

汇编语言与逆向工程

北京邮电大学
崔宝江

北邮网安学院 崔宝江



第四章 逆向初体验

- ① 一. HelloWorld程序逆向分析
- ② 二. 快速定位关键函数
- ③ 三. 逆向牛刀小试



一. HelloWorld程序逆向分析

@1. HelloWorld程序

@2. IDA载入分析

@3. 动态调试



1. HelloWorld程序

@用VC++ 6.0编译Hello World程序

○（VC++ 6.0 Debug版本）

4-1

```
#include<windows.h>
```

```
Int main() {
```

```
    MessageBox(NULL,"Welcome to the world of reverse!","Helloworld!",MB_OK);
```

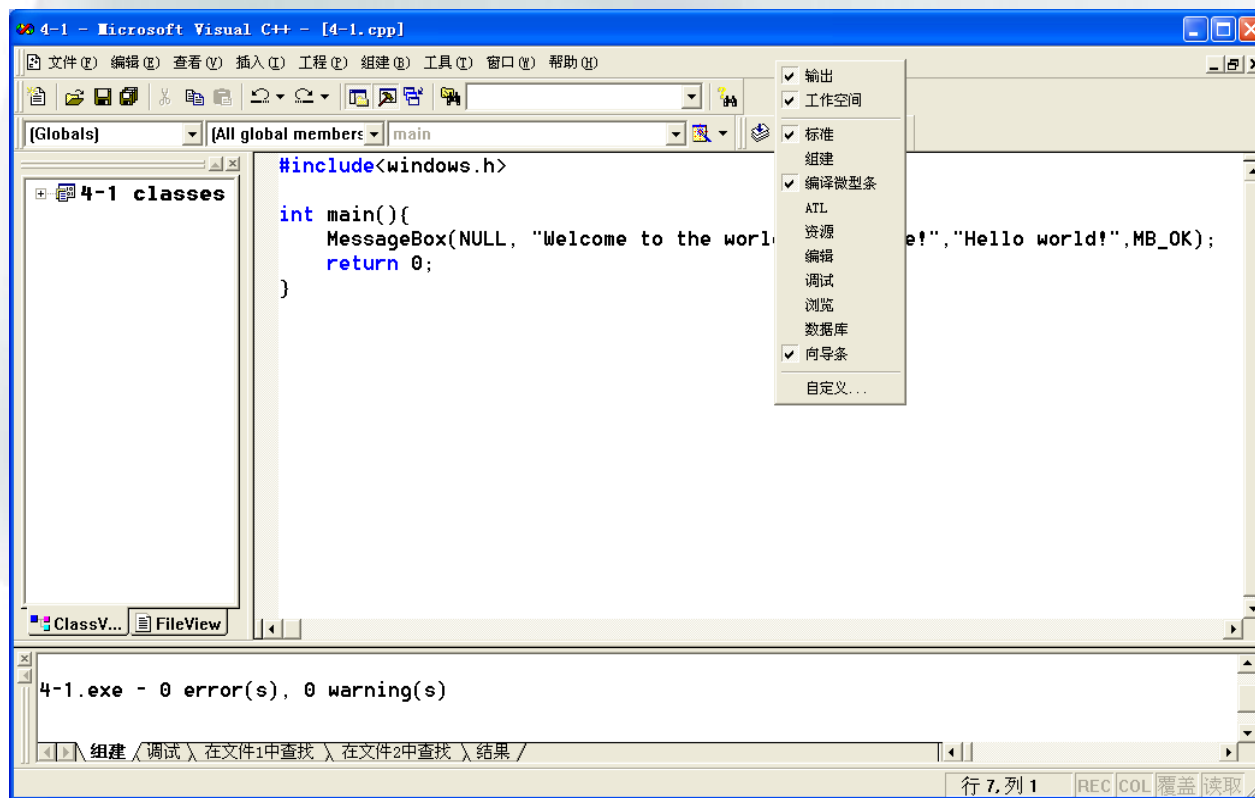
```
    return 0;
```

```
}
```



1. HelloWorld程序

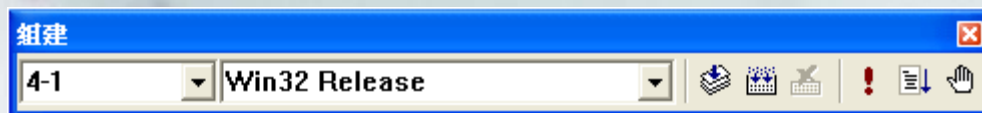
□ 在菜单栏中鼠标右击选中“组建”一栏



1. HelloWorld程序

□ 选择编译Release版本的程序

- 相对于Debug版本，Release版本的程序更加简洁，方便调试



- 运行一下程序，会弹出如下图的框



一. HelloWorld程序逆向分析

@1. HelloWorld程序

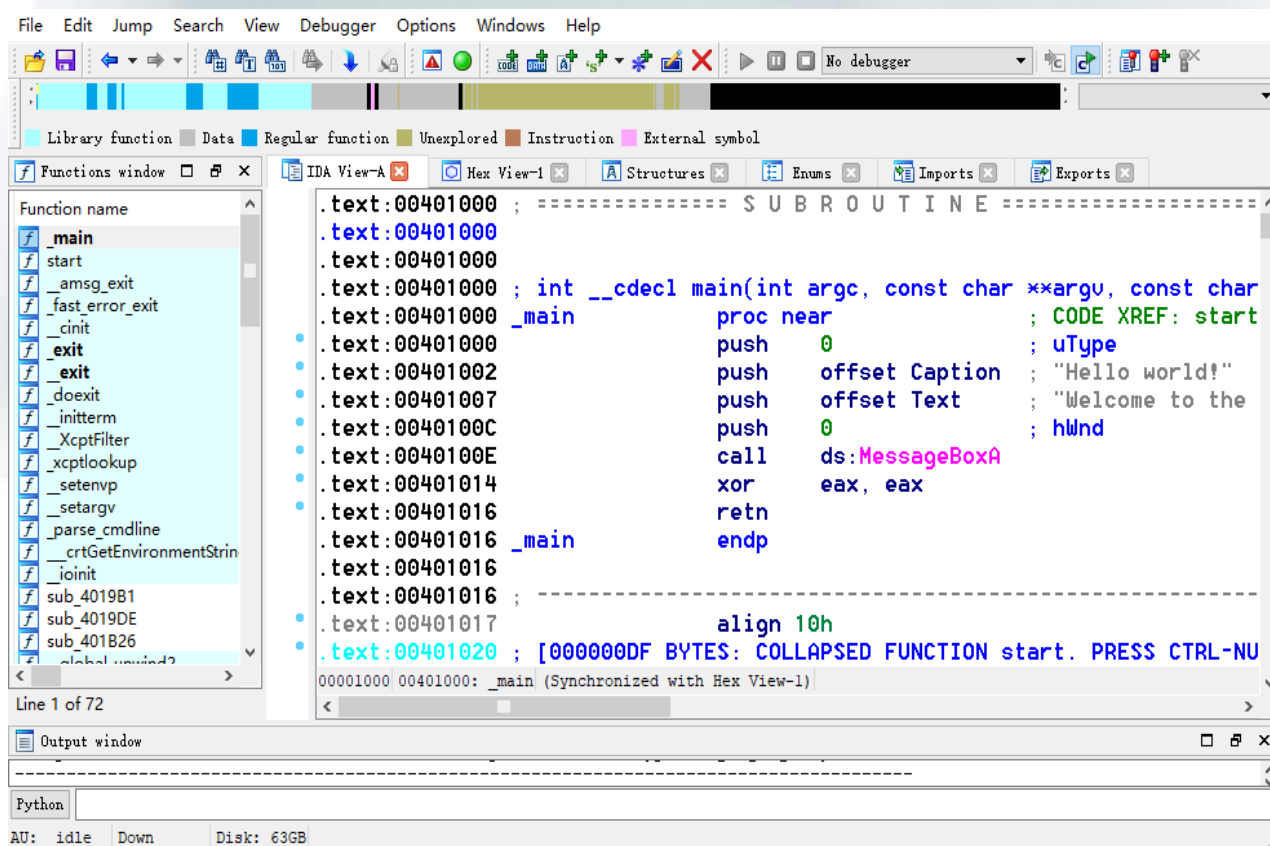
@2. IDA载入分析

@3. 动态调试



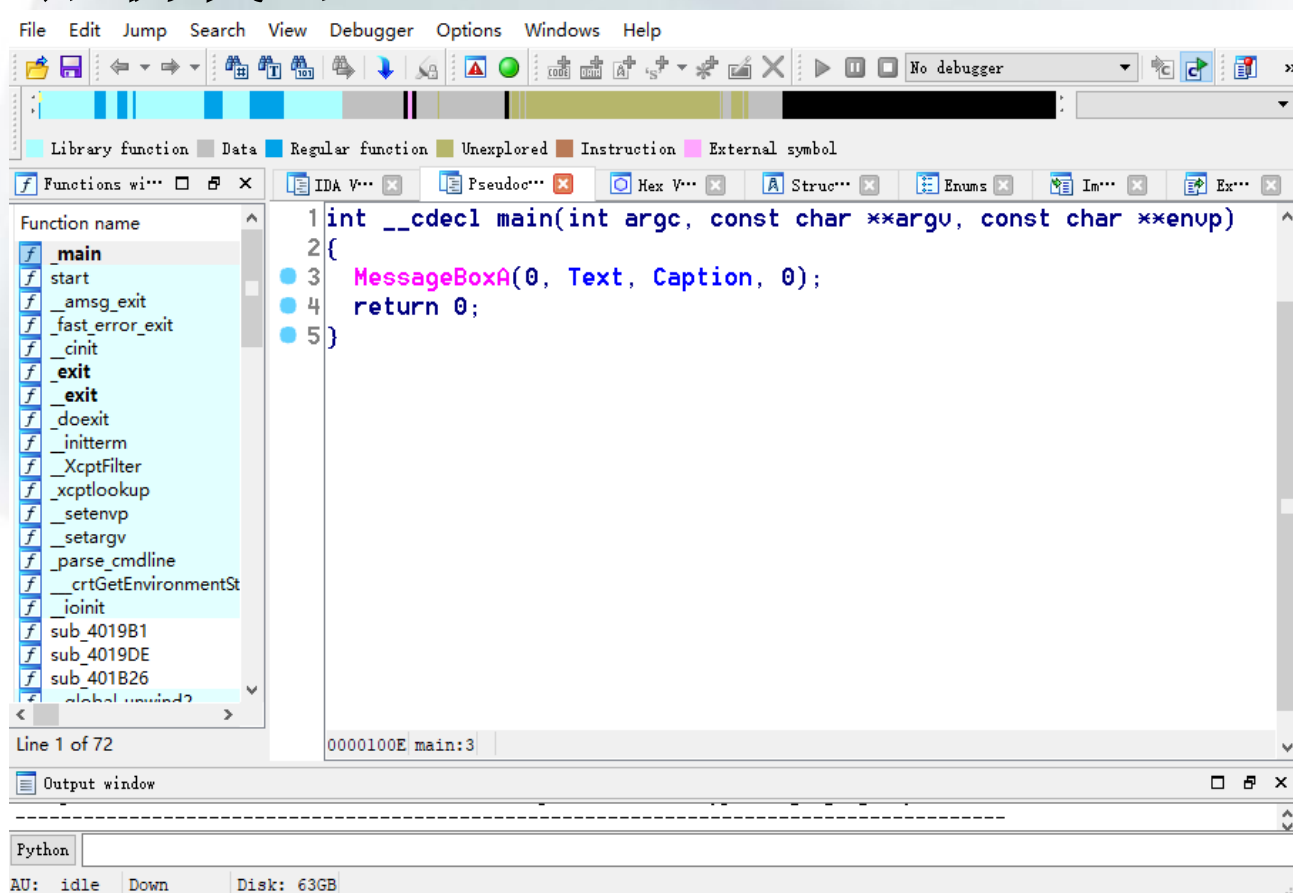
2. IDA载入分析

□ 由于是VC++ 6.0编译的程序，所以用IDA加载该程序后，能识别出main函数



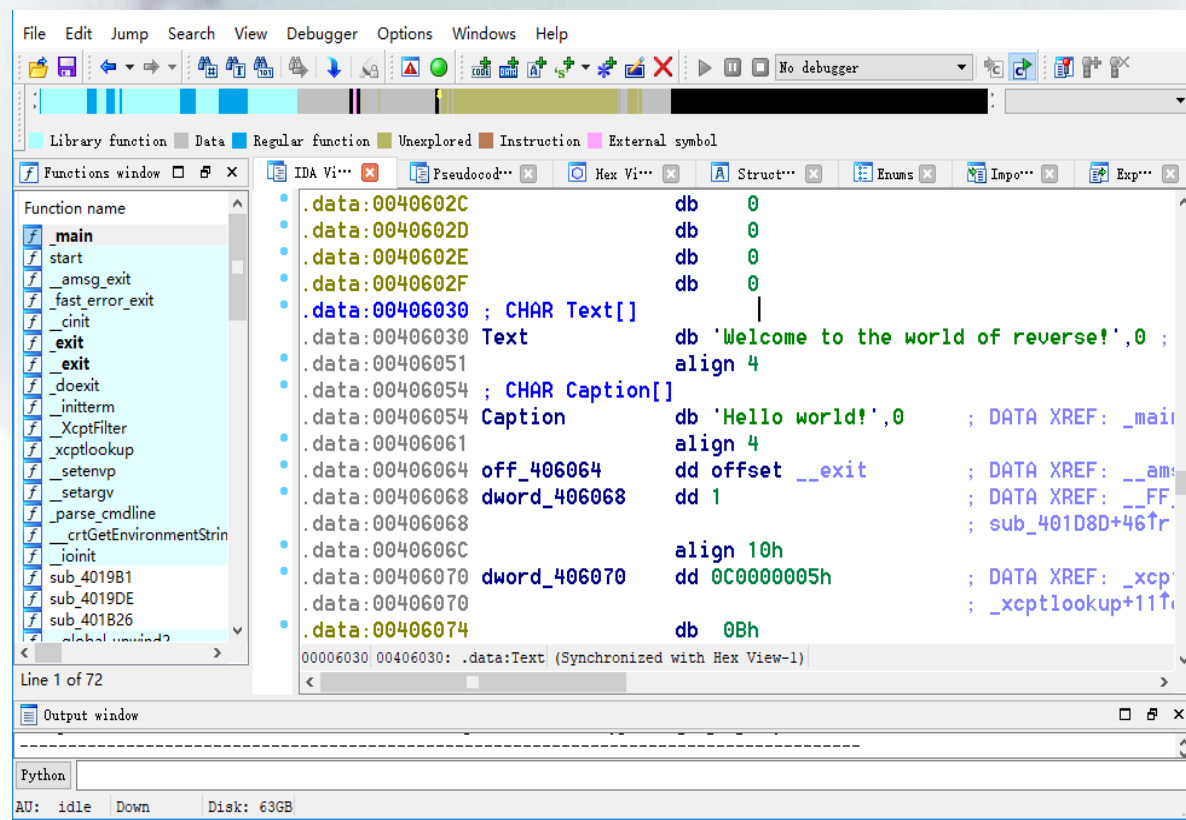
2. IDA载入分析

- ❑ 鼠标选中main函数里的任一区域，F5反编译生成伪代码



2. IDA载入分析

- 程序的功能是调用MessageBoxA函数输出消息框，消息框的内容和标题通过双击Text和Caption变量即可查看



一. HelloWorld程序逆向分析

@1. HelloWorld程序

@2. IDA载入分析

@3. 动态调试



3. 动态调试

- ❑ 动态调试的目的，是利用逆向分析方法，找到主要功能函数，并分析出其功能
- ❑ 从程序入口点开始
 - 使用OllyICE动态调试Hello World程序，来确定main函数的位置
 - 用调试器载入Hello World程序，程序停在地址0x401020处
 - 这便是Hello World程序的代码入口点（EP, Entry Point），即该程序最先执行的代码的起始位置



暂停

```

00401020 55      push    ebp
00401021 8BEC    mov     ebp, esp
00401023 6AFF    push    -1
00401025 68A8504000 push    004050A8
0040102A 687C1C4000 push    00401C7C
0040102F 64:A1 00000000 mov     eax, dword ptr fs:[0]
00401035 50      push    eax
00401036 64:8925 00000000 mov     dword ptr fs:[0], esp
0040103D 83EC 10    sub     esp, 10
00401040 53      push    ebx
00401041 56      push    esi
00401042 57      push    edi
00401043 8965 E8    mov     dword ptr [ebp-18], esp
00401046 FF15 0C504000 call    dword ptr [&KERNEL32.GetVe
0040104C 33D2     xor     edx, edx
0040104E 8AD4     mov     dl, ah
00401050 8915 24854000 mov     dword ptr [408524], edx
00401056 8BC8     mov     ecx, eax
00401058 81E1 FF000000 and     ecx, 0FF
0040105E 890D 20854000 mov     dword ptr [408520], ecx
00401064 C1E1 08    shl     ecx, 8
00401067 03CA     add     ecx, edx
00401069 890D 1C854000 mov     dword ptr [40851C], ecx
0040106F C1E8 10    shr     eax, 10

```

ebp=0012FFF0

```

00406000 00 00 00 00 00 00 00 00 00 00 00 00 3F 25 40 00 ..
00406010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
00406020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..
00406030 57 65 6C 63 6F 6D 65 20 74 6F 20 74 68 65 20 77 We
00406040 6F 72 6C 64 20 6F 66 20 72 65 76 65 72 73 65 21 or
00406050 00 00 00 00 48 65 6C 6C 6F 20 77 6F 72 6C 64 21 ..
00406060 00 00 00 00 86 11 40 00 01 00 00 00 00 00 00 00 ..
00406070 05 00 00 C0 0B 00 00 00 00 00 00 00 1D 00 00 C0 Y
00406080 04 00 00 00 00 00 00 00 96 00 00 C0 04 00 00 00 |.
00406090 00 00 00 00 8D 00 00 C0 08 00 00 00 00 00 00 00 ..
004060A0 8E 00 00 C0 08 00 00 00 00 00 00 00 8F 00 00 C0 ?
004060B0 08 00 00 00 00 00 00 00 90 00 00 C0 08 00 00 00 ■
004060C0 00 00 00 00 91 00 00 C0 08 00 00 00 00 00 00 00 ..

```

寄存器 (FPU)

```

EAX 00000000
ECX 0012FFB0
EDX 7C92E4F4 ntdll.KiFastSystemCallRet
EBX 7FFD4000
ESP 0012FFC4
EBP 0012FFF0
ESI FFFFFFFF
EDI 7C930208 ntdll.7C930208
EIP 00401020 4-1.<模块入口点>
C 0 ES 0023 32位 0(FFFFFFFF)
P 1 CS 001B 32位 0(FFFFFFFF)
A 0 SS 0023 32位 0(FFFFFFFF)
Z 1 DS 0023 32位 0(FFFFFFFF)
S 0 FS 003B 32位 7FFDF000(FFF)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_MOD_NOT_FOUND (0000007E)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty -UNORM 8BB0 01050104 00000000
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0

```

```

0012FFC4 7C817067 返回到 kernel32.7C817067
0012FFC8 7C930208 ntdll.7C930208
0012FFCC FFFFFFFF
0012FFD0 7FFD4000
0012FFD4 80545BFD
0012FFD8 0012FFC8
0012FFDC 8631E020
0012FFE0 FFFFFFFF SEH 链尾部
0012FFE4 7C839AC0 SE处理程序
0012FFE8 7C817070 kernel32.7C817070
0012FFEC 00000000
0012FFF0 00000000
0012FFF4 00000000

```

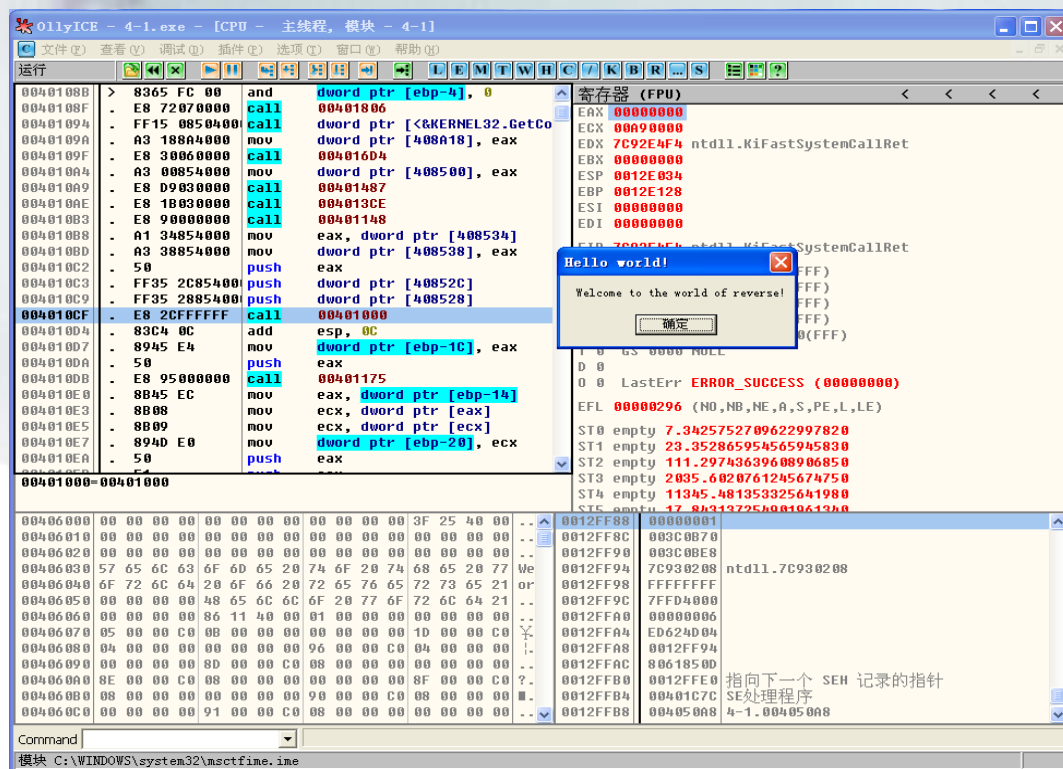
Command

程序入口点

3. 动态调试

□ 单步跟踪

- 一直用F8（单步步过）命令调试Hello World程序
- 直到运行到地址0x4010CF处，程序弹了一个消息框



3. 动态调试

- ❑ 可以猜想，该地址处调用的**0x401000**函数可能就是我们要找的**main**函数
- ❑ 为了验证猜想，在**0x4010CF**地址用**F2**命令下断点，重新开始调试程序
- ❑ **F9**命令执行到断点处，然后**F7**单步步入，就进入到了**0x401000**函数中



OlllyICE - 4-1.exe - [CPU - 主线程, 模块 - 4-1]

文件(F) 查看(V) 调试(D) 插件(P) 选项(O) 窗口(W) 帮助(H)

暂停

地址	汇编	注释
00401000	6A 00	push 0
00401002	68 54604000	push 00406054
00401007	68 30604000	push 00406030
0040100C	6A 00	push 0
0040100E	FF15 9C504000	call dword ptr [<USER32.MessageBoxA>]
00401014	33C0	xor eax, eax
00401016	C3	ret
00401017	90	nop
00401018	90	nop
00401019	90	nop
0040101A	90	nop
0040101B	90	nop
0040101C	00	nop
0040101D		
0040101E		
0040101F		
00401020		
00401021		
00401023		
00401025		
0040102A	08 7C7C4000	push 00407C7C
0040102F	64:A1 00000000	mov eax, dword ptr fs:[0]
00401035	50	push eax
00401036	64:8925 0000	mov dword ptr fs:[0], esp

寄存器 (FPU)

EAX 003C0BE8

ECX 003C0768

EDX 003C0000

EBX 7FFD6000

ESP 0012FF84

EBP 0012FFC0

ESI FFFFFFFF

EDI 7C930208 ntdll.7C930208

EIP 00401000 4-1.00401000

C 0 ES 0023 32位 0(FFFFFFFF)

D 4 SS 0040 32位 0(FFFFFFFF)

00007E)

本地调用来自 <模块入口点>+0AF

地址	汇编	注释
00406000	00 00 00 00 00 00 00 00 00 00 00 00 3F 25 40 00?%
00406010	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00406020	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00406030	57 65 6C 63 6F 6D 65 20 74 6F 20 74 68 65 20 77	Welcome to the
00406040	6F 72 6C 64 20 6F 66 20 72 65 76 65 72 73 65 21	orld of revers
00406050	00 00 00 00 48 65 6C 6C 6F 20 77 6F 72 6C 64 21	...Hello worl
00406060	00 00 00 00 86 11 40 00 01 00 00 00 00 00 00 00	...?@.f.....
00406070	05 00 00 C0 0B 00 00 00 00 00 00 00 1D 00 00 C0	Y.?......
00406080	04 00 00 00 00 00 00 00 96 00 00 C0 04 00 00 00	!.....?..
00406090	00 00 00 00 8D 00 00 C0 08 00 00 00 00 00 00 00	...?..?
004060A0	8E 00 00 C0 08 00 00 00 00 00 00 00 8F 00 00 C0	?..?..?..?
004060B0	08 00 00 00 00 00 00 00 90 00 00 C0 08 00 00 00	■.....?..
004060C0	00 00 00 00 91 00 00 C0 08 00 00 00 00 00 00 00?..?

0012FF84 00401004 返回到 4-1.<模块入口点>+0AF

0012FF88 00000001

0012FF8C 003C0B70

0012FF90 003C0BE8

0012FF94 7C930208 ntdll.7C930208

0012FF98 FFFFFFFF

0012FF9C 7FFD6000

0012FFA0 00000006

0012FFA4 ED7BED04

0012FFA8 0012FF94

0012FFAC 8061850D

0012FFB0 0012FFE0 指向下一个 SEH 记录的指针

0012FFB4 00401C7C SE处理程序

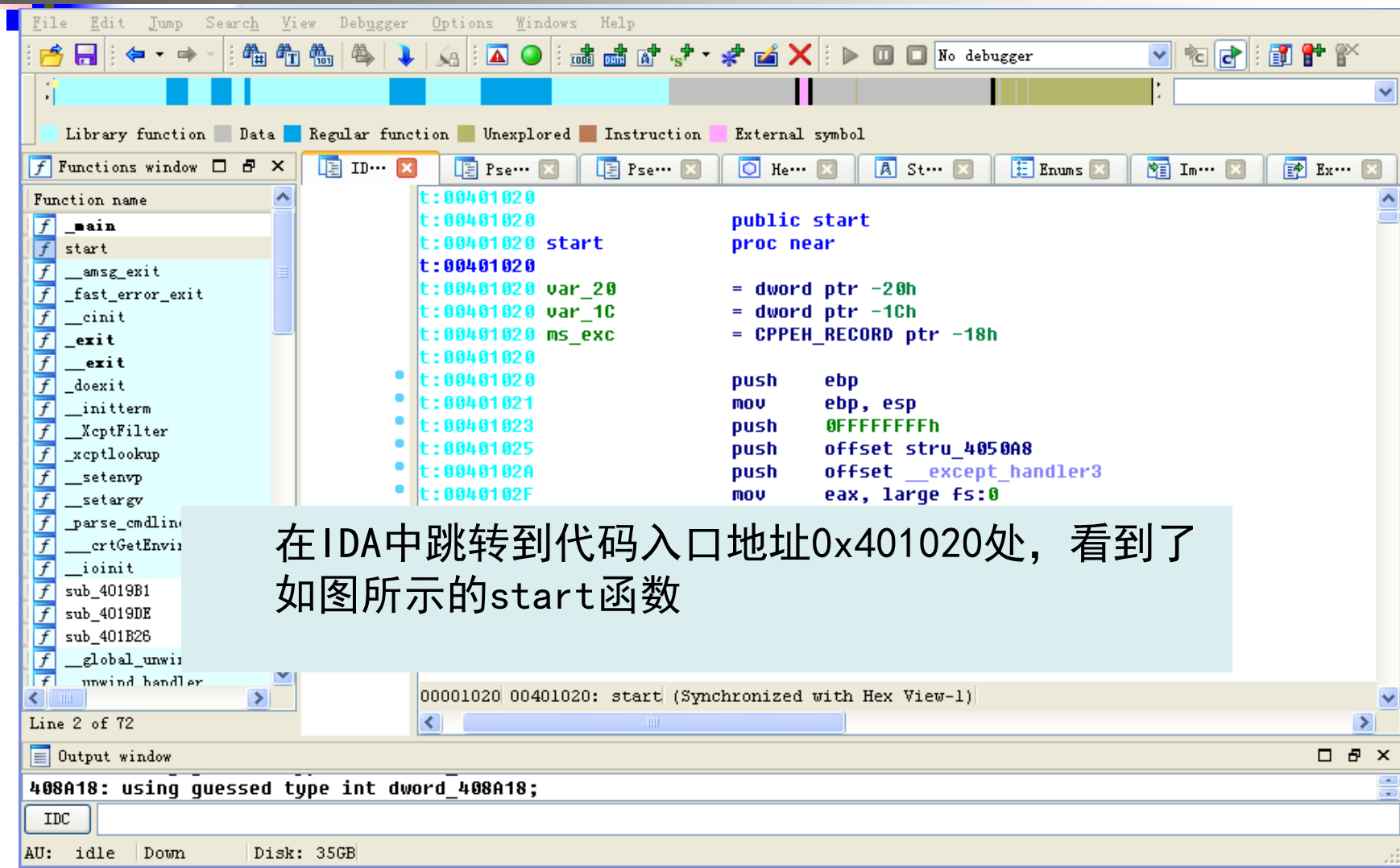
Command

3. 动态调试

- ❑ 学C和C++的时候，总说main函数是程序的入口函数，但我们在调试的时候发现，从程序的代码入口点到main函数执行之前，程序明明还做了其他操作，这些操作又是什么呢？
- ❑ 分析start函数

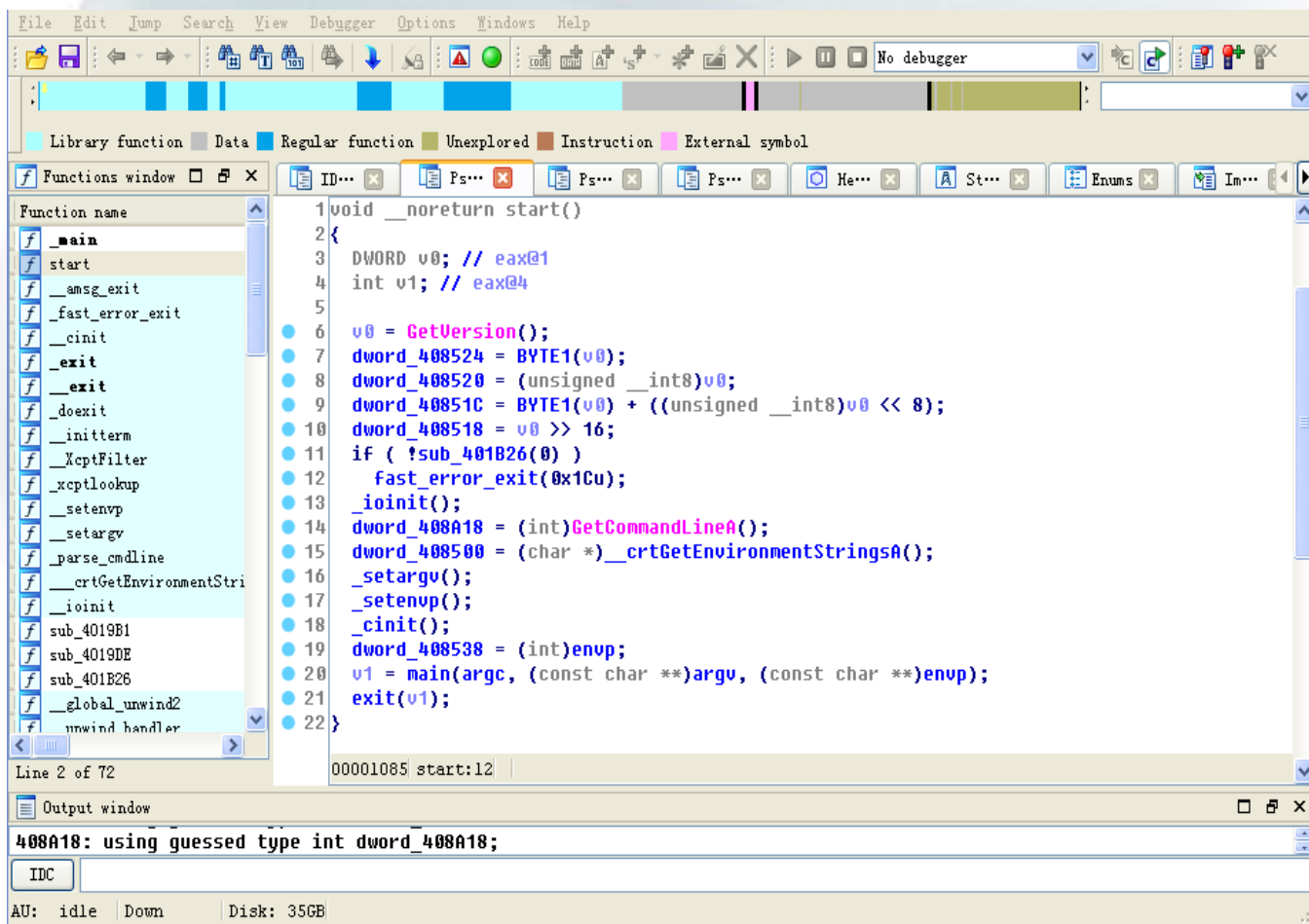


3. 动态调试



3. 动态调试

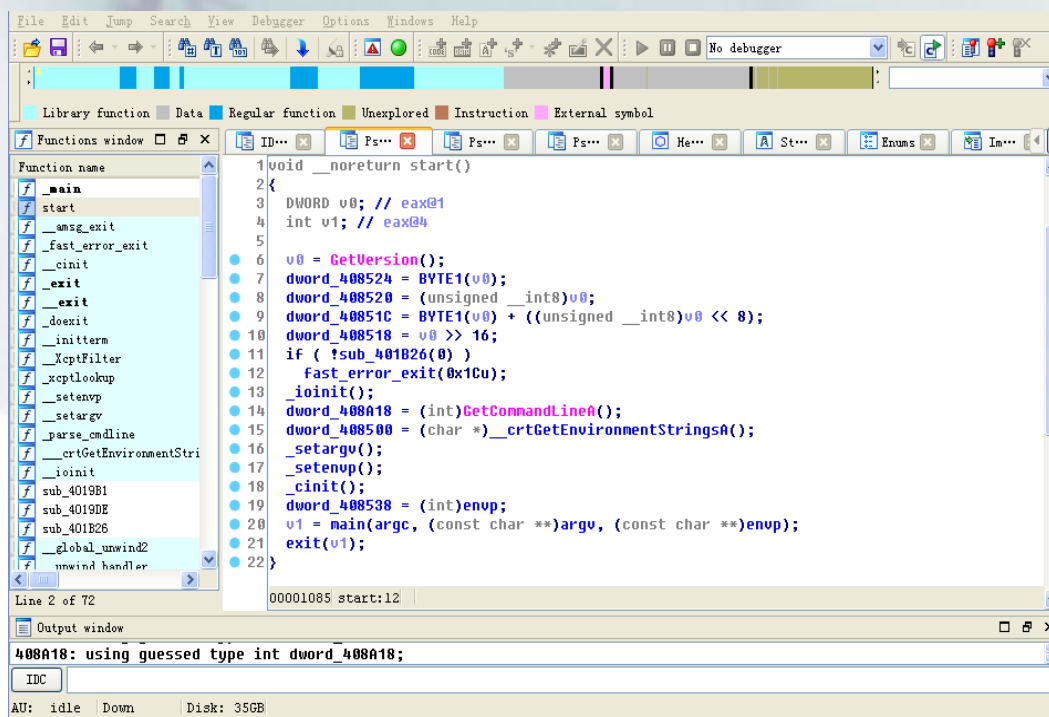
❑ F5生成的伪代码



3. 动态调试

❑ 分析start函数的伪代码可以得知，它在执行一些初始化操作

○ 如获取命令行参数、获取环境变量值、初始化全局变量等，一切准备工作完成之后，再调用main函数



The screenshot shows a debugger window with the following components:

- File Edit Jump Search View Debugger Options Windows Help** (Menu bar)
- Library function Data Regular function Unexplored Instruction External symbol** (Filter tabs)
- Functions window** (Left pane): Lists functions including `_main`, `start`, `_amsi_exit`, `_fast_error_exit`, `_cinit`, `_exit`, `_doexit`, `_initterm`, `_xcptFilter`, `_xcptlookup`, `_setenvp`, `_setargv`, `_parse_cmdline`, `_crtGetEnvironmentStrings`, `_iointit`, `sub_4019B1`, `sub_4019D2`, `sub_401B26`, `_global_unwind2`, and `_unwind_handler`.
- Code window** (Main pane): Displays the assembly code for the `start` function. The code is as follows:

```
1 void __noreturn start()
2 {
3     DWORD v0; // eax@1
4     int v1; // eax@4
5
6     v0 = GetVersion();
7     dword_408524 = BYTE1(v0);
8     dword_408520 = (unsigned __int8)v0;
9     dword_40851C = BYTE1(v0) + ((unsigned __int8)v0 << 8);
10    dword_408518 = v0 >> 16;
11    if ( !sub_401B26(0) )
12        FastErrorExit(0x1Cu);
13    _iointit();
14    dword_408A18 = (int)GetCommandLineA();
15    dword_408500 = (char *)_crtGetEnvironmentStringsA();
16    _setargv();
17    _setenvp();
18    _cinit();
19    dword_408538 = (int)envp;
20    v1 = main(argc, (const char **)argv, (const char **)envp);
21    exit(v1);
22 }
```
- Output window** (Bottom pane): Shows the message `408A18: using guessed type int dword_408A18;`.
- Status bar** (Bottom): Shows `AU: idle Down Disk: 35GB`.



第四章 逆向初体验

- ① 一. HelloWorld程序逆向分析
- ② 二. 快速定位关键函数
- ③ 三. 逆向牛刀小试



二. 快速定位关键函数

④ 为了快速找到我们希望分析的关键函数

- ❑ 1. 长驱直入法
- ❑ 2. 字符串查找法
- ❑ 3. API引用法
- ❑ 4. API断点法



1.长驱直入法

@长驱直入法的原理

- ❑ 当程序功能非常明确时，从程序入口处一步一步分析，逐条执行指令，直到找到关键函数
- ❑ 不过长驱直入法仅适用于被调试的代码量不大、且程序功能明确的情况
 - 倘若被调试的程序比较复杂时，这种方法就不适合了。



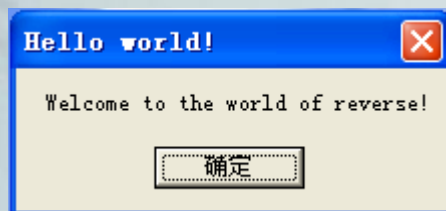
1.长驱直入法

- ❑ 运行程序的时候会弹出消息对话框，因此通过确定**MessageBox**函数的调用位置,即可确定**main**函数的位置
- ❑ 可以通过单步调试**HelloWorld**这个程序（**F8**，**Step Over**），当执行到某条指令时，程序弹出消息对话框，就能锁定**main**函数的位置



2. 字符串查找法

- ❑ 程序运行时弹出的消息对话框上显示了两个字符串，一个是标题“**Hello world!**”，一个是内容“**Welcome to the world of reverse!**”



- ❑ 可以通过查找这两个字符串来确定main函数的位置

2. 字符串查找法

@ 字符串查找法

- ❑ IDA PRO的字符串查找法
- ❑ OllyDbg的字符串查找法



2. 字符串查找法

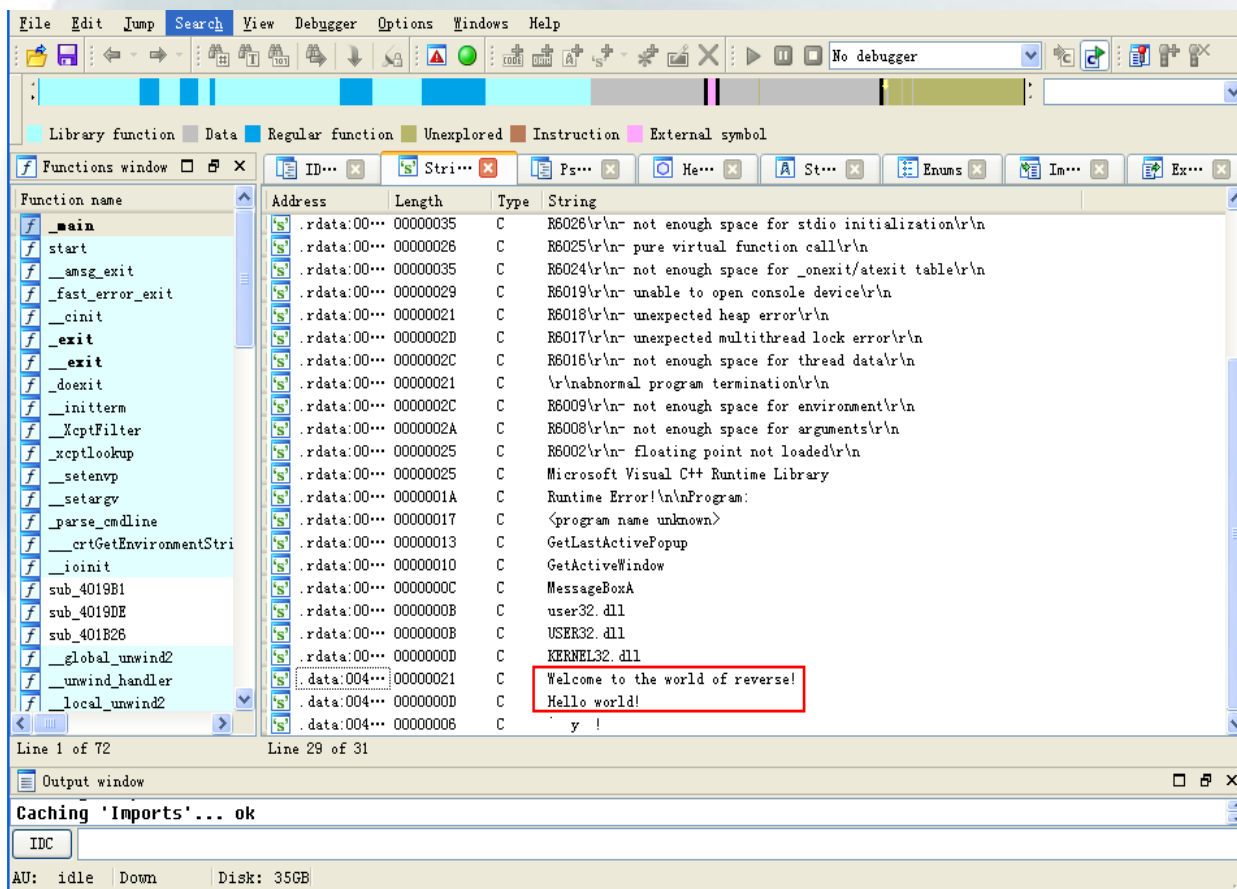
□ IDA PRO的字符串查找法

- 用IDA PRO加载目标程序后，通过三连击迅速确定关键函数位置
- 三连击：“Shift F12 + x + F5”



2. 字符串查找法

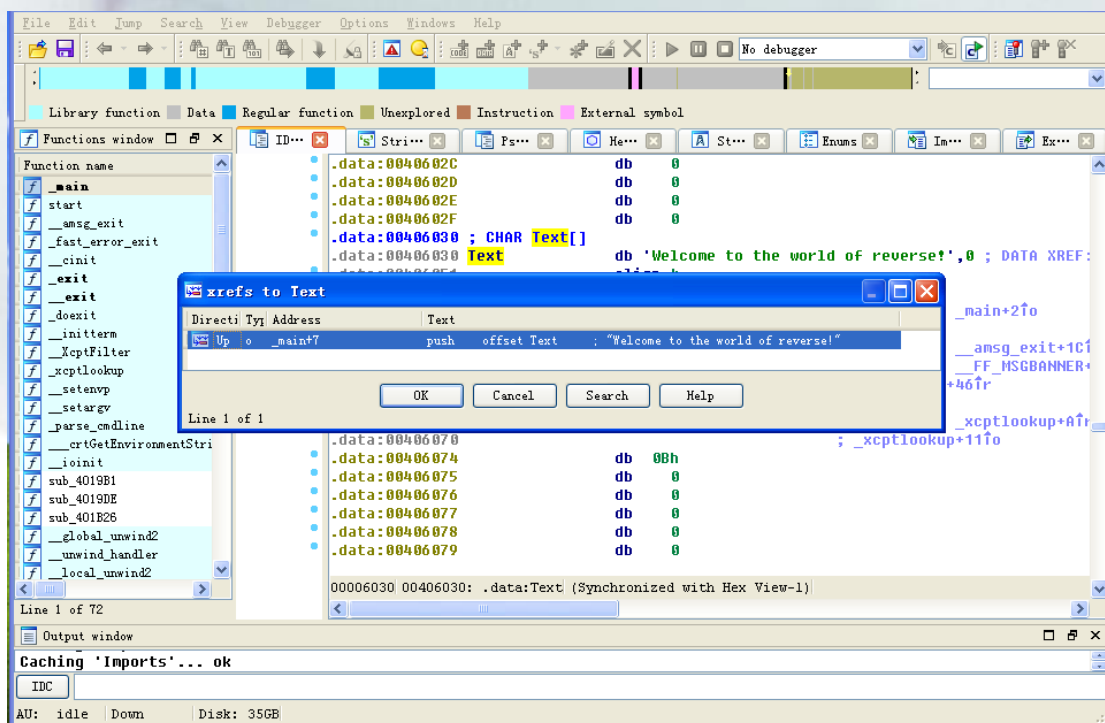
❑ (1) Shift + F12搜索字符串



2. 字符串查找法

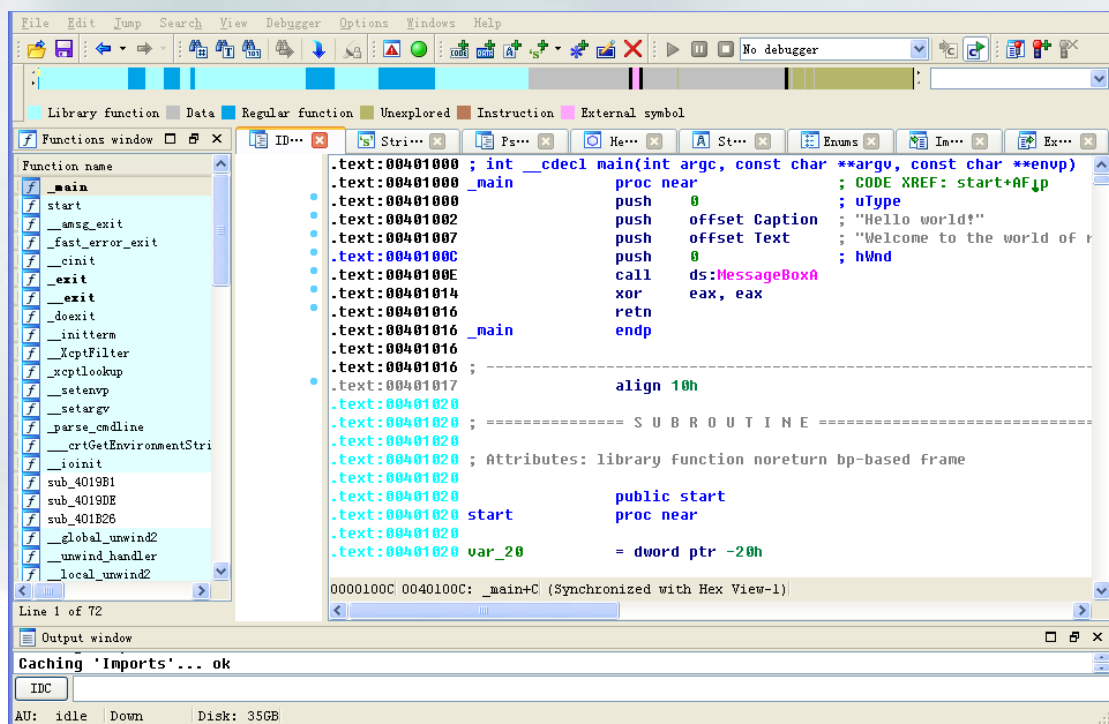
❑ (2) x查看交叉引用

- 选中一个字符串，双击跳转到汇编窗口，然后按快捷键x查看调用该字符串的地方



2. 字符串查找法

○ 点击OK即可来到关键函数的汇编代码段



The screenshot shows a debugger window with the assembly code for the `_main` function. The code is as follows:

```
.text:00401000 ; int __cdecl main(int argc, const char **argv, const char **envp)
.text:00401000 _main      proc near      ; CODE XREF: start+AF1p
.text:00401000          push      0
.text:00401002          push      offset Caption ; "Hello world!"
.text:00401007          push      offset Text   ; "Welcome to the world of r
.text:0040100C          push      0              ; hWnd
.text:0040100E          call     ds:MessageBoxA
.text:00401014          xor      eax, eax
.text:00401016          retn
.text:00401016 _main      endp

.text:00401016 ;
.text:00401017          align 10h

.text:00401020 ; ===== SUBROUTINE =====
.text:00401020 ; Attributes: library function noreturn bp-based frame
.text:00401020 ;
.text:00401020 public start
.text:00401020 start      proc near
.text:00401020          = dword ptr -20h
.text:00401020 var_20

0000100C 0040100C: _main+C (Synchronized with Hex View-1)
```

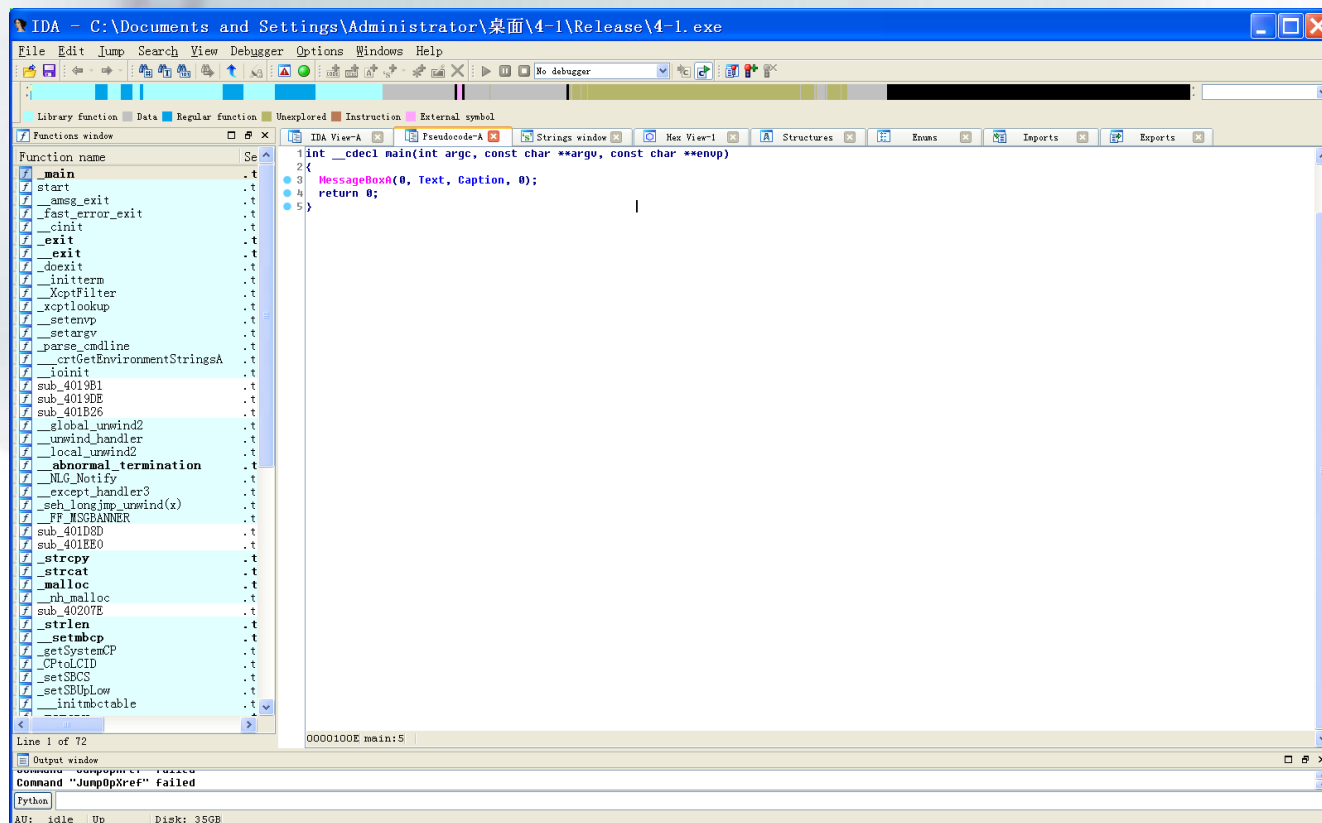
The debugger interface includes a menu bar (File, Edit, Jump, Search, View, Debugger, Options, Windows, Help), a toolbar, and a status bar at the bottom showing "AU: idle | Down | Disk: 35GB". The "Functions window" on the left lists various functions, and the "Output window" at the bottom shows "Caching 'Imports'... ok".



2. 字符串查找法

❑ (3) F5生成伪代码

- 在关键函数处，按快捷键F5生成伪代码，然后分析关键函数的代码逻辑



2. 字符串查找法

@ 字符串查找法

□ IDA PRO的字符串查找法

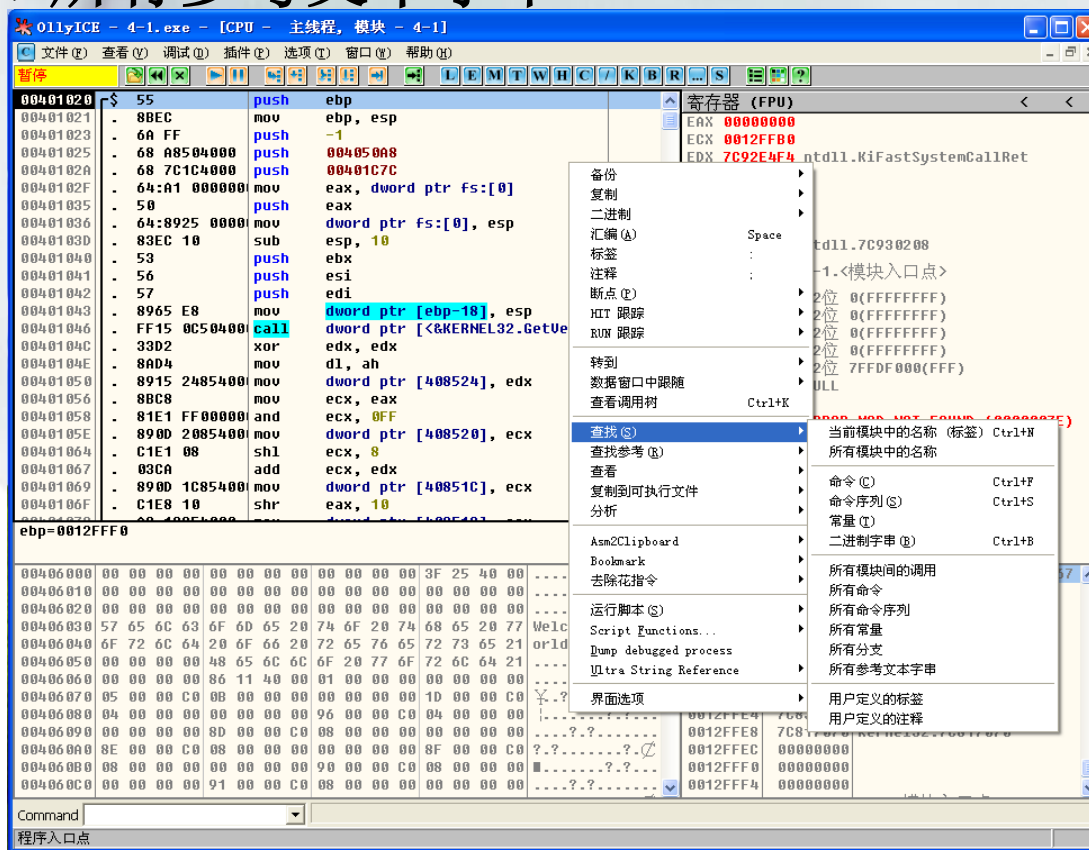
□ OllyDbg的字符串查找法



2. 字符串查找法

❑ OllyDbg的字符串查找法

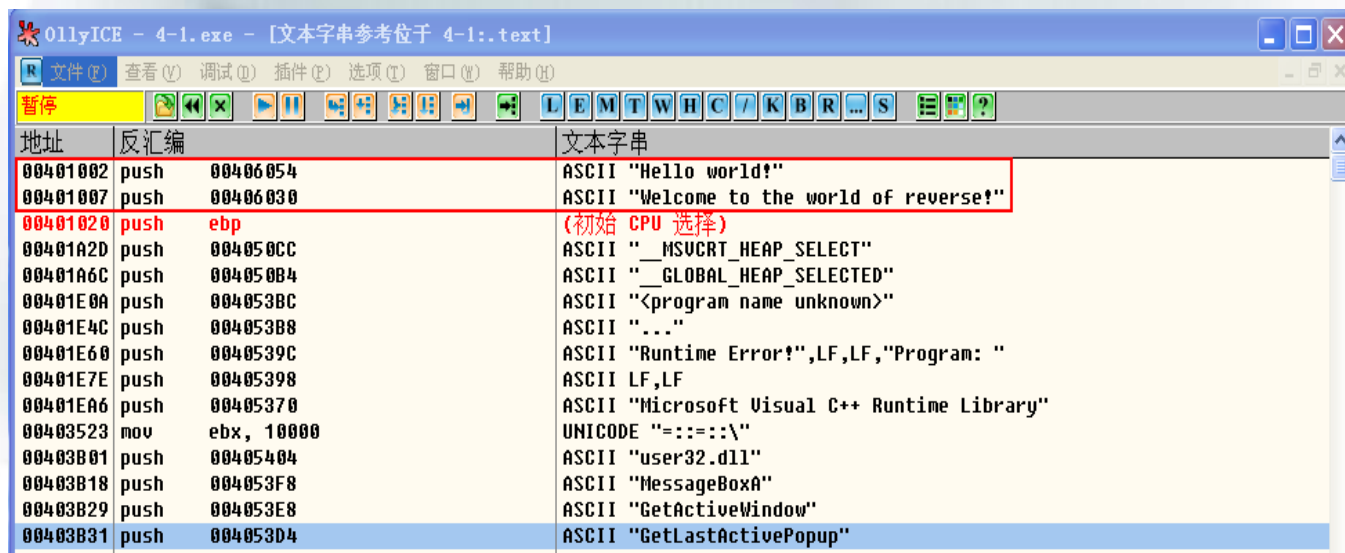
○ OllyDbg加载目标程序后，鼠标右键菜单—>查找—>所有参考文本字符串



2. 字符串查找法

❑ OllyDbg的字符串查找法

○ 字符串查找结果



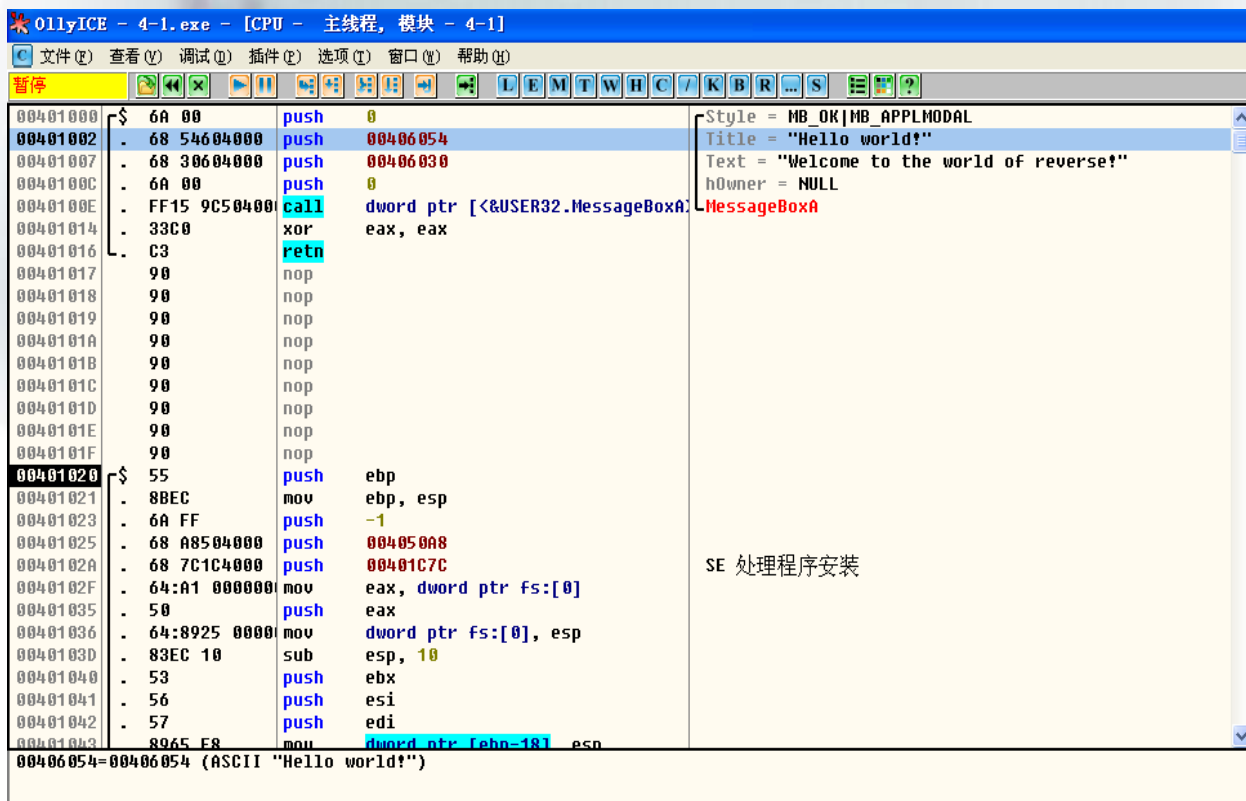
地址	反汇编	文本字符串
00401002	push 00406054	ASCII "Hello world!"
00401007	push 00406030	ASCII "Welcome to the world of reverse!"
00401020	push ebp	(初始 CPU 选择)
0040102D	push 004050CC	ASCII "_MSUCRT_HEAP_SELECT"
0040106C	push 004050B4	ASCII "_GLOBAL_HEAP_SELECTED"
00401E0A	push 004053BC	ASCII "<program name unknown>"
00401E4C	push 004053B8	ASCII "..."
00401E60	push 0040539C	ASCII "Runtime Error!",LF,LF,"Program: "
00401E7E	push 00405398	ASCII LF,LF
00401EA6	push 00405370	ASCII "Microsoft Visual C++ Runtime Library"
00403523	mov ebx, 10000	UNICODE "=::::\\"
00403801	push 00405404	ASCII "user32.dll"
00403818	push 004053F8	ASCII "MessageBoxA"
00403829	push 004053E8	ASCII "GetActiveWindow"
00403831	push 004053D4	ASCII "GetLastActivePopup"



2. 字符串查找法

❑ OllyDbg的字符串查找法

○ 双击“Hello world!”或者“Welcome to the world of reverse!”即可来到main函数位置



The screenshot shows the OllyDbg interface for the file 4-1.exe. The assembly window displays the following code:

```
00401000 6A 00 push 0
00401002 68 54604000 push 00406054
00401007 68 30604000 push 00406030
0040100C 6A 00 push 0
0040100E FF15 9C504000 call dword ptr [&USER32.MessageBoxA]
00401014 33C0 xor eax, eax
00401016 C3 retn
00401017 90 nop
00401018 90 nop
00401019 90 nop
0040101A 90 nop
0040101B 90 nop
0040101C 90 nop
0040101D 90 nop
0040101E 90 nop
0040101F 90 nop
00401020 55 push ebp
00401021 8BEC mov ebp, esp
00401023 6A FF push -1
00401025 68 A8504000 push 004050A8
0040102A 68 7C1C4000 push 00401C7C
0040102F 64:A1 000000 mov eax, dword ptr fs:[0]
00401035 50 push eax
00401036 64:8925 0000 mov dword ptr fs:[0], esp
0040103D 83EC 10 sub esp, 10
00401040 53 push ebx
00401041 56 push esi
00401042 57 push edi
00401043 8965 F8 mov dword ptr [ebp-18], esp
```

The right-hand pane shows the search results for the string "Hello world!". The results are:

- Style = MB_OK|MB_APPLMODAL
- Title = "Hello world!"
- Text = "Welcome to the world of reverse!"
- hOwner = NULL
- MessageBoxA

At the bottom, the search results for the string "Hello world!" are displayed as: 00406054=00406054 (ASCII "Hello world!")



二. 快速定位关键函数

- 1. 长驱直入法
- 2. 字符串查找法
- 3. **API**引用法
- 4. **API**断点法



3. API引用法

- ❑ **Windows**编程中，有些功能需要通过调用**Win32 API**来实现，认真观察一个程序的功能后，能够大致推测出它在运行时调用了哪些**Win32 API**
- ❑ 以**HelloWorld**程序为例，它在运行时会弹出一个消息窗口，因此推断出该程序调用了**MessageBox**函数，通过查找哪里调用了该**API**即可确定关键函数的位置



3. API引用法

- ❑ (1) IDA PRO
- ❑ (2) OllyDbg



3. API引用法

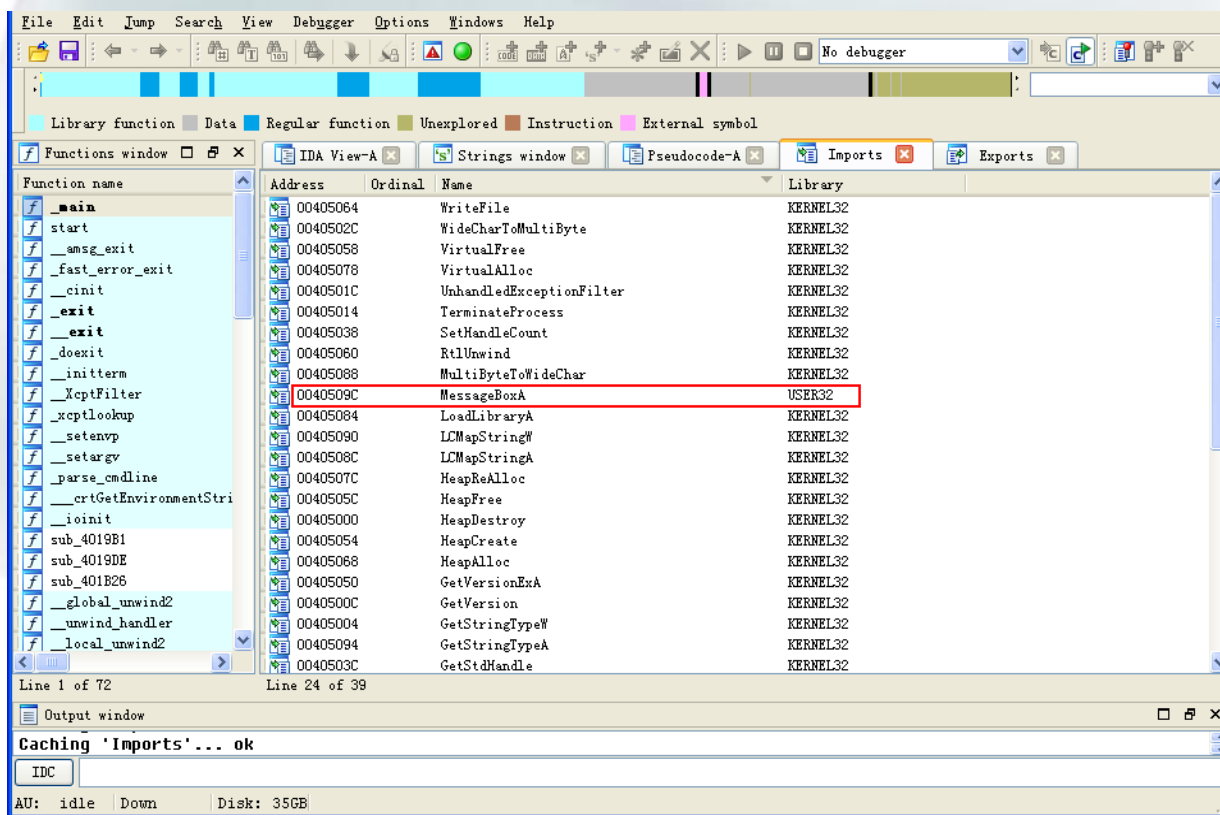
④ (1) IDA PRO

□ IDA中通过导入函数窗口（Imports）可以迅速找到MessageBoxA函数（因为HelloWorld程序编译时用的是ASCII字符集，所以调用的是MessageBoxA函数，若是Unicode字符串，调用的就是MessageBoxW函数）



3. API引用法

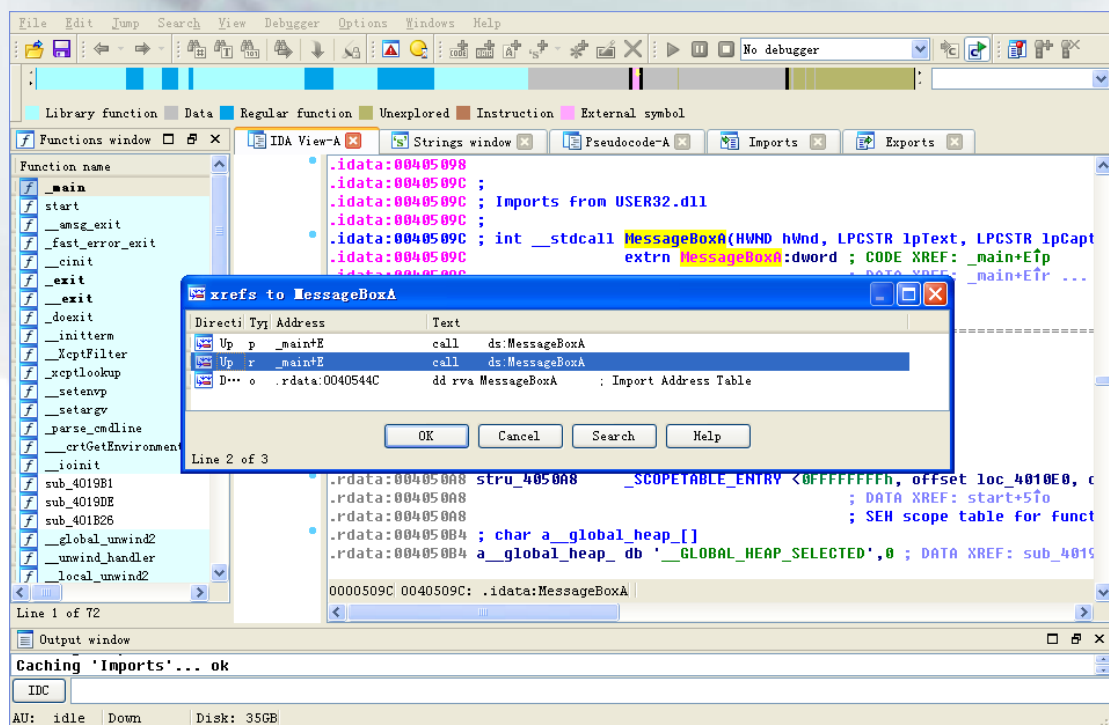
□ (1) IDA PRO



3. API引用法

① (1) IDA PRO

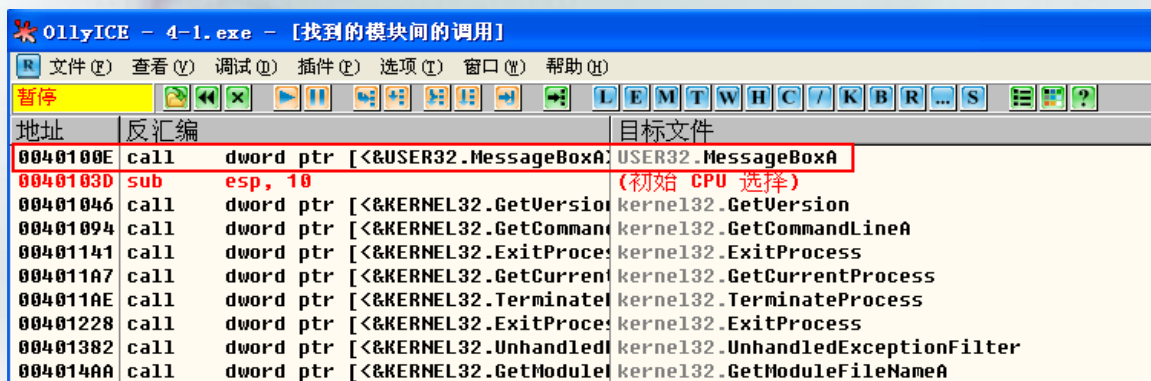
❑ IDA PRO双击MessageBoxA函数来到汇编代码窗口，通过查看交叉引用即可确定关键函数位置



3. API引用法

❑ (2) OllyDbg

❑ OllyDbg加载目标程序后，鼠标右键菜单—>查找—>所有模块间的调用



❑ 双击MessageBoxA即可来到调用它的地址处（0x0040100E），即可确定关键函数位置



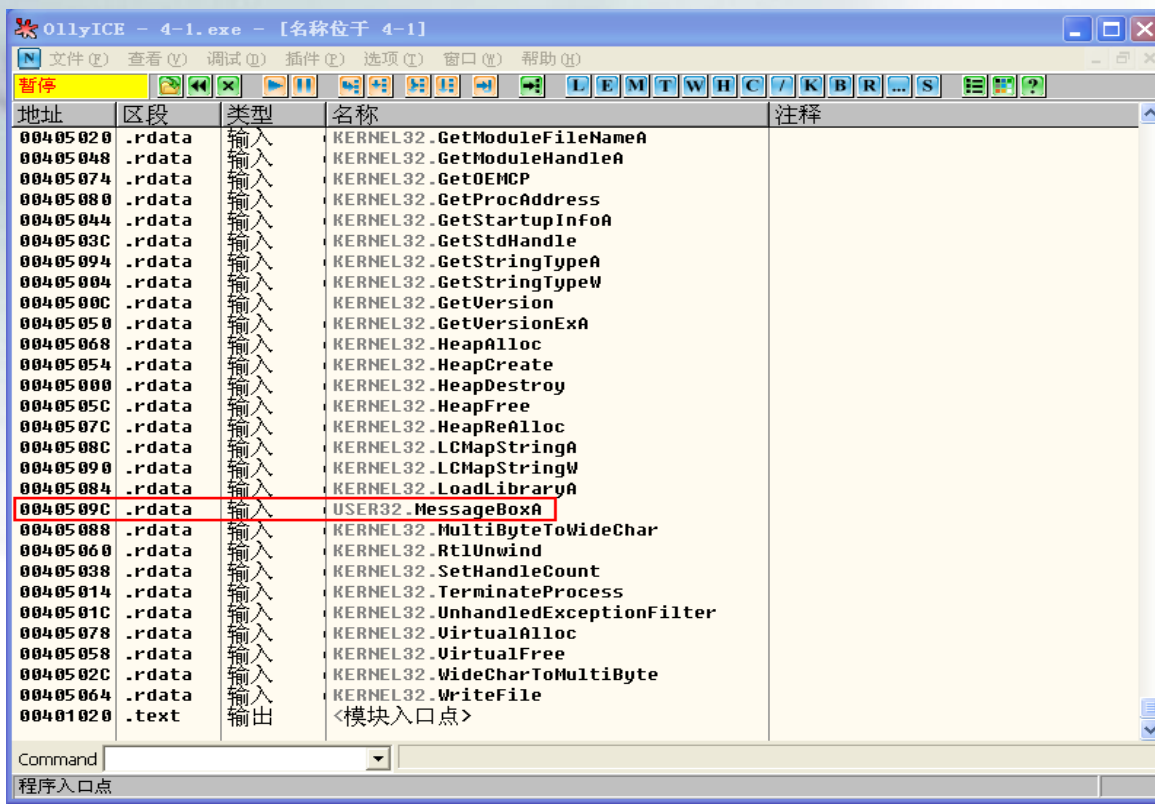
二. 快速定位关键函数

- 1. 长驱直入法
- 2. 字符串查找法
- 3. API引用法
- 4. API断点法



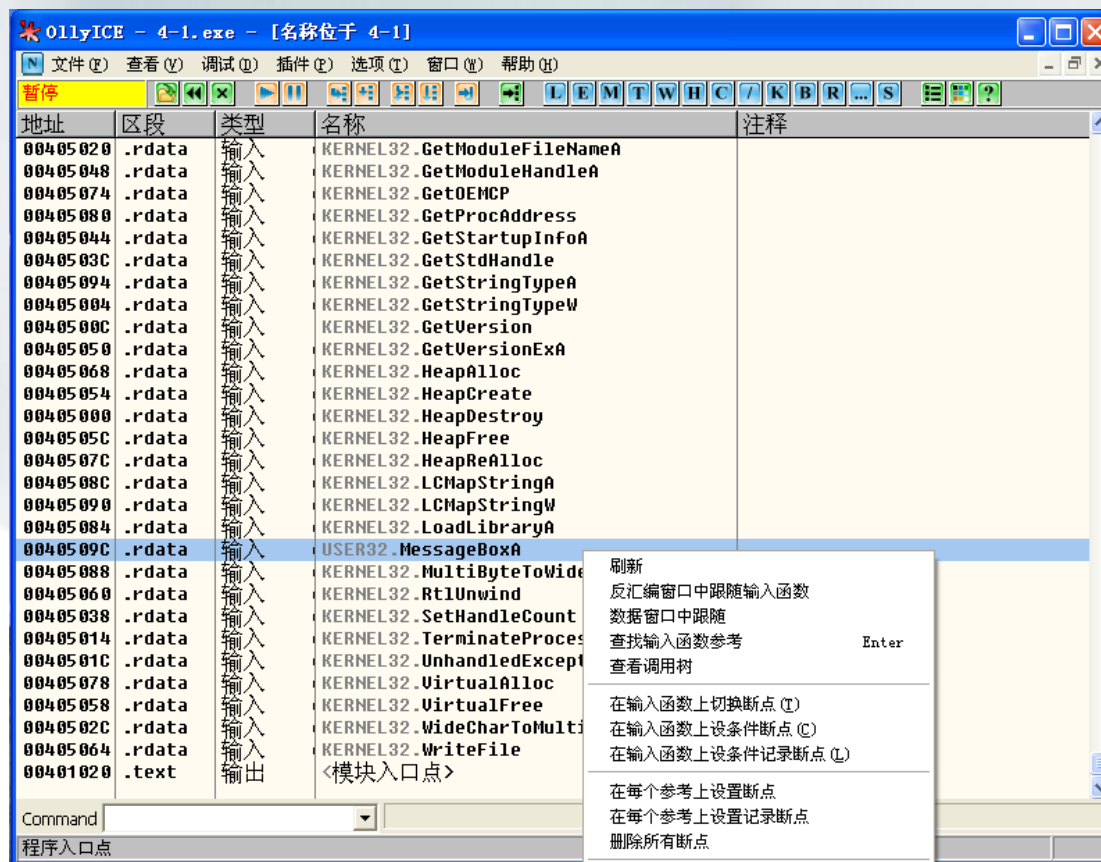
4. API断点法

- ❑ 通过在API上设置断点来确定关键函数位置
- ❑ 用OllyDbg加载目标程序，鼠标右键菜单—>查找—>当前模块中的名称，或者用“ctrl + n”快捷键



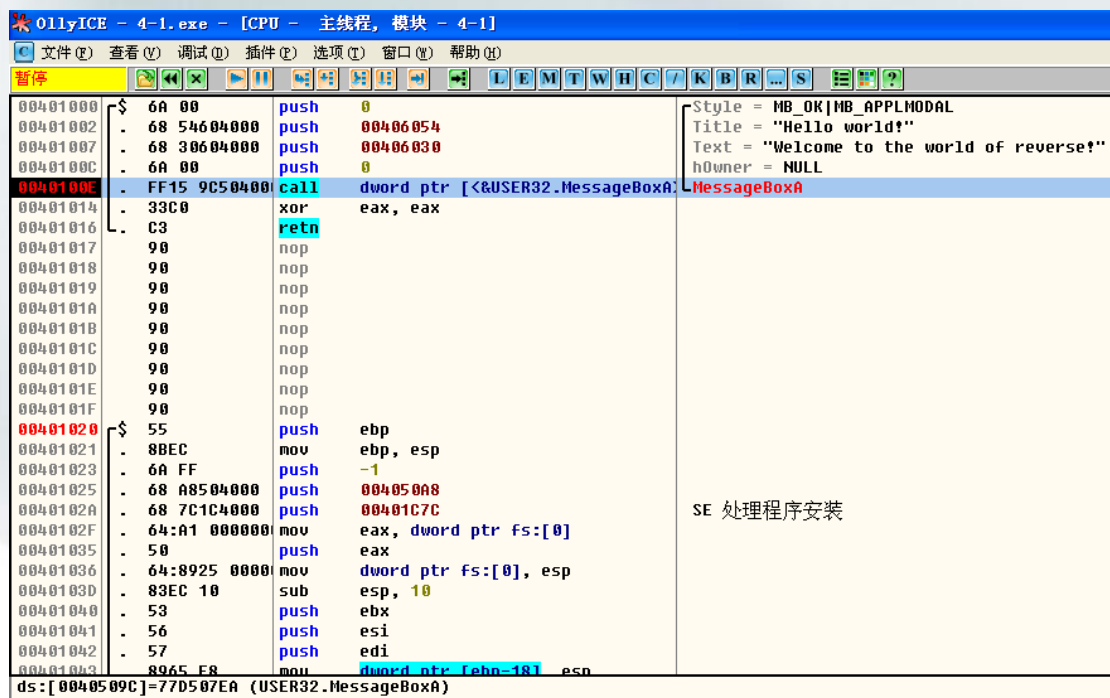
4. API断点法

❑ 鼠标右键MessageBoxA函数，选择“在每个参考上设置断点”



4. API断点法

Ⓢ F9运行程序，便会在关键函数处停下来



The screenshot shows the OllyICE debugger interface. The title bar indicates the target is '4-1.exe' at the CPU level, main thread, module '4-1'. The menu bar includes File, View, Debug, Plugins, Options, Windows, and Help. The toolbar shows various debugging actions, with the 'Breakpoint' icon (a red circle with a white 'X') highlighted. The main window displays assembly code with the following instructions:

```
00401000 6A 00 push 0
00401002 68 54604000 push 00406054
00401007 68 30604000 push 00406030
0040100C 6A 00 push 0
0040100E FF15 9C504000 call dword ptr [ <USER32.MessageBoxA@77D507EA> ]
00401014 33C0 xor eax, eax
00401016 C3 ret
00401017 90 nop
00401018 90 nop
00401019 90 nop
0040101A 90 nop
0040101B 90 nop
0040101C 90 nop
0040101D 90 nop
0040101E 90 nop
0040101F 90 nop
00401020 55 push ebp
00401021 8BEC mov ebp, esp
00401023 6A FF push -1
00401025 68 A8504000 push 004050A8
0040102A 68 7C1C4000 push 00401C7C
0040102F 64:A1 00000000 mov eax, dword ptr fs:[0]
00401035 50 push eax
00401036 64:8925 00000000 mov dword ptr fs:[0], esp
0040103D 83EC 10 sub esp, 10
00401040 53 push ebx
00401041 56 push esi
00401042 57 push edi
00401043 8965 F8 mov dword ptr [ebp-18], esp
```

The instruction at address 0040100E is highlighted in blue, indicating it is the current instruction being executed. The right-hand pane shows the parameters for the selected instruction: 'Style = MB_OK|MB_APPLMODAL', 'Title = "Hello world!"', 'Text = "Welcome to the world of reverse!"', and 'hOwner = NULL'. The status bar at the bottom shows the current instruction pointer (EIP) as 'ds:[0040509C]=77D507EA (USER32.MessageBoxA)'.



第四章 逆向初体验

- ①. HelloWorld程序逆向分析
- ②. 快速定位关键函数
- ③. 逆向牛刀小试



三. 逆向牛刀小试

④ 以**firstRe.exe**作为本章的练习题目，更好的练习如何定位关键函数代码段



三. 逆向牛刀小试

- ① 1. 定位关键函数
- ② 2. 程序逻辑分析



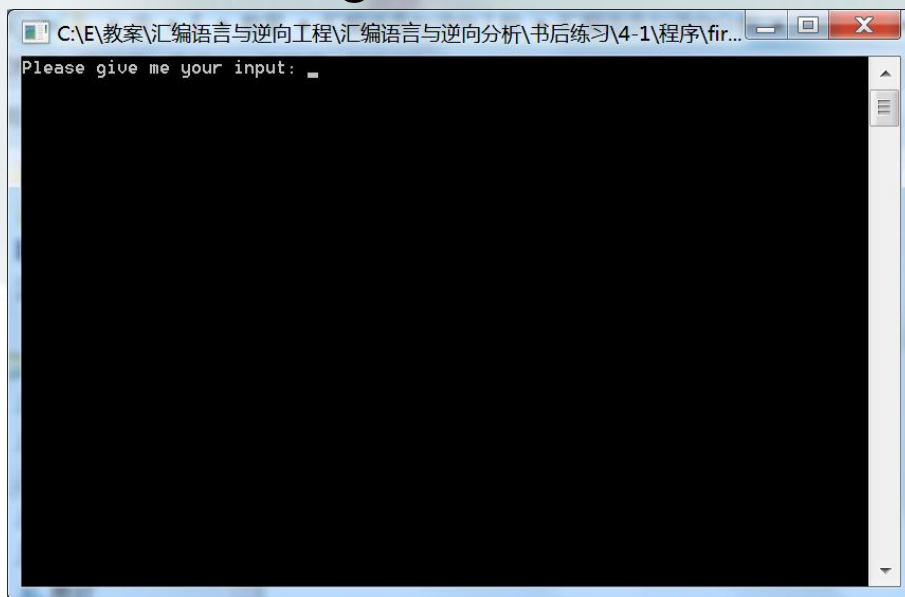
三. 逆向牛刀小试

@ 运行程序，会让我们输入一串字符串，然后会提示我们输入错误

D:\>firstRe.exe

Please give me your input: aaaaaaaa

Sorry! You are wrong!!!

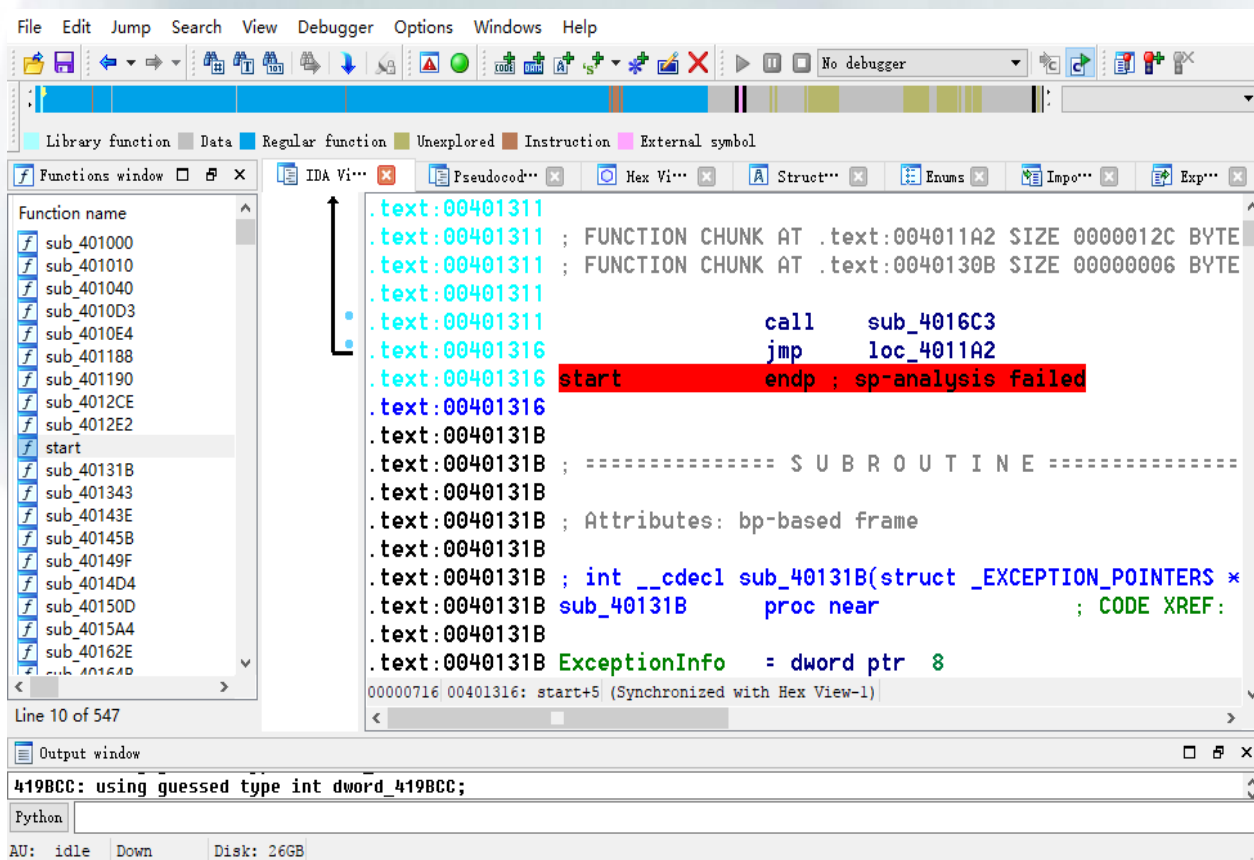


崔宝江



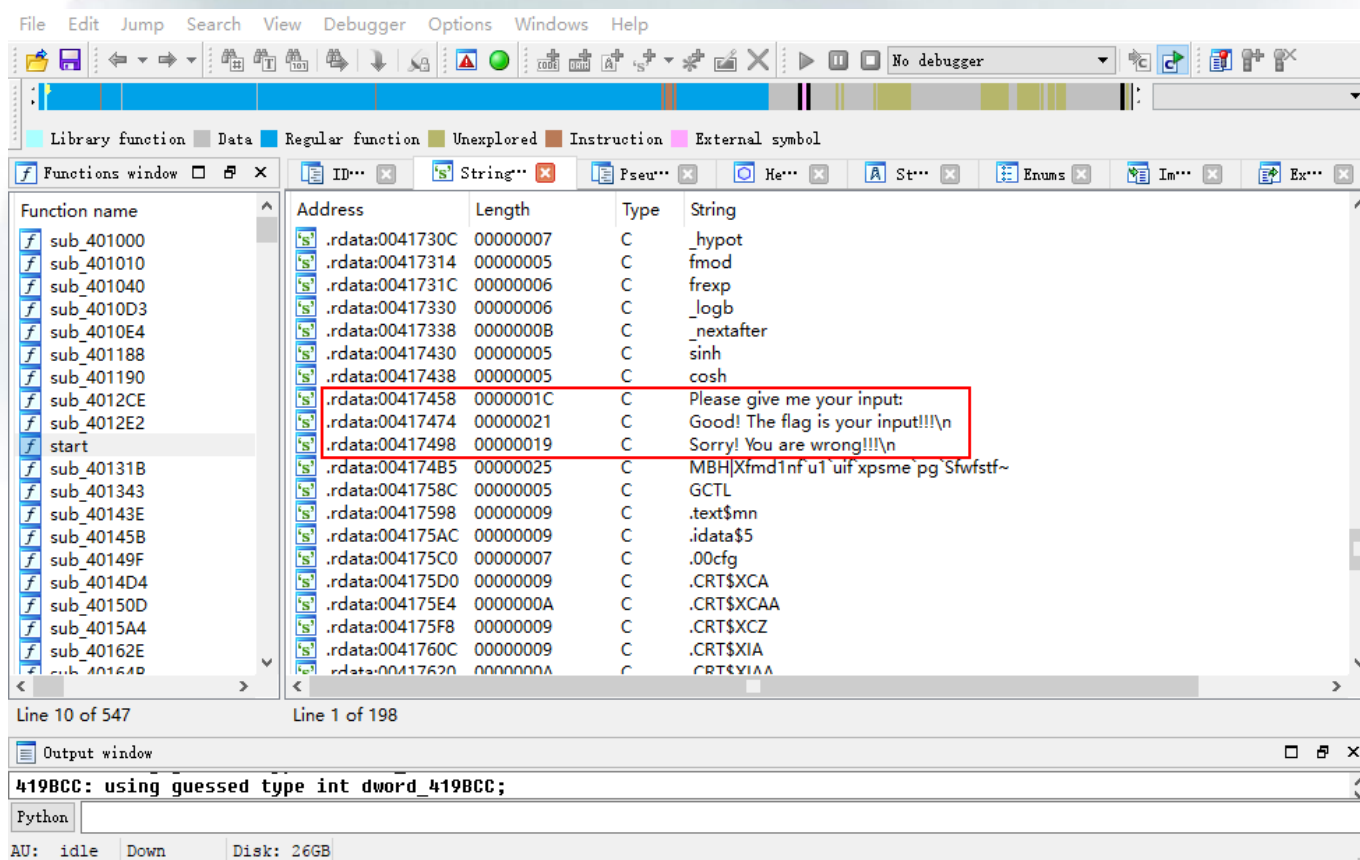
三. 逆向牛刀小试

① 用IDA加载目标程序，在函数名称一栏并没能找到main函数



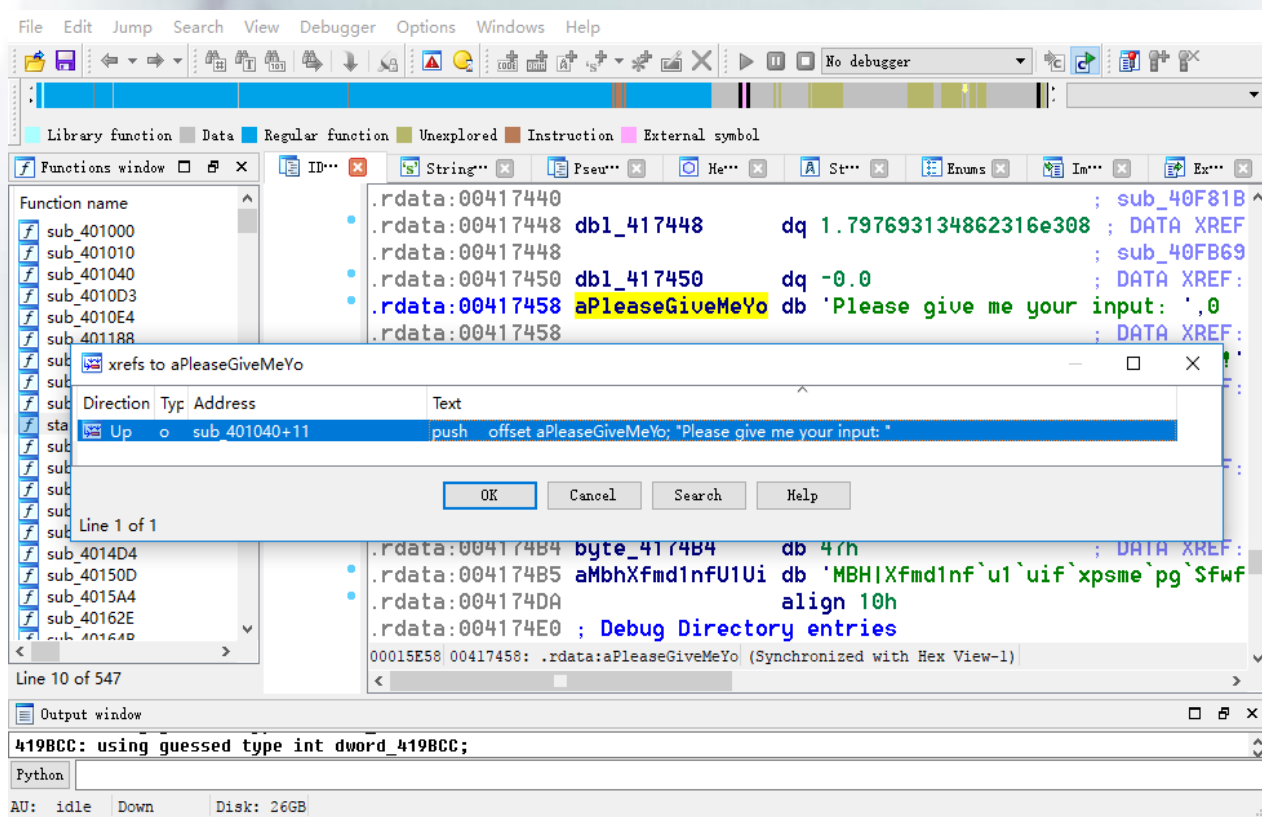
三. 逆向牛刀小试

通过字符串查找法来确定关键函数代码段，按快捷键“**Shift + F12**”搜索字符串



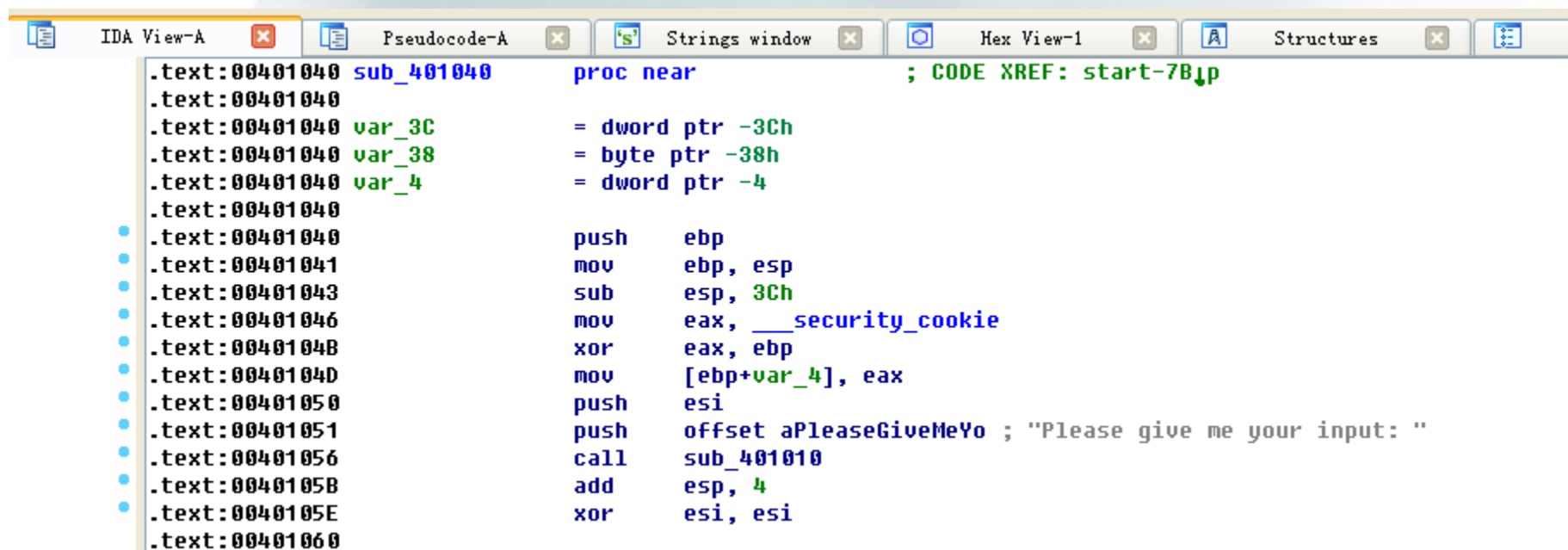
三. 逆向牛刀小试

- 可以看到跟程序功能有关的字符串，选中“**Please give me your input:**”并双击，来到该字符串的汇编代码窗口，并按快捷键“x”查看其交叉引用



三. 逆向牛刀小试

□ 来到函数sub_401040的汇编代码



```
.text:00401040 sub_401040 proc near ; CODE XREF: start-7B↓p
.text:00401040
.text:00401040 var_3C = dword ptr -3Ch
.text:00401040 var_38 = byte ptr -38h
.text:00401040 var_4 = dword ptr -4
.text:00401040
.text:00401040 push ebp
.text:00401041 mov ebp, esp
.text:00401043 sub esp, 3Ch
.text:00401046 mov eax, __security_cookie
.text:00401048 xor eax, ebp
.text:0040104D mov [ebp+var_4], eax
.text:00401050 push esi
.text:00401051 push offset aPleaseGiveMeYo ; "Please give me your input: "
.text:00401056 call sub_401010
.text:0040105B add esp, 4
.text:0040105E xor esi, esi
.text:00401060
```



三. 逆向牛刀小试

□ 按快捷键“F5”查看其伪代码，如图4-29所示，成功定位到关键函数部分

```
1 int sub_401040()  
2 {  
3     signed int v0; // esi@1  
4     char v1; // al@2  
5     int v2; // eax@5  
6     int v4; // [sp+4h] [bp-3Ch]@0  
7     char v5[52]; // [sp+8h] [bp-38h]@3  
8  
9     sub_401010("Please give me your input: ");  
10    v0 = 0;  
11    do  
12    {  
13        v1 = sub_403B40();  
14        if ( v1 == 10 )  
15            break;  
16        v5[v0++] = v1;  
17    }  
18    while ( v0 < 40 );  
19    v5[v0] = 0;  
20    if ( strlen(v5) == 37 )  
21    {  
22        v2 = 0;  
23        while ( v5[v2] + 1 == byte_4174B4[v2] )  
24        {  
25            if ( ++v2 >= 37 )  
26            {  
27                if ( v2 == 37 || v4 )  
28                {  
29                    sub_401010("Good! The flag is your input!!!\n");  
30                    return 0;  
31                }  
32                break;  
33            }  
34        }  
35    }  
36    sub_401010("Sorry! You are wrong!!!\n");  
37    return 0;  
38 }
```

崔宝江



三. 逆向牛刀小试

@1. 定位关键函数

@2. 程序逻辑分析



2. 程序逻辑分析

- 结合OllyDbg动态调试和代码，可以确定函数sub_401010为输出函数（printf），函数sub_403B4D为读取字符函数（getchar）。
- 程序一开始会获取用户的输入，当输入超过40个字符或者遇到换行（v1=10）就停止读取。
 - ‘\n’的ASCII码值为10：用鼠标选中10这个数字，然后按快捷键‘r’转换为字符\n

```
10  v0 = 0;  
11  do  
12  {  
13      v1 = sub_403B4D();  
14      if ( v1 == 10 )  
15          break;  
16      v5[v0++] = v1;  
17  }  
18  while ( v0 < 40 );
```

```
11  do  
12  {  
13      v1 = sub_403B4D();  
14      if ( v1 == '\n' )  
15          break;  
16      v5[v0++] = v1;  
17  }  
18  while ( v0 < 40 );
```



2. 程序逻辑分析

❑ 如果需要程序输出 “**Good! The flag is your input!!!**”，就需要我们的输入满足两个条件：

- 1、输入字符串长度为37。
- 2、将输入的每个字符**ASCII**码值加1后，与内存中的字符串**byte_4174B4**相等，双击**byte_4174B4**。

```
19 v5[v0] = 0;  
20 if ( strlen(v5) == 37 )  
21 {  
22     v2 = 0;  
23     while ( v5[v2] + 1 == byte_4174B4[v2] )  
24     {  
25         if ( ++v2 >= 37 )  
26         {  
27             if ( v2 == 37 || v4 )  
28             {  
29                 sub_401010("Good! The flag is your input!!!\n");  
30                 return 0;  
31             }  
32             break;  
33         }  
34     }  
35 }
```



2. 程序逻辑分析

- ❑ 通过关键代码分析，很容易找到比较的字符串地址为**0x4174B4**，其内容为

```
.rdata:004174B4 ; char byte_4174B4[]  
.rdata:004174B4 byte_4174B4 db 47h ; DATA XREF: sub_401040+55↑r  
.rdata:004174B5 aMbHxfmd1nfU1Ui db 'MBH|Xfmd1nf`u1`uif`xpsme`pg`Sfwfstf~',0
```

- ❑ **0x4174B4**为**47h**；即**G**

- ❑ **0x4174B5**为

“**MBH|Xfmd1nf`u1`uif`xpsme`pg`Sfwfstf~**”

- ❑ 故应该为:

“**GMBH|Xfmd1nf`u1`uif`xpsme`pg`Sfwfstf~**”



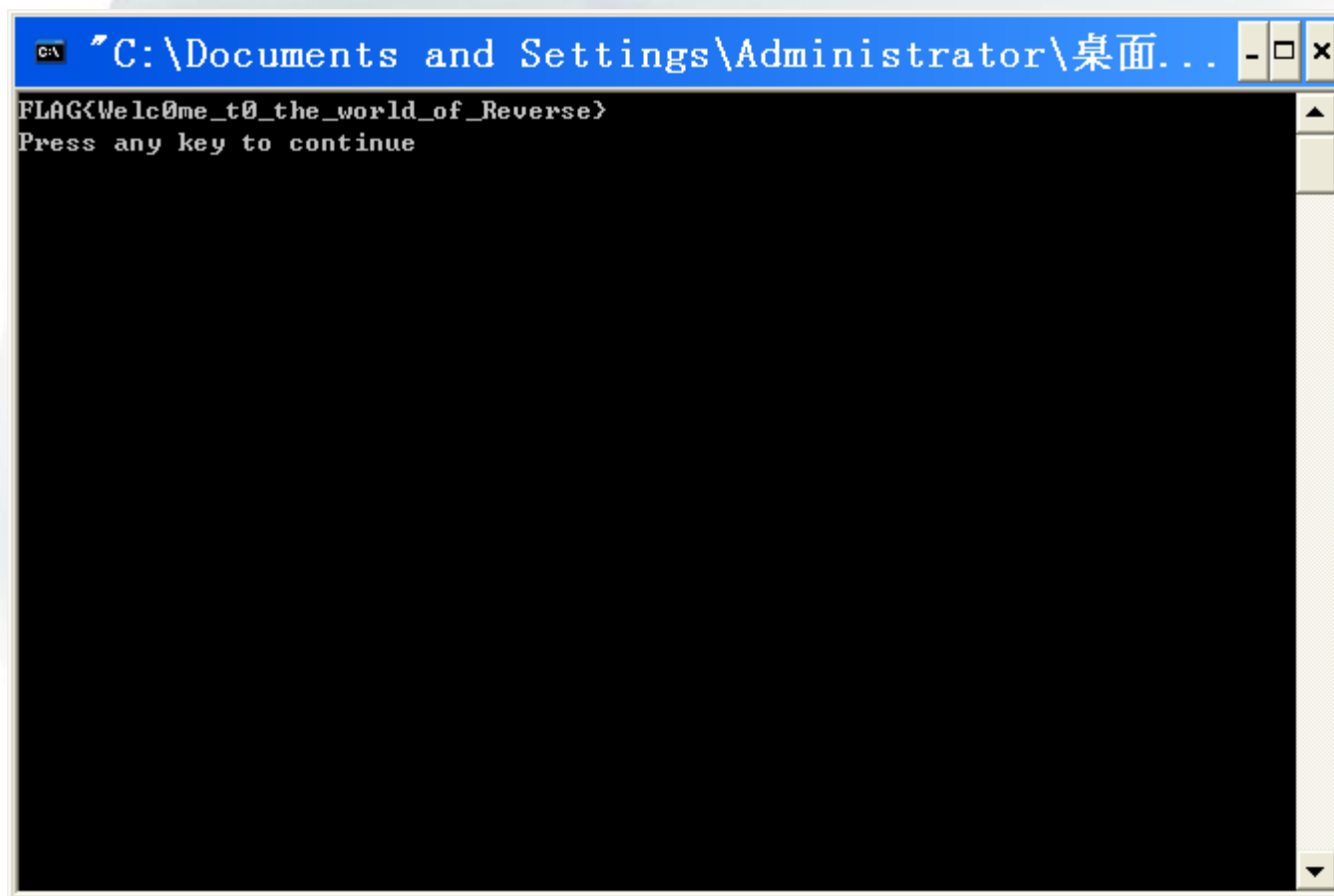
2. 程序逻辑分析

❑ 将该字符串的每个字符**ASCII**码值减1即可得到正确的**flag**。

```
#include<stdio.h>
#include<string.h>
Int main() {
    char check[]="GMBH|Xfmdlnf`u1`uif`xpsme`pg`Sfwfstf~";
    char flag[40]={0};
    for(int i=0;i<strlen(check);i++){
        flag[i]= check[i]-1;
    }
    printf("%s\n", flag);
    return 0;
}
```



2. 程序逻辑分析



```
C:\Documents and Settings\Administrator\桌面...  
FLAG>Welcome_to_the_world_of_Reverse>  
Press any key to continue
```





Q & A

谢谢!

