x86 & PE

$berlinsides_

28th December 2011

Ange Albertini

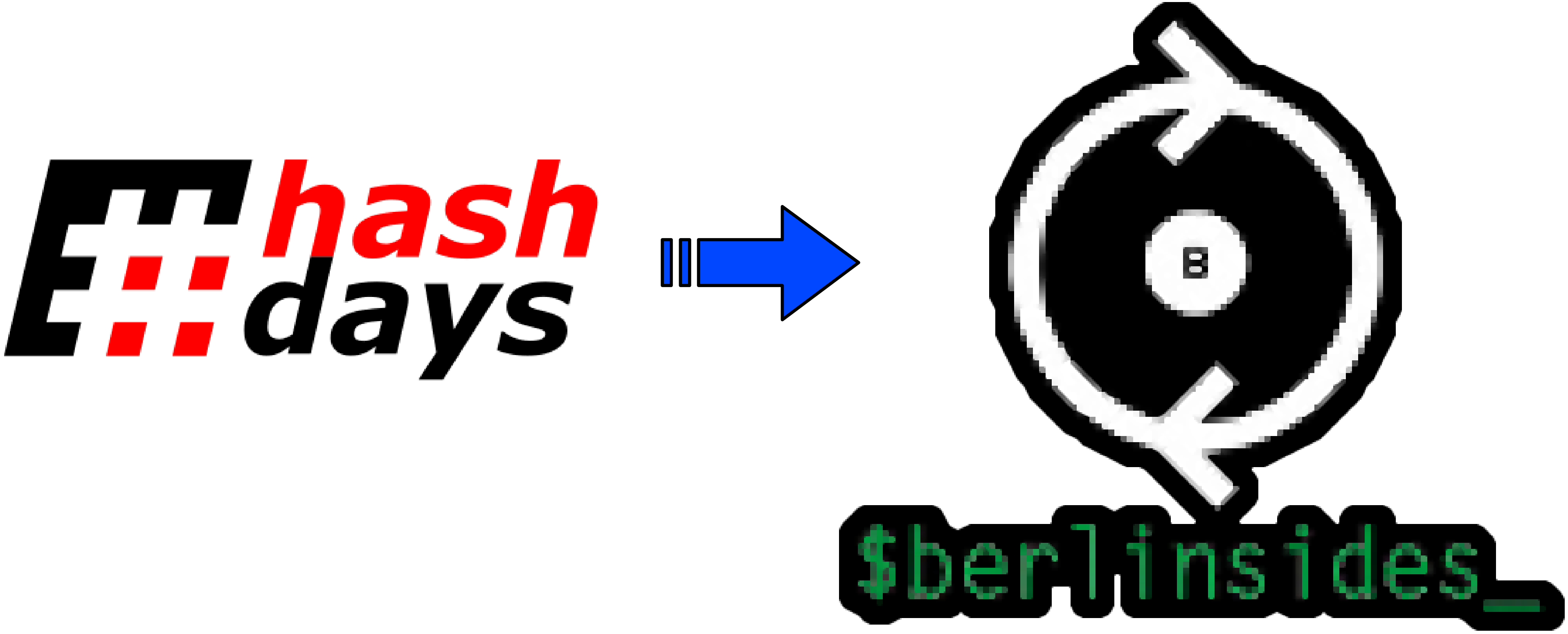
HIDDEN SLIDE

before you decide to read further...

Contents of this slide deck:

1. Introduction
    1. introduce Corkami, my reverse engineering site
    2. explain (in easy terms)
        1. why correct disassembly is important for analysis
        2. why undocumented opcodes are a dead end
2. Main part
    1. a few examples of undocumented opcodes and CPU weirdness
    2. theory-only sucks, so I created CoST for practicing and testing.
    3. CoST also tests PE, but it's not enough by itself
    4. So I documented PE separately, and give some examples.

Improved, but similar

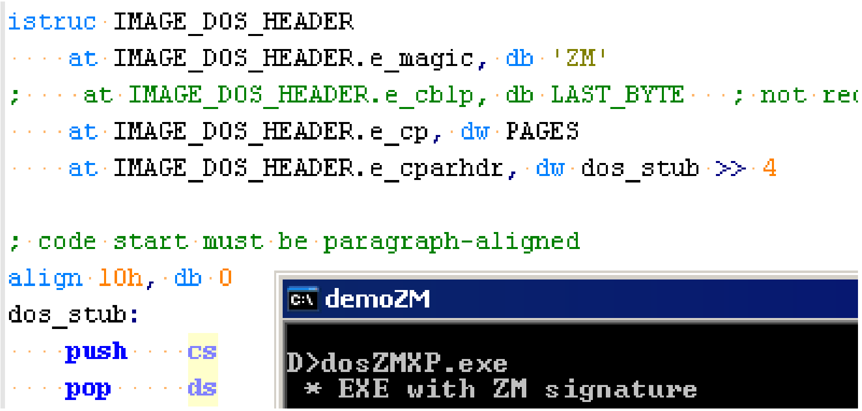


Author

- Corkami
  - reverse engineering
  - technical, really free
  - MANY handmade and focused PoCs
    - nightly builds
    - summary wiki pages
  - but... only a hobby!

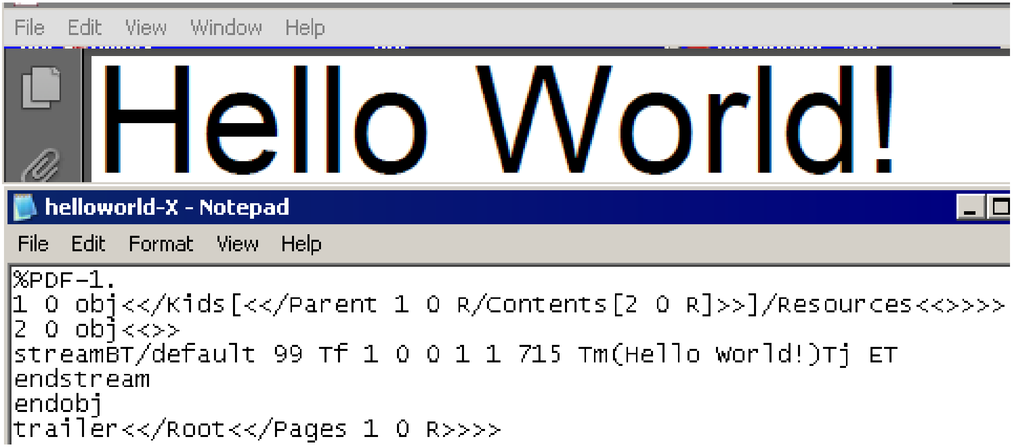
“there's a PoC for that”

and if there's none yet, there will be soon ;)



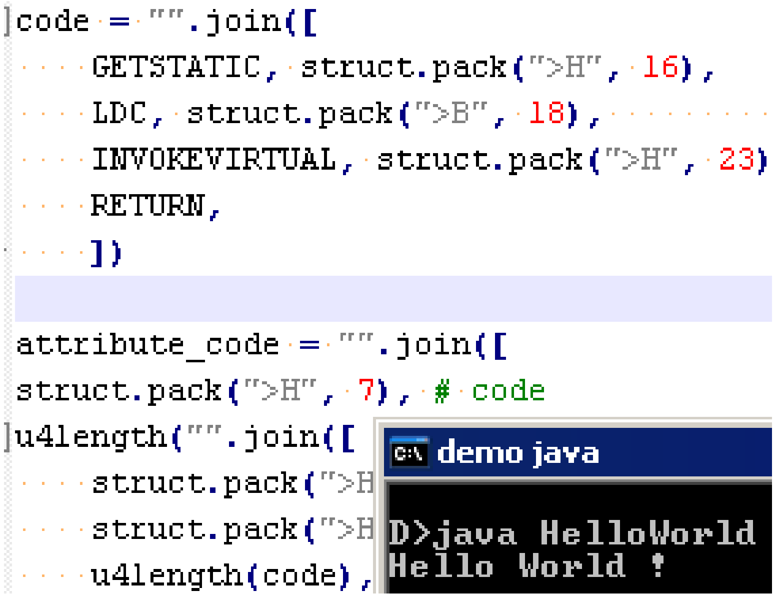
```
istruc IMAGE_DOS_HEADER
    at IMAGE_DOS_HEADER.e_magic, db 'ZM'
;    at IMAGE_DOS_HEADER.e_cblp, db LAST_BYTE   ; not rec
    at IMAGE_DOS_HEADER.e_cp, dw PAGES
    at IMAGE_DOS_HEADER.e_cparhdr, dw dos_stub >> 4

; code start must be paragraph-aligned
align 10h, db 0
dos_stub:
    push    cs
    pop     ds
```

```
D>dosZMXP.exe
 * EXE with ZM signature
```



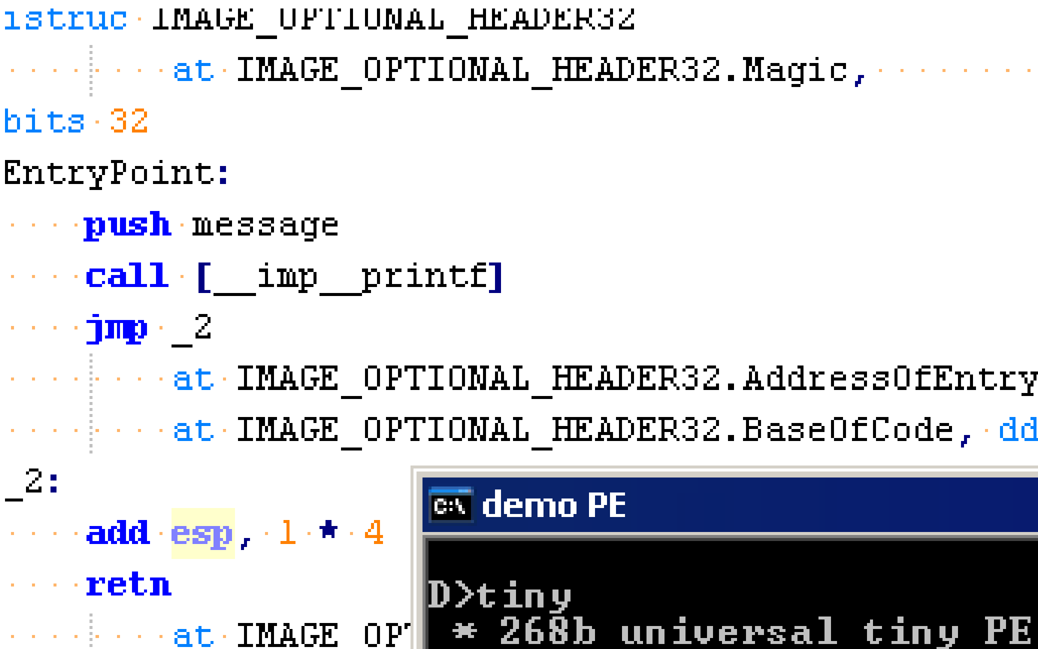
Hello World!

```
%PDF-1.
1 0 obj<</Kids[<</Parent 1 0 R/Contents[2 0 R]>>]/Resources<<>>>>
2 0 obj<<>>
streamBT/default 99 Tf 1 0 0 1 1 715 Tm(Hello World!)Tj ET
endstream
endobj
trailer<</Root<</Pages 1 0 R>>>>
```



```
code = "".join([
    GETSTATIC, struct.pack(">H", 16),
    LDC, struct.pack(">B", 18),
    INVOKEVIRTUAL, struct.pack(">H", 23)
    RETURN,
    ])

attribute_code = "".join([
struct.pack(">H", 7), # code
u4length("".join([
    struct.pack(">H
    struct.pack(">H
    u4length(code),
```

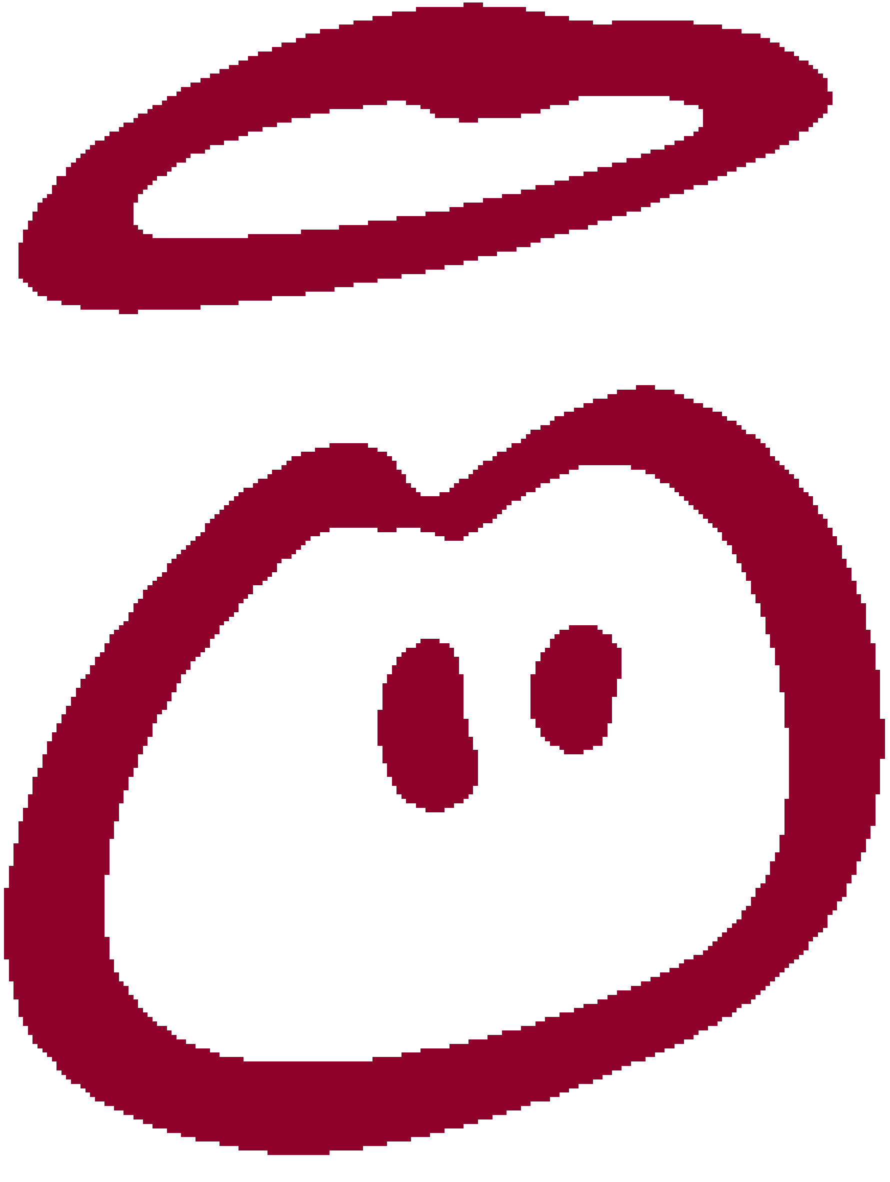
```
D>java HelloWorld
Hello World !
```

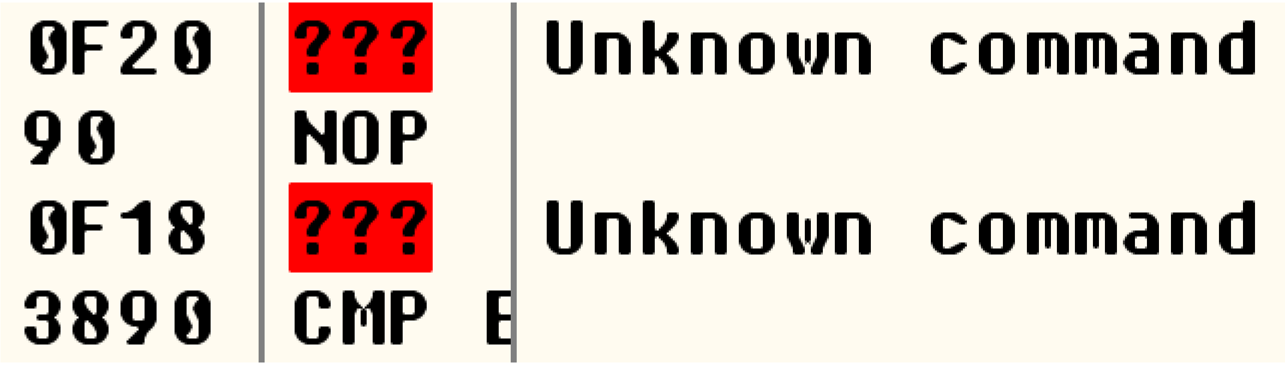


```
istruc IMAGE_OPTIONAL_HEADER32
        at IMAGE_OPTIONAL_HEADER32.Magic,
bits 32
EntryPoint:
    push message
    call [__imp__printf]
    jmp _2
        at IMAGE_OPTIONAL_HEADER32.AddressOfEntry
        at IMAGE_OPTIONAL_HEADER32.BaseOfCode, dd
_2:
    add esp, 1 * 4
    retn
        at IMAGE_OP
```

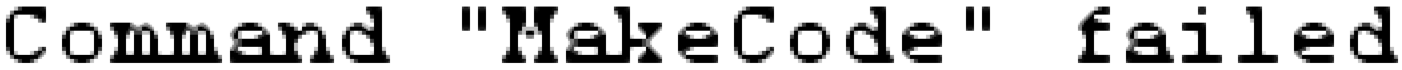
```
D>tiny
 * 268b universal tiny PE
```

the story behind this presentation

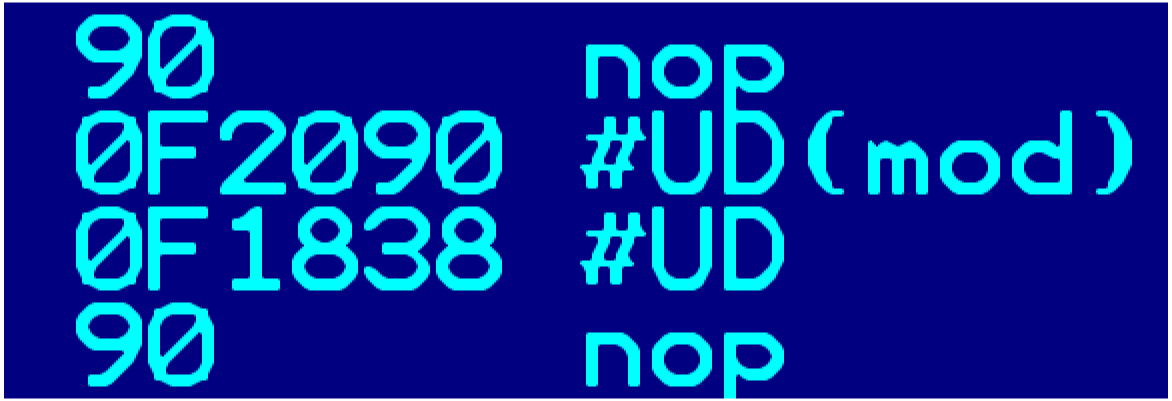




```
0F20  ???     Unknown command
90    NOP
0F18  ???     Unknown command
3890  CMP B
```

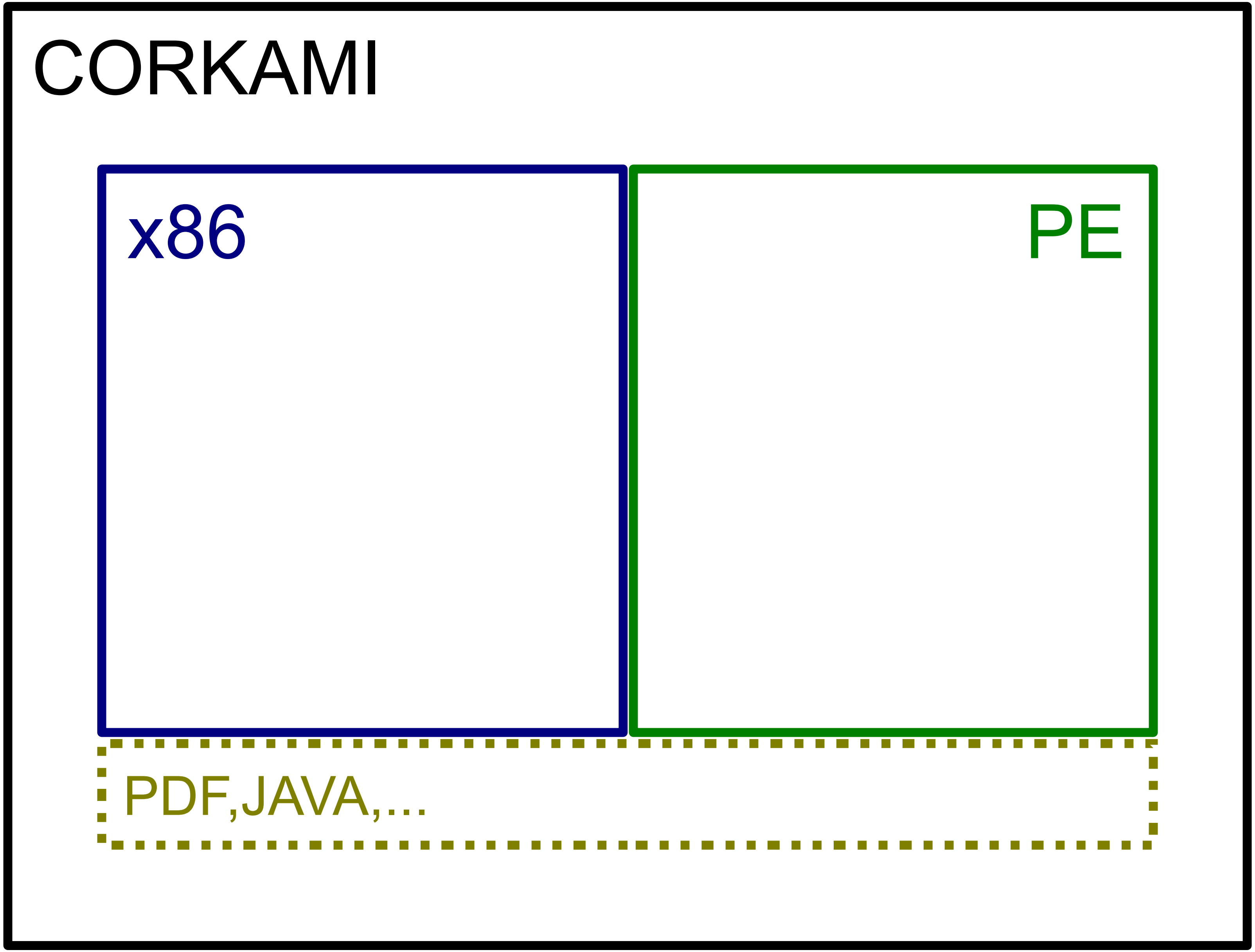


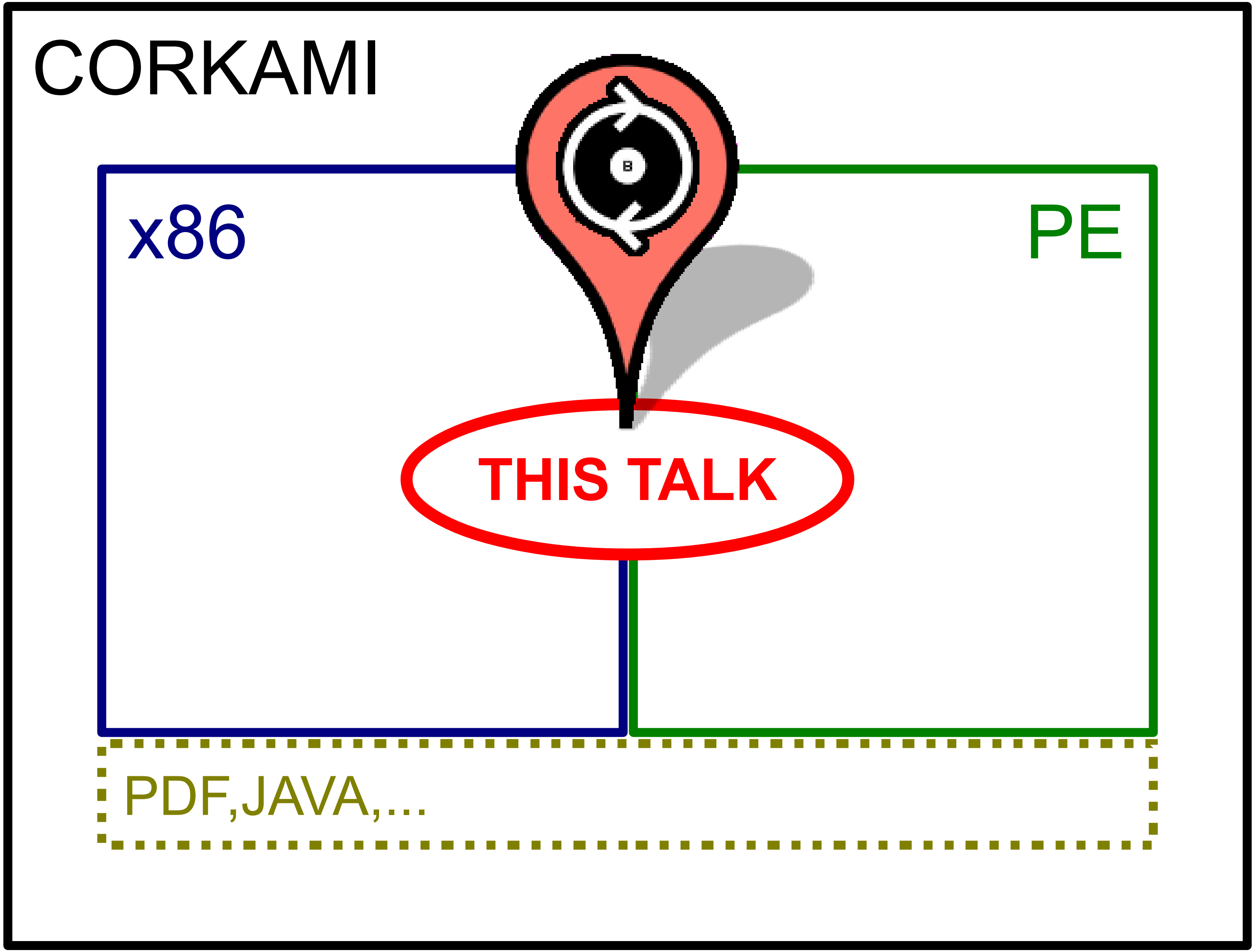
```
Command "MakeCode" failed
```



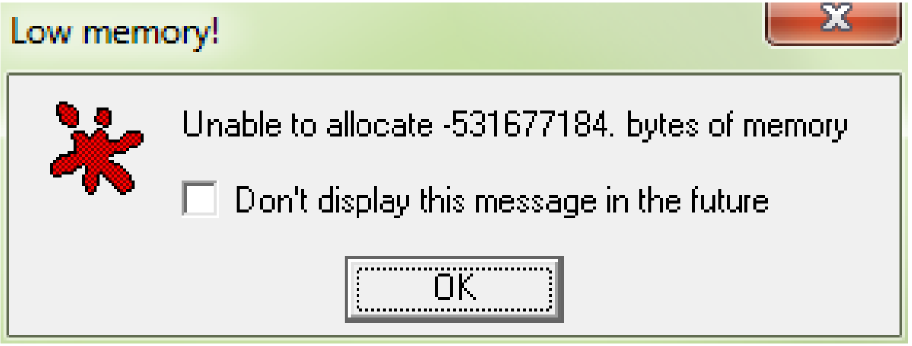
```
 90      nop
 0F2090  #UD(mod)
 0F1838  #UD
 90      nop
```

BACK TO THE BASICS


CORKAMI
x86
PE
PDF,JAVA,...

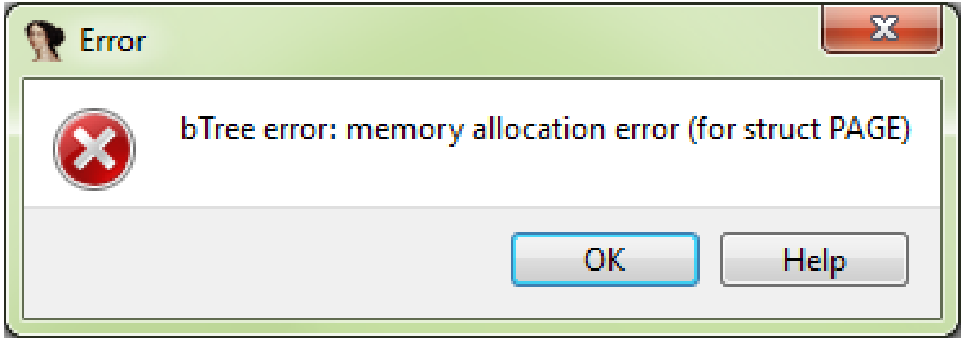

CORKAMI
x86
PE
THIS TALK
PDF,JAVA,...

“Achievement unlocked”

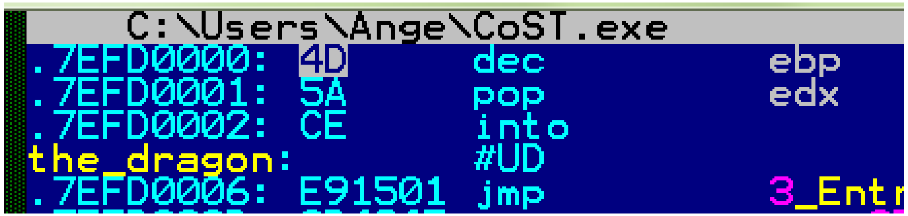

Low memory!

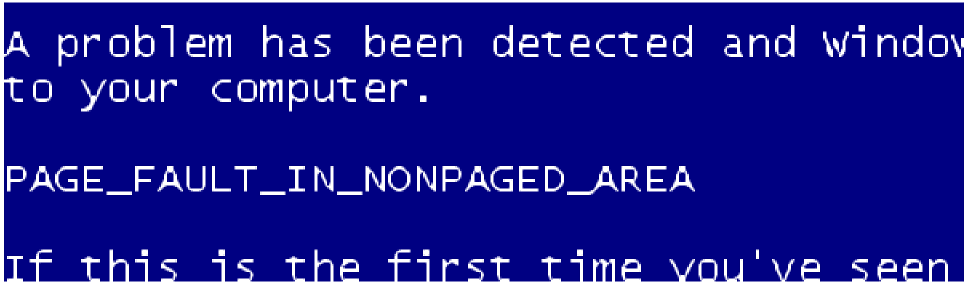
Unable to allocate -531677184. bytes of memory

□ Don't display this message in the future


Error

bTree error: memory allocation error (for struct PAGE)


```
    C:\Users\Ange\CoST.exe
.7EFD0000: 4D       dec          ebp
.7EFD0001: 5A       pop          edx
.7EFD0002: CE       into
the_dragon:         #UD
.7EFD0006: E91501   jmp          3_Entr
```


```
A problem has been detected and windov
to your computer.

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen
```

(Authors notified, and most bugs already fixed)

Agenda

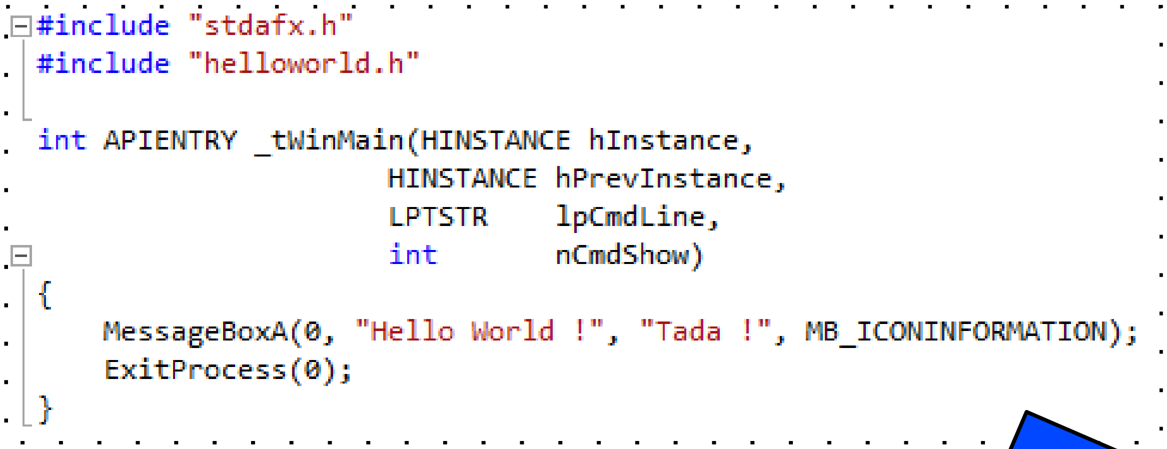
I. why does it matter?
    I. assembly
    II. undocumented assembly
II. x86 oddities

(technical stuff starts now)

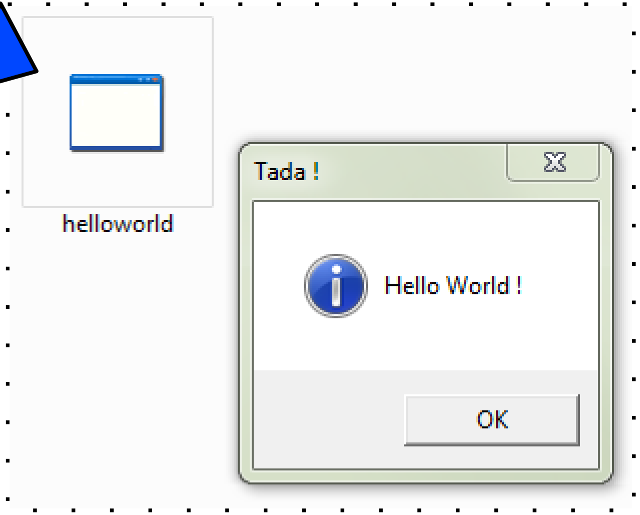
III. CoST
IV. a bit more of PE

assembly, in 8 slides

from C to binary

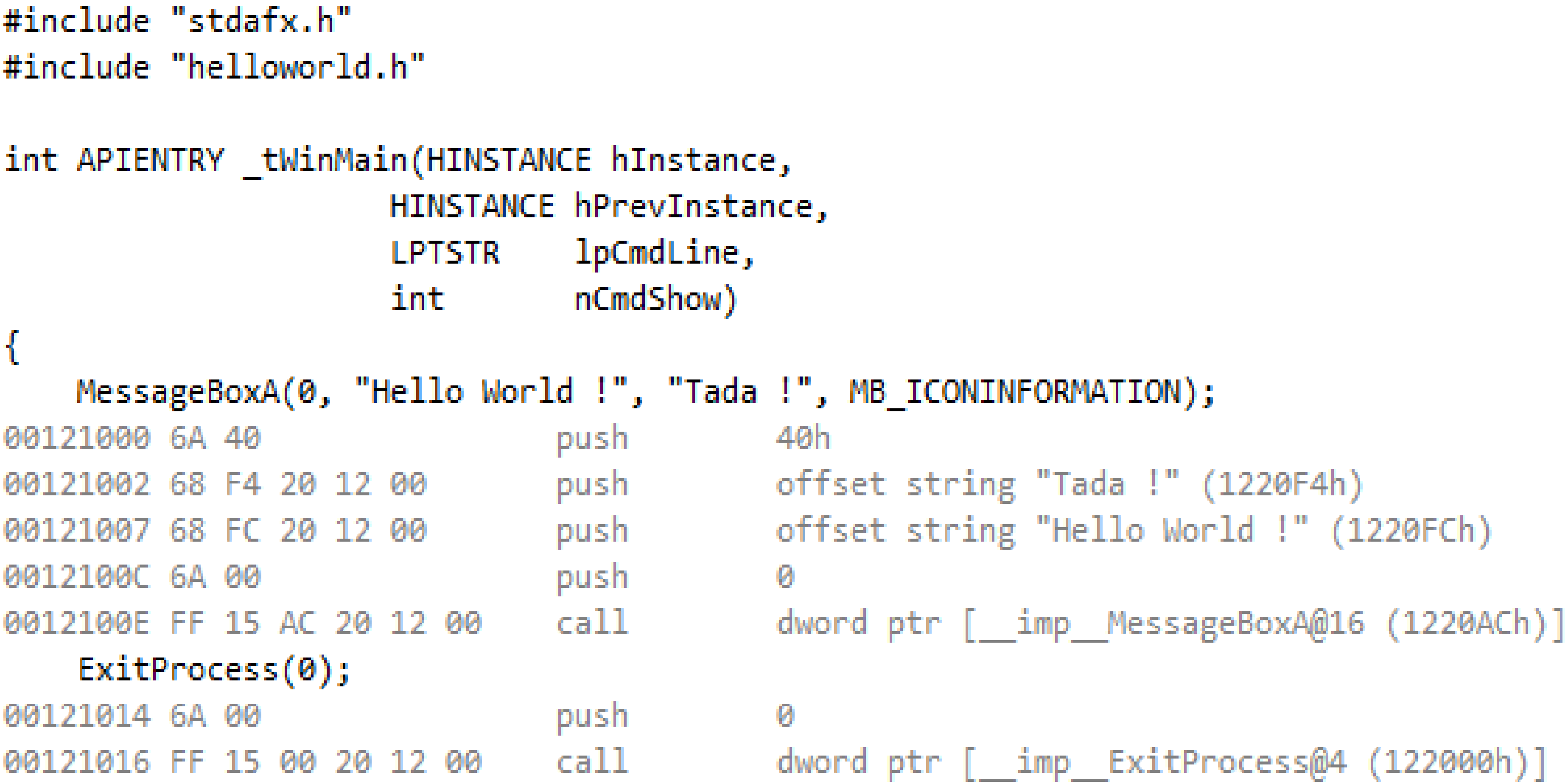


```
#include "stdafx.h"
#include "helloworld.h"

int APIENTRY _tWinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPTSTR    lpCmdLine,
                     int       nCmdShow)
{
    MessageBoxA(0, "Hello World !", "Tada !", MB_ICONINFORMATION);
    ExitProcess(0);
}
```



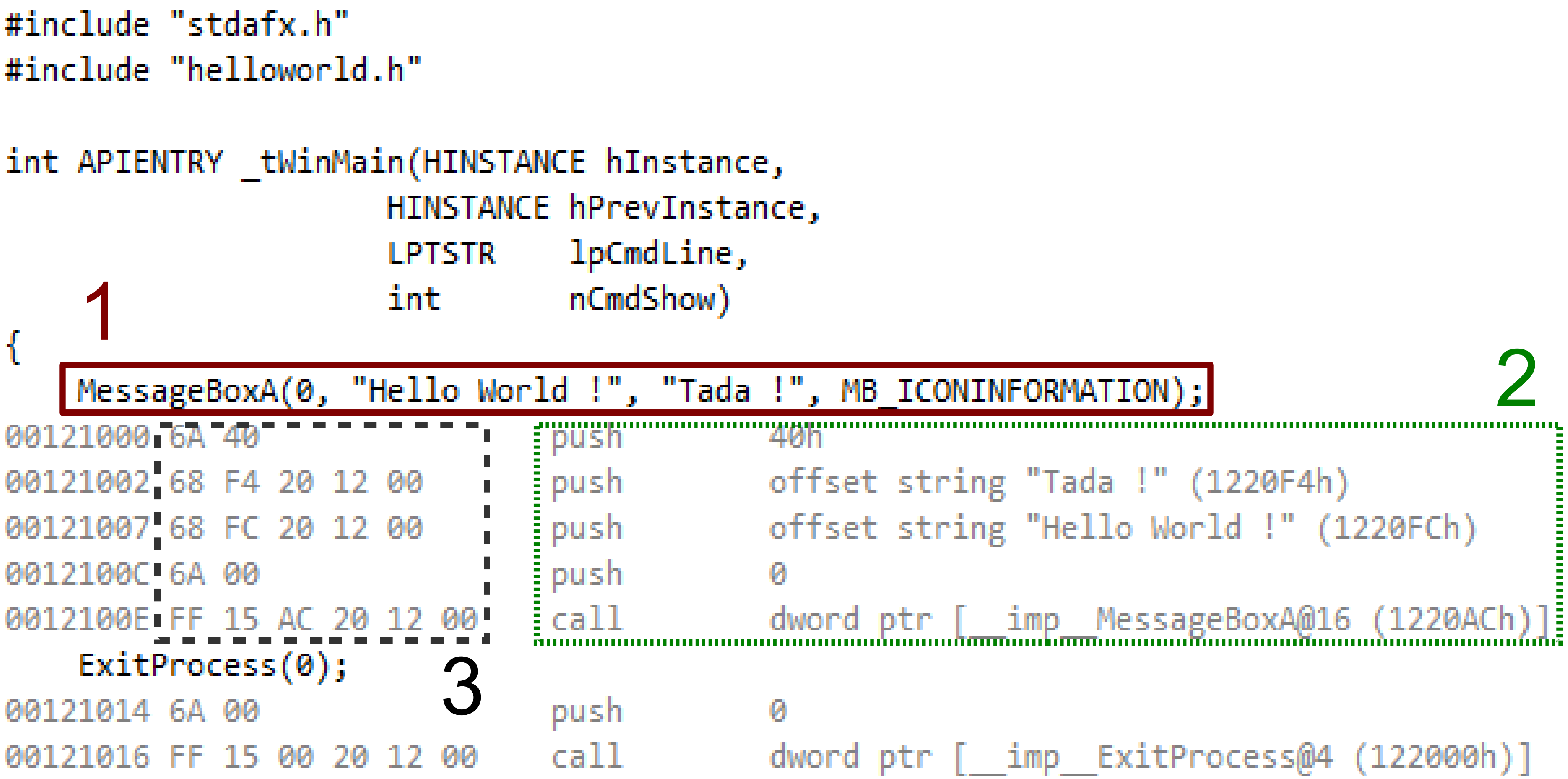
Hello World !

inside the binary



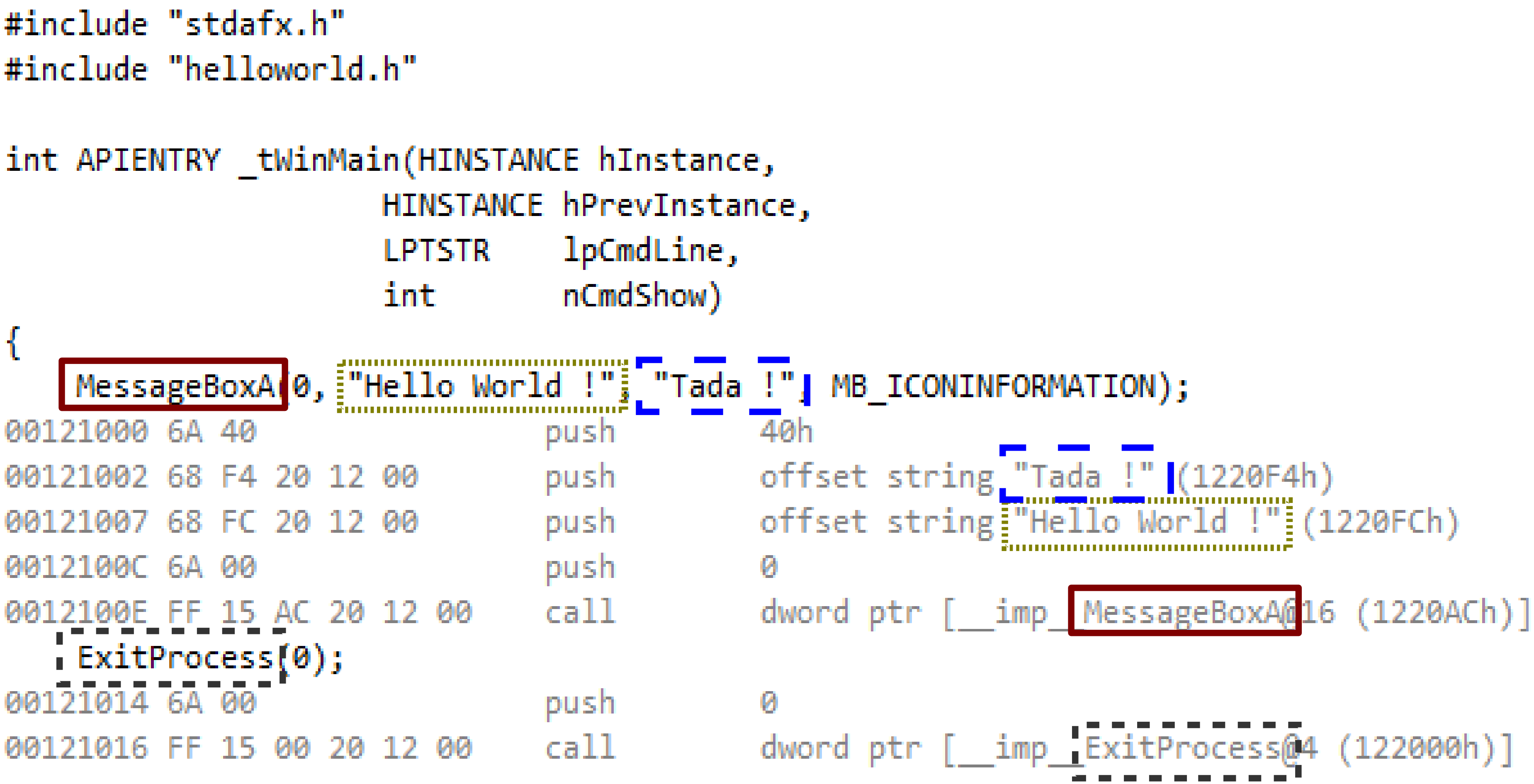
```
#include "stdafx.h"
#include "helloworld.h"

int APIENTRY _tWinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPTSTR    lpCmdLine,
                     int       nCmdShow)
{
    MessageBoxA(0, "Hello World !", "Tada !", MB_ICONINFORMATION);
00121000 6A 40                push        40h
00121002 68 F4 20 12 00       push        offset string "Tada !" (1220F4h)
00121007 68 FC 20 12 00       push        offset string "Hello World !" (1220FCh)
0012100C 6A 00                push        0
0012100E FF 15 AC 20 12 00    call        dword ptr [__imp__MessageBoxA@16 (1220ACh)]
    ExitProcess(0);
00121014 6A 00                push        0
00121016 FF 15 00 20 12 00    call        dword ptr [__imp__ExitProcess@4 (122000h)]
```

order



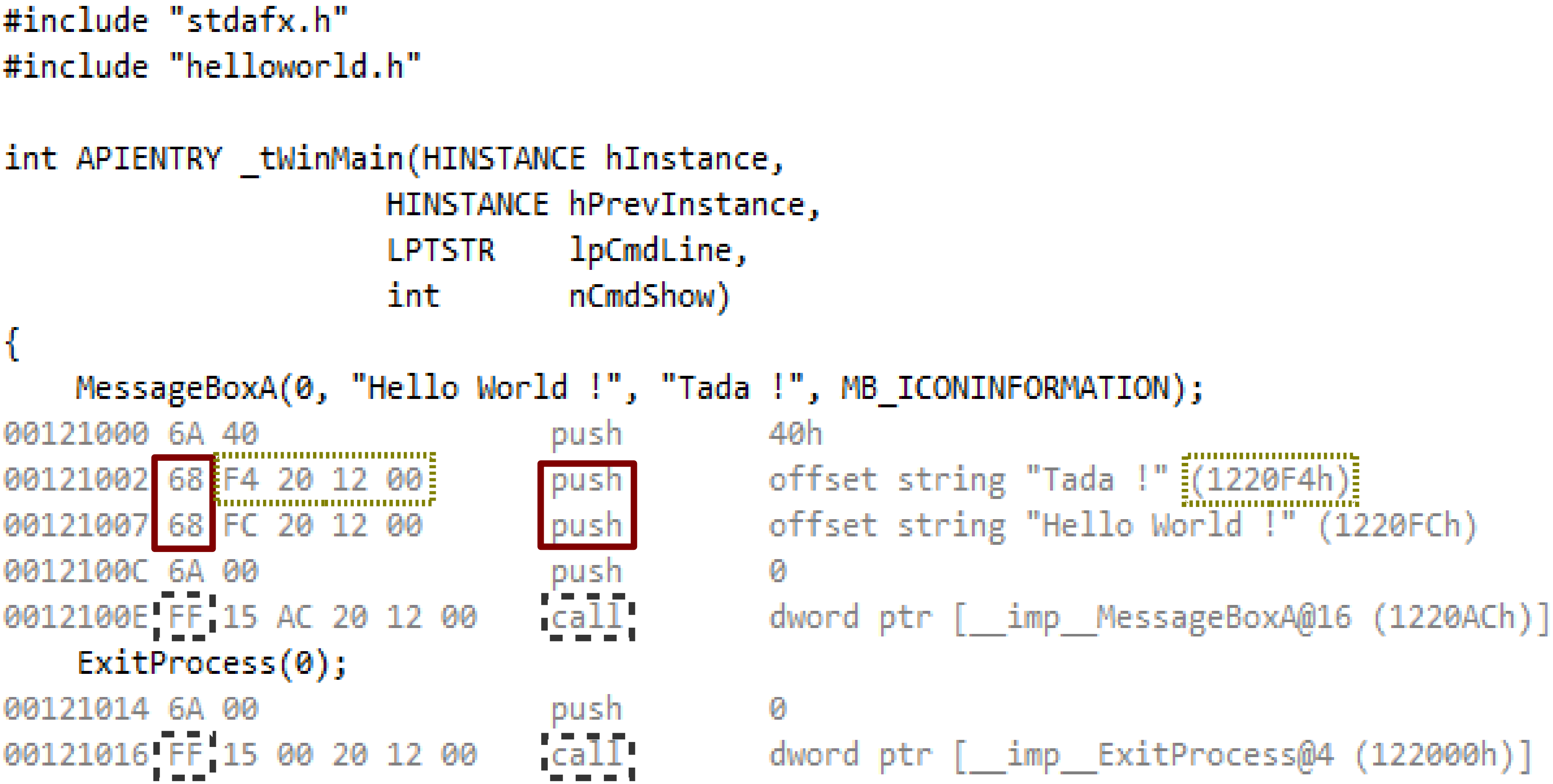
```
#include "stdafx.h"
#include "helloworld.h"

int APIENTRY _tWinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPTSTR    lpCmdLine,
                     int       nCmdShow)
{
    MessageBoxA(0, "Hello World !", "Tada !", MB_ICONINFORMATION);
00121000 6A 40                push        40h
00121002 68 F4 20 12 00       push        offset string "Tada !" (1220F4h)
00121007 68 FC 20 12 00       push        offset string "Hello World !" (1220FCh)
0012100C 6A 00                push        0
0012100E FF 15 AC 20 12 00    call        dword ptr [__imp__MessageBoxA@16 (1220ACh)]
    ExitProcess(0);
00121014 6A 00                push        0
00121016 FF 15 00 20 12 00    call        dword ptr [__imp__ExitProcess@4 (122000h)]
```

our code, 'translated'



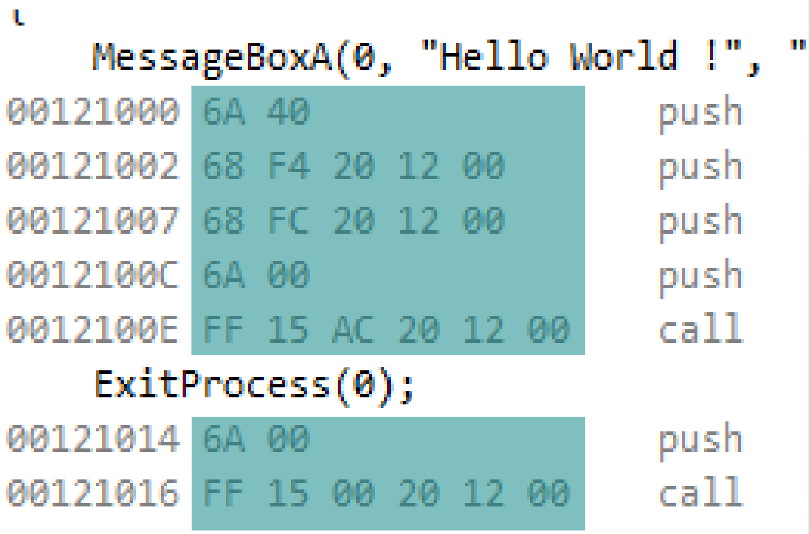
```
#include "stdafx.h"
#include "helloworld.h"

int APIENTRY _tWinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPTSTR    lpCmdLine,
                     int       nCmdShow)
{
    MessageBoxA(0, "Hello World !", "Tada !", MB_ICONINFORMATION);
00121000 6A 40                push        40h
00121002 68 F4 20 12 00       push        offset string "Tada !" (1220F4h)
00121007 68 FC 20 12 00       push        offset string "Hello World !" (1220FCh)
0012100C 6A 00                push        0
0012100E FF 15 AC 20 12 00    call        dword ptr [__imp__MessageBoxA@16 (1220ACh)]
    ExitProcess(0);
00121014 6A 00                push        0
00121016 FF 15 00 20 12 00    call        dword ptr [__imp__ExitProcess@4 (122000h)]
```

opcodes ⇔ assembly

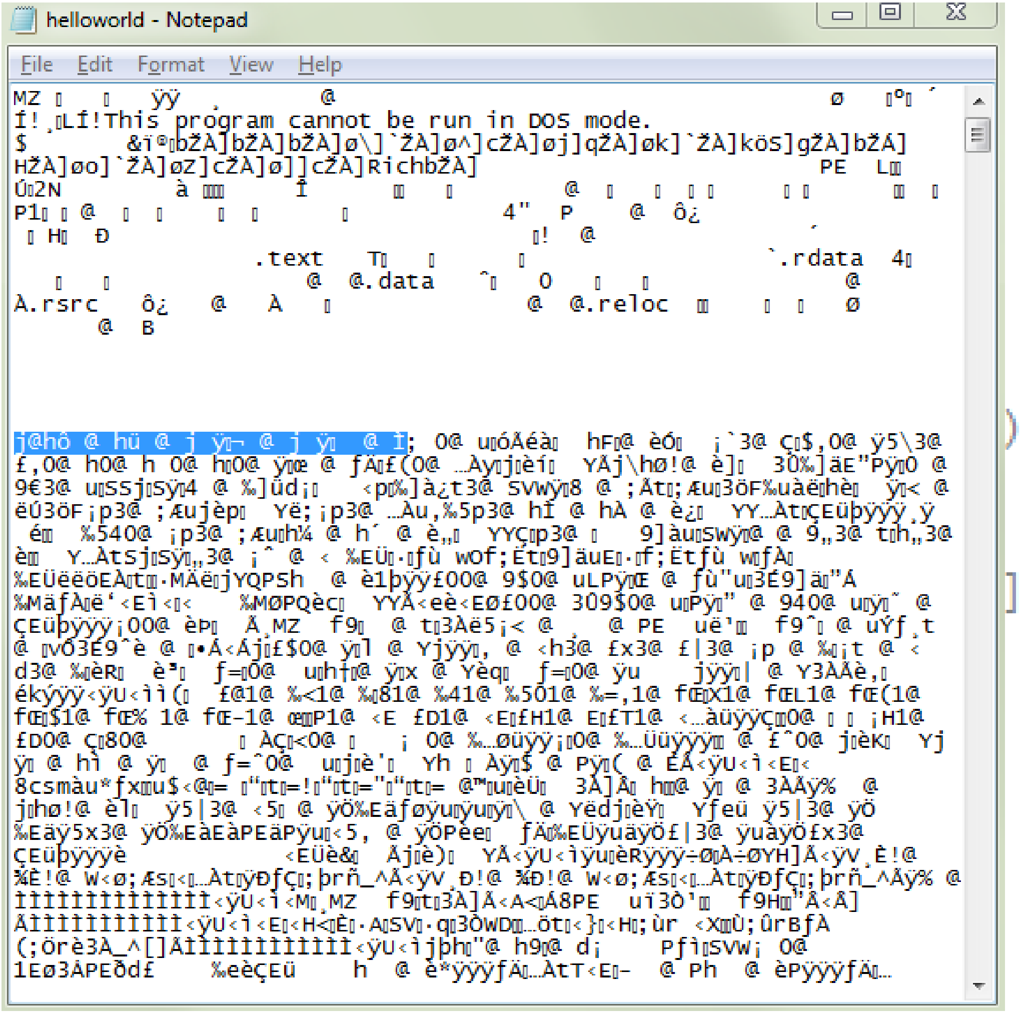


```
#include "stdafx.h"
#include "helloworld.h"

int APIENTRY _tWinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPTSTR    lpCmdLine,
                     int       nCmdShow)
{
    MessageBoxA(0, "Hello World !", "Tada !", MB_ICONINFORMATION);
00121000 6A 40                push        40h
00121002 68 F4 20 12 00       push        offset string "Tada !" (1220F4h)
00121007 68 FC 20 12 00       push        offset string "Hello World !" (1220FCh)
0012100C 6A 00                push        0
0012100E FF 15 AC 20 12 00    call        dword ptr [__imp__MessageBoxA@16 (1220ACh)]
    ExitProcess(0);
00121014 6A 00                push        0
00121016 FF 15 00 20 12 00    call        dword ptr [__imp__ExitProcess@4 (122000h)]
```

what's (only) in the binary

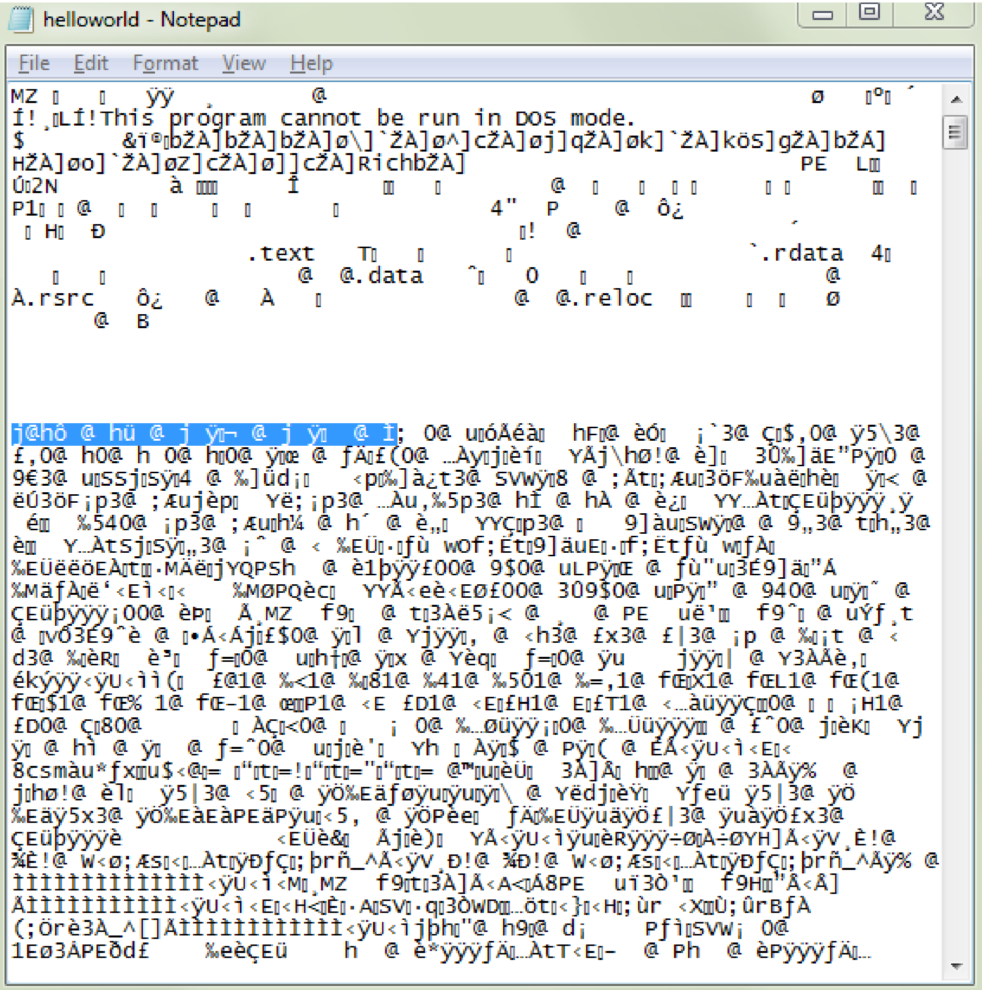


```
    MessageBoxA(0, "Hello World !", "
00121000  6A 40                push
00121002  68 F4 20 12 00       push
00121007  68 FC 20 12 00       push
0012100C  6A 00                push
0012100E  FF 15 AC 20 12 00    call
    ExitProcess(0);
00121014  6A 00                push
00121016  FF 15 00 20 12 00    call
```

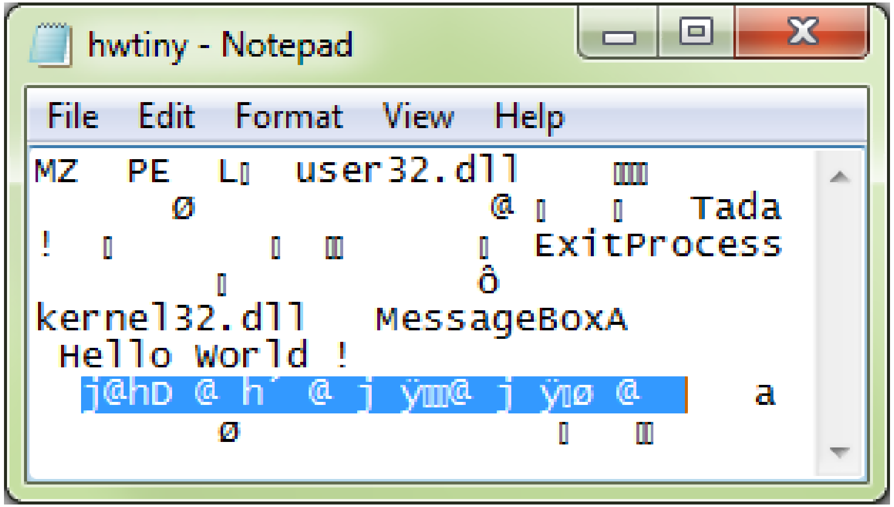


```
MZ ▯   ▯   ÿÿ  ¸        @                                  ø    ▯º▯ ´
Í!¸▯LÍ!This program cannot be run in DOS mode.
$        &ï®▯bŽÀ]bŽÀ]bŽÀ]ø\]`ŽÀ]ø^]cŽÀ]øj]qŽÀ]øk]`ŽÀ]kös]gŽÀ]bŽÀ]
HŽÀ]øo]`ŽÀ]øz]cŽÀ]ø]]cŽÀ]RichbŽÀ]                           PE  L▯
Ú▯2N          à ▯▯    Î      ▯   ▯        @  ▯   ▯  ▯ ▯    ▯ ▯     ▯  ▯
P1▯ ▯ @  ▯  ▯    ▯  ▯     ▯           4"  P    @  ô¿
 ▯ H▯  Ð                                 ▯!  @                ´
                 .text   T▯   ▯       ▯                    `.rdata  4▯
   ▯   ▯               @  @.data   ˆ▯   O   ▯   ▯                @
À.rsrc   ô¿   @   À   ▯              @  @.reloc  ▯    ▯  ▯   ø
      @  B



j@hô @ hü @ j ÿ▯¬ @ j ÿ▯  @ Ì; 0@ u▯óÃéà▯  hF▯@ èÓ▯  ¡`3@ Ç▯$,0@ ÿ5\3@
£,0@ h0@ h 0@ h▯0@ ÿ▯œ @ ƒÄ▯£(0@ …Ày▯j▯èí▯  YÃj\hø!@ è]▯  3Ò‰]äE"Pÿ▯0 @
9€3@ u▯SSj▯Sÿ▯4 @ ‰]üd¡▯   ‹p▯‰]à¿t3@ SVWÿ▯8 @ ;Át▯;Æu▯3öF‰uàë▯hè▯  ÿ▯< @
ëÚ3öF¡p3@ ;Æujèp▯  Yë;¡p3@ …Àu,‰5p3@ hÌ @ hÀ @ è¿▯  YY…Àt▯ÇEüþÿÿÿ¸ÿ
 é▯  ‰540@ ¡p3@ ;Æu▯h¼ @ h´ @ è„▯  YYÇ▯p3@ ▯   9]àu▯SWÿ▯@ @ 9„3@ t▯h„3@
è▯  Y…ÀtSj▯Sÿ▯„3@ ¡ˆ @ ‹ ‰EÜ▯·▯fù wOf;Ët▯9]äuE▯·▯f;Ëtfù w▯fÀ▯
‰EÜëëöEÀ▯t▯·MÄë▯jYQPSh  @ è1þÿÿ£00@ 9$0@ uLPÿ▯Œ @ fù"u▯3É9]ä▯"Á
‰MäƒÀ▯ë'‹Eì‹▯‹   ‰MØPQèc▯  YYÃ‹eè‹EØ£00@ 3Ò9$0@ u▯Pÿ▯" @ 940@ u▯ÿ▯˜ @
ÇEüþÿÿÿ¡00@ èP▯  Ã¸MZ  f9▯  @ t▯3Àë5¡< @  ¸   @ PE  uë'▯  f9ˆ▯ @ uÝƒ¸t
@ ▯vÕ3É9ˆè @ ▯•Á‹Áj▯£$0@ ÿ▯l @ Yjÿÿ▯, @ ‹h3@ £x3@ £|3@ ¡p @ ‰▯¡t @ ‹
d3@ ‰▯èR▯  è³▯  ƒ=▯0@  u▯h†▯@ ÿ▯x @ Yèq▯  ƒ=▯0@ ÿu    jÿÿ▯| @ Y3ÀÃè,▯
ékýÿÿ‹ÿU‹ìì(▯  £@1@ ‰<1@ ‰▯81@ ‰41@ ‰501@ ‰=,1@ fŒ▯X1@ fŒL1@ fŒ(1@
fŒ▯$1@ fŒ% 1@ fŒ-1@ œ▯▯P1@ ‹E £D1@ ‹E▯£H1@ E▯£T1@ ‹…àüÿÿÇ▯▯0@ ▯ ▯ ¡H1@
£D0@ Ç▯80@       ▯ ÀÇ▯<0@ ▯    ¡ 0@ ‰…Øüÿÿ¡▯0@ ‰…Üüÿÿÿ▯ @ £ˆ0@ j▯èK▯  Yj
ÿ▯ @ hì @ ÿ▯  @ ƒ=ˆ0@  u▯j▯è'▯  Yh ▯ Àÿ▯$ @ Pÿ▯( @ ÉÃ‹ÿU‹ì‹E▯‹
8csmàu*ƒx▯▯u$‹@▯= ▯"▯t▯=!▯"▯t▯="▯"▯t▯= @™▯u▯èÜ▯  3À]Â▯ h▯▯@ ÿ▯ @ 3ÀÃÿ%  @
j▯hø!@ èl▯  ÿ5|3@ ‹5▯ @ ÿÖ‰EäƒøÿuÞÿu▯ÿ▯\ @ YëdjèŸ▯  Yƒeü ÿ5|3@ ÿÖ
‰Eäÿ5x3@ ÿÖ‰EàEàPEäPÿu▯‹5, @ ÿÖPèe▯  ƒÄ▯‰EÜÿuäÿÖ£|3@ ÿuàÿÖ£x3@
ÇEüþÿÿÿè             ‹EÜè&▯  Ãj▯è)▯  YÃ‹ÿU‹ìÿu▯èRÿÿÿ÷Ø▯À÷ØYH]Ã‹ÿV¸È!@
¾È!@ W‹ø;Æs▯‹▯…Àt▯ÿÐƒÇ▯;þrñ_^Ã‹ÿV¸Ð!@ ¾Ð!@ W‹ø;Æs▯‹▯…Àt▯ÿÐƒÇ▯;þrñ_^Ãÿ% @
ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ‹ÿU‹ì‹M▯¸MZ  f9▯t▯3À]Ã‹A<▯Á8PE  uï3Ò'▯  f9H▯"Â‹Â]
ÃÌÌÌÌÌÌÌÌÌÌÌÌ‹ÿU‹ì‹E▯‹H<▯È▯·A▯SV▯·q▯3ÒWD▯…öt▯‹}▯‹H▯;ùr ‹X▯Ù;ûrBƒÂ
(;Örè3À_^[]ÃÌÌÌÌÌÌÌÌÌÌÌÌÌ‹ÿU‹ìjþh▯"@ h9▯@ d¡    Pƒì▯SVW¡ 0@
1Eø3ÅPEÕd£     ‰eèÇEü    h  @ è*ÿÿÿƒÄ▯…ÀtT‹E▯-  @ Ph  @ èPÿÿÿƒÄ▯…
```

execution ⇔ CPU + opcodes



```
MZ ▯   ▯   ÿÿ  ¸        @                                    ø   ▯º▯ ´
º! ¸▯Lº!This program cannot be run in DOS mode.
$       &ï®▯bŽÀ]bŽÀ]bŽÀ]ø\]`ŽÀ]ø^]cŽÀ]øj]qŽÀ]øk]`ŽÀ]kös]gŽÀ]bŽÀ]
HŽÀ]øo]`ŽÀ]øz]cŽÀ]ø]]cŽÀ]RichbŽÀ]                               PE  L▯
Ú▯2N            à ▯▯      º            ▯    ▯           @  ▯   ▯  ▯ ▯      ▯ ▯       ▯  ▯
P1▯ ▯ @  ▯   ▯      ▯   ▯         ▯                4"   P      @   ô¿
 ▯ H▯   Ð                                         ▯!    @
                     .text    T▯     ▯           ▯                   `.rdata   4▯
   ▯    ▯                  @  @.data    ˆ▯     0     ▯    ▯                   @
À.rsrc    ô¿     @     À    ▯                  @   @.reloc   ▯     ▯  ▯      ø
      @    B



j@hö @ hü @ j ÿ▯¬ @ j ÿ▯  @ ì▯; 0@ u▯óÅéà▯  hF▯@ èó▯   ¡`3@ Ç▯$,0@ ÿ5\3@
£,0@ h0@ h 0@ h▯0@ ÿ▯œ @ ƒÄ▯£(0@ …Ày▯jèí▯  YÃj\hø!@ è]▯  3Û‰]äE"Pÿ▯0 @
9€3@ u▯SSj▯Sÿ▯4 @ ‰]üd¡▯     ‹p▯‰]à¿t3@ SVWÿ▯8 @ ;Æt▯;Æu▯3öF‰uàë▯hè▯   ÿ▯< @
ëÜ3öF¡p3@ ;Æujè▯    Yë;¡p3@ …Àu,‰5p3@ hì @ hÀ @ è¿▯   YY…Àt▯ÇEüþÿÿÿ¸ÿ
 é▯  ‰540@ ¡p3@ ;Æu▯h¼ @ h´ @ è„▯   YYÇ▯p3@ ▯     9]àu▯SWÿ▯@ @ 9„3@ t▯h„3@
è▯   Y…ÀtSj▯Sÿ▯„3@ ¡ˆ @ ‹ %EÜ▯·▯fù wOf;Ët▯9]äuE▯·▯f;Ët▯fù w▯fÀ▯
%EÜëëöEÀ▯t▯·MÄë▯jYQPSh  @ è1þÿÿ£00@ 9$0@ uLPÿ▯Œ @ fù"u▯3É9]ä▯"À
%MäfÀë'‹Eì‹▯‹   %MøPQèç▯  YYÃ‹eè‹Eø£00@ 309$0@ u▯Pÿ▯" @ 940@ u▯ÿ▯ˆ @
ÇEüþÿÿÿ¡00@ èÞ▯   À MZ  f9▯   @ t▯3Àë5¡< @     @ PE   uë'▯  f9ˆ▯ @ uÝƒ¸t
@ ▯vÓ3É9ˆè @ ▯•À‹Áj▯£$0@ ÿ▯l @ Yjÿÿ▯, @ ‹h3@ £x3@ £|3@ ¡p @ ‰▯¡t @ ‹
d3@ ‰▯èR▯   è"▯   f=▯0@   u▯h†▯@ ÿ▯x @ Yèq▯   f=▯0@ ÿu      jÿÿ▯| @ Y3ÀÃè,▯
ékÿÿÿ‹ÿU‹ìì(▯   £@1@ ‰<1@ ‰▯81@ ‰41@ ‰501@ ‰=,1@ fŒ▯X1@ fŒL1@ fŒ(1@
fŒ▯$1@ fŒ% 1@ fŒ-1@ œ▯P1@ ‹E £D1@ ‹E▯£H1@ E▯£T1@ ‹…àüÿÿÇ▯0@ ▯ ▯  ¡H1@
£D0@ Ç▯80@        ▯ ÀÇ▯<0@ ▯     ¡  0@ ‰…Øüÿÿ¡▯0@ ‰…Üüÿÿÿ▯ @ £ˆ0@ jèK▯   Yj
ÿ▯ @ hì @ ÿ▯   @ f=ˆ0@   u▯jè'▯   Yh ▯ Àÿ▯$ @ Pÿ▯( @ ÉÃ‹ÿU‹ì‹E▯‹
8csmàu*ƒxu▯$‹@▯= ▯"“▯t▯=!▯"“▯t▯="▯"“▯t▯= @™▯u▯èÜ▯   3À]Â▯ hœ@ ÿ▯ @ 3ÀÃÿ%  @
j▯hø!@ èl▯   ÿ5|3@ ‹5▯ @ ÿÖ‰Eäƒøÿu▯ÿu▯ÿ▯\ @ YëdjèY▯   Yƒeü ÿ5|3@ ÿÖ
‰Eäÿ5x3@ ÿÖ‰EàÈàPEäPÿu▯‹5, @ ÿÖPèe▯   ƒÄ‰EÜÿuäÿÖ£|3@ ÿuàÿÖ£x3@
ÇEüþÿÿÿè           ‹EÜè&▯   Ãj▯è)▯   YÃ‹ÿU‹ìÿuèRÿÿÿ÷Ø▯À÷ØYH]Ã‹ÿV¸È!@
¾È!@ W‹ø;Æs▯‹…At▯ÿÐƒÇ▯;þrñ_^Ã‹ÿV¸Ð!@ ¾Ð!@ W‹ø;Æs▯‹…At▯ÿÐƒÇ▯;þrñ_^Ãÿ% @
ÌÌÌÌÌÌÌÌÌÌÌÌÌÌ‹ÿU‹ì‹M▯¸MZ  f9▯t▯3À]Ã‹A<▯Á8PE   uï3Ò¹▯   f9Hˆ”Â‹Â]
ÃÌÌÌÌÌÌÌÌÌÌÌ‹ÿU‹ì‹E▯‹H<▯È▯·ASV▯·q▯3ÒWD▯…öt▯‹}▯‹H▯;ùr  ‹X▯Ù;ûrBƒÂ
(;Òrè3À_^[]ÃÌÌÌÌÌÌÌÌÌÌÌÌ‹ÿU‹ìjþhˆ"@ h9▯@ d¡     Pƒì▯SVW¡  0@
1Eø3ÅPEðd£      ‰eèÇEü     h   @ è*ÿÿÿƒÄ▯…ÀtT‹E▯-   @ Ph   @ èPÿÿÿƒÄ▯…
```



```
MZ   PE   L▯  user32.dll     ▯▯▯
        ø                     @ ▯    ▯    Tada
!  ▯          ▯  ▯▯         ▯  ExitProcess
           ▯                 ô
kernel32.dll    MessageBoxA
 Hello World !
  j@hD @ h´ @ j ÿ▯@ j ÿ▯ø @      a
        ø                    ▯    ▯▯
```

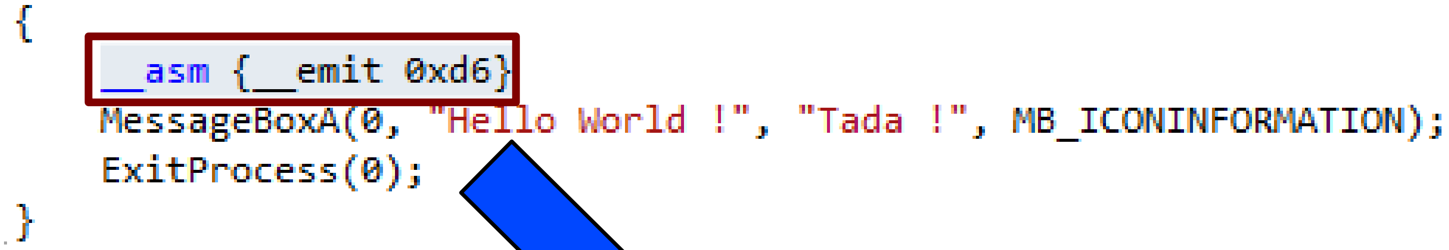
opcodes

- generated by compilers, tools,...
  - or written by hand
- executed directly by the CPU
- the only code information, in a standard binary
  - what 'we' read
    - **after** disassembly

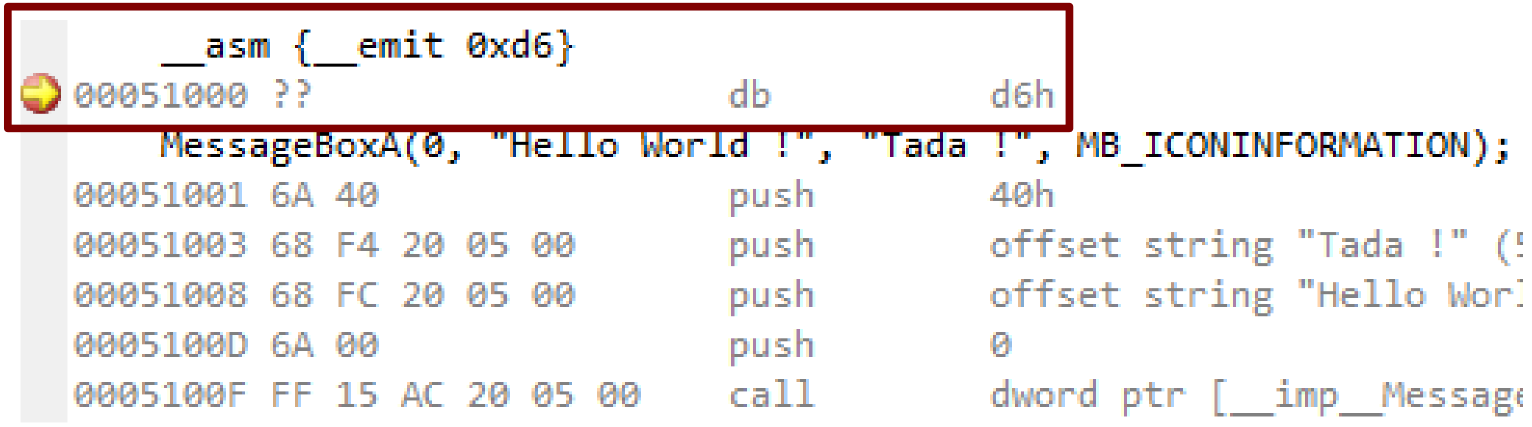
- disassembly is only for humans
  - no text code in the final binary

let's mess a bit now...

let's insert 'something'

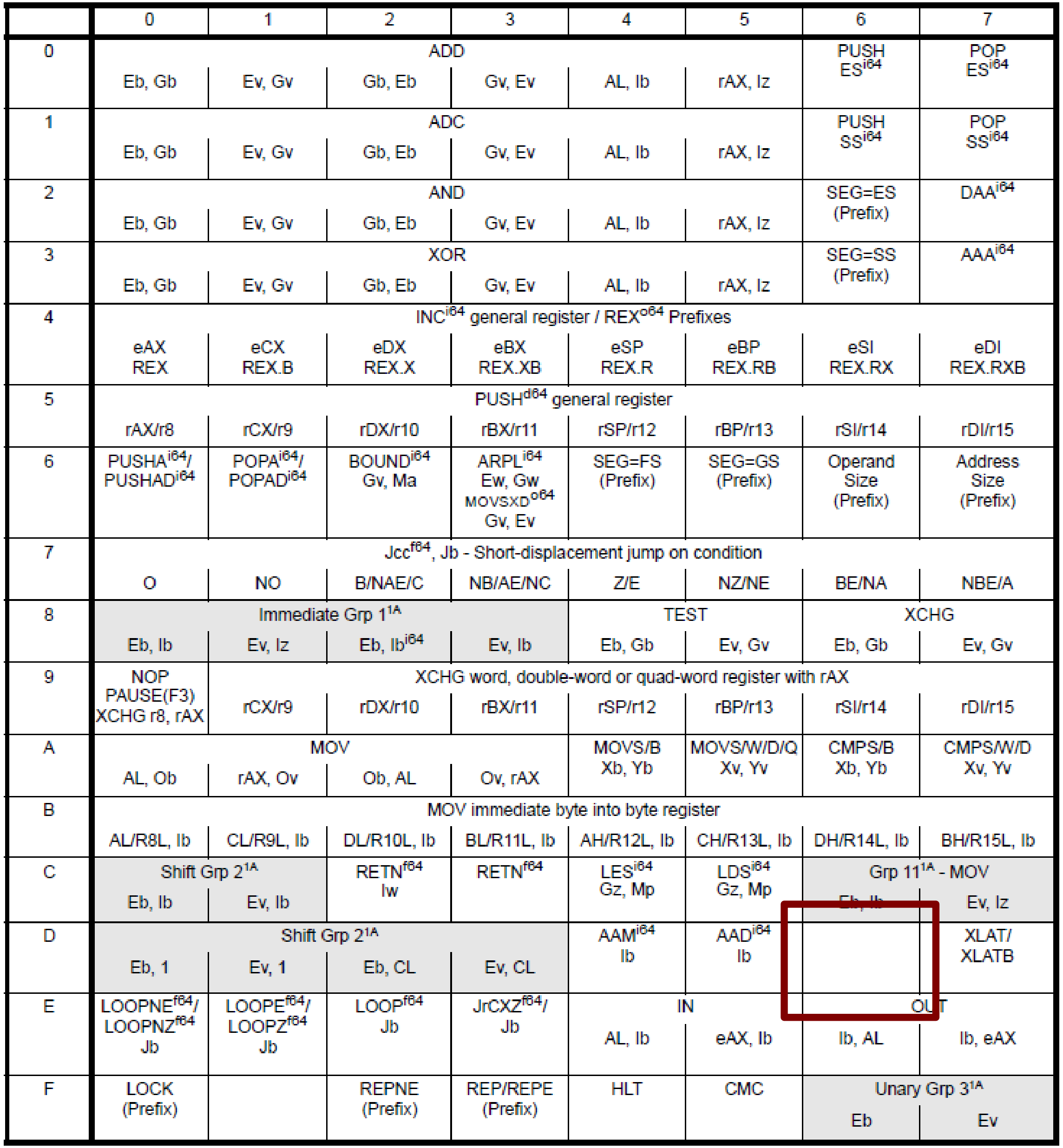


```
{
    __asm {__emit 0xd6}
    MessageBoxA(0, "Hello World !", "Tada !", MB_ICONINFORMATION);
    ExitProcess(0);
}
```



```
    __asm {__emit 0xd6}
00051000 ??                  db          d6h
    MessageBoxA(0, "Hello World !", "Tada !", MB_ICONINFORMATION);
00051001 6A 40               push        40h
00051003 68 F4 20 05 00      push        offset string "Tada !" (5
00051008 68 FC 20 05 00      push        offset string "Hello Worl
0005100D 6A 00               push        0
0005100F FF 15 AC 20 05 00   call        dword ptr [__imp__Message
```

Table A-2. One-byte Opcode Map: (00H — F7H) *



	0	1	2	3	4	5	6	7
0	ADD						PUSH ES^{i64}	POP ES^{i64}
	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz		
1	ADC						PUSH SS^{i64}	POP SS^{i64}
	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz		
2	AND						SEG=ES (Prefix)	DAA^{i64}
	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz		
3	XOR						SEG=SS (Prefix)	AAA^{i64}
	Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz		
4	INC^{i64} general register / REX^{o64} Prefixes							
	eAX REX	eCX REX.B	eDX REX.X	eBX REX.XB	eSP REX.R	eBP REX.RB	eSI REX.RX	eDI REX.RXB
5	PUSH^{d64} general register							
	rAX/r8	rCX/r9	rDX/r10	rBX/r11	rSP/r12	rBP/r13	rSI/r14	rDI/r15
6	PUSHA^{i64}/ PUSHAD^{i64}	POPA^{i64}/ POPAD^{i64}	BOUND^{i64} Gv, Ma	ARPL^{i64} Ew, Gw MOVSXD^{o64} Gv, Ev	SEG=FS (Prefix)	SEG=GS (Prefix)	Operand Size (Prefix)	Address Size (Prefix)
7	Jcc^{f64}, Jb - Short-displacement jump on condition							
	O	NO	B/NAE/C	NB/AE/NC	Z/E	NZ/NE	BE/NA	NBE/A
8	Immediate Grp 1^{1A}				TEST		XCHG	
	Eb, Ib	Ev, Iz	Eb, Ib^{i64}	Ev, Ib	Eb, Gb	Ev, Gv	Eb, Gb	Ev, Gv
9	NOP PAUSE(F3) XCHG r8, rAX	XCHG word, double-word or quad-word register with rAX						
		rCX/r9	rDX/r10	rBX/r11	rSP/r12	rBP/r13	rSI/r14	rDI/r15
A	MOV				MOVS/B Xb, Yb	MOVS/W/D/Q Xv, Yv	CMPS/B Xb, Yb	CMPS/W/D Xv, Yv
	AL, Ob	rAX, Ov	Ob, AL	Ov, rAX				
B	MOV immediate byte into byte register							
	AL/R8L, Ib	CL/R9L, Ib	DL/R10L, Ib	BL/R11L, Ib	AH/R12L, Ib	CH/R13L, Ib	DH/R14L, Ib	BH/R15L, Ib
C	Shift Grp 2^{1A}		RETN^{f64} Iw	RETN^{f64}	LES^{i64} Gz, Mp	LDS^{i64} Gz, Mp	Grp 11^{1A} - MOV	
	Eb, Ib	Ev, Ib					Eb, Ib	Ev, Iz
D	Shift Grp 2^{1A}				AAM^{i64} Ib	AAD^{i64} Ib		XLAT/ XLATB
	Eb, 1	Ev, 1	Eb, CL	Ev, CL				
E	LOOPNE^{f64}/ LOOPNZ^{f64} Jb	LOOPE^{f64}/ LOOPZ^{f64} Jb	LOOP^{f64} Jb	JrCXZ^{f64}/ Jb	IN		OUT	
					AL, Ib	eAX, Ib	Ib, AL	Ib, eAX
F	LOCK (Prefix)		REPNE (Prefix)	REP/REPE (Prefix)	HLT	CMC	Unary Grp 3^{1A}	
							Eb	Ev

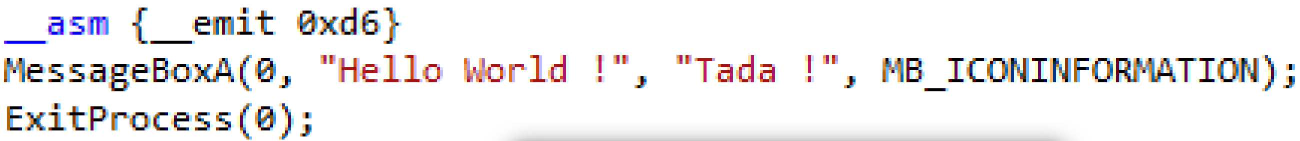
what did we do?

- Inserting an unrecognized byte
  - directly in the binary
    - to be executed by the CPU
  - not even documented, nor identified!

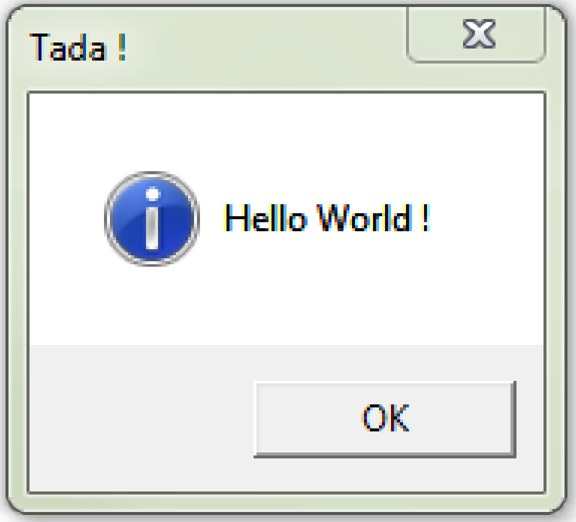
“kids, don't try this at home!”

the CPU doesn't care

- **it** knows
  - and does its own stuff



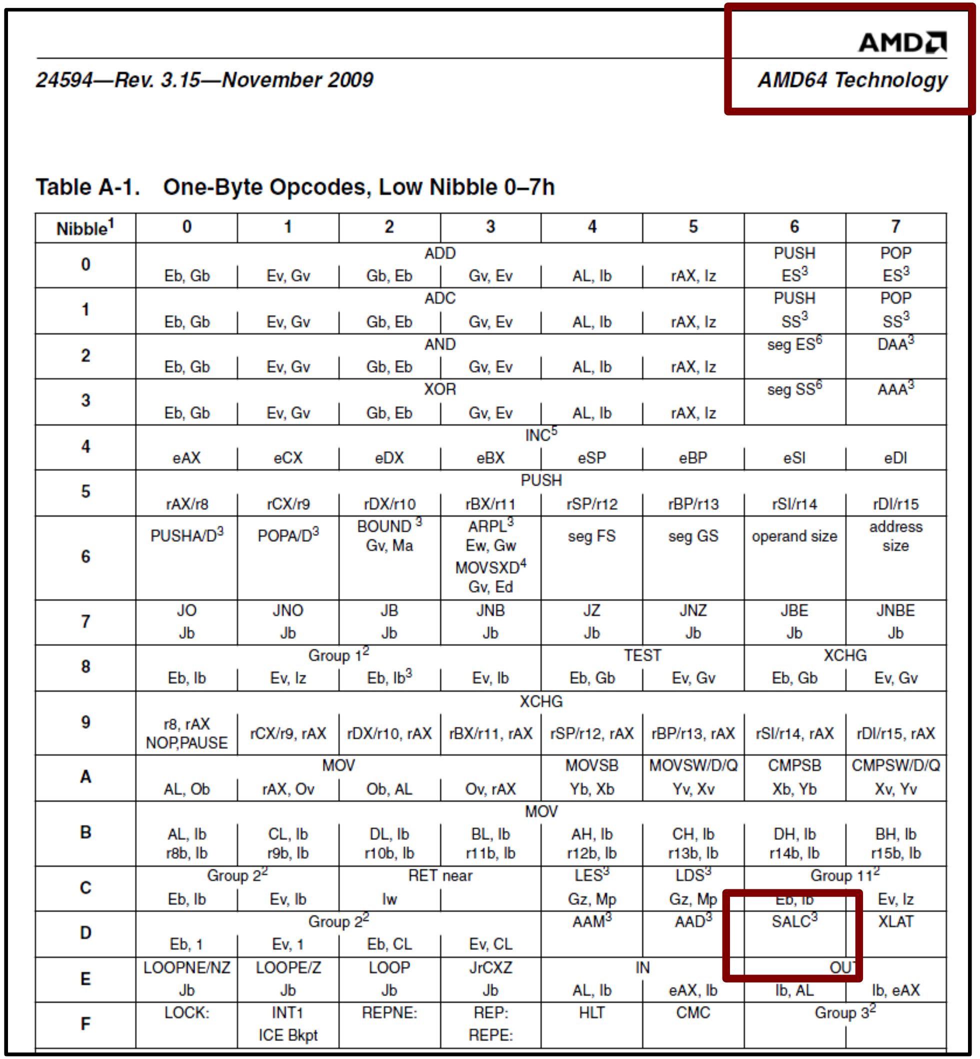
```
__asm {__emit 0xd6}
MessageBoxA(0, "Hello World !", "Tada !", MB_ICONINFORMATION);
ExitProcess(0);
```



Hello World !

what happened ?

- D6 = S[ET]ALC
  - Set AL on Carry
    - AL = CF ? -1 : 0
- trivial
- but not documented
  - unreliable, or shameful ?



AMD

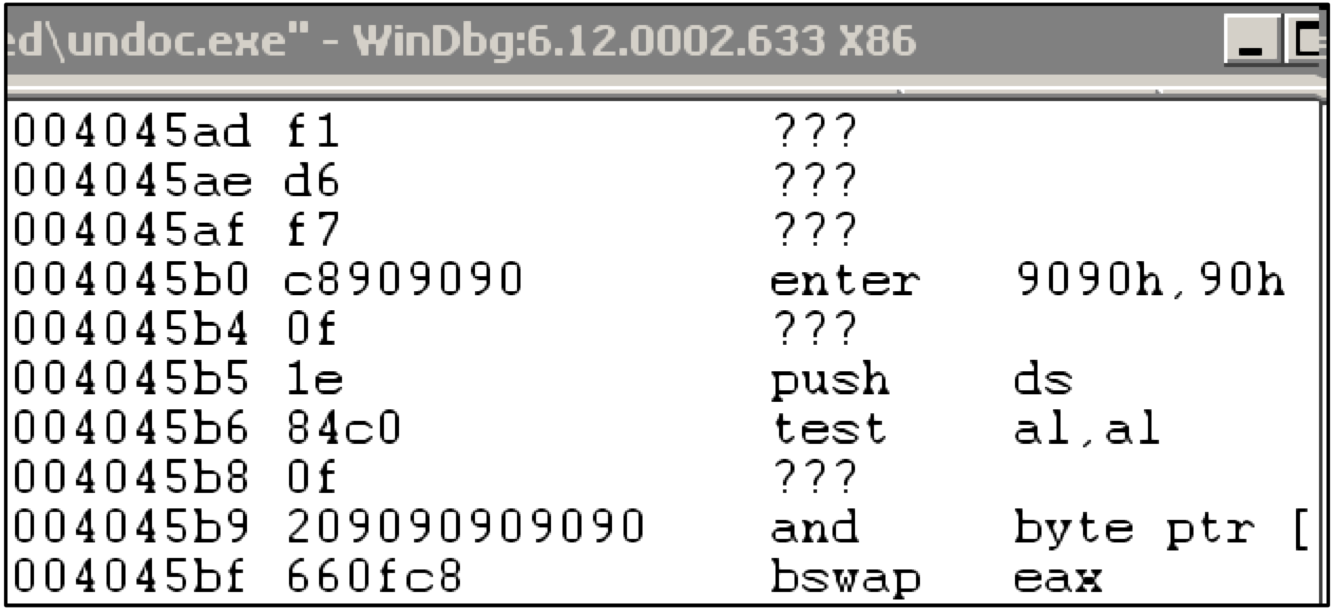
24594—Rev. 3.15—November 2009

AMD64 Technology

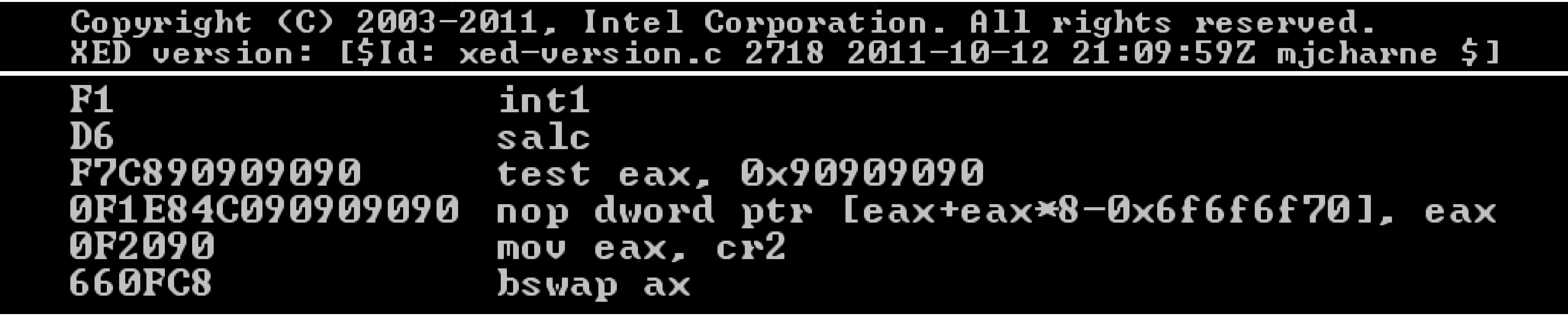
Table A-1. One-Byte Opcodes, Low Nibble 0–7h

Nibble[1]	0	1	2	3	4	5	6	7
0	ADD Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz	PUSH ES[3]	POP ES[3]
1	ADC Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz	PUSH SS[3]	POP SS[3]
2	AND Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz	seg ES[6]	DAA[3]
3	XOR Eb, Gb	Ev, Gv	Gb, Eb	Gv, Ev	AL, Ib	rAX, Iz	seg SS[6]	AAA[3]
4	INC[5] eAX	eCX	eDX	eBX	eSP	eBP	eSI	eDI
5	PUSH rAX/r8	rCX/r9	rDX/r10	rBX/r11	rSP/r12	rBP/r13	rSI/r14	rDI/r15
6	PUSHA/D[3]	POPA/D[3]	BOUND [3] Gv, Ma	ARPL[3] Ew, Gw MOVSXD[4] Gv, Ed	seg FS	seg GS	operand size	address size
7	JO Jb	JNO Jb	JB Jb	JNB Jb	JZ Jb	JNZ Jb	JBE Jb	JNBE Jb
8	Group 1[2] Eb, Ib	Ev, Iz	Eb, Ib[3]	Ev, Ib	TEST Eb, Gb	Ev, Gv	XCHG Eb, Gb	Ev, Gv
9	XCHG r8, rAX NOP,PAUSE	rCX/r9, rAX	rDX/r10, rAX	rBX/r11, rAX	rSP/r12, rAX	rBP/r13, rAX	rSI/r14, rAX	rDI/r15, rAX
A	MOV AL, Ob	rAX, Ov	Ob, AL	Ov, rAX	MOVSB Yb, Xb	MOVSW/D/Q Yv, Xv	CMPSB Xb, Yb	CMPSW/D/Q Xv, Yv
B	MOV AL, Ib r8b, Ib	CL, Ib r9b, Ib	DL, Ib r10b, Ib	BL, Ib r11b, Ib	AH, Ib r12b, Ib	CH, Ib r13b, Ib	DH, Ib r14b, Ib	BH, Ib r15b, Ib
C	Group 2[2] Eb, Ib	Ev, Ib	RET near Iw		LES[3] Gz, Mp	LDS[3] Gz, Mp	Group 11[2] Eb, Ib	Ev, Iz
D	Group 2[2] Eb, 1	Ev, 1	Eb, CL	Ev, CL	AAM[3]	AAD[3]	SALC[3]	XLAT
E	LOOPNE/NZ Jb	LOOPE/Z Jb	LOOP Jb	JrCXZ Jb	IN AL, Ib	eAX, Ib	OUT Ib, AL	Ib, eAX
F	LOCK:	INT1 ICE Bkpt	REPNE:	REP: REPE:	HLT	CMC	Group 3[2]	

“do what I do...”



```
004045ad f1                  ???
004045ae d6                  ???
004045af f7                  ???
004045b0 c8909090            enter   9090h,90h
004045b4 0f                  ???
004045b5 1e                  push    ds
004045b6 84c0                test    al,al
004045b8 0f                  ???
004045b9 209090909090        and     byte ptr [
004045bf 660fc8              bswap   eax
```



```
Copyright (C) 2003-2011, Intel Corporation. All rights reserved.
XED version: [$Id: xed-version.c 2718 2011-10-12 21:09:59Z mjcharne $]
```

```
F1                int1
D6                salc
F7C890909090      test eax, 0x90909090
0F1E84C090909090  nop dword ptr [eax+eax*8-0x6f6f6f70], eax
0F2090            mov eax, cr2
660FC8            bswap ax
```

the problem (1/2)

- the CPU does its stuff
  - whatever we (don't) know
- if we/our tools don't know what's next, we're blind.

the problem (2/2)

no exhaustive or clean test set

- deep into malwares or packers
- scattered

→ Corkami

let's start exploring x86...

Questions

Generalities

- opcodes
- registers
  - relation
  - initial values

Specificities

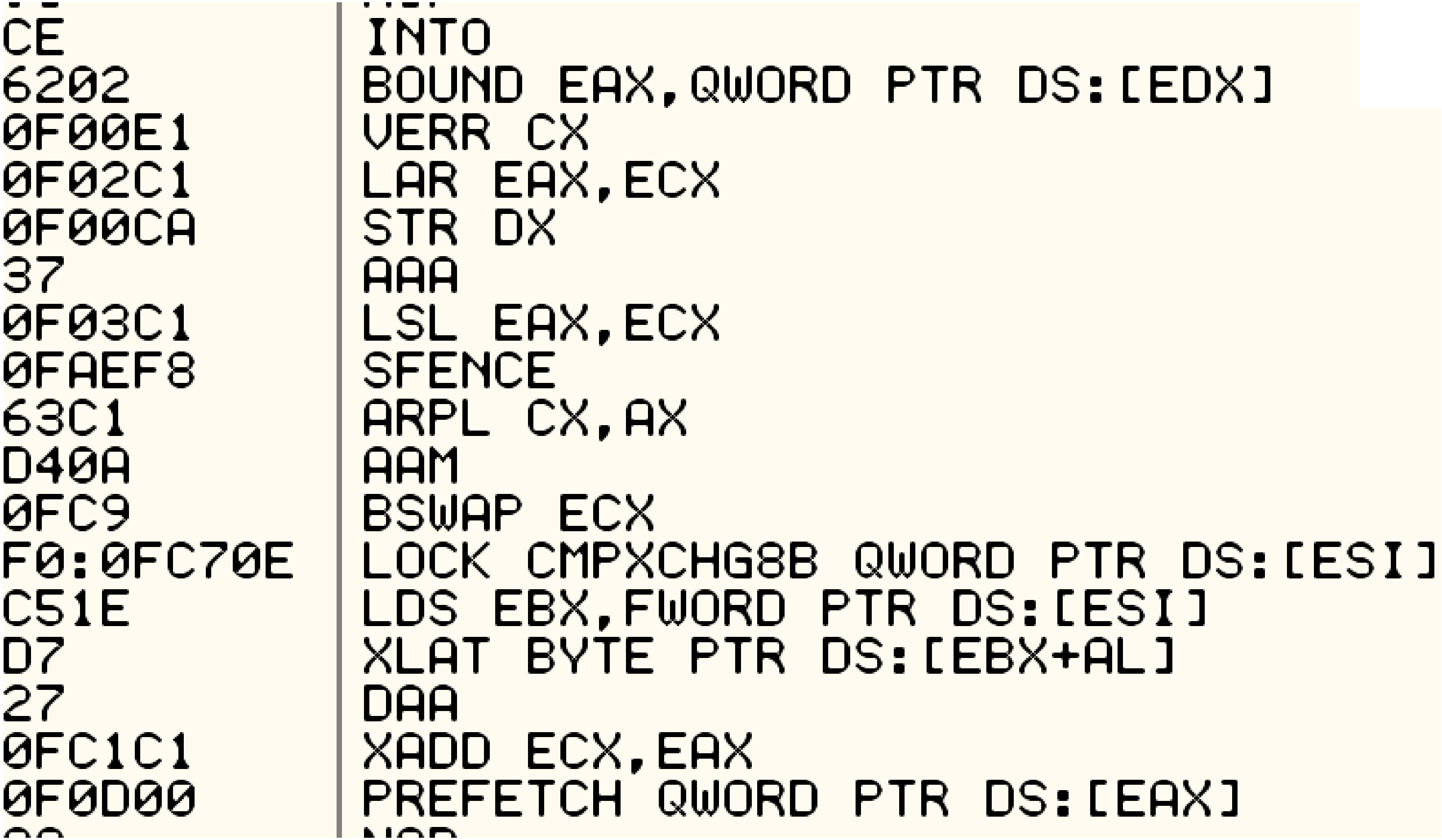
a multi-generation CPU: modern...

English	Assembly
let's go!	*push*
you win	*mov*
sandwich	*call*
hello	*retn*
f*ck	*jmp*

...shakespeare...

thou	*aaa*
porpentine	*xlat*
enmity	*verr*
hither	*smsw*
unkennel	*lsl*

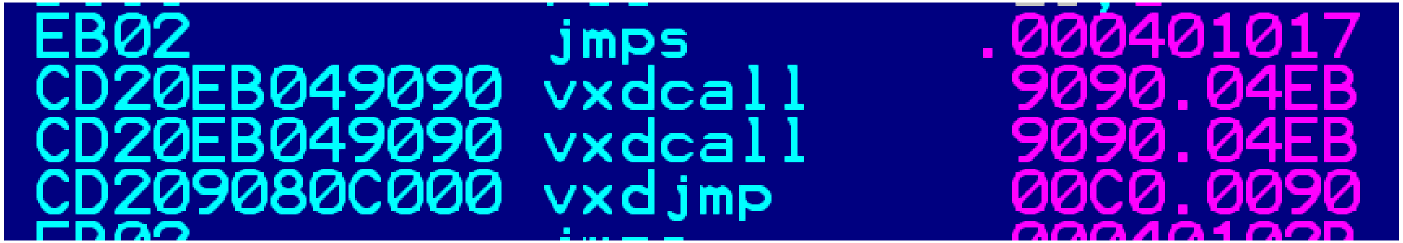
(old, but fully supported)



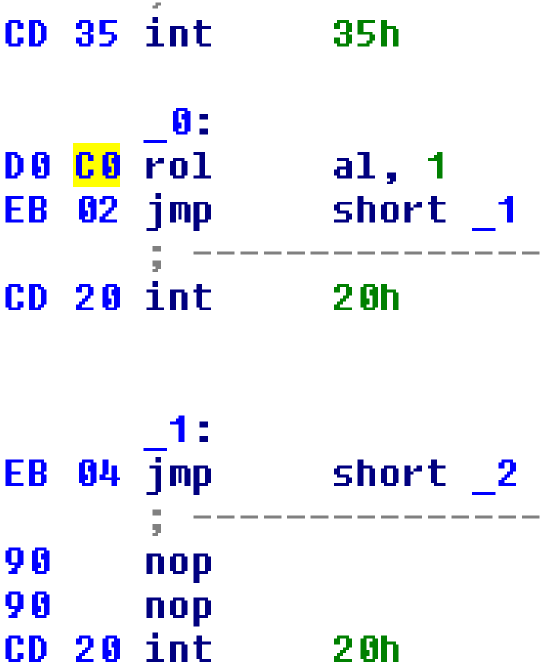
```
CE          INTO
6202        BOUND EAX,QWORD PTR DS:[EDX]
0F00E1      VERR CX
0F02C1      LAR EAX,ECX
0F00CA      STR DX
37          AAA
0F03C1      LSL EAX,ECX
0FAEF8      SFENCE
63C1        ARPL CX,AX
D40A        AAM
0FC9        BSWAP ECX
F0:0FC70E   LOCK CMPXCHG8B QWORD PTR DS:[ESI]
C51E        LDS EBX,FWORD PTR DS:[ESI]
D7          XLAT BYTE PTR DS:[EBX+AL]
27          DAA
0FC1C1      XADD ECX,EAX
0F0D00      PREFETCH QWORD PTR DS:[EAX]
```

'over-disassembling'

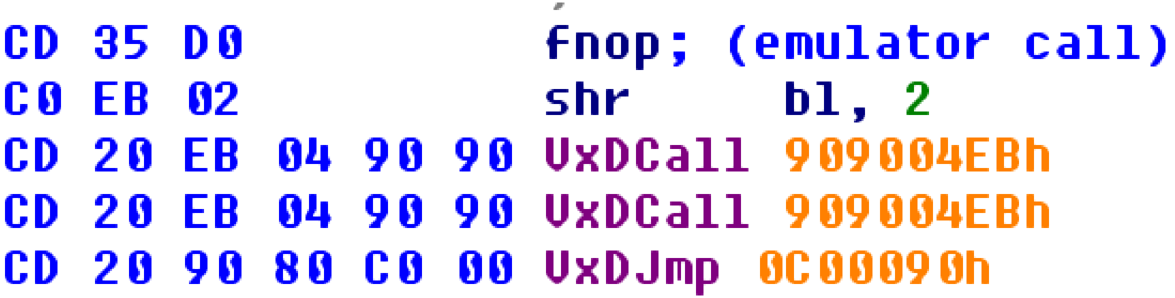
- CD XX: int XX
- deprecated behaviors:
  - int 20h = VXD, int 35-39 = FPU



```
EB02          jmps     .000401017
CD20EB049090  vxdcall   9090.04EB
CD20EB049090  vxdcall   9090.04EB
CD209080C000  vxdjmp    00C0.0090
```



```
CD 35 int     35h

      _0:
D0 C0 rol     al, 1
EB 02 jmp     short _1
      ; ---------------
CD 20 int     20h


      _1:
EB 04 jmp     short _2
      ; ---------------
90    nop
90    nop
CD 20 int     20h
```

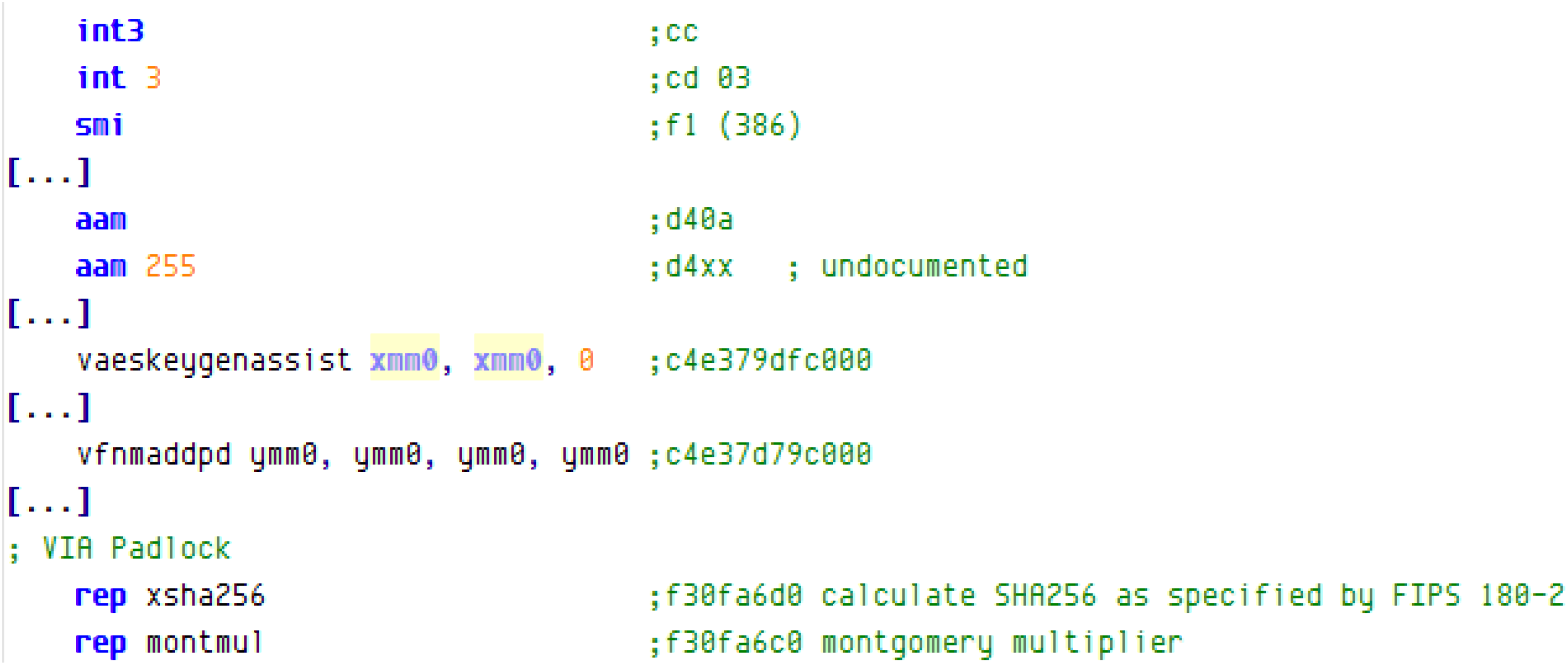


```
CD 35 D0          fnop; (emulator call)
C0 EB 02          shr     bl, 2
CD 20 EB 04 90 90 VxDCall 909004EBh
CD 20 EB 04 90 90 VxDCall 909004EBh
CD 20 90 80 C0 00 VxDJmp  0C00090h
```

...next generation

tweet	*crc32*
poke	*aesenc*
google	*pcmpistrm*
pwn	*vfmsubadd132ps* Fused Multiply-Alternating Subtract/Add of Packed Single-Precision Floating-Point Values
apps	*movbe* only in netbooks!

all opcodes PoC



```
    int3                             ;cc
    int 3                            ;cd 03
    smi                              ;f1 (386)
[...]
    aam                              ;d40a
    aam 255                          ;d4xx    ; undocumented
[...]
    vaeskeygenassist xmm0, xmm0, 0   ;c4e379dfc000
[...]
    vfnmaddpd ymm0, ymm0, ymm0, ymm0 ;c4e37d79c000
[...]
; VIA Padlock
    rep xsha256                      ;f30fa6d0 calculate SHA256 as specified by FIPS 180-2
    rep montmul                      ;f30fa6c0 montgomery multiplier
```

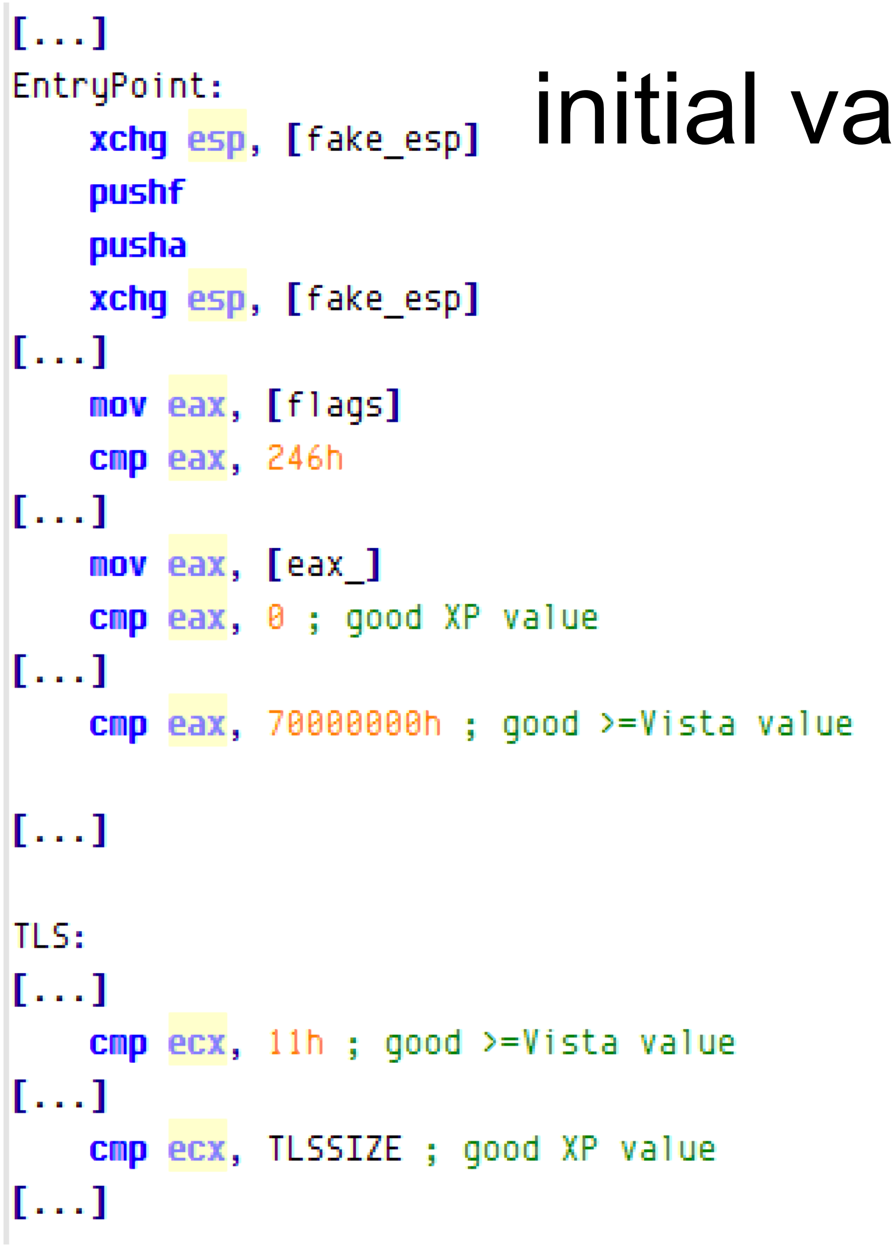
registers

- Complex relations
  - FPU changes FST, STx, Mmx (ST0 overlaps MM7)
    - also changes CR0 (under XP)

- Initial values
  - AX = <OS generation>
    - OS = (EAX == 0) ? XP : newer
  - GS = <number of bits>

    bits = (GS == 0) ? 32 : 64

initial values PoC



```
[...]
EntryPoint:
    xchg esp, [fake_esp]
    pushf
    pusha
    xchg esp, [fake_esp]
[...]
    mov eax, [flags]
    cmp eax, 246h
[...]
    mov eax, [eax_]
    cmp eax, 0 ; good XP value
[...]
    cmp eax, 70000000h ; good >=Vista value

[...]

TLS:
[...]
    cmp ecx, 11h ; good >=Vista value
[...]
    cmp ecx, TLSSIZE ; good XP value
[...]
```

	XP	W7
Flags	🟦	🟦
TLS		
eax	🟥	🟦
ecx	⬛	🟦
edx	🟥	🟩
ebx	🟦	🟥
EntryPoint		
eax	🟦	🟩
ecx	🟩	🟦
edx	🟩	🟥

fully ctrl-ed

controlled

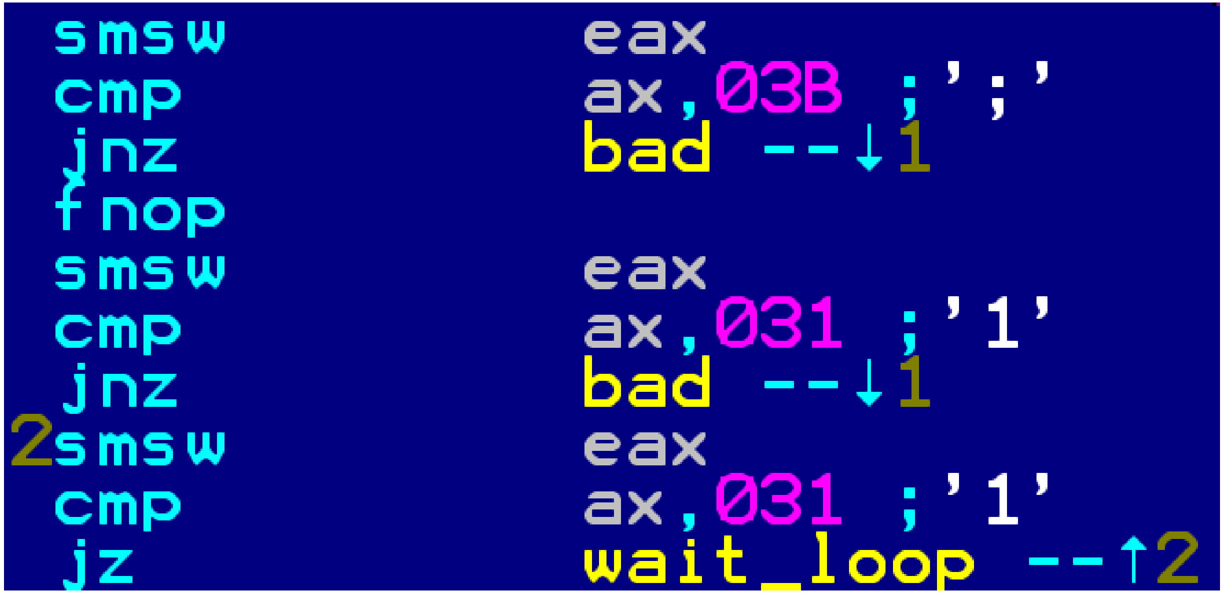
fixed

range

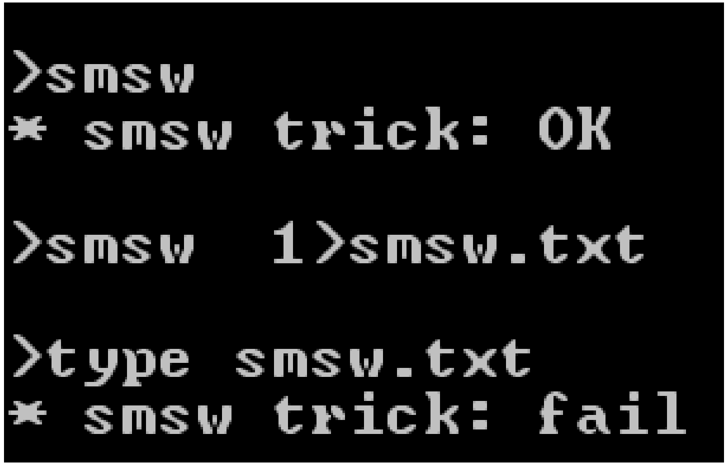
smsw

- CR0 access, from user-mode
  - 286 opcode
- higher word of reg32 'undefined'
- under XP
  - influenced by FPU
  - eventually reverts

DEMO



```
  smsw              eax
  cmp               ax,03B ;';'
  jnz               bad --↓1
 ƒnop
  smsw              eax
  cmp               ax,031 ;'1'
  jnz               bad --↓1
2smsw               eax
  cmp               ax,031 ;'1'
  jz                wait_loop --↑2
```

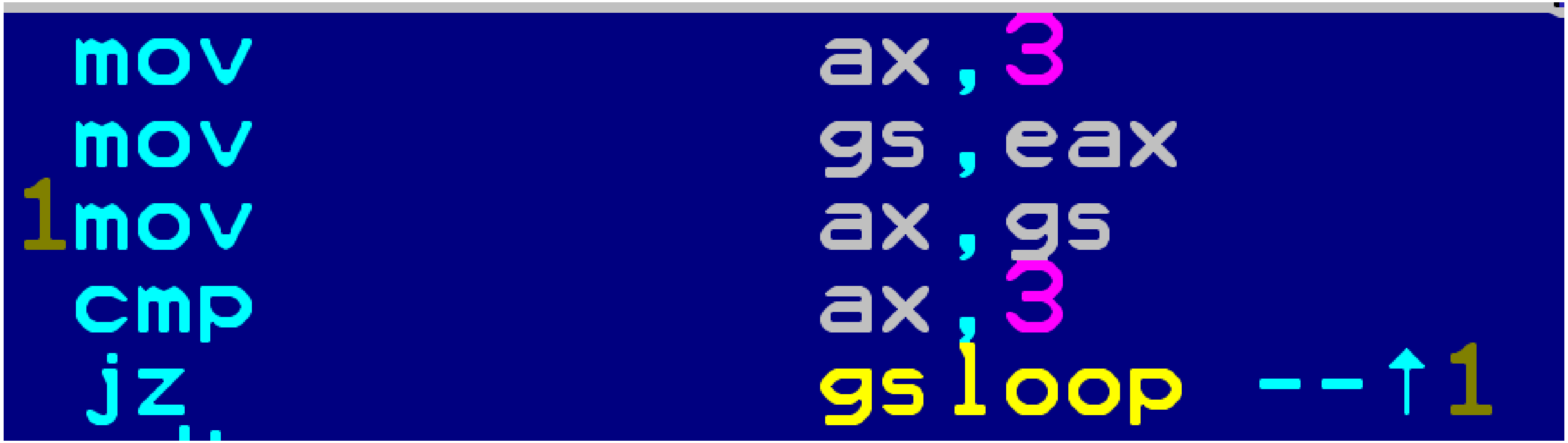


```
>smsw
* smsw trick: OK

>smsw  1>smsw.txt

>type smsw.txt
* smsw trick: fail
```

GS

- unused on Windows 32b
  - on 64b: FS, GS = TEB32, TEB64
- reset on thread switch
  - eventually reset
    - debugger stepping
    - wait
    - timings

DEMO



```
 mov                     ax,3
 mov                     gs,eax
1mov                     ax,gs
 cmp                     ax,3
 jz                      gsloop --↑1
```

nop

- *nop* is *xchg *ax, *ax*
  - but *xchg *ax, *ax* can **do** something, in 64b !

    87 c0: xchg eax, eax

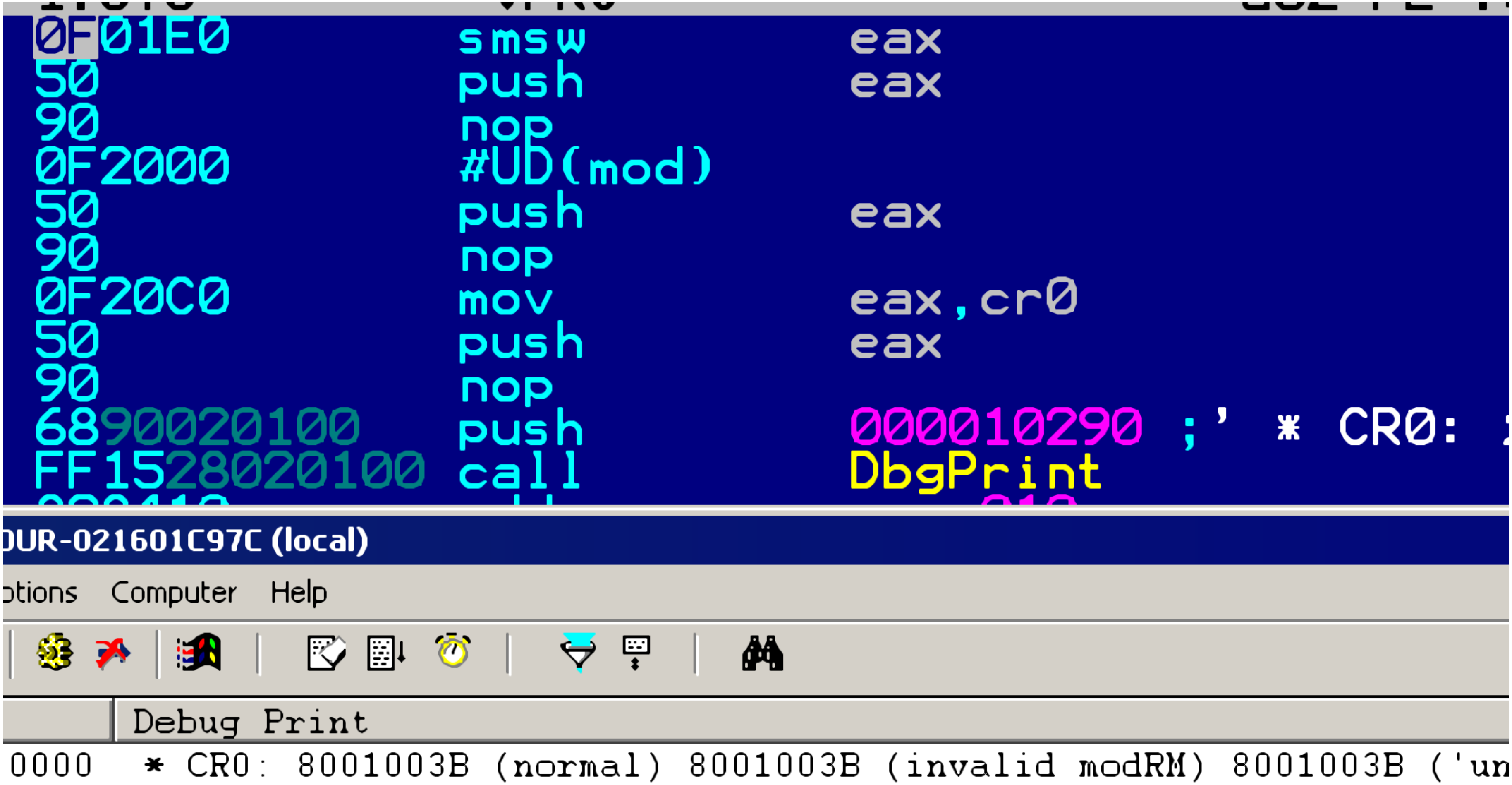
```
.. .. .. .. 01 23 45 67 => 00 00 00 00 01 23 45 67
```

- *hint nop* `0F1E84C090909090 nop dword ptr [eax+eax*8-0x6f6f6f70], eax`
  - partially undocumented, actually 0f 18-1f
  - can trigger exception

mov

- documented, but sometimes tricky
    - *mov [cr0], eax*   *mov cr0, eax*
        - mod/RM is ignored
    - *movsxd eax, ecx*   *mov eax, ecx*
        - no REX prefix
    - *mov eax, cs*   *movzx eax,cs*
        - 'undefined' upper word

non standard CR0 access



```
0F01E0          smsw    eax
50              push    eax
90              nop
0F2000          #UD(mod)
50              push    eax
90              nop
0F20C0          mov     eax,cr0
50              push    eax
90              nop
6890020100      push    00010290 ;' * CR0: 
FF1528020100    call    DbgPrint
```

```
0000    * CR0: 8001003B (normal) 8001003B (invalid modRM) 8001003B ('un
```

bswap

rax

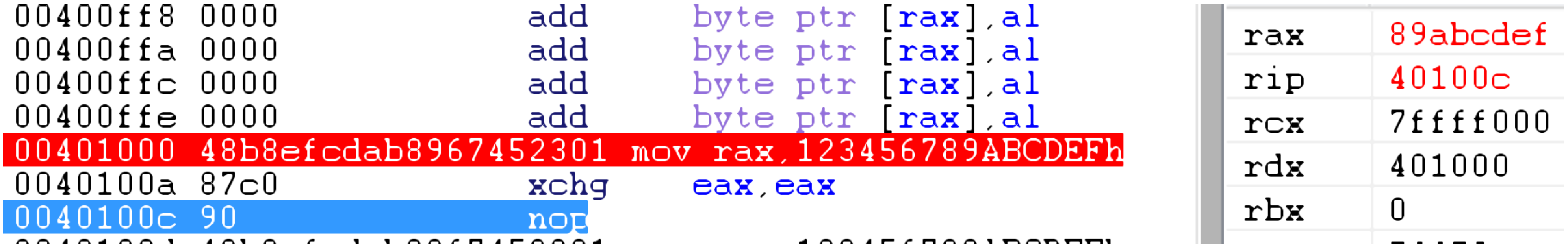
```
12 34 56 78 90 ab cd ef => ef cd ab 90 78 56 34 12
```

eax

```
.. .. .. .. 01 23 45 67 => 00 00 00 00 67 45 23 01
```

ax

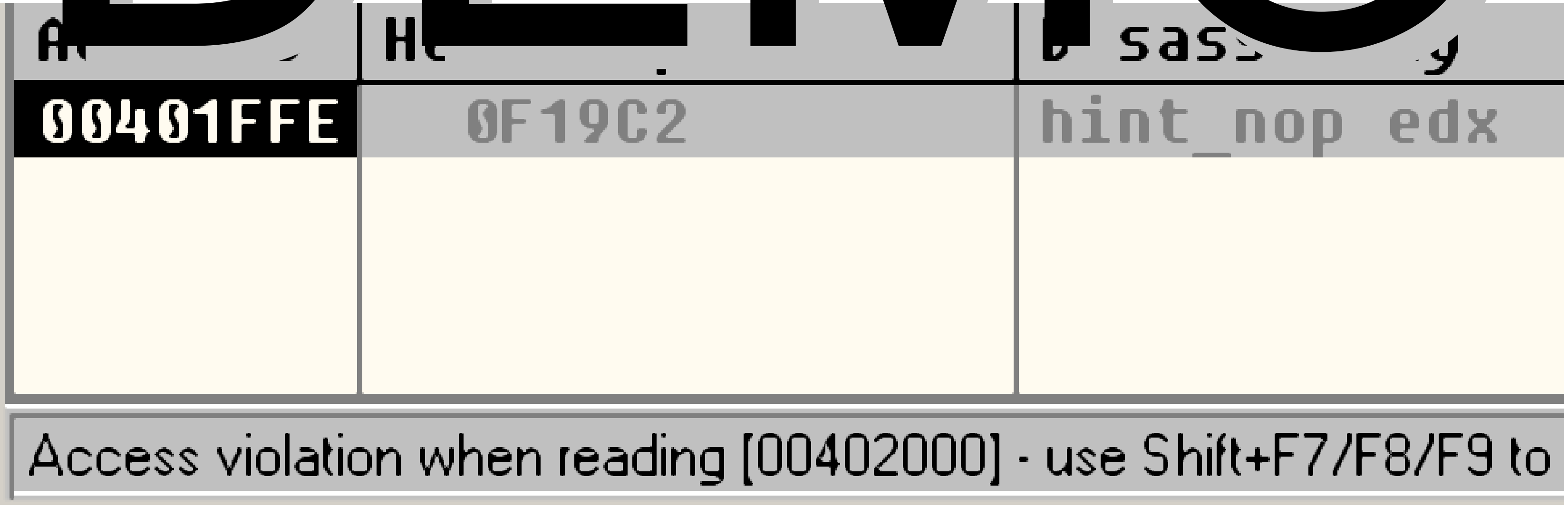
```
.. .. .. .. .. .. 01 23 => .. .. .. .. .. .. 00 00
```



```
00400ff8 0000                 add     byte ptr [rax],al
00400ffa 0000                 add     byte ptr [rax],al
00400ffc 0000                 add     byte ptr [rax],al
00400ffe 0000                 add     byte ptr [rax],al
00401000 48b8efcdab8967452301 mov rax,123456789ABCDEFh
0040100a 87c0                 xchg    eax,eax
0040100c 90                   nop
```

```
rax   89abcdef
rip   40100c
rcx   7ffff000
rdx   401000
rbx   0
```

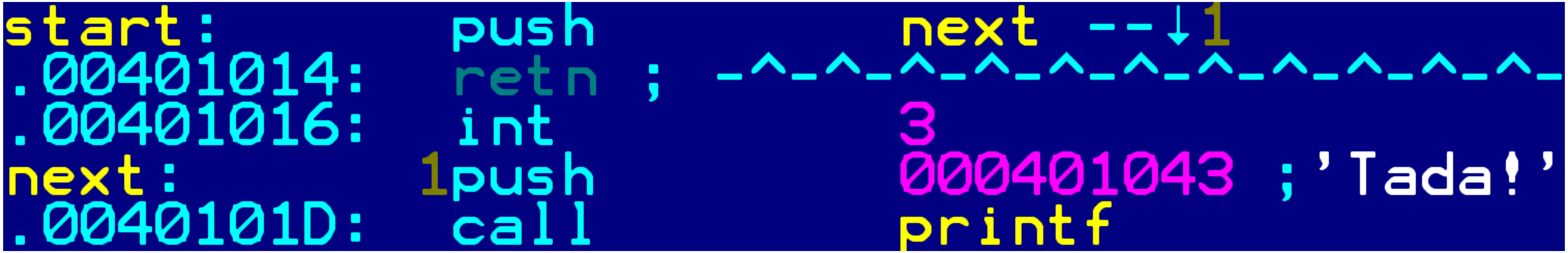
DEMO



```
00401FFE   0F19C2   hint_nop edx
```

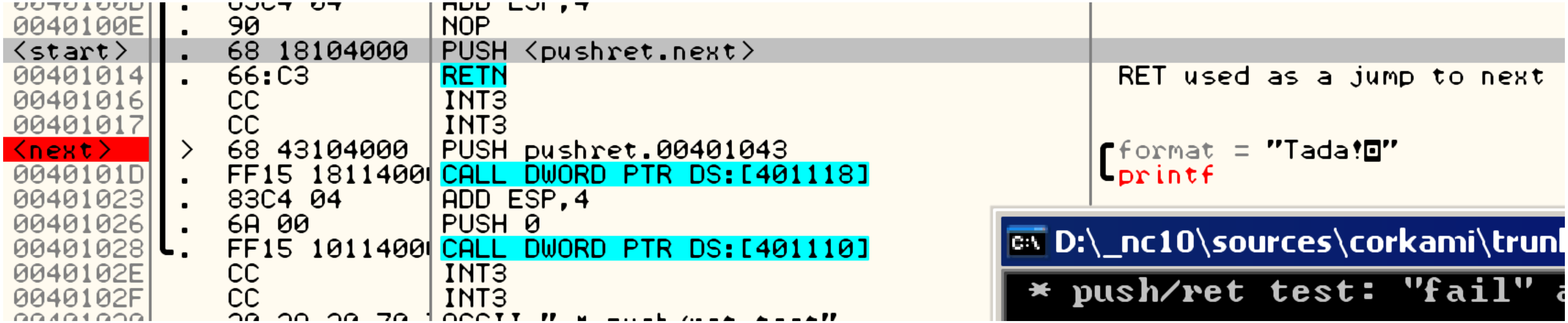
Access violation when reading [00402000] - use Shift+F7/F8/F9 to

push+ret



```
start:          push            next --↓1
.00401014:      retn ; _^_^_^_^_^_^_^_^_^_^_^_
.00401016:      int             3
next:          1push            000401043 ;'Tada!'
.0040101D:      call            printf
```

DEMO



```
0040100B  .  83C4 04          ADD ESP,4
0040100E  .  90               NOP
<start>   .  68 18104000      PUSH <pushret.next>
00401014  .  66:C3            RETN                            RET used as a jump to next
00401016     CC               INT3
00401017     CC               INT3
<next>    >  68 43104000      PUSH pushret.00401043          [format = "Tada!□"
0040101D  .  FF15 1811400     CALL DWORD PTR DS:[401118]     [printf
00401023  .  83C4 04          ADD ESP,4
00401026  .  6A 00            PUSH 0
00401028  .  FF15 1011400     CALL DWORD PTR DS:[401110]
0040102E     CC               INT3
0040102F     CC               INT3
00401030  .  20 20 20 70      ASCII " * push/ret test"
```

```
D:\_nc10\sources\corkami\trun
 * push/ret test: "fail" a
```

...and so on...

- much more @ http://x86.corkami.com
  - also graphs, cheat sheet...

- too much theory for now...

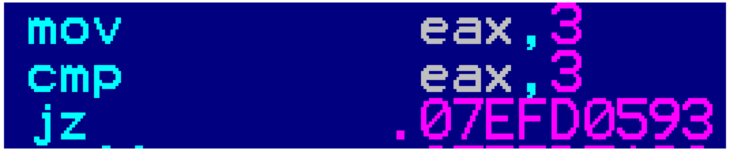
Corkami Standard Test

CoST

- http://cost.corkami.com
- testing opcodes
- in a hardened PE
  - available in easy mode

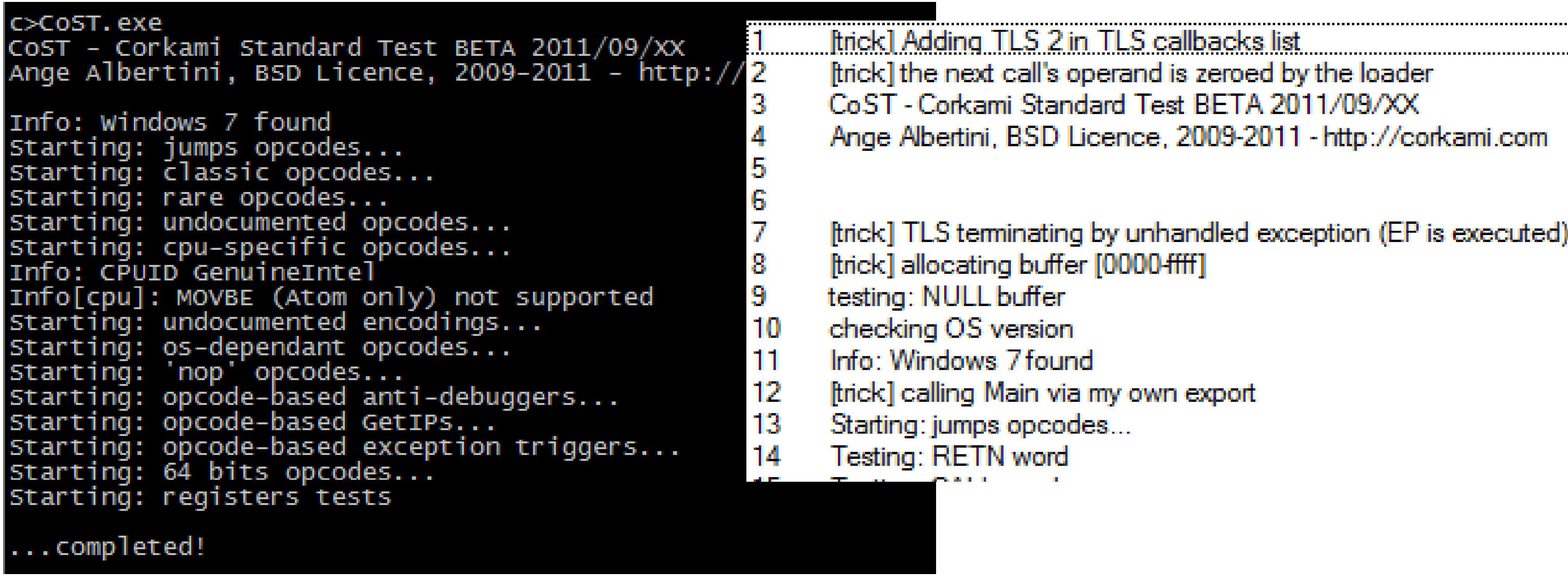
more than 150 tests

- classic, rare
- jumps (JMP to IP, IRET, …)
- undocumented (IceBP, SetALc...)
- cpu-specific (MOVBE, POPCNT,...)
- os-dependant, anti-VM/debugs
- exceptions triggers, interrupts, OS bugs,...
- ...



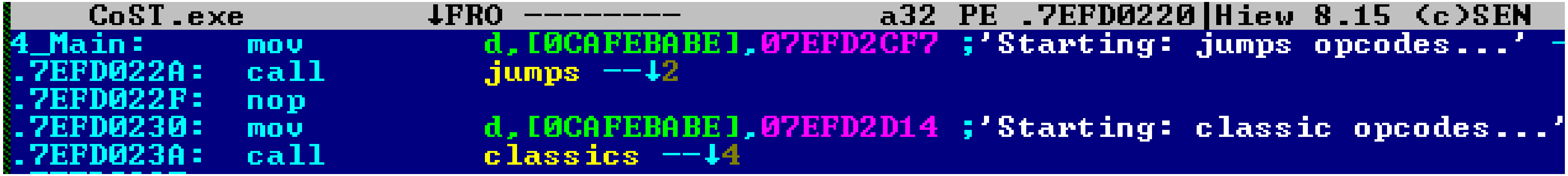
```
mov                     eax,3
cmp                     eax,3
jz                     .07EFD0593
```

CoST's internals



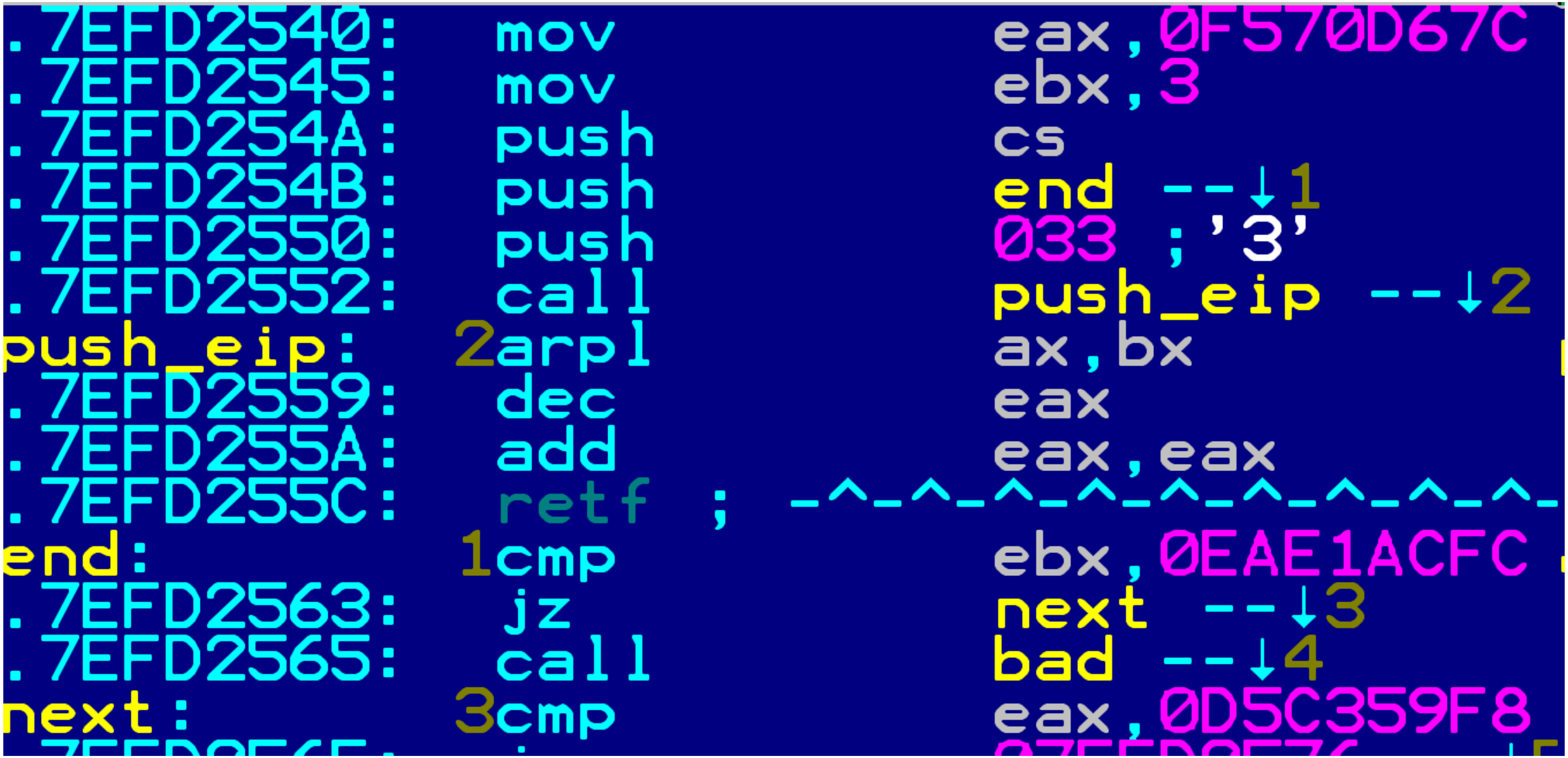
```
c>CoST.exe
CoST - Corkami Standard Test BETA 2011/09/XX
Ange Albertini, BSD Licence, 2009-2011 - http://

Info: Windows 7 found
Starting: jumps opcodes...
Starting: classic opcodes...
Starting: rare opcodes...
Starting: undocumented opcodes...
Starting: cpu-specific opcodes...
Info: CPUID GenuineIntel
Info[cpu]: MOVBE (Atom only) not supported
Starting: undocumented encodings...
Starting: os-dependant opcodes...
Starting: 'nop' opcodes...
Starting: opcode-based anti-debuggers...
Starting: opcode-based GetIPs...
Starting: opcode-based exception triggers...
Starting: 64 bits opcodes...
Starting: registers tests

...completed!
```

```
[trick] Adding TLS 2 in TLS callbacks list
[trick] the next call's operand is zeroed by the loader
CoST - Corkami Standard Test BETA 2011/09/XX
Ange Albertini, BSD Licence, 2009-2011 - http://corkami.com


[trick] TLS teminating by unhandled exception (EP is executed)
[trick] allocating buffer [0000-ffff]
testing: NULL buffer
checking OS version
Info: Windows 7 found
[trick] calling Main via my own export
Starting: jumps opcodes...
Testing: RETN word
```

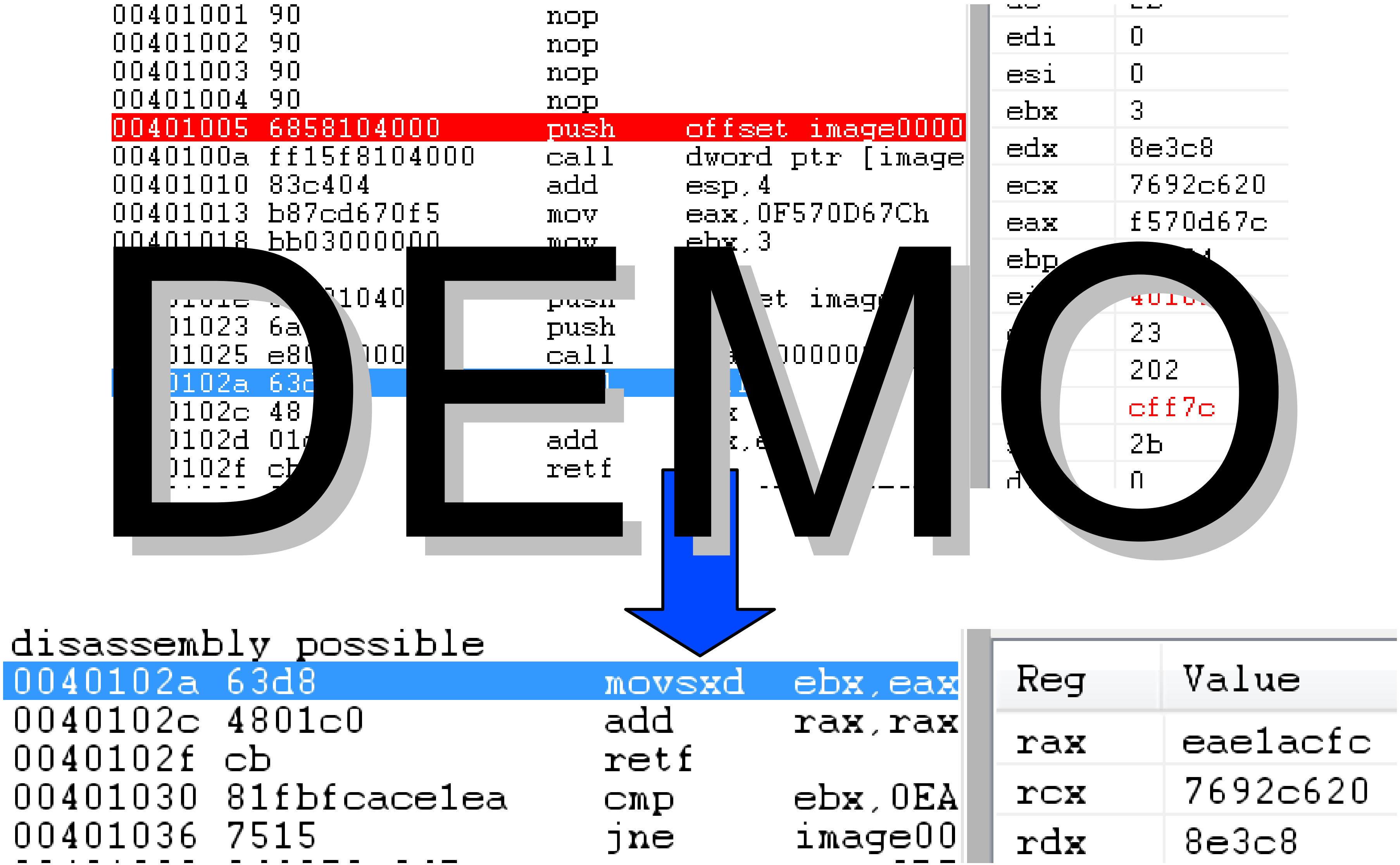


```
   CoST.exe           ↓FRO --------          a32 PE .7EFD0220|Hiew 8.15 (c)SEN
4_Main:       mov          d,[0CAFEBABE],07EFD2CF7 ;'Starting: jumps opcodes...'
.7EFD022A:    call         jumps --↓2
.7EFD022F:    nop
.7EFD0230:    mov          d,[0CAFEBABE],07EFD2D14 ;'Starting: classic opcodes...'
.7EFD023A:    call         classics --↓4
```

32+64 = ...



```
.7EFD2540:   mov                 eax,0F570D67C
.7EFD2545:   mov                 ebx,3
.7EFD254A:   push                cs
.7EFD254B:   push                end --↓1
.7EFD2550:   push                033 ;'3'
.7EFD2552:   call                push_eip --↓2
push_eip:   2arpl                ax,bx
.7EFD2559:   dec                 eax
.7EFD255A:   add                 eax,eax
.7EFD255C:   retf ; _^_^_^_^_^_^_^_^_^_
end:        1cmp                 ebx,0EAE1ACFC
.7EFD2563:   jz                  next --↓3
.7EFD2565:   call                bad --↓4
next:       3cmp                 eax,0D5C359F8
```



```
00401001 90               nop
00401002 90               nop
00401003 90               nop
00401004 90               nop
00401005 6858104000       push    offset image0000
0040100a ff15f8104000     call    dword ptr [image
00401010 83c404           add     esp,4
00401013 b87cd670f5       mov     eax,0F570D67Ch
00401018 bb03000000       mov     ebx,3
```

edi	0
esi	0
ebx	3
edx	8e3c8
ecx	7692c620
eax	f570d67c

DEMO

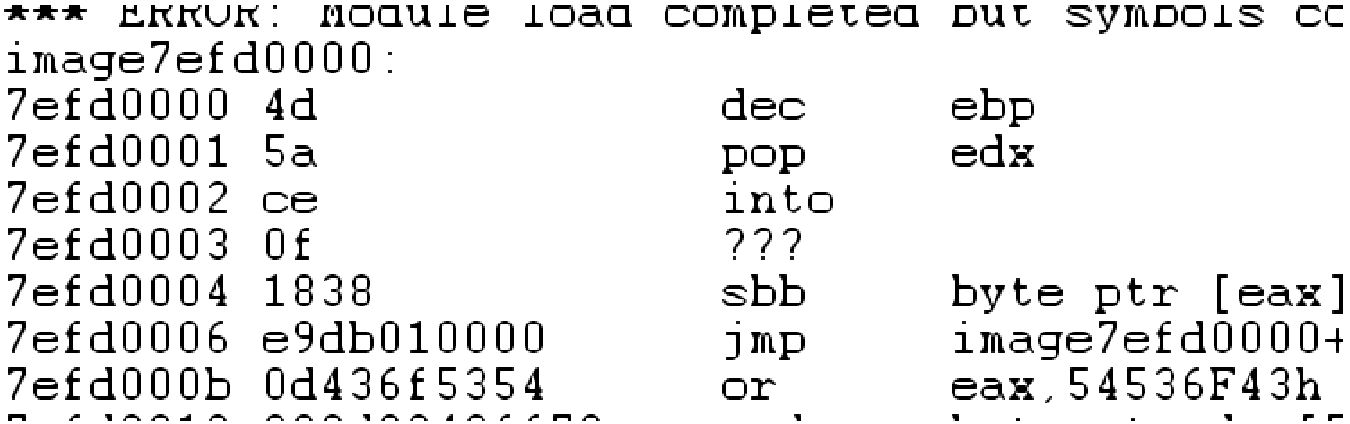
disassembly possible

```
0040102a 63d8             movsxd  ebx,eax
0040102c 4801c0           add     rax,rax
0040102f cb               retf
00401030 81fbfcace1ea     cmp     ebx,0EA
00401036 7515             jne     image00
```

Reg	Value
rax	eae1acfc
rcx	7692c620
rdx	8e3c8

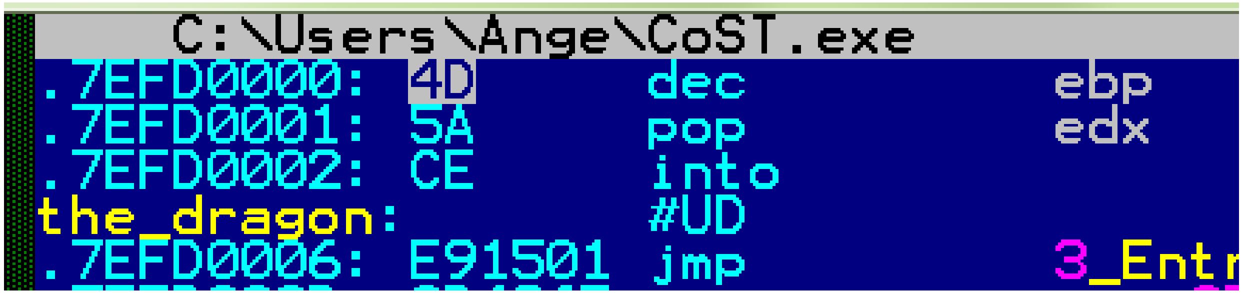
CoST vs WinDbg & Hiew

WinDbg 6.12.0002.633



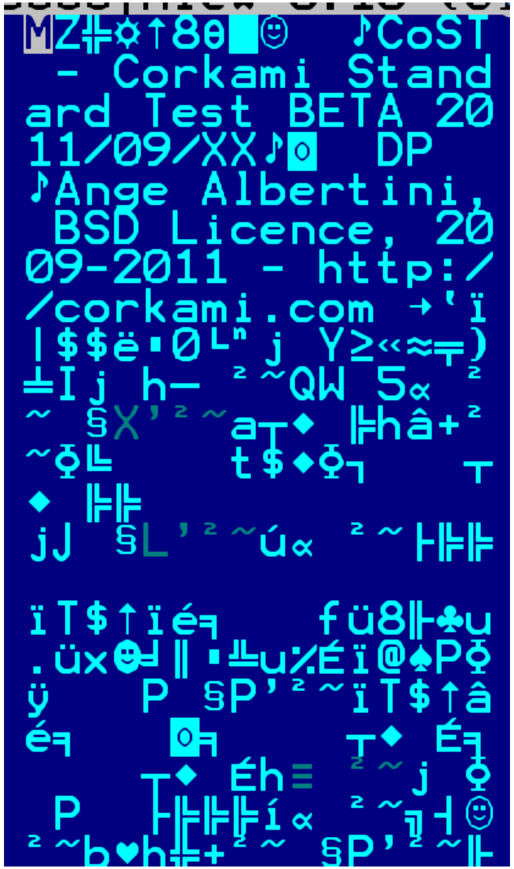
```
*** ERROR: Module load completed but symbols cc
image7efd0000:
7efd0000 4d                  dec     ebp
7efd0001 5a                  pop     edx
7efd0002 ce                  into
7efd0003 0f                  ???
7efd0004 1838                sbb     byte ptr [eax]
7efd0006 e9db010000          jmp     image7efd0000+
7efd000b 0d436f5354          or      eax,54536F43h
```

Hiew 8.15



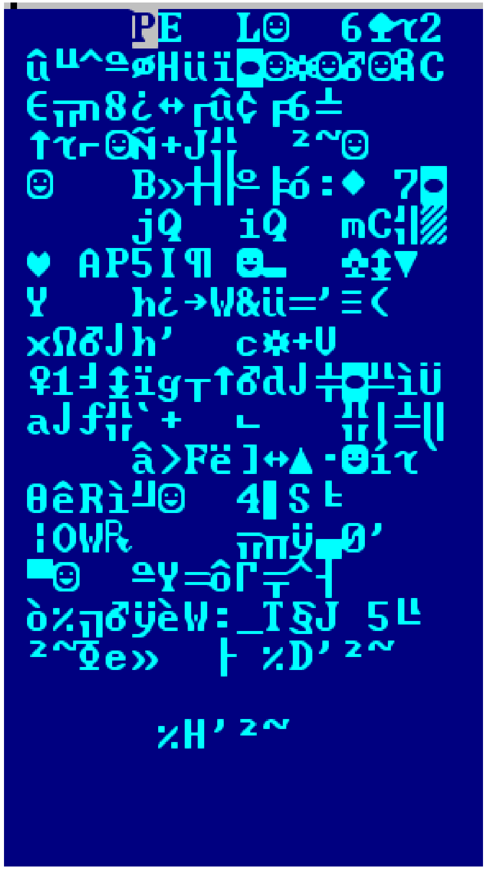
```
      C:\Users\Ange\CoST.exe
.7EFD0000: 4D      dec          ebp
.7EFD0001: 5A      pop          edx
.7EFD0002: CE      into
the_dragon:        #UD
.7EFD0006: E91501  jmp          3_Entr
```

a hardened PE



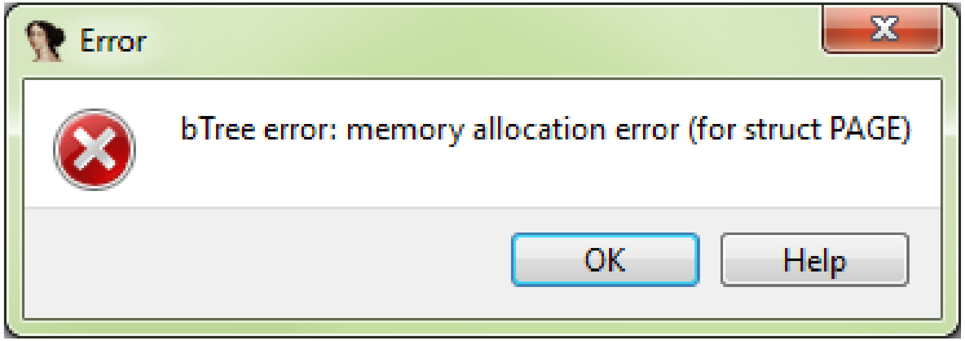
```
MZ╬☼↑8θ █☺   ♪CoST
  - Corkami Stand
ard Test BETA 20
11/09/XX♪◙  DP
♪Ange Albertini,
 BSD Licence, 20
09-2011 - http:/
/corkami.com
```

Top

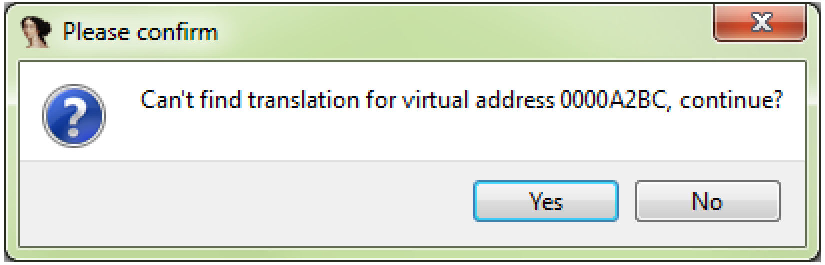


PE 'footer'

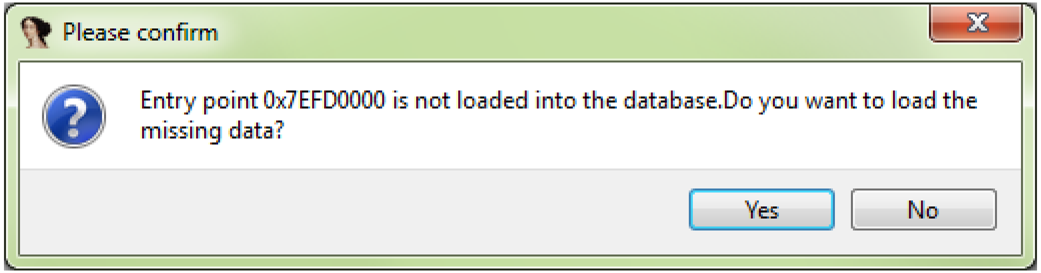
CoST vs IDA



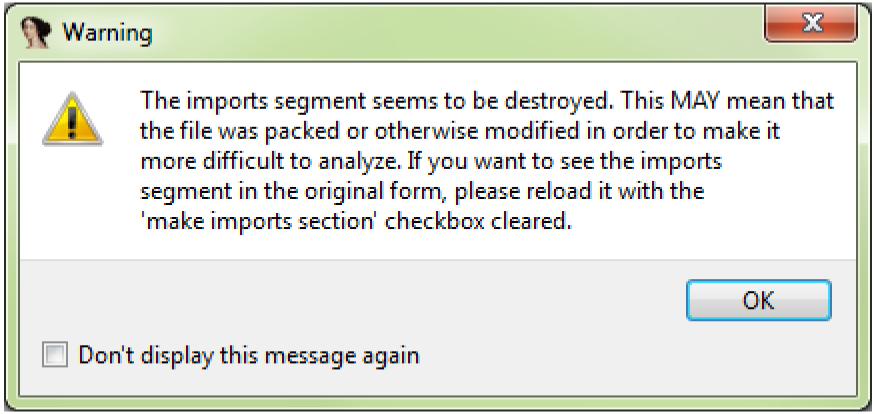
bTree error: memory allocation error (for struct PAGE)



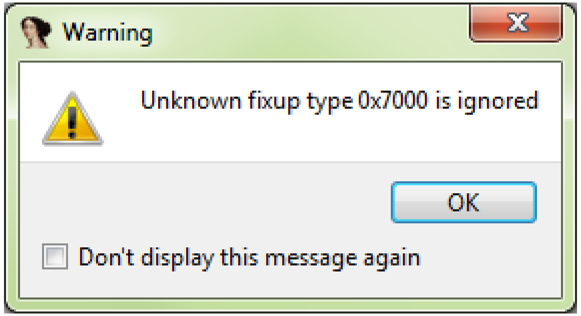
Can't find translation for virtual address 0000A2BC, continue?



Entry point 0x7EFD0000 is not loaded into the database.Do you want to load the missing data?



The imports segment seems to be destroyed. This MAY mean that the file was packed or otherwise modified in order to make it more difficult to analyze. If you want to see the imports segment in the original form, please reload it with the 'make imports section' checkbox cleared.



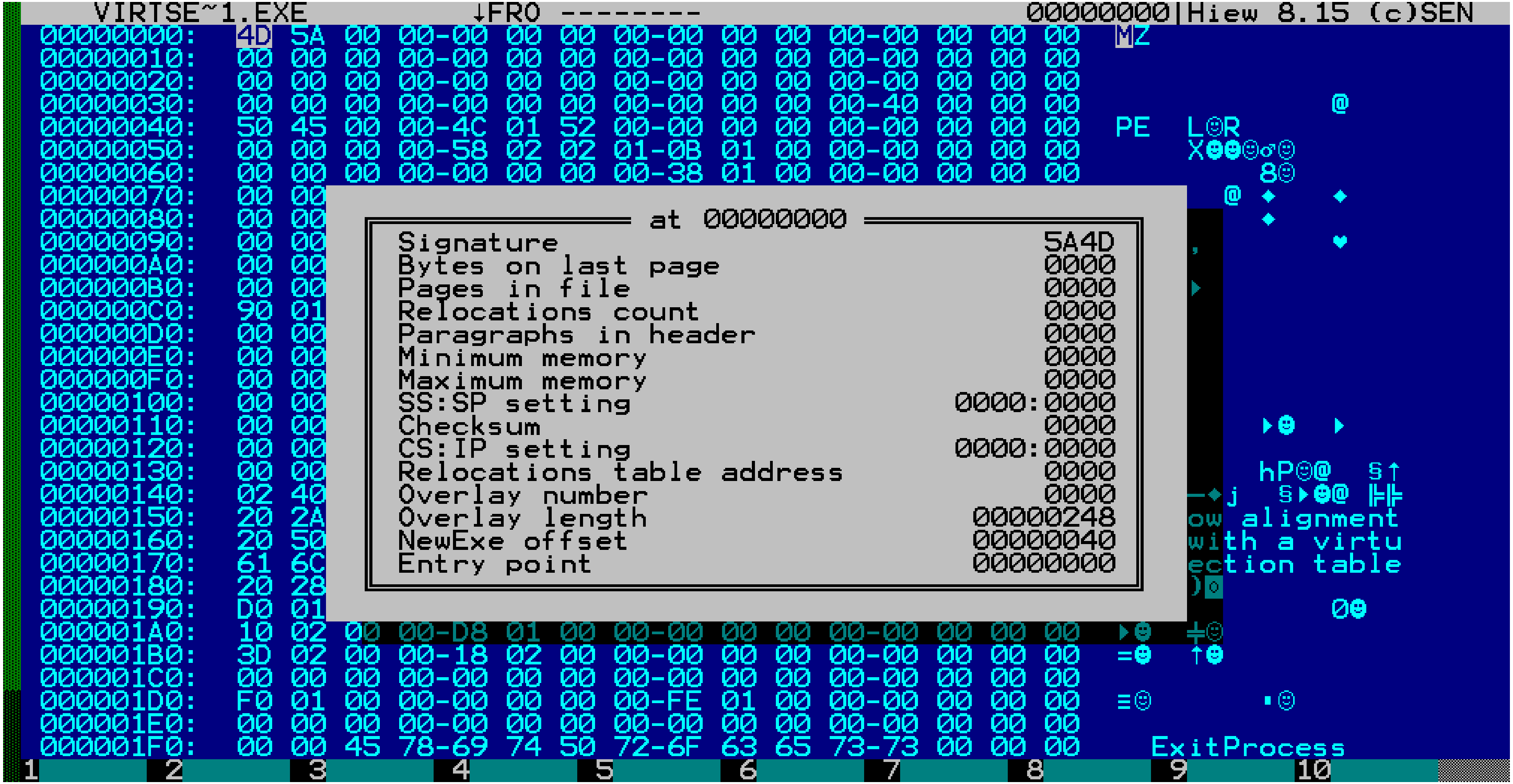
Unknown fixup type 0x7000 is ignored

a bit more of PE...

PE on Corkami

- still in progress
- more than 120 PoCs
  - covering many aspects
  - good enough to break <you name it>
- 'summary' page http://pe.corkami.com
- printable graphs

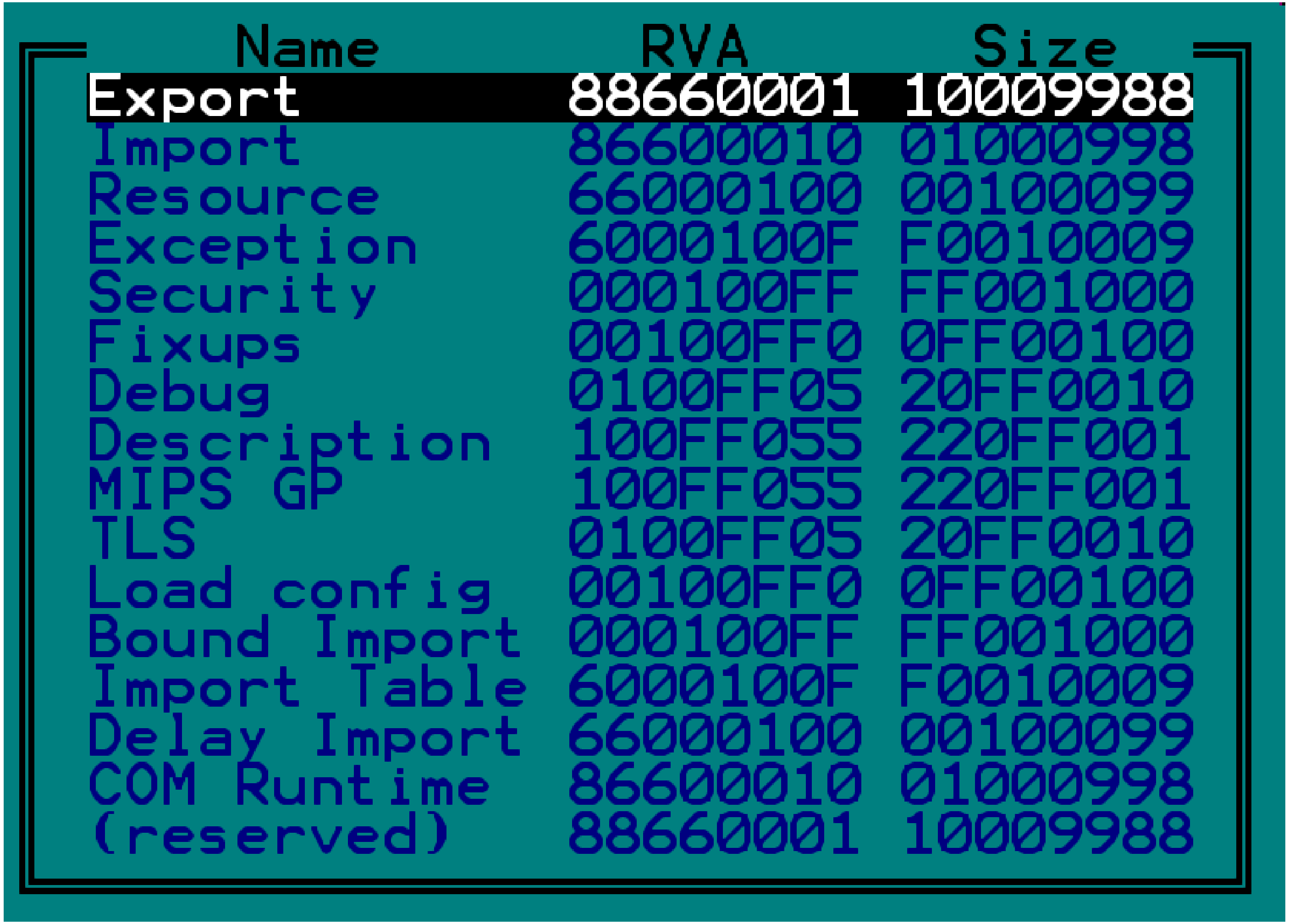
virtual section table vs Hiew



```
    VIRTSE~1.EXE            ↓FRO --------         00000000|Hiew 8.15 (c)SEN
00000000:  4D 5A 00 00-00 00 00 00-00 00 00 00-00 00 00 00  MZ
00000010:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000020:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
00000030:  00 00 00 00-00 00 00 00-00 00 00 40-00 00 00 00              @
00000040:  50 45 00 00-4C 01 52 00-00 00 00 00-00 00 00 00  PE  L☺R
00000050:  00 00 00 00-58 02 02 01-0B 01 00 00-00 00 00 00      X☻☻☺♂☺
00000060:  00 00 00 00-00 00 00 00-38 01 00 00-00 00 00 00          8☺
00000070:  00 00                                                  @ ♦     ♦
00000080:  00 00                                                    ♦
00000090:  00 00                                               ,          ♥
000000A0:  00 00
000000B0:  00 00                                               ►
000000C0:  90 01
000000D0:  00 00
000000E0:  00 00
000000F0:  00 00
00000100:  00 00
00000110:  00 00                                                    ►☻    ►
00000120:  00 00
00000130:  00 00                                                    hP☺@  §↑
00000140:  02 40                                               ─♦j  §►☻@ ╟╟
00000150:  20 2A                                               ow alignment
00000160:  20 50                                               with a virtu
00000170:  61 6C                                               ection table
00000180:  20 28                                               )○
00000190:  D0 01                                                          0☻
000001A0:  10 02 00 00-D8 01 00 00-00 00 00 00-00 00 00 00  ►☻  ╪☺
000001B0:  3D 02 00 00-18 02 00 00-00 00 00 00-00 00 00 00  =☻  ↑☻
000001C0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001D0:  F0 01 00 00-00 00 00 00-FE 01 00 00-00 00 00 00  ≡☺      ■☺
000001E0:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
000001F0:  00 00 45 78-69 74 50 72-6F 63 65 73-73 00 00 00    ExitProcess
1       2       3       4       5       6       7       8       9       10
```

```
═══════════════ at 00000000 ═══════════════
 Signature                              5A4D
 Bytes on last page                     0000
 Pages in file                          0000
 Relocations count                      0000
 Paragraphs in header                   0000
 Minimum memory                         0000
 Maximum memory                         0000
 SS:SP setting                     0000:0000
 Checksum                               0000
 CS:IP setting                     0000:0000
 Relocations table address              0000
 Overlay number                         0000
 Overlay length                     00000248
 NewExe offset                      00000040
 Entry point                        00000000
```

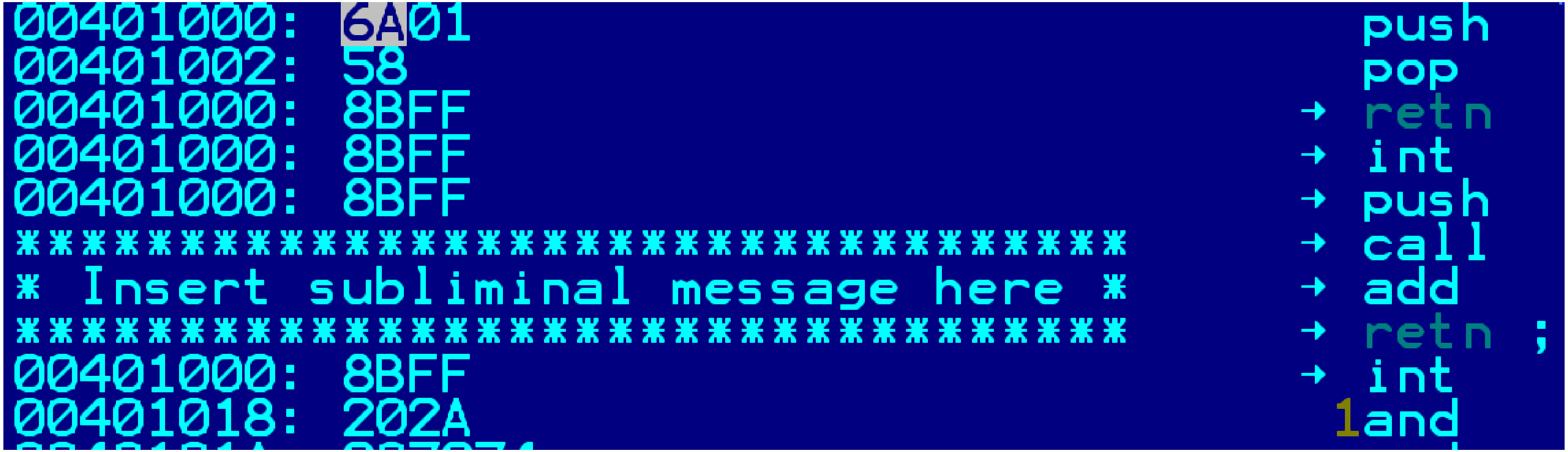
Folded header



```
      Name              RVA       Size
Export               88660001  10009988
Import               86600010  01000998
Resource             66000100  00100099
Exception            6000100F  F0010009
Security             000100FF  FF001000
Fixups               00100FF0  0FF00100
Debug                0100FF05  20FF0010
Description          100FF055  220FF001
MIPS GP              100FF055  220FF001
TLS                  0100FF05  20FF0010
Load config          00100FF0  0FF00100
Bound Import         000100FF  FF001000
Import Table         6000100F  F0010009
Delay Import         66000100  00100099
COM Runtime          86600010  01000998
(reserved)           88660001  10009988
```

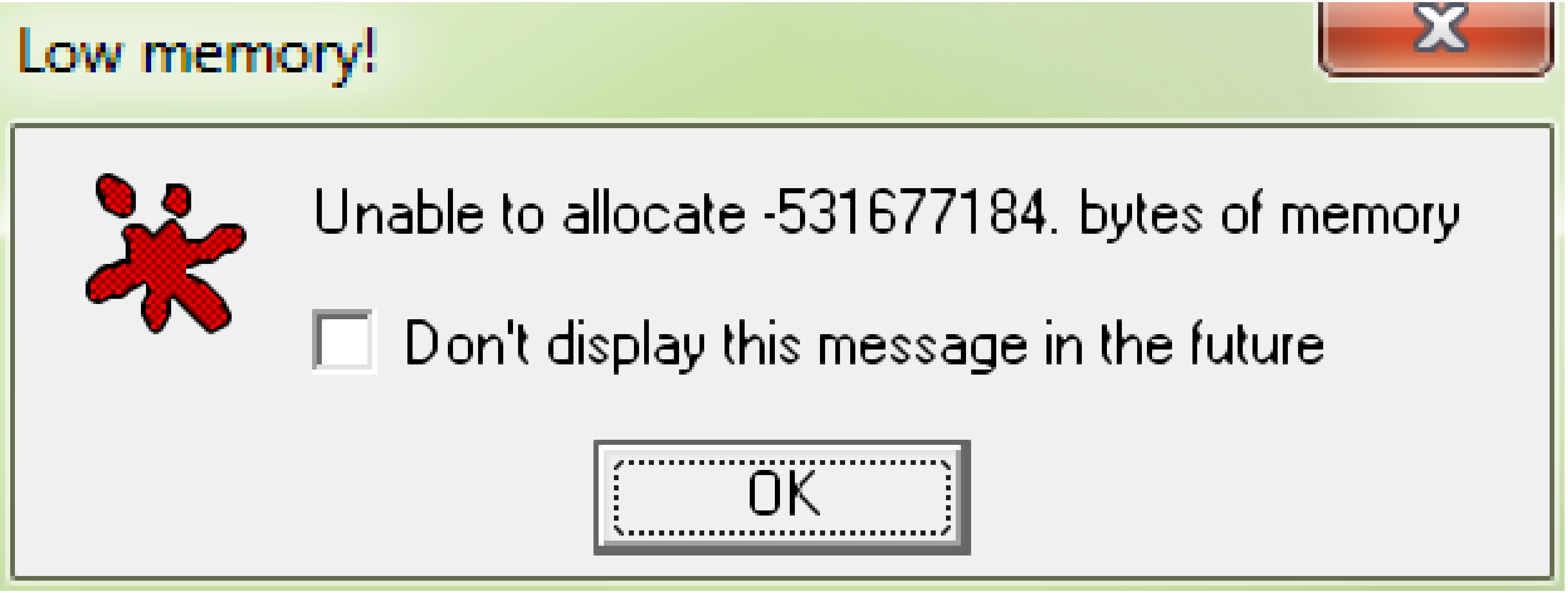
Weird export names

- exports = <anything non null>, 0



```
00401000: 6A01                                 push
00401002: 58                                   pop
00401000: 8BFF                               → retn
00401000: 8BFF                               → int
00401000: 8BFF                               → push
*********************************            → call
* Insert subliminal message here *           → add
*********************************            → retn ;
00401000: 8BFF                               → int
00401018: 202A                                1and
```

65535 sections vs OllyDbg



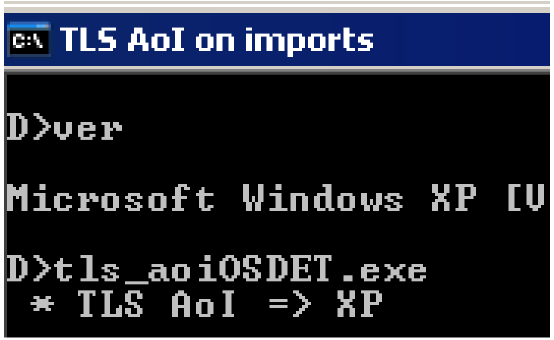
Low memory!

Unable to allocate -531677184. bytes of memory

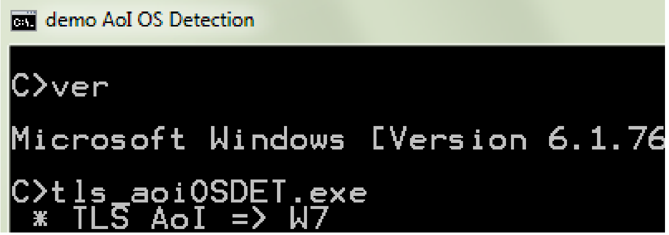
□ Don't display this message in the future

a last one...

- TLS AddressOfIndex is overwritten on loading
- Imports are parsed until Name is 0
- under XP, overwritten after imports
  - imports are fully parsed
- under W7, before
  - truncated



```
D>ver

Microsoft Windows XP [V

D>tls_aoiOSDET.exe
 * TLS AoI => XP
```



```
C>ver

Microsoft Windows [Version 6.1.76

C>tls_aoiOSDET.exe
 * TLS AoI => W7
```

same PE, loaded differently

Conclusion (1/2)

- x86 and PE are far from perfectly documented

official docs ⇒ FAIL

Conclusion (2/2)

1. visit Corkami
2. download the PoCs
   - read the doc / source
3. fix the bugs ;)
   - or answer my bug reports ?#$!

Acknowledgments

- Peter Ferrie
- Ivanlef0u

Adam Błaszczyk, BeatriX, Bruce Dang, Candid Wüest, Cathal Mullaney, Czerno, Daniel Reynaud, Elias Bachaalany, Ero Carrera, Eugeny Suslikov, Georg Wicherski, Gil Dabah, Guillaume Delugré, Gunther, Igor Skochinsky, Ilfak Guilfanov, Ivanlef0u, Jean-Baptiste Bédrune, Jim Leonard, Jon Larimer, Joshua J. Drake, Markus Hinderhofer, Mateusz Jurczyk, Matthieu Bonetti, Moritz Kroll, Oleh Yuschuk, Renaud Tabary, Rewolf, Sebastian Biallas, StalkR, Yoann Guillot,...

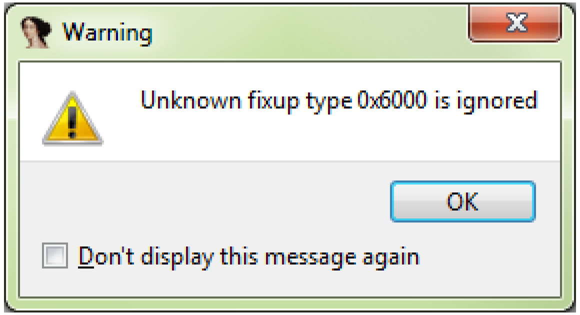
Questions?

Thank YOU!

@ange4771

Bonus

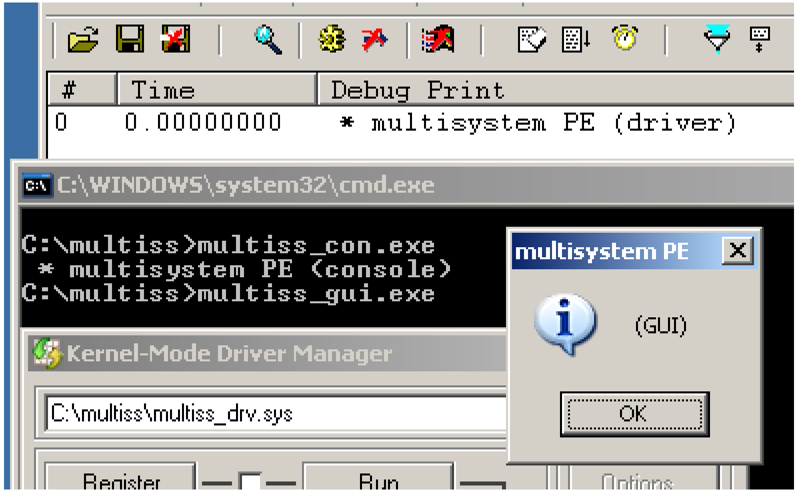
- Mips relocs (on relocs)
- ImageBase reloc
- multi-subsystem PE
- regs on TLS & DllMain



Warning

Unknown fixup type 0x6000 is ignored

□ Don't display this message again



```
#   Time         Debug Print
0   0.00000000   * multisystem PE (driver)
```

C:\WINDOWS\system32\cmd.exe

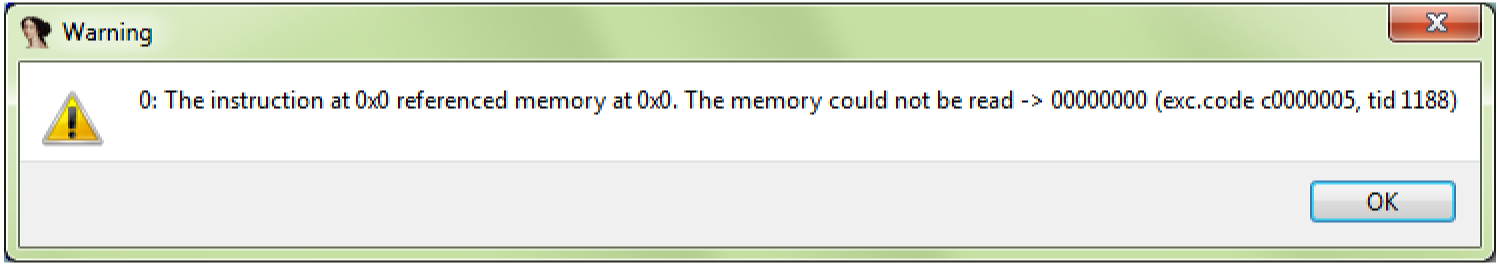
```
C:\multiss>multiss_con.exe
 * multisystem PE (console)
C:\multiss>multiss_gui.exe
```

Kernel-Mode Driver Manager

C:\multiss\multiss_drv.sys

multisystem PE

(GUI)



Warning

0: The instruction at 0x0 referenced memory at 0x0. The memory could not be read -> 00000000 (exc.code c0000005, tid 1188)