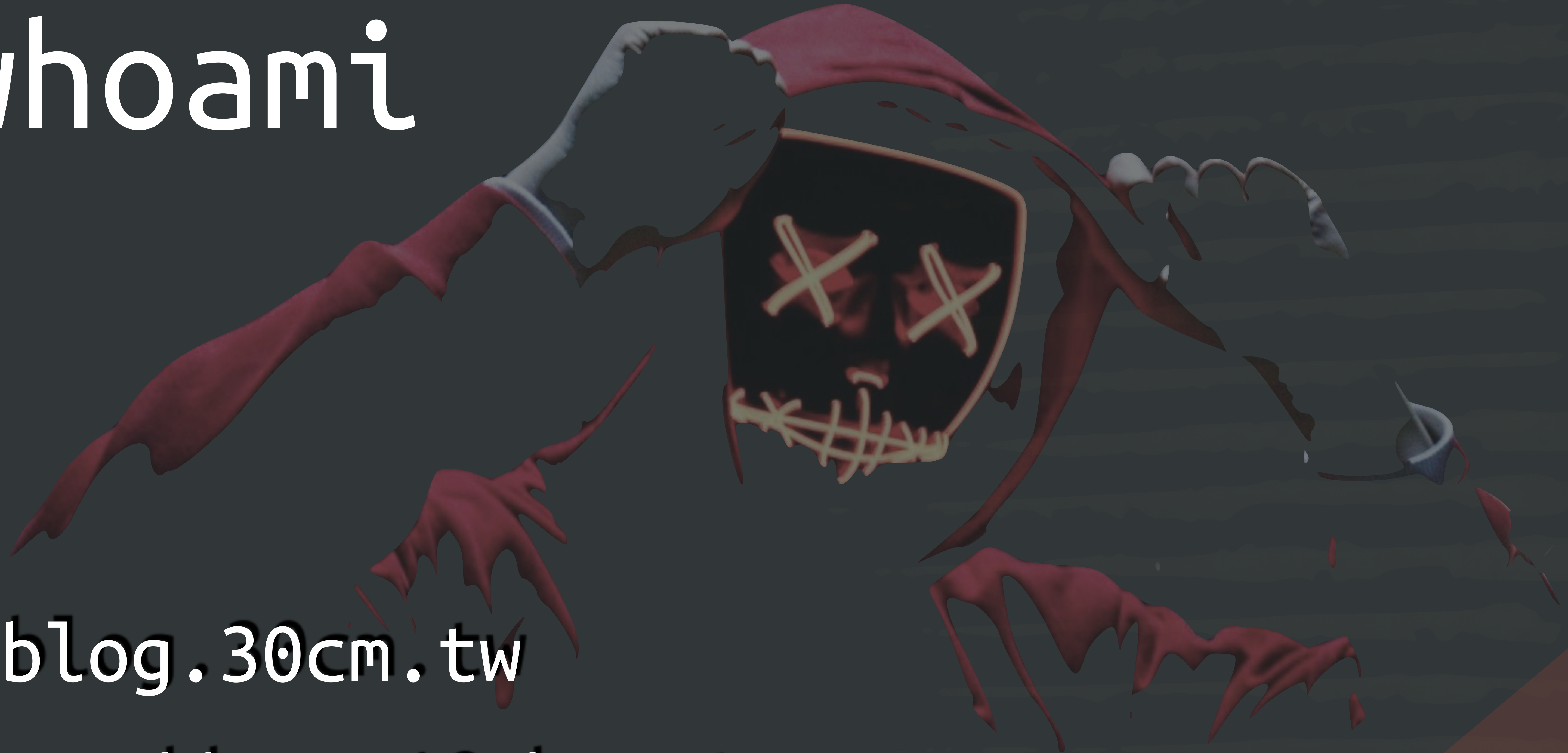
Windows Reversing Basic

aaaddress1@chroot.org

台灣科技大學 資安實務

$_whoami

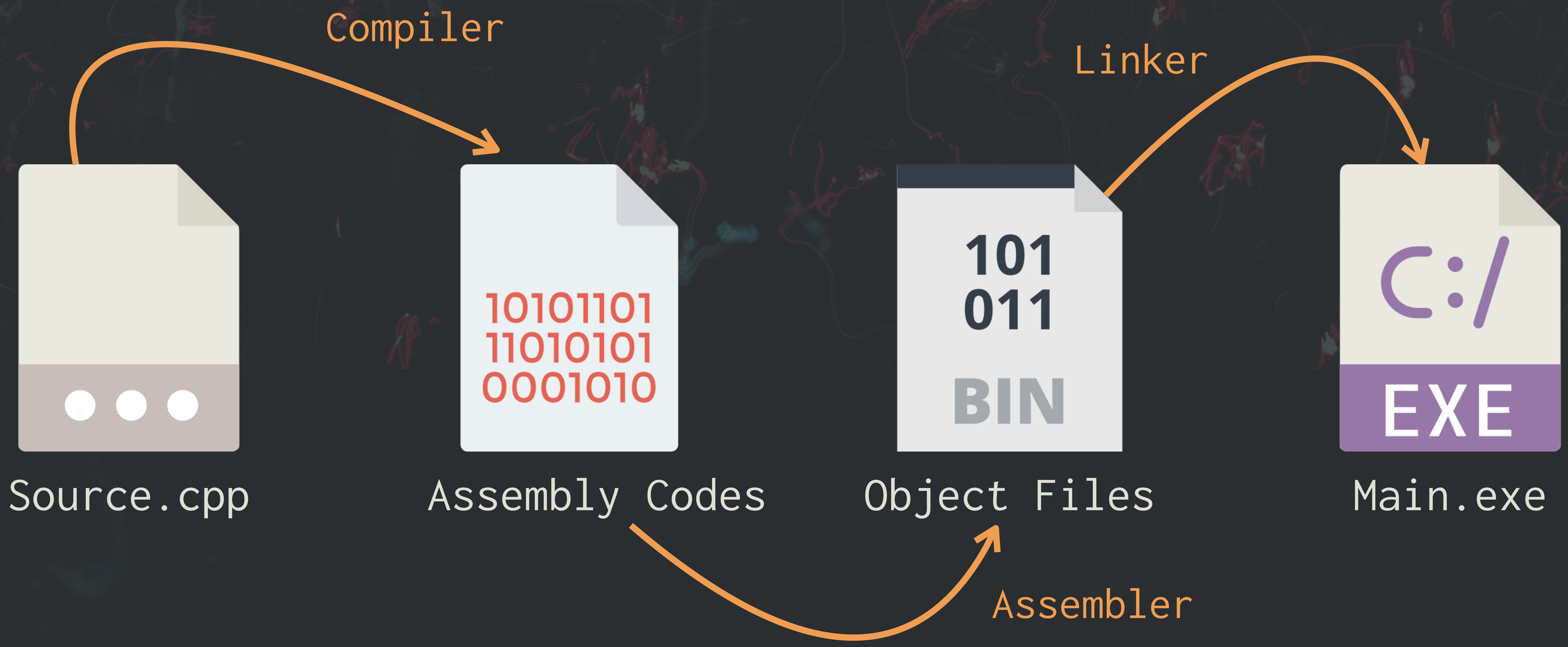


- blog.30cm.tw
- aaaddress1@chroot.org
- Master degree at CSIE, NTUST
- **Security Researcher** - chr0.ot
- **Speaker** - BlackHat, DEFCON, VXCON, HITCON
- aaaddress1@chroot.org

#Windows #Reversing #Pwn #Exploit #EoP

cd Compiler

# compiler


Compiler
Linker
10101101
11010101
0001010
101
011
BIN
C:/
EXE
Source.cpp
Assembly Codes
Object Files
Main.exe
Assembler

# compiler

```
#include <Windows.h>
int main()
{
    MessageBoxA(
        0, "hi there.", "info", 0
    );
   return 0;
}
```

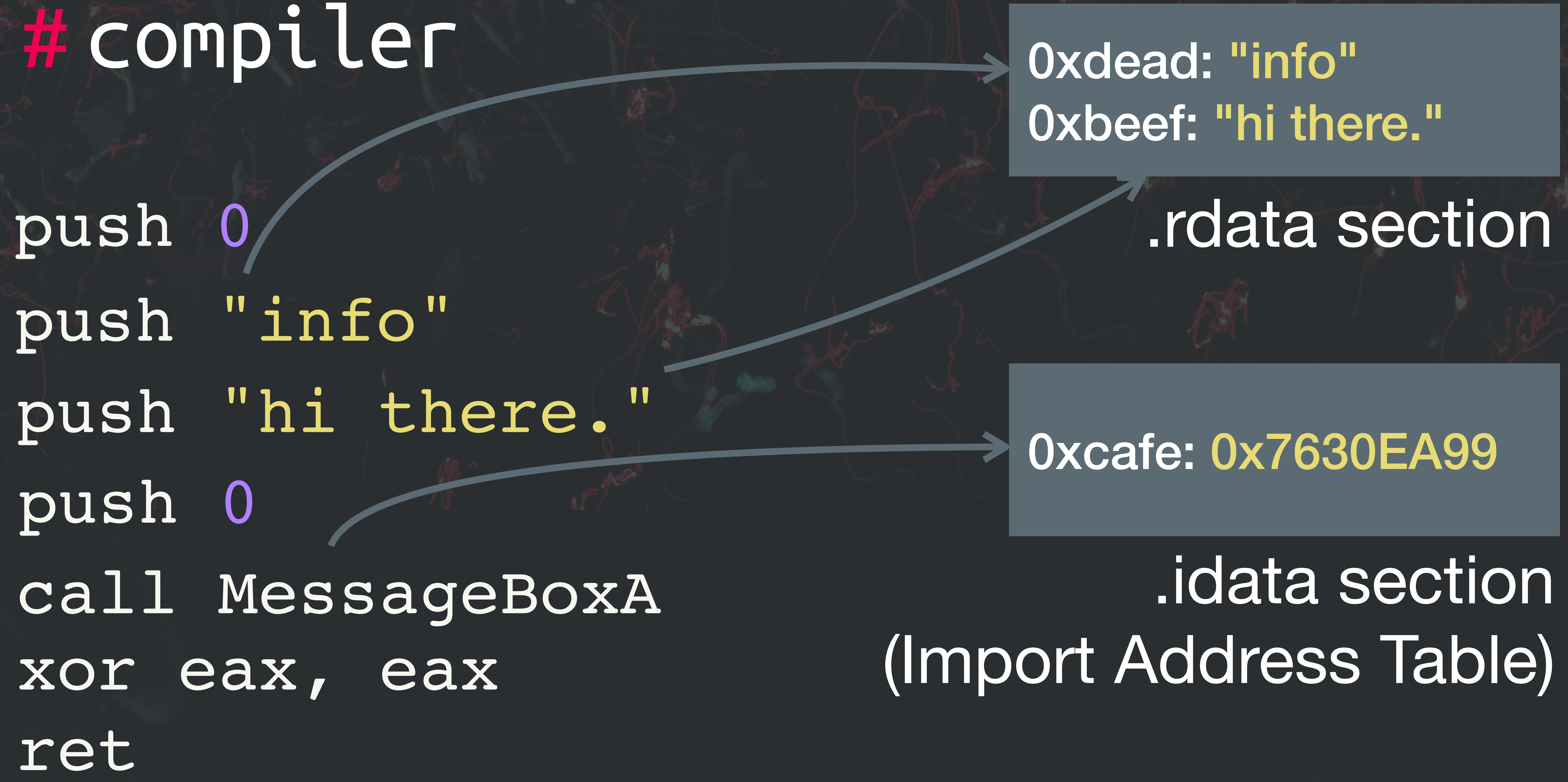
compiler

```
#include <Windows.h>
int main() {
  MessageBoxA(
   0,
   "hi there.",
   "info", 0
  );
  return 0;
}
```

```
push 0
push "info"
push "hi there."
push 0
call MessageBoxA
xor eax, eax
ret
```

en.wikipedia.org/wiki/X86_calling_conventions

#compiler



```
push 0
push "info"
push "hi there."
push 0
call MessageBoxA
xor eax, eax
ret
```

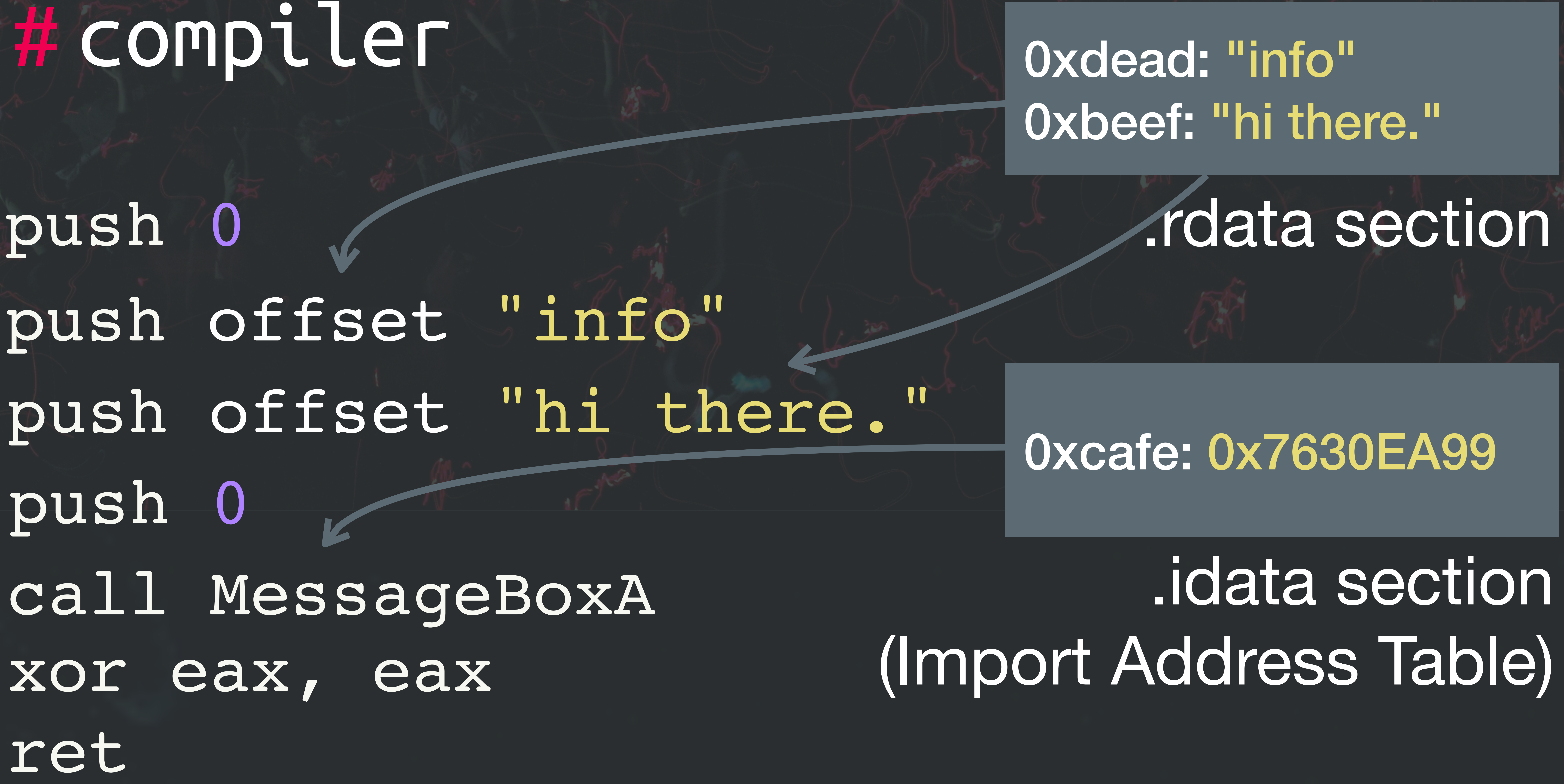
0xdead: "info"
0xbeef: "hi there."

.rdata section

0xcafe: 0x7630EA99

.idata section
(Import Address Table)

compiler



```
push 0
push offset "info"
push offset "hi there."
push 0
call MessageBoxA
xor eax, eax
ret
```

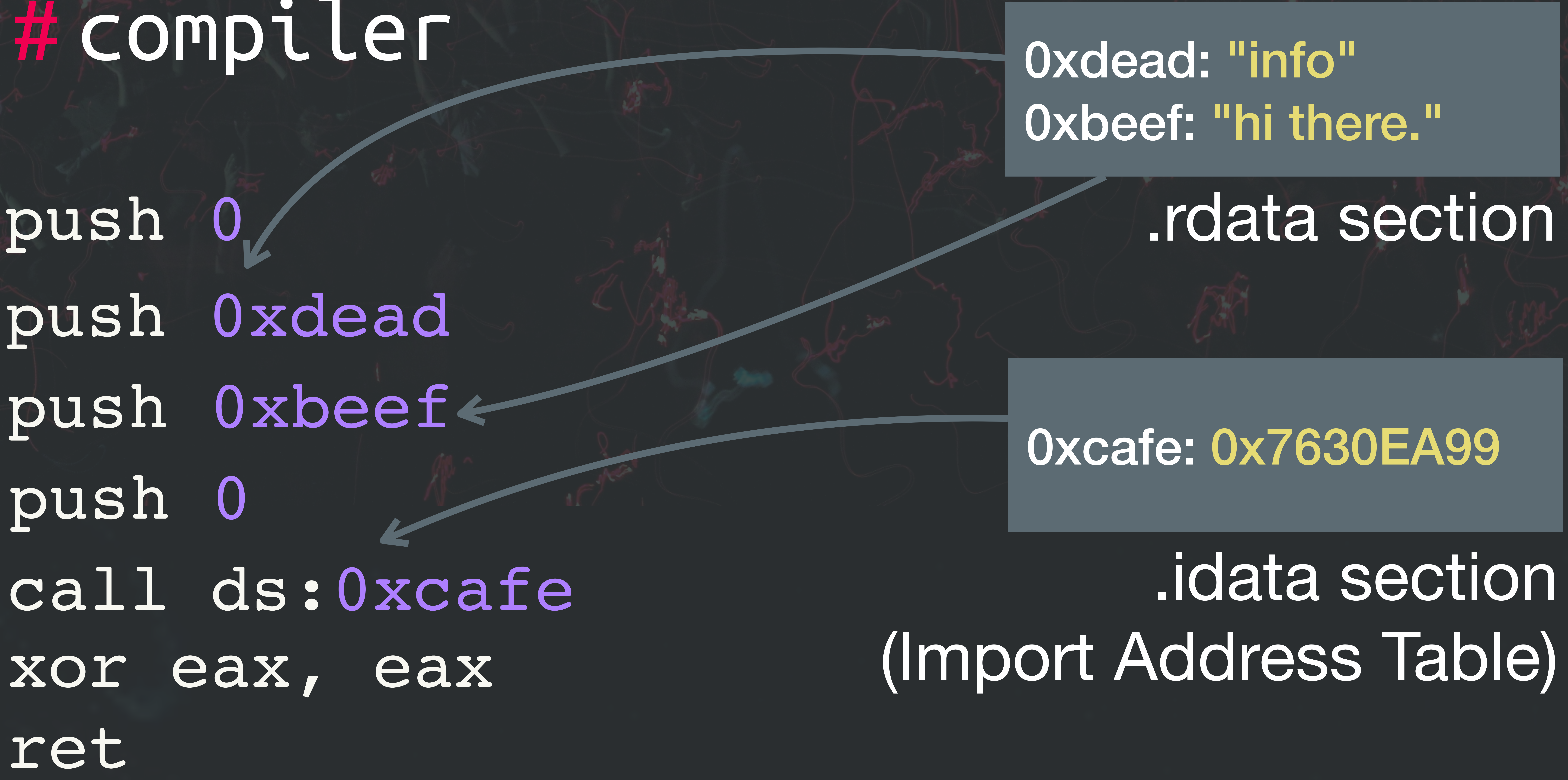
0xdead: "info"
0xbeef: "hi there."

.rdata section

0xcafe: 0x7630EA99

.idata section
(Import Address Table)

compiler



```
push 0
push 0xdead
push 0xbeef
push 0
call ds:0xcafe
xor eax, eax
ret
```

0xdead: "info"
0xbeef: "hi there."

.rdata section

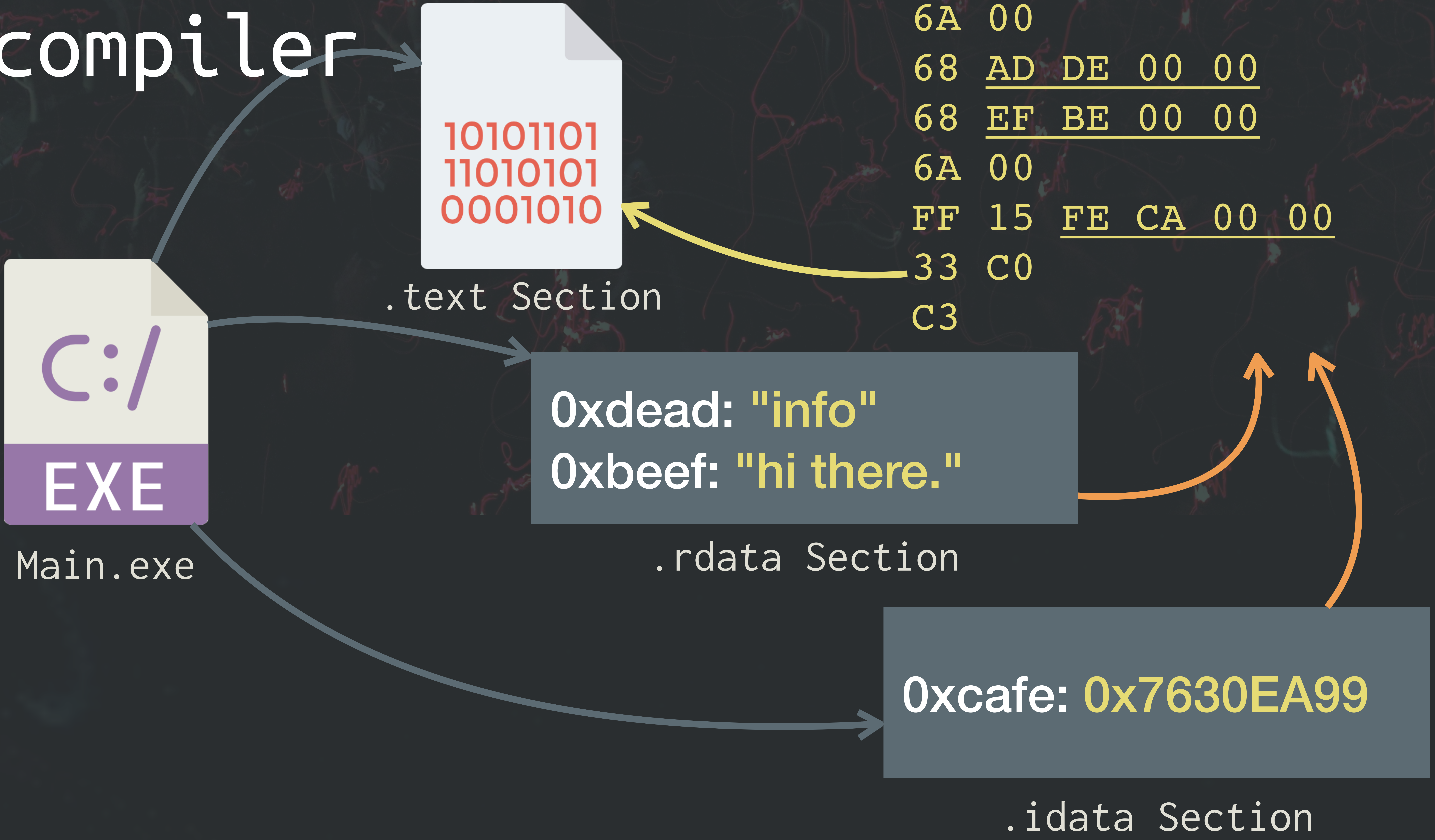
0xcafe: 0x7630EA99

.idata section
(Import Address Table)

compiler

```
push    0         ; 6A 00
push    0xdead    ; 68 AD DE 00 00
push    0xbeef    ; 68 EF BE 00 00
push    0         ; 6A 00
call    ds:0xcafe ; FF 15 FE CA 00 00
xor     eax, eax  ; 33 C0
ret               ; C3
```

compiler



.text Section

```
6A 00
68 AD DE 00 00
68 EF BE 00 00
6A 00
FF 15 FE CA 00 00
33 C0
C3
```

Main.exe

0xdead: "info"
0xbeef: "hi there."

.rdata Section

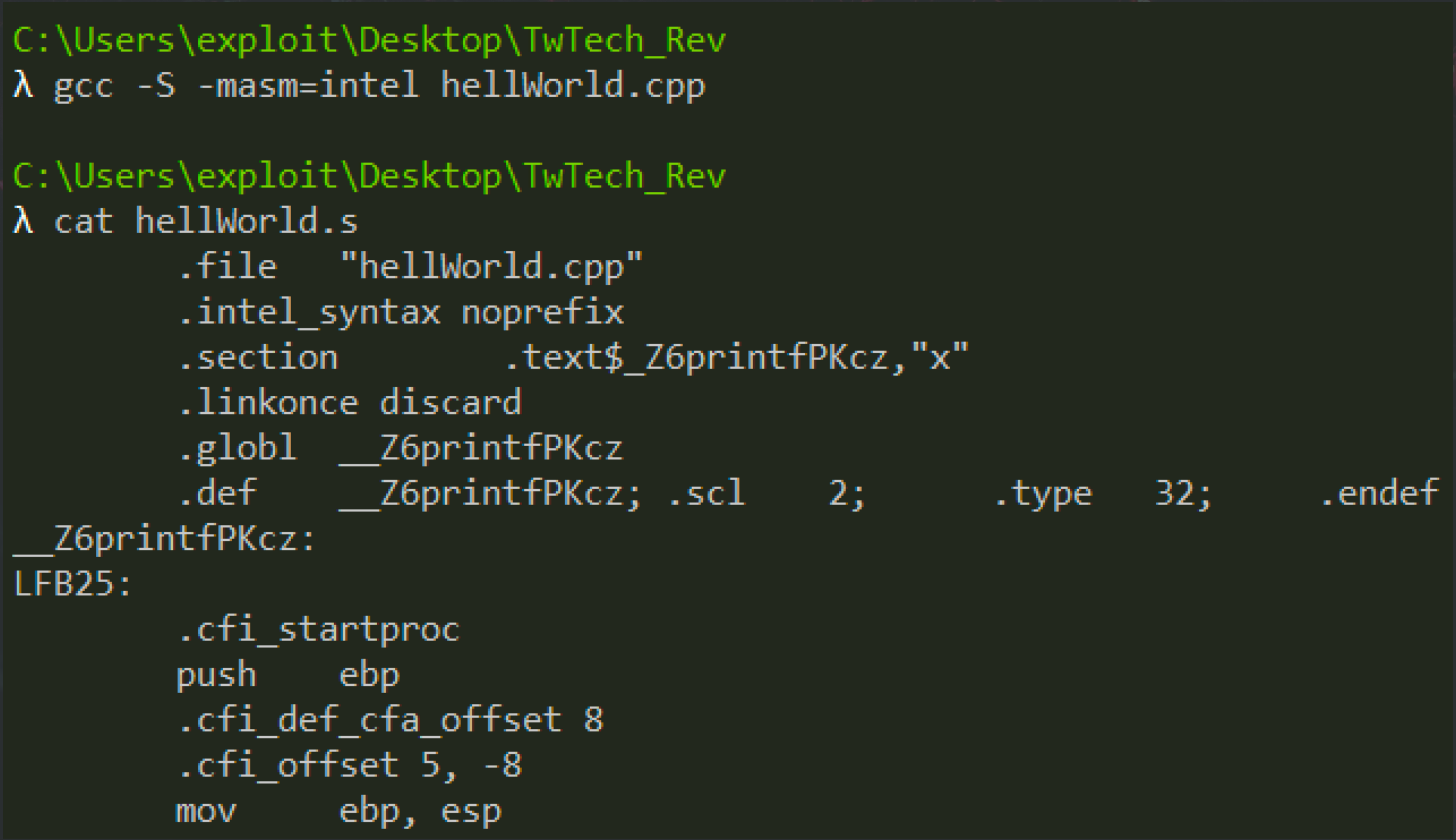
0xcafe: 0x7630EA99

.idata Section

```
cd Hell_World.c
```

# Compiler

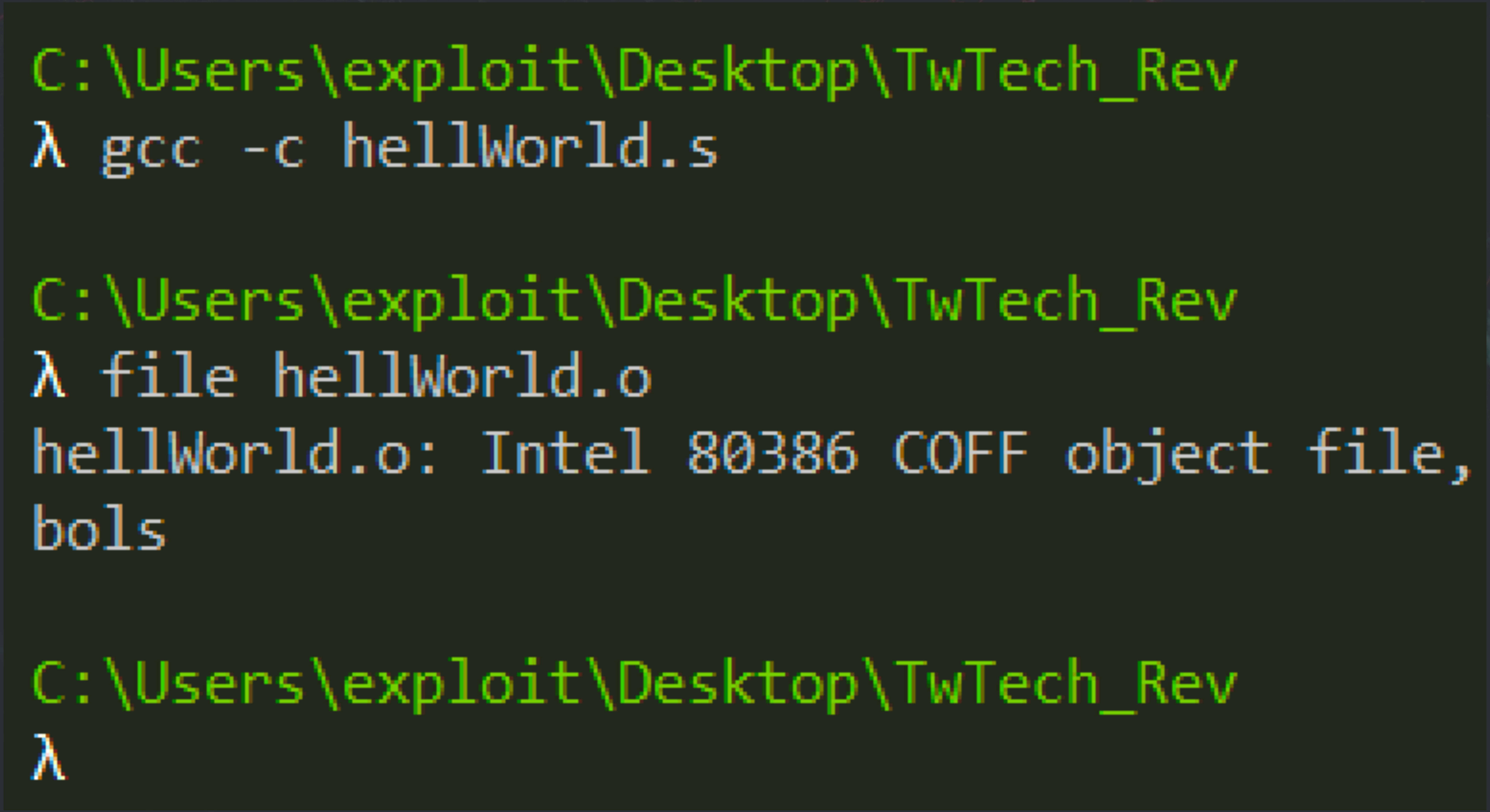
$ gcc -S hellWorld.cpp



```
C:\Users\exploit\Desktop\TwTech_Rev
λ gcc -S -masm=intel hellWorld.cpp

C:\Users\exploit\Desktop\TwTech_Rev
λ cat hellWorld.s
        .file   "hellWorld.cpp"
        .intel_syntax noprefix
        .section        .text$_Z6printfPKcz,"x"
        .linkonce discard
        .globl  __Z6printfPKcz
        .def    __Z6printfPKcz; .scl    2;      .type   32;     .endef
__Z6printfPKcz:
LFB25:
        .cfi_startproc
        push    ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        mov     ebp, esp
```

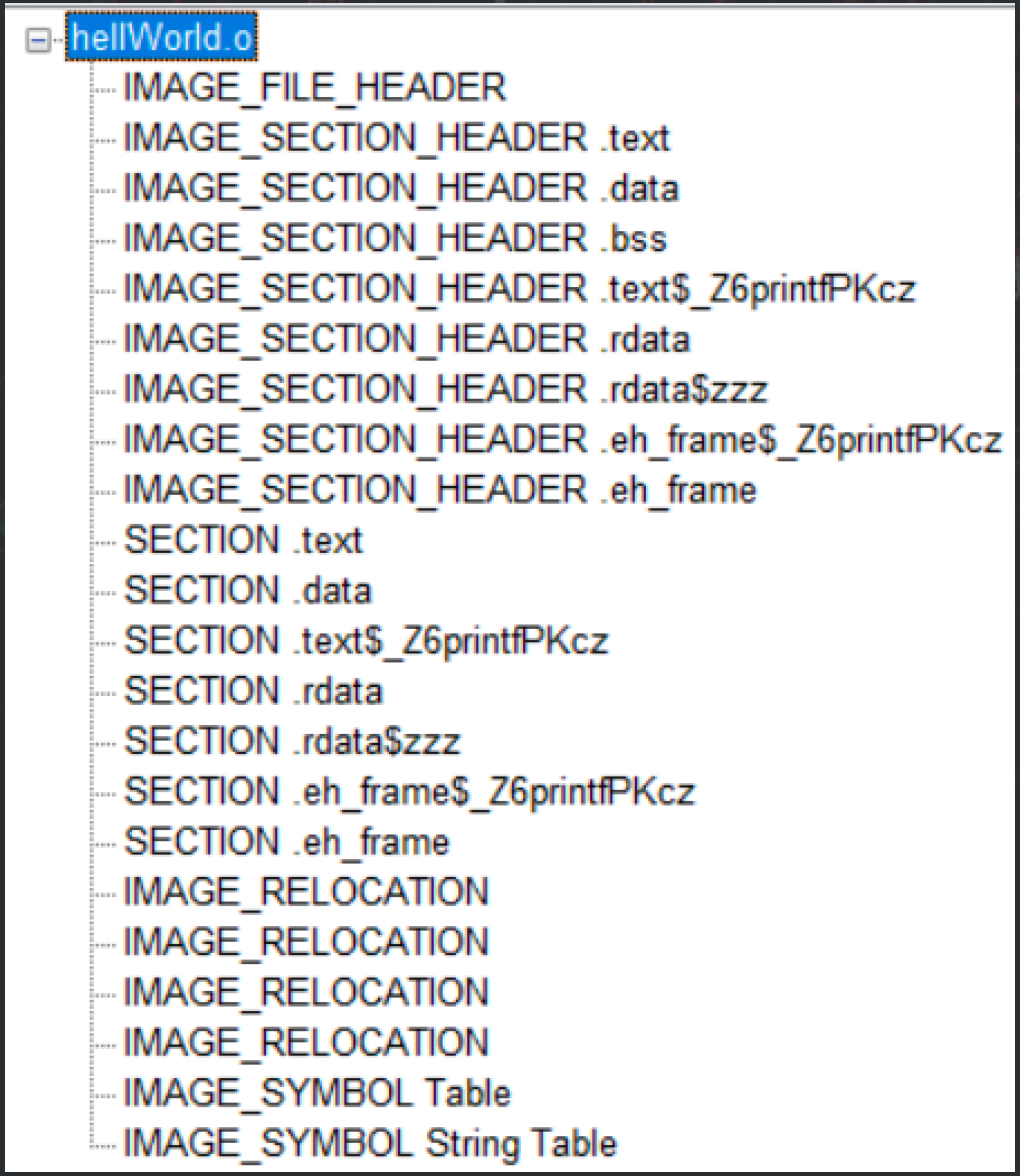
# Assembler

$ gcc -O0 -c hellWorld.s



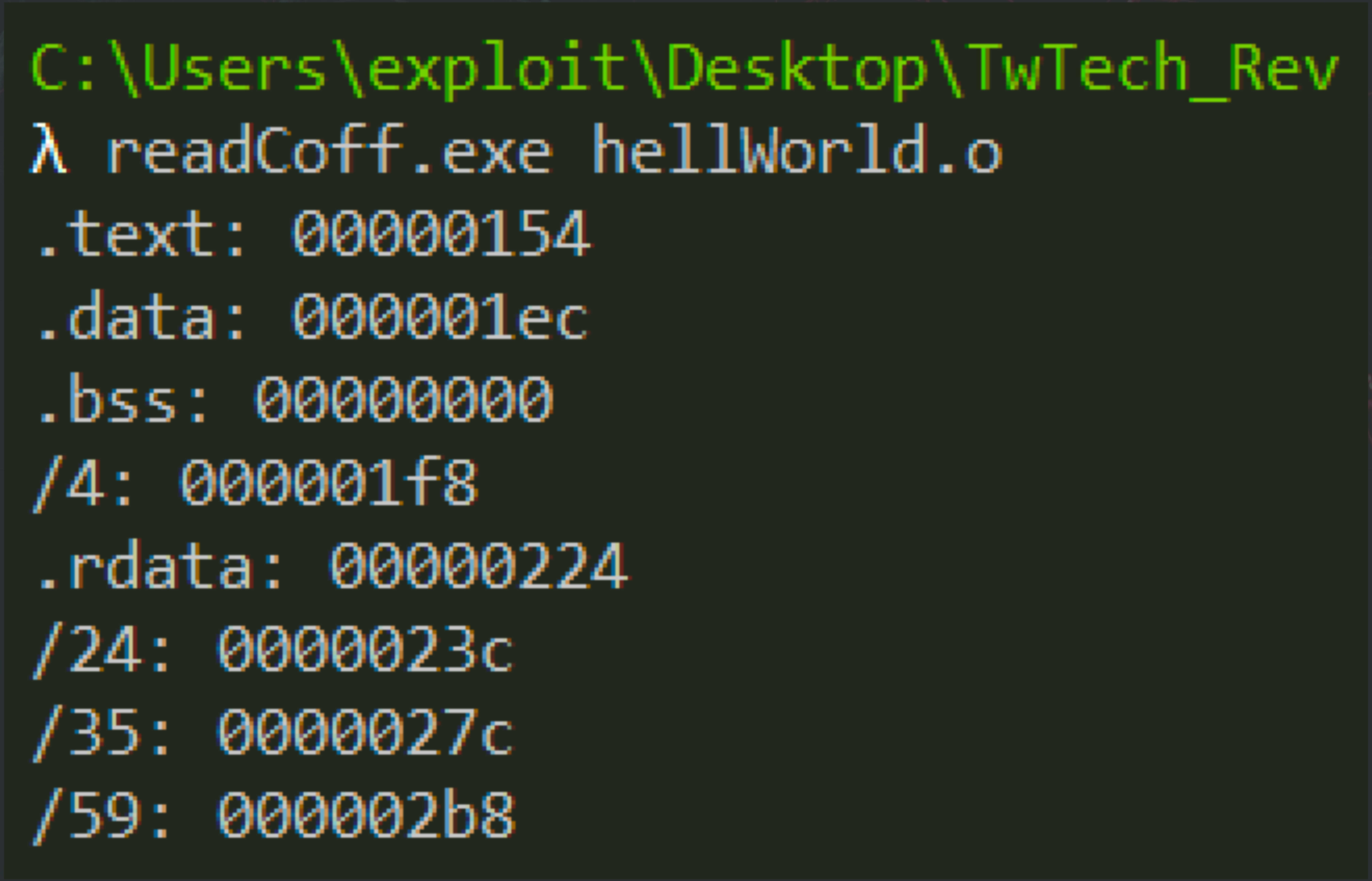
```
C:\Users\exploit\Desktop\TwTech_Rev
λ gcc -c hellWorld.s

C:\Users\exploit\Desktop\TwTech_Rev
λ file hellWorld.o
hellWorld.o: Intel 80386 COFF object file,
bols

C:\Users\exploit\Desktop\TwTech_Rev
λ
```

# COFF File?



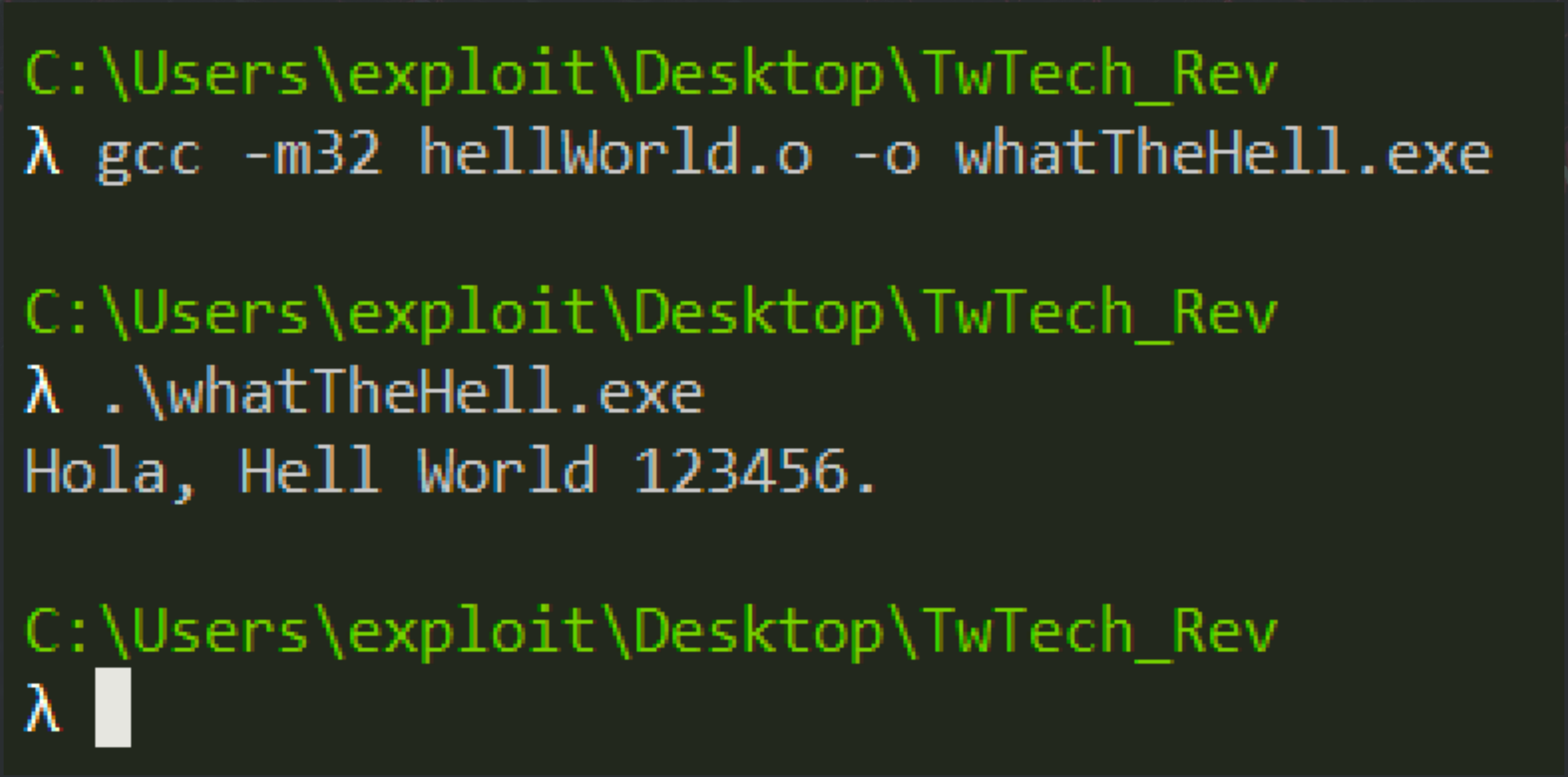
```
⊟ hellWorld.o
    IMAGE_FILE_HEADER
    IMAGE_SECTION_HEADER .text
    IMAGE_SECTION_HEADER .data
    IMAGE_SECTION_HEADER .bss
    IMAGE_SECTION_HEADER .text$_Z6printfPKcz
    IMAGE_SECTION_HEADER .rdata
    IMAGE_SECTION_HEADER .rdata$zzz
    IMAGE_SECTION_HEADER .eh_frame$_Z6printfPKcz
    IMAGE_SECTION_HEADER .eh_frame
    SECTION .text
    SECTION .data
    SECTION .text$_Z6printfPKcz
    SECTION .rdata
    SECTION .rdata$zzz
    SECTION .eh_frame$_Z6printfPKcz
    SECTION .eh_frame
    IMAGE_RELOCATION
    IMAGE_RELOCATION
    IMAGE_RELOCATION
    IMAGE_RELOCATION
    IMAGE_SYMBOL Table
    IMAGE_SYMBOL String Table
```

# COFF File



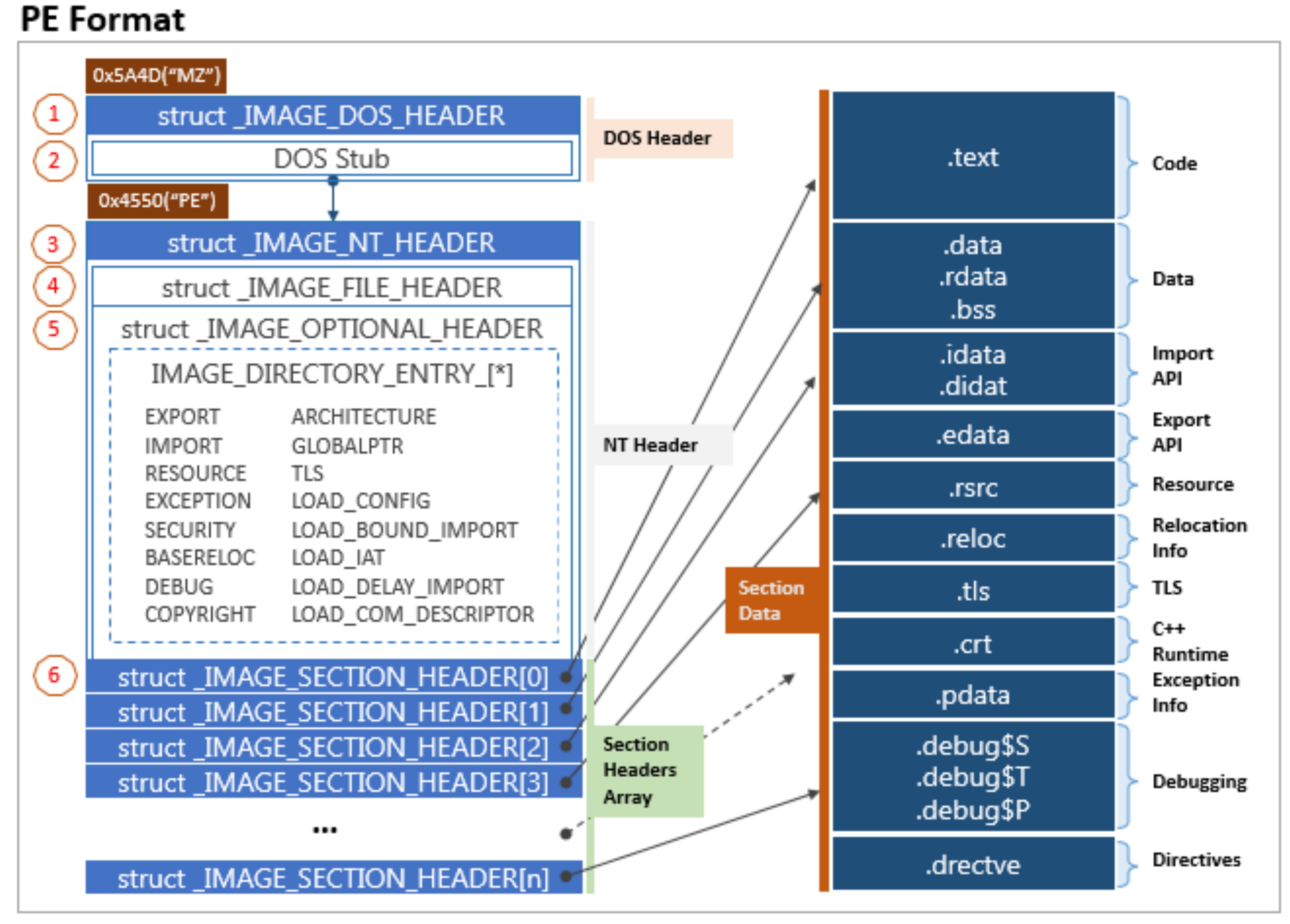
```
C:\Users\exploit\Desktop\TwTech_Rev
λ readCoff.exe hellWorld.o
.text: 00000154
.data: 000001ec
.bss: 00000000
/4: 000001f8
.rdata: 00000224
/24: 0000023c
/35: 0000027c
/59: 000002b8
```

# Linker



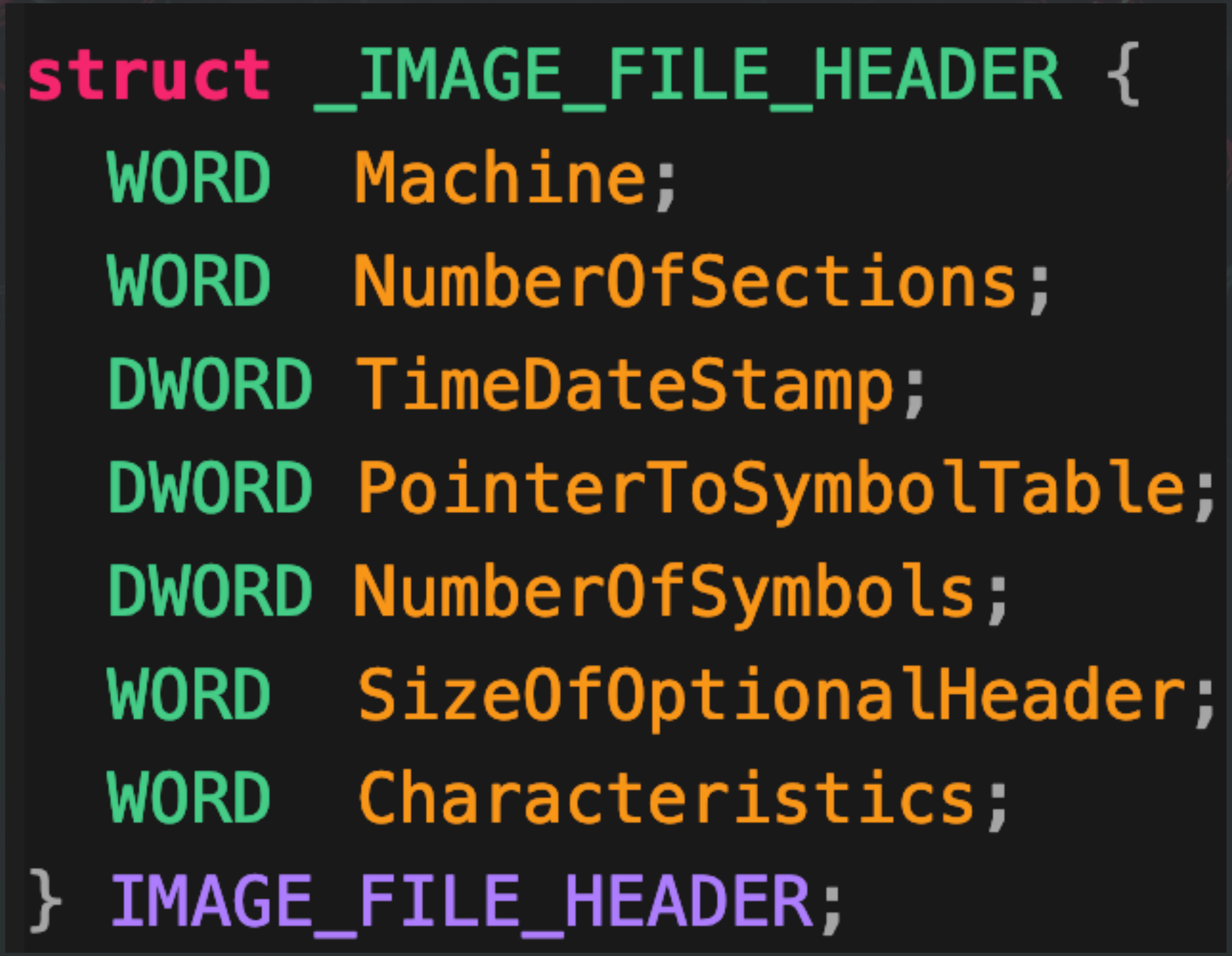
```
C:\Users\exploit\Desktop\TwTech_Rev
λ gcc -m32 hellWorld.o -o whatTheHell.exe

C:\Users\exploit\Desktop\TwTech_Rev
λ .\whatTheHell.exe
Hola, Hell World 123456.

C:\Users\exploit\Desktop\TwTech_Rev
λ █
```

# Linker

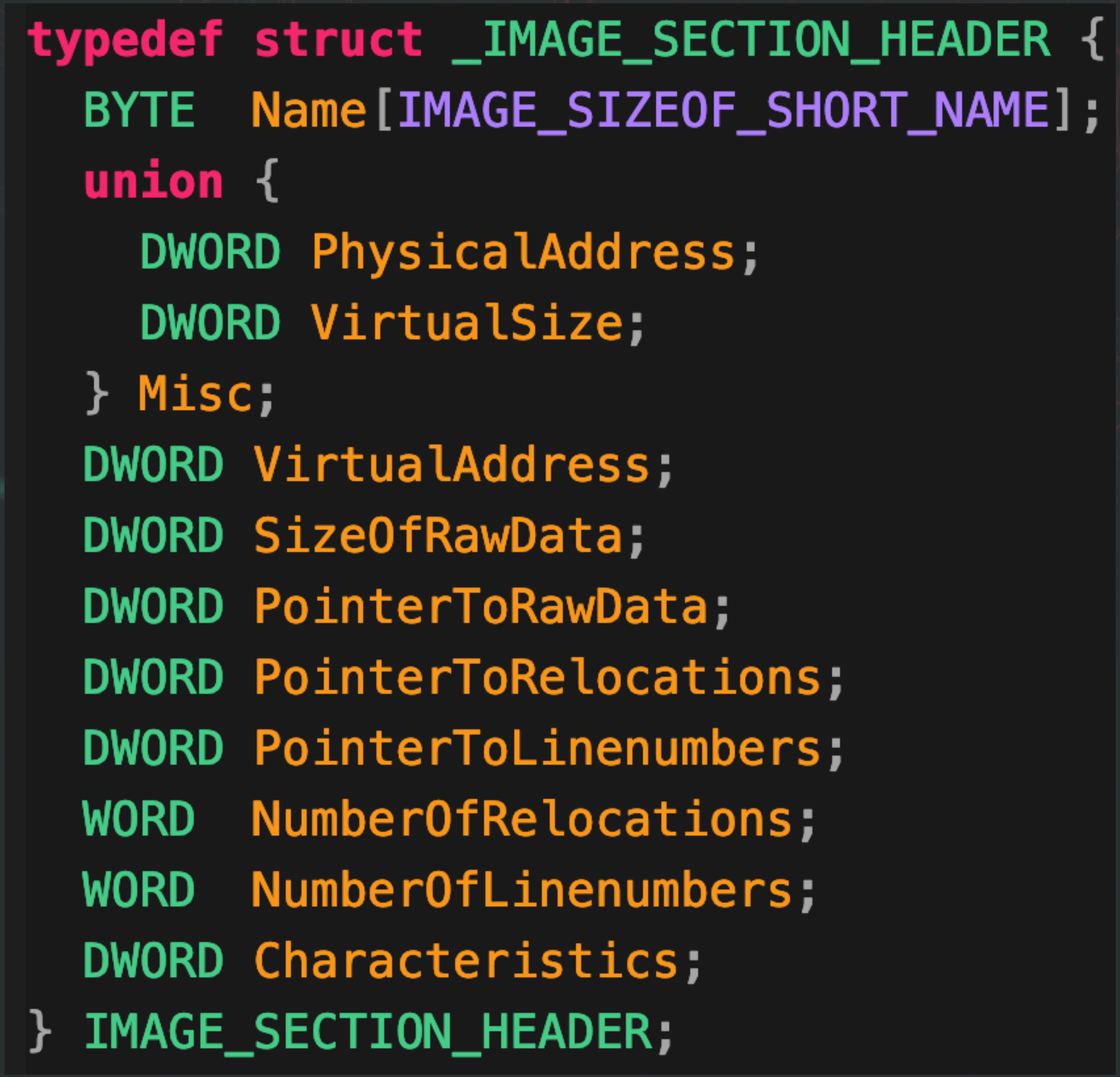

PE Format
0x5A4D("MZ")
1
struct _IMAGE_DOS_HEADER
2
DOS Stub
DOS Header
0x4550("PE")
3
struct _IMAGE_NT_HEADER
4
struct _IMAGE_FILE_HEADER
5
struct _IMAGE_OPTIONAL_HEADER
IMAGE_DIRECTORY_ENTRY_[*]
EXPORT
IMPORT
RESOURCE
EXCEPTION
SECURITY
BASERELOC
DEBUG
COPYRIGHT
ARCHITECTURE
GLOBALPTR
TLS
LOAD_CONFIG
LOAD_BOUND_IMPORT
LOAD_IAT
LOAD_DELAY_IMPORT
LOAD_COM_DESCRIPTOR
NT Header
6
struct _IMAGE_SECTION_HEADER[0]
struct _IMAGE_SECTION_HEADER[1]
struct _IMAGE_SECTION_HEADER[2]
struct _IMAGE_SECTION_HEADER[3]
...
struct _IMAGE_SECTION_HEADER[n]
Section Headers Array
Section Data
.text
Code
.data
.rdata
.bss
Data
.idata
.didat
Import API
.edata
Export API
.rsrc
Resource
.reloc
Relocation Info
.tls
TLS
.crt
C++ Runtime
.pdata
Exception Info
.debug$S
.debug$T
.debug$P
Debugging
.drectve
Directives

cat ./a.o

# COFF Overview



```
struct _IMAGE_FILE_HEADER {
  WORD   Machine;
  WORD   NumberOfSections;
  DWORD  TimeDateStamp;
  DWORD  PointerToSymbolTable;
  DWORD  NumberOfSymbols;
  WORD   SizeOfOptionalHeader;
  WORD   Characteristics;
} IMAGE_FILE_HEADER;
```



```
typedef struct _IMAGE_SECTION_HEADER {
  BYTE  Name[IMAGE_SIZEOF_SHORT_NAME];
  union {
    DWORD PhysicalAddress;
    DWORD VirtualSize;
  } Misc;
  DWORD VirtualAddress;
  DWORD SizeOfRawData;
  DWORD PointerToRawData;
  DWORD PointerToRelocations;
  DWORD PointerToLinenumbers;
  WORD  NumberOfRelocations;
  WORD  NumberOfLinenumbers;
  DWORD Characteristics;
} IMAGE_SECTION_HEADER;
```

cat ./a.o

# COFF Overview

.Machine

.NumberOfSections

.TimeDateStamp

.PointerToSymbolTable

.NumberOfSymbols

.Characteristics

Image File Header

.VA

.RVA

.size

.name

Section Header 1

Section Header 2

Section Header 3

...

Section Header Array →

Section Data

Section Data

cat ./PE

# PE Overview

.ImageBase
.SizeOfHeaders
.AddressOfEntryPoint
.DataDirectory
.SizeOfImage
.NumberOfSections
.text
.size
.RVA
.VA
Section Header 1
'MZ'
DOS
'PE'
File Headr
Opt Header
Section Header Array →
.text (Data)

cat ./PE

# COFF or PE

.ImageBase
.SizeOfHeaders
.AddressOfEntryPoint
.DataDirectory
.SizeOfImage
.NumberOfSections
.text
.size
.RVA
.VA
Section Header 1
'MZ'
DOS
'PE'
File Headr
Opt Header
Section Header Array →
.text (Data)

cd Win32 Process

Parent Process

(A.) CreateProcess

(B.) Child Proess Created,
EXE File Mapped, Gained the Same Privilege
and New Thread pointed to RtlUserThreadStart

(C.)
Kernel Create a new Thread:
RtlUserThreadStart
→LdrInitializeThunk
→LdrpInitializeProcess

Child Process

(D.) Jump into AddressOfEntry

Ring3
syscall
Ring0

lifecycle

Source.cpp

Assembly Codes

Object Files

Main.exe

File Mapping

Process

Stack Memory

PE Module

NT Header

File Header

Optional Header

Section Header Array

Section[0]: .text

Section[1]: .data

Section[2]: .rdata

. . .

[DATA] .text

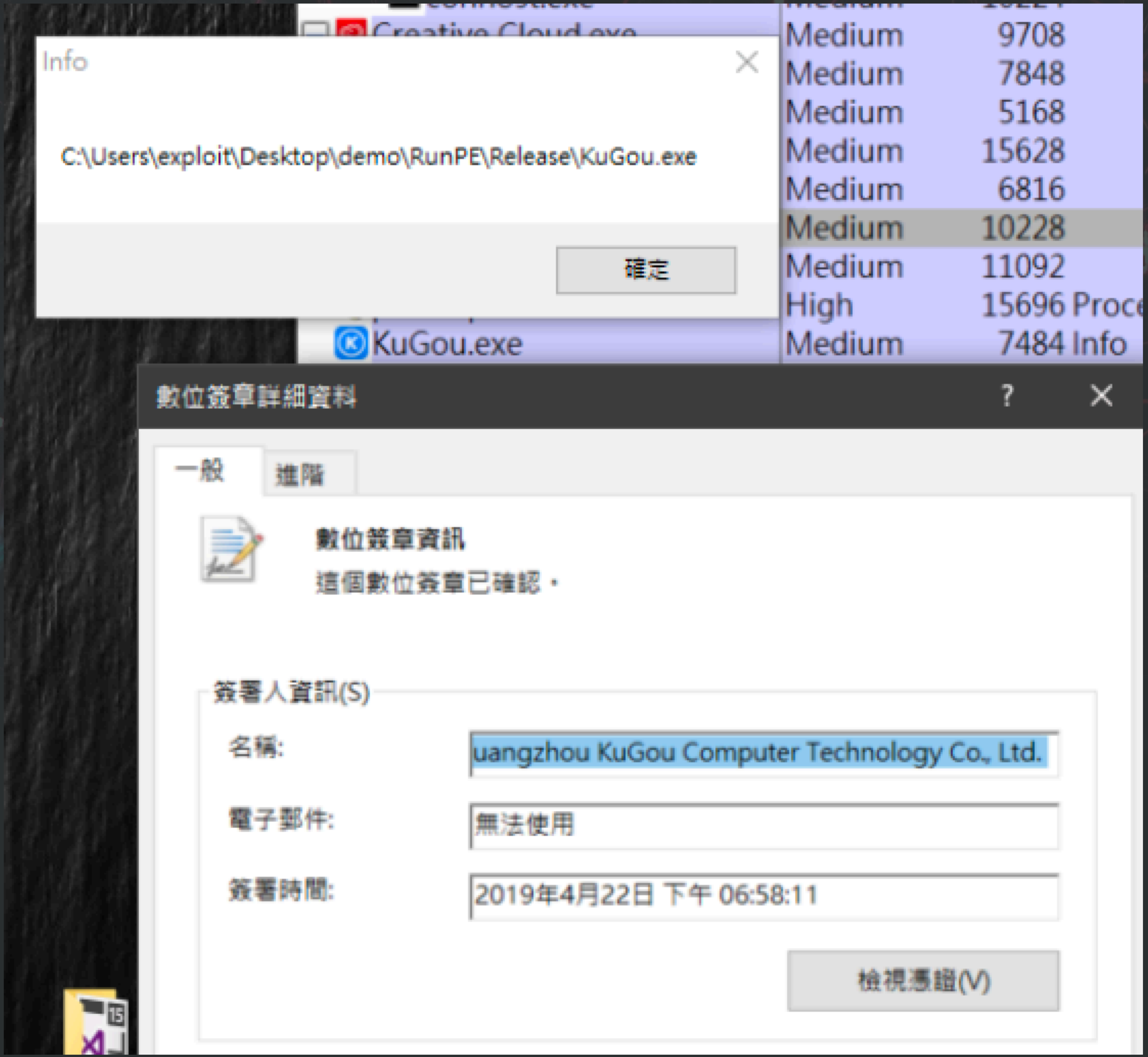
[DATA] .data

[DATA] .idata

10101101
11010101
0001010
101
011
BIN
C:/
EXE

cat ./RunPE

# Process Hollowing



```
Creative Cloud.exe   Medium   9708
                     Medium   7848
                     Medium   5168
                     Medium  15628
                     Medium   6816
                     Medium  10228
                     Medium  11092
                     High    15696 Proce
KuGou.exe            Medium   7484 Info
```

Info

C:\Users\exploit\Desktop\demo\RunPE\Release\KuGou.exe

確定

數位簽章詳細資料

一般　進階

數位簽章資訊

這個數位簽章已確認。

簽署人資訊(S)

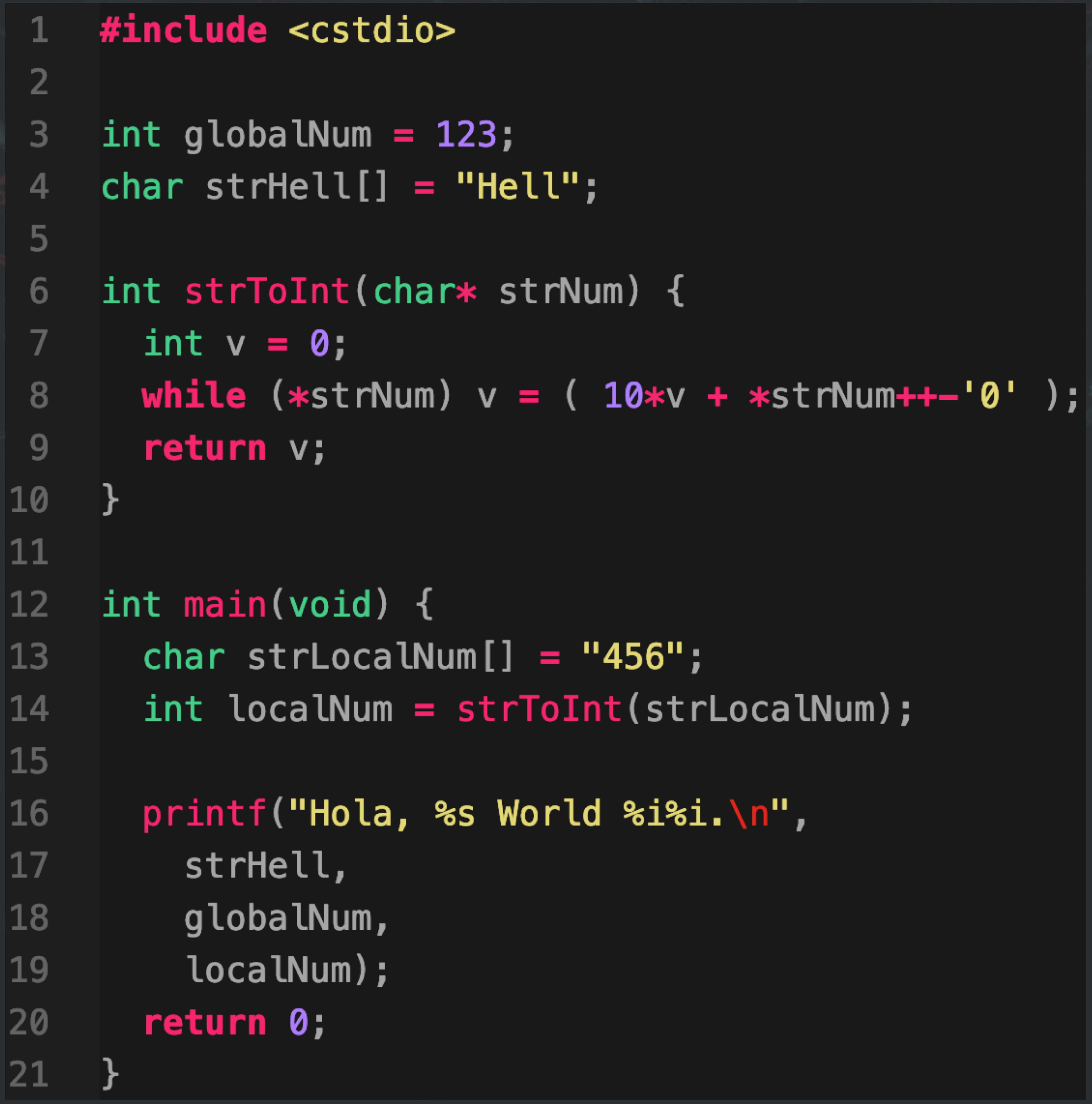
名稱: uangzhou KuGou Computer Technology Co., Ltd.

電子郵件: 無法使用

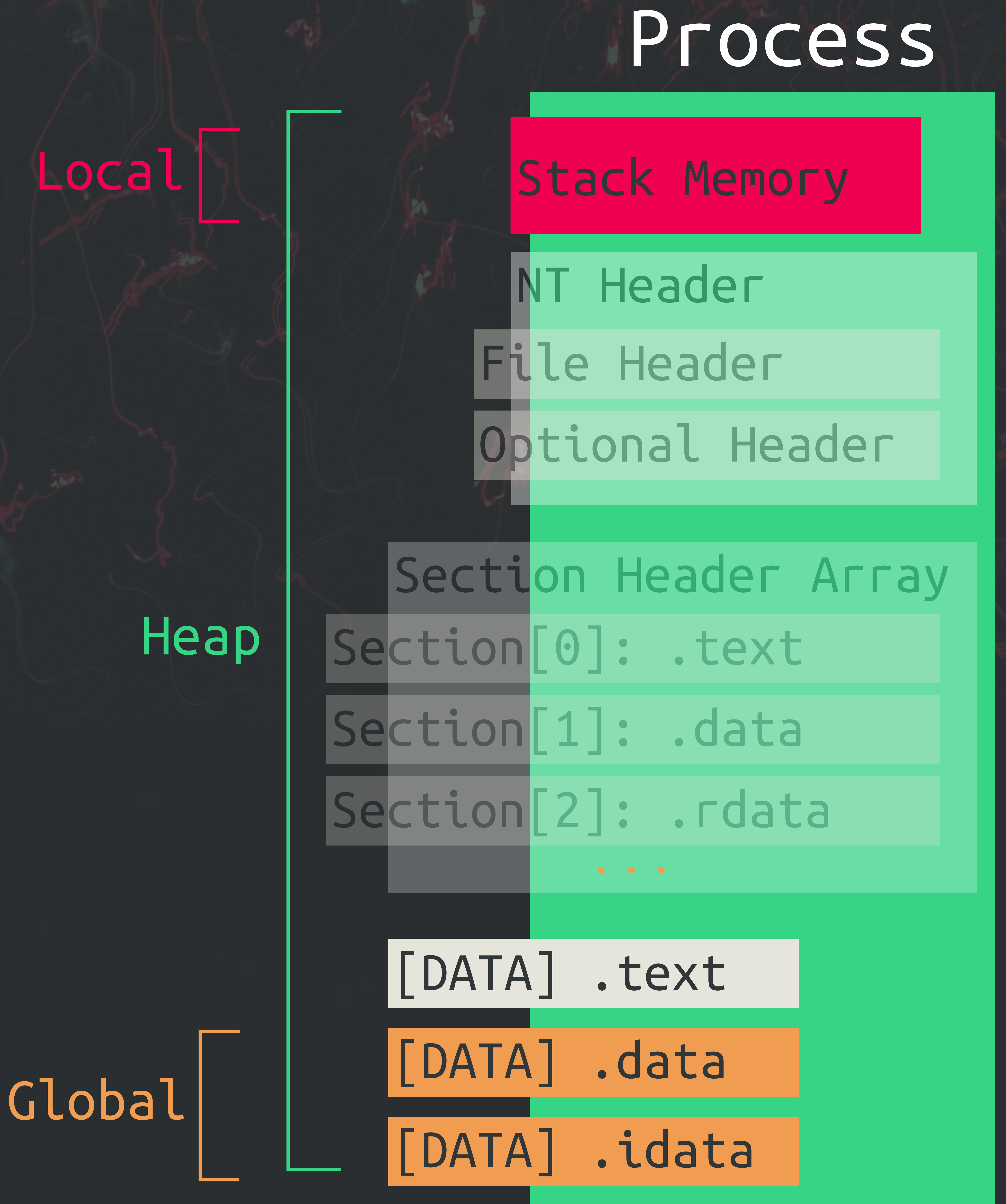
簽署時間: 2019年4月22日 下午 06:58:11

檢視憑證(V)

lifecycle

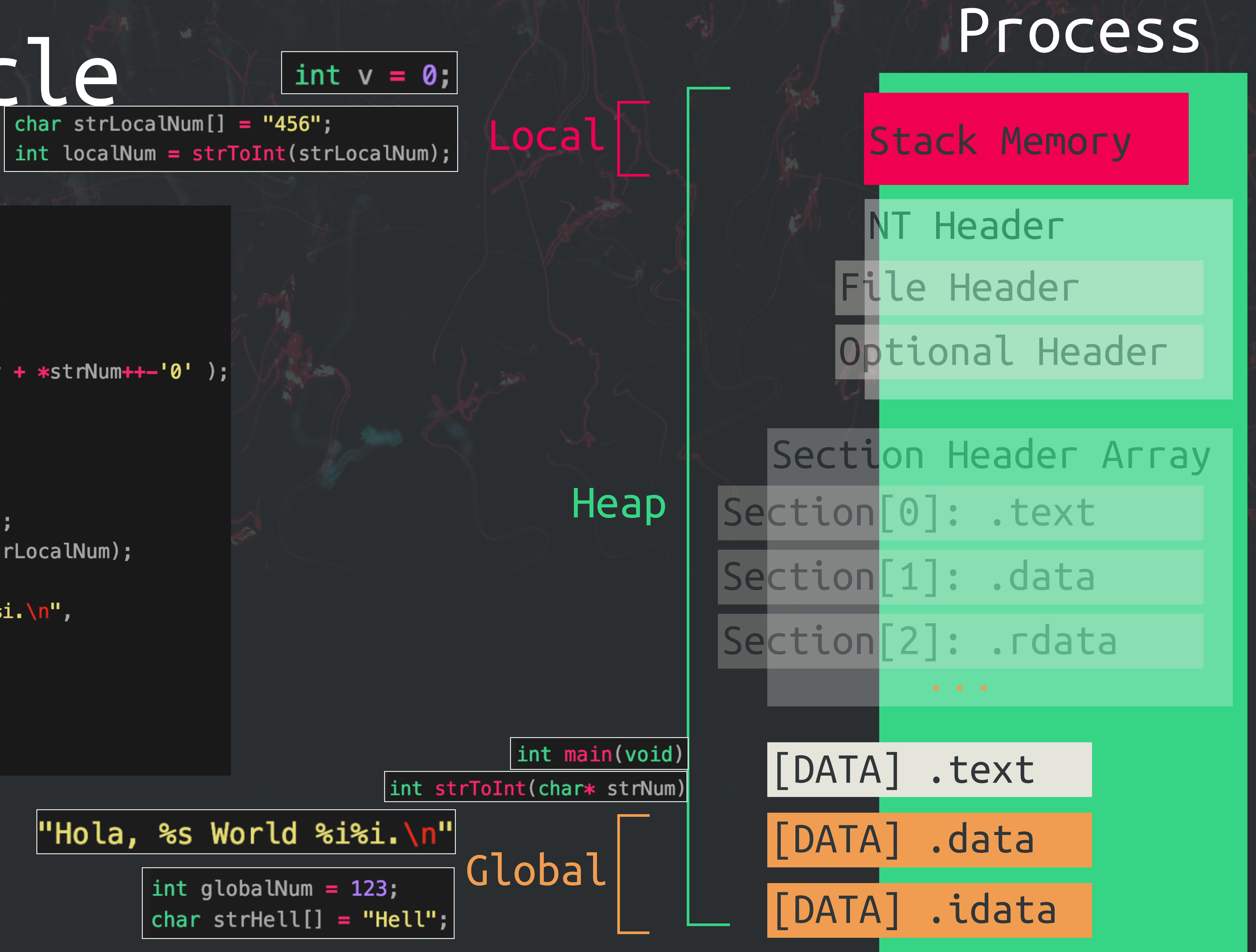


```
#include <cstdio>

int globalNum = 123;
char strHell[] = "Hell";

int strToInt(char* strNum) {
  int v = 0;
  while (*strNum) v = ( 10*v + *strNum++-'0' );
  return v;
}

int main(void) {
  char strLocalNum[] = "456";
  int localNum = strToInt(strLocalNum);

  printf("Hola, %s World %i%i.\n",
    strHell,
    globalNum,
    localNum);
  return 0;
}
```



lifecycle

```c
int globalNum = 123;
char strHell[] = "Hell";

int strToInt(char* strNum) {
  int v = 0;
  while (*strNum) v = ( 10*v + *strNum++-'0' );
  return v;
}

int main(void) {
  char strLocalNum[] = "456";
  int localNum = strToInt(strLocalNum);

  printf("Hola, %s World %i%i.\n",
    strHell,
    globalNum,
    localNum);
  return 0;
}
```



Process

```c
int v = 0;
```

```c
char strLocalNum[] = "456";
int localNum = strToInt(strLocalNum);
```

```c
int main(void)
int strToInt(char* strNum)
```

```c
"Hola, %s World %i%i.\n"
```

```c
int globalNum = 123;
char strHell[] = "Hell";
```

lifecycle

Process

Source.cpp

10101101
11010101
0001010

Assembly Codes

101
011
BIN

Object Files

C:/
EXE

Main.exe

File Mapping

new Thread

hellWorld.exe

[DATA] .text

[DATA] .idata

iat:imp_printf

iat:imp_WinExec

msvcrt.dll

[DATA] .text

kernel32.dll

[DATA] .text

module_a.dll

module_b.dll

lifecycle

我怎麼知道這一次是哪個模組的執行緒啦，森77。

分時多工

Thread[0]

Thread[1]

Thread[2]

Process

hellWorld.exe

[DATA] .text

[DATA] .idata

msvcrt.dll

[DATA] .text

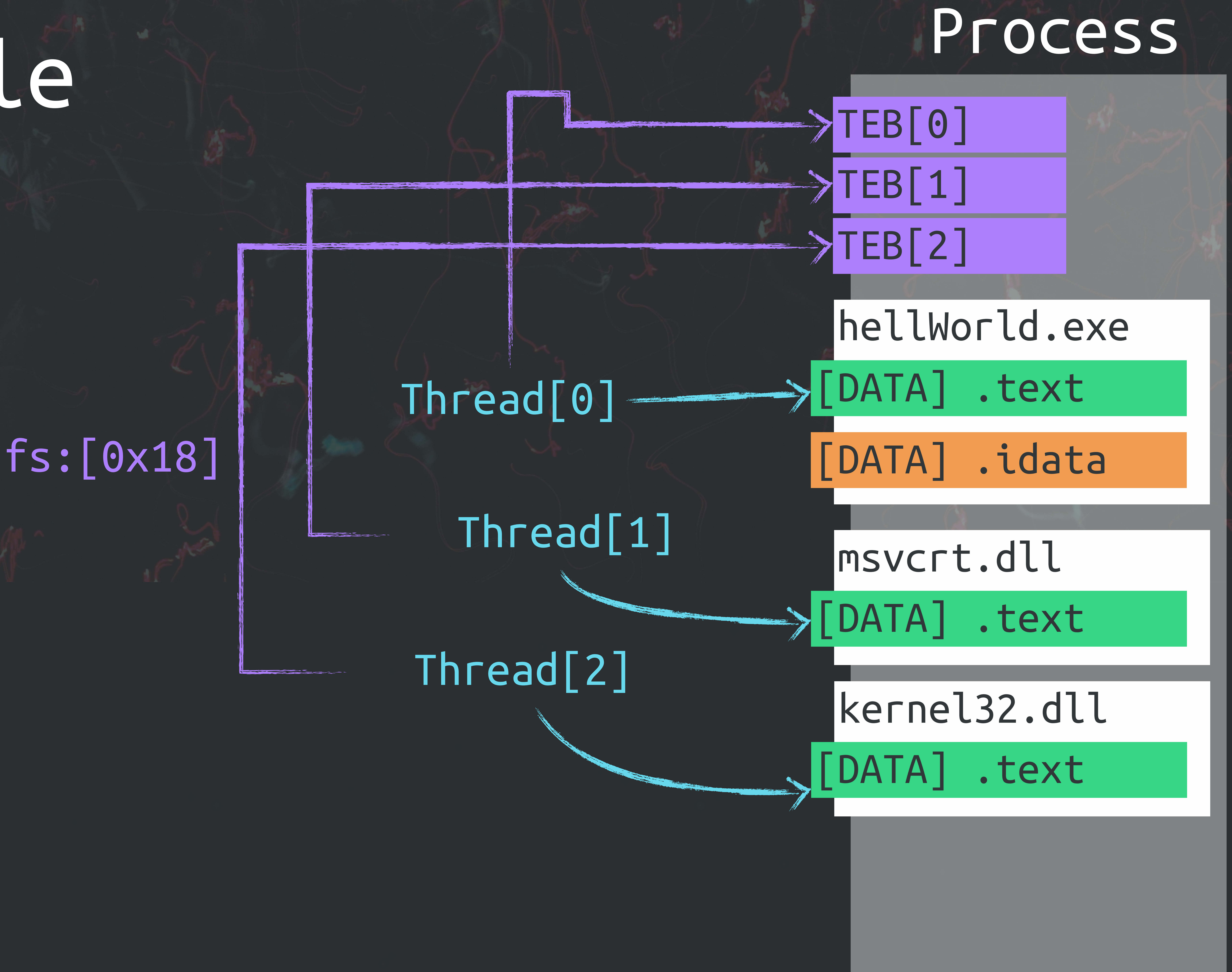
kernel32.dll

[DATA] .text

module_a.dll

module_b.dll

lifecycle


Process
TEB[0]
TEB[1]
TEB[2]
hellWorld.exe
[DATA] .text
[DATA] .idata
msvcrt.dll
[DATA] .text
kernel32.dll
[DATA] .text
fs:[0x18]
Thread[0]
Thread[1]
Thread[2]

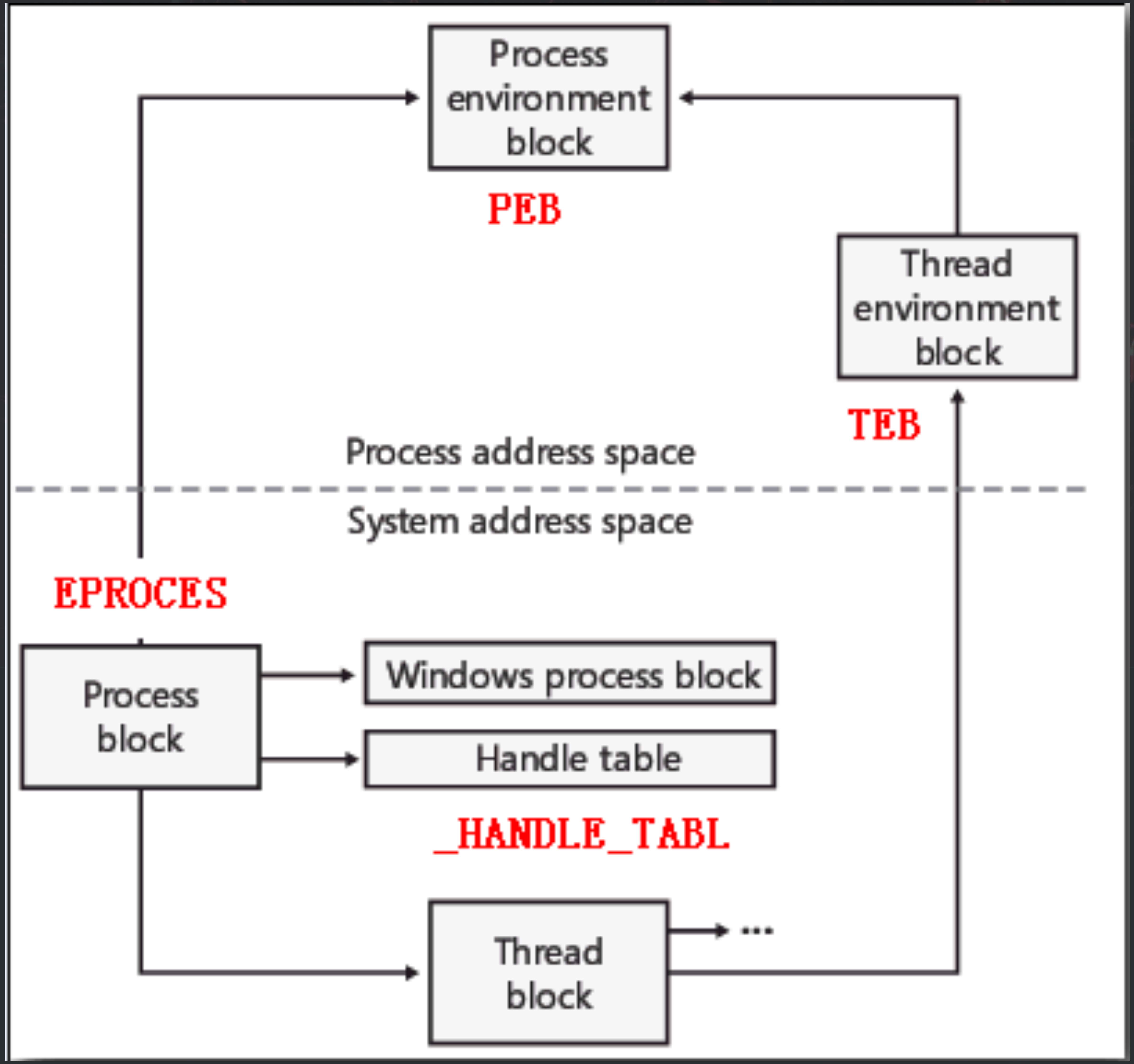
/? TIB

In computing, the Win32 Thread Information Block (TIB) is a data structure in Win32 on x86 that stores information about the currently running thread. This structure is also known as the Thread Environment Block (TEB).

The TIB can be used to get a lot of information on the process without calling Win32 API. Examples include emulating GetLastError(), GetVersion(). Through the pointer to the PEB one can obtain access to the import tables (IAT), process startup arguments, image name, etc. It is accessed from the FS segment register when operating on 32 bits, and from GS in 64 bits.

en.wikipedia.org/wiki/Win32_Thread_Information_Block

/? TIB

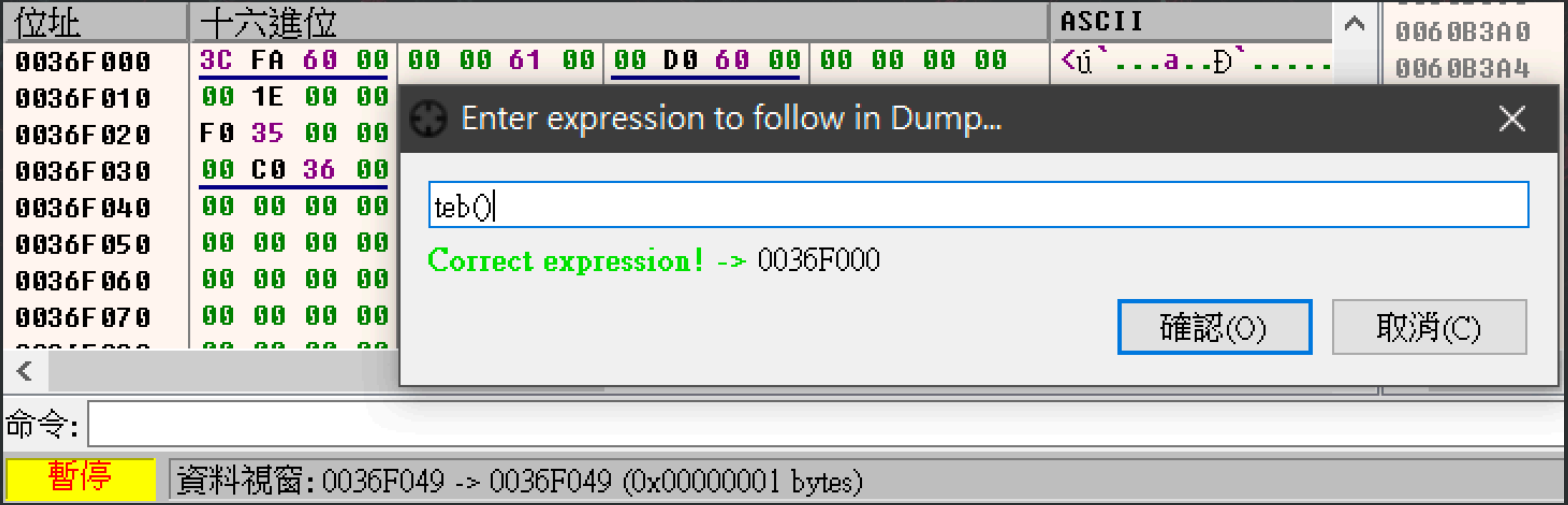

Process environment block
PEB
Thread environment block
TEB
Process address space
System address space
EPROCES
Process block
Windows process block
Handle table
_HANDLE_TABL
Thread block
...

/? TIB # Undocumented

```
struct TEB {
    //NT_TIB structure portion
    EXCEPTION_REGISTRATION*     ExceptionList;  //0x0000 / Current Structured Exception Handling frame
    void*       StackBase;                      //0x0004 / Bottom of stack (high address)
    void*       StackLimit;                     //0x0008 / Ceiling of stack (low address)
    void*       SubSystemTib;                   //0x000C
    union {
        void*   FiberData;                      //0x0010
        DWORD   Version;                        //0x0010
    } dword10;
    void*       ArbitraryUserPointer;           //0x0014
    TEB*        Self;                           //0x0018
    //NT_TIB ends (NT subsystem independent part)

    void*       EnvironmentPointer;         //0x001C
    CLIENT_ID   ClientId;                   //0x0020
    //          ClientId.ProcessId          //0x0020 / value retrieved by GetCurrentProcessId()
    //          ClientId.ThreadId           //0x0024 / value retrieved by GetCurrentThreadId()
    void*       ActiveRpcHandle;            //0x0028
    void*       ThreadLocalStoragePointer;  //0x002C
    PEB*        ProcessEnvironmentBlock;    //0x0030
    ...
```

bytepointer.com/resources/tebpeb32.htm

/? x64dbg



```
位址        十六進位                                              ASCII
0036F000   3C FA 60 00  00 00 61 00  00 D0 60 00  00 00 00 00   <ú`...a..Ð`.....
0036F010   00 1E 00 00
0036F020   F0 35 00 00
0036F030   00 C0 36 00
0036F040   00 00 00 00
0036F050   00 00 00 00
0036F060   00 00 00 00
0036F070   00 00 00 00
```

```
0060B3A0
0060B3A4
```

Enter expression to follow in Dump...

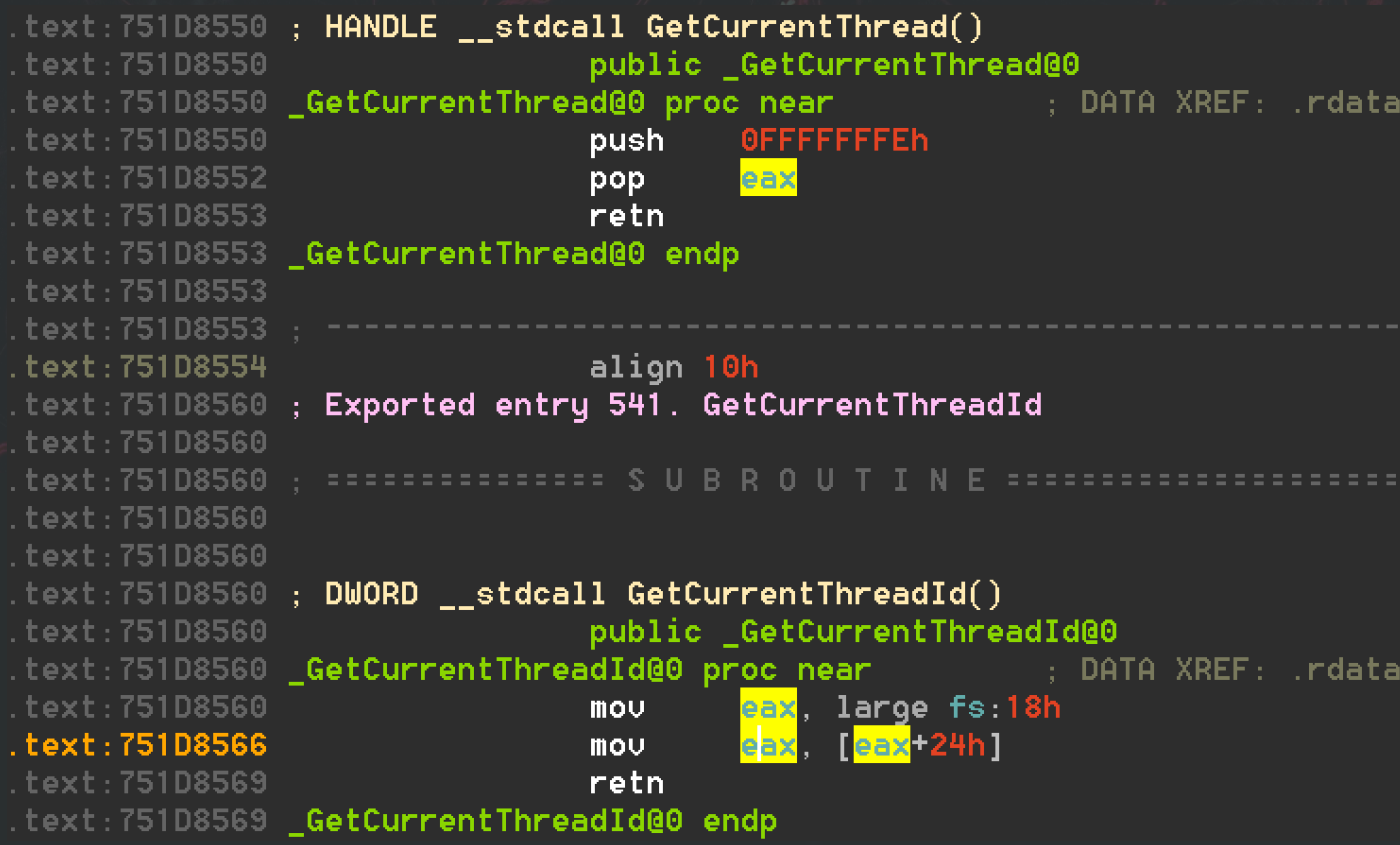
```
teb()
```

Correct expression! -> 0036F000

資料視窗: 0036F049 -> 0036F049 (0x00000001 bytes)

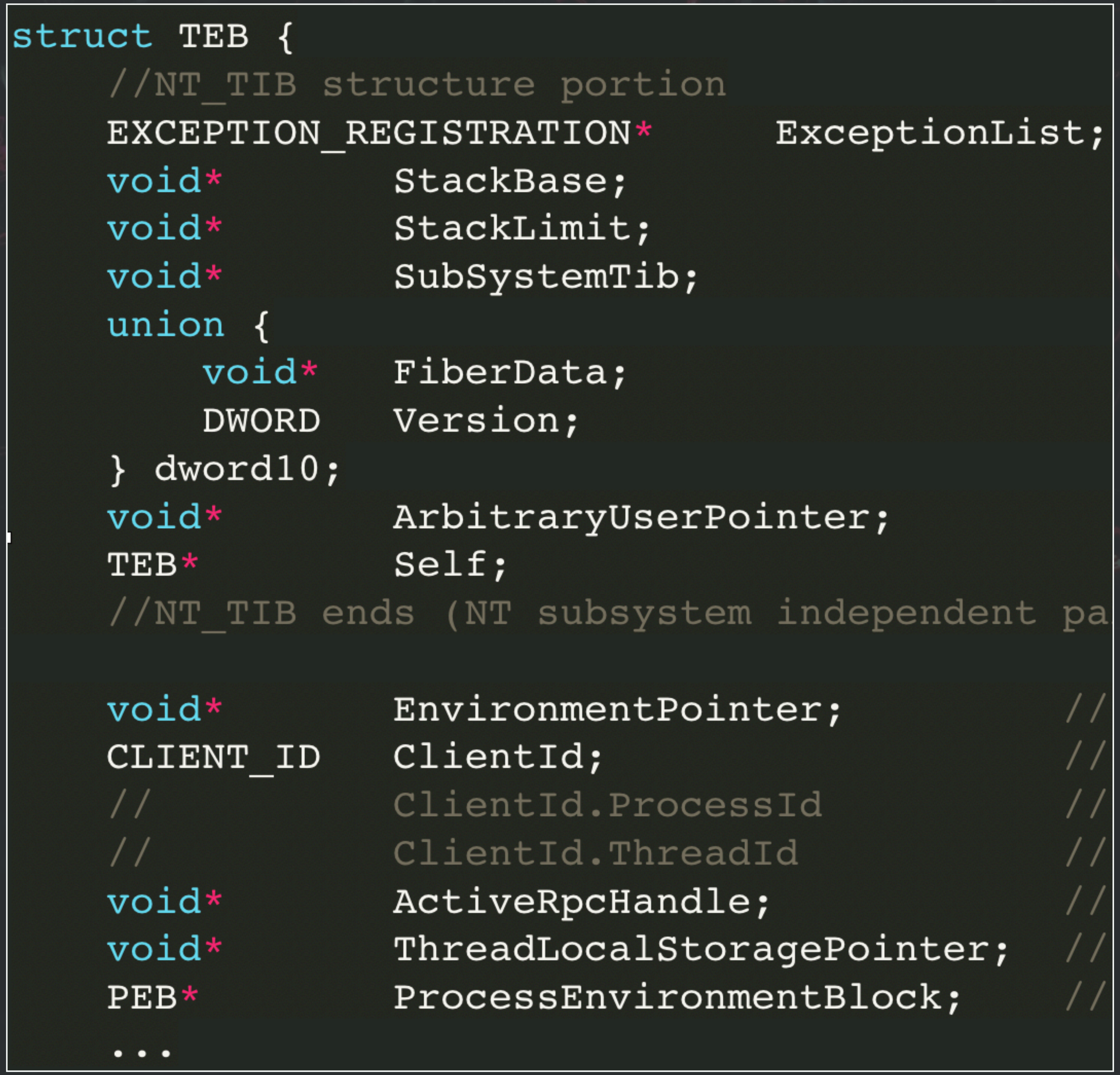
/? C$Windows\Sys32\Kernel32

- GetCurrentThread
- GetModuleHandleW
- GetCurrentThreadId
- GetCurrentThread
- IsDebuggerPresent

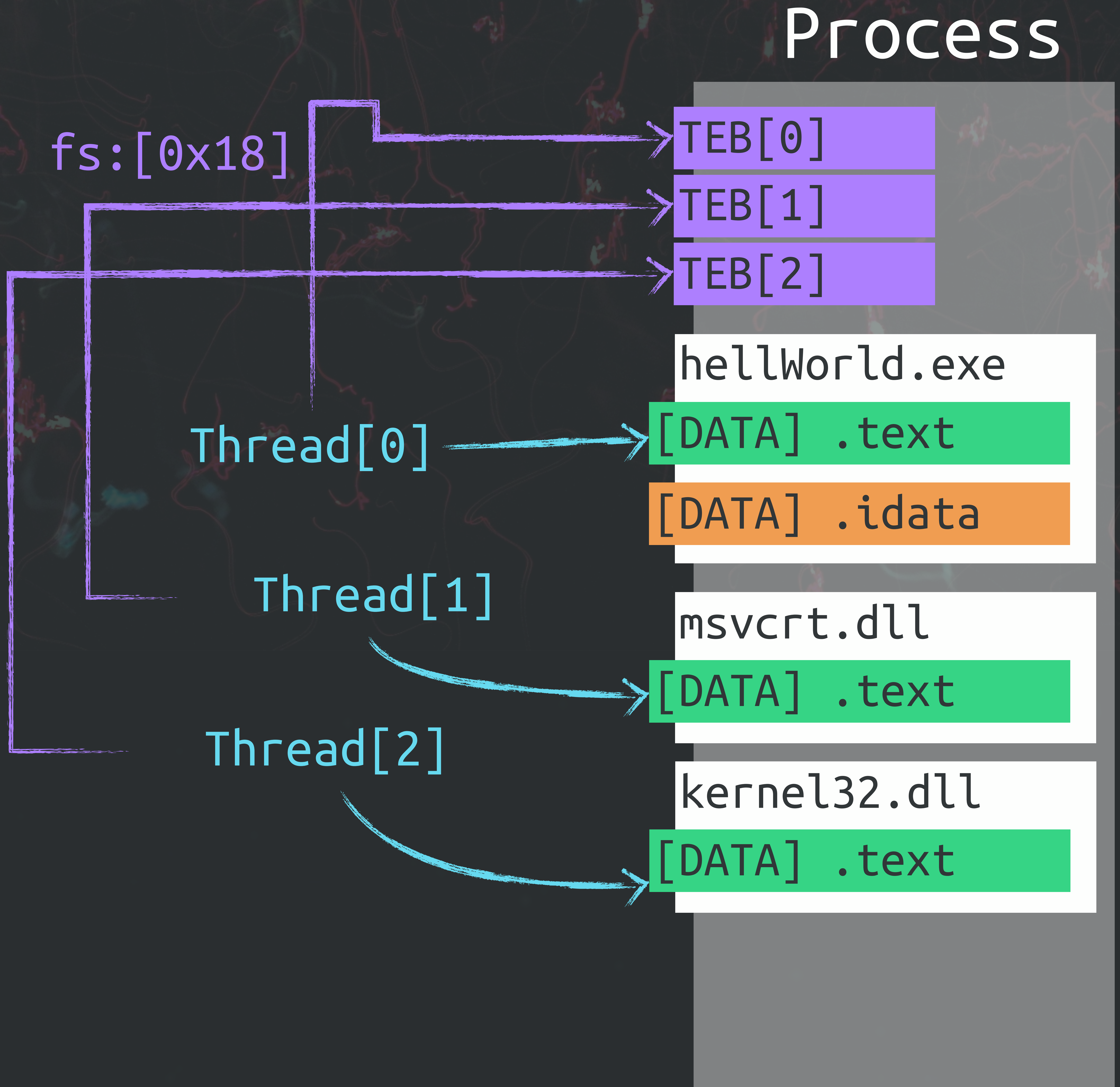


```
.text:751D8550 ; HANDLE __stdcall GetCurrentThread()
.text:751D8550                 public _GetCurrentThread@0
.text:751D8550 _GetCurrentThread@0 proc near          ; DATA XREF: .rdata
.text:751D8550                 push    0FFFFFFFEh
.text:751D8552                 pop     eax
.text:751D8553                 retn
.text:751D8553 _GetCurrentThread@0 endp
.text:751D8553
.text:751D8553 ; ---------------------------------------------------------
.text:751D8554                 align 10h
.text:751D8560 ; Exported entry 541. GetCurrentThreadId
.text:751D8560
.text:751D8560 ; =============== S U B R O U T I N E ====================
.text:751D8560
.text:751D8560
.text:751D8560 ; DWORD __stdcall GetCurrentThreadId()
.text:751D8560                 public _GetCurrentThreadId@0
.text:751D8560 _GetCurrentThreadId@0 proc near        ; DATA XREF: .rdata
.text:751D8560                 mov     eax, large fs:18h
.text:751D8566                 mov     eax, [eax+24h]
.text:751D8569                 retn
.text:751D8569 _GetCurrentThreadId@0 endp
```

lifecycle



```
struct TEB {
    //NT_TIB structure portion
    EXCEPTION_REGISTRATION*     ExceptionList;
    void*         StackBase;
    void*         StackLimit;
    void*         SubSystemTib;
    union {
        void*     FiberData;
        DWORD     Version;
    } dword10;
    void*         ArbitraryUserPointer;
    TEB*          Self;
    //NT_TIB ends (NT subsystem independent pa

    void*         EnvironmentPointer;           //
    CLIENT_ID     ClientId;                     //
    //            ClientId.ProcessId            //
    //            ClientId.ThreadId             //
    void*         ActiveRpcHandle;              //
    void*         ThreadLocalStoragePointer;    //
    PEB*          ProcessEnvironmentBlock;      //
    ...
```



Process

fs:[0x18]

TEB[0]

TEB[1]

TEB[2]

hellWorld.exe

[DATA] .text

[DATA] .idata

Thread[0]

Thread[1]

Thread[2]

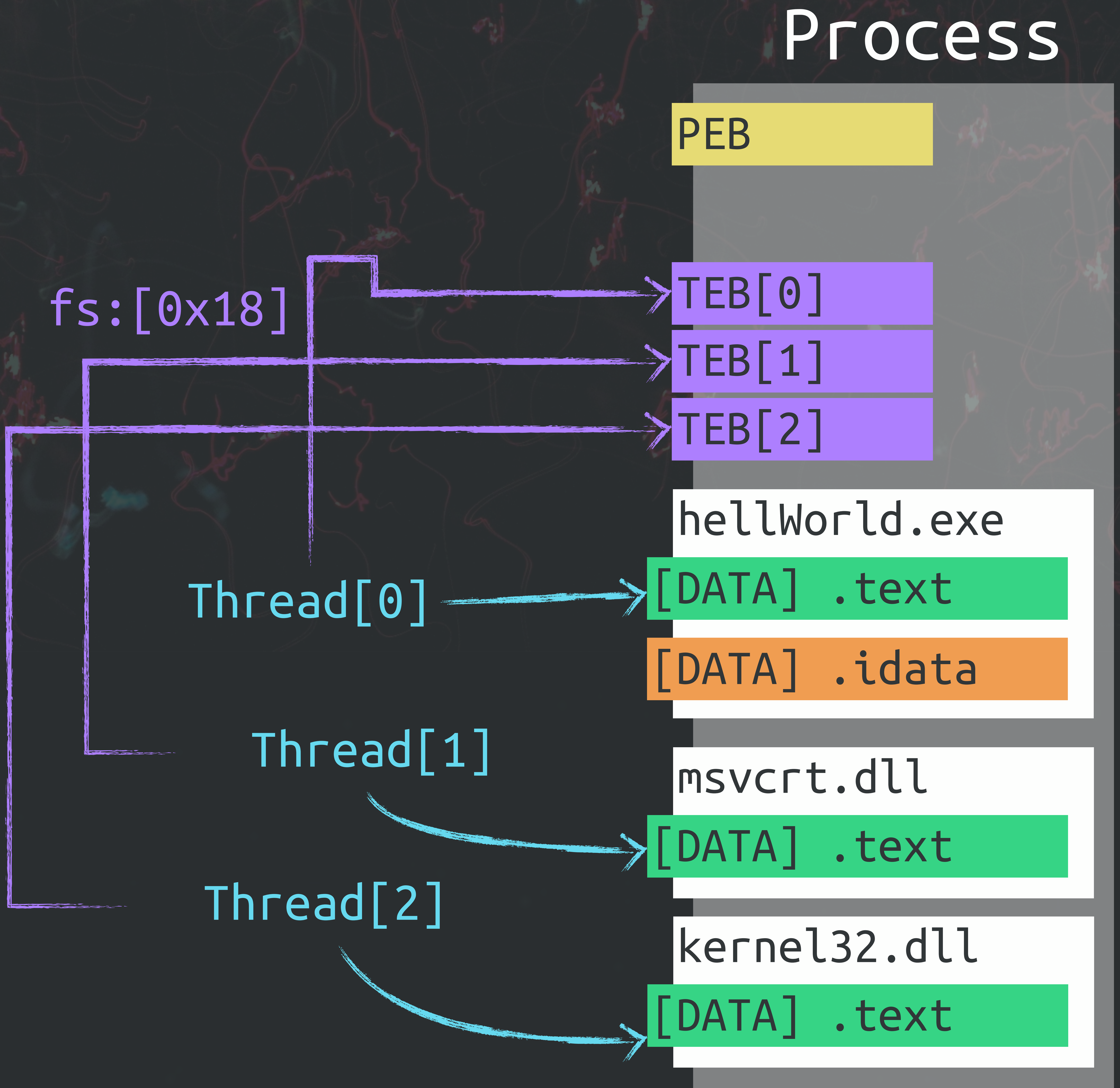
msvcrt.dll

[DATA] .text

kernel32.dll

[DATA] .text

lifecycle


Process
PEB
fs:[0x18]
TEB[0]
TEB[1]
TEB[2]
hellWorld.exe
Thread[0]
[DATA] .text
[DATA] .idata
Thread[1]
msvcrt.dll
[DATA] .text
Thread[2]
kernel32.dll
[DATA] .text

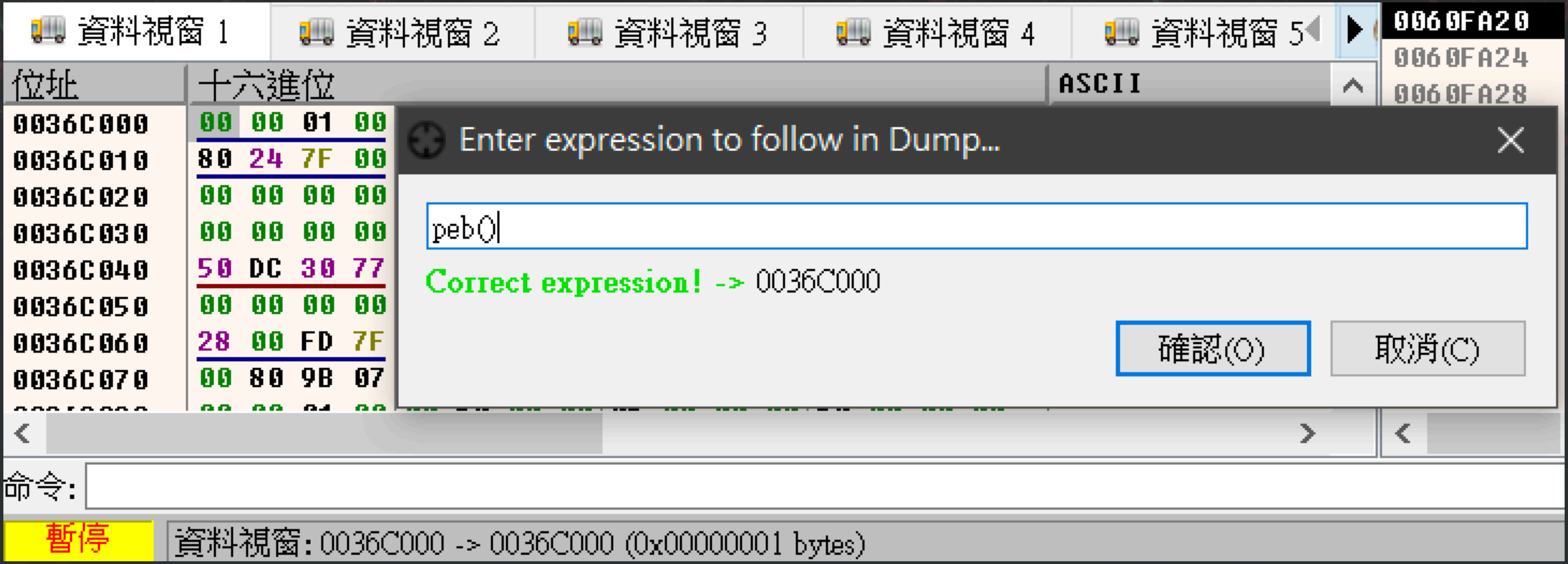
/? PEB

In computing the Process Environment Block (abbreviated PEB) is a data structure in the Windows NT operating system family. It is an opaque data structure that is used by the operating system internally, most of whose fields are not intended for use by anything other than the operating system.

Microsoft notes, in its MSDN Library documentation — which documents only a few of the fields — that the structure "may be altered in future versions of Windows". The PEB contains data structures that apply across a whole process, including global context, startup parameters, data structures for the program image loader, the program image base address, and synchronization objects used to provide mutual exclusion for process-wide data structures.

en.wikipedia.org/wiki/Process_Environment_Block

/? x64dbg



```
0036C000  00 00 01 00
0036C010  80 24 7F 00
0036C020  00 00 00 00
0036C030  00 00 00 00
0036C040  50 DC 30 77
0036C050  00 00 00 00
0036C060  28 00 FD 7F
0036C070  00 80 9B 07
```

Enter expression to follow in Dump...

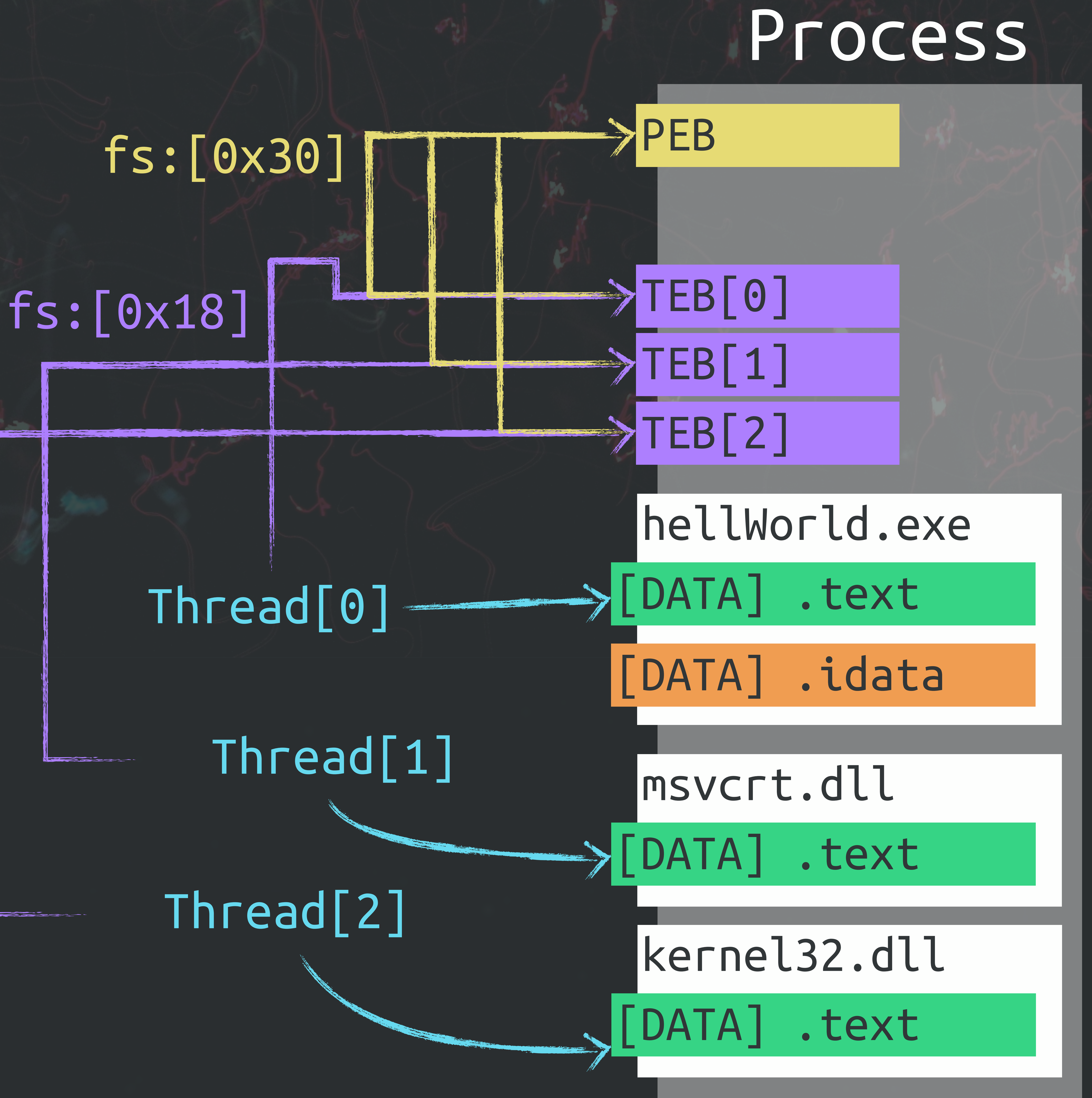
peb()

Correct expression! -> 0036C000

資料視窗: 0036C000 -> 0036C000 (0x00000001 bytes)

lifecycle

```
typedef struct _PEB32 {
  UCHAR InheritedAddressSpace;
  UCHAR ReadImageFileExecOptions;
  UCHAR BeingDebugged;
  UCHAR BitField;
  ULONG Mutant;
  ULONG ImageBaseAddress;
  PPEB_LDR_DATA Ldr;
  ULONG ProcessParameters;
  ULONG SubSystemData;
  ULONG ProcessHeap;
  ULONG FastPebLock;
  ULONG AtlThunkSListPtr;
  ULONG IFEOKey;
  ULONG CrossProcessFlags;
  ULONG UserSharedInfoPtr;
  ULONG SystemReserved;
  ULONG AtlThunkSListPtr32;
  ULONG ApiSetMap;
} PEB32, *PPEB32;
```



/?main

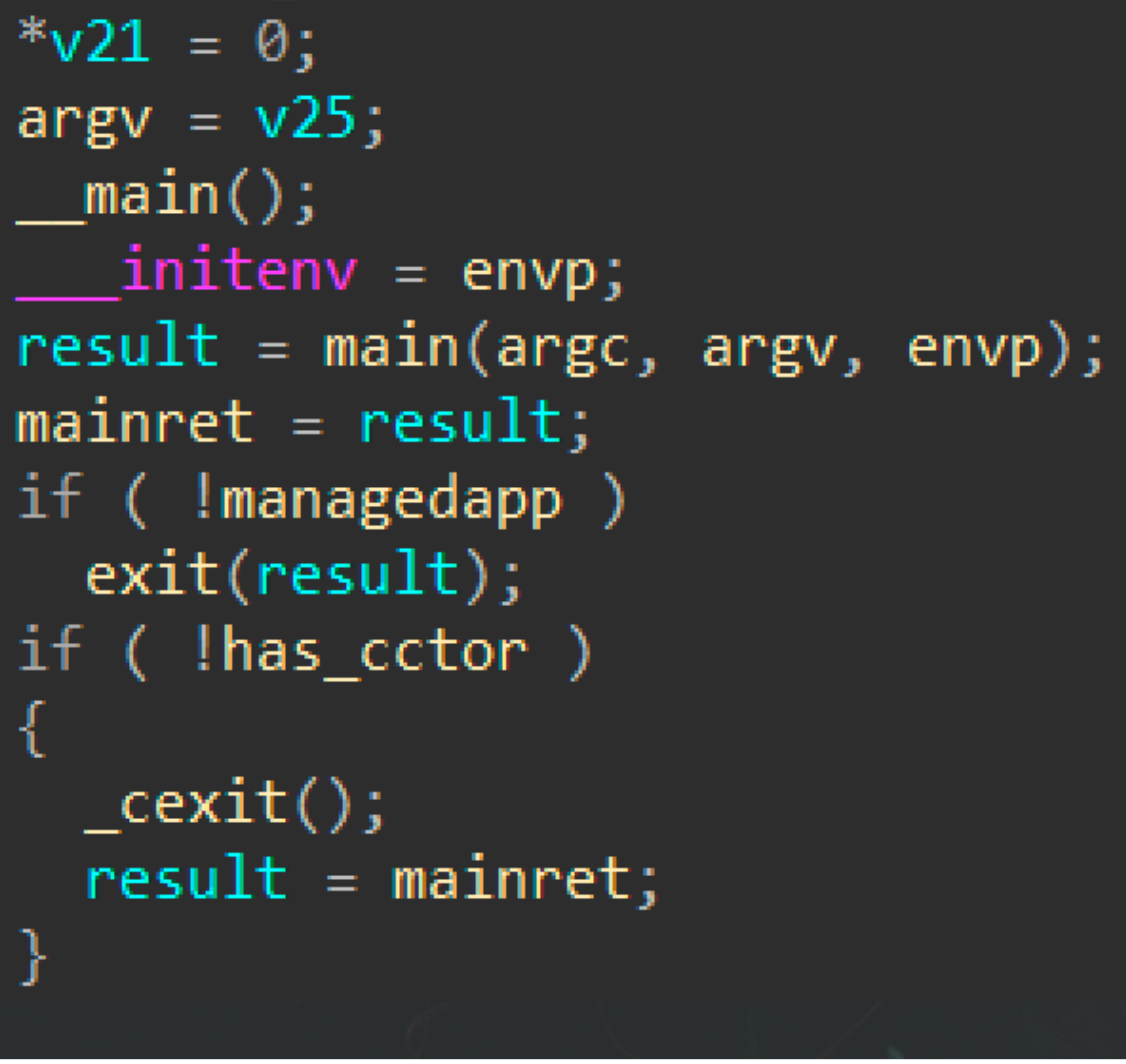
cat ./PE

# PE Overview

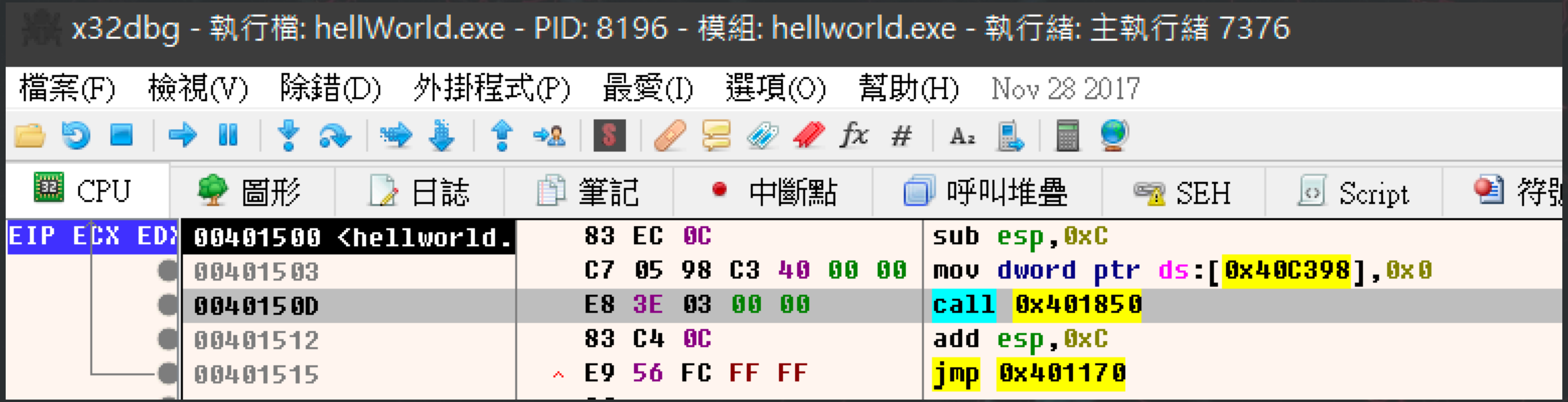
.ImageBase
.SizeOfHeaders
.AddressOfEntryPoint
.DataDirectory
.SizeOfImage
.NumberOfSections
.text
.size
.RVA
.VA
Section Header 1
'MZ'
DOS
'PE'
File Headr
Opt Header
Section Header Array →
.text (Data)

$_ ./Lab1

Lab1

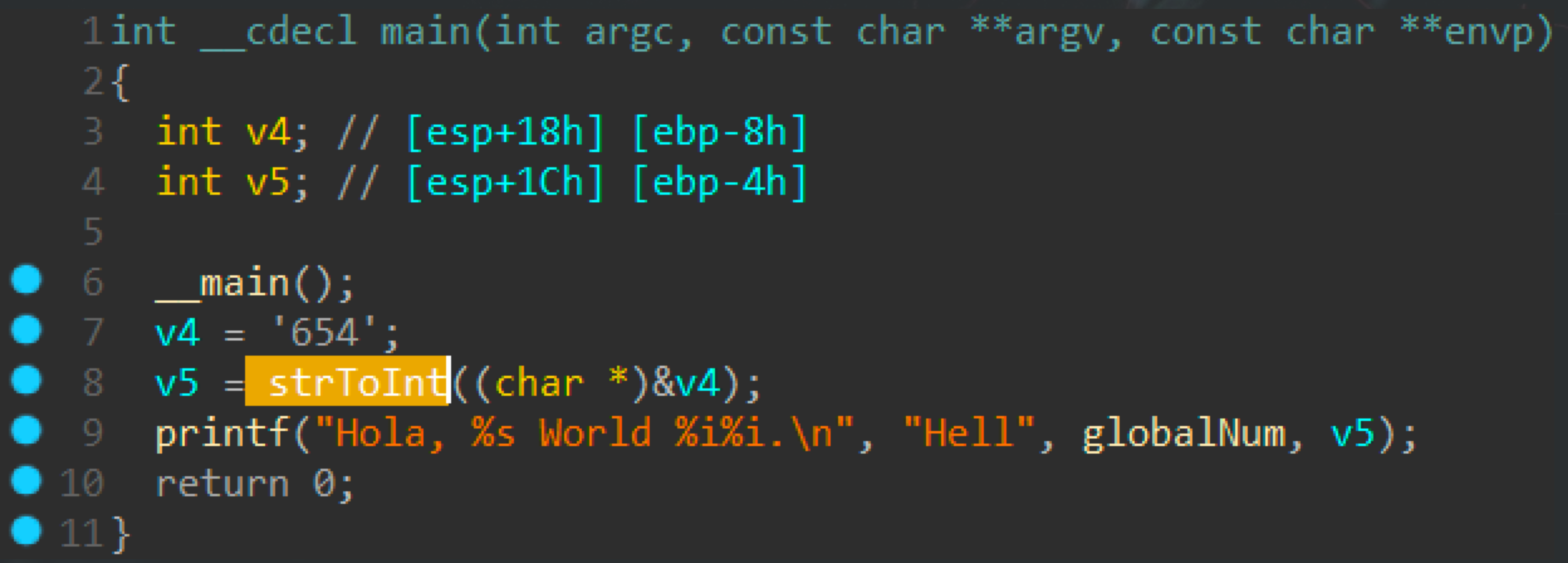


```
*v21 = 0;
argv = v25;
__main();
___initenv = envp;
result = main(argc, argv, envp);
mainret = result;
if ( !managedapp )
  exit(result);
if ( !has_cctor )
{
  _cexit();
  result = mainret;
}
```

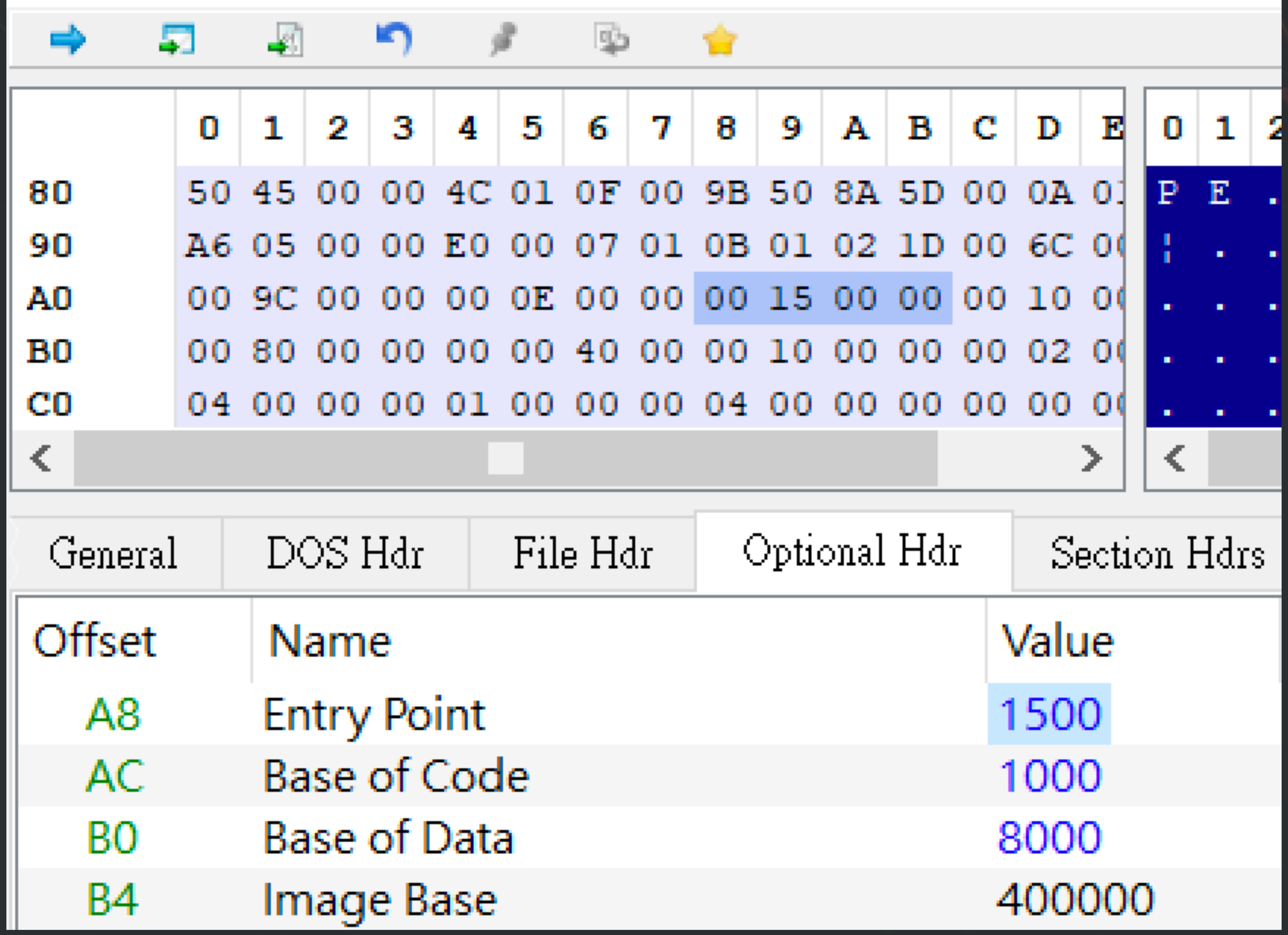


x32dbg - 執行檔: hellWorld.exe - PID: 8196 - 模組: hellworld.exe - 執行緒: 主執行緒 7376
檔案(F) 檢視(V) 除錯(D) 外掛程式(P) 最愛(I) 選項(O) 幫助(H) Nov 28 2017
CPU 圖形 日誌 筆記 中斷點 呼叫堆疊 SEH Script 符號

```
EIP ECX EDX  00401500 <hellworld.    83 EC 0C              sub esp,0xC
             00401503                C7 05 98 C3 40 00 00  mov dword ptr ds:[0x40C398],0x0
             0040150D                E8 3E 03 00 00        call 0x401850
             00401512                83 C4 0C              add esp,0xC
             00401515              ^ E9 56 FC FF FF        jmp 0x401170
```



```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int v4; // [esp+18h] [ebp-8h]
  int v5; // [esp+1Ch] [ebp-4h]

  __main();
  v4 = '654';
  v5 = strToInt((char *)&v4);
  printf("Hola, %s World %i%i.\n", "Hell", globalNum, v5);
  return 0;
}
```

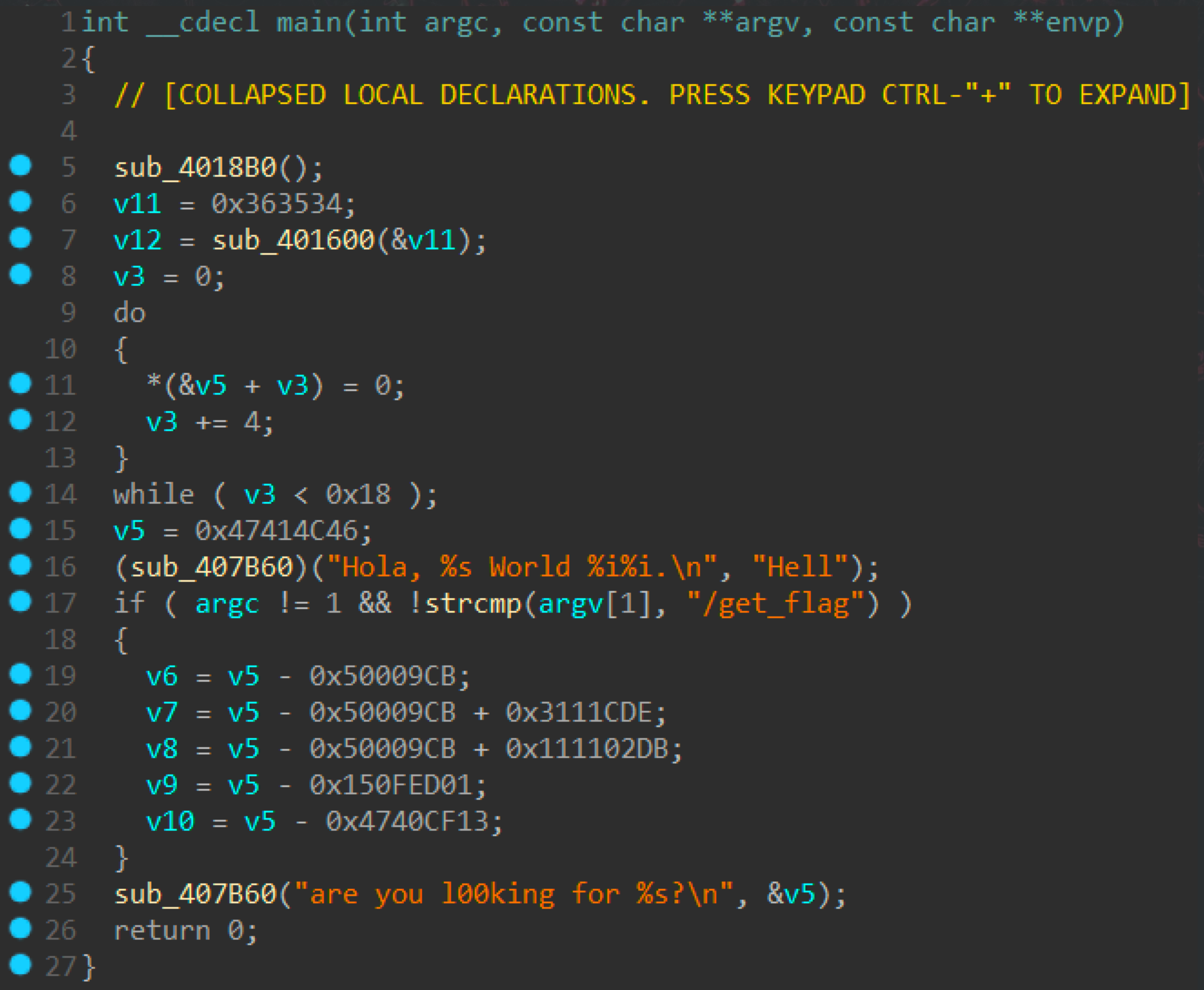


```
     0  1  2  3  4  5  6  7  8  9  A  B  C  D  E    0 1 2
80   50 45 00 00 4C 01 0F 00 9B 50 8A 5D 00 0A 0    P E .
90   A6 05 00 00 E0 00 07 01 0B 01 02 1D 00 6C 0    ¦ . .
A0   00 9C 00 00 00 0E 00 00 00 15 00 00 00 10 0    . . .
B0   00 80 00 00 00 00 40 00 00 10 00 00 00 02 0    . . .
C0   04 00 00 00 01 00 00 00 04 00 00 00 00 00 0    . . .
```

Offset	Name	Value
A8	Entry Point	1500
AC	Base of Code	1000
B0	Base of Data	8000
B4	Image Base	400000

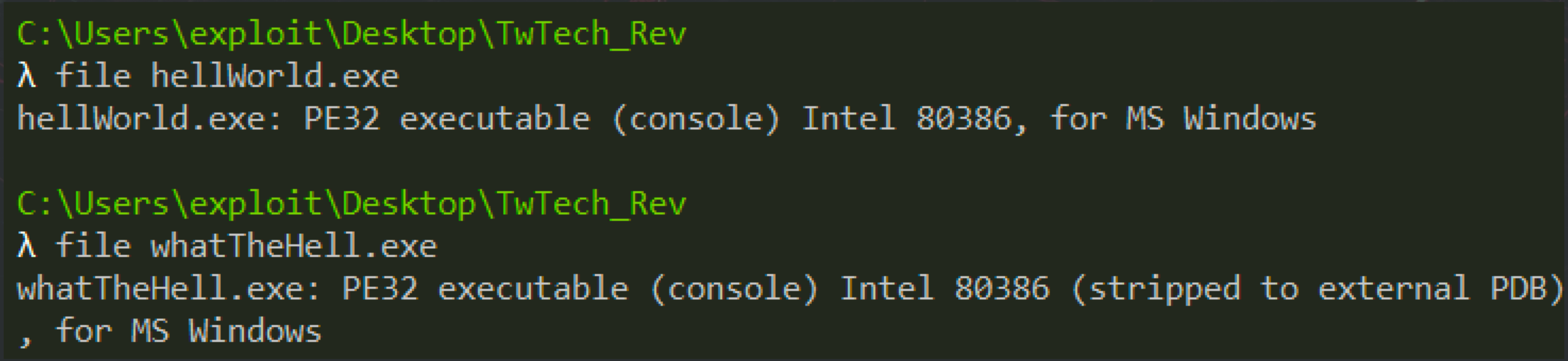
$_ ./Lab2

Lab2



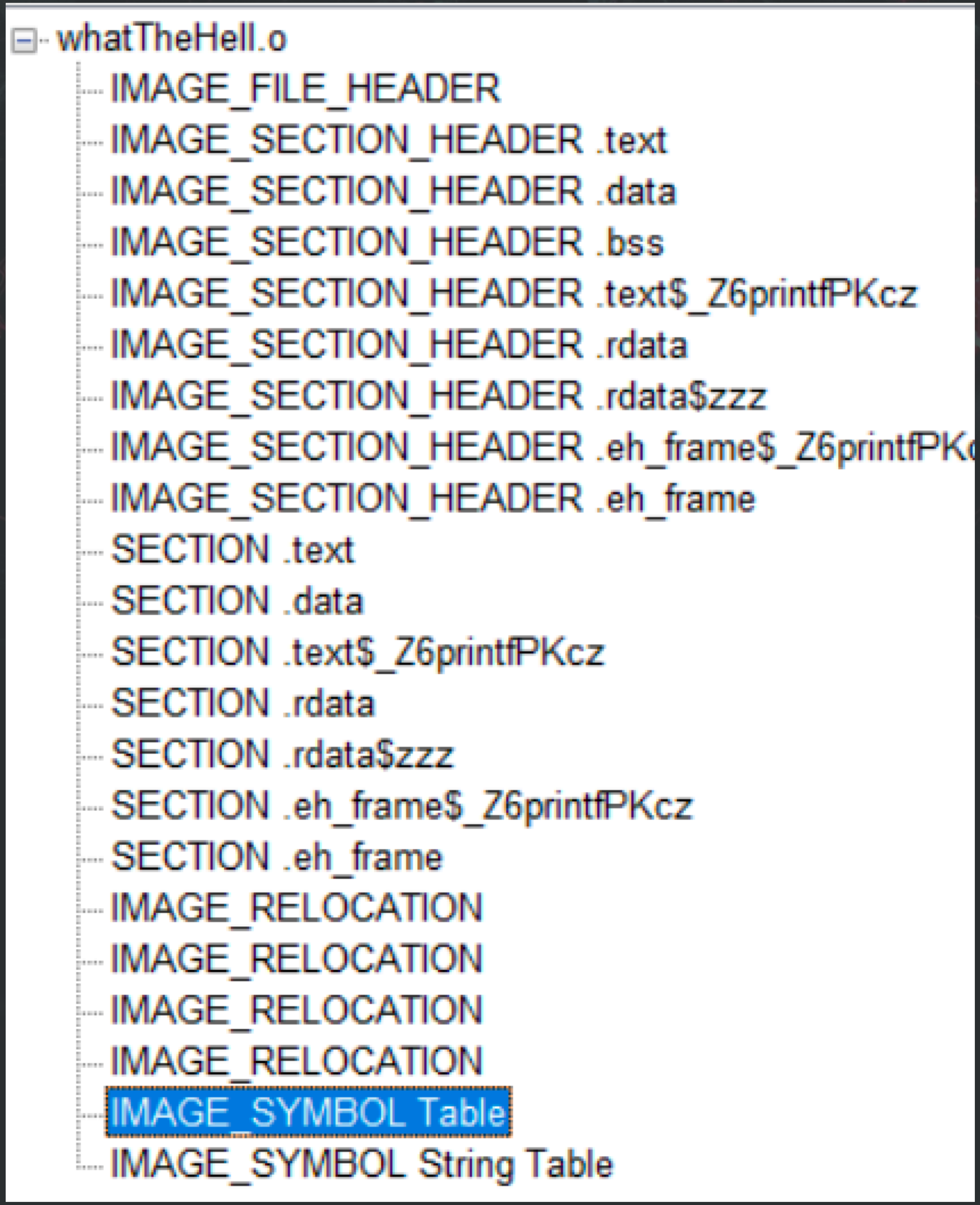
```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  // [COLLAPSED LOCAL DECLARATIONS. PRESS KEYPAD CTRL-"+" TO EXPAND]

  sub_4018B0();
  v11 = 0x363534;
  v12 = sub_401600(&v11);
  v3 = 0;
  do
  {
    *(&v5 + v3) = 0;
    v3 += 4;
  }
  while ( v3 < 0x18 );
  v5 = 0x47414C46;
  (sub_407B60)("Hola, %s World %i%i.\n", "Hell");
  if ( argc != 1 && !strcmp(argv[1], "/get_flag") )
  {
    v6 = v5 - 0x50009CB;
    v7 = v5 - 0x50009CB + 0x3111CDE;
    v8 = v5 - 0x50009CB + 0x111102DB;
    v9 = v5 - 0x150FED01;
    v10 = v5 - 0x4740CF13;
  }
  sub_407B60("are you l00king for %s?\n", &v5);
  return 0;
}
```

Lab 1 & 2

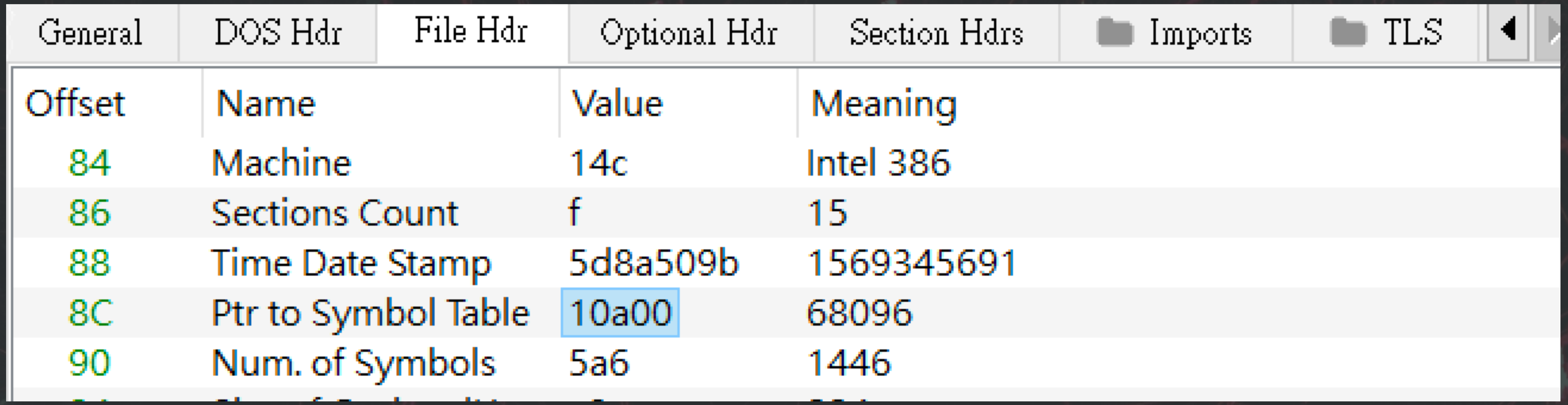


```
C:\Users\exploit\Desktop\TwTech_Rev
λ file hellWorld.exe
hellWorld.exe: PE32 executable (console) Intel 80386, for MS Windows

C:\Users\exploit\Desktop\TwTech_Rev
λ file whatTheHell.exe
whatTheHell.exe: PE32 executable (console) Intel 80386 (stripped to external PDB)
, for MS Windows
```

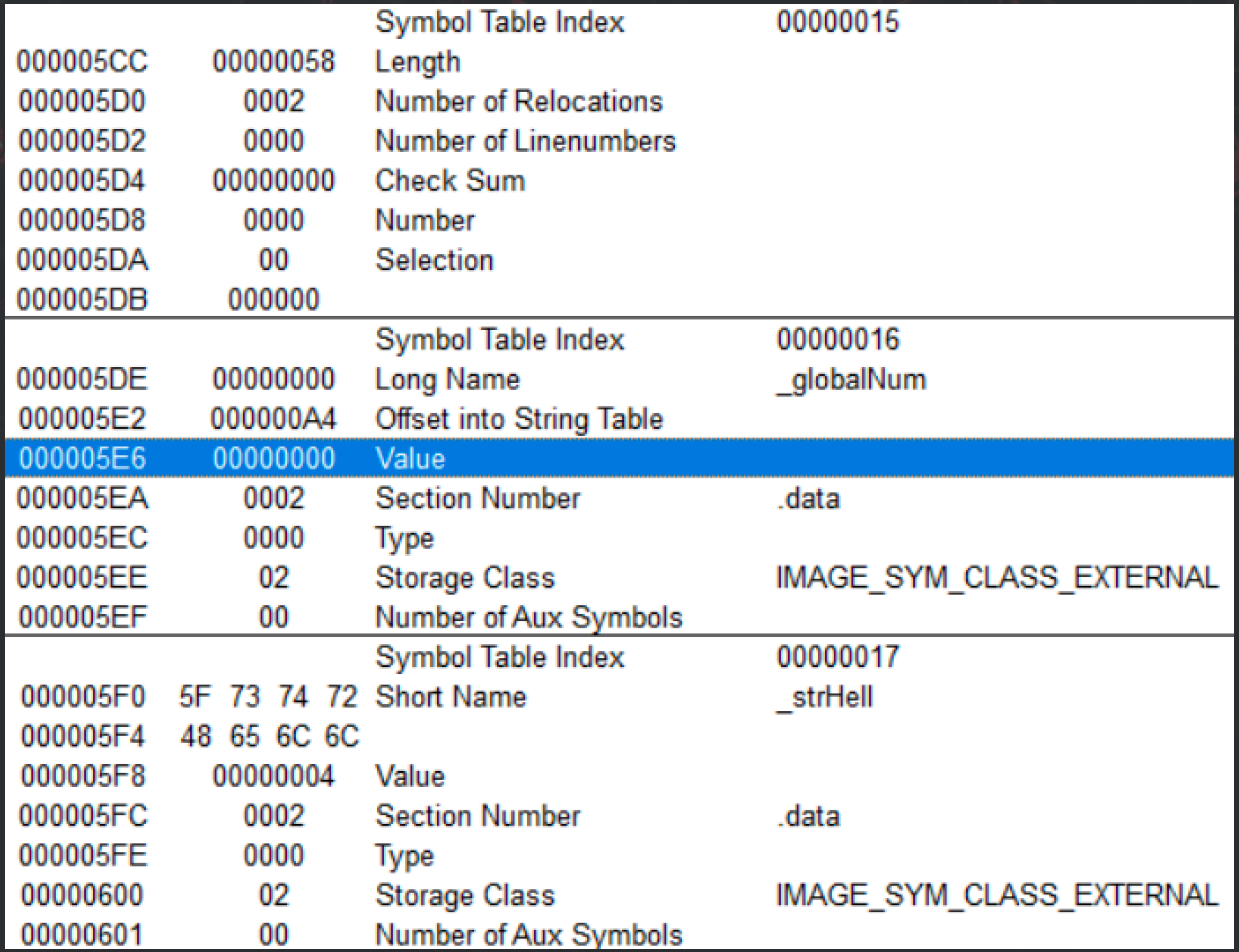
# Lab1



```
⊟ whatTheHell.o
    IMAGE_FILE_HEADER
    IMAGE_SECTION_HEADER .text
    IMAGE_SECTION_HEADER .data
    IMAGE_SECTION_HEADER .bss
    IMAGE_SECTION_HEADER .text$_Z6printfPKcz
    IMAGE_SECTION_HEADER .rdata
    IMAGE_SECTION_HEADER .rdata$zzz
    IMAGE_SECTION_HEADER .eh_frame$_Z6printfPKc
    IMAGE_SECTION_HEADER .eh_frame
    SECTION .text
    SECTION .data
    SECTION .text$_Z6printfPKcz
    SECTION .rdata
    SECTION .rdata$zzz
    SECTION .eh_frame$_Z6printfPKcz
    SECTION .eh_frame
    IMAGE_RELOCATION
    IMAGE_RELOCATION
    IMAGE_RELOCATION
    IMAGE_RELOCATION
    IMAGE_SYMBOL Table
    IMAGE_SYMBOL String Table
```



Offset	Name	Value	Meaning
84	Machine	14c	Intel 386
86	Sections Count	f	15
88	Time Date Stamp	5d8a509b	1569345691
8C	Ptr to Symbol Table	10a00	68096
90	Num. of Symbols	5a6	1446

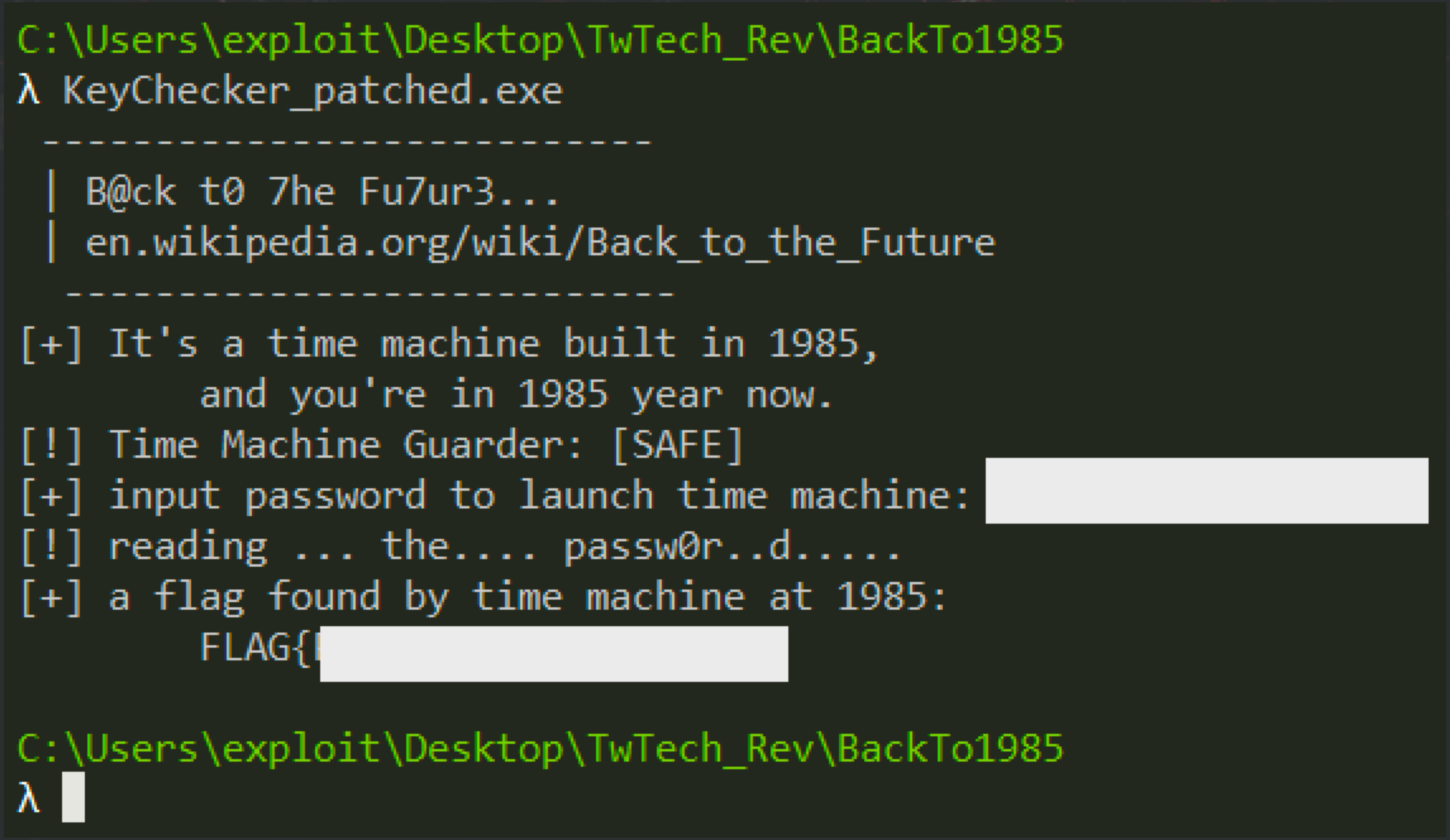


```
                                Symbol Table Index          00000015
000005CC     00000058          Length
000005D0       0002            Number of Relocations
000005D2       0000            Number of Linenumbers
000005D4     00000000          Check Sum
000005D8       0000            Number
000005DA        00             Selection
000005DB      000000
                                Symbol Table Index          00000016
000005DE     00000000          Long Name                   _globalNum
000005E2     000000A4          Offset into String Table
000005E6     00000000          Value
000005EA       0002            Section Number              .data
000005EC       0000            Type
000005EE        02             Storage Class               IMAGE_SYM_CLASS_EXTERNAL
000005EF        00             Number of Aux Symbols
                                Symbol Table Index          00000017
000005F0   5F  73  74  72      Short Name                  _strHell
000005F4   48  65  6C  6C
000005F8     00000004          Value
000005FC       0002            Section Number              .data
000005FE       0000            Type
00000600        02             Storage Class               IMAGE_SYM_CLASS_EXTERNAL
00000601        00             Number of Aux Symbols
```

cd ./r3versing

/? homework

# Back To The Future



```
C:\Users\exploit\Desktop\TwTech_Rev\BackTo1985
λ KeyChecker_patched.exe
 -----------------------------
 | B@ck t0 7he Fu7ur3...
 | en.wikipedia.org/wiki/Back_to_the_Future
  -----------------------------
[+] It's a time machine built in 1985,
        and you're in 1985 year now.
[!] Time Machine Guarder: [SAFE]
[+] input password to launch time machine: [redacted]
[!] reading ... the.... passw0r..d.....
[+] a flag found by time machine at 1985:
        FLAG{[redacted]

C:\Users\exploit\Desktop\TwTech_Rev\BackTo1985
λ
```

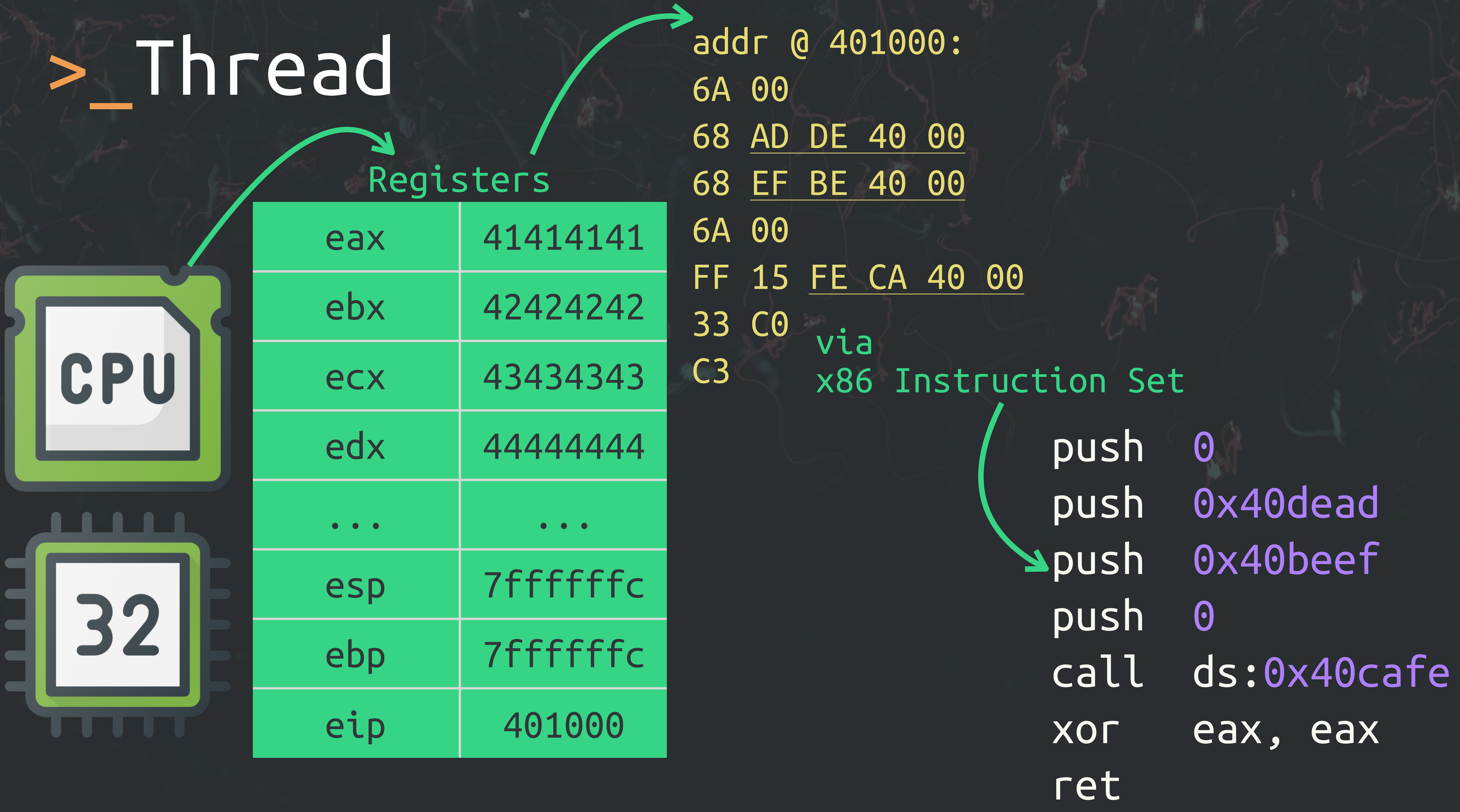
/? homework

```
sub_4019E0();
v3 = GetModuleHandleA(0);
v6 = (_DWORD *)((char *)v3 + *((_DWORD *)v3 + 0xF));
v5 = __readfsdword(0x30u);
if ( *(_WORD *)v3 == 0x5A4D && *v6 == 0x4550 )
{
  sub_407CA0(
    " ---------------------------- \n"
    " | B@ck t0 7he Fu7ur3...  \n"
    " | en.wikipedia.org/wiki/Back_to_the_Future\n"
    "  --------------------------- \n");
  dword_40C040 = sub_401600(v6[2]);
  sub_407CA0("[+] It's a time machine built in 1985, \n\tand you're in %i ye
  if ( dword_40C040 != 0x7C1 )
    puts("[!] WARNING: \n\tit might be some trouble if you're not in 1985 ye
  *(_BYTE *)(v5 + 2);
  sub_407CA0("[!] Time Machine Guarder: %s\n");
  sub_407CA0("[+] input password to launch time machine: ");
  gets(byte_40C060);
  for ( i = 0; strlen(byte_40C060) > i; ++i )
    byte_40C060[i] |= 0x20u;
  sub_407CA0("[!] reading ... the.... passw0r..d.....\n");
  for ( j = 0; j <= 0x12; ++j )
  {
    byte_40C060[j] ^= 2 * (dword_40C040 + 0x3F) + *(_BYTE *)(v5 + 2) + 0x7F;
    if ( byte_40C060[j] != byte_408008[j] )
    {
      puts("[!] oops... time machine g0t some                tim3... ");
      break;
    }
  }
  for ( k = 0; k <= 0x12; ++k )
    byte_40C060[k] ^= byte_40801C[k];
  sub_407CA0("[+] a flag found by time machine
}
else
{
  puts("time machine broken, oohoho. please don't patch me ;)");
}
```

我知道在場一堆IDA狗，再不學動態分析試試看，看怎麼你逆ㄏㄏ

```
cd ./stack
```

>_Thread



Registers

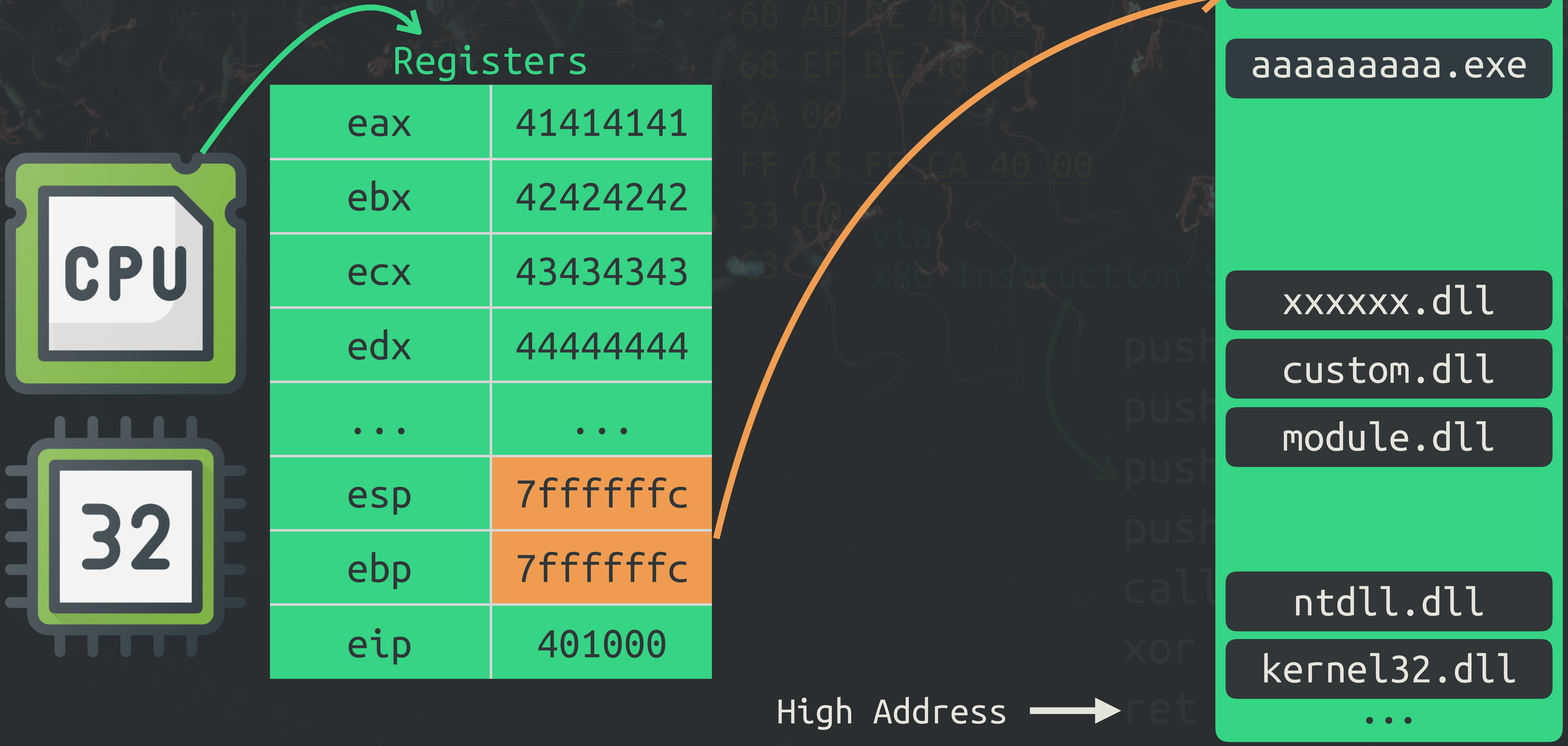
eax	41414141
ebx	42424242
ecx	43434343
edx	44444444
...	...
esp	7ffffffc
ebp	7ffffffc
eip	401000

```
addr @ 401000:
6A 00
68 AD DE 40 00
68 EF BE 40 00
6A 00
FF 15 FE CA 40 00
33 C0
C3
```

via
x86 Instruction Set

```
push  0
push  0x40dead
push  0x40beef
push  0
call  ds:0x40cafe
xor   eax, eax
ret
```

>_Thread



Registers

eax	41414141
ebx	42424242
ecx	43434343
edx	44444444
...	...
esp	7ffffffc
ebp	7ffffffc
eip	401000

Process

- Thread Stack
- aaaaaaaaa.exe
- xxxxxx.dll
- custom.dll
- module.dll
- ntdll.dll
- kernel32.dll
- ...

>_Heap

&(buf[0]) = 0x100
buf[0] = 1
&(buf[1]) = 0x104
buf[1] = 0xAAAAAAAA
&(buf[2]) = 0x108
buf[2] = 0x12345678
00
00
00
01
AA
AA
AA
AA
78
56
34
12
Low Address = 0x100
(buf)
High Address

```
uint32_t buf[3] = { 1, 2, 3 };
buf[1] = 0xAAAAAAAA;
buf[2] = 0x12345678;
```

>_Stack

esp = 0x100 + sizeof(uint32_t) * 99

Allocate Local Memory

Release Local Memory

Low Address = 0x100
(stack)

High Address

```
uint32_t stack[100];
uint32_t index = 99;
void x86_push(uint32_t in) { stack[--index] = in; }
void x86_pop(&out)         { x = stack[index++];  }
```

>_Stack

esp = 0x28c
index = 99

Low Address = 0x100
(stack)

High Address

```
push eax

push ebx

pop edx
```

eax	1
ebx	2
edx	3

>_Stack

esp = 0x288
index = 98
esp = 0x28c
index = 99
00
00
00
01
Low Address = 0x100
(stack)
High Address

```
push eax
push ebx
pop edx
```

eax	1
ebx	2
edx	3

>_Stack

esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
00
00
00
02
00
00
00
01
Low Address = 0x100
(stack)
High Address

```
push eax
push ebx
pop edx
```

eax	1
ebx	2
edx	3

>_Stack

esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
00
00
00
02
00
00
00
01
Low Address = 0x100
(stack)
High Address

```
push eax
push ebx
pop edx
```

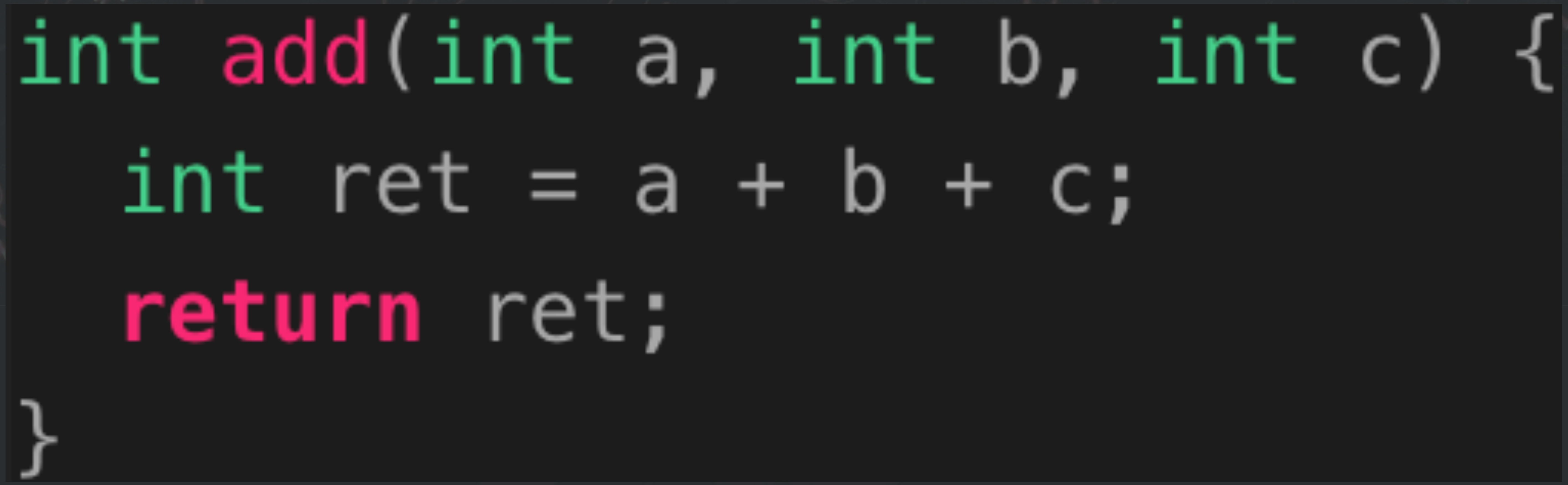
eax	1
ebx	2
edx	2

x86 Calling Convention

aaaddress1@chroot.org

>_Calling Convention

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```



```c
int add(int a, int b, int c) {
  int ret = a + b + c;
  return ret;
}
```

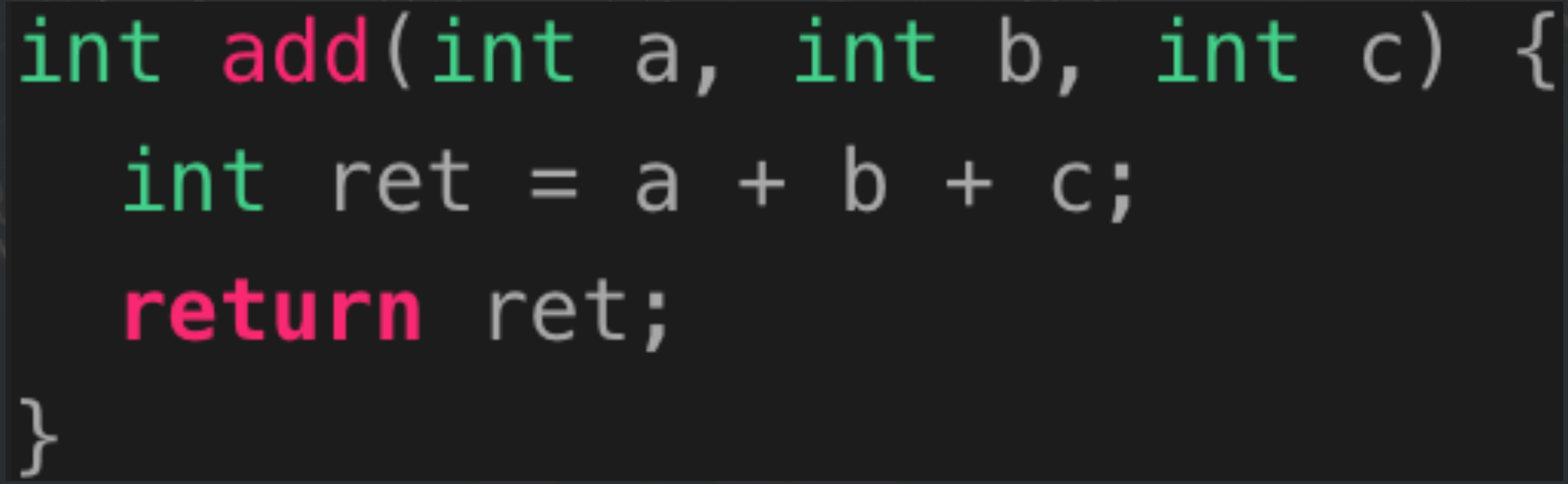
en.wikipedia.org/wiki/X86_calling_conventions

>_Calling Convention

The Begin of function

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

The end of function



```
int add(int a, int b, int c) {
  int ret = a + b + c;
  return ret;
}
```

en.wikipedia.org/wiki/X86_calling_conventions

>_Function

esp = 0x28c
index = 99

Low Address = 0x100
(stack)

High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

esp = 0x288
index = 98
esp = 0x28c
index = 99
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
00
00
00
02
00
00
00
03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

esp = 0x27c
index = 95
esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
ret addr
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

esp = 0x278
index = 94
esp = 0x27c
index = 95
esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
old ebp
ret addr
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

ebp = 0x278 (the base pointer for the current stack frame)
esp = 0x278
index = 94
esp = 0x27c
index = 95
esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
old ebp
ret addr
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

ebp = 0x278 (the base pointer for the current stack frame)
esp = 0x274
index = 93
esp = 0x278
index = 94
esp = 0x27c
index = 95
esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
local buf
old ebp
ret addr
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

ebp = 0x278 (the base pointer for the current stack frame)
esp = 0x274
index = 93
esp = 0x278
index = 94
esp = 0x27c
index = 95
esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
ebp
ebp+4
arg1:
ebp+8
arg2:
ebp+0x0c
arg3:
ebp+0x10
local
buf
old ebp
ret addr
00
00
00
01
00
00
00
02
00
00
00
03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

ebp = 0x278 (the base pointer for the current stack frame)

esp = 0x274
index = 93

esp = 0x278
index = 94

esp = 0x27c
index = 95

esp = 0x280
index = 96

esp = 0x284
index = 97

esp = 0x288
index = 98

esp = 0x28c
index = 99

ebp

ebp+4

arg1:
ebp+8

arg2:
ebp+0x0c

arg3:
ebp+0x10

local buf	old ebp	ret addr	00	00	00	01	00	00	00	02	00	00	00	03

Low Address = 0x100
(stack)

High Address

```
push 3
push 2
push 1
call add
add   esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

ebp = 0x278 (the base pointer for the current stack frame)
esp = 0x274
index = 93
esp = 0x278
index = 94
esp = 0x27c
index = 95
esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
ebp
ebp+4
arg1:
ebp+8
arg2:
ebp+0x0c
arg3:
ebp+0x10
6
old ebp
ret addr
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

ebp = 0x278 (the base pointer for the current stack frame)
esp = 0x278
index = 94
esp = 0x27c
index = 95
esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
ebp
ebp+4
arg1:
ebp+8
arg2:
ebp+0x0c
arg3:
ebp+0x10
6
old ebp
ret addr
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

esp = 0x27c
index = 95
esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
ebp
ebp+4
arg1:
ebp+8
arg2:
ebp+0x0c
arg3:
ebp+0x10
6
old ebp
ret addr
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

esp = 0x280
index = 96
esp = 0x284
index = 97
esp = 0x288
index = 98
esp = 0x28c
index = 99
ebp
ebp+4
arg1:
ebp+8
arg2:
ebp+0x0c
arg3:
ebp+0x10
6
old ebp
ret addr
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

esp = 0x28c
index = 99
ebp
ebp+4
arg1:
ebp+8
arg2:
ebp+0x0c
arg3:
ebp+0x10
6
old ebp
ret addr
00 00 00 01
00 00 00 02
00 00 00 03
Low Address = 0x100
(stack)
High Address

```
push 3
push 2
push 1
call add
add  esp,0x0c
// add(1, 2, 3)
```

```
add:
push ebp
mov  ebp, esp
sub  esp, 0x04
mov  eax, [ebp+0x08]
add  eax, [ebp+0x0C]
add  eax, [ebp+0x10]
mov  [ebp-0x04], eax
mov  eax, [ebp-0x04]
mov  esp, ebp
pop  ebp
ret
```

Structured Exception Handling (SEH)

aaaddress1@chroot.org

>_SEH

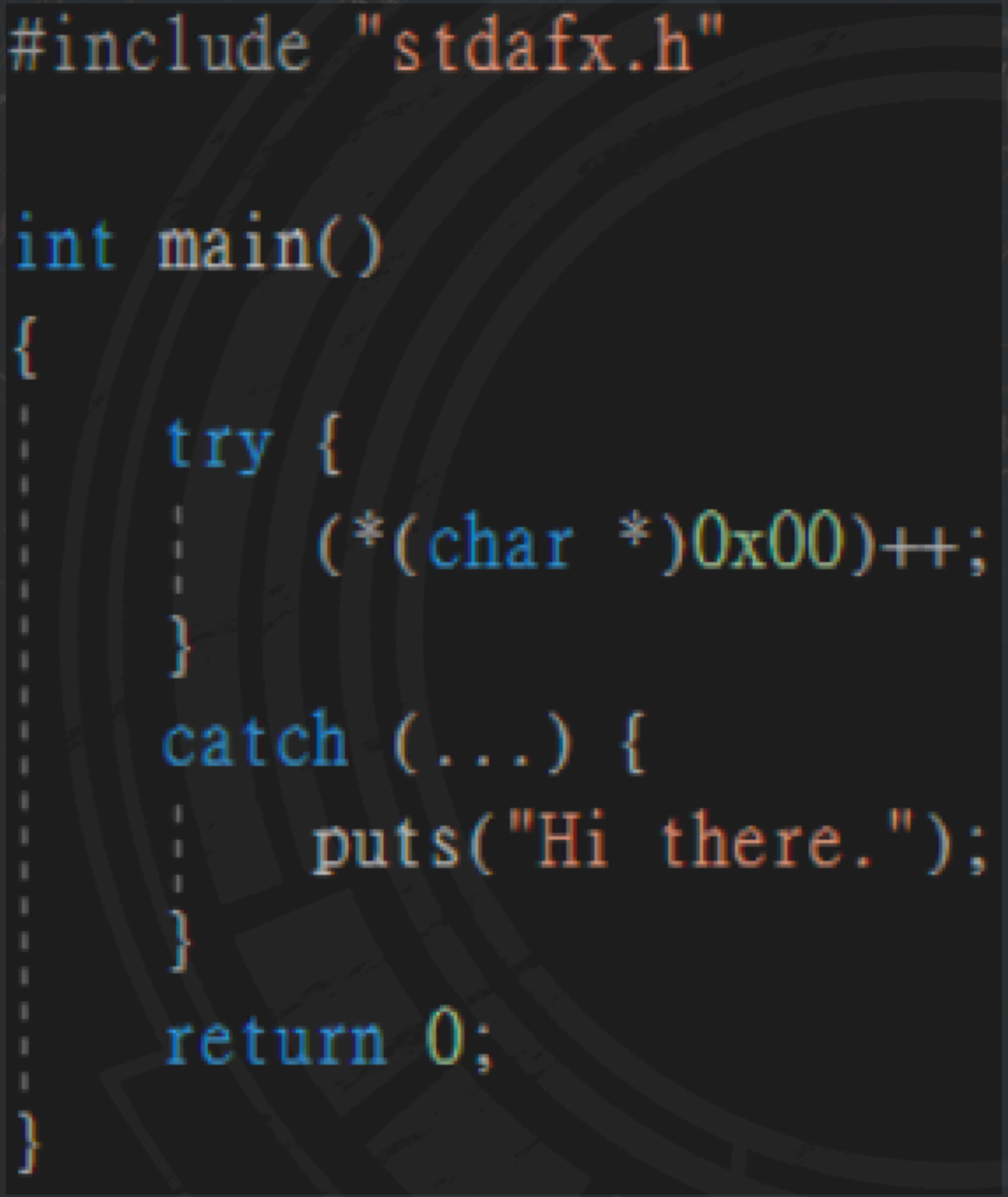
Structured exception handling enables you to have complete control over the handling of exceptions, provides support for debuggers, and is usable across all programming languages and machines. Vectored exception handling is an extension to structured exception handling.

docs.microsoft.com/en-us/windows/desktop/debug/structured-exception-handling

>_Visual C++

Actually SEH is a feature to support try {} catch (...) {}

and...



```
#include "stdafx.h"

int main()
{
    try {
        (*(char *)0x00)++;
    }
    catch (...) {
        puts("Hi there.");
    }
    return 0;
}
```

Microsoft Corporation

Press OK to terminate the application.
Press Cancel to debug the application, which you probably won't do.

s a stub. You can help Microsoft by expanding it.

Critical Error

An error has occurred while trying to display an error message.

This is real.

System report:

Everything is fine. Nothing is ru

Warning

Analysis result:

You are pregnant.

Just an error.

For the sake o

Microsoft Word

Not enough memory. Delete Windows to free memory?

Error

Error

MS Paint Error

Operation completed successfully.
There must be something wrong.

Error

Too many errors. Please close some error message boxes.

Internet Explorer Error

Error 404: File not found.

Internet Explorer will be termi

An unh
Contin
If you c

Could n
files\u

Not an error. For diversity.

NOBODY EXPECTS THE SPANISH INQUISITION!!!

Microsoft Visual Cpluspl

abnormal program t

end of this message for details on invoking
e (JIT) debugging instead of this dialog box.

***** Exception Text *****************
O.DirectoryNotFoundException: Could n
em.IO.__Error.WinIOError(Int32 errorCo
em.IO.FileStream..ctor(String path, FileM

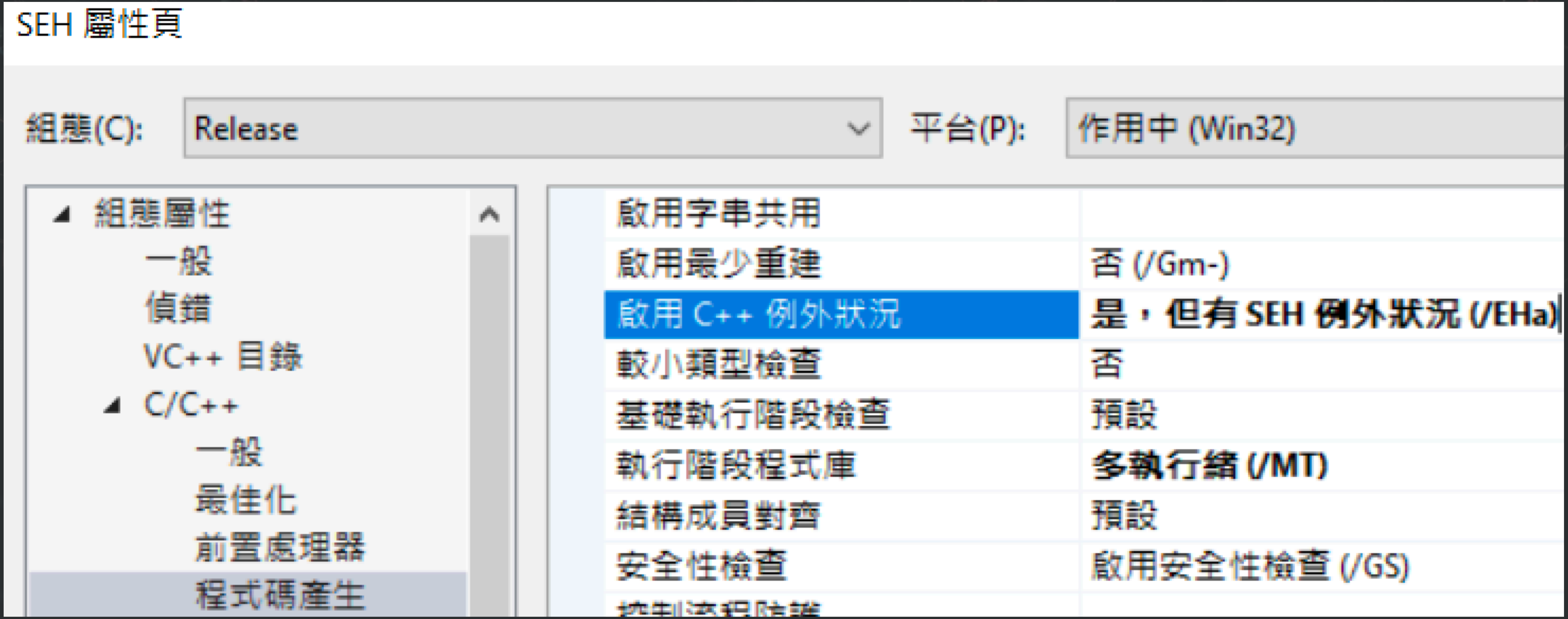
T3h ult1m@t3 spamm0r 3xp3r14nc3

T3h ult1m@t3 spamm0r 3xp3r14nc3

T3h ult1m@t3 spamm0r 3xp3r14nc3

Exclusive!

>_Visual C++



SEH 屬性頁

組態(C): Release

平台(P): 作用中 (Win32)

- 組態屬性
  - 一般
  - 偵錯
  - VC++ 目錄
  - C/C++
    - 一般
    - 最佳化
    - 前置處理器
    - 程式碼產生

啟用字串共用	
啟用最少重建	否 (/Gm-)
啟用 C++ 例外狀況	**是，但有 SEH 例外狀況 (/EHa)**
較小類型檢查	否
基礎執行階段檢查	預設
執行階段程式庫	**多執行緒 (/MT)**
結構成員對齊	預設
安全性檢查	啟用安全性檢查 (/GS)
控制流程防護	

Thread Information Block

—— Break into how Win32 API works

aaaddress1@chroot.org

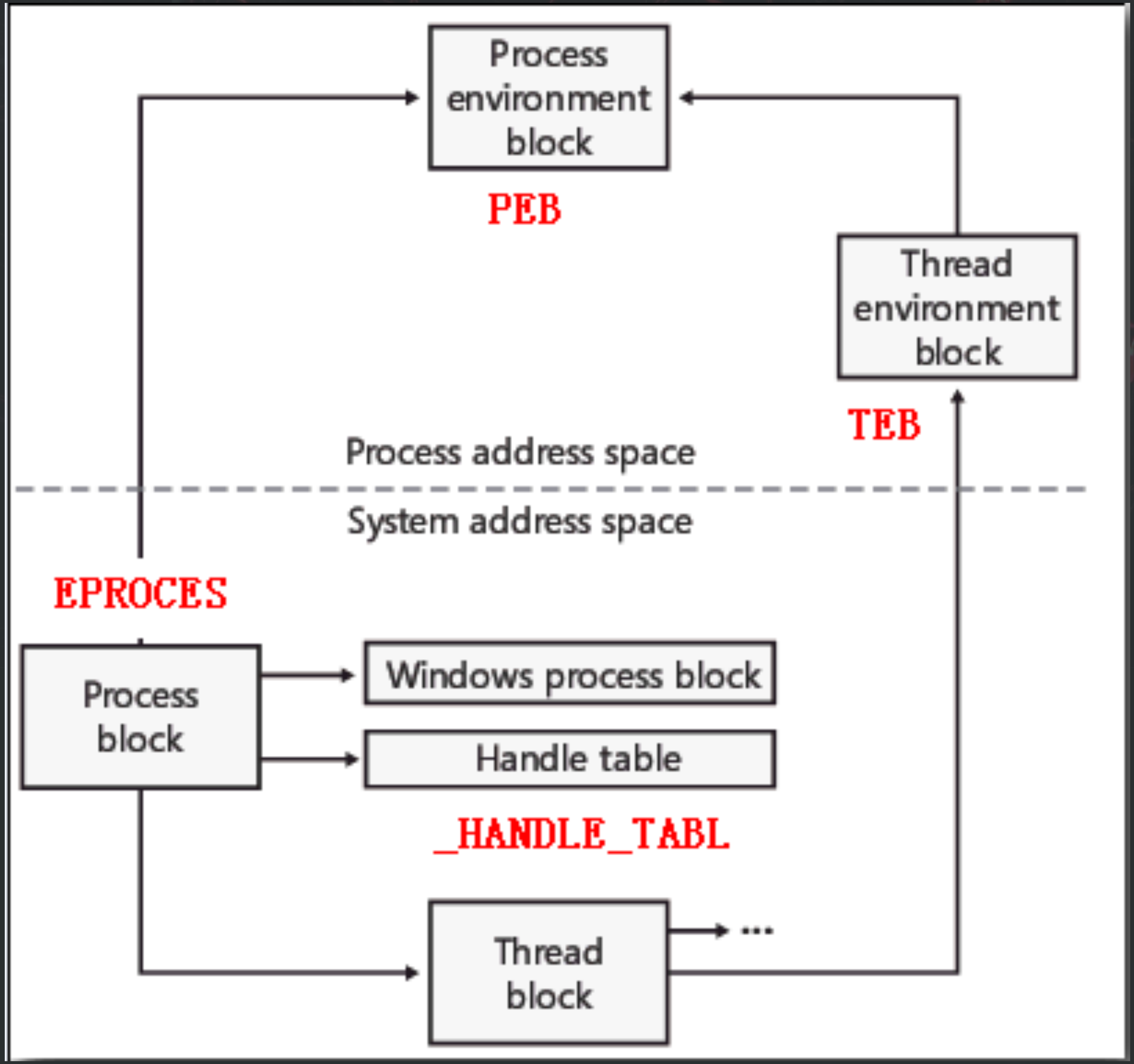
/? TIB

In computing, the Win32 Thread Information Block (TIB) is a data structure in Win32 on x86 that stores information about the currently running thread. This structure is also known as the Thread Environment Block (TEB).

The TIB can be used to get a lot of information on the process without calling Win32 API. Examples include emulating GetLastError(), GetVersion(). Through the pointer to the PEB one can obtain access to the import tables (IAT), process startup arguments, image name, etc. It is accessed from the FS segment register when operating on 32 bits, and from GS in 64 bits.

en.wikipedia.org/wiki/Win32_Thread_Information_Block

/? TIB

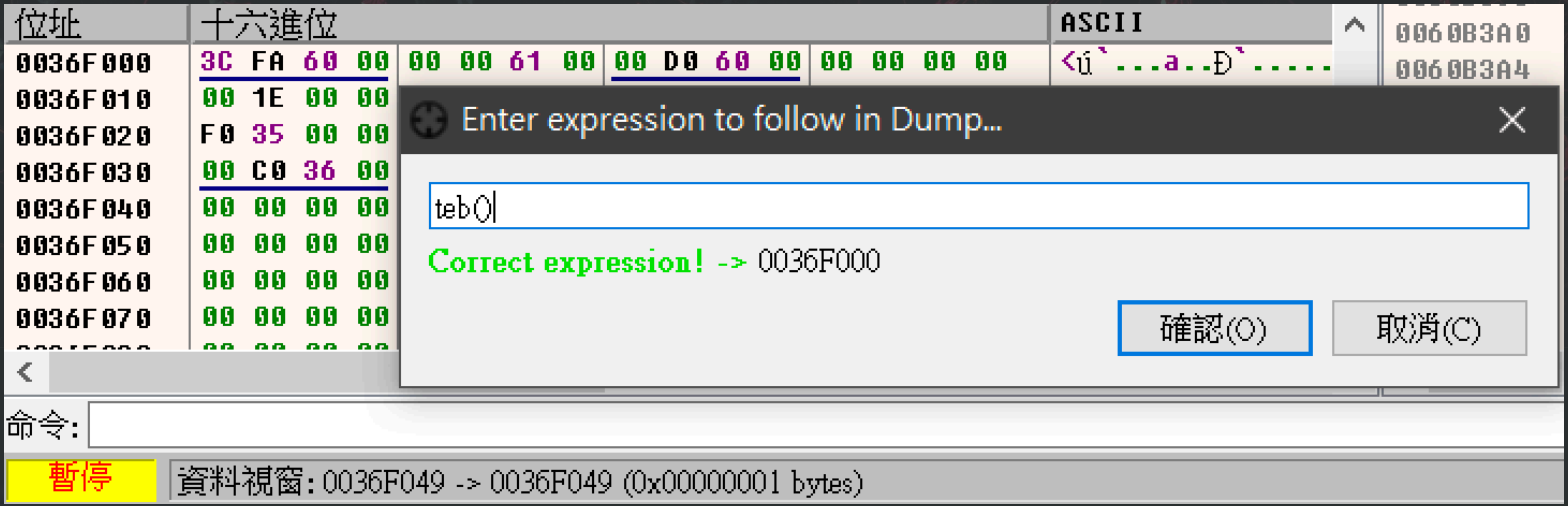

Process environment block
PEB
Thread environment block
TEB
Process address space
System address space
EPROCES
Process block
Windows process block
Handle table
_HANDLE_TABL
Thread block
...

/? TIB # Undocumented

```
struct TEB {
    //NT_TIB structure portion
    EXCEPTION_REGISTRATION*     ExceptionList;  //0x0000 / Current Structured Exception Handling frame
    void*       StackBase;                      //0x0004 / Bottom of stack (high address)
    void*       StackLimit;                     //0x0008 / Ceiling of stack (low address)
    void*       SubSystemTib;                   //0x000C
    union {
        void*   FiberData;                      //0x0010
        DWORD   Version;                        //0x0010
    } dword10;
    void*       ArbitraryUserPointer;           //0x0014
    TEB*        Self;                           //0x0018
    //NT_TIB ends (NT subsystem independent part)

    void*       EnvironmentPointer;         //0x001C
    CLIENT_ID   ClientId;                   //0x0020
    //          ClientId.ProcessId          //0x0020 / value retrieved by GetCurrentProcessId()
    //          ClientId.ThreadId           //0x0024 / value retrieved by GetCurrentThreadId()
    void*       ActiveRpcHandle;            //0x0028
    void*       ThreadLocalStoragePointer;  //0x002C
    PEB*        ProcessEnvironmentBlock;    //0x0030
    ...
```

bytepointer.com/resources/tebpeb32.htm

/? x64dbg



```
位址      十六進位                                          ASCII
0036F000  3C FA 60 00 00 00 61 00 00 D0 60 00 00 00 00 00  <ú`...a..Ð`.....
0036F010  00 1E 00 00
0036F020  F0 35 00 00
0036F030  00 C0 36 00
0036F040  00 00 00 00
0036F050  00 00 00 00
0036F060  00 00 00 00
0036F070  00 00 00 00
```

```
0060B3A0
0060B3A4
```

Enter expression to follow in Dump...

```
teb()
```

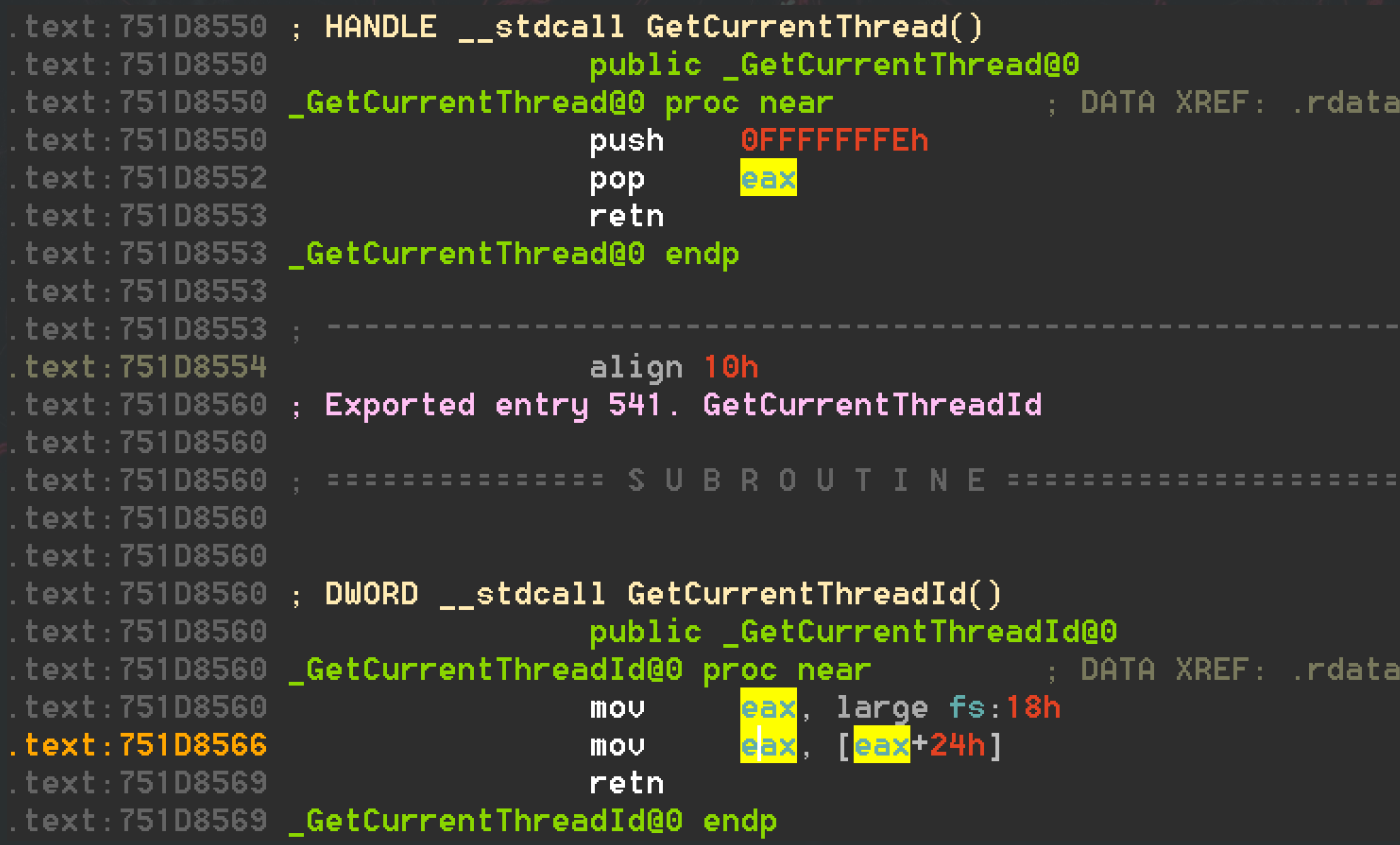
Correct expression! -> 0036F000

命令:

暫停 資料視窗: 0036F049 -> 0036F049 (0x00000001 bytes)

/? C$Windows\Sys32\Kernel32

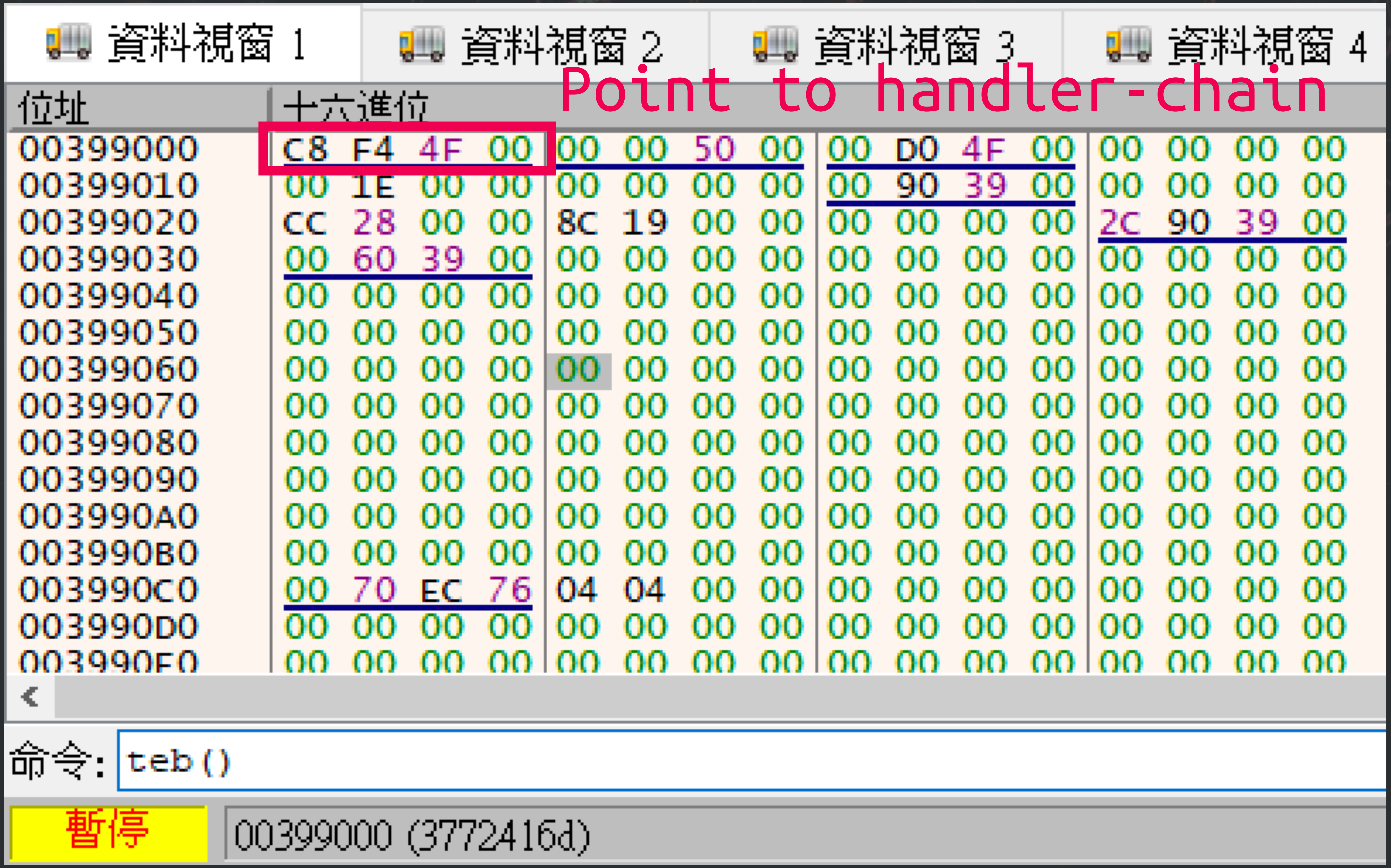
- GetCurrentThread
- GetModuleHandleW
- GetCurrentThreadId
- GetCurrentThread
- IsDebuggerPresent



```
.text:751D8550 ; HANDLE __stdcall GetCurrentThread()
.text:751D8550                 public _GetCurrentThread@0
.text:751D8550 _GetCurrentThread@0 proc near          ; DATA XREF: .rdata
.text:751D8550                 push    0FFFFFFFEh
.text:751D8552                 pop     eax
.text:751D8553                 retn
.text:751D8553 _GetCurrentThread@0 endp
.text:751D8553
.text:751D8553 ; ---------------------------------------------------------
.text:751D8554                 align 10h
.text:751D8560 ; Exported entry 541. GetCurrentThreadId
.text:751D8560
.text:751D8560 ; =============== S U B R O U T I N E =====================
.text:751D8560
.text:751D8560
.text:751D8560 ; DWORD __stdcall GetCurrentThreadId()
.text:751D8560                 public _GetCurrentThreadId@0
.text:751D8560 _GetCurrentThreadId@0 proc near        ; DATA XREF: .rdata
.text:751D8560                 mov     eax, large fs:18h
.text:751D8566                 mov     eax, [eax+24h]
.text:751D8569                 retn
.text:751D8569 _GetCurrentThreadId@0 endp
```

>_x64dbg

We can use the command "teb()" to fetch the current TEB table address.

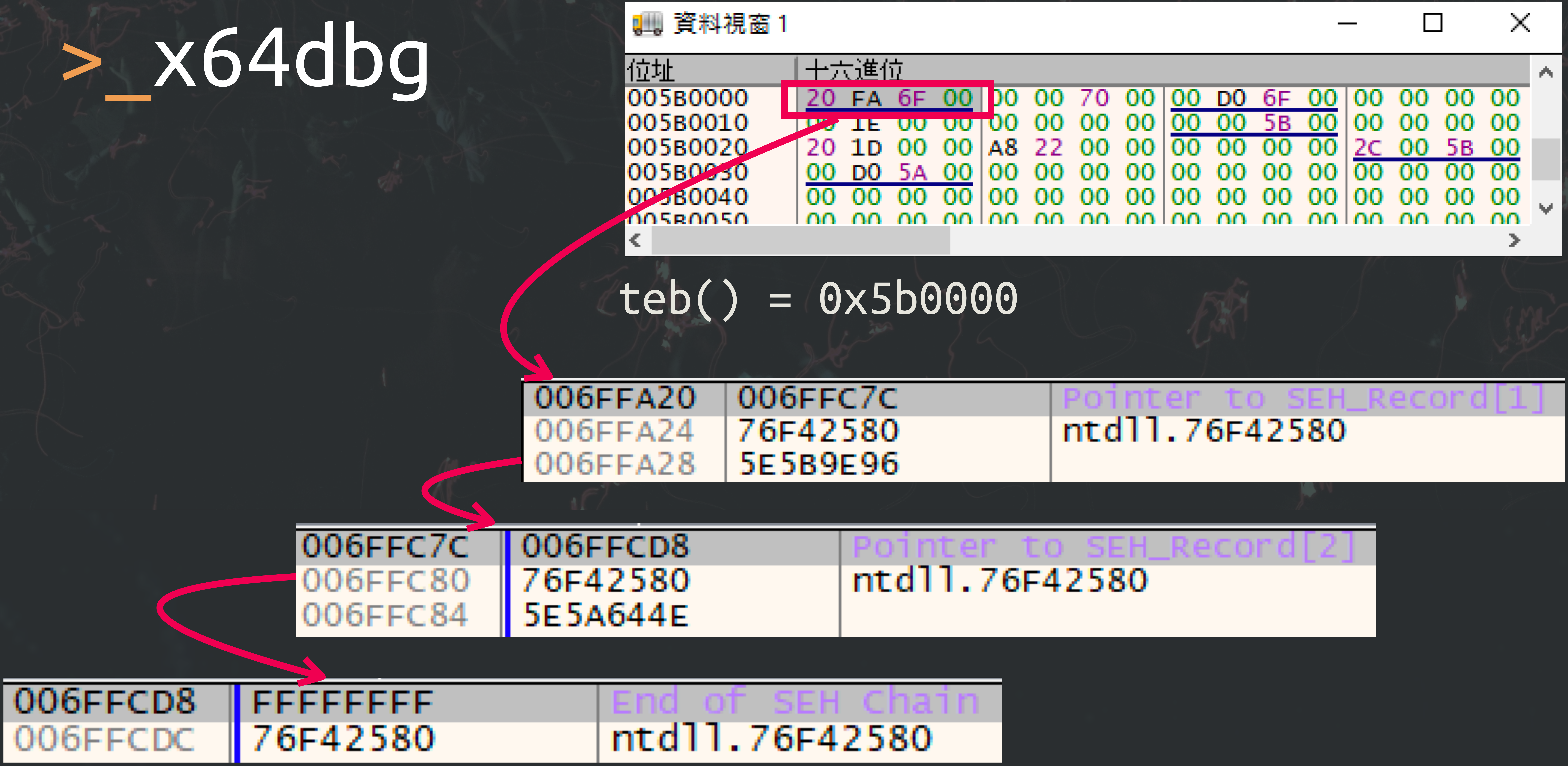


Point to handler-chain

```
位址       十六進位
00399000  C8 F4 4F 00 00 00 50 00 00 D0 4F 00 00 00 00 00
00399010  00 1E 00 00 00 00 00 00 00 90 39 00 00 00 00 00
00399020  CC 28 00 00 8C 19 00 00 00 00 00 00 2C 90 39 00
00399030  00 60 39 00 00 00 00 00 00 00 00 00 00 00 00 00
00399040  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00399050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00399060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00399070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00399080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00399090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
003990A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
003990B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
003990C0  00 70 EC 76 04 04 00 00 00 00 00 00 00 00 00 00
003990D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
003990E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
命令: teb()
00399000 (3772416d)
```

>_x64dbg



資料視窗 1

```
位址        十六進位
005B0000   20 FA 6F 00 00 00 70 00 00 D0 6F 00 00 00 00 00
005B0010   00 1E 00 00 00 00 00 00 00 00 5B 00 00 00 00 00
005B0020   20 1D 00 00 A8 22 00 00 00 00 00 00 2C 00 5B 00
005B0030   00 D0 5A 00 00 00 00 00 00 00 00 00 00 00 00 00
005B0040   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
005B0050   00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

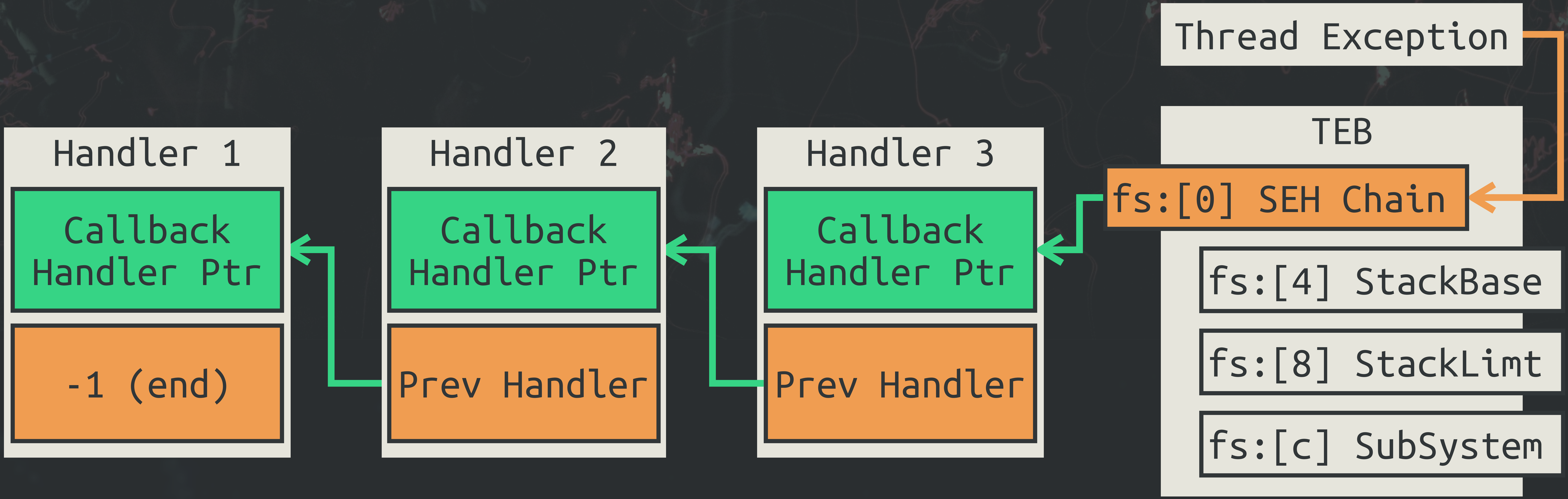
teb() = 0x5b0000

```
006FFA20   006FFC7C   Pointer to SEH_Record[1]
006FFA24   76F42580   ntdll.76F42580
006FFA28   5E5B9E96
```

```
006FFC7C   006FFCD8   Pointer to SEH_Record[2]
006FFC80   76F42580   ntdll.76F42580
006FFC84   5E5A644E
```

```
006FFCD8   FFFFFFFF   End of SEH Chain
006FFCDC   76F42580   ntdll.76F42580
```

>_SEH Record



Thread Exception

TEB

fs:[0] SEH Chain

fs:[4] StackBase

fs:[8] StackLimt

fs:[c] SubSystem

Handler 1

Callback Handler Ptr

-1 (end)

Handler 2

Callback Handler Ptr

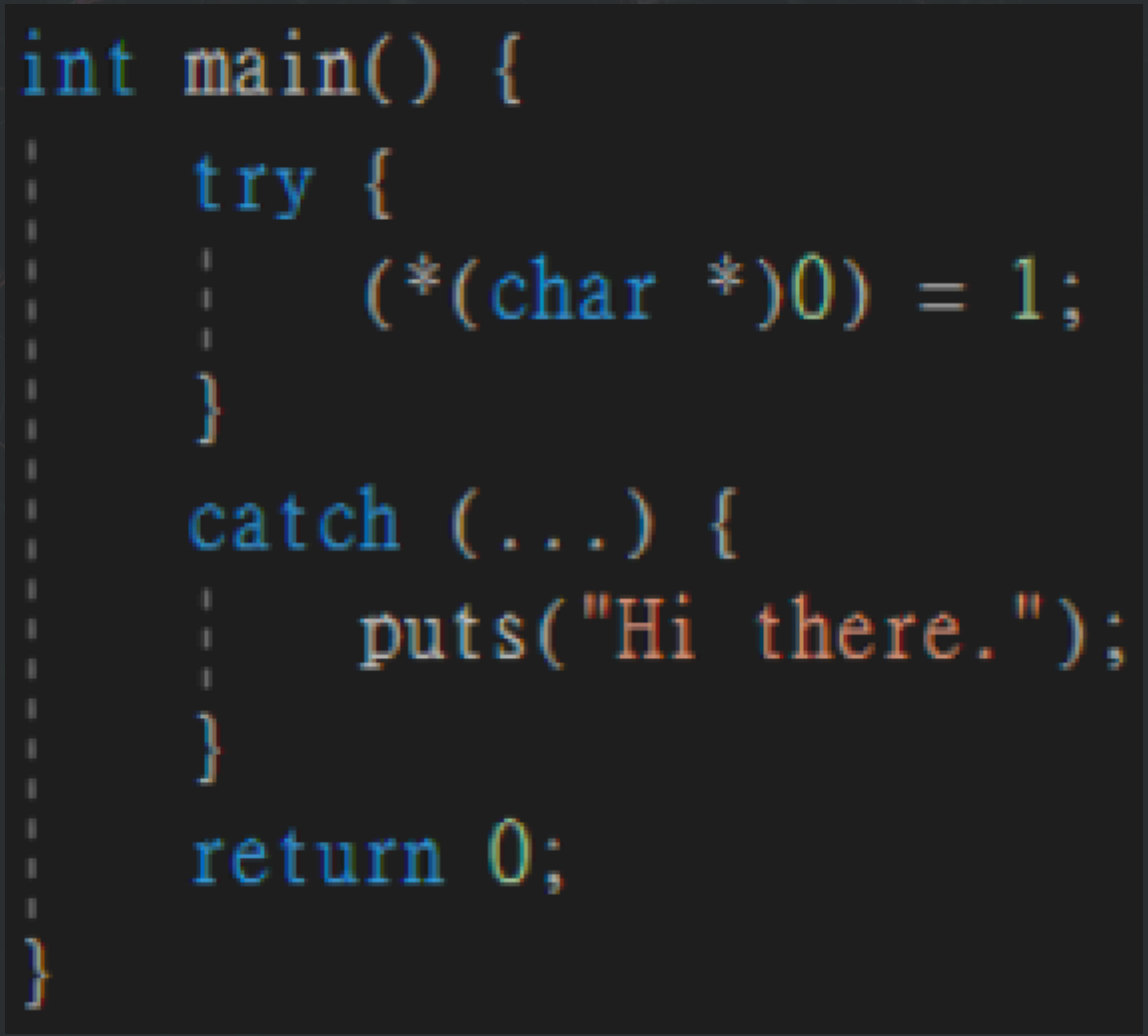
Prev Handler

Handler 3

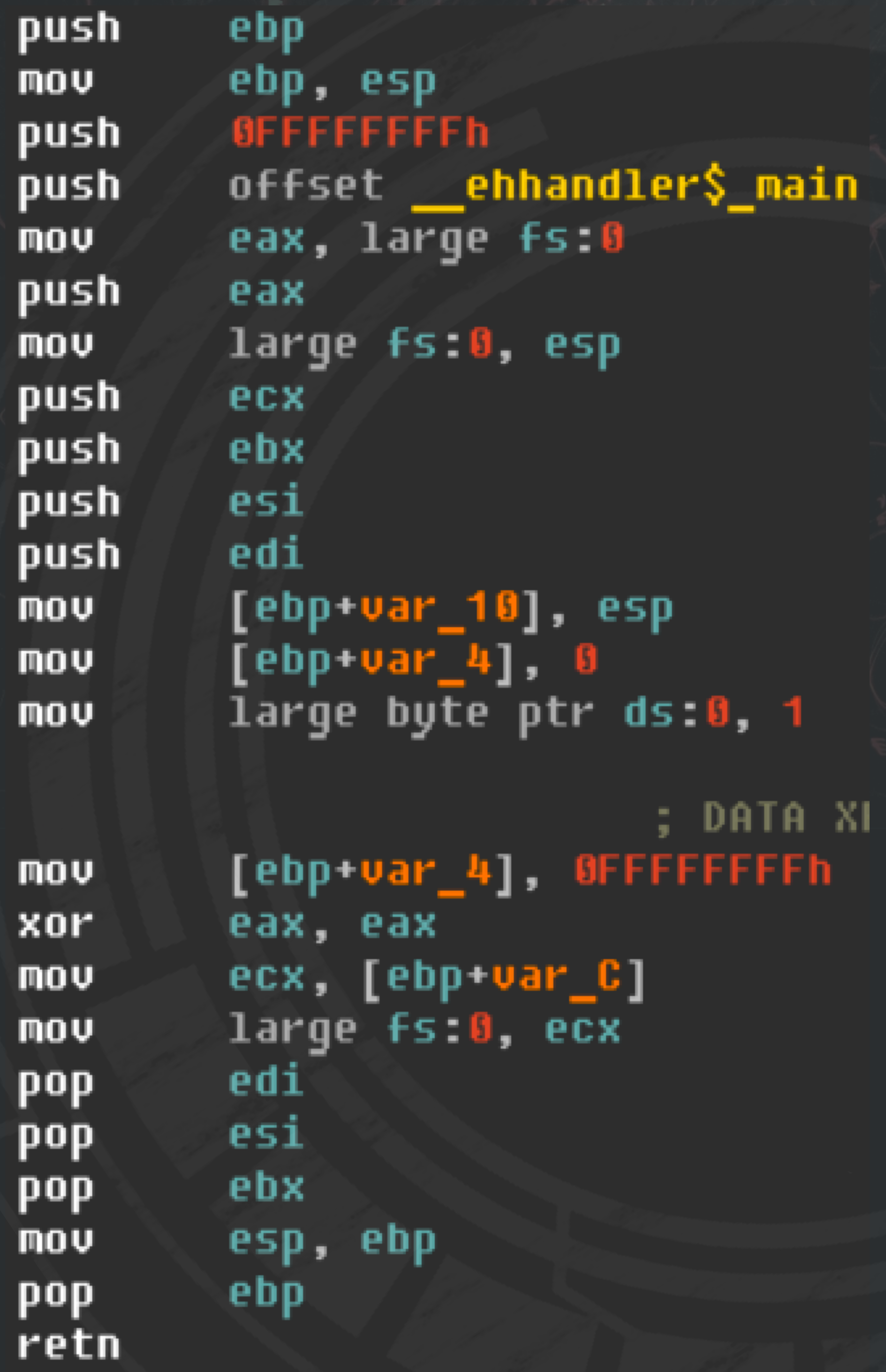
Callback Handler Ptr

Prev Handler

>_SEH



```
int main() {
    try {
        (*(char *)0) = 1;
    }
    catch (...) {
        puts("Hi there.");
    }
    return 0;
}
```



```
push    ebp
mov     ebp, esp
push    0FFFFFFFFh
push    offset __ehhandler$_main
mov     eax, large fs:0
push    eax
mov     large fs:0, esp
push    ecx
push    ebx
push    esi
push    edi
mov     [ebp+var_10], esp
mov     [ebp+var_4], 0
mov     large byte ptr ds:0, 1

                                ; DATA XI
mov     [ebp+var_4], 0FFFFFFFFh
xor     eax, eax
mov     ecx, [ebp+var_C]
mov     large fs:0, ecx
pop     edi
pop     esi
pop     ebx
mov     esp, ebp
pop     ebp
retn
```

>_SEH

```
push    ebp                    The begin of function
mov     ebp, esp
push    offset __ehhandler$_main
push    fs:[0]                                         Register a handler
mov     fs:[0], esp
mov     [0], 1        <--- Function codes
xor     eax, eax      <----- Return value
mov     ecx, [esp]
mov     large fs:0, ecx                                Unregister a handler
mov     esp, ebp
pop     ebp                    The end of function
retn
```

>_Stack Frame

ebp (current stack frame)
esp
ebp
ebp+4
ebp+8
ebp+0x0c
ebp+0x10
buffer
SEH record
canery
old ebp
ret addr
arg1
arg2
arg3
Low Address
(stack)
High Address
SEH record
Previous SEH Record addr
Current Handler

>_Buffer Overflow

ebp (current stack frame)
esp
ebp
ebp+4
ebp+8
ebp+0x0c
ebp+0x10
buffer
SEH record
canery
old ebp
ret addr
arg1
arg2
arg3
Low Address (stack)
High Address
SEH record
Previous SEH Record addr
Current Handler

>_Buffer Overflow

ebp (current stack frame)
esp
Buffer Overflow from low addr to high addr
ebp
ebp+4
ebp+8
ebp+0x0c
ebp+0x10
SEH record
canery
old ebp
ret addr
arg1
arg2
arg3
buffer
Low Address
(stack)
High Address
SEH record
Previous SEH Record addr
Current Handler

>_Buffer Overflow

ebp (current stack frame)
memory out of bounds
esp
Buffer Overflow from low addr to high addr
ebp
ebp+4
ebp+8
ebp+0x0c
ebp+0x10
SEH record
canery
old ebp
ret addr
arg1
arg2
arg3
buffer
buffer
Low Address (stack)
High Address
SEH record
Previous SEH Record addr
Current Handler

Lab 2:
Knock down the handler

aaaddress1@chroot.org

Github

Slide

Facebook

@aaaddress1

Windows Reversing Basic

aaaddress1@chroot.org