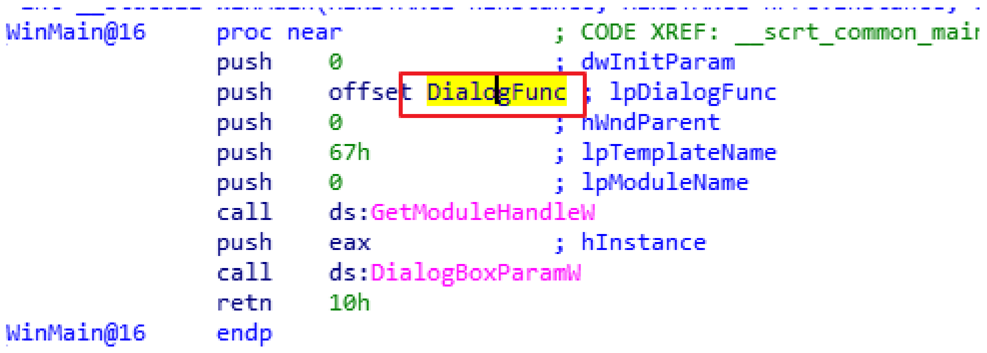
2016-pediy-ctf-04

笔记本： CTF
创建时间： 2019/3/18 9:24 **更新时间：** 2019/3/18 14:32

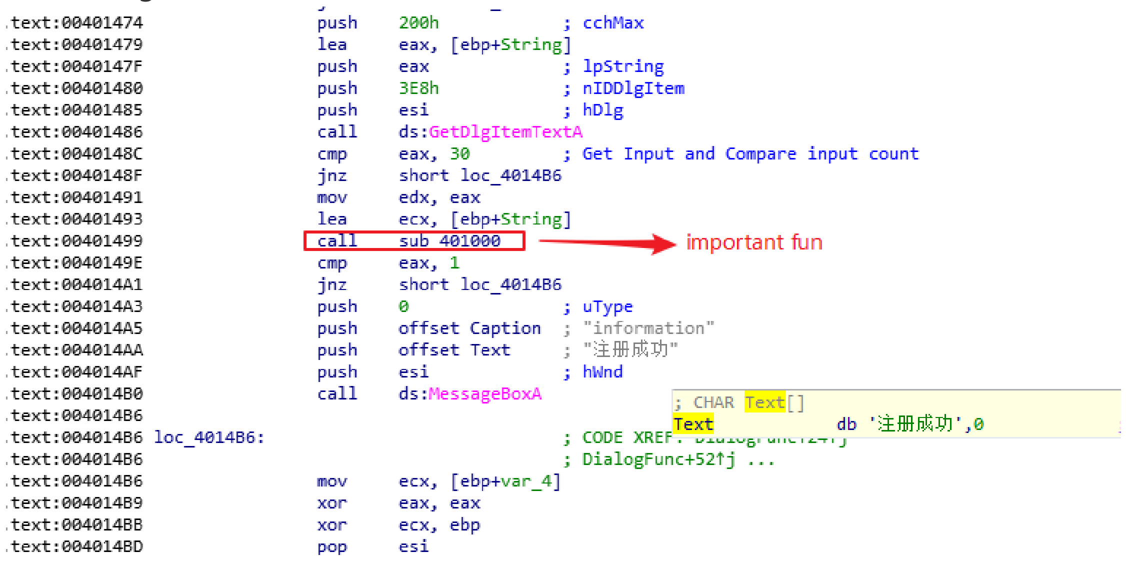
2016-pediy-ctf-04

发现主程序很简单，主要过程在窗口回调



```
WinMain@16      proc near               ; CODE XREF: __scrt_common_mai
                push    0               ; dwInitParam
                push    offset DialogFunc ; lpDialogFunc
                push    0               ; hWndParent
                push    67h             ; lpTemplateName
                push    0               ; lpModuleName
                call    ds:GetModuleHandleW
                push    eax             ; hInstance
                call    ds:DialogBoxParamW
                retn    10h
WinMain@16      endp
```

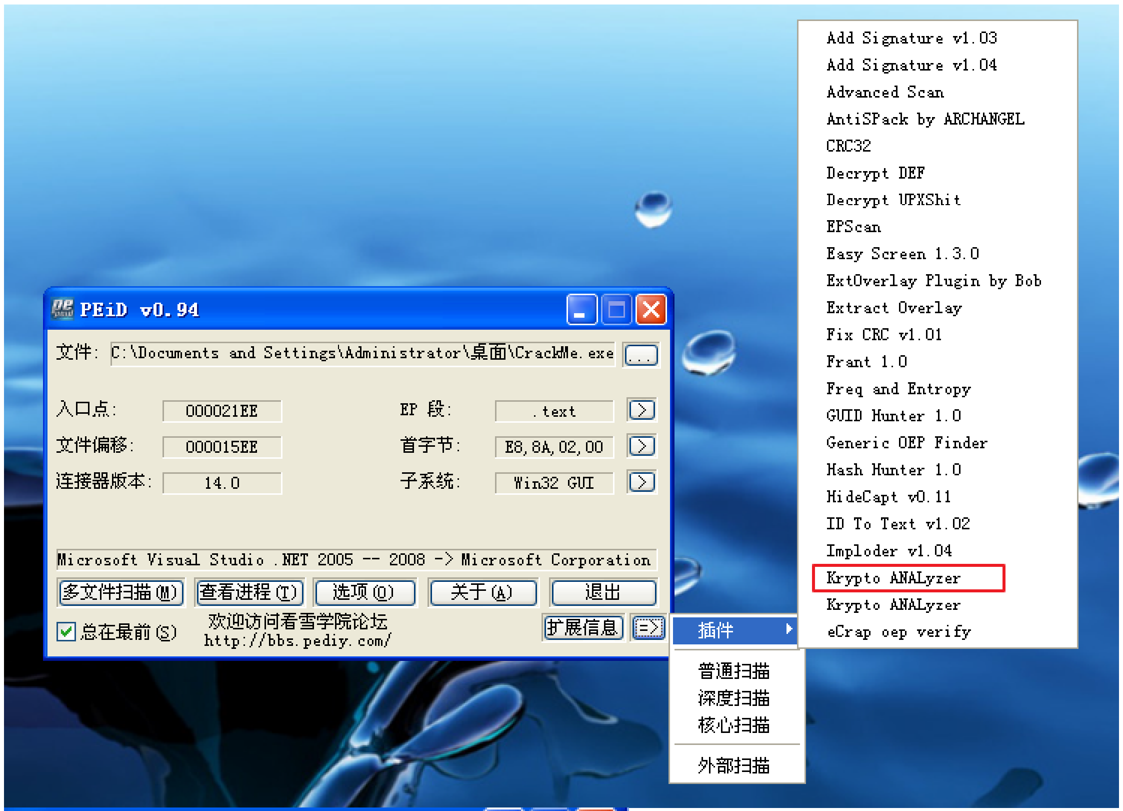
通过GetDlgItemText得到输入，并对其字数进行比较，锁定关键函数

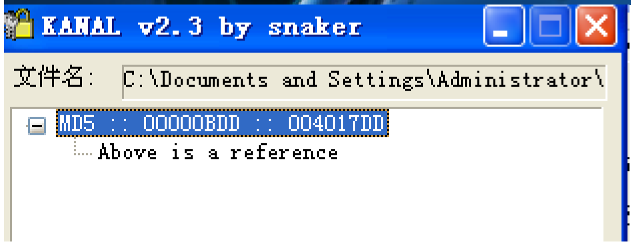


```
.text:00401474                 push    200h            ; cchMax
.text:00401479                 lea     eax, [ebp+String]
.text:0040147F                 push    eax             ; lpString
.text:00401480                 push    3E8h            ; nIDDlgItem
.text:00401485                 push    esi             ; hDlg
.text:00401486                 call    ds:GetDlgItemTextA
.text:0040148C                 cmp     eax, 30         ; Get Input and Compare input count
.text:0040148F                 jnz     short loc_4014B6
.text:00401491                 mov     edx, eax
.text:00401493                 lea     ecx, [ebp+String]
.text:00401499                 call    sub_401000
.text:0040149E                 cmp     eax, 1
.text:004014A1                 jnz     short loc_4014B6
.text:004014A3                 push    0               ; uType
.text:004014A5                 push    offset Caption  ; "information"
.text:004014AA                 push    offset Text     ; "注册成功"
.text:004014AF                 push    esi             ; hWnd
.text:004014B0                 call    ds:MessageBoxA
.text:004014B6
.text:004014B6 loc_4014B6:                             ; CODE XREF:
.text:004014B6                                         ; DialogFunc+52↑j ...
.text:004014B6                 mov     ecx, [ebp+var_4]
.text:004014B9                 xor     eax, eax
.text:004014BB                 xor     ecx, ebp
.text:004014BD                 pop     esi
```

```
; CHAR Text[]
Text            db '注册成功',0
```

查看关键函数，发现里边的代码很混乱，利用PEID和exeinfo查壳，发现并没有检测到壳，顺便利用PEID工具检测一下加密算法，发现MD5算法



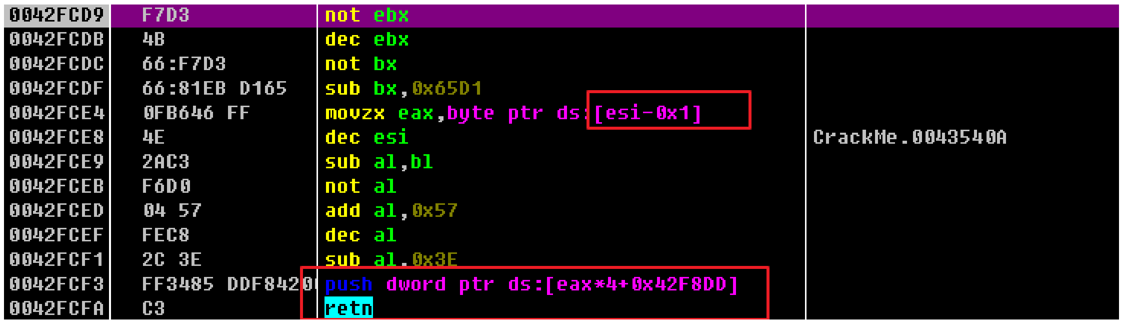

KANAL v2.3 by snaker
文件名: C:\Documents and Settings\Administrator\
MD5 :: 00000BDD :: 004017DD
Above is a reference

搜索字符串并不能发现其他有用的东西

所以我们直接进行动态调试

输入 1234567890三次，得到30位输入进行调试关键函数

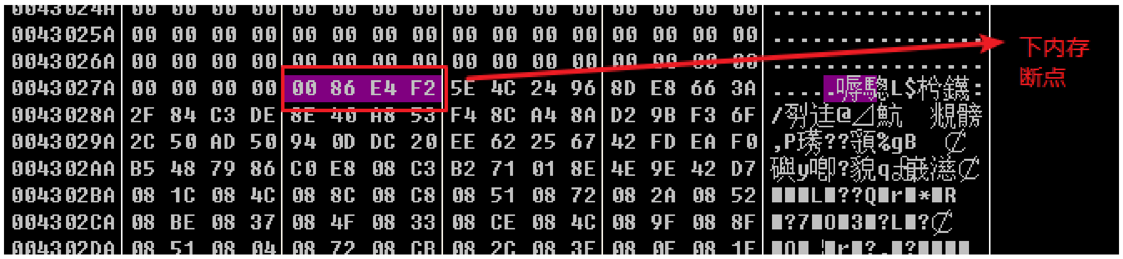
动态调试发现该过程存在循环



```
0042FCD9   F7D3              not ebx
0042FCDB   4B                dec ebx
0042FCDC   66:F7D3           not bx
0042FCDF   66:81EB D165      sub bx,0x65D1
0042FCE4   0FB646 FF         movzx eax,byte ptr ds:[esi-0x1]              CrackMe.0043540A
0042FCE8   4E                dec esi
0042FCE9   2AC3              sub al,bl
0042FCEB   F6D0              not al
0042FCED   04 57             add al,0x57
0042FCEF   FEC8              dec al
0042FCF1   2C 3E             sub al,0x3E
0042FCF3   FF3485 DDF84200   push dword ptr ds:[eax*4+0x42F8DD]
0042FCFA   C3                retn
```

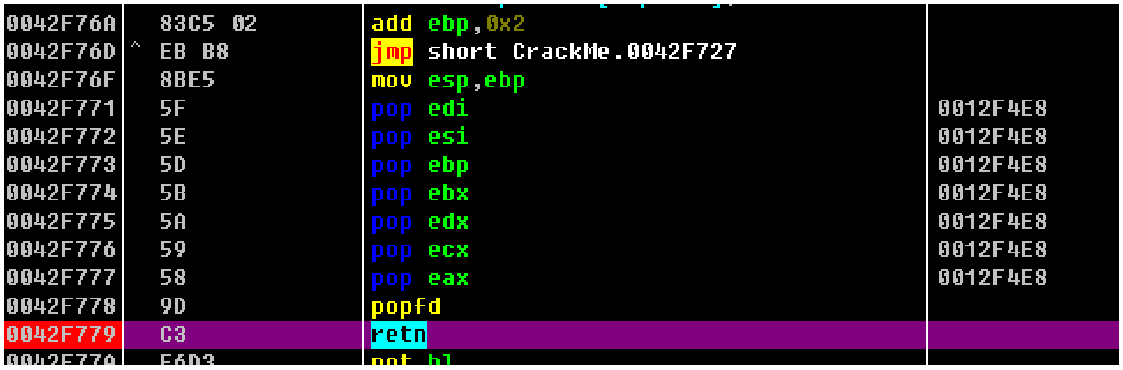
其实该过程为虚拟机，即VMP，上图esi为handler，

直接查看esi全部handle，在最后几个下内存断点



```
0043025A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0043026A  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0043027A  00 00 00 00 00 86 E4 F2 5E 4C 24 96 8D E8 66 3A  .....嗕騳L$杩鑛:
0043028A  2F 84 C3 DE 8E 40 A8 53 F4 8C A4 8A D2 9B F3 6F  /刭迬@⊿魧   觐鬍
0043029A  2C 50 AD 50 94 0D DC 20 EE 62 25 67 42 FD EA F0  ,P璓??躓%gB  
004302AA  B5 48 79 86 C0 E8 08 C3 B2 71 01 8E 4E 9E 42 D7  礖y唧?觊q嶯濨
004302BA  08 1C 08 4C 08 8C 08 C8 08 51 08 72 08 2A 08 52  ...L.??Q.r.*.R
004302CA  08 BE 08 37 08 4F 08 33 08 CE 08 4C 08 9F 08 8F  .?7.O.3.?L.?
```

注意这个操作，出虚拟机的操作，我们直接下断点



```
0042F76A    83C5 02      add ebp,0x2
0042F76D ^  EB B8        jmp short CrackMe.0042F727
0042F76F    8BE5         mov esp,ebp
0042F771    5F           pop edi                          0012F4E8
0042F772    5E           pop esi                          0012F4E8
0042F773    5D           pop ebp                          0012F4E8
0042F774    5B           pop ebx                          0012F4E8
0042F775    5A           pop edx                          0012F4E8
0042F776    59           pop ecx                          0012F4E8
0042F777    58           pop eax                          0012F4E8
0042F778    9D           popfd
0042F779    C3           retn
0042F77A    F6D3         not bl
```

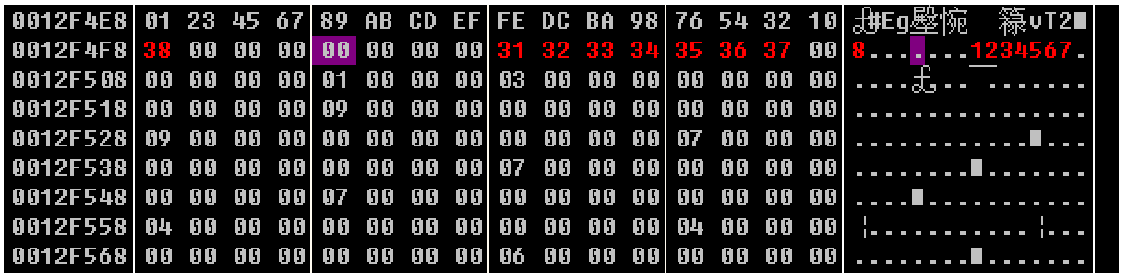
> 笔者由于个人水平有限，还是不能说清楚关于虚拟机的种种操作，所以这里只是在关键位置提一下，只能说关联代码前后凭感觉去找出虚拟机的操作，并且作者在关键代码并没有VM，如果作者所有代码都进行VM操作，可能以笔者水平，并不能搞完这个题

出虚拟机后

004014F0

发现我们之前用插件得到的MD5， 进入该函数发现第一次作比较把我们输入的前七位拷贝之后ret，不过很可能之后依旧会进入该函数，并很可能进行MD5操作

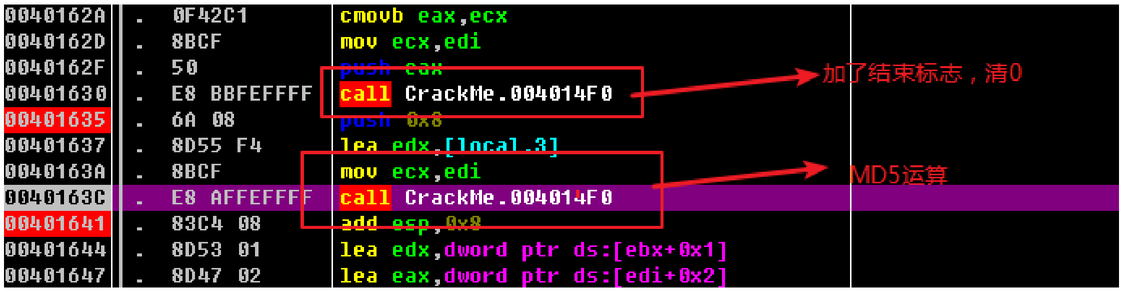
ret后发现再次进入虚拟机，我们直接运行到虚拟机出口



```
0012F4E8  01 23 45 67  89 AB CD EF  FE DC BA 98  76 54 32 10  #Eg壂惋  簶vT2
0012F4F8  38 00 00 00  00 00 00 00  31 32 33 34  35 36 37 00  8.......1234567.
0012F508  00 00 00 00  01 00 00 00  03 00 00 00  00 00 00 00  ....£...........
0012F518  00 00 00 00  09 00 00 00  00 00 00 00  00 00 00 00  ................
0012F528  09 00 00 00  00 00 00 00  00 00 00 00  07 00 00 00  ................
0012F538  00 00 00 00  00 00 00 00  07 00 00 00  00 00 00 00  ................
0012F548  00 00 00 00  07 00 00 00  00 00 00 00  00 00 00 00  ................
0012F558  04 00 00 00  00 00 00 00  00 00 00 00  04 00 00 00  ¦...........¦...
0012F568  00 00 00 00  00 00 00 00  06 00 00 00  00 00 00 00  ................
```

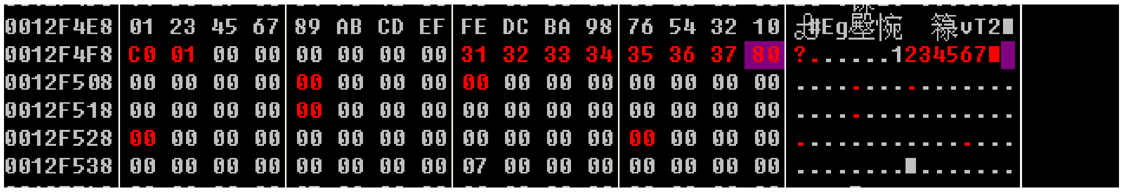
004015c0

该函数的两个函数主要操作



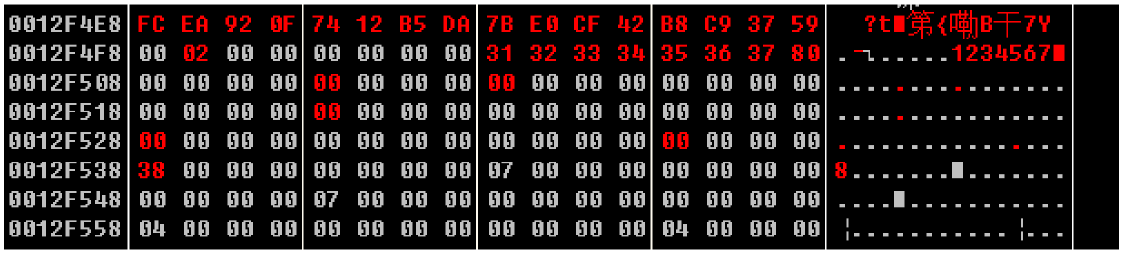
```
0040162A  .  0F42C1       cmovb eax,ecx
0040162D  .  8BCF         mov ecx,edi
0040162F  .  50           push eax
00401630  .  E8 BBFEFFFF  call CrackMe.004014F0
00401635  .  6A 08        push 0x8
00401637  .  8D55 F4      lea edx,[local.3]
0040163A  .  8BCF         mov ecx,edi
0040163C  .  E8 AFFEFFFF  call CrackMe.004014F0
00401641  .  83C4 08      add esp,0x8
00401644  .  8D53 01      lea edx,dword ptr ds:[ebx+0x1]
00401647  .  8D47 02      lea eax,dword ptr ds:[edi+0x2]
```

第一次进入004014F0加了结束标志符，并把后边清0



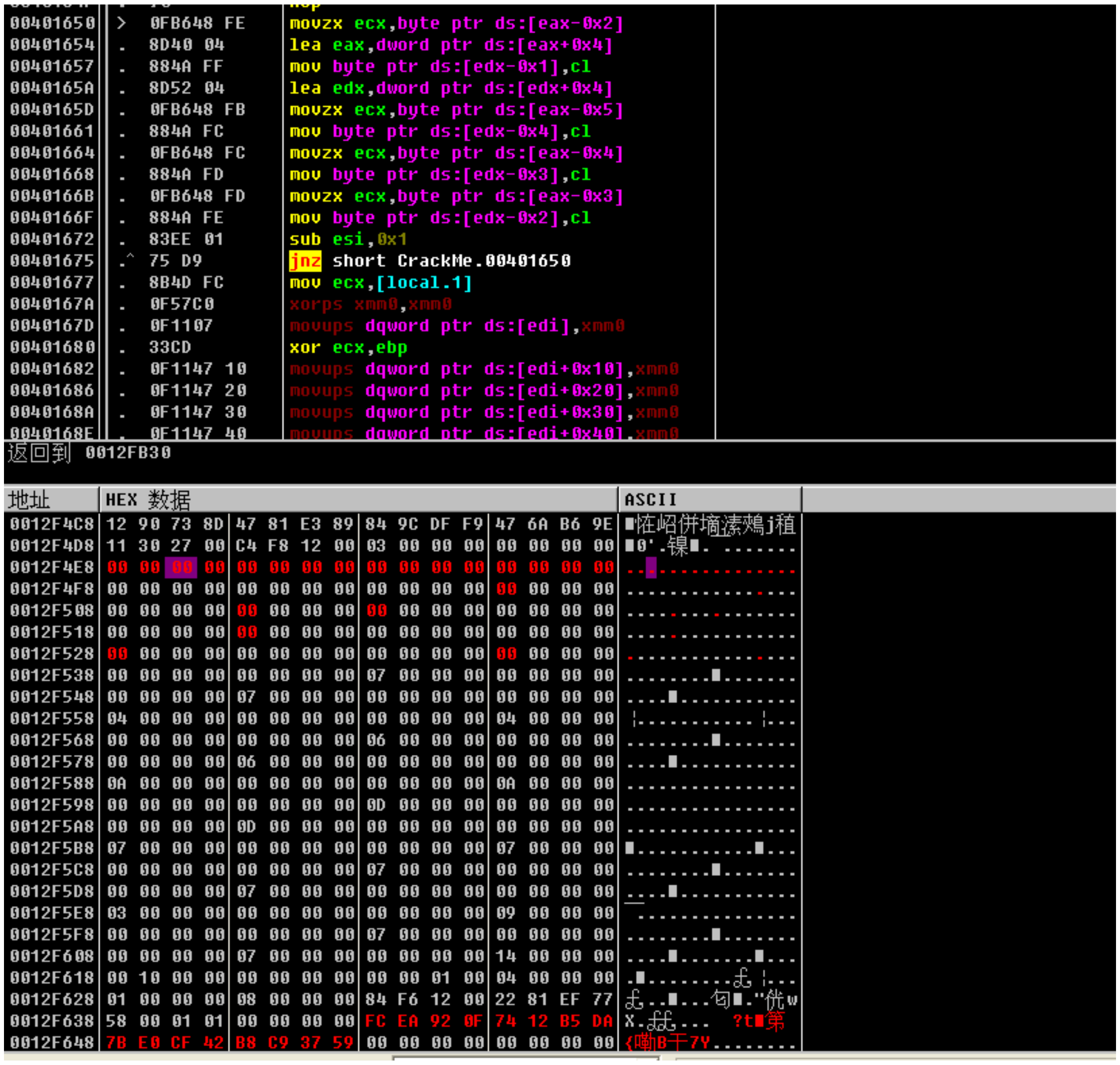
```
0012F4E8  01 23 45 67  89 AB CD EF  FE DC BA 98  76 54 32 10  #Eg壂惋  簶vT2
0012F4F8  C0 01 00 00  00 00 00 00  31 32 33 34  35 36 37 80  ?.......1234567
0012F508  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
0012F518  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
0012F528  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
0012F538  00 00 00 00  00 00 00 00  07 00 00 00  00 00 00 00  ................
```

第二次,MD5操作



```
0012F4E8 FC EA 92 0F 74 12 B5 DA 7B E0 CF 42 B8 C9 37 59  ?t■第{嘞B干7Y
0012F4F8 00 02 00 00 00 00 00 00 31 32 33 34 35 36 37 80  .┐......1234567■
0012F508 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012F518 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012F528 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0012F538 38 00 00 00 00 00 00 00 07 00 00 00 00 00 00 00  8.......■.......
0012F548 00 00 00 00 07 00 00 00 00 00 00 00 00 00 00 00  ....■...........
0012F558 04 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00  ¦...........¦...
```

拷贝MD5值并把内存清0

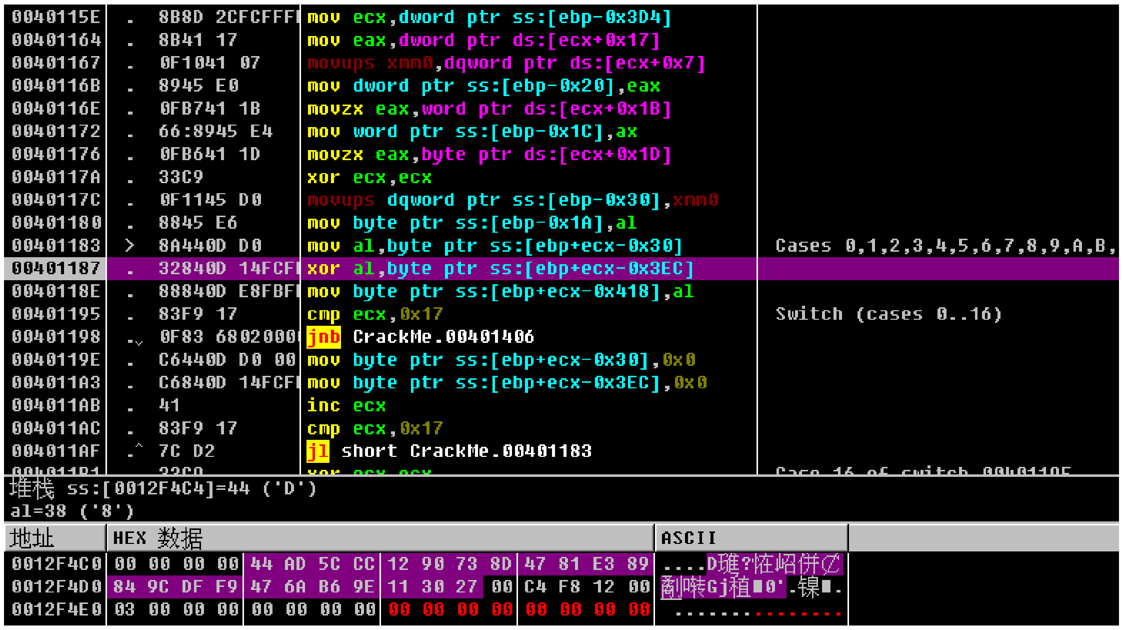


```
00401650  >  0FB648 FE      movzx ecx,byte ptr ds:[eax-0x2]
00401654  .  8D40 04        lea eax,dword ptr ds:[eax+0x4]
00401657  .  884A FF        mov byte ptr ds:[edx-0x1],cl
0040165A  .  8D52 04        lea edx,dword ptr ds:[edx+0x4]
0040165D  .  0FB648 FB      movzx ecx,byte ptr ds:[eax-0x5]
00401661  .  884A FC        mov byte ptr ds:[edx-0x4],cl
00401664  .  0FB648 FC      movzx ecx,byte ptr ds:[eax-0x4]
00401668  .  884A FD        mov byte ptr ds:[edx-0x3],cl
0040166B  .  0FB648 FD      movzx ecx,byte ptr ds:[eax-0x3]
0040166F  .  884A FE        mov byte ptr ds:[edx-0x2],cl
00401672  .  83EE 01        sub esi,0x1
00401675  .^ 75 D9          jnz short CrackMe.00401650
00401677  .  8B4D FC        mov ecx,[local.1]
0040167A  .  0F57C0         xorps xmm0,xmm0
0040167D  .  0F1107         movups dqword ptr ds:[edi],xmm0
00401680  .  33CD           xor ecx,ebp
00401682  .  0F1147 10      movups dqword ptr ds:[edi+0x10],xmm0
00401686  .  0F1147 20      movups dqword ptr ds:[edi+0x20],xmm0
0040168A  .  0F1147 30      movups dqword ptr ds:[edi+0x30],xmm0
0040168E  .  0F1147 40      movups dqword ptr ds:[edi+0x40],xmm0
返回到 0012FB30

地址      HEX 数据                                          ASCII
0012F4C8 12 90 73 8D 47 81 E3 89 84 9C DF F9 47 6A B6 9E ■恡峈併墒潆鵊j稙
0012F4D8 11 30 27 00 C4 F8 12 00 03 00 00 00 00 00 00 00 ■0'.镍■.........
0012F4E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0012F4F8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0012F508 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0012F518 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0012F528 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
0012F538 00 00 00 00 00 00 00 00 07 00 00 00 00 00 00 00 ........■.......
0012F548 00 00 00 00 07 00 00 00 00 00 00 00 00 00 00 00 ....■...........
0012F558 04 00 00 00 00 00 00 00 00 00 00 00 04 00 00 00 ¦...........¦...
0012F568 00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 ........■.......
0012F578 00 00 00 00 06 00 00 00 00 00 00 00 00 00 00 00 ....■...........
0012F588 0A 00 00 00 00 00 00 00 00 00 00 00 0A 00 00 00 ................
0012F598 00 00 00 00 00 00 00 00 0D 00 00 00 00 00 00 00 ................
0012F5A8 00 00 00 00 0D 00 00 00 00 00 00 00 00 00 00 00 ................
0012F5B8 07 00 00 00 00 00 00 00 00 00 00 00 07 00 00 00 ■...........■...
0012F5C8 00 00 00 00 00 00 00 00 07 00 00 00 00 00 00 00 ........■.......
0012F5D8 00 00 00 00 07 00 00 00 00 00 00 00 00 00 00 00 ....■...........
0012F5E8 03 00 00 00 00 00 00 00 00 00 00 00 09 00 00 00 ................
0012F5F8 00 00 00 00 00 00 00 00 07 00 00 00 00 00 00 00 ........■.......
0012F608 00 00 00 00 07 00 00 00 00 00 00 00 14 00 00 00 ....■.......■...
0012F618 00 10 00 00 00 00 00 00 00 00 01 00 04 00 00 00 .■........£ ¦...
0012F628 01 00 00 00 08 00 00 00 84 F6 12 00 22 81 EF 77 £...■...匂■."侁w
0012F638 58 00 01 01 00 00 00 00 FC EA 92 0F 74 12 B5 DA X.££....?t■第
0012F648 7B E0 CF 42 B8 C9 37 59 00 00 00 00 00 00 00 00 {嘞B干7Y........
```

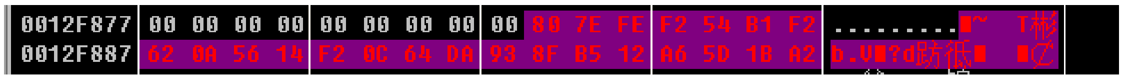
00401187

获取之后的23位，与指定数进行xor



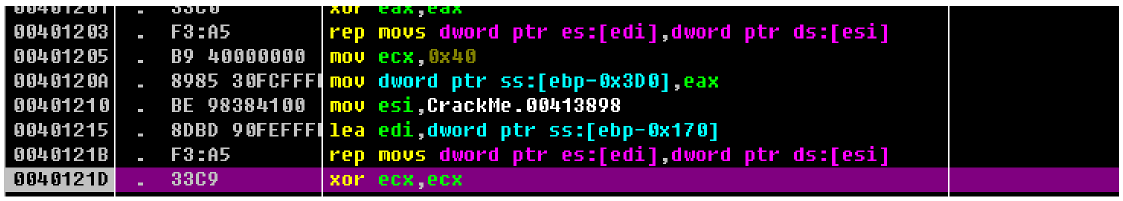
```
0040115E  .  8B8D 2CFCFFFF mov ecx,dword ptr ss:[ebp-0x3D4]
00401164  .  8B41 17       mov eax,dword ptr ds:[ecx+0x17]
00401167  .  0F1041 07     movups xmm0,dqword ptr ds:[ecx+0x7]
0040116B  .  8945 E0       mov dword ptr ss:[ebp-0x20],eax
0040116E  .  0FB741 1B     movzx eax,word ptr ds:[ecx+0x1B]
00401172  .  66:8945 E4    mov word ptr ss:[ebp-0x1C],ax
00401176  .  0FB641 1D     movzx eax,byte ptr ds:[ecx+0x1D]
0040117A  .  33C9          xor ecx,ecx
0040117C  .  0F1145 D0     movups dqword ptr ss:[ebp-0x30],xmm0
00401180  .  8845 E6       mov byte ptr ss:[ebp-0x1A],al
00401183  >  8A440D D0     mov al,byte ptr ss:[ebp+ecx-0x30]      Cases 0,1,2,3,4,5,6,7,8,9,A,B,
00401187  .  32840D 14FCFF xor al,byte ptr ss:[ebp+ecx-0x3EC]
0040118E  .  88840D E8FBFF mov byte ptr ss:[ebp+ecx-0x418],al
00401195  .  83F9 17       cmp ecx,0x17                           Switch (cases 0..16)
00401198  .v 0F83 6802000( jnb CrackMe.00401406
0040119E  .  C6440D D0 00  mov byte ptr ss:[ebp+ecx-0x30],0x0
004011A3  .  C6840D 14FCFF mov byte ptr ss:[ebp+ecx-0x3EC],0x0
004011AB  .  41            inc ecx
004011AC  .  83F9 17       cmp ecx,0x17
004011AF  .^ 7C D2         jl short CrackMe.00401183
堆栈 ss:[0012F4C4]=44 ('D')
al=38 ('8')
地址      HEX 数据                                         ASCII
0012F4C0  00 00 00 00 44 AD 5C CC 12 90 73 8D 47 81 E3 89  ....D璀?恠岎併
0012F4D0  84 9C DF F9 47 6A B6 9E 11 30 27 00 C4 F8 12 00  劀唼Gj稙■0'.镎■.
0012F4E0  03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

之后该值会与之后的MD5值进行XOR，注意MD5值16位，重复循环，xor，知道23位得到值



```
0012F877  00 00 00 00 00 00 00 00 00 80 7E FE F2 54 B1 F2  .........■~  T彬
0012F887  62 0A 56 14 F2 0C 64 DA 93 8F B5 12 A6 5D 1B A2  b.V■?d跡祇■  ■∅
```

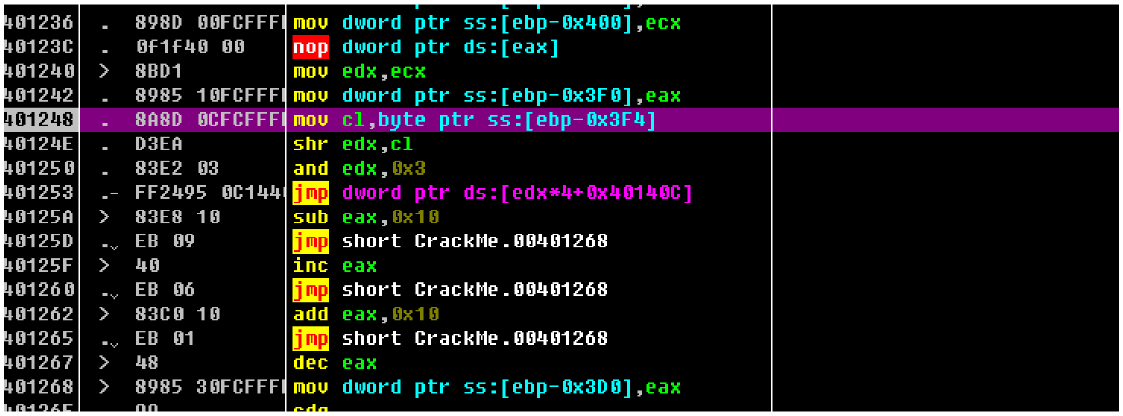
这两个赋值操作是对迷宫的初始化，后期会用到的



```
00401203  .  F3:A5         rep movs dword ptr es:[edi],dword ptr ds:[esi]
00401205  .  B9 40000000   mov ecx,0x40
0040120A  .  8985 30FCFFFF mov dword ptr ss:[ebp-0x3D0],eax
00401210  .  BE 98384100   mov esi,CrackMe.00413898
00401215  .  8DBD 90FEFFFF lea edi,dword ptr ss:[ebp-0x170]
0040121B  .  F3:A5         rep movs dword ptr es:[edi],dword ptr ds:[esi]
0040121D  .  33C9          xor ecx,ecx
```

，而且此时的数值才是正确的，因为之后经过VM后，有些数值会发生变化，导致后边计算错误

之后就是迷宫走法的核心计算，下边是如何计算每一步及其规则



```
401236  .  898D 00FCFFFF mov dword ptr ss:[ebp-0x400],ecx
40123C  .  0F1F40 00     nop dword ptr ds:[eax]
401240  >  8BD1          mov edx,ecx
401242  .  8985 10FCFFFF mov dword ptr ss:[ebp-0x3F0],eax
401248  .  8A8D 0CFCFFFF mov cl,byte ptr ss:[ebp-0x3F4]
40124E  .  D3EA          shr edx,cl
401250  .  83E2 03       and edx,0x3
401253  .- FF2495 0C1440 jmp dword ptr ds:[edx*4+0x40140C]
40125A  >  83E8 10       sub eax,0x10
40125D  .v EB 09         jmp short CrackMe.00401268
40125F  >  40            inc eax
401260  .v EB 06         jmp short CrackMe.00401268
401262  >  83C0 10       add eax,0x10
401265  .v EB 01         jmp short CrackMe.00401268
401267  >  48            dec eax
401268  >  8985 30FCFFFF mov dword ptr ss:[ebp-0x3D0],eax
```

详细说一下这里吧

首先00401248这里cl被赋值为6，4，2，0，

0040124E这个edx取我们输入后处理的值，and 3，得到其值

然后通过00401253计算跳到哪里，这里可以得到

0 ---> 向上走

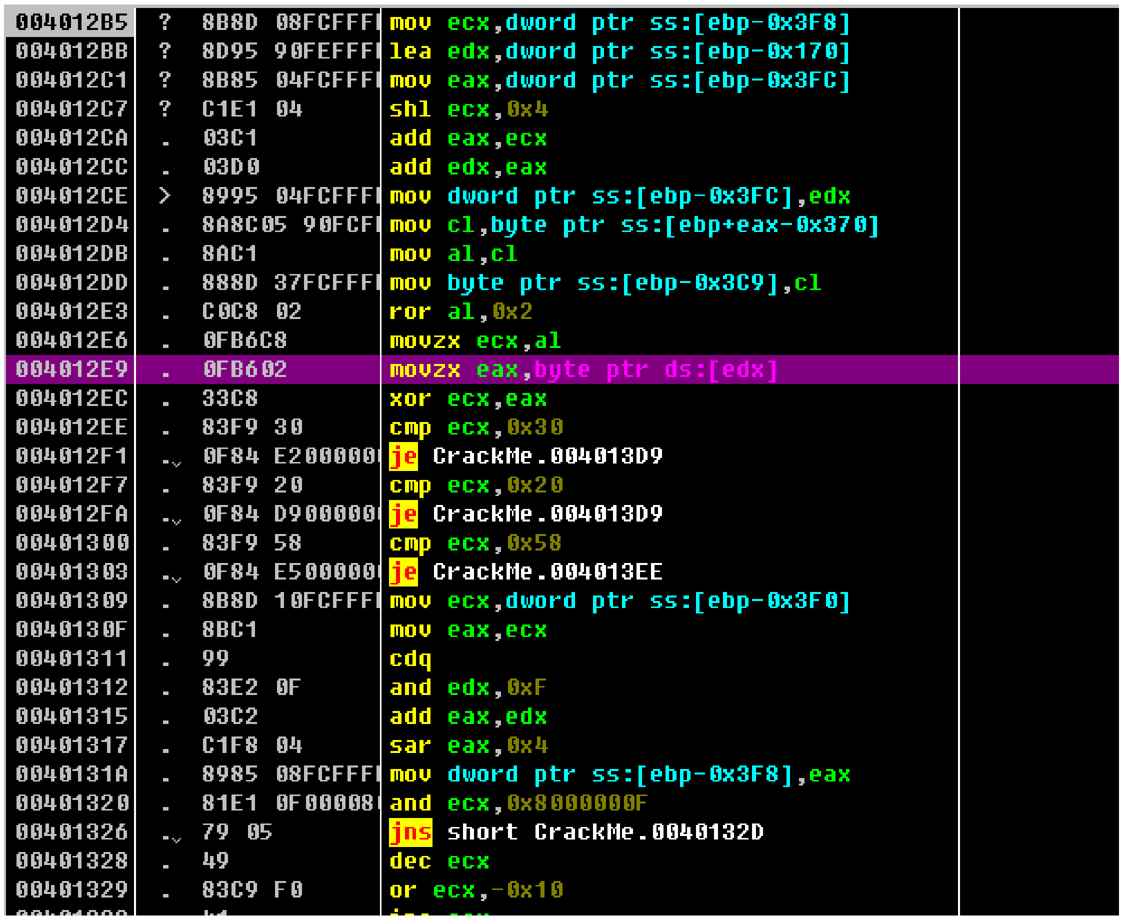
1 ---> 向前走

2 ---> 向下走

3 ---> 向后走

所以一个字节可以走四步

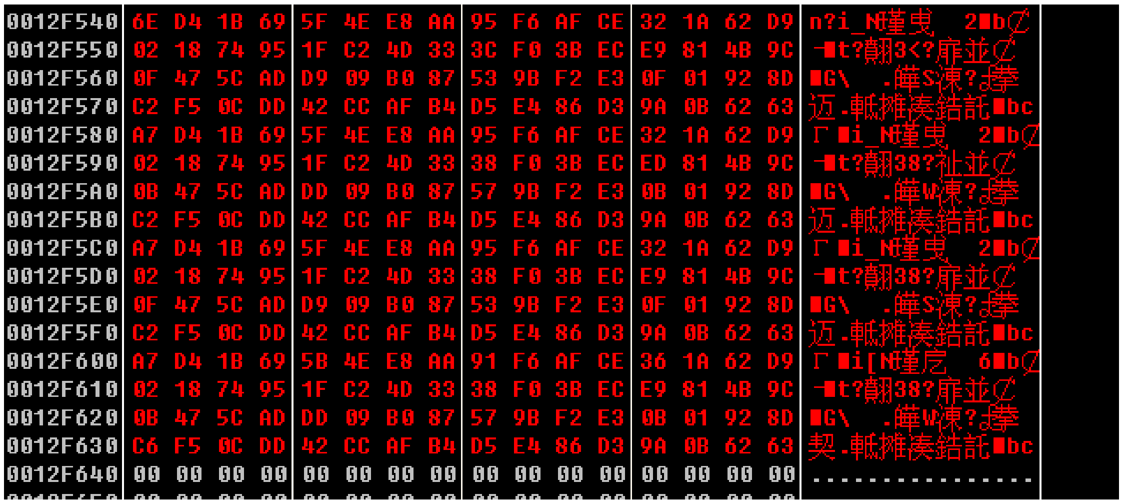
然后接下来是迷宫的计算



```
004012B5  ?  8B8D 08FCFFFF  mov ecx,dword ptr ss:[ebp-0x3F8]
004012BB  ?  8D95 90FEFFFF  lea edx,dword ptr ss:[ebp-0x170]
004012C1  ?  8B85 04FCFFFF  mov eax,dword ptr ss:[ebp-0x3FC]
004012C7  ?  C1E1 04        shl ecx,0x4
004012CA  .  03C1           add eax,ecx
004012CC  .  03D0           add edx,eax
004012CE  >  8995 04FCFFFF  mov dword ptr ss:[ebp-0x3FC],edx
004012D4  .  8A8C05 90FCFI  mov cl,byte ptr ss:[ebp+eax-0x370]
004012DB  .  8AC1           mov al,cl
004012DD  .  888D 37FCFFFF  mov byte ptr ss:[ebp-0x3C9],cl
004012E3  .  C0C8 02        ror al,0x2
004012E6  .  0FB6C8         movzx ecx,al
004012E9  .  0FB602         movzx eax,byte ptr ds:[edx]
004012EC  .  33C8           xor ecx,eax
004012EE  .  83F9 30        cmp ecx,0x30
004012F1  .v 0F84 E2000000  je CrackMe.004013D9
004012F7  .  83F9 20        cmp ecx,0x20
004012FA  .v 0F84 D9000000  je CrackMe.004013D9
00401300  .  83F9 58        cmp ecx,0x58
00401303  .v 0F84 E5000000  je CrackMe.004013EE
00401309  .  8B8D 10FCFFFF  mov ecx,dword ptr ss:[ebp-0x3F0]
0040130F  .  8BC1           mov eax,ecx
00401311  .  99             cdq
00401312  .  83E2 0F        and edx,0xF
00401315  .  03C2           add eax,edx
00401317  .  C1F8 04        sar eax,0x4
0040131A  .  8985 08FCFFFF  mov dword ptr ss:[ebp-0x3F8],eax
00401320  .  81E1 0F000080  and ecx,0x8000000F
00401326  .v 79 05          jns short CrackMe.0040132D
00401328  .  49             dec ecx
00401329  .  83C9 F0        or ecx,-0x10
```

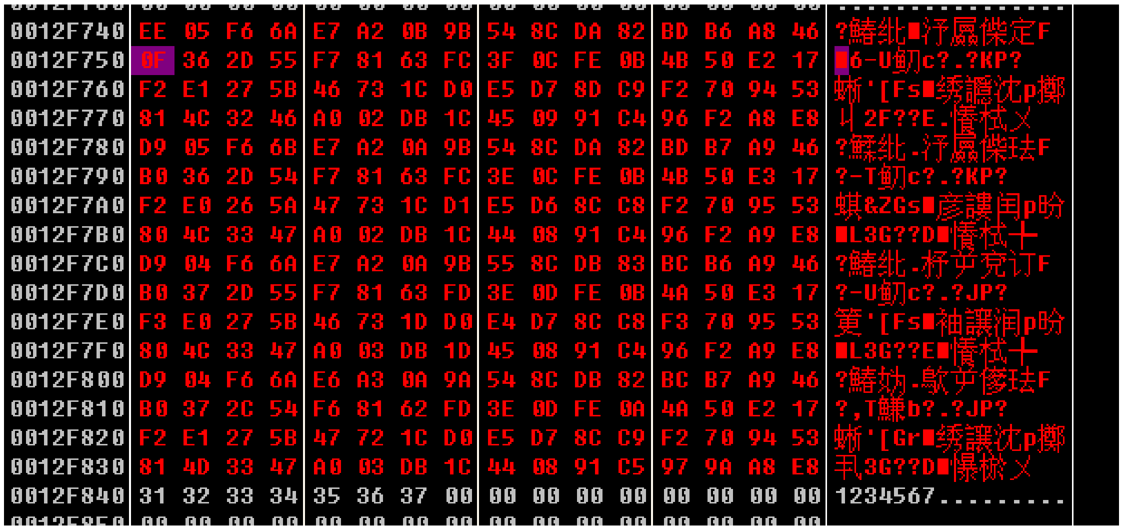
我们首先得到

part 1



```
0012F540  6E D4 1B 69  5F 4E E8 AA  95 F6 AF CE  32 1A 62 D9  n?i_N槿曳  2■b⊄
0012F550  02 18 74 95  1F C2 4D 33  3C F0 3B EC  E9 81 4B 9C  ┤■t?翽3<?扉並⊄
0012F560  0F 47 5C AD  D9 09 B0 87  53 9B F2 E3  0F 01 92 8D  ■G\  .皣S凍?₫摮
0012F570  C2 F5 0C DD  42 CC AF B4  D5 E4 86 D3  9A 0B 62 63  迈.軙摊凑鍣託■bc
0012F580  A7 D4 1B 69  5F 4E E8 AA  95 F6 AF CE  32 1A 62 D9  Γ■i_N槿曳  2■b⊄
0012F590  02 18 74 95  1F C2 4D 33  38 F0 3B EC  ED 81 4B 9C  ┤■t?翽38?祉並⊄
0012F5A0  0B 47 5C AD  DD 09 B0 87  57 9B F2 E3  0B 01 92 8D  ■G\  .皣W凍?₫摮
0012F5B0  C2 F5 0C DD  42 CC AF B4  D5 E4 86 D3  9A 0B 62 63  迈.軙摊凑鍣託■bc
0012F5C0  A7 D4 1B 69  5F 4E E8 AA  95 F6 AF CE  32 1A 62 D9  Γ■i_N槿曳  2■b⊄
0012F5D0  02 18 74 95  1F C2 4D 33  38 F0 3B EC  E9 81 4B 9C  ┤■t?翽38?扉並⊄
0012F5E0  0F 47 5C AD  D9 09 B0 87  53 9B F2 E3  0F 01 92 8D  ■G\  .皣S凍?₫摮
0012F5F0  C2 F5 0C DD  42 CC AF B4  D5 E4 86 D3  9A 0B 62 63  迈.軙摊凑鍣託■bc
0012F600  A7 D4 1B 69  5B 4E E8 AA  91 F6 AF CE  36 1A 62 D9  Γ■i[N槿戹  6■b⊄
0012F610  02 18 74 95  1F C2 4D 33  38 F0 3B EC  E9 81 4B 9C  ┤■t?翽38?扉並⊄
0012F620  0B 47 5C AD  DD 09 B0 87  57 9B F2 E3  0B 01 92 8D  ■G\  .皣W凍?₫摮
0012F630  C6 F5 0C DD  42 CC AF B4  D5 E4 86 D3  9A 0B 62 63  契.軙摊凑鍣託■bc
0012F640  00 00 00 00  00 00 00 00  00 00 00 00  00 00 00 00  ................
```

part 2



```
0012F740 EE 05 F6 6A E7 A2 0B 9B 54 8C DA 82 BD B6 A8 46 ?鳍纰■泞羼傑定F
0012F750 0F 36 2D 55 F7 81 63 FC 3F 0C FE 0B 4B 50 E2 17 ■6-U觔c?.?KP?
0012F760 F2 E1 27 5B 46 73 1C D0 E5 D7 8D C9 F2 70 94 53 蛳'[Fs■绣謕沈p擲
0012F770 81 4C 32 46 A0 02 DB 1C 45 09 91 C4 96 F2 A8 E8 ↓2F??E.憤杖乂
0012F780 D9 05 F6 6B E7 A2 0A 9B 54 8C DA 82 BD B7 A9 46 ?鲧纰.泞羼傑珐F
0012F790 B0 36 2D 54 F7 81 63 FC 3E 0C FE 0B 4B 50 E3 17 ?-T觔c?.?KP?
0012F7A0 F2 E0 26 5A 47 73 1C D1 E5 D6 8C C8 F2 70 95 53 蜞&ZGs■彦謓闰p盼
0012F7B0 80 4C 33 47 A0 02 DB 1C 44 08 91 C4 96 F2 A9 E8 ■L3G??D■憤杖十
0012F7C0 D9 04 F6 6A E7 A2 0A 9B 55 8C DB 83 BC B6 A9 46 ?鳍纰.杍屰兊订F
0012F7D0 B0 37 2D 55 F7 81 63 FD 3E 0D FE 0B 4A 50 E3 17 ?-U觔c?.?JP?
0012F7E0 F3 E0 27 5B 46 73 1D D0 E4 D7 8C C8 F3 70 95 53 笆'[Fs■袖譟润p盼
0012F7F0 80 4C 33 47 A0 03 DB 1D 45 08 91 C4 96 F2 A9 E8 ■L3G??E■憤杖十
0012F800 D9 04 F6 6A E6 A3 0A 9A 54 8C DB 82 BC B7 A9 46 ?鳍妫.歀屰傪珐F
0012F810 B0 37 2C 54 F6 81 62 FD 3E 0D FE 0A 4A 50 E2 17 ?,T鰊b?.?JP?
0012F820 F2 E1 27 5B 47 72 1C D0 E5 D7 8C C9 F2 70 94 53 蛳'[Gr■绣譟沈p擲
0012F830 81 4D 33 47 A0 03 DB 1C 44 08 91 C5 97 9A A8 E8 刊3G??D■憬榢乂
0012F840 31 32 33 34 35 36 37 00 00 00 00 00 00 00 00 00 1234567.........
```

首先是对Part 1根据上边的走法取对应值然后ror，**这条指令我们用程序模拟还是挺麻烦的**，在找到对应的part 2值，进行xor

之后我们通过爆破，发现0x58是出口

了解了具体的步骤

我们开始写程序爆破

程序后边代码