indescomp

# INSTRUCCIONES

# INSTRUCCIONES GENERALES PARA CARGA DE PROGRAMAS

1.º Rebobinar el casette hasta el principio antes de proceder a la carga del programa. Presione **PLAY,** y escuchando la cinta colóquela en la zona de silencio, y pare el casette.

2.º Posicione el nivel de volumen de su magnetófono a 3/4 del máximo y el control de tono de agudos al máximo.

3.º Teclee **LOAD " "**.

4.º Presione **PLAY** en su magnetofón y pulse **NEW LINE.**

IMPORTANTE: En caso de dificultad en la carga del programa, consulte el capítulo 16 del manual de su ZX 81.

## INSTRUCCIONES DEL ENSAMBLADOR DEL CODIGO MAQUINA

Este ensamblador ocupa 5K de RAM, y se coloca en la parte alta de la memoria, entre la posición 27648 y la 32767. La variable del sistema RAMTOP se ajusta automáticamente para evitar el borrado del ensamblador por algún programa Basic, o con el comando NEW.
Para utilizar el ensamblador, simplemente cargarlo de la cinta usando el nombre ZXAS, despúes RUN. Esto transfiere el ensamblador a la parte alta de la memoria. Se pueden ahora borrar las líneas 10, 20 y 30, y el ensamblador está preparado para ser utilizado.
Los nemónicos standard de ZILOG son interpretados por el ensamblador (con la única excepción del uso de puntos en lugar de comas).
Las instrucciones son introducidas dentro del programa Basic en sentencias REM, admitiendo más de una instrucción por línea si están separadas por punto y coma.
El listado en lenguaje ensamblador ya completo se coloca entre parentesis, que se colocan también en sentencias REM. Se pueden colocar comentarios en el listado, escribiendo: *, seguido por el comentario.
Cada vez que una instrucción es procesada por el ensamblador, aparecen varias combinaciones de números y letras en la pantalla, un ejemplo típico es:

```
100     408A     76     HALT
```

Esto simplemente significa que la instrucción HALT está en la línea 100 del listado, el código máquina para HALT es 76 (hexadecimal), y que la instrucción fue ensamblada en la posición 408A (hexadecimal).
Esta información va apareciendo en la pantalla, y cuando está llena, tienes que pulsar una tecla para que el ensamblador continue (pulsando BREAK en cualquier momento terminará el programa). El ensamblador acabará con un mensaje de error en la parte baja de la pantalla; teniendo los siguientes significados:

Error 0 - Sin equivocaciones
Error 1 - No encontró: ''(''
Error 2 - No encontró: '')''
Error 3 - Etiqueta ilegal
Error 4 - Instrucción ilegal
Error 5 - Número fuera del rango
Error 6 - Salto relativo fuera del rango

Para ensamblar el código máquina, haz simplemente GOTO 9000, y te preguntará la dirección de comienzo del código máquina (mira más adelante los valores típicos). Puesto que no todas las instrucciones pueden ser ensambladas inmediatamente, por ejemplo: saltos a una etiqueta colocada más adelante, el ensamblador procesará cada instrucción dos veces.
Los números se suponen decimales,a menos que estén precedidos por $, que se toman como hexadecimales. Asi las dos instrucciones ADD A.34 y ADD A.$ 22 son equivalentes. Se permiten hasta 256 etiquetas, indicadas por ''L'' seguido por el número de la etiqueta.
Por tanto se permiten etiquetas en el rango :L0 a :L255. Estas etiquetas son muy útiles para saltos y llamadas a subrutinas, y como almacén temporal de datos. Este último uso puede verse en el siguiente ejemplo que uti-

liza L30 como una variable.

```
LD A.2
LD (L30). A
RET
:L30NOP
```

La etiqueta 30 se ha usado para definir la posición inicialmente, y debe cargarse con una instrucción inútil. En vez de usar etiquetas con saltos relativos, puede usarse el byte de desplazamiento, con el signo apropiado + ó -, así, estas dos combinaciones de instrucciones son equivalentes:

```
i) JR Z. + 1          ii) JR Z.L5
   INC HL                 INC HL
   LD (HL). A             :L5LD (HL).A
```

Como ejemplo de una rutina, escribe las siguientes instrucciones, que producen un retardo de hasta 25,6 segundos en pasos de 0,1 segundos (el retardo está determinado por el valor con que está inicializado B, dentro de la línea 20).

```
10 REM (
20 REM LD B.0 ; LD DE.$FFFF
30 REM :L1LD HL.14130;: L2ADD HL.DE
40 REM JR C.L2; DJNZ.L1; RET
50 REM )
```

Al final del código máquina debes retornar a Basic pues mientras se ejecuta una rutina máquina no se atiende a la tecla BREAK. Las siguientes instrucciones realizarán un lazo hasta que pulses la tecla BREAK, en que vuelves a Basic, fíjate que usa una rutina de la ROM del ZX-81.

```
10 REM (
20 REM CALL $F46; JR C.-5;RET
30 REM )
```

Los programas en Código Máquina generalmente tienen como resultado la pérdida del programa, por tanto es recomendable grabar el programa antes de ejecutarlo. (Para ejecutarlo haz simplemente una llamada mediante una USR al principio de la rutina).
El contenido de memoria que se graba al hacer SAVE, es todo el contenido hasta la dirección almacenada en la variable E-LINE, por tanto, a menos que el código esté ensamblado en parte del programa, no se grabará. El listado contenido en REM, no obstante se graba. Si quieres grabar el código máquina, sigue el siguiente procedimiento. Escribe en la primera línea de programa una REM bastante larga, como para acomodar los códigos ensamblados. Ensambla el programa máquina a partir de la posición 16514 (es la dirección del primer caracter de la REM), y borra todo el resto. El programa se puede grabar (el ensamblador no es grabado por el comando SAVE), con el código máquina dentro del programa. Si no necesitas grabar el código ensamblado, entonces puede ser ensamblado en una zona vacia de memoria.

Su comienzo viene indicado por STKEND (pag. 171 del manual de ZX-81). Debes recordar que si posteriormente a desconectar el ZX-81, quieres añadir instrucciones máquina, a tu programa escrito en sentencias REM, el ensamblador debe estar cargado en el

microordenador, por ello haz las siguientes operaciones:

Carga el ''ZXAS'', haz ''RUN'', haz ''NEW'', y a continuación carga en el ZX-81 el programa con las instrucciones máquina colocadas en REM, y añade o quita las instrucciones que quieras.

Desafortunadamente problemas de espacio no permiten comentarios detallados de las instrucciones del Z-80, recomendamos al usuario que consulte libros tal como:
Libro de instrucciones del Z−80, o cualquier texto sobre programación del Z-80. Esto es solo una descripción de como usar ZXAS, y no es un curso de programación en código máquina.

**Los nemónicos**

Se usan varias abreviaturas en la siguiente lista de instrucciones:

r ..... Es cualquiera de los registros: A,B, C,D,E,H, ó L.
dis ..... Es un desplazamiento de un Byte dentro del rango - 128 a + 127
data ..... Es un dato inmediato, sea de uno o dos Bytes.
mem ..... Es: (HL), (IX + dis). ó (IY + dis).
address ..... Dirección.

Las tres primeras letras de la mayoría de las instrucciones son importantes, debes por tanto, procurar que sean correctas, **especialmente los espacios**.

ADC A.mem
ADC HL.DE
ADD A.r
ADD HL.HL
ADD IX.IX
ADD IY.IY
AND data
BIT 1.r
BIT 3.r
BIT 5.r
BIT 7.r
CALLNC.address
CALLPO.address
CP r
CPI
DEC mem
DEC HL
DI
EX (SP).IX
EXX
IM2
INC r

ADC A.r
ADC HL.HL
ADD A.data
ADD HL.SP
ADD IX.SP
ADD IY.SP
BIT 0.mem
BIT 2.mem
BIT 4.mem
BIT 6.mem
CALL address
CALLNZ.address
CALLZ.address
CP data
CPIR
DEC r
DEC IX
DJNZ.dis
EX (SP).IY
HALT
IN r(C)
INC BC

ADC A.data
ADC HL.SP
ADD HL.BC
ADD IX.BC
ADD IY.BC
AND mem
BIT 0.r
BIT 2.r
BIT 4.r
BIT 6.r
CALLC.address
CALLP.address
CCF
CPD
CPL
DEC BC
DEC IY
EI
EX AF.AF
IM0
IN A.port
INC DE

ADC HL.BC
ADD A.mem
ADD HL.DE
ADD IX.DE
ADD IY.DE
AND r
BIT 1.mem
BIT 3.mem
BIT 5.mem
BIT 7.mem
CALLM.address
CALLPE.address
CP mem
CPDR
DAA
DEC DE
DEC SP
EX (SP).HL
EX DE.HL
IM1
INC mem
INC HL

| | | | |
|---|---|---|---|
| INC IX | INC IY | INC SP | IND |
| INDR | INI | INIR | JP (HL) |
| JP (IX) | JP (IY) | JP address | JP C.address |
| JP M.address | JP NC.address | JP NZ.address | JP P.address |
| JP PE.address | JP PO.address | JP Z.address | JR dis |
| JR C.dis | JR NC.dis | JR NZ.dis | JR Z.dis |
| LD (address) .A | LD (address) .BC | LD (address) .DE | LD (address) .HL |
| LD (address) . IX | LD (address) .IY | LD (address) .SP | LD (BC) .A |
| LD (DE) .A | LD mem.r | LD mem.data | LD A.(address) |
| LD A.(BC) | LD A.(DE) | LD r.r | LD r.data |
| LD r.mem | LD A.I | LD A.R | LD BC.(address) |
| LD BC.data | LD DE.(address) | LD DE.data | LD HL.(address) |
| LD HL.data | LD I.A | LD IX(address) | LD IX.data |
| LD IY.(address) | LD IY.data | LD R.A | LD SP.(address) |
| LD SP.data | LD SP.HL | LD SP.IX | LD SP.IY |
| LDD | LDDR | LDI | LDIR |
| NEG | NOP | OR mem | OR r |
| OR data | OTDR | OTIR | OUT (C) .r |
| OUT port.A | OUTD | OUTI | POP AF |
| POP BC | POP DE | POP HL | POP IX |
| POP IY | PUSH AF | PUSH BC | PUSH DE |
| PUSH HL | PUSH IX | PUSH IY | RES 0.mem |

| | | | |
|---|---|---|---|
| RES 0.r | RES 1.mem | RES 1.r | RES 2.mem |
| RES 2.r | RES 3.mem | RES 3.r | RES 4.mem |
| RES 4.r | RES 5.mem | RES 5.r | RES 6.mem |
| RES 6.r | RES 7.mem | RES 7.r | RET |
| RET C | RET M | RET NC | RET NZ |
| RET P | RET PE | RET PO | RET Z |
| RETI | RETN | RL mem | RL r |
| RLA | RLC mem | RLC r | RLCA |
| RLD | RR mem | RR r | RRA |
| RRC mem | RRC r | RRCA | RRD |
| RST 00 | RST 08 | RST 10 | RST 18 |
| RST 20 | RST 28 | RST 30 | RST 38 |
| SBC A.mem | SBC A.r | SBC A.data | SBC HL.BC |
| SBC HL.DE | SBC HL.HL | SBC HL.SP | SCF |
| SET 0.mem | SET 0.r | SET 1.mem | SET 1.r |
| SET 2.mem | SET 2.r | SET 3.mem | SET 3.r |
| SET 4.mem | SET 4.r | SET 5.mem | SET 5.r |
| SET 6.mem | SET 6.r | SET 7.mem | SET 7.r |
| SLA mem | SLA r | SRA mem | SRA r |
| SRL mem | SRL r | SUB mem | SUB r |
| SUB data | XOR mem | XOR r | XOR data |

PROGRAMA 2.°

## DESENSAMBLADOR Y DEPURADOR DEL CODIGO MAQUINA

ZXDB ocupa los primeros 4K de memoria y se almacena en una sentencia REM en la línea 1. Para usar el ZXDB cargarlo con LOAD ''ZXDB'' y RUN.
Aparecerá un puntero (*) que indica que el ZXDB está esperando una letra (comando).

NOTA: Para usar ZXDB y ZXAS juntos el procedimiento es:
a) LOAD ''ZXAS''
b) RUN
c) NEW
d) LOAD ''ZXDB''

Tendrás ahora el ZXDB en los 4K de abajo de la memoria y el ZXAS en los 5K de arriba.
Si no usas ambos juntos entonces todas las líneas siguientes a la 10 se pueden quitar.

¿DONDE COLOCAS TU CODIGO MAQUINA?:
Introduciendo el desensamblador, hay una zona de memoria libre entre 50C0 - 52FF (hexadecimal) que es el mejor lugar para almacenarlo hasta que esté depurado. Puedes entonces transferirlo hasta una REM aparte y después borrar el ZXDB, dejando más espacio para tus programas BASIC.

Nota: Todas las entradas a (y salidas desde) el ZXDB se hacen en hexadecimal del que solo los últimos cuatro caracteres son válidos. Pulsando EDIT durante cualquier entrada volverás al puntero (*).

LISTA DE COMANDOS:
Por cada comando, se asigna una letra que debe ser escrita al aparecer el puntero. En estas instrucciones verás que el símbolo ( = ó>ó combinaciones) es generado por el ZXDB esperando tu respuesta.

Ejemplo: = dirección de comienzo: significa que un caracter = aparece en la pantalla, al que tu deberás responder escribiendo una dirección.

La mejor forma para aprender como se usa el ZXDB es cargarlo y experimentar con cada comando.

Q-Abandona el ZXDB y vuelve a BASIC.
G-Ejecuta un programa. = dirección de comienzo.
JP 4100 al final de un programa devuelve al ZXDB, mientras que JP 410 devuelve al BASIC.
V-Contenido de la memoria en hexadecimal. = dirección de comienzo.

El ZXDB te muestra una línea de 8 Bytes de memoria. Pulsa New Line para ver la siguiente línea. Pulsando el 0 retornas al puntero (*).

A-Muestra la memoria como caracteres. = dirección de comienzo.

Una línea de 32 Bytes de memoria se muestran como caracteres. Los códigos erróneos son mostrados como espacios. Pulsa New Line para ver la siguiente línea y Q para retornar al puntero ( = ), para escribir una nueva dirección, y Q otra vez retorna al puntero (*).

W-Hace una muestra de una porción (ventana) de la memoria.
= número de líneas a usar. Indica el número de líneas usadas por ZXDB (contando desde abajo) para su muestra en pantalla.
Este comando tiene interés en la ejecución paso a paso que se verá más adelante.

F-Llena un bloque de memoria.
>Valor a ser colocado en el bloque.
= Comienzo del bloque.
= Fin del bloque.
Llena el bloque definido con el valor deseado.

E-Cargador hexadecimal.
= Dirección de comienzo. Este es un cargador hexadecimal en el que el valor anterior es mostrado. Te pide un nuevo valor.
Solo las entradas acabadas en New Line cambiarán el valor.

C-Compara dos bloques de memoria.
= Comienzo del primer bloque.
= Fin del primer bloque.
= Comienzo del segundo bloque. ZXDB muestra la diferencia entre dos bloques de memoria. En cada pausa New Line continuará el proceso. EDIT devuelve al puntero (*).

M-Movimiento de un bloque de memoria.
= Comienzo del bloque de memoria.
= Final del bloque.
= Comienzo del bloque destino. Mueve un bloque de memoria a una nueva localización. Moviendo el bloque de memoria hacia abajo, acaba la última entrada con I. En caso contrario acaba con New Line.

D-Desensamblaje de un bloque de memoria.
= Dirección de comienzo. La memoria es desensamblada por el ZXDB usando nemónicos standard de ZILOG con las siguientes excepciones (que son principalmente las del INTEL 8080).

| ZILOG | ZXDB |
|---|---|
| LD B, (IX + 10) | LD B, IX + 10 |
| LD A, (1234) | LDA 1234 |
| LD (1234), A | STA 1234 |
| LD A, (DE) | LDAX DE |
| LD (DE), A | STAX DE |
| LD HL, (1234) | LHLD 1234 |
| LD(1234),HL | SHLD 1234 |
| LD SP, HL | SPHL |
| EX DE, HL | XCHG |
| EX AF, A'F' | EX |
| JP (HL) | PCHL |
| EX (SP), HL | XTHL |
| IN A, (FE) | IN FE |
| OUT (FE), A | OUT FE |
| OUT (C), A | COUT A |
| IN A, (C) | CIN A |

Nota: (HL) siempre se llama M.

Lo anterior es aplicable a otros registros. Es decir: LD IX, (1234) aquí es: LIXD 1234.

Escribe una nueva dirección seguido de New Line

para mover el desensamblador a este sitio. Escribe Q para retornar al puntero (*). Escribiendo cualquier otra cosa desensambla la siguiente instrucción.

S-Busqueda en memoria de una **entrada** de caracteres en forma de cadena. Después de escribir S, ZXDB espera la introducción de la cadena que va a ser buscada, que es introducida como una cadena de valores hexadecimales separados por puntos y terminada por New Line. El último caracter antes de New Line debe ser un punto.
SHIFT detiene una búsqueda en la que no encuentra la cadena.
Si utilizas cuatro dígitos hexadecimales, entonces los dos primeros se tomarán como máscara para los dos segundos. El hecho de encontrar la cadena de memoria se define como:
( (MEM) XOR (**INPUT**) ) AND NOT-MASK = 0
En efecto, si el bit de máscara es uno, entonces el bit de la **entrada** correspondiente no es necesario que se encuentre en la posición de memoria.

Ejemplo: Escribiendo:
CD.FF00.1F00. (New Line)
= 0000 (New Line)
busca en memoria desde el primer CALL (Código CD) de la ROM.

Escribiendo C en cada paso te permite cambiar la cadena que hay en memoria por una nueva cadena, introduciendo la nueva cadena de la misma forma que la anterior, y puede ser más larga o más corta que la original.

B-Coloca un Breakpoint (punto de parada) y ejecuta el programa.
= Dirección del Breakpoint.
= Dirección de comienzo de ejecución del programa.
Este comando es el más importante. Se coloca un Breakpoint en el programa de usuario, de forma que cuando se alcanza, para el programa, mostrando el contenido de todos los registros, el estado de las banderas, los nemónicos de las instrucciones en ejecución, la siguiente instrucción, y una ventana de memoria definida por el usuario (8 Bytes).
El programa puede continuarse paso a paso, de forma que las instrucciones se ejecuten de una en una, mostrandose el contenido de todos los registros.
Esta potente herramienta permite al usuario ver exactamente que está haciendo su programa (y si se ve que va mal, pararlo antes de que se rompa).
El formato de muestra es como sigue:

| | |
|---|---|
| PC SP IX IY | Banderas (acarreo, cero...) |
| (PC) (SP) (IX) (IY) | Nemónico de la instrucción en ejecución |
| AF BC DE HL | |
| (AF) (BC) (DE) (HL) | Nemónico de la siguiente instrucción. |
| Ventana — | Ventana de memoria de 8 Bytes. |

Lo siguiente es una línea de comandos específicos paso a paso (completamente aparte de los principales comandos ZXDB).
El comando se introduce como una simple letra, o número y letra cuando se requiera. New Line ejecuta el siguiente paso.

G Hace que el programa corra normalmente.

nW Coloca la ventana de memoria en la dirección n (HEX).

R Usado correctamente después de un CALL, ejecuta la subrutina y vuelve al modo paso a paso después de retornar de la subrutina.

nP Coloca el Breakpoint en n (HEX).

Q Retorno al puntero (*).

L Coloca el Breakpoint en la instrucción presente (solo de tres bytes) y permite que el programa funcione. Es útil en bucles.

O Muestra solo los nemónicos de la instrucción.

nN Ejecuta las siguientes n instrucciones en el modo paso a paso.

Z Permite colocar datos en los registros por simple escritura del registro, seguido del dato.
Ejemplo: AFF pone en A el valor FF.
Cualquiera de los registros (A,B,C,D,E,F,H,L ó M) (M es (HL)) pueden ser modificados.
Escriba NewLine para volver al paso a paso.

nT El programa solo retornará al paso a paso cuando un registro de 16 BITS contenga n o se hace una referencia a la dirección de memoria n.