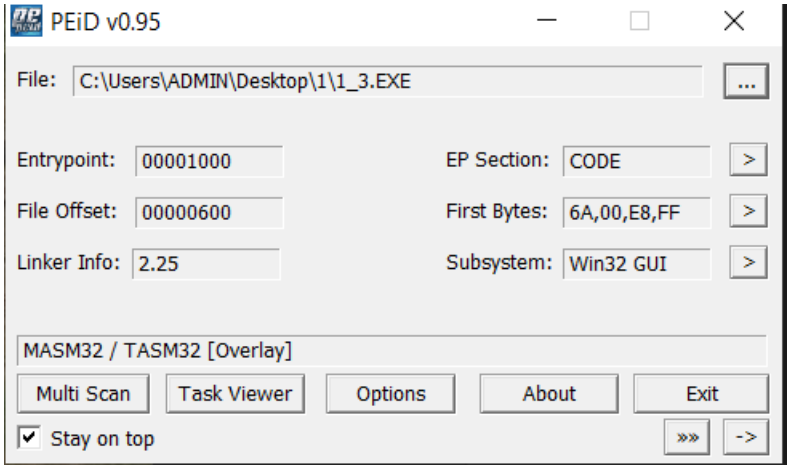
1.3

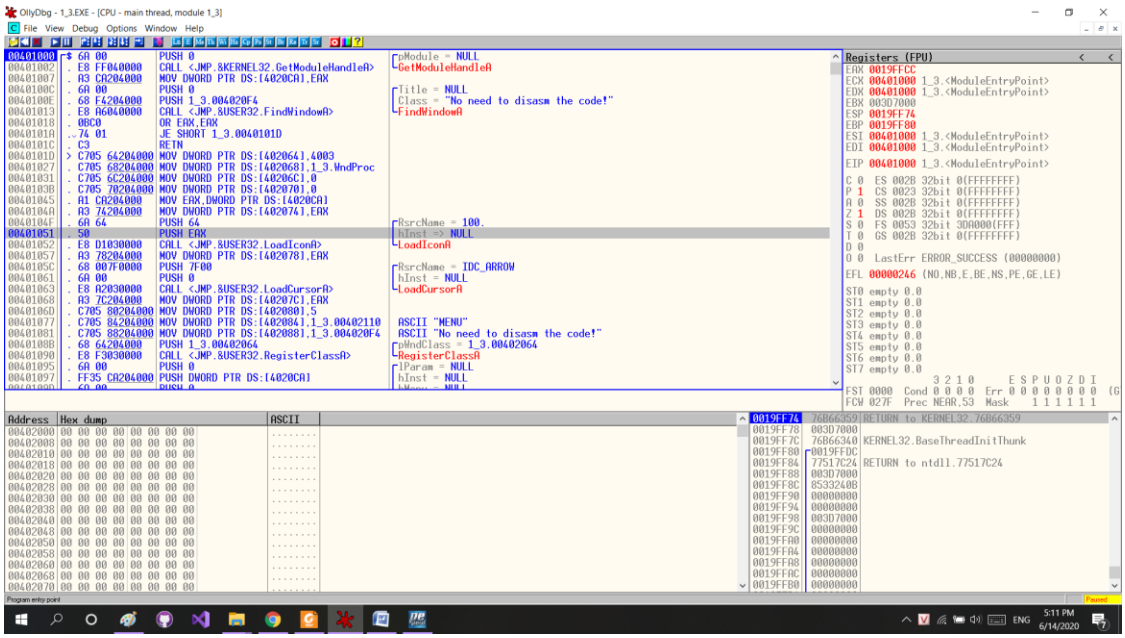
1)Đầu tiên chúng ta sẽ Scan with PEID 1.1exe để kiểm tra tệp này đã bị pack hay chưa

(Packer là một kiểu chương trình nén hoặc che dấu file thực thi (executable file). Các chương trình này ra đời bắt nguồn từ mục đích giảm kích thước của file, làm cho việc tải file nhanh hơn.)



Sau khi Scan xong ra kết quả thì ta có thể thấy kết quả là chương trình không bị pack và có thể được code bằng MASM32 hoặc TASM32.

2)Chạy OllyDbg mở 1_3.EXE



Có nhiều cách để tiếp cận như

.Thông qua việc nhập dữ liệu(Hàm GetDlgItemTextA)

.Hàm MessageBoxA

.Các từ khóa GoodBoy,BadBoy

...

Trong bài này mình chọn theo cách 1 là GetDlgIteamTextA

Sơ lược về hàm GetDlgIteamtextA

(https://docs.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getdlgitemtexta)

Syntax

C++ Copy

```
UINT GetDlgItemTextA(
  HWND  hDlg,
  int   nIDDlgItem,
  LPSTR lpString,
  int   cchMax
);
```

Parameters

`hDlg`

Type: **HWND**

A handle to the dialog box that contains the control.

`nIDDlgItem`

Type: **int**

The identifier of the control whose title or text is to be retrieved.

`lpString`

Type: **LPTSTR**

The buffer to receive the title or text.

`cchMax`

Type: **int**

The maximum length, in characters, of the string to be copied to the buffer pointed to by *lpString*. If the length of the string, including the null character, exceeds the limit, the string is truncated.

Return value

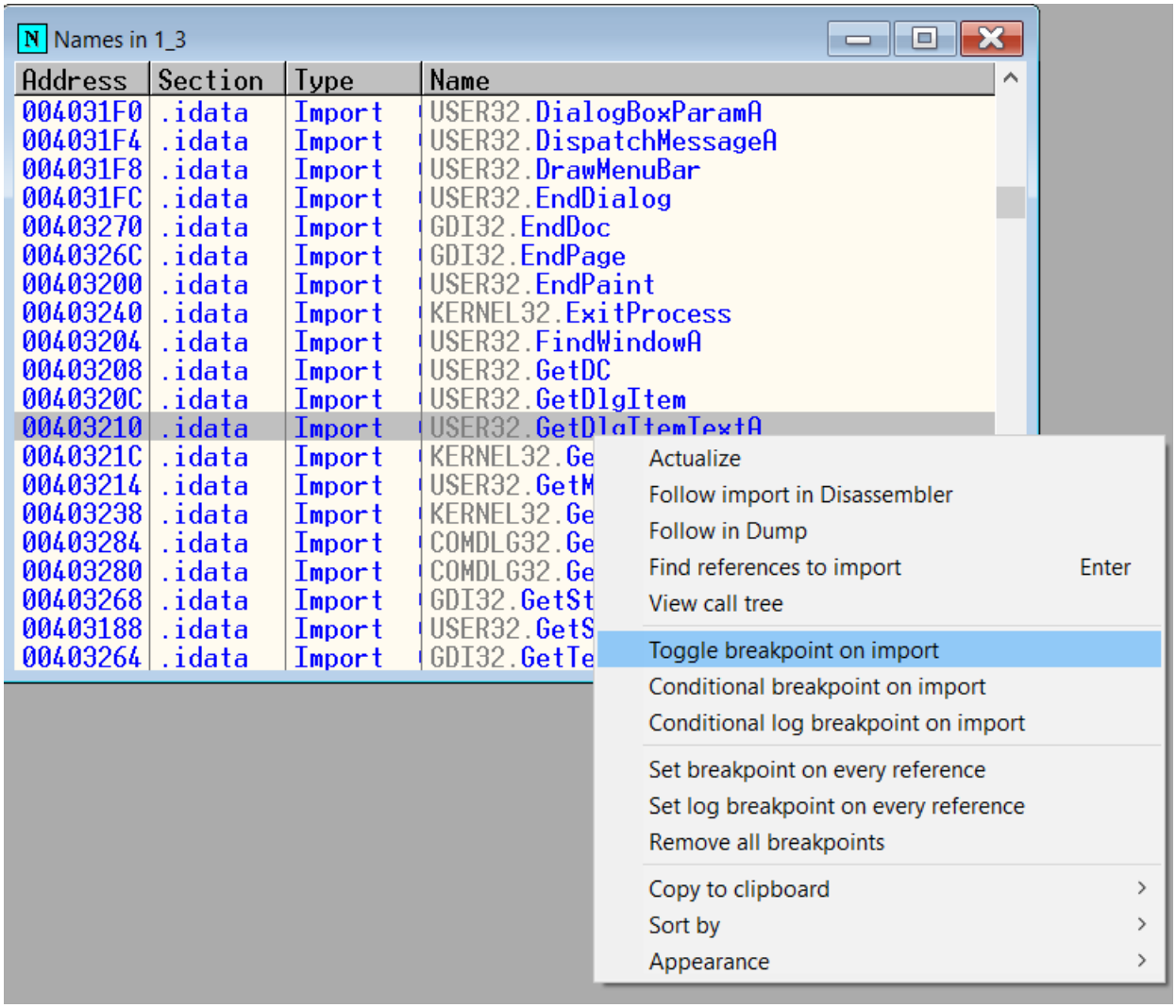
Type: **UINT**

If the function succeeds, the return value specifies the number of characters copied to the buffer, not including the terminating null character.

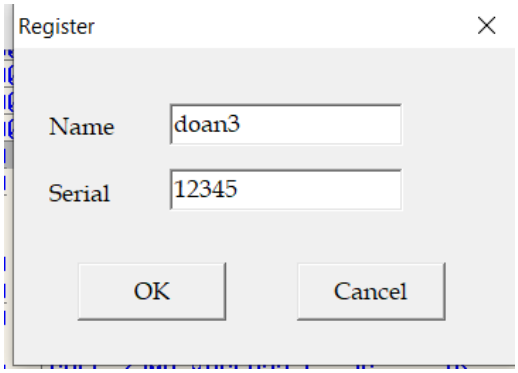
If the function fails, the return value is zero. To get extended error information, call GetLastError.

Để hiểu rõ hơn các dữ liệu khi nhập vào sẽ được lưu trữ ở đâu chúng ta đặt 1 BreakPoint vào Hàm

GetDlgItemTextA

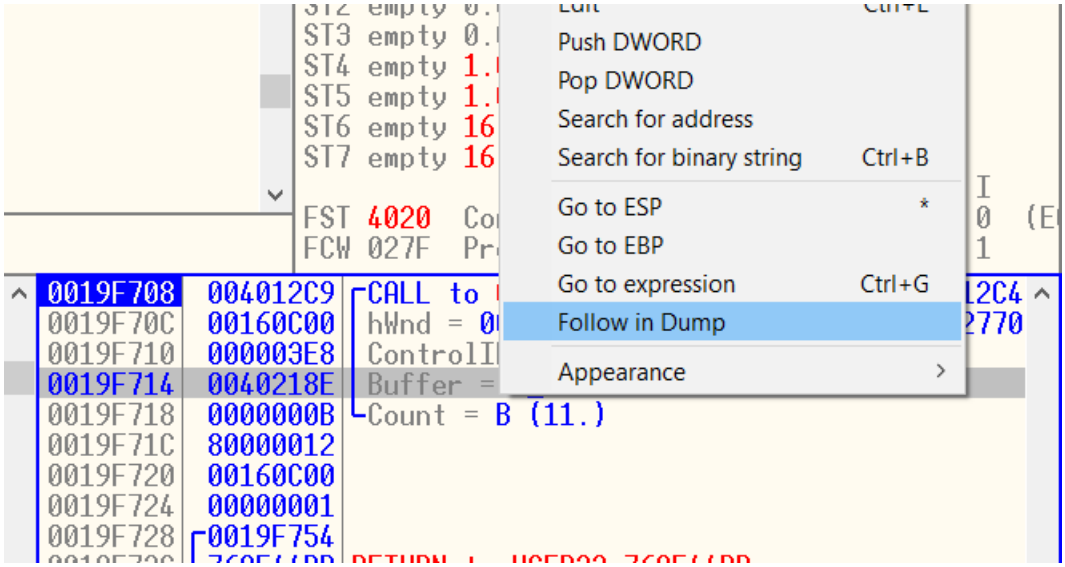


Nhấn F9 để Debug chương trình

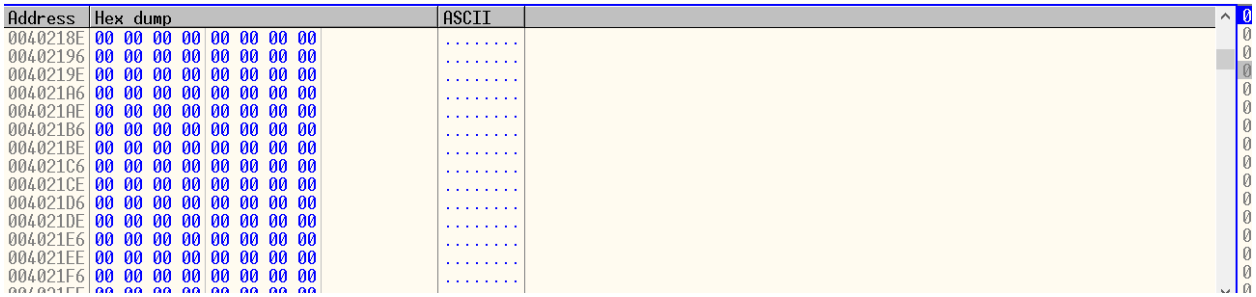


Nhập 1 bộ ngẫu nhiên để test

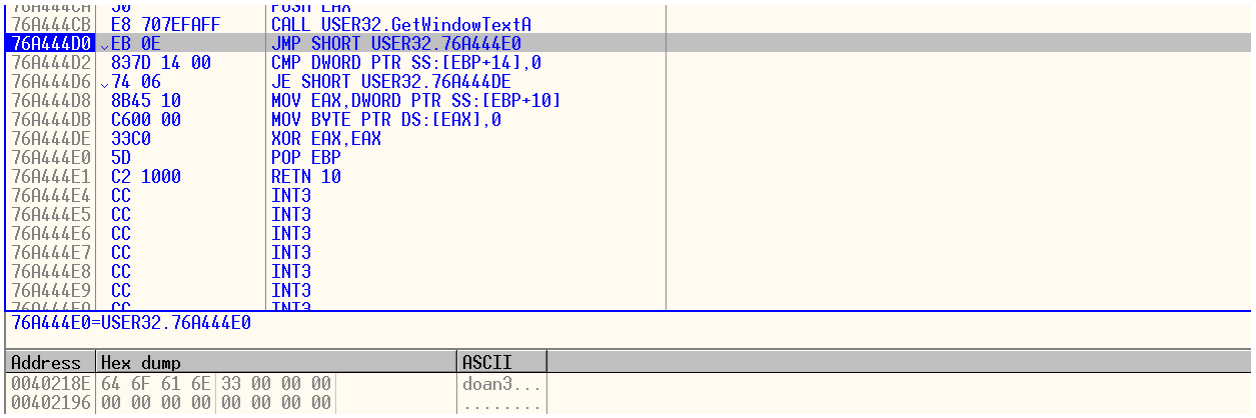
Sau khi nhấn OK chương trình sẽ dừng lại ngay hàm mình đã đặt BP.Ở dòng Count =11 cho chúng ta biết dữ liệu tối đa là 11.Suy ra số kí tự Name <11



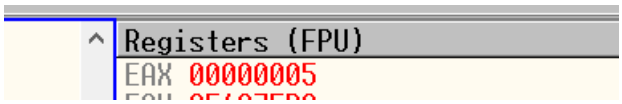
Nhấn vào Follow in Dump để xem dữ liệu



Dữ liệu chưa có gì tức là Hàm chưa được chạy.Nhấn F8 để trace qua các câu lệnh để thực thi hàm

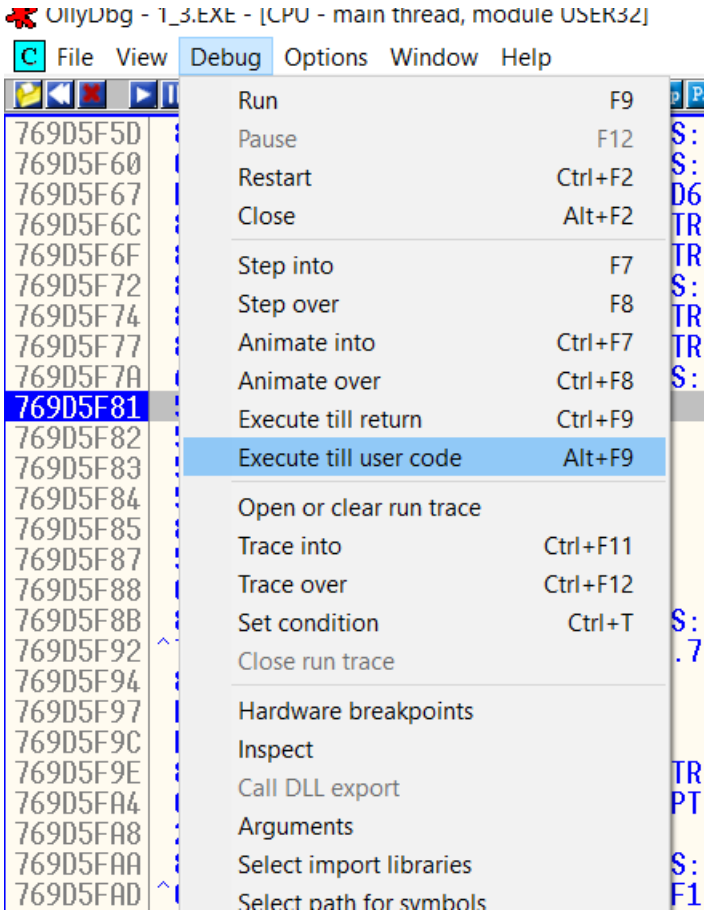


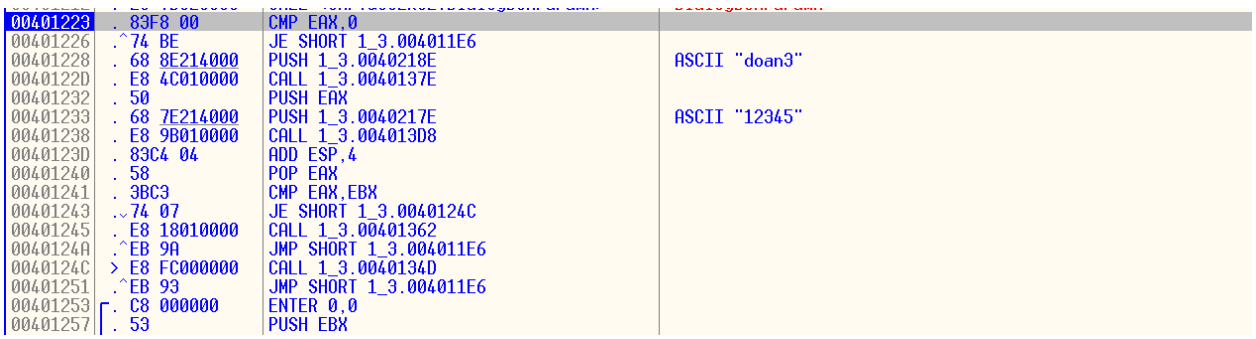
F8 đến câu lệnh này thì ta thấy Name đã được thêm vào .Chú ý đến thanh ghi EAX bằng 5 bằng với số kí tự của Name là Doan3.



Tiếp tục F8 thu được giá trị của serial

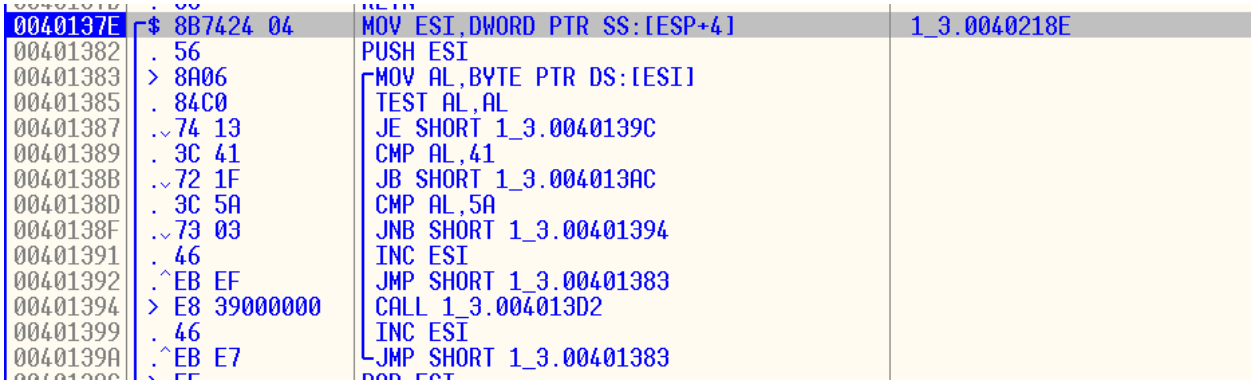
Sau khi tìm hiểu xong hàm GetDlgIteamtextA ta sẽ quay trở lại code chính bằng cách





Ta nhận thấy sau khi PUSH name vào STACK thì sẽ thực hiện 1 lệnh Call,và sau khi PUSH Serial cũng thực hiện 1 lệnh Call.Ta tiến hành phân tích lệnh CALL 1_3.0040137E trước

Khi F8 tới đoạn CALL ta nhấn F7 để vào trong hàm CALL



Phân tích đoạn code trên như sau.

MOV dữ liệu từ STACK[ESP+4] mà cụ thể ở đây dữ liệu là Name vào ESI

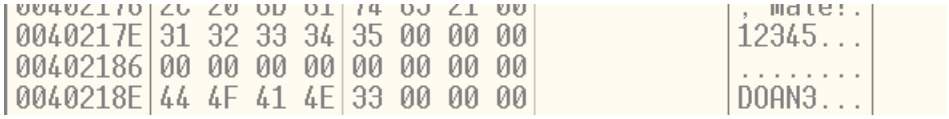
PUSH ESI vào STACK

MOV 1 BYTE từ ESI vào AL(Thanh ghi Low của AX)

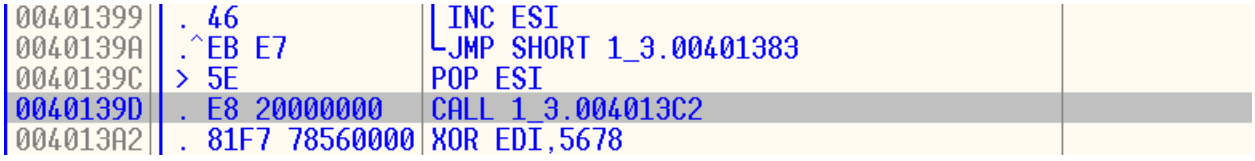
So sánh các Byte vừa lấy với 41.(41 trong mã Hex = 65 trong mã Dec và là mã của Kí tự A trong ASCII)

So sánh các Byte vừa lấy với 5A.(5A trong mã Hex = 90 trong mã Dec và là mã kí tự Z trong ASCII).

Tóm tắt 1 cách dễ hiểu thì đoạn code trên có nhiệm vụ đọc từng kí tự trong chuỗi Name có nằm trong khoảng từ A->Z hay không.Nếu không thì sẽ CALL đến 1_3.004013D2 để CONVERT



Sau khi CONVERT xong thì sẽ đến lệnh CALL



Nhấn F7 để xem lệnh CALL này làm gì

```
004013C2  $ 33FF        XOR EDI,EDI          EDX=0
004013C4  . 33DB        XOR EBX,EBX          EBX=0
004013C6  > 8A1E        MOV BL,BYTE PTR DS:[ESI]   MOV 1 BYTE ESI -> BL(LOW BX)
004013C8  . 84DB        TEST BL,BL           BL AND BL. IF BL=0 -> ZF =1
004013CA  .v74 05       JE SHORT 1_3.004013D1   JE IF ZF =1
004013CC  . 03FB        ADD EDI,EBX          EDI=EDI+EBX
004013CE  . 46          INC ESI              ESI+1
004013CF  .^EB F5       JMP SHORT 1_3.004013C6   JMP.JUMP K DIEU KIEN.VONG LAP
004013D1  > C3          RETN
004013D2  $ 2C 20       SUB AL,20
```

Đoạn code này làm nhiệm vụ cộng dồn toàn bộ các kí tự trong chuỗi Name lại vào lưu vào thanh ghi edi.

```
EBP 0019FDDC
ESI 0040218E ASCII "DOAN3"
EDI 00000111
```

111(Hex)=273(Dec)

Sau khi kết thúc hàm CALL

```
0040139D  . E8 20000000   CALL 1_3.004013C2
004013A2  . 81F7 78560000 XOR EDI,5678
004013A8  . 8BC7          MOV EAX,EDI
004013AA  .vEB 15         JMP SHORT 1_3.004013C1
004013AC  > 5E            POP ESI
004013AD  . 6A 30         PUSH 30                  Style = MB_OK|MB_ICONEXCLAMATION|MB_APPLMODAL
004013AF  . 68 60214000   PUSH 1_3.00402160        Title = "No luck!"
004013B4  . 68 69214000   PUSH 1_3.00402169        Text = "No luck there, mate!"
004013B9  . FF75 08       PUSH DWORD PTR SS:[EBP+8]   hOwner = 00130BB6 ('CrackMe v1.0',class='No need to disasm the code!')
004013BC  . E8 79000000   CALL <JMP.&USER32.MessageBoxA>   MessageBoxA
004013C1  > C3            RETN
004013C2  $ 33FF          XOR EDI,EDI              EDX=0
```

Giá trị EDI sẽ được XOR với 5678.Kết quả thu được sẽ MOV vào EAX

```
00401220  . E8 4C010000   CALL 1_3.0040137E
00401232  . 50            PUSH EAX
00401233  . 68 7E214000   PUSH 1_3.0040217E          ASCII "12345"
00401238  . E8 9B010000   CALL 1_3.004013D8
0040123D  . 83C4 04       ADD ESP,4
00401240  . 58            POP EAX
```

Kết thúc hàm Call xử lí Name.Kết quả sau khi được lưu vào EAX sẽ được push lên STACK

Tiếp tục PUSH Serial lên STACK và gọi 1 hàm Call để xử lí Serial

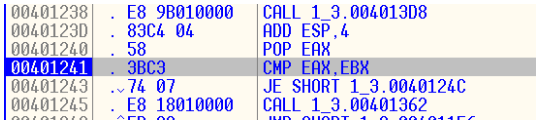
Nhấn F7 để vào hàm CALL này

```
004013D8  $ 33C0          XOR EAX,EAX          EAX = 0
004013DA  . 33FF          XOR EDI,EDI          EDI =0
004013DC  . 33DB          XOR EBX,EBX          EBX =0
004013DE  . 8B7424 04     MOV ESI,DWORD PTR SS:[ESP+4]   ESI = SERIAL
004013E2  > B0 0A         MOV AL,0A            AL = 0A
004013E4  . 8A1E          MOV BL,BYTE PTR DS:[ESI]   MOV 1 BYTE FROM ESI -> BL
004013E6  . 84DB          TEST BL,BL           AND BL BL. IF BL = 0 -> FLAG ZF =1
004013E8  .v74 0B         JE SHORT 1_3.004013F5   EXIT LOOP
004013EA  . 80EB 30       SUB BL,30            BL = BL -30
004013ED  . 0FAFF8        IMUL EDI,EAX         EDI = EDI +EAX
004013F0  . 03FB          ADD EDI,EBX          EDI = EDI + EBX
004013F2  . 46            INC ESI              ESI++
004013F3  .^EB ED         JMP SHORT 1_3.004013E2
004013F5  > 81F7 34120000 XOR EDI,1234
004013FB  . 8BDF          MOV EBX,EDI
004013FD  . C3            RETN
```

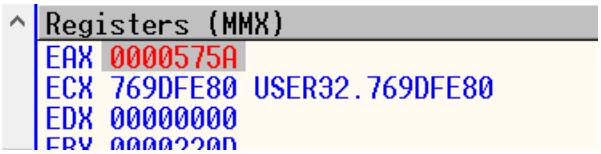
Đoạn code trên có nhiệm vụ Chuyển chuỗi Serial về dạng Hex và lưu vào thanh ghi EDI .

Sau đó sẽ đem EDI XOR với 1234 rồi lưu vào EBX

Sau đó trở về lại màn hình chính và tới phần so sánh giữa EAX và EBX



Bây giờ ta sẽ tim Serial chính xác với chuỗi Name vừa nhập.Vì lúc ta nhập test là doan3 có kí tự 3 không thuộc A-Z nên sẽ bị thông báo lỗi .Nên ta sẽ kiểm tra lại sau với Name là doan.Và sẽ chạy lại chương trình để tìm lại giá trị EAX.Giá trị của EAX khi có Name là doan là

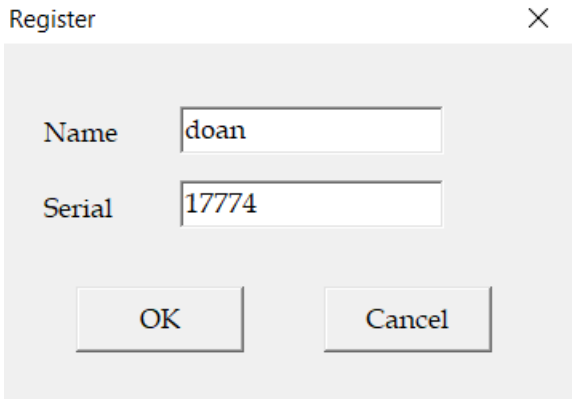


575A(HEX)

Serial của chúng ta sẽ là giá trị tính toán được của chuỗi Name và đem
xor với 0x1234

Lấy 575A XOR 1234 =456e(Chuyển sang Dec =17774)

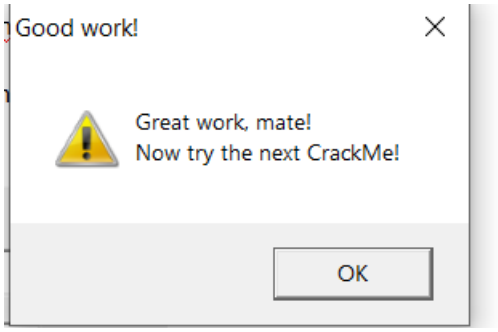
Thử chạy lại chương trình với Name doan và Serial 17774



Ta nhận được thông báo thành công

CHÚ Ý:

Dữ liệu Name phải là kí tự chữ và có độ dài < 11


Good work!
Great work, mate!
Now try the next CrackMe!
OK