Cómo modificar registros

Si introducimos un número hexadecimal después del símbolo " " seguido de un punto ".", el número dado se introduce en el registro de Z80 señalado por la flecha " ".

Al empezar, la flecha " " apunta al PC. Si quieres modificar cualquier otro registro, introduce "." solo, y la flecha se moverá de PC a AF. No es posible cambiar los registros SP o IR.

Ejemplo:

Supongamos que la flecha apunta al PC.

```
.       apunta a Iy
.       apunta a Ix
0.      Ix=0
.       apunta a HL
123.    HL=#123
etc.
[CTRL] J
```

Este comando nos introduce en GENA3, en caso de que ya haya sido usado previamente; si no, no tendrá efecto.

ENSAMBLADOR DESENSAMBLADOR Devpac

Como sabrás, las computadoras piensan en código binario; todas las memorias y las instrucciones que éstas obedecen no son más que una larga serie de **1** (unos) y **0** (ceros) (o bien, encendido o apagado). Un lenguaje de alto nivel, como BASIC o PASCAL, agrupa estas instrucciones para formar comandos en pseudo-inglés, como por ejemplo PRINT, END, etc. La ventaja de estos lenguajes es su facilidad de comprensión y uso, la desventaja que presentan es que una instrucción en alto nivel puede equivaler a cientos o incluso miles de ellas en código máquina. Por eso, los programas escritos en lenguajes de alto nivel son fáciles de escribir, pero lentos y demasiado largos. Así que si queremos escribir programas rápidos y compactos, deberemos hacerlo en código máquina, que es entendido directamente por la máquina. Sin embargo, como ya dijimos, estas instrucciones están compuestas por unos y ceros (o bits, dígitos Binarios), y nadie puede ni quiere recordar el grupo de Bits correspondientes a cada instrucción. No te preocupes, no es necesario; el lenguaje ensamblador se ocupa de otorgar «palabras" bastante fáciles de recordar (nemónicos). De este modo puedes escribir programas en lenguajes de bajo nivel mediante nemónicos, uno por cada instrucción. Después, tu programa en ensamblador debe ser transformado al lenguaje de máquina; de esto se encarga el ensamblador.

GENA3 es un esamblador para el microprocesador Z80. Contiene su propio editor que te permite crear un programa en assembler y ensamblarlo (convertirlo en Código Máquina) de forma muy fácil.

Tal vez, te gustaría ver un poco de Código Máquina y tratar de entender cómo funciona. Para esto, te gustará convertir el Código Máquina en assembler de nuevo para entenderlo mejor;

pues bien, para eso está el programa desensamblador. También querrás ver cómo afecta cada instrucción al procesador y ejecutarlas una por una, viendo los cambios en los registros y banderas (esto ayuda mucho a la hora de buscar errores en los programas).

Puedes incluso ejecutar un grupo de instrucciones y parar donde quieras colocando un "breakpoint". Todas estas posibilidades te las ofrece el MONA3, que es un instrumento indispensable cuando quieras buscar errores en tus programas.

Esto es lo que DEVPAC te ofrece. ¿Confundido? Esperamos que no, ¿impaciente? Muy bien, hagamos un programa en código máquina.

Pon tu cassette DEVPAC en el CPC 64, pulsa [PLAY] y escribe RUN" [ENTER] en el teclado.

Espera un rato y aparecerá el mensaje

Load Address?

en pantalla. Deja el cassette como está y escribe

1000 [ENTER]

El ensamblador será cargado en la memoria del CPC 64 a partir de la dirección 1000. Cuando termine de cargar el programa (después de aproximadamente 135 segundos), el ensamblador se ejecutará automáticamente, imprimiendo un mensaje en pantalla, un signo ">" seguido del cursor normal.

Pulsa [CAPS LOCK] y teclea:

I 10, 10 [ENTER]

y verás cómo aparece el número 10 a la izquierda de la pantalla seguido de un espacio. Ahora podemos introducir líneas de nemónicos seguidas cada una de [ENTER], y éstas serán añadidas al programa según el orden indicado por los números de línea. Así que, después del 10, escribe:

ENT $ [ENTER]
LD B, 26 [ENTER]
LD A, "A" [ENTER]
CICLO: CALL # BB5A [ENTER]
INC A [ENTER]
DJNZ CICLO [ENTER]
RET [ENTER]

y finalmente [CTRL] C. Es decir, mantén la tecla [CTRL] pulsada y aprieta la tecla C. Así vuelves al editor apareciendo el símbolo ">". Lista tu programa de la siguiente forma:

L [ENTER]

verás que saldrá tabulado.

Veamos ahora qué hace el programa.

10 ENT $
Esto simplemente dice al editor que, en caso de que quieras ejecutar el programa empezará a hacerlo a partir de aquí.

ENT no genera ninguna instrucción en Código Máquina, es lo que se llama una orden al ensamblador.

20 LD B,26
Otorga el valor 26 a uno de los registros del Z80: el registro B.

30 LD A, "A"
El registro A vale en este momento el código que el carácter "A" tenga asignado en ASCII.

40 CICLO: CALL # BB5A
La instrucción CALL se utiliza para llamar a una subrutina que esté colocada en cualquier otro sitio de la memoria de la máquina. La subrutina # BB5A (hexadecimal BB5A) es parte del sistema operativo e imprime en pantalla el carácter que el registro A guarde y vuelve a la instrucción siguiente al CALL. De este modo, ahora se imprime una "A".

50 INC A
Incrementa el valor del registro A en una unidad.

Resulta además que el ASCII tiene valores secuenciales para las letras, de tal modo que la "B" viene detrás de la "A". Por eso, el registro A toma el valor "B" al ejecutarse esta instrucción.

60 DJNZ CICLO

Poderosa instrucción esta. Dice: resta uno al valor guardado en el registro B y si no es cero, vuelve a la instrucción cuya etiqueta es CICLO (instrucción 40). Esta instrucción la podemos utilizar para ejecutar algo muchas veces, un poco como los ciclos FOR-NEXT en BASIC. Aquí, el registro B valía 26, así que DJNZ CICLO lo deja en 25, y salta a CICLO, que de nuevo imprime el valor de A (en este caso "B"), después A se ve incrementada de nuevo (ahora a "C") y volvemos a DJNZ, secuencia de instrucciones que se repetirá 26 veces hasta que B valga cero, momento en el cual se ejecutará:

70 RET

vuelve al editor. Generalmente RET señala el final de una subrutina y la vuelta al programa principal.

En este caso no estábamos en una subrutina, pero si ejecutamos el programa desde el editor, éste lo considera como tal y el control es devuelto al editor. Bueno, ¿qué tal si lo ejecutamos? Hasta el momento, lo que tenemos es solamente el texto fuente de nuestro programa: no lo hemos convertido en Código Máquina, no lo hemos ensamblado. Para ello utilizamos el comando del editor

A [ENTER]

y verás aparecer el mensaje

Table size: ("tamaño de la tabla")

Por el momento no necesitamos dar ningún valor aquí a la tabla, así que pulsa [ENTER]. Después aparece

Options:

Hay muchas opciones posibles, pero ahora simplemente pulsa [ENTER]. Verás aparecer un listado en pantalla: el listado ensamblado. La parte principal de éste te muestra el Código Máquina correspondiente a tu programa en assembler y dónde ha sido colocado en memoria, los 4 primeros caracteres corresponden a la dirección de memoria (en hexadecimal) de la instrucción en Código Máquina colocada a continuación.

De este modo, LD B,26 genera 061A en la máquina. Este 061A, es en realidad, otra representación de los bits. Cada carácter corresponde a 4 bits, es decir 0 es 0000, 6 es 0110, 1 es 0001 y A es 1010. No te preocupes de todo esto, lo importante es que tu programa ya está en memoria y listo para ser ejecutado. Hagámoslo: escribe

R [ENTER]

¿Qué ha sucedido? Si todo ha ido bien, habrás visto escribirse el alfabeto en pantalla. Ejecútalo de nuevo... ¿ves qué fácil?

Hagamos un pequeño cambio. Teclea:

20 LD B, 60 [ENTER]
A [ENTER]

Pulsa [ENTER] después de "table size?" y "Optións:" y ejecuta el nuevo programa (R [ENTER], recuerda).

Esta vez verás 60 caracteres.

Los ensambladores están hechos para facilitar la labor al programador en Código Máquina y GENA3 contiene muchas posibilidades que te permiten crear programas más fácilmente. Borremos el anterior y vamos a rehacerlo de una forma más legible.

D 10, 70 [ENTER]
Esta instrucción tan solo borra el código fuente (es decir, los nemónicos), no el programa en Código Máquina de la memoria (para comprobarlo, pulsa R [ENTER] de nuevo). Copia este programa:

```
I 10, 10 [ENTER]
LONGITUD: EQU26 [ENTER]
PRIMERO: EQU "A" [ENTER]
CONOUT: EQU # BB5A [ENTER]
[ENTER]
ENT $ [ENTER]
LD B, LONGITUD [ENTER]
LD A, PRIMERO [ENTER]
CICLO: CALL CONOUT [ENTER]
```

```
DJNZ CICLO [ENTER]
RET [ENTER]
[CTRL/C]
```

Lista el programa usando L [ENTER], ¿es más legible o no?

Hemos utilizado la orden EQU (las órdenes al ensamblador no generan Código Máquina) para asignar valores o pseudovariables. No son realmente variables pues no se pueden cambiar, pero dan al programa más sentido. Ensambla ahora el programa con A [ENTER], pulsando [ENTER] después de "Table size" y "Options" y ejecútalo (R [ENTER]). Como verás, el resultado sigue siendo el mismo que con el programa primitvo.

Bueno, hasta aquí hemos llegado. Esto es todo lo que podemos hacer sin escribir un libro del tema, cosa que, por lo demás, ya se ha hecho. Así que si eres nuevo en esto del Código Máquina deberás hacerte con un buen libro sobre Código Máquina del Z80 y leerlo. En cuanto a qué libro utilizar, no te puedo decir más que hay muchos y variados. Cualquiera, en principio, es bueno. Que tengas suerte, y sólo para que veas por qué la gente usa Código Máquina, borra tu programa:

D 10, 110 [ENTER]

y copia este otro:

```
I 10, 10 [ENTER]
ENT $ [ENTER]
COUNT: EQU 10000 [ENTER]
PLOT: EQU # BBEA [ENTER]
MODE: EQU # BC0E [ENTER]
GRPEN: EQU # BBDE [ENTER]
[ENTER]
LD A, 1 [ENTER]
CALL MODE [ENTER]
LD HL, 300 [ENTER]
LD (RANUM), HL [ENTER]
LD BC, COUNT [ENTER]
[ENTER]
LOOP: PUSH BC [ENTER]
BIT 4,C [ENTER]
JP Z, NOSET [ENTER]
LD A,C [ENTER]
CALL GRPEN [ENTER]
NOSET: CALL RANDOM [ENTER]
PUSH HL [ENTER]
CALL RANDOM [ENTER]
POP DE [ENTER]
CALL PLOT [ENTER]
POP BC [ENTER]
DEC BC [ENTER]
LD A,B [ENTER]
OR C [ENTER]
JP NZ, LOOP [ENTER]
RET [ENTER]
[ENTER]
RANDOM: LD DE, (RANUM) [ENTER]
LD H,E [ENTER]
LD L,—3 [ENTER]
LD A,D [ENTER]
SBC HL,DE [ENTER]
SBC A,0 [ENTER]
SBC HL,DE [ENTER]
SBC A,0 [ENTER]
LD E,A [ENTER]
LD D,0 [ENTER]
SBC HL,DE [ENTER]
JP NC, RAN1 [ENTER]
INC HL [ENTER]
RAN1: LD (RANUM), HL [ENTER]
LD A, H [ENTER]
AND %00000001 [ENTER]
LD H,A [ENTER]
RET [ENTER]
[ENTER]
```

```
RANUM: DEFS 2 [ENTER]
[CTRL/C]
```

Ensambla el programa mediante A [ENTER]. Quizá querrás usar la opción 4 después del mensaje "Options:" para no ver todo el listado, lo cual debe ser aburrido. Ejecútalo como ya sabes y mira lo que pasa. La velocidad de impresión de puntos en pantalla es de aproximadamente 1500 por segundo. Podría ser incluso más rápida si accediésemos directamente a la memoria de pantalla. Echemos un vistazo a un programa similar en BASIC. Desde el ensamblador pulsa B [ENTER], lo cual te devuelve al BASIC, y copia:

```
NEW [ENTER]
AUTO [ENTER]
CLS [ENTER]
J%=1 [ENTER]
FOR I%=1 TO 10000 [ENTER]
X%=RND*639 : Y%=RND*399 [ENTER]
IF (I%MOD 20)=0 THEN J%=INT(RND*3) +1 [ENTER]
PLOT X%,Y%,J% [ENTER]
NEXT I% [ENTER]
[ESC]
RUN [ENTER]
```

El resultado es mucho más lento que la rutina en Código Máquina, aproximadamente 20 veces. Sin embargo, ésta es una velocidad realmente buena para el BASIC, pues normalmente el Código Máquina suele ir hasta 1000 veces más deprisa que el equivalente en BASIC. De todos modos, la versión en Código Máquina es mucho más difícil de entender. Todo tiene sus ventajas y sus inconvenientes.

Para tu información, los tiempos de esta rutina de "ciclo nocturno" en BASIC, PASCAL y Código Máquina son:

150, 22 y 8 segundos, respectivamente

Esperamos que esta sección te haya abierto el apetito y estés deseando aprender Código Máquina cuanto antes y continuar con una lectura cuidadosa de este manual, probando las cosas según vayan apareciendo. Hay un extenso ejemplo de cómo usar el ensamblador en el apéndice 3 y otro del desensamblador en la sección 2.11 del manual de MONA3

SECCION 1. ENSAMBLADOR Y EDITOR

GENA3 es un poderoso ensamblador fácil de usar, muy próximo al ensamblador standard de Zilog. Pocos ensambladores son tan útiles como GENA3 y te invitamos a estudiar las siguientes secciones junto al ejemplo del apéndice 3. Si eres novato, trabaja con éste primero y/o con la sección 0.

GENA3 ocupa más o menos 7K una vez relocalizado. Contiene su propio editor de líneas que coloca el archivo de texto inmediatamente después del código de GENA3, mientras que la tabla de símbolos va detrás de dicho archivo. Por eso, cuando cargues GENA3 debes dejar suficiente espacio para el ensamblador la tabla de símbolos y longitud de texto mayores que vayas a necesitar.

Será conveniente que coloques GENA3 en la parte baja de la memoria.

Para cargar GENA3 en el Amstrad, teclea

RUN" [ENTER]

y pulsa [PLAY]. Primero se cargará un programa pequeño en BASIC que servirá para cargar el Código Máquina automáticamente.

Cuando aparezca el mensaje

Load Address? (¿Dirección de carga?)

deberás introducir un número entre 1000 y 30000 y pulsar [ENTER].

Dicho número es la dirección en que el ensamblador será cargado —un buen sitio es 1000—. Después de esto, aparecerá

Please wait... Loading GENA3...
(Espere, por favor,... cargando GENA3)

y el cassette se pondrá en marcha automáticamente. Después de unos 130 segundos el ensamblador estará ya listo para utilizarse.

Lo primero que verás será un corto mensaje y una pantalla de ayuda con los comandos en letra mayúscula.

NOTA: Si quieres tener en memoria GENA3 y MONA3 al tiempo, siempre carga el GENA3 en la parte baja (digamos en 1000) y MONA3 en la parte alta de la memoria (30000). En este caso, GENA3 debe ser cargado el último.

SECCION 2. DETALLES DE GENA3

2.0. Cómo funciona GENA3

GENA3 es un rápido ensamblador de dos pasos Z80, que se puede adaptar fácilmente a casi todos los sistemas de Z80. El ensamblador (o assembler) ensambla todos los nemónicos de Z80. Además incluye ensamblado condicional, comandos del ensamblador y una tabla de símbolos de árbol binario.

Cuando utilices el comando "A" para ensamblar (ver sec. 3) se te pedirá el tamaño de la tabla (table size?) en decimal. Esto representa el espacio que se le da a la tabla de símbolos durante el ensamblado. Si no pones nada (si pulsas [ENTER]), entoces GENA3 escogerá una longitud para la tabla que considere suficiente. Normalmente, este espacio será bastante. Cuando escojas la opción de incluir archivos, puede ser necesario que especifiques el tamaño de la tabla, pues el ensamblador no puede predecir la longitud del fichero que vas a añadir.

Después de "Table size?" se te pedirá la opción que deseas utilizar cuando aparezca el mensaje "Options:".

Veamos cada una de ellas:

Opción 1 Produce un listado de la tabla de símbolos después del segundo paso del ensamblado.

Opción 2 No produce código objeto. Es decir, no genera Código Máquina (sirve para descubrir errores en el programa sin ensamblar realmente cuando no tenemos espacio en memoria para el programa en Código Máquina).

Opción 4 No lista el programa ensamblado.

Opción 8 Lista el programa ensamblado en impresora.

Opción 16 Coloca el código objeto después de la tabla. El puntero se coloca en la dirección señalada por ORG, de manera que el Código Máquina puede ser puesto en una zona de memoria pero ejecutado en cualquier otro sitio.

Opción 32 No revisa la zona de memoria en la que el código objeto va a ser colocado. Es útil para dar mayor rapidez al ensamblado.

Ejemplo: la opción 36 produce un ensamblado rápido, no inspecciona la zona donde el código objeto está siendo enviado y no lista el programa. Esto es así puesto que 36=32+4, así que hemos escogido, en realidad, las opciones 32 y 4 simultáneamente. Del mismo modo, si queremos elegir las opciones 16 y 8 al tiempo deberemos introducir la opción 24 (16+8).

Notese que al escoger la opción 16, la orden ENT tendrá efecto. Puedes saber la colocación del código objeto en este caso usando el comando "X". El segundo número que aparezca corresponderá al final del texto. Sumando a esta cifra el tamaño de la tabla +2 obtendremos la dirección de comienzo del Código Máquina.

El ensamblado tiene lugar en dos pasos. Durante el primero, GENA3 busca errores y compila la tabla. En el segundo, genera el código objeto (si no has elegido la opción 2). Durante el primer paso nada aparece en pantalla a no ser que haya errores, en cuyo caso, la línea donde se encuentra el fallo será impresa con un número debajo (ver apéndice 1). El ensamblado se para en este momento. Si pulsas [CTRL] C vuelves al editor, cualquier otra tecla hace continuar el ensamblado.

Después del primer paso verás el mensaje:

Pass 1 errors : nn (Paso 1, errores : nn)

Donde nn es el nº de errores de teclado. Si hay alguno, el ensamblado será interrumpido y no se procederá al segundo paso. Si has utilizado alguna etiqueta inexistente en la zona destinada a ellas un mensaje de aviso (*WARNING* etiqueta absent) aparecerá por cada una que falte.

Es durante el segundo paso cuando el Código Máquina se genera (insistimos; siempre que no se haya escogido la opción 2) y cuando se imprime listado del programa ensamblado (excepto en la opción 4 o si se ha incluido en el programa el comando *L—). El listado es normalmente de la forma:

C000	210100	25	etique	LD	HL,1
1	6	15	21	28	33

El primer grupo numérico C000 corresponde a la posición del puntero antes de procesar esta línea, a no ser que en ella se encuentre uno de los pseudo-nemónicos ORG, EQU o ENT (ver sección 2.6), en cuyo caso, el primer número representa la posición de la instrucción en el campo de operandos. Este número está normalmente en hexadecimal pero aparecerá en decimal usando el comando *D+ (ver sección 2.8).

El siguiente número, a partir de la columna 6, puede tener hasta 8 caracteres de longitud (es decir, puede representar hasta 4 bytes) y corresponde al código objeto generado por la instrucción de nuestra línea.

Después viene el número de línea; un entero entre 1 y 32767 ambos inclusive. Las columnas 21-26 de la línea contienen los primeros 6 caracteres de la etiqueta que puede (o no) encontrarse en esta línea. Es por eso por lo que en el ejemplo anterior de arriba dice "etique" en lugar de "etiqueta", pues ésta última es demasiado larga (8 caracteres).

Al final se encuentran los nemónicos, operandos (si hay) y los comentarios. El ensamblado puede pararse pulsando cualquier tecla y subsecuentemente pulsar [CTRL] C para volver al editor o cualquier otra tecla para continuar. Los únicos errores que pueden aparecer durante el segundo paso son *ERROR*10 (ver apéndice 1) y "Bad Org!" (ocurre cuando el código objeto va a invadir el sitio de GENA3, el archivo de texto o la tabla —la detección de este error no ocurrirá en la opción 32—). El *ERROR*10 no es fatal y puedes continuar el ensamblado, mientras que "Bad Org!" sí que lo es, e inmediatamente devuelve el control al editor. ¡Mucho cuidado con la opción 32! Puede aparecer el mensaje **File closed** (fichero cerrado) si se ha usado el directivo ORG, cuando se esté grabando el código objeto mediante el comando *T, esto avisa de que el actual archivo de cassette ha sido cerrado y el comando *T inutilizado.

Al final del segundo paso se imprimirá el mensaje:

Pass 2 errors: nn (Errores del paso 2: nn)

y los avisos (**Warning**) de las etiquetas inexistentes.

También te mostrará cuánto espacio ha usado de la tabla de símbolos:

Table used: XXXX from YYYY (Tabla usada: XXXX de YYYY)

En este momento, si el comando ENT ha sido utilizado correctamente, verás aparecer el mensaje:

Executes: nnnn

que te indica la dirección donde se empezará a ejecutar el código objeto. Puedes hacerlo utilizando el comando "R" (siempre que el ensamblado haya sido correcto y el mensaje anterior se haya impreso en pantalla, sino, cuidado con usarlo).

Finalmente, si has escogido la opción 1, aparecerá un listado en orden alfabético de las etiquetas y los valores asociados.

El control vuelve al editor.

2.1. Formato de las instrucciones en ensamblador

Cada línea procesada por GENA3 debe tener este formato, siendo algunas partes de ella opcionales:

```
ETIQUETA:     NEMONICO     OPERANDOS     COMENTARIOS
PRINCI:       LD           HL,etique     ; carga hl con etique
```

Los espacios y tabulaciones (insertadas por el editor) generalmente se ignoran.

La línea se procesa de la siguiente forma:

Se comprueba si el primer caracter de la línea es ";", "," "*" ó [ENTER]. Si es éste ";", toda la línea se trata de un comentario, y por tanto ignorada. Si es éste "*", el ordenador sabe que el/los siguiente/s carácter/es es un comando del ensamblador (ver sección 2.8). Trata todos los caracteres después del comando como comentario.

Si éste es un [ENTER] simplemente ignora la línea.

Si el primer carácter de la línea se trata de cualquier otro aparte de los listados anteriormente, entonces el ordenador busca un espacio, una tabulación o dos puntos ":". En este último caso, todos los caracteres precedentes formarán parte de la etiqueta, de cualquier otro modo, el primer carácter que no sea un espacio o un tabulador se tomará como principio del nemónico.

Observa que si un nemónico está a continuación de una etiqueta, sólo los 6 primeros caracteres de la etiqueta aparecerán en la etiqueta. Sin embargo, si una etiqueta está sola en una línea, en el listado aparecerán todos sus caracteres aunque sólo los 6 primeros entrarán en la tabla de símbolos.

Después de procesar una etiqueta válida, o si no la hay, el ensamblador busca el siguiente carácter que no sea espacio o tabulador, y ve si éste se trata o bien el final de la línea o el comienzo de un nemónico de Z80 (ver apéndice 2) de hasta 4 caracteres de longitud y terminado por un espacio/tabulador o ENTER. Si el nemónico es válido y requiere uno o más operandos, el ordenador se salta los espacios/tabs y procesa dichos operandos.

Las etiquetas pueden encontrarse solas en una línea para hacer más legible el listado. Los comentarios pueden encontrarse en cualquier sitio detrás de la zona de operandos o, si el nemónico no necesita argumentos, después de la zona de nemónicos. Los comentarios se colocan automáticamente en la columna 40, tanto en los listados del ensamblador como en los del editor. Los nemónicos deben ir seguidos de un espacio, [TAB] o [ENTER].

2.2. Etiquetas

Una etiqueta es un conjunto de símbolos de hasta 16 bits de información. Puede usarse para especificar la dirección de una instrucción particular, área de datos o, como constante, dándole un valor con EQU (ver sección 2.6.).

Si una etiqueta se asocia a un valor de más de 8 bits y se utiliza en un contexto donde debería haber una constante de 8 bits, el ensamblador dará un mensaje de error (*ERROR*10) en el segundo paso. Un ejemplo de este tipo de error lo constituye el siguiente programa:

```
VALOR : EQU # 1234
        LD    A,VALOR
```

Las etiquetas pueden tener la longitud deseada (ver más abajo), aunque sólo los primeros 6 se toman en consideración; estos 6 caracteres no deben ser repetidos pues una etiqueta no se puede redefinir (*ERROR*4). Una etiqueta debe terminar con ":" y no debe constituir una palabra reservada, aunque una de ellas puede estar incluida como parte de una etiqueta.

Los caracteres utilizables en una etiqueta son: 0-9, $, A-Z (esto significa que todas son letras, tanto mayúsculas como minúsculas, se admiten), (, \ ,), ?, 'y—, aunque deben comenzar con una letra. Ejemplos de etiquetas válidas:

```
CICLO:
ciclo:
etiqueta— larga:
L [1]
L [2]
```

```
a:
LDIR:       (LDIR no es una palabra reservada)
dos:5:
etc.
```

2.3. Puntero

El ensamblador tiene un puntero de tal manera que un símbolo en la zona de etiquetas puede asociarse con una dirección e incorporarse a la tabla de símbolos. Este puntero puede colocarse a voluntad mediante ORG (ver sección 2.6). El símbolo "$" puede utilizarse para referirse a la posición actual del puntero, por ejemplo: LD HL, $+5 cargaría el par HL con la posición del puntero +5.

2.4. Tabla de símbolos

Cuando una etiqueta aparece por primera vez, entra a formar parte de una tabla junto con dos punteros que indican qué relación alfabética tiene con las demás etiquetas de la tabla. Si la primera aparición de la etiqueta es en el campo de etiquetas, su valor se introduce en la tabla de símbolos. De otro modo, el valor se introduce en la tabla cuando la etiqueta aparece por primera vez en el campo de etiquetas.

Este tipo de tabla se denomina "tabla de símbolos en árbol binario" y su estructura permite la introducción de nuevos símbolos y su recuperación en muy poco tiempo (esencial en programas largos). La longitud de la entrada en la tabla varía entre 8 y 13 bytes dependiendo del tamaño de la etiqueta.

Si, durante el primer paso, una etiqueta ha sido definida más de una vez, aparecerá el *ERROR*4 pues el ensamblador no sabe qué valor de todos asociar (recuérdese que basta con repetir los 6 primeros carácteres para incurrir en el error 4).

Si no se le asocia un valor a un símbolo, el mensaje de aviso "*WARNING* símbolo absent" se imprimirá en pantalla.

La ausencia de dicha definición del símbolo no interrumpirá el ensamblado.

Al final del ensamblado se te informará de cuánta memoria fue utilizada por la tabla durante el ensamblado. Puedes dar la cantidad de memoria que quieres a la tabla respondiendo al mensaje "Table size?" con el número de bytes que creas suficiente al principio del ensamblado (ver sección 2.0.).

2.5. Expresiones

Una expresión es un operando consistente en: o bien un TERMINO sencillo o bien una combinación de TERMinos separados por un OPERADOR. Las definiciones de TERMino y OPERADOR son:

TERMino: Constante decimal, p. ej.: 1029
constante hexadecimal, p. ej.: #405
constante binaria, p. ej.: % 10001010
carácter, p.ej.: "a"
etiqueta, p. ej.: LA23

También $ puede utilizarse para referirse al valor del puntero del programa.

OPERADOR: "+" adición
"—" sustracción
"&" AND lógico
"@" OR lógico
"!" XOR Lógico
"*" producto de enteros
"?" función MOD (a ? b=a—(a/b)*b)

Notas: "#" se usa para especificar que el número que sigue se encuentra en base 16 (hexadecimal), % para números en base 2 (binarios) y " " para caracteres. Cuando lee un número (decimal, hexadecimal o binario) GENA3 toma los 16 bits menos significativos de éste, por ejemplo: 70016 se convierte en 4480 y #5A2C4 en A2C4.

No existe prioridad de operadores; las expresiones se evalúan de izquierda a derecha. Los operadores "*", "/" y ?, se han añadido por mera conveniencia y no como parte de un manipulador de expresiones, lo que aumentaría el tamaño de GENA3.

Si una expresión está entre paréntesis, se toma como si se tratase de una dirección de memoria. Por ejemplo: en la instrucción LD HL, (loc +5), el par HL tomaría el valor (de 16 bits) contenido en la posición loc +5.

Ciertas instrucciones de Z80 (JR y DJNZ) funcionan sólo con operandos de 8 bits. Este tipo de instrucciones se llaman de "direccionamiento o salto relativo". Cuando GENA3 encuentra una de ellas, substrae el valor del puntero en la siguiente instrucción del valor dado en la zona de operandos en la instrucción actual, para obtener la dirección relativa de la instrucción en curso. El intervalo de valores permitido para una dirección relativa es: valor del puntero —128 ó +127, es decir (puntero —128, puntero +127).

Si en cambio, quieres referirte a un "offset" relativo desde la posición del puntero en la instrucción actual, debes utilizar el signo "**$**" (palabra reservada) seguido del desplazamiento deseado. Puesto que ahora éste se refiere al valor del puntero en la instrucción en que nos encontramos, el intervalo es (—126, 129) ambos inclusive.

Ejemplos de expresiones válidas:

```
# S000— label
% 1001101 ! % 1011      da 1000110
#3456 ? #1000           da #456
4+5*3—8                 da 19
$ —label +8
2345/7 —1               da 334
"A" +128
"y" — ";" +7
(5*label—#1000 & % 1111)
17 @ % 1000             da 25
```

Pueden insertarse espacios entre TERMinos y OPERADORES y viceversa, pero no entre medias de TERMinos.

Si una multiplicación es mayor en valor absoluto a 32767, el mensaje "*ERROR*15" aparecerá, mientras que si intentamos dividir por 0, en este caso obtendremos un "*ERROR*14". Cualquier otro desbordamiento (overflow) se ignora. Todas las operaciones utilizan el complemento a dos, por el cual los números mayores de 32767 se consideran negativos. Por ejemplo: 60000=—5536 (60000—65536).

2.6. Directivos del ensamblador

Ciertas instrucciones pseudonemónicas son reconocidas por GENA3. Estos directivos no tienen efecto en el procesador Z80 al ejecutar el programa, es decir, no se traducen en código objeto, simplemente dirigen al ensamblador mientras ensambla. Las acciones que estas órdenes mandan efectuar al ensamblador cambian de alguna manera el código objeto generado por éste.

Los pseudonemónicos se ensamblan exactamente igual que las instrucciones ejecutables; pueden ser precedidas por una etiqueta (necesaria para EQU) y seguidas por un comentario. Los directivos existentes son:

ORG expresión
Coloca el puntero en el valor dado por "expresión". Si no eliges la opción 2 ni la 16 y ORG provocara que el programa invadiera la zona asignada a GENA3, al archivo de texto o a la tabla de símbolos, el mensaje "Bad ORG!" aparecerá en pantalla y el ensamblador parará. Ver la sección 2.0 para más detalles de cómo las opciones 2 y 16 afectan al uso de ORG.

EQU expresión
Debe ir precedido por una etiqueta. Otorga a la etiqueta el valor de "expresión". Esta expresión no puede contener ningún símbolo que no haya sido definido previamente (o si no, obtendremos el *ERROR*13).

DEFB expresión, expresión...
Cada expresión no puede tener más de 8 bits; el byte correspondiente al valor del puntero recibe el valor de la expresión, y el puntero aumenta su valor en 1. Se repite este proceso con cada expresión.

DEFW expresión, expresión:
Coloca el valor "expresión" (de 2 bytes) en la dirección del puntero y avanza al puntero en 2. Se coloca primero el byte menos significativo seguido del más significativo. El proceso se repite en cada expresión.

DEFS expresión.
Aumenta el valor del puntero en el número de bytes que "expresión" indique. Esto equivale a reservar un bloque de memoria de longitud igual al valor numérico de la expresión. A todas las posiciones de memoria de este bloque se les dará el valor cero.

DEFM "S"
Coloca el valor ASCII de la cadena "S" en n bytes, donde n es la longitud de dicha cadena y puede encontrarse, en teoría, en el intervalo entre 1 y 255 inclusive, aunque en la práctica, lo que limita la longitud de la cadena es la longitud de línea que el editor te permita introducir. El primer carácter de la zona de operandos (en nuestro caso las comillas) se toma como el delimitador de la cadena, y ésta se define como los caracteres existentes entre dos delimitadores; el final de línea también actúa como delimitador.

ENT expresión
El valor de la expresión corresponde a la dirección desde donde el código objeto será ejecutador cuando utilicemos el comando "R" (ver sección 3). No puede faltar esta dirección de ejecución en el programa.

2.7. Pseudonemónicos condicionales

Este tipo de pseudonemónicos permiten al programador quitar y poner determinadas zonas de código fuente para ser o no ensambladas.

IF expresión
Primero se evalúa la expresión; si ésta es cero, no se ensamblaran las líneas siguientes hasta que aparezca "ELSE" o "END" en el programa. Si la expresión no vale cero, el ensamblado continúa normalmente.

ELSE
Cuando el ensamblador encuentra un "ELSE" hace lo contrario de lo que venía haciendo hasta ese momento. Si no estaba ensamblando el programa, a partir de aquí empezará a hacerlo y viceversa.

END
Activa el ensamblado.

2.8. Comandos del ensamblador

Estos comandos, al igual que las órdenes al ensamblador, no afectan al procesador al ejecutar el programa pues no se convierten en código objeto. Sin embargo, aunque no producen ningún cambio en el código objeto generado por el ensamblador, modifican el formato del listado.

Un comando es una línea de texto fuente que comienza con un asterisco "*". La letra que sigue determina el tipo de comando, debe ser una letra mayúscula. El resto de la línea puede ser cualquier cosa, excepto en los comandos "L", "T" y "D" que deben ir seguidos de un signo "+" o un signo "—".

Comandos:
*E
Imprime tres líneas en blanco en la pantalla. Es útil para separar partes. En caso de utilizar la impresora, ésta iniciará una nueva página al encontrar este comando.

*Hs
Imprime la cadena s como encabezamiento después de cada *E. *Hs incluye *E, así que este último comando no será necesario en caso de incluir *Hs.

*S
Detiene el listado del programa. Este continuará pulsando cualquier tecla. Es útil para leer direcciones de memoria en medio del listado.

*L—
A partir de esta línea no se lista el programa.

*D+
El valor del puntero se encontrará en base 10 al principio de cada línea, en lugar de su formato normal en hexadecimal.

*D—
Efecto contrario al comando anterior. El puntero vuelve a aparecer en hexadecimal.

*F nombre del archivo
Este poderoso comando permite ensamblar texto del cassette. Dicho texto se lee del cassette y se introduce en un buffer, un bloque cada vez y después se ensambla desde el buffer. Esto permite crear grandes cantidades de código objeto puesto que éste no ocupa memoria real.

El nombre del archivo (hasta 8 caracteres) de texto que quieres incluir en ese momento del ensamblador debe ser especificado después de "*F" y debe ir precedido por un espacio. Si no dices el nombre del archivo, se ensamblará el primero que encuentre el ordenador. Cualquier archivo que quieras incluir de esta manera deberá ser grabado en un cassette usando el comando "P" del editor. Cuando el ensamblador detecte un comando "F" se imprimirá el mensaje

"Press PLAY then any key"

antes del primero y segundo paso, pues el archivo deberá ser revisado en cada paso. El cassette buscará el archivo con el nombre especificado, o en su defecto, el primero de todos. Cuando encuentre uno que no coincida con el que queremos, imprimirá "Found nombre" y si se trata del que queremos incluir: "Loading nombre". Ver apéndice 3 para más información y ejemplos de este comando.

*T+ nombre del fichero
Este comando vuelca código objeto al cassette (con el nombre dado) durante el ensamblado. Este volcado puede ser interrumpido por el comando *T—, por ORG o por el final del ensamblado. El código así grabado puede ser recargado desde BASIC o con MONA3.

*T—
Cierra el archivo de cassette de código objeto. Si el ensamblado ha sido interrumpido por un pseudonemónico condicional, este comando deja sin efecto cualquier otro.

SECCION 3. EL EDITOR INTEGRAL

3.1. Introducción

El editor de GENA3 es un sencillo instrumento pero eficiente a la hora de editar programas rápidamente.

Para reducir el tamaño del archivo de texto, el editor comprime algunos espacios de la siguiente manera. Cuando tú introduces una línea, ésta se introduce carácter a carácter en un buffer interno. Cuando pulsas [ENTER] se transfiere del buffer al archivo y es durante esta transferencia cuando se comprimen los espacios. Se busca el primer carácter de la línea; si es un espacio, se introduce un tabulador en el archivo de texto y los demás espacios se saltan. Si no es un espacio, todos los caracteres hasta el primer espacio se introducen en el archivo, y después del primer espacio ocurre lo mismo que si el siguiente carácter fuese el primero de la línea. Esto se repite otra vez, introduciendo tabuladores al principio de la línea, entre etiqueta y nemónico y entre nemónico y operandos.

Si aparece [ENTER], la transferencia finaliza y el control vuelve al editor.

Los espacios no se comprimen en los comentarios, ni deberían formar parte de una etiqueta, nemónico u operando.

El editor genera el mensaje:

Hisoft GENA3 Assembler
Copyright Hisoft 1984
All rights reserved

y el símbolo " > ".

En respuesta a este símbolo (que indica que el ordenador se encuentra preparado para recibir órdenes), puedes introducir una línea de comandos con el siguiente formato:

C N1, N2, S1, S2 [ENTER]

C es el comando a ejecutar (ver sección 3.2)
N1 número entre 1 y 32767 ambos inclusive
N2 número entre 1 y 32767 ambos inclusive
S1 cadena de caracteres de hasta 20 caracteres de longitud
S2 cadena de caracteres de hasta 20 caracteres de longitud

Las comas se usan para separar los diversos argumentos (aunque esto puede cambiarse —ver comando "S"—) y los espacios son ignorados, excepto en las cadenas. Ninguno de los anteriores argumentos son obligatorios, aunque, por ejemplo, el comando "D" no funcionará si no se especifican N1 y N2. El editor recuerda los últimos números y cadenas que introdujiste y los utiliza si no especificas alguno de ellos. El valor inicial de N1 y N2 es 10 y de S1 y S2 la cadena vacía. Si introduces una línea no válida como: F—1, 100, HOLA, la línea será ignorada y se imprimirá el mensaje "Pardon?" y deberás introducirla correctamente: F1, 100, HOLA. Este mensaje se imprimirá tambien si la longitud de S2 es mayor que 20. Si S1 tiene más de 20 caracteres, el sobrante será ignorado. Los comandos pueden introducirse en mayúsculas o minúsculas.

[CTRL] X borra hasta el principio de la línea.

[TAB] desplaza el cursor hasta la posición de la siguiente tabulación las siguientes subsecciones detallan los comandos existentes en el editor. Nótese que cuando un argumento está encerrado entre los símbolos "<>", es necesario para que el comando tenga efecto.

3.2.1. Comandos del editor

El texto puede introducirse, o bien escribiendo el número de línea, un espacio y después el texto o usando el comando "I". Si introduces un número de línea seguido de [ENTER], ésta será borrrada del programa si existía previamente. [CTRL] C vuelve al editor y ya conocemos las funciones de [CTRL] X y [TAB]. La tecla [DEL] borra un carácter (no más allá del principio de la línea). Si el buffer en el que se introduce la línea se llena, el ordenador te impedirá seguir escribiendo, debiendo utilizar [DEL] ó [CTRL] X para hacer hueco. Si, durante la inserción de texto, aparece el mensaje "Bad Memory!", esto querrá decir que nos estamos acercando al final de la RAM, no se puede introducir más texto y el programa en cuestión (o al menos parte) deberá ser grabado para posterior recuperación.

Comando: I n,m
Modo de inserción automático. Los números de línea aparecen automáticamente empezando por el número n y con incrementos de m.

Para salir de este modo pulsa [CTRL] C.

Si introduces una línea con un número que existía previamente, la anterior línea remunerada una unidad más y la nueva línea insertada antes, después de pulsar [ENTER]. Si el incremento automático del número de línea produce una línea con un número mayor de 32767, el ordenador saldrá de ese modo automáticamente.

3.2.2.

Comando: L n,m
Lista el programa desde la línea n hasta la m. Si no se especifican, n tomará el valor de 1, y m el valor 32767, con lo cual se listará todo el programa. La tabulación es automática y el número de líneas que caben en la pantalla es 24. Cuando haya listado 24, el editor hará una pausa. Pulsa [CTRL] C para volver al editor o cualquier otra tecla para continuar.

3.2.3. Edición de textos

Una vez creado un programa, inevitablemente habrá que editar algunas líneas. Hay varios comandos que permiten arreglar, borrar y renumerar líneas. Puedes utilizar el editor propio del CPC 464, y mover el cursor libremente por la pantalla con las

teclas de cursor y copiar texto de la pantalla al buffer del editor mediante la tecla [COPY] hasta llegar al punto donde quieres hacer la corrección. Después de hacerla, copia el resto de la línea, finalizando con [ENTER].

3.2.3. Edición de textos

Una vez creado un texto, inevitablemente será necesario corregir algunas líneas. Hay muchos comandos disponibles para posibilitar la corrección, borrado, movimiento y renumeración de las líneas introducidas. Estos comandos los trataremos más adelante. Por otro lado también existen formas sencillas de edición de pantalla que funcionan de la forma siguiente:

Siempre que estés en modo comando en el editor (es decir, con un '>' en la parte izquierda de la pantalla) puedes seleccionar entre cursor de lectura y cursor de escritura pulsando la tecla [SHIFT] a la vez que una de las teclas de cursor. El cursor de escritura permanecerá en la posición original mientras que el cursor de lectura puede moverse por toda la pantalla usando [SHIFT] y las teclas de cursor. Una vez colocado en la posición deseada, suelta [SHIFT] y la tecla de cursor. Puedes ahora teclear directamente y los caracteres aparecerán en la posición del cursor de escritura o pulsar [COPY], en cuyo caso se transferirán los caracteres desde la posición del cursor de lectura a la posición del cursor de escritura incrementándose los dos punteros.

Para salir de este modo pulsa simplemente [ENTER], desapareciendo el cursor de lectura y la línea que contiene el cursor de escritura será normalmente por el editor.

Además de estas posibilidades de edición de pantalla, existen varios comandos de edición de línea:

Comando: D <n, m>
Borra todas las líneas comprendidas entre *n* y *m*. Si m < n o si falta uno de los dos números, el comando no funcionará. Para borrar una línea haz n=m o bien teclea el número de línea y pulsa [ENTER].

Comando N > M n,m,d
Traslada el bloque de texto comprendido entre las líneas n,m o una posición anterior a la línea d, borrando el texto inicial.

El bloque trasladado se renumerará empezando con un número de línea superior en una unidad al número de línea anterior al d.

Comando N <n,m>
Renumera el programa, dando el número n a la primera línea n+m a la segunda. n+m+m a la tercera, y así sucesivamente. Si al renumerar, sobrepasamos con algún número de línea el valor 32767, se conservará la numeración antigua.

Comando F n,m,f,s
Busca la cadena f en todas las líneas desde n hasta m. Si f aparece en alguna de ellas, ésta será impresa y el ordenador se introduce en el modo Editor automáticamente (ver siguiente comando). Puedes, entonces, usar los comandos del Editor para buscar otra vez la cadena f en el intervalo de instrucciones dado, sustituirla por la cadena S. (n,m,f,s pueden haber sido dadas anteriormente, y por tanto no ser necesario escribirlas de nuevo para hacer la misma búsqueda posteriormente. Bastará para ello simplemente con introducir "F". Ver sección 3.3 para aclaraciones.

Comando E n
Edita la línea de número n. Si no existe, no ocurrirá nada, pero si existe, la línea será copiada en un buffer e impresa en pantalla.

El número de línea se imprimirá inmediatamente debajo. Todos los cambios a partir de aquí tendrán lugar en el buffer, no en el programa mismo, siendo posible recuperar la línea original en cualquier momento. En este modo, un puntero se coloca al principio en el primer carácter de la línea y varios subcomandos permiten arreglar los errores. Estos son:

"—" (espacio) incrementa el puntero en uno. No seguirá después del final de la línea.

[DEL] coloca el puntero en el carácter anterior. No seguirá retrocediendo más allá del principio de la línea.

[ENTER] finaliza la edición de la línea guardando los cambios hechos.

Q — Sale del modo de edición sin tener en cuenta los cambios hechos.

R — Ignora los cambios hechos y recarga el buffer con la línea como estaba originalmente.

L — Lista el resto de la línea desde la posición del puntero. Permaneces en el modo de edición con el puntero al principio de la línea.

K — Borra el carácter en el que se encuentra el puntero.

Z — Borra todos los caracteres desde la posición del puntero hasta el final de la línea.

F — Explicado anteriormente.

S — Sustituye la cadena f encontrada por "F" por la cadena s y continúa la búsqueda de f más adelante.

I — Inserta caracteres en la posición del cursor. Continuarás en este submodo hasta que pulses [ENTER], tras lo cual volverás al modo editor con el puntero colocado en el último carácter que has insertado. [DEL] borra el carácter a la izquierda del puntero mientras estemos en el submodo I (en este submodo el cursor será un asterisco "*").

X — Avanza el puntero hasta el final de la línea e introduce el modo I.

C — Cambia el carácter en el que se encuentre el puntero por el que tú escribas e incrementa el puntero en uno. Permanecerás en este modo hasta pulsar [ENTER], tras lo cual volverás al editor colocado en el último carácter que cambiaste. [DEL] hace retroceder el puntero una posición y [TAB] no tiene efecto El cursor en este submodo es el signo "+".

3.2.4. Comandos del cassette

Los programas pueden ser grabados y cargados usando los comandos "P", "V" y "G" mientras que el código objeto será grabado el comando "O".

Comando: Pn,m,s
El intervalo de líneas entre n y m se graba bajo el nombre s. Recuerda que estos argumentos pueden tener valores asignados previamente. Asegúrate de que el cassette está en modo RECORD antes de introducir este comando.

Comando: Q,,s
Es igual que el comando P pero el texto se graba en ASCII sin números de línea y con CRLF (retorno de carro/fin de línea) al final de cada línea.

Comando: G,,s
El cassette busca un archivo con el nombre s, si lo encuentra, será cargado detrás del programa que esté en memoria. Si s es la cadena vacía, se cargará el primer programa que aparezca.

Después de introducir el comando "G", pulsa PLAY en el cassette. Si encuentra el programa deseado el mensaje "Loading nombre" aparecerá, si no, el mensaje será "found nombre" y la búsqueda continúa. Si hay algún otro programa en memoria, el programa cargado con "G" se añadirá al final del anterior y ambos serán renumerados desde la línea 1, con intervalo de 1 entre líneas y línea.

Comando: V,,s
Verifica el programa s del cassette con el existente en la memoria. Si la verificación es correcta, el mensaje que aparece es "Verified" sino obtendremos el mensaje "Failed!".

Comando: O,,s
Graba el código objeto generado en el último ensamblado bajo el nombre s. Para cargar este código objeto utiliza MONA3 o BASIC.

Comando: Tn

Cambia la velocidad de grabación en cassette. "T" seguido de [ENTER] selecciona la velocidad lenta, mientras que "T" seguido de un número mayor que cero y [ENTER] selecciona el modo rápido.

3.2.5. Ensamblado y ejecución desde el editor

Comando A
Ensambla el programa fuente desde la primera línea. Ver la sección 2 para más detalles.

Comando: R
Si el código fuente ha sido ensamblado sin errores y has dado la dirección de ejecución con ENT, el comando "R" ejecutará el código objeto. Este programa puede incluir una instrucción RET (#C9) para volver al editor puesto que el stack se encuentra en la misma situación al finalizar la ejecución que al principio. ENT no tendrá ningún efecto si has escogido la opción 16 al ensamblar.

3.2.6. Otros comandos

Comando: H
Imprime una pantalla de ayuda donde aparecen los comandos de GENA3 en mayúsculas.

Comando: B
Devuelve el control al sistema operativo. Para reentrar en el ensamblador puedes hacer una llamada a la dirección original de carga +2, en cuyo caso borrarás el programa fuente de la memoria o bien llamar a la dirección original +4, conservándolos.

Comando: S,,d
Te permite cambiar el delimitador que se toma para separar argumentos en una línea. Al entrar en el editor, se toma la coma "," como delimitador, pero tras utilizar este comando, el papel de delimitador pasará al primer carácter de la cadena d, y hasta que no se defina otra vez, dicho carácter será el delimitador. No se puede utilizar el espacio para este fin.

Comando: C
Imprime el valor del delimitador, N1, N2, S1 y S2. Es útil para comprobarlos en caso de que pienses omitir alguno de ellos en algún comando.

Comando: Zn,m
Lista en impresora el trozo de programa entre las líneas n y m. Si ambos valores se omiten, todo el programa fuente será listado. En caso de que la impresora no esté conectada, se imprimirá el mensaje "No printer!" y no ocurrirá nada. Puedes parar el listado pulsando cualquier tecla. Si pulsas [CTRL] C volverás al editor, mientras que pulsando cualquier otra cosa, el listado continuará.

Comando: Yn
Pone el número de líneas por página al valor n. Esto te permite ajustar la salida por impresora a las diferentes longitudes de papel de impresora.

Comando: X
Imprime la dirección de comienzo y la final del código fuente en decimal. Es útil en caso de que quieras grabar el texto desde BASIC o para ver cuánta memoria tienes libre tras dicho programa "X" se usa también en conjunción con el desensamblador de MONA3 para desensamblar un programa para uso de GENA3. Para unir un texto generado por MONA3 al ensamblador, debemos mover primero dicho texto (o programa en código fuente) al TEXTSTART de GENA3 o generar el texto directamente en esa dirección. Ahora, la primera dirección impresa con el comando "X" corresponde al principio del texto de GENA3. Esta es la dirección en la que el código fuente desensamblado por MONA3 debe empezar, así que habrás de ingeniártelas para desensamblar el código en esa posición y anota la dirección del final del texto que te da el desensamblador, valor que deberás dar al TEXTEND de GENA3. Este TEXTEND está almacenado en la dirección "Principio de GENA3"+7 así que si cargamos GENA3 en 1000, TEXTEND estaría localizado en 1007. Toma el valor del final del texto dado en hexadecimal

por MONA3 y "poke" el byte menos significativo en 1007 y el más significativo en 1008 y llama a GENA3 via "dirección original"+4. El programa desensamblado puede ser editado y ensamblado por GENA3 ahora.

Comando: U
Muestra el número de líneas de la última línea del fichero de texto, es útil para encontrar rápidamente donde termine el fichero de texto y añadir más texto.

Comando: W
Cambia el display de 40 a 80 caracteres o al revés.

Comando: J
Te introduce en el programa para descubrir errores MONA3 si éste ha sido utilizado previamente, si no, no tiene ningún efecto.

3.3. Ejemplo de uso del Editor

Supongamos que has copiado el siguiente programa (utilizando I 10, 10):

```
10*h NUMEROS ALEATORIOS DE 16 BITS
20
30  ; INPUT: HL contiene números aleatorios anteriores
      o semillas
40  ; OUTPUT: HL contiene el número aleatorio nuevo
50
60    Random: PUSH AF ; salva registros
70      PUS BC
80      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 ; número aleatorio antiguo
160     ADD HL, DE
170     LD DE, 41
180     ADD HL, DE
190     POP C ; recuperar registros
200     POP AF
210     REY
```

Este programa contiene los siguientes errores:

10: una "h" minúscula en lugar del comando "*H".
40: "randon" en lugar de "random"
150: "PIP" en lugar de "POP"
160: necesita un comentario. No es un error, simplemente es aconsejable.
210: "REY" en vez de "RET"

También 2 líneas ADD HL, HL deberían añadirse entre las líneas 140 y 150, y todas las referencias al par DE en 160 y 180 deberían ser al par BC.

Para arreglar todo esto, procedemos como sigue:

E10 [ENTER] espacio en blanco C (modo de cambio) H [ENTER] [ENTER]
F40, 40, randon, random (ENTER)s (comando)
I 142,2 [ENTER]
142 ADD HL, HL ; *128
144 ADD HL, HL ; *256
[CTRL] C
F 150, 150, PIP, POP [ENTER]s (comando)
E 210 [ENTER], 10 espacios C T [ENTER] [ENTER]
N 10, 10 [ENTER] para renumerar el programa.

Te recomendamos que trabajes con el ejemplo anterior directamente en el editor de GENA3

APENDICE 1 *Errores y sus significados*

*ERROR*1	Error en el contexto de la línea
*ERROR*2	Nemónico no reconocido
*ERROR*3	Sentencia mal construida
*ERROR*4	Simbolo definido previamente

*ERROR*5 La línea contiene un carácter ilegal, es decir inválido teniendo en cuenta el contexto de la línea.
*ERROR*6 Operando ilegal
*ERROR*7 Palabra reservada utilizada como símbolo
*ERROR*8 Registros que no casan
*ERROR*9 Demasiados registros en la línea
*ERROR*10 Expresión de más de 8 bits donde debería haber una de 8
*ERROR*11 Las instrucciones JP(IX±n) y JP(IY±n) son ilegales
*ERROR*12 Error en la formación de una orden al ensamblador
*ERROR*13 Referencia ilegal, por ejemplo: en EQU se ha utilizado un símbolo todavía no definido.
*ERROR*14 División por cero
*ERROR*15 Desbordamiento en la multiplicación
Bad Org! El origen ha sido colado en una posición que provocaría la invasión de GENA3, el texto o la tabla de símbolos.

El control vuelve al editor.

Out of table space! Aparece durante el primer paso si la memoria otorgada a la tabla es insuficiente. El control vuelve al editor.

Bad Memory! Aparece si no hay memoria para introducir más programa pues estamos llegando al final del RAM.

Tendrás que grabar parte, o la totalidad, del programa en cassette.

APENDICE 2 *Palabras reservadas, nemónicos, etc.*

La lista siguiente corresponde a las palabras reservadas de GENA3. Estos símbolos no pueden utilizarse como etiquetas como tal, aunque pueden formar parte de ellas. Todas las palabras reservadas están escritas en mayúscula.

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

Lista de nemónicos válidos del Z80, órdenes al ensamblador y comandos (también 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								
★D	★E	★H	★L	★S	★C	★F	★T	

APENDICE 3 *Un ejemplo paso a paso*

Reproducimos una sesión típica para un usuario de GENA3. Si eres novato, leelo cuidadosamente. Cada línea **debe** ir seguida de [ENTER], no lo olvides.

Objetivos:

Escribir una rutina de multiplicación de enteros rápida, grabar el programa fuente con el comando "P" para ser incluida en futuros programas.

Plan de trabajo:

1. Escribir la rutina como si fuera una subrutina y grabarla en cassette como ya hemos indicado para recuperarla posteriormente y corregir errores si los hubiere.
2. Eliminar los errores.
3. Grabar la rutina sin errores mediante el comando "P" para ser incluida en otros programas.

Paso 1 — Escribir la rutina.

Utilizamos el comando "I" para introducir el texto, utilizando TAB, si así lo deseamos, para obtener una lista tabulada, aunque esto se hará automáticamente. No importa que las direcciones

de memoria en el ejemplo no sean iguales que las generadas por tu máquina, simplemente sirven como ilustración.

```
>I10,10 [ENTER]
 10 ; Subrutina rápida
 20 ; para multiplicar enteros
 30 ; multiplica HL por DE y
 40 ; devuelve el resultado en HL
 50 ; C=1 si hay desbordamiento
 60
 70 ORG #7F00
 80
 90 Mult: OR A
100 SBC HL,DE;HL > DE?
110 ADD HL,DE
120 JR NC,Mu1;yes
130 EX DE,HL
140 Mu1: OR D
150 SCF
160 RET NZ; desbordamiento
170 OR E
180 LD E,D
190 JR NZ,MU4 ; no
220 EX DE,HL;0
210 RET
220
230; Rutina principal
240
250 Mu2: EX DE, HL
260 ADD HL,DE
270 EX DE,HL
280 Mu3: ADD HL,HL
290 RET C; desbordamiento
300 Mu4: RRA
310 JR NC,Mu3
320 OR A
330 JR NZ,MU2
340 ADD HL,DE
350 RET
360 [CTRL] C
> P10,350,Mult [ENTER]
```

Este programa crea el texto de la rutina y la graba en cassette. Recuerda poner el cassette en RECORD antes de pulsar [ENTER] tras el comando "P".

Paso 2 — Eliminar errores

Primero, veamos si el texto ensambla correctamente. Utilizaremos la opción 6 para no ver el listado y no se produzca código objeto:

```
> A
Table size: [ENTER]
Optións: 6 [ENTER]
Hisoft GENA3 Assembler. Page 1
Pass 1 errors: 00
Pass 2 errors: 00
*WARNING* MU4 absent
Table used: 74 from 156
>
```

Vemos que tenemos un error en la línea 190, y hemos introducido MU4 en lugar de Mu4, que es la etiqueta apropiada. Editemos la línea 190:

```
> F 190,190,MU4,Mu4 [ENTER]
    190     JR  NZ,          utilicemos el sub:comando "S"
>
```

Ensambla ahora y verás cómo todo funciona perfectamente. Ahora escribamos un programita para probar la rutina.

```
N 3000,10     renumera para poder escribir más texto
 >I10,10
 10; Comprobador de la
 20; rutina de multiplicación
 30
```

```
 40 LD HL,50
 50 LD DE,20
 60 CALL Mult ; Multiplicar
 70 LD A,H; o/p resultado
 80 CALL Aout
 90 LD A,L
100 CALL Aout
110 RET
120
130; rutina o/p A en hex
140
150 Aout: PUSH AF
160 RRCA
170 RRCA
180 RRCA
190 RECA
200 CALL Nibble
210 POP AF
220 Nibble : AND % 1111
230 ADD A, #90
240 DAA
250 ADC A, #40
260 DAA
270 ; salida de un caracter
280 CALL # BB5A
290 RET
300 [CTRL]
```

Ensamblemos las dos rutinas juntas, también con la opción 6

```
>A
>Tab. siz: (ENTER)
>Op: 6 (ENTER)
Hisoft .....
268D   190 RECA
*ERROR*02
hit any key to continue (pulsa tecla para continuar)
Pass 1 errors: 01
Table used: 88 from 201
>
```

En la línea 190, RECA debería decir RRCA, así que:

```
>E 190
>190 RECA
190 (9 espacios) C R [ENTER] [ENTER]
```

Al ensamblar otra vez, utiliza simplemente la opción 4 (sin listado) y suponiendo que no haya habido errores, estamos en el momento de ver cómo funciona nuestra subrutina, así que debemos decirle al editor desde dónde debe ejecutar el código objeto, lo hacemos mediante ENT:

```
>35 ENT $ [ENTER]
```

Ensamblemos de nuevo. El ensamblado debe ser correcto y terminar con los mensajes:

```
Table used: 88 from 202
Executes: 9847
```

o algo por el estilo. Ahora, al ejecutar "R", vamos a multiplicar 50×20, lo cual debe resultar 1000 o #3E8 en hexadecimal.

```
>R [ENTER] 0032>
```

¡No funciona! ¿Por qué? Lista las líneas desde 380 hasta 500 (L380,500). Verás que en la línea 430 la instrucción es OR D seguida de RET NZ. Lo que esto está haciendo es OR lógico entre el registro D y el acumulador A, volviendo con la bandera de error puesta (labanderaC) si el resultado no es cero. El propósito de esto es asegurar que DE < 256 para que la multiplicación no produzca un desbordamiento (overflow), y esto se hace comprobando si D es cero... pero OR sólo funcionará bien en este caso si el acumulador A vale cero al principio, y no estamos nada seguros de eso.

Debemos asegurarnos que A=0 antes de hacer OR D, de otra manera obtendremos desbordamientos impredecibles. Después de investigar un poco, vemos que OR A de la línea 380 podría convertirse en XOR A, con lo cual A vale cero y la bandera de la instrucción SBC HL,DE puesta a su valor adecuado.

>E380 [ENTER]
380 Mult: OR A
380 (8 espacios) I [modo inserción] X [ENTER] [ENTER]
>

Ensambla de nuevo y ejecútalo. La respuesta ahora será #3E8. Podemos comprobar que la subrutina funciona, cambiando las líneas 40 y 50 para multiplicar diferentes números; después ensamblar y ejecutar. Ahora que sabemos que funciona, vamos a grabarla en el cassette.

>P300, 999, Mult [ENTER]

Recuerda poner el cassette a grabar antes de pulsar [ENTER]. Una vez grabada la subrutina, puede ser incluída en un programa tal y como se muestra abajo:

```
500 RET
510
520; incluir la subrutina Mult aquí.
530
540 *F Mult
550
560 ; Siguiente rutina.
```

Al ensamblar lo anterior, el ordenador te pedirá que pulses PLAY en el cassette cuando llegue a la línea 530 tanto en el primero como en el segundo paso. Así que después del primer paso deberás rebobinar la cinta hasta el principio de "Mult", o bien puedes grabarlo dos veces para más comodidad; una para el primer paso y la otra para el segundo. Estudia cuidadosamente el ejemplo anterior y pruébalo tú mismo.

SECCION 1. DESENSAMBLADOR Y MONITOR

MONA3 es relocalizable; simplemente lo cargas en la dirección en que tú deseas ejecutarlo y una vez cargado llamas (CALL) a esa dirección. Si en algún momento vuelves a BASIC y quieres reentrar en MONA3, deberás hacerlo en la dirección anterior +2. Para cargar MONA3 en tu Amstrad CPC-464, simplemente escribe:

RUN" [ENTER]

y pulsa PLAY en el cassette. Primero se cargará un corto programa en BASIC que se autoejecutará e imprimirá el mensaje:

Please wait... loading Mona3

Después de aproximadamente, 100 segundos, el programa estará cargado, e imprimirá un mensaje en pantalla, preparado para que introduzcas algún comando (ver sección 2).

NOTA: Si quieres tener GENA3 y MONA3 al tiempo en memoria, carga siempre primero el que se encuentre localizado más arriba en la memoria. Es conveniente tener GENA3 en la memoria baja (aprox. 1000) y MONA3 más arriba (digamos en 30000). Carga primero MONA 3 en este caso.

Los comandos se ejecutan inmediatamente; no necesitas pulsar [ENTER] después. Todo el panel que tienes en pantalla se renueva después de cada comando para poder observar los resultados de cada uno de ellos.

Muchos comandos requieren hexadecimales. Para introducirlos, puedes escribir tantos dígitos hexadecimales (0-9, A-F ó a-f) como el número tenga seguido de un dígito no hexadecimal. Si éste es un comando válido, se ejecutará tras haber sido procesado cualquier comando anterior. Si el último dígito es un signo menos "—" entonces se toma el número hexadecimal negativo y se devuelve contemplado al dos. Por ejemplo:

1800— da como resultado E800

Si introduces más de 4 dígitos en un número hexadecimal, sólo los 4 últimos se tienen en cuenta.

SECCION 2 COMANDOS EXISTENTES

Los siguientes comandos son los que incluye MONA3. En esta sección, cuando utilizamos [ENTER] para terminar un número hexadecimal, dicho [ENTER] puede ser sustituido por otro carácter no hexadecimal (ver sección 1). "—" significa espacio.

[CTRL] X vuelve a BASIC o al programa desde que el MONA3 fue ejecutado.

[CTRL] D

cambia la base en la que las direcciones de memoria estaban siendo impresas de 10 a 16 o al revés. Al principio, todas las direcciones se escriben en hexadecimal. Pulsando una vez el comando anterior, las direcciones se imprimirán en decimal y pulsando de nuevo, volvemos a la base 16. Este comando no afecta a los contenidos de las memorias (los números de 8 bits se expresan siempre en hexadecimal, y los números introducidos desde el teclado deben encontrarse en dicha base).

[CTRL] A

Imprime una página de código máquina, empezando por la dirección del puntero. Es útil para ver qué instrucciones vienen detrás. Pulsando [CTRL] A de nuevo, volvemos al panel o pulsando cualquier otra tecla, obtendremos otra página de código máquina desensamblada.

"→" cursor derecha

Incrementa el puntero en una unidad.

"←" cursor izquierda

Resta uno al valor del puntero.

"↑" cursor arriba

Resta ocho al valor del puntero. Es útil para retroceder rápidamente.

"↓" cursor abajo

Suma ocho al valor del puntero. " " avanzar ".

"G"

Recorre la memoria buscando una cadena determinada.

Aparecerá un símbolo ":" a lo que deberás responder con el primer byte de la cadena que quieres buscar seguido de [ENTER], sigue introduciendo bytes (y [ENTER]) hasta que hayas definido toda la cadena. Pulsa [ENTER] otra vez y la memoria empezará a ser rastreada. Cuando el ordenador encuentra la cadena, el panel será "actualizado"; el puntero se colocará en el primer carácter de la cadena. Ejemplo: supongamos que quieres buscar #3E#FF (2 bytes) desde la posición de memoria #8000. Procedemos como sigue:

M: 8000 [ENTER] coloca el puntero en //8000
G: 3E [ENTER] define el primer byte de la cadena
FF [ENTER] define el segundo byte de la cadena
[ENTER] fin de la cadena

Ahora la memoria está siendo rastreada hasta que se encuentre dicha cadena por primera vez, momento en el cual verás el panel remodelarse. Para buscar sucesivas apariciones de la cadena en memoria, utiliza el comando "N".

"H"

Convierte un número decimal a su equivalente hexadecimal. Para introducir el número, basta con escribir el número seguido de un carácter no decimal y obtendrás el número traducido a hexadecimal. Pulsa cualquier tecla para volver a MONA3.

Ejemplo:

H: 41472 = A200 (el espacio, en este caso, es el carácter no decimal)

"I"

Copia inteligente. Se utiliza para copiar un bloque de memoria de una posición a otra. Es inteligente puesto que el bloque puede copiarse a posiciones que solaparían la zona anterior. Es decir, podemos trasladar un bloque de, digamos, 500 bytes desde su posición inicial a, simplemente, una posición más adelante (por ejemplo, de 1000 a 1001). "I" te pedirá la dirección de comienzo y final del bloque a copiar, y la posición en donde quieres copiarlo. Todos los números deben ser hexadecimales. Si la dirección de comienzo es mayor que la del final, el comando no tendrá efecto alguno.

"J"

El número que se te pedirá después de lso dos puntos ":" es la dirección desde la que debe ejecutarse el código objeto en hexadecimal. Una vez hecho esto, el stack interno se reinicializa,

se borra la pantalla y comienza la ejecución. Si quieres volver al panel en un punto determinado de la ejecución, deberás colocar un "breakpoint" en dicha posición (ver el comando "!"). Ejemplo:

J: B000 [ENTER] ejecuta desde la posición #B000

pulsando [ESC] antes de terminar la dirección, el comando no será ejecutado.

[CTRL] C

Continúa la ejecución desde la dirección actual del contador de programa (PC). Este comando se usa casi siempre después de "!". Un ejemplo nos aclarará todo esto: supongamos que has utilizado [CTRL] S para ejecutar el programa de debajo instrucción por instrucción y has llegado a la posición #8920. Ahora estás interesado en ver el estado de los flags después de la llamada a la subrutina en #8800.

Procede como sigue:

Sitúa un "breakpoint" en #892D utilizando el comando "!" (recuerda colocar el puntero en el sitio adecuado mediante "M"), pulsa [CTRL] C. La ejecución continúa desde donde el puntero de programa está (en este caso #9820) hasta encontrar el "breakpoint" en #892D. Pulsa una tecla y aparecerá el panel con el estado de las banderas tras ejecutar la subrutina #8800 como deseabas.

```
891E  3EFF          LD    A,—1
8920  CD0090        CALL  #9000
8923  2A0080        LD    HL,(#8000)
8926  7E            LD    A,(HL)
8927  111488        LD    DE,#8814
892A  CD0088        CALL  #8800
892D  2003          JR    NZ,lab1
892F  320280        LD    (#8002),A
8932  211488 lab1:  LD    HL,#8814
```

"L"

Tabula, o lista, un bloque de memoria desde la dirección del puntero. Borra la pantalla e imprime la representación hexadecimal y los equivalentes ASCII de los 160 bytes, empezando desde el valor del puntero. Las direcciones se escriben en decimal o hexadecimal dependiendo del estado el panel (ver [CTRL] D). La pantalla consiste en 20 filas de 8 bytes cada una, con el ASCII al final de ellas. A cualquier valor mayor de 127 se le resta 128 y entre 0-31 inclusive se representan como ".".

Al final de una página de listado, puedes volver al panel pulsando [ESC] o continuar con la siguiente página pulsando cualquier tecla.

"M"

Coloca el puntero en una dirección determinada. Se te pedirá una dirección hexadecimal (ver sección 1). El puntero se colocará en dicha dirección y el panel mostrará la memoria de ahí en adelante. "M" es fundamental antes de introducir código máquina, tabular memoria, etc.

"N"

Busca la siguiente aparición de la cadena hexadecimal especificada por el comando "G".

"O"

Va al destino de un salto relativo. El comando toma el byte de la dirección del puntero, lo trata como un salto relativo y actualiza la memoria de acuerdo a este salto.

Ejemplo:

Supongamos que el puntero está en #6800 y que los contenidos de #67FF y #6800 son #20 y #16 respectivamente; esto podría interpretarse como JR NZ, $+22. Para ver dónde saltaría en caso de darse la condición NZ, simplemente pulsa "O" cuando el puntero apunte el byte de desplazamiento.

Recuerda que los saltos relativos mayores de #7F (127) se tratan como negativos. Ver comando "U"

"P"

Llena la memoria entre dos límites de un valor dado. "P" pide la dirección de comienzo ("First:"), la de finalización ("Last:"), y el

valor con el que hay que llenar el trozo de memoria elegido ("With:"). Ejemplo:

First: 7000
Last: 77FF
With: 55
Llena desde #7000 hasta #77FF con #55.

"R"

Lee código objeto del cassette, creado por el comando "W" de MONA3 o bien los comandos "O" o "T" de GENA3. Se te pedirá el nombre del programa (pulsa [ENTER] si no lo sabes) y la dirección donde quieres cargarla. Esta dirección **no** puedes omitirla.

" > "

Coloca un "breakpoint" después de la instrucción en la que está el puntero y continúa la ejecución. Ejemplo:

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

Estás ejecutando el programa de arriba instrucción por instrucción y llegas a #9001 con un valor en el acumulador A distinto de cero, así que después de OR A, se dará la condición NZ. Si continúas ejecutando paso a paso con [CTRL] S el programa se dirigirá a la dirección #9800. Si no quieres continuar con la opción "instrucción por instrucción", utiliza el comando " " en la dirección #9001, el CALL se ejecutará automáticamente y el programa se parará en #9004 para seguir ejecutándolo paso a paso.

"S"

Actualiza el puntero de memoria, dándote el valor actual del stack (indicando por SP). Es útil para ver la dirección a donde una subrutina debe volver.

"T"

Desensambla un bloque de código con posibilidad de enviarlo a la impresora. Se te preguntará el primer y último byte a desensamblar en hexadecimal, y si quieres copiarlo en impresora, en caso afirmativo, contesta "Y" o cualquier otra tecla para impresión en pantalla. También se te pedirá la dirección de comienzo del archivo de texto que quieras que el desensamblador produzca. Si no quieres producir tal archivo, simplemente pulsa [ENTER]. El archivo creado será compatible con GENA3.

Si quieres utilizar un archivo con GENA3 debes generarlo en la primera dirección dada por el comando "X" del ensamblador o bien moverlo a ella. También debes especificar la dirección del final del archivo, que te la da el desensamblador e introdúcela en el TEXTEND de GENA3 (ver manual de GENA3 sección 3.2), llama a GENA3 reinicializando de forma que conserves el texto original.

Si durante el desensamblado, el texto invade MONA3, el desensamblado se detiene. Pulsa cualquier tecla para volver al panel. Si quieres una copia en impresora, se te pedirá la dirección ("Workspace:") de comienzo de una zona vacía de memoria que se utiliza como tabla de símbolos para las etiquetas generadas por el desensamblador. Se necesitan 2 bytes para cada etiqueta. No se puede omitir esta dirección. Después se te pedirán, repetidamente, las direcciones iniciales y finales de cualquier área de datos que existan en el bloque que quieres desensamblar. Estas áreas contienen datos que no quieres que se interpreten como instrucciones de Z80 (DEFB, etc.).

Si el valor del byte en la zona de datos está entre 32 y 127, se imprimirá la representación en ASCII. Si has acabado de definir áreas de datos, pulsa simplemente [ENTER] a ambas preguntas. El comando "T" usa una zona al final de MONA3 para cargar la dirección del área de datos, así que puedes definir tantas áreas como memoria tengas; cada área requiere 4 bytes para almacenarse. "T" destruye todos los "breakpoints" (ver comando "!").

La pantalla se borrará. Si pediste que se creara un archivo de texto, habrá un corto retardo mientras la tabla de símbolos se construye. Después, empezará a aparecer la lista desensam-

blada en pantalla o impresora. Puedes detener el listado pulsando cualquier tecla, y después [ESC] para volver al panel o cualquier otra para continuar. Si una instrucción inválida aparece, se desensambla como NOP con un asterisco "*".

Al final del desensamblado, si has pedido un archivo, aparecerá en pantalla el mensaje "End of Text xxxx"; xxxx es la dirección en decimal o hexadecimal que debes POKEar (primero el byte menos significativo) en el TEXTEND de GENA3 para que éste pueda manejar este archivo.

Las etiquetas se generan en la forma LXXXX (donde XXXX es la dirección hexadecimal de la etiqueta), pero sólo si dicha dirección se encuentra dentro de los límites de la zona a desensamblar, si no es así, simplemente se dará la dirección.

Ejemplo:

```
T
First: 8B [ENTER]
Last: 9E [ENTER]
                        Printer? Y
Text: [ENTER]
First: 95 [ENTER]
Last: 9E [ENTER]
First: [ENTER]
Last: [ENTER]
008B   FE16        GP     #16
008D   3801        JR     C,L0090
008F   23            INC    HL
0090   37   L0090    SCF
0091   225D5C        LD     (#5C5D),HL
0094   C9            RET
0095   BF524E        DEFB   #BF, "R","N"
0098   C4494E        DEFB   #C4,"I","N"
009B   4B4559        DEFB   "K","E","Y"
009E   A4            DEFB   #A4
```

"U"

Se utiliza en conjunción con el comando "O". Recuerda que "O" actualiza la pantalla de acuerdo a un desplazamiento relativo, es decir: muestra los efectos de JR o DJNZ. "U" devuelve la memoria a donde al comando "O" fue ejecutado (tiene efecto contrario a "O").

Ejemplo:

```
7200  47          71F3  77
7201  20          71F4  C9
7202  F2          7175  F5
7203  06          71F6  C5
display 1         display 2
```

Estás en el display 1 y quieres saber a dónde va el salto 20F2, as- que pulsa "O" y la pantalla se colocará en el display 2. Si al cabo de un rato quieres volver a donde estabas, pulsa "U" y volveremos al display 1.

"V"

En conjunción con el comando "x". "V" es similar al comando "U", excepto que actualiza el display memoria a donde se encontraba antes del último comando "X". El ejemplo anterior es perfectamente aplicable a esta caso con pequeñas modificaciones triviales.

"W"

Graba un bloque de memoria en cassette bajo el nombre que se te pedirá, asimismo se pide el byte inicial y final del bloque a grabar.

"!"

Coloca un "breakpoint" en el puntero. Un "breakpoint" de MONA3 es simplemente una llamada a la subrutina que imprime el panel para permitir al programador revisar los registros y banderas de Z80. Si quieres interrumpir la ejecución en #9876, utiliza "M" para colocar el puntero en #9876 y "!" para colocar el "breakpoint". Los 3 bytes de código que estaban originalmente en dicha dirección se guardan para reponerlos cuando suprimamos el "breakpoint". Cada breakpoint requiere

5 bytes de la zona de memoria destinada a ellos, esto es, al final de MONA3.

Los "breakpoints" sólo pueden colocarse en RAM por razones obvias, así que cuidado con saltar al segundo o tercer byte de los trece que un "breakpoint" utilice pues los resultados pueden ser fatales.

"X"

Se utiliza para colocar el puntero en la dirección de CALL ó JP. "X" toma la dirección de 16 byts especificada por el byte donde se encuentra el puntero y el siguiente, y actualiza la pantalla situándose en esa dirección.

Ejemplo:

Supongamos que quieres dirigirte a la subrutina que CD0563 llama. Para hacerlo, coloca el puntero en 05 y pulsa "X".

"Y"

Introduce caracteres ASCII desde el puntero. "Y" te da una línea nueva en la que puedes introducir caracteres ASCII directamente del teclado, introduciéndose en la memoria, empezando por la posición a la que el puntero apunta. La cadena de caracteres debe finalizar con [ESC] y puedes usar [DEL] para borrar caracteres. Al terminar, el puntero se coloca después del final de la cadena.

[CTRL] S

Paso a paso.

[CTRL] S ejecuta la instrucción en la que nos encontremos y actualiza el panel para reflejar los cambios acusados. No se puede ejecutar paso a paso las instrucciones EXX o EX AF,AF', aunque sí cualquier zona de RAM o ROM.

Ahora sigue un ejemplo que deberá aclarar el uso de muchos comandos de MONA3.

Supongamos que tenemos 3 secciones de programa en el ordenador. La 1a sección es el programa principal que carga HL y DE con números y llama a una subrutina que los multiplica (2a sección) y finalmente llama a otra subrutina que imprime el resultado en pantalla (3a sección).

```
7080  2A0072     LD     HL,(#7200)   ;SECTION 1
7083  ED5B0272   LD     DE,(#7202)
7087  CDC071     CALL   Mult
708A  7C         LD     A,H
798B  CD1471     CALL   Aout
708E  7D         LD     A,L
708F  CD1D71     CALL   Aout
7092  210000     LD     HL,0
7100  AF         Mult:  XOR    A       ;SECTION 2
7101  ED52              SBC    HL,DE
7103  19                ADD    HL,DE
7104  3001              JR     NC,Mu1
7106  EB                EX     DE,HL
7107  B2         Mu1    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  F5         Aout:  PUSH   AF      ;SECTION 3
```

```
711E  0F                  RRCA
7120  0F                  RRCA
7121  0F                  RRCA
7122  CD2671              CALL   Nibble
7125  F1                  POP    AF
7126  E60F        Nibble: AND    %1111
7128  C690                ADD    A,#90
712A  27                  DAA
712B  CE40                ADC    A,#40
712D  27                  DAA
712E  CD5ABB              CALL   #BB5A
7131  C9                  RET
   .
   .
7200  1B2A                DEFW   10779
7202  0200                DEFW   2
```

Ahora vamos a ver cómo funciona. Para eso ejecutamos los siguientes comandos:

```
M:7080 [ENTER]    —Coloca el puntero en #7080
7080              —Coloca el PC en #7080
[CTRL] S          —Paso a paso
[CTRL] S          —Paso a paso
[CTRL] S          —Ejecuta el CALL
M:7115 [ENTER]    —Salta el pre-proceso de los números
!                 —Coloca un "breakpoint"
[CTRL] S          —Paso a paso
[CTRL] S          —Ve al salto relativo
[CTRL] S          —Paso a paso
[CTRL] S            "    "
[CTRL] S            "    "
[CTRL] S            "    "
[CTRL] S            "    "
[CTRL] S            "    "
[CTRL] S            "    "
[CTRL] S          —Vuelve a la subrutina de multiplicar
[CTRL] S          —Paso a paso
[CTRL] S          —Ejecuta el CALL
M:7128 [ENTER]    —Coloca el puntero en el bit interesante
.                 —Coloca un "breakpoint"
[CTRL] C [ENTER]  —y continúa.
[CTRL] S          —Paso a paso
[CTRL] S            "    "
[CTRL] S            "    "
[CTRL] S            "    "
S
!
[CTRL] C [ENTER]  —y continúa
[CTRL] S          —paso a paso
S                 —vuelve de la rutina Aout.
!
[CTRL] C [ENTER]
[CTRL] S          —paso a paso
                  —ejecuta CALL Aout.
```

Trabaja con el ejemplo.

[CTRL] L

lista el programa en la impresora.

Cómo modificar la memoria

Los contenidos de una dirección dada por el puntero de memoria pueden ser modificados introduciendo un número hexadecimal seguidos de cualquier carácter no hexadecimal. Si este carácter es un comando, después de introducir el número en la posición del puntero, será ejecutado. Si el carácter no es un comando, será ignorado.

Ejemplo:

```
F2—       #F2 se introduce y el puntero avanza en 1
EM:E00    #0E se introduce en la posición del puntero y
          éste se coloca en #E00. "—" no tiene efecto.
2A5D—     #5D se introduce y el puntero no cambia pues
          el último carácter es inocuo.
```