TUTORIAL: AÑADIR NUEVA FUNCIONALIDAD A UN JUEGO DE NES

Este proceso es un poco complicado y en algunos juegos no se puede realizar.

Para el ejemplo, vamos a utilizar el juego “Adventures of Dino Riki, The (U) [!]” que no permite pasar de nivel utilizando una combinación de botones pero sin embargo, podríamos añadir dicha funcionalidad.

El juego que he utilizado tiene una cabecera de 16 bytes ($10).

En este juego se utiliza la dirección $0005 de la RAM para almacenar el botón que hemos pulsado el juego con los siguientes valores:

01: Derecha

02: Izquierda

04: Abajo

08: Arriba

10: Start

20: Select

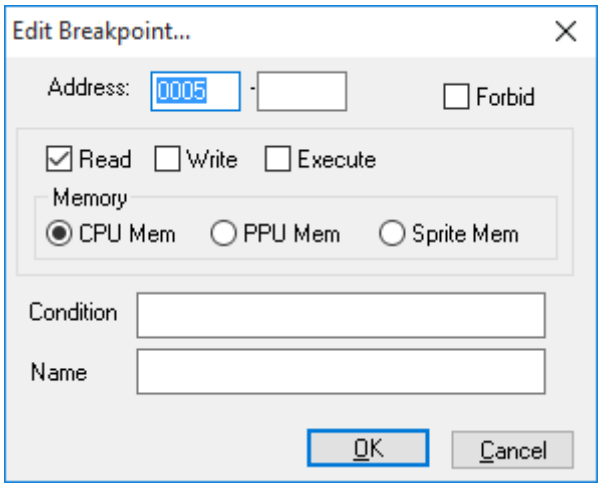
40: B

80: A

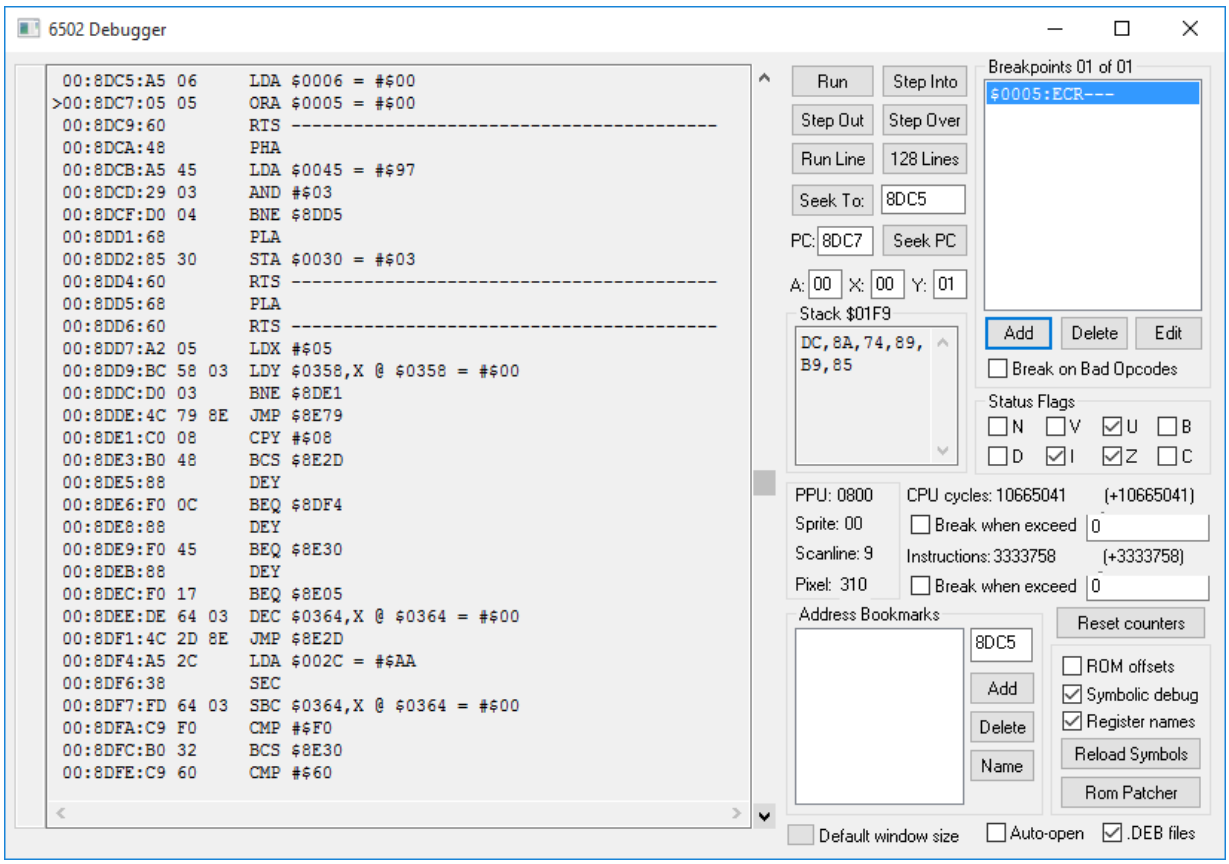
La combinación de botones “A”+”B” correspondería al valor $40+$80=$C0 en hexadecimal. Vamos a hacer una modificación en el juego para que cuando se pulse dicha combinación de botones pasemos de nivel.

Vamos a cargar el juego con el emulador FCEUX 2.2.3 y vamos a empezar en el nivel:

Vamos a poner un breakpoint cuando se lea el valor que hay en la dirección $0005 para conocer el fragmento de código donde se lee el botón pulsado en el juego.



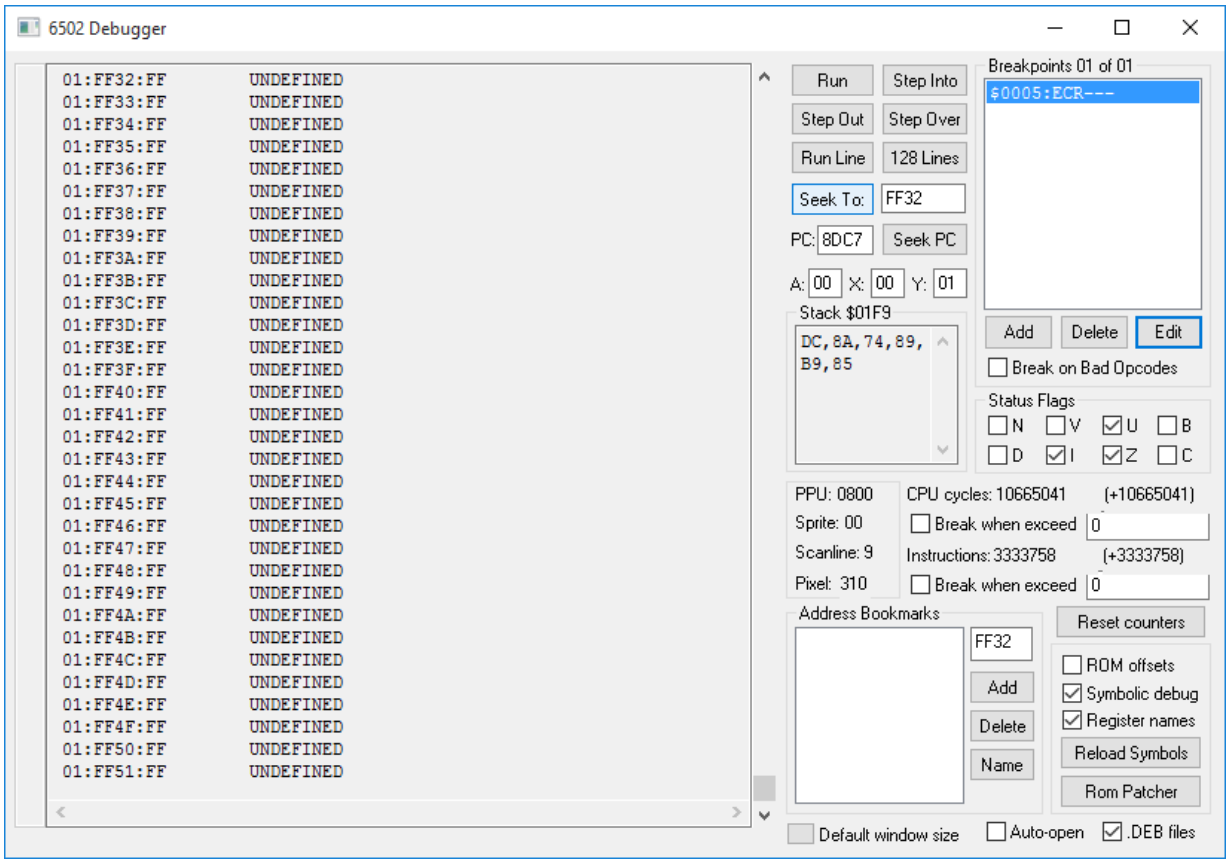
Si seguimos jugando veremos cómo pronto se para el emulador y nos aparece la siguiente pantalla del “Debugger”:



Observamos cómo se para la ejecución en $00:8DC5 donde está el cursor “>”.

Puede que la dirección $0006 también tengo que ver con el botón pulsado.

Vamos a buscar un área que pueda estar vacía para poner añadir un fragmento de código. En $01:FF32:



Aquí parece que hay espacio suficiente en blanco para poner la rutina corta que vamos a utilizar. Deberíamos asegurarnos que estos valores no corresponden a datos del juego aunque normalmente el final de los bancos con valores $FF o $00 se utilizan para llenar el espacio en blanco.

La rutina que aparece en $00:8DC5:

00:8DC5:A5 06 LDA $0006 = #$00

00:8DC7:05 05 ORA $0005 = #$00

00:8DC9:60 RTS ----------

Vamos a cambiar parte de las instrucciones de esta rutina para que salte a la dirección $01:FF32 quedando de esta forma:

00:8DC5:20 32 FF JSR $FF32

00:8DC8:60 RTS ----------

00:8DC9:60 RTS ----------

Hemos puesto en $00:8DC5 una instrucción para que salte a la subrutina que vamos a crear que se encuentra en $01:FF32.

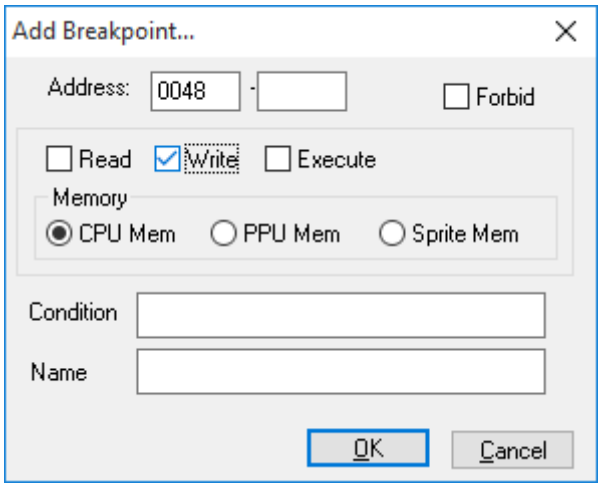
Como las 2 primeras instrucciones se codifican como “A5 06 05 05” y la instrucción “JSR FF32” que hemos puesto se codifica “20 32 FF” necesitamos añadir un byte adicional para lo cual hemos puesto “60” que equivale a la instrucción “RTS” (Regresa de la subrutina) para que vuelva a la rutina que llamo a ésta.

También podríamos haber puesto “00” que corresponde a la instrucción NOP (Not OPeration) que equivale a no hacer nada con lo que pasaría a la siguiente instrucción que hay en $00:8DC9 que es un “RTS” para regresar de la subrutina.

De las dos formas se obtendría el mismo resultado.

Este juego almacena en la dirección $0048 de la RAM el nivel donde nos encontramos correspondiendo el valor $00 al nivel 1-1, $01 al enemigo final del nivel 1 (Pterodáptilo),…

Si jugamos al juego y ponemos un breakpoint en el emulador cuando se escriba en la dirección $0048 para obtener el fragmento de código donde se incrementa el nivel.



Cuando pasamos de nivel el emulador se para en la dirección $00:86B3 del código fuente:

00:86B3:E6 48 INC $0048 = #$04

00:86B5:A6 48 LDX $0048 = #$04

00:86B7:E0 0E CPX #$0E

Ahí se utiliza “INC $0048” para incrementar el valor que hay en la dirección $0048=Nivel.

Sin embargo voy a irme un poco más arriba de este código:

00:869C:20 5A 87 JSR $875A

00:869F:A9 01 LDA #$01

Voy a poner un salto a la dirección $00:869F después de la instrucción “JSR”.

Ahora que ya hemos modificado la rutina en $00:8DC5 tenemos que poner la rutina en $01:FF32 que permite que cuando pulsemos “A”+”B” pasemos de nivel.

01:FF32:A5 05 LDA $0005 = #$00

01:FF34:C9 C0 CMP #$C0

01:FF36:D0 03 BNE $FF3B

01:FF38:20 9F 86 JSR $869F

01:FF3B:A5 06 LDA $0006 = #$00

01:FF3D:05 05 ORA $0005 = #$00

01:FF3F:60 RTS ----------

Lo que hace esta rutina es leer el valor que hay en la dirección $0005 utilizando la instrucción “LDA” y compara dicho valor con “C0” que corresponde a “A”+”B” (A: $80, B: $40. A+B=$80+$40=$C0).

Si no es igual saltaría a la instrucción en $01:FF3B que es realmente la rutina que había originalmente en el juego para que no se modifique como actúa el juego cuando se pulse cualquier otro botón.

Pero si es igual a $C0 el valor que hay en la dirección $0005=Botón pulsado se pasaría a la instrucción que hay en $01:FF38 en la que tenemos la instrucción “JSR $869F” que saltaría a la subrutina que hay en la dirección $00:869F que se encarga de incrementar en nivel.

Una vez guardada la modificación en el juego podemos ver como si se pulsa dicha combinación de botones pasamos de nivel.

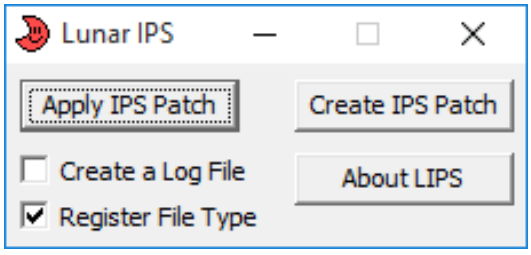
Para la modificación podemos utilizar un editor hexadecimal.

La rutina que lee el botón en $00:8DC5 corresponde a los bytes $DD5 (al tener cabecera de $10 bytes) en adelante.

La rutina que hemos añadido en $01:FF32 corresponde a los bytes $7F42 en adelante.

Adicionalmente podríamos crear un fichero .IPS para parchear el juego original para distribuir dicho parche en páginas como Romhacking.net.

Para hacerlo podemos utilizar la herramienta “Lunar IPS” que encontramos en Romhacking.net.



Para hacerlo basta con seleccionar la opción “Create IPS Patch” y seleccionar el juego original sin cambios, luego el juego que hemos modificado y finalmente donde queremos guardar el fichero .IPS que es el parche para modificar el juego.

Lo deseable cuando hacemos esta modificación es que el fragmento que utilizamos para poner la rutina se encuentre en el mismo banco pero puede que en el banco $00 no haya espacio en blanco por lo que he puesto la rutina en el banco $01.

Como en el banco $00 en este juego se utilizan las direcciones $8000-$BFFF y para el banco $01 las instrucciones $C000-$FFFF no va a haber problemas ya que en el banco $00 no hay dirección $FF32 por lo que el juego al saltar a la subrutina asume que es la que se encuentra en $01:FF32 al igual que cuando salta a $869F asume que es a $00:869F y cuando salta a $869F asume que es a $00:869F.