ENSAMBLADOR

INSTRUCCIONES
SOFTWARE SPECTRUM

SPECTRUM
INSTRUCCIONES GENERALES DE CARGA

1. Rebobina la cinta hasta su principio.
2. Conecta el (EAR) del ZX-SPECTRUM con el (EAR) del cassette.
3. Ajusta el volumen del cassette a 3/4 del volumen máximo (aproximadamente). Si el cassette tiene control de agudos, posiciónalo en el máximo, y el de graves en el mínimo.
4. Escribe LOAD " " o bien sigue las instrucciones particulares de carga de cada cinta, y a continuación 'ENTER'.
5. Pulsa PLAY en el cassette.

ENSAMBLADOR — SPECTRUM (48 K)

MANUAL DE INSTRUCCIONES

INTRODUCCION
El objetivo del SPECTRUM ENSAMBLADOR es simplificar el proceso de programación en lenguaje de máquina en el ZX SPECTRUM. Proporciona una amplia serie de facilidades ocupando muy poco de la memoria RAM disponible para el usuario. El programa ocupa 9K y se sitúa en la parte superior de la memoria. De esta forma el BASIC es casi igual que si no estuviese presente el ensamblador, lo que permite utilizar subrutinas en ensamblados dentro de los programas en Basic.

DESCRIPCION GENERAL
Hay dos partes principales en el programa:

1) El EDITOR / ENSAMBLADOR. Puedes introducir tu programa en ensamblados con el EDITOR y luego ensamblarlo con el ENSAMBLADOR. El EDITOR ha sido especialmente diseñado para la programación en lenguaje de máquina: utiliza scroll hacia arriba y hacia abajo en la pantalla, búsqueda de cadenas, inserción y borrados de líneas y caracteres, repetición automática en todas las teclas, sistema de introducción de etiquetas y movimiento del cursor en las cuatro direcciones. Permite hasta 31 caracteres por línea. El ENSAMBLADOR es muy versátil: puede ensamblar todas las memorias y puede manejar etiquetas. Permite la entrada de números en decimal y hexadecimal, y permite la introducción de texto y comentarios.
2) EL MONITOR proporciona utilidades que permiten probar, ejecutar y corregir el programa. También proporciona algunas útiles subrutinas (tales como "imprimir en pantalla" o "ver qué tecla está pulsada").

El Objeto de este manual es explicar el uso del EDITOR/ENSAMBLADOR y el MONITOR, proporcionar algunos trucos para los que comienzan y una lista del juego de instrucciones del Z80. El SPECTRUM ASSEMBLER es un programa muy potente y, correctamente usado, ayuda al programador de lenguaje máquina.

COMENZANDO
Lee el capitulo 20 del manual de Sinclair sobre cómo cargar un programa del cassette. Para cargar el ensamblador escribe:

y "LOAD". La cinta tiene dos copias, una en cada cara.
El programa tarda casi un minuto en cargar. Está situado en los 9K superiores por encima del RAMTOP. Comienza por la dirección de DB00 h (56064 decimal). De esta forma, con el ensamblador presente puedes cargar cualquier programa en BASIC (máximo 31K) o en lenguaje de máquina (ten cuidado para no situarte encima del ensamblador). De hecho, estos programas pueden estar presentes en el SPECTRUM antes de cargar el ensamblador.
Para tener acceso al ensamblador, escribe:

RANDOMIZE USR 6E 4

Estarás situado entonces en el Modo de Ordenes. Cualquier orden se introduce a través del teclado.
Para volver al BASIC, pulsa Q dos veces.
Si otros programas no dejan suficiente memoria para el ENSAMBLADOR, al escribir RANDOMIZE USR 6E 4 y pulsar ENTER obtendrás un error "4 Out of Memory".

GRABANDO LOS PROGRAMAS

El código FUENTE (el texto que introduces utilizando el Editor) es situado automáticamente en una REM en la línea 2.
El código OBJETO (el lenguaje de máquina creado por el ENSAMBLADOR) es situado en una REM en la línea 1, a no ser que hubiese una instrucción ORG al principio del código fuente.
De esta forma, el lenguaje de máquina se convierte en una parte integral del programa de BASIC.
Para grabar, pues, los códigos FUENTE y OBJETO (si no has utilizado ORG), simplemente vuelve a BASIC y graba el programa normalmente.
NOTA: *Si no has terminado el programa, puedes borrar el código OBJETO (línea 1) antes de grabar, ya que puedes, sencillamente, reensamblar el código FUENTE más tarde. Si ya has terminado el programa, no necesitas el código FUENTE y. por tanto, borra la línea 2. De todas formas, es mejor guardar también una copia que contenga sólo el código Fuente por si hay que hacer arreglos más tarde.*
Si has utilizado una instrucción ORG, puedes grabar el código OBJETO utilizando la instrucción SAVE del ENSAMBLADOR.

MODO DE INTRODUCCION DE INSTRUCCIONES
(MODO COMANDO)

Estarás en este modo cuando aparezca el mensaje "SPECTRUM ASSEMBLER". Al pulsar la tecla H aparecerá una lista de todas las instrucciones. Pulsa cualquier otra tecla para volver al modo normal.
En la parte inferior de la pantalla hay información sobre la cantidad de memoria ocupada (redondeada a los 0'25K más próximos -256 Bytes) por:

SOURCE:	La longitud de la línea 2 REM.
OBJECT:	La longitud de la línea 1 REM. Si utilizaste una instrucción ORG, este número será 0, ya que no habrá línea 1.
BASIC:	Memoria ocupada por el BASIC sin contar las líneas 1 y 2 REM.
SPARE:	Esta es la memoria no ocupada por todo lo anterior menos la que ocupa la tabla de etiquetas. De esta forma, cualquier aumento en la longitud (1), el código fuente (2), el código objeto si no utilizamos ORG (3), tabla de etiquetas (4), el programa en BASIC, tendrá como consecuencia una disminución de esta memoria sin usar.

EL EDITOR

Para acceder al EDITOR desde el modo de instrucciones directas (modo comando), pulsa E.
Aparece una pantalla de texto con el cursor en la parte superior de la pantalla y siete caracteres en la parte izquierda.
El cursor se colocará al principio del texto si:

a) Acabas de introducir el ENSAMBLADOR.
b) Acabas de ENSAMBLAR sin errores.
c) Has obtenido un error "Out of memory".

Si dejas el EDITOR, al volver a él más tarde el cursor estará en la misma posición en el que lo dejaste, siempre que no hayas ensamblado o hayas dejado el ENSAMBLADOR mientras.
El EDITOR está compuesto por dos partes principales: El EDITOR de PANTALLA y el EDITOR de LINEA.
Al principio controlas el de PANTALLA, y podrás utilizar los siguientes comandos:

Pulsa: CAPS SHIFT 6: baja el cursor 1 línea.
CAPS SHIFT 7: sube el cursor 1 línea.

(Con estas dos órdenes, la pantalla asciende para mantener el cursor en pantalla).

SYMBOL SHIFT D: Borrar la línea actual.
SYMBOL SHIFT E: Inserta una línea en la posición del cursor.
SYMBOL SHIFT G: Imprime el texto desde el cursor en adelante en impresora. Puedes pulsar BREAK para pararlo, en cuyo caso volverás a BASIC.
SYMBOL SHIFT O: Vuelta al modo de instrucciones directas.
SYMBOL SHIFT S: para buscar una cadena. Pulsa SHIFT A para buscar una etiqueta. El n.° de espacios no es importante. 1 espacio = muchos espacios. Por eso. PEDRO no es lo mismo que P _ EDRO. pero P _ EDRO si es lo mismo que P…….EDRO.
Si no encuentra la cadena, el cursor se queda en la misma posición, y si la encuentra, éste se moverá. Pulsa SYMBOL SHIFT O para parar la búsqueda.
SYMBOL SHIFT T: Mueve el cursor al principio del texto.

Te introduces al EDITOR de LINEA, pulsando:

a) Una letra, número u otro símbolo.
b) Cursor izquierdo o derecho.
c) SHIFT A.
d) RUBOUT o INSERT.

Nótese que no ocurre nada en pantalla para cuando accedes a este EDITOR, excepto la función de la tecla que pulsaste. Los comandos del EDITOR de PANTALLA no son accesibles ahora.

COMANDOS EN EL EDITOR DE LINEA

Pulsa:

CAPS SHIFT 8:	Mover el cursor 1 carácter hacia la derecha.
CAPS SHIFT 5:	Mover el cursor 1 carácter hacia la izquierda.
CAPS SHIFT 9:	Inserta un carácter.
CAPS SHIFT 0:	Borra un carácter.
SYMBOL SHIFT Q:	Deja la línea como estaba cuando accediste al EDITOR de LINEA y al volver al EDITOR de PANTALLA.
SYMBOL SHIFT A:	Introduce una etiqueta. Mueve el cursor al extremo izqu. de la pantalla. Sólo funciona si el cursor está en la posición del 7.º carácter (su posición normal) y si no hay ya etiqueta allí. Deja líneas en blanco para que sea más fácil de leer.
ENTER:	Para introducir la línea y volver al EDITOR de PANTALLA.

GENERAL
Todas las teclas tienen AUTO-REPEAT (excepto SHIF Q) después de tenerlas presionadas durante más de un segundo.
Si se acaba la memoria volverás al modo comando y aparecerá un mensaje de error: la línea que estabas introduciendo se borrará, así que puedes volver al EDITOR para tener algo de espacio.
Si quieres cambiar una etiqueta. SHIFT A no moverá el cursor a la izquierda, así que usa el cursor de la izquierda.
Para borrar una etiqueta debes situar el cursor en la posición del 7.° carácter (o al final de la etiqueta si su longitud es mayor que 6 caracteres). y pulsa entonces la tecla RUBOUT; con la ayuda de la auto-repetición borrará la etiqueta, y el último RUBOUT desplazará toda la línea 6 caracteres hacia la izquierda.
Puedes usar las letras minúsculas con el EDITOR.
Pulsa CAPS LOCK para pasar a letras minúsculas. Esto sólo funcionará si ya estabas en minúsculas antes de cargar el Ensamblador. Los memóricos en minúscula se ensamblarán igual que los que se escriban en mayúscula. Sin embargo sí hay diferencia en las etiquetas: así FRED no será lo mismo que fred.
Se pueden mezclar libremente mayúsculas y minúsculas.
Para volver a mayúsculas, vuelve a pulsar CAPS LOCK.
Al abandonar el EDITOR, todo el texto se encontrará en una sentencia 2 REM.

ESCRIBIENDO CODIGO MAQUINA

El objeto de este manual no es enseñar código máquina, aunque se dan algunas pistas y se recomiendan libros.
Para ensamblar el código fuente, selecciona el modo comando: Pulsa A, luego ENTER. El código fuente se ensamblará ahora: Si no se da ningún mensaje de error, el ensamblado ha funcionado bien. Los posibles mensajes de error están listados en la página 11.
Almacenamiento del código objeto.
El código objeto se almacena normalmente en una REM en la línea 1, que se crea automáticamente. La primera dirección de este código es 5CD2H o 23762 decimal. Sin embargo tú puedes ensamblar a partir de cualquier dirección usando la introducción ORG A000; ensamblará a partir de A000 h. En este caso no se creará ninguna línea 1 y cualquier línea ya existente con este número resultará borrada.
Al final del código ensamblado se pone automáticamente una instrucción de salto al ensamblador (EA90 h). Puedes poner esta instrucción en cualquier lugar de tu programa, incluso en una subrutina para comprobar el correcto funcionamiento de tu programa. No hay problema si la longitud tamaño del stack ha cambiado desde que pulsaste RUN por primera vez, sino que simplemente recibirás un mensaje STACK ERROR. (Recibirás este mensaje siempre que vuelvas al Ensamblador desde una subrutina). Es más, el Ensamblador reajusta el puntero que alteró el stack, así que tu puedes ejecutar tu programa corruptor de stacks todas las veces que quieras sin peligro de perder el programa. Si usas una instrucción RET en su lugar, volverás al BASIC con el peligro de que se quede bloqueado si el STACK está desajustado.
NOTA: para ejecutar, correr tu rutina, usa el comando del monitor RUN (Pág. 13).

NUMEROS
El ensamblador usa principalmente números hexadecimales(base 16) de:
8 bit — 0 a FF (1 dígito ó 2 dígitos).
16 bit — 000 a FFFF (3 o 4 dígitos).
Se tomarán como decimales si van precedidos de los signos + o —.

NUMEROS EN «MODO INMEDIATO»

«Modo Inmediato» significa simplemente, que no forma parte de otra instrucción o memórico. Estos números se introducen en el código objeto directamente. Esto es: 21 789A

Si ahora miras el contenido de memoria en 5CD2 usando el modo de edición de memoria (pág. 13) verás:

5CD2 21
5CD3 9A los números de 2-byte llevan siempre
5CD4 78 el byte de menos peso delante.

Nótese que ésto significa lo mismo que LO HL. 789A pero con un ahorro de tiempo y espacio, si puedes recordar todos los códigos. También se pueden usar números decimales en modo inmediato, se interpretarán siempre como números de 16-bit.

NUMEROS CON ETIQUETAS Y MEMORICOS

El ensamblador decidirá él mismo si un número debe ser de 8 bit o de 16 bit y procederá según el caso.

Ejemplo: Carga de etiquetas.

```
TEST = + 64
        LD HL. TEST: Se espera un número de 16 bit.
        LD A. TEST: Se espera un número de 8 bit.
```

La etiqueta TEST se tratará como un entero de 16 bit o de 8 bit como se indica.

Sin embargo, si ASSEMBLER está esperando un número de 8 bit y el número es mayor de FF o 255 se dará un mensaje NUMBER ERROR durante el proceso de ensamblaje.

Ejemplo: Números.

```
        LDHL. 1      :OK
        LOA. +13     :OK
        LD A. 101    :ERROR
```

Si se espera un número de 16 bit y se da un número mayor, el ensamblador contestará con SYNTAX ERROR, ya que ningún número será nunca mayor que FFFF o + 32767 o menor que — 32767.

Si quieres cargar un registro con un número que también es un registro, tienes que escribir un 0 delante del número.

Ejemplo: LD A. B : Copia el registro B en A
LD A.OB : Guarda el número B en A

ETIQUETAS

Una etiqueta es una colección de letras y números usados para definir una dirección o un número. Siempre que has pulsado SHIFT A en el EDITOR y has movido el cursor al extremo izquierdo, el ASSEMBLER entiende que sigue una definición de una etiqueta.

Las etiquetas se pueden definir de dos formas:

a) Etiquetas predefinidas: A una etiqueta se le asigna un número.
b) Etiquetas que definen una posición: La etiqueta está definida por su posición en el «código objeto».

Ejemplo: Rutina de puntuación.

```
SCORE1 :    A000: etiqueta predefinida.
            JRBEGIN
SCORE :     000: etiqueta definida por su posición reserva 2 bytes para SCORE.
BEGIN :     LD HL, (SCORE): carga en HL el contenido de SCORE.
            INC HL: lo incrementa HL.
            LD (SCORE). HL; guarda el nuevo valor de SCORE.
            LD (SCORE 1). HL; almacena una copia de SCORE en A000 HEX.
```

Se puede acceder a cualquier etiqueta en un rango de —9a +9.

```
Ejemplo:    LD A, (PTR + 4)
            JR END
            PTR 0 1 2 3 4 5; números de 1 byte.
            END
```

Después de ejecutar el programa A contendrá 4.
Si no se especifica ninguna diferencia, se asume que ésta es cero, y en el caso anterior A contendría 0.
Si la diferencia no se encuentra entre los limites ± 9 se producirá un OFFSET ERROR al ensamblar.
N. B. No se deben usar etiquetas con el mismo nombre que un número normal, un registro o un memórico. Por ejemplo DAB no es válido.
También se pueden usar etiquetas en modo inmediato. Esto es: 2A FRED es lo mismo que LD HL. (FRED).
El valor de cualquier FRED que aparezca en tu código fuente resultará insertado en el código objeto. Así, si te equivocas al escribir un memórico, el ensamblador pensará que estás intentando acceder a una etiqueta que no está definida y se producirá un mensaje de error.
Nótese que FRED es una etiqueta diferente a Fred.
El ensamblador almacena todas las etiquetas con sus direcciones en una tabla de etiquetas mientras está ensamblando. La cantidad de memoria que usa la tabla de etiquetas no está limitada, y puede crecer tanto como le permita la memoria libre.
Los memóricos son los 280 memóricos standard de Zilog con la excepción de los siguientes:

```
EX — AF     en lugar de EX AF, AF'
IM0         en lugar de IM — 0
IM 1        en lugar de IM — 1
IM2         en lugar de IM — 2
HALT        en lugar de HLT
JP y JR no tienen comas, esto es:    JR NZ + 7
                                     JR JM
```

SEPARADORES

Cada instrucción puede separarse por espacios finales de líneas, o un número de líneas en blanco (es conveniente separar las subrutinas por una o dos líneas en blanco para facilitar la lectura del programa).
Las secciones dentro de una introducción pueden separarse por espacios, pero no por finales de línea.
Esto es: LD A, (HL) con la excepción de los siguientes que deben teclearse exactamente igual que como sigue:

```
LD — A,I        — es un espacio.
LD — I,A
LD — R.A
LD — A.R
EX — A.F
EX — DE,HL
EX — (SP) HL
```

NOTA: Los memóricos pueden teclearse tanto en mayúsculas como en minúsculas, o una mezcla de ambos.

TEXTO

Se puede ensamblar directamente en el código objeto cualquier texto contenido entre comillas " " (shift P). Esto es una forma simple de introducir mensajes que luego deben ser impresos, y el texto también puede usarse en comandos de lenguaje de ensamblador.

```
Ejemplo:    Impresión de un mensaje.
            CALL FCEI; subrutina para la impresión de texto.
            "ASI SE INTRODUCE EL TEXTO" FF; el texto termina con FF.
Ejemplo:    Texto en lenguaje de ensamblador.
            LD HL, FFFF; dirección de comienzo —1
NEXT        INC HL; incrementa el puntero
            LD A, (HL); lee el contenido
            CP " *"; comprueba si es *
            JR NZ NEXT; si no es así salta a NEXT
```

Este calcula la dirección en que" * " aparece por primera vez empezando en la dirección 0000 en LH. De esta forma no es necesario saber los códigos de los caracteres.

COMENTARIOS

Los comentarios deben procederse por un punto y coma; El texto que siga será ignorado por el ensamblador.

USANDO LOS REGISTROS INDEXADOS

Solamente debes usar el registro IX en el Spectrum, a no ser que deshabilites la interrupción (instrucción DJ), porque I.Y contiene el comienzo de las variables del sistema y se usa cada vez que se interrumpe el programa (50 veces por segundo). Si deshabilitas las interrupciones, debes rehabilitarlas (y restaurar su contenido) antes de:

a) Volver al BASIC.
b) Volver al ensamblador.
c) Usar la subrutina de lectura del teclado del ensamblador.

Siempre que se use un registro indeseado, se debe especificar un desplazamiento entre O y FF.
O —128 y + 127.

```
Ejemplo:    LD A, (JX 4E); desplazamiento en hexadecimal.
            LD A. (IX + 42): desplazamiento en decimal.
```

Si el desplazamiento está fuera de rango o tiene una sintaxis incorrecta se producirá un OFFSET ERROR.

USANDO EL SET DE REGISTROS ALTERNATIVOS

Solamente debe usarse el par de registros alternativos AF' en el Spectrum; si usa los otros pueden producirse resultados impredecibles. Sin embargo puedes usar la instrucción EXX para experimentar.

MENSAJES DE ERROR

Si se imprime un mensaje de error al ensamblar un programa, presionando E puede listarse el texto desde la línea donde se produjo el error.
Los mensajes de error que puedes obtener son:

LABEL ERROR o error de etiquetado.

1. Has intentado acceder a una etiqueta que no está definida.

2. Definición ilegal de una etiqueta, esto es, que empieza con un número. Ten en cuenta que si escribes mal un memórico probablemente obtengas este error, ya que el ensamblador pensará que estás intentando acceder a una etiqueta en modo inmediato.

RELATIVE JUMP ERROR o error de salto relativo.
Esto se produce si un salto relativo esta fuera de rango:

a) El salto no está entre +127 o —128.
b) Si el salto contenido en una etiqueta está fuera de rango.

OFFSET ERROR o error de desplazamiento.

1. Al cargar un registro con una etiqueta más un desplazamiento que está fuera de rango, esto es: LD HL, (FRED + 10) ya que sólo está permitido ±9.
2. Usando registros indeseados con un desplazamiento ilegal, esto es: ADD A, (IX + *) o uno que esté fuera de rango.

IX, Y + 127 }
- 128 } están permitidos.

NUMBER ERROR
Este error se producirá cuando intentes usar un número demasiado grande para una determinada instrucción. Por ejemplo: LD A, 1000.
OUT OF MEMORY
Cuando el código objeto es demasiado largo, la tabla de etiquetas es demasiado grande, o por una combinación de ambos.
SYNTAX ERROR
Se produce con todos los demás errores. Por ejemplo: ADD A, *
NOTA: Si se produce un error al ensamblar, no se debe correr en ningún caso el programa, va que éste solamente se habrá ensamblado parcialmente.

MONITOR

El MONITOR de ZX-ASSEMBLER es un complemento esencial del Ensamblador/Editor. Permite depurar y comprobar rápidamente programas en código máquina.
Todos los comandos se inician pulsando una sola tecla en «modo de comando» y muchos necesitan una dirección de 4 dígitos HEX.
La entrada de HEX se realiza con el sistema de desplazamiento a la izquierda; pulsa ENTER para introducir la dirección. Cualquier entrada puede interrumpirse pulsando Q antes de ENTER.

COMANDOS

A Ensambla el código fuente. Pulsa ENTER después de aparecer A. El programa queda ensamblado al volver al «modo de comando».

C Copia un bloque de memoria de un sitio a otro

COPY BLOCK:	FIRST ADDR	Dirección de comienzo del bloque que se va a copiar.
	LAST ADDR	Ultima dirección del bloque.
	TO ADDR	Dirección del destino.

D Saca por impresora un bloque de memoria en hexadecimal.

DUMP FROM ADDR	Dirección de comienzo
TO ADDR	Dirección final Se imprimirán 6 bytes por línea.

E Para pasar al EDITOR.

F Búsqueda de un entero de 2 bytes.

FIND	Número de 2 bytes que se quiere buscar.
FROM ADDR	Dirección en que se debe iniciar la búsqueda.

Busca por toda la memoria excepto en el stack (ignora 32 bytes desde el stack pointer o puntero del stack).
Muestra en pantalla la dirección en que el número aparece por primera vez. Si ahora pulsas M, va directamente al «modo de edición de memoria» en esta dirección. Pulsa Q para volver al «modo de comando». Si no se encuentra el número se presentará un mensaje «NOT FOUND». El número que se está buscando es el mismo que el que se usó la última vez, y la dirección en que se comienza la búsqueda será la última dirección en que se encontró el número incrementada en una unidad. Así, si quieres buscar, la siguiente vez que aparezca tu número pulsa simplemente F ENTER ENTER. Si quieres cambiarlo, escribe los números nuevos.

I Inspección y/o modificación de registros.
Presenta los registros BC, DE, HL, AF y IX. F también aparece en binario con una etiqueta sobre cada bandera (S = Signo, Z = Cero, H = medio acarreo, O = paridad, I = desbordamiento, N = negativo o menos, C = acarreo).
Cuando ejecutes un programa en código máquina, los valores que se presentan en pantalla se almacenan en los registros y al volver, los valores de los registros se almacenan para este comando. El número valor hexadecimal que escribas se almacena en el registro al que apunte el cursor. Para introducir un número y mover el cursor pulsa ENTER.

L LOAD carga desde el cassette un bloque de memoria.
LOAD FRONT ADDR primera dirección.
TO ADDR última dirección.
Solamente puedes cargar un bloque de memoria que se haya almacenado con cinta mediante el comando SAVE de este ensamblador.

M Modo de edición de memoria. Puedes introducir y editar Código máquina directamente. En el borde izquierdo de la pantalla hay una columna de 24 direcciones de dos bytes. y junto a ésta se encuentra otra columna con el contenido de estas direcciones. (1—byte). El cursor señala la dirección cuyo contenido puedes cambiar (la dirección del cursor) para hacer esto, escribe simplemente un número en hexadecimal y pulsa ENTER. Si pulsas Q antes de ENTER se interrumpirá la entrada del número. Los comandos disponibles en este modo son:

- J Calcula el desplazamiento en los saltos relativos: introduce los dos últimos dígitos de la dirección de destino, pulsa ENTER y el desplazamiento se situará en la dirección del cursor.
- L Mueve el cursor continuamente hacia abajo en la memoria (es equivalente a pulsar ENTER repetidamente)

ENTER mueve el cursor un solo paso hacia abajo.

- O Mueve el cursor continuamente hacia arriba (esto es: P repetida).
- P Mueve el cursor un solo paso hacia arriba.
- R Repite la entrada de un valor: escribe tu valor y luego pulsa ENTER. El valor seleccionado se almacenará en todas las direcciones que señale el cursor. Se puede usar cualquiera de los comandos de movimiento del cursor. Para abandonar este modo pulsa Q.

P Llena un bloque de memoria con números de 1 byte.
PUT dato con que se quiere llenar el bloque
FROM ADDR Dirección de comienzo
TO ADDR Ultima dirección

Q Quit — vuelta al BASIC — púlsala dos veces.

R Ejecuta un programa en código maquina desde la dirección 5CD2 o la dirección elegida. La vuelta al ensamblador se produce por una instrucción de salto que se

ensambla automáticamente al final del programa. La dirección a la que se efectúa el salto es EA90. Se borra la pantalla (situándose el cursor en la esquina superior izquierda) antes de ejecutar el programa, y al volver, el ensamblador espera a que se pulse una tecla antes de borrar la pantalla de nuevo y volver al «modo de comando». Se producirá un mensaje de error en el stack si al volver el puntero del stack está en una posición diferente a la que se encontraba antes de ejecutar el programa.

S SAVE Almacena un bloque de memoria en cassette.
SAVE FROM ADDR dirección de comienzo.
TO ADDR última dirección.
Un bloque de memoria que se ha almacenado con este comando no se puede volver a cargar en el ordenador desde BASIC.

NOTA: Pulsando Q (o SHIFT Q en el EDITOR) se interrumpirá siempre lo que estés haciendo, siempre y cuando el ensamblador esté esperando que pulses una tecla.

SUBRUTINAS UTILES DEL ENSAMBLADOR

El ensamblador contiene muchas subrutinas que puedes usar en tus programas. A continuación aparece una lista de las más útiles.
El contenido de todos los registros (a excepción de F) están protegidos a no ser que se indique lo contrario.

DISPLAY CHAR — FCF2 Imprime en la pantalla el carácter contenido en A e incrementa la posición del cursor en 1 (dirección FFOO). Se pueden usar también los caracteres de control del cursor (08—OB).
Si A = 0, se borra la pantalla.
Si A= 1, se sitúa el cursor en la posición de comienzo (esquina superior izquierda)

DISPLAY TEXT — FCE1 Imprime en pantalla el texto que sigue a la instrucción CALL de llamada a esta subrutina. El final del texto debe señalizarse con un FF.

```
Ejemplo:    CALL FCEI
            12 1            modo FLASH
            "HI — "
            12 0            desconecta FLASH
            102 114         tinta roja, papel verde
            16 10 12        AT 16,18 en decimal
            "—THERE" FF
```

Los colores, FLASH, etc., se establecen al retornar al ensamblador.

Los códigos para INK, PAPER, etc., se encuentran en la página 183 del manual SINCLAIR.

SCROLL DOWN FE23 Posición del primer carácter a efectuar el SCROLL (0—2FFh) en HL. Tiene que estar al principio de una línea.
Ejemplo: 0h, 20h, 40h, 50h... provoca un fallo.

SCROLL UP — E72A La posición del primer carácter en BC. Se deben aplicar las mismas restricciones que en la instrucción anterior.

1 byte HEX NUMBER — FCBE imprime el contenido del registro A en hexadecimal.

2 byte HEX NUMBER ECB3 imprime el contenido de HL en hexadecimal. El contenido del registro A se pierde.

KEYBOARD — E086 deja en el registro A el código del carácter de la tecla que se está pulsando.
A = FF cuando no se pulsa ninguna tecla.

WAIT FOR KEY — FC6E Espera a que se pulse una tecla y deja su código en el registro A.

1 HEX — BYTE IN — FC20 Presenta el valor del registro A.
La bandera CARRY se pondrá a 1 si se pulsa ENTER y a 0 si se pulsa Q.

2 HEX — BYTES IN — EF28 presenta el valor de HL.
La bandera CARRY se modificará como arriba se ha indicado.

CREATE REM — E7F2 crea una línea REM en BASIC.

BC — número de línea.

DE — máxima dirección sobre la que se puede escribir. El acarreo toma valor 1 y no se crea la REM si se sobrescribe esta dirección.

HL — longitud sin incluir EAOD.

FIND LINE — E860 busca la dirección de una línea en BASIC.
Entrada: número de línea en BC.
Salida: dirección en HL.

DELETE LINE — E6F5 borra una línea de BASIC.
Entrada: número de línea en BC

COPY BLOCK — EEFD copia un bloque de memoria como el comando copy del MONITOR.

BC dirección de comienzo del bloque.

DE dirección final.

HL nueva dirección de comienzo.

EA90 vuelve al monitor desde un programa en c/m.

5CD2 primera dirección del código objeto en una línea 1 REM.

CONSEJOS PARA EL PRINCIPIANTE

El código máquina no es un arte misterioso reservado a extraños seres con electrones circulando por sus venas. Es simplemente otra forma de programar. como el BASIC. Tienes que ser más cuidadoso que en BASIC, ya que no hay mensajes de error cuando se ejecute el programa y se produzca algún fallo. Si cometes errores de bulto, la consecuencia más común es que se bloquee el ordenador.
Sin embargo, usando el ensamblador y el monitor. y siguiendo unas reglas simples, pronto estarás en condiciones de escribir, depurar, y ejecutar con éxito tus programas en código máquina.

REGLAS

1. Guarda todos los registros dentro de las subrutinas en el STACK con PUSH. Antes de volver de la subrutina, recupéralos en orden inverso.
   Por ejemplo: Subrutina DISP para llenar la pantalla de δ.

```
DISP PUSH BC PUSH AF; guarda los registros.
     LD BC, + 768; ajusta el contador.
LOOP CALL FCE1" δ " FF; imprime δ.
     DEC BC; decrementa contador.
     LD A, B; chequea si es cero.
     OR C
     JR NZ LOOP; si no es así, salta a LOOP.
     POP AF POP BC; recupera registros.
     RET; return.
```

Recuerda: Cuando trabajes con el stack, el primer dato que recoges es el último que has cargado.

2. Todo lo que has introducido en el stack en una subrutina debe volverse a recoger en la misma subrutina antes de llegar a una instrucción RET. Esto es asi porque la instrucción CALL coloca en el stack la dirección de retorno, y asi, si modificas el stack después de una CALL y no se vuelve a ajustar, al encontrar el programa una instrucción RET saltará a un sitio incorrecto, y seguramente se bloqueará el ordenador.
   Supón que FRED es una subrutina para imprimir un *
   FRED PUSH AF;AF al stack
   LD A, "* ": se almacena en A el valor
   CALL FCF2; imprime el carácter
   RET; return ?
   Esta rutina no funcionará, ya que debería haber un POP AF antes de RET.

Para comprobar la subrutina. puede usarse JP EA90 en vez de RET (salto al MONITOR), dando un mensaje de error de stack.
Comprueba todas las subrutinas de esta forma para evitar sorpresas desagradables.
NOTA: Tendrás que usar un JP FRED al principio de tu código fuente para saltar por encima de cualquier programa que pudiera haber entre medias.

3. Recuerda donde está tu stack. No pongas una rutina de código máquina demasiado cerca del stack, ya que el stack podría comerse tu rutina al engordar.
4. Los saltos relativos tienen una limitación de rango.
   + 127 bytes hacia delante.
   —128 bytes hacia atrás.
   Solamente te enfrentarás a este problema si estas programando directamente en hexadecimal; usando el ensamblador se producirá un error de salto relativo.
5. Lee toda la información sobre programación en código máquina que puedas.

Escribe tus programas con subrutinas. a no ser que tengas una razón muy buena para no hacerlo así, y luego compruébalas por separado.

LIBROS RECOMENDADOS

Machine language programming made simple for your Sinclair.
Understanding your ZX81 ROM *Melbourne House.*
Programming the Z80 ZAKS SYBEX
Z80 Introduction Handbook.

PROGRAMAS EN CODIGO MAQUINA

1. Este programa imprime mensajes con letras grandes.

```
            JR START
MES         "Hola, esto es tu"      ; mensaje
            "SPECTRUM" FF           ; termina con FF
TMP         00
START       LD DE, MES              ; DE es el puntero de caracteres
NXTC        LD A, (DE)              ; toma carácter
            CP FF                   ; se ha acabado ?
            JRZ END
            SUB 20
            LD H, 0 LD L, A         ; multiplica por 8
            ADD HL, HL              ; para dar el desplazamiento
            ADD HL, HL              ; en la tabla de caracteres
```

```
            ADD HL, HL          ;
            LD B, 8             ; bucle para carácter
NXT L       PUSH BC             ; guarda el contador
            LD A, (HL)          ; calcula la línea en este carácter
            LD (TMP), A         ; guárdalo en TMP
            LD B, 8             ; número del bit en la línea
NXT B       LD A, (TMP)
            RLA
            LD (TMP), A         ; comprueba bit
            LD A, (D)
            JR C DISP           ; si está a 1 vete a DISP
            3E 20               ; si no es así, pinta un espacio
DISP        CALL FCF2           ; imprime carácter
            DJNZ NXT B          ; siguiente bit
            CALL FCEI D FF      ; imprime Nenuline
            INC HL
            POP BC
            DJNZ NXT L          ; siguiente línea de caracteres
            INC DE
            JR NXT C            ; siguiente carácter
END
```

ESTE PROGRAMA ATERRIZA SUAVEMENTE UNA NAVE ESPACIAL

```
            LD DE, + 700
LOOP        CALL FCEI " " 90 91     ; imprime la nave
            8 8 FF
            DEC DE
            LD BC, 740
            DEC BC                  ; delay
            LD A, B
            OR C
            JR NZ —5
            LD A, D                 ; contador = 0 ?
            OR E
            JR NZ LOOP              ; si no es así, vete a LOOP
            CALL FCEI 12 I 13 I
            D "* * ATERRIZASTE **" FF
```

También debes introducir los siguientes datos en memoria en «modo de edición de memoria». Estos serán los gráficos definidos por el usuario a y b.

```
            FF58 03 C3 F3 FF FF F3
            FF5E C3 03 F8 FC FE FF
            FF64 FF FE FC F8
```