C O M P I L A D O R S O F T E K

Los compiladores 'IS' y 'FP' se venden para crear software comercial y su uso debe anunciarse claramente en el mismo, no existiendo otra restrinción para su uso. Los compiladores Softek son potentes y de fácil uso.

INTRODUCCION

¿QUE ES UN COMPILADOR?

Un compilador transforma un programa escrito en BASIC en el mismo escrito en CODIGO DE MAQUINA, que en cierto sentido es la "lengua materna" del Spectrum, ya que él trabaja en este lenguaje. El BASIC lo comprende el Spectrum porque su ROM (Read Only Memory) contiene el llamado " BASIC interpreter" el cual está escrito en código de máquina.

El corazón del Spectrum es la CPU (Central Processing Unit) que no entiende el lenguaje BASIC. Su Spectrum emplea un montón de tiempo, mientras corre su programa, interpretando el BASIC para la CPU.

El compilador Softek, al contrario, convierte su BASIC en código de máquina y por ello sus programas compilados correrán diez veces más rápidos que el BASIC (menos, para la versión "FP"), pero incluso se llega a 200 veces más rápido con el compilador "IS" (Integer Compiler).

Ambos compiladores Softek le permiten escoger dónde situar en la memoria el código compilado y ésta versión final del código de máquina puede ser salvado en la cinta con CODE. Lo bueno de los compiladores Softek, es que despues de compilar, su BASIC aún está presente, por lo que para las modificaciones que se requieran no es necesario cargar la cinta ó microdrive También se pueden compilar subrutinas.

CARGA DEL COMPILADOR

Teclear LOAD"" y pulsar ENTER. Arrancar la cinta. Los programas se cargan en dos partes, el BASIC y el Compilador. Cuando esté cargado, le preguntará si la posición del RAMTOP es correcta (40000 para la versión de 48K y 26000 para la de 16K). Si es correcta pulse 'y' y ENTER, de lo contrario pulse 'n' y ENTER y responda con el RAMTOP que desée. Note que el RAMTOP es la dirección donde empezará su programa compilado.

Cada compilador tiene en la cinta las versiones de 16K y 48K. Cualquiera de estas versiones se pueden usar en el Spectrum de 48K, pero naturalmente en el de 16K sólo correrá la versión de 16K.

Una vez cargado y determinado el RAMTOP el programa le dirá que puede "NEW" el BASIC cargador. Hecho ésto su compilador aún está en memoria (protegido arriba de RAMTOP) y Vd podria empezar a programar ó a cargar el BASIC que quira compilar.

UTILIZACION DEL COMPILADOR

El uso del compilador Softek no puede ser más sencillo. Escriba su BASIC como siempre ó carguelo de la cinta ó microdrive. El compilador FP comprendrá prácticamente todos los comandos y funciones BASIC, pero si utiliza el IS, más adelante se indicarán sus limitaciones.

Cuando su BASIC está listo para compilar, 'entrar' RANDOMISE USR 59300 (para 48K ó 26600 para 16K). Hecho ésto su compilador empieza a trabajar. Mientras su compilador pasa dos veces por su BASIC traduciendolo a código de máquina aparecerán varios mensajes:

1 "START ADDRESS"

Indica la dirección inicial donde está el programa compilado (siempre es un byte por encima del RAMTOP)

Este dato se actualiza durante la compilación y muestra la última dirección del código de máquina mientras se va formando.

3 "VARIABLES END"

Este mensaje aparece cuando su programa contiene STRINGS ó ARRAYS. Estas se almacenan inmediatamente después de su Codigo de Máquina. Cuando la compilación ha terminado se indica la última dirección del ARRAY.

4 "FIRST PASS","SECOND PASS"

Para completar la compilación, su compilador Softek, debe de dar dos pasadas por su BASIC. Estos mensajes le indican en qué pasada ó vuelta de su programa se está compilando.

5 "ERROR", "NO ERRORS"

Si al final de la compilación aparece "NO ERRORS" entonces Vd ha conseguido su programa en Código de Máquina. Sin embargo si aparece "ERROR" se detiene la compilación e indica la linea del BASIC y la parte de esta linea que el compilador no entiende. Vd verá un signo de interrogación (?) justo detrás del error y el compilador habrá corrido el cursor al número de esta linea. Así pues, pulsando EDIT bajará esta linea molesta del BASIC

RUNNING COMPILED PROGRAMS

Tome nota, al finalizar la compilación de la START ADDRESS del Código de Máquina y después, desde el BASIC, se llamará al Código de Máquina con la instrucción RANDOMISE USR (START ADDRESS)

¿DONDE SE COLOCA EL CODIGO DE MAQUINA?

Como se indica arriba, el Código de Máquina, se coloca en el RAM, comenzando un byte por encima del RAMTOP que Vd puso. Se puede 'resetear' RAMTOP en cualquier momento 'entrando' CLEAR N. Donde N es uno menos que la direccion inicial que se escogió. Por ello, es muy fácil compilar subrutinas ó construir un programa en partes, ajustando y reseteando RAMTOP con cuidado. (Se debe tomar nota de los valores de "END ADDRESS" ó "VARIABLES END" para asegurarse que no se sobrepongan las distintas partes del programa)

¿QUE OCURRE CON EL BASIC?

Su BASIC sigue en memoria después de compilado. Sin embargo, si el Codigo de Máquina trabaja como se planificó, se puede salvar con CODE (ver más adelante) ó se puede borrar "NEW" para permitir compilar más BASIC.

USO DE LA MEMORIA

La memoria se puede agotar de varias formas.

1- Durante la compilación se construye una tabla de números de LINE. Si se usan demasiadas lineas ocorrirá "OUT OF MEMORY".

2- Si el Código de Máquina ó las variables amenazan con superponerse a lo escrito, entonces también aparaecerá el "OUT OF MEMORY".

VARIBLES, CADENAS, MATRICES Y NUMEROS

Las variables se tratan igual que en el BASIC, permitiendose el uso de los nombres completos en mayusculas ó minusculas. Sólo el compilador 'FP' trata las matrices. pero únicamente si no son multidimensionales.

El compilador 'FP' trata números con coma flotante igual que el BASIC. El compilador 'IS' trata números enteros en el margen de -32767 al 32767 . Algunas funciones tomarán un número almacenado comp Ø á 65536 (por ejemplo : PEEK, POKE, USR).

COMANDOS QUE MANEJA EL COMPILADOR

Tecla	
e	Representa una expresión.
s	Una secuencia de sentencias separadas por dos puntos.
n	Un número entero positivo.
c	Una secuencia de colores separados por ',' ó ';' ó '''.
x,y	Un conjunto de expresiones.
$	Una expresión cadena.
BEEP x,y	Como normal
BORDER e	Como normal
BRIGHT e	Como normal
CIRCLE c;x,y,z	Como normal
CLEAR	Sólo la versión 'FP'
CLS	Como normal.
CLOSE #,n,$	Sólo la versión 'FP'
COPY	Como normal
DATA	En ambas versiones no se pueden usar 'expresiones' en DATA. Solo se pueden usar números ó cadenas en la forma ".." y éstas se pueden mezclar.
DIM A(n) ó A$(n)	Sólo en la versión 'FP'
DRAW c;x,y	Como normal
DRAW c;x,y,z	Como normal
FLASH e	Como normal
FOR a = x TO y	Como normal
FOR a=x TO y STEP z	"
GOSUB n	Como normal
GOTO n	Como normal
IF e THEN s	Igual que en BASIC (totalmente Booleano)
INK e	Como normal
INPUT	Igual que en BASIC excepto para LINE. El editor de lineas BASIC se utiliza permitiendo el control del cursor y se pueden incluir sentencias PRINT. Se pueden escribir varibles si van entre paréntesis.
INVERSE e	Como normal
LET a = e	Como normal
LOAD $ CODE n	Sólo un parámetro
LPRINT	El '#' podría tambien usarse
NEW	Como normal
OPEN #n, $	Sólo en la versión 'FP'.
OUT x, y	Como normal
OVER e	Como normal
PAPER e	Como normal
PAUSE e	Como normal
PLOT c; x, y	Como normal
POKE x, y	Como normal
PRINT	El '#' podría también usarse
RANDOMISE	Como normal
RANDOMISE e	Como normal
READ	Como normal
REM	Como normal
RESTORE	Como normal
RESTORE n	Como normal
RETURN	Como normal

COMANDOS QUE MANEJA EL COMPILADOR (viene de la página anterior)

Tecla	
SAVE $ CODE e, e	Como normal
STOP	Como normal
VERIFY $ CODE e	'e' es opcional

SENTENCIAS REM

Las sentencias REM se usan en el compilador Softek para ofrecer comandos y facilidades que no se obtienen con el BASIC del Spectrum. En cada caso una simple letra mayúscula sigue a REM, pero si esta letra no es una de las del teclado de abajo entonces la linea se trata como un REM normal.

VERSION 'IS' DE ENTEROS

REM B	Comprueba si la tecla BREAK está pulsada.
REM M, n, n..	Entra el código en el programa y permite llamar una rutina en código de máquina en este punto del programa.
REM S, a,x,y	Escribe un carácter (ASCII código 'a') en la pantalla en la posición PLOT x,y. Esto es muy útil para juegos que requieran movimientos suaves de Alta Resolución.

VERSION 'FP' COMPLETA

REM B	Comprueba el BREAK.
REM M,n,n,n,..	Igual que en la versión 'IS'
REM E,n	Pone una trampa para errores que envia a la linea 'n'. Asi pues cualquier error que ocurra se hace un salto a la linea 'n'. Si un programa utiliza ésta faceta el STACK será alterado cuando ocurra un error. Asi pues, si luego se hace un retorno al BASIC se deberá hacer saltando mas allá del programa ó usando la sentencia STOP.
REM N	Restablece la detención de errores al modo normal. Esto debe de hacerse antes de volver al BASIC.
REM O,a,n,n,..	Esto simula la función BASIC ON n GOTO n,n,n,.. 'a' debe ser sólo una simple variable y si es mayor que el número de lineas presente, el programa continua pasada la sentencia.

CADENAS Y SU MANEJO

Las cadenas son totalmente flexibles y pueden tener cualquiera longitud (no sólo 256 caracteres como en BASIC). Las cadenas se almacenan después de las variables normales, las cuales a su vez se almacenan inmediatamente despues del Código Compilado.

Conjuntos de cadenas pueden existir en la versión 'FP' y sólo deben ser de una dimensión. Igual que en el BASIC cuando una cadena está dimensionada se completa con espacios.

Dos tipos de tratamiento son posibles, por ejemplo :
LET A$ (3 TO 7) = "HH" ó LET A$ = "cualquier abode" (3 TO 5) ó una mezcla de ello.

Los números se pueden omitir como en el BASIC p.ej. A$(3 TO) y todos los siguientes son también posibles:
LEN STR$ CODE CHR$ CODE INKEY$ SCREEN$ "+" (para concatenación)

Las cadenas se pueden comparar como en el BASIC.

NOTA

En ambos compiladores lo escrito con PRINT no se evalúa. Por ello si Vd. incluye algo como "abodef" (3 TO 6) ó "y" = "h", eto Vd. debe de encerrar la expresión en parentesis para forzar al compilador a evaluar antes de hacer PRINTing.

Tener en cuenta tambien que tampoco se usan VAL ni VAL$, porque éstas solo se manejan por el BASIC en tiempo real.

COMPILADOR DE ENTEROS : FUNCIONES, etc

FUNCIONES:

ABS, AND, ATTR, CHR$, CODE, IN, INKEY$, LEN, NOT, OR, PEEK, RND (un nº entre Ø y 32767) SCREEN$, SGN, STR$, USRn, USR$, +, /, * y todos los operandos de comparación.

BEEP

Vd podría hacer un BEEP de menos de un segundo incluyendo la división, por ejemplo : BEEP 1/expresión, n . La expresión puede ser cualquier número ó cualquier expresión. La expresión se calcula de forma entera y después se llama al cálculo de coma flotante para hacer la división final.

CLEAR

Limpia las variables simples y de control, borrando tambien las cadenas variables.

FULL COMPILER : FUNCIONES, etc

El compilador 'FP' tratará todas las funciones posibles con el compilador 'IS'. Además tratará las funciones de coma flotante tales como SIN, COS, ATN, etc CLEAR también se puede usar para sus otras funciones ajustando RAMTOP en esta versión 'FP' sólamente.

El comando CLEAR limpiará todos los conjuntos y tambien cambiará el complejo STACK. Así pues, se debe de usar la sentensia STOP para RETURN al BASIC despues de CLEAR, sino, el programa se perderá.

En general, no deberia RETURN al BASIC desde un programa compilado 'FP' excepto muy al final del programa ó usando la sentencia STOP.

DIFERENCIAS GENERALES ENTRE LOS COMPILADORES

Los compiladores 'FP' y 'IS' necesitan ambos una cantidad de memoria RAM muy similar (unos 6K). El 'FP' es cláramente más capaz de compilar todo el rango del BASIC del Spectrum y por ello es más apropiado para usos cientificos y técnicos. Los números compilados en el 'FP' son tratados como con coma flotante (5 bytes cada uno), mientras que en la versión 'IS' son todos enteros (2 bytes cada uno). Por esta razón, el compilador 'FP' es más lento que el 'IS'. Mientras que el 'IS' es 10 veces más rápido que el BASIC (puede llegar a 500 veces), el 'FP' viene a ser de 2 á 10 veces más rápido que el BASIC.

LOS COMPILADORES : USO Y SITUACION EN MEMORIA

El código compilado se situará siempre en RAM, un byte por encima del RAMTOP. Asi pues, Vd puede escoger dónde colocarlo en la RAM, usando 'CLEAR n' para cambiar RAMTOP (ver el cargador del compilador). El compilador debe estar en la memoria cuando el código debe correr el programa en código de máquina, porque el código compilado usa 'run time routine' en el compilador.

Después de esto, el código compilado es un bloque de 130 bytes el cual almacena todo acerca de las 36 cadenas permitidas. Así pues, hay normalmente espacios libres, a menos que ésto se llene con cadenas cuando el programa M/C las crea,

prog	arrays	linea direcciones	reserva	M Code	Vars	$ data 130bytes	$'s'	reserva

vars ads	Compiler

VARS STKEND RAMTOP

El compilador 'FP' almacena todos los conjuntos en el área variable del interpretador del BASIC. Se usa una rutina de búsqueda de tal forma que el

tiempo para buscar ésta variable es extremadamente corto.

Cuando se compila, el compilador construye una tabla de direcciones de lineas del programa que se produce. Esta es almacenada justo encima de STKEND en la zona de memoria de reserva. El compilador tambien construye una tabla de direcciones de las diferentes variables usadas en el programa. Esta tabla crece hacia abajo en la memoria desde el principio del compilado.

El compilador provee espacio suficiente para las variables que se usan, p.ej. si sólo se usa una variable, 2 bytes (ó 5 para la versión 'FP') se reservan para aquella variable. Pero si aquella variable se usa en un lazo FOR NEXT, entonces 8 bytes (ó 17 para el 'FP') se reservan para ella.

Las variables pueden ser de cualquier longitud y se puede usar cualquier número de variables (si por supuesto Vd tiene suficiente memoria).

SALVAR EL CODIGO COMPILADO

Para que el compilador produzca un Código MÁquina rápido, debe haber un conjunto de rutinas RUN TIME, a las que se llama desde el Código. El Código de MÁquina no puede funcionar sin estas rutinas, asi que hay que salvarlas junto con el Código.

Sus programas se pueden salvar como sigue :

Para version 48K: SAVE "nombre" CODE start address, 65536 - start address
Para versión 16K: SAVE "nombre" CODE start address, 37767 - start address

EJEMPLOS

```
1Ø  FOR A= Ø TO 1ØØ : OUT 254, 53 : FOR B= Ø TO A : NEXT B : OUT 254, Ø :
    FOR B= A TO Ø STEP-1 : NEXT B : NEXT A
2Ø  REM B
3Ø  GOTO 1Ø
```

```
1Ø  FOR A= 1 TO 128
2Ø  LET B= (255 - A)
3Ø  REM S, 87,,A,,A
4Ø  REM S,,88, B, A
5Ø  REM B
6Ø  BEEP 1/5Ø,, 1
7Ø  NEXT A
```

```
SOFTEK 'FP'
FULL COMPILER
©1983 Martin Lewis
V1.1:48K RAM
IS LOADING
```

```
SOFTEK 'FP'
FULL COMPILER
©1983 Martin Lewis
V1.1:48K RAM
LOADED
```

```
  10 PAPER 0: BORDER 0: INK 6: C
LEAR 39999
  20 PLOT 30,100: DRAW 155,0: DR
AW 0,-70: DRAW -155,0: DRAW 0,70
  30 PRINT AT 5,11;"SOFTEK 'FP'"
;AT 7,10;"FULL COMPILER"
  40 PRINT AT 9,7;"©1983 Martin
Lewis"
  50 PRINT AT 11,10;"V1.1:48K RA
M"
  60 PRINT AT 14,11;"IS LOADING"
  70 LOAD ""CODE
  80 PRINT AT 14,11;"   LOADED
"
  90 PRINT AT 19,0;"RAMTOP at 6
0000?(Y or N)";: INPUT LINE Q$
 100 IF Q$="Y" THEN GO TO 120
 110 PRINT AT 19,0;"RAMTOP?
";: INPUT RT: CLEAR
RT
 120 CLS : PRINT AT 10,0;"You a
y now NEW the BASIC loader"
   Please refer to manual"
 130 STOP
 200 SAVE "      COMPILER" LINE 1
0
 210 SAVE "MACHINE CODE "CODE 59
300,6050
```

"CLAVE"

1º) Cargar en el ordenador el programa a proteger. Acceder al listado y comprobar que las líneas no coincidan en ambos (el progr. "Clave" va desde 9975 a 9999)

2º) Cargar el programa "CLAVE" con MERGE

3º) Fijar la palabra clave: hacer LET P$="la clave"

4º) Salvar en el cassette: SAVE "nombre" LINE 9976

Poner Línea Ø

```
LET a= PEEK 23635+256*PEEK 23636:POKE a,Ø:
       POKE a+1,Ø
```

Quitar Línea Ø (pasa a línea 1

```
POKE PEEK 23635 +256 *PEEK 23636 +1,1
```

HECHO POR

PASCUAL

[illegible]