkargs: function is already typed
1015052: propagate_stkargs: function is already typed
1011969: propagate_stkargs: function is already typed
103BF84: propagate_stkargs: function is already typed
Function argument information is propagated
The initial autoanalysis has been finished.
Command "ChartXrefsTo" failed
Command "ChartXrefsTo" failed
```

Graphing Options

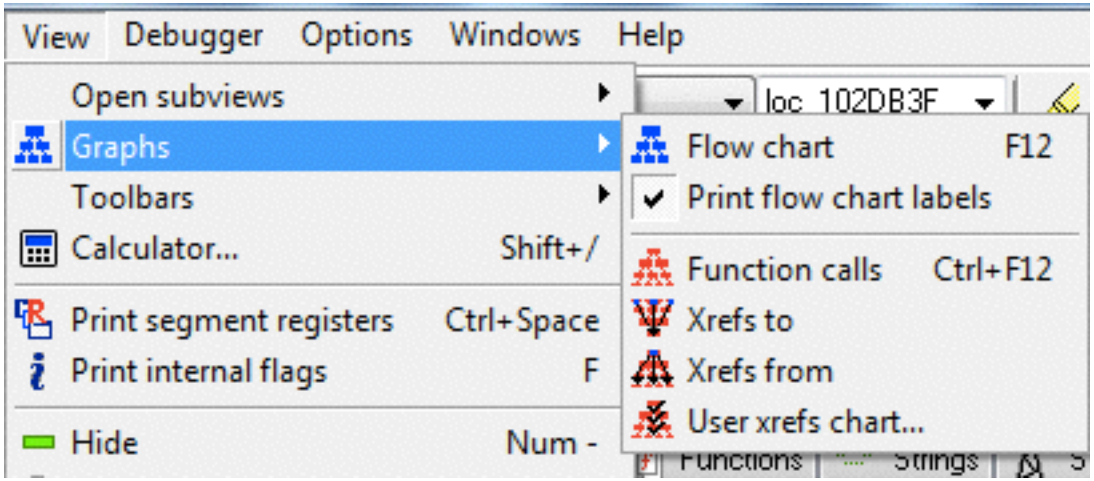


- **Xrefs from**
  - Graphs XREFs from selected XREF
  - Can show all the paths that exit from a function


WinGraph32 - Xrefs from _CheckWindowsGenuineStatus@0
File
View
Zoom
Move
Help
_CheckWindowsGenuineStatus@0
@__security_check_cookie@4
__imp__NtQueryLicenseValue@20
__chkstk
__imp__RtlInitUnicodeString@8

116.67% (0,0) 5 nodes, 4 edge segments, 0 crossings

Graphing Options



- **User xrefs chart…**
  - Customize graph's recursive depth, symbols used, to or from symbol, etc.
  - The only way to modify legacy graphs

Enhancing Disassembly

Warning

- There's no Undo, so if you make changes and mess them up, you may be sorry

Renaming Locations

- You can change a name like **sub_401000** to **ReverseBackdoorThread**
- Change it in one place, IDA will change it everywhere else

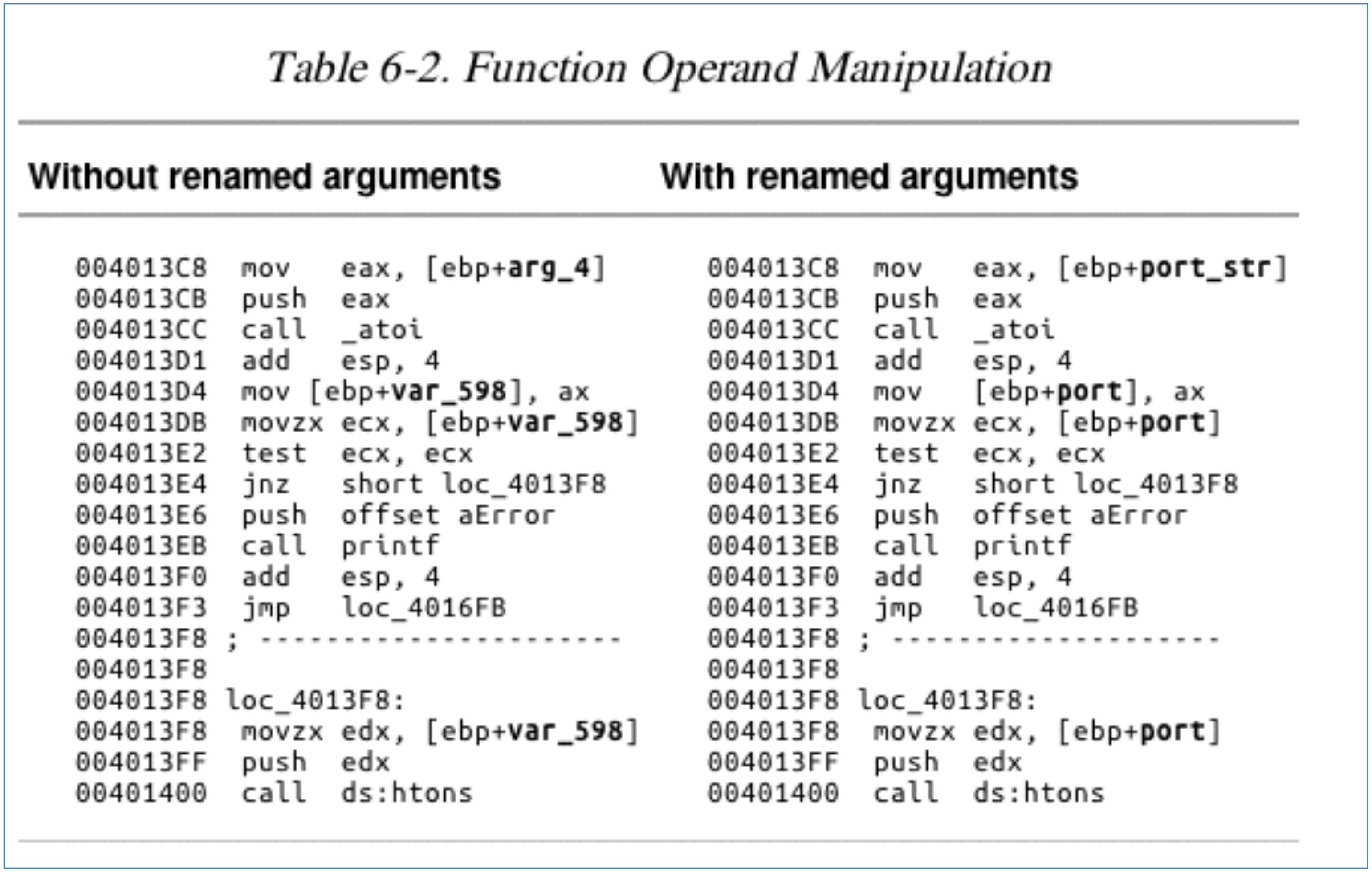


Table 6-2. Function Operand Manipulation

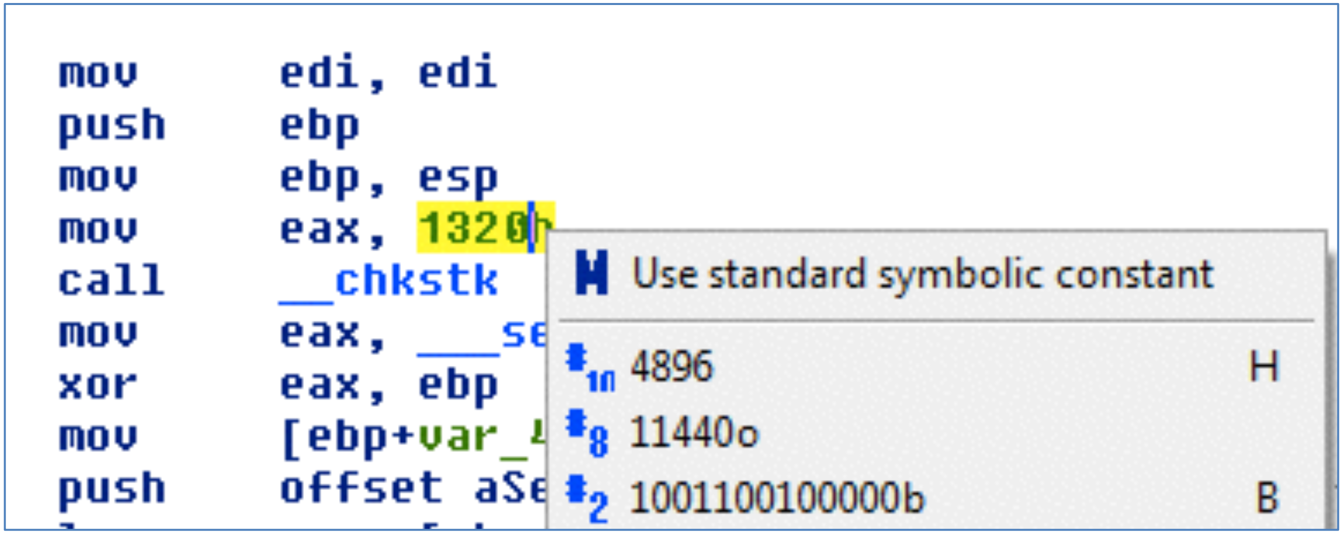
Without renamed arguments	With renamed arguments
004013C8 mov eax, [ebp+**arg_4**]	004013C8 mov eax, [ebp+**port_str**]
004013CB push eax	004013CB push eax
004013CC call _atoi	004013CC call _atoi
004013D1 add esp, 4	004013D1 add esp, 4
004013D4 mov [ebp+**var_598**], ax	004013D4 mov [ebp+**port**], ax
004013DB movzx ecx, [ebp+**var_598**]	004013DB movzx ecx, [ebp+**port**]
004013E2 test ecx, ecx	004013E2 test ecx, ecx
004013E4 jnz short loc_4013F8	004013E4 jnz short loc_4013F8
004013E6 push offset aError	004013E6 push offset aError
004013EB call printf	004013EB call printf
004013F0 add esp, 4	004013F0 add esp, 4
004013F3 jmp loc_4016FB	004013F3 jmp loc_4016FB
004013F8 ; ---------------------	004013F8 ; -------------------
004013F8	004013F8
004013F8 loc_4013F8:	004013F8 loc_4013F8:
004013F8 movzx edx, [ebp+**var_598**]	004013F8 movzx edx, [ebp+**port**]
004013FF push edx	004013FF push edx
00401400 call ds:htons	00401400 call ds:htons

Comments

- Press colon (:) to add a single comment
- Press semicolon (;) to echo this comment to all Xrefs

Formatting Operands

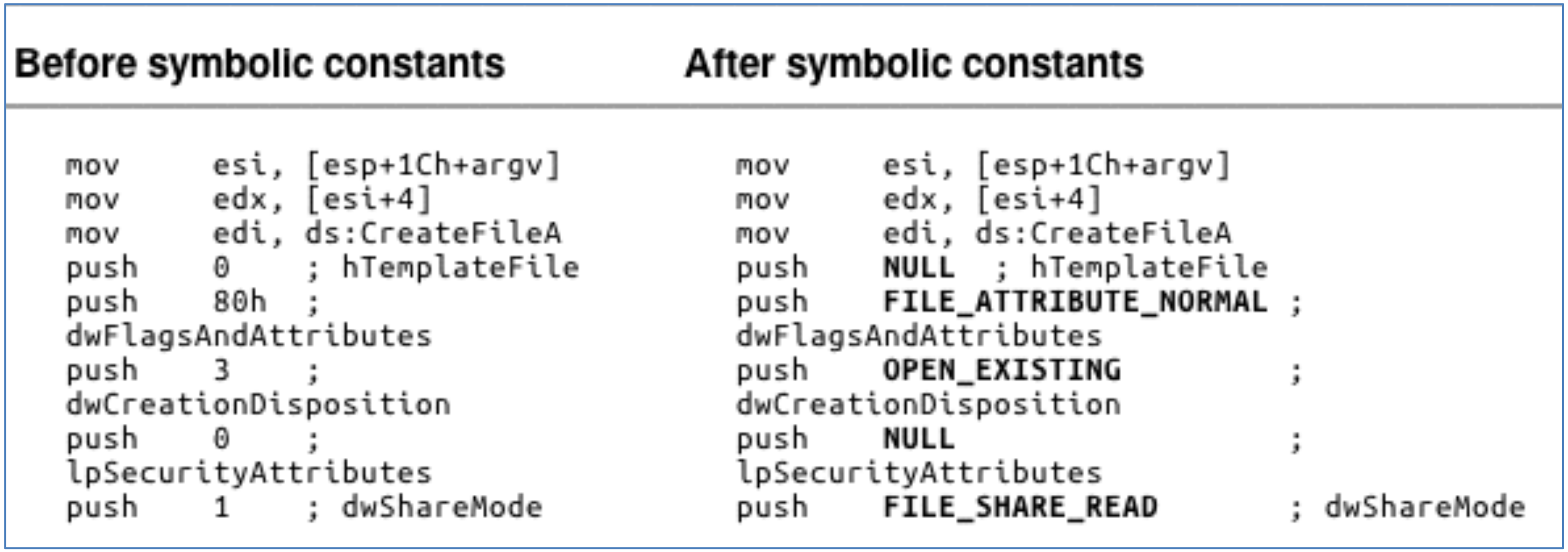
- Hexadecimal by default
- Right-click to use other formats



```
mov     edi, edi
push    ebp
mov     ebp, esp
mov     eax, 1320h
call    __chkstk
mov     eax, ___se
xor     eax, ebp
mov     [ebp+var_4
push    offset aSe
```

Using Named Constants

- Makes Windows API arguments clearer



Before symbolic constants

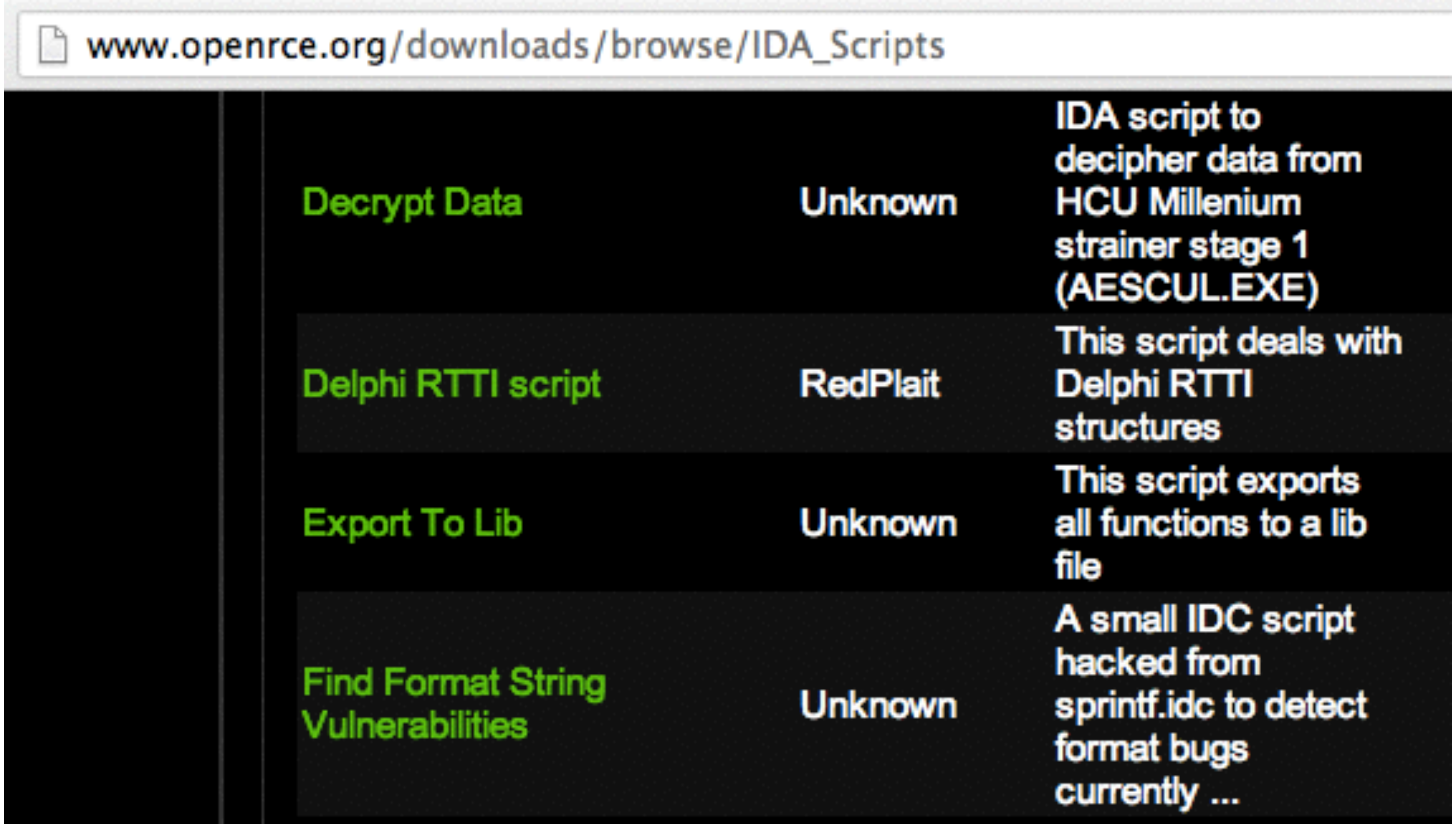
```
mov     esi, [esp+1Ch+argv]
mov     edx, [esi+4]
mov     edi, ds:CreateFileA
push    0    ; hTemplateFile
push    80h  ;
dwFlagsAndAttributes
push    3    ;
dwCreationDisposition
push    0    ;
lpSecurityAttributes
push    1    ; dwShareMode
```

After symbolic constants

```
mov     esi, [esp+1Ch+argv]
mov     edx, [esi+4]
mov     edi, ds:CreateFileA
push    NULL  ; hTemplateFile
push    FILE_ATTRIBUTE_NORMAL ;
dwFlagsAndAttributes
push    OPEN_EXISTING         ;
dwCreationDisposition
push    NULL                  ;
lpSecurityAttributes
push    FILE_SHARE_READ       ; dwShareMode
```

Extending IDA with Plug-ins

- IDC (IDA's scripting language) and Python scripts available (link Ch 6a)



www.openrce.org/downloads/browse/IDA_Scripts

Decrypt Data	Unknown	IDA script to decipher data from HCU Millenium strainer stage 1 (AESCUL.EXE)
Delphi RTTI script	RedPlait	This script deals with Delphi RTTI structures
Export To Lib	Unknown	This script exports all functions to a lib file
Find Format String Vulnerabilities	Unknown	A small IDC script hacked from sprintf.idc to detect format bugs currently ...