11
150 pts.

RUN

Enciclopedia Práctica del Spectrum

Nueva Lente/Ingelek

RUN

Enciclopedia Práctica del Spectrum

una publicación de

EDICIONES NUEVA LENTE, S. A., Y EDICIONES INGELEK, S. A.

Director editor por NUEVA LENTE
MIGUEL J. GOÑI
Director editor por INGELEK
ANTONIO M. FERRER
Director de producción
RICARDO ESPAÑOL
Jefe de producción
SANTOS ROBLES
Director de la obra
FERNANDO LOPEZ MARTINEZ
Coordinación
VICENTE ROBLES
Colaboradores
JUAN MANUEL LOPEZ MARTINEZ
CARLOS DE LA OSSA
JUAN MANUEL MAYORAL
Diseño gráfico
JOSE OCHOA
Maquetación
CARLOS GONZALEZ-AMEZUA
Ilustraciones
JOSE OCHOA
ALFONSO MENDEZ
ANTONIO PERERA
Fotografía
(Equipo Gálata)
ALBINO LOPEZ y
EDUARDO AGUDELO
Ediciones Nueva Lente, S. A.
Dirección y Administración:
Benito Castro, 12
28028 Madrid. Tel. 2454598
Ediciones Ingelek, S. A.
Números atrasados y suscripciones:
Avda. Alfonso XIII, 141
28016 Madrid. Tel. 2505820
Publicidad:
LOLA GONZALEZ
CARMINA FERRER
Tel. 4576923

Plan general de la obra:
52 fascículos de aparición semanal encuadernables en cuatro tomos de 13 fascículos

Distribución en España:
COEDIS, S. A. Valencia, 245.
08007 Barcelona
Distribución en Argentina:
Capital: Ayerbe
Interior: DGP
Distribución en Colombia:
DISUNIDAS, Ltda.
Distribución en Chile:
Alfa Ltda.
Distribución en Ecuador:
Muñoz Hermanos, S. A.
Distribución en México:
IMTERMEX, S. A.
Lucio Blanco, 435
México D. F.
Distribución en Paraguay:
Selecciones, SAC.
Distribución en Perú:
DISELPESA
Distribución en Puerto Rico:
Agencia de Publicaciones de Puerto Rico, Inc.
Distribución en Uruguay:
Ledian, S. A.
Distribución en Venezuela:
CONTINENTAL
Editor para Chile:
Editorial Andina, S. A.
La Concepción, 311. Santiago-9
Importador exclusivo Cono Sur:
CADE, SRL
Pasaje Sud América 1532. Tel. 212464
Buenos Aires-1.290. Argentina.

Fotomecánica: OCHOA
Ricardo Ortiz, 74. Madrid.
Impresión: Gráficas Reunidas, S. A.
Avda. de Aragón, 56. Madrid
ISBN de la obra: 84-7534-118-7
ISBN del fascículo: 84-7534-119-5
ISBN del tomo primero: 84-7534-120-9
Depósito legal: M-9896-1985
PRINTED IN SPAIN

Ediciones Nueva Lente, S. A. y Ediciones Ingelek, S. A. garantizan la publicación de todo los fascículos que componen esta obra y el suministro de cualquier número atrasado o tapa mientras dure la publicación y hasta un año después de terminada. El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra si las circunstancias del mercado así lo exigen.

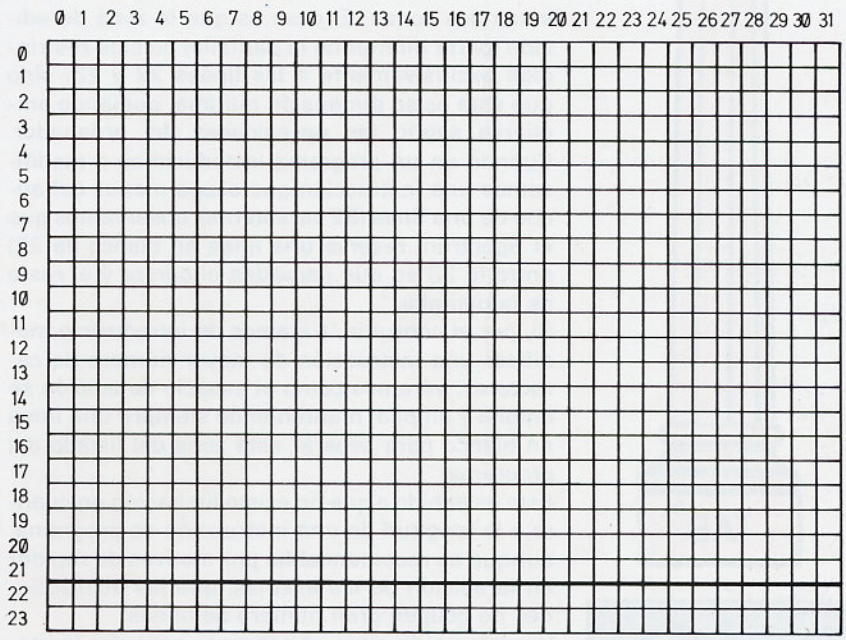
JUNIO 1985

SITUACION EN LA PANTALLA

NO cabe duda de que la parte más espectacular de la programación, es la impresión en la pantalla de literales y resultados; la sentencia BASIC que permite efectuar estas salidas al «exterior», de los contenidos de la memoria del ordenador es **PRINT**.

Hasta ahora hemos analizado la sentencia **PRINT** por sí misma, y en combinación con algunos signos de puntuación. Podemos hacer aquí un breve repaso de todo lo dicho, y ampliar conocimientos con nuevas posibilidades de **PRINT**, la más versátil de las palabras BASIC.

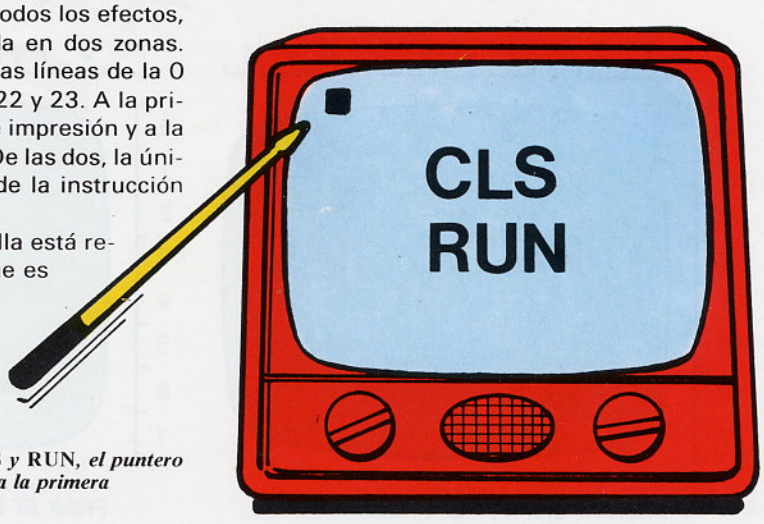
LA ESTRUCTURA DE LA PANTALLA



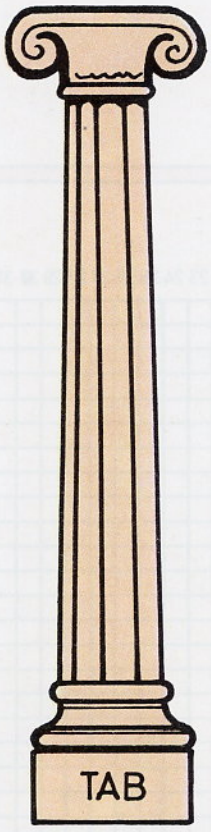
La pantalla del Spectrum está compuesta por 24 filas de 32 columnas, de las cuales las dos últimas filas están habitualmente reservadas para el uso del sistema.

La pantalla del Spectrum consta de 24 líneas de 32 columnas, numeradas simbólicamente de la 0 a la 23 y de la 0 a la 31, respectivamente, cuyo origen se encuentra en el ángulo superior izquierdo de la pantalla. De esta forma, es posible imaginarnos la pantalla del ordenador como una hoja de papel cuadriculado de dimensiones 24×32, sobre la que podemos escribir y reescribir a nuestro gusto, como si de una pizarra se tratase. Debemos saber también que, a todos los efectos, la pantalla se encuentra dividida en dos zonas. La primera de ellas comprende las líneas de la 0 a la 21, y la segunda las líneas 22 y 23. A la primera zona la denominaremos de impresión y a la segunda, zona del «ordenador». De las dos, la única que es accesible por medio de la instrucción **PRINT** es la primera.

El segundo apartado de la pantalla está reservado al ordenador, puesto que es la zona que se emplea para la edición de las instrucciones BASIC, para su introducción, modificación o borrado. Son también las líneas encar-



Cuando se ejecutan los comandos **CLS** *y* **RUN**, *el puntero de pantalla (posición de* **PRINT**) *indica la primera posición de la misma.*



La función BASIC **TAB** *se utiliza para el desplazamiento de la posición de impresión por columnas.*

gadas de recoger los mensajes del Spectrum, tanto de advertencia como de error.

Si hemos sido observadores durante la introducción de los programas de ejemplo, nos habremos dado cuenta de la existencia bien diferenciada de estas dos zonas. El caso es que la zona de edición (parte inferior de la pantalla) no está restringida exclusivamente a las líneas 22 y 23, sino que ésta es su dimensión mínima, pudiendo ampliarse según las necesidades del ordenador.

Cuando en un programa introducimos o modificamos una instrucción que ocupa menos del ancho de una línea (32 caracteres), observamos que el Spectrum reserva una línea en blanco (la 22) entre la 23 en que parpadea el cursor y el resto de la pantalla.

Si, por el contrario, tratamos de introducir o modificar una instrucción de mayor número de caracteres, veremos cómo el espacio de edición se amplía y amplía, manteniendo siempre una línea en blanco para separar esta zona del listado del programa.

Esto es debido a que no existe limitación en cuanto a la longitud de una instrucción de programa, aunque es recomendable, por motivos de rapidez en la edición de las mismas, que las instrucciones no ocupen gran número de líneas.

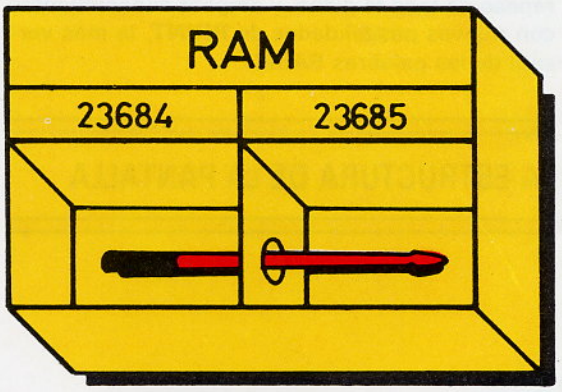
Veremos más claro el concepto reticular (cartesiano) de la pantalla por medio de un ejemplo:

```
10 REM - ESTRUCTURA DE LA PANTALLA
20 FOR I=0 TO 31
30 PRINT (STR$ I)(LEN STR$ I);
40 NEXT I
50 FOR I=1 TO 21
60 PRINT (STR$ I)(LEN STR$ I)
70 NEXT I
80 STOP
```

Por medio de este programa hemos podido ver, claramente diferenciadas, las dos zonas en que se divide la pantalla. La zona de impresión convencional y la reservada a los mensajes de error y advertencia del Spectrum. En nuestro caso, una vez ejecutado el programa, observamos en la línea 23 el mensaje:

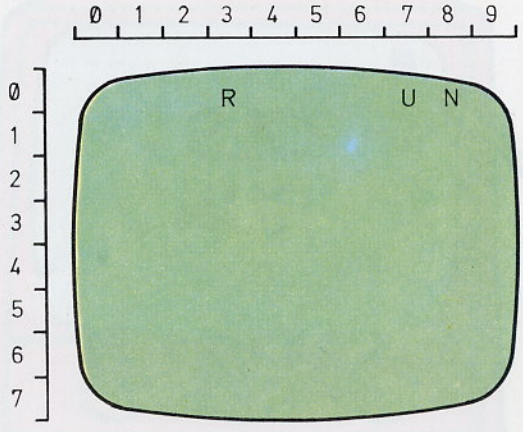
9 STOP statement, 80:1

En la primera línea de pantalla aparecen los dígitos de las unidades de los posibles valores de las columnas. Del mismo modo, en la primera columna de todas las filas, aparecen los correspondientes a los números de línea. Estos son los márgenes en que podemos trabajar por medio de la sentencia **PRINT**.

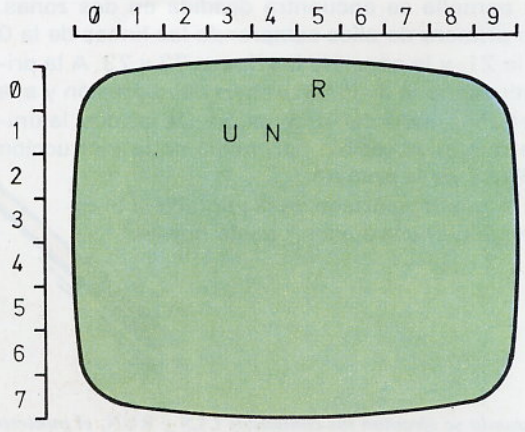


Las posiciones de memoria 23684 y 23685 (dentro de las Variables del Sistema), albergan el puntero de la pantalla.

Los argumentos para una serie de funciones **TAB**, *siempre deben estar en orden creciente, lo cual no es imprescindible utilizando* **AT**.



PRINT AT 0,3;"R";TAB 7;"UN"



PRINT AT 0,5;"R";TAB 3;"UN"

Existe una Variable del Sistema, que le indica a la sentencia **PRINT** a qué posición de la pantalla debe efectuarse la siguiente salida de un carácter. Este puntero de impresión es manejado directamente por el ordenador y, por lo tanto, no debemos preocuparnos de él.

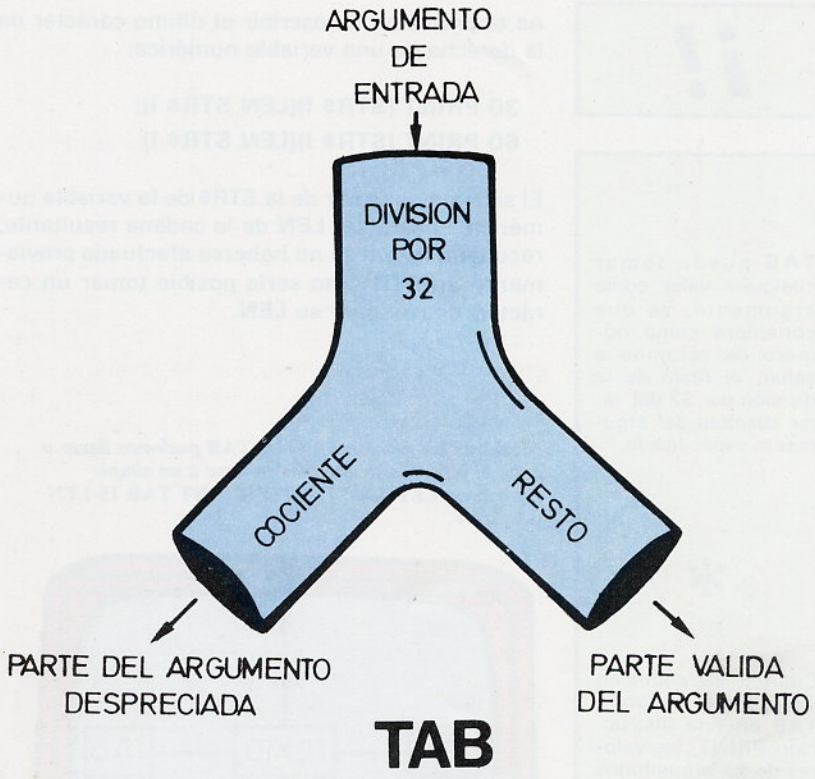
Cuando efectuamos un borrado de pantalla por medio de las sentencias **CLS** o **RUN**, este puntero señala a la primera dirección lógica de la pantalla, que es la columna 1 de la línea 1.

PUNTERO DE IMPRESION

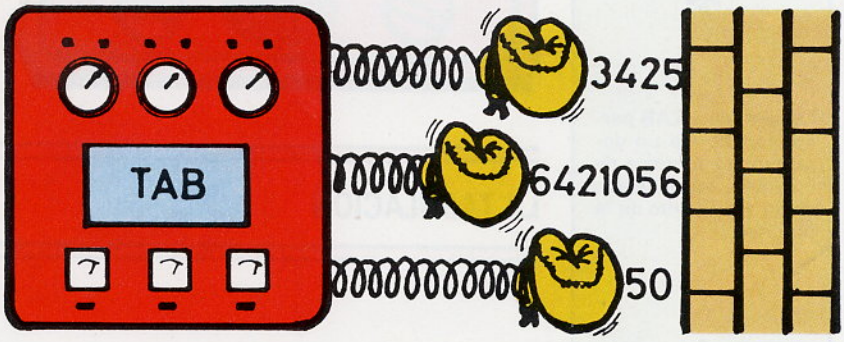
Existe una Variable del Sistema, que le indica a la sentencia **PRINT** en qué posición de la pantalla debe efectuarse la siguiente salida de un carácter. Este puntero de impresión es manejado directamente por el ordenador y, por lo tanto, no debemos preocuparnos de él. Sin embargo, hay algunas cosas sobre su funcionamiento que nos interesa saber: cuando efectuamos un borrado de pantalla por medio de las sentencias **CLS** o **RUN**, este puntero señala a la primera dirección lógica de la pantalla, que es la columna 0 de la línea 0. De esta forma, el contenido de la siguiente instrucción con un **PRINT** vendrá a parar a este lugar concreto de la pantalla.

En el caso del programa ejemplo anterior, al ejecutar **RUN**, el puntero se coloca señalando a la primera dirección lógica de la pantalla. Como a continuación existe un bucle de 32 impresiones de un carácter con punto y coma (;), cada una de ellas se coloca inmediatamente a la derecha de la anterior, sin efectuarse ningún retorno de carro, hasta completar la línea. Esto sucede en las líneas 20 a 40.

Observamos que las líneas 30 y 60 son extraordinariamente parecidas, difiriendo tan sólo en que la última no termina con punto y coma (;). Sin embargo, los resultados son completamente diferentes, puesto que en la instrucción 60 los caracteres son impresos uno a continuación de otro, pero en columna y no en la misma fila.

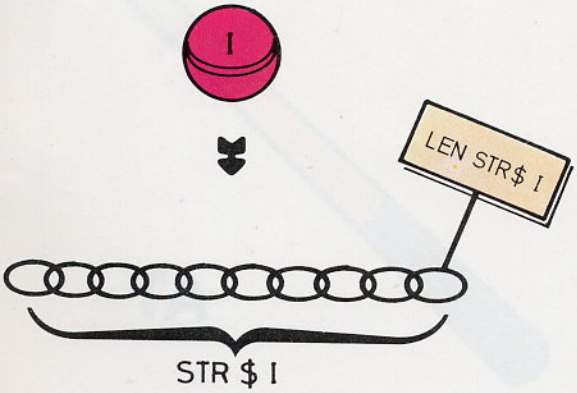


La función **TAB** *opera con el módulo 32 del valor absoluto de su argumento.*



Apoyándonos en la función **TAB**, *podemos llevar a cabo la edición de valores numéricos columnados por la derecha.*

Para tomar el último dígito de un valor numérico, primeramente es necesario convertirlo en cadena (STR$), y posteriormente hallar el carácter correspondiente a su longitud (LEN).



Esto es debido a que al no especificarse el punto y coma (;) al final del **PRINT** de dicha instrucción, el BASIC supone que se desea un retorno de carro, es decir, una puesta del puntero de impresión al valor de la primera columna de la siguiente línea de la pantalla.

Del programa anterior, merecen especial comentario las líneas 30 y 60; además de por lo dicho hasta el momento, porque en ambas se solucio-

¡!

TAB puede tomar cualquier valor como argumento, ya que considera como número de columna a saltar, el resto de la división por 32 del valor absoluto del argumento especificado.

*

Cuando se codifican varias expresiones **TAB** en una instrucción **PRINT**, los valores de los argumentos deben aparecer en secuencia ascendente.

*

La sentencia **TAB** permite «saltar» a un determinado número de columna, especificado como argumento de la función.

*

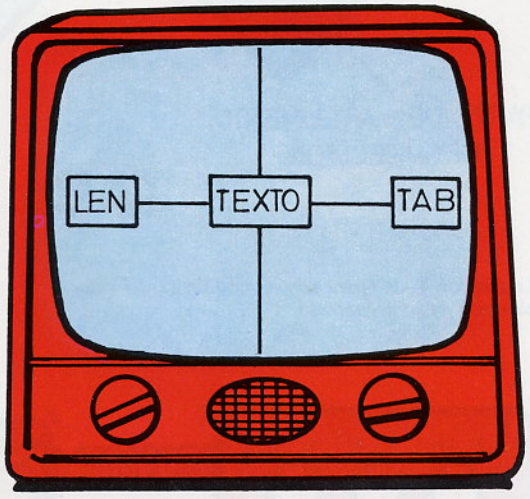
Cuando se especifica como argumento de **TAB** un número de columna ya sobrepasado, se genera un cambio automático a la siguiente línea de la pantalla, antes de saltar a la columna especificada por el argumento.

na el problema de escribir el último carácter de la derecha de una variable numérica:

```
30 PRINT (STR$ I)(LEN STR$ I);
60 PRINT (STR$ I)(LEN STR$ I)
```

El sistema es tomar de la STR$ de la variable numérica, el carácter **LEN** de la cadena resultante; recordemos que de no haberse efectuado previamente un **STR$**, no sería posible tomar un carácter, ni averiguar su **LEN**.

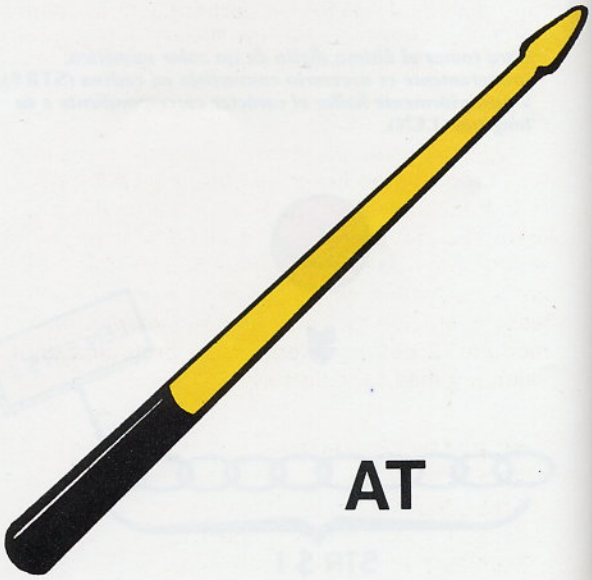
Mediante las funciones* LEN *y* TAB *podemos llevar a cabo el centrado de literales en base a un simple algoritmo: LET A$="TEXTO":PRINT TAB 15-LEN A$/2;A$.

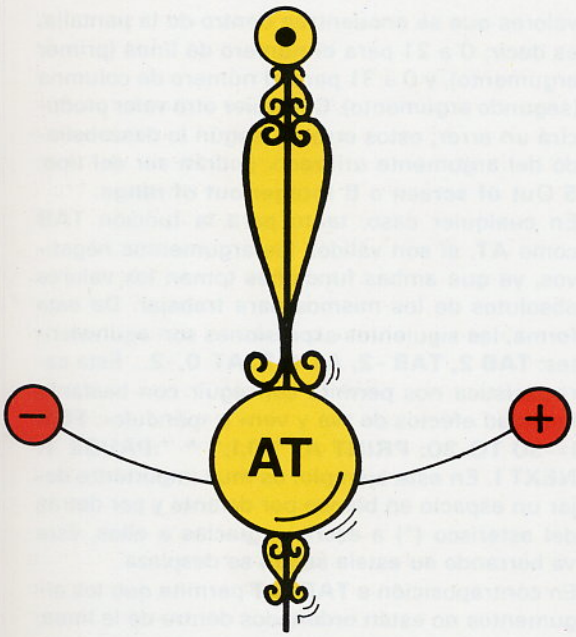


LA TABULACION

El empleo de la sentencia **PRINT** con coma (,) lo hemos visto en capítulos anteriores. Como recordatorio diremos que propicia un salto al comienzo de la siguiente «mitad» de línea, suponiendo que ésta se divide simbólicamente en dos mitades de 16 columnas cada una.
La sentencia **TAB** es mucho más precisa, permitiendo «saltar» a un determinado número de columna, especificado como argumento de la función. **TAB** toma el valor absoluto de los argumentos negativos, y puede utilizar expresiones mayores que 31. Esto es posible gracias a que opera realmente con el resto de la división entre 32 del argumento. De esta forma, las expresiones siguientes producen el mismo efecto: **PRINT TAB 5, PRINT TAB 37.**
Una restricción en cuanto a los argumentos de la función **TAB** es que éstos, en el caso de existir más de uno, deben estar ordenados en forma ascendente, es decir, el **TAB** nunca da «marcha atrás». Un ejemplo correcto sería: **PRINT TAB 5;1;TAB 10;2;TAB 15;3.** También lo sería, si nos fijamos en el tratamiento que la función **TAB** da a sus argumentos, haber escrito 47 en lugar de 15, en la última tabulación de la línea del ejemplo.
Si escribimos: **PRINT TAB 10;1;TAB 5;2;TAB 15;3** los resultados que obtenemos son diferentes. Esto es debido a que, como ya hemos dicho, no podemos especificar como argumento de la función **TAB** un número inferior al de la columna de la última tabulación realizada.
En la línea del ejemplo, la primera tabulación se realiza correctamente a la columna 10, pero la segunda, al no poder efectuarse en esa línea un salto a una columna anterior, genera automáticamente un cambio a la siguiente línea, y después efectúa la tabulación a la columna 5. El **TAB** a la columna 15 se realiza correctamente por ser esta columna mayor que 5, pero se efectúa en la misma línea que su tabulación precedente.
TAB nos sirve para imprimir a partir de una columna concreta de la pantalla, pero nos puede brindar posibilidades más concretas. Por ejemplo, obtener columnas de números en el lugar deseado:

La función* AT *sitúa el puntero de pantalla en cualquier posición de la misma que nosotros indiquemos.





Gracias a que la función **AT** *actúa sobre el valor absoluto de sus parámetros, podemos apoyarnos en ella para la programación de móviles con desplazamiento pendular.*

```
10 REM - COLUMNAS DE NUMEROS
20 LET X=0
30 FOR I=0 TO 21
40 FOR J=0 TO 28 STEP 4
50 LET X=X+1:PRINT TAB J;X;
60 NEXT J
70 NEXT I
```

En este programa hemos hecho un recorrido por las líneas de pantalla de la 0 a la 21, imprimiendo en cada una de ellas sólo en las columnas múltiplos de 4 **(STEP 4)**, partiendo de la columna 0. Hemos efectuado un **PRINT** con punto y coma (;), para no generar cambios de línea intermedios, sumando 1 a la variable contadora X por cada nueva impresión.
Pero este mismo programa se puede simplificar haciendo uso de la forma con que **TAB** calcula la columna en que imprimir, a partir del argumento de la función; ya que **TAB** divide por 32 el argumento que especificamos, utilizando el resto de la operación como columna a la que saltar (esta operación se conoce como «extracción del módulo 32 del argumento»); un programa equivalente y más sencillo sería:

```
10 REM - COLUMNAS DE NUMEROS
20 LET X=0
30 FOR I=1 TO 704 STEP 4
40 LET X=X+1:PRINT TAB I;X;
50 NEXT I
```

También, en combinación con las funciones **LEN** y **STR$**, podemos emplear **TAB** para obtener las columnas de números justificadas por la derecha:

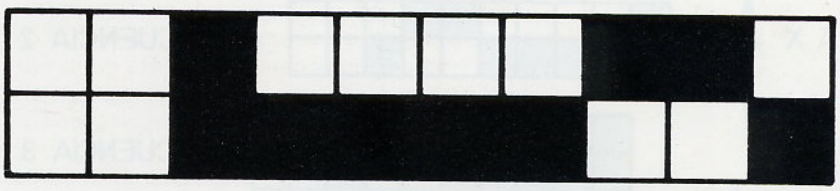
```
10 REM - COLUMNAS POR LA DERECHA
20 LET X=0
30 FOR I=4 TO 704 STEP 4
40 LET X=X+1:PRINT TAB I-LEN STR$ X;X;
50 NEXT I
```

Este programa es muy similar al anterior, pero además de presentar los valores de X en intervalos de cuatro posiciones, justifica dichos valores por la derecha. De comienzo, situamos el **TAB** en la última posición del intervalo, para restarle después la longitud del valor *string* (cadena) de la variable numérica X, e imprimir su valor con punto y coma (;).

IMPRESION EN UN PUNTO DETERMINADO

Del mismo modo que **TAB** nos permite saltar a una columna concreta, dentro de la línea actual sobre la que se está produciendo la impresión, existe otra función, más completa, que permite el posicionamiento directo en cualquier punto de la pantalla: **AT**. Esta vez, se han de suministrar dos valores separados por comas, de los cuales, el primero indica el número de línea (0 a 21), y el segundo el número de columna de la línea especificada en el valor anterior (0 a 31).
Debemos hacer ahora algunas matizaciones referentes a los argumentos empleados por las funciones **TAB** y **AT**. Hemos dicho que **TAB** permite argumentos mayores que 31, al tomar el resto de la división por 32 del argumento especificado, como número de columna a saltar; pero no es este el caso de **AT**. La función **AT** sólo permite

Sprite *utilizado para el programa «ATERRIZAJE».*



¡!

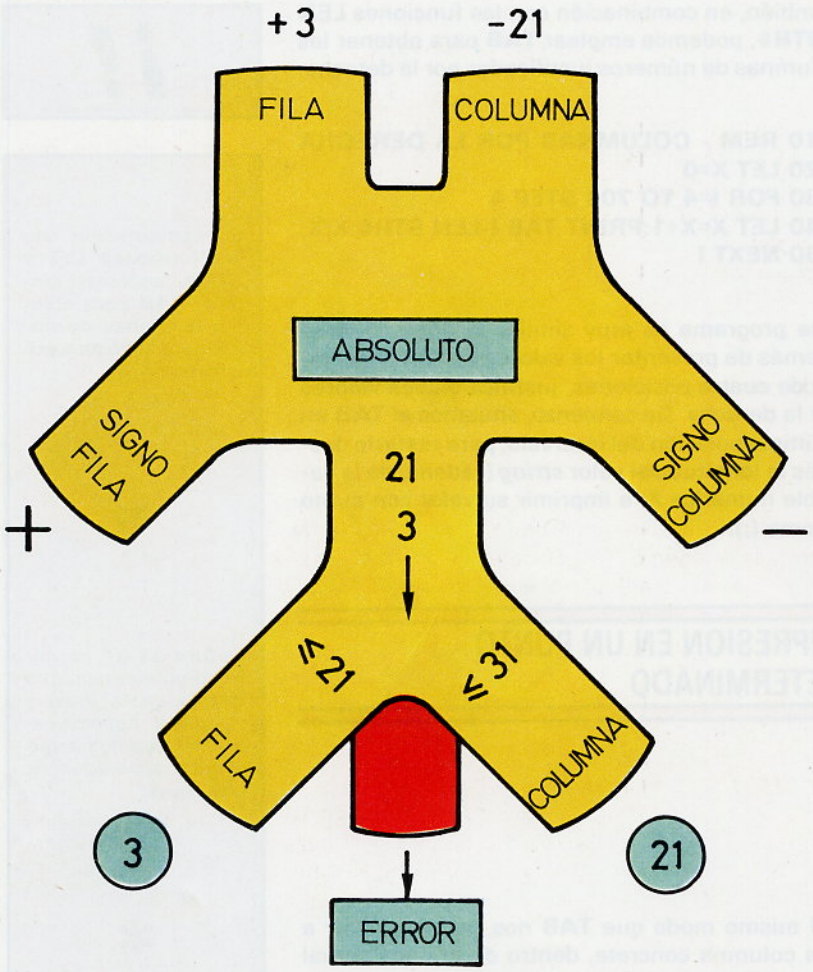
En combinación con las funciones **LET** y **STR$**, podemos emplear **TAB** para obtener columnas de números justificadas por la derecha.

*

La función **AT**, permite un posicionamiento directo en la línea y columna especificadas en sus dos argumentos, separados por una coma.

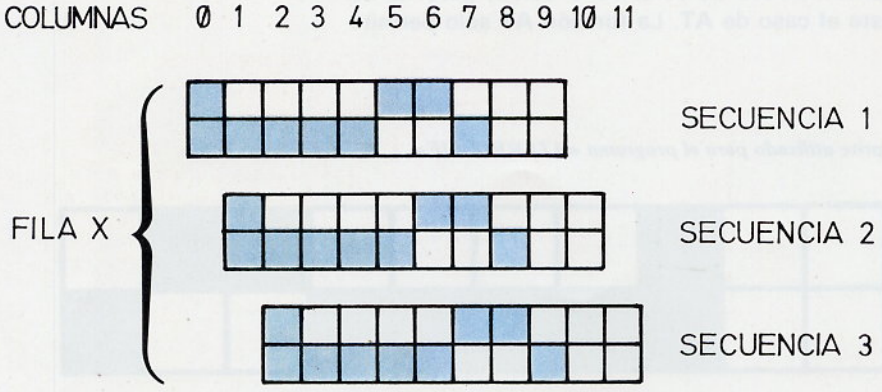
*

La función **AT** sólo permite valores que se encuentren dentro de la pantalla, es decir, 0 a 21 para el número de línea (primer argumento), y 0 a 31 para el número de columna (segundo argumento).



En el proceso de ejecución de la función **AT**, *primeramente se realiza un acondicionamiento de los parámetros (hallando su valor absoluto), para acto seguido llevar a cabo una depuración de los valores admisibles.*

En el desplazamiento de móviles a posiciones consecutivas de una misma línea de pantalla, es necesario dejar un espacio en blanco en su final, para que automáticamente la propia impresión realice el borrado de la estela.



valores que se encuentren dentro de la pantalla, es decir, 0 a 21 para el número de línea (primer argumento), y 0 a 31 para el número de columna (segundo argumento). Cualquier otro valor producirá un error; estos errores, según lo descabellado del argumento utilizado, podrán ser del tipo: **5 Out of screen** o **B Integer out of range**.

En cualquier caso, tanto para la función **TAB** como **AT**, sí son válidos los argumentos negativos, ya que ambas funciones toman los valores absolutos de los mismos para trabajar. De esta forma, las siguientes expresiones son equivalentes: **TAB 2, TAB -2, AT 0,2, AT 0,-2**... Esta característica nos permite conseguir con bastante facilidad efectos de «va y ven» o «péndulo»: **FOR I=-30 TO 30: PRINT AT 10,I;" * ":PAUSE 1: NEXT I.** En este ejemplo, es muy importante dejar un espacio en blanco por delante y por detrás del asterisco (*) a escribir; gracias a ellos, éste va borrando su estela según se desplaza.

En contraposición a **TAB, AT** permite que los argumentos no estén ordenados dentro de la línea, es decir, podemos escribir a cualquier línea y columna, justamente en el orden deseado, y sin preocuparnos de que los argumentos vayan en orden creciente, como sucede en el caso de **TAB**, ya que los resultados finales serán idénticos.

Antes de continuar hablando de **AT**, haremos un pequeño alto en el camino para aclarar el comportamiento exacto de la función **TAB** y el carácter coma (,). Hasta ahora, para simplificar en lo posible las explicaciones sobre estos elementos de edición, hemos dicho que tanto **TAB** como coma **(,)** producen un salto en su misma línea (o a la siguiente, si precisan retroceder en columnas).

De forma similar, **AT** puede saltar a cualquier punto de la pantalla. Pero existe una diferencia fundamental en la forma de desplazarse de la última función con respecto a las primeras: mientras que **AT** simplemente desplaza la posición de impresión, tanto **TAB** como coma **(,)** escriben espacios hasta situarse en su lugar.

Esto quiere decir que si en una línea en la que se realiza un **TAB** (o una edición con coma), existiera cualquier serie de caracteres entre la posición original y la de destino, ésta quedaría inmediatamente borrada..

Una prueba evidente de las posibilidades de **AT** es la simulación de movimiento:

```
10 REM - MANIOBRA DE ATERRIZAJE
20 FOR I=0 TO 20
30 FOR J=0 TO 31
40 PRINT AT I,J;" 1366"
50 NEXT J
60 NEXT I
70 STOP
```

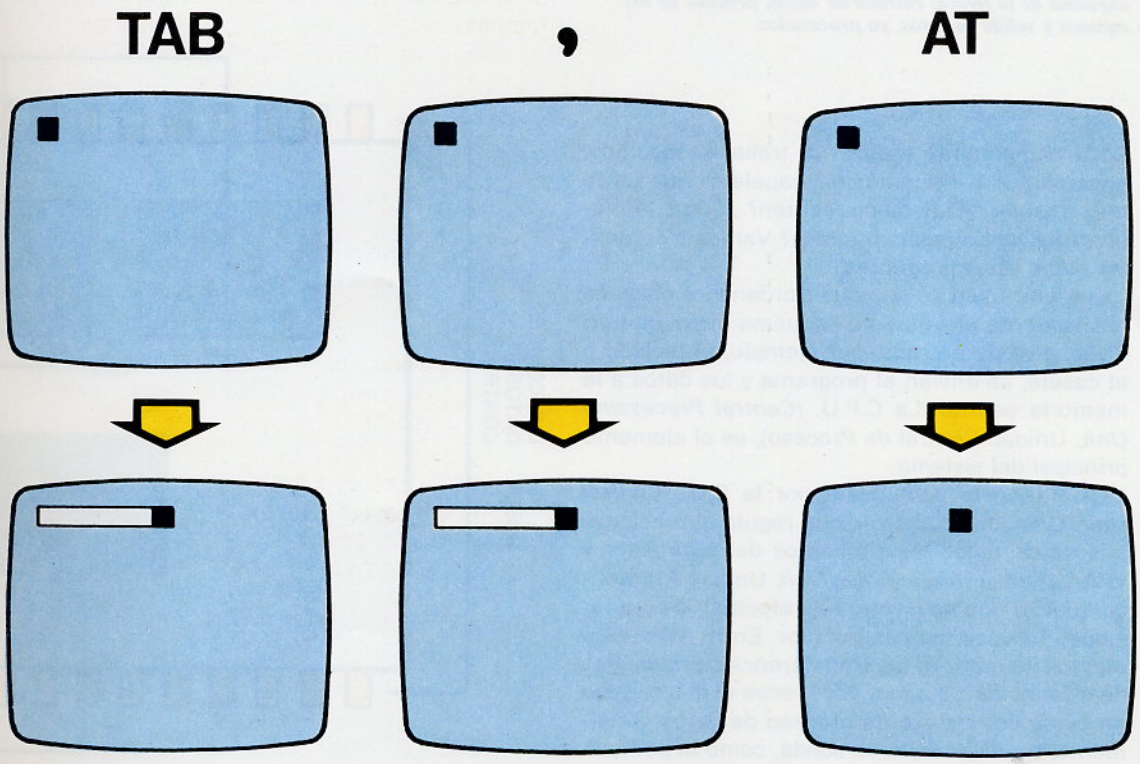
En nuestro ejemplo, hacemos aterrizar una tosca avioneta, que va perdiendo altura en sucesivas

pasadas, comenzando su vuelo desde la línea superior de la pantalla, para aterrizar sobre el mensaje del **STOP** de la línea 70.
En la línea 40, situamos el móvil en las coordenadas **I** (línea) y **J** (columna), siendo el recorrido del mismo el marcado en las líneas 20 (filas) y 30 (columnas).
Para la descripción de los caracteres que componen el móvil, hemos utilizado la convención de que los números subrayados representan a los gráficos asignados al dígito correspondiente, significando los subrayados doblemente los que se obtiene con **CAPS SHIFT**, dentro del modo **GRAPHICS**.
Otra característica a significar, dentro de esta misma línea, es el espacio en blanco que precede a los cuatro gráficos que componen el *sprite* (gráfico móvil). Este espacio cumple una importante función, que es la de borrar la estela dejada por el móvil, en su desplazamiento de izquierda a derecha de la pantalla, ya que por cada instrucción **PRINT** ejecutada, el *sprite* se dibuja una posición más adelante.
Desde luego, este programa de ejemplo podría haberse codificado por medio de la función **TAB**, ya que se produce un incremento sucesivo de la posición de tabulación, aunque otros tipos de movimiento como el de despegue, sólo se puede simular con **AT**:

```
10 REM - MANIOBRA DE DESPEGUE
20 FOR I=20 TO 0 STEP -1
30 FOR J=0 TO 31
40 PRINT AT I,J;" 1366"
50 NEXT J
60 PRINT AT I+1,0;"      "
70 NEXT I
```

Para conseguir que la avioneta despegue, han sido necesarias ciertas modificaciones en el programa. La más importante, la de la instrucción 20, para que la secuencia de líneas sobre las que imprimir vaya desde abajo hacia arriba. La segunda, la introducción de la línea 60, que antes no era necesaria. La misión de la línea 60 es la de borrar la imagen de la avioneta, cada vez que desaparece por completo por el margen derecho de la pantalla, ya que al imprimir un literal de 5 caracteres de longitud en la columna 31 de una línea, los cuatro últimos (la parte de avioneta del *sprite)* pasan a imprimirse como los primeros de la línea siguiente.
Por supuesto que el empleo de **AT** no está restringido a la simulación de movimiento, sino que es la forma más rápida y fácil de imprimir en la pantalla, cuando los literales o datos no tienen por qué ir unos detrás de otros, en una misma línea.

Los elementos de edición* TAB *y coma (,), además de producir un desplazamiento de la posición de impresión, dejan una estela (rastro) de espacios en blanco a su paso.



¡!

En contraposición a **TAB, AT** permite que los argumentos no estén ordenados dentro de la línea, es decir, podemos escribir en cualquier línea y columna y en el orden deseado.

*

La pantalla del Spectrum consta de 24 líneas de 32 columnas, numeradas simbólicamente de la 0 a la 23, y de la 0 a la 31, respectivamente, cuyo origen se encuentra en el ángulo superior izquierdo de la pantalla.

*

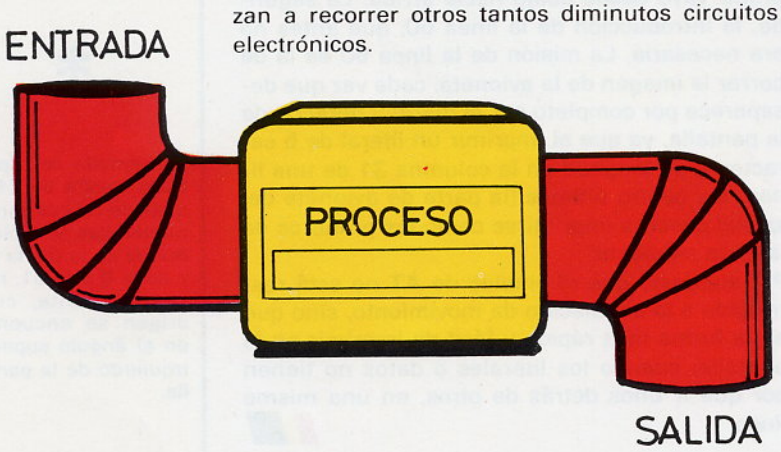
La pantalla se encuentra dividida en dos zonas. La primera de ellas comprende las líneas de la 0 a la 21, y la segunda las líneas 22 y 23. La primera zona es accesible por medio de **PRINT**, mientras que la segunda está reservada para uso del ordenador.

*

La zona inferior de la pantalla, es utilizada por el Spectrum para la edición de líneas de programa, e impresión de los mensajes de error y advertencia.

EL INTERIOR DEL SPECTRUM

CONECTAMOS a la red nuestro Spectrum. Encendemos el televisor y sobre la pantalla aparece el mensaje inicial de Sinclair. La labor es rutinaria; seguramente, la hemos realizado cientos de veces sin pararnos a pensar que, a partir de ese momento, miles de impulsos eléctricos comienzan a recorrer otros tantos diminutos circuitos electrónicos.



La organización de un ordenador responde básicamente al esquema de la figura: entrada de datos, proceso de los mismos y salida de éstos, ya procesados.

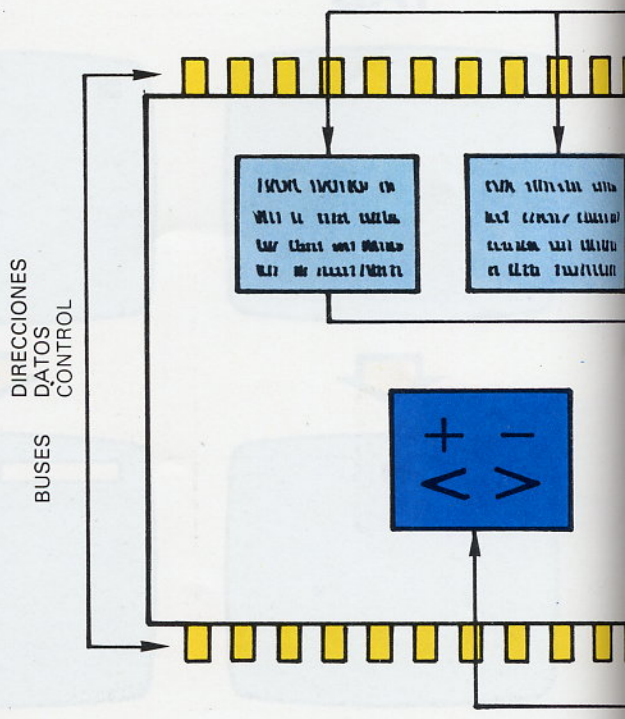
Cada componente realiza un trabajo específico, encaminado a solventar la «papeleta» que se le encomienda. ¿Qué labor realizan? ¿Cómo la ejecutan? ¿Dónde están situados? Vamos a contestar todas estas preguntas.

La organización de cualquier ordenador obedece básicamente al siguiente esquema: mediante un dispositivo de entrada, por ejemplo, el teclado o el casete, se envían el programa y los datos a la memoria central. La C.P.U. *(Central Processing Unit,* Unidad Central de Proceso), es el elemento principal del sistema.

La C.P.U. está compuesta por la C.U. *(Control Unit,* Unidad de Control), que regula el funcionamiento de todos los elementos del ordenador y la A.L.U. *(Arithmetic-Logic Unit,* Unidad Aritmético-Lógica) que se encarga de efectuar las operaciones básicas del computador. Entre estos elementos se realizan las transferencias principales de información, y sobre ellos recae el mayor peso en cualquier trabajo de proceso de datos. Finalmente, un dispositivo de salida, como la pantalla o la impresora, muestra los resultados obtenidos. Pero, entre la memoria y los dispositivos de entrada/salida, otros muchos componentes intervienen en la preparación de la información que ha de ser enviada o recibida. Vamos a ver dónde están albergados cada uno, y las funciones que desempeñan.

EL MICROPROCESADOR

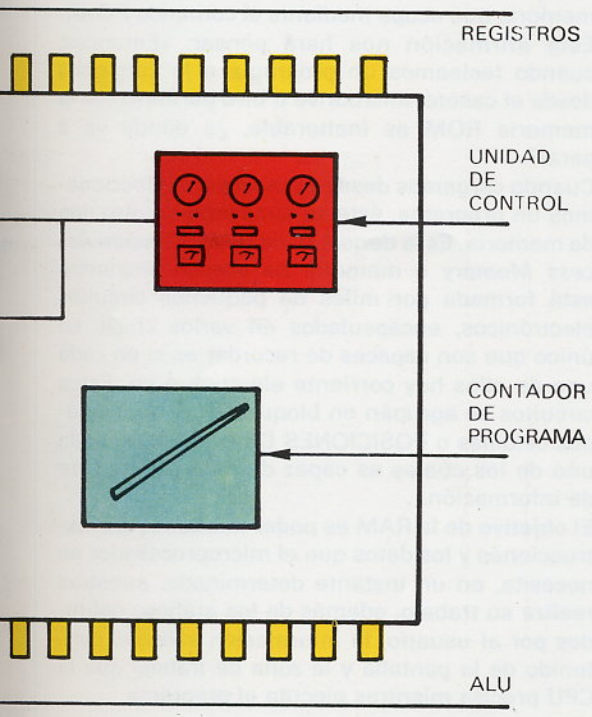
El cerebro de nuestro Spectrum es el microprocesador Z-80 A. Este es una versión más rápida del popular Z-80, fabricado por la firma ZILOG. Se trata de un conjunto de cerca de 8000 transistores integrados dentro de una pastilla (chip) de color negro, de la que sobresalen 40 patillas



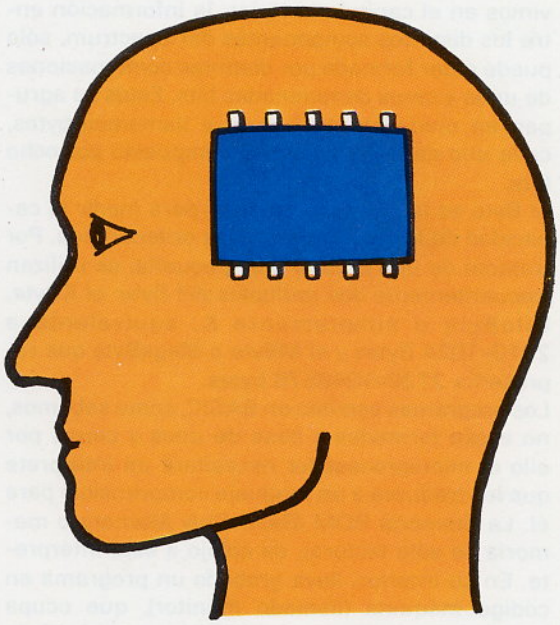
metálicas. En su interior se encuentran: la C.P.U., con la Unidad de Control y la Unidad Aritmético-Lógica; el registro de instrucciones, que la C.P.U. utiliza para anotar la instrucción que corresponde ejecutar; el contador de programa, que recuerda dónde está la siguiente celdilla de memoria donde recoger la próxima instrucción a ejecutar y, finalmente, los 24 registros libres para el usuario.

No entraremos en más detalles sobre los registros, que serán abordados adecuadamente en los capítulos dedicados al código máquina dentro de la sección BASIC.

La C.P.U., mediante su juego de instrucciones, puede realizar tareas como la lectura de datos desde un dispositivo de entrada, enviar datos hacia un dispositivo de salida, leer o escribir datos en la memoria del sistema, y efectuar operaciones aritmético-lógicas con los datos que le lleguen. Todo ello, a base de mover bits (1 ó 0) de un lado a otro.



Los componentes fundamentales del microprocesador Z-80 son: Unidad de Control, Unidad Aritmético-Lógica, Contador de Programa y Registros.



La C.P.U. es el «cerebro» de nuestro ordenador.

Todas estas operaciones, no podrían llevarse a cabo sin un elemento que fuera indicando a la Unidad de Control cuándo debe ejecutarlas, de forma que no se arme un fenomenal lío. Un circuito electrónico, que en el caso del Spectrum vibra 3,5 millones de veces cada segundo, hace las veces de reloj. Es algo así como el corazón del Spectrum; sincroniza todos los pasos que han de seguirse cada vez que el microprocesador debe ejecutar cualquier acción. De esta manera, el Z-80 A es capaz de realizar 875.000 instrucciones de las más sencillas cada segundo.

Fabuloso..., podríamos pensar. Sin embargo, no debemos creer que estas instrucciones son las mismas que utilizamos cuando escribimos un programa en BASIC. Entonces, ¿por qué se ejecuta nuestro programa, si el microprocesador no lo entiende? La razón se encuentra en otro chip, denominado memoria ROM.

¡!

LAS MEMORIAS

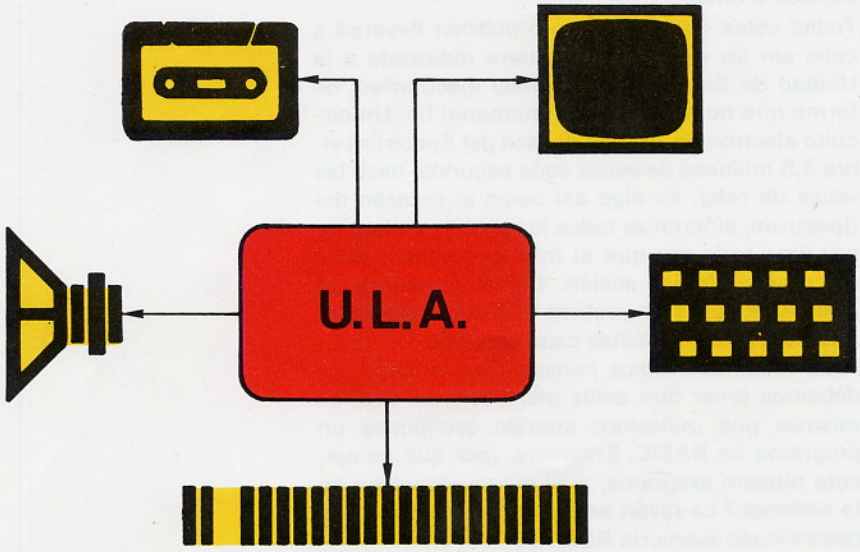
En el proceso de emisión de imagen, los componentes del Spectrum que intervienen son:
* U.L.A.: Toma la imagen de la zona de la memoria R.A.M. destinada a su almacenaje, y la envía al codificador P.A.L.
* P.A.L.: Convierte las señales eléctricas recibidas de la U.L.A. en una señal de vídeo con destino al modulador.
* Modulador: Modula la señal de vídeo recibida del P.A.L., y la envía por el canal 36 de U.H.F.

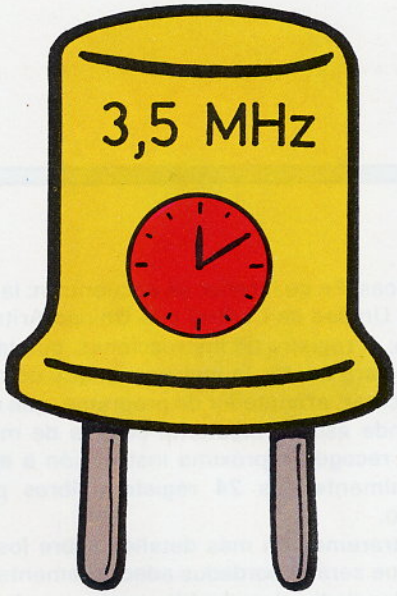
Antes de continuar, conviene recordar lo que ya vimos en el capítulo anterior: la información entre los distintos componentes del Spectrum, sólo puede estar formada por distintas combinaciones de unos y ceros denominadas bits. Estos se agrupan en unidades mayores que llamamos bytes, cada uno de los cuales está compuesto por ocho bits.

El Byte es la unidad empleada para medir la capacidad de las memorias de los ordenadores. Por tratarse de una unidad muy pequeña, se utilizan frecuentemente dos múltiplos del Byte: el KByte, KiloByte o simplemente K, equivalente a 2↑10=1024 Bytes y el MByte o MegaByte que representa 2↑20=1048576 bytes.

Los programas escritos en BASIC, como sabemos, no están formados a base de unos y ceros, por ello el microprocesador necesitará un intérprete que los traduzca a un lenguaje comprensible para él. La memoria ROM *(Read Only Memory* o memoria de sólo lectura), da cobijo a este intérprete. En su interior, lleva grabado un programa en código máquina (llamado monitor), que ocupa 16K Bytes de información (16384 bytes), desde la dirección 0 hasta la 16383.

La U.L.A. relaciona todos los periféricos con la C.P.U.





El reloj interno del Spectrum emite tres millones y medio de impulsos por segundo, marcando la cadencia de funcionamiento del ordenador.

De ellos, 7K se dedican al sistema operativo, 8K al intérprete de BASIC, y la K restante al generador de caracteres. En su interior, van incluidas las rutinas de entrada/salida, el control de la sintaxis en las expresiones BASIC, las rutinas de altavoz y teclado, etc.

Su contenido es inalterable; por más que nos empeñemos en conectar y desconectar la alimentación, o en intentar modificar las direcciones de memoria que ocupa mediante el comando **POKE**.

Esta afirmación nos hará pensar: «Entonces, cuando tecleamos un programa o lo cargamos desde el casete, microdrive u otro periférico, si la memoria ROM es inalterable, ¿a dónde va a parar?»

Cuando cargamos desde el casete o confeccionamos un programa, éste se almacena en otro tipo de memoria. Es la denominada RAM *(Random Access Memory* o memoria de acceso aleatorio); está formada por miles de pequeños circuitos electrónicos, encapsulados en varios chips. Lo único que son capaces de recordar es si en cada uno de ellos hay corriente eléctrica o no. Estos circuitos se agrupan en bloques de ocho, llamados celdillas o POSICIONES DE MEMORIA, cada uno de los cuales es capaz de albergar un byte de información.

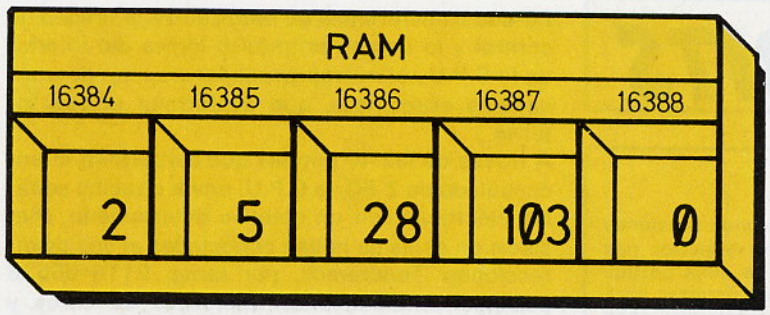
El objetivo de la RAM es poder mantener las instrucciones y los datos que el microprocesador no necesita, en un instante determinado, mientras realiza su trabajo, además de los gráficos definidos por el usuario, la información sobre el contenido de la pantalla y la zona de trabajo que la CPU precisa mientras ejecuta el programa.

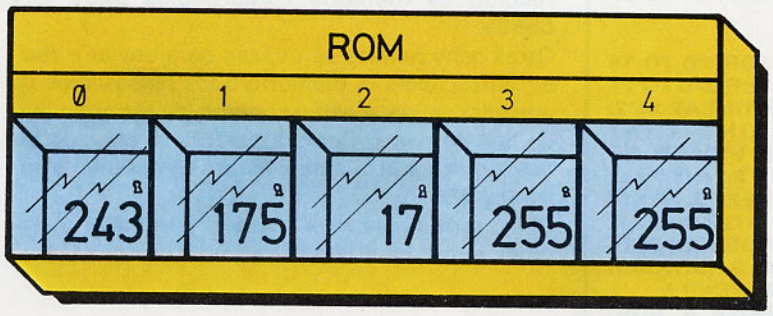
Cada celdilla dentro de la memoria lleva asignada una dirección, con objeto de identificar en cada momento su contenido. Es algo así como un enorme casillero, con un byte dentro de cada casilla. El primer byte de la memoria RAM se encuentra en la dirección 16384, mientras que el último dependerá del modelo de Spectrum (32767 en el de 16K, y 65535 en el de 48K). Las primeras 16K de memoria RAM están formadas por 8 chips, con 2K de memoria cada uno. En el Spectrum de 48K, las 32K adicionales van integradas en otros 8 chips, pero estos pueden almacenar, cada uno, 4K de información.

Incluso en el caso de tener conectado nuestro ordenador a la alimentación, el contenido de cada celdilla pierde intensidad con el paso del tiempo, debido a fugas de corriente internas. Así pues, debe ser sometido a un proceso de «refresco»; el Spectrum tiene prevista tal eventualidad y cada cierto período de tiempo, nunca superior a dos milisegundos, el microprocesador accede a cada dirección de memoria leyendo su contenido y volviéndolo a escribir.

Se trata de memorias volátiles, es decir, en el momento que cortemos la alimentación, se pierde lo que allí hubiera almacenado, al contrario de lo que ocurre en la ROM, cuyo contenido permanece inalterable.

Resumiendo, el Z-80 ejecuta las instrucciones que le van marcando los programas almacenados en las memorias RAM y ROM. La ROM se encarga de interpretar las líneas escritas en BASIC (presentes en la RAM), liberando una serie de órdenes comprensibles por el microprocesador, de manera que éste pueda realizar su trabajo.





En las posiciones de memoria ROM podemos leer datos, pero no escribirlos; al contrario que en la RAM, en la que podemos efectuar ambas operaciones.

LA U.L.A.

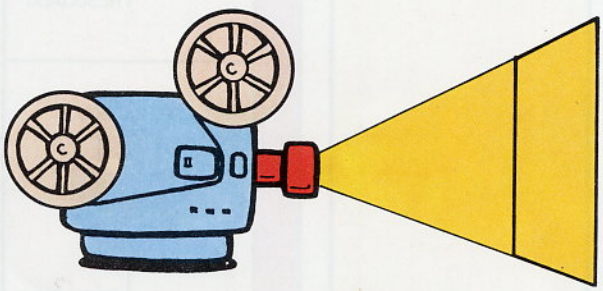
Uno de los componentes que justifica el pequeño tamaño y reducido costo de nuestro Spectrum es la U.L.A. *(Uncommited Logic Array)*. Este chip, viene a sustituir un gran número de componentes electrónicos más sencillos, con los que se coordinan las labores de entrada/salida de datos. Puede detener el funcionamiento del reloj de manera que no se produzcan interferencias, en caso que la C.P.U. intente acceder, simultáneamente, a la misma dirección de memoria dentro del fichero de representación visual.

En el proceso de emisión de imagen hacia el televisor intervienen tres elementos fundamentales: la U.L.A., el codificador P.A.L. y el modulador de U.H.F.

TRANSFERENCIAS DE INFORMACION

La comunicación entre los distintos componentes que hemos analizado, se realiza a través de una serie de conductores denominados buses. Son los encargados de que la información circule desde y hasta los dispositivos de entrada/salida a los que se desea acceder.

ARCHIVO DE PRESENTACION VISUAL



BITS

En muchas ocasiones nuestro televisor nos juega la mala pasada de desajustarse; para estos momentos, nada mejor que una buena carta de ajuste que permita afinar los colores.

```
10 FOR I=0 TO 15
20 FOR J=0 TO 21
30 PRINT AT J,I*2;
BRIGHT (INT
(I/2)=I/2); INK
(I-1)/2;"■■"
40 NEXT J: NEXT I
50 BORDER 1:
BORDER 2: BOR-
DER 3: BORDER
4: BORDER 5:
BORDER 6: BOR-
DER 5: BORDER
4: BORDER 3:
BORDER 2:
BODER 1:::::GO
TO 50
```

*

Combinando adecuadamente la sentencia **PAPER** y el separador coma (,), podemos evitar en algunas ocasiones escribir líneas en blanco mediante la sentencia **PRINT**. Por ejemplo, la instrucción **PRINT PAPER 2,,** es equivalente a **PRINT PAPER 2;"32 espacios en blanco"**, pero con una ocupación de espacio considerablemente menor.

Para la transferencia de datos entre la unidad de control y la unidad aritmético-lógica del interior de la C.P.U., el microprocesador dispone de unos canales específicos, que conforman el bus interno.

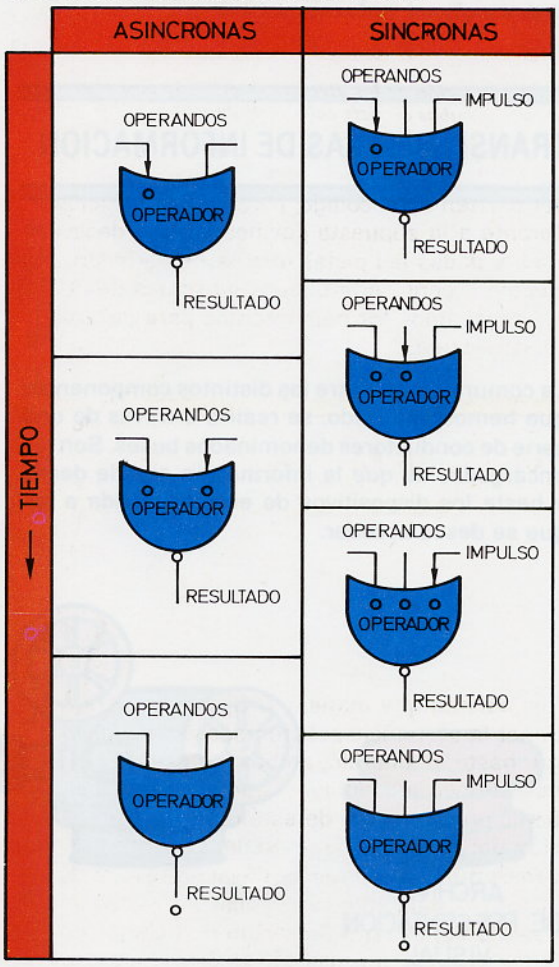
A través de las 40 patillas que conforman el encapsulado de Z-80, la C.P.U. envía o recibe señales eléctricas con un objetivo determinado. Dieciseis de estas se hallan conectadas al bus de direcciones. Tendremos, por tanto, 2↑16=65536 posibles combinaciones diferentes de ceros y unos, según pase corriente o no por cada una de ellas. Es decir, se puede acceder a cualquier dirección de memoria comprendida entre 0 y 65535.

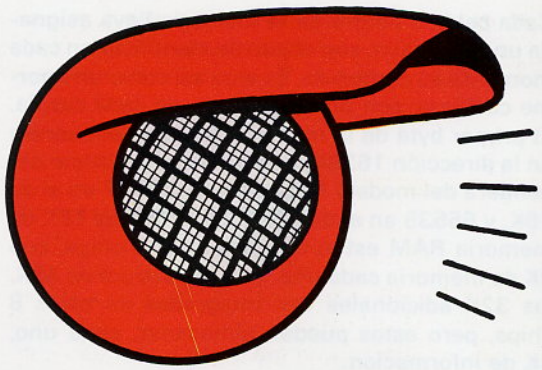
Otras ocho patillas se utilizan para enviar o recibir datos hacia la memoria o los dispositivos de entrada/salida, y están conectadas al denominado bus de datos. Como 2↑8=256, el valor de los datos que por él circulan, estará comprendido entre 0 y 255.

Cinco terminales más, reciben y transfieren señales que tienen que ver con el funcionamiento interno del microprocesador, conformando el denominado bus de control.

A los buses de datos, direcciones y control, se puede acceder directamente desde el exterior, a través de la tarjeta de expansión (port de expansión). Este elemento es el nexo de unión entre el Spectrum y los periféricos.

El ordenador realiza los procesos de manera síncrona, es decir, hace esperar la ejecución de una operación hasta que llega la señal del reloj que le da vía libre.





La emisión de sonidos en el Spectrum se efectúa a través de un pequeño zumbador (buzzer).

OTROS COMPONENTES

Las señales que se reciben o parten hacia el casete, necesitan de un interface que realice la conversión analógica/digital o digital/analógica, para transformar las señales sonoras en eléctricas binarias o viceversa. De esta manera, la C.P.U., durante el proceso de carga, realiza ciclos de lectura sucesivos, asesorada por la U.L.A., que transfiere la información al bus de datos. Es interesante señalar que en este caso el flujo de información circula en serie, es decir, bit a bit.

El modulador de color es controlado igualmente por la U.L.A., para que genere las señales correspondientes a los ocho colores de que dispone el Spectrum. A partir de él, se emite la señal de alta frecuencia que finalmente recibe la antena del televisor.

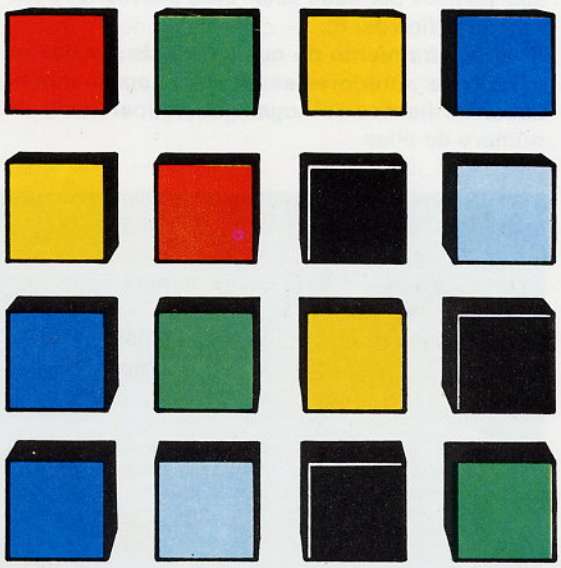
Por último, cabe mencionar la fuente de alimentación, que se encarga de suministrar al ordenador la tensión necesaria para su correcto funcionamiento. A partir del transformador de corriente exterior, que proporciona 9 voltios, el circuito 7805, colocado en el interior del Spectrum (sobre una placa metálica disipadora de calor), regula la tensión a los niveles de operación adecuados para los componentes del ordenador.

MASTER CUBOS

CUB es nuestra versión particular para RUN del conocido juego Mastermind. Por si aún queda alguien que no conozca las reglas de este pasatiempo, cosa bastante improbable, pasaremos a exponer rápidamente las mismas.

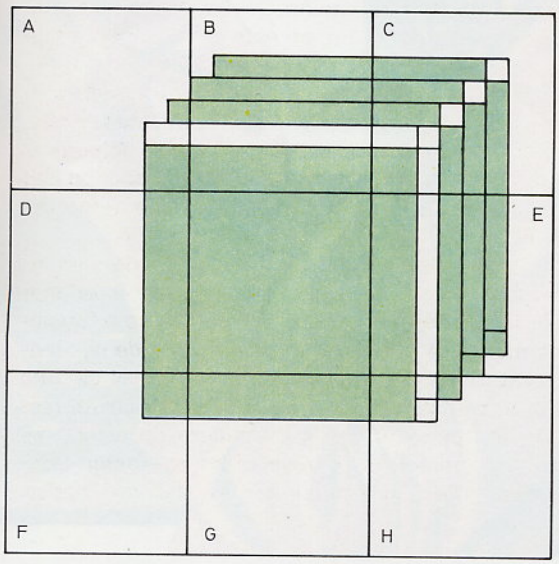
Ante todo es necesaria la presencia de dos jugadores. Uno de ellos pensará una secuencia de cuatro colores, con la condición de no repetir ninguno de ellos. En el caso concreto de CUB, este primer jugador será nuestro fiel compañero: el Spectrum.

El jugador restante, deberá adivinar dicha secuencia de colores, ejecutando combinaciones de cuatro elementos, que pueden ser o no repetidos. Lógicamente, en CUB este segundo participante seremos nosotros mismos. Para nuestras tareas detectivescas contamos con dos indispensables ayudas: por una parte, conocemos los diferentes colores con los que se ha podido formar el código secreto (siete colores), y por otra, por cada vez que realizamos un intento de descubrir el código, el Spectrum nos facilita una pista sobre lo cerca o lejos que estamos de la solución correcta. Estas pistas consisten en indicar qué colores están bien colocados en nuestra proposición, y cuáles existen en el código, pero en una posición diferente a la supuesta por nosotros. A decir verdad, y dadas las pistas que se nos brindan, nos podemos considerar buenos jugadores de CUB si no excedemos los seis intentos para adivinar la clave secreta.



La presentación de los diversos intentos, se lleva a cabo mediante cubos de siete colores diferentes.

Para la presentación de los cubos en la pantalla, se emplean los nueve gráficos definidos de la figura.



EL PROGRAMA

Los colores que maneja el Spectrum para componer la secuencia policromada van desde el negro hasta el amarillo, ambos inclusive. El blanco está descartado, debido a que el fondo de pantalla es precisamente de este color, y la presentación del mismo sería invisible. Echando cuentas vemos que los colores a utilizar son siete: negro, azul, rojo, magenta, verde, cian y amarillo.

Realmente, nuestro Spectrum no trabaja con colores propiamente dichos, sino con los códigos

¡!

Al introducir el programa, no olvidemos sustituir los caracteres subrayados, por los gráficos definidos correspondientes a las teclas indicadas.

*

Durante la ejecución del programa, si pulsamos la tecla SPACE, aparecerá en la pantalla cada uno de los intentos introducidos con anterioridad.

*

Una vez introducido el programa correctamente, podemos grabarlo mediante el comando **SAVE "CUB" LINE 1070.**

*

En las pistas facilitadas por el ordenador, un cuadrado azul representa un acierto parcial, y dos cuadraditos intermitentes en diagonal, un acierto completo.

que corresponden a los mismos, aunque esto es un detalle del programa que no nos afecta en el modo de operar con él.

Cada una de las secuencias de colores que nosotros introduzcamos, será correspondida por el Spectrum con otra secuencia de pistas:

a) Un cuadrado azul por cada color existente pero mal colocado.

b) Dos cuadritos intermitentes unidos por dos de sus vértices por cada color bien posicionado (carácter gráfico del 6).

El encuadramiento de cualquiera de las dos indicaciones anteriores es aleatorio, por lo que no debemos fiarnos del lugar que ocupen, sino del número de ellas.

En el momento en que la pantalla está saturada de información, el ordenador efectúa un desplazamiento de la misma hacia arriba *(scroll)* de cinco líneas. Esto hará que vayamos perdiendo de vista las primeras pistas.

Si durante el juego queremos saber cuáles han sido las jugadas anteriores, que han podido desaparecer de la pantalla por el efecto de *scroll* antes citado, pulsaremos SPACE (barra espaciadora en el PLUS). Inmediatamente aparecerán en la pantalla, por orden de introducción, cada uno de los intentos fallidos con su pista correspondiente.

CUB puede «recordar» un total de 40 intentos, número más que suficiente para averiguar la secuencia secreta. Esto es debido a las instrucciones **DIM I(40,4)** de la línea 60 y **DIM R(40,4)** de la línea 70, que controlan, respectivamente, las series de intentos y pistas memorizadas. Este programa puede ser utilizado tanto en el modelo de 48 K como en el de 16 K; y en ambos casos, debido a la limitación de las matrices de memorización, el programa se detendrá con el mensaje *Subscript wrong* en caso de que se llegue a rebasar este número de tope de intentos; y es que 40 intentos para adivinar cuatro colores... ¡ya está bien!

Para la introducción de los colores en nuestra combinación, utilizaremos las teclas numéricas, según el código de color que cada una de ellas tiene asignado. Hemos de tener cuidado en esta operación, ya que una vez introducido un código, no se contempla la posibilidad de su borrado.

Un dato anecdótico a subrayar en el programa, es la forma en que el mensaje de autoría del programador aparece en la pantalla, así como la forma de utilizar la sentencia **RANDOMIZE** en la línea 1290 para que los cubos que aparecen en la presentación del programa sean siempre del mismo color, regenerándolos cada vez que se ejecuta el programa mediante la función **RND**, sin ocupar un espacio de memoria innecesario con una **DATA**.

Y hablando de subrayar en el programa, no olvidemos sustituir los caracteres que aparecen de este modo en el listado, por los gráficos definidos de las teclas correspondientes. En cuanto a los números doblemente subrayados, recordemos que son los gráficos cambiados de las teclas afectadas, es decir, los que se obtienen pulsando simultáneamente la tecla **CAPS SHIFT** y el número, al encontrarnos en el modo gráfico.

Para almacenar el programa en cinta, utilizaremos el comando **SAVE "CUB" LINE 1070**, con el fin de que al cargarse se produzca una autoejecución a partir de la línea 1070.

```
  10 REM ************************
  20 REM * J.M.MAYORAL SERRANO *
  30 REM ************************
  40 BORDER 7: PAPER 7: CLS
  50 REM  SPECTRUM ELIGE NUMERO
  60 LET M$="    ": LET I$="    ": DIM I(40,4): DIM M
(4): DIM D(7)
  70 DIM R(40,4)
  80 LET FILA=1
  90 LET F1=0
 100 LET SW=0
 110 FOR N=1 TO 4
 120 LET NUM=INT (RND*7)
 130 IF D(NUM+1) THEN  GO TO 120
 140 LET D(NUM+1)=1: LET M(N)=NUM: LET M$(N)=STR$ NUM
 150 NEXT N
 160 CLS
 170 REM    JUEGO
 180 FOR I=1 TO 100
 190 PRINT  PAPER 2; FLASH 1;AT 21,0;"       INTRODUC
E NUMERO"
 200 FOR C=1 TO 4
 210 LET K$=INKEY$
 220 IF INKEY$=" " THEN  GO SUB 790
 230 IF CODE K$<48 OR CODE K$>54 THEN  GO TO 210
 240 BEEP .2,35
 250 LET I$(C)=K$: LET I(I,C)=VAL K$
 260 PRINT AT FILA,C+17;K$
 270 NEXT C
 280 LET F1=F1+1
 290 PRINT  PAPER 7;AT 21,0;"
"
 300 GO SUB 630
 310 GO SUB 420
 320 IF FILA>21 THEN  GO TO 350
 330 IF SW=1 THEN  RETURN
 340 NEXT I
 350 LET FILA=FILA-3
 360 POKE 23692,5
 370 FOR Q=1 TO 5
 380 PRINT
```

```
390 NEXT Q
400 IF SW=1 THEN  RETURN
410 NEXT I
420 REM VERIFICACION DE CODIGO
430 FOR Q=1 TO 4
440 PRINT  PAPER 7;AT FILA-3,Q+17;" "
450 NEXT Q
460 IF I$=M$ THEN  GO TO 1710
470 FOR Q=1 TO 4
480 FOR W=1 TO 4
490 IF I$(Q)=M$(W) AND Q=W THEN  LET R(F1,Q)=1
500 IF I$(Q)=M$(W) AND Q<>W THEN  LET R(F1,Q)=-1
510 NEXT W
520 NEXT Q
530 LET PL=1
540 FOR Q=4 TO 1 STEP -1
550 IF R(F1,Q)>0 THEN  GO TO 590
560 IF R(F1,Q)<0 THEN  GO TO 610
570 NEXT Q
580 RETURN
590 PRINT AT FILA-3,13+PL*2; FLASH 1; INK 2; PAPER 6
;"6"
600 LET PAS=F1*RND: LET PL=PL+1: GO TO 570
610 PRINT AT FILA-3,13+PL*2; INK 5;"8"
620 LET PL=PL+1: GO TO 570
630 REM DISPLAY JUGADA
640 LET COLUM=1
650 FOR N=1 TO 4
660 PRINT AT FILA-1,COLUM-1; INK I(F1,N);"A"
670 PRINT AT FILA-1,COLUM; INK I(F1,N);"B"
680 PRINT AT FILA-1,COLUM+1; INK I(F1,N);"C"
690 PRINT AT FILA,COLUM-1; INK I(F1,N);"D"
700 PRINT AT FILA,COLUM; PAPER I(F1,N);I(F1,N)
710 PRINT AT FILA,COLUM+1; INK I(F1,N);"E"
720 PRINT AT FILA+1,COLUM-1; INK I(F1,N);"F"
730 PRINT AT FILA+1,COLUM; INK I(F1,N);"G"
740 PRINT AT FILA+1,COLUM+1; INK I(F1,N);"H"
750 LET COLUM=COLUM+3
760 NEXT N
770 LET FILA=FILA+3
780 RETURN
790 REM   LISTADO DE JUGADAS
800 CLS
810 LET FILA=1
820 LET SW=1
830 LET FF1=F1
840 LET F1=0
850 FOR S=1 TO 4
860 LET F1=F1+1
870 GO SUB 320
880 GO SUB 630
890 GO SUB 530
900 IF F1=FF1 THEN  GO TO 930
910 NEXT S
920 GO TO 850
930 LET SW=0: LET F1=FF1
940 IF INKEY$="" THEN  GO TO 940
950 CLS : LET FILA=1
960 PRINT  PAPER 2; FLASH 1;AT 21,0;"        INTRODUC
E NUMERO "
970 RETURN
980 REM  SBR. PRESENTACION
990 DATA 0,0,0,0,1,0,3,3
1000 DATA 0,0,127,255,255,0,255,255
1010 DATA 0,0,248,244,236,28,220,220
1020 DATA 3,3,3,3,3,3,3,3
1030 DATA 220,220,220,220,220,220,220,216
1040 DATA 3,3,0,0,0,0,0,0
1050 DATA 255,255,0,0,0,0,0,0
1060 DATA 208,192,0,0,0,0,0,0
1070 LET A$="ABCDEFGH"
1080 FOR N=1 TO LEN A$
1090 FOR F=0 TO 7
1100 READ A
1110 POKE USR A$(N)+F,A
1120 NEXT F
1130 NEXT N
1140 BORDER 7: PAPER 7: INK 9:
1150 CLS : GO TO 1280
1160 REM
1170 LET CO=INT (RND*7)
1180 PRINT  INK CO;AT F-1,C-1;"A"
1190 PRINT  INK CO;AT F-1,C;"B"
1200 PRINT  INK CO;AT F-1,C+1;"C"
1210 PRINT  INK CO;AT F,C-1;"D"
1220 PRINT  PAPER CO;AT F,C;" "
1230 PRINT  INK CO;AT F,C+1;"E"
1240 PRINT  INK CO;AT F+1,C-1;"F"
1250 PRINT  INK CO;AT F+1,C;"G"
1260 PRINT  INK CO;AT F+1,C+1;"H"
1270 BEEP .1,30: RETURN
1280 REM
1290 RANDOMIZE 2: DIM P(13,29)
1300 DATA 1,1,1,1,0,1,1,1,1
1310 DATA 1,0,0,1,0,1,1,0,1
1320 DATA 1,0,0,1,0,1,1,1,0
1330 DATA 1,0,0,1,0,1,1,0,1
1340 DATA 1,1,1,1,1,1,1,1,1
1350 DATA 127,32,74,46,77,46,32,77,65,89,79,82,65,76
1360 DATA 32,83,69,82,82,65,78,79
1370 FOR F=1 TO 13 STEP 3
1380 FOR C=1 TO 7 STEP 3
1390 READ A
1400 LET P(F,C)=A
1410 NEXT C
1420 FOR U=12 TO 18 STEP 3
1430 READ A
1440 LET P(F,U)=A
1450 NEXT U
1460 FOR B=23 TO 29 STEP 3
1470 READ A
1480 LET P(F,B)=A
1490 NEXT B
1500 NEXT F
1510 REM
1520 FOR F=1 TO 13 STEP 3
1530 FOR C=1 TO 29
1540 IF P(F,C)=0 THEN  GO TO 1560
1550 GO SUB 1160
1560 NEXT C
1570 NEXT F
1580 REM
1590 FOR C=1 TO 30
1600 PRINT  BRIGHT 1; INK 6; PAPER 2;AT 20,C;"3"
1610 PRINT  BRIGHT 1; INK 5; PAPER 4;AT 21,C;"3"
1620 NEXT C
1630 REM
1640 FOR N=1 TO 22
1650 READ A
1660 PRINT AT 17,N+4;CHR$ A
1670 NEXT N
1680 FOR N=1 TO 500
1690 NEXT N
1700 RUN
1710 REM  CODIGO ACERTADO
1720 GO SUB 470
1730 FOR N=1 TO 50
1740 NEXT N
1750 CLS
1760 LET FILA=10
1770 LET COLUM=10
1780 GO SUB 650
1790 PLOT 60,70
1800 DRAW 120,0
1810 DRAW 0,45
1820 DRAW -120,0
1830 DRAW 0,-45
1840 LET K=1
1850 LET FILA=7
1860 LET PL=-1: GO SUB 540
1870 PRINT  PAPER 6;AT 17,1;"LO HAS RESUELTO EN ";F1;
" INTENTOS!"
1880 PRINT  INK 4;AT 21,0;"  PULSA UNA TECLA PARA EMP
EZAR "
1890 FOR C=1 TO 31
1900 PRINT  PAPER 6; INK 9; OVER 1;AT 17,C;" "; INK 9
; PAPER K; OVER 1;AT 17,C-1;" "
1910 BEEP .01,C
1920 IF INKEY$="" THEN  GO TO 1940
1930 GO TO 1970
1940 NEXT C: PRINT  PAPER K;AT 17,31;" "
1950 LET K=K+1: IF K>5 THEN  LET K=1
1960 GO TO 1890
1970 RUN 50
```

1er CERTAMEN DE INFORMATICA JUVENIL

tu Micro Feria PRIMAVERA 85

LOS DIAS 13 Y 14 DE JUNIO TIENES UNA CITA CON LA INFORMATICA EN EL HOTEL PRINCESA PLAZA. PRINCESA, 40.

ALLI SE CELEBRA LA PRIMERA FERIA DE MICROS Y SOFTWARE ESPECIALMENTE PENSADA PARA LA GENTE JOVEN.

EN «TU MICROFERIA» DE PRIMAVERA TENDRAS A TU DISPOSICION TODOS LOS MICROS, PERIFERICOS Y PROGRAMAS QUE PUEDAS IMAGINAR. Y ESTARAN ALLI PARA JUGAR.
CON ELLOS, O PARA QUE CONOZCAS LOS NUEVOS PROGRAMAS DE GESTION PARA ORDENADORES PERSONALES O —SI LO PREFIERES— EL SOFTWARE EDUCATIVO. ADEMAS HABRA INTERESANTES SORTEOS ENTRE NUESTROS VISITANTES.

HOTEL PRINCESA PLAZA. PRINCESA, 40.
MADRID 13-14 DE JUNIO.
HORARIO: 11 A 21 H.

todos los programas
todos los ordenadores

INVITACION

IMPORTANTE: RELLENA EL FORMULARIO QUE FIGURA AL DORSO Y PARTICIPA EN EL SORTE DE

EDICIONES INGELEK S. A. TIENE EL GUSTO DE INVITARLE AL PRIMER CERTAMEN DE INFORMATICA JUVENIL

tu Micro Feria PRIMAVERA 85

1er CERTAMEN DE INFORMATICA JUVENIL

13-14 DE JUNIO. HOTEL PRINCESA PLAZA. PRINCESA, 40. HORARIO: 11 A 21 H. MADRID.