HISOFT DEVPAC

ENSAMBLADOR/DESENSAMBLADOR PARA SPECTRUM +3

VERSIÓN ESPAÑOLA POR MASTRESS SOFT

HISOFT GENS4

ENSAMBLADOR/EDITOR

CONTENIDO

SECCION 1 EMPEZANDO A MANEJAR GENS4

1.1 Introducción e instrucciones de carga

GENS4 es un potente y fácil de usar ensamblador del microprocesador Z80 el cual es muy parecido en definición al ensamblador estándar Zilog. Al contrario que muchos otros ensambladores disponibles para el Spectrum. GENS4 es una completa y profesional pieza de software y te deberías estudiar las siguientes secciones, junto al ejemplo del Apéndice 3, muy cuidadosamente antes de comenzar a usar el ensamblador. Si eres novato, primero trabaja a través del Apéndice 3, o consulta los excelentes libros sugeridos en la Bibliografía.
Disponemos de versiones del Devpac 4 en disco, para el Disciple y la unidad de disco Opus Discovery, y para el Spectrum Plus 3. Estas versiones trabajan de la misma manera que aquí describimos, simplemente sustituye la palabra 'Microdrive' por Disco Opus, Disco del Plus 3 o Disco Disciple, según lo apropiado. Hay algunos puntos referentes al Plus 3 que se encuentran descritos en un apartado extra.
GENS4 ocupa unos 10k de memoria, una vez reubicado. y usa sus propias variables, por lo que es una pieza de software totalmente independiente. Contiene su propio editor de líneas que sitúa el fichero de texto inmediatamente después de GENS4 mientras que la tabla del ensamblador es creada después del fichero de texto. Al cargar GENS4, debes dejar memoria suficiente para contener el programa principal, la tabla de símbolos y el fichero de texto que vayas a usar en esta ocasión. A menudo será conveniente cargar GENS4 en una dirección de memoria baja.
Hay dos versiones del ensamblador en el disco. La primera posee 51 caracteres por línea, mientras que la segunda posee 32 caracteres por línea. Sus nombres son GESN4-51 y GENS42 respectivamente. Puedes usar la que más te guste, la segunda versión ocupa unos 400 bytes menos que la primera y es ligeramente más rápida.
Para cargar GENS4, teclea:

```
LOAD "" CODE XXXXX "ENTER si la cargas desde cinta
LOAD "GENS4" CODE XXXXX "ENTER | o
LOAD "GENS4-51" CODE XXXXX " ENTER | si la cargas del disco
```

donde XXXXX es la dirección en decimal donde queremos cargar el programa.

Una vez que hayas cargado el programa en la memoria deberás entrar en él tecleando RANDOMIZE USR XXXXX. donde XXXXX es la dirección en la que ha sido cargado. Si en cualquier momento deseas re-entrar en él, sólo deberás volver a teclear RANDOMIZE USR XXXXX, sin que se pierda el texto previamente creado.
Por ejemplo, si deseas cargar GENS4 de manera que se ejecute desde la dirección 26000, deberás teclear lo siguiente:

```
LOAD "" CODE 26000 "ENTER |
RANDOMIZE USR 26000 "ENTER |
```

Une vez que hayas entrado en GEN, aparecerá un pantalla de ayuda y aparecerá un signo '>', lo que indica que el editor está esperando que introduzcas algún comando o línea - Consulta la sección 3 para saber cómo entrar el texto y la sección 2 para saber qué entrar.

1.2 Realización de una copia de seguridad

Una vez que GENS4 se encuentre en la memoria de tu Spectrum, puedes realizar una copia de seguridad de la siguiente manera:

```
SAVE "GENS4-51" CODE XXXXX.11392 "ENTER |o
SAVE "GENS4" CODE XXXXX. 11010 "ENTER | para el casette.
SAVE *"M"¡1;"GENS4-51" CODE XXXXX.11392 "ENTER | o
SAVE *"M"¡1;"GENS4" CODE XXXXX,11010 "ENTER | para Microdrive.
```

Donde XXXXX es la dirección en la que ha sido grabado. Deberás hacer esto antes de entrar en GENS, de manera que se preserve la información de la reubicación del programa.
Los usuarios de Plus 3 deberán ver el folleto referente a su ordenador.

IMPORTANTE: Hemos permitido que se pueda realizar una copia de seguridad de GENS4 para tu propio uso. Por favor no copies GENS4 para dar (o mucho peor, vender) a tus amigos, nosotros suministramos el software a un precio más que razonable y además incorporamos un completo servicio post-venta, pero si mucha gente copia nuestro software. no podremos continuar nuestro trabajo y a la larga los usuarios serán los más perjudicados.
Por favor, COMPRA, no ROBES

SECCION 2 DETALLES DE GENS4

2.0 Cómo trabaja GENS4, opciones del Ensamblador, formato del listado

GENS4 es un rápido ensamblador del Z80 en dos pasos, y puede ensamblador todos los mnemónicos estándar del Z80 y posee otros rasgos como macros, ensamblado condicional, muchos comandos del ensamblador y un árbol binario de la tabla de símbolos.
Cuando invocas un ensamblado, usas el comando A de esta manera:

```
A4,2000,1:TEST "ENTER |
```

El primer número (4 arriba) después de la A especifica las opciones de ensamblado que deseas, estas opciones las encontraras un poco más adelante. Si no deseas ninguna opción, simplemente no teclees ningún numero, sólo una coma.
El segundo número (2000 arriba) es el espacio de la tabla de símbolos del ensamblador, en bytes decimales. Si no introduces ningún numero (usando simplemente una coma en lugar del número) GENS4 elegirá un espacio que él crea suficiente para el espacio que usa el texto - normalmente será un valor perfectamente aceptado. Sin embargo, cuando usas la opción Incluir, puedes especificar una tabla de símbolos más grande de lo normal; ya que el ensamblador no puede predecir con absoluta certeza el espacio necesario.
Después de la tabla de símbolos puedes teclear un nombre de fichero (1:TEST arriba). Si lo haces, el código objeto resultante del ensamblado será grabado en el microdrive, de manera automática. No importa la cantidad de código que generes, todo será grabado. Si no quieres que esto se haga, no teclees ningún nombre (no teclees tampoco una coma, ya que si lo haces, GENS supondrá que el nombre es una cadena de espacios). Mira el comando A en la sección 3 para más detalles y su efecto en el uso de ORG.

OPCIONES DEL ENSAMBLADOR

Opción l: Producir un listado de la tabla de símbolos al final del segundo paso del ensamblado.
Opción 2: No generar el código objeto.
Opción 4: Producir un listado del texto, observa que esto es justo lo contrario a versiones anteriores del ensamblador.
Opción 8: Dirigir cualquier listado ensamblador a la impresora.
Opción 9: Simplemente situar el código objeto, si es generado, justo después de la tabla de símbolos. El contador de dirección sigue manteniéndose en la posición marcada ORG y por lo tanto el código puede ser situado en una parte de la memoria y estar creado para ejecutarse en cualquier lugar.
Opción 32 : Desconectar la comprobación de dónde el código va siendo generado - útil para aumentar la velocidad del ensamblado.

Para cambiar las opciones, basta con sumarlas juntas, por ejemplo: A33 produce un ensamblado rápido, no se produce ningún listado, no se hacen comparaciones de dónde el código es generado (opción 32) y al finalizar el segundo paso se muestra un listado de la tabla de símbolos (opción 1). Observa que si has usado la opción 16 la orden del ensamblador ENT no tendrá ningún efecto. Puedes trabajar en otro lugar de donde vas a situar el código si la opción 16 ha sido especificada.

Puedes usar el comando Y para saber la dirección del final del texto (el segundo número que se muestra) y entonces sumándole el espacio que se ha usado para la tabla de símbolos + 2.
El ensamblado se realiza en dos pasos; durante el primer paso GENS4 busca los errores y compila la tabla de símbolos, y en el segundo paso genera el código objeto (a no ser que seleccionemos la opción 2). Durante el primer paso no se nos mostrará nada en la pantalla o impresora a menos que se detecte un error, en este caso se mostrara la línea equivocada seguida de un código de error (ver APENDICE 1). El ensamblado es parado - pulsa E para volver al editor o otra tecla para continuar el ensamblado desde la siguiente línea.
Al finalizar el primer paso se nos muestra el siguiente mensaje:

Paso 1 errores: nn

Si se ha detectado algún error el ensamblado será parado y no se procederá al segundo paso. Si en el texto se hacía referencia a alguna etiqueta no definida el mensaje:

WARHING etiqueta absent

será mostrado en pantalla por cada etiqueta no encontrada.
Durante el segundo paso es cuando se crea el código objeto (a no ser que la generación haya sido desconectada mediante la Opción 2). No se nos muestra ningún listado ensamblador durante este paso a no ser que éste sea activado con la opción 4 o mediante el comando del ensamblador *L+.
En la versión de 32 columnas, el listado ensamblador será normalmente de dos líneas y en la forma siguiente:

```
    C000   210100              25 ETIQUETA
                                  LD    HL.l
    1      6       15      21      26
```

mientras que en la versión de 51 columnas el listado es continuo y en una línea, sólo usando dos líneas si la línea es demasiado grande para caber en una sola.
La primera entrada en la línea es el valor del contador de dirección al inicio del proceso de esta línea, a no ser que el nemónico de esta línea sea uno de los pseudo-mnemónicos ORG, EQU o ENT (ver la Sección 2.7) en este caso la primera entrada representará el valor en el campo del operando de la instrucción. Esta entrada normalmente es mostrada en hexadecimal pero puede hacerlo en decimal sin signo a través del uso del comando del ensamblador *D+ (ver Sección 2.9).
La siguiente entrada, desde la columna 6, puede ser hasta 8 caracteres de largo (representando hasta 4 bytes) y es el código objeto generado por las instrucciones actuales - pero mira el comando del ensamblador *C descrito más adelante.
Después viene el numero de línea - un entero en el rango 1 a 32767 ambos inclusive.
Después de cualquier etiqueta viene el mnemónico que será mostrado entre las columnas 21 a 24 (en la versión de 32 columnas esto se hará en una nueva línea, a no ser que usemos el comando *C)
Seguidamente viene el campo del operando desde la columna 26 de esta línea y finalmente cualquier comentario que hayamos insertado en el final de la línea y se generaran nuevas líneas si son necesarias para imprimir este comentario.
El comando *C puede ser usado para producir un listado de la línea más corto - su función es omitir los 9 caracteres que contiene el código objeto y de esta manera podemos conseguir que más líneas de listado ensamblador quepan en una línea en la versión de 32 columnas. Ver la sección 2.8.

Modificación del formato del listado - sólo versión de 32 cols.
Puedes modificar la manera en que cada línea del listado es partida mediante la introducción de POKEs en tres direcciones de la versión de 32 columnas del GENS4. Los detalles para hacerlo los encontrarás un poco más adelante. Nosotros distinguiremos entre la línea de ensamblador la cual es la línea del listado ensamblador que actualmente se encuentra en un buffer y la línea de pantalla que es la línea que actualmente aparece en la pantalla. Una línea de ensamblador normalmente generará más de una línea de la pantalla.

1. Dirección Inicio de GENS4 * 51 (R33) marca en la columna - 5 en la cual la primera línea de la pantalla de la línea del ensamblador acabará. Puedes elegir como máximo el valor 255.
2. Dirección Inicio de GENS4 * 52 (S34) contiene la posición de columna (desde 1) en la que cada línea posterior de la pantalla debe comenzar.
3. Dirección Inicio de GENS4 + 53 (R35) informa de la cantidad de caracteres del resto de la línea del ensamblador deben ser mostrados en cada línea de la pantalla después de la primera.

Por ejemplo, digamos que quieres que la primera línea de pantalla de la línea del ensamblador contenga 20 caracteres (por ejemplo, sin incluir el campo de la etiqueta) y entonces que todas las siguientes líneas de la pantalla comiencen en la columna 1 y ocupen toda la línea. Además supongamos que has cargado GENS4 en la dirección 26000 decimal. Para efectuar estos cambios, ejecuta las siguientes instrucciones POKE mediante el BASIC:

```
POKE 26051,20
POKE 26052,1        debe haber un espacio como mínimo al inicio
POKE 26053,31       de cada una de las líneas posteriores.
```

Las modificaciones anteriores serán sólo disponibles si el comando *C no ha sido usado - el uso de *C hace que la línea vaya girando donde sea necesario.
El listado ensamblador puede ser parado al final de una línea pulsando las teclas "CAPS SHIFT" y "SPACE" juntas - y seguidamente podemos pulsar E para volver al editor o otra tecla para continuar el listado.
Los únicos errores que pueden ocurrir en el segundo paso son *ERROR* 10 (ver el apéndice 1) y Bad ORG ! (que sucede cuando el código objeto sobrescribiría el espacio ocupado por GENS4). *ERROR* 10 no es fatal y puedes continuar el ensamblado como en el primer paso, mientras que Bad ORG ! sí es fatal y causa inmediatamente el retorno al editor.
Al final del segundo paso aparece el siguiente mensaje:

Pass 2 errors: nn

Seguido de las posibles etiquetas inexistentes. El siguiente mensaje será ahora mostrado:

Table used: xxxxx from yyyyy

Esto informa la cantidad de la tabla de símbolos usada comparada con el espacio de que disponía.
En este punto, si la orden ENT ha sido usada correctamente, el mensaje Executes: nnnnn aparecerá. Esto muestra la dirección de ejecución del código objeto - puedes ejecutarlo con el comando del editor R. Ten cuidado al usar este comando si el ensamblado no ha acabado con el mensaje Executes: nnnnn.
Finalmente, si la opción 1 ha sido especificada, una lista alfabética de las etiquetas usadas y sus valores asociados será producida. El número de ellas mostrado en una línea puede ser cambiado POKEando la dirección Inicio de GENS4 + 50 con el valor apropiado, el valor por defecto es 2.
El control retorna al editor.

2.1 Formato de las sentencias del ensamblador

Cada línea de texto que debe ser procesada por GENS4 debe tener el siguiente formato donde algunos campos son opcionales:

```
ETIQUETA   MNEMÓNICO      OPERANDOS          COMENTARIO
start      LD             HL,etiqueta ; coger el valor de 'etiqueta'
```

Espacios y tabuladores (insertados por el editor) son generalmente ignorados. La línea será procesada como sigue:
El primer carácter de la línea es comprobado y la siguiente acción dependerá de su naturaleza como sigue:

; La línea entera es tratada como un comentario, y por ello será ignorada.

***** espera que el siguiente carácter o caracteres sean un comando del ensamblador (ver Sección 2.8). Trata todos los caracteres después del comando como un comentario.

<**CR**> (carácter de fin de línea) simplemente ignora la línea.

_ (espacio o código de tabulador) si el carácter es una espacio o un código de tabulador GENS4 supone que los siguientes caracteres que no sean ni espacio ni tabulador serán un mnemónico del Z80.

Si el primer carácter de una línea es un carácter distinto de estos, el ensamblador supone que se trata de una etiqueta - Ver Sección 2.2. Después de procesar una etiqueta válida, o si el primer carácter era un espacio o un tabulador. el ensamblador busca el siguiente carácter que no sea espacio ni tabulador y supone que éste será un código de fin de línea o un mnemónico del Z80 (ver apéndice 2) de hasta 4 caracteres de largo y acabado con un código de espacio o tabulador o un código de fin de línea.

Si el mnemónico es valido y necesita uno o mas operandos ignora los siguientes espacios / tabuladores hasta que encuentra el campo del operando y lo procesa.

Las etiquetas pueden estar presentes solas en un sentencia del ensamblador; esto es útil para aumentar la facilidad de lectura del listado.

Los comentarlos pueden aparecer en cualquier lugar después del campo del operando o, si el Mnemónico no necesita operandos, después del campo del mnemónico.

2.2 Etiquetas

Una etiqueta es un símbolo que representa hasta 16 bits de información. Una etiqueta puede ser usada para especificar la dirección de una instrucción en particular o un área de datos o también puede ser usada como una constante mediante la orden EQU (ver Sección 2.7)

Si una etiqueta está asociada con un valor mayor de 8 bits y es usada en el contexto donde una constante de 8 bits es necesaria, el ensamblador generará un mensaje de error, por ejemplo:

```
etiqueta    EQU     R1234
            LD      A, etiqueta
```

Provocara *ERROR* 10 cuando la segunda sentencia sea procesada durante el segundo paso.

Una etiqueta puede contener cualquier número o carácter válido pero sólo los 6 primeros son tratados como significantes; estos 6 caracteres deben ser únicos ya que una etiqueta no puede ser definida más de una vez (*ERROR* 4). Una etiqueta no pueda ser una Palabra Reservada (ver Apéndice 2) aunque una Palabra Reservada puede ser insertada en una etiqueta como parte da ella.

Los caracteres que pueden ser usados dentro de una etiqueta son 0-9, S y A-z. Observa que A-z incluye tanto las mayúsculas como las minúsculas junto a los caracteres |, Ñ, ¿,^, R y _. Una etiqueta debe comenzar con un carácter alfabético. Algunos ejemplos de etiquetas válidas son:

```
LOOP
loop
Una_etiqueta_larga
L¡1¿
L¡2¿
a
LDIR        LDIR no es una Palabra Reservada
Dos^5
```

2.3 Contador de dirección

El ensamblador mantiene un Contador de Dirección de manera que un símbolo en el campo de la etiqueta puede ser asociado con la dirección entrada en la tabla de símbolos. Este contador puede ser seleccionado mediante la orden ORG del ensamblador (ver Sección 2.7).

El símbolo$ puede ser usado para referirse al valor que posee actualmente este contador. Ejemplo: LD RL,$+5 generara el código de manera que cargues en el par de registros RL el valor 5 + el valor actual del contador de dirección.

2.4 Tabla de símbolos

Cuando se encuentra una etiqueta por primera vez, ésta es entrada en una tabla junto a dos punteros que indicarán, para usos posteriores, cómo está relacionada alfabéticamente con las otras etiquetas dentro de la tabla. Si la primera aparición de una etiqueta es en el campo de las etiquetas, su valor (obtenido mediante el valor del contador de dirección o del valor de la expresión que sigue a una orden EQU) es entrado en la tabla de símbolos. De otra forma, el valor es entrado cuando el símbolo sea seguidamente encontrado en el campo de las etiquetas.

Este tipo de tabla de símbolos es llamado Árbol binario de tabla de símbolos y su estructura permite a los símbolos ser entrados y recobrados de la tabla en un tiempo muy corto - esencial para programas largos.

El espacio de una entrada en la tabla da símbolos varía entre 8 bytes y 13 bytes dependiendo de la longitud del símbolo.

Si durante el primer paso un símbolo es definido más de una vez, se generará un error (*ERROR* 4) al final del ensamblado. La ausencia de una definición de un símbolo no permitiré al ensamblador continuar el proceso.

Observa que sólo los primeros 6 caracteres de un símbolo son introducidos en la Tabla de Símbolos para así ahorrar espacio de ésta.

Al final de un ensamblado se te informará del espacio que se ha usado para la Tabla de Símbolos durante el ensamblado - puedes cambiar la memoria reservada para la Tabla de Símbolos cuando comienzas el ensamblado (mira la Sección 2.0).

2.5 Expresiones

Una expresión es una entrada de operando que consiste de un solo TERMINO o una combinación de términos esperados por un OPERADOR. Las definiciones de término y operador vienen a continuación:

TERMINOS

una constante decimal, (ej. 1029)
una constante hexadecimal. (ej. R405)
una constante binaria, (ej. %10000000101)
una constante de carácter. (ej. "a")
una etiqueta, (ej. L1029)

además $ puede ser usado para indicar el valor de la posición actual del contador de dirección.

OPERADORES

+	suma
-	resta
&	operación lógica AND
O	operación lógica OR
!	operación lógica XOR
*	multiplicación entre enteros
/	división entre enteros
?	función MOD (a ? b = a - (a/b)*b)

Notas: R es usado para indicar el inicio de un numero hexadecimal. % para un numero binario y " para un carácter. Cuando se lee un número (decimal, hexadecimal o binario) GENS4 coge los 16 bits menos significativos del número (ej. MOD 65536) , ej. 70016 se convierte en 4480 y R5A2C4 en RA2C4.

Se dispone de una gran cantidad de operadores, pero estos no siguen ninguna preferencia; *LAS EXPRESIONES SON EVALUADAS ESTRICTAMENTE DE IZQUIERDA A DERECHA*. Los operadores *, / y ? han sido incluidos por conveniencia pero no pueden formar parte de una expresión completa ya que esto aumentaría el espacio usado por GENS4. Si una expresión se cierra entre paréntesis entonces ésta será interpretada como una dirección da memoria, de manera que la

instrucción LD HL, (loc+5) cargaría en el par de registros HL el valor de 16 bits contenido en la dirección de memoria loc+5.
Algunas instrucciones del Z80 (JR y DJNZ) esperan un valor de 8 bits y no uno de 16 - esto es llamado direccionamiento relativo. Cuando el direccionamiento relativo es especificado GENS 4 automáticamente resta el valor del contador de dirección de la siguiente instrucción al valor dado en el campo de los operandos para así obtener la dirección relativa para la instrucción actual. E1 rango permitido para un dirección relativa es la posición del contador de dirección de la siguiente instrucción -128 hasta +127.
Si, por cualquier motivo, deseas especificar una dirección relativa al actual valor del contador de dirección deberes usar el símbolo $ (una Palabra Reservada) seguido del desplazamiento requerido. Desde esta. el rango relativo deberá ser entre el contador de dirección de la actual instrucción -126 hasta +129 ambos inclusive.

Ejemplos de expresiones válidas

E 5000 – etiqueta
%1001101 ! %1011 da %1000110
R3456 ? R1000 da R456
4 * 5 + 3 8 da 19
S-etiqueta + 8
2345 / 7 -1 da 334
"y"-"¡" +7
17 O %1000 da 25
Observa que los espacios pueden ser insertados entre términos y operadores y viceversa pero no dentro de los términos.
Si una multiplicación da como resultado un valor mayor de 32768 se nos mostrará el *ERROR* 15 mientras que si una división se produce entre 0 aparecerá *ERROR* 14 - el desbordamiento es ignorado. Todos loa usos aritméticos de números en complemento a dos donde su valor sea mayor de 32767 son tratados como negativos. Ej.: 60000 = -5536 (60000-65536)

2.6 Macros
Loa Macros te permiten escribir programas en ensamblador más cortos y claros pero deben ser usados con cuidado y no deben ser confundidos con subrutinas. Una definición macro consta de una serie de sentencias de ensamblador, junto a un nombre del macro; cuando usemos ese nombre posteriormente en el campo de mnemónicos éste será sustituido por las sentencias que habíamos usado en la definición. Por ejemplo, e1 macro HSUB puede ser definido como:

```
HSUB    MAC
        OR    A
        SBC   HL.DE
        ADD   HL.DE
        ENDM
```

y entonces, cada vez que usemos HSUB como un mnemónico, éste generará tres sentencias de ensamblador OR A, SBC HL.DE y ADD HL.DB. Esto te ahorra tener que teclear lo mismo una y otra vez y hace que el listado de tu programa sea más fácil de entender, pero debes recordar que cada aparición de HSUB será sustituida por código, lo que ocupará memoria, mientras que si usases una subrutina y la llamas con un CALL puede que sea más eficiente. Seguidamente damos el formato de las definiciones de una macro y su invocación junto a algunos ejemplos. Te rogamos que los estudies cuidadosamente.

Una definición macro posee la siguiente forma:

```
nombre          MAC
                  .
                  .
                definición del macro
                  .
                  .
                ENDM
```

donde 'nombre' es el nombre que se le da al macro y que será usado para invocarlo posteriormente, MAC indica el inicio de una definición de un Macro y ENDH indica el final da la misma.
Loa parámetros de un macro pueden ser indicados en el momento de invocarlo, disponiendo de hasta 32 de ellos (0-31). Para darle ese valor a algún registro sólo debemos introducir un signo = seguido del número de parámetros que deseamos. Ejemplo: el macro:

```
MOVE            MAC
                LD          HL. =0
                LD          DE. =l
                LD          BC. =2
                LDIR
                ENDM
```

usa tres parámetros, dirección de inicio, dirección de destino y longitud, carga estoa valores en los registros HL, DE y BC y realiza un instrucción LDIR. Para invocar este macro en cualquier otra parte posterior del programa, simplemente debemos usar el nombre del macro en el campo de mnemónicos seguido de los valores que quieres dar a los 3 parámetros, de esta manera:

```
        MOVE    16384, 16385, 4096
```

Hemos usado direcciones específicas, pero podemos, si lo deseamos, usar cualquier expresión válida para especificar el valor, ej.:

```
        MOVE    inicio,  inicio+1, long
```

Piensa... ¿es el uso anterior un buen uso de un macro? ¿Podría haber sido una subrutina?
Dentro de la definición del macro. los parámetros pueden aparecer en cualquier expresión válida:

```
        HMS     MAC
                LD          HL. = 0 * 3600
                LD          DE. = 1 *60
                ADD         HL. DE
                LD          DE. =2
                ADD         HL. DE
                ENDM
```

este macro, tomando tres parámetros - horas, minutos y segundos - produce en el registro HL el número total de segundos. Podrías usarlo así:

```
        horas       EQU     2
        minutos     EQU     30
        segundos    EQU     12
        inicio      EQU     0
```

```
        HMS     horas, minutos, segundos
        LD      DE. inicio
        ADD     HL.DE           ¡ HL daría el tiempo final
```

Los macros no pueden anidarse de manera que no puedes definir un macro dentro de una definición de otro macro, y tampoco puedes invocar a un macro desde otro macro.
En el momento del ensamblado, cada vez que se encuentra un nombre de un macro en el campo de los mnemónicos, el texto del macro es ensamblado. Normalmente este texto no es listado en el listado ensamblador - solo se muestre el nombre del macro. Pero puedes forzar un listado del contenido del macro usando el comando del ensamblador *M+ antes de donde deseas que comiencen a listarse los macros - usa *M- para desconectarlo.
Si no queda espacio para el buffer de macros, un mensaje se mostrará en la pantalla y el ensamblado será abortado; usa el comando del editor C pera aumentar el espacio reservado a los macros.

2.7 Órdenes del ensamblador

Ciertos pseudo-mnemónicos son reconocidos por GENS4. Estas órdenes del ensamblador, como son llamadas, no tienen efecto durante le ejecución en el Z80, ya que no son codificadas, son simplemente órdenes que indican al ensamblador algunas acciones que debe realizar en el tiempo del ensamblado. Estas acciones tienen la función de cambiar, de alguna manera, el código objeto producido por GENS4.
Loa pseudo-mnemónicos son ensamblados exactamente igual que las instrucciones ejecutables; pueden estar precedidas por una etiqueta (imprescindible para EQU) y seguidos de un comentario. Las órdenes disponibles son:

```
        ORG     expresión
```

reselecciona el valor del contador de dirección. Si la opción 2 y la opción 16 no están seleccionadas y un ORG provocaría la destrucción de GENS4 (por realizarse en las posiciones ocupadas por él) o del fichero de texto o la tabla de símbolos se nos mostrará el mensaje Bad ORG! (ORG erróneo) y el ensamblado es abortado. Mira la Sección 2.0 para mas detalles de cómo afectan las opciones 2 y 16 a ORG. Mira también el comando A de la Sección 3 para saber algunas precauciones en el uso de ORG cuando usamos la opción de grabación automática del código generado.

```
        EQU     expresión
```

debe ir precedido de una etiqueta. Selecciona el valor de una etiqueta como el valor de 'expresión'. Esta expresión no puede contener un símbolo que todavía no haya sido definido (*ERROR*)

2.9 Comandos del ensamblador

Los comandos del ensamblador, al igual que las órdenes, no producen ningún efecto en el Z80 durante la ejecución, dado que no son codificados. Sin embargo, al contrario que las órdenes, tampoco producen ningún efecto en el código objeto producido - los comandos del ensamblador simplemente modifican el formato del listado. Un comando del ensamblador es una línea del texto fuente que comienza con un asterisco (*).
La letra después del asterisco determina el tipo de comando y debe estar en mayúsculas. El resto de la línea puede ser cualquier tipo de texto excepto que los comandos L y D esperan un signo + o un - después del comando.

```
        * E
```

imprime tres líneas en blanco en la pantalla o la impresora - útil para separar algunas partes del listado.

* Hs

provoca que la cadena s sea tomada como una cabecera que será imprimida después de cada comando *E. *H automáticamente realiza un * E.

* S

obliga al listado a pararse en esta línea. El listado puede ser reactivado pulsando cualquier tecla del teclado. Es útil para leer las direcciones en medio del listado.
*Nota: *S todavía es reconocido después de un comando *L-. pero no parará la impresión.*

* L-

hace que el listado se desconecte al principio de esta línea.

* L+

realiza la operación inversa al comando anterior, es decir, reactiva la impresión.

* D+

hace que el valor del contador de dirección sea dado en números decimales al principio de cada línea, que normalmente se da en hexadecimal. Para ello se usan números decimales sin signo.

* D-

funciona igual que el anterior, pero provoca que el contador de direcciones salga en hexadecimal.

* C-

Reduce el listado ensamblador comenzando en la siguiente línea. El listado es abreviado sin incluir el código objeto generado en la línea actual - esto ahorra 9 caracteres y permite al ensamblador poner más líneas de ensamblador en la pantalla de la versión de 32 caracteres y de esta manera el listado es más legible.

* C+

Retornar al listado del ensamblador completo, como se describe en la Sección 2.9

* M+

Conecta el listado expresión de los macros.

* M-

Desconecta el listado de las expresiones de los macros.

* F nombre

Donde 'nombre’ hace referencia al nombre de un fichero.
Este es un comando muy potente que te permite ensamblar texto desde la cinta o el microdrive - el fichero de texto es leído desde la cinta o el microdrive en un buffer, un bloque cada vez, y es entonces ensamblado desde el buffer; esto te permite crear grandes cantidades de código si estos no caben junto al texto en la memoria.
El nombre del fichero (de hasta 10 caracteres) del fichero de texto que desees 'incluir' en este punto en el ensamblador puede, opcionalmente, ser especificado después de F y debe estar precedido de un espacio. Si el fichero es del microdrive deberás indicar el nombre del fichero junto a él. Ejemplo:

* F 2:TEST para incluirlo de la unidad de microdrive 2

* F TEST para hacerlo desde la cinta

Si no se entra el nombre del fichero se asumirá que se desea incluir el primer fichero que se encuentre en le cinta.
Si el fichero está en el microdrive deberás haberlo grabado con el comando P del editor de la manera normal.
Pero si está en la cinta, el fichero deberá haber sido grabado con el comando del editor T y no con el comando P - esto es necesario porque el fichero de texto de la cinta debe ser ensamblado en bloques y estos deben disponer de espacio entre ellos que sea suficiente para que dé tiempo a ensamblarlo.
El espacio designado para cada bloque usado por este comando (y el comando del editor T) es seleccionado con el comando del editor C (mira la siguiente sección). La posibilidad de cambiar este espacio te permite optimizar la relación espacio/velocidad; por ejemplo, si no vas a usar el comando *F puedes encontrar útil seleccionar el buffer a 1 pare minimizar el espacio y así conseguir que GENS4 disponga de más memoria libre pera tu uso.
Cada vez que se detecta un comando F se busca en la cinte o el microdrive el fichero indicado; esto sucederá en el primero y el segundo paso ya que el fichero debe ser comprobado en ambos pasos. Si el fichero está grabado en la cinta, GENS4 buscará el fichero con el nombre indicado, o en su defecto, el siguiente fichero. Si se encuentre un fichero cuyo nombre no corresponde al indicado aparecerá el mensaje 'Found nombre-del-fichero' y la búsqueda continuará, y une vez que se encuentre el fichero buscado aperecerá el mensaje 'Using nombre-del-fichero', el fichero será cargado bloque por bloque y será incluido.
Mira el Apéndice 3 pera ver un ejemplo del uso de este comando.
Los demás comandos, distintos de *F, sólo son reconocidos durante el segundo paso.
Si el ensamblado ha sido desconectado, también se desconectarán todos los comandos del ensamblador.
Nota: La facilidad de incluir ficheros no está disponible para cinta en las versiones de Devpack 4 para el Disciple, Opus y el disco del Plus 3. Esto es así porque es mucho más rápido y fácil incluir los ficheros desde el disco. En estas versiones el comando C no te permitirá especificar el espacio reservado al buffer y el comando T no existe.

SECCION 3 EL EDITOR

3.1 Introducción al editor

El editor que incorporan todas las versiones de GENS4 es sencillo y fácil de manejar y se basa en líneas. Ha sido creado para trabajar con todos los sistemas operativos del Z80 y proporciona una gran rapidez y eficiencia en la edición de programas.
Para intentar reducir al máximo el espacio ocupado por el fichero de texto, el editor realiza una cierta compresión del mismo . Esta se realiza según el siguiente esquema: cada vez que tecleamos una línea desde el teclado, ésta es guardada, carácter a carácter dentro de un buffer interno del editor: entonces, cuando la línea se ha acabado (pulsando ¡ ENTER¿) es transferida desde este buffer al fichero de texto.
Es durante esta transferencia cuando algunos espacios son comprimidos: la línea es inspeccionada desde el primer carácter, y si éste es un espacio se introduce en su lugar un código de tabulador y se suprimen todos los espacios posteriores. Si el primer carácter no es un espacio entonces se transfieren los caracteres siguientes hasta encontrar un espacio y en ese momento se hace la misma acción que si este espacio hubiese estado en el primer carácter. Esto es repetido de manera que se obtienen códigos de tabuladores entre la etiqueta y el mnemónico y entre el mnemónico y los operandos. Por supuesto, si se encuentre un código de retorno de carro ¡ENTER¿ la transferencia será finalizada y el control volverá al editor.
Este proceso de compresión es transparente y puedes simplemente usar la tecla de ¡CURSOR DERECHA¿ para producir directamente un código de tabulador que además, ocupa menos espacio.

Observa que los espacios no son comprimidos en los comentarios y que una etiqueta no puede contener espacios entre ella, o en el mnemónico o en el campo de los operando.
Entramos automáticamente en el editor cuando ejecutamos GENS4 y se nos muestra una pantalla de ayuda, seguida del signo >.
En respuesta a este signo puedes entrar una línea de comandos en el siguiente formato:

C N1, N2, S1, S2 seguido de ¡ENTER¿

C es el comando a ser ejecutado (ver Sección 3.2). N1 es un número en el rengo 1 - 32767 inclusive. N2 es otro número en el mismo rango. S1 es una cadena da una longitud máxima de 20 caracteres. S2 es otra cadena de la misma longitud máxima.
Las coma es usada para separar los distintos argumentos (pero este signo puede ser cambiado - ver el comando S) y los espacios son ignorados, excepto dentro de las cadenas. Ninguno de los argumentos es obligatorio aunque algunos comandos (como el comando de borrar líneas - D) no funcionarán si no indicamos tanto N1 como N2.
El editor almacena los últimos números y cadenas entrados y usa estos valores anteriores, si son aplicables, si no especificas un argumento particular dentro de la línea de comandos. Los valores N1 y N2 valen inicialmente 10 y leas cadenas están vacías. Si entras una línea de comando errónea como F- 1 , 100, HOLA entonces la línea será ignorada y aparecerá el mensaje 'Pardon?'- deberás teclear de nuevo la línea correctamente (en el ejemplo serie F1.100.HOLA) . Este mensaje de error también se nos mostrara si la longitud de S2 supera los 20 caracteres: si la longitud de S1 es mayor de 20 caracteres los caracteres sobrantes simplemente son ignorados.
Los comandos pueden ser introducidos tanto en mayúsculas como en minúsculas.
Mientras estás entrando un comando, algunas combinaciones de teclas pueden ser usadas. ¡CURSOR IZQUIERDA¿ borra la línea hasta el inicio, ¡CURSOR DERECHA¿ lleva el cursor a la siguiente posición de tabulador. ¡CAPS SHIFT¿ 0 o ¡DELETE¿ borran el carácter anterior.
La siguiente subsección contiene la lista de los distintos comandos disponibles dentro del editor - observa que cuando un argumento está cerrado entre los símbolos < > esto indicará que ese argumento DEBE estar presente para que el comando se ejecute.

3.2 Los Comandos del Editor

3.2.1 Inserción de texto

El texto puede ser insertado dentro del fichero de texto tanto tecleando un número de línea, un espacio y después el texto correspondiente como usando el comando I. Observa que si tecleas un número de línea seguido de ¡ENTER¿ (es decir una línea sin texto) esta línea será borrada del texto si existía. Cada vez que el texto va a ser introducido ¡CURSOR IZQUIERDA¿ (borra hasta el principio de la línea). ¡CURSOR DERECHA¿ (mueve el cursor a la siguiente posición de tabulador) y ¡EDIT¿. (retornar al bucle de introducción de comandos) pueden ser usados.
La tecla ¡DELETE¿ (¡CAPS SHIFT¿ 0) borrará el carácter que se encuentre a la izquierda del cursor. El texto es introducido en un buffer interno y GENS4 se encarga de impedir que tecleemos una línea excesivamente grande (lo que provocaría el desbordamiento de este buffer). Si durante la inserción de texto, el editor detecta que el final del texto está casi al final de la RAM nos mostrará el mensaje Bad Memory!. Esto indica que no podemos introducir más texto en el fichero de texto actual, así que deberá ser grabado en cinta/microdrive y deberemos teclear el resto del programa en otro fichero (en estos casos es muy útil el comando del ensamblador *F)

Comando: I n.m

Cuando usamos este nodo, el editor pasa a un modo automático de inserción de líneas. Los números de líneas se nos van mostrando comenzando en 'n' e incrementándose en pasos 'm'. Puedes entrar el texto requerido después del número mostrado, usando los distintos códigos de control y si deseamos acabar el texto de una línea solo debemos pulsar ¡ENTER¿. Para salir de este modo pulsa ¡EDIT¿. Si entras una línea con un número que ya existía, el texto de la línea existente será borrado y sustituido por la nueva línea, en el momento en que pulses ¡ENTER¿. Si el incremento automático de un número de línea produce un número mayor que 32767 el modo de inserción de líneas será abandonado inmediatamente.
Si cuando estemos tecleando texto llegamos al final de la línea de la pantalla sin haber tecleado 64 caracteres (el máximo admitido por el buffer) la pantalla será movida hacia arribas (lo que se denomina 'scroll') y podrás seguir tecleando.
Después del número de línea se introducirá automáticamente un espacio de manera que los números queden separados del texto. Tanto el número de línea como este espacio no pueden ser editados ni borrados.

3.2.2 Listado de texto

El texto puede ser inspeccionado mediante el uso del comando L; la cantidad de líneas mostradas cada vez durante le ejecución de este comando son fijadas inicialmente pero pueden ser cambiadas a través del comando K.

Comando: L n,m

Este comando lista el texto desde el número de línea 'n' hasta el número de línea 'm' ambos inclusive. El valor por defecto de 'n' siempre es 1 y el de 'm' 32767. Estos valores no son cambiados por los argumentos entrados previamente.
Para listar el texto entero simplemente usa L sin argumentos. Las lineas de la pantalla son formateadas con un margen izquierdo de manera que los números queden claramente mostrados. La tabulación del texto es automática, lo que realiza una clara separación entre los distintos campos de la línea. El número de la cantidad de líneas que se muestran en la pantalla puede ser controlado con el comando R - después de una cierta cantidad de líneas el listado se para (si todavía no se ha llegado a la línea 'm'), pulsa ¡EDIT¿ para retornar al editor o cualquier otra tecla para continuar listando.

Comando: K n

K selecciona el número de lineas de la pantalla que son listadas antes de parar el listado como se describe en el comando L. El valor (n MOD 256) es computado y almacenado. Por ejemplo, si usamos K5 y luego realizamos un listado este se producirá de 5 en 5 lineas.

3.2.3 Edición de texto

Una vez que se ha creado algún texto, inevitablemente tendrás que editar alguna línea. Pare ello dispones de varios comandos que te permitirán hacerlo:

Comando: D <n,m>

Todas las lineas desde n hasta m inclusive son borradas del fichero de texto. Si m < n, o si se especifican menos de dos argumentos, no se producirá ninguna acción; esto es así para evitar

borrados accidentales. Si sólo deseas borrar una línea puedes hacer m=n, pero esto pueda hacerse de una manera más rápida: sólo tienes que teclear el número de la línea y pulsar ¡ENTER¿

Comando: M n.m

Este comando mueve el texto de la línea n a la línea m borrando cualquier texto que ya existiese. Observa que la línea n permanece inalterada. Si el número de línea n no existe no se producirá ninguna acción.

Comando: N <n,m>

Este comando renumera el texto con un primer número de línea n y sumando m parea cada línea posterior. Ambos argumentos deben ser especificados y si la renumeración provocaría una línea mayor de 32767 no se producirá ninguna renumeración.

Comando: F n,m,f,s

En el texto existente entre n y m se busca la cadena f. Si se encuentra, la línea aparecerá en pantalla y se entrara en modo Edit (mira el comando E n).
Puedes entonces usar los comandos del modo edición para buscar las siguientes apariciones de la cadena f o para sustituir esta cadena por la cadena s y seguir buscando la siguiente aparición. Observa que el rango y las dos cadenas pueden haber sido definidas previamente mediante otro comando de manera que sólo es necesario usar F para iniciar la búsqueda - lee el ejemplo de la Sección 3.3 para más detalles.

Comando: E n

Editar la línea con número n. Si n no existe no se producirá ninguna acción: de otra forma la línea será copiada en un buffer y mostrada en pantalla (con su número), el número de línea es mostrado de nuevo debajo de la línea original y se entra en el modo edición. Cualquier cambio que realicemos en ella se producirá en el buffer y no en el fichero de texto: de esta manera podemos recuperar la línea original si nos hemos equivocado.
En este modo un puntero se va moviendo imaginariamente a través de la línea (comenzando desde el primer carácter) y se incorporan algunos subcomandos que te permiten editar la línea. Los subcomandos son:

(espacio)	incrementar el puntero en uno, es decir, se mueve el cursor al siguiente carácter de la línea. No puedes pasar del final de la línea.
¡DELETE¿	decrementa el puntero en uno. No puedes pasar del primer carácter de la línea.
CURSOR DERECHA	mover el puntero adelante hasta la siguiente posición de tabulador de cade línea de la pantalla.
¡ENTER¿	finalizar la edición almacenando todos los cambios realizados.
Q	anular la edición de esta línea, dejando la línea original como estaba, ignorando los cambios realizados.
R	retomar la línea original, es decir, olvidar todos los cambios hechos en esta línea y volver e empezar con una línea igual a la original.
L	lista el resto de la línea que estamos editando, es decir, el texto posterior a la posición del puntero (cursor). Te mantienes en el modo edición y con el puntero en el inicio de la línea.

K	borrar el carácter que se encuentra en la posición actual del puntero.
Z	borrar todos los caracteres desde la posición del puntero (incluida) hasta el final de la línea.
F	buscar la siguiente aparición de la cadena previamente definida (mira el comando F). Este sub-comando automáticamente saldrá de la edición de la línea actual (manteniendo los cambios) si no encuentre ninguna aparición de la cadena en la línea actual. Si se detecta en una línea posterior (dentro del rango previamente especificado) la cadena se volverá al modo edición con la línea en que se encontró la misma. El puntero siempre es posicionado al inicio de la cadena encontrada.
S	sustituir la actual aparición de la cadena f (buscada) por la cadena s (de sustitución) ambas previamente seleccionadas con el comando F y buscar la siguiente aparición de la cadena f. Este sub-comando, junto al anterior, son usados pare buscar a través del fichero de texto y opcionalmente cambiar algunas cadenas f por la cadena s - mira la Sección 3.3 pare tener un ejemplo.
I	Insertar caracteres en la posición actual del puntero. Estarás en este modo hasta que pulses ¡ENTER¿ - esto te retornara al modo edición con el puntero en la posición posterior al último carácter introducido. El uso de ¡DELETE¿ dentro de este sub-modo causará que el carácter a la izquierda del puntero sea borrado del buffer y CURSOR DERECHA avanzará el puntero a la siguiente posición de tabulador de la pantalla, insertando espacios.
X	este comando lleva el cursor al final de la línea y entra en el sub-modo del comando I, descrito anteriormente.
C	Este comando te permite entrar en el sub-modo de cambio. Este modo te permite sobrescribir el carácter que se encuentre en la posición actual del puntero y entonces avanza un carácter a la derecha. Para salir de este sub-modo deberás pulsar ¡ENTER¿ lo que te llevará de nuevo el modo de edición con el puntero en la posición posterior el ultimo carácter cambiado. ¡DELETE¿ simplemente mueve el cursor un caracter a la derecha, mientras que CURSOR DERECHA no produce ningún efecto.

3.2.4 Comandos de cinta/microdrive

El texto puede ser grabado en cinta/microdrive o cargado desde cinta/microdrive usando los comandos P, G y T. El código objeto generado puede ser grabado en cinta/microdrive con el comando O. Los nombres de los ficheros pueden llegar hasta un máximo de 10 caracteres de largo.

Comando: Pn, m, s
El rango de líneas n-m es grabado en cinta o microdrive bajo el nombre especificado en la cadena s. El texto será grabado en microdrive si el nombre del fichero comienza con un número de unidad seguido de los dos puntos (:). Recuerda que estos argumentos pueden haber sido seleccionados con un comando anterior. Ejemplos:

P10, 200, EJEMPLO	graba las líneas 10-200 en cinta como EJEMPLO.
P500, 900, 1:TEXTO	graba las líneas 500-900 en el microdrive 1 y bajo el nombre TEXTO.

Antes de introducir este comando asegúrate que el cassette está ya en modo grabación, si grabas en cinta. No uses este comando si quieres, en otro momento. 'incluir' (con el comando del ensamblador *F) el texto desde la cinta, para ello usa al comando T. Si lo quieres incluir de microdrive sí que deberás usar este comando.

Cuando introduzcas el nombre del fichero para microdrive y actualmente ya exista un fichero con ese nombre. se te preguntará:

File Exists Delate (Y/N)? - (el fichero existe, borrarlo (Y/N) ?)

Si respondes Y (si) si fichero se borrará y continuará la grabación normalmente. Si pulsas cualquier otra tecla retornarás al editor sin grabar el fichero.

Comando: G , , s
Se busca en la cinta o microdrive un fichero de nombre 's': cuando es encontrado, se carga al final del texto actual. Si se introduce una cadena de nombre nula, se buscará en cinta el siguiente fichero de texto y éste será cargado. Para microdrives, debes especificar el nombre del fichero y éste debe comenzar con un número de unidad, seguido de dos puntos (:).
Si usas el cassette, después de haber entrado el comando G, aparecerá el mensaje 'Start tape..' - debes ahora pulsar PLAY en el cassette. Se realiza una búsqueda del fichero de texto con el nombre especificado, o del primer fichero si el nombre se ha especificado nulo. Cuando se encuentra el fichero se nos muestra al mensaje 'Using ' seguido del nombre del fichero, pero si se encuentra un fichero que no es de texto aparecerá 'Found ‘ seguido del nombre del fichero y la búsqueda continuará.
Si usas microdrive y el fichero no es encontrado aparecerá el mensaje ‘Absent' (ausente).
Observa que si ya hay algún texto en la memoria el fichero de texto será cargado justo a continuación del mismo, sin borrarlo, y los números de línea serán renumerados de 1 en 1.

Comando: Tn, m, s
Grabar un bloque de texto entre los números de líneas n-m (ambos inclusive), en cinta en un formato que permita la posterior inclusión (con el comando del ensamblador *F) - lee la Sección 2.9. El fichero es grabado con el nombre de fichero s. La grabación se haca justo en el momento en que pulses ¡ENTER¿ . por lo que el cassette ya deberá estar en modo grabación antes de entrar este comando.
Si intentas incluir de microdrive deberás usar con el comando P para grabar el texto, de la manera normal, y no este comando T.
Observa que este comando es sólo usado si quieres ensamblar el texto desde la cinta en otro momento. No se encuentra disponible en las versiones de disco de Devpac 4.

Comando: O , , s

Grabar el código objeto en casette o microdrive. El nombre del fichero s puede tener hasta 6 caracteres y debe comenzar con un numero de unidad (1-8) y el signo de dos puntos (:) si deseas grabarlo en microdrive.
Sólo el último 'bloque' de código producido por el ensamblador puede ser grabado de esta manera, es decir, si tienes más de un ORG en el texto fuente de tu programa sólo el código producido por el ultimo ORG será grabado.
El código debe haber sido producido antes de usar O.
A menudo será más conveniente grabar el código objeto generado directamente usando el comando A , , nombre-de-fichero en lugar del comando O. Mira el siguiente comando.

3.2.5 Ensamblado y ejecución desde el editor

Comando: Ao, s, f
Este comando ensembla el texto desde la primera línea.

'o' te permite especificar las opciones que van a ser usadas en el ensamblado (ver Sección 2.0). Normalmente, podrás usar las opciones por defecto tecleando una coma.

's' te da la posibilidad de cambiar el espacio de la tabla de símbolos (es decir, las etiquetas). Mira las Secciones 2.0 y 2.4 para saber su significado. De nuevo, el valor de la tabla por defecto, será a menudo usado satisfactoriamente, excepto cuando 'incluimos' texto (con el comando del ensamblador *F).

‘f’ puede ser un nombre de fichero válido, comenzando con un número de unidad de microdrive seguido de dos puntos: este nombre puede tener hasta 10 caracteres de largo. La presencia del nombre del fichero aquí provoca que el ensamblado se realice de una manera diferente a lo normal.

En lugar del simple ensamblado del código objeto en la memoria y la parada del mismo si se detecta el fin de la memoria, el ensamblador ensamblará en la memoria hasta que llegue el tope de la memoria (puedes alterar este tope con el comando U) y entonces grabará el código objeto generado en el microdrive en el fichero introducido por ti.

El ensamblado continuará de nuevo desde el principio de la memoria y el proceso se repetirá hasta que se haya grabado todo el código objeto.

Por lo tanto, no hay un límite (excepto el espacio libre en el cartucho de microdrive) para el espacio del programa que puedes ensamblar.

Hay algunos puntos que es importante resaltar sobre el uso de la orden ORG y esta facilidad de grabar directamente en un fichero:

1. La orden ORG hace que el código objeto sea situado en una posición específica y TAMBIEN lo hará cada vez que el código objeto sea en un fichero a menos que la opción 16 se use para asegurarse que el código objeto es situado directamente después de la tabla de símbolos. Generalmente será mejor usar la opción 16 cuando ensamblemos directamente en microdrive, ya que esto da el espacio máximo para el buffer en el que se va guardando el código objeto, ten en cuenta que le dirección de ejecución de tu programe no será alterado para nada.
2. Deberías evitar usar más de un ORG en tu programa y en lugar de esto rellenar la memoria con ceros, usando DBPS. Ejemplo:

```
ORG    50000
; algo de código
RET

ORG    60000
; más código
```

no sería grabado a microdrive correctamente, porque el segundo ORG redefine el inicio del buffer código objeto. Sin embargo:

```
ORG       50000
; algo de código
RET
; rellenar con ceros hasta 60000:
DEF$      60000-$
; más código
```

sería grabado correctamente dado que DEF$ 60000-$ genera ceros suficientes para asegurarse de que el siguiente código comience en la posición 60000.

Esto es obviamente ineficiente dada la cantidad de código almacenada en microdrives, pero su simplicidad mantiene el ensamblador corto y rápido.
Ejemplos del comando A:

A20 . . 1 : TEST ¡ENTER¿

Ensamblar, mostrando el listado, poniendo el código objeto inmediatamente después de la tabla de símbolos (lo que significa dar el máximo espacio al buffer del código objeto) usando la tabla de símbolos por defecto y grabando el código objeto en el microdrive 1 bajo el nombre TEST.

A.3000 ¡ENTER¿

Ensamblar el programa, usando las opciones por defecto y con un espacio de tabla de símbolos de 3000 (en decimal) bytes.
Mira la Sección 2 para detalles complementarios de lo que sucede durante el ensamblado.

Comando: R
Si el código fuente ha sido ensamblado sin errores y se ha especificado una dirección de ejecución con la orden del ensamblador ENT, el comando R puede ser usado para ejecutar el código objeto del programa. Este código puede usar la instrucción RET (cuyo código es RC9) para retornar al editor si la pila se encuentra en la misma posición al final de programa que la que había al inicio. Observa que ENT no tendrá efecto si usamos la Opción 16 del ensamblador.
Antes de ejecutar el código, las interrupciones son conectadas y el registro IY es cargado con el valor R5C3A, imprescindible para las rutinas que usen Rom del Spectrum.

3.2.6 Otros comandos

Comando: B
Este comando simplemente devuelve el control al sistema operativo (es decir, el Basic). Para re-entrar en el ensamblador debes usar RANDOMIZE USR xxxxx donde xxxxx es la dirección donde GENS ha sido cargado.

Comando: C
Este comando te permite configurar el espacio del buffer de 'inclusión' (sólo en las versiones de cinta de Devpac) y el de las funciones Macro.
El buffer de inclusión es el buffer en el cual el texto es almacenado cuando ensamblamos directamente desde cinta o microdrive - cuanto más largo sea, más texto será leído cada vez desde cinta o microdrives, lo que hará que el ensamblado sea más veloz. Pero por otra parte más memoria será ocupada. Por lo tanto, tú puedes elegir el espacio/velocidad que más te corresponda: para eso puedes usar el comando C.
El buffer de los macros es usado para almacenar el texto de cada definición que quieras usar.
El comando C te pregunta 'Include buffer' (buffer de inclusión) y después "Macro buffer' (buffer de macros). En ambos casos simplemente debes entrar el valor de la cantidad de bytes a reservar (en decimal), seguido de ¡ENTER¿. Si pulsas ¡ENTER¿ por sí solo sin introducir ningún número no se producirá ninguna acción. Si especificas el valor del buffer de inclusión, el espacio mínimo a reservar es de 256 bytes. Puedes abortar esta opción pulsando EDIT (o, lo que es lo mismo, CAPS SHIFT + 1)
Observe que en las versiones de disco de Devpac 4, sólo puedes cambiar el espacio reservado para el buffer de macros.

C no destruye tu texto; éste es movido hacia arriba o abajo en la memoria según el espacio que reservemos. Es mejor definir los buffers al inicio de la sesión si crees que los vas a necesitar.

Comando: S,,d
Este comando te permite cambiar el delimitador que es tomado como separador de loa argumentos en las líneas de comando. Cuando entramos en el editor este delimitador es una coma (,); esto puede ser cambiado usando el comando S, el delimitador será tomado del primer carácter de la cadena 'd'. Recuerda que una vez que hayas definido un nuevo delimitador éste debe ser usado (también con el comando S) hasta que definamos otro. No confundas este comando con el sub-comando de sustitución dentro del modo de edición de líneas.
Observa que el delimitador no puede ser un espacio.

Comando: Un
Te permite seleccionar el valor del tope de la memoria a ‘n’. Si no se introduce 'n' (tecleando solo U y ¡ENTER¿) se nos mostrará el valor actual del tope de la memoria.
GENS4 no permitirá al fichero de texto o al código objeto pasar del tope de memoria y nos devolverá un mensaje de error si se llega a este tope.
Por defecto, el tope de la memoria es seleccionado inicialmente con el tope de la pila del Spectrum.

Comando: V
El comando V nos muestra útil información: los valores por defecto de los parámetros N1 y N2, el delimitador por defecto, el inicio y el final del texto (en decimal) y el valor de la primera cadena. S1.

Comando: W n,m
El comando W causa que la sección del texto entre las líneas n y m ambas inclusive sean enviadas a la impresora. Si no se introducen n ni m se imprimirá todo el texto. La impresión se parará cada vez que se haya imprimido el número de líneas especificado por el comando K - pulsa una tecla para continuar la impresión.

Comando: X n
El comando X nos da un catalogo de la unidad de microdrive n. En el modo de 51 columnas, la pantalla es borrada primero. El catálogo siempre es mostrado en 32 columnas.

Comando: Z
Este comando borra todo el texto que haya en memoria, por supuesto, antes de hacerlo pide confirmación antes de proceder a hacerlo: responde Y (sí) o N (no) para borrarlo o no. Aparte de que es más rápido y corto que usar el comando D1, 32767. el comando Z te permite borrar todo el texto si éste ha sido corrompido de alguna manera. Por ejemplo, puedes haber cargado un fichero de código en lugar de uno de texto por error.

Comando: H
Muestra la pantalla de ayuda, la cual es una lista de los comandos disponibles en dos columnas con la letra de cada comando en mayúsculas, por ejemplo. el comando V se nos muestra como 'current values'.

3.3 Un ejemplo del uso del editor

Vamos a suponer que has tecleado el siguiente programa (usando I10. 10)

```
10      *h                  números aleatorios de 16 bits
20
30      ;ENTRADA:           HL contiene la semilla
40      ;SALIDA:            HL contiene el número aleatoria
50
60      Random              PUSH AF             ; guardar los registros
70      PUSH BC
80                          PUSH HL
90                          ADD  HL, HL         ;* 2
100                         ADD  HL, HL         ;* 4
110                         ADD  HL, HL         ;* 8
120                         ADD  HL, HL         ;*16
130                         ADD  HL, HL         ;*32
140                         ADD  HL; HL         ;* 64
150                         PIP  BC             ;antiguo número semilla
160                         ADD  HL, DE
170                         LD   DE, 41
180                         ADD  HL, DE
190                         POP  BC
200                         POP  AF             ;recuperar los registros
210                         REY
```

Este programa tiene una buena cantidad de errores:

- **Línea 10**: una h minúscula ha sido usada para el comando del ensamblador *H.
- **Línea 40**: se ha colocado aleatoria en lugar de aleatorio.
- **Línea 70**: PUSH BC comienza en el campo de las etiquetas.
- **Línea 150**: PIP en lugar de POP.
- **Línea 160**: necesita un comentario (no es un error-puro estilo)
- **Línea 210**: REY en lugar de RET.
- Además hay que insertar dos líneas ADD HL, HL entre las líneas 140 y 150 y todas las referencias al par de registros DE en las líneas 160 a 180 deberían hacer referencia al par de registros BC.

Para poner todo esto correctamente debemos hacer los siguiente:

```
E10   ;ENTER¿     entonces (1 espacio) C (modo cambios) H ¡ENTER¿
      !ENTER¿

F40,40,aleatoria,aleatorio ¡ENTER¿
      entonces el sub-comando S.

E70   ;ENTER¿ entonces I (modo inserción) (1 espacio) ¡ENTER¿
      ;ENTER¿

I142.2 ;ENTER¿

142                 ADD  HL,HL        ;*128
144                 ADD  HL,HL        ;*256
;EDIT¿
```

F150, 150, PIP, POP ;ENTER¿
seguido del sub-comando S

E160 ¡ENTER¿ X (dos espacios) ; *257 + 41 ¡ENTER¿ ! ENTER¿

F160, 180, DE, BC ¡ENTER¿
entonces usa repetidamente el sub-comando S.

E210 ;ENTER¿ ¡CURSOR DERECHA? (2 espacios) C (modo cambios) T
¡ENTER¿ ¡ENTER¿

N10, 10 ¡ENTER¿ para renumerar el texto.

Es muy importante que trabajes a través de este ejemplo usando el editor.

APENDICE 1 NUMEROS DE ERROR Y SUS SIGNIFICADOS

ERROR	1	Error en el contexto de esta línea.
ERROR	2	Mnemónico no reconocido.
ERROR	3	Sentencia mal formada.
ERROR	4	Símbolo (etiqueta) definido más de una vez.
ERROR	5	La línea contiene un carácter erróneo, es decir, uno que no es valido en un contexto particular.
ERROR	6	Uno de los operandos de esta línea es erróneo.
ERROR	7	La etiqueta de esta línea es una Palabra Reservada.
ERROR	8	Error en los registros.
ERROR	9	Demasiados registros en esta línea.
ERROR	10	Una expresión que debería dar 8 bits da un número mayor de 8 bits.
ERROR	11	Las instrucciones JP (IX+n) y JP (IY+n) son erróneas.
ERROR	12	Error en la formación de una orden del ensamblador.
ERROR	13	Error de referencia, por ejemplo: un EQU se ha hecho sobre una etiqueta que todavía no ha sido definida.
ERROR	14	División entre cero.
ERROR	15	Desbordamiento en una multiplicación.
ERROR	16	Definiciones de macros anidadas.
ERROR	17	El identificador no es un macro.
ERROR	18	Llamada anidada a un macro.
ERROR	19	Sentencias condicionales anidadas.
Bad ORG!		Un ORG ha sido hecho en una dirección que corrompería al GENS, su fichero de texto o la tabla de símbolos. El control volverá al editor.
Out of Table Space!		Sucede durante el primer paso si se ha reservado memoria insuficiente para la tabla de símbolos. El control volverá al editor.
Bad Memory!		No hay suficiente espacio para insertar más texto, ya que el texto está cerca del final de la memoria. Deberás grabar el fichero de texto, o una parte de él.

APENDICE 2 PALABRAS RESERVADAS, MNEMONICOS, ETC.

Lo siguiente es una lista da las Palabras Reservadas de GENS. Estos símbolos no puedan ser usados como etiquetas, pero sí que lo pueden hacer como parte de ellas. Todas las palabras reservadas están compuestas por letras mayúsculas.

A	B	C	D	E	H	L	I	R	S
AF	AF’	BC	DE	HL	IX	IY	SP	NC	Z
NZ	M	P	PE	PO					

Y esto es una lista de los mnemónicos válidos para el Z80, las órdenes del ensamblador y los comandos del mismo. Estos también deben ser introducidos en mayúsculas.

ADC	ADD	AND	BIT	CALL
CCF	CP	CPD	CPDR	CPI
CPIR	CPL	DAA	DEC	DI
DJNZ	EI	EX	EXX	HALT
IM	IN	INC	IND	INDR
INI	INIR	JP	JR	LD
LDD	LDDR	LDI	LDIR	NEG
NOP	OR	OTDR	OTIR	OUT
OUTD	OUTI	POP	PUSH	RES
RET	RETI	RETN	RL	RLA
RLC	RLCA	RLD	RR	RRA
RRC	RRCA	RRD	RST	SBC
SCF	SET	SLA	SRA	SRL
SUB	XOR			

DEFB	DEFM	DEFS	DEFW	ELSE
END	ENT	EQU	IF	ORG

MAC	ENDM

*D	*E	*H	*L	*S
*C	*F	*M		

APENDICE 3 UN EJEMPLO PRACTICO

Aquí sigue un ejemplo de una típica sesión de trabajo usando GENS4 - si eres un usuario nuevo en el mundo de loas programas en ensamblador o si simplemente no estás completamente seguro de entender el funcionamiento del editor/ensamblador, es muy importante que trabajes con este ejemplo cuidadosamente. Observa que ;ENTER¿ es usado pare indicar que debes pulsar la tecla ENTER del teclado.

Objetivo de la Sesión:

Escribir y comprobar una rutina de multiplicación de enteros, el texto de la cual va a ser grabado en cinta usando el comando del editor T para luego ser incluida desde cinta en posteriores programas.

Plan da trabajo da la sesión:

1. Escribir la rutina de multiplicación como una subrutina y grabarla en cinta usando el comando P para que podamos revisarlo fácilmente durante esta sesión, por si hay algunos errores.
2. Corregir los errores de la rutina, editando cuando sea necesario.

3. Grabar finalmente la rutina corregida en cinta, usando el comando T para que la podamos 'incluir' de cinta en otros programas.

Antas de comenzar debemos cargar GENS4 en la memoria - para hacerlo teclea:

LOAD “” CODE 26000 ;ENTER¿

para cargarlo en la dirección 26000. Ahora teclea:

RANDOMIZE USR 26000 ;ENTER¿

Ahora entras en el editor y ya puedes comenzar a crear programas en ensamblador.

Paso 1 - Escribir la rutina de multiplicación de enteros

Usamos el comando I para introducir el texto y usamos ¡CURSOR DERECHA¿ (el carácter de tabulador) para obtener un listado ya tabulado. No necesitamos usar ¡CURSOR DERECHA¿, un listado del texto realizará automáticamente la tabulación. No hemos indicado dónde debemos usar los tabuladores en el listado pero puedes suponer que ellos han sido usados antes del mnemónico y entre el mnemónico y los operandos. Observa que las direcciones mostradas en las direcciones que siguen no corresponderán a las producidas en tu ordenador: sólo están con un propósito ilustrativo.

```
T10.10 ;ENTER¿
    10    ;Una rápida rutino de multiplicar enteros ¡ENTER¿
    20    ;Multiplica el valor de HL ¡ENTER¿
    30    ;por DE. Retorna el resultado ¡ENTER¿
    40    ;en HL. El flag C activado si ¡ENTER¿
    50    ;se produce desbordamiento ¡ENTER¿
    60    ¡ENTER¿
    70              ORG   R7F00        ¡ENTER¿
    80    ¡ENTER¿
    90    Mult      OR    A            ¡ENTER¿
   100              SBC                HL, DE      ;HL> DE?     ¡ ENTER¿
   110              ADD                HL, DE      ¡ENTER¿
   120    JR.       NC, Mul            ;si         ¡ENTER¿
   130    EX        DE, HL             ¡ENTER¿
   140    Mul       OR          D      ¡ENTER¿
   150    SCF       ;desbordamiento si ¡ENTER¿
   160    RET        NZ ;DE>255        ¡ENTER¿
   170    OR        E ;0 veces?   ¡ENTER¿
   180    LD        E, D    ¡ENTER¿
   190    JR        NZ, .MU4 ;no ¡ENTER¿
   200    EX        DE, HL;0 ¡ENTER¿
   210    RET    ¡ENTER¿
   220    ¡ENTER¿
   230    ;Rutina principal.      ¡ENTER¿
   240    ¡ENTER¿
   250    Mu2       EX    DE, HL       ¡ENTER¿
   260              ADD   HL, DE       ¡ENTER¿
   270              EX    DE, HL       ¡ENTER¿
   280    Mu3       ADD   HL, HL       ¡ENTER¿
```

```
290             RET   C       ;desbordamiento      ¡ENTER¿
300     Mu4     RRA   ¡ENTER¿
310             JR    NC, Mu3       ¡ENTER¿
320             OR    A             ¡ENTER¿
330             JR    NZ, Mu2       ¡ENTER¿
340             ADD   HL, DE        ¡ENTER¿
350             RET   ¡ENTER?
360     ;EDIT¿
```

Lo anterior creará el texto de la rutina, ahora lo grabamos en cinta usando:

```
>P10, 350, Mult        ¡ENTER¿
```

Recuerda que el cassette debe estar en modo de grabación antes da pulsar ENTER.

Paso 2 - corrección de la rutina

Primero, vamos a ver si el ensamblado es correcto. Usaremos la opción 2 para que no se produzca ningún listado y el código objeto no sea generado.

```
>A2    ¡ENTER¿

*HISOFT GENS4 ASSEMBLER*
Copyright Misoft 1983, 84, 87
ALL Rights Reserved

Pass 1 errors 00
Pass 2 errors 00

*WARNING* MU4 absent Table used:                74 from 161
>
```

Vamos, que hemos cometido un error en la línea 190, ya que hemos entrado MU4 en lugar de Mu4 que es la etiqueta a la queremos ir.
Por lo tanto, editamos la línea 190:

```
>F190, 190, MU4, Mu4 !ENTER?
    190         JR      NZ,     (ahora usa el sub-comando S)
```

Ahora vuelve a ensamblar el texto y deberás encontrar que esto se produce sin errores. Ahora vamos a escribir algo más detexto para comprobar que la rutina funciona:

```
>N300, 10 ¡ENTER¿          (renumerar de manera que podamos escribir algo más de texto.)

>I10, 10        ¡ENTER¿

10      ;Código para verificar ¡ENTER¿
20      ;la rutina Mult.        ¡ENTER¿
30      ¡ENTER¿
```

```
40                  LD      HL, 50          ¡ENTER¿
50                  LD      DE, 20          ¡ENTER?
60                  CALL    Mult    ;multiplicar    ¡ENTER¿
70                  LD      A, H    ;imprimir el resultado ¡ENTER?
80                  CALL            Aout    ¡ENTER¿
90                  LD      A, L    ¡ENTER¿
100                 CALL    Aout    ¡ENTER¿
110                 RET     ;volver al editor  ¡ENTER¿
120     ;ENTER¿
130     ;Rutina para imprimir A en hexadecimal      ¡ENTER¿
140     ¡ENTER¿
150     Aout        PUSH    AF      ¡ENTER¿
160                 RRCA            ¡ENTER¿
170                 RRCA            ¡ENTER¿
180                 RRCA            ¡ENTER¿
190                 RECA            ¡ENTER¿
200                 CALL            Nibble ¡ENTER¿
210                 POP             AF      ¡ENTER¿
220     Nibble      AND             %1111 ¡ENTER¿
230                 ADD             A,R90 ¡ENTER¿
240                 DAA             ¡ENTER¿
250                 ADC             A,R40 ¡ENTER¿
260                 DAA             ¡ENTER'
270                 LD              IY, R5C3A     ;para ROM     ¡ENTER¿
280                 RST             R10           ;llamada a la ROM ¡ENTER¿
290                 RET     ¡ENTER¿
300 ¡EDIT¿
```

Ahora ensambla la rutina de comprobación y la de multiplicación juntas:

```
>A2    ¡ENTER¿

*HISOFT GENS4 ASSEMBLER*
Copyright Hisoft 1983, 84, 87
ALL Rights Reserved

7EAC 190            RECA
*ERROR* 02          (pulsa cualquier tecla para continuar)

Pass 1 errors: 01

Table used:         88 from 210
```

Tenemos un error en nuestra rutina; RECA en lugar de RRCA en la línea 190, por lo tanto:

```
> E190
190                 RECA
190     ¡CURSOR DERECHA¿ (1 espacio) C (modo cambio) R ¡ ENTER¿
        ¡ENTER¿
>
```

Ahora vuelve a ensamblar la rutina usando las opciones por defecto (solo usa A ¡ENTER¿) y el texto deberá ser ensamblado correctamente. Suponiendo que lo hace, ahora podemos comprobar si la rutina funciona, por lo tanto necesitamos que el editor sepa dónde debe ejecutar la rutina. Esto lo hacemos con la orden ENT de la siguiente forma:

```
>35          ENT          $          ¡ENTER¿
```

Volvemos a ensamblar al texto; el ensamblado finalizará correctamente con los mensajes:

```
Table used:   88 from 211
Executes:     32416

>
```

o algo similar. Ahora podemos ejecutar nuestro código mediante el comando del editor R. Debemos esperar que la rutina multiplique 50 por 20 lo que da 1000 (R3E8 en hexadecimal).

```
>R      ¡ENTER¿
0032>
```

¡No funciona! ¿Por qué? Lista las líneas 380 a 500 (L380.500). Verás que en la línea 430 hay una instrucción OR D seguida, efectivamente de RET NZ. Lo que esto hace es la operación lógica OR entra el registro D y el acumulador A y retorna con un flag de error seleccionado (el flag C) si el resultado no es 0. El objetivo de esto es asegurarse de que DE<256 para que la multiplicación no produzca desbordamiento - y para hacerlo comprueba si D es cero... pero el OR no funcionará correctamente si el acumulador A no es cero y no nos hemos asegurado de ello. Por lo tanto:

```
>E380        ¡ENTER¿
   380       Mult   OR     A
   380       !CURSOR DERECHA¿ I (modo inserción) X ¡ENTER¿ ¡ENTER¿
>
```

Ahora ensamblamos y ejecutamos el código de nuevo, usando R. La respuesta debería ser correcta:

83E85

Ahora podemos aseguraros más, editando las líneas 40 y 50 para multiplicar diferentes números, ensamblando y volviendo a ejecutar; deberías comprobar que la rutina ahora ya funciona correctamente.
Ahora que ye hemos perfeccionado la rutina podemos grabarla en cinta en el formato de 'inclusión'.

```
>T300.999. Mult ¡ENTER¿
```

Recuerda que debes pulsar RECORD en el cassette antes de pulsar la tecla ¡ENTER¿.
Si quieres incluirle de microdrive no necesitas usar el comando T; los programas grabados normalmente con P pueden eser incluidos desde el microdrive.
Une vez que hayas grabado el texto puedes incluirlo en tus programas de la manera siguiente:

```
500                RET
```

```
510
520     ;incluir la rutina de multiplicación aquí
530
540     *F Mult
550
560     :aquí va la siguiente rutina
```

Cuando ensamblamos el texto anterior el ensamblado muestra el mensaje 'Start Tape..' cuando llega a la línea 540 tanto en el primer paso como en el segundo. Esto significa que deberemos rebobinar la cinta después del primer paso. Puedes grabar dos veces el texto con T, uno seguido del otro, para que uno sea usado en el primer paso y el otro en el segundo.
Cuando incluyes del microdrives, no se muestran mensajes; todo sucede automáticamente.
Es conveniente que estudies este ejemplo cuidadosamente y comiences a introducir algunos cambios o introduzcas otros programas hechos por ti. de manera que consigas dominar a la perfección el sistema de edición y ensamblado de textos.

Hisoft MONS

Desensamblador/Corrector

CONTENIDO

situar un 'breakpoint'
salto direccionado
entrar ASCII
paso simple
ejemplo preáctico
imprimir el listado
Modificación de Memoria
Modificación de Registros

APENDICE UN EJEMPLO DEL ASPECTO DE LA PANTALLA

SECCION 1 *EMPEZANDO A MANEJAR MONS4*

MONS4 -incorpora un sistema de auto-reubicación: simplemente tienes que cargarlo en la dirección dende desees ejecutarlo y entonces entrar en MONS4 mediante esa dirección. Si deseas volver a entrar en él (habiendo dejado MONS4 para volver al BASIC) deberás ejecutar de nuevo la dirección en la que lo cargaste.
Los usuarios de Plus 3 deberán leer el folleto referente a su ordenador, puesto que ellos disponen de un MONS4 que se introduce en la memoria del disco RAM y sólo necesita 100 bytes de la memoria RAM del 48k normal.
Ejemplo:
Digamos que quieres cargar MONS4 en la dirección BC000 (49152 en decimal) - deberás hacer lo siguiente:

LOAD "" CODE 49152 ¡ENTER¿
RANDOMIZE USR 49152 ¡ENTER¿

Para volver a MONS4 usa:

RANDOMIZE USR 49152 ¡ENTER¿

MONS4 ocupa cerca de 6K de longitud una vez que ha sido reubicado pero deberás dejarle unas 7K para cargarlo, ya que posee una tabla de reubicación después del código principal. MONS4 contiene su pila interna por lo que es un programa auto-contenido.
MONS4 es cargado, por defecto, en la dirección 55000, pero normalmente querrás cargarlo en una dirección especial; normalmente es conveniente cargar MONS4 en una dirección de memoria alta.

Realización de una copia de seguridad
Una vez que hayas cargado MONS4 en la memoria de tu Spectrum puedes hacer una copia de seguridad de él de la mamera siguiente:

SAVE "MONS4" CODE xxxxx. 6656 ¡ENTER¿ en casette.
SAVE *"M";1;"MONS4" CODE xxxxx.6656 ¡ENTER¿ en Microdrive.
SAVE *"M";1;"MONS4" CODE xxxxx.6780 ¡ENTER¿ para el disco Opus.

donde: xxxxx es la dirección en la cual has cargado MONS4.

Observa que hemos permitido que se pueda hacer una copia del programa para tu PROPIO uso. por lo tanto puedes usarla para programar en confidencia. No copies MONS4 para dar (o mucho peor. vender, a tus amigos). Creemos dar una relación calidad/precio mas que aceptable y damos

una gran cantidad de servicios post-venta, ten en cuenta que si mucha gente copia nuestros programas no podremos continuar haciendo esto:

POR FAVOR: COMPRA, NO ROBES

Habiendo entrado en MONS4 se te presentará un panel frontal (ver el APÉNDICE para ver un ejemplo). Esto consiste en los registros y flags del Z80 junto a sus contenidos más una sección de 24 bytes de memoria centrados alrededor del Puntero de Memoria, inicialmente seleccionado a 0. En la línea superior se encuentra el desensamblado de la instrucción direccionada por el Puntero de Memoria.
Cuando entramos en MONS4, todas las direcciones se nos muestran en hexadecimal (es decir, sobre base 16); puedes cambiar esto de manera que las direcciones se muestren en decimal usando el comando ¡SYMBOL SHIFT¿ 3 - mira la siguiente sección. Observa, sin embargo, que las direcciones deben ser SIEMPRE introducidas en hexadecimal. Los comandos son entrados desde el teclado como respuesta al signo > que se encuentra bajo los bytes de memoria y pueden ser entrados tanto en mayúsculas como en minúsculas.
Algunos comandos, cuyo efecto podría ser desastroso si se usan erróneamente, necesitan que se pulse ¡SYMBOL SHIFT¿ junto a la letra del comando. A través de este manual el uso de ¡SYMBOL SHIFT¿ será sustituido por el símbolo ^. ejemplo: ^Z significa que debes mantener pulsada la tecla ¡SYMBOL SHIFT¿ y, sin soltarla, pulsar Z.
Los comandos se producen inmediatamente, no es necesario pulsar ¡ENTER¿. Los comandos inválidos simplemente son ignorados. El 'panel frontal' se vuelve a escribir cada vez que se pulsa algún comando y así puedes observar el resultado del mismo.
Muchos comandos necesitan que entremos números en hexadecimal - cuando se entra un número en hexadecimal se hace introduciendo los dígitos hex. (0-9 y A-F o a-f) y cuando quieras acabar la introducción del número sólo debes pulsar una tecla que no sea un dígito hexadecimal válido. Si el carácter introducido finalmente es un comando válido, éste será procesado después de efectuar la acción del anterior. Si es un signo menos '-' el número hexadecimal entrado será retornado en negativo - en complemento a dos. Ejemplo: 1800- da E800. Si entras más de cuatro dígitos cuando teclees un número en hexadecimal sólo los últimos 4 serán almacenados y mostrados en pantalla.
Para volver al Basic simplemente pulsa ¡STOP¿ (^A).

LOS USUARIOS DE PLUS 3 DEBERAN CONSULTAR LAS INSTRUCCIONES ADICIONALES PARA ESTE ORDENADOR ANTES DE CONTINUAR LEYENDO.

SECCION 2 *LOS COMANDOS DISPONIBLES*

Lo siguiente es la lista de todos los comandos disponibles por MONS4. En esta sección, cada vez que ¡ENTER¿ es usado para terminar un número hexadecimal también lo puede ser cualquier carácter no hexadecimal (ver Sección 1). Además _ es usado para representar un espacio.

¡SYMBOL SHIFT¿ 3 o R cambiar hex/dec

Cambiar la base en la que las direcciones son mostradas entre base 16 (hexadecimal) y base 10 (decimal). Al entrar en MONS4, las direcciones son mostradas en hexadecimal. Usa ^3 para cambiarlas a decimal y pulsa de nuevo ^3 para volver a ponerlas en hexadecimal. Esto afecta a todas las direcciones mostradas por MONS4 incluyendo aquellas que genere el desensamblador pero no cambia los contenidos de las direcciones de memoria - éstas siempre se muestran en hexadecimal.

¡SYMBOL SHIFT¿ 4 o S desensamblado de una página

Mostrar una página de desensamblado comenzando por la dirección contenida en el Puntero de Memoria. Es útil para examinar las direcciones actuales para saber las instrucciones que vienen a continuación. Pulsa ^4 o ¡EDIT¿ para volver al Panel Frontal o cualquier otra tecla para avanzar otra página.

¡ENTER¿ avanzar 1

Incrementar el Puntero de Memoria en uno de manera que los 24 bytes de memoria mostrados se centran en la siguiente posición.

CURSOR ARRIBA retroceder 1

Decrementar el Puntero de Memoria en uno.

CURSOR IZQUIERDA retroceder 8

Decrementar el Puntero de Memoria en 8 - usado para retroceder rápidamente.

CURSOR DERECHA avanzar 8

Incrementar el Puntero de Memoria en 8 - usado para avanzar rápidamente.

, (coma) tomar le pile

direccionar el Puntero de Memoria de manera que contenga la dirección actualmente contenida por la pila (indicada por SP). Esto es útil cuando quieres ver la dirección de retorno de una rutina, etc.

G buscar una cadena

Buscar en la memoria una cadena de bytes especifica.
Se te muestra el signo **:** y deberás entrar en ese momento el primer byte que quieres buscar seguido de ¡ENTER¿ Ahora puedes seguir introduciendo los valores siguientes (y ¡ENTER¿) como respuesta a **:** hasta que hayas introducido la cadena entera.
Entonces pulsa ¡ENTER¿ como respuesta a los dos puntos: esto finalizará la definición de la cadena y la buscará en la memoria comenzando a partir de la dirección contenida por el Puntero de Memoria. Cuando se encuentren el panel frontal y el Puntero de Memoria serán posicionados en el primer carácter de la cadena. Ejemplo:
Digamos que quieres buscar en la memoria, empezando en R8000, las apariciones de la cadena R3E RFF (2 bytes) - deberías hacerlo de la siguiente manera:

M:8000 ¡ENTER¿	seleccionar el Puntero de Memoria en R8000
G:3E ¡ENTER¿	definir el primer byte de la cadena.
FF ¡ENTER¿	definir el segundo byte de la cadena.
¡ENTER¿	terminar la cadena.

Después del ¡ENTER¿ final (o cualquier otro carácter no hexadecimal) G procede a buscar en la memoria desde R8000 la primera aparición de R3E RFF. Cuando la encuentra nos muestra su dirección - para buscar las siguientes apariciones usa el comando N.

H convertir a hex

Se te pregunta con un **:** un número decimal terminado en un carácter que no sea un dígito (cualquier carácter menos 0...9 inclusive). Une vez que el número ha sido terminado, se nos mostrará un signo = seguido del valor hexadecimal equivalente al número decimal. Ahora puedes pulsar cualquier tecla para volver al modo de entrada de comandos.
Ejemplo:

H:41472_=A200 aquí se ha usado un espacio como terminador

I copia inteligente

Esto es usado para copiar un bloque de memoria de una dirección a otra - es inteligente en el sentido de que el bloque puede copiarse en zonas en las que se sobre-escriba. I te pregunta el inicio y el final (ambos inclusive) del bloque a ser copiado (First:, Last:) y entonces la dirección en la que deseamos mover el bloque (To:); entra los números hexadecimales como respuesta a cada pregunta. Si la dirección de inicio es mayor que la del final el comando no producirá ningún efecto - de cualquier otra manera el bloque será movido.

J saltar a una dirección

Ejecutar código desde una dirección especifica.
Este comando te pregunta, mediante **:** ,un número hexadecimal - una vez que éste haya sido entrado la pila interna de MONS4 es reseleccionada, la pantalla es borrada y la ejecución transferida a la dirección especificada. Si deseas retornar al Panel Frontal después de la ejecución, deberás introducir un breakpoint (mira el comando W para saber lo que son) en el punto donde desees que se retorne.
Ejemplo:

J:B000 !ENTER¿ ejecuta el código empezando en RB000.

Puedes abortar este comando antes de terminar la dirección usando ¡EDIT¿ Observa que J corrompe los registros del Z80 antes de ejecutar el código; de esta forma el programa a ejecutar no deberá esperar ningún valor de estos registros. Si deseas ejecutar código con los registros seleccionados en unos valores particulares deberás usar ¡SYMBOL SHIFT¿ K.

¡SYMBOL SHIFT¿ K continuar la ejecución
Continuar la ejecución desde la dirección contenida por el Contador de Programa (PC).
Este comando será probablemente usado en conjunción con el comando W - un ejemplo te aclarará mejor su uso:
Digamos que estás ejecutando paso a paso con el comando ^Z (paso simple) una rutina y has examinado hasta la dirección B8920. Ahora no te interesa ver la rutina en R9000 pero quieres saber cómo afecta la rutina que hay en R8800 a los flags (indicadores) después de ser llamada.

```
891E    3EFF        LD    A, -l
8920    CD0090      CALL  R9000
8923    2A0080      LD    HL, (R8000)
8926    7E          LD    A, (HL)
8927    111488      LD    DE, R8814
892A    CD0088      CALL  R8800
892D    2003        JR    NZ, etiq
```

```
892F    320280          LD      (R8002), A
8932    211488  etiq    LD      HL, R8814
```

Procede como sigue: selecciona un breakpoint, usando W, en la dirección R892D (recuerda usar M primero para posicionar el Puntero de Memoria) y entonces manda un comando ^K. La ejecución continua desde la dirección contenida en PC que, en este caso, ese R8920. La ejecución continuará hasta que se llegue a la dirección en la que se haya posicionado el breakpoint. Una vez llegado a este punto, se devolverá el control el MONS4, y podrás inspeccionar el estado de los flags. etc. después de la llamada a la subrutina en R8800. Ahora puedes seguir usando ^Z para avanzar instrucción a instrucción.
Por lo tanto, ^K es útil para ejecutar el código sin que se borre la pila interna o se corrompan los registros, como lo hace J.

L **Listar memoria**

Tabular, o listar un bloque de memoria comenzando en la dirección contenida actualmente por el Puntero de Memoria.
L borra la pantalla y muestra la representación hexadecimal y los caracteres ASCII equivalentes de 80 bytes de memoria comenzando en el valor actual del Puntero de Memoria. Las direcciones serán mostradas en hexadecimal o decimal dependiendo del estado actual del Panel Frontal (mira el comando ^3).
La pantalla se nos muestra en 20 líneas a 4 bytes por cada una. Los códigos ASCII se muestran al final de cada línea. El código ASCII es representado de la siguiente manera: los códigos mayores de 127 son decrementados en 128 y los valores entre 0 y 31 inclusive son mostrados como un punto ‘**.**’
Al final de cada página tienes la opción de retornar al panel frontal pulsando ¡EDIT¿ o continuar con la siguiente página de 80 bytes pulsando cualquier otra tecla.

M **seleccionar la dirección de Memoria**

Seleccionar el Puntero de Memoria a una dirección especifica.
Se te pregunta mediante un '**:**' una dirección hexadecimal (ver la Sección 1). El Puntero de Memoria se posicionará en la dirección entrada y el panel frontal será cambiado acorde a ello.
M es útil como un preludio para entrar código, listar la memoria, etc.

N **siguiente búsqueda**

Encontrar la siguiente aparición de una cadena de bytes entrada anteriormente con el comando G.
G te permite definir une cadena y entonces buscar la primera aparición de ella; si quieres buscar más apariciones deberás usar N. N comienza la búsqueda desde el Puntero de Memoria y cuando lo encuentra selecciona el puntero en esa dirección.

O **salto relativo**

Ir a una dirección de un desplazamiento relativo.
El comando coge al byte que se encuentre actualmente direccionado por el Puntero de Memoria, lo trata como un desplazamiento relativo y sitúa el puntero de memoria en la posición correspondiente a él.
Ejemplo:
Digamos que el Puntero de Memoria está en la dirección R6800 y que el contenido de las direcciones R67FF y R6800 son R20 y R16 respectivamente - esto puede ser interpretado como una instrucción JR NZ, $+24. Para encontrar la dirección a la que se iría si la condición No-Cero fuese

cierta (NZ) simplemente pulsa O cuando el Puntero de Memoria está en la dirección del byte de desplazamiento (R16). El puntero de memoria sería situado en la posición R6817, la dirección a la que saltaría la instrucción JR NZ, $+24.
Recuerda que los desplazamientos relativos mayores de R7F (127) son tratados como negativos por el Z80: O lo tiene en cuenta.
Mira también el comando U en conexión con el comando O.

P **rellenar memoria**

Rellenar la memoria entre los límites especificados con un byte determinado.
P te pregunta 'First:* (primero), 'Last:' (último) y 'With:' (con). Entra los números decimales correspondientes en respuesta a cada pregunta; respectivamente el inicio y las direcciones finales (ambas inclusive) del bloque a rellenar y el byte con el que queremos rellenar la memoria.
Ejemplo:

```
P
First:7000 ¡ENTER¿
Last:77FF ¡ENTER¿
With:55 ¡ENTER ¿
```

rellenaría las direcciones R7000 a R77FF (inclusive) con el byte R55 (U). Si el inicio es mayor que el final el comando P es abortado.

Q **cambiar registros**

Cuando entramos en el panel frontal, éste nos muestra los registros estándar (AF, HL, DE, BC). Si usamos Q se nos mostrarán los registros Alternativos (AF', HL', DE', BC') que se distinguen de los primeros por el apóstrofe ' después del nombre del registro.
Si se usa Q cuando se nos muestran los registros alternativos se pasará a mostrar los estándar y viceversa.

¡SYMBOL SHIFT¿ T **ignorar llamada**

Selecciona un 'breakpoint' después de la instrucción actual y continúa la ejecución.
Ejemplo:

```
9000   B7       OR     A
9001   C20098   CALL   NZ, R9800
9004   010000   LD     BC,0
9800   21FFFF   LD     HL, -1
```

Estás ejecutando paso a paso el código anterior y estás en la posición R9001 donde el valor de A no es cero, por lo tanto el flag de cero (Z) sería NZ después de la instrucción OR A. Si usas ahora ^Z para continuar la ejecución paso a paso, la ejecución continuará a partir de R9800, la dirección de la subrutina. Si no quieres inspeccionar esa rutina y deseas ignorar la llamada deberás hacer un ^T cuando estés en la dirección R9001 y el CALL será automáticamente ejecutado y la ejecución se parará cuando se vuelva a la dirección R9004 para que sigas inspeccionando.
Recuerda. ^T colocó un 'breakpoint' (ver comando W) después de la instrucción actual y entonces manda un comando ^K.
Mira el comando ^Z para ver la ejecución paso a paso.

T **desensamblar**

Desensamblar un bloque de código, opcionalmente a la impresora y/o el microdrive.
Primero se te pregunta que entres las direcciones inicial y final ('First:'.'Last:') del código a desensamblar: entra estos valores en hexadecimal como se describe en la Sección 1.
Si el inicio es mayor que el final, el comando será abortado. Después de entrar estas direcciones se nos preguntará si deseamos imprimir por la impresora ('Printer?'): si respondemos Y (sólo la letra Y mayúscula) dirigiremos el listado a la impresora y si pulsamos cualquier otra tecla, el listado saldrá por pantalla.
Ahora se te pregunta 'Text:' y puedes entrar, en hexadecimal, la dirección inicial en la que deseas que se almacene el texto desensamblado. Si no quieres crear un fichero de texto simplemente pulsa ¡ENTER¿ después de esta pregunta. Si especificas una dirección, el fichero será almacenado, comenzando en esa dirección, en un formato compatible con el GENS4. Si deseas cargar este texto desde el GENS4 deberás anotar los valores del inicio y el final del texto que se te dan al acabar el desensamblado, retornar al BASIC y grabar el texto como un fichero CODE.
De esta manera podrás cargarlo en GENS4 usando al comando G.
Por otra parte, en lugar de introducir una dirección como respuesta a 'Text:'. puedes entrar un nombre válido de microdrive y el texto será grabado directamente en microdrive a medida que éste es desensamblado. Ejemplo:

Text: 2:PEPE ¡ENTER¿

grabaría el texto desensamblado en el microdrive 2 bajo el nombre PEPE. Si introduces un nombre de microdrive no se mostrará ningún listado en pantalla. El fichero producido en el microdrive puede ser cargado directamente con el comando del editor G del GENS4.
Si, mientras se genera el fichero de texto, el texto se escribiría encima de MONS4, entonces el desensamblado será abortado - pulsa cualquier tecla para volver al Panel Frontal.
Si especificas una dirección se te preguntará 'Workspace:' (espacio de trabajo), esto debe ser la dirección de una zona da memoria libre que es usada como una tabla de símbolos de 2 bytes para cada etiqueta generada por el desensamblador. Si pulsas ¡ENTER¿ se reservarán 4K de espacio por debajo de MONS4.
Después de esto, se te pregunta repetidamente 'First:' y 'Last:' (inclusive) que son las direcciones inicial y final de algunas áreas del código que son datos. Estas áreas en lugar de ser desensambladas como instrucciones lo harán como órdenes DEFB.
Si el valor de un dato está entre 32 y 127 (R20 y R7F) inclusive el carácter ASCII correspondiente será almacenado. Ejemplo: R41 será cambiado por una A después un DEFB. Cuando has acabado de especificar áreas de datos, o si no quieres especificar ninguna, simplemente pulsa ¡ENTER¿ como respuesta a ambas preguntas. El comando T usa un área al final de MONS4 para almacenar las áreas de datos de manera que puedes introducir tantas como memoria quede disponible: cada una ocupa 4 bytes de espacio. Observa que T destruye todos los 'breakpoints' que se hayan podido introducir previamente - ver el comando W.
El byte después de una instrucción RST 8 es desensamblado como un DEFB dado que este byte es tomado por la ROM del Spectrum y nunca ejecutado.
La pantalla se borrará. Si has pedido que se cree un fichero de texto habrá un corto retraso (dependiendo de cómo sea de larga la acción que quieres desensamblar) mientras la tabla de símbolos es construida durante el paso 1.
Una vez hecho esto, el listado desensamblado aparecerá en la pantalla o impresora a no ser que estemos usando un fichero de microdrive. Puedes parar el listado al final de cada línea pulsando

¡ENTER¿ o ¡SPACE¿. seguidamente pulsa ¡EDIT¿ si deseas volver el panel frontal o cualquier otra tecla para continuar el desensamblado.

```
711D        F5
```

Si se entra un código inválido, éste será desensamblado como un asterisco y un NOP parpadeante.
Al final del desensamblado, si hemos introducido que se genere un archivo de texto en memoria, se nos mostrará un mensaje 'End of text xxxxx' (fin del texto xxxxx).
Cuando se acabe el desensamblado, pulsa cualquier tecla para volver al panel frontal.
Las etiquetas son generadas, donde hacen falta (por ejemplo, en C30078), en la forma Lxxxx donde xxxx es la dirección absoluta en hexadecimal de la etiqueta. Si la dirección se encuentra en una posición exterior a los márgenes seleccionados al generar el código, simplemente se sustituirá por la dirección hexadecimal o decimal correspondiente.
Por ejemplo, si estamos desensamblando entre R7000 y R8000, la instrucción C30078 será desensamblada por JP L7800; por otro lado, si desensamblamos entre R9000 y R98000 ésta será desensamblada por JP R7800 o JP 30720, dependiendo del estado del panel frontal (ver comando ^3).
Si una dirección ha sido referida por alguna instrucción dentro del listado desensamblado, la instrucción de esa dirección aparecerá etiquetada con Lxxxx en el campo de etiquetas (antes del mnemónico), pero sólo si el listado se dirige a un fichero de texto.
Ejemplo:

```
T
First: 8B ¡ENTER¿
Last: 9E ¡ENTER¿
Printer? Y
Text: ¡ENTER¿
First: 95 ¡ENTER¿
Last: 9E ¡ENTER¿
First: ¡ENTER¿
Last: ¡ENTER¿
```

produciría algo como esto:

```
008B    FE16            CP      R16
008D    3801            JR      C, L0090
008F    23              INK     HL
0090    37              SCF
0094    C9              RET
0095    BF524E          DEFB    RBF, "R·, "N"
0098    C4494E          DEFB    RC4, "I", "N"
009B    4B4599          DEFB    "K", "E", "Y"
009E    A4              DEFB    RA4
```

U **retornar a un salto relativo**

Usado en conjunción con el comando O.
Recuerda que O lleva el Puntero de Memoria a la posición resultante de efectuar un desplazamiento relativo (el efecto producido por un instrucción JR o DJNZ). U se usa para volver a la posición en la que se introdujo el último comando O. Ejemplo:

7200	47	71F3	77
7201	20	71F4	C9
>7202	F2<	>71F5	F5<
7203	06	71F6	C5

Listado 1 Listado 2

Supongamos que se te está mostrando en el panel frontal el listado 1 y quieres ver la rutina que hay en R842F. Por lo tanto, pulsa X con el panel centrado como se muestra; se pasará a mostrarnos el listado 2. Examinas esta rutina durante un rato y entonces deseas volver a la dirección que contenía el listado 1. Pulsa V y retornarás al modo inicial en el que se te mostraba el listado 1. Como U puede ser usado sólo con el último comando X que se haya enviado, todos los demás son perdidos.

W situar un ‘breakpoint’

Un 'breakpoint', tal como lo concibe MONS4, es una simple instrucción CALL a la rutina de MONS4 que muestra al panel frontal de manera que se permite al programador detener la ejecución del programa e inspeccionar los registros del Z80, los flags y las direcciones de memoria que desee. Para ello, si quieres parar la ejecución de un programa en la dirección, por ejemplo, R9876, usa el comando M para situar el Puntero de Memoria en la dirección R9876 y acto seguido usa W para situar en esa dirección un breakpoint. Los tres bytes de código que había originalmente en B9876 son almacenados y sustituidos por una instrucción CALL que parará la ejecución cuando llegue a ese punto. Cuando se ejecuta esta instrucción CALL, los 3 bytes almacenados son recuperados en R9876 y se vuelve al panel frontal con los registros y flags en el estado en que se encontraban antes de ejecutar esa instrucción. Puedes ahora usar todas las posibilidades de MONS4 de la manera usual.
Notas sobre el uso de los breakpoints:
Cuando se encuentra el breakpoint (o si lo prefieres, punto de ruptura), MONS4 emitirá un tono por el altavoz del Spectrum y esperará a que pulses una tecla.
MONS4 usa el área, al final de él mismo, que originalmente contenía la tabla de reubicación, para almacenar la información acerca de los breakpoints. Esto significa que puedes seleccionar tantos como quieras, siempre que quede memoria; cada breakpoint ocupa 5 bytes. Cuando se ejecuta un breakpoint, MONS4 automáticamente restaura los valores originales de esa dirección.
Observa que, dado que el comando T usa este área, todos los breakpoints se pierden cuando usamos este comando. Los breakpoints sólo pueden ser situados en RAM. Dado que cada breakpoint consiste en 3 bytes de una instrucción CALL hay que tener mucho cuidado en algunos casos excepcionales: Ejemplo:

8000	3E	8008	00
8001	01	8009	00
8002	18	800A	06
8003	06	800B	02
>8004<	AF<	800C	18
8005	OE	800D	F7
8006	FF	800E	06
8007	01	800F	44

Si sitúas un breakpoint en R8004 y ejecutas el código desde 8000 el registro A será cargado con el valor 1 (3E 01), la ejecución transferida a R800A, el registro B cargado con el valor 2, la ejecución transferida a la dirección R8005. Pero R8005 ha sido sobre-escrita al introducir loa bytes del

breakpoint, por lo que ahora hemos corrompido el código y puede ocurrir cualquier cosa. Este tipo de situación es bastante inusual pero debes tenerla siempre en cuenta, y contra ella puedes usar la ejecución paso a paso (^Z) para saber la respuesta.

X **salto direccionado**

Usado para direccionar el Puntero de Memoria con la destinación de una instrucción absoluta CALL o JP.
X coge la dirección de 16 bytes especificada por el byte contenido en la dirección contenida en el Puntero de Memoria y el byte siguiente. Recuerda que primero va el byte menos significativo de la dirección y seguidamente el más significativo. Como ejemplo, digamos que quieres saber dónde llama una rutina da código CD0563; selecciona el Puntero de Memoria (usando el comando M) de manera que se encuentre en la dirección donde está el 05 y pulsa X. El Puntero de Memoria pasará a la dirección R6305. Mira también el comando V en conexión con X.

Y **entrar ASCII**

Y te da una nueva línea en la que puedes entrar caracteres ASCII directamente desde el teclado. Estos caracteres son convertidos a su valor decimal equivalente e introducidos en la memoria comenzando por la dirección contenida en el Puntero de Memoria. La cadena de caracteres deberá terminar con ¡EDIT¿ y DELETE (o ¡CAPS SHIFT¿ 0) puede ser usado para borrar caracteres de la cadena.

¡SYMBOL SHIFT¿ Z **ejecución paso a paso**

Antes de usar ^Z (o ^T) el contador de programa (PC) debe estar en la dirección de la instrucción que quieres ejecutar.
^Z simplemente ejecuta la instrucción actual y entonces direcciona el panel frontal de manera que se reflejen en él los cambios causados por la ejecución de esta instrucción.
Observa que la ejecución paso a paso puede hacerse en cualquier dirección de memoria (RAM o ROM) pero no puede hacerse en la ROM del Interface 1.
Aquí sigue un amplio ejemplo que te debe aclarar el uso de la mayoría de los comandos disponibles por MONS4 - es urgente que lo estudies cuidadosamente y lo intentes hacer por ti mismo.

Un ejemplo Práctico

Vamos a suponer que tenemos 3 secciones de código en el ordenador, la primera es el programa principal y carga en HL y DE los números y llama a una rutina de multiplicar (la segunda sección) y sacar el resultado en HL y finalmente llama a otra rutina dos veces para imprimir al resultado de la multiplicación en la pantalla (sección 3). El código sería el siguiente:

```
7080    2A0072      LD      HL, (R7200)     ;   SECCION 1
7083    ED5B0272    LD      DE, (R7202)
7087    CD0071      CALL    MULT
708A    7C          L, D    A, H
708B    CD1D71      CALL    AOUT
708E    7D          LD      A, L
708F    CD1D71      CALL    AOUT
7092    210000      LD      HL, 0
```

```
7100    AF    MULT   XOR     A    ;     SECCION 2
7101    ED52         SBC     HL, DE
7103    19           ADD     HL, DE
7104    3001         JR      NC, MUL
7106    EB           EX      DE, HL
7107    B2    MUL    OR      D
7108    37           SCF
7109    C0           RET     NZ
710A    B3           OR      E
710B    5A           LD      E, D
710C    2007         JR      NZ, MU4
710E    EB           EX      DE, HL
710F    C9           RET
7110    EB    MU2    EX      DE, HL
7111    19           ADD     HL, DE
7112    EB           EX      DE, HL
7113    29    MU3    ADD     HL, HL
7114    D8           RET     C
7115    1F    MU4    RRA
7116    30FB         JR      NC, MU3
7118    B7           OR      A
7119    20F5         JR      NZ, MU2
711B    19           ADD     HL, DE
711C    C9           RET

711D    F4    AOUT   PUSH    AF        ;     SECCION 3
711E    0F           RRCA
711F    0F           RRCA
7120    0F           RRCA
7121    0F           RRCA
7122    CD2671       CALL    NIBBLE
7125    F1           POP     AF
7126    E60F  NIBBLE AND     R1111
712B    C690         ADD     A, R90
712D    27           DAA
712E    FD213A5C     LD      IY, R5C3A
7132    D7           RST     R10
7133    C9           RET

7200    1B2A         DEFW    10779
7202    0300         DEFW    3
```

Ahora queremos investigar el código anterior para ver si trabaja o para saber cómo lo hace. Esto lo podemos hacer con la siguiente lista de comandos - es necesario decir que esto es únicamente un ejemplo de cómo ejecutar paso a paso un código, no es necesariamente eficaz pero puede demostrar cómo hacerlo:

```
M:7080 ¡ENTER¿      situar el Puntero de Memoria en R7080
7080.               situar el contador de programa (PC) en R7080
^Z                  paso simple
```

^Z	paso simple
^Z	seguir el CALL
M:7115 ¡ENTER¿	Ignorar el pre-proceso de los números
W	situar un breakpoint
^K	continuar desde R7100 hasta el breakpoint.
^Z	paso simple
^Z	seguir el salto relativo
^Z	paso simple
^Z	‘’
^Z	‘’
^Z	‘’
^Z	‘’
^Z	‘’
^Z	‘’
^Z	retornar de la rutina de multiplicar
^Z	paso simple
^Z	seguir el CALL
M:7128 ¡ENTER¿	situar el Puntero de Memoria.
W	situar un breakpoint
^K	continuar desde R711D hasta el breakpoint
^Z	paso simple
^Z	‘’
^Z	‘’
^Z	‘’
,	mirar la dirección de retorno
W	situar un breakpoint allí
^K	y continuar
^Z	paso simple
,	retornar de la rutina AOUT
W	
^K	
^Z	paso simple
^T	ejecutar el CALL a AOUT.

Es importante que trabajes a través de este ejemplo, primero tecleando el código (ver Modificación de Memoria) o usando GENS4, y efectuando los comandos detallados antes. Encontrarás el ejemplo como una buena ayuda para comprender cómo examinar un programa paso a paso.

‘’ ¡SYMBOL SHIFT¿ P imprimir el listado

Este comando es exactamente el mismo que L, excepto que la salida se hace por la impresora en lugar de la pantalla. Recuerda que, al final de cada página, puedes pulsar ¡EDIT¿ para retornar al panel frontal u otra tecla para seguir.

Modificación de Memoria

El contenido de una dirección dada por el Puntero de Memoria puede ser modificada entrando el número hexadecimal seguido de un terminador (ver Sección 1). Los dos últimos dígitos hexadecimales (si sólo se entra uno éste es rellenado con un cero a la izquierda) y entrada en la dirección actual direccionada por el Puntero de Memoria y entonces el comando (si hay alguno)

especificado por el terminador es obedecido. Si el terminador no es un comando válido simplemente será ignorado. Ejemplo:

F2 ¡ENTER¿	se entra RF2 y el Puntero de Memoria avanza 1
123 ¡CAPS SHXFT¿ 8	se entra R23 y el Puntero de Memoria avanza 8
EM:E00_	se entra R0E en la posición del Puntero de Memoria y se sitúa el mismo en la dirección RE00.
8CO	Se entra R8C y el Puntero de Memoria se sitúa en la posición correspondiente al salto relativo (el comando O) producido por R8C, es decir, a su valor - 115.
2A5D_	R5D es entrado y el puntero no es alterado dado que el terminador es un espacio, no un comando.

Modificación de Registros

Si se entra un número hexadecimal como respuesta a la pregunta '>' y éste se finaliza con un signo '**.**' el numero especificado será situado en el registro actualmente marcado por la flecha > en la parte de los registros del panel frontal.
Al entrar al MONS4 > está situado en la posición del contador de programa (PC) y usando **.** como terminador de un número hexadecimal se cambiará su valor. Para modificar otro registro deberás usar '**.**' sin usarlo de terminador (es decir, un **.** solo) y el puntero de memoria circulará entre los registros PC-AF. Observa que no es posible alterar tanto la pila (SP) como los registros IR.
Ejemplos:
Supongamos que el puntero está situado en PC:

.	situarse en IY
.	situarse en IX
0**.**	seleccionar IX a cero.
.	situarse en H1
123**.**	seleccionar HL a R123
.	situarse en DE
.	situarse en BC
E2A7**.**	seleccionar BC a RE2A7
.	situarse en AF
FF00.	seleccionar a RFF y borrar todos los flags.
.	situarse en PC
8000.	seleccionar PC a R8000.

Observa que **.** puede también modificar loa registros alternativos si estos se encuentran en pantalla. Usa el comando Q para cambiar entra alternativos y estándar.

APENDICE ***UN EJEMPLO DEL PANEL FRONTAL***

(ver la página MON-21 del manual inglés)

En ella se muestra un ejemplo típico de la pantalla cuando te encuentras en el panel frontal - esta pantalla en concreto es una de las que aparecen al ejecutar paso a paso la rutina MULT del ejemplo del comando Z.
Las primeras 9 líneas contienen los registros del Z80; primero el nombre de los registros (PC a IR), seguidamente (de PC a BC) el valor contenido por cada uno y finalmente el contenido de 7 direcciones de memoria comenzando desde la dirección de este registro. El registro F (de flags) se decodifica de manera que muestre los flags activados - si el registro F estuviese a RFF se nos mostraría como 00FF SZ H VNC. A la derecha de IR está la palabra ON o OFF que indica el estado de las interrupciones. Un puntero > indica el registro actual; ver le Sección 2 - Modificación de Registros
Los 24 bytes de memoria mostrados en la parte de abajo son organizados como la dirección (2 bytes, 4 caracteres) seguido de los contenidos (1 byte, 2 caracteres) de la memoria de esa dirección. Estos bytes son centrados alrededor del Puntero de Memoria, indicado por > <.
Los comandos (ver Sección 2) son entrados en la línea inferior de la pantalla como respuesta al signo >. El panel frontal se vuelve a imprimir cada vez que se procesa un comando.

BIBLIOGRAFIA

Lo siguiente es la lista de libros recomendados para programar el lenguaje ensamblador en el Spectrum.

The Complete Spectrum ROM disassembly	Dr. Ian Logan Dr. F. O'Hara	Melbourne House ISBN 0 86759 117 X
Programación del Z80	Rodney Zaks	Sybex 1992
Mastering Machine Code on the ZX Spectrum	Toni Baker	Interface
Z80 Assembly Language Programming Manual	Zilog	Zilog (UK) (0628) 39200
Master your ZX Microdrive	Andrew Pennell	Sunshine ISBN 0 946408 19 X

Y:

Lenguaje Máquina avanzado para Zx Spectrum	David Webb	Ediciones Anaya Multimedia S.A.. 1995 Villafranca, 22. 28928 Madrid
Código Maquina del Zx Spectrum	Jesús Alonso Rodríguez	Publicado en fascículos en la revista MICROHOBBY
Programación del Z80	Rodnay Zaka	Ediciones Anaya Multimedia Villafranca, 22. 28928 Madrid

APENDICE A
HISOFT DEVPAC - VERSION PLUS 3

Devpac en el Plus 3 funciona de un modo parecido al de los otros modelos. En el disco se incorporan otros programas: GENS451 (versión 51-columnas del GENS4 normal), GENP3 (ensamblador de Plus 3), GENP351 (versión 51-columnas). MONS4 (versión del GENS4 normal). MONP3 (desensamblador especial de Plus 3). GENS451 y MONS4 se encuentran descritos en el manual, mientras que GENP3, GENP351 y MONP3 son ligeramente diferentes, estas diferencias son las explicadas a continuación. Debes leer el manual y las siguientes notas detenidamente.

QUÉ HACER PRIMERO

Antes de hacer nada con el programa, protege el disco contra escritura y realiza una copia de seguridad. Para hacerlo deberás seguir los siguientes pasos:

-Introduce en la unidad el disco original y teclea:

```
COPY "A: *.* " TO "M:"
```

Estando en modo Plus 3 BASIC. Cuando se hayan copiado los ficheros, introduce un disco nuevo virgen y teclea:

```
FORMAT "A:"
COPY “M *.* TO "A:*
ERASE “M:”
```

Hecho esto es importante que guardes la copia original en un lugar seguro y uses la copia de seguridad.

GEN Versión 5.1

Por defecto, cualquier operación de carga o grabación (como los comandos del editor G o P), se usará la unidad A: si deseas utilizar otra unidad deberás iniciar el nombre del fichero con la letra del disco seguida de dos puntos (:). Si quieres usar la cinta, tendrás que usar como unidad la T: y situarla antes del nombre del fichero. Ejemplos:

G , , TEST	Carga TEST desde la unidad A
G , , B: TEST	Carga TEST desde la unidad B
P , , T: SPRITE	Graba todo el texto que haya en memoria a la cinta con nombre SPRITE
A , , M: OBJECT	Ensambla el texto actual que haya en memoria y sitúa el código generado en un fichero llamado OBJECT en el disco RAM.

Se puede usar cualquier nombre válido para el Plus DOS. Es decir, 8 letras del nombre con tres de la extensión.
El disco RAM es totalmente soportado como unidad M:
El comando del editor X da un catalogo de la unidad de disco elegida (A: por defecto). Usa X, , s donde s es cualquier especificador válido de fichero o unidad; el valor por defecto as A:*.* .
Ejemplos:

```
X, , *.GEN
X, , M: SPRITE
X, , B:
```

Los mensajes de error del Plus3 DOS son indicados por su número; los números de error y sus mensajes los encontrarás en el manual del Plus 3 (páginas 235 y 236).
La inclusión de un texto dentro de otro mientras se ensambla sólo se encuentra disponible para cinta, no para disco.

MON Versión 5.1

La mayor diferencia que hay entre MONS4 y MONP3 es que el segundo se sitúa en la memoria del disco RAM, dejando toda la RAM normal de 48k (menos 160 bytes) libre. Se necesitan 160 bytes en algún lugar de la memoria que esté libre (y que sea menor de 49152) y al cargarlo se te pedirá esta dirección. En este momento deberás darle esa dirección (en decimal); estos 160 bytes serán usados para las rutinas de cambio de página, la pila, etc. Si quieres volver al MONP3 desde el BASIC deberás hacerlo mediante un RANDOMIZE USR xxxxx, donde xxxxx es la dirección que has introducido para los 160 bytes.
Inicialmente, MONP3 deberá ser cargado en la dirección 32768 y ejecutado en la misma; en este momento se te pide la dirección (en decimal) del espacio libre pera situar los 160 bytes; se reubicará en la memoria del disco RAM y reducirá el espacio del mismo en 16k.
La versión de Plus 3 de MON almacena tanto la pantalla como las variables del sistema cuando es llamado. Esto significa que puedes inspeccionar progresos que realicen algún tipo de efecto en el archivo visual (la pantalla), como imprimir caracteres o gráficos, y podrás ver en todo momento lo que sucede en ella, sin que en ningún momento se borre (para verla, usa el comando S) y que puedes ejecutar paso a paso los programas que usen las variables del sistema sin peligro. Este es un aspecto muy importante (y te recomendemos que lo examines detenidamente).
Los números pueden ser introducidos en decimal en cualquier momento si los precedemos con la eñe (obtenida pulsando ¡SYMBOL SHIFT¿-D) y tecleando acto seguido el número decimal.
Las instrucciones RST R28 son desensambladas seguidas de un DEFW XX dado que el Plus 3 usa esta instrucción con una dirección de línea interna.
Se puede desensamblar y almacenar el texto en el disco usando el comando T; simplemente entre la unidad (A, B, M) seguida de dos puntos (:) y seguidamente el nombre del fichero como respuesta a la pregunta 'Text?'
El comando H ahora muestra una pantalla de ayuda. Antiguamente el comando H servía para pasar los números de decimal a hexadecimal, pero esto es obviamente innecesario dado que los números decimales pueden ser introducidos directamente precediéndolos de ¡SYMBOL SHIFT¿-D.
El comando R te pide que introduzcas la RAM y ROM que quieres ver. Los números de ROM pueden ser 0, 1, 2 o 3 y de RAM puedes introducir cualquiera desde 0 hasta 7: las páginas de RAM se paginan a partir de la dirección RC000 (49152) y pueden ser inspeccionadas, modificadas, desensambladas, etc. Tanto la ROM como la RAM actualmente seleccionadas se muestran en el panel frontal, antes de los bytes de memoria y a la derecha.
El comando S te permite inspeccionar el estado de la pantalla antes de entrar en MONP3 (desde un breakpoint o directamente llamando a la dirección donde se almacenaron los 160 bytes). Pulsa SPACE para volver el Panel Frontal.

Esperamos que te guste usar el Devpac para el Plus 3 y aceptamos cualquier comentario que quieras hacer acerca del mismo y de la traducción.