INTRODUCCION AL CRACKING CON OLLYDBG PARTE 6

COMPARACIONES Y SALTOS CONDICIONALES

En términos generales, las comparaciones entre dos operandos, y según el resultado de esa comparación la decisión si el programa saltara o no en un salto condicional posterior, suele ser lo primero que se menciona en cualquier tute básico que comienza desde cero de cracking.

Sabemos que si el programa nos pide un serial para registrarnos, en algún momento deberá decidir si es correcto no y para ello deberá comparar y realizar uno o varios saltos y la dirección adonde saltará variará según si pusiste bien el serial o no.

Ahora como funciona en profundidad la comparación y el salto es lo que veremos a continuación.

Sabemos porque lo hemos visto ya en partes anteriores de esta introducción, que según el resultado una instrucción se activan o desactivan los flags, el caso mas conocido es el flag que se activa cuando el resultado de una instrucción es cero el FLAG Z y hemos visto diferentes formas de activar los flags según el resultado de una instrucción.

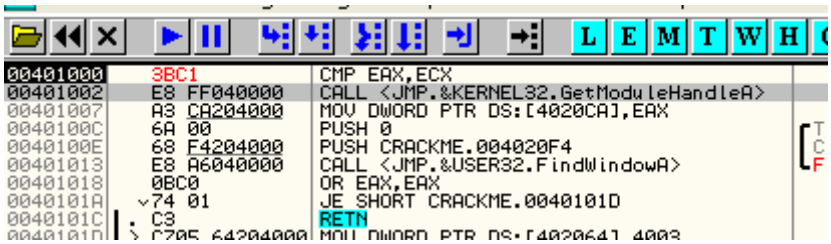
CMP

Esta es la instrucción mas conocida de comparación y lo que hace es comparar dos operandos, en realidad es una instrucción SUB, que no guarda el resultado de la resta en el primer operando, ambos operandos quedan igual, lo que cambian son los flags según el resultado.

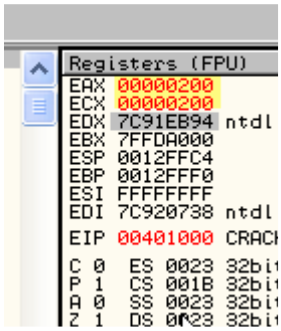
Como vimos si hacemos por ejemplo

CMP EAX, ECX

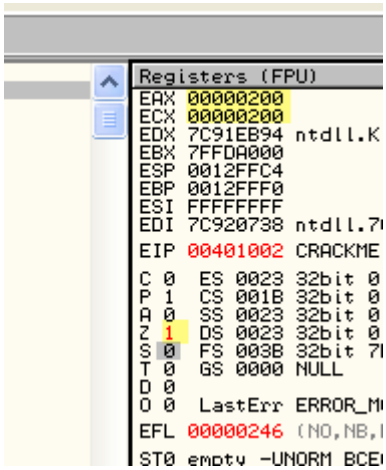
Siendo EAX y ECX iguales, se restan ambos, no se modifican, pero el resultado de la resta es cero lo que hace activar el flag Z, veamos el ejemplo en OLLYDBG.



Allí escribí la instrucción y ahora modificare EAX y ECX para que sean iguales.



A apretar F7 veo que EAX y ECX no modificaron su valor pero se activo el flag Z, al haber realizado una resta y ser su resultado CERO aunque no podamos ver el resultado ya que no lo guarda.



En realidad no nos importa el resultado numérico de la resta en si, sino que de alguna forma podemos saber según los flags si EAX era igual a ECX, o si son desiguales, cual es mayor.

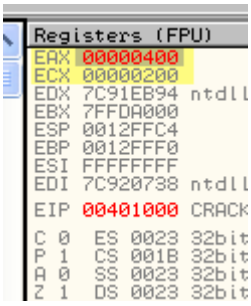
Como comentario ya que aun no hemos llegado a los saltos condicionales, los tienen dos posibilidades, deciden si saltar o no saltar según el estado de los flags, el mas claro ejemplo y que trabajaría en conjunto con la comparación anterior es el salto JZ que salta si el FLAG Z esta activo o a UNO y no salta si esta inactivo o a CERO.
De esta forma se logra que el programa tome una decisión, si fueran dos seriales que esta comparando, por ejemplo en EAX esta el serial que vos ingresaste para registrar un programa y en ECX esta el serial correcto, el programa podría decidir con un CMP que si son iguales se activa el flag Z, y el salto JZ al ver el flag activo, nos llevara a una zona para usuarios registrados, y si el serial que tipeaste y esta en EAX no es igual al de ECX que es el correcto, seguí intentando jeje, el flag Z sigue a CERO, y no salta manteniéndote en la zona de usuario no registrado.

Ya veremos los ejemplos más concretos al estudiar los saltos condicionales.

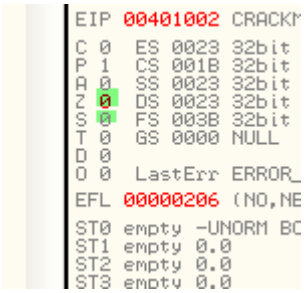
De la misma forma con el flag S o de signo podemos ver si en una comparación el primer operando era mayor que el segundo, o al revés.

Miremos el ejemplo

Repitamos el CMP EAX, ECX pero ahora pongamos EAX mayor que ECX

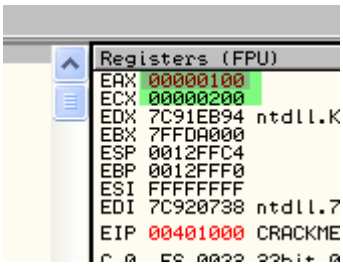


Si apretamos F7

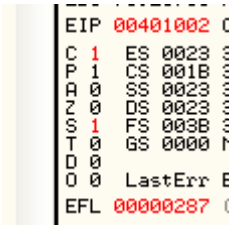


Vemos que el FLAG Z es cero por lo cual ya sabemos que no son iguales, y a la vez el flag S es cero lo cual quiere decir que al hacer la resta EAX-ECX el resultado es positivo, lo cual significa que EAX es mayor que ECX.

De la misma forma si repetimos la operación con EAX menor que ECX

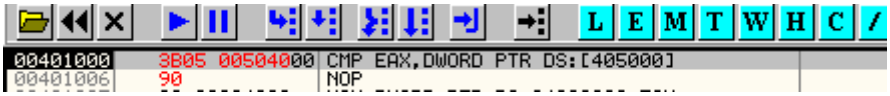


Y apretamos F7

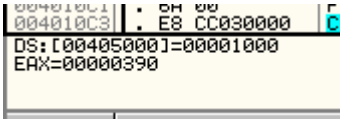


Allí vemos que al dar la resta de EAX con ECX negativa ya que ECX es mayor, se activa el FLAG S que recordamos se activa con resultados negativos.

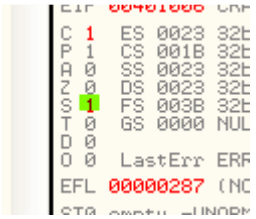
Las diferentes posibilidades de comparación activaran los correspondientes flags y según lo que el programa necesite saltara o no de acuerdo a chequear la activación de uno o mas de dichos flags.
También permite la comparación entre registros y posiciones de memoria utilizando DWORD, WORD y BYTE .



Allí compararía EAX con el contenido de 405000 y como siempre la ventana de aclaraciones del OLLY nos da el valor de cada operando en mi caso



Al apretar F7 en este ejemplo y EAX es menor que el contenido de 405000 que es 1000, dará un resultado negativo y activara el FLAG S.



Existen en forma similar

CMP AX,WORD PTR DS:[405000]

Y

CMP AL,BYTE PTR DS:[405000]

En el caso de comparar 2 bytes o 1 byte del contenido de la memoria.

TEST (Logical Compare)

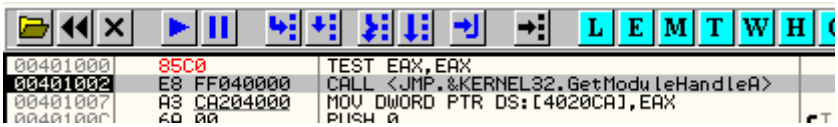
El principio de esta instrucción es, en cierto modo, el mismo de cmp, es decir, una operación entre dos valores que no se guarda, sino que puede modificar el estado de algunos flags (en este caso, SF, ZF y PF) que determinan si debe efectuarse el salto que también suele acompañar a esta instrucción. La diferencia está en que en este caso, en vez de tratarse de una resta, se trata de una operación AND.
Esto nos dice nuestro amigo CAOS en su tute de ASM veremos ejemplos para aclarar la definición, generalmente el comando TEST lo veremos en este formato

TEST EAX,EAX

Ustedes dirán para que quiere testearse contra si mismo el valor de EAX o el registro que sea?
Pues se usa esta instrucción, para saber si EAX en este caso es cero o no, y como funciona?

Escribamos en OLLY

TEST EAX,EAX



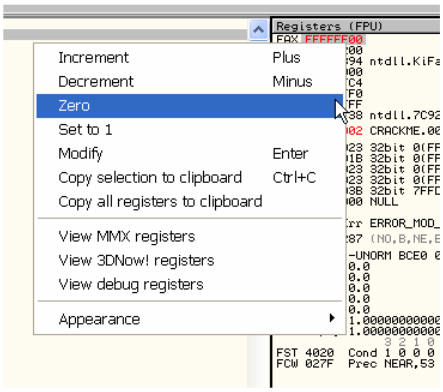
La tablita de la operación AND era

. El resultado es 1 si los dos operandos son 1, y 0 en cualquier otro caso.

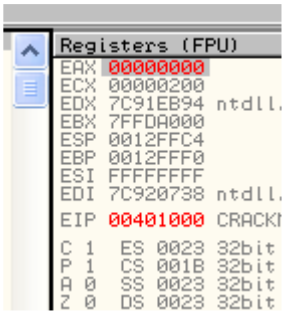
1 and 1 = 1
1 and 0 = 0
0 and 1 = 0
0 and 0 = 0

Vemos que la única forma que el resultado sea cero es que ambos operandos sean cero (no interesan en este caso los casos de bytes diferentes pues EAX esta operando contra si mismo, por lo que ambos bytes siempre son iguales) pues si EAX en binario tiene algún bit que es 1, al hacer AND contra si mismo daría 1 y ya no podría ser cero el resultado.

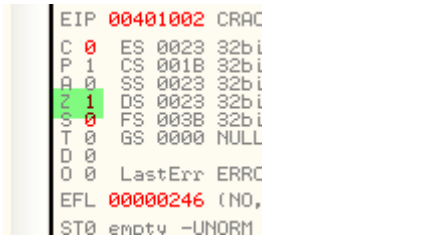
Pongamos EAX igual a CERO



En el OLLYDBG eso sea realiza fácilmente haciendo CLICK DERECHO en el registro a modificar y eligiendo ZERO.

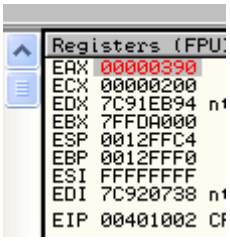


Allí esta apreto F7

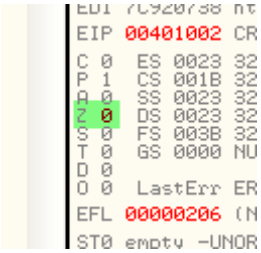


Vemos que se activo el flag Z o sea como sabíamos la operación AND entre dos valores que son cero es igual a cero y se activara.

Si repito la operación con EAX diferente de cero



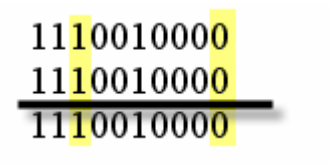
Apreto F7



Y el FLAG Z no se activa al ser el resultado diferente de cero.

Si uso la calculadora puedo comprobar que 390 AND 390 da como resultado 390

En binario 390 es 1110010000



Como la operación AND si ambos operandos son CEROS dará como resultado cero y si son dos UNOS dará como resultado UNO, vemos que el resultado no cambiará o sea será 390 en hexadecimal, y no se activara el flag Z.

Hay alguna instrucciones mas de comparación pero estas son las principales llego el momento de ver los saltos

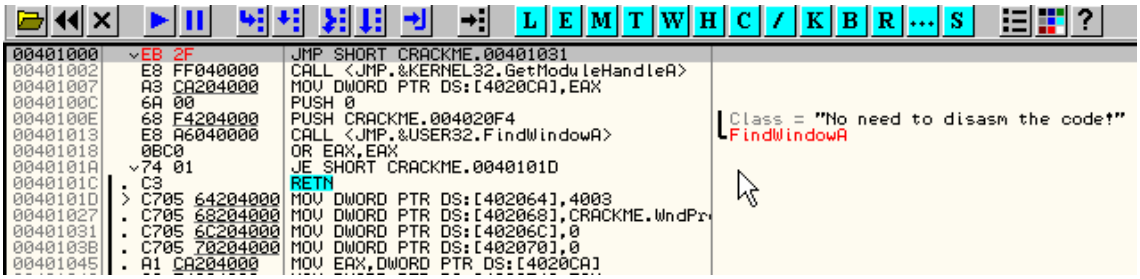
SALTOS

Todos as instrucciones de salto tienen un solo operando que es la dirección adonde saltaría el programa, veamos los diferentes tipos aquí en la tabla y aclarémoslos.

JMP	salta
JE, JZ	salta si es igual a cero
JNE, JNZ	salta si no igual a cero
JS	salta si signo negativo
JNS	salta si signo no negativo
JP, JPE	salta si paridad par
JNP, JOP	salta si paridad impar
JO	salta si hay capacidad excedida
JNO	salta si no hay capacidad excedida
JB, JNAE	salta si por abajo (no encima o igual)
JNB, JAE	salta si no está por abajo (encima o igual)
JBE, JNA	salta si por abajo o igual (no encima)
JNBE, JA	salta si no por abajo o igual (encima)
JL, JNGE	salta si menor que (no mayor o igual)
JNL, JGE	salta si no menor que (mayor o igual)
JLE, JNG	salta si menor que o igual (no mayor)
JNLE, JG	salta si no menor que o igual (mayor)

JMP

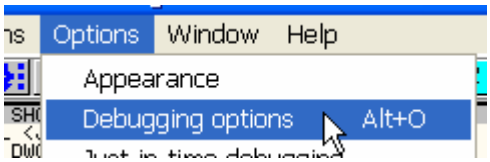
Es el salto directo o incondicional, aquí no hay ninguna decisión SIEMPRE saltara a la dirección que nos muestra el operando por ejemplo veamos en OLLY



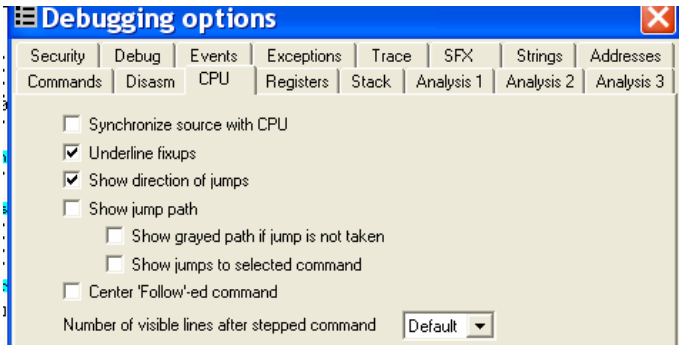
Al ejecutar ese salto el programa ira a 401031 y seguirá ejecutándose desde allí.

Tenemos un par de lindas configuraciones del OLLY para los saltos que les enseñare aquí, así son mas visibles los mismos.

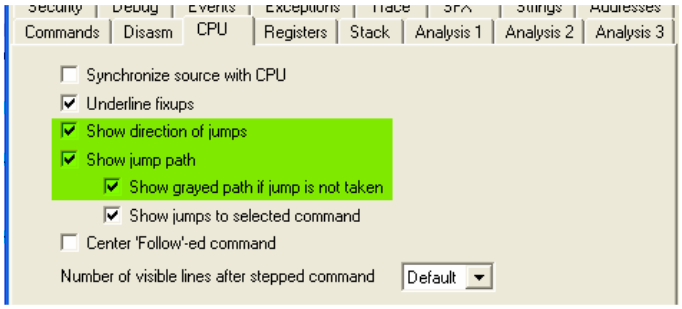
Si vamos a OPTIONS-DEBUGGING OPTIONS



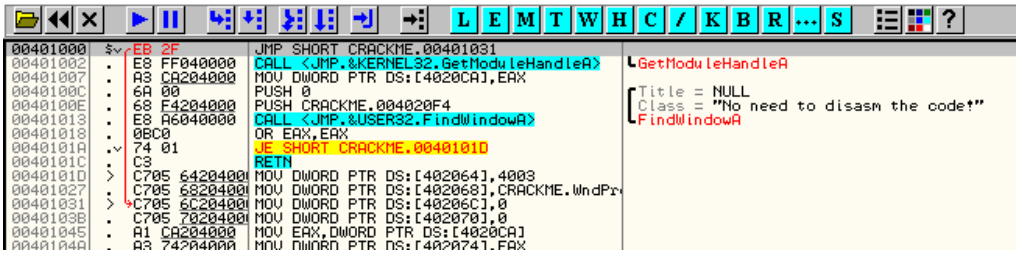
Y allí en la pestaña CPU



Marcamos estas tres tildes

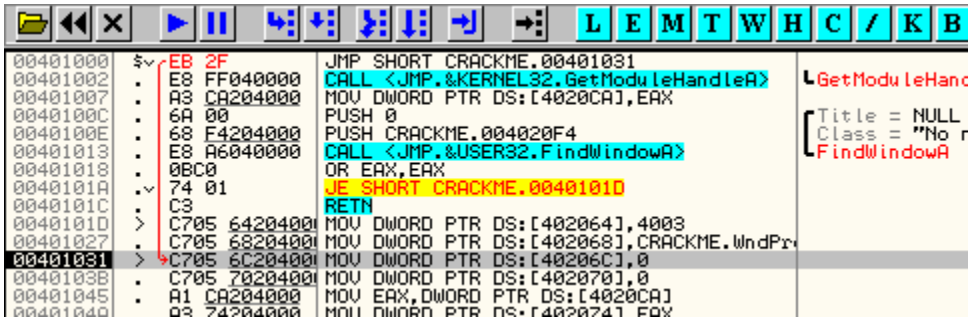


Vemos que la información es mucho mayor ahora y mas visible

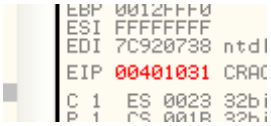


Vemos que OLLYDBG ahora nos muestra con una flecha roja que va a saltar y además a adonde va a saltar, exactamente a 401031.

Si ejecuto ahora con F7



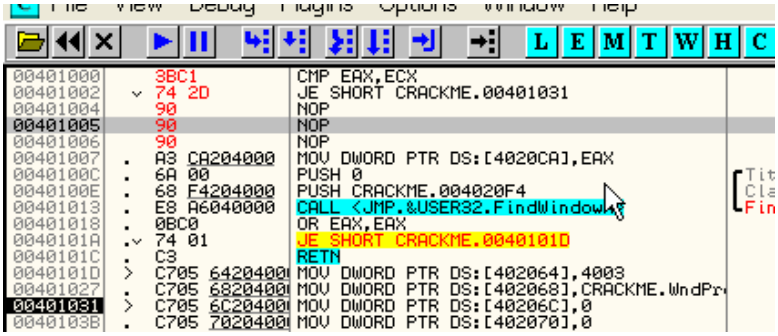
Se realizo el salto y EIP es 401031



Ya que EIP apunta a la instrucción que se va a ejecutar a continuación, en este caso 401031.

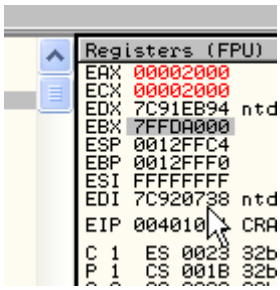
JE o JZ

Ambos son el mismo tipo de salto condicional y pueden escribirse de las dos formas ya vimos que JZ saltara cuando el flag Z sea cero.

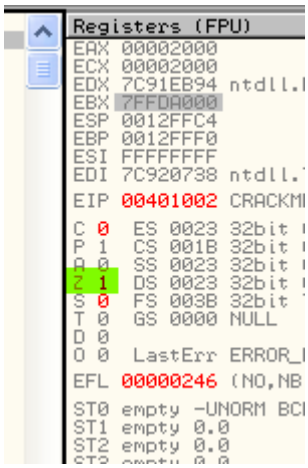


Escribamos en OLLYDBG dos instrucciones una comparación y el salto así verificamos como funciona.

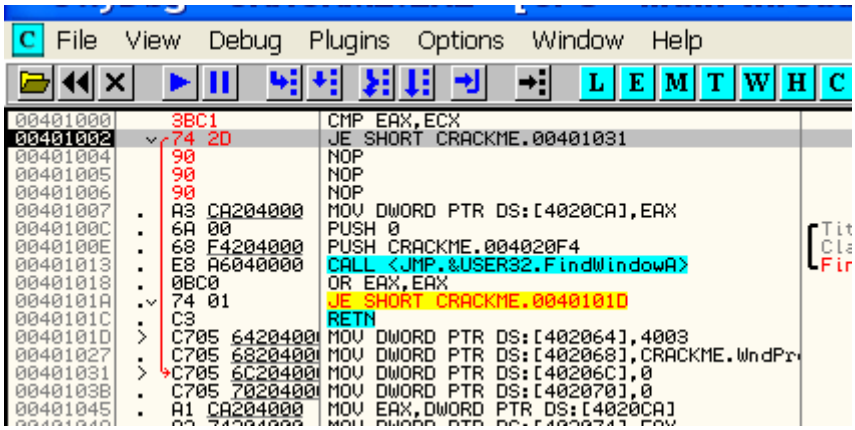
Pongo EAX y ECX iguales



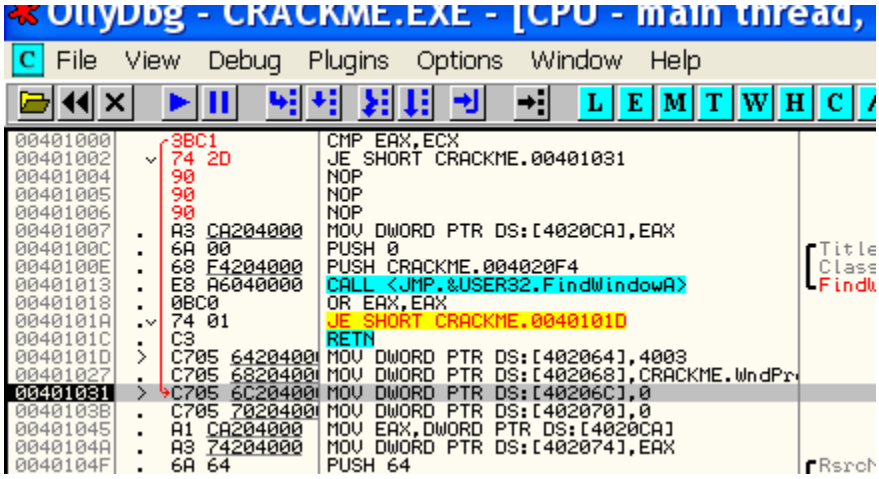
Al ejecutar la comparación se realiza la resta y como ambos son iguales se activa el FLAG Z al ser el resultado CERO.



La siguiente instrucciones es el salto condicional veamos

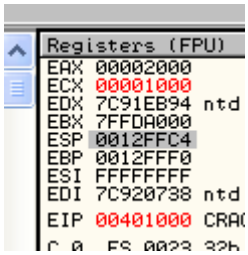


OLLYDBG ya nos avisa la decisión y como el FLAG Z esta activado nos muestra en rojo que va a saltar, si la indicación estuviera en gris quiere decir que la decisión es no saltar, en este caso saltara, apreto F7.

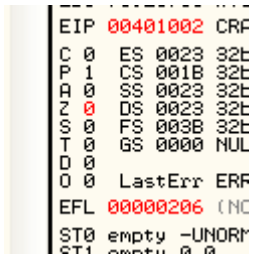


Allí vemos que salto y EIP ahora es 401031.

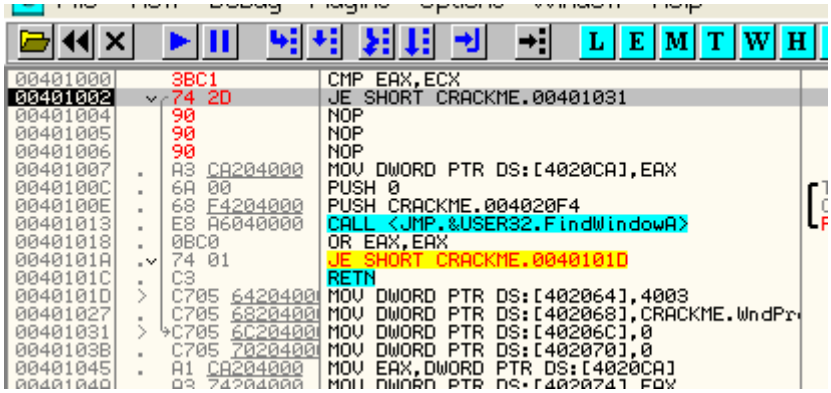
Repitamos el ejemplo con EAX diferente de ECX



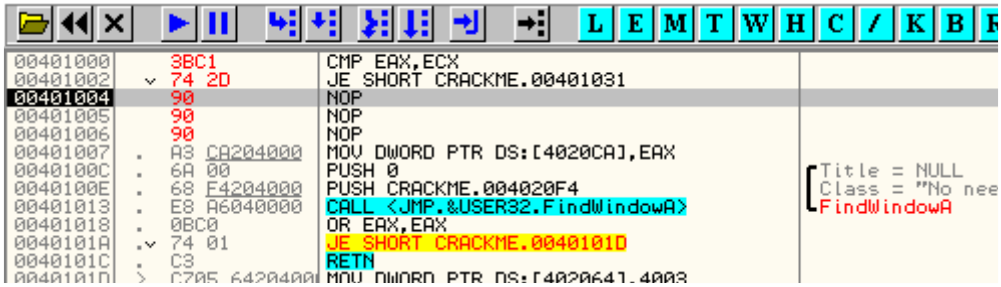
Al apretar F7 en la primera instrucción como el resultado no es cero, el flag Z no se activa.



Y vemos en OLLY

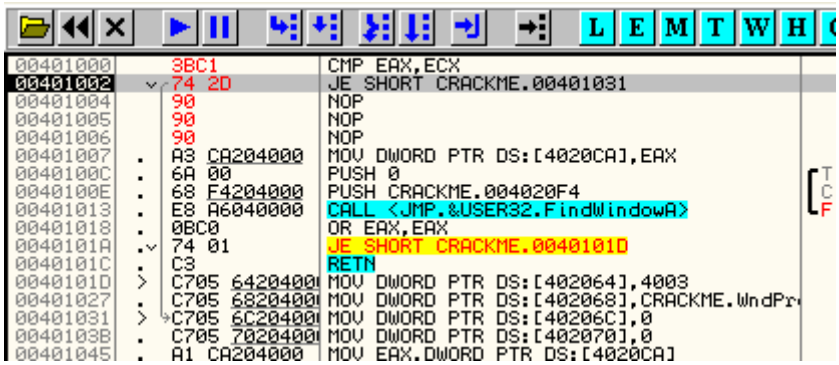


Que como el JE salta si el flag Z es UNO, en este caso no saltara, y la flecha del salto esta gris, al apretar F7 nuevamente

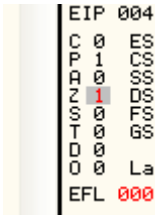


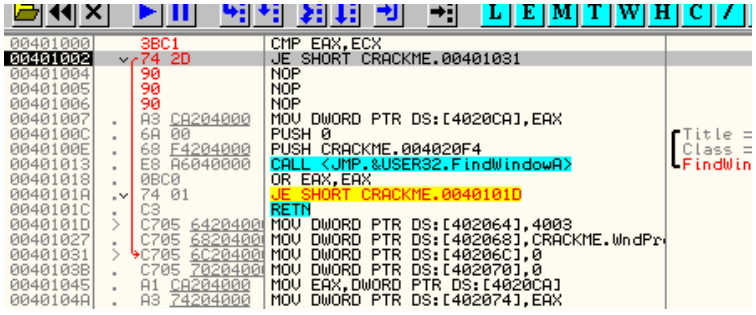
Vemos que no salto y siguió a continuación en 401004, esto que parece tan tonto es la base de la comparaciones y decisiones en todos los programas.

Si repito el ejemplo anterior y llego al salto, nuevamente y no va a saltar



Sabemos que no salta porque el flag Z esta a cero, ahora que ocurre si hago doble click en el flag Z y lo cambio a 1.





Vemos que la flecha cambio a rojo y OLLYDBG saltara, independientemente de la comparación, manipulando directamente el flag, ya que la decisión se toma sobre el estado actual del mismo, si cambia el flag cambiara el salto.

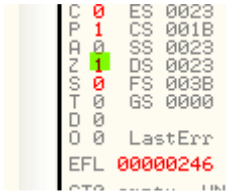
Los otros saltos los veremos rápidamente con un ejemplo cada uno

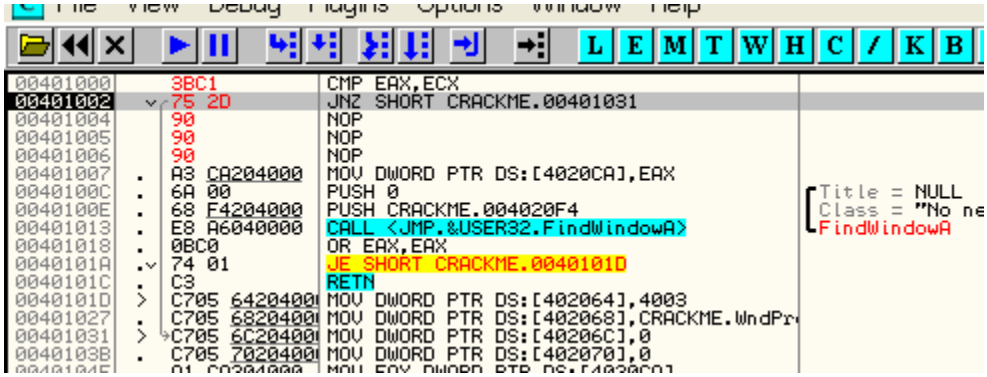
JNE o JNZ

Es el opuesto al salto anterior en este caso, salta si el flag Z no esta activo o sea si el resultado de la operación fue distinto de cero.

Allí escribo la comparación y el JNZ

Si EAX y ECX son iguales se activa el FLAG 0 al ser la diferencia CERO



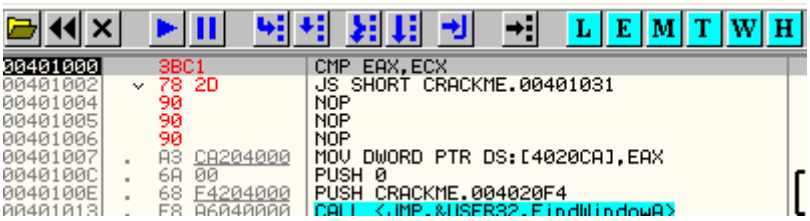


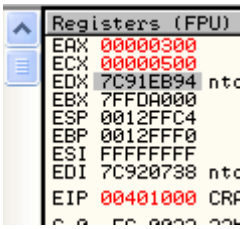
Y al revés que el JZ que saltaba al estar el FLAG Z activo este es el opuesto, salta cuando el FLAG Z es cero o esta inactivo.

Se puede entender que si pongo EAX diferente de ECX, el resultado será diferente de cero y el flag Z quedara inactivo y allí si saltara.

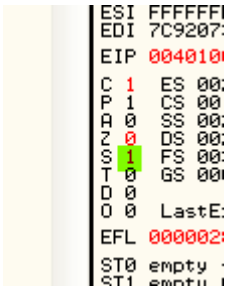
JS

Como vemos en la tabla saltara si la comparación da un resultado negativo o sea si EAX en menor que ECX en el ejemplo anterior.

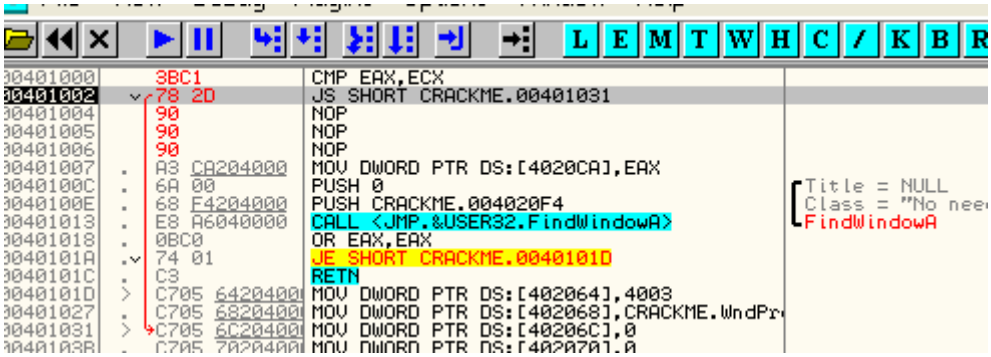




Al apretar F7



El FLAG S se pone a 1 y saltara al ser negativo,



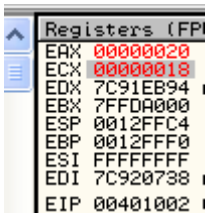
Allí vemos que la flecha roja nos indica que saltara, en el caso que EAX sea mayor que ECX el flag S no se activara al ser un resultado positivo y el salto JS no saltara.

JNS

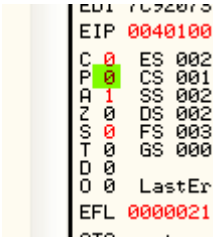
Es el opuesto al anterior saltara cuando el FLAG S este a cero o sea cuando el resultado sea positivo, en el ejemplo anterior cuando EAX sea mayor que ECX.

JP o JPE

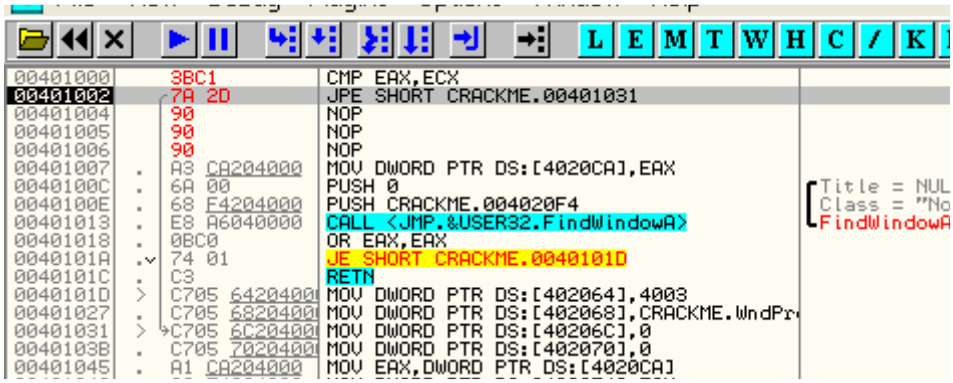
En esta caso el salto condicional JP saltara cuando el FLAG P este activo y esto ocurrirá como habíamos visto cuando el resultado de la comparación tenga paridad par o par cantidad de unos al verlo en binario.



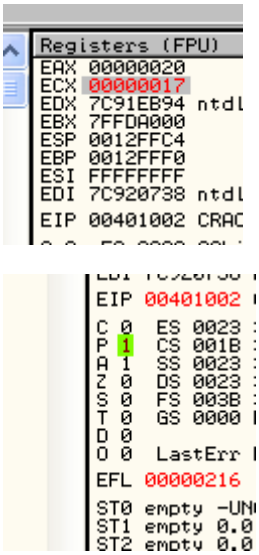
Ponemos EAX a 20 y ECX a 18 y apretamos F7



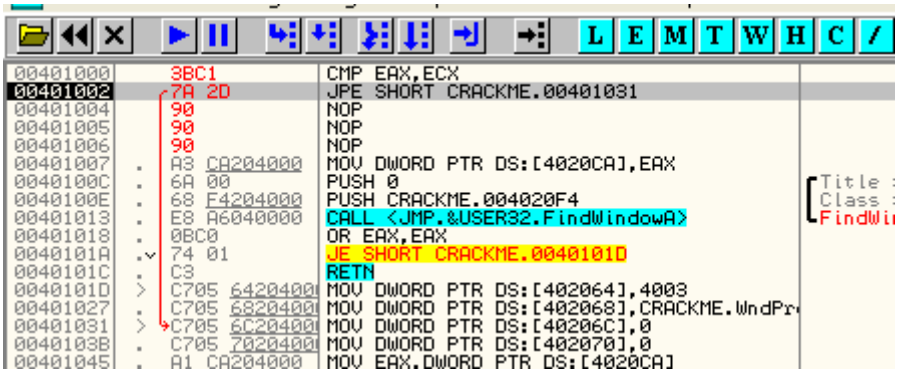
Como la diferencia entre EAX y ECX es 2 y este pasado a binario es 10 que tiene 1 solo uno o sea una cantidad impar de unos, el flag P no se activa y el JPE no saltara.



Ahora si cambio ECX a 17 y repito apreto F7



Vemos que al ser el resultado 3 que en binario es 11 y tiene un número par de unos, entonces allí si se activa el FLAG P y el JPE saltara.

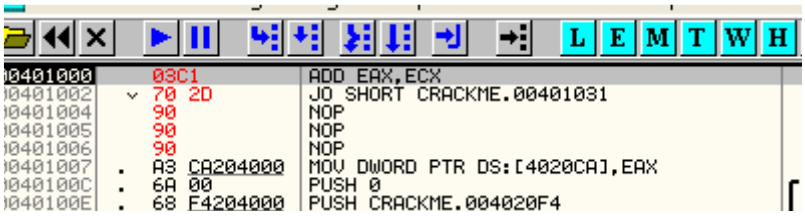


JNP o JNPE

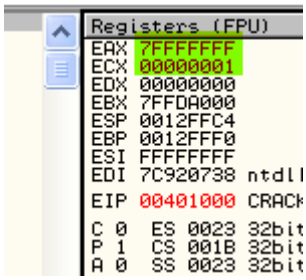
Es el opuesto del anterior o sea este salta cuando el flag P esta a cero o sea la paridad es impar, en el ejemplo anterior hubiera saltado la primera vez cuando el resultado era 2 y no hubiera saltado cuando el resultado era 3, al revés del JP.

JO

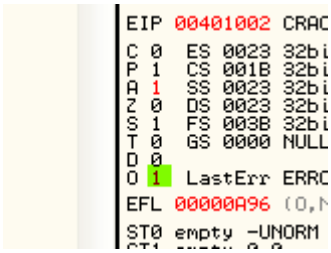
Este salta cuando hay overflow o capacidad excedida lo cual activa el flag O.

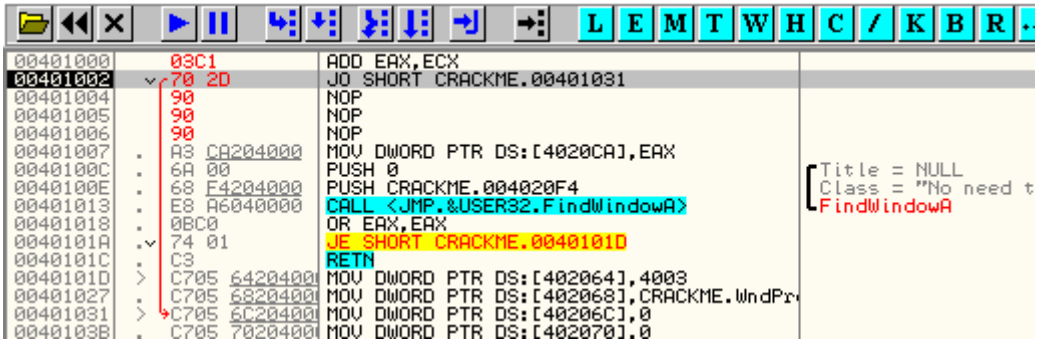


Aquí cambiamos la comparación porque para activar el FLAG O hay que hacer OVERFLOW y esto es posible mediante una suma.



Apreto F7





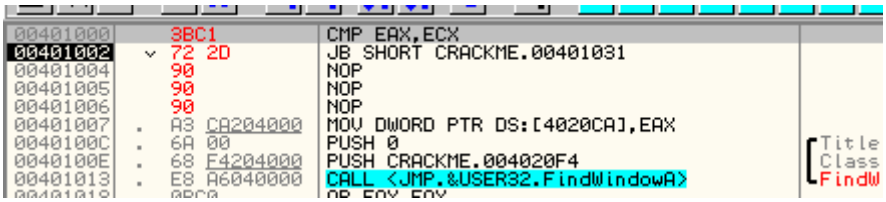
Y el JO saltara al haberse activado el FLAG O por OVERFLOW o capacidad excedida.

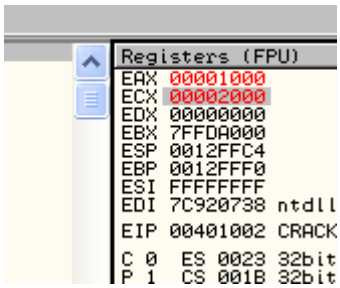
JNO

Es el opuesto al anterior este salta cuando el FLAG O esta a CERO o sea no hay OVERFLOW

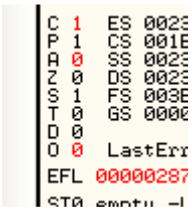
JB

Salta si es mas bajo, veamos el ejemplo





Vemos que EAX es mas bajo que ECX o sea que debería saltar, al apretar F7



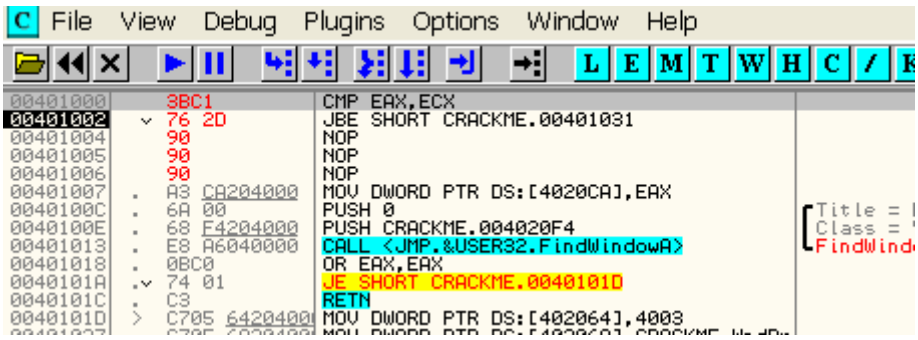
El flag C se activa, ya que tiene que al hacer la diferencia que da un numero negativo, en el bit mas significativo o sea el primero habrá acarreo y eso activa el FLAG C, y según eso decide el JB en resumidas cuentas si EAX era menor que ECX.

JNB

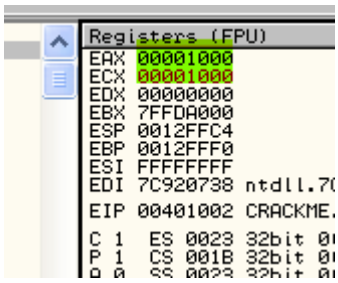
Es el opuesto de JB saltara si el FLAG C es cero o sea no hay acarreo porque el resultado fue positivo, lo cual supone que EAX fue mayor que ECX y al revés que el anterior este saltara en ese caso.

JBE

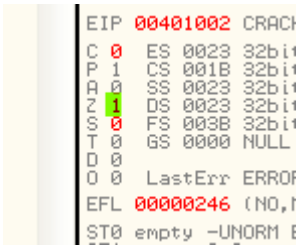
Este salta si es mas bajo o igual o sea testea dos flags a la vez ve si el FLAG C esta activo en ese caso salta, y también verifica si el FLAG Z esta activo con el cual también salta, o sea si EAX es igual a ECX saltara y si es menor también.



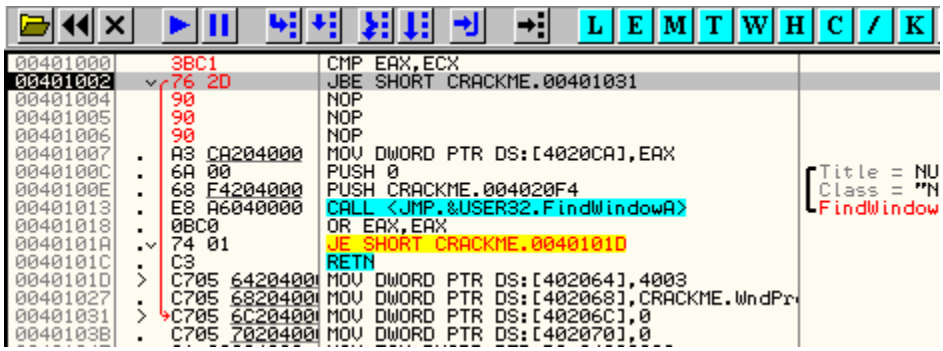
Ponemos como primer caso que EAX y ECX sean iguales



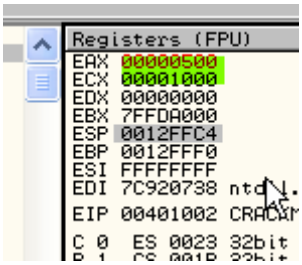
Apreto F7



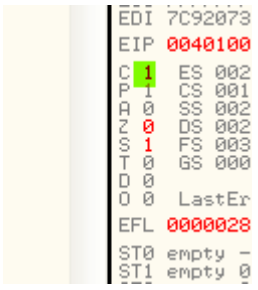
Al ver que el flag Z esta activo salta



Si EAX es menor que ECX

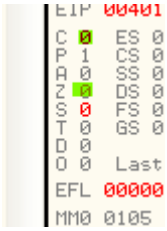


Apreto F7

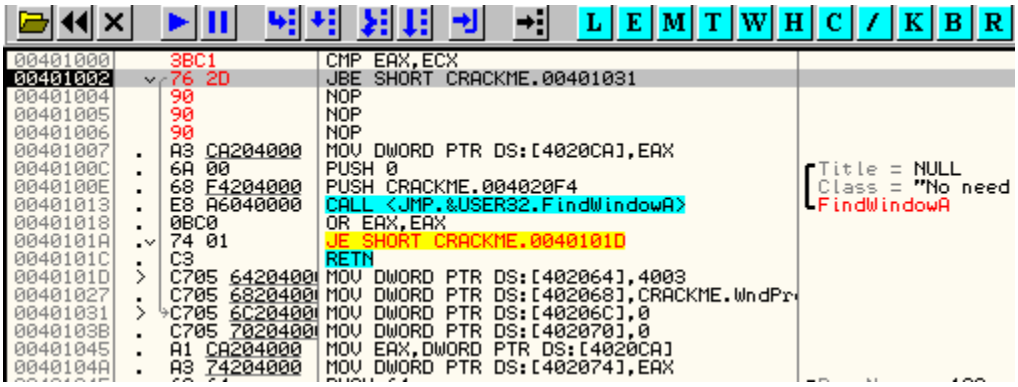


En este caso se activa el FLAG C al ser resultado negativo que usa un carry en el bit mas significativo, o sea salta cuando EAX es menor a ECX también

En el ultimo caso que EAX es mayor a ECX repito el ejemplo apreto F7



Ambos flags Z y C están a cero por lo tanto no saltara



O sea la conclusos es que JBE salta si EAX es mas bajo o igual que ECX en el ejemplo.

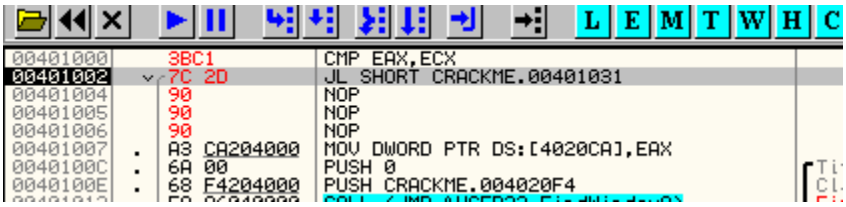
JNBE

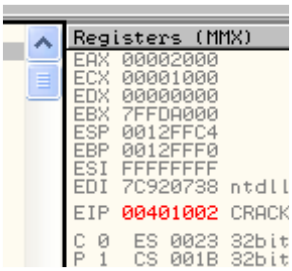
Es el opuesto al anterior, salta si el flag Z y el FLAG C son cero ambos o sea solo en el ultimo ejemplo anterior cuando EAX es mayor que ECX, allí ambos flags están a cero y este JNBE saltara.

JL

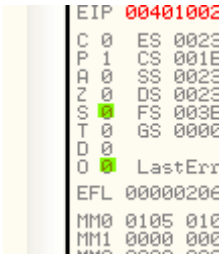
en este caso JL salta si es menor, pero en diferente forma, veamos aquí mira si FLAG S es diferente a FLAG O y en ese caso salta

Veamos en los ejemplos cuando se da esto si ambos EAX y ECX son positivos siendo EAX mayor que ECX



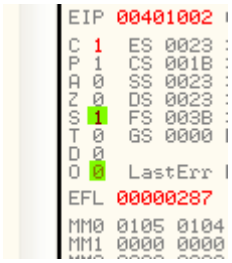


Al apretar F7 no salta ya que EAX es mayor que ECX y el resultado es positivo lo que no activa el FLAG O ni el FLAG S.

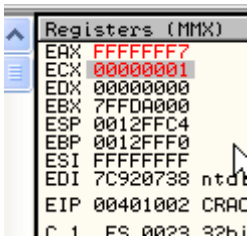


Si EAX es menor que ECX pero ambos son positivos repito el ejemplo

Apreto F7

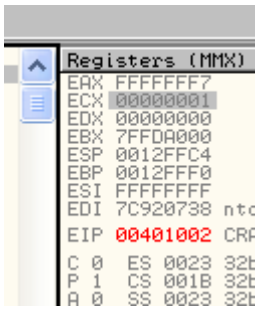


Y como S y O son diferentes salta o sea que salta cuando es menor si ambos son positivos, veamos otro caso.

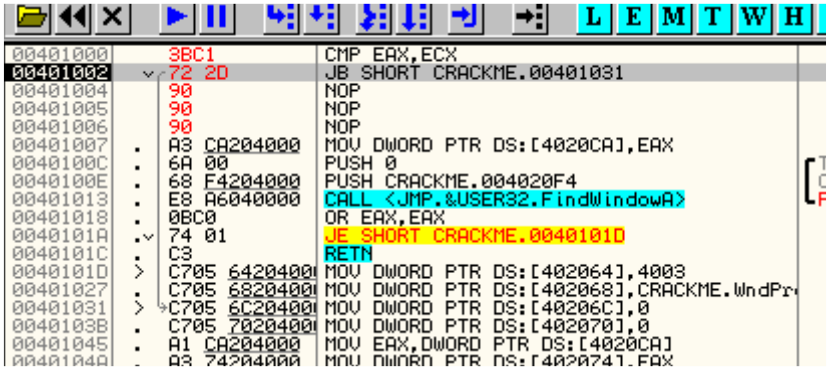


En este caso EAX es menor que ECX, ya que es un número negativo veamos que pasa

Salta perfectamente porque es menor pero que pasa si usamos estos dos mismos valores con el otro salto que parece realizar el mismo trabajo el JB



Al apretar F7



Vemos que el JB no salta quiere decir que JB compara ambos como si fueran positivos o SIN SIGNO mientras que si el salto debe considerar el signo de lo que compara se utiliza JL esa es la principal diferencia entre ambos.

	La comparación se realiza con	
Para saltar si	**Números sin signo**	**Números con signo**
El destino es más grande que la fuente	JA	JG
Destino es igual a la fuente	JE	JE
El destino no es igual a la fuente	JNE	JNE
El destino es menor que la fuente	JB	JL
El destino es menor o igual a la fuente	JBE	JLE
El destino es más grande o igual a la fuente	JAE	JGE

Aquí vemos en esta los saltos condicionales según queramos considerar el signo de lo que comparamos o no,

Vemos que JA, JB JBE y JAE consideran ambos operandos positivos mientras que JG, JL, JLE y JGE consideran los operandos con signo, JE y JNE se comportan igual en ambos casos.

Creo que con esto hemos aclarado el tema de los saltos condicionales y las comparaciones, en la práctica mas adelante veremos su utilización en los programas lo cual nos será mucho mas liviano que esta pesada tarea de revisarlos a casi todos.
Queda en la ultima parte de ASM por suerte ver los calls y ret, y los modos de direccionamiento ya casi llegamos al fin de lo duro paciencia jeje..
Hasta la parte 7
Ricardo Narvaja
16 de noviembre de 2005