AÑO II N.º 25

MICROHOBBY

AMSTRAD Semanal

REVISTA INDEPENDIENTE PARA USUARIOS DE ORDENADORES AMSTRAD

AÑO II N.º 25

160 Ptas.

Canarias **165** pts.

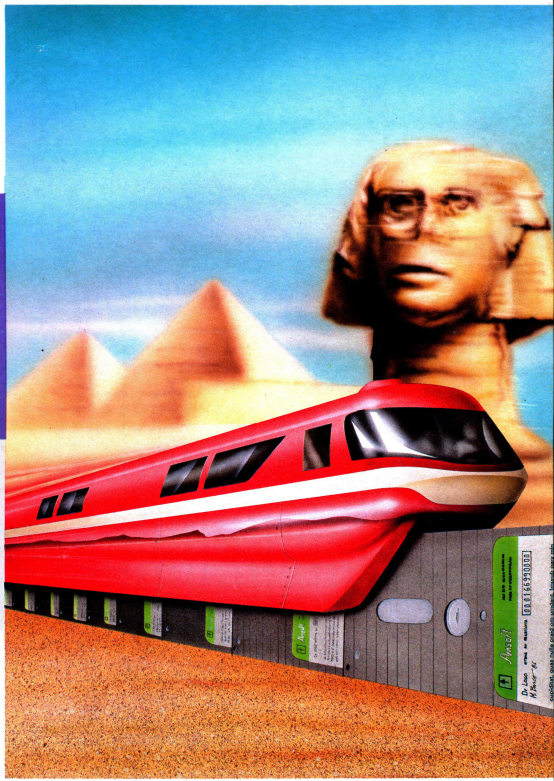
DAR FORMA AL SONIDO: LA ENVOLVENTE

LOS SECRETOS DEL DISCO EN CODIGO MAQUINA

Valor numérico de las operaciones lógicas

SOFTWARE

Sir Fred, un valeroso caballero made in Spain



HOBBY PRESS

&

¡Juegos de Simulación Estratégica para poner a prueba su inteligencia y vivir la emoción de situaciones reales!

¿Se atreve Ud. a dirigir la Campaña del Desierto y derrotar a Rommel antes de que lo hiciera el General Montgomery en el Alamein?

Un WAR GAME de estrategia que reúne todos los condimentos necesarios para hacer de él un auténtico *«plato fuerte»*, inteligencia, emoción, sorpresa y realismo.

Sólo 1.125 pts. (I.V.A. incluido)

RATAS del DESIERTO

Operación Norte de Africa

¡Desde uno a tres jugadores!

Disponible para

Spectrum

y

Amstrad

HOBBY PRESS
Para gente inquieta.

um 48, Plus, 128 K

Recorte o copie este cupón y envíelo a Hobby Press, S. A. Apdo. de Correos 54.062. Madrid. Deseo recibir en mi domicilio **sin gastos de envío alguno por mi parte**, la cinta RATAS DEL DESIERTO al precio de 1.125 ptas. (I.V.A. incluido).

Nombre y Apellidos
Dirección
Localidad
Código
Provincia
Teléfono
Edad
El modelo que elijo es para Amstrad Spectrum
La forma de pago elegida es la que señalo con una cruz
Giro postal nº
Tarjeta Visa nº
Talón nominativo a HOBBY PRESS, S. A.
Fecha de caducidad de la tarjeta
Fecha y firma

Director Editorial
José I. Gómez-Centurión
Director Ejecutivo
Victor Prieto
Subdirector
José María Díaz
Redactora Jefe
Marta García
Diseño
José Flores
Colaboradores
Francisco Portalo, Pedro Sudón
Miguel Sepúlveda,
Francisco Martín,
Jesús Alonso, Pedro S. Pérez
Amalio Gómez
Juan J. Martínez,
David Sopuerta, Alberto Suñer,
Eduardo R. Velasco
Secretaria Redacción
Carmen Santamaría
Fotografía
Carlos Candel
Javier Martínez
Portada
M. Barco
Ilustradores
Javier Igual, J. Pons, F. L.
Frontán, J. Septien, Pejo, J. J.
Mora, Luigi Pérez

Edita
HOBBY PRESS S.A.

Presidente
María Andrino
Consejero Delegado
José I. Gómez-Centurión

Jefe de Publicidad
Concha Gutiérrez
Publicidad Barcelona
José Galán Cortes
Tel: (93) 303 10 22/313 71 62

Secretaria de Dirección
Marisa Cogorro

Suscripciones
M.ª Rosa González
M.ª del Mar Calzada

Redacción, Administración y Publicidad
La Granja, 39
Polígono Industrial de Alcobendas
Tel.: 654 32 11
Telex: 49 480 HOPR

Dto. Circulación
Carlos Peropadre

Distribución
Coedis, S. A. Valencia, 245
Barcelona

Imprime
ROTEDIC, S. A. Crta. de Irún,
Km. 12,450 (MADRID)
Fotocomposición
Novocomp, S.A.
Nicolás Morales, 38-40
Fotomecánica
GROF
Ezequiel Solana, 16
Depósito Legal:
M-28468-1985

Representante para Argentina, Chile, Uruguay y Paraguay, Cía. Americana de Ediciones, S.R.L. Sud América 1.532. Tel.: 21 24 64. 1209 BUENOS AIRES (Argentina).

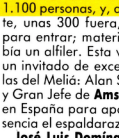
Se solicitará control OJD

MICROHOBBY
AMSTRAD
sumario

Año II • Número 25 • 18 de Febrero al 24 de Febrero de 1986
160 ptas. (incluido I.V.A.)
Canarias, 155 ptas + 10 ptas sobretasa aérea
Ceuta y Melilla, 155 ptas.

AMSTRAD ESPAÑA APUESTA POR EL PCW8256

El Scala del hotel Meliá Castilla fue de nuevo el escenario escogido por Indescomp para anunciar a bombo y platillo las buenas nuevas acerca del ordenador que, sorprendemente, «amenaza» con convertirse en la estrella de Amstrad para el difícil y competitivo campo de la gestión, de los llamados ordenadores serios.



El acontecimiento sólo se puede calificar como de masas, y probablemente no seamos capaces de transmitir al lector la enorme multitud que se congregó en el Scala del Meliá Castilla: dentro, unas 1.100 personas, y, desgraciadamente, unas 300 fuera, imposibilitadas para entrar; materialmente, no cabía un alfiler. Esta vez se contó con un invitado de excepción en las salas del Meliá: Alan Sugar, **chairman** y Gran Jefe de **Amstrad** se presentó en España para apoyar con su presencia el espaldarazo al PCW8256.

José Luis Domínguez, presidente de **Indescomp**, habló a los asistentes describiendo detalladamente la ascendente trayectoria de su compañía y de los productos **Amstrad**, cosa que nadie, hoy por hoy, puede negar: **Amstrad** es número 1 en Europa.

Domínguez explicó que la estrategia de su compañía y la de **Amstrad** sigue siendo la misma de siempre: ofrecer a los consumidores un ordenador llave en mano, listo para funcionar inmediatamente y al mínimo precio posible, esto es, 129.900 ptas. La idea subyacente está muy clara: aumentar el sector del mercado accesible a la informática, en general, y a **Amstrad** en particular.

Nuestros lectores recordarán que vaticinamos que el 8256 tendría software de utilidades y lenguajes a muy largo plazo; pues bien, tenemos la satisfacción de confesar que nos equivocamos completamente.

El **PCW8256** va a tener soft de todo tipo, y en cantidad.

¿**Ejemplos**? Microsoft Multiplán, DBaseII, toda la caterva de lenguajes y utilidades de Nevada y Digital Research (*PASCAL/MT+, DR GRAPAH, DR DRAW, etc)* y, lo que es más prometedor, todos rondarán las 15.000 ptas de precio y algunos ya están en la calle, como Multiplán.

Un programa del que es preciso hacer un aparte es un generador de bases de datos, el BORIAR situado en este margen de precio y cuyo equivalente IBM PC cuesta ciento y muchas mil pesetas.

Se espera que en un plazo de 2 ó 3 semanas todo esto, o gran parte, esté disponible, según fuentes de la propia Indescomp.

José Luis Domínguez explicó también a sus distribuidores que el **relanzamiento** del PCW obedece, entre otras cosas, a la posibilidad de convertir unos meses de malas ventas en unos de muy buenas ventas, gracias al ofrecimiento al público de un buen equipo.

Para apoyar sus intenciones con algo más que palabras, Domínguez mostró a la gente un resumen audiovisual de unas de las campañas de publicidad más masivas que se han echo nunca, verdaderamente impresionante.

El costo de la campaña oscilará entre unos 180 millones de pesetas, y abarcará radio, periódicos, tel:evisión, etc. La imagen del PCW estará en todas partes, según José Luis.

Después de la parte seria de la reunión, llegó la diversión y la alegría: se sortearon muchas cosas, algunas más grandes, otras más pequeñas, religiosamente entregadas por Alan Sugar y José Luis Domínguez.

Entre las grandes, una furgoneta, así, como suena, para algún distribuidor.

Entre las pequeñas, 7 ordenadores y 500.000 ptas. de publicidad gratis en la agencia de publicidad Arge, creadora de la campaña para el 8256.

También el público recogió con agrado *(se lo juro)* revistas e incluso suscripciones de regalo a **AMSTRAD SEMANAL** (para gran sorpresa de la redacción de la misma) y **Amstrad** User. Los maliciosos que sugieran que eso es imposible, recuerden que en un ambiente festivo y popular como el de la fiesta uno se siente inclinado a ser tolerante, ¡qué diablos! Para remachar el clavo, Martes y trece tuvo una actuación tan buena y caústica como siempre, y así, entre plato y plato, se fue pasando el rato.

Hablando de platos, aquí tienen el menú de la cena:

Pastel del chip.
Diskette de mariscos.
Software de escalpines flameante.
Soufle helado al Locoscript.
Café con Sugar.
Bebidas hasta 256K.
Periféricos.

Bromas aparte, el PCW8256, alias el Deseado, como Fernando VII, ya está entre nosotros, arropado con un plantel de software más que suficiente y a unos precios que creemos no exagerar al calificarlos de revolución.

MAS LENGUAJES

Los sistemas pequeños tienen derecho a poseer sistemas de desarrollo de software, como por ejemplo el SkyForth creado por Tosch.

Esta implementación del lenguaje Forth, de momento sólo existe para el Apple II, e incluye un editor de código fuente que a la vez, como en todos los sistemas Forth, es también el compilador. Esto permite crear, depurar y corregir programas rápidamente.

Los programas, una vez finalizados, pueden salvarse en el disco como código objeto y/o overlays.

En el paquete se incluye también un ensamblador completo y utilidades para la depuración de programas, junto con un programa de runtime que permite al código del compilador ejecutarse sin la presencia del propio compilador.

SkyForth puede manejar aritmética en coma flotante, enteros de 32-bits, estructuras de datos en forma de lista y gráficos basados en ventanas.

CP/M PARA MACINTOSH

La empresa norteamericana IQ Software comercializa para el Mac de 128 Kbytes el sistema operativo CP/M-68K.

IQ soft incluye, junto con el DOS, el compilador de C de Digital Research y el Macro Ensamblador.

La emulación del CP/M 2.2 *(sistema operativo del* ***Amstrad)***, también está disponible, pero requiere al **«Fat Mac»**, el de 512 K para poder ejecutarse.

Los discos usados por el CP/M-68K no son compatibles con el resto de discos del Mac.

NACE UN LENGUAJE DE PROGRAMACION

La empresa americana LUCID Software Corp. ha creado y comercializado un nuevo lenguaje de programación de alto nivel específicamente diseñado para aplicaciones de negocios. El recién nacido, llamado LUCID, está basado en el lenguaje de programación C, pero no tiene declaraciones de variables-tipo.

LUCID posee 188 funciones incluidas que le permiten, entre otras cosas, manejar la entrada/salida del ordenador, acceder a ficheros secuenciales y aleatorios, manipular cadenas alfanuméricas y funciones matemáticas.

El lenguaje funciona bajo los sistemas operativos MS-DOS, PC/IX, UNIX y XENIX. Por ahora, no existe una versión CP/M.

Hasta tal punto está pensado este lenguaje para negocios, que puede leer datos directamente de los programas dBASEII y dBASEIII, e intercambiarlos con Lotus 1-2-3, Simphony y Framework.

En cuanto a los precios, oscilan entre 300 y 400 dólares, según el ordenador y el sistema operativo.

Primera plana

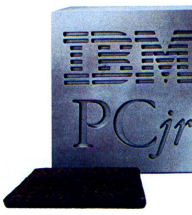
LENGUAJE SMALLTALK PARA PCs

Smalltalk es un lenguaje de programación orientado a objetos, basado en ventanas y **«pop-up»** menús *(los típicos menús en los que uno escoge un ítem desplazándose por él mediante las teclas del cursor o algo parecido)*.

Software Systems ha creado para IBM PC y compatibles el Method, una versión del Smalltalk 80 que viene con un compilador, un depurador y un editor de textos.

El paquete se ha configurado como un sistema de desarrollo, y requiere un PC con 512 Kbytes de RAM y dos unidades de disco de 360 Kbytes.

Se encuentran también en desarrollo versiones para el Apple II y sistemas basados en el 68000 de Motorola.



OPERACIONES BOOLEANAS

Todos los ordenadores realizan sus funciones trabajando internamente con ceros y unos* (sistema binario). *Vamos a intentar ver de una forma clara y detallada una serie de operaciones que actúan directamente sobre la representación binaria de cualquier número. Se trata de los «operadores lógicos».

En el artículo anterior vimos cómo podíamos relacionar dos o más condiciones mediante una serie de nuevas palabrejas. El que ahora les hablemos de OR, ANDZ, NOT y XOR no debe sonarles ya a chino y seguro que

```
IF A < B OR B > C THEN...
```

será una instrucción que no le causará ningún problema cuando se la encuentre dentro del listado de un programa.

Pero esta no es la única utilización de los operadores lógicos.

Operaciones lógicas con números: álgebra de Boole

Hasta ahora solamente hemos realizado con los números las clásicas operaciones aritméticas a las que todos ya estamos acostumbrados. Suponemos que ya serán capaces de decirle a su **Amstrad** que haga cualquier suma, por muy larga que sea, y que evaluar la fórmula del «interés simple» no tendrá ningún misterio.

Existen otra serie de operaciones que quizá se apartan de los conceptos clásicos que tenemos sobre cómo podemos manipular un número, o una serie de ellos. Se trata de las **«operaciones lógicas»**, también llamadas **«BOOLEANAS»** por pertenecer al Algebra de Boole. ¡No se asuste con el nombrecito!

¿Y, cuáles son sus características? Vamos a verlas. Dentro del Algebra de Boole sólo vamos a poder utilizar como operando dos valores diferentes **«ceros»** y **«unos»**. La solución también será de este estilo —no es bueno salirse de las normas—.

Es decir, vamos a realizar unas operaciones lógicas sobre unos operandos binarios —*cero o uno*— que nos van a dar resultados también binarios. Sólo esos valores.

No es muy difícil observar la semejanza que hay con el sistema binario de numeración en el que representamos cualquier número decimal como una sucesión de ceros y unos. Una suma de números binarios se haría sumando las cifras binarias (o bits) de cada uno de los órdenes *(o columnas)* correspondientes.

Pues estas operaciones lógicas se harían de una forma muy parecida. El **Amstrad** pasaría primeramente los operadores a binario —en realidad ya los tiene así en la memoria— y después operaría con las cifras binarias (o bits) de las columnas que se correspondan.

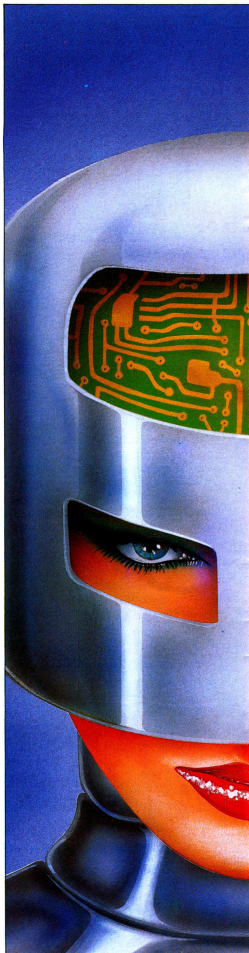
Por eso podemos decir que estos operadores lógicos actúan directamente sobre cada uno de los bits que forman parte de los operandos.

Pero hasta ahora no hemos visto nada sobre qué es lo que hace cada una de estas operaciones. Vayamos por partes.

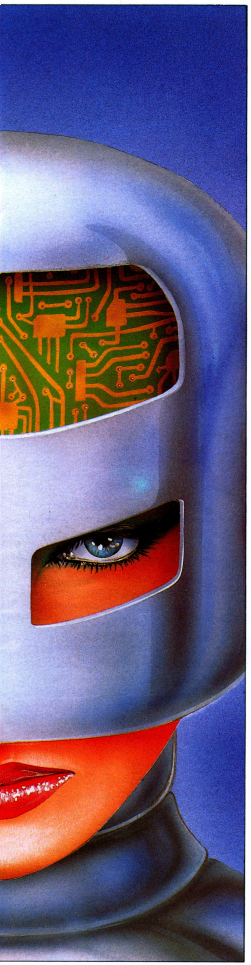
Operador AND

La primera de las palabras clave que vimos en el capítulo anterior fue AND. Empecemos también ahora con este operador.

La operación AND entre dos bits nos da como resultado 1 cuando ambos bits son 1. En cualquier otro caso, el valor que nos devuelve es 0.



Javier Igual



Según esta definición podemos construirnos una tabla de verdad semejante a las que vimos en el artículo anterior. Inténtelo. Recuerde, solamente nos dará 1 cuando los dos operandos sean 1.

Por si lo cree oportuno, echando un vistazo a la figura 1 comprobará si coincide con la suya. Esperamos que así haya sido.

Veámoslo ahora en la práctica. Nuestra tarea va a consistir en realizar la operación lógica:

12 AND 6

Si tiene prisa por conocer el resultado, teclee:

PRINT 12 AND 6

y en la pantalla aparecerá la solución: 4.

¿Cómo ha ocurrido esto? Primero pasemos a binario ambos números.

12 = 1100

y

6 = 0110

Sería muy interesante que ahora nosotros hiciéramos de ordenador y calculáramos el resultado de la operación. Cojemos estos números binarios y los escribimos en un papel como si se tratara de una operación aritmética clásica:

```
  1100
AND
  0110
  ----
```

y a continuación calculamos la solución operando con los dos bits *(o cifras binarias)* de cada una de las columnas.

Con la ayuda de la Tabla de Verdad de esta operación AND no le habrá sido muy difícil encontrar que:

0100

es el resultado correcto. Pero este número binario **¿a qué decimal es equivalente?** Si lo desarrolla comprobará que nuestro ordenador manual nos dará un número que, si no se ha equivocado, ha de coincidir con el 4 que nos dio nuestro **Amstrad**.

Intente hacer manualmente varios de estos ejemplos y después compruebe si los resultados son correctos utilizando el ordenador para obtenerlos.

¿Qué le parece si ahora hacemos un programita que nos muestre gráficamente cómo funciona todo esto? Vamos a ello.

Lo primero que hace es pedirnos los valores de los operandos —líneas 50 y 60. En la 80 calcula el resultado de la operación AND hecha entre los dos valores enteros que le hemos dado.

Las líneas 90 y 120 nos muestran gráficamente en la pantalla el método que habíamos seguido en el caso de realizar esta operación manualmente.

En la 90 calculamos e imprimimos una cadena de 8 dígitos binarios que representan el valor del primer operando *(contenido en la variable* ***«operando1»****)*. Lo hacemos por medio de la función BIN$ con dos parámetros.

Primeros pasos

El primero contiene el valor entero que queremos transformar y el segundo, el número de caracteres que van a formar la parte de la cadena.

¡Ojo con el uso de esta función! Deberá tener mucho cuidado al utilizarla y no olvidar que nos devuelve un valor que es una **«cadena de caracteres»** y que por tanto no podremos hacer operaciones matemáticas con dicho valor.

Del mismo modo actuamos en la línea 100, pero esta vez con el segundo operando —**«operando2»**—.

En la línea 120 imprimimos por un lado la cadena de dígitos binarios *(BIN$)* que representa el valor de la variable **«resultado»** y a continuación el valor decimal del mismo.

Creemos que después de hacer unos cuantos ejemplos con este Programa no tendrá ninguna duda a la hora de hacer esta nueva operación. No olvide que AND solamente nos da 1 cuando ambos operandos son 1.

Operador OR

Con su permiso vamos a pasar al siguiente operador. OR es una operación lógica que nos da cero cuandos los dos operandos tienen el valor 0. En cualquier otro caso el resultado será uno.

Basta, por tanto, que uno de los operandos sea 1 para que el valor que obtengamos con la operación sea también 1. No se puede decir que sea muy complicado, ¿verdad?

Esta definición la tiene resumida en la tabla II, donde podemos encontrar el resultado viendo los operandos que vamos a utilizar.

¿Sigue todavía con ganas de hacer de ordenador? La experiencia precedente no ha sido muy pesada e imaginamos que no tendrá inconveniente en volver a repetirla.

Vamos a realizar la operación lógica:

12 OR 6

Si como en el caso anterior teclea:

PRINT 12 OR 6

la respuesta es inmediata: 14.

Pero hemos quedado que primero lo vamos a hacer manualmente. ¡Ya haremos trabajar después al **Amstrad**!

Ya habíamos calculado antes los números binarios correspondientes a los operandos, por tanto tenemos un trabajo menos,

12 = 1100

y

6 = 0110

A continuación formamos la operación en el modo tradicional:

```
  1100
OR
  0110
  ----
```

y calculamos el resultado operando los dos dígitos binarios de cada una de las columnas conforme a los valores dados en la tabla II.

Y sin ningún problema sacaríamos el resultado:

```
  1110
```

que resulta ser el binario correspondiente al número 14 que calculó el **Amstrad**. No cabe duda que nuestro ordenador ha sido mucho más rápido pero es conveniente repetir unas cuantas veces este proceso para comprender la filosofía de estas operaciones lógicas. Hágamos caso y pruebe varios ejemplos. Los valores de los operandos se los dejamos a su elección.

Y una vez que ya sea un maestro en el manejo de esta operación, pese a ver el Programa II que le permitirá comprobar si los resultados obtenidos anteriormente son válidos o tiene que seguir practicando.

Su estructura es muy semejante a la del Programa I con la diferencia que ahora empleamos otra operación. Seguimos utilizando la función BIN$ para representar con una **«cadena»** de dígitos binarios todos los valores decimales. ¡No olvide que es una cadena!

La siguiente operación lógica de la que nos vamos a ocupar es NOT. **¿Qué es lo que hace?**

Operador NOT

El operador NOT aplicado a un dígito binario (o bit) invierte el valor que tiene dicho dígito. O sea, si vale 1 lo transforma en 0 o viceversa. Así de sencillo. Su Tabla de Verdad será también muy sencilla, puede comprobarlo en la tabla III.

Y, **¿cómo actúa sobre un número entero?**. Paciencia. Para verlo vamos a emplear el mismo método que hemos seguido hasta ahora: hacerlo manualmente. Por ejemplo, vamos a calcular:

NOT 6

El número binario equivalente es:

6 = 0110 *(ya lo teníamos antes)*

AND	OPERANDO 1	OPERANDO 2	RESULTADO
—	0	0	0
—	0	1	0
—	1	0	0
—	1	1	1

TABLA I

Y ahora nos toca a nosotros el trabajo de cambiar ceros por unos y unos por ceros. Luego:

NOT 6 = 1001

Esto es lo que en Algebra de Boole llamamos calcular el **«complemento a 2»** de un número cualquiera.

Con el Programa III lo verá mucho más gráficamente. El **Amstrad** supone que cada número entero está representado por 16 dígitos binarios, por eso será que en este Programa así los representamos. No se asuste que no es para tanto.

Suponemos que, vistos los anteriores, el seguirlo no le será nada difícil.

Con una pequeña regla práctica le podemos decir que el operador nos transforma un número entero siguiendo una sencilla formulita:

NOT 6 = —(6+1) = —7

Le recomendamos que pruebe que así ocurre con varios números ejecutando el Programa III: el **Amstrad** nunca falla, no lo olvide.

Y la última operación lógica que podemos realizar con nuestro ordenador es XOR. Como cualquier operación booleana solamente opera con dígitos binarios —ceros y unos— y su resultado también es una cifra binaria.

Este operador nos da **«uno»** cuando los operandos a los que afecta son diferentes: el primero es 1 y el segundo es 0 o viceversa.

En el momento que ambos operandos sean iguales —bien **«unos»** o bien **«ceros»**, el resultado de la operación es 0.

Modo de hacerlo. **¿Seguimos con el método del ordenador manual?** Sí, **¿verdad?**

Y, **¿para qué cambiar de números?** Volvemos con nuestros viejos conocidos 6 y 12 y a ver lo que sale.

```
  1100
  0110
XOR
  ----
```

Volvemos a colocar las cifras binarias como para una operación tradicional:

y el resultado será: 1010.

Hemos operado los dos bit que se encuentran en la misma columna conforme la definición dada para esta operación y sólo obtenemos el valor 1 cuando ambos kit son diferentes.

Si queremos, podemos hallar el número decimal equivalente al resultado: 10.

Cuando domine el mecanismo de esta operación teclee:

PRINT 12 XOR 6

y el ordenador calculará el valor 10 un poco más rápido que si nosotros lo hacemos manualmente.

La Tabla de Verdad de este operador podemos encontrarla en la tabla IV. Compruebe, si le parece, que está de acuerdo con la definición que hemos dado de la operación.

Como para los demás operadores lógicos, a continuación les damos un pequeño programa para que con el intente despejar las posible dudas —*grandes o pequeñas*— que todavía pueda tener.

OR	OPERANDO 1	OPERANDO 2	RESULTADO
—	0	0	0
—	0	1	1
—	1	1	1

TABLA II

Aunque no le contemos cómo funciona no tendrá ningún problema en seguirlo, **¿verdad?**

Bueno, ya hemos hecho un pequeño recorrido por todos los operadores lógicos, pero, **¿para qué vale todo esto?**

Teclee:

PRINT **«PEPE»** = **«JUAN»**

y se encontrará con la sorpresa de que el ordenador no le da ningún mensaje de error sintáctico sino que el valor 0 aparece en la pantalla.

Después de la instrucción PRINT tenemos la relación.

«PEPE» = **«JUAN»**

¿se cumple esta condición? Es evidente que no, ya que estamos comparando dos literales que son constantes y distintos. ¿Y el cero que ha salido en la pantalla?

La explicación a todo esto es muy sencilla. El ordenador primero ha evaluado si la condición se cumple o no. Cuando no se cumple, como en este caso, la evaluación da como resultado el cero que aparece en la pantalla. **¿Y si se cumple?** Compruébelo usted mismo con:

PRINT **«PEPE»** < > **«JUAN»**

Ahora que la condición se cumple —**«JUAN»** es siempre distinto de **«PEPE»**— en la pantalla aparece —1. Vamos a sacar conclusiones.

Cuando la condición es falsa, su evaluación da como resultado 0, que es una sucesión de bits iguales a 0. Sin embargo, si la condición es verdadera tenemos un —1 en la pantalla *(—1 es una sucesión de bits en la que todos son iguales a 1)*. **¿Va comprendiendo?**

Esto quiere decir que cuando relacionábamos dos condiciones, en el artículo anterior, por medio de algún operador lógico en realidad lo que estábamos haciendo era comprobar si se cumplía cada una de las condiciones dándonos como resultado 0 ó *—1 como hemos visto*— y después realizábamos cualquier operación lógica con los resultados de ambas evaluaciones.

El Programa V nos dará una clara visión del camino seguido.

Vamos a seguirlo. En la línea 30 el **Amstrad** nos informa de su nombre, por si hubiera dudas. Mediante la línea 40 nos pregunta por el nuestro y el valor que le damos lo almacena en la variable **«nombre$»**.

NOT	OPERANDO	RESULTADO
—	0	1
—	1	0

TABLA III

A llegar a la línea 50 nos encontramos algo nuevo. **¿Cómo funciona?** Lo primero que hace es evaluar la condición que hay entre los paréntesis.

nombre$ = **«AMSTRAD»**

Si se cumple —el nombre que hemos metido por el teclado es el mismo— el programa meterá en la variable «condición» un —1. En caso que no se cumpla meterá un 0 como ya vimos.

En la siguiente línea —60— preguntamos si existe «condición» o su valor es 0.

IF condicion THEN... ELSE...

mira si el valor de la variable «condición» es 0 o es distinto de 0 *(—1 por ejemplo)*.

Si es distinto de cero *(o lo que es lo mismo, si la condición se cumple)* el programa continuará por las instrucciones que sigan a THEN.

Si es igual a cero *(en este caso la condición no se ha cumplido)* seguiremos por la rama del ELSE. **¿Verdad que no es tan complicado como parece?** Visto cómo se evalúan las condiciones vamos a intentar aplicar alguno de estos operadores lógicos.

Este pequeño programa nos va a preguntar por dos números y va a intentar imprimirnos el más pequeño de los dos. Observemos cómo lo hace.

Metemos los dos números en las variables **«número1»** y **«número2»** por medio de las instrucciones INPUT de las líneas 30 y 40. Hasta aquí todo en orden.

La línea 50 es la del lío. Lo primero que vamos a hacer es calcular el valor de los paréntesis.

El primero es:

número1 AND número1 < número2

Hagamos de ordenador. Si se cumple la condición que número1 < número2, la evaluación de esta condición es —1 *(o una sucesión de bits con valor 1)*.

Si hacemos un AND de **«número1»** con —1, el resultado de la operación es **«número1»**. Puede emplear el Programa 1 para verlo.

PROGRAMAS

```
10 REM PROGRAMA I
20 CLS
30 PRINT TAB(10);"OPERADOR LOGICO A
ND"
40 PRINT
50 INPUT "VALOR DEL PRIMER OPERANDO
: ",operando1
60 INPUT "VALOR DEL SEGUNDO OPERAND
O: ",operando2
70 PRINT
80 resultado=operando1 AND operando
2
90 PRINT TAB(15);BIN$(operando1,8)
100 PRINT TAB(10);"AND";TAB(15);BIN
$(operando2,8)
110 PRINT TAB(15);"--------"
120 PRINT TAB(15);BIN$(resultado,8)
;" =";resultado
```

```
10 REM PROGRAMA II
20 CLS
30 PRINT TAB(10);"OPERADOR LOGICO O
R"
40 PRINT
50 INPUT "VALOR DEL PRIMER OPERANDO
: ",operando1
60 INPUT "VALOR DEL SEGUNDO OPERAND
O: ",operando2
70 PRINT
80 resultado=operando1 OR operando2
90 PRINT TAB(15);BIN$(operando1,8)
100 PRINT TAB(10);"OR";TAB(15);BIN$
(operando2,8)
110 PRINT TAB(15);"--------"
120 PRINT TAB(15);BIN$(resultado,8)
;" =";resultado
```

```
10 REM PROGRAMA III
20 CLS
30 PRINT TAB(10);"OPERADOR LOGICO N
OT"
40 PRINT
50 INPUT "VALOR DEL OPERANDO: ",ope
rando
60 PRINT
70 resultado=NOT operando
80 PRINT TAB(5);"NOT";TAB(10);BIN$(
operando,16)
90 PRINT TAB(10);"----------------"
100 PRINT TAB(10);BIN$(resultado,16
);" =";resultado
```

```
10 REM PROGRAMA IV
20 CLS
30 PRINT TAB(10);"OPERADOR LOGICO X
OR"
40 PRINT
50 INPUT "VALOR DEL PRIMER OPERANDO
: ",operando1
60 INPUT "VALOR DEL SEGUNDO OPERAND
O: ",operando2
70 PRINT
80 resultado=operando1 XOR operando
2
90 PRINT TAB(15);BIN$(operando1,8)
100 PRINT TAB(10);"XOR";TAB(15);BIN
$(operando2,8)
110 PRINT TAB(15);"--------"
120 PRINT TAB(15);BIN$(resultado,8)
;" =";resultado
```

```
10 REM PROGRAMA V
20 CLS
30 PRINT"ME LLAMO AMSTRAD, Y TU";
40 INPUT nombre$
50 condicion=(nombre$="AMSTRAD")
60 IF condicion THEN PRINT"TENEMOS
EL MISMO NOMBRE" ELSE PRINT"NO SOMO
S TOCAYOS"
```

```
10 REM PROGRAMA VI
20 CLS
30 INPUT "PRIMER NUMERO: ",numero1
40 INPUT "SEGUNDO NUMERO: ",numero2
50 PRINT "EL NUMERO MENOR ES:";(num
ero1 AND numero1<numero2)+(numero2
AND numero1>=numero2)
```

Por tanto, si se cumple que número1 < número2, el valor del primer paréntesis es el contenido de **«número1»**. Pero no se cumplirá la condición del segundo paréntesis:

Si número1 < número2 es VERDADERO
número1 > = número2 es FALSO

y su evaluación será 0.

Al hacer en el segundo paréntesis AND de un número con **«cero»**, el resultado es evidentemente 0.

Si sumamos los valores obtenidos nos quedará algo así como:

número1 + 0

o lo que es igual:

número1

Luego si se cumple que

número1 < número2

el programa nos imprime el valor del número menor tal y como queríamos.

XOR	OPERANDO 1	OPERANDO 2	RESULTADO
—	0	0	0
—	0	1	1
—	1	0	1
—	1	1	0

TABLA IV

Le sugerimos como trabajo personal ver qué pasa si

número1 > número2

Bueno, lo dejamos por el momento. No olvide que cualquier operador lógico que actúa sobre números nos da como resultado un valor numérico. **¿De acuerdo?** Hasta pronto.

ESCRIBIENDO NUESTROS PROPIOS PROGRAMAS

Primeros repasos

Hasta ahora, nos hemos limitado a darle a la máquina una instrucción cada vez. Sin embargo, a poco compleja que sea la tarea que pretendemos realizar, será necesario dividirla en parte y darle al AMSTRAD muchas instrucciones. Por ejemplo, supongamos que queremos que aparezca en la pantalla el siguiente mensaje:

PROGRAMAR
ES
FACIL

Con nuestro método **«paso-a-paso»** tendríamos que teclear:

PRINT **«PROGRAMAR»** [ENTER]
PRINT **«ES»** [ENTER]
PRINT **«FACIL»** [ENTER]

Rápidamente se ve que de esta forma no funciona, porque cada instrucción sucesiva aparece en la pantalla «entre medias». Lo que necesitamos es darle al micro tres instrucciones tales como:

1. Escribe **«programar»**
2. Escribe **«es»**
3. Escribe **«fácil»**

en secuencia, de modo que el ordenador las ejecute sin detenerse a proguntar qué queremos que haga después.

Un programa es una secuencia de instrucciones

Tal secuencia de instrucciones es llamada PROGRAMA y, como podéis ver, va numerada en orden creciente, para que el micro pueda saber a donde debe dirigir su atención primero. Ahora puede comenzar nuestro programa, tecleando:

10 PRINT **«PROGRAMA»** [ENTER]

Hemos elegido para numerar la primera instrucción el número 10 en lugar del 1. En la práctica de la programación, se suelen numerar las líneas de 10 en 10 por razones que más tarde os parecerán obvias. Llamamos al número de una instrucción su NUMERO DE LINEA. EL ORDENADOR NO EJECUTA INMEDIATAMENTE LA INSTRUCCION. Ningún **«PROGRAMAR»** aparece en la pantalla. Esta reacción inesperada se debe al número de línea, el cual le informa a la máquina que lo que siga no debe ser obedecido ahora, sino atesorado ciudadosamente en la memoria junto con el resto de las instrucciones que vengan después, si existen, para ser ejecutadas conjuntamente en el momento oportuno.

Acabamos, pues, de descubrir que todo ordenador posee dos MODOS DE OPERACION, uno, llamado MODO INMEDIATO, en el cual la orden es obedecida en cuanto se pulsa [ENTER] y un segundo, conocido como MODO PROGRAMA, en el que las instrucciones se depositan en memoria, esperando pacientemente a cobrar vida. Este es el que nos interesa. Ahora, por favor, teclead:

20 PRINT **«ES»** [ENTER]
30 PRINT **«FACIL»** [ENTER]

y, en modo directo, CLS [ENTER], para borrar la pantalla.

Todo nuestro texto ha desaparecido, pero no hay que preocuparse demasiado. Existe un comando Basic que nos mostrará en pantalla nuestro programa completo, lo que se llama el LISTADO del mismo. Para verlo y creerlo:

LIST [ENTER]

y debería aparecer el programa número 1.

De sabios es rectificar

En previsión de algún posible **«despiste»** en éste o posteriores programas, vamos a ver cómo podríamos corregirlo. Trataremos de alterarlo para escribir:

PROGRAMAR
ES
SIMPLE

Pues bien, el método más simple de todos *(y tosco)* es ¡teclearla de nuevo! La nueva versión reemplazará a la antigua en la memoria del AMSTRAD, y eso es todo; para demostrarlo, teclése: 30 PRINT **«SIMPLE» [ENTER] a continuación LIST [ENTER] y, ¡hops!, tenemos el programa. Como prueba final y decisiva, RUN [ENTER] mostrará en la pantalla:**

PROGRAMAR
ES
SIMPLE

Hay formas más sofisticadas de corregir *(EDITAR)* una línea, pero las revisaremos posteriormente. De momento, éste será nuestro método «antierrores».

Naturalmente, si nos percatamos de un error de escritura MIENTRAS introducimos una línea de programa, usaríamos la tecla DEL para borrarlo, siguiendo a partir de ese punto hasta finalizar con ENTER la instrucción.

Supongamos que de nuevo queremos alterar el programa para que produzca como salida:

PROGRAMAR ES
BASTANTE SIMPLE

Obviamente, necesitamos una nueva línea de programa entre la 20 y la 30, que podemos probar con:

25 PRINT **«BASTANTE»** [ENTER]

Vamos a bregar ahora con algo completamente diferente *(programa número 2)*. El efecto es bastante impresionante, **¿verdad?**

Hasta este momento, nuestros programas se limitaban a copiar en la pantalla el contenido entrecomillado de las líneas; aquí, con la adición de una más, la cantidad de output *(salida en pantalla)* se incrementa considerablemente, y lo único que hemos hecho es ordenar al AMSTRAD que repita una y otra vez la misma secuencia de operaciones *instrucción 40;* **«vete a la línea 10»,** *(en inglés GOTO 10)*.

Si la salida del programa va demasiado rápida, podemos detenerla pulsando, la tecla ESC; el programa continuará ejecutándose a la pulsación de cualquier otra tecla excepto ESC, en cuyo caso se detendría, devolviéndonos el control.

PROGRAMAS

```
 5  REM * PROGRAMA 1 *
10 PRINT"PROGRAMAR"
20 PRINT"ES"
30 PRINT"SIMPLE"

 5  REM * PROGRAMA 2 *
10 PRINT "ME"
20 PRINT "DIVIERTE"
30 PRINT "PROGRAMAR"
40 GOTO 10
```

Los bucles: 1.ª clave de la programación

Lo que sucede en el interior del ordenador es lo siguiente: ME *(línea 10)*
DIVERTE *(línea 20)*
PROGRAMAR *(línea 30)*
y entonces llega a la línea 40 que le ordena volver a repetir el ciclo, observad cómo al **«acabarse»** la pantalla, el AMSTRAD, él solito, hace sitio para más texto por el expeditivo procedimiento de subir hacia arriba el existente.

AMPLIA LAS POSIBILIDADES DE TU AMSTRAD

IMPRESORA PRINTER 130

Especialmente recomendada para ordenadores AMSTRAD. **54.900 Pts**

JOYSTICKS

Los famosos SVI de la serie Quickshot. Desde **1.600 Pts**

LAPIZ OPTICO

Diseña gráficos y menús de comunicación en la pantalla a color. Incluye software. **4.500 Pts**

INTERFACE SERIE RS 232 C

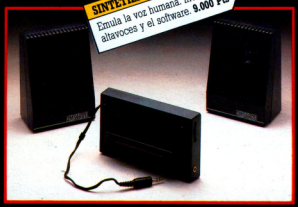
Para conectar con modems, impresoras serie y otros ordenadores **11.750 Pts.**

UNIDAD DE DISCO

Incluye Sistema Operativo CP/M y lenguaje LOGO.
(con controlador) **45.500 Pts**
(sin controlador) **39.500 Pts**

SINTETIZADOR DE VOZ

Emula la voz humana. Incluye dos altavoces y el software. **8.000 Pts**



MODULADOR TV COLOR

Para utilizar el TV como pantalla a todo color. **8.000 Pts** (CPC 464).
9.450 Pts (CPC 664 y 6128)

DAR FORMA AL SONIDO

En nuestro último encuentro estuvimos viendo cómo conseguir que el Amstrad produzca sencillos sonidos utilizando una versión muy básica del comando SOUND.

Si nos siguió le será muy fácil comprender por qué:

SOUND 1,200,100,5

produce una nota de volumen 5, por el canal A con un tono 200, durante un segundo.

Estamos seguros de que ha sido así, pero nuestra insistencia es debida a que usaremos esta nota en particular como base para nuestra exploración de la envolvente de volumen del **Amstrad**.

Y, se preguntará, **¿qué es una envolvente de volumen?**

Concepto de envolvente

Cuando introducimos en el ordenador:

SOUND 1,200,100,5

seremos recompensados con una nota bastante sonora. Observe que el sonido no varía mientras dura la nota.

El problema es que en la vida real las notas no permanecen con el mismo nivel de sonido. Su volumen sube o baja mientras las estamos oyendo. Nosotros no podemos conseguir este efecto con el simple comando SOUND que hemos usado hasta el momento.

La nota comienza con un nivel 5 y permanece en él hasta que deja de sonar un segundo después. No obstante, utilizando una envolvente de volumen definida previamente podemos hacer que la nota cambie de nivel mientras suena.

Así que vamos a definir la envolvente de volumen con:

ENV 1,5,2,20

No se preocupe si por el momento no comprende esta instrucción, acepte solamente que este comando define una envolvente de volumen a la que a partir de ahora nos referiremos como la número 1.

Escuchemos el efecto que produce en una nota teclenado:

SOUND 1,200,100,5,1

Hemos podido escuchar que la nota se hace más y más alta. Sigue sonando durante un segundo pero ahora el volumen varía.

El comando es el mismo que habíamos utilizado anteriormente excepto que hemos añadido un 1 al final para decirle al micro que utilice la envolvente definida previamente como la número 1.

Disponemos de hasta 15 envolventes

Podemos definir hasta 15 de estas envolventes, numeradas del 1 al 15. Una vez definidas, podemos llamarlas añadiendo el número apropiado al final de nuestro sencillo comando de sonido. Entonces el **Amstrad** reproduce la nota variando el nivel de acuerdo con la envolvente de volumen especificada.

Antes que entremos en los mecanismos de cómo se define una envolvente de volumen, queremos primero admitir que la semana pasada les dijimos algo que puede considerarse como una pequeña mentira.

Recordará que escribimos que el parámetro de volumen podía variar desde 0 a 7 y que tenía un valor por defecto de 4. Bueno, esto es cierto, pero sólo cuando no utilizamos la envolvente de volumen.

Si la utilizamos, el parámetro de volumen puede variar realmente desde 0 *(silencio)* a 15 *(nivel máximo)*. No hay diferencia en la intensidad absoluta de máximo volumen, un parámetro de volumen de valor 7 sin envolvente nos da el mismo nivel que uno de 15 con una envolvente.

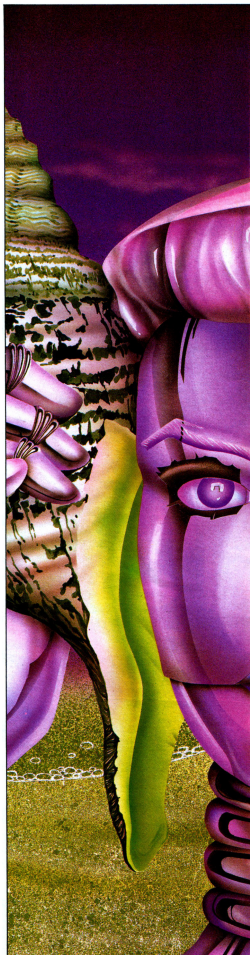
Justamente por ello, cuando especificamos una envolvente, el rango del parámetro volumen puede tener 16 valores en lugar de los 8 que utilizamos habitualmente.

Mostramos este nuevo rango de los valores de los parámetros en la figura 1.

De este modo podemos definir hasta 15 envolventes de volumen y, añadiendo el número apropiado al final del comando SOUND, podemos usarlas para variar la intensidad de una nota mientras está sonando. Nuestra pregunta es, **¿cómo definiremos una envolvente de volumen?**

Análisis del comando ENV

La respuesta es fácil: utilizando el comando ENV. Esta nueva instrucción tiene un as-



pecto feroz al poder estar seguida de hasta 16 parámetros.

Cuando conseguimos conocerlo, sin embargo, no es tan fiero como parecía. El secreto está en no dejarle ver que le tenemos miedo.

Oigamos de nuevo los efectos que produce la envolvente 1. A menos que la hayamos redefinido o tengamos desconectado el **Amstrad**, podemos encontrarla guardada en nuestro micro, pero para estar seguros tecleemos:

ENV 1,5,2,20

Observe que el comando ENV no produce ningún sonido por sí mismo. Podemos teclearlo hasta que se nos congestione la cara por el esfuerzo pero, a pesar de ello, el **Amstrad** permanecerá mudo. Todo lo que hace la envolvente de volumen es *(cuando se le pide)* influir en el comando SOUND. Este es el que hace ruido. Si no nos cree, teclee:

SOUND 1,200,100,5,1

y oirá una nota que ha sido afectada por la envolvente de volumen.

Si escucha con atención, podrá oir claramente cinco cambios de intensidad durante el segundo que está sonando la nota. El volumen de ésta se incrementa en intervalos de 2 unidades y cada intervalo dura 20 centésimas de segundo.

Es muy importante hacer que la subida o caída de volumen causada por la envolvente no sea progresiva. Se lleva a cabo en escalones. Formalmente diremos que el cambio es discreto, no continuo.

Vamos a examinar detalladamente la envolvente 1. Su forma más general es:

ENV N,P,Q,R

es decir, el comando envolvente seguido de cuatro parámetros.

Todo lo que hace N es **«etiquetar»** la envolvente definiéndola con un número comprendido entre 1 y 15. Hay una envolvente 0 pero es la definida por defecto, lo mismo que su duración fijada en dos segundos en un volumen que es el del parámetro normal de la instrucción SOUND.

El parámetro P especifica al micro en cuántos escalones queremos que evolucione la nota. Puede haber hasta 127 de estos intervalos. En su envolvente de volumen 1 hay 5 etapas.

El parámetro Q dice al **Amstrad** el incremento o decremento en intensidad que tiene que variar en cada escalón. El volumen inicial se toma del parámetro correspondiente en el comando SOUND. En el caso de la envolvente 1 el volumen aumenta 2 unidades por cada escalón de la envolvente. El valor del parámetro Q puede variar entre —128 y 127.

Finalmente el parámetro R decide la duración de cada uno de los escalones en los que está dividida la envolvente. Se mide en centésimas de segundo y puede tener un rango de valores comprendido entre 0 y 255.

La tabla II es una recapitulación de todos estos parámetros y sus rangos correspondientes.

Sonido

La figura 1 representa los efectos de:

SOUND 1,200,100,5

vistos de un modo gráfico.

Como podemos ver, el nivel de volumen permanece igual durante todo el segundo que dura la nota.

Ahora probemos con nuestra envolvente favorita:

ENV 1,5,2,20

(si es que hemos apagado el **Amstrad** o redefinido la envolvente) y

SOUND 1,200,100,5,1

La figura II nos presenta lo que ocurre. La envolvente provoca que el volumen de la nota se incremente durante el segundo que suena.

Hay cinco escalones y cada uno dura 20 centésimas de segundo. Cada etapa provoca un incremento de dos unidades en el volumen, el resultado al final de todos los escalones es que el volumen aumenta de 7 a 15 en el segundo que suena la nota.

A continuación vamos a realizar un cambio y definiremos otra envolvente de volumen con:

ENV 2,5,1,20

Podemos **«oírlo»** en acción introduciendo:

SOUND 1,200,100,5,2

Llamamos a la envolvente de volumen 2 con el parámetro del final del comando SOUND. Esta vez la intensidad de la nota aumenta también, pero como el parámetro Q es 1, no consigue el mismo nivel que antes.

A propósito, la envolvente de volumen número 1 sigue guardada en la memoria del **Amstrad**. Podemos comprobarlo por nosotros mismos tecleando:

SOUND 1,200,100,5,1

y obtendremos el mismo resultado que antes.

Con la envolvente podemos variar el volumen

La envolvente de volumen no sirve siempre para hacer que el volumen aumente. También podemos hacerle disminuir. Si no se lo cree defina una nueva envolvente con:

ENV 3,5,—1,20

y llámela con:

SOUND 1,200,100,5,3

Ahora el volumen se decrementa uno en cada escalón conforme avanza el tiempo.

Mientras decrementamos el volumen, pruebe:

ENV 4,5,—2,20

y

SOUND 1,200,100,5,4

Parece una campana, **¿verdad?** Observe que el parámetro de volumen del comando SOUND *—la cuarta cifra—* es 14. Quiere de-

cir que la envolvente de volumen comienza por debajo del volumen máximo y va disminuyendo de 2 en 2 en cada etapa.

Intente definir algunas envolventes propias y vea los efectos que puede crear. Es muy divertido y, mientras se entretiene, puede encontrarle algún que otro efecto extraño.

En los ejemplos anteriores hemos usado siempre el comando SOUND con una duración de 1 segundo. Del mismo modo, siempre le hemos dividido en 5 etapas, cada una de las cuales tiene una duración de 20 centésimas de segundo.

En otras palabras, los 5 escalones de la envolvente de volumen suman un segundo, que es la duración del comando SOUND. Podríamos preguntarnos qué ocurre si la duración de SOUND fuese más larga o más corta que el tiempo que hace falta para todos los escalones de la envolvente de volumen. Veamos.

Definamos una nueva versión de la envolvente 1 con:

ENV 1,5,2,10

De nuevo hay 5 etapas, cada una de ellas con una duración de 10 centésimas de segundo de modo que la envolvente dura 0.5 segundos. El tiempo que necesita una envolvente de volumen se obtiene de multiplicar P×R, el número de escalones por el tiempo que dura cada uno de ellos.

A continuación veamos qué ocurre cuando tecleamos:

SOUND 1,200,100,5,1

que le dice al **Amstrad** que toque durante un segundo.

Como podemos escuchar la nota alcanza rápidamente el volumen máximo y se mantiene en este nivel hasta que finaliza el comando SOUND. Dicho de otro modo, la envolvente sólo afecta a la nota durante el tiempo que abarca la misma (P×R). La nota continúa durante todo el tiempo especificado en SOUND, sonando con el volumen alcanzado cuando terminó la envolvente.

Veamos qué ocurre en el caso contrario, volviendo a definir la envolvente 2 con:

ENV 2,5,1,20

que tiene 5 etapas con una duración total de un segundo.

Ahora produciremos un sonido que dure medio segundo y escucharemos lo que ocurre. Teclee:

SOUND 1,200,100,5,2

y oiremos cómo corta a las primeras de cambio a nuestra pobre envolvente. Si quiere obtener el efecto completo utilice:

SOUND 1,200,100,5,2

Aparte de las discrepancias entre la duración de la envolvente de volumen y la del comando SOUND que le llama, puede presentarse otro problema. **¿Qué ocurre si hay demasiados escalones en la envolvente?**

Probándolo se verá. Mientras

ENV 1,10,2,20

espera 10 escalones con una duración de un quinto de segundo cada uno de ellos y nece-

Tabla I: Rango de parámetros del comando SOUND.

	Canal	Tono	Duración	Volumen		Envolvente de volumen
				Sin envolvente	Con envolvente	
Rango	1=A 2=B 4=C	0 A 4095	1 A 32767	0 A 7	0 A 15	0 A 15
Defecto				4	12	0

Tabla II: Rangos de parámetro para el comando ENV.

Parámetro	Número N	Número de escalones en sección P	Cambio de volumen por escalón Q	Longitud de cada escalón R
Rango	0 A 15	0 A 127	−128 A 127	0 A 255

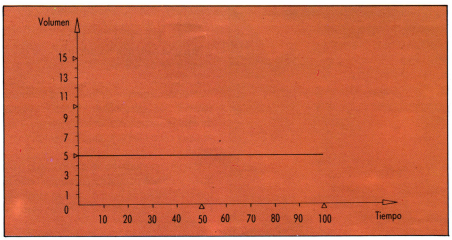


Figura I: SOUND 1, 200, 100, 5

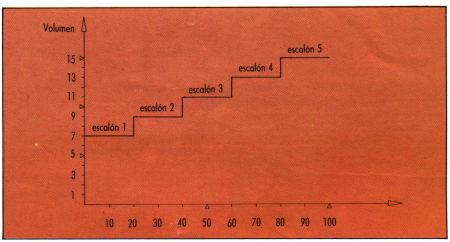


Figura II: SOUND 1, 200, 100, 5, 1.

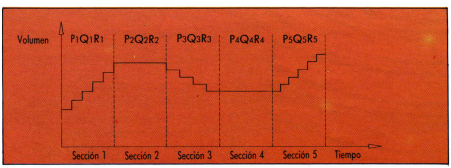


Tabla III: Parámetros para las cinco secciones de envolvente de sonido.

sita dos segundos para completar su trabajo, por desgracia

```
SOUND 1,200,100,5,1
```

sólo dura un segundo, así que la envolvente de volumen solamente tendrá cinco etapas.

El caso contrario, en el que sólo vamos a tener 2 etapas en la envolvente de volumen, lo ilustramos con los siguientes comandos:

```
ENV 1,2,2,20
```

y

```
SOUND 1,200,100,5,1
```

Como podremos escuchar, obtendremos dos escalones ocupando 40 centésimas de segundo y la nota mantiene el volumen final durante las 60 centésimas restantes.

Echemos de nuevo una ojeada a la tabla 1. Observe que cuando usamos una envolvente, el parámetro de volumen puede tener un rango de valores comprendidos entre 0 y 15. Hasta aquí, todos los ejemplos han sido elegidos para que los parámetros permanezcan dentro de este rango.

¿Qué pasa si la envolvente de volumen ha intentado sacarlos fuera de rango?

Definamos una envolvente con:

```
ENV 5,5,3,20
```

Aumenta el volumen en escalones de 3 unidades cada vez. Vamos a considerar el efecto que produce esta envolvente sobre un comando SOUND semejante a:

```
SOUND 1,200,100,5,5
```

El volumen inicial es 5. La primera etapa de la envolvente lo convertirá en 8, la segunda en 11 y la tercera en 14. Pero, ¿qué ocurre cuando la cuarta intenta darle un valor de 17 que está fuera de rango? Inténtelo y verá qué pasa.

El volumen es cíclico

Como podemos escuchar el **Amstrad** es una máquina astuta y, en vista que le hemos invitado a hacer una travesura, llega solamente a 15. Oimos cómo el volumen vuelve a empezar de nuevo en 0 como si se tratara de un **«cuentakilómetros»**, pasando de golpe de una intensidad muy alta al silencio otra vez.

También puede ocurrir de otra forma. La envolvente definida por:

```
ENV 6,5,–3,20
```

disminuye en tres la intensidad de volumen de cada escalón. Cuando somos malvados y le hacemos trabajar con el comando:

```
SOUND 1,200,100,5,6
```

el volumen pasa de 5 a 2 y entonces, antes de que se salga del rango, **«da la vuelta del cuentakilómetros»**, se coloca en 14 y sigue decreciendo otra vez. Pruébelo y lo verá.

Por ahora debemos sentirnos bastante cómodos con la existencia de la envolvente de volumen. Pero también debemos ser conscientes de alguna de sus limitaciones. Una de ellas es que hasta aquí sólo podemos usar la envolvente para que el volumen de una nota aumente o disminuya.

PROGRAMAS

```
10 REM PROGRAMA I
20 REM ENVOLVENTE DE VOLUMEN
30 DIM p(5),q(5),r(5)
40 WHILE -1
50 MODE 1
60 INPUT "Numero de secciones en la
 envolvente de volumen? ",secciones
70 IF secciones<1 OR secciones>5 TH
EN CLS:GOTO 60
80 CLS
90 FOR bucle=1 TO secciones
100 LOCATE 3,5:PRINT"Seccion"bucle
110 LOCATE 3,8:PRINT"Numero de esca
lones?"
120 LOCATE 30,8:INPUT "",p(bucle)
130 IF p(bucle)<0 OR p(bucle)>127 T
HEN LOCATE 30,8:PRINT SPACE$(8):GOT
O 120
140 LOCATE 3,13:PRINT"Altura de cad
a escalon?"
150 LOCATE 30,13:INPUT "",q(bucle)
160 IF q(bucle)<-128 OR q(bucle)>12
7 THEN LOCATE 30,13:PRINT SPACE$(8)
:GOTO 150
170 LOCATE 3,18:PRINT"Tiempo de pau
sa?"
180 LOCATE 30,18:INPUT "",r(bucle)
190 IF r(bucle)<0 OR r(bucle)>255 T
HEN LOCATE 30,18:PRINT SPACE$(8):GO
TO 180
200 LOCATE 14,23:PRINT"PULSA ESPACI
O"
210 WHILE INKEY(47)=-1:WEND:CLS
220 WHILE INKEY$<>"":WEND
230 NEXT bucle
240 ENV 1,p(1),q(1),r(1),p(2),q(2),
r(2),p(3),q(3),r(3),p(4),q(4),r(4),
p(5),q(5),r(5)
250 duracion=p(1)*r(1)+p(2)*r(2)+p(
3)*r(3)+q(4)*r(4)+p(5)*r(5)
260 SOUND 1,200,duracion,5,1
270 CLS
280 duracion$=RIGHT$(STR$(duracion)
,LEN(STR$(duracion))-1)
290 PRINT"SOUND 1,200,";duracion$;"
,5,1
300 FOR bucle=1 TO secciones
310 bucle$=RIGHT$(STR$(bucle),1)
320 PRINT"P(";bucle$;")  "p(bucle)
330 PRINT"Q(";bucle$;")  "q(bucle)
340 PRINT"R(";bucle$;")  "r(bucle)
350 NEXT
360 LOCATE 14,23:PRINT"PULSA ESPACI
O"
370 WHILE INKEY(47)=-1:WEND:CLS
380 WEND
```

En la vida real, sin embargo, las notas a veces hacen las dos cosas, creciendo gradualmente su intensidad para luego decaer y desvanecer en la lejanía. Con su actual forma:

```
ENV N,P,Q,R
```

nuestra envolvente de volumen no puede hacerlo. Y una vez más debemos confesarles que solamente hemos dicho la verdad a medias.

La verdadera definición de la envolvente de volumen no es:

```
ENV N,P,Q,R
```

sino que es en realidad:

```
ENV N,P1,Q1,R1,P2,Q2,R2
P3,Q3,R3,P4,Q4,R4,
P5,Q5,R5
```

En lugar de nuestro simpatiquillo ENV con cuatro parámetros, nos hemos encontrado con un enorme monstruo con 16 números detrás. No obstante no nos demos por vencidos ni sintamos miedo, con un poco de paciencia veremos que es bastante más sencillo de lo que parece.

En lo que hemos hecho hasta ahora sólo hemos utilizado una parte de los parámetros de la envolvente de volumen. Cada envolvente puede tener hasta cinco secciones que afectan a un mismo comando SOUND de manera distinta.

Por el momento solamente hemos utilizado la primera sección como referencia hacia aquéllos que poseen una disposición nerviosa.

Cada una de las cinco secciones de la envolvente de volumen funciona exactamente de la misma manera que la primera, sobre la que nos hemos estado concentrando. Todo lo que tenemos que hacer cuando queramos comprender una envolvente es tratar cada una de estas cinco secciones como hemos dicho anteriormente.

La única diferencia es que en lugar de P, Q, y R, la primera sección tiene los parámetros P1, Q1 y R1, la segunda P2, Q2, y R2 y así sucesivamente. La figura III nos muestra cómo se relacionan los parámetros con las secciones.

Aunque podemos tener cinco secciones en la envolvente de sonido, como es evidente por todo lo dicho anteriormente, no es obligatorio utilizarlas todas.

Vamos a definir una envolvente que subirá el volumen, luego lo bajará y de nuevo lo volverá a incrementar.

Lo podemos hacer con:

```
ENV 1,5,2,20,5,–2,20,5,2,20
```

Aquí la primera sección de la envolvente tiene cinco escalones y el volumen aumenta en cada uno de ellos dos unidades. La segunda sección hace lo contrario ya que en cada uno de sus cinco escalones el volumen disminuye en 2 unidades. Dejamos que imagine lo que ocurre en la tercera sección:

```
SOUND 1,200,300,5,1
```

nos permitirá escuchar esta envolvente a la que hemos dividido en tres partes. Observe que hemos alargado la duración del comando SOUND a tres segundos. Nos aseguramos así que todas la secciones tengan una oportunidad de sonar.

Y esto es todo lo que hay sobre la envolvente de volumen. Aunque parezca difícil no lo es con tal que vayamos analizando sección por sección. Todo lo que necesitamos en un poco de práctica que nos familiarice con ella y es aquí cuando aparece el Programa I.

Ejecútelo y se encontrará con que le permitirá crear sus propias envolventes de sonido y escuchar los resultados. Después de una hora más o menos descubrirá que lo comprende completamente.

Y hasta puede que esté esperando con ilusión el artículo de la semana que viene sobre envolventes de tono.

Análisis

BUSQUEDA CON INSTR

Autor: **Fco. Javier Barceló T.**

AMSTRAD Análisis estudia esta semana un comando muy útil a la hora de buscar determinados datos dentro de un archivo en disco o cinta. Este comando es INSTR. Veámoslo en el programa. Para mayor simplicidad, el programa incluye los datos en sentencias DATA, simulando los registros de un archivo.

10-40 Comentarios.

50-80 Lee las líneas de DATA y las asigna a las cadenas A$(1) a A$(5).

90 Borra la pantalla, y establece el modo de pantalla en 80 caracteres.

100 Pregunta el dato a buscar, lo almacena en la variable B$, calcula la longitud de B$ y da su valor a BL.

120 Pregunta la posición desde la que tiene que buscar B$ en A$ y la almacena en B.

130 Inicia el bucle de búsqueda del dato.

140 Selecciona el valor de X.

150 A partir de la posición de búsqueda dada (B), busca en la cadena origen [A$ (X)] el dato dado anteriormente *(B$)* Esta función produce un valor numérico *(E)*. Si no encuentra el dato dicho valor será 0, y si lo encuentra el valor será igual a la posición dentro de la cadena origen [A$ (X)] que ocupe en 1.º carácter del dato que se busca (B$).

160 Si no ha encontrado el dato de la cadena (E=0), va a la línea 240 para continuar el bucle principal.

170 Comentario.

180 Si ha encontrado el dato (E < > 0), indica la cadena en que lo ha encontrado, y la posición dentro de dicha cadena donde empieza el dato buscado *(E)*.

190 Imprime una línea en blanco.

200 Imprime la cadena donde ha encontrado el dato [A$ (X)].

210 Imprime espacios hasta la posición donde empieza el dato encontrado en la cadena anterior, y luego imprime flechas señalando el dato encontrado.

220 Imprime una línea en blanco.

230 Da a I$ el valor **«encontrado»**. Ver línea 260.

240 Fin del bucle principal.

250 Comentarios.

260 Si I$ no tiene el valor «encontrado» imprime que no ha encontrado el dato. En el momento que encuentre el dato, pasará por la línea 230, y por tanto no imprimirá nada.

270 Pregunta si se quiere hacer otra búsqueda. Almacena la respuesta en R$.

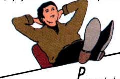
280 Si respondemos S, anula el valor de I$ y vuelve al principio para empezar otra vez.

290 Si la respuesta es N, acaba el programa. Si se ha dado por error un valor distinto e S y de N, vuelve a la línea 270 para volver a preguntar.

300 Comentarios.

310, 320, 330, 340 y 350 Datos de las cadenas A$ donde se realiza la búsqueda.

Si se le da *—por ejemplo—* como posición de búsqueda 4, y encuentra lo que se busca 2 posiciones después, el valor que devuelve no es 2 sino 6, es decir, la posición absoluta de dicho dato dentro de la cadena. Si no lo encuentra, devuelve el valor 0. Las únicas limitaciones de esta instrucción son que no busca espacios en blanco, *(siempre que se intenta, da el valor 0)*, y que si damos un dato en mayúsculas y ese dato está en minúsculas, no lo reconoce.



Para que tus dedos no realicen el trabajo duro, M.H. AMSTRAD lo hace por ti. Todos los listados que incluyan este logotipo se encuentran a tu disposición en un cassette mensual, solicítanoslo.

...descubre el N.º 3

ya está en tu quiosco

AMSTRAD

también disponible para

SPECTRUM, PLUS, 128

COMMODORE 64

CARNIVAL

Si te gusta el tiro al blanco, con este programa podrás practicar sin necesidad de salir de casa. Si tienes buena puntería obtendrás disparos gratis.

BLOCKER

Demuestra tu habilidad esquivando las paredes y a tus enemigos. Cuantos más destruyas, más aparecerán ante ti.

SPACE

Al cargar este programa aparecerá ante ti un batallón de alienígenas. Tu misión es destruirlos, pero cuidado, su intención es eliminarte lo antes posible.

HAUNTED

En este caso debes coger todos los puntos que aparecen en el laberinto. ¡Atención!, los fantasmas están enfadados e intentarán deshacerse de ti a toda costa.

VAMPIRO

Es un programa en el cual pueden participar dos jugadores. La misión de cada uno será pintar las lápidas de un color distinto. La destrucción del enemigo significa la victoria.

SPLIT

Es una rutina en código máquina, que te permitirá siete colores en pantalla en MODO 1, en el cual normalmente sólo se pueden utilizar cuatro.

795 pts.
(Incluido IVA)

YOUR COMPUTER

Si no lo encontrara en su quiosco, solicítelo directamente a nuestra editorial.

SINTAX, S.A.

Paseo de la Castellana, 268.
28046 Madrid. Tel. (91) 733 25 99

La mejor selección de programas de juegos y utilidades, publicados en la revista de mayor difusión de ordenadores de Europa. Ahora reproducidos en cassette, en auténtica exclusiva mundial.

SIR FRED

Sir Fred, el diestro espadachín, entra en la dura contienda del mundo del sotfware, bajo el emblema de Made in Spain. Los oponentes son fuertes y poderosos, sus armas: el dinero, la fama y una gran experiencia, el campo de batalla; el Reino Unido, una empresa de titanes.

Una gran conmoción se ha apoderado de la villa de Castlecity: todos están pendientes de la arrogante figura del caballero, que, desafiando a la guardia del castillo, pretende raptar a la princesa Margaret.

Sir Fred no es un hombre precisamente guapo; rasgos morunos, piernas cortas y resistentes, prominente barriga que le cuelga por encima del cinturón y un considerable apéndice nasal, que, de no llevar la cabeza muy alta, puede hacer que nuestro héroe pierda el equilibrio.

Pero **¿qué tiene que ver el valeroso caballero, con España?**

Pocas personas conocen su turbulento pasado.

Sir Fred, nació en una pequeña aldea de Burgos, donde su padre ostentaba el título de condestable mayor del reino.

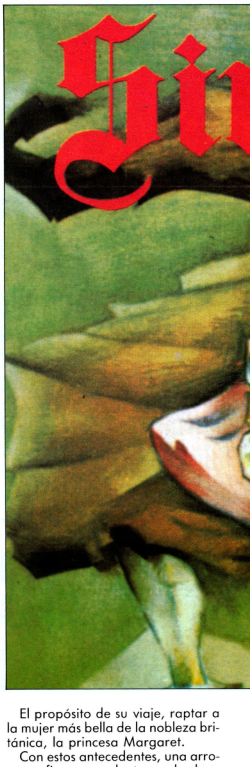
De la familia de los Pérez de Sotillejo, como hijo primogénito, su padre eligió un nombre digno de la grandeza y arraigo de la casa, quedando para la historia como Don Leoncio Pérez de Sotillejo.

Con ese nombre, cualquier mortal estaría eternamente agradecido a su progenitor y Leoncio, de hecho lo estuvo, hasta que llegó a Madrid, donde los villanos de la capital se mofaron tanto, que nuestro caballero decidio emprender una nueva vida en un país diferente, para dejar de ser el hazmerreír del reino.

De esta forma, Leoncio reniega de su nombre y se embarca hacia las costas inglesas, donde aparece como Sir Fred Pérez de Sotillejo y Villareal, que viste mucho más.

Lo único que no ha traicionado, es el escudo heráldico de su familia; un león rampante con una espada entre los dientes, sobre campo de gules y plata, enseña que Sir Fred ostenta con orgullo y defiende allá donde va.

Compatible: CPC/464, CPC/664 y CPC/6128.



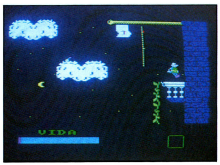
El propósito de su viaje, raptar a la mujer más bella de la nobleza británica, la princesa Margaret.

Con estos antecedentes, una arrogante figura se adentra en los bosques que rodean al misterioso castillo. En la maleza, criaturas diabólicas nos acechan, Sir Fred salta, trepa, corre y nada, combate y se oculta, acercándose a los infranqueables muros.

Experto espadachín, estoqueará a guardianes y vigilantes, en su paso por los corredores del castillo, hasta conseguir apoderarse de Margaret y regresar con ella a su tierra natal, para dejar constancia de su gran hazaña.

Fred

Mr. Joystick



Nos encontramos ante un programa que se basa en la archiconocida idea del caballero que rescata a una princesa; menos mal que esta aventura se puede ambientar en lugares y decorados diferentes, porque si no, habría como veinte o treinta programas iguales.

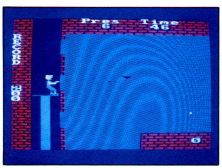
Nuestro héroe, puede: cojer objetos, batirse a espada, correr, saltar, andar, trepar, nadar, bucear y morir, formando una amplia gama de movimientos y posibilidades, las cuales debemos utilizar en las distintas pantallas del juego.

Todo tratado con unos gráficos aceptables, inundados de rojo, en los cuales lo más destacable es la animación del personaje central, que aunque de reducida dimensiones, ejecuta una gran variedad de movimientos.

Es importante reseñar la diversidad de pantallas que incluye el juego, cada una con sus elementos característicos, agua, murallas, escaleras, lianas, obstáculos y terraplenes, que nos obligan a exprimir al máximo las distintas modalidades de movimiento que permite don Leoncio Pérez de Sotillejo, más conocido en el mundo del software como el simpar Sir Fred.

CAZA DE ALTURA

Caza de altura es un juego de mucha acción, que contiene una idea clásica tratada con gran originalidad y acierto en el manejo de los gráficos.



Programa realizado por
René Rodríguez Lleonart

El programa sólo funciona con joystick; el autor no ha previsto el empleo del teclado para manejar el programa, pensando sin duda en la comodidad que ofrece el joystick y en que todo el mundo tiene uno (!).

Aunque esto último tal vez no sea muy realista, el lector interesado no encontrará dificultad en adaptarlo, haciendo uso de la tabla de rutinas principales que René ha incluido en su comentario del programa.

Descripción:

El juego consiste en cazar el máximo número de patos que podamos en un tiempo determinado o hasta que se nos acaben las balas. Al empezar disponemos de 6 disparos que se incrementan cada vez que conseguimos destruir alguno de los huevos que los patos sueltan al pasar ante el edificio que nos encontramos.

Nuestro movimiento en el edificio lo controlamos mediante un ascensor que nos permite subir y bajar a velocidad normal o bajar hasta la base con caída libre *(sin peligro para nuestra integridad física)*. Si nos quedamos sin munición termina el juego.

VARIABLES PRINCIPALES

Pres	Número de disparos disponibles.
Tem	Tiempo que queda para finalizar.
Dalt	Coordenada **«y»** de la pistola en todo momento.
Ocell	Toma valores 0 ó 1 para determinar si hay o no un pato en pantalla.
Ou	Igual que la anterior pero para los huevos.
Ocx,ocy	Coordenadas del pato
Ox,oy	Coordenadas del huevo
Pun	Puntos acumulados.
Rec	Récord de puntuación.
Vx	Coordenada **«x»** de la bala.
a$,b$,c$,d$,t,g,	Variables auxiliares.

ESTRUCTURA DEL PROGRAMA

1-1000	Inicializaciones y dibujo de la pantalla.
1000-1100	Bucle principal del programa.
1110-1150	Mensajes finales del programa. Nueva partida.
1199-1230	Control del movimiento del ascensor.
1399-1470	Movimiento de la bala y de detección de choques.
1599-1620	Mueve el pato si no ha sido cazado.
1799-1810	Mueve el huevo hasta que es destruido.
1999-2010	Escribe la puntuación en pantalla.
2199-2210	Lanza un huevo aleatoriamente.
2399-2420	Crea un pato a una latura aleatoria.
2799-2820	Determina si lo que se ha cazado es un pato o un huevo y actualiza las variables correspondientes.
2999-3010	Retardo de tiempo.
3199-3230	Mueve el pato cuando ha sido cazado.
399-4005	Actualiza el tiempo y lo escribe en pantalla.
4099-4110	Escribe el tiempo que queda.
4199-4210	Escribe los puntos que llevamos.
4299-4310	Escribe el récord de puntuación.
4399-4410	Borra los puntos de la partida anterior.

Serie Oro

```
1 REM ============================
======
2 REM ===== RENE RODRIGUEZ LLEONART
====
3 REM =====            PRESENTA
====
4 REM =====
====
5 REM =====             EDIF1
====
6 REM =====
====
7 REM =====     SEPTEMBRE 1.985
====
8 REM =====
====
9 REM ============================
#===
10 MODE 0
11 INK 1,15:INK 2,26:INK 6,13:INK 7
,2:INK 8,11:INK 9,7:INK 10,16:INK 1
1,24:INK 12,11,16:INK 13,18:INK 14,
23:INK 15,8
12 PAPER 7:PEN 3:CLS:PAPER #2,5:PAP
ER #3,5
13 RANDOMIZE TIME:pres=6:tem=51:EVE
RY 100,3 GOSUB 4000
14 WINDOW #1,1,5,1,25:PAPER #1,5:CL
S#1:WINDOW #1,3,5,1,25:WINDOW#4,7,2
0,19,22:PAPER#4,7
15 SYMBOL AFTER 143
16 SYMBOL 144,0,251,251,251,0,223,2
23,223
17 SYMBOL 145,0,240,240,240,0,208,2
08,208
18 SYMBOL 146,240,240,240,240,240,2
40,240,240
19 PRINT CHR$(22)"1"
20 FOR t=1 TO 2:FOR g=1 TO 25:LOCAT
E t,g:PRINT CHR$(144):NEXT:NEXT
30 FOR t=1 TO 22 STEP 7:FOR g=0 TO
3:LOCATE 6,t+g:PEN 5:PRINT CHR$(146
)CHR$(8)::PEN 3:PRINT CHR$(145):NEX
T:NEXT
31 LOCATE 1,1
40 MOVE 68,0:DRAW 154,0,7:MOVE 68,2
:DRAW 154,2,7
60 GOSUB 4400
62 WINDOW #3,6,20,1,3:CLS#3:PEN #3,
3:FOR t=1 TO 45:PRINT#3,CHR$(144);:
NEXT:LOCATE#3,1,1:PRINT#3,CHR$(22)"
1";:PEN #3,2:LOCATE #3,3,1:PRINT#3,
"Pres":LOCATE#3,9,1:PRINT#3,"Time":
PRINT#3,CHR$(22)"0"
63 GOSUB 2000
65 PEN 2:PRINT CHR$(22)"1":LOCATE 1
,4:PRINT"R":PRINT"E":PRINT"C":PRINT
"O":PRINT"R":PRINT"D"
66 GOSUB 4300
100 SYMBOL 147,0,3,2,3,3,7,7,2
110 SYMBOL 148,15,12,12,12,12,8,0,0
120 SYMBOL 149,0,0,1,0,0,0,0,0
130 SYMBOL 150,0,128,128,192,128,0,
128,0
140 SYMBOL 151,192,64,0,0,0,0,0,0
150 SYMBOL 152,2,7,6,4,6,7,7,0
160 SYMBOL 153,0,0,1,3,1,0,0,0
170 SYMBOL 154,0,0,0,0,0,0,0,7
180 SYMBOL 155,0,0,128,112,0,0,0,0
190 SYMBOL 156,0,0,0,128,240,240,0,
0
200 SYMBOL 157,0,0,0,8,8,0,0,0
210 SYMBOL 158,0,15,14,4,0,8,0,0
220 SYMBOL 159,7,7,7,3,1,0,0,0
230 SYMBOL 160,0,0,0,4,2,1,0,0
240 SYMBOL 161,128,128,192,224,240,
240,56,56
250 SYMBOL 162,0,0,0,0,0,0,192,192
260 SYMBOL 163,24,24,24,0,0,0,0,0
270 SYMBOL 164,192,192,192,0,0,0,0,
0
280 SYMBOL 165,0,0,0,216,0,0,0,0
290 SYMBOL 166,0,0,0,0,254,182,0,0
300 SYMBOL 167,0,28,81,126,30,30,0,
0
310 SYMBOL 168,0,0,0,128,0,0,4,2
320 SYMBOL 169,0,65,126,30,30,0,0,0
330 SYMBOL 170,0,0,128,0,0,4,2,0
340 SYMBOL 171,0,3,3,0,0,0,0,0
350 SYMBOL 172,0,0,14,22,38,102,102
,1
360 SYMBOL 173,0,0,0,0,0,0,8,40
400 PRINT#1,CHR$(22)"1":LOCATE #1,2
,22
410 PEN#1,10:PRINT#1,CHR$(147)CHR$(
150)CHR$(8)CHR$(8)::PEN#1,11:PRINT#
1,CHR$(148)CHR$(151)CHR$(8)CHR$(8);
:PEN#1,12:PRINT#1,CHR$(149)
420 LOCATE #1,2,23
430 PEN#1,13:PRINT#1,CHR$(152)CHR$(
155)CHR$(8)CHR$(8)::PEN#1,14:PRINT#
1,CHR$(153)CHR$(156)CHR$(8)CHR$(8);
:PEN#1,9:PRINT#1,CHR$(154)::PEN#1,1
```

```
0:PRINT#1,CHR$(157)CHR$(8);:PEN#1,1
5:PRINT#1,CHR$(158)
440 LOCATE#1,2,24
445 PEN #1,8:PRINT#1,CHR$(159)CHR$(
161)CHR$(8)CHR$(8);:PEN #1,7:PRINT#
1,CHR$(160)CHR$(162)
450 LOCATE#1,3,25
460 PEN#1,8:PRINT#1,CHR$(163)CHR$(8
);:PEN#1,7:PRINT#1,CHR$(164)CHR$(8)
;:PEN#1,10:PRINT#1,CHR$(165)CHR$(8)
;:PEN#1,6:PRINT#1,CHR$(166)" ";:PRI
NT#1,CHR$(143)
490 LOCATE #1,1,1:PRINT#1,CHR$(22)"
0"
999 dalt=22
1000 REM============================
====
1010 REM ============================
1020 REM ============================
====
1030 WHILE pres>0 AND Tem>0
1050                GOSUB 1200:REM mou
re ascensor
1060                IF JOY(0)>15 THEN
 IF (dalt-1) MOD 7<4 THEN SOUND 1,12
00,5:pres=pres-1:GOSUB 2000 ELSE GO
SUB 1400 ELSE GOSUB 3000
1070                IF ocell>0 THEN GO
SUB 1600 ELSE GOSUB 2400
1075                IF ou=1 THEN GOSUB
 1800 ELSE IF ocx<19 THEN GOSUB 220
0
1100 WEND:v=REMAIN (1)+REMAIN (2)+R
EMAIN (3)
1110 FOR t=pres TO 1 STEP-1:pun=pun
+15:pres=pres-1:GOSUB 2000:GOSUB 42
00:SOUND 4,300,5:FOR g=1 TO 300:NEX
T:NEXT
1120 PEN#4,2:LOCATE#2,1,5:IF rec<=p
un THEN IF rec=pun THEN PRINT#4," L
O IGUALASTE" ELSE rec=pun:PRINT#4,"
ERES EL MEJOR":ELSE PRINT#4," No es
 Record"
1125 WINDOW#5,1,1,13,24:PAPER #5,5:
PEN#5,3:FOR t=13 TO 24:LOCATE#5,1,t
:PRINT#5,CHR$(144);:NEXT
1130 PRINT#4," Joystick  ";CHR$(243
):PRINT#4," para seguir"
1140 IF JOY(0)=8 THEN pun=0:pres=6:
tem=51:CLS#4:GOSUB 4000:GOSUB 4400:
GOSUB 4200:GOSUB 4300:GOSUB 2000:EV
ERY 100,3 GOSUB 4000:GOTO 1030 ELSE
 1140
1150 END
1199 REM=======moure ascensor======
====
1200 IF JOY(0)=4 THEN IF ou=1 THEN
EVERY 10,2 GOSUB 1800 ELSE ELSE GOT
O 1210
1205 EVERY 10,1 GOSUB 1600:WHILE da
lt<23:DI:LOCATE #1,1,1:PRINT#1,"
":d
alt=dalt+1:EI:WEND:v=REMAIN (1)+REM
AIN (2):SOUND 2,1150,20,7:GOTO 1230
1210 IF (JOY(0)=1 OR JOY(0)=17) AND
 dalt>3 THEN LOCATE#1,2,25:PEN#1,6:
PRINT#1,"
"CHR$(143):dalt=dalt-1:SOUND 2,1140
,21,2 ELSE IF(JOY(0)=2 OR JOY(0)=18
)AND dalt<23 THEN LOCATE #1,1,1:PRI
NT#1,"
":dalt=dalt+1:SOUND 2,1150,21
,2 ELSE GOSUB 3000
1230 RETURN
1399 REM=======disparar============
====
1400 SOUND 4,794,5,7:vx=6
1410 GOSUB 1600:EVERY 10,2 GOSUB 16
00:IF ou=1 THEN GOSUB 1800:EVERY 10
,1 GOSUB 1800
1420 WHILE NOT(((vx=ocx OR vx=ocx+1
) AND dalt=ocy)OR(vx=ox AND dalt=oy
))AND vx<20
1430 DI:PEN 0:vx=vx+1:LOCATE vx-1,d
alt:PRINT" "CHR$(171):EI
1440 WEND
1445 v=REMAIN (1)+REMAIN (2)
1450 IF vx<20 THEN GOSUB 2800 ELSE
LOCATE vx,dalt:PRINT" ":pres=pres-1
:GOSUB 2000
1470 RETURN
1599 REM ==========moure ocell======
====
1600 IF ocell=2 THEN 3200
1605 PEN 5:n=(n+2)MOD 4
1610 IF ocx>7 THEN LOCATE ocx,ocy:P
RINT" ";:ocx=ocx-1:LOCATE ocx,ocy:P
RINT CHR$(22)"1"CHR$(167+n);:PEN 1:
PRINT CHR$(8)CHR$(168+n)CHR$(22)"0"
 ELSE LOCATE ocx,ocy:PRINT" ";:ocel
l=0
1620 RETURN
1799 REM ==========moure ou========
====
1800 IF oy<22 OR (oy<24 AND ox<14)
THEN PEN 4:LOCATE ox,oy:PRINT" "CHR
$(10)CHR$(8)CHR$(171):oy=oy+1 ELSE
IF ou=1 THEN ou=0:LOCATE ox,oy:PRIN
T" "
1810 RETURN
1999 REM=========puntuacio=========
====
```

Serie Oro

```
2000 DI:b$=STR$(pres):LOCATE #3,6-L
EN(b$),2:PEN#3,3:PRINT#3,CHR$(144);
:PEN#3,2:PRINT#3,MID$(b$,2):EI
2010 RETURN
2199 REM=========llencar ou========
====
2200 IF RND(1)<0.2 THEN SOUND 2,850
,5:oy=ocy+1:ox=ocx:PEN 4:LOCATE ox,
oy:PRINT CHR$(171);:ou=1
2210 RETURN
2399 REM==========crear ocell======
====
2400 ocy=INT((RND(1)*10)+4):ocx=20
2410 LOCATE ocx,ocy:PEN 5:PRINT CHR
$(22)"1"CHR$(167);:PEN 1:PRINT CHR$
(8)CHR$(168)CHR$(22)"0":ocell=1
2420 RETURN
2799 REM==========explosio========
====
2800 SOUND 1,1300,5:IF DALT=ocy THE
N ocell=2:pun=pun+50:pres=pres-1 EL
SE LOCATE vx,oy:PRINT" ":ou=0:pres=
pres+1:oy=25
2810 GOSUB 2000:GOSUB 4200
2820 RETURN
2999 REM==========Retard==========
====
3000 FOR t=1 TO 30:NEXT:RETURN
3199 REM========mort================
====
3200 IF ocy<22 OR (ocy<24 AND ocx<1
4) THEN PEN 5:LOCATE ocx,ocy:PRINT"
 "CHR$(10)CHR$(8)CHR$(172);:PEN 1:P
RINT CHR$(22)"1"CHR$(8)CHR$(173)CHR
$(22)"0":ocy=ocy+1 ELSE LOCATE ocx,
ocy:PRINT" ":GOSUB 2400
3205 ou=1
3230 RETURN
3999 REM=========temps=============
====
4000 tem=tem-1:GOSUB 4100
4005 RETURN
4099 REM========print temps========
====
4100 c$=STR$(tem):LOCATE#3,13-LEN(c
$),2:PRINT#3,MID$(c$,2);:IF tem<11
THEN SOUND 4,200,5 :IF tem=9 THEN P
EN#3,3:PRINT#3,CHR$(8)CHR$(8)CHR$(1
44):PEN#3,2
4110 RETURN
4199 REM========print punts========
====
4200 a$=STR$(pun):LOCATE#2,8-LEN(a$
),2:PRINT#2,MID$(a$,2);
4210 RETURN
4299 REM========print record=======
====
4300 PRINT CHR$(22)"1":PEN 2:d$=STR
$(rec):LOCATE 1,13:FOR t=1 TO LEN(d
$):PRINT MID$(d$,t,1):NEXT:PRINT CH
R$(22)"0"
4310 RETURN
4399 REM========borra punts========
====
4400 WINDOW #2,14,20,23,25:CLS#2:PE
N #2,3:FOR t=1 TO 21:PRINT#2,CHR$(1
44);:NEXT:PEN#2,2
4410 RETURN
```

Gestión del disco en lenguaje máquina

Todos sabemos salvar y gravar programas en cinta o disco con los conocidos comandos SAVE, LOAD, y también conocemos cómo abrir ficheros para introducir u obtener datos desde el disco o el cassette.
Ahora bien, seguro que muchos de nosotros nos habremos preguntado alguna vez, cómo hacer todo lo dicho anteriormente desde código máquina.

Cuando hayáis terminado de leer este artículo estamos seguros de que estas preguntas quedarán resueltas, y seréis capaces de salvar y cargar programas ya sea en Basic o Binarios, sin ningún problema, desde código máquina.

Estudiaremos paso a paso cada uno de los comandos de manejo de disco y cassette desde Basic y confeccionaremos dichos comandos en lenguaje máquina, aprovechándonos por supuesto de las grandes posibilidades que nos ofrece el firmware de nuestro ordenador.

Cómo se hace un SAVE en máquina

Veremos en primer lugar la forma de salvar un programa en Basic. Para salvar un programa de este tipo ejecutaríamos la instrucción:

SAVE"PROGRAMA

Vamos a ver pues cuál sería la rutina en código máquina capaz de producir el mismo efecto.

El programa número uno nos muestra cómo hacerlo, así pues vamos a intentar explicar cada paso y cuáles son las llamadas que se hacen al firmware.

#BC8C Abre canal y buffer para efectuar SAVE. Antes de llamar a esta rutina debemos tener en cuenta los parámetros que necesita para producir los efectos deseados. Estos son los siguientes:

B Longitud del nombre del programa.

HL Dirección donde se encuentra el nombre.

DE Dirección del buffer de 2 K.

#BC98 Pasa el contenido del buffer de 2 K directamente al cassette o disco. Los parámetros necesarios son los siguientes:

HL Contiene la dirección donde se encuentra almacenado el programa.

DE Longitud del programa.

BC Dirección de entrada si queremos que se ejecute automáticamente con RUN"PROGRAMA.

A Contiene el tipo de programa.

0 Programa Basic

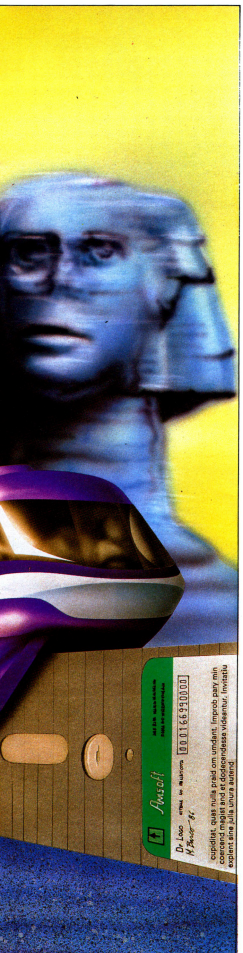
1 Programa Basic protegido

2 Programa en código máquina

#BC8F Cierra el canal de SAVE y pasa el último buffer al cassette o disco. No necesita condiciones de entrada.

Si nos fijamos en la rutina que efectúa la funciones de SAVE, podemos ver que en primer lugar introducimos en el registro B el valor 6, que es el número de caracteres que tiene el nombre del programa 'Programa', a continuación apuntamos el registro HL a la dirección NAME, que es donde se encuentra situado el nombre del programa, y luego cargamos en DE la dirección del Buffer de SAVE, que hemos colocado en la dirección #A900, y con estos datos llamamos a la rutina #BC8C.

A continuación debemos introducir la dirección de inicio del programa en el registro HL, como hemos dicho que se trataba de un programa Basic, la dirección de inicio de éste es siempre la dirección #170, por lo tanto cargaremos dicho varlo en HL. Seguidamente debemos calcular la longitud de dicho programa. Para obtener este dato, en primer lugar cargamos DE con el valor que se encuentra en la dirección #AE83, la cual nos indica dónde termina el programa Basic, por lo tanto si a esta dirección de final de programa le restamos la dirección de inicio que era #170, ob-



tendremos la longitud total del programa basic.

Este proceso se efectúa de la siguiente forma:

```
LD HL,#170
LD DE,(#AE83)
EX DE,HL
SBC HL,DE
EX DE,HL
```

A continuación deberemos cargar en BC el punto de entrada del programa, pero dado que es un programa Basic, no será necesario introducir ningún valor por lo que hemos cargado BC con cero.

Por último deberemos introducir en el registro A el tipo de programa. Si queremos salvarlo como Basic protegido, dicho valor será uno, de lo contrario deberá ser cero.

Ahora llamamos a la rutina del firmware que pasa el contenido del buffer a cassette o disco, y finalmente cerramos el canal de SAVE llamando a #BC8F.

Comando SAVE para ficheros binarios

Para efectuar un SAVE de un programa en código máquina deberemos seguir el mismo camino mencionado anteriormente, teniendo en cuenta las siguientes modificaciones.

En primer lugar, la dirección de inicio de programa no será ya la #170, sino la dirección de donde se encuentra ubicado nuestro programa, asimismo la longitud del programa será la correspondiente a nuestro programa en memoria que podemos calcular de la forma siguiente:

(Dirección final - Dirección inicial) + 1

y por último el valor con que deberemos cargar el acumulador será 2 ya que éste es el correspondiente a un programa tipo binario.

El programa número dos muestra un ejemplo de SAVE para un programa tipo binario.

Si queremos que un programa salvado en código máquina se ejecute automáticamente cuando se carga con la instrucción RUN'', deberemos introducir en el registro BC, la dirección de ejecución de dicho programa.

El comando Basic que efectúa este segundo programa es el siguiente:

SAVE''Programa'',B,&6000,&500,&6000

El equivalente del comando LOAD

Vamos a ver a continuación cómo podemos simular el comando LOAD desde un programa en código máquina. Veremos en primer lugar cuáles son las direcciones del firmware que utilizaremos para dicha rutina.

#BC77 Abre canal para LOAD y lee el primer bloque. Las condiciones de entrada deben ser:

ProgramAcción

B Debe contener la longitud del nombre del programa.

HL Contiene la dirección donde está situada la cadena de caracteres que componen el nombre.

DE Contiene la dirección del buffer de 2 k para efectuar la lectura.

#BC83 Coloca el programa directamente en la zona de memoria que va a ocupar dicho programa. Los parámetros de entrada son los siguientes:

HL Debe contener la dirección de memoria donde va a ir el programa. Dicha dirección debe estar en la RAM.

El primer programa simula el comando Basic:

LOAD''PROGRAMA

con el que cargaremos un programa ya sea en Basic o código máquina, desde cinta o disco.

Si el programa que deseamos cargar es un programa confeccionado en Basic, la dirección que debe contener el registro doble HL, será la #170, puesto que en dicha dirección es donde empiezan todos los programas Basic.

Para cargar un programa en código máquina deberemos cargar en HL, la dirección inicial donde debe ir dicho programa.

Para simular la instrucción Basic:

LOAD'' ''

deberemos indicar a la correspondiente rutina del firmware, que la longitud del nombre debe ser cero, para ello, antes de llamar a la dirección #BC77, deberemos cargar en el registro B el valor cero que indicará que la longitud de la cadena que forma el nombre, es nula. Asimismo no hará falta cargar en HL la dirección donde se encuentra dicha cadena de caracteres.

Un ejemplo de actuación de este comando lo podemos ver en el segundo programa que trata de la sentencia LOAD.

Otra forma de ejecutar la carga de un programa desde Basic, es desabilitando los mensajes que se ofrecen durante la carga del mismo.

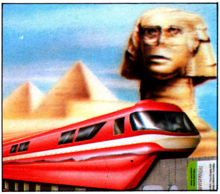
Para ello deberemos llamar a la rutina situada en el firmware que se encargará de ello. Dicha rutina es la que se explica a continuación:

#BC6B desabilita mensajes en pantalla. No aparecen tampoco los mensajes de error. Los parámetros en entrada son los siguientes:

A 0 para habilitar mensajes

A cualquier valor distinto de 0 para desabilitarlos.

Por lo tanto el programa que simula esta sentencia Basic, cargará el programa 'progra-



ma' sin darnos ningún tipo de mensajes durante su carga. Esto se consigue cargando en el acumulador el valor 255 y llamando seguidamente a la rutina del firmware #BC6B.

De la misma forma colocando las siguientes líneas:

```
LD A,255
CALL #BC6B
```

al principio del programa SAVE''PROGRAMA'', conseguiremos desabilitar mensajes durante la grabación de programas.

El último programa nos muestra cómo efectuar la sentencia Basic siguiente:

LOAD''!''

esto lo conseguimos desabilitando los mensajes y además cargando en el registro B el valor 0 que indicará que la cadena de caracteres que forman el nombre del programa está vacía, y por lo tanto cargará el primer programa que encuentre en la cinta. Esto último no es posible hacerlo para efectuar la carga desde disco, ya que como sabéis para efectuar un LOAD desde disco necesitamos indicar el nombre del programa.

ENSAMBLADOR / CARGADOR BASIC

PROGRAMA 1

```
                    10 :SAVE"PROGRAMA"
                    20 :
A000                30          ORG  #A000
A000  0608          40          LD   B,8
A002  2122A0        50          LD   HL,NAME
A005  1100A9        60          LD   DE,#A900
A008  CD8CBC        70          CALL #BC8C
A00B  217001        80          LD   HL,#170
A00E  ED5B83AE      90          LD   DE,(#AE83)
A012  EB           100          EX   DE,HL
A013  ED52         110          SBC  HL,DE
A015  EB           120          EX   DE,HL
A016  010000       130          LD   BC,0
A019  3E00         140          LD   A,0
A01B  CD98BC       150          CALL #BC98
A01E  CD8FBC       160          CALL #BC8F
A021  C9           170          RET
A022  50524F47     180 NAME:    DEFM "PROGRAMA"
```

ETIQUETAS

```
NAME   A022
```

PROGRAMA 2

```
                    10 :SAVE"PROGRAMA",B,&6000,&500,&6000
                    20 :
A000                30          ORG  #A000
A000  3A2BA0        40          LD   A,(LNAME)
A003  47            50          LD   B,A
A004  2123A0        60          LD   HL,NAME
A007  ED5B2CA0      70          LD   DE,(BUFFER)
A00B  CD8CBC        80          CALL #BC8C
A00E  2A2EA0        90          LD   HL,(DIRINI)
A011  ED5B30A0     100          LD   DE,(LONG)
A015  ED4B32A0     110          LD   BC,(ENTRA)
A019  3A34A0       120          LD   A,(TIPO)
A01C  CD98BC       130          CALL #BC98
A01F  CD8FBC       140          CALL #BC8F
A022  C9           150          RET
A023  50524F47     160 NAME:    DEFM "PROGRAMA"
A02B  08           170 LNAME:   DEFB 8
A02C  00A9         180 BUFFER:  DEFW #A900
A02E  0060         190 DIRINI:  DEFW #6000
A030  0005         200 LONG:    DEFW #500
A032  0060         210 ENTRA:   DEFW #6000
A034  02           220 TIPO:    DEFB 2
```

ETIQUETAS

```
BUFFER A02C  DIRINI A02E  ENTRA  A032
LNAME  A02B  LONG   A030  NAME   A023
TIPO   A034
```

PROGRAMA 3

```
                    10 :LOAD"PROGRAMA"
                    20 :
A000                30          ORG  #A000
A000  3A18A0        40          LD   A,(LNAME)
A003  47            50          LD   B,A
A004  2119A0        60          LD   HL,NAME
A007  ED5B23A0      70          LD   DE,(BUFF)
A00B  CD77BC        80          CALL #BC77
A00E  2A21A0        90          LD   HL,(DIREC)
A011  CD83BC       100          CALL #BC83
A014  CD7ABC       110          CALL #BC7A
A017  C9           120          RET
A018  08           130 LNAME:   DEFB 8
A019  50524F47     140 NAME:    DEFM "PROGRAMA"
A021  0080         160 DIREC:   DEFW #8000
A023  00A9         170 BUFF:    DEFW #A900
```

ETIQUETAS

```
BUFF   A023  DIREC  A021  LNAME  A018
NAME   A019
```

PROGRAMA 4

```
                    10 :LOAD""
                    20 :
A000                30          ORG  #A000
A000  0600          40          LD   B,0
A002  ED5B13A0      50          LD   DE,(BUFF)
A006  CD77BC        60          CALL #BC77
A009  2A15A0        70          LD   HL,(DIREC)
A00C  CD83BC        80          CALL #BC83
A00F  CD7ABC        90          CALL #BC7A
A012  C9           100          RET
A013  00A9         110 BUFF:    DEFW #A900
A015  0080         120 DIREC:   DEFW #8000
```

ETIQUETAS

```
BUFF   A013  DIREC  A015
```

PROGRAMA 5

```
                    10 :LOAD"!PROGRAMA
                    20 :
A000                30          ORG  #A000
A000  3EFF          40          LD   A,255
A002  CD6BBC        50          CALL #BC6B
A005  0600          60          LD   B,0
A007  ED5B18A0      70          LD   DE,(BUFF)
A00B  CD77BC        80          CALL #BC77
A00E  2A1AA0        90          LD   HL,(DIREC)
A011  CD83BC       100          CALL #BC83
A014  CD7ABC       110          CALL #BC7A
A017  C9           120          RET
A018  00A9         130 BUFF:    DEFW #A900
A01A  0080         140 DIREC:   DEFW #8000
```

ETIQUETAS

```
BUFF   A018  DIREC  A01A
```

PROGRAMAS

```
10 REM * PROGRAMA 1*
20 FOR N=&A000 TO &A02A
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&117A THEN PRINT "ERROR
 EN DATAS"
70 DATA 6,8,33,34,160,17,0
80 DATA 169,205,140,188,33,112,1
90 DATA 237,91,131,174,235,237,82
100 DATA 235,1,0,0,62,0,205
110 DATA 152,188,205,143,188,201,80
120 DATA 82,79,71,82,65,77,65
```

```
10 REM * PROGRAMA 2 *
20 FOR N=&A000 TO &A035
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&1554 THEN PRINT "ERROR
 EN DATAS"
70 DATA 58,43,160,71,33,35,160
80 DATA 237,91,44,160,205,140,188
90 DATA 42,46,160,237,91,48,160
100 DATA 237,75,50,160,58,52,160
110 DATA 205,152,188,205,143,188,20
1
120 DATA 80,82,79,71,82,65,77
130 DATA 65,8,0,169,0,96,0
140 DATA 5,0,96,2,0,0,0
```

```
10 REM * PROGRAMA 3 *
20 FOR N=&A000 TO &A025
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&F6B THEN PRINT "ERROR
EN DATAS"
70 DATA 58,24,160,71,33,25,160
80 DATA 237,91,35,160,205,119,188
90 DATA 42,33,160,205,131,188,205
100 DATA 122,188,201,8,80,82,79
110 DATA 71,82,65,77,65,0,128
120 DATA 0,169,0,0,0,0,0
```

```
10 REM *PROGRAMA 4 *
20 FOR N=&A000 TO &A017
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&AE1 THEN PRINT "ERROR
EN DATAS"
70 DATA 6,0,237,91,19,160,205
80 DATA 119,188,42,21,160,205,131
90 DATA 188,205,122,188,201,0,169
100 DATA 0,128,0,0,0,0,0
```

```
10 REM * PROGRAMA 5 *
20 FOR N=&A000 TO &A01C
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&E1C THEN PRINT "ERROR
EN DATAS"
70 DATA 62,255,205,107,188,6,0
80 DATA 237,91,24,160,205,119,188
90 DATA 42,26,160,205,131,188,205
100 DATA 122,188,201,0,169,0,128
```

DISPONIBLE PARA ZX SPECTRUM
AMSTRAD

SOFTWARE

Sound on Sound es una marca registrada
producida y distribuida por Iberofón, s. a.
Telf. 671.22.50 / 04 / 08 / 12 / 16

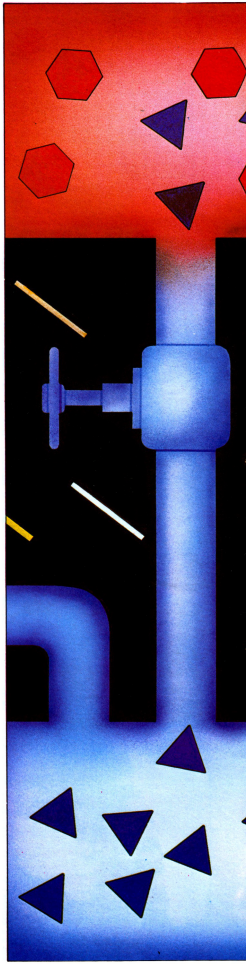
Sound-on-Sound

JUEGA CON EL FUTURO

¡¡¡NO LO SUEÑES!!! ¡JUEGALO!
SIENTE LA EMOCION DE LO DESCONOCIDO
CORRE TU PROPIO RIESGO
SALVA A TU COMPAÑERO/A ATRAPADO/A
REUNE LOS FRAGMENTOS DEL CUADRO
SON TU AMULETO

¡¡¡POR FIN EN CASTELLANO!!!
LA PRIMERA COMEDIA MUSICAL EN VIDEO-JUEGO

INSTRUCCIONES DE CAMBIO TRANSFERENCIA Y BUSQUEDA



F. L. Fronten

Hoy empezaremos la descripción de otro gran bloque de instrucciones, que son las llamadas de cambio, transferencia y búsqueda. Dada su extensión, las dividiremos en tres apartados diferentes. En el primero trataremos las instrucciones de cambio, en otro hablaremos de las de transferencia y por último el que tratará las de búsqueda.

Empezaremos analizando las instrucciones de cambio, que como su nombre indica, realizan un intercambio de contenidos entre diferentes registros.

La primera de las instrucciones con la que nos encontramos se representa de la siguiente forma:

EX DE,HL

Tras su ejecución los dos bytes de los registros dobles DE y HL se cambian entre sí.

Así pues si cargamos DE y HL con las posiciones indicadas a continuación:

LD HL,#7000
LD DE,#5000

después de la ejecución de EX DE, HL, el contenido de HL será #5000, y el del registro DE será #7000.

En el primer programa que hemos preparado podemos ver una aplicación práctica. Cargamos el registro doble DE con la dirección #7000 y HL con la dirección #7001, luego colocamos en la primera posición el código de la letra 'A', y en la segunda el código de la letra 'a'. Luego pasamos a un bucle que se repetirá 26 veces, en el cual se llama en primer lugar a la rutina de imprimir, la cual nos pintará en pantalla una letra mayúscula, luego hacemos EX DE,HL y volvemos a llamar a la rutina de impresión, y ahora ésta nos pintará una letra minúscula, y por último volvemos a intercambiar los contenidos de los registros DE y HL para que cuando se vuelva a repetir el bucle nos imprima otra mayúscula.

De este modo obtendremos en pantalla un abecedario doble constituido uno por letras mayúsculas y el otro por letras minúsculas.

Tenemos dos intrucciones más utilizando los registros dobles IX e IY:

EX DE,IX
EX DE,IY

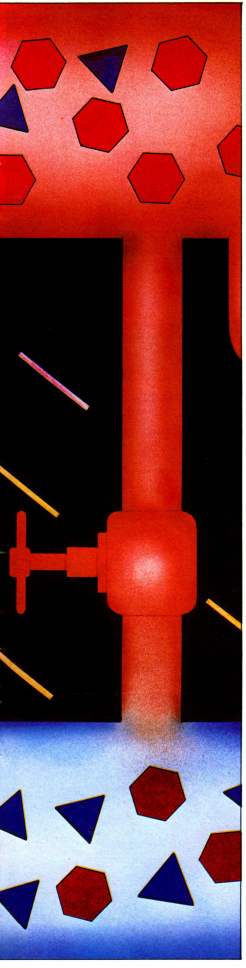
que actúan exactamente de la misma forma que la descrita anteriormente, pero como es lógico, utilizando los registros correspondientes.

Antes de pasar a describir la próxima instrucción de cambio, hablaremos de los registros alternativos que posee el microprocesador Z80.

Hasta ahora hemos estado trabajando con los registros, AF, HL, DE, BC, IX, IY, pero el Z80 además tiene la capacidad de trabajar con los registros alternativos, que para diferenciarlos de los ordinarios se le coloca un apóstrofe. Así pues estos registros alternativos serán:

AF' HL' DE' BC'

Ahora bien, el microprocesador sólo puede trabajar con los registros normales, así pues éstos se podrán utilizar como almacén, para en un momento determinado poder recu-



perar unos valores guardados anteriormente.

Ahora podemos ya pasar a describir la siguiente instrucción de cambio que se representan así:

EX AFF,AF'

Después de la ejecución de esta instrucción los dos bytes del registro AF pasan al registro AF' y los dos bytes de este último pasan al primero.

Así pues si el contenido de AF fuera #200 y el contenido de AF' fuera #7000, una vez ejecutado EX AF,AF' obtendríamos el valor #7000 en el registro doble AF y AF' contendría #200.

Dicha instrucción podríamos utilizarla, si por ejemplo tenemos un valor en el acumulador que queremos recuperar más tarde, entonces lo que haremos es ejecutar EX AF,AF' y cuando queramos recuperar ese valor deberemos ejecutar de nuevo la misma instrucción.

En el programa números dos podemos ver un ejemplo de utilización, hemos cargado el acumulador con el valor 49, y luego lo hemos pasado al registro alternativo AF'.

Deberemos tener mucho cuidado en el uso de los registros alternativos, ya que si llamamos a rutinas del firmware, estos registros se corromperán, ya que el sistema operativo del ordenador los utiliza para sus propias rutinas.

Por este motivo deberemos recuperar los valores almacenados en los registros alternativos, antes de llamar a cualquier rutina que utilice llamadas al firmware.

La instrucción que estudiaremos a continuación es: EXX

Su ejecución produce un intercambio entre los registros normales y los alternativos. Así pues los dos bytes de los pares BC, DE y HL son intercambiados respectivamente por los bytes de los pares BC', DE' y HL'.

Deberemos tener cuidado al ejecutarla, ya que como hemos dicho anteriormente, si llamamos a cualquiera de las rutinas del firmware, se corromperán los valores que habíamos guardado en los registros alternativos.

Podemos ver un ejemplo de actuación de esta instrucción en el programa número tres. En primer lugar damos unos valores a los registros dobles BC, DE y HL y luego los pasamos a los registros alternativos, para recuperarlos más tarde cuando sean necesarios.

Estas instrucciones podríamos utilizarlas, si en un momento determinado deseamos saltar a una rutina determinada, pero preservando todos los registros. Entonces dicha rutina debería tener la siguiente estructura:

```
ENTRADA A LA RUTINA
EX AF,AF'
EXX
.
.
RESTO DE LA RUTINA
.
.
EXX
EX AF,AF'
SALIDA DE LA RUTINA
```

Debo volver a insistir que dentro de esta rutina no puede haber ninguna llamada al firmware, ya que si la hubiera se corromperían los registros alternativos y perderíamos la información almacenada en ellos.

Previamente a la explicación de la próxima instrucción de cambio repasaremos brevemente el funcionamiento del STAK que utiliza el microprocesador para su funcionamiento interno.

Sabemos que éste es utilizado para guardar direcciones de memoria hacia las cuales debe saltar el control de programa cuando éste se encuentra con una instrucción RET.

La forma en que se almacenan esas direcciones de memoria, es semejante a la forma en que nosotros podríamos almacenar una serie de bandejas en una caja. Pondríamos en primer lugar la bandeja número 1 seguida de la número 2 y así sucesivamente hasta llegar a la bandeja número 10 por ejemplo. Una vez almacenadas éstas, cuando queremos recuperarlas, la primera que cogeremos será la bandeja que hemos almacenado en último lugar, o sea el número 10, seguida de la número 9 y así sucesivamente hasta llegar a la número 1 que es la primera que hemos almacenado.

El STACK o PILA, funciona exactamente de la misma forma, por lo tanto cuando se quiere recuperar un dato, se recupera en primer lugar el último que se ha almacenado.

Veamos ahora cómo se representa la última de las instrucciones de cambio:

EX (SP),HL

su ejecución produce un intercambio de los dos bytes superiores del STACK por los dos bytes del registro doble HL.

Por las explicaciones dadas anteriormente y observando la actuación

de la anterior instrucción, podemos hacernos una idea de las consecuencias catastróficas que podría acarrerar el mal uso de dicha instrucción.

Esto es así debido a que con EX (SP), HL se alteran los dos bytes superiores del stack que el microprocesador toma como retorno de una rutina. Si estos bytes se modifican injustificadamente, el retorno se produce en una dirección en la cual el micro no encuentra nada y por lo tanto se quedará colgado sin remisión.

Esta última instrucción es aplicable también con los registros dobles IX e IY, de la forma que indicamos a continuación:

EX (SP),IX

EX (SP),IY

éstas actúan exactamente igual que la anteriormente comentada pero lógicamente los dos bytes superiores del STACK, se intercambiarán respectivamente por los dos bytes de los registros IX e IY.

Debido al manejo de la PILA por el sistema operativo del ordenador, las anteriores instrucciones únicamente las podremos utilizar en programas construidos en su totalidad en código máquina, ya que si intentáramos volver al Basic, el sistema se bloquearía.

En el programa número cuatro se muestra un ejemplo de actuación de EX (SP),HL. En primer lugar cargamos en HL con la dirección de la rutina NAME y luego mediante la anterior instrucción, pasamos ese valor al STACK, con lo que el micro cuando encuentre un RET, saltará inmediatamente a la rutina NAME, y luego volverá al programa principal que nos imprimirá una B en pantalla, y se quedará en ese lugar colgado en un bucle infinito, ya que no podemos retornar al Basic.

El último programa es similar al anterior, únicamente que en éste repetimos el proceso tres veces, una con cada una de las instrucciones siguientes:

EX (SP),HL
EX (SP),IX
EX (SP),IY

cada uno de los procesos nos imprimirá en pantalla un número, con lo que al final del programa tendremos en pantalla los tres número de cada uno de los procesos y un último número que es imprimido al volver al programa principal.

GRAFICO 1

```
                        10 ;PROGRAMA-1
                        20 ;
A000                    30          ORG  #A000
A000  110878            40          LD   DE,#7800
A003  210170            50          LD   HL,#7001
A006  3E61              60          LD   A,"a"
A008  12                70          LD   (DE),A
A009  3E41              80          LD   A,65
A00B  77                90          LD   (HL),A
A00C  061A              100         LD   B,26
A00E  CD19A0            110 BUC:    CALL PINTA
A011  EB                120         EX   DE,HL
A012  CD19A0            130         CALL PINTA
A015  EB                140         EX   DE,HL
A016  10F6              150         DJNZ BUC
A018  C9                160         RET
A019  7E                170 PINTA:  LD   A,(HL)
A01A  CD5ABB            180         CALL #BB5A
A01D  34                190         INC  (HL)
A01E  C9                200         RET
```

ETIQUETAS

```
BUC     A00E  PINTA  A019
```

```
                        10 ;PROGRAMA-2
                        20 ;
A000                    30          ORG  #A000
A000  3E31              40          LD   A,49
A002  08                50          EX   AF,AF'
A003  C9                60          RET
```

```
                        10 ;PROGRAMA-3
                        20 ;
A000                    30          ORG  #A000
A000  018040            40          LD   BC,#4080
A003  210020            50          LD   HL,#2000
A006  110070            60          LD   DE,#7000
A009  D9                70          EXX
A00A  C9                80          RET
```

```
                        10 ;PROGRAMA-4
                        20 ;
A000                    30          ORG  #A000
A000  CD0AA0            40          CALL RUTI
A003  3E42              50          LD   A,"B"
A005  CD5ABB            60          CALL #BB5A
A008  18FE              70 INFI:    JR   INFI
A00A  210FA0            80 RUTI:    LD   HL,NAME
A00D  E3                90          EX   (SP),HL
A00E  C9                100         RET
A00F  E3                110 NAME:   EX   (SP),HL
A010  C9                120         RET
```

ETIQUETAS

```
INFI    A008  NAME   A00F  RUTI   A00A
```

```
                        10 ;PROGRAMA-5
                        20 ;
A000                    30          ORG  #A000
A000  CD0AA0            40          CALL RUT
A003  3E34              50          LD   A,"4"
A005  CD5ABB            60          CALL #BB5A
A008  18FE              70 INFI:    JR   INFI
A00A  210FA0            80 RUT:     LD   HL,NAME
A00D  E3                90          EX   (SP),HL
A00E  C9                100         RET
A00F  3E31              110 NAME:   LD   A,"1"
A011  CD5ABB            120         CALL #BB5A
A014  DD211BA0          130         LD   IX,NAME1
A018  DDE3              140         EX   (SP),IX
A01A  C9                150         RET
A01B  3E32              160 NAME1:  LD   A,"2"
A01D  CD5ABB            170         CALL #BB5A
A020  FD2127A0          180         LD   IY,NAME2
A024  FDE3              190         EX   (SP),IY
```

```
A026  C9                200         RET
A027  3E33              210 NAME2:  LD   A,"3"
A029  CD5ABB            220         CALL #BB5A
A02C  E3                230         EX   (SP),HL
A02D  C9                240         RET
```

ETIQUETAS

```
INFI    A008  NAME   A00F  NAME1  A01B
NAME2   A027  RUT    A00A
```

PROGRAMAS

```
10 REM * PROGRAMA 1 *
20 FOR N=&A000 TO &A01F
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&DA3 THEN PRINT "ERROR
EN DATAS"
70 DATA 17,8,112,33,1,112,62
80 DATA 97,18,62,65,119,6,26
90 DATA 205,25,168,235,205,25,160
100 DATA 235,16,246,201,126,205,90
110 DATA 187,52,201,187,253,33,39
```

```
10 REM * PROGRAMA 2 *
20 FOR N=&A000 TO &A005
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&140 THEN PRINT "ERROR
EN DATAS"
70 DATA 62,49,8,201,0,0,0
```

```
10 REM * PROGRAMA 3 *
20 FOR N=&A000 TO &A00B
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&2A5 THEN PRINT "ERROR
EN DATAS"
70 DATA 1,0,64,33,0,32,17
80 DATA 0,112,217,201,0,0,0
```

```
10 REM * PROGRAMA 4 *
20 FOR N=&A000 TO &A011
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&917 THEN PRINT "ERROR
EN DATAS"
70 DATA 205,10,160,62,66,205,90
80 DATA 187,24,254,33,15,160,227
90 DATA 201,227,201,0,0,0,0
```

```
10 REM * PROGRAMA 5 *
20 FOR N=&A000 TO &A02E
30 READ A:SUMA=SUMA+A
40 POKE N,A
50 NEXT
60 IF SUMA<>&1BCF THEN PRINT "ERROR
   EN DATAS"
70 DATA 205,10,160,62,52,205,90
80 DATA 187,24,254,33,15,160,227
90 DATA 201,62,49,205,90,187,221
100 DATA 33,27,160,221,227,201,62
110 DATA 58,205,90,187,253,33,39
120 DATA 160,253,227,201,62,51,205
130 DATA 90,187,227,201,0,0,0
```

MICRO-1

el IVA lo paga MICRO-1

C/ Duque de Sesto, 50. 28009 Madrid
Tel.: (91) 275 96 16/274 53 80
(Metro O'Donell o Goya)

SOFTWARE: por cada programa GRATIS ¡¡1 BOLIGRAFO CON RELOJ DE CUARZO!!

HYPER SPORTS	**2.300** ptas.
TORNADO LOW LEVEL	**1.950** ptas.
EXPLODING FISTT	**2.300** ptas.
JUMP JET	**2.495** ptas.
ZORRO	**2.600** ptas.
SABREWULF	**1.650** ptas.
GHOSTBUSTERS	**1.950** ptas.
GYROSCOPE	**2.300** ptas.
HYGHWAY ENCOUNTER	**1.750** ptas.
HIGHWAY ENCOUNTER DISCO	**3.300** ptas.

DYNAMITE DAN	**2.100** ptas.
RAID OVER MOSCOW	**2.300** ptas.
THEY SOLD A MILLION	**2.500** ptas.
FIGHTER PILOT	**1.975** ptas.
MASTER OF T. LAMP	**1.950** ptas.
NIGHTSHADE	**1.950** ptas.
HACKER	**1.950** ptas.
SUPER TEST	**2.300** ptas.
MAPGAME	**2.700** ptas.
TONADO LOW LEVEL DISCO	**3.300** ptas.

JOYSTICK QUICK SHOTT II.. **2.295** ptas.
JOYSTICK QUICK SHOT V ... **2.595** ptas.

PC-COMPATIBLE IBM 256 K
MONITOR FOSFORO VERDE
2 BOCAS DISKETTE 360 K
SOLO ¡¡**243.900**!!

TAPA METACRILATO PARA TECLADO ¡¡**1.900** ptas.!!

UNIDAD DISKETTE 5.25"
¡¡**45.900** ptas.!!
(incluido controlador)

LAPIZ OPTICO
¡¡**4.900** ptas.!!

IMPRESORA MARGARITA
¡¡**49.900** ptas.!!

CASSETTE ESPECIAL
ORDENADOR **5.295** ptas.

PRECIOS SUPER-EXCEPCIONALES PARA
AMSTRAD CPC-472 Y CPC-6128
¡¡LLAMANOS, TE ASOMBRARAS!!

IMPRESORAS
¡¡20% DTO. SOBRE P.V.P.!!

SINTETIZADOR DE VOZ
Y AMPLIFICADOR:
7.900 ptas.

MODULADOR TV
8.400 ptas.

INTERFACE DISCO
5 1.4" **5.300** ptas.

CINTA C-15 ESPECIAL
ORDENADOR **85** ptas.
DISKETTE 3" **990** ptas.

UNIDAD DE DISCO 3" CON
CONTROLADOR: **49.900** ptas.

Libros:

Curso autodidáctico Basic I	**2.525** ptas.
Curso autodidáctico Basic II	**2.525** ptas.
Programando con Amstrad	**2.195** ptas.
Juegos sensacionales Amstrad	**1.950** ptas.
Hacia la Inteligencia Artific.	**1.295** ptas.
Música y sonidos con Amstrad	**995** ptas.

Pedidos contra reembolso sin ningún gasto de envío. Tels. (91) 275 96 16/274 53 80, o escribiendo a Micro-1. C/ Duque de Sesto, 50. 28009 Madrid

Sin duda alguna

A través de esta sección se pretende resolver, en la medida de lo posible, todas las posibles dudas que **«atormenten»** a todas las personas interesadas en el mundo del AMSTRAD, sean o no poseedores de uno y, si lo son, se encuentren en cualquier nivel de destreza en su manejo.

Semanalmente, aparecen en estas páginas las consultas de la mayor cantidad de usuarios posible; ello redundará en un mejor servicio y en un contacto más estrecho entre todos nosotros a través de la revista.

SIN DUDA ALGUNA está abierta a todos.

En el Banco de pruebas, referente al PCW 8256 hay una frase que dice: «poder usar cualquier programa que funcione bajo el mismo sistema operativo o sea el CP/M y que esté en igual formato de 3 pulgadas». ¿Esto significa entonces que puede usar cualquier programa con el mismo CP/M y en 3 pulgadas, de otro ordenador?

Carlos Veja Sancho

Por desgracia, la respuesta es sí y no. Depende del programa: al estar en 3 pulgadas y correr bajo CP/M, sí.

Pero, si el programa está especialmente adaptado a las características de consola del otro ordenador, habría que readaptarlo al ***Amstrad.***

Las mayores o menores dificultades de este último proceso dependen de muchos factores, y no se puede asegurar que siempre le sea factible al usuario.

Me dirijo a la sección de Sin duda alguna:

En el **MICROHOBBY AMSTRAD** número 5, en la página 5 dice que estudiaba la medida de ampliar la memoria del CPC 464. ¿Me podrían decir si ya se ha ampliado?

¿Podrían decirme también si las radiaciones que emite el monitor sobre el ordenador cuando están uno enfrente de otro, pueden perjudicar a la cinta que esté en el cassette? Se lo digo porque es que ya me ha pasado, y no sé si es por culpa del cabezal o de las radiaciones.

Jesús Alegría

1) Como comentábamos en Sin duda alguna en su momento, existen en Inglaterra ampliaciones de memoria, hasta 256 K, para la serie CPC. En España, que nosotros sepamos, no.

2) Probablemente se deba al cabezal. Las «radiaciones» emitidas por el monitor del ***Amstrad*** *no pueden perjudicar a la cinta.*

—¿La versión del Dr. Logo que acompaña a este ordenador es la misma que la que posee el modelo 664 o está ampliada aprovechando la mayor cantidad de memoria disponible? Lo digo porque se suministra en el mismo disco que el sistema operativo CP/M 2.2.

—¿Es posible que por causa de un decreto de homologación se modifique en un futuro inmediato el teclado del CPC-6128 con el fin de adaptarse a las lenguas peninsulares?

Diego

1) Con el 6128 vienen dos versiones de LOGO, una, adaptada al 128, con muchos más comandos y capaz de usar más memoria. La segunda es análoga a la del CPC664.

2) El ***Amstrad*** *6128 deberá adaptarse, como todas las máquinas, a las normas de homologación que entrarán en vigor, salvo nuevas prórrogas, dentro de aproximadamente 5 meses.*

¿Dónde se puede conseguir el libro de Firmware del **Amstrad**?, o ¿lo podrían publicar en su revista?

También quisiera saber cómo utilizar los comandos HIMEN y MEMORY a la hora de mezclar en un programa Basic y código máquina, y si se pueden presentar problemas al hacer éste.

Esperando que me resuelvan estas dudas, se despide de ustedes.

Antonio Sánchez Tejero *(Granada)*

Indescomp vende el libro de Firmware del ***Amstrad.****.*

No, no lo podemos publicar en nuestra revista debido a su gran extensión, y a que se trata de una publicación sometida al copyrigth.

2) El comando HIMEM nos da el valor de la posición más alta de memoria resevada al Basic.

El comando MEMORY altera dicha posición, y se suele utilizar para reservar espacio en la memoria a rutinas de código máquina.

Amstrad Ideas

AMSTRAD Semanal comunica a todos sus lectores la apertura de una nueva sección dedicada a recoger las mejores ideas que exploten al máximo las posibilidades del ordenador, materializadas en programas claros y cortos *(máximo 25 líneas)*. Los mejores de entre todos ellos serán publicados con el nombre de su autor en la revista, recibiendo como premio, gratuitamente en su domicilio los cuatro primeros números de nuestra cinta mensual. Los programas enviados deberán incluir:

— Cinta de cassette con el programa o programas grabados.

— Explicación detallada del funcionamiento y propósito del programa, mecanografiado a 2 espacios o con letra clara.

Es imprescindible indicar en el sobre claramente: **AMSTRAD IDEAS.**

La dirección es:

Hobby Press, S. A.
La Granja, 39
Polígono Industrial de Alcobendas.
Madrid

Mercado común

Con el objeto de fomentar las relaciones entre los usuarios de AMSTRAD, *MERCADO COMUN* te ofrece sus páginas para publicar los pequeños anuncios que relacionados con el ordenador y su mundo se ajusten al formato indicado a continuación.

En *MERCADO COMUN* tienen cabida, anuncios de ventas, compras, clubs de usuarios de AMSTRAD, programadores, y en general cualquier clase de anuncio que pueda servir de utilidad a nuestros lectores.

Envíanos tu anuncio mecanografiado a: **HOBBY PRESS, S.A.**
AMSTRAD SEMANAL.
Apartado de correos 54.062
28080 MADRID
¡ABSTENERSE PIRATAS!

Vendo consola vídeo-juego marca Intellivision, nuevo (del 6-1-85) con 6 cartuchos de juegos: fútbol, baloncesto, tenis, boxeo, space battle, triple action y adaptador para TV. 15.000 ptas. Eulogio Marzo. C/ Lérida, 9 - 1.º-2.ª. Sant Vicenç dels horts *(Barcelona)* Tel. (93) 656 39 78.

Poseedor de Amstrad CPC-464 desearía contactar con usuarios del mismo ordenador para el cambio de programas, tanto de juegos como de utilidades. Tengo más de 100 programas entre juegos y utilidades. Les ruego manden lista. Escribir a: Fernando Martínez Martínez. C/ Alicante, 7 - 3.º A. 30003 Murcia.

Desearía contactar con usuarios del **Amstrad**, para intercambio de programas, información e ideas. Escribir o llamar a: Claudio Rivero Armas. C/ Antonio Collado, 19. 35015 Las Palmas (Gran Canaria). Tel. (928) 31 23 98.

Urgente vendo **Amstrad** CPC 664 con monitor en color 2 discos originales con CP/M, logo, base de datos, etc., y manual en español. Garantía oficial **Amstrad** España a estrenar. Todo por 110.000 pts. Tel. (91) 206 75 21.

Desearía intercambiar programas comerciales para **Amstrad** CPC 464/664/6128. Estoy interesado también en relacionarme personalmente con usuarios de **Amstrad** en Málaga. Interesados escribir a John Stubbs Cruz. C/ Mirador «Albion». Urb. «Cerrado de Calderón». 29018 Málaga. Aquellos residentes en Málaga pueden llamar al Tel. 29 15 74. Gracias.

Desearía contactar con usuarios del 464/664/6128 para intercambio de programas. Llamar de 10 a 11 de la noche o escribir a: Julián Calero. C/ Cataluña, 16-5.º izqda. Basauri *(Vizcaya)*. Tel. (94) 440 46 88.

Vendo Amstrad CPC-664, monitor fósforo verde, unidad de discos, garantía **Amstrad** España (6 meses), comprado el 11-9-85, manuales español e inglés, dos discos originales de regalo (C/PM, logo, procesador de textos, random files, diseñador de gráficos, puzzle, animal, vegetal y mineral). Y además regalo disco contabilidad (PCAN o ENERAL contable, P.V.P.: 14.500 pts.). Todo por 95.000 pts. Razón: Aurelio Sanchis Llopis. C/ San Pacual, 4. Genovés *(Valencia)*. Tel. (96) 227 67 22.

Desearía contactar con propietarios de CPC 664 y CPC 6128 para formar un club de usuarios en Zaragoza y provincias limítrofes. Mariano de la Iglesia. Tel. (976) 56 01 07. Zaragoza.

Amstrad 664/6128 desearía intercambio de programas en disco o cinta, utilidades o juegos.

Amstrad 664/6128 desearía intercambio de programas en disco o cinta, utilidades o juegos. Manuel Díaz Fernández. C/ Foncalada, 11 - 1.º dcha. 33002 Oviedo. Tel. 21 14 17.

Me gustaría intercambiar programas, juegos, ideas, etc., con usuarios del CPC 464 que vivan en Burgos. Llamar por la tarde al Tel. 26 06 89. Preguntar por Javier.

Intercambio programas comerciales de todo tipo para **Amstrad** 464, 664 y 6128: juegos, utilidades y copiones. Por favor adjuntar lista. Prometo contestar a todos. F.º Manuel Gijón Romero. C/ San Pío X, 4 - 2.º B. 18007 Granada. Tel. (958) 12 70 31. Preguntar por Francis.

Vendo Amstrad CPC-464 monitor f/verde, joystick, más de 40 programas y mauales en español, por sólo 45.000 pts. Idem y además una unidad de disco y programas, por 87.000 pts. Tel. (91) 796 15 46. Llamar de 9-13 h. preguntar por Ignacio. Madrid.

Vendo Amstrad 464 verde 50.000. Disco con controlador 45.000. Todo por 90.000. Garantía 4 meses. Con manuales, libros, y 20 juegos y utilidades, los mejores. Tel. 888 58 41. José Luis. Cenas. Madrid.

Desearía contactar con usuarios del Amstrad CPC 6128 para intercambio, de ideas, comentarios, etc. Tengo algunos programas interesantes. A ser posible de Madrid. Carlos. Tel. (91) 233 05 74 (comidas o cenas).

Vendo, cambio programas para **Amstrad** CPC 664. Interesados escribir a Juan Mucientes Rasilla. C/ Nicaragua, 14-5.º C. 15005 La Coruña.

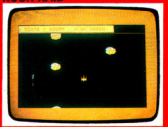
Micro 6128 PRESENTA... AMSTRAD

NUEVOS PROGRAMAS EN CASSETTE Y DISCO

ARGO NAVIS

El comandante de nave AMSTRAD-1 se encuentra atrapado en las profundidades de una central nuclear y debe salir con vida. Excelentes gráficos y sonido. **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

ROCK RAID



Debes pilotar con acierto la nave que a lo largo de su viaje galáctico sufrirá encuentros con meteoritos, residuos planetarios, etc. Gran movilidad, y excelentes efectos. **P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.**

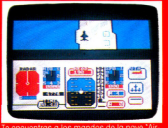
WIZARD'S LAIR

Te encuentras atrapado en las profundidades de una caverna, llena de obstáculos, adversidades, etc. ¿Serás capaz de salir con vida? **P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.**

MACADAM FLIPPER

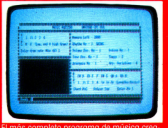
A través de este programa que nos traslada al manejo de la máquina-flipper del mejor casino de Las Vegas. Posibilidad de creación del tablero, puntuaciones, etc. **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

JUMP JET



Te encuentras a los mandos de la nave "Aircraft". En una perfecta maniobra debes despegar del portaviones. (Excelente versión simulador vuelo-combate) **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

MUSIC MAESTRO



El más completo programa de música creado para el AMSTRAD. Permite crear sonidos, melodías y convertir tu ordenador en la mejor "caja de música". **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

PAZAZZ

Programa que permite de una manera sencilla la creación de pantallas con gráficos, datorles de movimiento, acompañados de música. **P.V.P.: DISCO 2.900 pts.**

SYCLONE 2

Programa de utilidad que permite realizar copias de seguridad (back-ups) a distintas velocidades (baudios). **P.V.P.: CASSETTE 1.800 pts. DISCO 2.500 pts.**

ZEDIS II

Editor-desensamblador del Z-80, para el programador más avanzado **P.V.P.: CASSETTE 1.900 pts. DISCO 2.600 pts.**

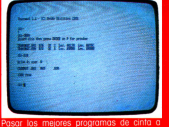
SYSTEM X

Ampliación del lenguaje Basic. Conjunto de 30 nuevas instrucciones (fill, circle, profec) para ayudar en la programación. **P.V.P.: CASSETTE 2.200 pts. DISCO 2.900 pts.**

ODDJOB

La mejor utilidad para el mejor conocimiento del disco.
(Copias de disco, Disk map, Disk track, sector, etc.) **P.V.P.: DISCO 2.600 pts.**

TRANSMAT

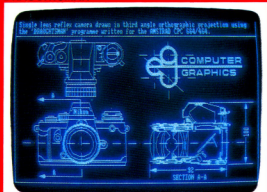


Pasar los mejores programas de cinta a disco ya no es problema. Con Transmat este proceso será fácil y sencillo. **P.V.P.: DISCO 2.600 pts.**

OTROS PROGRAMAS EN STOCK

MINI OFFICE	P.V.P. CASS. 3.200 pts. P.V.P. DIS. 3.900 pts.
WORLD CUP FOOTBALL	P.V.P. CASS. 1.800 pts.
BATLE FOR MIDWAY	P.V.P. CASS. 1.800 pts.
FIGHTER PILOT	P.V.P. CASS. 2.200 pts.
SURVIVOR	P.V.P. CASS. 1.800 pts.
MOON BUGGY	P.V.P. CASS. 1.800 pts.
TECHNICIAN TED	P.V.P. CASS. 1.800 pts.
FRUITY FRANK	P.V.P. CASS. 1.800 pts.
DATABASE	P.V.P. CASS. 2.100 pts.
LOGO TURTLE GRAPHICS	P.V.P. CASS. 2.400 pts.
TASCOPY Y TASPRINT	P.V.P. CASS. 2.600 pts.
FONT EDITOR	P.V.P. CASS. 1.900 pts.

DRAUGHTSMAN



Solicitada programa de dibujo que permite tratar la pantalla del AMSTRAD como un sencillo tablero de dibujo. sus resultados son espectaculares. **P.V.P.: CASSETTE 4.500 pts. DISCO 5.200 pts.**

ENVIENOS A **MICROBYTE** P.S.

P.º Castellana, 179, 1.º - 28046 Madrid

Nombre

Apellidos

Dirección

Población

D.P. Teléfono

ENVIOS GRATIS

JUEGO	C	D	Precio	TOTAL

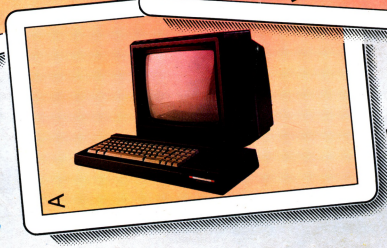
PRECIO TOTAL PESETAS

Incluyo talón nominativo ☐

Contra-Reembolso ☐

Pedidos por teléfono **91 - 442 54 33 / 44**

TRIO DE ASES.



Al comprar tu ordenador, CURSO BASIC de regalo

Solicita el lote de REGALOS al comprar tu ordenador

Apúntate a lo último. En SINCLAIR STORE tenemos las últimas novedades de este otoño. Desde el Spectrum de 128K al QL en español. Desde el nuevo AMSTRAD CPC 6128 a las últimas novedades mundiales en periféricos. Ven a vernos. Podrás comprobarlo personalmente. Y no olvides pedir tu tarjeta del CLUB SINCLAIR STORE, con la que conseguirás el 10% de descuento en tus próximas compras.

QL

- 128K RAM
- Procesador de 32 bits
- Teclado profesional en castellano
- 2 Microdrives incorporados
- Color y alta resolución
- Software incluído:
  - Tratamiento de textos
  - Base de datos
  - Hoja electrónica de cálculo
  - Gráficos

* GARANTIA INVESTRONICA

AMSTRAD CPC 6128

- 128K RAM
- 48K ROM
- Unidad de disco de 3"
- Teclado profesional en castellano
- Monitor color o fósforo verde
- Sistema operativo:
  AMS—DOS CP/M 2.2 y CP/M Plus.
- DR. LOGO
- Se entrega con dos discos de los sistemas operativos y Dr. LOGO y un disco con 6 programas de obsequio.
- Manuales en castellano

* GARANTIA OFICIAL AMSTRAD ESPAÑA

SPECTRUM 128

- 128K RAM
- Teclado con caracteres españoles
- Teclado adicional para editar programas o textos, controlar juegos o como calculadora
- Editor de pantalla permanente
- Admite el software del Spectrum y Spectrum +
- Salida RS 232 y RED ZX
- Conectores: T.V., monitor RGB, cassette, microdrive, etc.
- Facilidad de conexión a diversos instrumentos musicales.
- Manuales en castellano.

* GARANTIA INVESTRONICA

sinclair store

SOMOS PROFESIONALES

BRAVO MURILLO, 2 (aparc. gratuito en C/. Magallanes, 1). Tel.: 446 62 31
DIEGO DE LEON, 25 (aparc. gratuito en C/. Núñez de Balboa, 114). Tel.: 261 88 01 MADRID
AVDA. FELIPE II, 12. Tel.: 431 32 33 MADRID