# HL ZX FORTH

Instrucciones de operación del programa SP48 (Versión 2).

## 1.- Puesta en marcha.

La cinta que se adjunta tiene una copia del HL ZX FORTH en cada cara. Ha sido producida por una firma profesional de duplicación. Si tiene agún problema con ella, devuelvamela para reviserla.
Cargue el HL ZX FORTH con LOAD "". Espere hasta que aparezca la petición ***). Esto indica modalidad de comando, y ya esta preparado para empezar.

## 2.- La palabra.

El FORTH es un lenguaje extraño. Trabaja completamente bajo el concepto de PALABRA. Todo en FORTH es una palabra. Usted puede pensar que hay números, símbolos y nombres de variable, pero realmente todo lo que hay son palabras. El principio de programación en FORTH es que usted le enseña al computador nuevas palabras para realizar nuevas cosas.
En un diccionario, una palabra que usted no conoce está definida en palabras que ya conoce. En FORTH, usted define nuevas palabras al computador en términos de palabras que él ya conoce.

Entre cada palabra debe haber una palabra separadora. Puede ser o un SPACE o un ENTER. Tienen exactamente el mismo efecto en todo momento en lo que concierne al computador. Con un poco de experiencia, usted habrá decidido cual prefiere utilizar y cuando. SPACE y ENTER son completamente intercanbiables.

## 3.- El Stack.

El stack es la característica más intersante del FORTH después de la palabra. Es como una pila de platos - el último plato puesto es el primero para sacar - pero consiste en números, no platos.
La palabra FORTH que pone un número dentro del stack es cualquier cosa que se parece a un número (un entero entre - 32768 y 32767). La palabra FORTH que saca un número fuera del stack y lo escribe se llama dot (punto), y parece sospechosamente como un stop total - realmente lo es.
Desde modalidad de comando, ponga un número dentro del stack. Escriba seis espacio:punto y coma enter. (El punto y coma marca el final de la instrucción). Ahora usted debe tener un elemento dentro del stack. Ponga media docena más de elementos solo por entretenimiento. Usted los puede poner todos a la vez simplemente separando los números con espacios o ENTERs. Acuerdese del punto y coma al final.

Ahora saquemos un número fuera. Dot espacio punto y coma enter escribe el último número que usted a puesto dentro. Recuerde, ultimo en entrar, primero en salir. Intente sacar todos los demas también.

3.1-Cálculos.

Hay varias palabras de cálculo. Por ejemplo, la palabra + (mas) saca el elemento de mas arriba (top-off-stack o TOS) y el siguiente elemento por debajo (second-on-stack o 2OS) y lo suma, dejando el resultado en el stack.Pruebe

```
***)6 2 ;
***)+ ;
***). ;          (no teclee ***).
```

y usted debe obtener 8.

Intente ahora en una sola línea

```
***)6 2 + . ;
```

y ahora solamente como práctica,

```
***)45 100 + . ;
***)100 45 + . ;
***)45 100 - . ;
***)100 45 - . ;
***)1024 2 / . ;
***)111 -7 * . ;
***)12 16 13 2 + + + . ;
***)1000 100 1 - + . ;
```

4.- Tabús.

No haga nada de lo que sigue ! En serio, tiene que tener mucho cuidado con lo que está haciendo o el computador le hará algún disparate. No debe utilizar sintaxis erronea o parámetros fuera de límites. No debe poner paréntesis en medio de comillas o comillas o parentesis dentro de parentesis. No debe tener mas de 800 elementos en el stack a la vez. No debe tener mas de 1000 palabras nuevas en el computador al mismo tiempo. La dirección de compilación (ver mas adelante) no debe exceder de 51000. No escriba nada después de utilizar funciones de gráficos de alta resolución sin haber utilizado antes AT. No defina ninguna palabra nueva con nombres empezando con el signo menos o un número.

5.- Almacenamiento.

Usted almacena sus palabras FORTH haciendo una nueva copia del HL ZX FORTH. En modalidad de comando, entre la letra minúscula s y dé enter. Entre un nombre de fichero (menor de diez letras, no espacios)como se pide y dé enter. Aparece la nota de copyright y entonces el programa se almacena en tres bloques - tendrá que pulsar enter al principio de cada uno de ellos.

Cuando se ha terminado el almacenamiento, el programa le pide que rebobine la cinta para verificar. Si ha ido bien, volvemos a modalidad de comando. Si falla, deberá BREAK y GO TO 2010 para volver a almacenar.

6.- Gráficos definidos por el usuario.

Hay 256 UDGs disponibles, con los números de 0 a 255. Pulse la tecla minúscula d (para definir) desde modalidad de comando. Dé los números, como se pide. Los números 32 al 127 estan disponibles en el teclado, el resto solamente desde FORTH. Se le pedirán ocho líneas de definición, y cada una debe estar compuesta exactamente por ocho unos y ceros, indicando negro y blanco en la definición. Si dispone de impresora se escribirá la definición completa. Entonces volvemos a modalidad de comando.

7.- Definición de palabras propias.

Desde modalidad de comando, teclee ~~una coma~~ (:) (pulse enter). Dé nombre a su palabra, como se pide. Puede ser teoricamente de cualquier longitud, pero solamente cuentan los seis primeros caracteres. No debe comenzar con el signo menos o con un número. Pruebe TEST como ejemplo. La palabra aparece en la parte superior de la pantalla con su direccion de compilación. Ahora usted define la palabra en terminos de palabras FORTH que la máquina ya conoce - es exactamente igual que entrar un comando directo. Como todavía no conocemos demasiado FORTH, deberá ser mas bien una palabra sin utilidad, pero pruebe.

6 2 + . ;  (pulse enter)

La impresora (si la tiene) escribirá la palabra. Si no tiene impresora quizá quiera apuntarla. Si tiene una, quizá desee BREAK y GO TO 100 para ahorrar papel (esto también es aplicable a la impresión de UDGs).

Desde modalidad de comando, teclee TEST espacio punto y coma enter. El computador reconoce TEST. Ahora puede utilizar TEST en definiciones posteriores de palabras como si hubiera estado alli junto con todas, ej.

***): TEST2 TEST 0 . TEST ;

Cuando programe en FORTH, usted construye palabras en palabras como éstas hasta que su programa sea solamente una palabra.

Mientras define, se puede mantener la pantalla limpia de dos maneras. Primeramente una palabra -separador empieza una línea nueva, y en segundo lugar se pueden poner comentarios (como REMs en BASIC) entre parentesis, dentro de los cuales se permiten espacios.

8.- Borrado de palabras.

Se pueden borrar TEST y TEST2 tecleando, en modalidad de comando, la minúscula f, espacio, TEST, enter. F permite FORGET(olvidar). Solo hace falta olvidar

TEST ya que la instrucción FORGET borra la palabra nombrada y todas aquellas definidas a partir de ella. (dirección de compilación posterior)

9.- Mas palabras aritméticas.

Número. (number) Cualquier palabra que empiece por un número o un signo menos se evalúa y se pone en el stack.

+ - * / > < = Cada una de estas siete palabras toman el 2OS como primer operando y TOS como segundo operando. El resultado del cálculo/comparación se pone en el stack. El resultado de una comparación es 1 si se cumple, 0 si no se cumple.

/MOD Nos da el resultado y el resto de una división, resultado como TOS y resto debajo. 13 4 /MOD nos da un 3 y un 1 como resultados.

MOD Nos da el resto de una división, ej. 13/4=3 tiene de resto 1, por lo que 13 4 MOD da 1.

AND OR XOR Estas tres palabras son para combinar resultados de comparaciones, como en BASIC. AND da verdadero si TOS y 2OS son verdaderos; OR da verdadero si cualquiera de ellos lo es; XOR da verdadero solamente si los dos son diferentes uno del otro.

NOT Cambia TOS de verdadero a falso o viceversa. (1er bit)

NEGATE Niega TOS.

ABS Quita el signo menos de TOS, si lo tenía.

MAX Toma TOS y 2OS y reemplaza solo el mas grande.

MIN Toma TOS y 2OS y reemplaza solo el mas pequeño.

RND . Pone un número pseudoaleatorio entre 0 y 255 dentro del stack. Esto no funciona bien dentro de un bucle rápido.

10.- Utilización eficaz del stack.

Aqui tenemos algunas palabras que nos ayudarán a tener el stack bajo control.

DROP Elimina el TOS, haciendo el stack de un elemento más corto.

DUP . (duplicar) copia el TOS por lo que aparece en el stack dos veces haciendolo un elemento mas largo - pruebe 3 2 DUP . . . ;

?DUP Hace DUP solamente si el TOS es diferente de cero. En caso contrario no hace nada.

DEPTH . Nos da el número de elementos en el stack antes de poner depth. Pruebe DEPTH . ;

SWAP Permuta los dos elementos primeros del stack.
Pruebe 5 4 3 2 1 SWAP . . . . ;

OVER : ~~2 elemento~~ lo copia en TOS

PICK Saca el TOS-avo elemento del stack y lo copia, por lo que 2 PICK es lo mismo que OVER. Pruebe 500 400 300 200 100 ; después 3 PICK . ;

ROT (rotar) lleva 30S a la parte de arriba, y empuja el antiguo 20S y TOS hacia el hueco por lo que la profundidad del stack no cambia. Pruebe 5 4 3 2 1 ROT . . . . . ;

ROLL Es a ROT lo que PICK es a OVER. Lleva el TOS-avo elemento del stack a la parte de arriba y empuja cualquier elemento por encima del hueco hacia el hueco. Pruebe 7 6 5 4 3 2 1 6 ROLL . . . . . ;

## 11.- Estructuras FORTH.

El HL ZX FORTH tiene todas las estructuras de control del FORTH.

c IF
a ELSE
b THEN

IF es similar a su homónimo del BASIC, pero busca si es verdadero o falso en el stack, no siguiendo al IF como en BASIC. Entre IF y ELSE van todas las palabras para comprobar, si TOS era verdadero. Entre ELSE y THEN van todas las palabras para comprobar, si TOS era falso. Vea que la posición de THEN no es igual que en BASIC. Habiendo ejecutado o un conjunto de palabras o el otro, el computador ejecuta 'entonces' las palabras después de THEN.

c = condición
a = palabra si c = 1
b = " si c falso

Pruebe 5 5 = IF 444 . ELSE 333 . THEN ;
5 6 = IF 444 . ELSE 333 . THEN ;

Siempre que cualquiera de estas tres palabras aparezcan en una definición o comando, debe ir acompañada por sus dos compañeras. IF no es anidable.

a b DO
c LOOP
+LOOP

DO es mas bien como el FORT de BASIC. Toma TOS como primer valor de bucle y 20S como primer valor no bueno (no el último valor bueno como en BASIC). FOR z=1 TO 1000 se convierte en 1001. 1 DO.NEXT y es reemplazado por LOOP, que no necesita operandos. Este tipo de loop (y solamente éste) es anidable a cualquier nivel. La palabra I (para índice) siempre pone el valor actual de bucle mas interior dentro del stack. Pruebe

a = valor final + 1
b = " inicial
c = palabra a ejecutar.
I = índice.

1001 0 DO 23 0 AT I . LOOP ;
256 0 DO I EMIT LOOP ; (éste escribe el conjunto de caractares)

+LOOP puede ser utilizado en lugar de LOOP. Hace que el valor de I se incremente en TOS en lugar de 1. Pruebe

51 0 DO I . FIELD 5 +LOOP ;
1000 1 DO I . FIELD I +LOOP ;

En cualquier definición de palabra o comando directo, el número de DOs debe exactamente igualar el número de LOOPs y +LOOPs.

BEGIN
UNTIL — Es una clase diferente de loop. Las operaciones entre BEGIN y UNTIL se ejecutarán hasta que UNTIL encuentre un valor verdadero en TOS. Pruebe 100 BEGIN 1 - DUP . FIELD ?DUP 0 = UNTIL ; estos bucles no son anidables.

BEGIN
WHILE
REPEAT — Las operaciones entre BEGIN y REPEAT se ejecutan continuamente mientras que WHILE encuentra valores verdaderos en TOS. Tan pronto como WHILE encuentra un valor falso, el control sale fuera del loop a la palabra siguiente REPEAT. Pruebe
8192 BEGIN DUP 1 WHILE DUP . FIELD 2 / REPEAT DROP ;

LEAVE — Frozara una salida antes de tiempo desde un bucle DO haciendo I igual al valor mas alto del bucle.

EXIT — Hace que el computador deje de ejecutar la palabra/comando actual. No se debe utilizar dentro de un bucle DO.

EXITLP — Como EXIT, pero para uso sólamente dentro de bucles DO.

QUIT — Hace que el computador vuelva a modalidad de comando FORTH.

ABORT — Borra el stack y después hace QUIT.

WAIT — No tiene ningún efecto si mantiene pulsada la tecla Y. Si no la tiene pulsada, espera hasta que la pulsa, mostrando un bloque intermitente en la parte inferior derecha de la pantalla, que le recuerda que está esperando. Pruebe 1001 0 DO FIELD I . WAIT LOOP ;

12.- La pantalla y el altavoz.

SPACE — Adelanta en uno la posición de impresión.

SPACES — Adelanta la posición de impresión en TOS espacios. No utilice 0 SPACES.

EMIT — Imprime CHR$ (TOS).

. — (dot) Imprime el número TOS.

FIELD — Adelanta la posición de impresión al siguiente cuarto de la pantalla.

CR — (carriage return) Adelanta la posición a la línea siguiente.

CLS — Borra la pantalla.

."string" Imprime una tira de caracteres. Se permiten espacios.

AT — Mueve la posición de impresión a la línea 20S columna TOS ej.
10 15 AT ."Hello" ; se debe emplear después de palabras de alta resolución, siempre que se utilicen, para restaurar la posición de impresión.

ATTR — Convierte la línea número 20S y columna TOS en una dirección en el fichero de atributos tal que ATTR C es lo mismo que el ATTR BASIC.

FLASH
BRIGHT
INK
PAPER — Similar al equivalente en BASIC pero solamente cambia el estado de un solo espacio de caracter. El espacio alterado es la línea 20S columna TOS (como en AT) y el operando, que debe ser como un operando BASIC, es 30S. Pruebe
0 15 10 0 15 10 1 15 10 1 15 10 FLASH BRIGHT WAIT FLASH BRIGHT ; o
127 EMIT 6 23 0 INK 2 23 0 PAPER WAIT ;

BORDER — Como en BASIC pero tomando TOS como operando. Pruebe 3 BORDER WAIT ; o 3000 0 DO 8 0 DO I BORDER LOOP LOOP ;

PLOT
UNPLOT
POINT
OVRPLT — Igual que en BASIC, tomando (20S,TOS) como coordenadas (x,y); x está entre 0 y 255, y entre 0 y 191. OVRPLT escribe encima. POINT deja 1 o cero en el stack (similar al POINT de BASIC).

INVERSE — Emplea aproximadamente un segundo en invertir todo el conjunto de caracteres. Pruebe simplemente INVERSE ;

BEEP — Toma 20S como duración y TOS como tono de sonido. El mismo valor de tono siempre da el mismo tono, pero la duración depende del 20S y del tono; le llegará a ser familiar. Pruebe 1000 100 BEEP ; y para sonidos raros (o algo) ,pruebe 200 0 DO I BEEP LOOP ;

13.- Variables.

En el HL ZX FORTH, no es necesario declarar sus variables - hay disponibles 25 que tienen los nombres A,B,C...X,Y,Z excpto la I. Como los nombres de variables son realmente palabras, les puede dar otro nombre mediante SCORE A ; o algo similar. Para almacenar TOS en una variable, pulse letra espacio almacena,. donde almacena es un signo de exclamación. Por ejemplo, para poner 3000 en A, utilice 3000 A ! ;

(POKE) A !
(ADDR) A @
(PEEK) A ?
(ADD) A +!

Para leer una variable dentro del stack, utilice letra espacio búsqueda, donde búsqueda es el simbolo@ . Para leer A por consiguiente, utilice A@ . ;
Al igual que almacenamiento y busqueda, se puede utilizar lectura, que es un interrogante y escribe la variable, ej. A ? ;
También hay incremento que se escribe como +! (sin espacio en medio). Para sumar 627 a A, escriba 627 A +! ; o para restar 628 haga -628 A +! ;
Los nombres de variables son realmente palabras que ponen una dirección dentro del stack. Si usted no sabe lo que esta haciendo, puede poner una dirección en lugar de un nombre de variable en cualquiera de las funciones anteriores y que vienen a ser un poco como PEEK y POKE.

14.- Entradas.

INKEY — Pone en el stack el valor ASCII de la tecla que se esta pulsando, o 255 si no hay ninguna pulseda.

KEY Espera a que usted pulse una tecla y entonces pone su valor ASCII dentro del stack.

QUERY Entra caracteres desde el teclado en el área de notas (pad) de la memoria hasta que pulse la tecla enter. El caracter ENTER (13) se guarda para indicar el final de los datos. No entre más de 100 caracteres en respuesta a un query.

WORD Lee un caracter desde el pad en el stack, trabajando a lo largo del pad cada vez que se requiera.

PAD Da la dirección del pad.

>IN Da la dirección donde usted puede ver hasta qué WORD ha leído a lo largo del pad, por lo que >IN C@ da el número real de caracteres en palabras que se han leido hasta ahora.

15.- Programacion de bajo nivel.

No es necesario entender esta sección.
Los programadores de bajo nivel ya tienen AND,OR y XOR, que son operaciones de 16 bits. También @ y ! tienen muchas aplicaciones. C@ y C! son como @ y ! pero solo para bytes, P@ y P! tienen la misma sintaxsis, pero direcciones de ports, no bytes. C? es lectura de byte, y similar a ? (lectura). Otras palabras son

>R (a R) Trasfiere un elemento desde el stack de datos al stack de retorno.

R> (de R) Transfiere un elemento desde el stack de retorno al stack de datos.

R @ (busqueda en R) Copia el elemento más alto del stack de retorno en el stack de datos.

FILL Ocupa 20S bytes con el valor de TOS empezando en la dirección 30S.

0 FILL <> m n ERASE Ocupa TOS bytes con valor cero empezando en la dirección 20S.

32 FILL <> m n DELETE Ocupa TOS bytes con el valor 32 (blanco en ASCII) empezando en la dirección 20S.

CMOVE Copia TOS bytes desde la dirección 30S al destino 20S. (MOVE hace lo mismo para números de 2 bytes). Los bloques origen y destino no deben solaparse si el destino esta más alto en memoria que el origen.

TYPE Emite TOS caracteres encontrados en la dirección 20S.

CDUMP Escribe los valores de TOS bytes encontrados en la dirección 20S.

DUMP Escribe los valores de TOS números de 2 bytes encontrados en la dirección 20S.

prstr4:

FIND Da la dirección de compilación de la siguiente palabra sin ejecutar esta palabra. Busque solamente sus propias palabras. Un apóstrofe puede ser utilizado como una abreviatura para FIND. Ej.' NAME . ; escribiría la dirección de compilación de NAME.

HIRES Toma las coordenadas de alta resolución (2OS,TOS) y coloca el DF_CC para apuntar al byta que contiene este pixel, dejando 8 veces el número de bit en el stack de datos.

wrdsch busca la palabra cuyo nombre está guardado en 6 bytes en la posición 23264,en el diccionario. Devuelve la dirección de la entrada del diccionario (o cero para no encontrado) en la pareja de registros BC. Cada entrada del diccionario consiste en 6 bytes de nombre de palabra seguida de 2 bytes de dirección de compilación.

flgtst Lee sólamente el bit 0 de TOS y fija el flag Z adecuadamente. Utilizado por IF, WHILE, y UNTIL, por lo que los números pares se toman como falsos y los impares como verdaderos.

EXECUTE Provoca un salto a la dirección TOS.

EXPECT Entra hasta un número de TOS caracteres desde el teclado y los pone en la dirección 2OS. La entrada se puede terminar antes pulsando enter, en cuyo caso se almacena el valor 13.

COUNT Para utilizar con una tabla de bytes en la que el primer byta contiene la longitud de la tabla, y la dirección de la tabla en TOS. TOS se incrementa y la longitud de la tabla se coloca en la parte superior de ella en el stack.

STKSWP (stack swap). Utilizado por primera vez, hace que SP apunta al stack de datos. Otras veces, el SP apuntará al stack de retorno.

Las variables del sistema utilizadas son: DF_CC para la posición de print, 23681 para desplazamiento de PALABRA dentro del PAD, 23728 para guardar el puntero del stack que no está actualmente en SP.
Entrada de código máquina: como si fuera una palabra FORTH, en comando directo o definición, entre mc n1 n2 n3 n4 .....n?.end donde n1 etc. son bytes de código máquina en decimal. Puede utilizar enter en vez de espacios, pero NO el truco de un doble separador para empezar nueva linea.

16.- Tiempos del HL ZX FORTH

Debajo están los tiempos de 1000 operaciones en segundos. Los tiempos del Jupiter Ace y Spectrum se han tomado de las propias referencias del Jupiter. Los tiempos del HL ZX FORTH son según nuestras propias referencias. Como puede ver, el HL ZX FORTH es más rápido que el Jupiter Ace, excepto para el mane

jo de pantallas, que no es aplicable por tener pantalla de alta resolución.

| | JAce | Spec | HLZXF |
|---|---|---|---|
| Bucle vacio | 0.12 | 4.2 | 0.06 |
| Impresión de un número | 7.5 | 19. | 1.9 |
| Impresión de un caracter | 0.62 | 7.5 | 0.7 |
| Suma | 0.45 | 7.5 | 0.23 |
| Multiplicación | 0.9 | 7.5 | 0.5 |

17.- Mapa de Memoria.