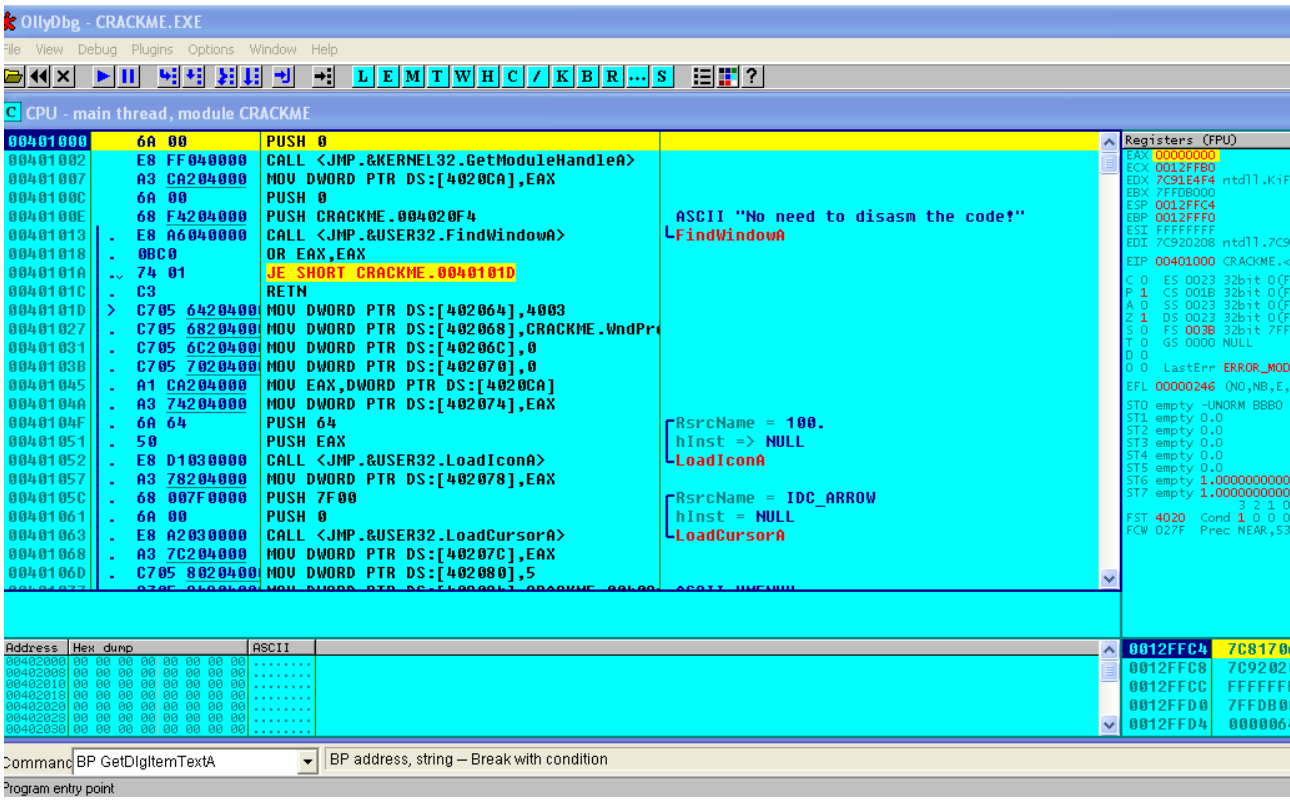
Crackme CruShead1999 – 14/08/2024 – KarasuRØØT

¡Buenas! Bueno la intención de este tute es tratar de llegar a resolver este viejo crackme que a muchos nos ha enseñado en nuestros comienzos.

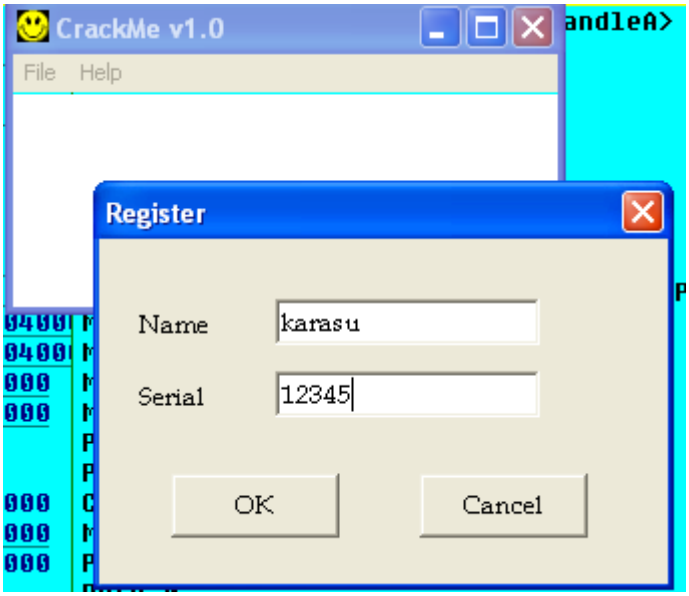
Particularmente Ricardo Narvaja mostraba en sus viejos tute cuando se usaba Olly como resolverlo – en INTRODUCCIÓN AL CRACKING CON OLLYDBG PARTE 16 – Si bien muestra cómo llegar a una solución, mi intención es abordarlo nuevamente después de tanto y realizar un keygen para que lo tome con cualquier nombre.

Voy a seguir con olly – obviamente hay cierto camino que es igual que en el tutorial original de Ricardo hasta un punto.

Bueno primero voy a poner un Bp en GetDlgItemTextA – esto se debe a que “Recupera el título o el texto asociado a un control en un cuadro de diálogo.” Según MSDN

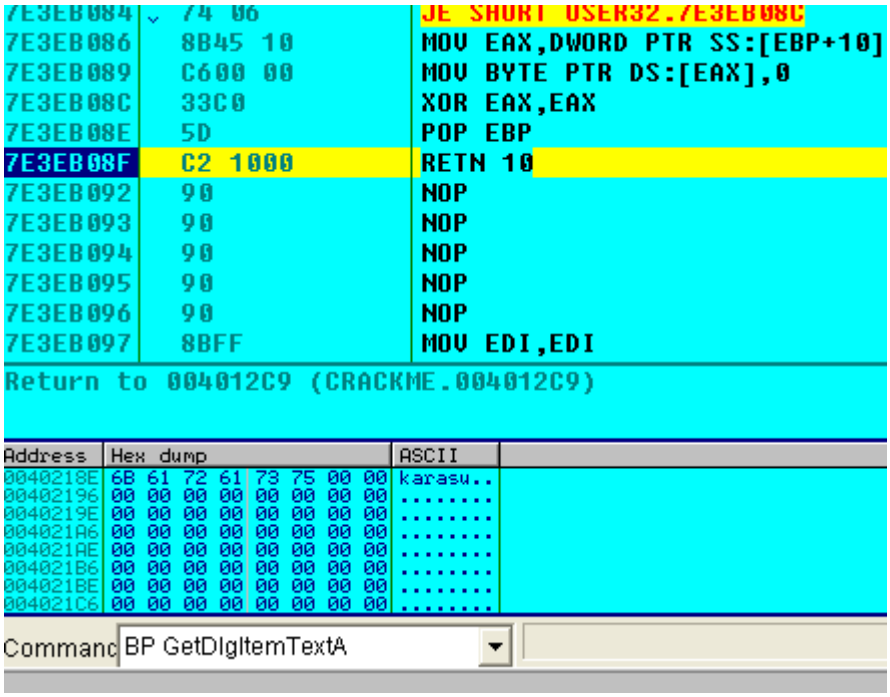


Le damos RUN y llegamos a la ventana correspondiente -> si damos en la opción “Help”-> Register aparece el famoso registro donde nos pide un nombre y un serial



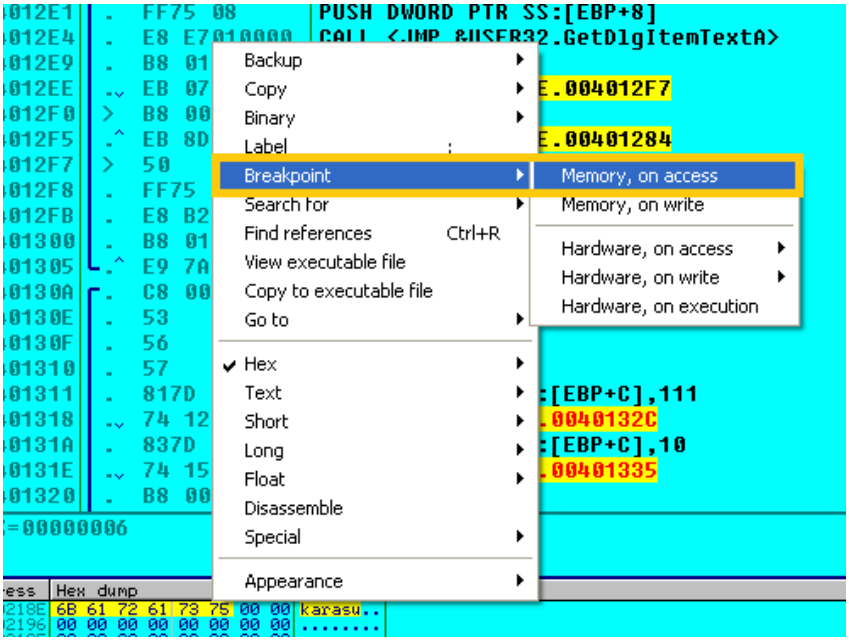
Así que comenzare a probar – el nombre usare “karasu” – el serial 12345 y le doy OK

Se detiene en la instrucción MOV EDI,EDI -> pero lo importante aquí es lo que aparece en el STACK -> allí se aloja lo que se ingresó en el Nombre – es decir “karasu” – bueno sigamos -> Click derecho -> Follow in Dump y si veo la ventana del dump está en 0 – esto es, como lo explica Ricardo – porque todavía no escribió, aun no se ejecutó la api todavía – le damos Ctrl + F9 o bien -> en la opción superior -> Debug -> Execute till return

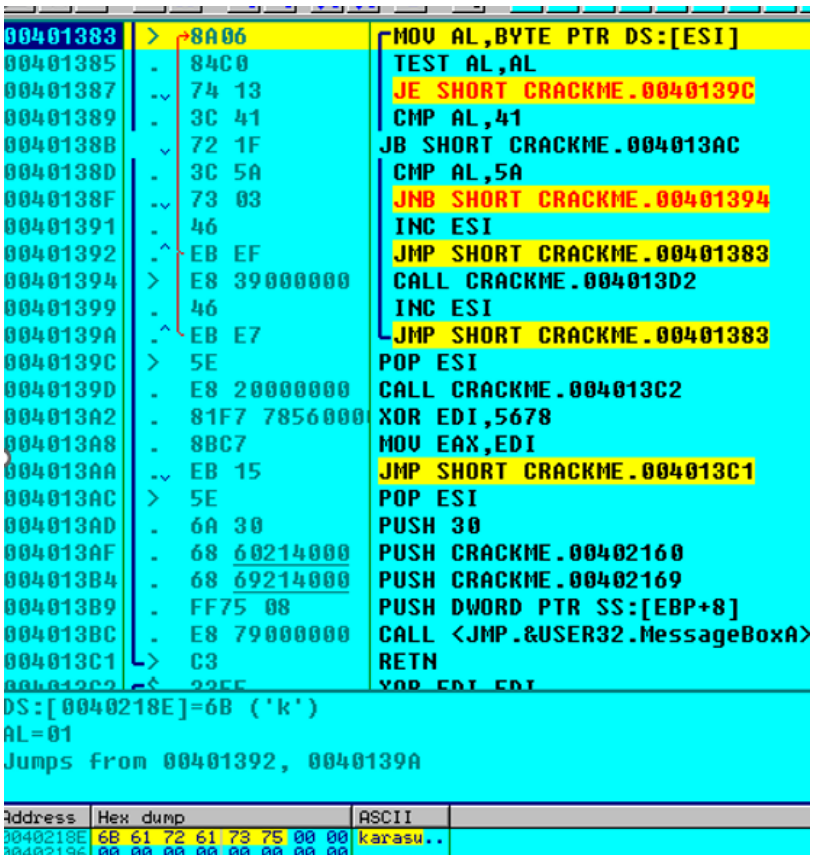


A partir de aquí voy a apartarme un poco de lo que se venía viendo en el tute original de Ricardo

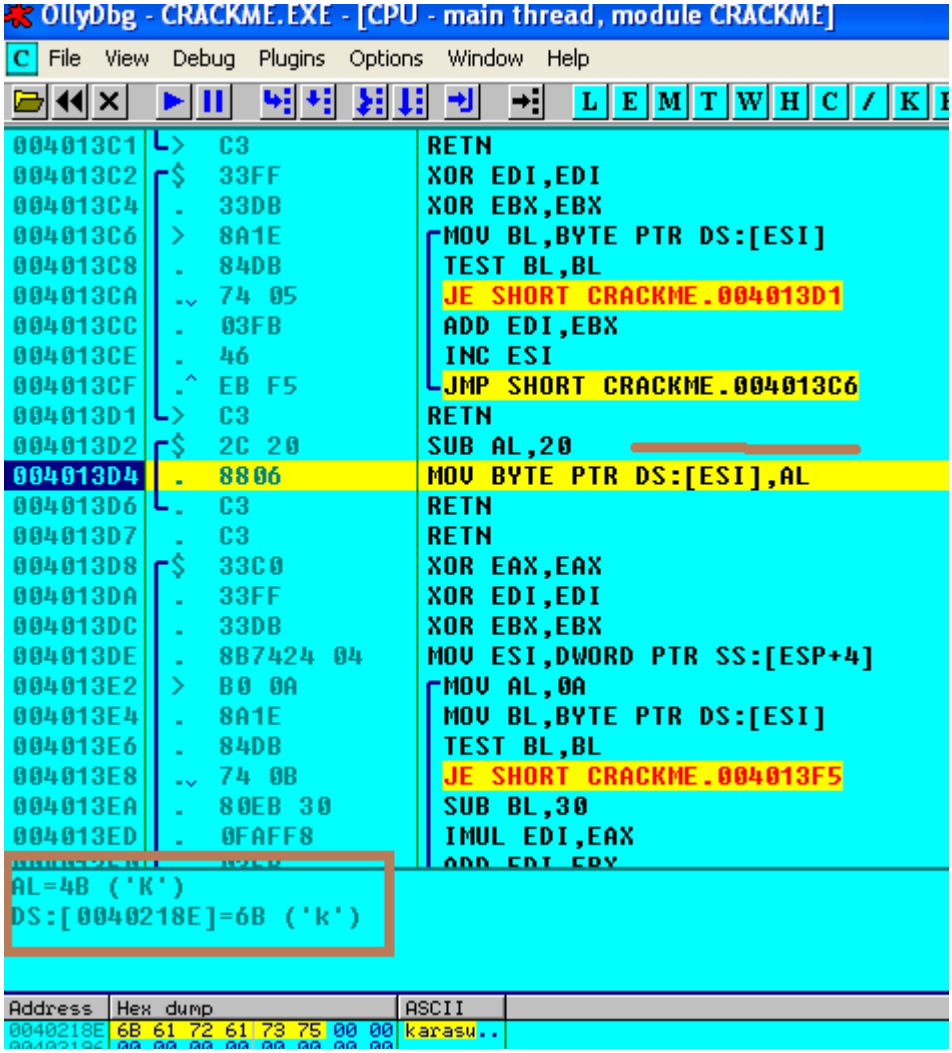
Ahora si vemos que escribió el nombre – OK -> F7 – ahora lo que voy a hacer es marcar el valor hexa decimal del dump – marco 6B 61 72 61 73 75 – Luego voy a poner un Breackpoint On ACCES -> para ello -> click derecho ->



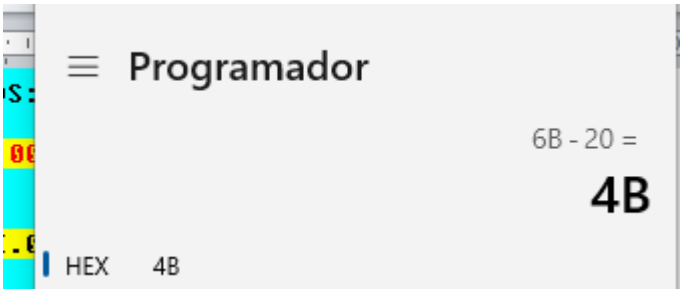
Y le damos F9 o RUN (x2 es decir dos veces para llegar a la instrucción) ->



Recordemos que para una instrucción antes –es decir que aún no se ejecuta “MOV AL, BYTE PTR DS:[EDI]” -> si vemos en la aclaración del Olly indica que [004018E] vale 6B (que en hexa es el valor de ‘k’ en la tabla ASCII) y se guarda en AL – sigo traceando y llego a este punto, creo yo que es importante:

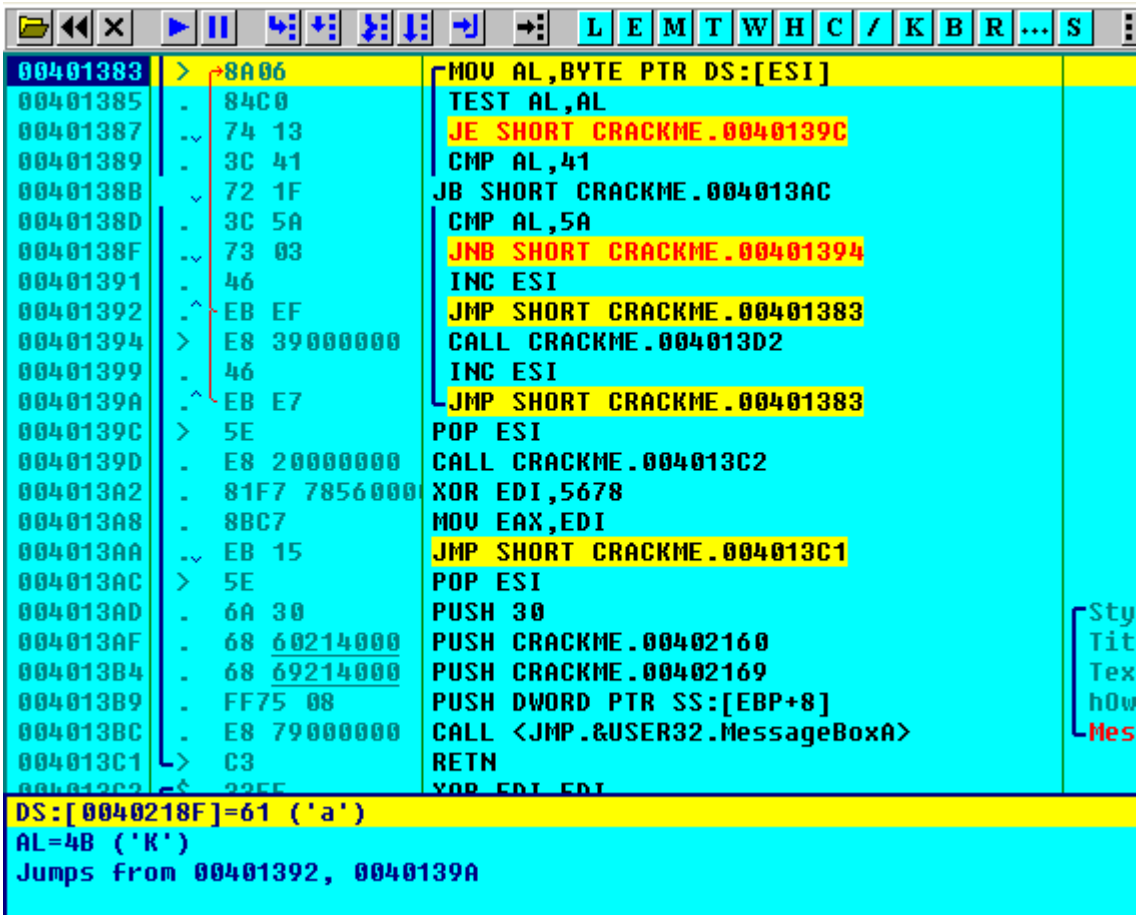


Recordemos que AL = 6B -> luego le resta 20 en SUB AL, 20 = entonces esto quiere decir que ahora AL vale 4B

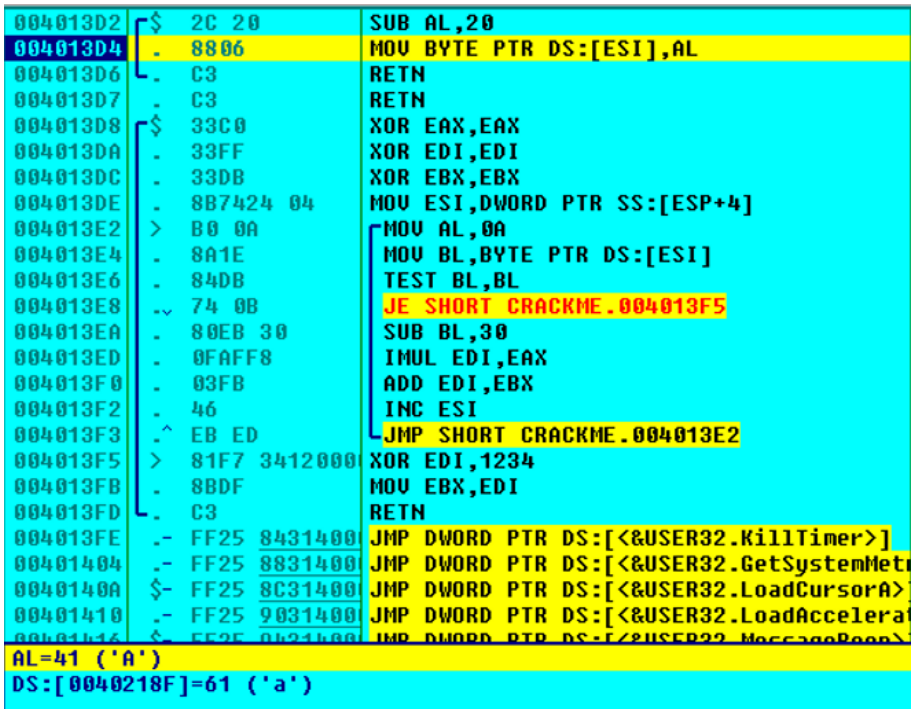


Y eso quiere decir que entonces que k – ahora es 4B (En ASCII “K”) y luego hay un RETUN (RET) -> entonces vuelve → Return to 00401399 (CRACKME.00401399)

Al seguís traceando -> llego otra vez al bucle -> observar que ahora toma la ‘a’ y AL conserva el valor 4B(‘K’)

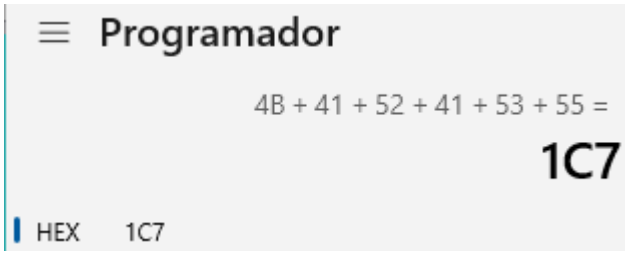


En la siguiente instrucción AL = 61 sigo traceando y otra vez:

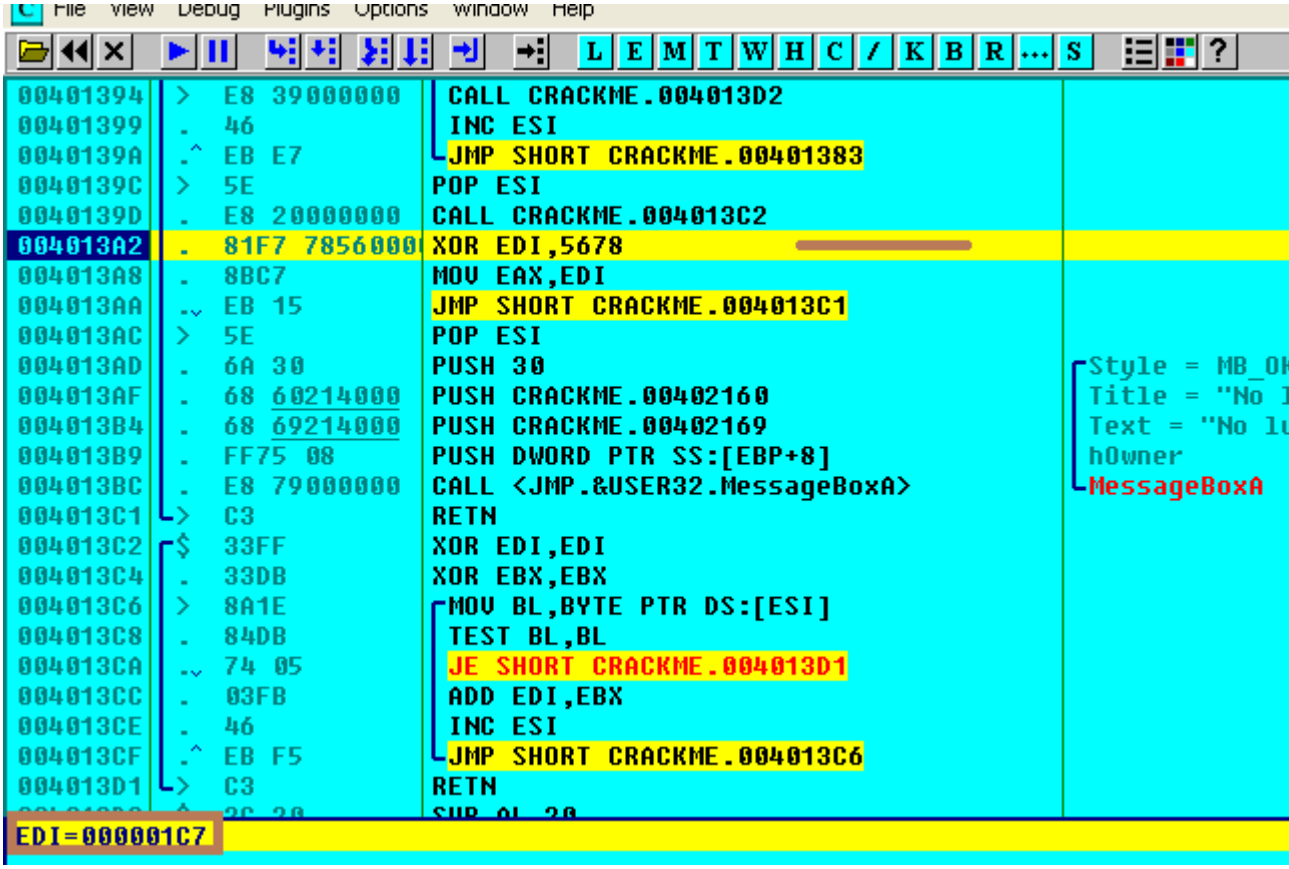


Es decir, otra vez toma la siguiente letra y la convierte en mayúscula -> y luego va juntando el resultado en EDI – para no hacerlo tedioso, lo que hace es que recorre la string que se haya puesto en el nombre -> lo pone todo en mayúsculas -> Luego suma cada carácter – entonces si yo ingrese “karasu” -> lo va a convertir en “KARASU” básicamente y luego suma cada carácter ➔ KARASU = 4B 41 52 41 53 55 (en Hexa)

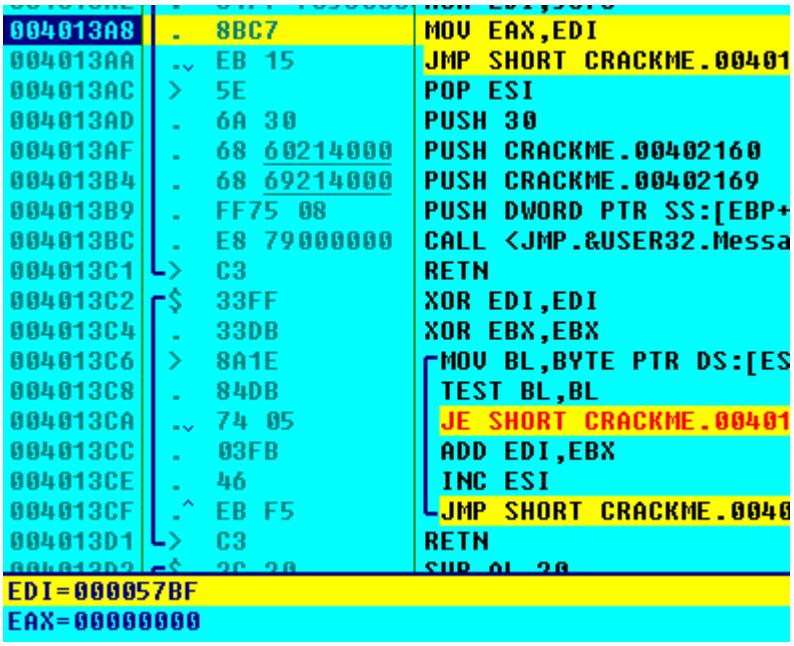
Luego suma carácter por carácter



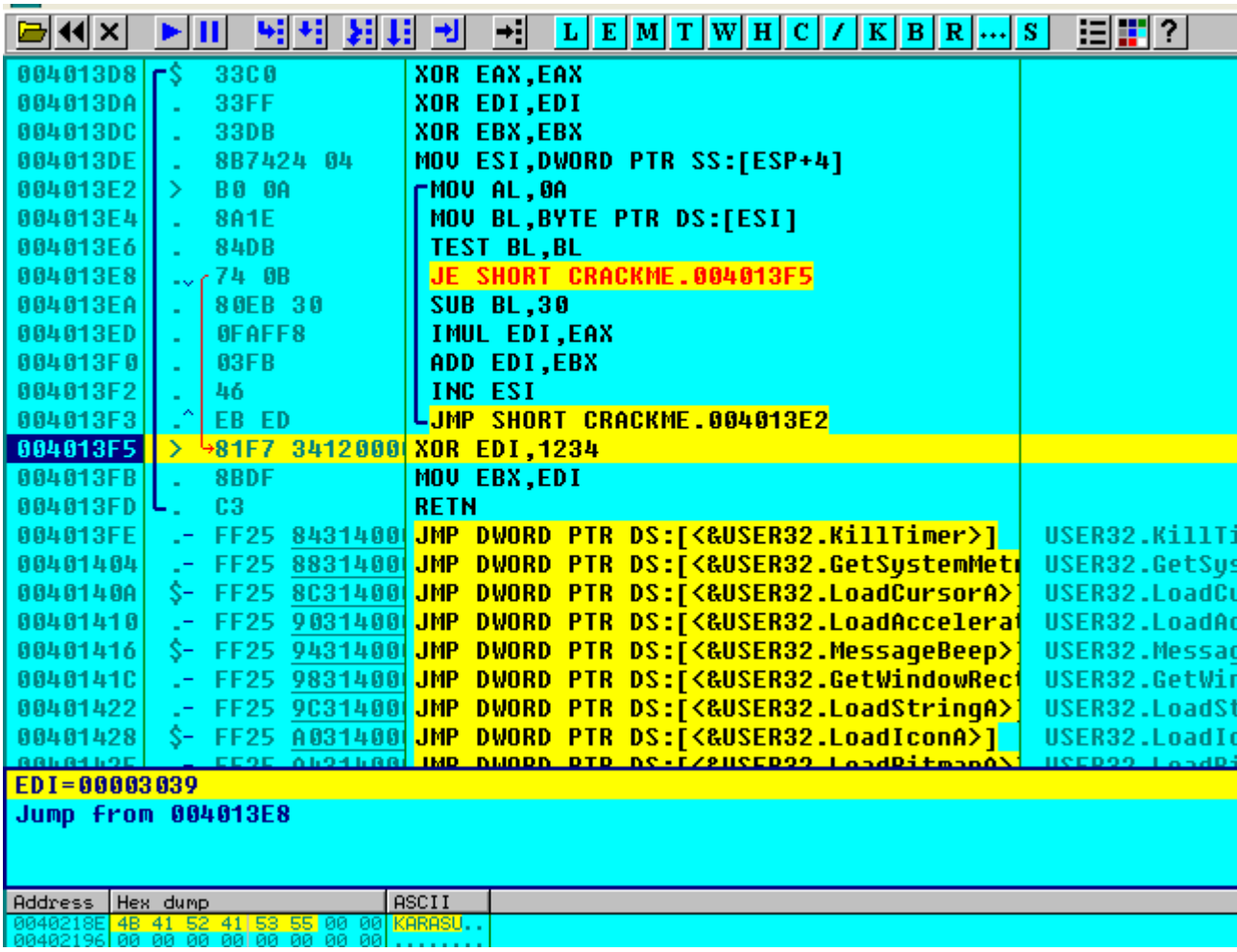
Una vez que recorre toda la stirng y sale del loop llegamos aquí:



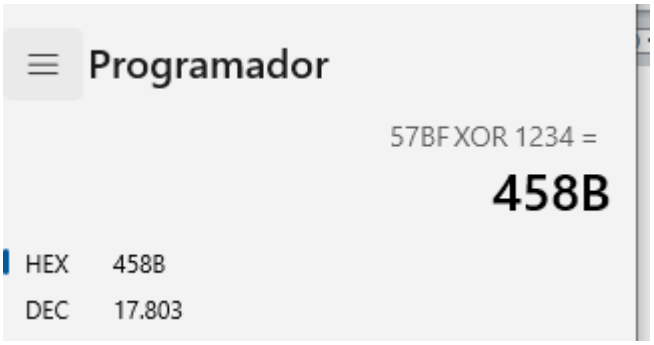
Notar que efectivamente EDI = 1C7 ➔ Luego hace un XOR EDI, 5678 = 57BF posteriormente mueve ese valor a EAX ->



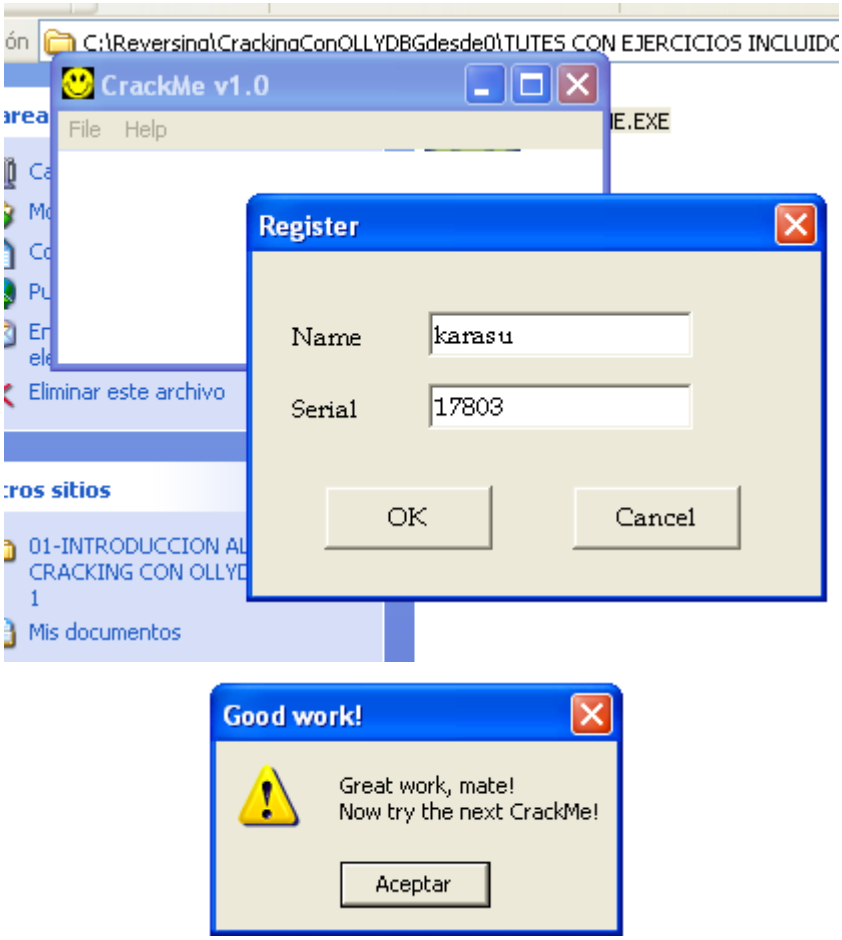
Luego seguimos traceando – es la parte donde evalúa nuestro serial – no lo mostrare ya que mi serial ya sé que es falso y llegamos a esta instancia:



Ahí básicamente lo que hace es XOR EDI, 1234 – entonces eso quiere decir, que SI EDI = 57BF => 57BF XOR 1234 =



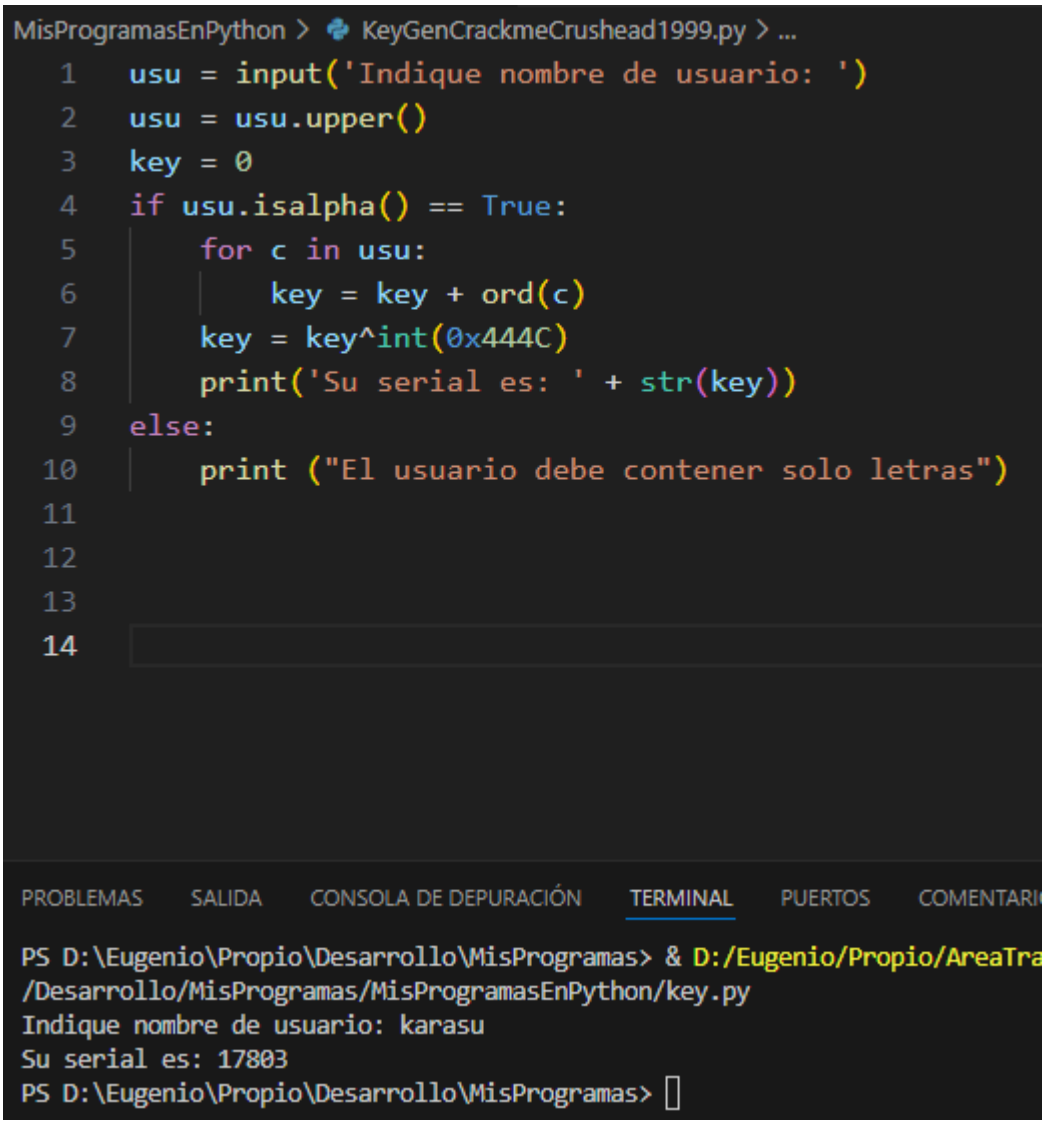
Es decir que ahora valdrá 458B = lo que quiere decir que el serial seria 17803 – el programa sigue y es cuando devuelve el cartel de chico malo – pero podemos probarlo –



Al dar OK -> devuelve el mensaje “Great work, mate! Now try the next Crackme!” → entonces le ganamos :D – algo que quizás no está demás, es que la clave 17803 sirve tanto como para la string “KARASU” como para “karasu”

Bueno llego el momento de hacer un script

Repasando entonces básicamente lo que hay que hacer es que el script me pida un nombre(string) -> Luego convertirlo a mayúsculas -> luego sumar carácter por carácter -> luego que haga XOR 5678 y a ese resultado que aplique XOR 1234 y luego que devuelva el valor numérico

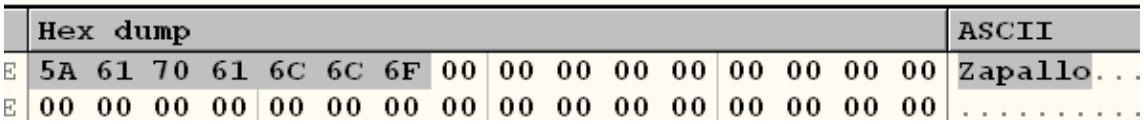


Corrección (15/08/2024)

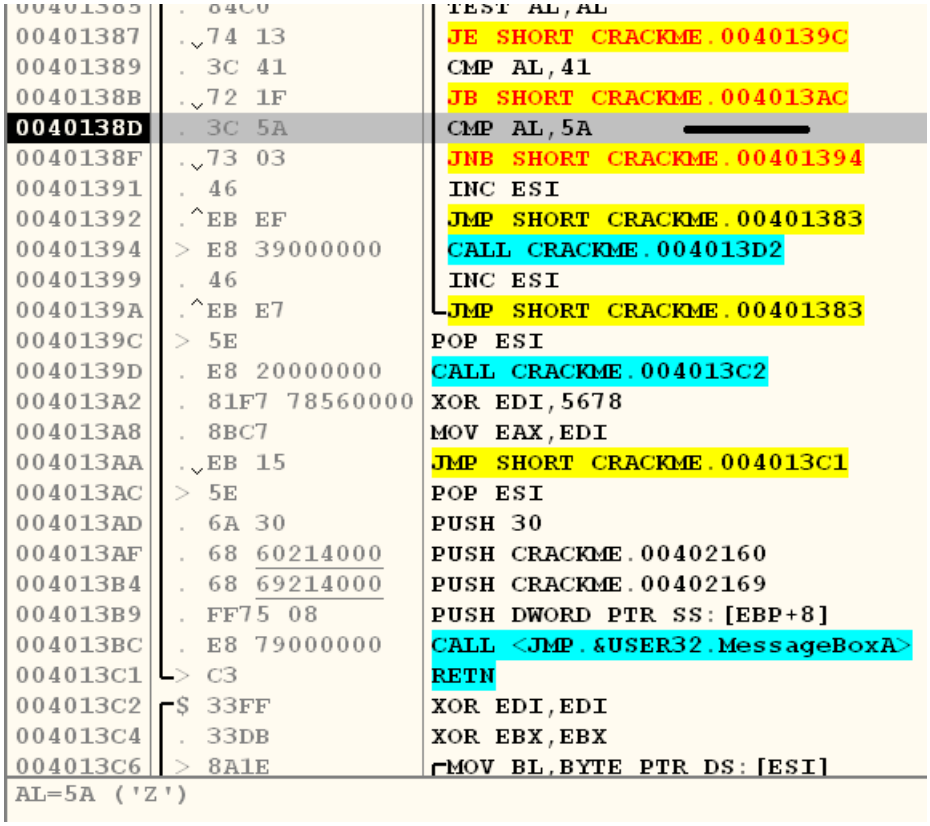
Alguien me hizo notar que si se usa por ejemplo Zapallo (hacer una mención sencilla – con zapallo el anterior keygen funciona sin problemas) como usuario, al generar el serial en ese keygen inicial – no funcionaría y daría cartel de chico malo.

Por ello me puse a investigar, por que algo se me había pasado por alto y efectivamente así fue:

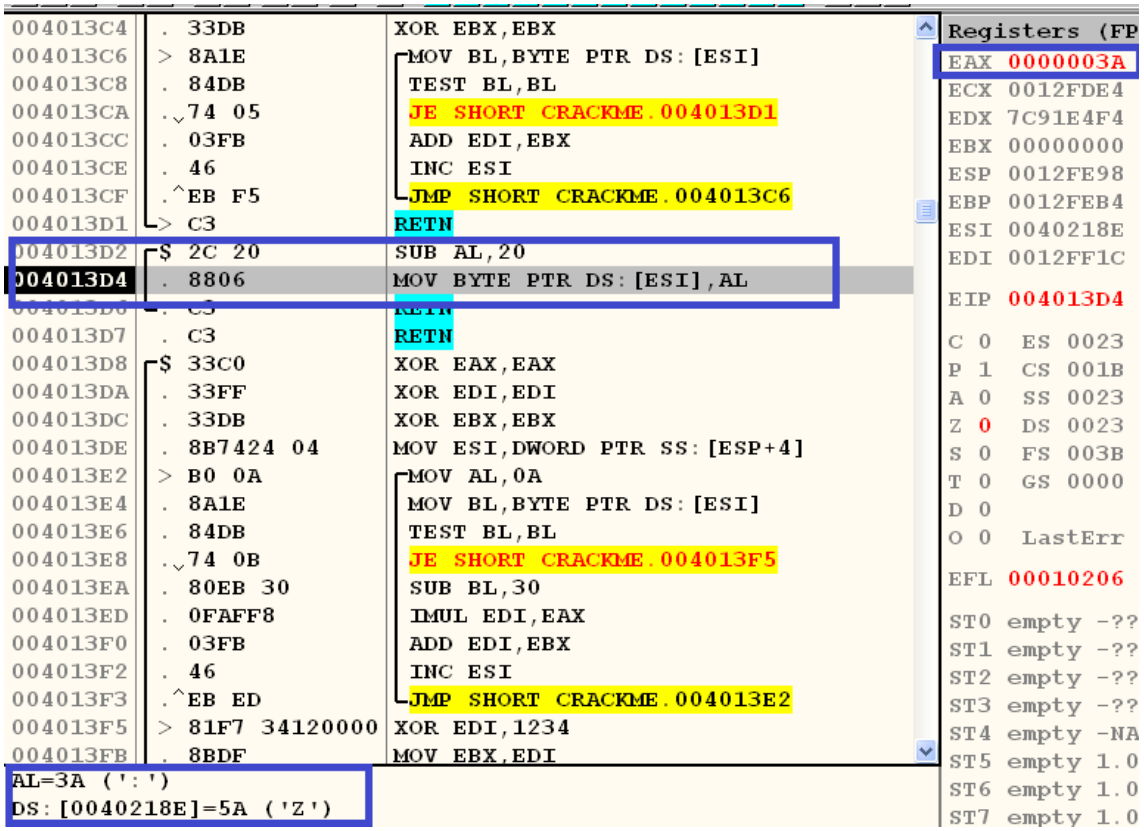
Al realizar el traceo nuevamente me doy cuenta que el problema es la Z



Observar que Z es 5A en Hexa → bueno por que existe una línea que complica las cosas:

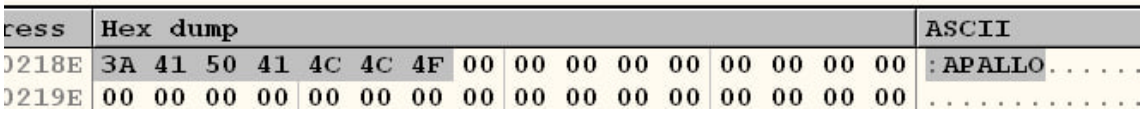


Allí cuando compara y comienza a hacer el bucle de convertir los caracteres a mayúsculas y etc.- Z(Siendo 5A) estando en AL – es comparado contra 5A esto causa que cuando llegue a la instrucción



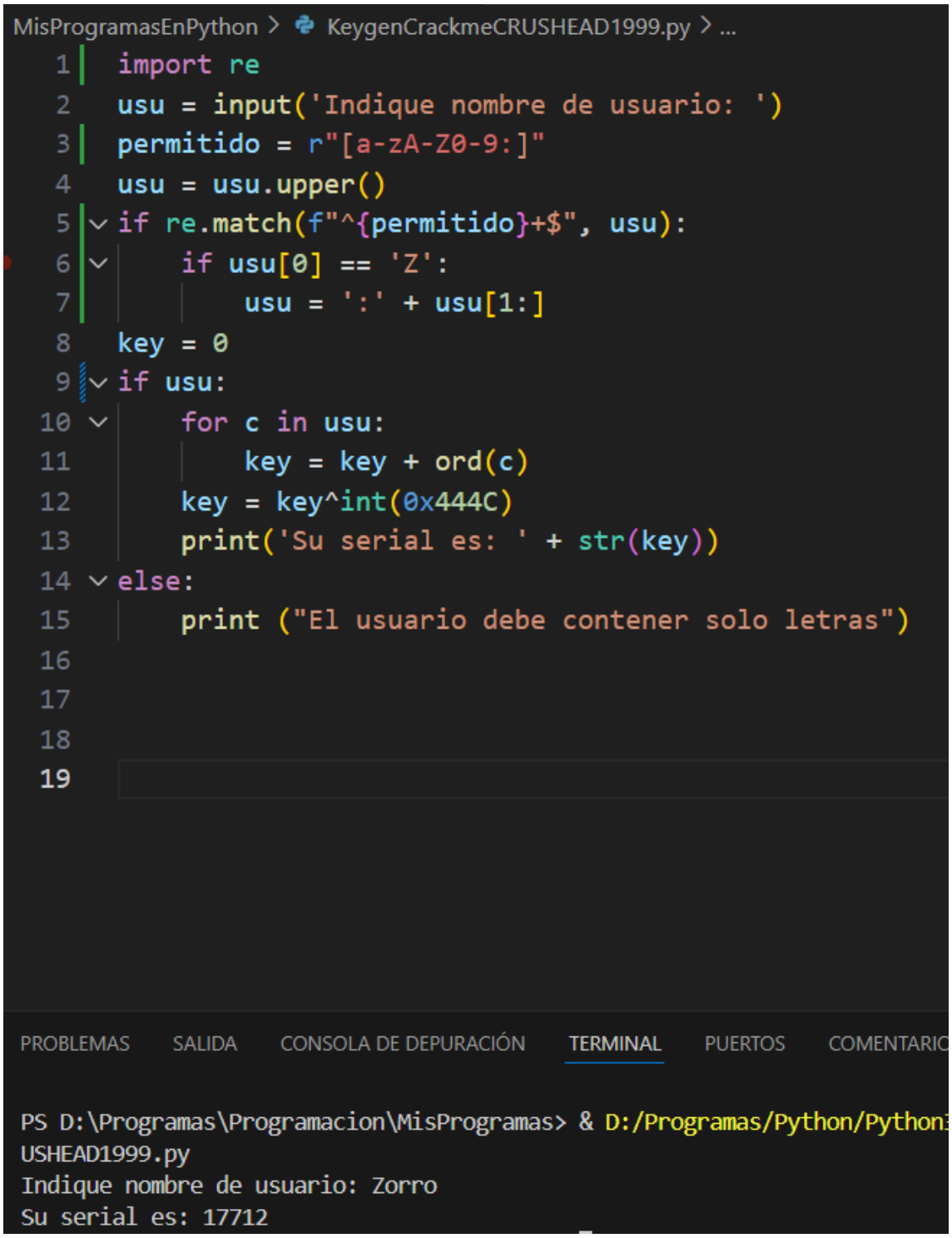
Al llegar al SUB AL,20 -> causa que 5A -20 = 3ª y eso quiere decir que eso es representado

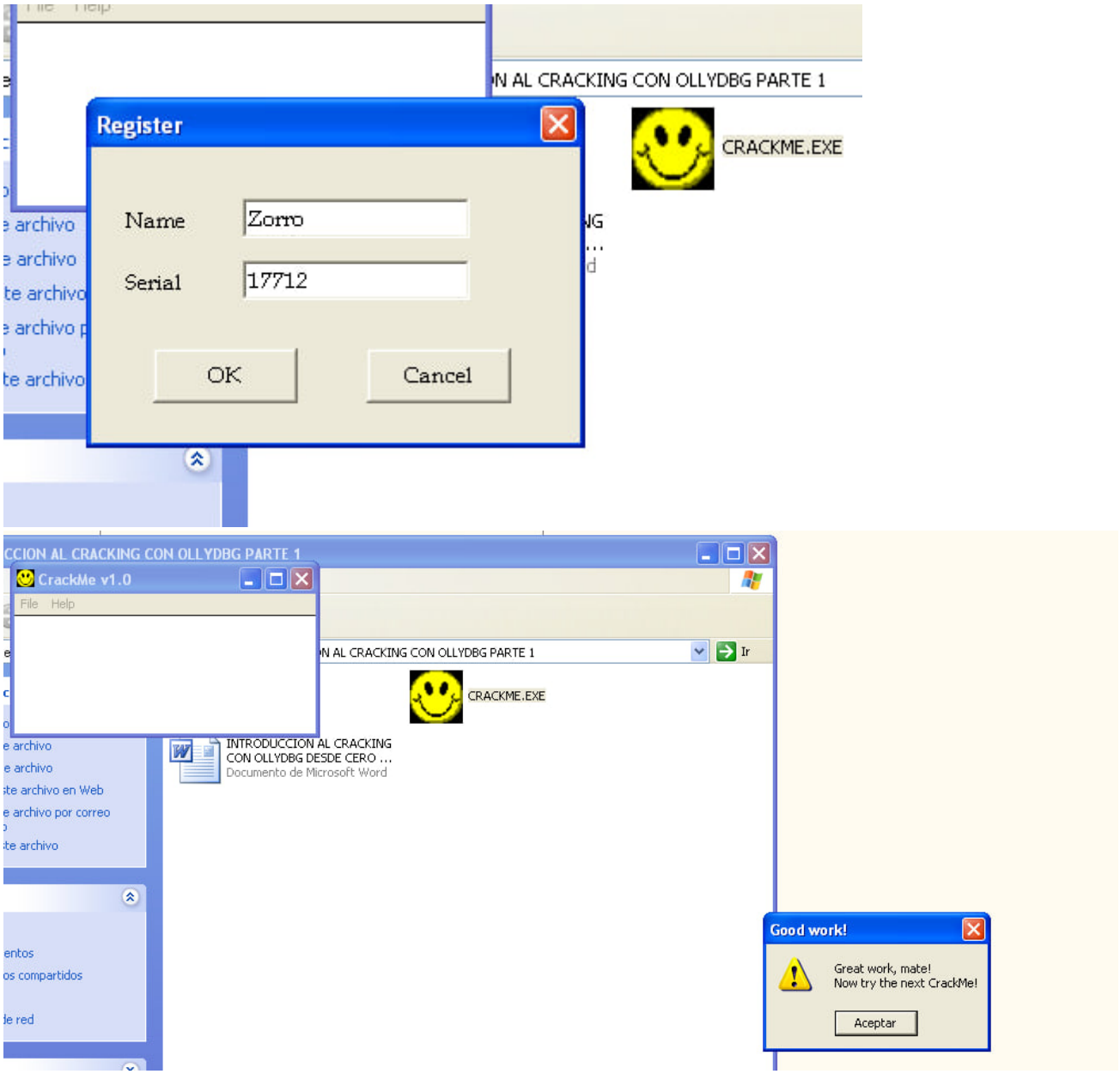
la salida seria así:



Entonces hay que modificar el script por que eso no lo tiene contemplado, por ello se modifica para que cunado se usa un nombre que comience con “Z” -> lo reemplace con “:” y realice las operaciones desde allí.

Usando “Zorro” como nombre: realicé los cambios en el script y tuve estos resultados:




N AL CRACKING CON OLLYDBG PARTE 1
CRACKME.EXE
Register
Name
Zorro
Serial
17712
OK
Cancel
CCION AL CRACKING CON OLLYDBG PARTE 1
CrackMe v1.0
File Help
N AL CRACKING CON OLLYDBG PARTE 1
Ir
CRACKME.EXE
INTRODUCCION AL CRACKING CON OLLYDBG DESDE CERO ...
Documento de Microsoft Word
Good work!
Great work, mate!
Now try the next CrackMe!
Aceptar