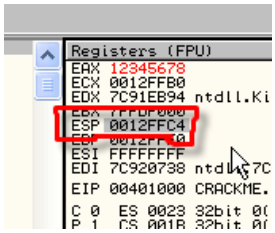
INTRODUCCION AL CRACKING CON OLLYDBG PARTE 3

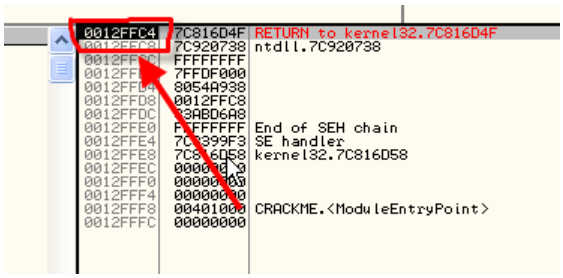
QUE SON LOS REGISTROS Y PARA QUE SIRVEN

Ahora para que sirven y que son exactamente los registros?
Bueno el procesador necesita asistentes en su tarea de ejecutar los programas.
Los registros lo ayudan en ello, cuando veamos las instrucciones ASM veremos por ejemplo que no se pueden sumar el contenido de dos posiciones de memoria directamente, el procesador tiene que pasar una de ellas a un registro y luego sumarla con la otra posición de memoria, este es un ejemplo pero por supuesto ciertos registros tienen usos mas específicos veamos.

ESP apunta al valor superior del stack, vemos en nuestro Crackme de Cruehead como ejemplo.

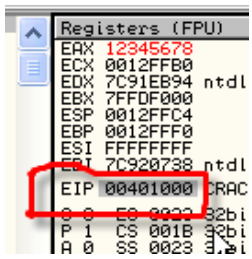


ESP vale 12FFc4 y si miramos el stack en OLLY en el mismo momento

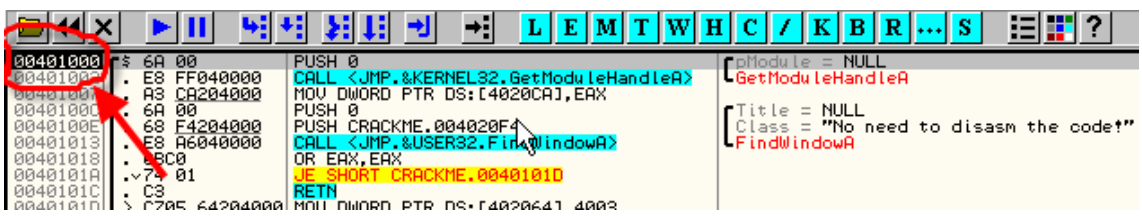


Vemos que apunta al valor superior de nuestro stack o dicho en forma simpática, a la carta superior de nuestro mazo de cartas o barajas.

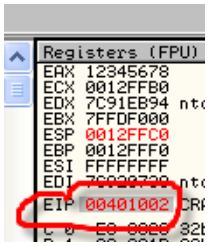
EIP es otro registro muy importante apunta a la instrucción que esta siendo ejecutada en este momento veamos



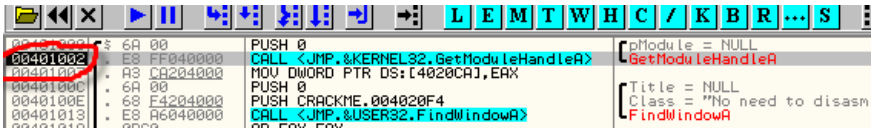
Veamos en el listado del OLLYDBG, que al arrancar el crackme de Cruehead, este paro allí en 401000, que es la primera instrucción a ejecutar y por supuesto el valor de EIP cuando esta detenido allí será 401000.



Si apreto F7 ejecuta la primera instrucción y pasa a la siguiente.

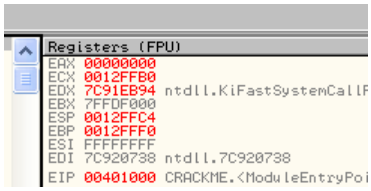


EIP ahora vale 401002 y en al listado vemos que se ejecuto la primera instrucción y ahora estamos en 401002.



Los otros registros pueden tomar valores variables y sirven para asistir al procesador en las ejecuciones de las instrucciones, ECX es usado casi siempre como contador los demás son fluctuantes y asisten en la ejecución de programas como veremos en la explicación de cada instrucción.

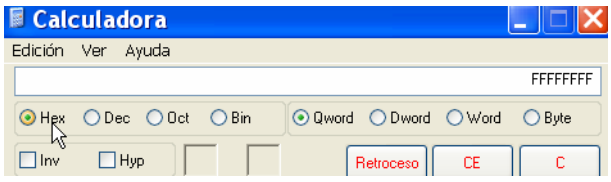
Recordamos donde el OLLYDBG nos mostraba el valor de los REGISTROS

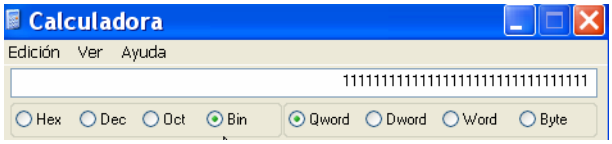


Vemos a simple vista que son EAX, ECX, EDX, EBX, ESP, EBP, ESI, EDI y EIP

Esos son los llamados REGISTROS de 32 bits

OLLYDBG expresa el contenido en hexadecimal, vemos por ejemplo que EAX vale 00000000, y el máximo valor que podría tener es FFFFFFFF, si lo pasamos a BINARIO seria 11111111111111111111111111111111.



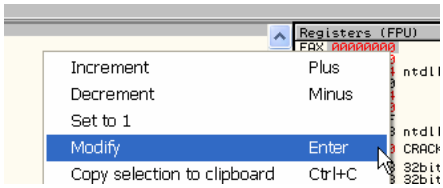


Vemos que son 32 bits cada uno con la posibilidad de ser 0 o 1 en numero binario, por eso se llama a estos, registros de 32 bits.

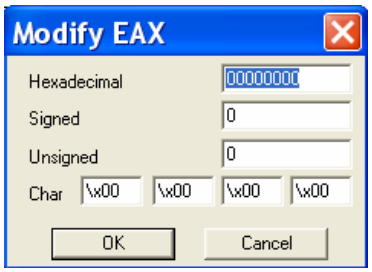
En el lenguaje ASM se pueden operar con partes de los registros de 32 bits, en este caso EAX, puede ser subdividido.

Veamos en el OLLY para más practicidad con un ejemplo:

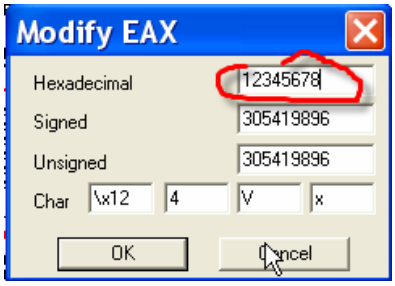
Cambiare el valor de EAX a uno que yo quiera en este caso 12345678.
Abro el OLLYDBG y allí mi programa será el CRACKME DE CRUEHEAD, aunque podría ser cualquiera.
Una vez que arranca y para en el inicio hago clic derecho en EAX y elijo MODIFY



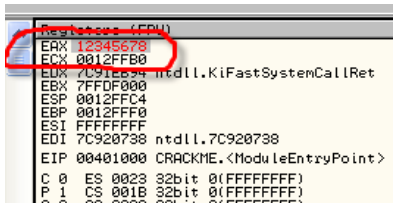
En la ventana que se abre escribo en la línea hexadecimal el valor 12345678



Queda así



Luego acepto con OK



Allí vemos como quedo cambiado al valor que yo deseaba, OLLYDBG tiene la particularidad de poner en ROJO los valores que se modifican.

Como decíamos se pueden usar solo partes de EAX, en este caso AX seria el registro de 16 bits o sea las cuatro ultimas cifras de EAX, por lo tanto AX valdría en este caso 5678, corroborémoslo en OLLY en el commandbar tipeemos

? AX (ya que el signo de interrogación sirve también para hallar el valor de una expresión o de un registro)

Cuando apreto ENTER

Vemos que dice 5678 que es lo que suponíamos, AX son las ultimas cuatro cifras de EAX.
También existen AL y AH, cuales son estos miremos en OLLYDBG

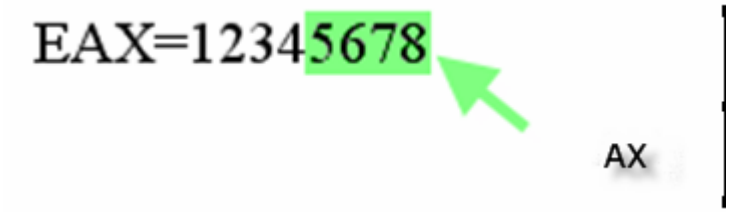
? AL

? AH

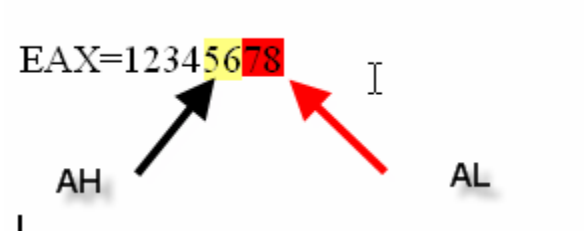
O sea si

EAX=12345678

AX son las ultimas cuatro cifras



AH la 5 y 6 cifra y a su vez AL las ultimas dos cifras



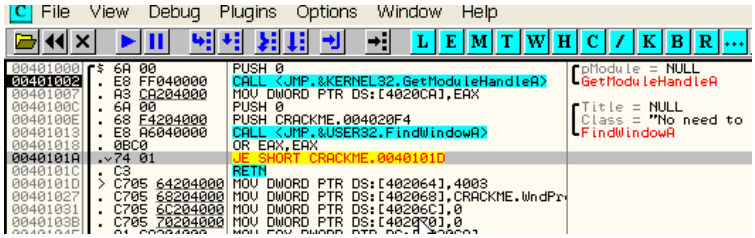
También en la misma forma EBX se puede subdividir en BX, BH y BL y así sucesivamente existen subdivisiones para casi todos los otros registros.

COMO CAMBIAR LOS VALORES DE LOS REGISTROS

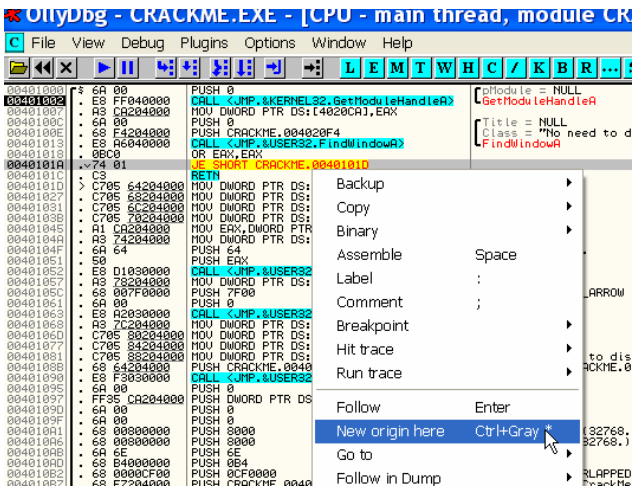
Ya vimos como se pueden cambiar valores de los registros en OLLYDBG, lo que hicimos en EAX se puede hacer en los otros registros de la misma forma, marcando el registro que deseamos cambiar de valor, luego haciendo CLICK DERECHO-MODIFY, salvo en el caso de EIP, dado que el mismo apunta a la instrucción que se esta ejecutando.

Para cambiar EIP operamos de la siguiente forma:

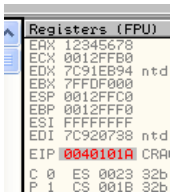
Ya que EIP siempre apunta a la instrucción que se va a ejecutar, elegimos una nueva instrucción en el listado.



Luego que esta marcada como en este ejemplo 40101A, hago en ella CLICK DERECHO-NEW ORIGIN HERE y cambiara EIP a 40101A, continuando el programa ejecutándose desde allí.

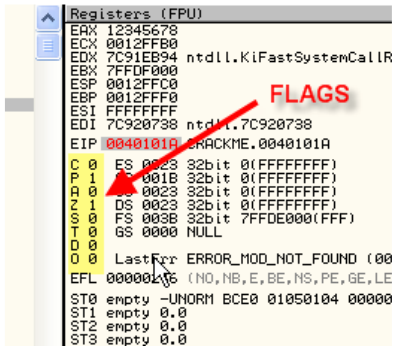


Como vemos queda EIP valiendo 40101A



QUE SON LOS FLAGS?

Como vimos en el primer tutorial en OLLYDBG debajo de los registros se encuentran los flags o banderas.



Vemos que los flags son C P A Z S T D y O

Vemos que solo pueden tener valores de cero o uno, que nos advierten que al ejecutar determinada instrucción, ha ocurrido algo, según el flag que sea.

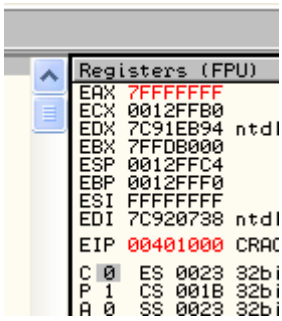
Vayamos mirando que indica cada uno:

EL FLAG O O FLAG OVERFLOW (DESBORDAMIENTO)

Se activa cuando al hacer una operación, el resultado cambia de signo dando un valor incorrecto.

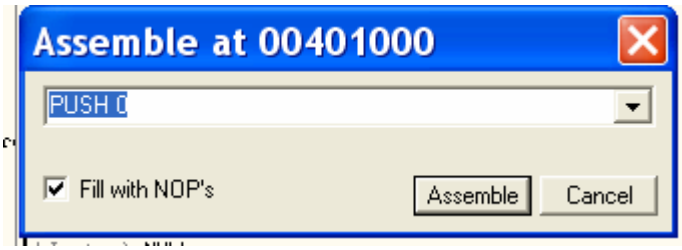
Miremos en OLLYDBG este ejemplo, como siempre en el CRACKME DE CRUEHEAD de paso vamos practicando usar el OLLYDBG.

Modifico como hicimos antes el valor de EAX a 7FFFFFFF que es el máximo positivo posible.

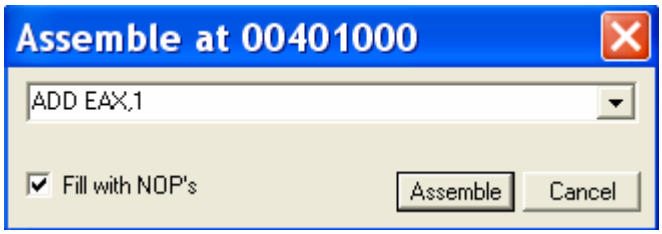


Ahora le sumare 1, lo cual excederá la posibilidad de EAX de mostrar un resultado positivo ya que 80000000 ya corresponde a un número negativo

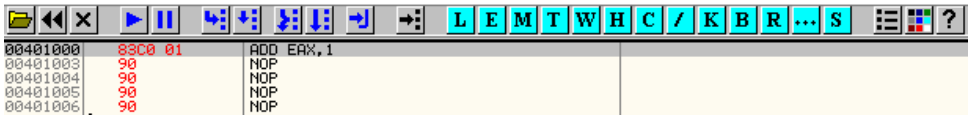
Para eso apreto la barra espaciadora que me permite escribir instrucciones.



Me sale esa ventana donde escribo ADD EAX,1.

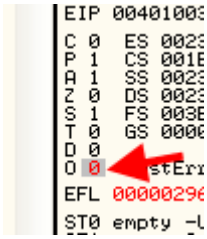


Al apretar el botón ASSEMBLE vemos que cambia la instrucción que había antes en 401000 por la que yo escribí.



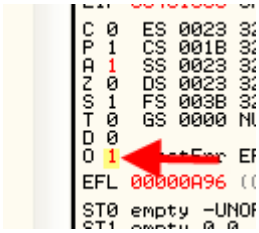
ADD EAX, 1 (ya lo veremos cuando enumeremos y expliquemos las instrucciones) seria sumarle a EAX el valor 1, y guardando el resultado en el mismo EAX.

Veo que antes de ejecutar la línea con F7 el flag O esta en cero



Si ejecuto la instrucción con F7 para ver que es lo que ocurre, al realizar dicha operación veo que EAX al sumarle 1 se desborda y me muestra 80000000 lo cual traspasa la línea del cambio de signo.

El FLAG O se activa poniéndose a 1 indicándome que la operación excedió el máximo resultado posible y esa es su función indicar cuando ocurra desborde al ejecutar una instrucción.



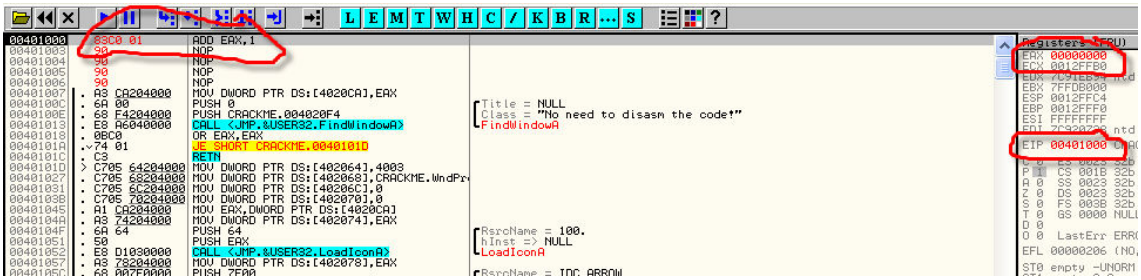
El FLAG A o AUXILIAR

Tiene una función similar pero para cuando se realizan operaciones con otros formatos que por ahora no nos interesan.

El FLAG P o PARIDAD

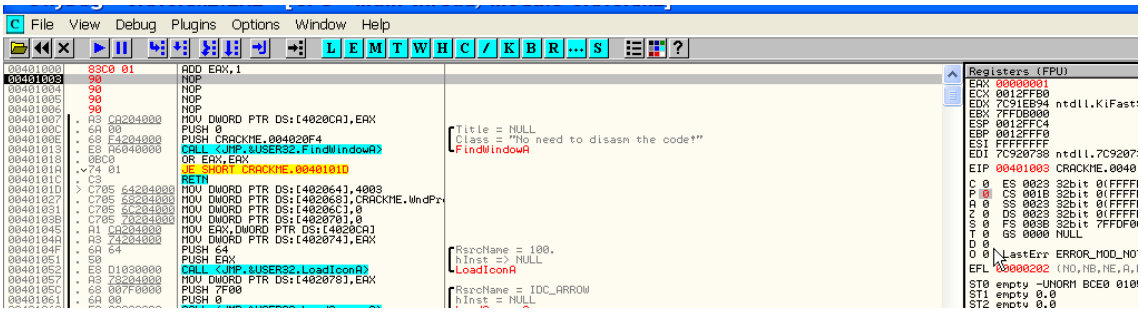
Dicho flag se activa cuando ejecutamos una instrucción y su resultado es un valor, que pasado a numero binario tiene una cantidad par de unos, como por ejemplo 1010, o 1100 o 1111000 que tienen resultados cuya cantidad de unos total es par.

Para probar esto ya que tenemos en el OLLYDBG escrito ADD EAX,1 y como ya ejecutamos esa línea para probar el flag anterior, pues la marcamos de nuevo y hacemos CLICK DERECHO-NEW ORIGIN HERE lo cual llevara EIP de nuevo a 401000 (volvemos atrás) y a que si apreto F7 se ejecute de nuevo la instrucción que escribimos ADD EAX,1.



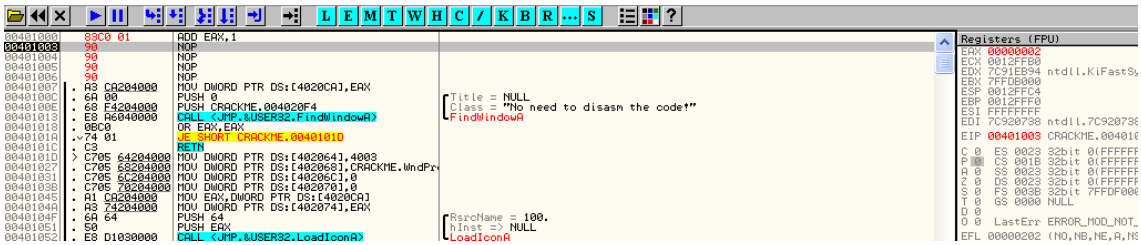
Allí tenemos pues de nuevo justo antes de ejecutar la suma, con EAX valiendo 00000000 y el flag P valiendo 1, porque quedo así de la operación anterior, veamos que ocurre cuando le sumamos 1 a EAX nuevamente.

Apretamos F7

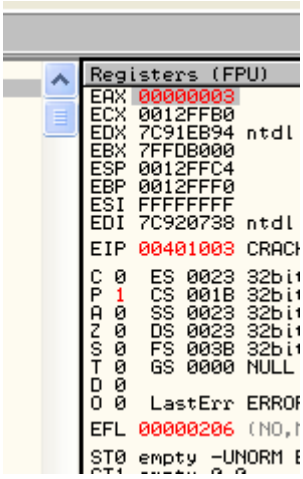


Vemos que P nos marca 0 porque el resultado que muestra EAX=00000001 que en binario es 1 y tiene un solo 1 o sea un numero impar de unos por eso no se activa.

Y vuelvo a hacer ahora click derecho en nuestro ADD EAX,1 y de nuevo CLICK DERECHO- NEW ORIGIN HERE para volver a sumar 1 y apreto F7.



Vemos que EAX que valía uno, al sumarle uno nuevamente, ahora vale 2 que es 10 en binario y sigue el resultado teniendo un solo uno por lo cual el flag P no se activo, si repito el procedimiento una vez mas, volviendo atrás y apretando F7 para sumarle 1 nuevamente a EAX.



Ahora EAX vale 3 que en BINARIO es 11 o sea el resultado tiene un numero par de unos por lo cual se activo el FLAG P o de paridad.

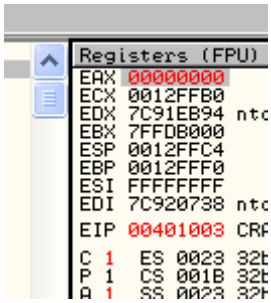
Con eso vemos como funciona el susodicho FLAG, al ejecutar una operación solo mira el resultado y si el mismo en BINARIO tiene cantidad par de unos, se activa.

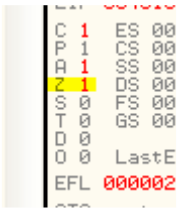
El FLAG Z o FLAG CERO

Uno de los más conocidos y usados en el cracking es el FLAG CERO el mismo se activa cuando ejecutamos una instrucción y el resultado es cero.

Podemos volver con CLICK DERECHO-NEW ORIGIN HERE a nuestro ADD EAX,1 de 401000, pero cambiemos ahora el valor de EAX a FFFFFFFF que es -1 decimal, de forma de que cuando apretemos F7 y ejecutemos ADD EAX,1 , sumemos -1 +1 el resultado sea cero a ver si se activa el FLAG Z.

Vemos que al apretar F7, EAX quedo en cero y como el resultado es cero, se activo el FLAG Z poniéndose a uno.

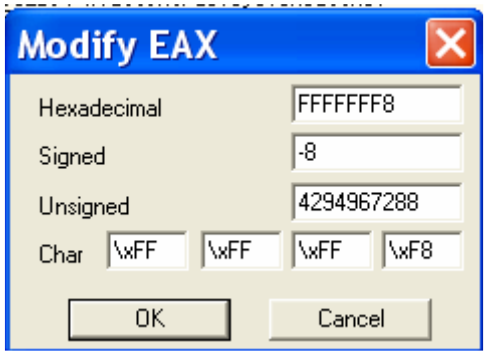




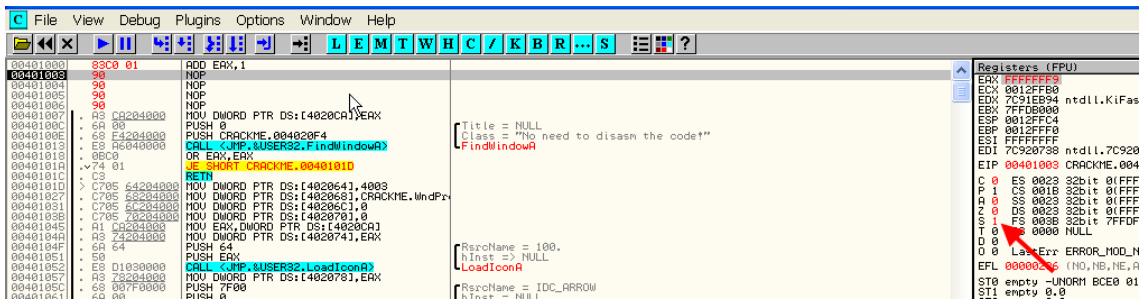
Creo que queda claro que dicho flag, se activa cuando el resultado de una instrucción es cero ya veremos diversas formas de activarlo mas adelante.

El FLAG S o FLAG DE SIGNO

Se activa cuando el resultado de una operación es negativo, o sea si quiero probarlo cambio EAX a FFFFFFF8 que es -8 decimal



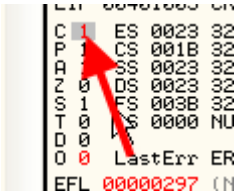
Y vuelvo con NEW ORIGIN HERE a mi ADD EAX,1 al apretar F7 y ejecutarlo, estoy sumando a -8 el valor 1, el resultado es FFFFFFF9 que es -7 decimal, el cual es negativo aun por lo cual debería activarse el flag de SIGNO probemos en OLLY.



Vemos que al apretar F7 y hacer la suma se activa el flag S de signo quedando a 1, queda claro como funciona, resultado negativo de una instrucción se activa el FLAG S.

EL FLAG C o CARRY FLAG

Se activa cuando se excede el máximo valor posible que se puede mostrar, si ponemos EAX a FFFFFFFF y le sumamos 1 como hicimos las veces anteriores veremos activarse el CARRY FLAG poniéndose a 1.



EL FLAG T , D e I

No los explicaremos por ahora pues son bastante complejos, si lo haremos mas adelante, no tiene mayor interés por ahora, ya que vamos a explicar las instrucciones mas sencillas, así que los dejaremos para mas adelante.

Bueno con esto tenemos una idea de que es cada registro y en que caso se activa cada FLAG, con esa información ya podremos en la tercera parte estudiar instrucción por instrucción ya que por ahora solo vimos la instrucción ADD para ayudarnos a comprender cuando se activaba cada FLAG.

Se que esta parte y la que viene son las mas indigestas de todas así que léanla con paciencia, practiquen con el OLLY activar los FLAGS al ejecutar nuestro ADD EAX,1 y nos vemos en la parte 3 de esta INTRODUCCION.
Es muy importante que queden bien grabados todos estos conceptos básicos, recomiendo leer practicar y releer hasta que no haya dudas.

Hasta la parte 4
Ricardo Narvaja
10 de noviembre de 2005