EL FORTH DE SPECTRUM

COMO COLOCAR EL FORTH EN MEMORIA
Cargar el compilador de Forth tecleando: LOAD " " CODE.
Parar el cassette cuando haya sido cargado.
El Editor Forth es lo siguiente en la cinta Para cargarlo, teclee esto:1 LOAD. Presione a continuación la tecla <ENTER>. La computadora responderá con el mensaje: READY CASSETTE. Presionando ENTER arrancará la rutina de carga del cassette
Ponga la cinta en marcha y presione ENTER. La primera pantalla tarda un par de segundos en cargar. Pare su aparato cuando desaparezcan de la pantalla las características franjas azules y amarillas. Espere a que aparezca de nuevo READY CASSETTE y entonces ponga su cassette en marcha y presione ENTER. Hay un total de 3 pantallas a cargar.
Si el programa no se carga de manera adecuada, retroceda la cinta hasta el principio de la pantalla.
Después de que se haya cargado la tercera pantalla, el mensaje 'OK' aparecerá. No se alarme si aparece el mensaje de error MSG#4.
Su Editor está cargado ya.
Ahora empiece a programar en Forth.
DISFRUTE PROGRAMANDO Y BUENA SUERTE

INDICE

EL FORTH DE SPECTRUM

1.0 OPERACIONES BASICAS

El modo más fácil do aprender FORTH es usarlo. Puesto que el Forth es un lenguaje interactivo, puedo usted sentarse y experimentar con él. En este manual del usuario hay muchos ejemplos para ilustrar las capacidades del Forth. Le sugerimos que los pruebe usted mismo

1.1 EMPEZANDO

El Forth se anunciará a sí mismo y le dirá de cuánta memoria dispone. El carácter gráfico  es su cursor y aparecerá cuando el sistema esté preparado para entradas desde el teclado. En este momento todo estará listo para que usted teclee un comando seguido de un <ENTER>. Hasta que no haya pulsado la tecla ENTER puede usted cambiar sus comandos usando la tecla DELETE para borrar cualquier carácter que no desee, presionándola una vez para borrar un carácter cualquiera y escribiendo entonces los que desee que constituyan la línea.
El comando más sencillo que puede usted dar al Spectrum-Forth es una línea vacía. Si presiona usted la tecla ENTER sin más, su Spectrum-Forth deberá responder con OK. Puesto que ha visto que no hay nada que hacer, ha terminado la línea y está esperando a que se teclee otro comando. Debe usted probar esto. Así podrá darse cuenta de que su Forth está vivo y esperando sus instrucciones.

1.1.1 EL EDITOR DE PANTALLA

Al Forth de Spectrum se le ha dotado de un editor de pantalla para ayudarle a usted a redefinir cualquier palabra con la que pueda haber cometido un error.
El Editor le proporciona un cursor de copia que se mueve con las teclas de movimiento de cursor (CAPS SHIFT 5 - 8) El Editor entra en funcionamiento presionando cualquiera de las teclas de movimiento de cursor. Coloque este cursor en la línea quo desea editar y presione entonces la tecla EDIT (CAPS SHIFT “1”). Esto hará que el cursor normal copie el carácter y moverá ambos cursores un espacio a la derecha. Todas las teclas están dotadas de un espacio hacia la derecha. Puesto que todas las teclas están dotadas de repetición automática, mantenga la tecla EDIT apretada. De esta manera puede usted copiar una línea completa.

Comandos del Editor

CAPS SHIFT 5	Cursor a la izquierda
CAPS SHIFT 6	Cursor abajo

CAPS SHIFT 7	Cursor arriba
CAPS SHIFT 8	Cursor a la derecha
CAPS SHIFT 1	Copia de un carácter y movimiento de ambos cursores un espacio a la derecha

1.2 PALABRAS

La unidad de comando básico del Forth es llamada una palabra. Una palabra consiste en una cadena de caracteres delimitada por espacios. Las únicas restricciones sobre las palabras son que ninguna palabra puede contener un espacio en un interior, o un ENTER, o un carácter gráfico. La palabra puede tener cualquier longitud, siendo significativos los primeros 31 caracteres, lo que hace que se puedan usar palabras con sentido. Las palabras pueden estar constituidas por letras mayúsculas y minúsculas.

Después de introducir una línea de texto presionando ENTER, EL INTERPRETADOR DE TEXTO analiza la línea, rompiéndola en unidades básicas, palabras, que serán ejecutadas en el orden de entrada. Cada palabra en Forth tiene un nombre (la manera de la que usted se refiere a ella) y una definición (su significado; lo que hace).

Para ejecutar una palabra, el intérprete busca en su Diccionario para determinar la definición de esa palabra. Si la palabra es encontrada, la definición se interpreta y ejecuta. Si no se encuentra, el interpretador intenta convertir la palabra en un entero de 16 bits. Si la palabra no es un número válido en la base actual, un mensaje de error #0 aparece en la pantalla. El sistema vuelve entonces con el cursor para la entrada de nuevas palabras. Este diccionario puede extenderse añadiendo nuevas palabras que se refieran a palabras ya existentes (Ver sección 1.7.)

1.3 NUMEROS

Los números pueden expresarse en cualquier base (desde base 2 a base 36). El sistema empieza trabajando en decimal. De todas formas, puede usted, en cualquier momento, utilizar los comandos (palabras) DECIMAL o HEX o puede definir otra base. Esto establece la base en la que se tratan los números tanto en entradas como en salidas. En general, lo mejor es que se use una sola base a través de todas sus definiciones para evitar confusiones con respecto a la interpretación de los números.

Los números puedan teclearse como enteros positivos o negativos. Se aceptan números positivos sin signo desde el 0 hasta el 65535.

El sistema acepta números enteros con signo (+ o -) (desde el -32768 hasta el 32767). Puede usar también los números de doble precisión que son enteros con signo (desde el - 2147483648 hasta 2147483647). Estos números de 32 bit y doble precisión deben ir precedidos de un punto "**.**".

Puesto que todos los números se almacenan de forma binaria, puede usted beneficiarse de la selección de base para llevar a cabo conversiones de números de decimal a hexadecimal. Por ejemplo, teclee: DECIMAL 258 HEX. <ENTER>, y recibirá usted la respuesta 102 OK(recuerde usted, sin embargo que ahora está en modo HEXADECIMAL). Si va a seguir programando en decimal, cambie de base.

1.4 EL STACK DE PARAMETROS

Todos los programas manipulan datos usando un grupo de parámetros establecido. En Forth, la mayoría de los parámetros se guardan en un stack en el que se EMPUJAN hacia abajo. Este stack se llama "stack de parámetros".

Un stack en el que se empujan hacia abajo los valores, es un tipo de almacenamiento de memoria muy particular, las palabras que quieren acceder a valores almacenados en este stack de parámetros sólo puedan hacerse con los que están más arriba. (El último valor empujado en el stack es el primero en salir).

Para colocar un número en el stack puede usted teclearlo como parte de su comando de entrada. La palabra Forth, "**.**" (un punto), elige el numero que está más arriba en el stack y lo imprime, en la base en la que está funcionando el sistema en ese momento, en le pantalla.
Por ejemplo, para colocar números en el stack 2 4 6 8 <ENTER>
Esta es la apariencia del stack ahora: 8
6
4
2
Si ahora teclea <ENTER>, la salida será 8 OK.
El stack quedará así ahora: 6
4
2
Ahora teclee <ENTER> 6 4 2. El stack estará ahora vacío.
Supongamos que ahora teclea; <ENTER>. La computadora responde: ? MSG # 1 (Error: no hay números que sacar en el stack).
El Forth tiene también otro stack llamado "el Stack de Retorno", que es usado por el interpretador para almacenar direcciones de retorno, Cualquier mensaje de error vacía ambos stackcs.

1.5 ARITMETICA

El Forth tiene un grupo de opearadores aritméticos predefinidos (ver tabla 1). Puesto que el Forth utiliza un stack en el que se empujan los valores hacia abajo y una notación invertida abreviada de parámetros deben estar en el stack antes de que la operación se lleve a cabo. Así pues, para sumar dos números e imprimir al resultado, teclee 5 27 + **.** <ENTER> 32 OK.
Rompiendo esta línea en las unidades que la componen, descubrirá que

5 Empuja el valor 5 al stack.
27 Empuja el valor 27 al stack.
+ Coge-los dos valores de encima del stack, los suma y coloca el resultado en el stack.
. Recoge el valor que más arriba está en el stack (el resultado) y lo imprime: 32 OK.
NOTA: El stack tiene una pérdida neta de 1 valor.

De esta manera ha dejado usted el stack exactamente igual que antes de empezar la suma
El proceso de comparar también puede resultarle extraño. El Forth utiliza las convenciones de la lógica positiva: **El valor de verdad**.

0 Falso
#0 Verdadero

Las palabras de relación del Forth (como <, >, = etc.) deben ser recordadas como escritas entre la segunda entrada de un valor en el stack, en la izquierda, y el valor superior en el stack, en la derecha. Así pues AB< hará la función de A > B y dejará solamente el valor de verdad en el stack. puesto que tanto A como B habrán sido retiradas para realizar la comprobación.

1.6 MANIPULACIONES DEL STACK

Otras operaciones frecuentemente realizadas son clasificadas como manipulaciones del stack, para 1as cuales el Forth de Spectrum proporciona un par de palabras. Estas palabras (descritas en la tabla 3) se usan, generalmente, para mantener orden en el stack cuando éste contiene parámetros. La práctica con estas palabras hará que se hagan útiles para usted rápidamente
Cuando practique, recuerde estas dos reglas elementales 1) Mantenga paridad (todo lo que se meta en el stack hay que sacarlo). 2) Nunca quite del stack más valores de los que ha metido.
Cuando se haga usted con los operadores aritméticos y las palabras manipuladoras del stack, podrá usted crear sus propias palabras. Por ejemplo, para hallar el cuadrado de un número, teclee:

CUADRADO DUP * . : <ENTER> OK

1.7 DEFINICIONES

Una parte de la versatilidad del Forth reside en que éste le permite que usted defina sus propias palabras. Por ejemplo, es posible que tengamos que calcular a menudo el cubo de un número. Es fácil definir una palabra nueva para realizar el trabajo:

: CUBO DUP DUP * * . ; <ENTER> OK

He aquí lo que hace cada componente de esta línea.

: Empieza la definición
CUBO. El nombre de la nueva palabra que va a ser añadida al diccionario
DUP DUP * * Las palabras Forth que definen lo que la nueva palabra va a hacer.
. Imprimir
; Fin de la definición

Después de hacer esta definición, podemos calcular e imprimir un cubo de un número cuando queramos. Por ejemplo:

4 CUBO <ENTER> 64 OK
3 CUBO <ENTER> 27 OK

¿Qué habría sucedido si ustod hubiese usado la palabra CUBO antes de definirla? El Forth no lo habría permitido. Habría imprimido en pantalla CUBO? MSG # (error, palabra indefinida). Por suerte para el programador novato, el Forth tiene un rico vocabulario de palabras predefinidas. Por ejemplo:

? Imprime el contenido de las posiciones de memoria indicadas por el último valor del stack. ? tiene una definición sencilla : : ? @ . ;

Otra combinación sencilla quo está predefinida es 1 + que suma 1 al valor superior del stack. 1 + tiene la definición . 1 + 1 + ;

1.8 MODOS

El interpretador de texto de Forth opera en los modos "Ejecución inmediata" y "Compilación". En el modo ejecución inmediata, cada palabra en una línea entrada es buscada en Diccionario y ejecutada inmediatamente. Durante la compilación, sin embargo, la mayoría de las palabras no son ejecutadas. En su lugar, una referencia de ellas es compilada en el diccionario. La palabra coloca el interpretador en modo de compilación, mientras que lo vuelve a modo de ejecución inmediata.

La forma compilada de la definición consiste en indicadores que señalan las posiciones de memoria de las rutinas que serán ejecutadas por el interpretador interior cuando la definición sea ejecutada. Esta forma de interpretación es extremadamente rápida. Para distinguir entre los modos de ejecución y compilación. pruebe los siguientes ejemplos:

905 <ENTER> 905 OK

Esto se ejecuta inmediatamente (note la interacción).

: MUESTRA 905 . : <ENTER> OK

Esto es compilador. No ocurre nada todavía

MUESTRA <ENTER> 905 OK

Esto ejecuta la rutina compilada y produce el efecto deseado. Para aprender más rápidamente, debe usted practicar con las palabras básicas del Forth y con palabras que usted produzca a partir de sus experimentos. Desarrolla un tipo de notación que le deje una idea de lo que ha hecho (esto le ayudará a evitar cometer los mismos errores dos veces).

EJERCICIOS

1. ¿Cuál es la diferencia entre DUP * DUP * y DUP DUP **?
2. ¿Cuál es la diferencia entre OVER SWAP y SWAP OVER?

TABLA 1

OPERADORES ARITMETICOS

PALABRA	DESCRIPCION	EJEMPLO DEL STACK ANTES	EJEMPLO DEL STACK DESPUES
+	Suma	9 6 2	9 8
-	Resta	9 6 2	9 4
*	Multiplica (con signo)	9 6 2	9 12
/	Divide	9 6 2	9 3
1+	Suma 1	9 6 2	9 6..3
2+	Suma 2	9 6 2	9 6 4
ABS	Deja el valor absoluto	9-6-2	9-6 2
MAX	Deja el mayor de los dos valores superiores del stack	9 6 2	9 6
MIN	Deja el menor de los dos valores superiores del stack	9 6 2	9 2
MINUS	Complementa a 2,5	9 6 2	9 6--2
MOD	Deja el resto de la división entre los 2 valores superiores del stack	9 6 2	9 0
1/	Multiplica el segundo y tercer valor y divide el resultado por el primero	9 6 2	27
2/	Igual que el ejemplo anterior solo que deja en el stack el resto de la división	9 6 2	27 0
+-	El primer número “cede” el signo al segundo	9 6-2	9--6
/MOD	Divide el segundo número por el primero dejando el resto y el cociente	9 6 2	9 0 3
AND	Efectúa un “Y” lógico bit a bit entre los dos primeros números dejando el resultado	9 6 3	9 2
OR	Efectúa un “O” lógico bit a bit entre los dos primeros números dejando el resultado	9 6 3	9 7
XOR	Efectúa un “O” exclusivo bit a bit entre los dos primeros números dejando el resultado	9 6 3	9 5

TABLA 2

OPERADORES DE COMPARACION

PALABRA	DESCRIPCION	ANTES	DESPUES
<	Compara: deja un 1 en el stack si el segundo número es menor que el primero. Si no deja un 0.	9 6 2	9 0
>	Compara: deja un 1 si el segundo es mayor que el primero. Si no, deja un 0.	9 6 2	9 1
0=	Busca un cero. Deja un 1 si el primer número del stack es 0. Si no, deja un 0	9 6 2	9 6 0
0>	Busca un número negativo. Deja un 1 si el primer número del stack es menor que cero. Si no, deja un 0.	9 6 2	9 6 0
=	Compara los primeros dos números para ver si son iguales. Deja un 1 si lo son. Si no, deja un 0.	9 6 2	9 6 0

TABLA III

OPERADORES DE MANIPULACION DEL STACK

PALABRA	DESCRIPCION	ANTES	DESPUES
*	Imprime el número superior en el stack	1 2 3	1 2
DROP	Descarta la última entrada	3 2 1	3 2
DUP	Duplica la última entrada	3 2 1	3 2 1 1
-DUP	Duplica la última entrada si no es cero	3 2 1	3 2 1 1
	es cero	o 3 2 0	3 2 0
OVER	Copia el segundo número en la parte superior del stack	3 2 1	3 2 1 2
ROT	Hace rotar las tres ultimas entradas	4 3 2 1	4 2 1 3
SWAP	Invierte las dos ultimas entradas	3 2 1	3 1 2
.R	Imprime el segundo número en un campo de la anchura del primer número.	3 2 1	3
R	Copia el número colocado en la parte superior del stack de retorno al stack de parámetros	ret. 20 30 20 30 1 2 3	ret. 1 2 3 30

2.0 DECLARACION DE DATOS

El Forth le permite reservar memoria para constantes, variables y cadenas.

2.1 CONSTANTES

Para asignar nombres a constantes se usa la palabra CONSTANT. Esto se hace porque es más fácil referirse a un nombre que a un número o porque el nombre se usa muy a menudo.
Por ejemplo: 5280 CONSTANT PIES/MILLA <ENTER> OK crea la nueva palabra PIES/MILLA y le asigna el valor 5280. Después de haber definido PIES/MILLA puedo usted usarlo como usaría 5280 para colocar ese valor en el stack. Ejemplo: 3 PIES/MILLA * calcula el número de pies en 3 millas.

NOTA: Una vez que un valor he sido definido como una constante, su valor binario es independiente de la base numérica en la que se esté trabajando.

2.2 VARIABLES

La palabra Forth VARIABLE da nombre a una localización cuyo valor es probable que cambie suponiendo que deseamos guardar la puntuación de un juego de marcianitos, entonces podemos declarar una variable de esta manera:

0 VARIABLE PUNTUACION < ENTER> OK
Valor inicial

Cuando usted invoca una variable por su nombre, su posición de memoria es colocada en el stack. La palabra Forth @ remplaza la posición de memoria guardada en el stack por el valor contenido en esa posición de memoria. Por ejemplo, para colocar su puntuación, use: PUNTUACION @ <ENTER> OK.

A veces se necesita examinar el contenido de una variable. La palabra Forth "?" imprime el valor de la variable cuya posición de memoria está guardada en la parte superior del stack. Por ejemplo

PUNTUACION? <ENTER> 0 OK

La palabra "I" se usa para almacenar un valor de 16 bit en una posición de memoria. “I” coge el valor que está en segunda posición en el stack y lo almacena en la posición de memoria almacenada encima del stack Por ejemplo, para poner la puntuación a 100:

100 PUNTUACION I <ENTER> OK

La palabra “+I” suma un valor a una variable. Por ejemplo, para aumentar cien puntos la puntuación

100 PUNTUACION +I <ENTER> OK

NOTA: Puesto que el stack de parámetros es usado para almacenar valores intermedios, la necesidad de variables temporales queda eliminada.

2.3 CAMBIANDO LA BASE NUMERICA

El Forth dispone de una variable que almacena la base numérica con la que está trabajando. Puede usted alterar esta variable y poner cualquier valor entro 2 y 36 para seleccionar otras bases además de la decimal y la hexadecimal. Por ejemplo, suponiendo que usted desee trabajar en binario, puede usted hacer esto de la siguiente manera

2 BASE / <ENTER> OK.

A partir de ahí todos los números que sigan serán impresos en binario. Recuerde que los números que usted teclee deben estar también en binario.

2.4 MATRICES

Las matrices de datos son muy importantes en muchas aplicaciones. Por ejemplo, en vez de tener 10 variables T0, T1, T2, etc. sería mejor usar 10 elementos sucesivos de datos TEMP. A través de operaciones aritméticas adecuadas de direccionamiento, usted puede computar el número adecuado del elemento que necesite en un momento determinado. Esto es más flexible para programar además de ahorrar espacio de diccionario.

Para reservar espacio en el diccionario para cadenas, se usa en Forth la palabra ALLOT. En el caso de arriba se teclearía:

0 VARIABLE TEMP 18ALLOT <ENTER> OK

donde:

0 VARIABLE TEMP define una variable capaz de almacenar un número de 2 bytes llamado TEMP.

18 Pone el número 18 en el stack.

ALLOT Asigna 18 bytes más a la variable TEMP.

Para reservar espacio para una cadena de n elementos coloque el número n en el stack y teclee a continuación: 2 * TEMP + @ <ENTER> OK.

NOTA: Los elementos se numeran del 0 al 9, no del 1 al 10. Valores de n fuera de esta escala darían resultados incontrolables.

Para inicializar el elemento enésimo, teclee: (valor) n 2 * TEMP + I <ENTER> OK.

2.5 OTRAS OPERACIONES EN LA MEMORIA

Existen otras cuatro palabras que puedan ser usadas para manipular contenidos de posiciones de memoria.

1. **CMOVE** Mover la cantidad de bytes especificada por el primer número del stack desde la posición de memoria indicada por el tercer número del stack hasta la posición de memoria especificada por el segundo número del stack los contenidos de la posición de memoria más baja son los primeros en moverse.

Ejemplo: 16396 8000 64 CMOVE <ENTER> OK. (Esto mueve 64 bytes de 16396 a 8000).

2. **FILL** Llena la memoria a partir de la posición de memoria indicada por el tercer número en el stack con el número indicado por el segundo número en el stack de bytos iguales al valor indicado por el primer número en el stack.

Ejemplo: 8000 64 0 FILL <ENTER> OK (Pone 64 bytes de valor 0 desde la posición de memoria 8000 en adelante.)

3. **ERASE** Llena un bloque de memoria con ceros. Es equivalente al 0 FILL (ejemplo superior).

Ejemplo: 8000 64 ERASE <ENTER> OK. (Borra 64 bytes empezando desde la posición de memoria 8000.)

4. **BLANKS** Llena un bloque de memoria con espacios (el código ASCII del espacio es 32). Esta instrucción es equivalente a un 32 FILL.

Ejemplo. 8000 64 BLANKS <ENTER> OK (Pone 64 bytes de valor 32 desde la posición de memoria 8000 en adelante)

TABLA IV
OPERACIONES EN LA MEMORIA

PALABRA	DESCRIPCION	ANTES	DESPUES
@	Coge el contenido de la posición de memoria colocada encima del stack.	100	-236
I	Almacena el segundo número en el stack en la posición de memoria indicada por el número colocado encima del stack	3 20000	vacío
?	Coge e imprime el contenido de la posición de memoria colocada encima del stack	100	vacío
+I	Suma el segundo número del stack al		

	número almacenado en la posición de memoria indicada por el número colocado encima del stack.	701	2000	vacío
C@	Almacena el byte cuya dirección se encuentra en la parte superior del stack en el stack		100	20
CI	Almacena un byte (el segundo número en el stack) en la posición de memoria colocado encima del stack	254	2000	vacío

CMOVE }
FILL }
} Ver 2.4
ERASE }
BLANK }

(Doble precisión) ver glosario.

EJERCICIOS

1. Defina la palabra INTERCAMBIO para que cambie entre sí los valores de dos variables. Esto es: si A y 8 son dos variables, entonces el resultado del comando A B INTERCAMBIO debería ser colocar el valor de A en B y el valor de B en A.
2. Defina la palabra TRANSFERIR para mover datos entre dos matrices de igual longitud.

3.0 ENTRADAS Y SALIDAS

Para realizar cualquier operación es necesario introducir datos en la computadora y obtener resultados. El Forth tiene varias formas de hacer esto.

3.1 EL CONJUNTO DE CARACTERES

Los caracteres gráficos definibles por el usuario (144-164) pueden ser definidos usando la palabra DEF. Para definir un carácter, ponga 8 bytes en el stack representando las formaciones de bits que configurarán el nuevo carácter desde la fila inferior a la superior, entonces, teclee el código del carácter que desea cambiar, seguido de la palabra DEF. Esto puede parecer complicado, pero he aquí cómo definir un hombrecito:

```
HEX. <ENTER>
81 81 66 3C FF 7E 18 18 <ENTER>
DECIMAL <ENTER>
144 DEF <ENTER> OK
```

El carácter 144 es ahora un pequeño hombrecito.

3.2 ENTRADA DE INFORMACION

El Forth no tiene comandos de entrada de información en sí, porque los números que son los parámetros para cada comando son almacenados en el stack de parámetros. Estos números se colocan generalmente en el stack precediendo la ejecución de un comando. Por ejemplo, supongamos que deseamos calcular. $(4x^3 - 3x + 2)$ para cualquier valor de x. Es fácil definir un comando que calcule x^3 y una vez hecho esto, otro que calcule el resto de la ecuación.

```
: CUBO DUP DUP * * ;
: ECUACION DUP CUBO 4 * SWAP 3 * - 2 + . ; <ENTER> OK.
```

Hemos definido la función ECUACION para calcular esta ecuación de grado tres. El valor del parámetro x que se necesita para el cálculo se coloca en el stack antes de que se use el comando. Esto se hace de esta manera:

8 ECUACION <ENTER> 2026 OK

Así pues, cualquier parámetro puede ser introducido para actuar con un comando y no es necesario ningún comando de entrada de datos. De todas maneras, el Forth tiene un comando para aceptar la pulsación de una tecla en el teclado. Esta instrucción es similar a INKEY$ en Basic. Sin embargo, KEY espera a que se presione una tecla mientras que INKEY$ pasa de largo. Supongamos que durante la ejecución del programa necesitamos una entrada de datos desde el teclado. Por ejemplo:

¿DESEA INSTRUCCIONES? (S/N).

KEY coloca el valor ASCII de la tecla presionada en el stack de parámetros donde puede ser examinado. En Forth no hay ninguna rutina para introducir un número de más de 1 dígito desde el teclado. De todas formas, hay una rutina descrita en el Apéndice para introducir un número llamada INPUT. Trate de escribir una rutina que haga usted mismo.

3.3 SALIDAS E IMPRESION

El Forth le ofrece varias maneras de sacar información. La más usada es la salida de una línea de texto. La palabra Forth que se usa para esto es **.** seguido del mensaje entre comillas.
Por ejemplo:

."ESTA ES UNA LINEA DE TEXTO" <ENTER>

Imprime ESTA ES UNA LINEA DE TEXTO en la pantalla. Cualquier otra salida de información posterior aparecerá en la misma línea. La palabra Forth CR efectúa un ENTER o retorno de carro. La siguiente impresión se hará entonces desde el principio de la línea siguiente. La palabra Forth del Spoctrum AT es la misma que en el Basic y puede utilizarse para colocar el cursor en una posición determinada de la pantalla para imprimir allí.
Por ejemplo, para imprimir en la tercera línea, segunda columna, teclee: 3 2 AT. "HOLA"
El Forth del Spectrum corre la pantalla automáticamente si el cursor se va a salir de la pantalla por debajo.
Otra manera de imprimir un carácter se lleva a cabo mediante la palabra Forth “EMIT” que imprime el carácter cuyo código ASCII está encima del stack de parámetros. Este valor puede colocarse en el stack con el comando KEY. Esto le permite además imprimir un carácter que no se puede obtener directamente del teclado.
Por ejemplo: 143 EMIT <ENTER> ■ OK imprime el carácter cuyo código ASCII es 143, que es el "espacio gráfico". Una salida de información puede también mandarse por impresora.
Hay una variable en Forth llamada PRINT. Si esta variable contiene el valor 0, entonces la impresión se realiza en la pantalla. Si contiene el valor 1, entonces la impresión se realiza en pantalla y en la impresora. Por ejemplo, para mandar información a la impresora, tecléese:

1 PRINT I <ENTER>

"Apague" la impresora tecleando: ç

0 PRINT I <.ENTER>

El equivalente de Copy en Basic es en Forth la misma palabra, COPY. Teclee COPY <ENTER> para hacer una copia de la pantalla en la impresora

3.4 IMPRESION DE NUMEROS

La manera más sencilla de imprimir un número es usar la palabra Forth "**.**", un punto, que ya conocemos. Esto imprime el número superior del stack en la mínima anchura de campo posible, esto es, no imprime ningún cero a la izquierda y deja un espacio detrás del número. El formateado de números se puede conseguir con las siguientes palabras Forth:

".R" imprime el número en un campo de una anchura dada.
Por ejemplo 103 4 .R <ENTER> 103 OK imprime 103 en un campo de 4 caracteres. El Forth le proporcionará también la posibilidad de formatear sus salidas como desee

"<#" empieza la definición de salidas formateadas y necesita que haya un número de doble precisón en el stack. Puesto que, generalmente, se utilizan números de precisión simple, es posible convertir un número de precisión simple en uno de precisión doble utilizando la palabra Forth "S — > D" que convierte el número encima del stack en uno de doble precisión.

Para un formateado de salidas, puede usted utilizar las siguientes palabras Forth:

"#" pone el siguiente dígito en el buffer de salidas empezando por el valor más bajo. Por ejemplo, si tenemos el número 112, el primer # pone un 2 en el buffer, el siguiente # pone un 1 en el buffer etc.

"#S" pone los dígitos que quedan en el buffer de salidas si no son ceros.

"HOLD" utilizado en la forma "46 HOLD" pone en la siguiente parte del buffer el carácter cuyo código es 46.

"#>" Fin de salidas formateadas. Deja la posición de memoria y la longitud del buffer de salidas en el stack. A partir de aquí, la cadena puede sacarse utilizando el comando "TYPE". Ejemplo: supongamos que deseamos definir el nuevo comando "FORM" para producir salidas formateadas:

```
: FORM < # # # 46 HOLD # # S #>; <ENTER > OK
```

Esto hará que se imprima cualquier número de doble precisión como (. . . . x.xx).
Ejemplo: 11437. FORM TYPE <ENTER> 114.73 OK
0. FORM TYPE <ENTER> 0.00 OK.
Se necesita práctica para dominar el arte de usar salidas numéricas formateadas. Existen comandos de doble precisión. (Ver Glosario.)

3.5 OTRAS OPERACIONES DE IMPRESION

El ZX-Forth tiene otros cuatro comandos para ayudar al formateado en la pantalla:

"SPACE" Imprime un espacio en la pantalla. (Equivalente a 32 EMIT.)

"SPACES" Esta palabra se utiliza para imprimir un número dado de espacios especificado por el número que hay encima del stack. Por ejemplo: 5 SPACES <ENTER> ….. OK imprime 5

espacios en la pantalla.

"HOME" Esta palabra coloca la posición de impresión en el rincón superior izquierdo de la pantalla. Cualquier impresión posterior empezará a partir de esa posición.

"CLS" Esta palabra borra la pantalla y mueve la posición de impresión al rincón superior izquierdo de la pantalla.

3.6 ALTA RESOLUCION EN COLOR Y SONIDO

La mayoría de los comandos de color y alta resolución están incluidos en el Forth del Spectrum. Los códigos de color son los mismos que en Basic: 1 = Rojo, etc. La única diferencia al usar estos comandos en Forth es que los parámetros preceden a la instrucción en vez de seguirla. Ejemplo, para cambiar la tinta a rojo:

BASIC	FORTH
INK 1 <ENTER>	1 INK <ENTER>

NOTA: en Forth estos colores son sólo temporales y se convierten en los que aparecen originalmente al encender le máquina cuando se ejecuta un comando CLS. Para hacer que estos colores sean permanentes, teclee la palabra PERM.

Ejemplo: Papel azul, tinta blanca.

2 Paper 7 Ink PERM <ENTER>

Comandos de color, sonido y alta resolución

BASIC			FORTH
INK	X	X	INK
PAPER	X	X	PAPER
FLASH	X	X	FLASH
BRIGHT	X	X	BRIGHT
INVERSE	X	X	INV
OVER	X	X	GOVER
BORDER	X	X	BORDER
PLOT	X, Y	X, Y	PLOT
CIRCLE	X, Y, radio	Radio, X, Y	CIRCLE
DRAW	X, Y	X, Y	DRAW
BEEP	X, Y	X, Y	BEEP

TABLA 5
CODIGO DE CARACTERES

0-7	Caracteres de control	
8	Borrado (DELETE) Caps Shift-0	
9-12	Caracteres de control	
13	Entradas (ENTER)	
14-31	No se usan	
32	Espacio (SPACE)	
33	I	Symbol shift “1”
34	“	Symbol shift “p”

35	#	Symbol shift “3”
36	$	Symbol-shift “5”
37	%	
38	—	Symbol shift “6”
39	o	Symbol shift “7”
40	(	Symbol shift “8”
41	)	Symbol shift “9”
42	*	Symbol shift “B”
43	+	Symbol shift "K"
44	o	Symbol shift “N”
45	—	Symbol shift “J”
46	.	Symbol-shift "M"
47	/	
48	0	
49	1	
50	2	
51	3	
52	4	
53	5	
54	6	
55	7	
56	8	
57	9	
58	:	Symbol-shift "Z"
59	;	Symbol-shift “O”
60	<	Symbol-shift "R"
61	=	Symbol-shift "L"
62	>	Symbol-shift "T"
63	?	Symbol-shift “C”
64	@	Symbol-shift "2"
65	A	
66	B	
67	C	
68	D	
69	E	
70	F	
71	G	
72	H	
73	I	
74	J	
75	K	
76	L	
77	M	
78	N	
79	O	
80	P	
81	Q	
82	R	
83	S	
84	T	

85	U	
86	V	
87	W	
88	X	
89	Y	
90	Z	
91	[	Symbol-shift "Y"
92	\	Symbol-shift "D"
93	]	Symbol-shift "U"
94	↑	Symbol-shift "H"
95	-	Symbol-shift "O"
96	£	
97	a	
98	b	
99	c	
100	d	
101	e	
102	f	
103	g	
104	h	
105	i	
106	j	
107	k	
108	l	
109	m	
110	n	
111	o	
112	p	
113	q	
114	r	
115	s	
116	t	
117	u	
118	v	
119	w	
120	x	
121	y	
122	z	
123	{	
124	|	
125	}	
126	~	
127	©	
128		
129		
130		
131		
132		
133		
134		

135
136
137
138
139
140
141
142
143
144-164 Caracteres gráficos definidos por el usuario.
165-255 Palabras clave Basic. No se usa.

4.0 RAMIFICACIONES CONDICIONALES Y BUCLES

El Forth proporciona instrucciones de ramificación condicional que alteran el orden en el que se ejecutan los comandos dependiendo de una condición dada. El Forth también proporciona estructuras de bucle para repetir una secuencia de comandos un número dado de veces.
NOTA: Las ramificaciones condicionales y los bucles no pueden ejecutarse inmediatamente y deben ir incluidos con una definición.

4.1 RAMIFICACIONES CONDICIONALES

Tres palabras compiladoras. "IF", ELSE, ENDIF (o THEN) se usan para compilar ramificaciones condicionales en una definición. En Forth. el comando "IF" examina la cima del stack para determinar qué ramificación se tomará. Una ramificación condicional tiene la siguiente estructura:

: DEFINICION condición IF (verdad) haz esto ELSE (falso) haz aquello THEN continúa.

Qué hace cada cosa

DEFINICION:	Empieza la definición
Condición:	El resultado de una comparación deja un 0 (falso) o un 1 (verdadero) en el stack
IF	
haz esto:	Coge y examina el valor del número dejado por la condición en el stack. Esto se ejecuta si el número no es cero (la condición era verdad)
ELSE	
haz aquello	Aquello se ejecuta si el número es cero (la condición no se da)
THEN	
continúa	Continúa el programa

La palabra IF marca al momento en el que se coge y examina el valor encima del stack. Si este valor no es cero, se ejecuta todo, hasta la palabra ELSE, y el ELSE manda la ejecución al THEN. En cambio, si el valor del stack examinado es cero, todos los comandos, hasta el ELSE, se saltan, no se ejecutan y sigue la ejecución a partir del ELSE.
La parte de la ramificación "ELSE haz aquello" es opcional y se puede omitir si no se necesita. El "valor de verdad" en el stack es generalmente el resultado de una comparación que haya usado uno de los operadores de comparación Forth. Como < **.**> = etc. (Ver capítulo 1)
Dos valores de verdad pueden combinarse con las palabras Forth "AND, OR y XOR". Por ejemplo:

AND Deja valor de verdad 1 si los dos valores de verdad encima del stack son verdaderos.

OR Dejar valor de verdad 1 si uno o los dos valores encima del stack son verdaderos.
XOR Deja valor de verdad 1 si uno de los dos valores es verdadero y el otro es falso.

Por ejemplo: 11 AND deja un 1
10 AND deja un 0. (Ver tablas de verdad en tabla 6)

Por ejemplo supongamos que deseamos definir una palabra Forth para evaluar exámenes, digamos que con el 50% de las preguntas bien se aprueba y con menos del 50% se suspende. Podemos definir esa palabra de esta manera:

: EXAMEN 50 < IF ." SUSPENSO" ELSE . "APROBADO" THEN CR; <ENTER> OK

Después, para usar esa palabra, se teclearía:

(% de preguntas bien contestadas) EXAMEN <ENTER>
Si el % es menos de 50, la computadora imprime SUSPENSO.
Si el % es mayor o igual a 50 la computadora imprime APROBADO.

4.2 BUCLES INDEFINIDOS

El Forth dispone también de una serie de estructuras de bucle que repiten un grupo de comandos bien hasta que una condición dada sea satisfecha, o bien un número dado de veces. En esta sección trataremos el primer tipo Este tipo de bucle puede ser de dos formas. He aquí la primera de ellas.

: EJEMPLO BEGIN proceso condición UNTIL continuar;
<ENTER> OK

donde:

:EJEMPLO	Empieza la definición.
BEGIN	Marca el comienzo de un bucle indefinido.
Proceso	Define la acción que debe llevarse a cabo
Condición	Deja un valor de verdad en el stack.
UNTIL	Coge ese valor de verdad del stack y vuelve al BEGIN si ese valor es (la condición no se da)
Continuar;	Se ejecuta lo que hay detrás del UNTIL si ese valor de verdad no es cero.

Ejemplo: supongamos que deseamos buscar a través de la memoria la posición de memoria en la que está almacenado un número de 16 bit e imprimirla cuando la encontremos. Podemos definir la palabra BUSCAR para que haga esto. Supongamos que estamos buscando el primer 0 que encontramos partiendo desde el principio de la memoria:

: BUSCAR 0 BEGIN DUP @ SWAP 1 + SWAP 0 = UNTIL 1 - .: <ENTER> OK

Las instrucciones entre el BEGIN y el UNTIL se ejecutan repetidamente hasta que el valor que se encuentra sea un 0. El 0 que va antes del bucle es la posición de memoria desde la que hay que comenzar la búsqueda. El comando detrás del UNTIL imprime la dirección de memoria en la que ha aparecido el primer 0.
La segunda forma de bucle indefinido es esta:

:EJ1 BEGIN condición WHILE proceso REPEAT continuar; <ENTER> OK

: EJ 7	Empieza la definición.
BEGIN	Marca el comienzo de un bucle indefinido.
Condición	Deja un valor lógico en el stack.
WHILE	Si el valor lógico no es cero.
Proceso	Entonces se ejecuta el proceso.
REPEAT	Vuelve al BEGIN.
Continuar;	Si el valor lógico es 0, entonces lo que hay detrás del REPEAT se ejecuta.

Ejemplo para buscar a través de la memoria como en el ejemplo anterior:

:BUSCAR 1 0 BEGIN DUP @ WHILE 1 + REPEAT.; <ENTER>OK

Intente descifrar esta línea usted mismo.
Otro tipo de bucle es el siguiente

EJZ BEGIN proceso AGAIN; <ENTER >.

Este bucle repite el proceso indefinidamente. Es un bucle infinito y sólo puede terminar si se pulsa la tecla Break.

4.3 EL STACK DE RETORNO

El Forth usa dos stacks: el stack de parámetros y el stack de retorno. Esto es así porque de otra manera podrían confundirse parámetros con direcciones de retorno. Hay varios comandos para transferir valores desde un stack al otro.

>R Coge un número del stack de parámetros y lo coloca en el stack de retorno.
R > Coge un número del stack de retorno y lo coloca en el stack de parámetros.
NOTA: Sise usan estos comandos en una definición, deben aparecer ambos para complementarse y no dejar parámetros en el stack de retorno y viceversa.
R o I Hace una copia del número colocado encima del stack de retorno y lo coloca en el stack de parámetros. El stack de retorno no es alterado.

Para que se acostumbre a estos comandos intente definir la palabra 2INTERCAMBIO para intercambiar los dos primeros números del stack de parámetros con el tercero y el cuarto. Esto es lo que debe suceder.

STACK
1 2 3 4 5 2 INTERCAMBIO
1 4 5 2 3

4.4 BUCLES CONTROLADOS

Un bucle controlado es aquel que se repite un cierto número de veces. El Forth nos proporciona la estructura DO ... LOOP para hacer esto. Esta estructura toma la siguiente forma:

: DIEZ VECES 10 0 DO proceso LOOP; ENTER OK

DIEZ VECES	Empieza la definición.
10	Da el valor en el que tiene que terminar el bucle.
0	Da el valor del que se parte.
DO	Transfiere los parámetros del bucle al stack de retorno.
Proceso	
LOOP	Vuelve al DO mientras no se haya alcanzado el valor 10.

Dentro del bucle se puede acceder al índice del bucle mediante la palabra Forth "I". Suponiendo que deseamos imprimir los números desde el 0 al 9 en la pantalla en una línea cada uno. Podemos definir la palabra NUM para que haga esto:

```
: NUM 10 0 DO CR I. LOOP : <ENTER> OK
```

Si desea que el índice no se incremente de 1 en 1, puede usted utilizar la estructura DO... +LOOP Por ejemplo:

```
: INCREMENTO2 10 0 DO 2 +LOOP; <ENTER> OK
                    ↑
              Incremento. Ahora el bucle aumenta de 2 en 2 .
```

INCREMENTO 2	Empieza la definición.
10 0	Parámetros del bucle.
DO	Comenzar bucle
2	Pone el valor del incremento en el stack.
+ LOOP	Coge el valor del incremento en el stack y lo suma al índice del bucle. Si el resultado es menor que el valor con el que debe terminar el bucle, el bucle se repite con el nuevo valor del índice.

De otra manera, el bucle termina. El incremento puede ser positivo o negativo. Usando un incremento negativo, el bucle realizará una cuenta atrás. Si el incremento es negativo, los parámetros deben colocarse invertidos, esto es: 0 10 en vez de 10 0.
Por ejemplo, para imprimir los números del 10 al 1, se puede definir la palabra DEL DIEZ AL UNO.

```
: DEL DIEZ AL UNO 0 10 DO CR I. - 1 +LOOP ; <ENTER> OK
```

En este caso LOOP comprueba si el valor en el índice del bucle es mayor que el valor con el que debe terminar el bucle y repite si es verdad. Hay una forma de dejar una estructura DO… LOOP antes de que el bucle se haya terminado. Por ejemplo, si una cierta condición se cumple, el comando LEAVE hará que el bucle termine en el siguiente LOOP o +LOOP. Ejemplo:

```
: EJEMPLO 10 0 DO I DUP 6 = IF LEAVE ELSE . THEN LOOP: <ENTER> OK
```

Esta es una manera un tanto torpe de imprimir los números del 0 al 5, pero sirve para ilustrarnos sobre el uso de esta palabra.
Ejercicio
1. Defina la palabra ELEVAR de tal manera que m n ELEVAR calcule el valor de m para n positivo.

4.5 ESTRUCTURAS ANIDADAS

Las estructuras DO … LOOP y IF … THEN pueden contener la una a la otra pero sólo si están correctamente anidadas. Esto es: un DO ... LOOP puede ir dentro de otro pero no puede saltar por encima de otro. Por ejemplo:

CORRECTO:

```
          /-------------------------------------------------------\
          /                        /----------------\              \
... IF   10      0       DO      ...      LOOP          THEN
```

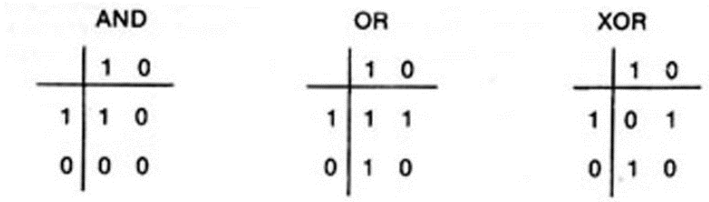
INCORRECTO:

```
          /----------------------------------------\
                                  /-------------------------------------\
… IF   10      0       DO      …        THEN          LOOP
```

EJERCICIOS

1. ¿Cómo definiría usted las palabras Forth MAX, MIN y ABS?
2. Defina le palabra FACTORIAL. Para ello calcule el factorial de un número.

TABLA 6



5.0 ALMACENAMIENTO EN CINTA

El Forth trabaja normalmente de forma interactiva y una vez que una definición ha sido introducida no hay ningún modo de cambiarla, sino de volverla a escribir. El Forth proporciona también un método para almacenar el código en una pantalla numerada. Una pantalla consista en 16 líneas de 64 caracteres Los programas se almacenan en una pantalla usando el Editor. (Ver el manual del Editor.)

5.1 GRABANDO PROGRAMAS

Una vez que el programa haya sido escrito en la pantalla, es posible grabarlo en una cassette. Para hacer esto debe preparar su computadora para la grabación (ver capítulo dedicado a grabación en el manual del Spectrum). El comando FLUSH indica al Forth del Spectrum que debe grabar la presente pantalla al cassette (no confunda la pantalla de su televisor con una pantalla de información). Si teclea FLUSH <ENTER> el Forth del Spectrum responderá con el mensaje READY CASSETTE. Apriete la tecla de grabación de su cassette y presione <ENTER>. Una pantalla tarda 30 segundos en grabarse. Si en vez de presionar <ENTER> presiona usted cualquier otra tecla, el comando FLUSH es cancelado y el cursor reaparece.

5.2 CARGANDO PROGRAMAS

El Forth del Spectrum le proporcionaré a usted dos palabras para cargar un programa del cassette. "LIST" se usa para listar una pantalla y se escribe en la forma ***n LIST*** donde ***n*** es el número de pantalla. Si la pantalla ***n*** está en la memoria, ésta es listada en su televisor. Si no está en memoria, el Forth de Spectrum la intentará cargar del cassette. Conecte las clavijas para cargar (ver manual del Spectrum) y coloque la cinta en el espacio sin grabar que precede a la pantalla que desea grabar. Presione la tecla <ENTER> y ponga en marcha su cassette. Si la pantalla se carga correctamente, aparecerá listada en su televisor. Si no se carga bien o ha intentado usted cargar la pantalla que no era, entonces el mensaje READY CASSETTE volverá a aparecer. Pruebe de nuevo. Use <ENTER> para seguir o cualquier otra tecla si desea cancelar el LIST como en FLUSH. La variable FIRST

contiene el número de cualquier pantalla que esté en ese momento en memoria. Puede examinar este número tecleando: FIRST ? <ENTER>.
Si desea dejar de cargar, presione la tecla "Space".
"LOAD" se usa para compilar las definiciones en una pantalla. Este comando debe ir precedido del número de pantalla al igual que el "LIST". Si la pantalla no está ya en memoria, es tratado como si hubiera sido escrito desde el teclado.

5.3 EL FORMATO DE LA PANTALLA

Cada pantalla tiene un número y consiste en 16 líneas de 64 caracteres. De todas maneras, para que el Forth pueda usarlas más eficazmente, es necesario terminar la pantalla con unas palabras Forth especiales.
La palabra "- -> " al final de la última línea de la pantalla ordena a Forth del Spectrum a "LOAD" (cargar) la siguiente pantalla consecutiva cuando se ha cargado la pantalla actual usando el comando "LOAD". Esto se usa cuando un programa ocupa más de una pantalla Forth. El Forth del Spectrum compila la pantalla en la memoria y entonces emite el mensaje READY CASSETTE para "LOAD" (cargar) la siguiente pantalla.
La palabra "; S " al final de la última línea de la pantalla hace que termine el comando "LOAD". Esto se usa en la última pantalla del programa. Si no se coloca este indicador de fin de programa en la última pantalla, el sistema se escapará de control.
NOTA. Sólo puede haber una pantalla almacenada en memoria. La pantalla que se cargue a continuación borrará la pantalla que hubiera residente en memoria.

6.0 OTROS COMANDOS UTILES

Puesto que el Forth tiene un vocabulario tan rico, es imposible listarlos todos aquí, la única manera de acostumbrarse a ellos es leer el glosario y probar los comandos allí descritos.
La palabra "FORGET" se usa para anular una definición. Se usa de la siguiente forma:

FORGET palabra <ENTER>

NOTA esto hará que se borre del diccionario la palabra que tiene por nombre "palabra" y todas las que se hayan definido después de ésta. Si por ejemplo definiéremos les siguientes palabras:

: PAL1 "HOL "; <ENTER>
: PAL2 “¿QUE TAL?” ; <ENTER>
: PAL 3 "ADIO " ; <ENTER>

el comando FORGET PAL2 <ENTER> OK borraría PAL2 y PAL3 del diccionario (PAL1 permanecería intacta).

"BYE"	Se usa para salir del Forth y volver al Basic.
“**IMMEDIATE"**	Normalmente cuando se encuentra una palabra dentro de una definición, se compila como parte de la definición. Si usted necesita que una palabra se ejecute cuando se encuentre, incluso si está dentro de una definición, entonces debe declarar esa palabra "Immediate" (inmediata) al final de su definición.
"TASK"	Esta palabra es una definición "postiza" que se usa para empezar un programa, de tal forma que cuando desee borrar un programa, sepa desde dónde debe borrar.
"VLIST"	Esta palabra hará que aparezcan todas las palabras que hay en el diccionario La palabra que haya sido definida más recientemente será la primera en aparecer. El listado de palabras puede pararse presionando la tecla BREAK (SHIFT SPACE).

Ejemplo:

: WD3 . “ESTA COMPILANDO" ; IMMEDIATE <ENTER> OK

Esto imprime el mensaje entre comillas cuando WD3 es encontrado y no está compilado.
" (" Y ") y "LITERAL"
A veces es conveniente calcular una constante con una definición sin tener que calcular después cada vez que la definición se ejecute. La palabra Forth "["coloca temporalmente a la computadora en modo inmediato, de tal manera que cualquier cosa que se teclee se ejecutará inmediatamente. La palabra Forth "]" coloca a la computadora de nuevo en modo de compilación, en la definición. La palabra Forth "LITERAL" coloca el valor encima del stack en la definición como una constante LITERAL es un comando inmediato y por tanto es ejecutado tan pronto como se encuentra.
Por ejemplo, las dos definiciones siguientes son equivalentes:

: DF1 3 [13 + 2 *] LITERAL + . ; <ENTER>
: DF2 3 8 +. ; <ENTER>

Esto se usa cuando no se conoce el resultado del cálculo. Las palabras descritas le evitan tener que realizar el cálculo en un papel aparte para poner el valor exacto de la constante en la definición.
"VOCABULARY"
El Forth le permite creer sus propios vocabularios, de tal manera que todas las palabras para un programa se almacenan juntas. Los vocabularios deben ser declarados inmediatos. Por ejemplo, para definir un vocabulario llamado MIO, teclee:

VOCABULARY MIO IMMEDIATE <ENTER>

Para colocar definiciones dentro de su vocabulario, teclee el nombre de su vocabulario seguido del comando DEFINITIONS. Por ejemplo:

MIO DEFINITIONS <ENTER>

Todas las definiciones hechas con anterioridad a esto serán colocadas en el vocabulario "MIO" hasta que usted cambie de nuevo su vocabulario. Desde dentro de un vocabulario, puede usted usar todas las palabras incluidas en ese vocabulario y todas las palabras dentro del vocabulario en cuyo interior se definió el vocabulario en el que esté. Si desea usar una palabra de otro vocabulario entonces debe preceder esa palabra con el nombre del vocabulario del que proceda. El vocabulario básico es el FORTH
"MEM"
Esta palabra imprime la memoria que queda libre. El número de bytes libres se imprime en la base numérica en la que se esté trabajando.

APENDICE "A" - CODIGOS DE ERROR

CODIGOS DE ERROR	MENSAJE DE ERROR
0	COMANDO INEXISTENTE EN EL DICCIONARIO
1	STACK VACIO
2	DICCIONARIO LLENO (FUERA DE LA MEMORIA)
3	MODO DE DIRECCIONAMIENTO INCORRECTO
4	AVISO NOMBRE YA UTILIZADO
7	STACK LLENO
17	PALABRA DEBE USARSE EN LA DEFINICION
18	EJECUCION SOLAMENTE
19	CONDICIONAL DESPAREJADO
20	DEFINICION NO CONCLUIDA

21 LOCALIZADA EN DICCIONARIO PROTEGIDO
22 USESE SOLO CUANDO SE CARGUE
23 FUERA DE LA ACTUAL PANTALLA DE EDICION
24 DECLARAR VOCABULARIO

APENDICE "B" - RUTINAS UTILES

La rutina de entrada de datos aquí listada recibe del teclado un número determinado con un ENTER cuando se ejecuta y lo deja en el stack.

```
: INPUT PAD 1 + 64 EXPECT .0 PAD

(NUMBER) DROP DROP ;
```

INPUT recibe del teclado un número de doble precisión

```
: INPUT. PAD 1 + 64 EXPECT .0 PAD

(NUMBER) DROP ;
```

GLOSARIO DEL FORTH DE LA SPECTRUM

Este glosario contiene las definiciones de todas las palabras incluidas en el primer lanzamiento de Forth para la Spectrum. Las definiciones se presentan en orden según su clasificación ASCII. La primera línea de cada entrada muestra una descripción simbólica de los procedimientos de acción en el stack de parámetros. Los símbolos indican el orden en que los parámetros entrantes han sido colocados en el stack. Tres guiones (“ ---“) indican el punto de ejecución; cualquier parámetro que quede en el stack se lista. En esta notación, la parte de arriba del stack corresponde a la derecha. Los símbolos son:

addr	(address) posición de memoria.
b	Byte de 8 bits (los ocho bits altos son cero).
c	Carácter ASCII de 9 bits (los ocho bits altos son cero),
d	Entero doble de 32 bits con signo, la porción más significativa con signo encima del stack.
f	Bandera boleana. 0 - falso, =/= 0 verdadero.
ff	Bandera boleana de falsedad-0.
n	Número entero con signo de 16 bit.
u	Número entero sin signo de 16 bit.
tf	Bandera boleana de verdad =/=0.

Las letras mayúsculas de la derecha muestran características referentes a la definición de esa palabra.

C	Se puede usar sólo en una definición limitada por: y; Un dígito indica el número de direcciones de memoria que usa, en caso de que no sea l.
E	Sólo se usa en modo de ejecución inmediata.
LO	Definición del nivel O del FORTH-78.
L1	Definición del nivel 1 del FORTH-78.
P	Tiene puesto el bit de prioridad. Se ejecutará incluso durante la compilación.
U	Variable del usuario.

Si no se indica lo contrario, todas las referencias a número son a enteros de 16 bit con signo. Para números de precisión doble con signo, la parte más significativa junto con el signo se almacena en la parte más alta del stack.

! n addr LO
- Almacena los 16 bits de n en la dirección addr.

!CSP - Almacena la posición del stack en CSP. Se usa como parte del sistema de seguridad del compilador.

d1 - - - d2 LO
- Genera a partir de un número doble d1, el siguiente carácter ASCII que situará en una cadena de salida.
El resultado d2 es el cociente que resulta al dividir por BASE, y se mantiene para operaciones sucesivas. Se usa entre <#y #>. Véase S.

#> d - - - addr n.° de caracteres LO
- Finaliza la conversión de salida quitando d, y dejando la dirección del texto y número de caracteres dispuesto para usarse TYPE.

#S d1 - - - d2 LO
- Genera un texto ASCII en el buffer de salida de texto, a través del uso de 4,
hasta que se obtenga un cero. Se usa entre <# y #>.

, - - - addr P,LO
Usado en la forma, nnn
— Deja la dirección del campo de parámetros de la palabra del diccionario número nnnn.
Como control del compilador, provoca que en una definición entre : y; se compile la dirección como un literal.

(Usado en la forma. (cccc) P.LO
—ignora un comentarlo delimitado por otro paréntesis) en la misma línea. Puede usarse tanto en ejecución como en compilación. Se necesita un espacio en blanco detrás del primer paréntesis.

(.”) C +
—El proceso inmediato que se compila por ." . Transmite la línea de texto siguiente al periférico de salida elegido. Véase."

(; CODE) —El proceso inmediato, compiladop por ; CODE, que reescribe el campo del código de la última palabra definida para apuntar a la subrutina de código máquina correspondiente.

(+LOOP) n - - - C2
—El proceso inmediato compilado por +LOOP, que incrementa el índice del bucle en n y comprueba si se ha completado el bucle.

(ABORT) —Se ejecuta después de un error cuando WARNING toma el valor —1. Esta palabra normalmente ejecuta ABORT, pero puede alterarse (con cuidado) para un uso alternativo.

(DO) —El proceso inmediato compilado por DO que transfiere parámetros de control del bucle al stack de RETURN. Véase DO.

(FIND) addr1 addr2 - - - pfa b tf (ok)
addr addr2 - - - ff (mal).
— Busca en el diccionario empezando en la dirección del campo de nombres addr2 hasta el texto de la dirección addr1. Al acabar deja la dirección del campo de parámetros, la longitud del campo del nombre y una señal lógica verdadera, si la búsqueda ha sido positiva. Si no, se dejará simplemente una señal lógica negativa.

(LINE) n1 n2 - - - addr count.
—Convierte la línea número n1 de la página n2 a la dirección del buffer del cassette que contenga los datos. Un "count" (cuenta) de 64 indica una longitud de una línea de texto entera.

(LOOP) —El proceso Inmediato compilado por LOOP que incrementa el índice del bucle y comprueba si se ha finalizado el bucle.

(NUMBER) d1 addrl - - - d2 addr2
—Conviene el texto ASCII que comienza en addr1 + 1 teniendo en cuenta la Base numérica.
El valor nuevo se acumula en el número de precisión doble d1, que se deja como d2. Addr2 es la dirección del primer dígito que no se puede convertir. Se usa por NUMBER.

*/ n1 n2 - - - prod
—Deja el producto con signo de dos números con signo.
n1 n2 n3 - - - n4
—Deja como n4 el resultado de la operación.
n1*n2/n3, donde n1, n2 y n3 son números con signo. Durante la operación se almacena temporalmente un número de 31 bit, lo que permite una mayor exactitud que con la secuencia n1 n2 * n3/

*/MOD n1 n2 n3 - - - n4 n5
—Calcula el cociente n5 y el resto n4 de la operación n1 + n2 / n3. También se utiliza un producto intermedio de 31 bit como en*/

+ n1 n2 - - - SUM
—Calcula la suma n1 + n2.

+! n addr - - -
—Suma n al valor contenido en esta dirección.

+ — n1 n2 - - - n3
—Se aplicará el signo de n2 a n1, que se dejará como n3.

+ LOOP n1 - - - (run)
—Se usa en una definición: y; de la forma:
DO . . . n1 + LOOP. Durante la ejecución del programa, LOOP provoca un salto a DO dependiendo de n1, el índice del bucle, y el límite del bucle. El incremento, (neg. o pos.) n1 se suma al índice, y entonces se compara con el límite. El salto a DO se produce hasta que el nuevo índice sea igual o mayor que el límite cuando (n1 > 0), o hasta que el nuevo índice sea igual o menor que el límite cuando (n1< 0). Después de abandonar el bucle, se descartan los parámetros y se continúa con la ejecución normal del programa.

+ ORIGIN n - - - addr
—Deja la dirección de memoria relativa a n del origen del área de parámetros. n es la unidad mínima de direcciones, tanto byte como palabra.
n - - - LO
—Guarda n en la siguiente celda de memoria del diccionario disponible, ajustando el puntero del diccionario.

— n1 n2 - - - diff LO
—Calcula la diferencia de n1 — n2.

- - > P,LO
—Continúa la interpretación en la siguiente página.

-DUP n1 - - - n1 si n1 =0
n1 - - - n1 ni si n1 =0

—Duplica n1 si no es cero. Esto se usa normalmente para copiar un valor justo antes de IF y no tener que usar ELSE para deshacerse de él.

—FIND - - - pfa tf (encontrado)
- - - ft (no encontrado)

—Acepta la siguiente palabra de texto (delimitado por espacios) en la cadena de entrada a HERE, busca en CONTEXT y luego en los vocabularios seleccionados una entrada válida. Si se encuentra, se dejarán en el stack la dirección del campo de parámetros de entrada, su longitud, y una señal lógica 1. De otra forma se dejará simplemente una señal 0.

—TRAILING addr n1 - - - addr n2

—Ajusta el contador de caracteres n1 de una cadena literal que comienza en addr para suprimir la salida de espacios de arrastre. Esto es, los caracteres desde addr + n1 hasta addr + n2 son espacios en blanco.

n - - - LO

—Imprime un número con signo de 16 bit en complemento a dos, de acuerdo con la base seleccionada.

Usado en la forma." cccc" P,LO

—Compila una cadena delimitada por " con una ejecución inmediata para transmitir el texto al periférico seleccionado.

Si se ejecuta fuera de una definición, ." imprimirá inmediatamente el texto hasta".

.R n1 n2 - - -

—Imprime un número n alineado al margen derecho de un campo de una anchura n2. No se imprimirá ningún espacio más.

/ n1 n2 - - - coc. LO

—Deja en el stack el cociente n1/n2.

/MOD n1 n2 - - - resto coc. LO

—Deja en el stack el resto del cociente n1/n2. Dicho resto lleva el signo del dividendo.

0 1 2 3 - - - n

—Estos números se usan tantas veces, que se han definido con un nombre en el diccionario como constantes.

0< n - - - f LO

—Deja un 1 si el número es menor que cero (negativo); en otro caso deja 0.

0= n - - - f LO

—Deja 1 si el número es igual a 0; si no, deja 0.

0 BRANCH f - - - C2

— El proceso inmediato de saltos condicionales. Si f es cero, el siguiente parámetro se suma al puntero del intérprete para saltar hacia delante o hacia atrás.

Se compila con IF, UNTIL y WHILE.

1+ n1 - - - n2 LI

—Se incrementa n1 en uno.

2+ n1 - - - n2

—Incrementa n1 en dos.

2DUP —Duplica un valor de 32 bits en el stack. S SIGN.

: P,E,LO

—Se usa para definir una palabra.

: cccc - - - ;

Crea una palabra cccc en el diccionario que equivale a la secuencia de palabras Forth que siguen a continuación (...) hasta encontrar; o ; CODE.

El proceso de compilación lo realiza el compilador hasta que STATE no sea cero.

Otros detalles son que el vocabulario CONTEXT se adopta como el vocabulario vigente y qué palabras con el bit de Prioridad activo se ejecutan i CODE inmediatamente en vez de compilarse.

P,C,LO

—Termina una definición y para cualquier compilación s que siga. Compila la ejecución de; S.

CODE —Se usa con el ensamblador.

S —Para interpretar una página ; S es también la palabra de ejecución inmediata compilada al final de una definición que retorna la ejecución al proceso que lo ha llamado.

< N1 n2 - - - f LO

-Deja 1 si n1 es menor que n2. Si no, deja un 0.

<# Se usa para la salida numérica de datos ajustando el formato a voluntad usando las palabras:

< # # #s sign #>

<BUILDS C,LO

—Se usa en una definición.

: cccc < BUILDS ….

— DOES> ... ;

Cada vez que se ejecute cccc, BUILDS define una nueva palabra con un proceso de ejecución de alto nivel. Ejecutando cccc nnnn usa < BUILDS para crear una palabra en el diccionario para nnnn con una llamada a la parte DOES > para nnnn. Cuando se ejecute nnnn más tarde, tiene la dirección de su área de parámetros en el stack y ejecuta las palabras después de DOES > en cccc.

<BUILDS y DOES > permiten escribir procesos inmediatos en un nivel alto en vez de en ensamblador.

= n1 n2 - - - F LO

—Deja en el acumulador 1 si n1 = n2; si no, 0.

> n1 n2 - - - F LO

—Deja un uno si n1 es mayor que n2; si no, 0.

>R —Coge un número del stack y lo coloca en la parte superior del stack de RETURN. Se debería escribir R > en la misma definición, cuando se use esta palabra.

? addr - - - LO

—Imprime el contenido de la dirección en formato libre y de acuerdo con la base seleccionada.

? COMP Se presenta un mensaje de error si no está compilando.

? CSP —Se presenta un mensaje de error si la posición del stack es diferente del valor de CSP.

?ERROR f n - - -

—Presenta un mensaje de error número n, si el valor del indicador lógico f es verdad (=1)

?EXEC Presenta un mensaje de error si no está ejecutando.

?LOADING Presenta un mensaje de error si no está cargando desde la cinta.

?PAIRS n1 n2 - - -

—Presenta un mensaje de error si n1 no es igual a n2. El mensaje indica que los condicionales compilados no se cumplen.

?STACK Presenta un mensaje de error si el stack está fuera de los límites. Esta definición depende del sistema que se esté usando.

@ addr - - - n LO

—Deja en el stack el contenido de 16 bit de la dirección especificada.

ABORT —Borra el stack y coloca el ordenador en modo de ejecución. Retorna el control al terminal del operador, imprimiendo un mensaje dependiendo de la instalación que se esté usando.

ABS n - - - u LO
—Deja en el stack como u el valor absoluto de n.

AGAIN —Se usa en una definición en la forma:
BEGIN ... AGAIN.
Cuando se ejecuta el programa, AGAIN fuerza un salto al BEGIN correspondiente. El stack permanece inalterado. No se puede abandonar el bucle hasta que R > DROP sea ejecutado un nivel más abajo.

ALLOT n --- LO
—Suma un número con signo al puntero del diccionario DP. Puede usarse para reservar espacio en el diccionario o para reiniciar la memoria.

AND n1 n2 - - - n3 LO
—Deja en el stack el resultado de una operación lógica AND entre n1 y n2 bit a bit. como n3.

AT n1 n2 - - -
—Posiciona el cursor de impresión en la pantalla en la línea n1, columna n2.

B/SCR - - - n
-Esta constante deja el número de bloques de la página que se está editando. Por convención, una página consta de 1024 bytes organizados como 16 líneas de 64 caracteres cada una.

BACK addr - - -
—Calcula el salto hacia atrás empezando en HERE hasta addr y compila en la siguiente libre del diccionario.

BASE addr U,LO
—Variable del sistema que contiene la base numérica seleccionada para la entrada y salida de datos.

BEEP n1 n2 - - -
— Efectúa un BEEP con la duración n1 y el pitch n2.

BEGIN —Se usa en una definición como sigue:
BEGIN... UNTIL
BEGIN.... AGAIN
BEGIN ...WHILE.... REPEAT
Durante la ejecución, BEGIN marca el comienzo de una secuencia que ejecutará repetidamente. Sirve como un punto de retorno desde UNTIL, AGAIN O REPEAT. Cuando se ejecuta UNTIL, se producirá un salto a BEGIN si el dato superior del stack es falso (= 0). Con AGAIN y REPEAT el salto a BEGIN se producirá siempre.

BL - - - c
—Constante que deja en el stack el valor ASCII de un espacio.

BLANKS addr número - - -
—Llena un área de memoria empezando en addr con el número de espacios especificados.

BLK addr U,LO
—Variable del sistema que contiene el número del bloque que se está interpretando. Si es cero, la entrada se está produciendo desde el buffer de entrada de un terminal.

BLOCK n - - - addr LO
—Deja la dirección de memoria del buffer de bloques que contiene el bloque n. Si éste no se encuentra en la memoria, se cargará desde la cinta.

BORDER n - - -

—Cambia el color del borde de la pantalla al número n.

BRANCH —El proceso inmediato para salto incondicional. Se añade un incremento al puntero del intérprete para un salto hacia delante o hacia atrás.

BRIGHT n - - -
—Brillo de los caracteres. 0-normal. 1-brillante.

BYE -Vuelve al BASIC.

C! b addr - - -
—Guarda 8 bits en la dirección addr.

C, b - - -
—Almacena 8 bits en el siguiente byte libre del diccionario incrementando el puntero del diccionario.

C@ addr - - - b
—Deja en el stack los 8 bit contenidos en esa posición de memoria.

CFA pfa - - - cfa
—Transforma la dirección del campo de parámetros de una definición en la dirección de su campo de código.

CIRCLE n1 n2 n3 - - -
Dibuja un círculo de radio n1 en las coordenadas (n2,n3).

CLS —Borra la pantalla.

CMOVE "desde" "a" número de bytes - - -
—Transfiere la cantidad de bytes especificada empezando en la dirección "desde" a la dirección "a". El contenido de la dirección "desde" es transferido primero.

COLD — Procedimiento para ajustar el puntero del diccionario al mínimo y comenzar otra vez con ABORT. Puede llamarse desde el terminal para quitar un programa y comenzar de nuevo

COMPILE —Cuando se ejecuta una palabra conteniendo COMPILE, la dirección de ejecución de la palabra que sigue a COMPILE se copiará (compilada) en el diccionario. Esto permite manejar situaciones de compilación específicas, además de la compilación de una dirección de ejecución (que es lo que hace el intérprete).

CONSTANT n --- LO
—Una palabra de definición usada en la forma:
n CONSTANT CCCC para crear la palabra CCCC, cuyo contenido del campo de parámetros sea n.
Cuando se ejecute CCCC se pondrá el valor n en el stack.

CONTEXT - - - addr.
—Variable del sistema que contiene un puntero al diccionario en el que empezará la búsqueda.

COPY —Manda una copia de la pantalla a la impresora.

COUNT addrl - - - addr2 número de bytes. LO
—Deja la dirección del byte addr2 y el número de bytes de un texto que empieza en addr1. Se supone que el primer byte en addr1 contiene el número de bytes, y el texto en sí comienza con el segundo byte. Normalmente después de COUNT se usa TYPE.

CR —Sitúa el cursor al principio de la siguiente línea. LO

CREATE —Usado en la forma: CREATE cccc por palabras como CODE y CONSTANT para crear un encabezamiento en el diccionario para definición FORTH. El campo de código contiene la dirección del campo de parámetros de las palabras. La palabra nueva se crea en el diccionario seleccionado en ese momento.

CSP - - - addr U
—Variable del sistema que contiene temporalmente la posición del puntero del stack, para detectar los errores de compilación.

D+ d1 d2 - - - dsum
— Deja en el stack la suma de doble precisión de dos números de doble precisión.

D+- d1 n - - - d2
—Aplica el signo de n al número en doble ejecución d1, quedando éste como d2.

D. d - - - LI
—Imprime un número con signo a partir de un valor de 32 bits en complemento a dos. Los 16 bits más significativos se almacenan encima del stack. La conversión se realizará de acuerdo con la base numérica seleccionada. Sigue un espacio en blanco.

D.R dn - - -
—Imprime un carácter de doble precisión d, alineado a la derecha en un campo de n caracteres.

DABS d - - - ud
—Deja el valor absoluto de un número de doble precisión.

DECIMAL —Selecciona la base numérica decimal para entrada/salida. LO

DEF b1 b2 b3 ... b8 n1 - - - LO
—Define un carácter gráfico.

DEFINITIONS—Usado en la forma: L1
CCCC DEFINITIONS
Selecciona el vocabulario CONTEXT como el vocabulario actual. En el ejemplo, la ejecución del nombre del vocabulario cccc, lo puso como el vocabulario CONTEXT y la ejecución de DEFINITIONS hará ambos vocabularios especificados CCCC.

DIGIT cn1 - - - n2 tf (ok)
cn1 - - - ff (mal)
—Convierte el carácter ASCII c (usando la base n1) a su equivalente binario n2, acompañado de una bandera lógica verdadera (1). Si la conversión no es válida, deja sólo una señal lógica falsa (0).

DLITERAL d - - - d (ejecución)
d - - - (compilación)
—Si está compilando, compila un número de doble precisión del stack en un literal. La ejecución posterior de la definición que contiene este literal, la situará en el stack. Si está ejecutando, el número permanecerá en el stack.

DMINUS di - - - d2
—Convierte d1 a un número de doble precisión en complemento dos.

DO n1 n2 - - - (ejecuta) P,C2,LO
—Se usa en una definición de la siguiente forma:
DO... LOOP
DO... + LOOP
Durante la ejecución, DO marca el comienzo de una secuencia que se ejecutará repetidamente, y que se controla por el límite del bucle n1 y un índice con valor inicial n2. DO quita estos valores del stack. Cuando se alcanza LOOP el índice se incrementa en uno. Mientras el nuevo índice no alcance el valor o sobrepase el límite, se producirá un salto a DO. De otra forma, se descartan los parámetros del bucle y se continúa con la ejecución normal del programa, n1 y n2 se determinan durante la ejecución, y pueden ser el resultado de otras operaciones. En un bucle la palabra I, LOOP, + LOOP, LEAVE.

DOES> LO
—Palabra que define la acción inmediata en una palabra definida en un nivel alto. DOES > altera el campo de código y el primer parámetro de la nueva palabra para ejecutar la secuencia de direcciones de palabras compiladas que siguen a DOES > .

Se usa en combinación con < BUILDS. Cuando se ejecuta DOES > , comienza con la dirección del primer parámetro de la palabra nueva que se encuentra en el stack. Esto permite la interpretación usando este área o sus contenidos. Se usa normalmente con el ensamblador FORTH, conjuntos multidimensionales, y generación del compilador.

DP - - - addr U,L

—Variable del sistema, puntero del diccionario. Contiene la siguiente dirección libre por encima del diccionario. Su valor puede leerse con HERE y alterarse con ALLOT.

DPL - - - addr U,LO

—Variable del sistema que contiene el número de dígitos a la derecha del punto decimal en entradas de enteros de doble precisión. Puede usarse también para imprimir un punto decimal, en un formato definido por el usuario.

DRAW n1 n2 - - -

—Dibuja una línea desde la posición del cursor gráfico, n1 en dirección x y n2 en dirección y.

DROP n - - - LO

—Elimina el dato superior del stack.

DUP n - - - nn LO

—Duplica el valor que está encima del stack. Su uso depende del caso. Véase R.

ELSE —Se usa en una definición de palabra como sigue:
IF...ELSE...ENDIF
ELSE se ejecuta después de la secuencia de comandos que siguen a IF. Esto solo ocurre si se cumple la condición anterior a IF. ELSE hace que se omitan las instrucciones que hay entre ELSE y ENDIF. La ejecución se reanudará después de ENDIF. No tiene ningún efecto sobre el stack.

EMIT c - - -

—Imprime el carácter ASCII c en la pantalla.

EMPTY-BUFFERS —Marca todos los buffers de bloques como vacío, sin afectar necesariamente a su contenido. Se puede usar como un procedimiento de iniciación antes de usar el cassette por primera vez.

ENCLOSE addr c - - -
addr1 n1 n2 n3

— La búsqueda de texto que usa WORD. Se determina desde la dirección del texto addr1, y un carácter ASCII delimitador c, la cantidad de bytes que hay hasta el primer carácter no delimitador n1, la cantidad de bytes hasta el primer carácter delimitador después del texto n2, y la cantidad de bytes hasta el primer carácter no incluido. Este procedimiento no continuará tras un carácter ASCII cero, ya que éste se tratará como un delimitador incondicional.

END -Tiene el mismo significado que UNTIL. P,LO

ENDIF -Se usa en una definición como sigue:
IF...ENDIF
IF... ELSE… ENDIF
Durante la ejecución del programa. ENDIF simplemente indica el destino de un salto desde IF o ELSE. Marca el final de una estructura condicional.
THEN tiene el mismo significado que ENDIF. Véase también IF y ELSE.

ERASE addr n - - -

—Borra una zona de n direcciones a partir de la dirección addr con ceros.

ERROR n - - - IN BLK

—Ejecuta una notificación de error y restablece el sistema. Primero se examina WARNING. Si es uno, se imprime el texto de la línea n, de la página 4 del controlador (de discos) 0. Este número de línea puede ser tanto positivo como negativo, pero limitándose únicamente a la página 4. SI WARNING es 0 (es decir, no se tiene instalada unidad de discos) n se imprimirá simplemente como un mensaje de error. Si WARNING es —1, se ejecuta la palabra (ABORT), que ocasionará un ABORT en el sistema.
El usuario puede alterar con cuidado el significado de (ABORT). ZX-FORTH guarda en el stack los contenidos de IN y BLK para facilitar la localización del error. Al final se ejecutará QUIT.

EXECUTE addr - - -
—Ejecuta una notificación de error y restablece el sistema. Primero se **cuentre** (*sic. No acertamos a deducir lo que se pretendió escribir, porque la palabra carece de significado en español)* en el stack. La dirección del campo de códigos, también se llama la dirección de compilación.

EXPECT addr número - - - LO
—Transfiere caracteres desde el terminal a la dirección especificada, hasta que se reciba un "RETURN" (NEW/LINE) o se complete el número de caracteres especificados. Se añaden uno o varios ceros al final del texto.

FENCE - - - addr U
—Variable del sistema que contiene una dirección por debajo de donde FORGET no puede actuar. Para borrar (FORGET) por debajo de este punto es necesario alterar el contenido de FENCE.

FILL addr con b - - -
—Llena una cantidad de bytes a partir de addr con el byte b.

FIRST - - - n
—Constante que deja en el stack la dirección del buffer de bloques.

FLASH n1 - - -
—Define si un carácter es intermitente o uniforme.
0 - Uniforme 1 - Intermitente.

FORGET —Ejecutando de la forma: FORGET cccc. E,LO
Borra la definición con nombre cccc del diccionario y todas las que se han definido después. En ZX-FORTH se producirá un mensaje de error si los vocabularios CONTEXT y CURRENT no son los mismos.

FORTH P,L1
—Nombre del vocabulario inicial. La ejecución selecciona FORTH como el vocabulario CONTEXT. Hasta que no se seleccione otro vocabulario, todas las definiciones de palabras se añaden al vocabulario FORTH. FORTH es inmediato, por lo tanto se ejecuta durante la creación de una palabra, para seleccionar este vocabulario durante la compilación de la palabra.

GOVER n - - -
—Controla la sobreimpresión de caracteres.
n = 0 El carácter es borrado por otros caracteres que se imprimen encima.
n = 1 Lo que se imprime es el resultado de un OR entre el carácter que había y el que se va a imprimir.

HERE - - - addr LO
—Deja en el stack la dirección de la siguiente posición libre del diccionario.

HEX LO
—Coloca la base numérica de conversión en 16 (hexadecimal).

HLD - - - addr LO

—Una variable del usuario que contiene las posiciones de memoria del último carácter de texto durante la conversión de salidas numéricas.

HOLD - - - addr

—Se usa entre < # y # > para insertar un carácter ASCII en una cadena de formato ajustable; por ejemplo:

2E HOLD pondrá un punto decimal.

HOME LO

—Sitúa el cursor en la esquina superior izquierda de la pantalla.

I - - - n C,LO

—Se usa en un bucle DO-LOOP para copiar en el stack el índice del bucle.

ID addr - - -

—Imprime el nombre de una definición a partir de su dirección en el campo de nombres. Otros usos dependen de la implementación. (Véase R)

IF f - - - (ejecución) P,C2,LO

—Se usa en una definición como:

IF (tp)… ENDIF

IF (tp)…ELSE (fp). .. ENDIF

Durante la ejecución, IF selecciona dónde se continúa con la ejecución dependiendo de un indicador lógico. Si f es verdad (no cero), la ejecución continúa normalmente con la secuencia de instrucciones verdaderas (tp).

Por el contrario, si f es cero (falso), la ejecución continuará a partir de ELSE para ejecutar la parte falsa (fp). Después de cualquiera de estos dos casos, la ejecución continuará a partir de ENDIF. ELSE y su secuencia de instrucciones "falsas" son opcionales. Si faltan, la ejecución en caso de que no se cumpla I condición seguirá tras ENDIF.

IMMEDIATE —Marca la última definición que se escribió para que cuando se encuentre durante la compilación, se ejecute inmediatamente en vez de compilarse, esto es, se pone a valor lógico 1 el bit de prioridad del puntero de esta palabra. Este método permite insertar en definiciones situaciones que no se pueden compilar. El usuario puede forzar la compilación de una palabra declarada inmediata precediéndola de la palabra (COMPILE).

IN - - - addr LO

—Variable del sistema que contiene la cantidad de bytes del buffer de texto actual (terminal o disco), del que se aceptará el siguiente texto. WORD usa y cambia el valor de IN.

INK n1 - - -

—Define el color de la tinta de los caracteres

INTERPRET

— El intérprete de texto exterior que compila o ejecuta secuencialmente texto desde un periférico de entrada (terminal de teclado o disco) dependiendo de STATE. Si la palabra no se encuentra después de buscar en CONTEXT y CURRENT, se convierte en un número de acuerdo con la base numérica seleccionada. Si esto falla, se imprimirá un mensaje con un signo de interrogación. La entrada de texto se hará de acuerdo con la convención para WORD. Si se encuentra un punto decimal como parte de un número, se dejará un número de doble precisión.

El punto decimal solamente tiene la misión de forzar esta acción. Véase NUMBER.

INV n - - -

—Controla la Inversión de los caracteres.
n = 0 normal n= 1 caracteres inversos

KEY - - - c LO
—Deja en el stack el valor ASCII de la siguiente tecla que se pulse.

LATEST - - - addr
—Deja en el stack la dirección del campo de nombres de la palabra superior del vocabulario CURRENT.

LEAVE C,LO
— Fuerza la salida de un bucle DO-LOOP, en la siguiente oportunidad asignando al límite del bucle el valor del índice de ese bucle. El valor del índice permanece inalterable, y la ejecución continúa normal hasta encontrar LOOP o +LOOP.

LFA pfa - - - lfa
—Convierte la dirección del campo de parámetros de una definición del diccionario en su dirección del campo de enlaces.

LIMIT - - - n
—Constante que deja en el stack la dirección por encima de la memoria más alta que puede usarse para buffer de la cinta. Normalmente es la última dirección de memoria del sistema.

LIST n - - - SCR,LO
—Presenta en la pantalla el texto de la página número n.
Contiene el número de esta pantalla durante y después de este proceso.

LIT - - - n C2,LO
—Es una definición de palabra. LIT se compila automáticamente antes de cada número de 16 bit que se encuentre en el texto de entrada.
La ejecución posterior de LIT provoca que el contenido de la siguiente dirección del diccionario se sitúe en el stack.

LITERAL n - - - (compilación) P,C2,LO
—Si se está compilando, entonces compila el valor del stack n como un número de 16 bit. Esta definición es inmediata, por lo tanto se ejecutará durante la compilación de una palabra. El uso para el que está pensada es:
:xxx (calculo) LITERAL;
La compilación se suspende para realizar el cálculo de un valor. A continuación se reanuda la compilación, y LITERAL compilará este valor del stack.

LOAD n - - - LO
—Comienza la interpretación de la página n. La carga se terminará con ;S. (Véase ;S y - - >)

LOOP —Se usa en una definición de palabra como:
DO... LOOP
Durante la ejecución, LOOP decide si saltar al DO correspondiente, basándose en el índice del bucle y en el límite. El índice del bucle se incrementa en uno y se compara con el límite. El salto a DO se produce hasta que el índice del bucle sea igual o sobrepase al límite: entonces se descartan los parámetros del bucle y se continúa con la ejecución normal del programa.

M* n1 n2 - - - d
—Operación matemática mixta que deja el producto de precisión doble con signo de dos números con signo.

M/ d n1 - - - n2 n3
—Operación matemática que deja el resto n2 y el cociente n3, de un dividendo de doble precisión y un divisor n1. El resto tiene el signo del dividendo.

M/MOD vd1 v2 - - - v3 v4

—Calcula el cociente de precisión doble vd4 y el resto v3, de un dividendo doble vd1 y un divisor normal v2.

MAX n1 n2 - - - max LO
—Deja en el stack el mayor de dos números.

MESSAGE n…
—Imprime ?MSG # n

MIN ni n2 - - - min LO
—Deja el menor de dos números.

MINUS n1 - - - n2 LO
—Deja el complemento a dos de un número.

MOD n1 n2 - - - mod LO
—Deja el resto n1/n2, con el mismo signo que n1.

NEXT —Se usa con el ensamblador FORTH.

NFA pfa - - - nfa
—Convierte la dirección del campo de parámetros de una definición en su campo de nombres.

NUMBER addr - - - d
—Convierte una cadena de caracteres que se encuentra en addr, y precedida de su longitud, en un número doble con signo, usando la base numérica seleccionada. Si se encuentra un punto decimal en el texto la posición se dará en DPL, pero no se producirá ningún otro efecto. Si no es posible la conversión numérica, se imprimirá un mensaje de error.

OR n1 n2 - - - or LO
—Deja el resultado de una operación or bit a bit.

OUT - - - addr
—Variable del sistema que contiene un valor incrementado por EMIT. El usuario puede alterar y examinar OUT para controlar el formato de salida.

OVER n1 n2 - - - n1 n2 n1
—Copia el segundo valor del stack y lo sitúa encima del stack.

PAD - - - addr
—Deja en el stack la dirección del buffer de texto de salida, que está situada una cantidad de bytes fija por encima de HERE.

PAPER n - - -
—Controla el color del papel (del fondo).

PERM - - -
-Hace que todos los colores temporales sean permanentes.

PLOT n1 n2 - - -
—Imprime un punto en las coordenadas (n1,n2) y mueve la posición de PLOT.

PFA nfa - - - pfa
—Convierte la dirección del campo de nombres de una definición compilada en su dirección del campo de parámetros.

QUERY —Introduce 80 caracteres de texto (o hasta un return) desde el terminal (teclado). El texto se almacena en la dirección contenida en TIB con IN puesto a cero.

QUIT LI
Borra el stack de return, para la compilación, y devuelve el control al terminal. No se imprime ningún mensaje.

R - - - n
—Copia el dato encima del stack de Return en el stack normal de computación.

R# - - - addr

—Variable del sistema que puede contener la posición de un cursor de edición, u otra función referente a ficheros.

R/W adr blk f - - -
—Es la unión entre el FORTH y un controlador de disco.
addr especifica la fuente o el destino de un buffer de bloques, blk es el número secuencial del bloque, y f indica: 0 - escribir, 1 - leer. R/W determina la localización en el disco, lee o escribe en éste y comprueba si hay algún error.

R > LO
Coge el primer valor del stack de Return y lo deja en el stack normal de computación. Véase >RyR.

RO - - - addr U
—Variable del sistema que contiene la dirección inicial del stack de Return. Cero. Véase RP!

REPEAT P,C2
—Se usa en una definición de palabra como: BEGIN...WHILE... REPEAT
Durante la ejecución, REPEAT provoca un sallo incondicional detrás del BEGIN correspondiente.

RSMUDGE Pone a cero el bit "smudge" de la última entrada del diccionario.

ROT n1 n2 n3 - - - n2 n3 n1 LO
—Rota los tres valores superiores del stack, llevando el tercero arriba.

RP! -Procedimiento dependiente del ordenador para inicializar el puntero del stack de return desde la variable del sistema RO.

S—> D n - - - d
—Transforma un número normal en uno de doble precisión.

S0 - - - addr U
—Variable del sistema que contiene el valor inicial del puntero del stack. Véase SP!

SCR - - addr U
—Variable del sistema que contiene el número de pantalla que se utilizó más recientemente con LIST.

SIGN nd - - - d LO
—Almacena un signo ASCII "—" justo delante de una cadena de salida numérica en el buffer de salida de textos cuando n es negativo; n se descarta, pero el número de precisión doble permanece inalterado. Se tiene que usar entre < # y # > .

SMUDGE —Se usa durante la definición de palabras para muestrear el bit "smudge" en el campo de nombres de definiciones.
Esto impide que una definición incompleta se encuentre durante la búsqueda en el diccionario, hasta que se compile sin error.

SP! —Procedimiento para inicializar el puntero del stack de SO.

SP@ - - - addr
—Procedimiento para colocar la dirección de la posición del stack en la parte superior de éste como estaba antes de ejecutarse SP@ (Esto es 12SP @ @ ... imprimirá 221).

SPACE —Imprime un espacio ASCII. LO

SPACES n - - - LO
—Imprime n espacios.

STATE - - - addr LO,LU
—Variable del sistema que contiene el estado de compilación, un valor distinto de cero indica compilación. El valor depende del caso.

SWAP n1 n2 - - - n2 n1 LO
—Intercambio los dos valores superiores del stack.

TASK -Palabra que no ejecuta nada. Puede usarse para marcar los límites entre programas. Borrando (FORGET) TASK y volviendo a compilar, se pueden descartar todas las palabras definidas para una determinada aplicación.

THEN Tiene el mismo significado que ENDIF. P,CO,LO

TIB - - - addr U
—Variable del sistema que contiene la dirección del buffer de entrada desde el teclado.

TOGLE addr b - - -
—Complementa el contenido de addr con la muestra b.

TRAVERSE addr 1n - - - addr2
—Desplazamiento a lo largo del campo numérico de la longitud de una variable FORTH. Addr1 es la dirección del byte de longitud o de la última letra. Si n = 1 el desplazamiento será hacia adelante en la memoria. Si n = —1 el desplazamiento será hacia atrás. El resultado addr2 es la dirección del otro extremo del nombre.

TYPE addr count - - -
—Transmite count caracteres empezando en la dirección addr en la pantalla.

U* v1 v2 - - - vd
—Calcula el producto de doble precisión sin signo de dos números simples sin signo.

U/ ud v1 - - - v3 v4
—Calcula el resto sin signo v2 y el cociente sin signo v3 de un dividendo de doble precisión sin signo ud y un divisor v1 simple sin signo

UNTIL f - - - (en ejecución)
—Se usa en una definición de palabra como:
BEGIN…UNTIL
Durante la ejecución. UNTIL controla un salto condicional al BEGIN correspondiente.
Si f es falso (= 0), la ejecución continúa justo después de BEGIN; si es verdad. (¥ 0). la ejecución continúa detrás de UNTIL.

USER n - - - LO
—Palabra usada en la forma:
n USER cccc
que crea una variable del usuario cccc. El campo de parámetros de cccc contiene n a una distancia fija al registro del puntero del usuario UP para esta variable.
Cuando se ejecuta cccc más tarde, sitúa la suma de esta distancia y la dirección de la base del área del usuario en el stack como la dirección de almacenamiento de una variable particular.

VARIABLE —Palabra usada como: E,LO
n VARIABLE cccc
Cuando se ejecute VARIABLE, crea las definiciones cccc con su campo de parámetros inicializado a n. Si se ejecuta cccc más tarde, se deja en el stack la dirección de su campo de parámetros, de tal manera que se pueda almacenar o leer un dato de esta posición.

VOC-LINK - - - addr U
-Variable del sistema que contiene la dirección de un campo en la definición del último vocabulario que se creó. Todos los nombres de un vocabulario están unidos mediante estos campos para permitir realizar un FORGET por varios vocabularios.

VOCABULARY —Palabra usada de la siguiente forma: E,L
VOCABULARY cccc
—Para crear una definición de vocabulario cccc.

El uso posterior de cccc hará que éste sea el vocabulario CONTEXT, que es donde busca primero el intérprete
La secuencia "cccc DEFINITIONS" también seleccionará cccc como el vocabulario CURRENT, donde se situarán las nuevas definiciones.
En ZX-FORTH, cccc está encadenado de tal forma que incluye todas las definiciones del vocabulario, en las que cccc se ha definido. Todos los vocabularios están encadenados al FORTH. Por convención, los nombres de vocabulario deben declararse inmediatos (INMEDIATE). Véase VOC-LINK.

VLIST —Lista los nombres de las definiciones del vocabulario CONTEXT. "BREAK" (shift SPACE), interrumpe el listado.

WARNING —Debe tener el valor 0 para una instalación con disco. P,C2

WHILE f - - - (en ejecución)
—Se usa en una definición de palabra.
BEGIN ... WHILE (tp)... REPEAT
Durante la ejecución del programa, WHILE, selecciona dónde se continuará la ejecución dependiendo de la señal lógica f. Si f es verdad (¥ 0), WHILE provoca que la ejecución continúe en la parte verdadera hasta llegar a REPEAT, y de ahí salta a BEGIN. Si f es falso (= 0) la ejecución salta detrás de REPEAT, saliendo de la estructura.

WIDTH —longitud máxima de una palabra. 31 en el ZX-FORTH.

WORD c - - -
—Lee los siguientes caracteres de texto desde una fuente de entrada que se está interpretando hasta que encuentra un delimitador c. Almacena la cadena de caracteres en el buffer del diccionario empezando en HERE. WORD deja el número de caracteres en los dos primeros o más espacios en blanco.
Las apariciones posteriores de c son ignoradas.
Si BLK es cero, el texto se coge del buffer de entrada del teclado, si no, se cogerá del bloque almacenado en BLK. Véase BLK y IN.

XOR n1 n2 - - - XOR
—Deja el resultado de una operación lógica or exclusiva bit a bit.

[—Se usa en una definición de palabra como sigue:
: xxx [palabras] continuación de la definición;
Suspende la compilación. Las palabras detrás de [se ejecutan y no se compilan. Esto permite hacer cálculos que no se incluyen en la definición hasta que se reanude la compilación con]. Véase LITERAL.

COMPILE —Se usa en una definición de palabra.
: XXX [COMPILE] FORTH;
[COMPILE] fuerza la compilación de una definición declarada como inmediata. El ejemplo anterior seleccionará el vocabulario FORTH cada vez que se ejecute XXX, en vez de hacerlo en el momento de la compilación.

] —Reanuda la compilación para completar una definición de palabra.

MANUAL DEL EDITOR

El Forth organiza su almacenamiento de datos en "pantallas" de 1024 caracteres. El Forth sólo permite que haya 1 pantalla en la memoria para el almacenamiento de texto. Las pantallas son números del 0 al 32768. Cada pantalla está dividida en 16 líneas de 64 caracteres por línea. Las pantallas Forth son meramente una organización de la memoria y no corresponden al formato de la pantalla del ZX-81 o el ZX-Spectrum.

SELECCION DE UNA PANTALLA Y ENTRADA DE TEXTO

Para empezar la edición, el usuario debe teclear EDITOR para invocar el vocabulario adecuado. A la pantalla se le da un número. Esto se hace así nCLEAR (borrar la pantalla n y prepararla para edición).

Para la entrada de nuevo texto en la pantalla n después del CLEAR, se usa el comando P (poner)

Ejemplo:

```
0  P  ESTA ES LA MANERA
1  P  DE COLOCAR TEXTO
2  P  EN LAS LINEAS 0, 1, 2 DE LA PANTALLA SELECCIONADA
```

EDICION DE LINEAS

Durante esta descripción del editor, se hace referencia al PAD. Esto es un buffer de texto que puede contener una línea de texto que haya que encontrar o borrar con un comando de edición de una cadena.

COMANDOS DE EDICION DE LINEAS

n D	Borrar la línea n pero almacenarla en el PAD. La línea 15 queda libre puesto que todas las cadenas almacenadas en líneas posteriores a la línea n se mueven una línea hacia arriba.
n E	Borrar la línea n con espacios.
n I	Insertar el texto almacenado en el PAD en la línea n, moviendo la antigua línea n y las posteriores hacia abajo. La línea 15 se pierde.
n H	Retener la línea n en el PAD (este comando es usado por el sistema más que por el usuario).
n R	Reemplazar la línea n con el texto que hayen el PAD.
n S	Extender en la línea n la antigua línea n y las siguientes líneas se mueven una línea hacia abajo. La línea n se queda vacía. La línea 15 se pierde

COMANDOS DE EDICION DE PANTALLAS

n CLEAR	Borrar la pantalla con n espacios y seleccionarla para edición.
FLUSH	Se usa al final de una sesión de edición para grabar la pantalla en una cinta.
nL	Listar la pantalla con la que se está trabajando. La línea del cursor se lista al final del listado de la pantalla para mostrar la posición del cursor.
n LIST	Listar la pantalla n y seleccionarla para edición si la pantalla n no es la que hay en este momento en memoria, el sistema intentara cargarla del cassette.

CONTROL DEL CURSOR Y EDICION DE CADENAS

La pantalla de texto que esté siendo editada reside en el área de un buffer de almacenamiento. El cursor de edición es una variable que guarda un offset en este área de buffer. El usuario utiliza unos comandos para posicionar el cursor, bien directamente o buscando una cadena de texto en el buffer, y para insertar o borrar texto en la posición del cursor.

COMANDOS PARA POSICIONAR EL CURSOR

n M	Mover el cursor n caracteres. La posición del cursor en la línea se muestra con el carácter.
TOP	Colocar el cursor al principio de la pantalla.

COMANDOS DE EDICION DE CADENAS

B	Usado después de F para volver al cursor atrás la longitud del texto más reciente
C text	Copiar el texto situado en la posición del cursor a la línea del cursor.

F text — Buscar a partir de la posición actual del cursor hasta que la cadena "texto" sea encontrada. El cursor se queda al final de la cadena y la línea del cursor es impresa. Si la cadena no es encontrada aparece un mensaje de error #0 y el cursor es colocado al principio de la pantalla.

N — Encontrar la siguiente aparición de la cadena encontrada por un comando "F".

TILL text — Borrar en la línea del cursor desde el cursor hasta el final de la cadena "text".

X text — Buscar y borrar la siguiente aparición de la cadena "text".

*NOTA: Teclear C sin **texto** copiará un nulo en el texto en le posición del cursor. Esto hará que se interrumpa abruptamente la compilación más tarde. Para arreglar este error, teclee TOP X "N/L".*

GLOSARIO DEL EDITOR

#LAG — - - - cursor - cuenta de posición de memoria - después - cursor - hasta que - EOL.

#LEAD — - - - línea - offset de posición de memoria - a - el cursor.

#LOCATE — - - - ni n2
Desde la posición del cursor fijar la línea - número n2 y el offset en la línea n1,

R# — - - - addr
Una variable del usuario que contiene el offset del cursor de edición desde el principio de la pantalla.

B — - - -
Retroceder el cursor el número de caracteres almacenados en el PAD.

C — - - -
Extender en la posición del cursor y copiar el siguiente texto en la línea del cursor n.

CLEAR — n - - -
Vaciar la pantalla n. puede usarse para seleccionar la pantalla n para edición.

D — n - - -
Borrar la línea n pero guardarla en el PAD.

DELETE — n - - -
Borrar los n caracteres anteriores al cursor.

E — n - - -
Borrar la línea n con espacios.

F — - - -
Colocar el texto siguiente en el PAD y buscar uno igual desde la posición del cursor hasta el final de la pantalla.

FIND — - - -
Buscar una cadena igual a la almacenada en el PAD, desde la posición del cursor hasta el final de la pantalla. Si no se encuentra ninguna, aparece un mensaje de error y se coloca el cursor al principio de la pantalla.

FLUSH — - - -
Grabar la pantalla actualizada en la cassette.

H — n - - -
Guarda la línea de número n en el PAD.

I — n - - -
Extiende en la línea n e inserta desde PAD.

L — - - -
Listar la pantalla en memoria.

LINE — n - - - addr
Dejar la posición de memoria de la línea n en la pantalla.

1 LINE - - - f
Examinar la línea de cursor, buscando un texto igual al del PAD. Devolver la bandera f y actualizar la variable R# del cursor al final del texto igual al del PAD, o al principio de la siguiente línea si no se encuentra ningún texto igual.

M - - -
Mover el cursor un número positivo o negativo de espacios e imprimir su línea.

MATCH Cursor - addr bytes a la izquierda - hasta que - EOL str - addr str - contar - tf cursor - avanzar - hasta - el final de - comparación – texto - ff bytes - izquierda - hasta – EOL
Comparar la cadena en la posición de memoria str con todas las cadenas en la línea de cursor a partir del cursor. Los argumentos que quedan permiten al R# del cursor que se actualice bien al final de un texto igual o al principio de la siguiente línea.

-MOVE - - -
addr número de línea.
Mover una línea de texto desde la posición de memoria addr a la línea actual de pantalla.

N ---
Encontrar la siguiente aparición del texto en el PAD.

P n - - -
Poner el texto que sigue en la línea n.

R n - - -
Reemplazar la línea n con el texto en el PAD.

S n - - -
Extender. La línea n y las siguientes se mueven para abajo, n queda vacía.

T n - - -
Imprimir la línea n y guardarla en el PAD.

TEXT c - - -
Aceptar el texto siguiente en el PAD. c es el delimitador de texto.

-TEXT - - -
Addr1 cuenta addr2 - - boleana
(Verdadero si concuerdan las dos cadenas).

TILL - - -
Borrar en la línea de cursor desde el cursor hasta el final del texto que sigue.

TOP - - -
Colocar el cursor de edición al principio de la pantalla.

WHERE n1 n2 - - -
n2 es el numero de bloque. n1 es un offset en el bloque. Si se encuentra un error en la fuente cuando se está cargando desde el cassette la rutina de recuperación ERROR deja estos valores en el stack para ayudar al usuario a localizar el error. WHERE usa estos valores para imprimir la pantalla y la línea y un esquema de dónde se dio el error.

X - - -
Borrar la siguiente aparición del texto que sigue.