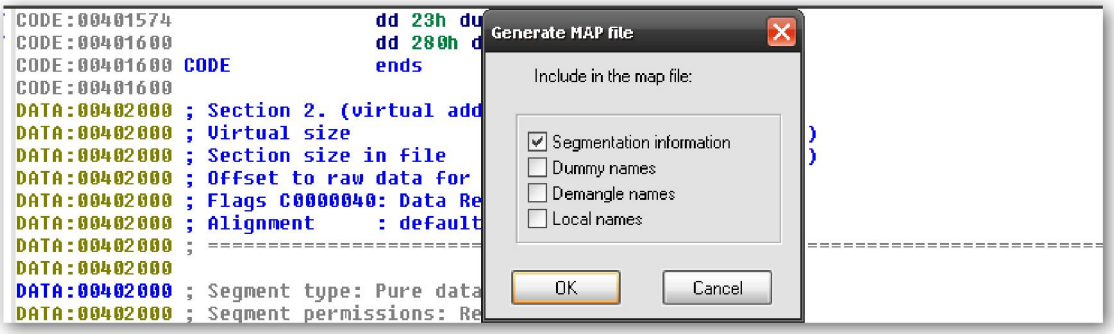
13.2.—Archivos de salida de IDA y generación de parches

Una de los menús más interesante de IDA es el menú **File > Produce File**. Según las opciones de dicho menú, IDA puede generar archivos tipo **MAP, ASM, INC, LST, EXE, DIF y HTML**. Estos tipos los describiremos seguidamente.

13.2.1.—Generar archivos MAP

Un archivo **.map** describe el esquema completo de un binario, lo cual incluye información sobre las secciones que componen el binario y la ubicación de los símbolos dentro de cada sección. Cuando generemos un archivo **.map**, se nos preguntará por el nombre que le daremos al archivo y los tipos de símbolos que queremos guardar en el archivo. En la siguiente figura, mostramos las opciones del diálogo de **MAP file**, en el cual seleccionaremos la información que queramos incluir en el archivo **.map**.



La información de las direcciones en un archivo **.map** se representan utilizando direcciones lógicas. Una dirección lógica describe la ubicación de un símbolo utilizando un número del segmento y un offset del segmento. Las primeras líneas de un archivo **.map** se muestran a continuación. En este listado vemos seis segmentos y los dos primeros símbolos.

```
Start         Length     Name                   Class
0001:00000000 000001000H CODE                   CODE
0002:00000000 000001000H DATA                   DATA
0003:00000000 000001000H .idata                 DATA
0004:00000000 000001000H .edata                 DATA
0005:00000000 000001000H .reloc                 DATA
0006:00000000 000002000H .rsrc                  DATA


 Address         Publics by Value

0001:00000000       start
0001:00000128       WndProc
```

La dirección lógica de **WndProc** indica que se encuentra en el **byte Offset 128h** dentro del primer segmento **CODE**.

Los archivos MAP generados por IDA son compatibles con Turbo Debugger de Borland. El propósito principal de los archivos .map es restituir los nombres de símbolos cuando depuramos binarios en los cuales pueden haberse eliminado dichos nombres o en aplicaciones que han sido compiladas de alguna forma especial y no los muestran.

13.2.2.—Generar archivos ASM

IDA puede generar un archivo **.asm** de la base de datos que tenemos cargada. La idea es crear un archivo que pueda ejecutarse a través de un ensamblador para recrear el archivo binario. Ida intentará volcar la información suficiente, entre otros los esquemas de

estructuras, para lograr realizar un buen desensamblado. Para lograr generar un buen archivo **.asm** depende de distintos factores, el más importante es que nuestro ensamblador comprenda la sintaxis utilizada por IDA.

El lenguaje ensamblador utilizado se determina con la opción **Target assembler** la cual se encuentra en la solapa **Analysis** del menú **Options > General menu**. Por defecto IDA generará un archivo ensamblado de toda la base de datos. Sin embargo, podemos limitar el alcance del listado "clickando" y arrastrando o utilizando flecha SHIFT-arriba o flecha SHIFT-abajo para desplazarnos y seleccionar la región que queremos volcar. En las versiones consola de IDA podemos utilizar la órden **Anchor o ALT-L** para habilitar el punto de inicio de la región seleccionada y utilizar las teclas de flechas para extender el tamaño de la región a seleccionar.

13.2.3.—Generar archivos INC
Un archivo **INC (include)** contiene definiciones de estructuras de datos y tipos de datos numerados. Este es esencialmente el volcado del contenido de la ventana **Structures** de una forma adecuada para su utilización por un ensamblador.

13.2.4.—Generar archivos LST
Un archivo **LST** no es nada más que en archivo de texto del volcado del contenido de la ventana de desensamblado de IDA. También podemos escoger el rango de direcciones deseado del volcado, de la misma forma que se realiza con el archivo ASM.

13.2.5.—Generar archivos EXE
Aunque esta es la opción más prometedora, también es la más "capada". En pocas palabras no funciona con la gran mayoría de tipo de archivos, y si es así recibiremos un mensaje de error **"This type of output file is not supported".**

Esta podría ser la característica ideal para parchear binarios, pero en general es muy difícil regenerar un archivo ejecutable desde una base de datos de IDA. La información presente en una base de datos de IDA está compuesta principalmente por el contenido de las secciones que componen el archivo fuente original. En muchos casos, sin embargo, IDA no procesa todas las secciones de un archivo fuente y cierta información se pierde cuando el archivo es cargado en la base de datos, por lo tanto realizar la generación de un ejecutable desde una base de datos es imposible. El ejemplo más simple de dichas pérdidas es el hecho de que IDA no carga, por defecto, la sección de recursos **.rsrc** de un archivo **PE**, lo cual hace imposible la restauración de dicha sección desde la base de datos.

En otros casos, IDA procesa la información de los binarios originales pero no se puede acceder a ella de la misma forma que en el original. Como ejemplos citaremos la tabla de símbolos, tabla de importación y tabla de exportación, las cuales requerirían un gran esfuerzo para su reconstrucción correcta a fin de generar un ejecutable funcional.

Un producto que proporciona la capacidad de generar EXE son los **pe_scripts** para IDA de **Atli Mar Gudmundsson**. Estos son un conjunto de scripts de IDA para funcionar con archivos PE. Uno de estos scripts es el llamado **pe_write.idc**, el cual tiene como fin volcar una imagen del archivo PE desde una base de datos del archivo. Si nuestra intención es parchear un archivo PE con los scripts, tendremos que realizar los siguientes pasos:

1. Cargar el archivo PE en IDA. Asegurarnos que tenemos deseleccionada la opción **Make imports section** en el diálogo de carga.

2. Ejecutar el script **pe_sections.idc** para mapear todas las secciones del binario original en la nueva base de datos.

3. Realizar cualquier cambio que deseemos en la base de datos.

4. Ejecutar el script **pe_write.idc** para volcar el contenido de la base de datos a un nuevo archivo PE.

La realización de scripts **IDC** la estudiaremos más adelante.

13.2.6.—Generar archivos DIF

Un archivo **DIF** de IDA es un archivo de texto plano con el listado de todos los bytes que han sido modificados dentro de una base de datos. Es el formato más útil si nuestra meta es parchear un binario original basándonos en los cambios hechos en la base de datos de IDA. El formato del archivo es muy simple, veamos un ejemplo

```
This difference file is created by The Interactive Disassembler

CRACKME.EXE
000002F8: 83 FF
000002F9: EC 75
000002FA: 04 EC
000002FB: FF 68
```

El archivo incluye una línea de comentario como encabezado seguido del nombre del archivo original y el listado de los bytes cambiados dentro del archivo. Cada línea de cambio especifica el Offset del archivo, no la dirección virtual, del byte cambiado, el valor original del byte y el valor actual de dicho byte en la base de datos. En este ejemplo, la base de datos para CRACKME.EXE ha sido modificada en cuatro ubicaciones correspondientes a los Offset de los byte **0x2F8-0x2FB** dentro del archivo original. Es una tarea trivial para parchear un programa el analizar los archivos **.dif** y aplicar los cambios al archivo original para generar una versión parcheada de dicho binario.

13.2.7.—Generar archivos HTML

IDA aprovecha las capacidades disponibles con **HTML** a fin de generar listados de desensamblado coloreados. Un archivo HTML generado por IDA es esencialmente un archivo LST con etiquetas HTML lo cual nos produce un listado coloreado similar al de la ventana de desensamblado de IDA. Por desgracia, el archivo generado no contiene ningún hiperenlace con el cual poder navegar fácilmente utilizando el listado de texto.