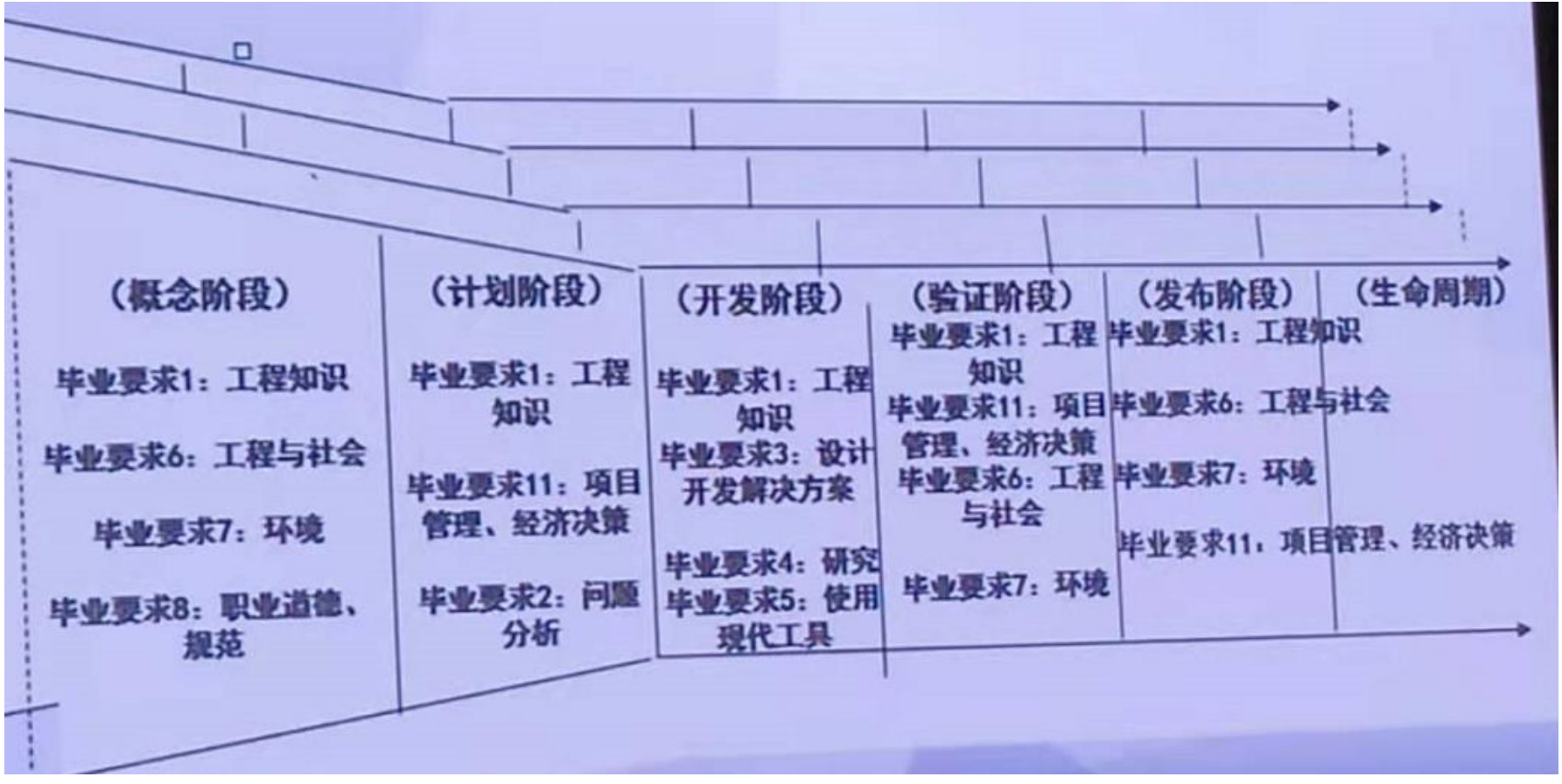
工程认证课程目标

- 课程目标3.1：使学生具有一定的自学能力和信息获取能力
- 课程目标3.2：使学生具有进行程序逆向分析，阅读复杂程序的能力
- 课程目标3.3：使学生具有对实际逆向分析问题研究分析、设计解决方案的能力

课程目标与OBE(outcome based education)毕业要求

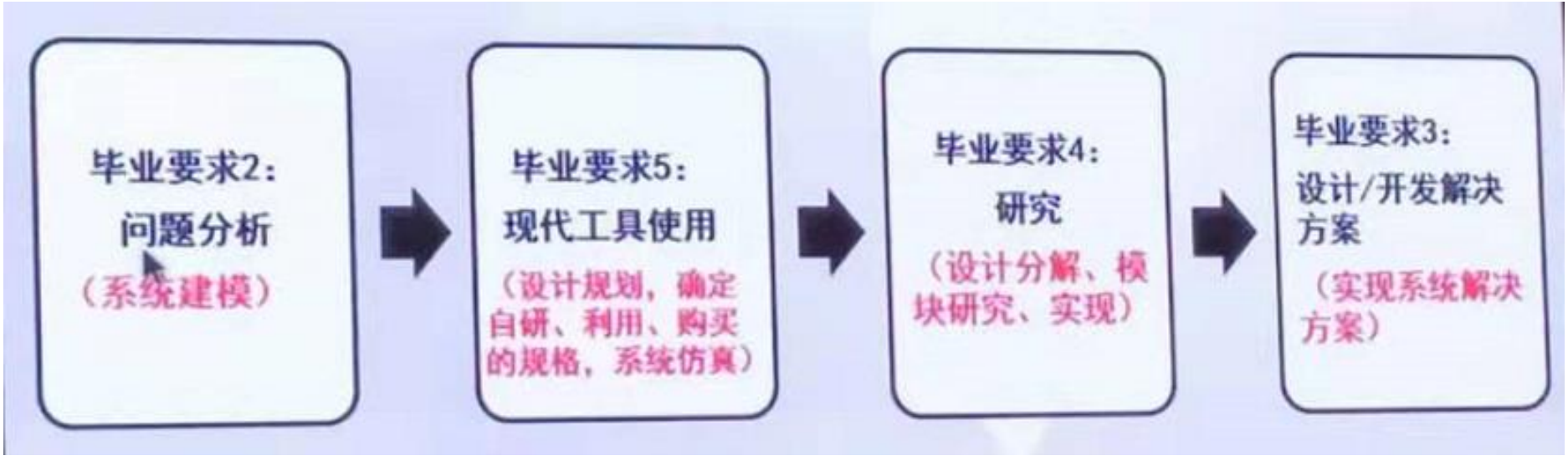
毕业要求	指标点	课程目标
4、研究	4-3-M 针对设计或开发的解决方案，能够基于信息安全领域科学原理对其进行研究，并能够通过理论证明、实验仿真或者系统实现等多种科学方案说明其有效性、合理性，并对解决方案的实施质量进行分析，通过信息综合得到合理有效的结论	3.1
5、使用现代工具	5-3-M 能够分析比较所使用的技术、资源和工具的优势和不足，并理解与表述问题解决方案的局限性。	3.2
3、设计/开发解决方案	3-3-M 充分理解信息安全领域软硬件系统的基础上，能够设计或开发满足特定需求和约束条件的信息安全系统、模块或算法流程，并能够进行系统级优化。	3.3

毕业要求与产品生命周期



毕业要求与产品生命周期

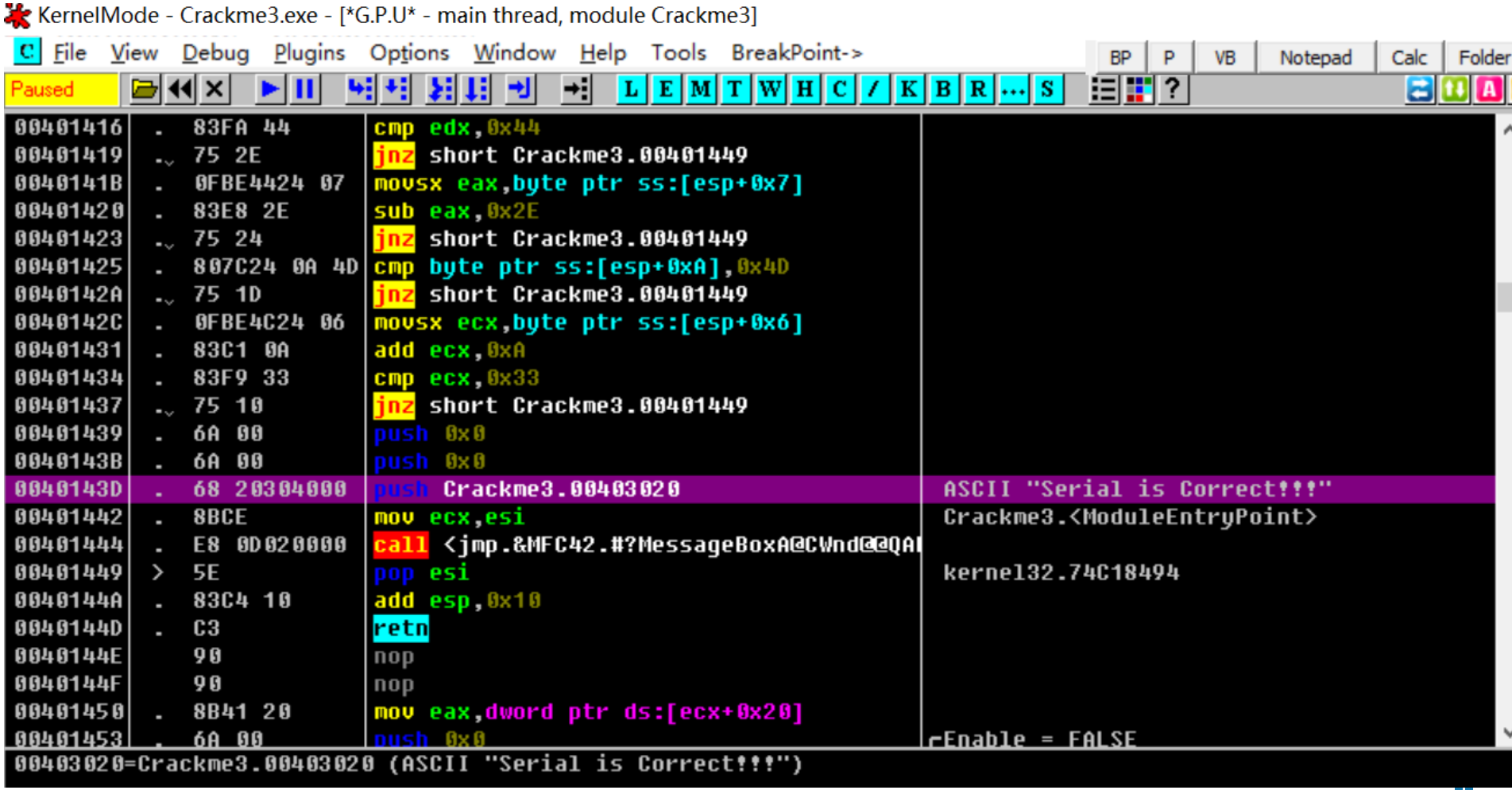
- 毕业要求2、3、4、5的能力协同



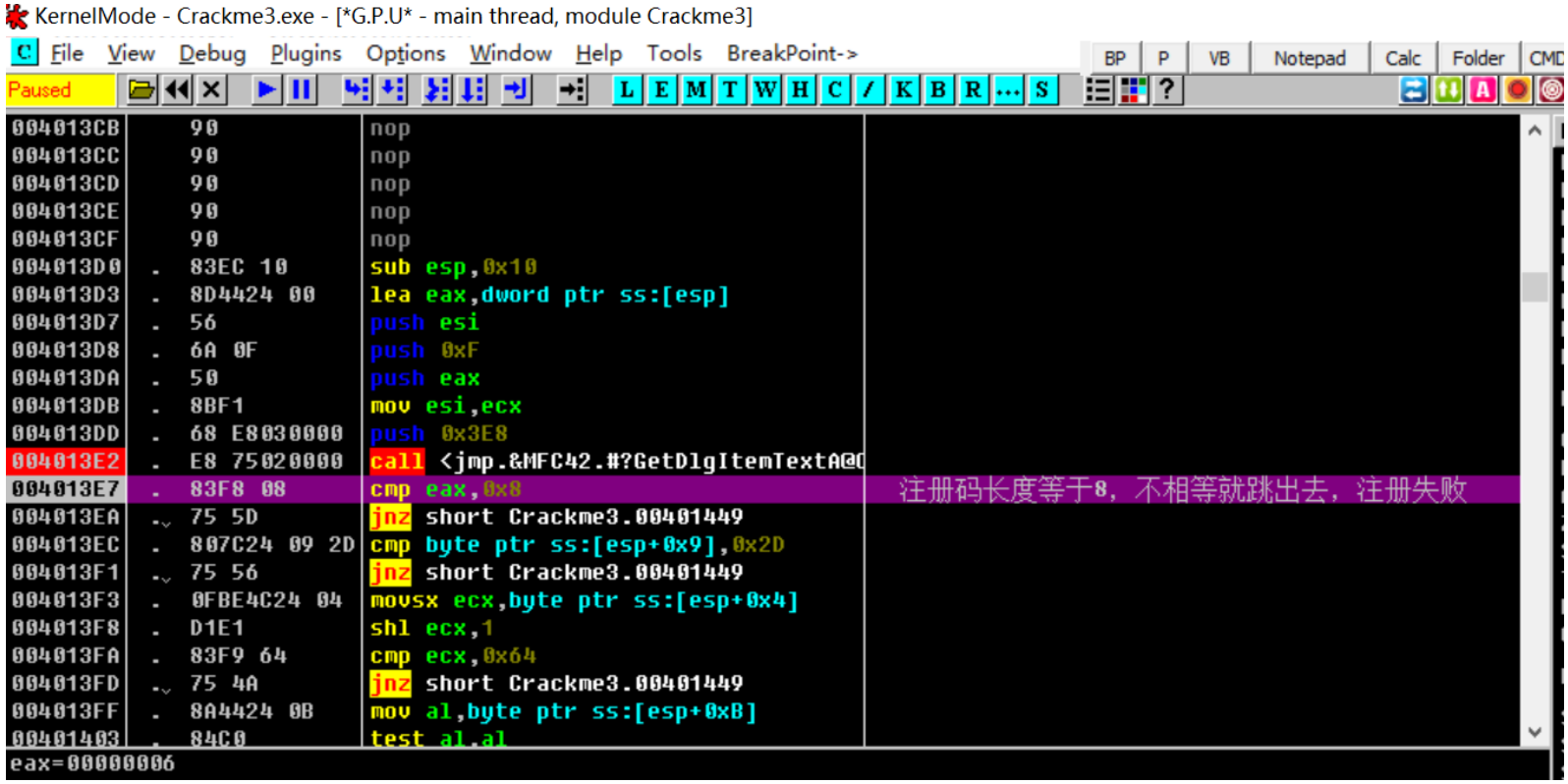
逆向分析技术

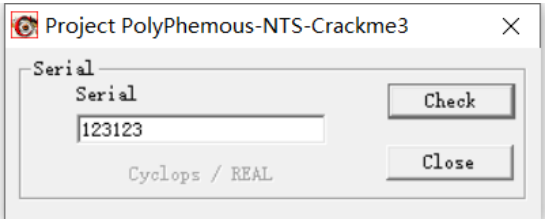
课程回顾

面向汇编的逐句解析 - 追踪字串

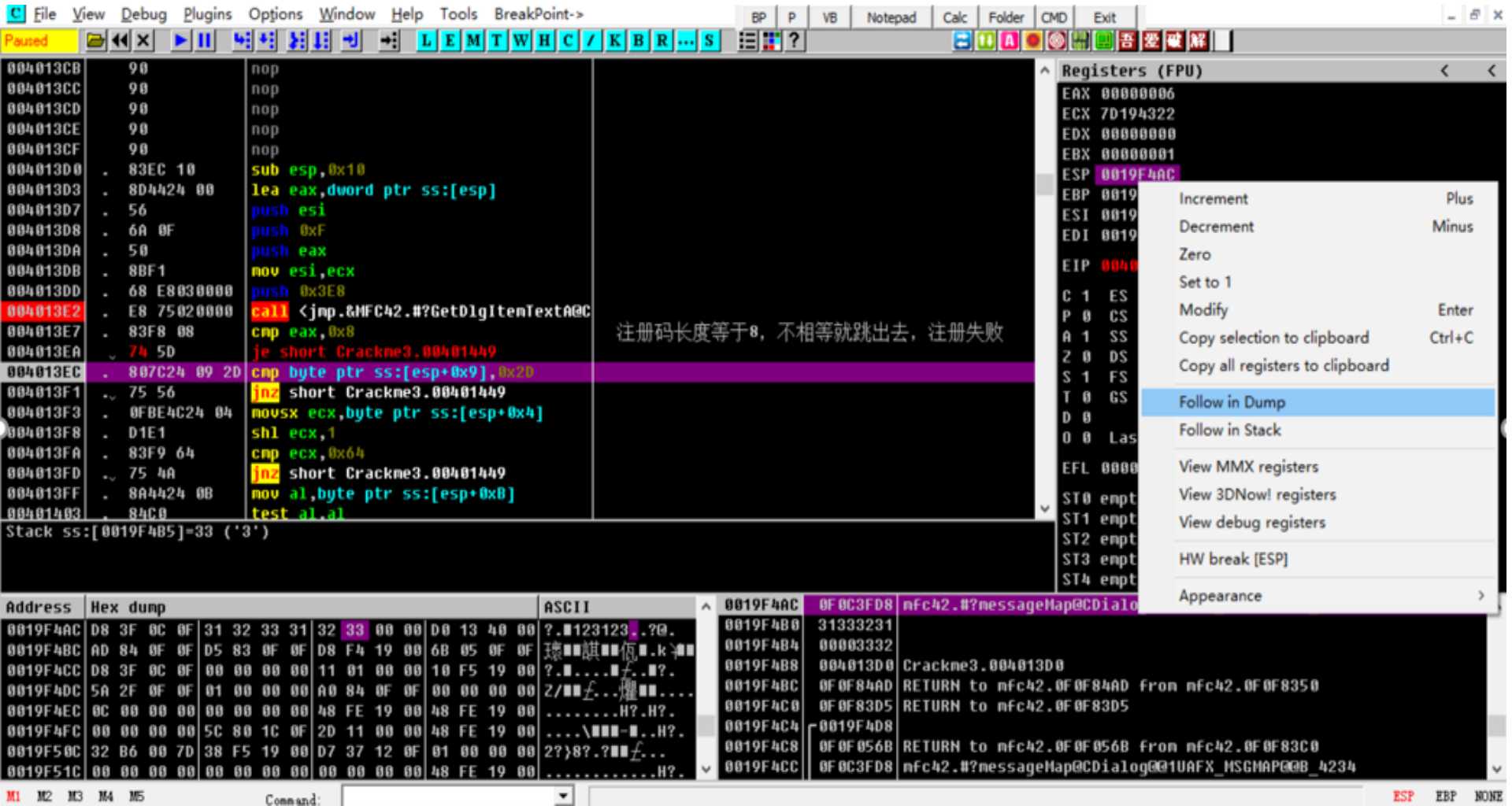


面向汇编的逐句解析 - 长度判断

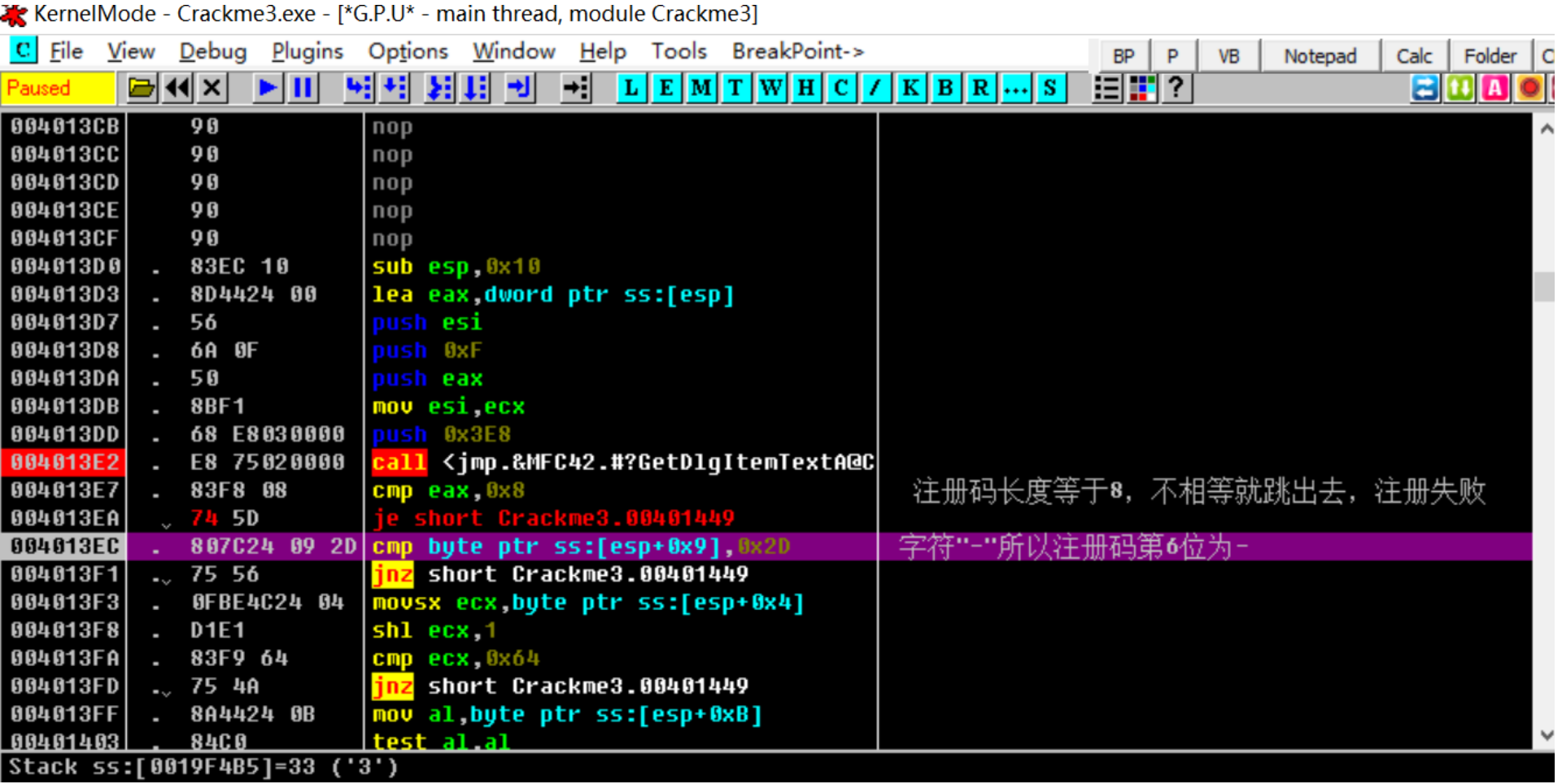




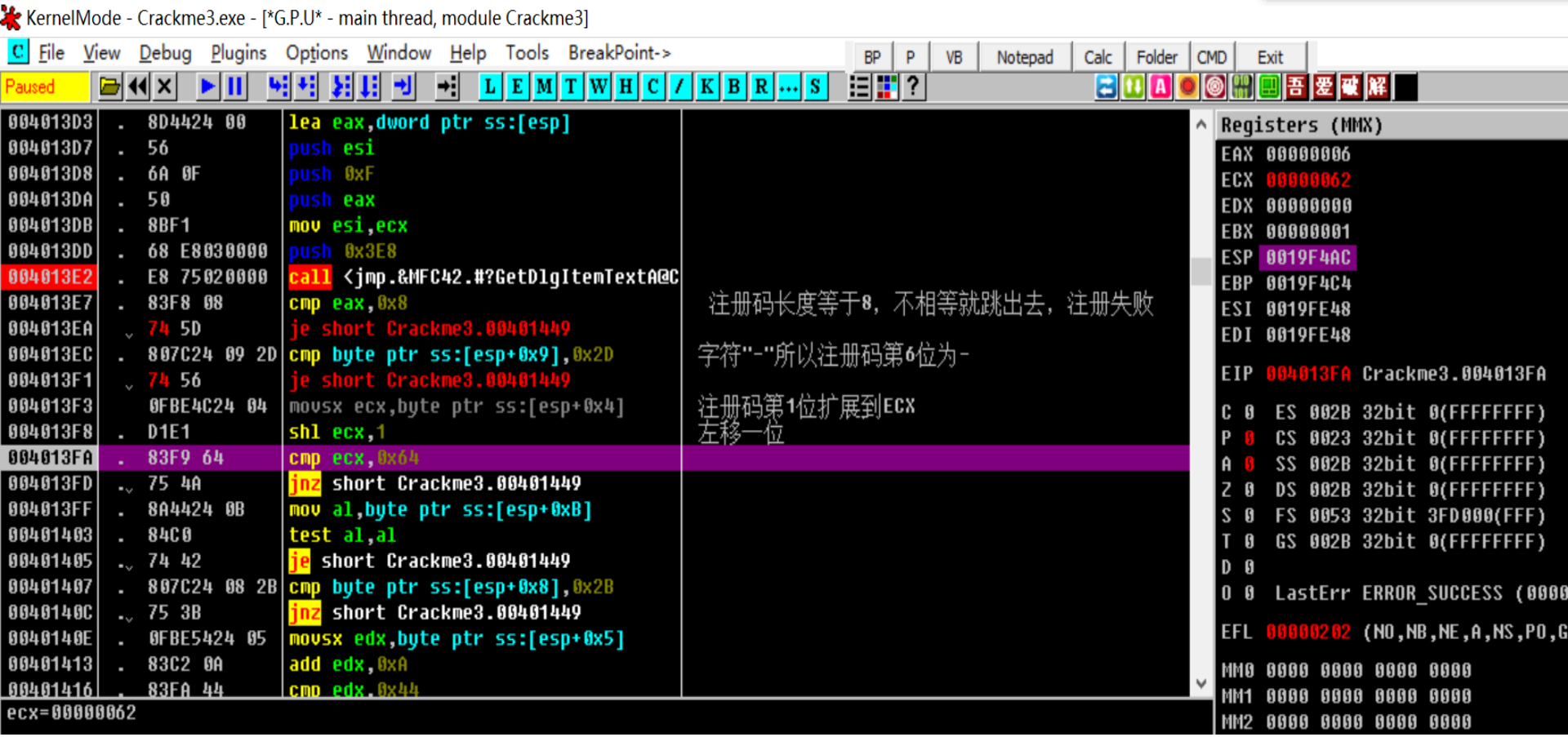
面向汇编的逐句解析 - SS:[esp+0x9]



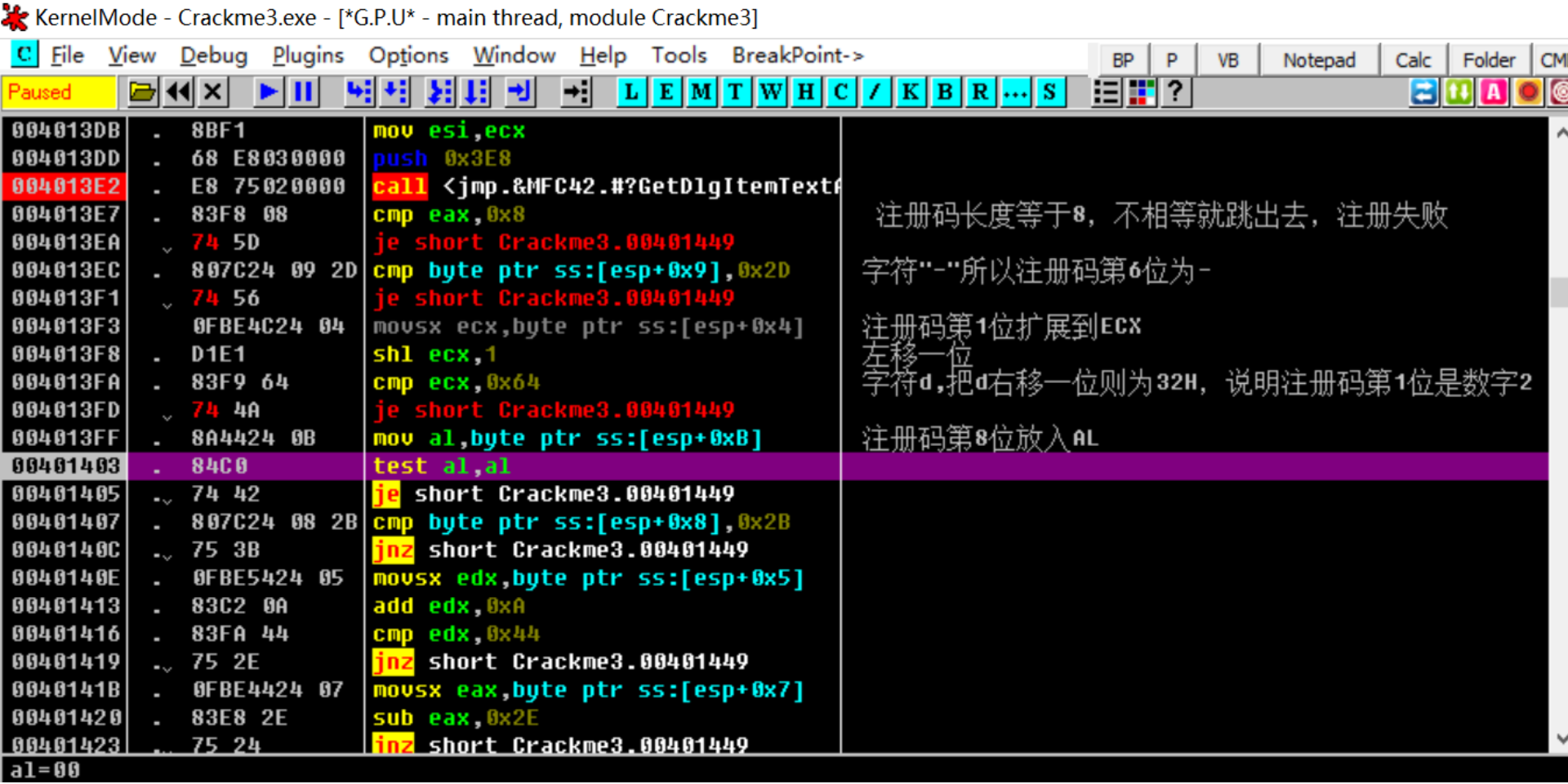
面向汇编的逐句解析 - 字符判断1



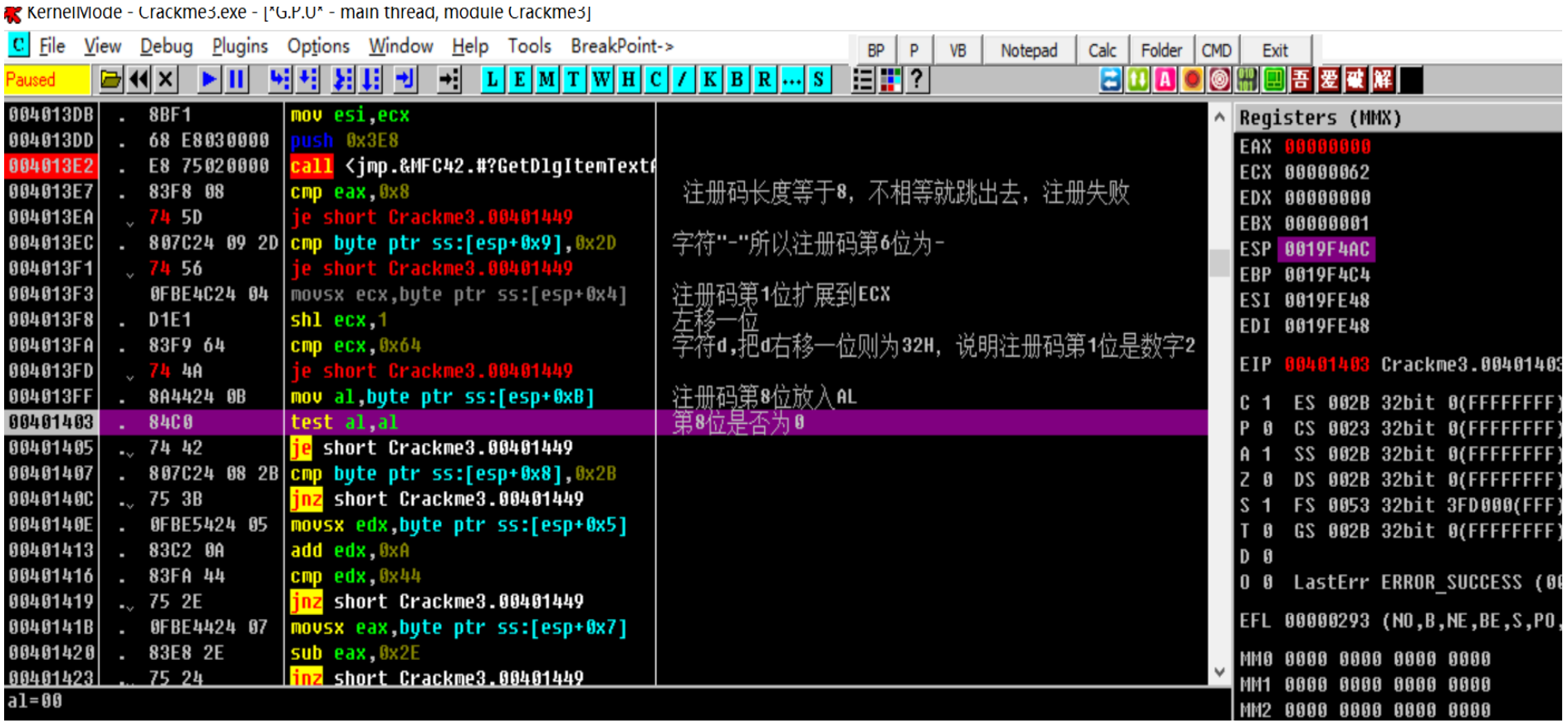
面向汇编的逐句解析 - 字符判断2



面向汇编的逐句解析 - 字符判断3



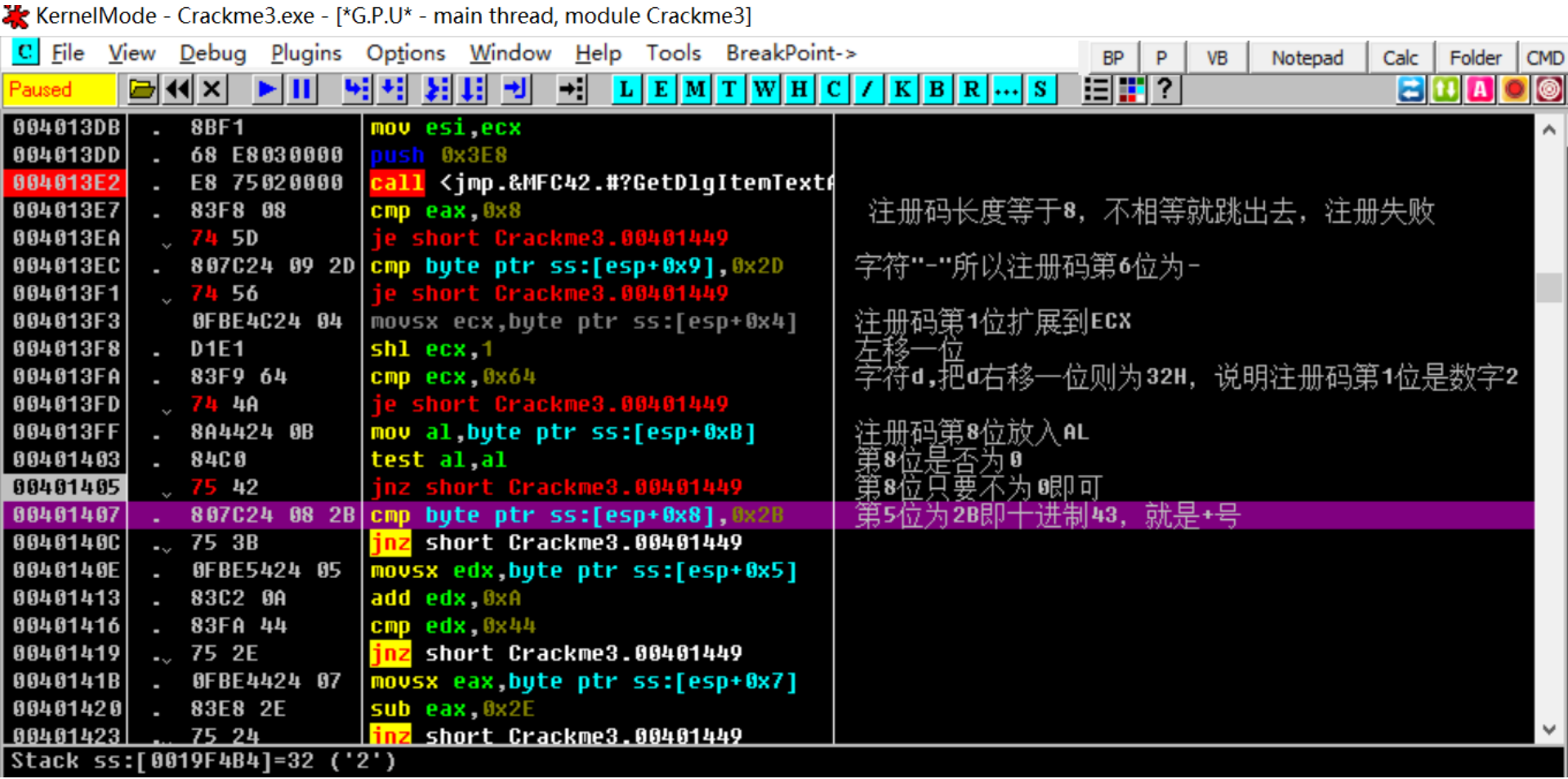
面向汇编的逐句解析 - 字符判断4



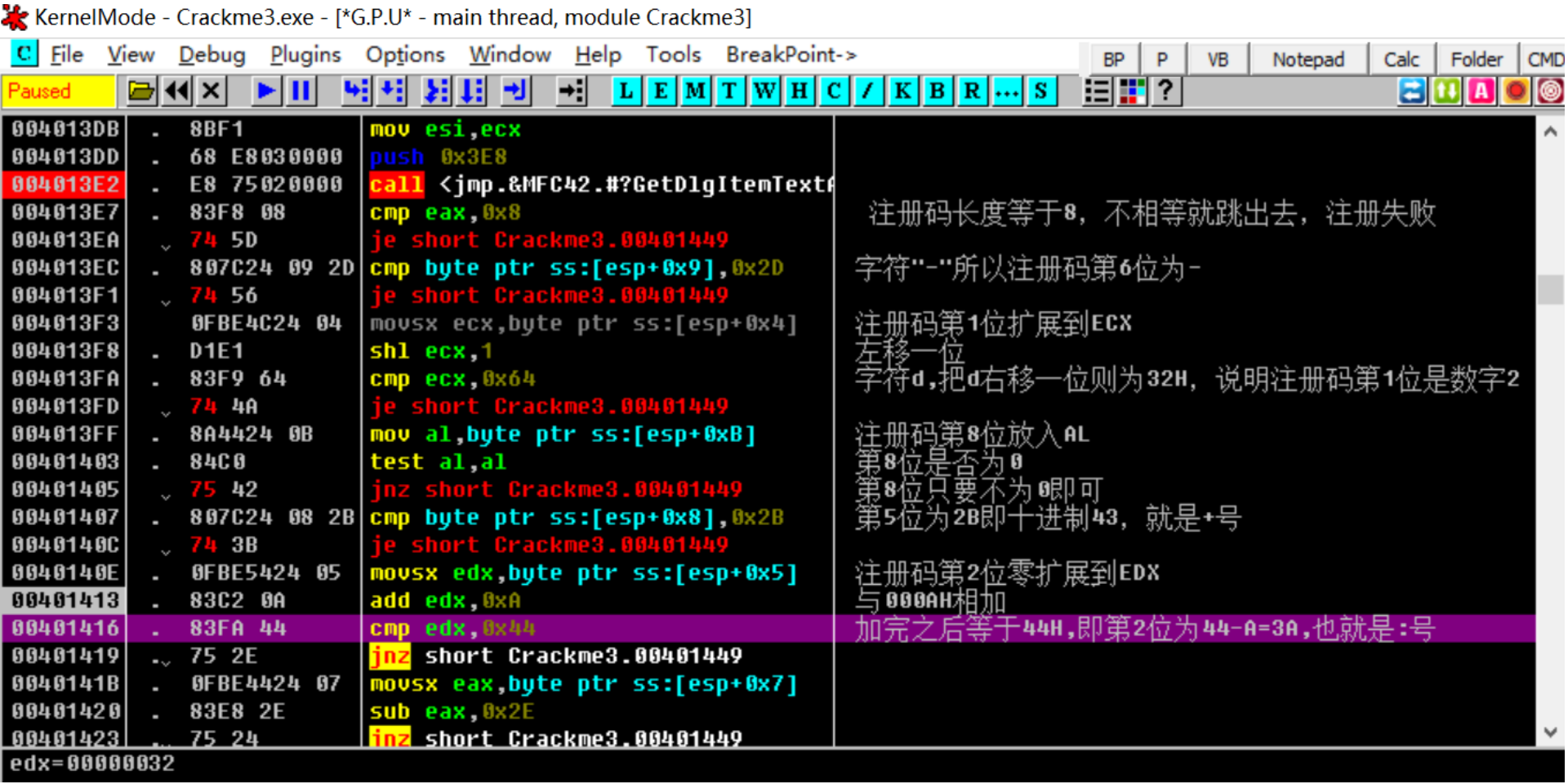
Test将两个操作数进行逻辑与运算：结果是0, ZF标志位为1；结果不是0, ZF为0

当ZF为1时，JE跳转

面向汇编的逐句解析 - 字符判断5



面向汇编的逐句解析 - 字符判断6



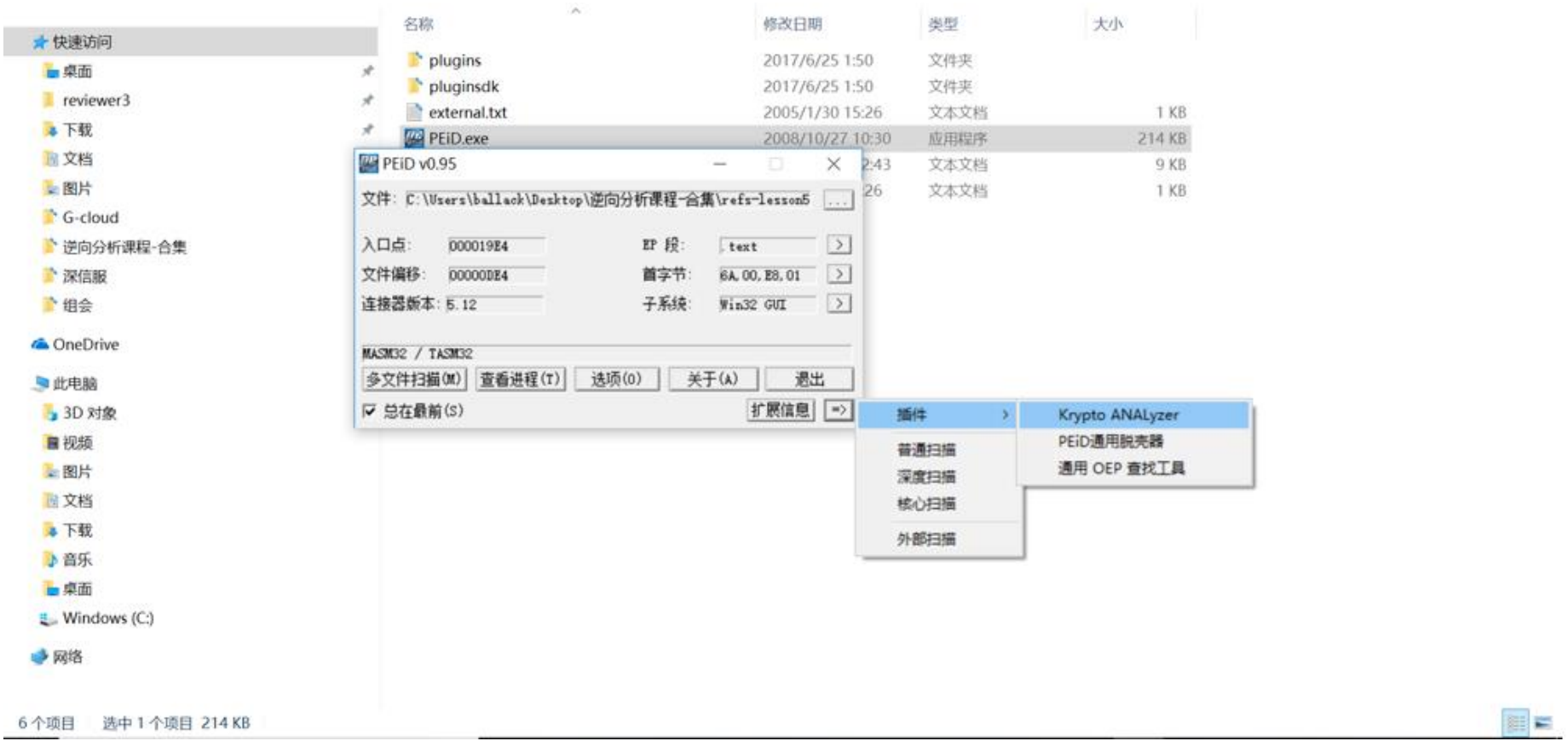
逆向分析技术

加密算法逆向

加密算法逆向 - 示例运行


BytePtr's Crypto kgme...
LilithCPH
Name
Serial
Try Now
About
Exit

加密算法逆向 - PEiD扫描

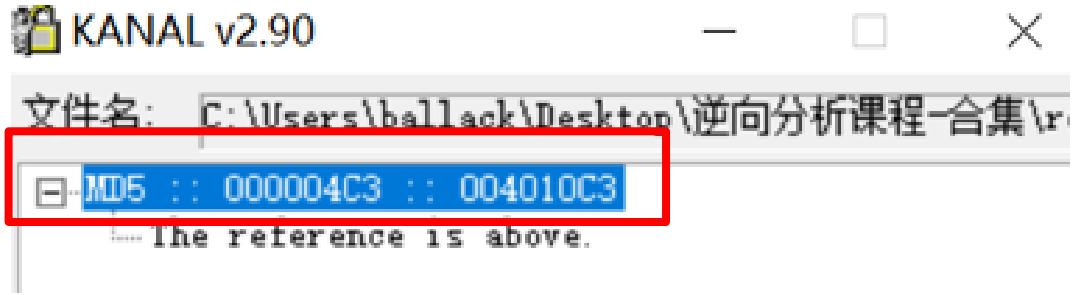


加密算法逆向 - PEiD扫描

CM2：加密算法逆向 - PEiD扫描

```
.text:004010B8                and     eax, ecx
.text:004010BA                or      eax, ebx
.text:004010BC                add     eax, [ebp+var_10]
.text:004010BF                add     eax, [edi+4]
.text:004010C2                add     eax, 0E8C7B756h
.text:004010C7                mov     cl, 0Ch
```

```
004010B3  D3 C0 03 45 F8 89 45 FC  8B 45 FC 8B 5D F8 8B 4D  ...E..E..E..]..M
004010C3  F4 23 D8 F7 D0 23 C1 0B  C3 03 45 F0 03 47 04 05  .#...#....E..G..
004010C3  56 B7 C7 E8 B1 0C D3 C0  03 45 FC 89 45 F0 8B 45  V........E..E..E
004010D3  F0 8B 5D FC 8B 4D F8 23  D8 F7 D0 23 C1 0B C3 03  ..]..M.#...#....
004010E3  45 F4 03 47 08 05 DB 70  20 24 B1 11 D3 C0 03 45  E..G...p $.....E
```



MD5算法是什么？

加密算法逆向 - MD5

1、数据初始化过程

首先要对需加密消息进行数据填充，使消息的长度对512取模后等于448，消息长度 mod 512=448。不足位数在消息（字符串）后面进行填充，填充第一位为1，其余为0。即第一个字节为0x80,其余字节为0x00。

加密算法逆向 - MD5

2、添加消息长度

再填充上**原消息的长度**，例如欲加密字符为16个ASCII，即16字节×8位=128位，128即16进制的0x80。但它是一个QWORD值，占用8个字节(64位)。

在此步骤进行完毕后，最终**消息长度就是512的整数倍**。(448 + 64 =512)

加密算法逆向 - MD5

3、数据加密处理

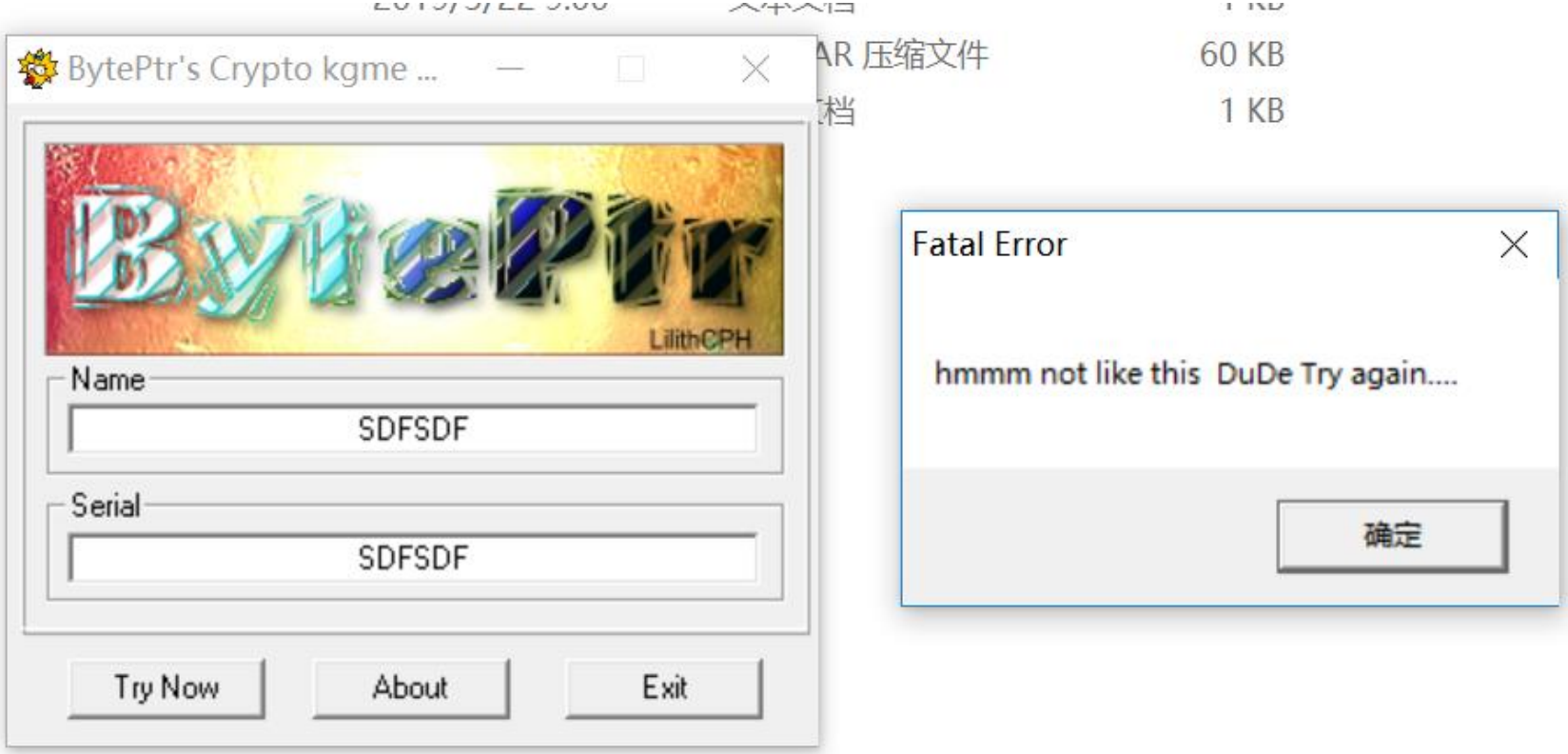
准备需要用到的数据：

4个种子值： A = 0x67452301, B = 0xEFCDAB89, C = 0x98BADCFE, D = 0x10325476;

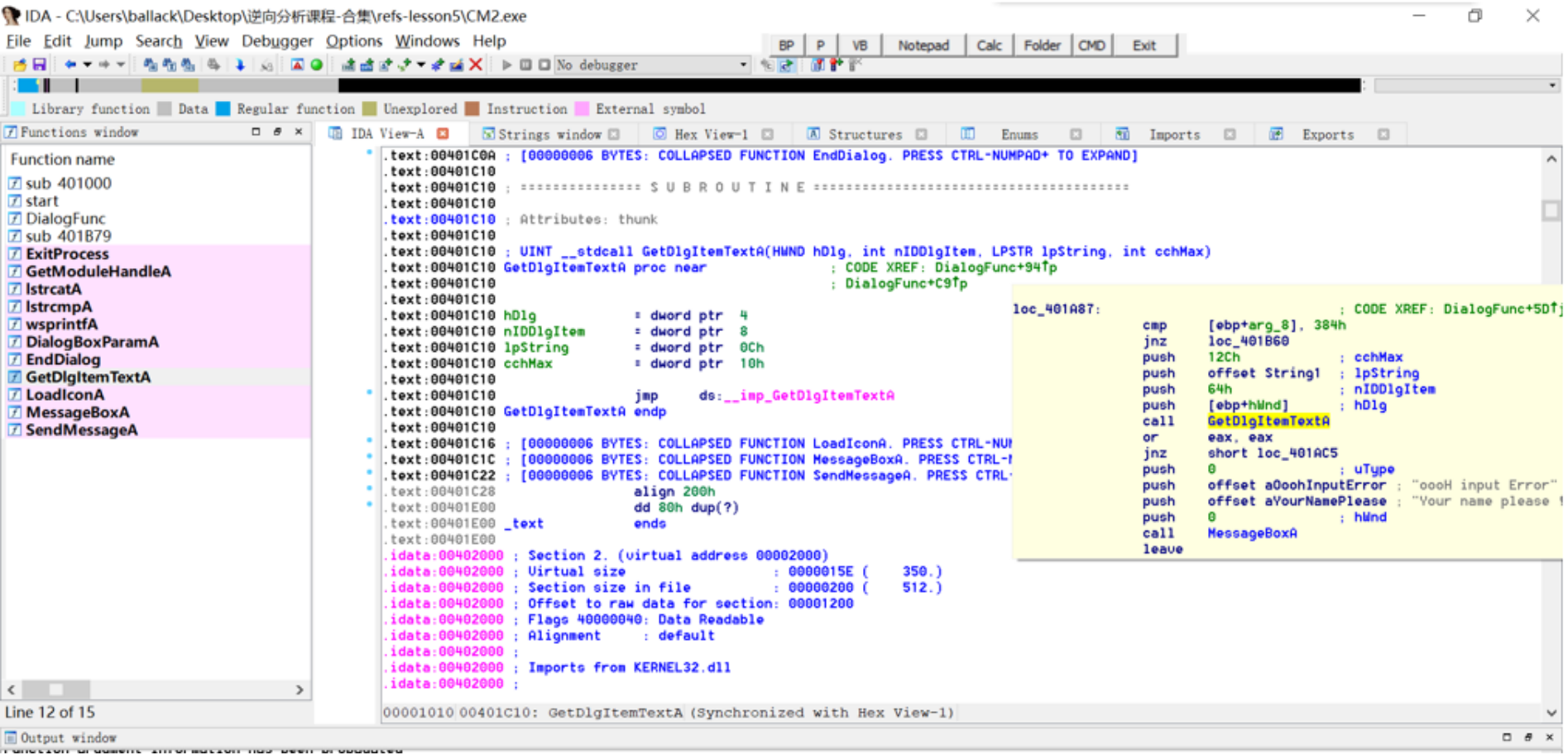
4个函数：F(X,Y,Z)=(X & Y) | ((~X) & Z); G(X,Y,Z)=(X & Z) | (Y & (~Z)); H(X,Y,Z)=X ^ Y ^ Z; I(X,Y,Z)=Y ^ (X | (~Z));

把消息分以512位为一分组进行处理，每一个分组进行4轮变换，以上面所说4个常数为起始变量进行计算，重新输出4个变量，以这4个变量再进行下一分组的运算，如果已经是最后一个分组，则这4个变量为最后的结果，即MD5值。

加密算法逆向 - MD5


AR 压缩文件 60 KB
1 KB
BytePtr's Crypto kgme ...
LilithCPH
Name
SDFSDF
Serial
SDFSDF
Try Now
About
Exit
Fatal Error
hmmm not like this DuDe Try again....
确定

加密算法逆向 - MD5 - 找入口

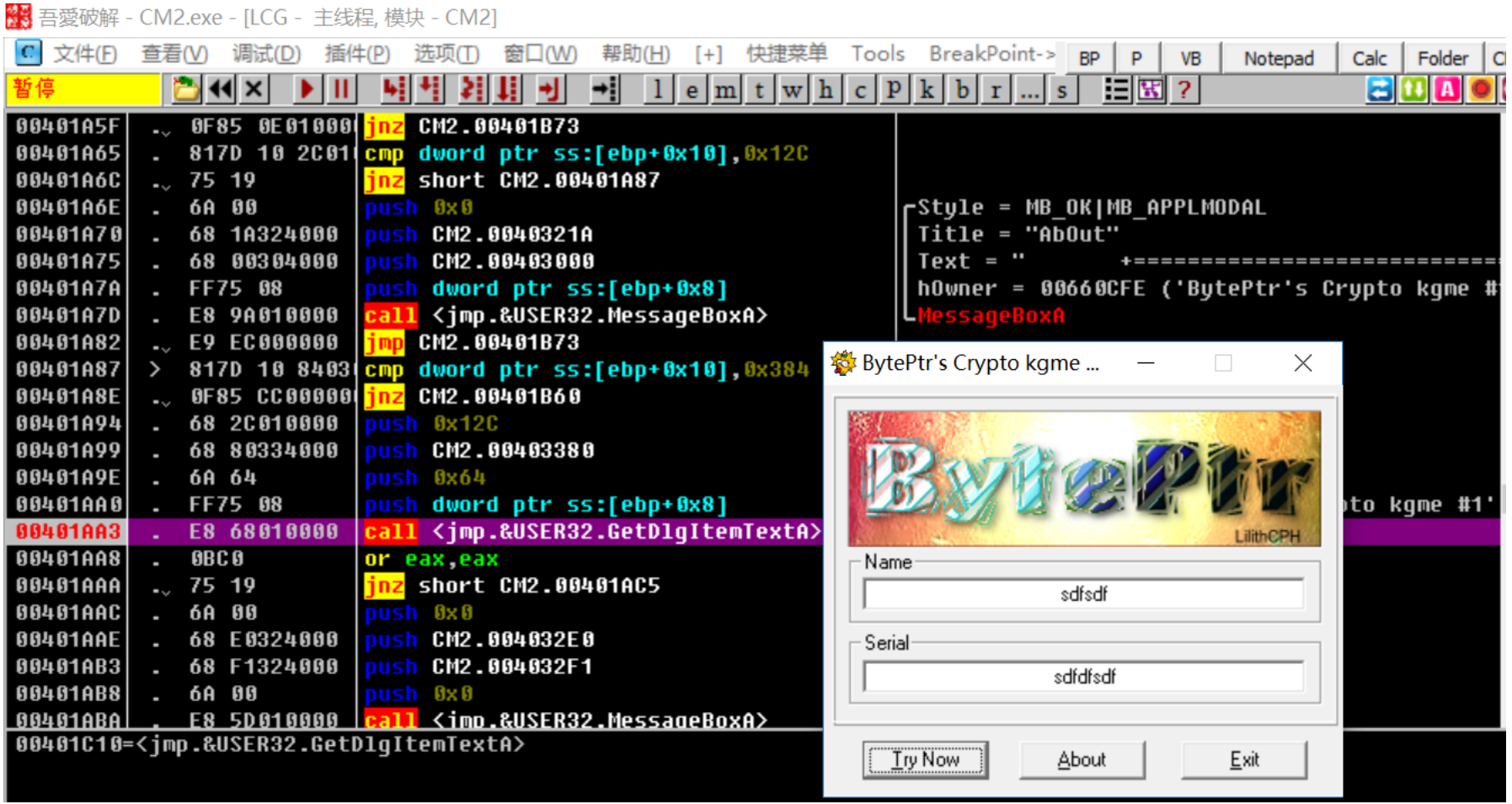


加密算法逆向 - MD5 - 找入口

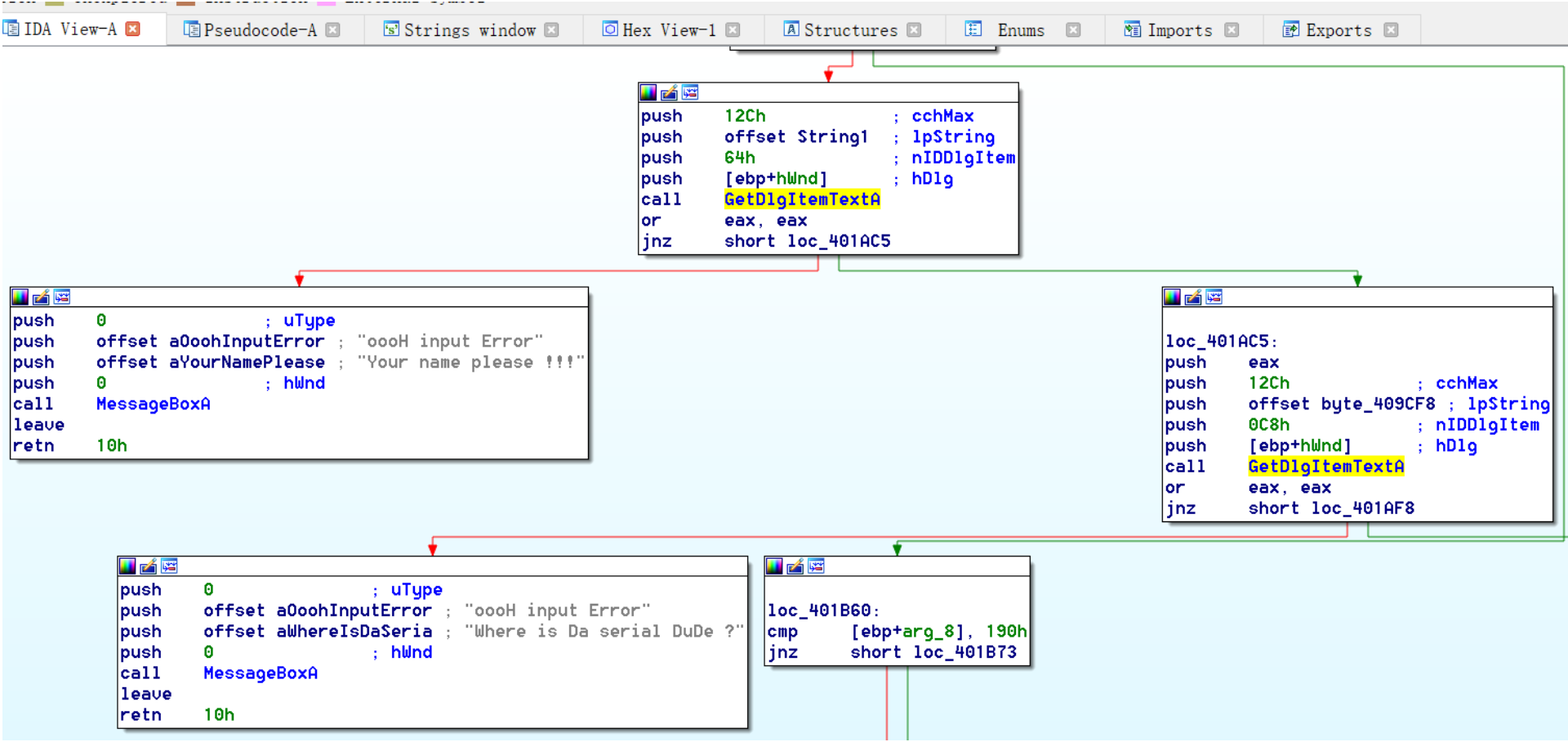
```
.text:00401A87 loc_401A87:                             ; CODE XREF: DialogFunc+5D↑j
.text:00401A87                 cmp     [ebp+arg_8], 384h
.text:00401A8E                 jnz     loc_401B60
.text:00401A94                 push    12Ch            ; cchMax
.text:00401A99                 push    offset String1  ; lpString
.text:00401A9E                 push    64h             ; nIDDlgItem
.text:00401AA0                 push    [ebp+hWnd]      ; hDlg
.text:00401AA3                 call    GetDlgItemTextA
```

```
00401A6E  .  6A 00          push 0x0                    ┌Style = MB_OK|MB_APPLMODAL
00401A70  .  68 1A324000    push CM2.0040321A           │Title = "AbOut"
00401A75  .  68 00304000    push CM2.00403000           │Text = "        +==============
00401A7A  .  FF75 08        push dword ptr ss:[ebp+0x8] │hOwner = 002D8000
00401A7D  .  E8 9A010000    call <jmp.&USER32.MessageBoxA> └MessageBoxA
00401A82  .v E9 EC000000    jmp CM2.00401B73
00401A87  >  817D 10 8403   cmp dword ptr ss:[ebp+0x10],0x384
00401A8E  .v 0F85 CC00000   jnz CM2.00401B60
00401A94  .  68 2C010000    push 0x12C                  ┌Count = 12C (300.)
00401A99  .  68 80334000    push CM2.00403380           │Buffer = CM2.00403380
00401A9E  .  6A 64          push 0x64                   │ControlID = 64 (100.)
00401AA0  .  FF75 08        push dword ptr ss:[ebp+0x8] │hWnd = 002D8000
00401AA3  .  E8 68010000    call <jmp.&USER32.GetDlgItemTextA> └GetDlgItemTextA
00401AA8  .  0BC0           or eax,eax
00401AAA  .v 75 19          jnz short CM2.00401AC5
00401AAC  .  6A 00          push 0x0                    ┌Style = MB_OK|MB_APPLMODAL
00401AAE  .  68 E0324000    push CM2.004032E0           │Title = "oooH input Error"
00401AB3  .  68 F1324000    push CM2.004032F1           │Text = "Your name please !!!"
00401AB8  .  6A 00          push 0x0                    │hOwner = NULL
00401ABA  .  E8 5D010000    call <jmp.&USER32.MessageBoxA> └MessageBoxA
```

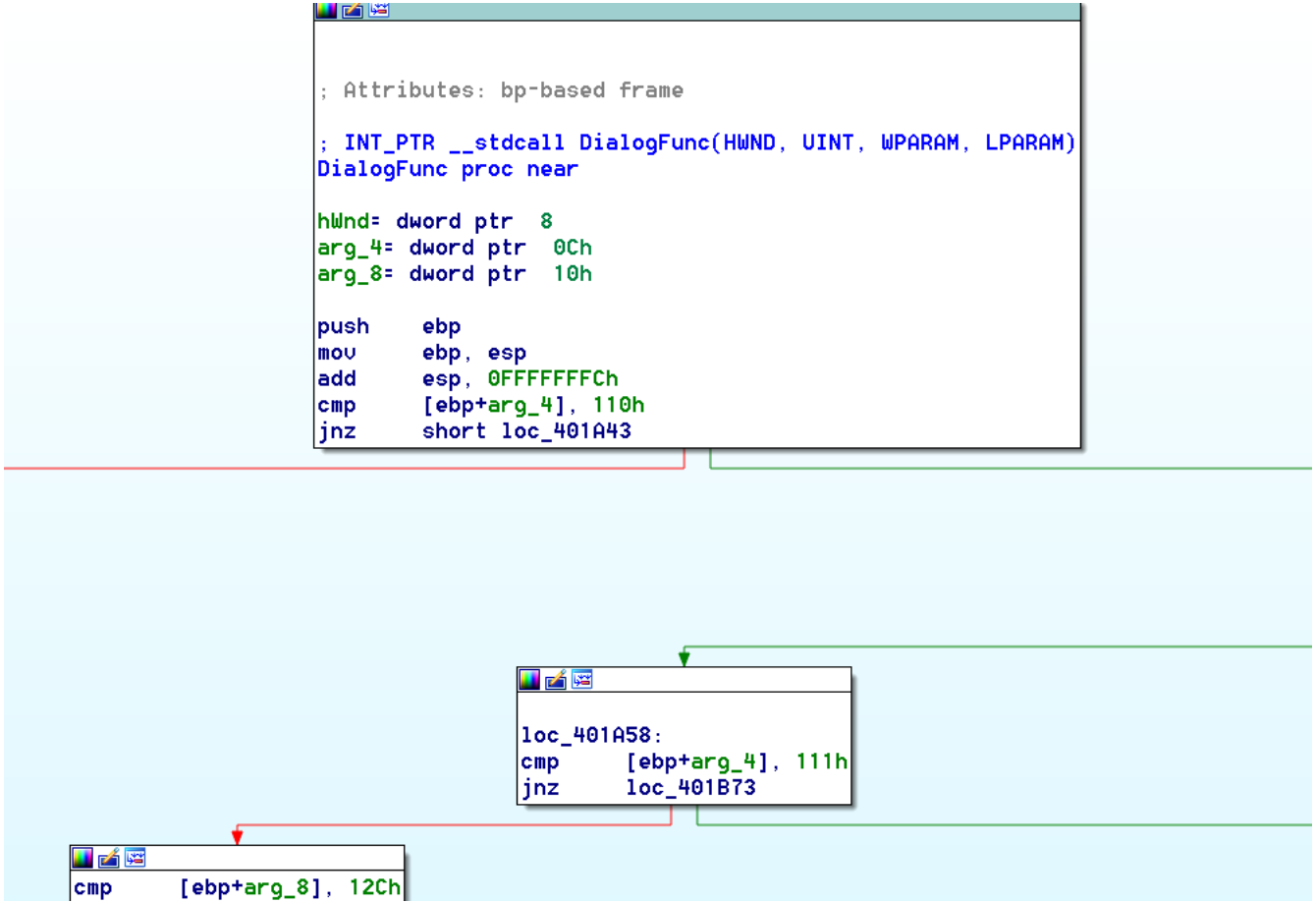
加密算法逆向 - MD5 - 取用户名



加密算法逆向 - MD5 - IDA看代码



加密算法逆向 - MD5 - 开启F5



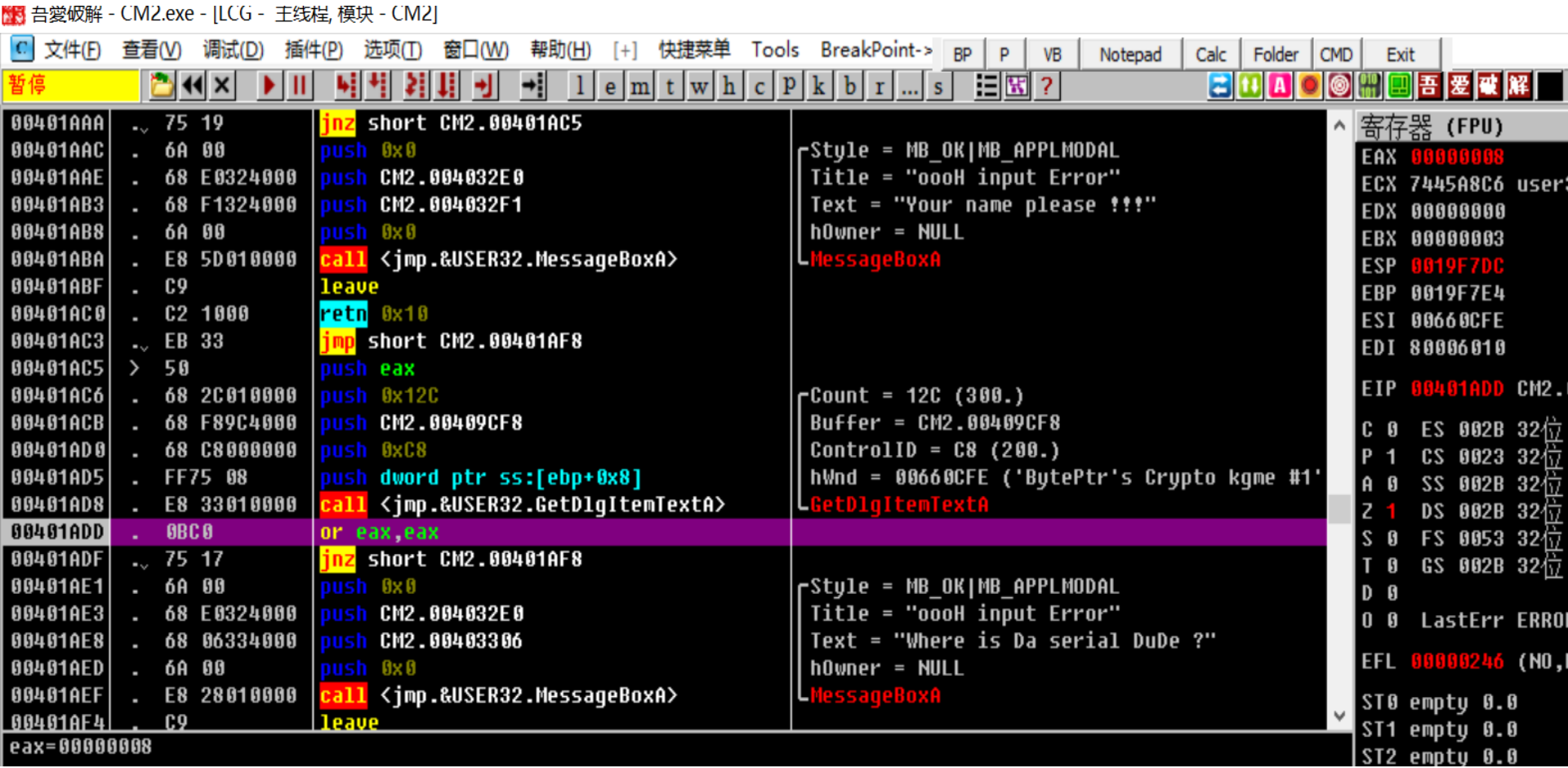
加密算法逆向 - MD5 - 开启F5

```
7  switch ( a2 )
8  {
9    case 0x110u:
10     v4 = LoadIconA(hInstance, (LPCSTR)0x1F4);
11     SendMessageA(hWnd, 0x80u, 0, (LPARAM)v4);
12     break;
13   case 0x10u:
14     EndDialog(hWnd, 0);
15     break;
16   case 0x111u:
17     switch ( a3 )
18     {
19       case 0x12Cu:
20         MessageBoxA(hWnd, Text, Caption, 0);
21         break;
22       case 0x384u:
23         v5 = GetDlgItemTextA(hWnd, 100, String1, 300);
24         if ( !v5 )
25           return MessageBoxA(0, aYourNamePlease, aOoohInputError, 0);
26         v7 = v5;
27         if ( !GetDlgItemTextA(hWnd, 200, byte_409CF8, 300) )
28           return MessageBoxA(0, aWhereIsDaSeria, aOoohInputError, 0);
29         lstrcatA(String1, String2);
30         sub_401000(String1, v7, &unk_4056A8);
31         sub_401B79();
32         if ( lstrcmpA(byte_409CF8, byte_4079D0) )
33         {
34           MessageBoxA(0, aHmmmNotLikeThi, aFatalError, 0);
35           return 0;
36         }
37         MessageBoxA(0, aGoodSerialNowS, aGoodWork, 0);
38         break;
39       case 0x190u:
40         EndDialog(hWnd, 0);
```

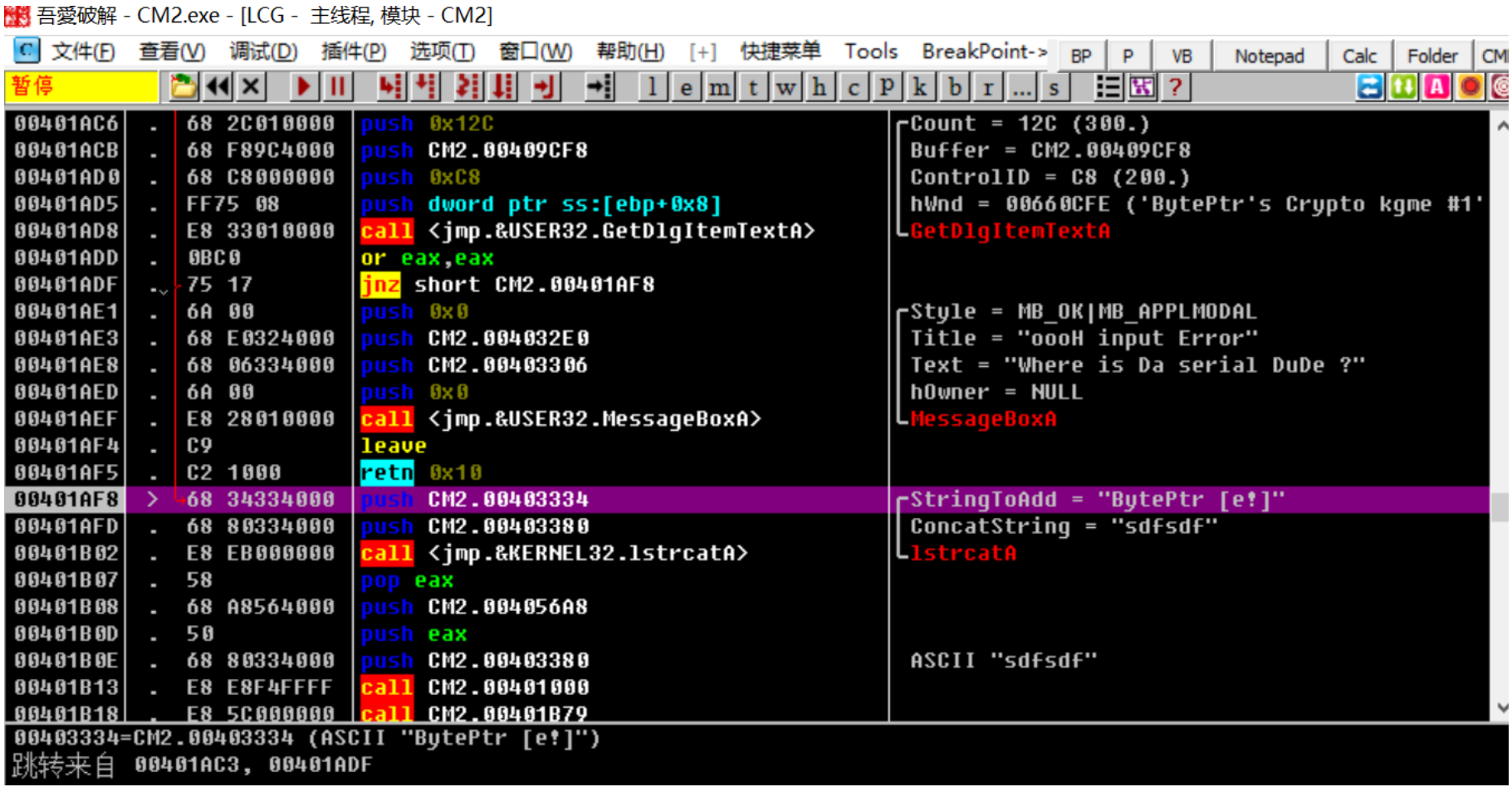
加密算法逆向 - MD5 - 核心位置

```
22          case 0x384u:
23            v5 = GetDlgItemTextA(hWnd, 100, String1, 300);
24            if ( !v5 )
25              return MessageBoxA(0, aYourNamePlease, aOoohInputError, 0);
26            v7 = v5;
27            if ( !GetDlgItemTextA(hWnd, 200, byte_409CF8, 300) )
28              return MessageBoxA(0, aWhereIsDaSeria, aOoohInputError, 0);
29            lstrcatA(String1, String2);
30            sub_401000(String1, v7, &unk_4056A8);
31            sub_401B79();
32            if ( lstrcmpA(byte_409CF8, byte_4079D0) )
33            {
```

加密算法逆向 - MD5 - 核心位置

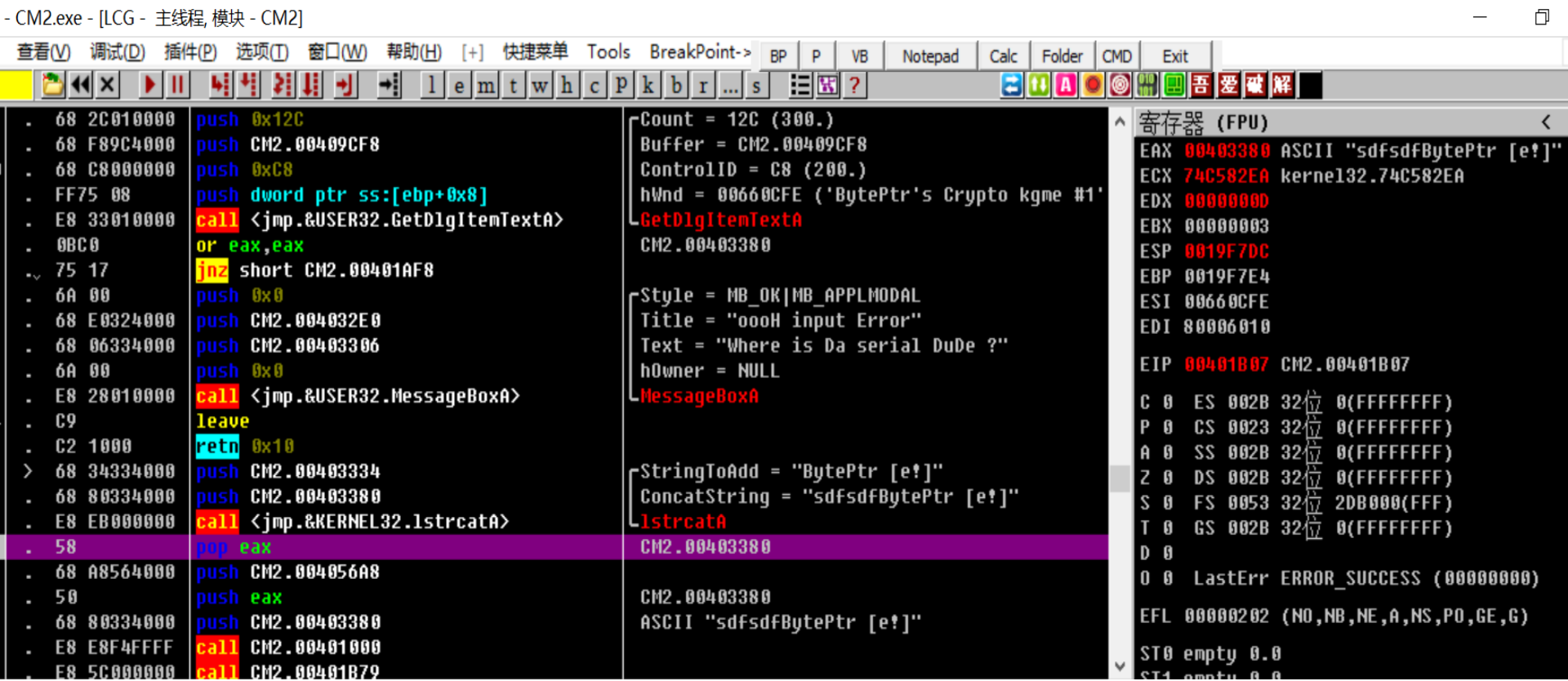


加密算法逆向 - MD5 - 开始比较

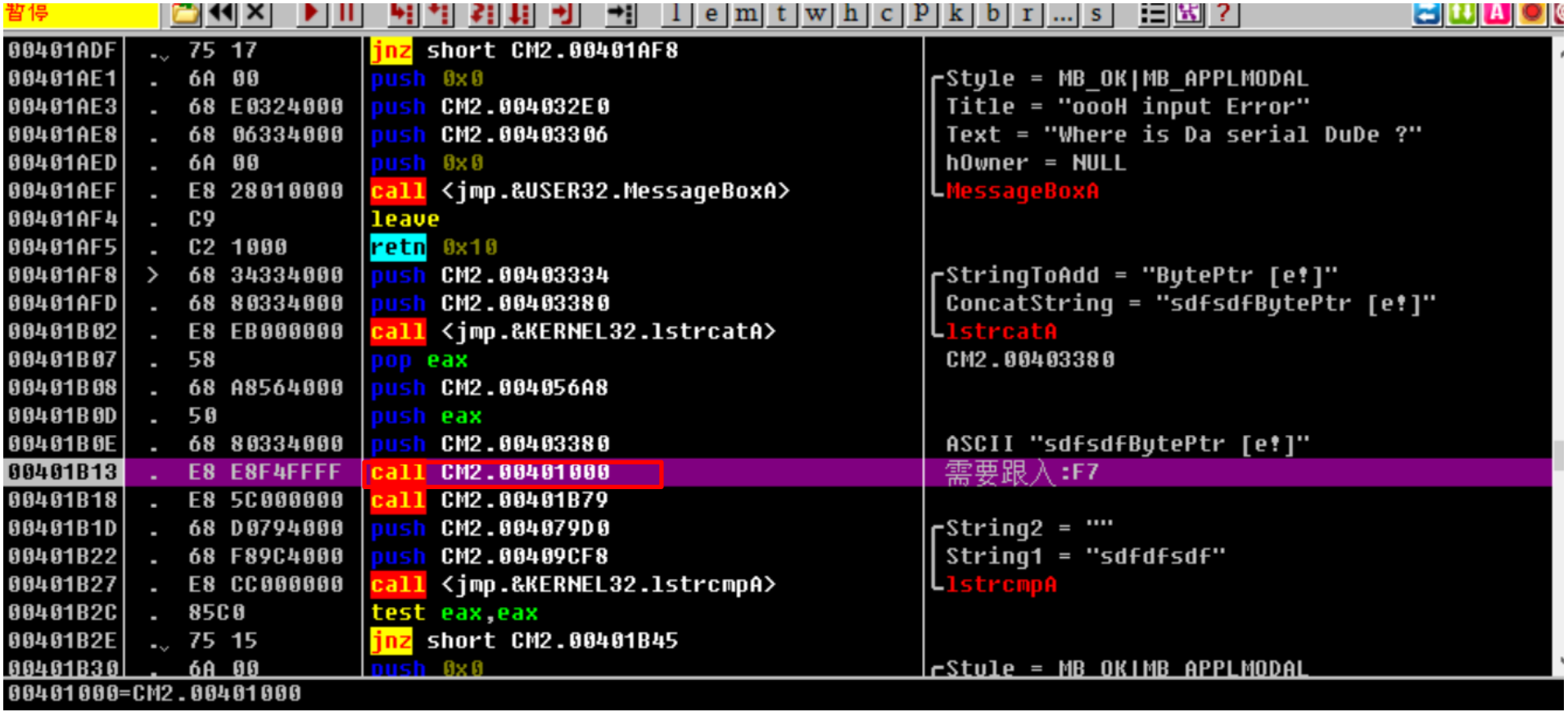


基于输入的用户名生成MD5值，然后与输入的序列号进行对比

加密算法逆向 - MD5 - 生成字符串X



加密算法逆向 - MD5 - 函数的步入处理



加密算法逆向 - MD5 - 定位MD5

```
.text:00401B07                pop     eax
.text:00401B08                push    offset unk_4056A8
.text:00401B0D                push    eax
.text:00401B0E                push    offset String1
.text:00401B13                call    sub_401000
.text:00401B18                call    sub_401B7
.text:00401B1D                push    offset b
.text:00401B22                push    offset b
.text:00401B27                call    lstrcmpA
.text:00401B2C                test    eax, eax
.text:00401B2E                jnz     short lo
.text:00401B30                push    0
.text:00401B32                push    offset a
.text:00401B37                push    offset a
```

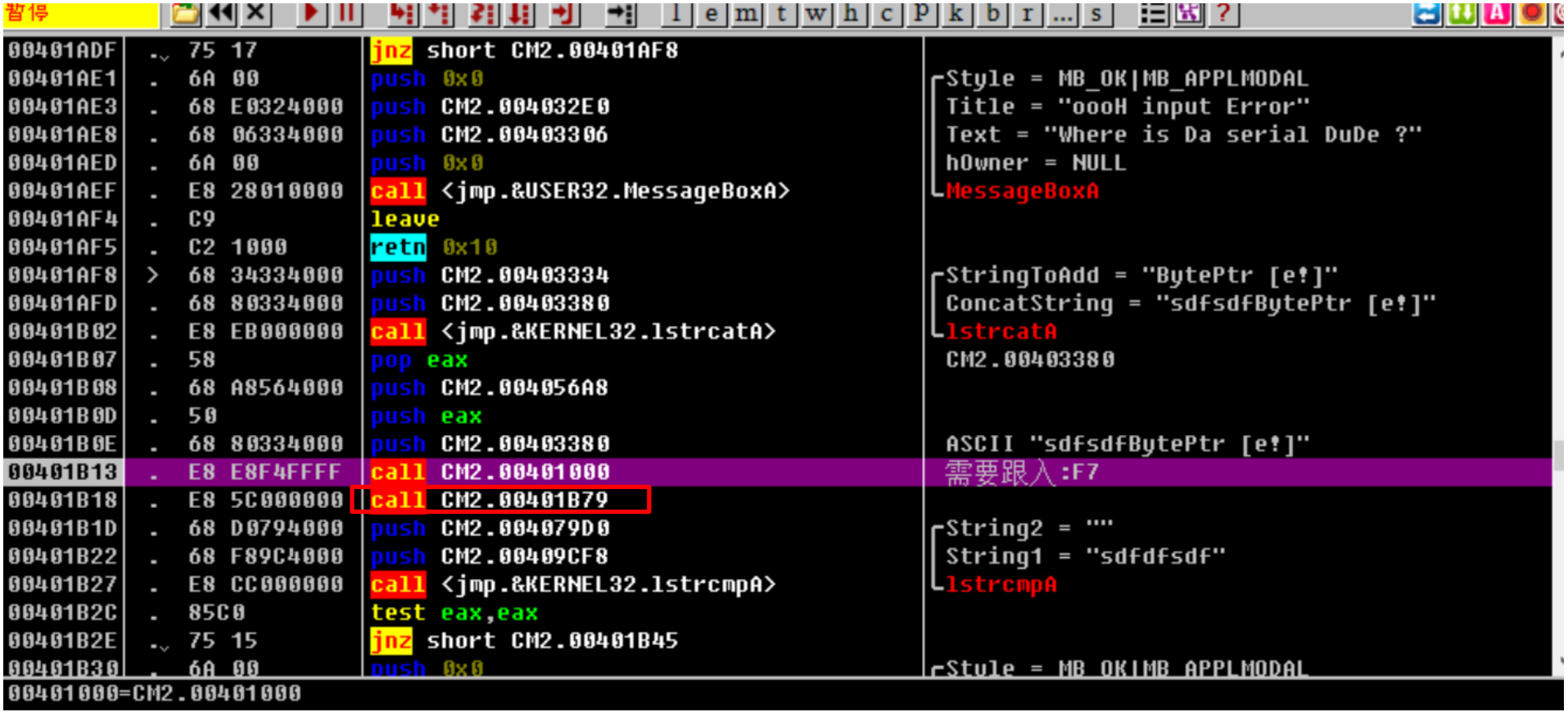
```
var_10          = dword ptr -10h
var_C           = dword ptr -0Ch
var_8           = dword ptr -8
var_4           = dword ptr -4
arg_0           = dword ptr  8
arg_4           = dword ptr  0Ch
arg_8           = dword ptr  10h
```

w send your Tut & KeyGen".

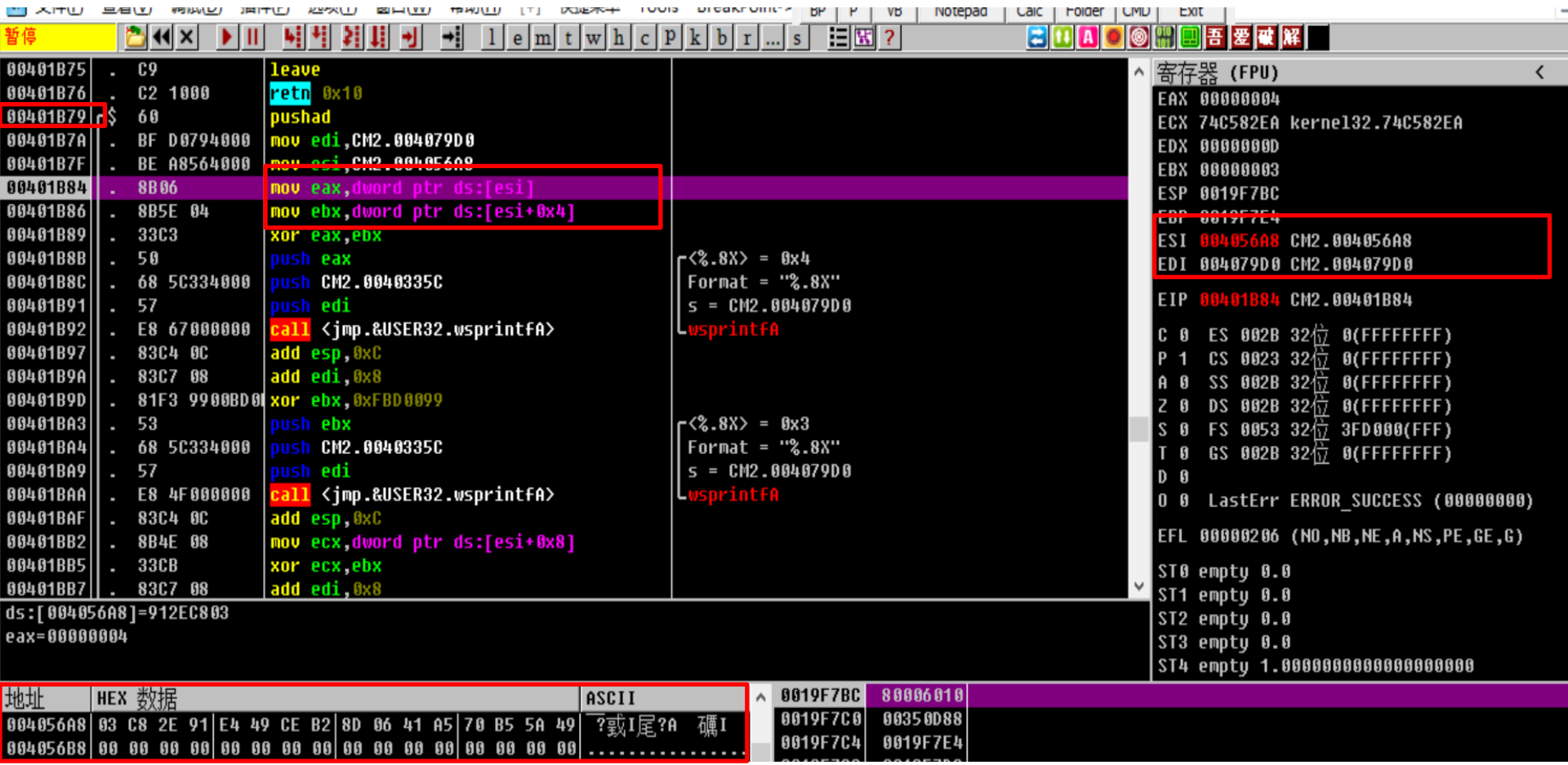
```
.text:00401048                mov     [edi-4], edx
.text:0040104B                mov     edx, [ebp+arg_4]
.text:0040104E                mov     edi, [ebp+arg_0]
.text:00401051                mov     esi, [ebp+arg_8]
.text:00401054                mov     dword ptr [esi], 67452301h
.text:0040105A                mov     dword ptr [esi+4], 0EFCDAB89h
.text:00401061                mov     dword ptr [esi+8], 98BADCFEh
.text:00401068                mov     dword ptr [esi+0Ch], 10325476h
.text:0040106F
.text:0040106F loc_40106F:                            ; CODE XREF: sub_401000+9BF↓j
```

种子值： A = 0x67452301, B = 0xEFCDAB89,
C = 0x98BADCFE, D = 0x10325476;

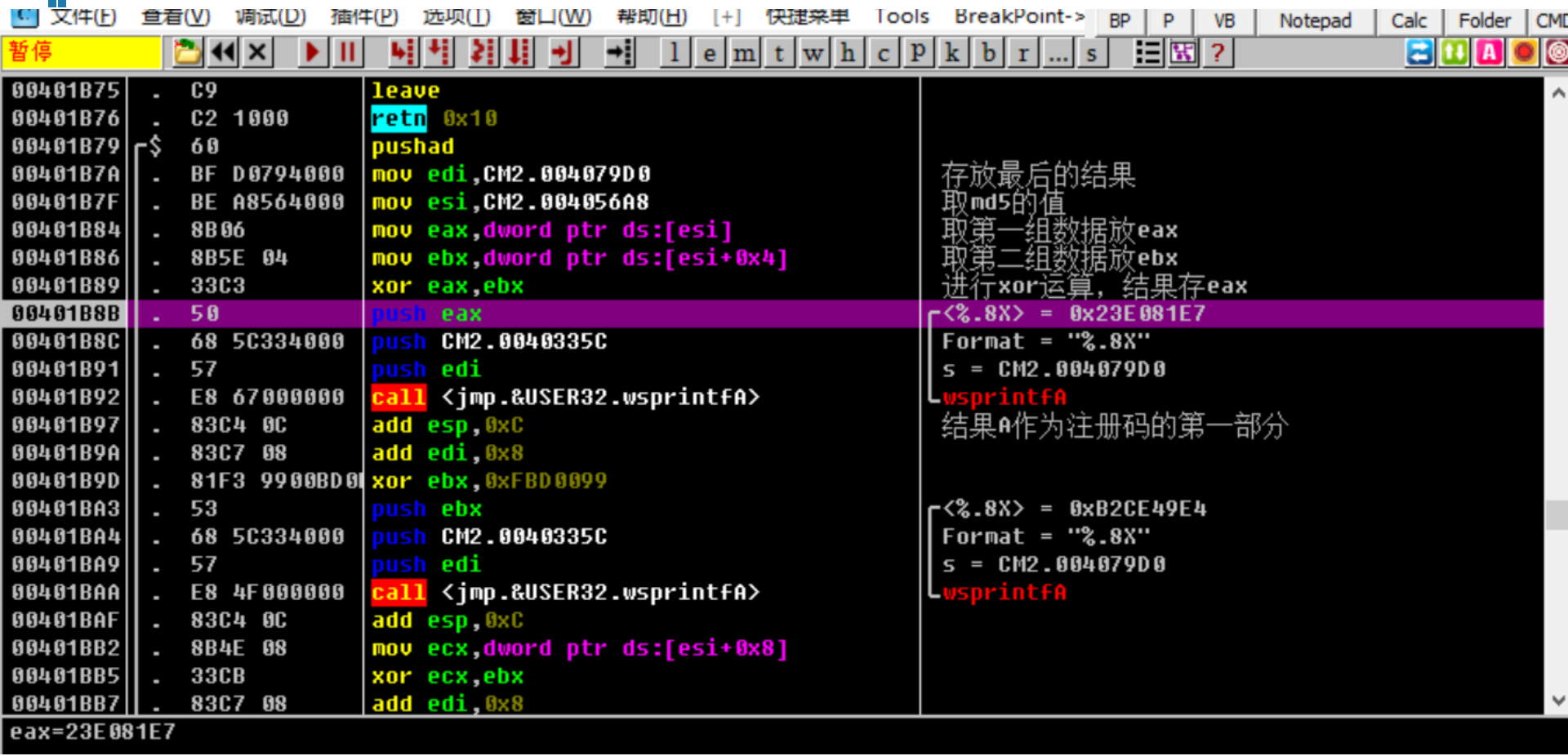
加密算法逆向 - MD5 - 函数的步入处理



加密算法逆向 - MD5 - 调试中比较



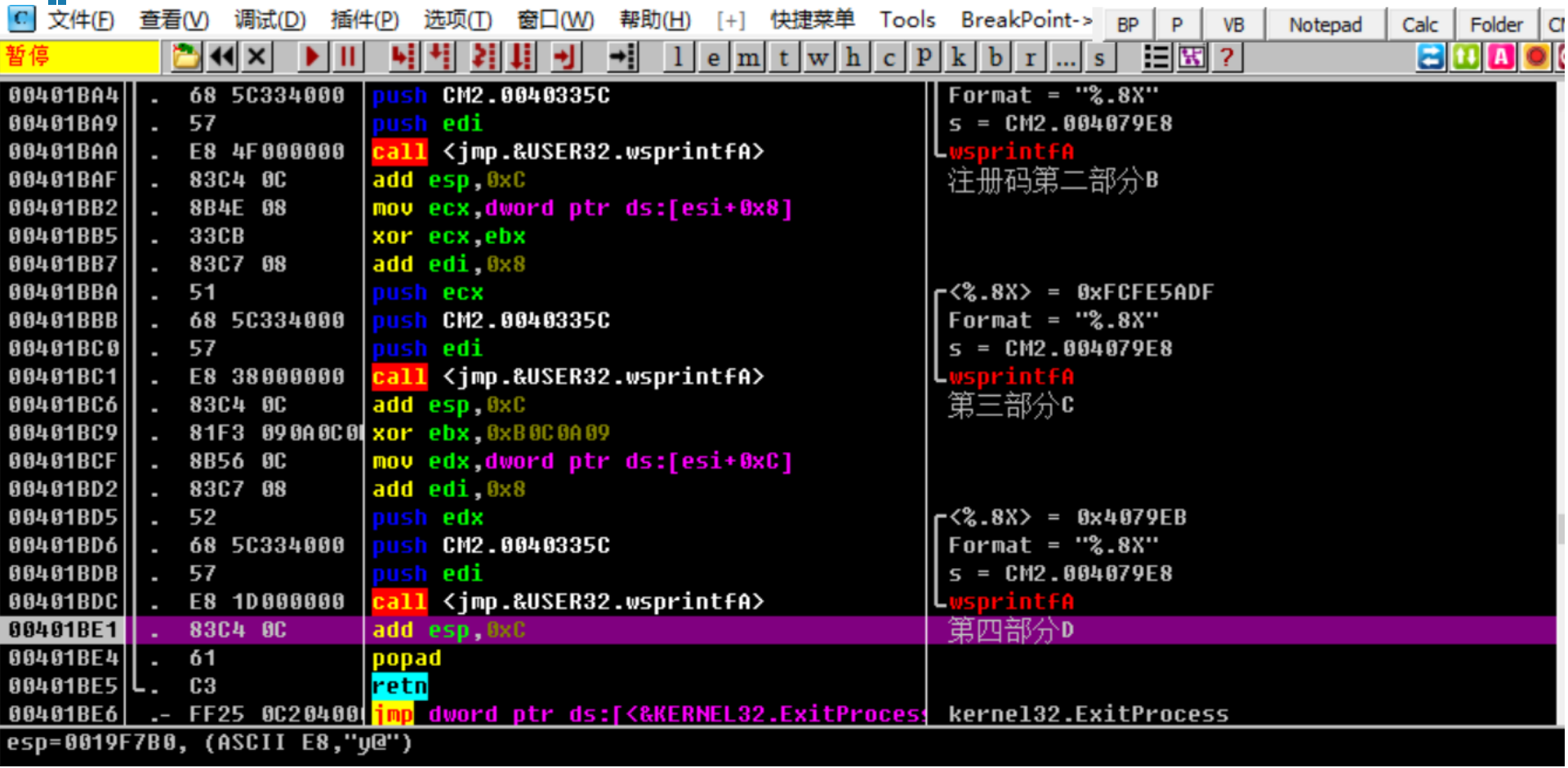
加密算法逆向 - MD5 - 注册码第一部分



加密算法逆向 - MD5 - 注册码第二部分

```
00401B75   .  C9              leave
00401B76   .  C2 1000         retn 0x10
00401B79  r$  60              pushad
00401B7A   .  BF D0794000     mov edi,CM2.004079D0          存放最后的结果
00401B7F   .  BE A8564000     mov esi,CM2.004056A8          取md5的值
00401B84   .  8B06            mov eax,dword ptr ds:[esi]    取第一组数据放eax
00401B86   .  8B5E 04         mov ebx,dword ptr ds:[esi+0x4] 取第二组数据放ebx
00401B89   .  33C3            xor eax,ebx                   进行xor运算，结果存eax
00401B8B   .  50              push eax                      <%.8X> = 0x8
00401B8C   .  68 5C334000     push CM2.0040335C             Format = "%.8X"
00401B91   .  57              push edi                      s = CM2.004079D8
00401B92   .  E8 67000000     call <jmp.&USER32.wsprintfA>  wsprintfA
00401B97   .  83C4 0C         add esp,0xC                   结果A作为注册码的第一部分
00401B9A   .  83C7 08         add edi,0x8
00401B9D   .  81F3 9900BD0I   xor ebx,0xFBD0099             获得结果B
00401BA3   .  53              push ebx                      <%.8X> = 0xBD73497D
00401BA4   .  68 5C334000     push CM2.0040335C             Format = "%.8X"
00401BA9   .  57              push edi                      s = CM2.004079D8
00401BAA   .  E8 4F000000     call <jmp.&USER32.wsprintfA>  wsprintfA
00401BAF   .  83C4 0C         add esp,0xC                   注册码第二部分
00401BB2   .  8B4E 08         mov ecx,dword ptr ds:[esi+0x8]
00401BB5   .  33CB            xor ecx,ebx
00401BB7   .  83C7 08         add edi,0x8
ebx=BD73497D
```

加密算法逆向 - MD5 - 注册码第三、四部分



加密算法逆向 - 最后进行比较

```
00401B7A  .  BF D0794000   mov edi,CM2.004079D0                存放最后的结果
```

```
00401B07  .  58            pop eax
00401B08  .  68 A8564000   push CM2.004056A8
00401B0D  .  50            push eax
00401B0E  .  68 80334000   push CM2.00403380
00401B13  .  E8 E8F4FFFF   call CM2.00401000
00401B18  .  E8 5C000000   call CM2.00401B79
00401B1D  .  68 D0794000   push CM2.004079D0
00401B22  .  68 F89C4000   push CM2.00409CF8
00401B27  .  E8 CC000000   call <jmp.&KERNEL32.lstrcmpA>
00401B2C  .  85C0          test eax,eax
00401B2E  .  75 15         jnz short CM2.00401B45
00401B30  .  6A 00         push 0x0
00401B32  .  68 20324000   push CM2.00403220
00401B37  .  68 2A324000   push CM2.0040322A
00401B3C  .  6A 00         push 0x0
00401B3E  .  E8 D9000000   call <jmp.&USER32.MessageBoxA>
00401B43  .  EB 2E         jmp short CM2.00401B73
00401B45  >  6A 00         push 0x0
00401B47  .  68 AD324000   push CM2.004032AD
00401B4C  .  68 B9324000   push CM2.004032B9
00401B51  .  6A 00         push 0x0
00401B53  .  E8 C4000000   call <jmp.&USER32.MessageBoxA>
00401B58  .  33C0          xor eax,eax
```

计算结果

输入的序列号

感谢大家！


南京郵電大學
Nanjing University of Posts and Telecommunications